Predicting Flight Duration

By:

Axel Perez - 00001053989

Dominic Magdaluyo - 00001283712

Alex Cherekdjian - 00001083236

Professor Anastasiu

COEN 281

March 10, 2020

Introduction:

The objective of this project is to use US domestic flight history to be able to predict how many minutes a given flight will take by with reasonable accuracy. By using flight information like the day, airline, origin, destination, and planned departure and arrival time, we can use machine learning techniques to build different models that are tuned to the patterns of how that flight information affects flight duration. With records of millions of flights over the span of ten years, we aim to keep our algorithms efficient but also capable of taking advantage of our multitude of data. Providing reliable estimates of flights can deliver a massive benefit to the consumer as it can deliver an overall better experience. In relation to the market for such development, with these estimates, users could plan their trips for picking up another individual at the airport more efficiently. This would reduce the amount of time spent circling the airport, overall unnecessary waiting, and possibly less traffic at the airport. Furthermore, airport courier companies would be very interested in such an invention as it would help diminish downtime between customer travels and stalling at the airport which would potentially yield more profit in the end.

System Design & Implementation Details:

The algorithms we selected to implement were a linear regression, a polynomial/cubic regression, and a random forest regression. We chose these implementations since we knew we needed to do a regression. A linear regression provides a good baseline for other techniques such as the polynomial/cubic regression and the random forest regression which felt like a unique addition. The technology and tools used throughout the project include Jupyter Notebook, numpy, pandas, sklearn, and joblib.

One aspect of consideration was whether to generate a model using all the data together or to generate multiple models depending on airport origin. Although there are hundreds of airports, the advantage of generating a model for each origin airport would be to improve the accuracy of the prediction. The problem with this strategy lies with the difference in the number of flights departing from each different airport. Figures 1 and 2 show the top 50 and lowest 50 airports in outbound flight volume from the dataset. Because of the number of outbound flights, a model built for an airport like Atlanta which has the highest volume of outbound flights would

potentially be trained better than an airport with less traffic. While the top airports have hundreds of thousands of flights to generate a model, the low volume airports have under a hundred flights in some cases. Later in the report, we show our findings for experimenting with creating models for some specific large and mid-sized airports. In the end, for the GUI, we created one model using all airports.

