CS230

Predicting Flight Arrival Delay: Final Report

James Rabe
Stanford University
jrabe@stanford.edu

Nico Stainfeld
Stanford University
nicost@stanford.edu

Abstract

In this research project, we attempt to predict flight arrival delays between two major domestic hubs based upon local weather patterns and basic flight information. After building three different models, we conclude that while weather data and basic flight information are essential for predicting flight arrival delays, this data alone is not enough to achieve the desired level of accuracy.

1 Description of Overall Task

In this research project, we intend to investigate the effects of various weather factors on US domestic flights. This problem is interesting to us as we are both graduating seniors who will be moving to New York City after graduation this summer to work jobs that require extensive travel. This tool will help us predict potentially lengthy arrival delays and allow for more time to reach our destination. More broadly, we foresee this tool helping airlines better allocate resources (planes, spare parts) and personnel (flight crew, mechanics), as delays are largely wasted time for the airlines.

2 Review of Previous Approaches

This project was inspired by our surveying of previous classes' projects [1], in particular one from Spring 2019 titled Predicting Delays in Flight Departure Time at SFO [2]. This piqued our interest, and we decided to adapt it to the NYC context. However, we also noted some potential areas for improvement over this previous project, particularly in the use of more up-to-date and accurate data. Critically, we propose the necessity of using destination weather information rather than just arrival weather data. Additionally, we believe the relevant metric for delay is not the flight departure delay but rather the flight's arrival delay. Often flights which depart less than 30 minutes late actually make up the lost time en-route, arriving on time. Data aside, this group used three different models, a simple neural net, an LSTM, and a CNN. We found that this paper's use of a CNN took advantage of having left the delay group of chronologically-close flights in the data. Thus, when predicting a flight's delay, this model could see the ultimate actual delay on flights leaving around the same time (using a sliding window based on time). While this solution elegantly achieved high accuracy, we believe that it is technically "seeing into the future" as these quantities could not be known when the data would be needed. This group's LSTM model similarly achieves high accuracy by making use of time dependent knowledge that would be unobtainable in a practical model.

Additionally, there are also projects concerning predicting flight delays from CS 229 which also provided useful background on successful and unsuccessful approaches. One such project, from Autumn 2017 [3], uses models beyond the scope of this course such as decision trees, as well as simple neural networks. They were able to achieve an enviable test accuracy of 91 percent when measuring arrival delay, however on closer inspection we see that this project similarly suffers from using future data which would be unknown to the user! In particular, they are able to achieve extremely high

CS230: Deep Learning, Spring 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

binary accuracy by using departure delay as the main predictor of arrival delay. Though this gives good results, in some ways we consider this approach less useful as the model's user would not be able to know the actual departure delay of similar flights beforehand, and thus could not use the model for decision-making. If anything, this model is more applicable to users such as arrival-city flight controllers, who can use the actual observed departure delay to predict arrival time while the aircraft is en route.

Lastly, beyond Stanford there are also a number of papers published on this area. One which we found most compelling was by researchers at the State University of New York Binghamton, in which the team used a somewhat-complex neural network to predict incoming flight delays at John F. Kennedy International Airport [4]. This approach benefits from interpretability, as each neuron represents an individual "delay factor". Though this approach was found to be in some ways superior to simple back propagation neural networks, we determined this to be beyond the scope of the course and thus this project.

3 Description of Dataset

Our data for this project comes from two primary sources. First, we use the Department of Transportation's Bureau of Transportation Statistics Airline On-Time Performance Data, which catalogs metrics for every domestic flight in the United States — roughly 600,000 flights each year [5]. We will be specifically focusing on data for 2019 flights between two cities: New York and Chicago. This amounts to roughly 3,300 flights per month, or around 38,000 flights over the 12-month period. Note that we originally planned to use a full 4 years of flights, but later concluded that more data would not significantly improve our model.

For each flight, metrics include day/date/time, carrier, origin, destination, scheduled departure and arrival, scheduled flight time, and distance. We will be using these data points to predict the class of arrival delay, among various distinct delay types. First, we predict binary delay. Later, we implement a model to classify three types of delay: early and on-time, less than 30 minutes delayed, and greater than 31 minutes delayed. Our final model predicts one of 15 different delay classes, which correspond to 15-minute delay intervals. For example, a flight that arrives 31 minutes late would be a "Class 2" delay, and a flight which arrives 10 minutes early would be a "Class -1" delay.

