Predicting Flight Durations

Axel Perez, Dominic Magdaluyo,
& Alex Cherekdjian

Introduction

- Objective - predict total flight time
- Compare different models
- Market Benefit:
    - More efficient airport pickups
        - Airport circling
    - Possibly less airport traffic
    - Airport courier companies
        - Less downtime
        - More profit

Dataset Qualities

- Airline Delay and Cancellation Data from 2009 - 2018
- Each year ~ 6 million flights
- Total flights ~ 61.56 million flights
- 27 Features:
    - Flight date
    - Airline
    - Flight number
    - Origin/Destination
    - Planned departure/arrival time
    - Actual departure/arrival time
    - Weather/security delay
    - Flight’s wheels off and on ground

Dataset Qualities

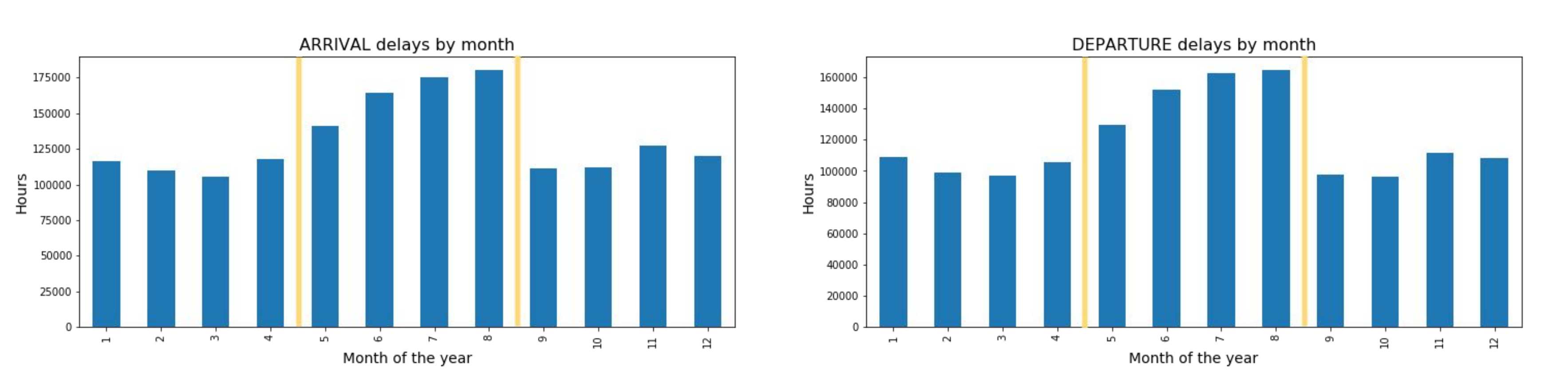
Data Types	Example	Feature Instances
Two letter op-codes	AA - American Airlines	Airline
Three letter op-codes	LAX, CLE, JFK	Origin, Destination
year-month-day	2019-01-01	Dates
Integers	1100, 1300	Planned departure and arrival times
Floats	1102.0, 1230.0	Actual departure and arrival times
NA / NULL	NULL, NA	Cancellation Code, Rarely in other columns

Pre-Processing

- Drop all columns that directly lead to prediction
    - Ie. delay, taxi time, wheels up, wheels down etc.
- Drop all information user will not have access to
    - Input - Flight origin, Destination, Date, Planned departure time, Airline, Planned arrival time
- Ensure no rows with NA or NULL data