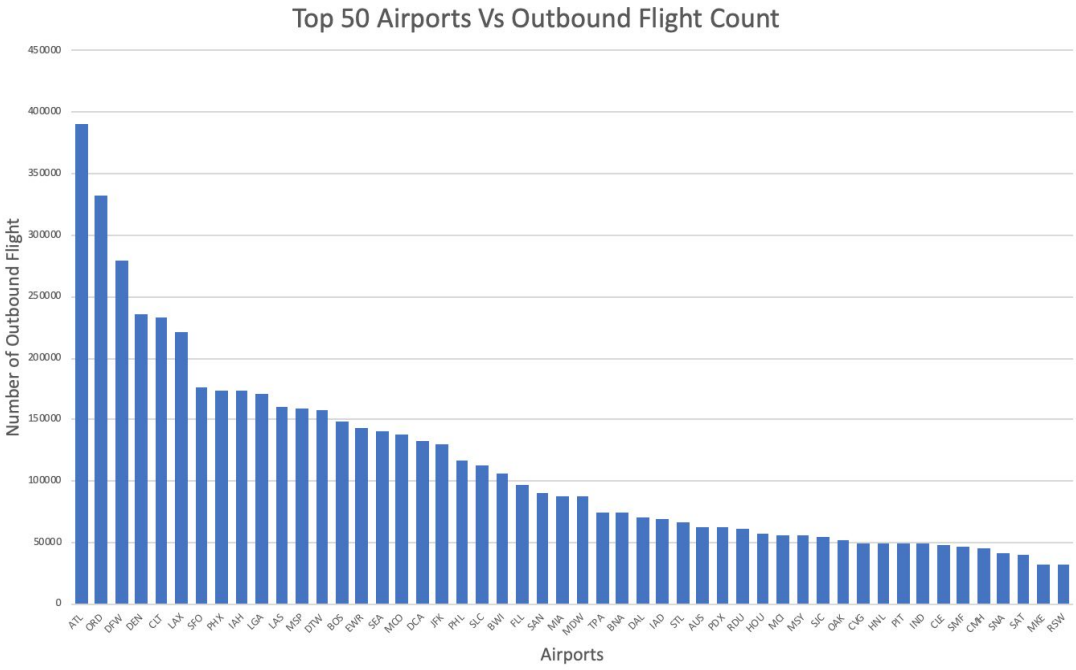

Figure 1: *Top 50 Airports versus Outbound Flight Count*

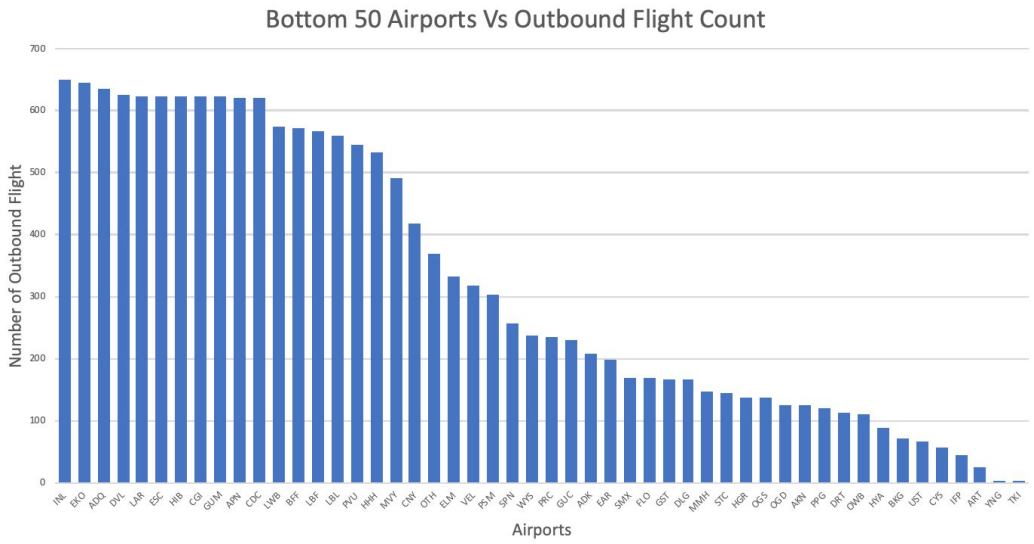

Figure 2: *Bottom 50 Airports versus Outbound Flight Count*

Experiments / Proof of concept Evaluation:

The US domestic flight dataset, "Airline Delay and Cancellation Data, 2009 - 2018", from Kaggle includes twenty-seven features. These include flight date, airline, flight number, origin, destination, planned departure time, planned arrival time, true departure time, true arrival time, along with specific flight information such as whether there is a weather or security delay, and when the flight's wheels took off and touched the ground. Only one column labeled cancelation code is extremely sparse for all the flights with ninety-eight percent of the values being NULL. This flight data is separated by year from 2009 to 2018 with roughly 6 million flights per year. In total, the entire dataset contains about 61.56 million flights. The type of data ranges from two letter op-codes for the airline (AA - American Airlines), three-letter op-codes for the destination or origin airport (LAX), dates in the format year-month-day, integers for values such as planned departure and arrival times, and floats for attributes such as actual departure and arrival times respectively.

For pre-processing the data, we wanted to select features that a customer using our software would have the details for. So, our first priority was to drop all of the columns that were not applicable to our predictions. The column that we were to predict was labeled the actual elapsed time. We first ensured to drop rows that included NA or NULL data. In keeping in line with our objective for user inputs to the model, the columns we decided to keep as potential candidates for inputs to our models were the flight origin, destination, date, planned departure time, airline, and planned arrival time. In order for our data to work with regression algorithms, the dates, origin, and destination columns needed to be turned into numbers. For the origin and destination airports, we used SkLearn's label encoder that maps each instance of a sequence to a specific number. For the dates, after conducting some research on the statistics of our dataset, we noticed that the best way to organize the dates was in bins of 4 months as there was a drastic increase in delay times in flights, thus increasing total flight time (Tomin).

