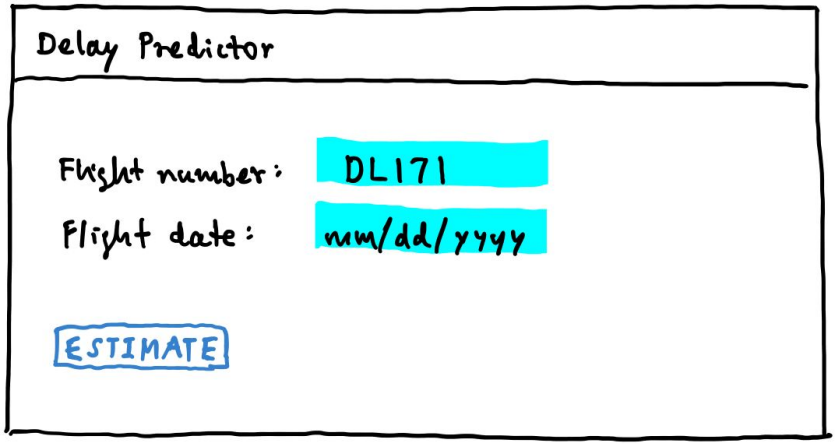
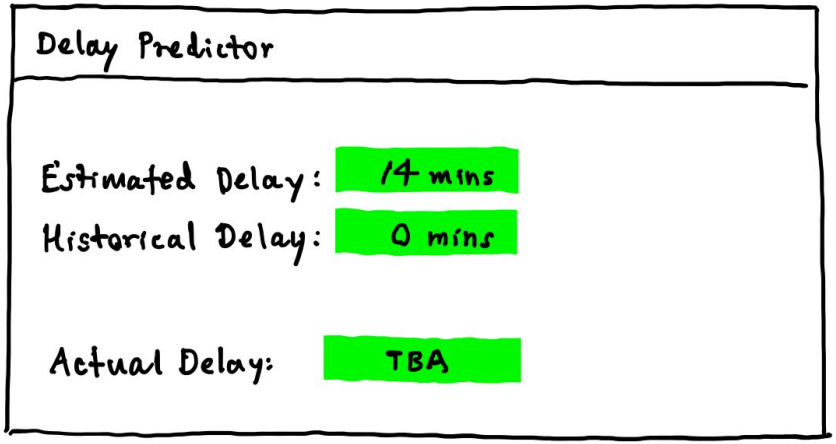
Flight Delay Predictor

Group 9 - Jialiang and Vaibhav

Initial Objective

- Predict if a flight will arrive late
  - Flight number
  - Date of departure
- Makes prediction without knowing the actual departure delay
  - Difference between ours and existing