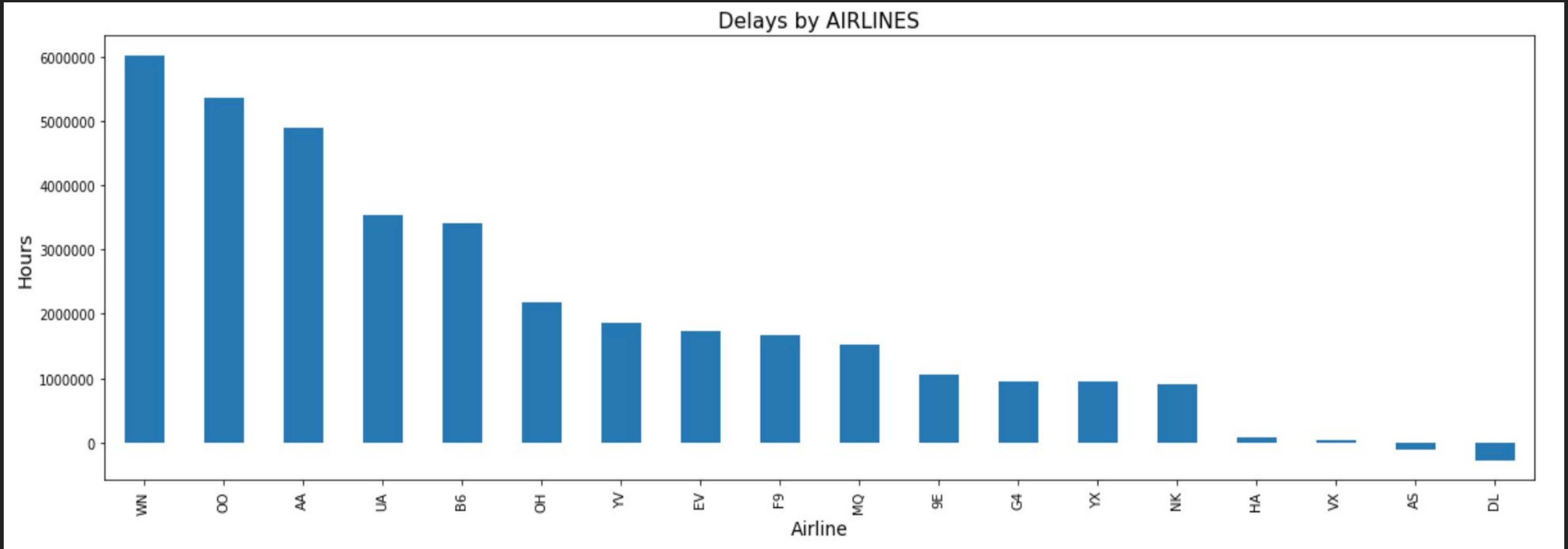
Pre-Processing

- Label encode airport origin and destinations
  - Encoder (strings) -> (unique integer)
    - Ie. LAX => 1, CLE => 2 etc.
- Split dates into four bins of three months