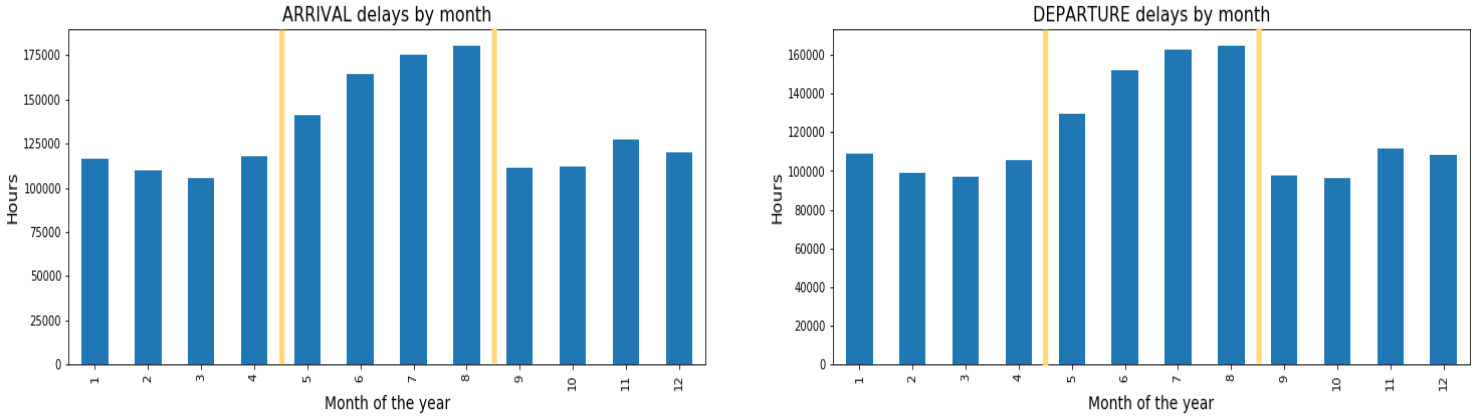

Figure 3: *Flight Delays vs Month (Tomin): Bins of 4 months shown by yellow bars*

After verifying this claim with our own scripts, we wrote a simple function to relabel each flight date accordingly. Furthermore, we analyzed the amount of delay each airline experienced over the 10 years and plotted them in figure 4. This clearly shows that certain airlines do typically have more delay than others and thus later arrival times. However, through implementing the algorithms, we noticed that models run with and without this feature ran with similar accuracy. Therefore we chose not to use the airline as an input.