To supplement this dataset, we also purchased historical hourly weather data from OpenWeatherMap [6]. This data provides hourly weather observations, including actual temperature, minimum and maximum temperature projections, atmospheric pressure, humidity, wind direction and speed, precipitation volume, cloud cover, and a unique weather-type identifier.

To combine the datasets, we matched each flight to the hourly weather observation, rounding down, for both origin and destination airports. For example, a flight departing at 10:35am from JFK to ORD would get the 10am data from the New York City and Chicago monitoring stations.

All together, each row of the dataset represents one flight and includes 112 columns, most of which are a result of the one-hot encoding format. We then proceeded with a 90/10 train/test split. Thus we had roughly 34,000 observations in the training set and 3,800 observations in the test set.

4 Methodology

We initially spent significant time wrangling our data from different sources, cleaning the data, and merging the data.

We began by using the Adam learning rate optimization algorithm with ReLU and sigmoid activation functions with and without dropout. Much of our project relies on code from the class Coursera modules, with some customizations [7]. We also consulted with beginner Tensorflow guides [8].

We began by attempting a baseline binary classification problem, determining whether the flight was delayed or not. Initially, our models performed with an accuracy that hovered around 60-62 percent on both the train and test sets. However, once we changed the loss function for the binary model from softmax cross entropy loss to the simpler sigmoid cross entropy loss.

This initial binary model had 70 percent accuracy on the test set (with similar accuracy on the train set). This model used the Adam optimizer and sigmoid cross entropy loss. Although we were

somewhat satisfied with this accuracy, our initial focus was more on getting the models to run on our data.

Next, we pivoted to the core multi-class classification problem, where we tried to predict a total of 15 different delay classes (0-15 minutes delay, 16-30, etc.). For this problem, we switched to softmax cross entropy loss with a one-hot representation based on delay categories 0-14. This is a significantly more difficult problem than simple binary classification, and we ultimately were only able to achieve roughly 27 percent train and 26 percent test accuracy for the 15 classes. Though this is significantly above mere randomly guessing (100/15 classes = 6 percent random accuracy), we nonetheless were disappointed with this performance.

Taking a step backwards, we decided to push towards implementing a multi-class model with a simpler set of delay buckets. We settled on using 3 buckets: early, less than 30 minutes delay, and greater than 30 minutes delay. This approach, once again using a 3-layer ReLU -> ReLU -> Softmax neural network, yielded 57 percent train and 56 percent test accuracy, also a significant improvement over expected random accuracy of 33 percent. We were more pleased with this result, but had to continue fine tuning the parameters of the loss function in order to more accurately account for the "magnitude" of the error. In other words, cross entropy merely knows if the prediction is right or wrong. We proceeded to use mean absolute error and mean squared error cost functions. These functions take into account the "magnitude" of the error and thus we expected to see a change in our accuracy. However, we found no noticeable change to either train or test accuracy after using these cost functions. Thus, we moved to implement our own loss function.

To try to better incorporate the user's preferences regarding false positives and false negatives (better to be unexpectedly early than unexpectedly late), we set out to generate a custom loss function to more harshly penalize grossly erroneous predictions. However, given the requirement for the loss function to be differentiable we had issues incorporating this vision into the project. The loss function we settled on effectively took the actual minus predicted value raised to the fourth power. This ensures that predictions of group 1 with a ground truth of group 3 (or vice versa) would be very harshly penalized compared to other wrong predictions (on a scale of 16 times). We decided that this was appropriate due to the intended purpose of our model: predict flight delays for the busy traveler trying to make a meeting.

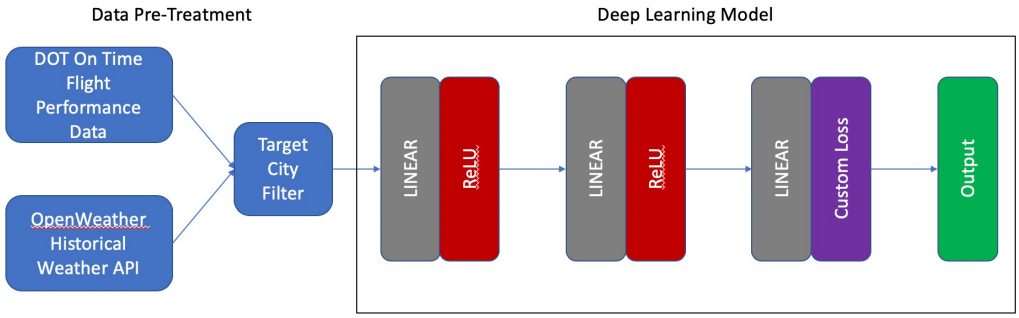

Figure 1: Data Pipeline and Project Architecture

5 Results Analysis

Taking into account these previous experiments with models, learning rates, and other hyperparameters, we determined that what we faced was clearly a bias issue and not a variance issue. This can be seen in the near-identical train and test error rates, implying we have very low variance. Thus, we realized it would not be an effective use of our time to purchase and import additional years of data. Instead, we chose to focus on reducing bias as much as possible.