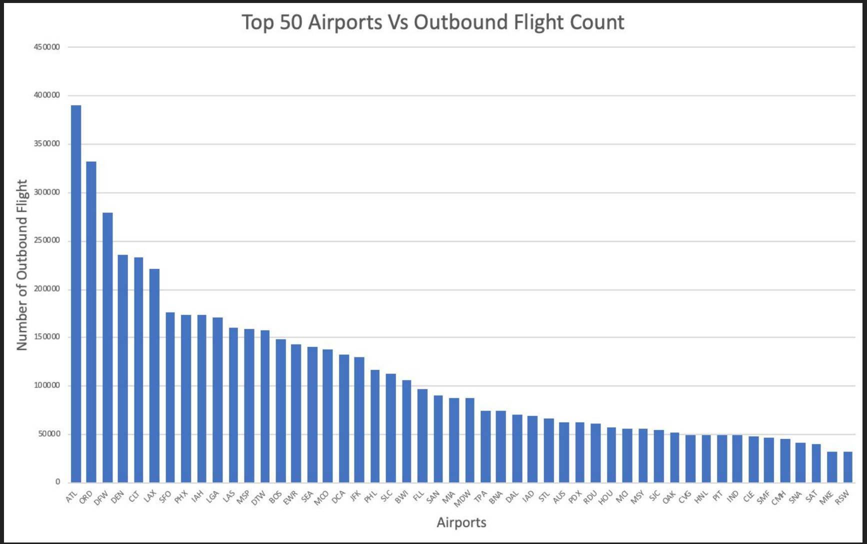
Pre-Processing

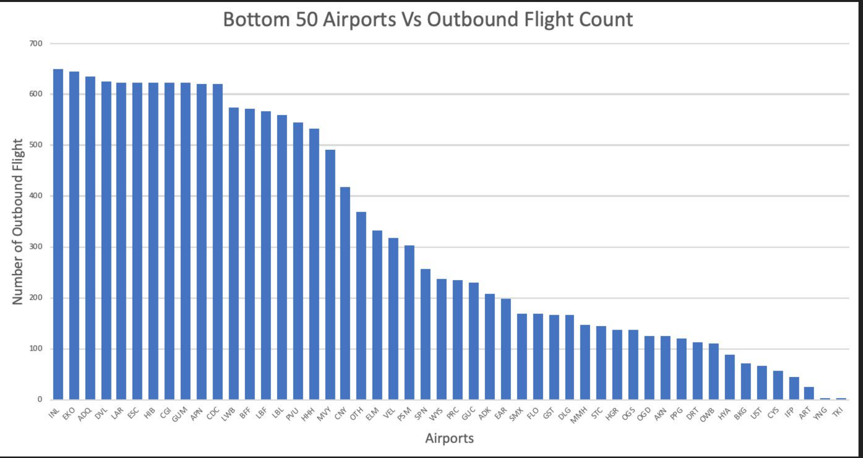
- Airline and delay correlation
  - Implementations with and without feature - Similar MSE values



Flight Count per Airport

- Considered creating a different model for each airport.
- Could be used to create a very accurate model overall
- Some airports lack the significant amount of data needed to create a good model
- Busy airports have hundreds of thousands of entries in our data set while slow airports can have less than a hundred over a 10 year period