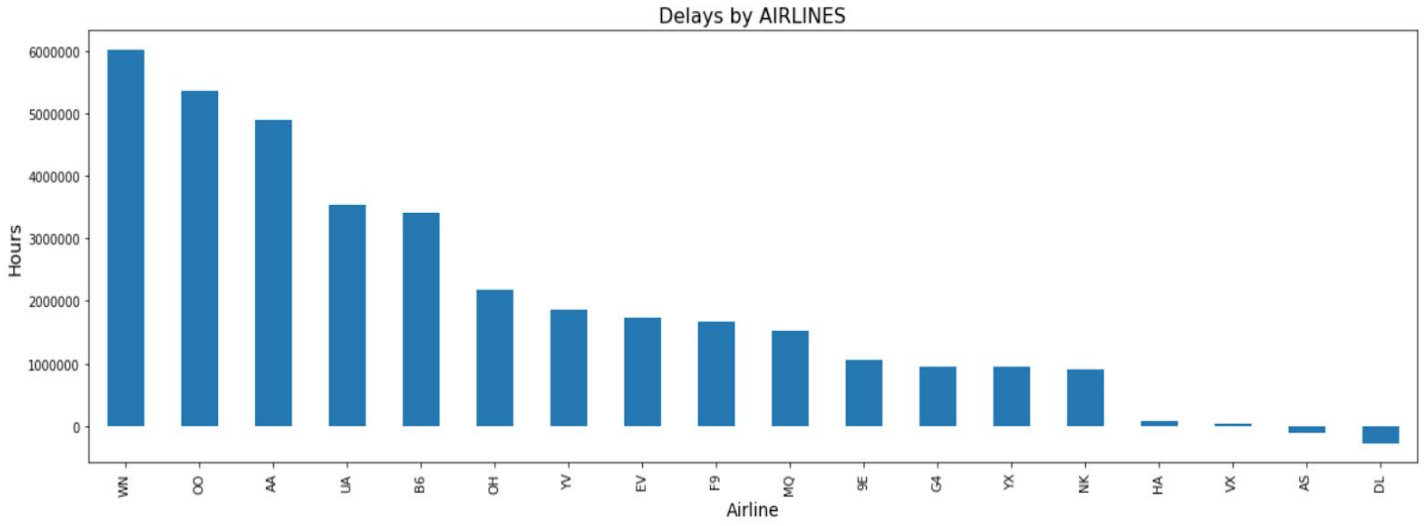

Figure 4: *Flight Delays Vs Airline*

The methodology behind utilizing the preprocessed data was to run our models on a single year's worth of data and choose the best models to use on the entire dataset. We will also use Sklearns "train test split" module to split our data into an 80/20 split for training and testing.

Linear, Polynomial, and Cubic Regressions:

The initial regressions that we planned on running our dataset through were the linear, polynomial and cubic regressions using an elastic net, magnetic lasso, and ridge regression techniques. These methods were implemented in order to prevent overfitting of our data.

Ridge regression proved to be the worst method with regard to these three. When trained on a year's worth of data from all airports, the best value of N and alpha were N=3 and alpha = 1/10 as shown in figure 5 with a minimum MSE score of 4399.40, averaging a miscalculation of around 66.32 minutes.

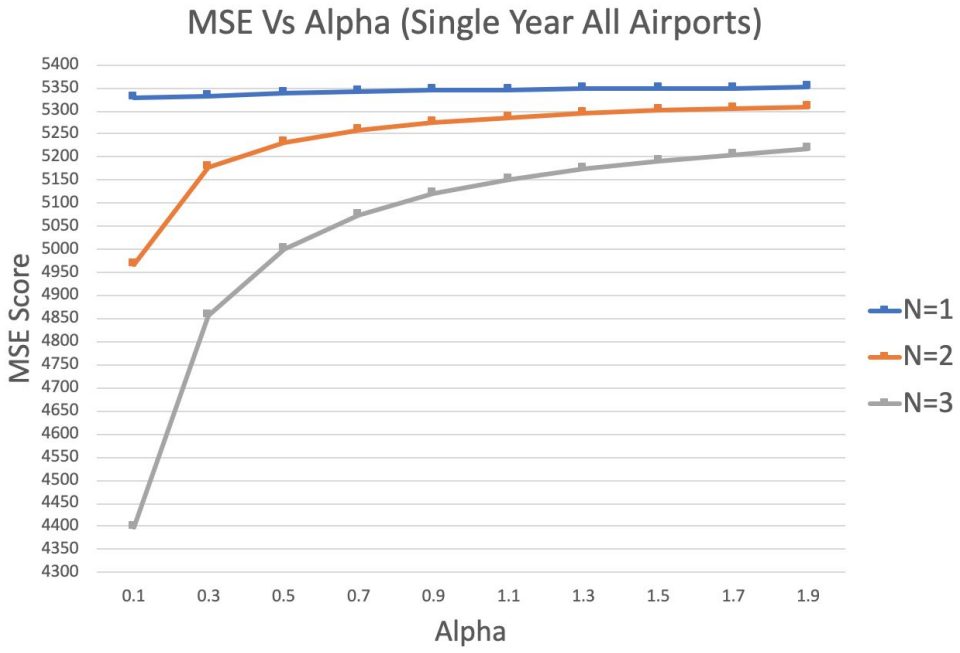

Figure 5: *Linear/Polynomial/Cubic Ridge MSE Vs. Alpha*

Using an Elastic Net proved to be a fairly decent method. When trained on a year's worth of data from all airports, the best value of N and alpha were N=3 and alpha = 1/10 as shown in figure 6 with a minimum MSE score of 3865.2, averaging a miscalculation of around 62.17 minutes.