Ultimately, our model was not able to improve much over the course of our experimentation and tweaking. From previous projects' attempts, we knew that this problem does not improve much beyond 50-100 epochs, which enabled us to rapidly iterate different custom loss functions, learning rates, and model types. This is also in line with what we saw in our own testing, with performance stabilizing around 100 epochs.

The final performance of our 3 models was as follows, with the expected random guess accuracy for reference:

	Best Train Accuracy	Best Test Accuracy	Random Guess Accuracy
Binary Classification	70.4%	70.3%	50%
3-Class Classification	56.7%	56.6%	33.3%
15-Class Classification	27.9%	27.7%	6.7%

Figure 2: Final Model Performance

Overall, while we are disappointed with our absolute accuracy numbers, we are pleased that our iterative process improved our initial baseline models somewhat. We also note consistently and significantly outperforming random guessing across all three classification problems. In particular, our binary classification model improved meaningfully after changing our loss functions and tweaking the model.

6 Future Work

After further research, we found the following statistic from the Bureau of Transportation Statistics: *only* 38.7 percent of total delayed minutes and 19.13 percent of total flight delays in 2019 were due to weather [9]. Although these percentages are likely higher for the Chicago to New York route we have modeled, as both cities have to contend with harsh winters, the point remains that well under a quarter of flight delays are due to adverse weather. This can help explain why our model did not perform according to our initial hopes. Therefore, we hypothesize that to make our model better, we would need to include data relevant to other significant sources of delays, namely, according to the Bureau of Transportation Statistics: air carrier delays, aircraft arriving late delays, national aviation system delays, and security delays. Thus if given the opportunity to continue work on this project in the future we would look to gather and include data related to these other delays to reduce bias in the models.

7 Contributions

Our team worked together on every aspect of the project. Nico Stainfeld handled the sourcing and preparation of the flight data while James Rabe handled the sourcing of the weather data and the joining of the two data sets. We both then worked together in real time on loading the data into Google Colab, preparing the data for use, and building the ML models. We enjoyed working together and feel that the burden of work was split evenly.

References

In our implementation of the model, we used the Google Colab programming environment and the following packages: pandas, tensorflow, tf-utils (from Coursera), math, numpy, h5py, matplot.pyplot, sklearn, and tensorflow.python.framework.

[1] Ethan Andrew Chi and Jillian Tang, *Predicting Flight Delays with Deep Learning*, http://cs230.stanford.edu/projects-winter-2019/reports/15813450.pdf.

[2] Yanqiu Wang and Yutong Coco Sun, *Predicting Delays in Flight Departure Time at SFO*, http://cs230.stanford.edu/projects-spring-2019/reports/18680989.pdf.

[3] Nathalie Kuhn and Navaneeth Jamadagni, *Application of Machine Learning Algorithms to Predict Flight Arrival Delays*, http://cs229.stanford.edu/proj2017/final-reports/5243248.pdf

[4] Sina Khanmohammadi, Salih Tutun, and Yunus Kucuk, *A New Multilevel Input Layer Artificial Neural Network for Predicting Flight Delays at JFK Airport*, https://www.sciencedirect.com/science/article/pii/S1877050916324942

[5] United States Department of Transportation Bureau of Transportation Statistics, *Airline On-Time Performance Data*, https://www.transtats.bts.gov/Tables.asp?DB-ID=120.

[6] OpenWeatherMap History Bulk data, https://openweathermap.org/history-bulk.

[7] A. Ng, K.Katanforoosh, and Y. Mourri. "Tensorflow," 2020, https://www.coursera.org/learn/deep-neural-network/notebook/AH2rK/tensorflow.

[8] "TensorFlow 2 quickstart for beginners," *TensorFlow.org* [Online]. https://www.tensorflow.org/tutorials/quickstart/beginner.

[9] *Understanding the Reporting of Causes of Flight Delays and Cancellations*, United States Department of Transportation Bureau of Transportation Statistics, https://www.bts.dot.gov/topics/airlines-and-airports/understanding-reporting-causes-flight-delays-and-cancellations.