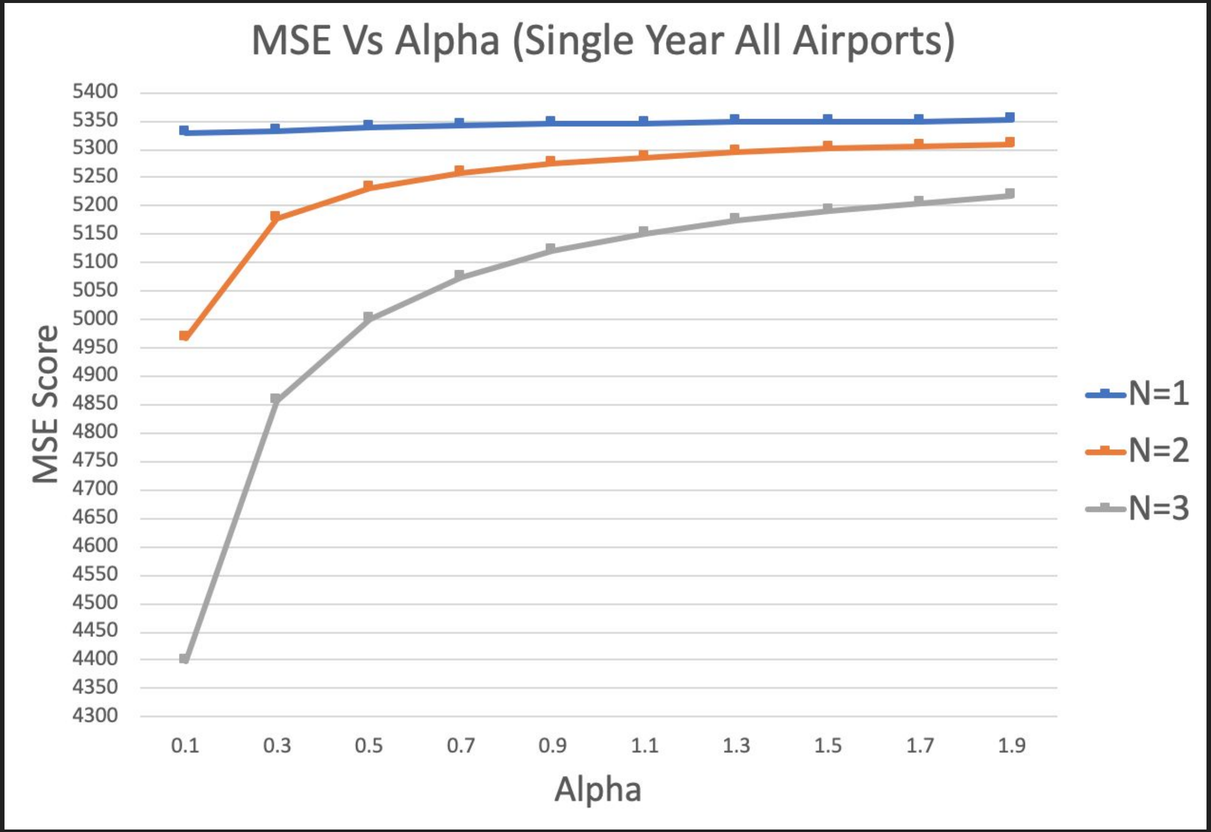
Methodology / Test Plan

- Test on a single year, then choose best models to run on all 10 years
- Use Sklearn test Split Module 80/20
- Tech used - Jupyter Notebook, numpy, pandas, sklearn, and joblib

Linear, Polynomial, & Cubic Regressions

- Tried using magnetic lasso, ridge, and elastic net
  - Prevent overfitting
- Ran regression over N=1, 2, 3
- Alpha's = 0.1 - 1.9 in steps of 0.2
- Implementation - SkLearn
- Testing methodology - SkLearn Test Split 80/20