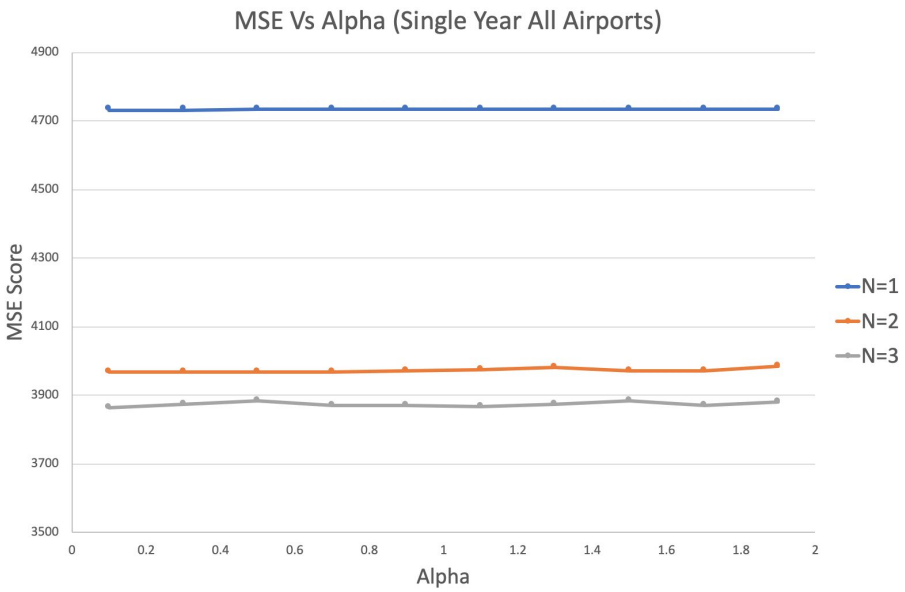

Figure 6: *Linear/Polynomial/Cubic Elastic Net MSE Vs. Alpha*

Overall, the Magnetic Lasso proved to be slightly better than the Elastic Net regression. When trained on a year's worth of data from all airports, the best value of N and alpha were N=3 and alpha = 1/10 as shown in figure 7 with a minimum MSE score of 3211.6, averaging a miscalculation of around 56.67 minutes.

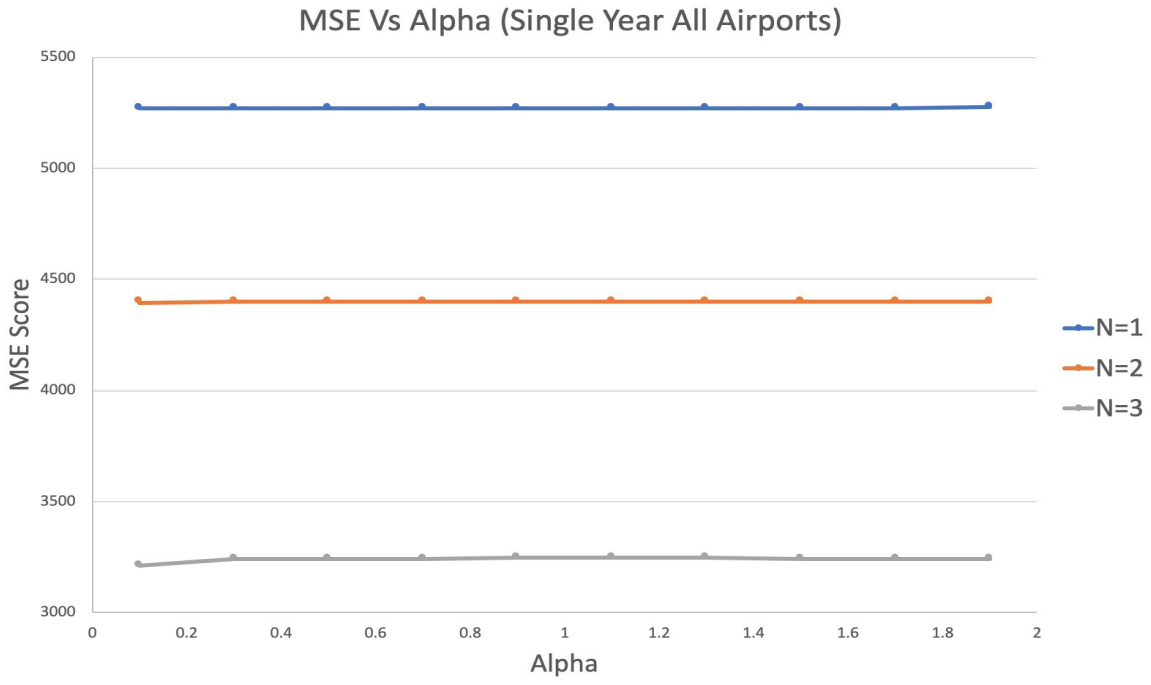

Figure 7: *Linear/Polynomial/Cubic Magnetic Lasso MSE Vs. Alpha*

Random Forest Regression:

After deciding we would do a linear, polynomial, and ridge regression, we thought it would be interesting to see how a random forest regressor would do on our flight duration data. Random forest is known for being an accurate model and good on large datasets. Decision trees in general tend to overfit but we were hoping by using a random forest and a large dataset, we would be able to get many trees on random parts of the data to reduce the chance of overfitting.

In order to get a full understanding of the capabilities of a random forest on our data, we tested random forests with various tree numbers on all airport data, CLE airport origin data, ATL airport origin data, and LAX airport origin data. As seen in Figure 8, all of the models follow the same diminishing return curve. After 10 trees, the improvement in accuracy was minimal while the increase in training time and model size were significant. Thus, we decided to use a 10 tree random forest for the GUI and most of the following experiments. We also observed that using all of the airports as an origin was typically more accurate than training random forest models on small airports but not as accurate as larger airports like ATL and LAX.

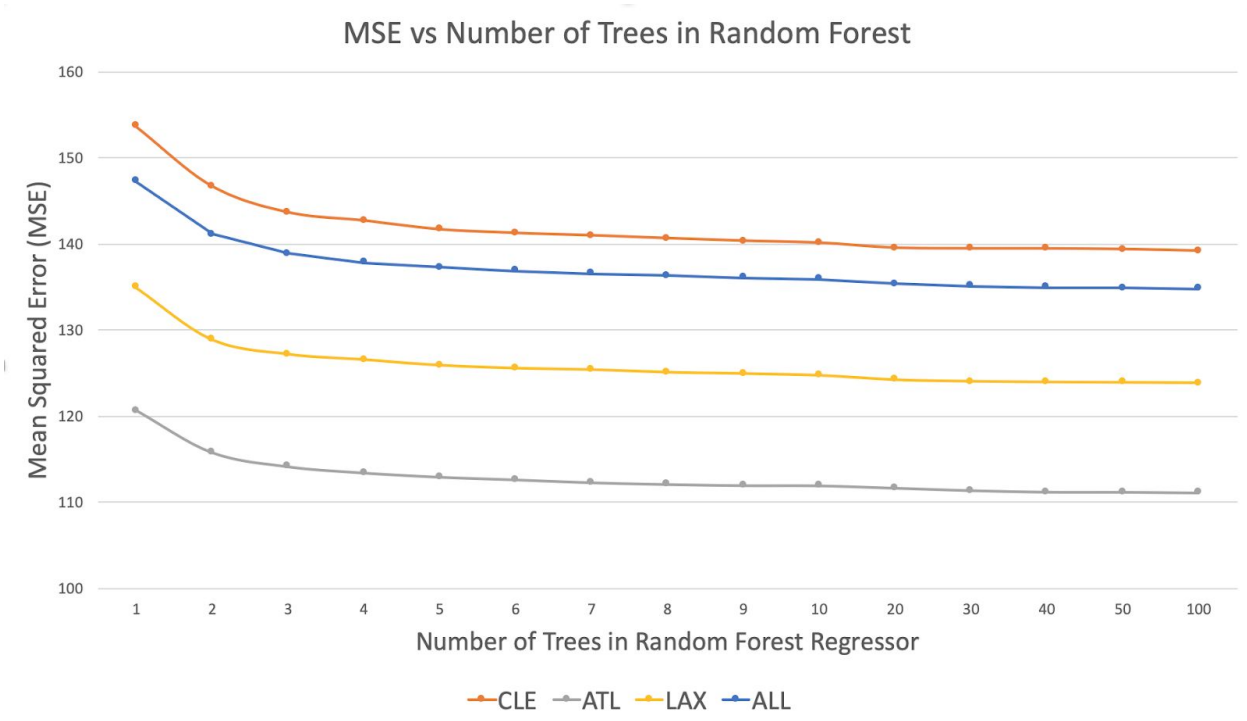

Figure 8: *Random Forest Regression MSE Vs. Number of Trees*

Model Results on Individual Airport Origins:

Figure 9 has the results of creating different models for specific origin airports to compare to the model created from data from all airports. In the case of the Cleveland (CLE) airport, the model shows little to no margin of improvement over the model created from all airports. The Los Angeles (LAX) and Atlanta (ATL) airports have a higher volume of departing flights in the data, so the models built for these specific airports perform slightly better than the model created from all airports. Even though we decided to use the model for all airports in our GUI, creating a model specific to the airport can be useful in the case where the user is only concerned about one departure airport and the departure airport being used for the model is among the top 25 airports in flight departure quantity.

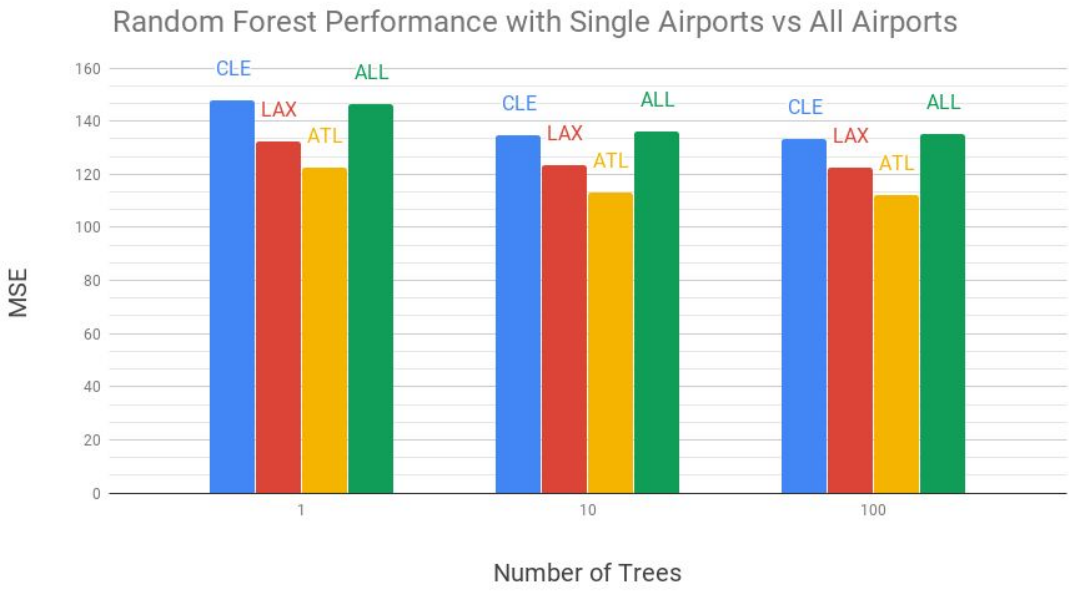

Figure 9: *Random Forest Performance Comparing Single Airport Models with All Airports*

Model Results without Outliers:

Methods were tested to increase this score, such as removing outlier flights that had delays past an hour. We attempted this on all of the regressions to see if the overall score could be improved. In removing outlier flights with delays over an hour from our dataset, we removed roughly 5% of our data as outliers which correlated to keeping all data that had total delays of flight time by 150 minutes.