Linear, Polynomial, & Cubic Regressions



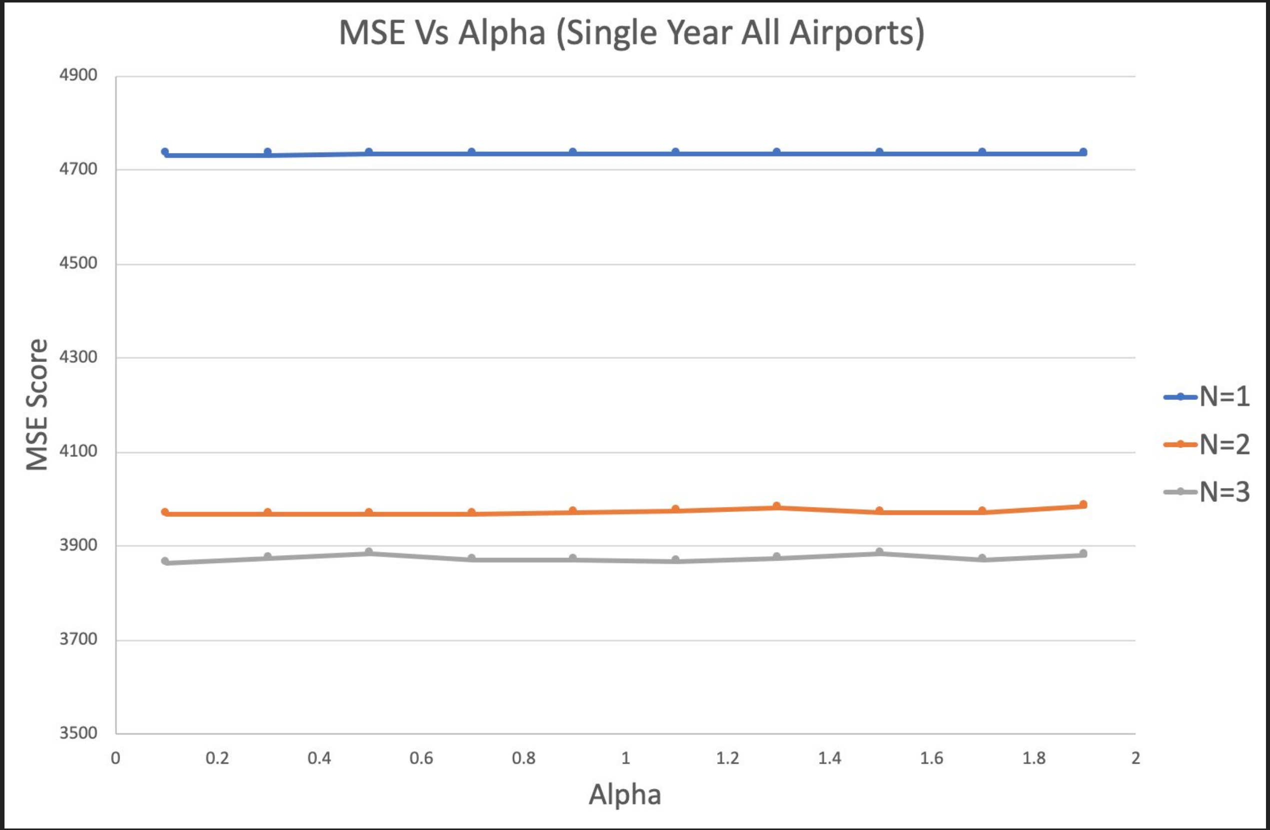
<u>**Ridge**</u>

Best MSE = 4399.40
~ 66.32 minutes
N = 3
Alpha = 0.1

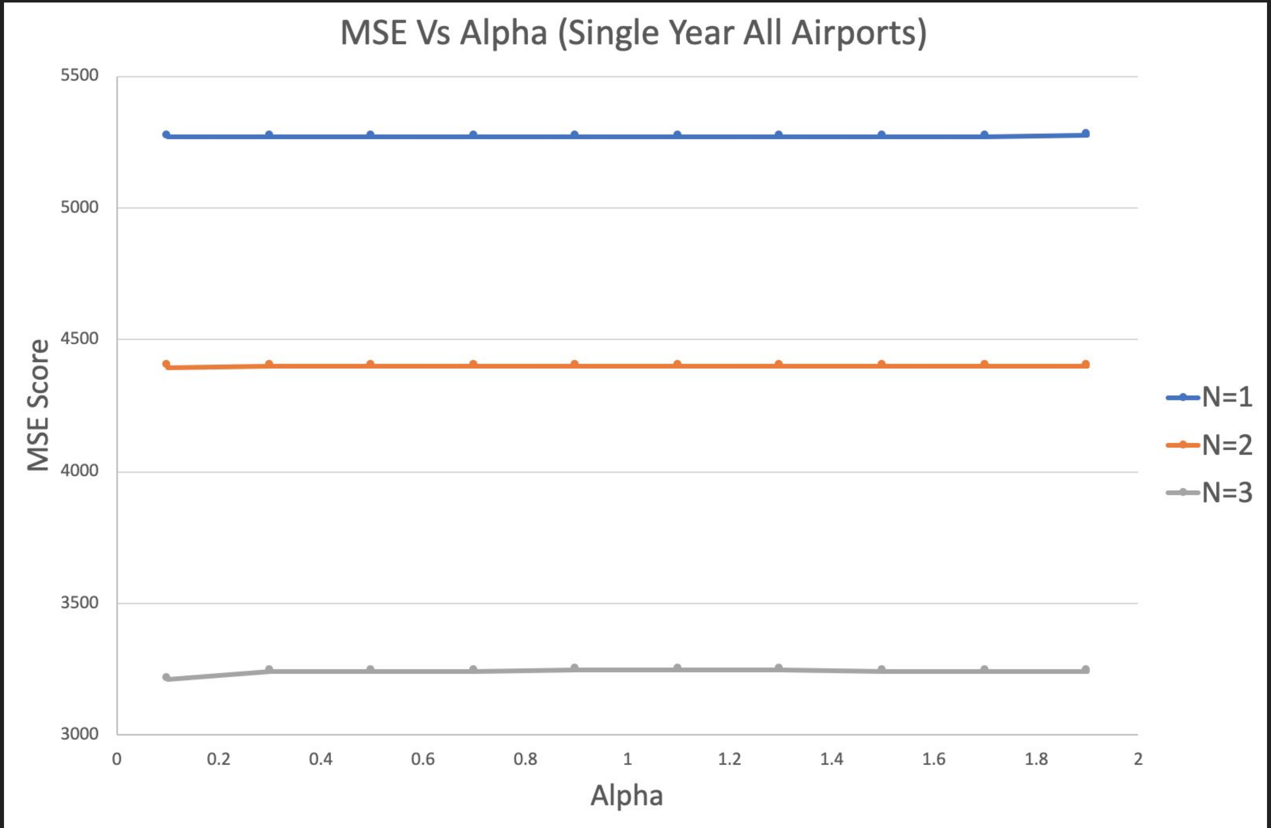
Linear, Polynomial, & Cubic Regressions

<u>Elastic Net</u>

Best MSE = 3865.2
~ 62.17 minutes
N = 3
Alpha = 0.1



Linear, Polynomial, & Cubic Regressions



Magnetic Lasso

Best MSE = 3211.6
~ 56.67 minutes
N = 3
Alpha = 0.1

Random Forest Regression

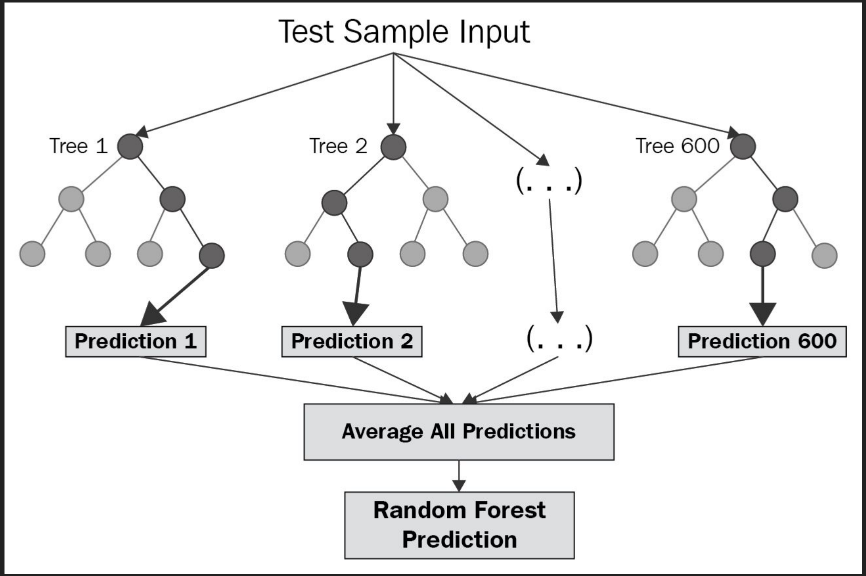
- Create n decision trees
- Each tree is on random sample of data
- Average all of the n predictions

Pros:

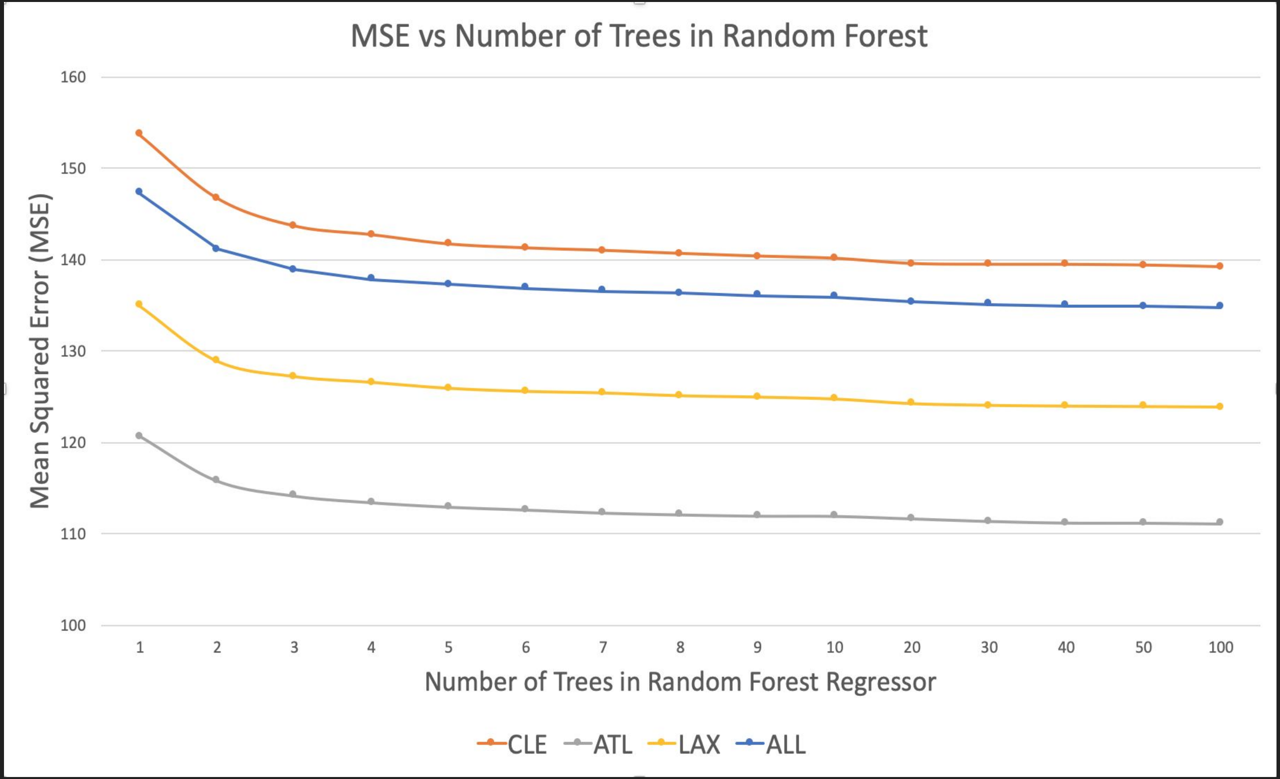
- Highly accurate
- Can handle large data