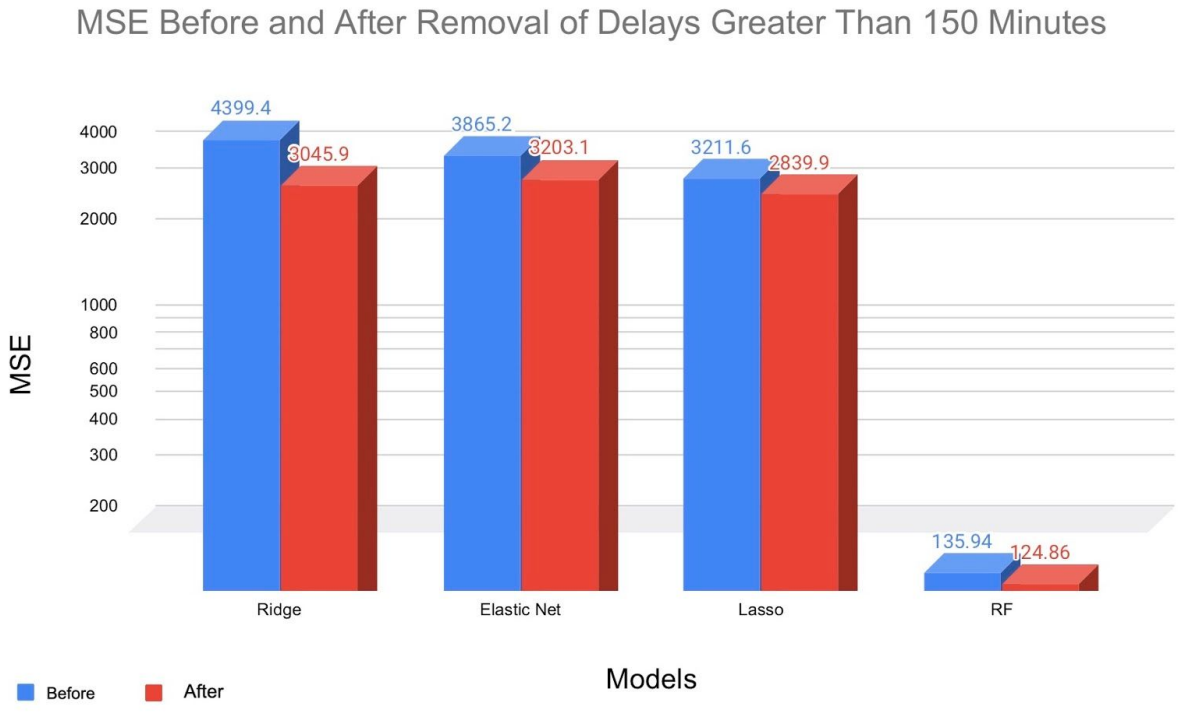

Figure 10: *MSE After Removing Outlier Flights Comparison*

The overall MSE score did improve in all scenarios, shown in figure 10. Most notably, the polynomial ridge regression score dropped from a MSE of 4399.40 to 3045.9.

Discussion & Conclusions:

Our GUI, shown in figure 11, uses a random forest model built from using data from all airports. Our decision to do it this way was to make a more general purpose model for users. We initially encountered some difficulty figuring out which columns to ignore for training the data. Some values, like actual departure time, were potentially useful to make an accurate prediction, but they weren't realistic for users to have. At one point, all of our models were giving really low MSE values after training. We discovered that we accidentally included the prediction value as a parameter when testing the training data. Our final GUI uses the random forest model of 10 trees because the 100 tree model showed minimal improvements over the 10 tree model, but generated a file that was 10 times larger. Through this project, we learned a lot about how the libraries that implement these algorithms as well as the pandas and numpy libraries.

```
bash-3.2$ python3 gui.py rf_model.joblib
Enter the 3 character origin airport: SFO
Enter the 3 character destination airport: JFK
Enter the month of departure (1-12): 3
Enter the scheduled departure time (24 hour HHMM format): 900
Enter the scheduled arrival time (24 hour HHMM format): 1745
[333.42987734]
bash-3.2$
```

Figure 11: *Sample Run of the GUI Script Running the Random Forest Model*

Project Plan / Task Distribution:

The task distribution is as follows:

- Preprocessing - Alex
- Initial linear regression - Dom
- Polynomial regression - Alex
- Ridge/Lasso/Elastic comparison - Alex
- Random Forest - Axel
- Model exporting - Axel
- GUI - Dom

Our plan was to initially attempt different regression strategies on a single year to figure out which model would work best for the total 10 years. Each of us tried a different regression algorithm and compared accuracies among the models. Random forest had the best results, but we also decided to compare additional regressions such as linear/polynomial with ridge, lasso, and elastic net regression.

Works Cited

Mu, Yuanyu. “Airline Delay and Cancellation Data, 2009 - 2018.” *Kaggle*, OST, 11 Aug. 2019, www.kaggle.com/yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018.

Tomin, Milan. “Airline Delay and Cancellation Data 2018.” *Kaggle*, Kaggle, 23 Aug. 2019, www.kaggle.com/milantomin/airline-delay-and-cancellation-data-2018.