Cons:

- Can overfit
- Long training times



Random Forest Regression



<u>At 10 trees:</u>

CLE 140.09 MSE
~ 11.83 minutes

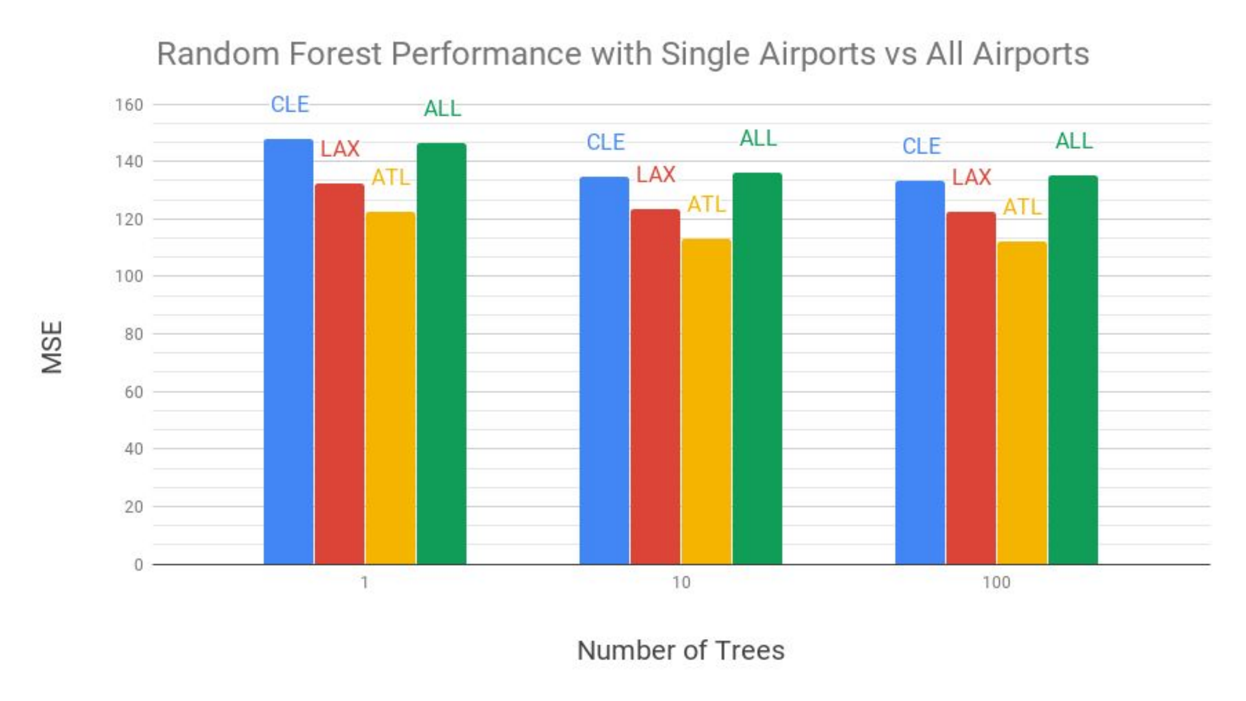
ALL 135.94 MSE
~ 11.66 minutes

LAX 124.72 MSE
~ 11.17 minutes

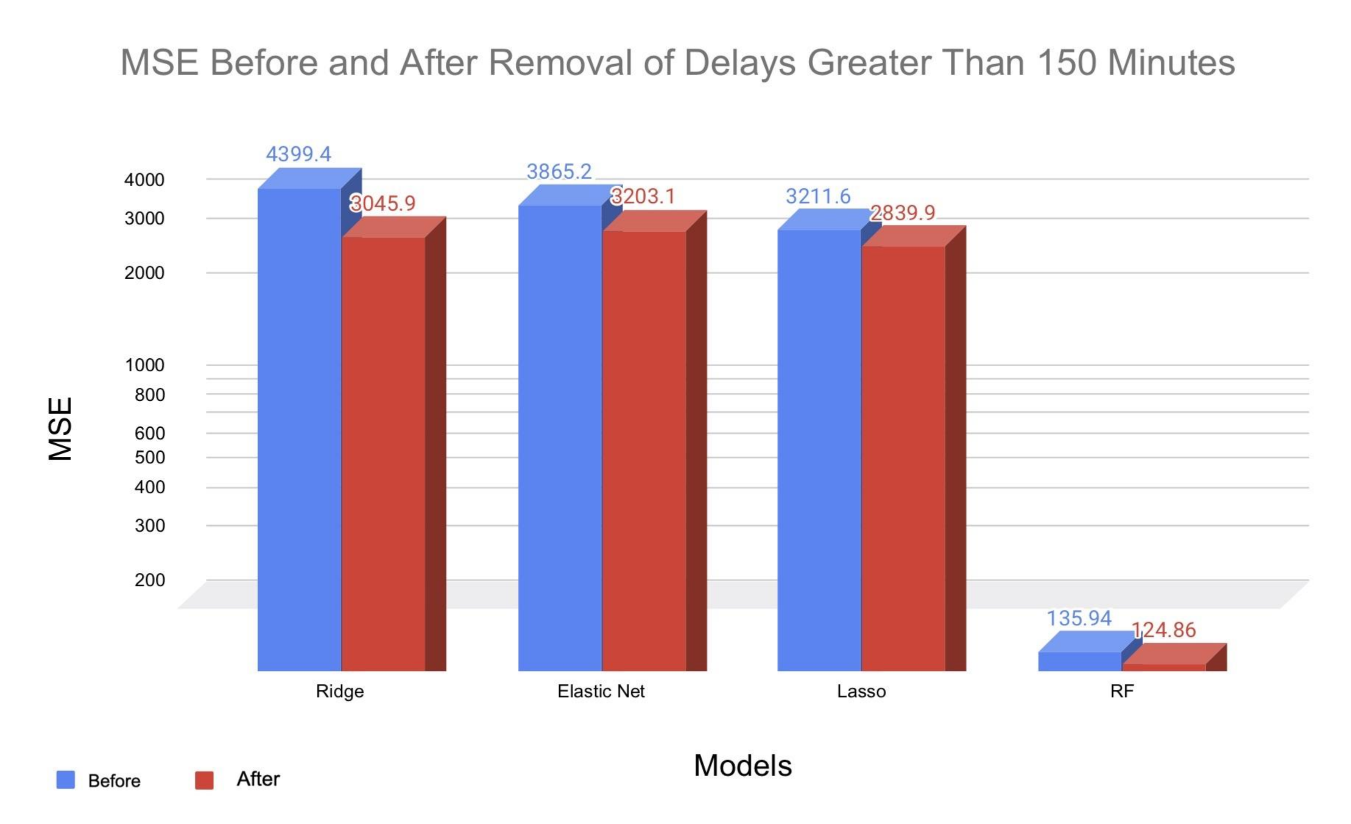
ATL 111.92 MSE
~ 10.58 minutes

Comparing Single Airport Models to All Airport Model

- CLE has 48,385 flights while ATL has 390,046 flights over a 10 year period
- Not realistic for our current dataset to create a model for each airport



Outlier Removal Attempt Overview



Conclusion

- GUI uses a 10-tree Random Forest model create from all airport data
  - Considered a 100-tree model, but it offered negligible improvements with a model file that was 10 times larger
- We learned:
  - Great models take time to create
  - Feature selection
  - Ignoring or forgetting one parameter can completely ruin a model

```
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$
```

Thank You!