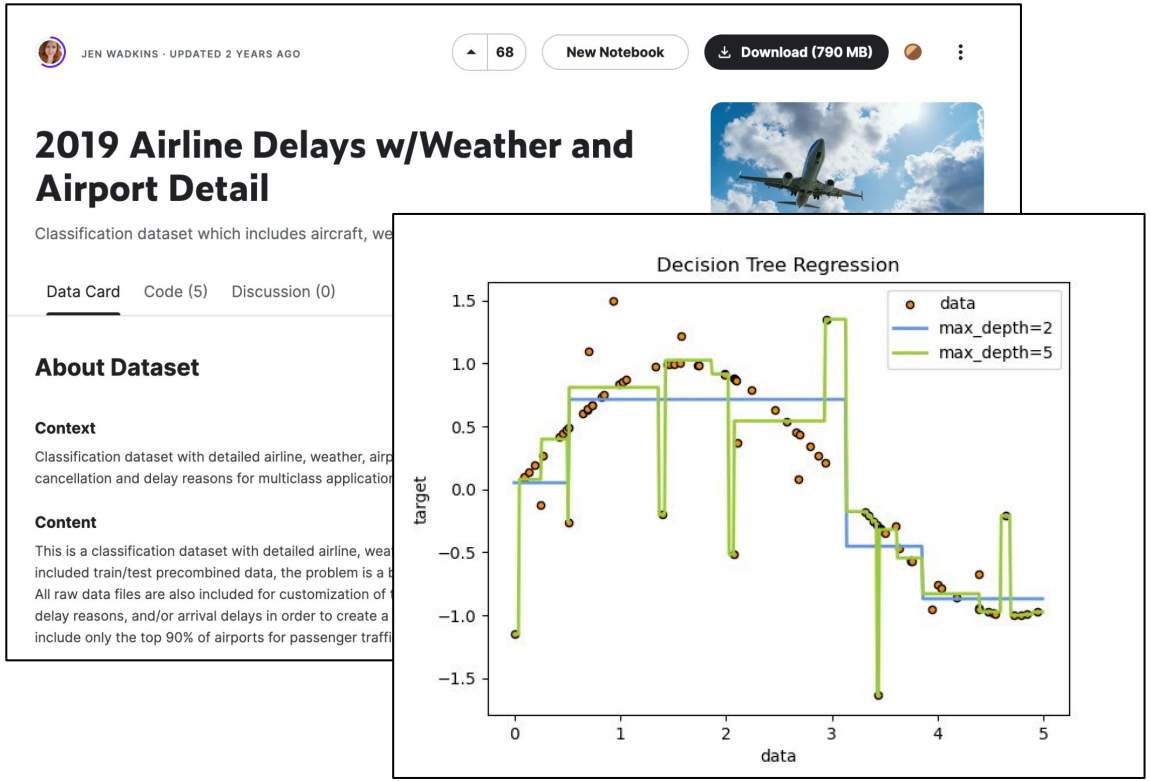




Existing Code and Datasets

We started by checking existing work and datasets for our ML model

- Kaggle
  - Full dataset with all necessary flight information
- scikit-learn
  - Python library with regression tree support
  - Previous examples



Initial Method Chosen: Decision Tree

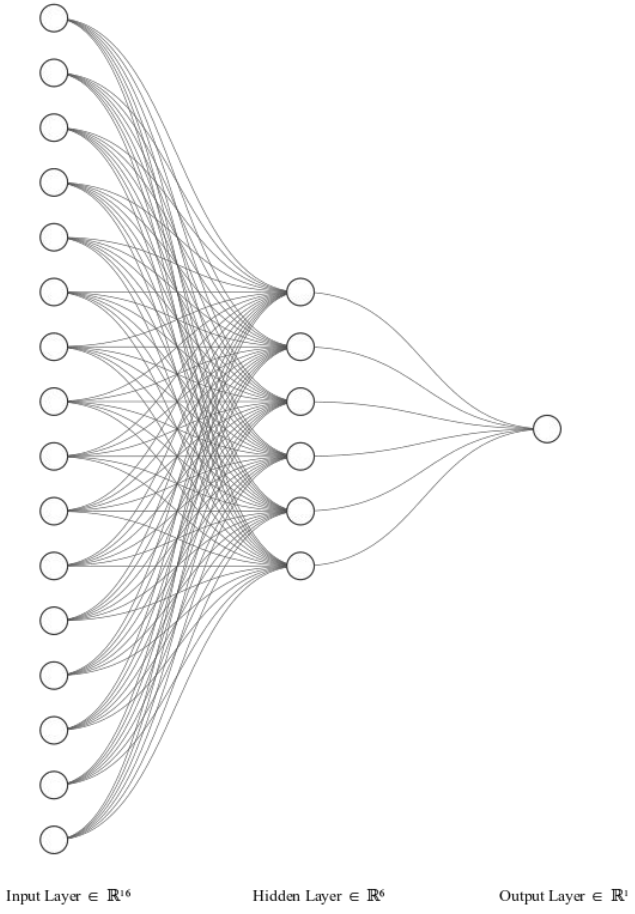
- Regression Tree
  - Kaggle dataset is entirely numerical
  - Lot of data to sort
  - Remove extreme delays
- Most effective method
  - Yuemin Tang, University of Southern California
- Flightlabs
  - Scrape data to validate prediction
  - Up-to-date flight information

Table 2. Results for estimation of flight delays

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.8675	0.8850	0.8675	0.8073
KNN	0.8661	0.7501	0.8661	0.8039
Gaussian Naïve Bayes	0.8487	0.8037	0.8487	0.8184
Decision Tree	**0.9778**	**0.9777**	**0.9778**	**0.9778**
SVM	0.8983	0.9067	0.8983	0.8707
Random Forest	0.9240	0.9250	0.9240	0.9126
Gradient Boosted Tree	0.9334	0.9377	0.9334	0.9237

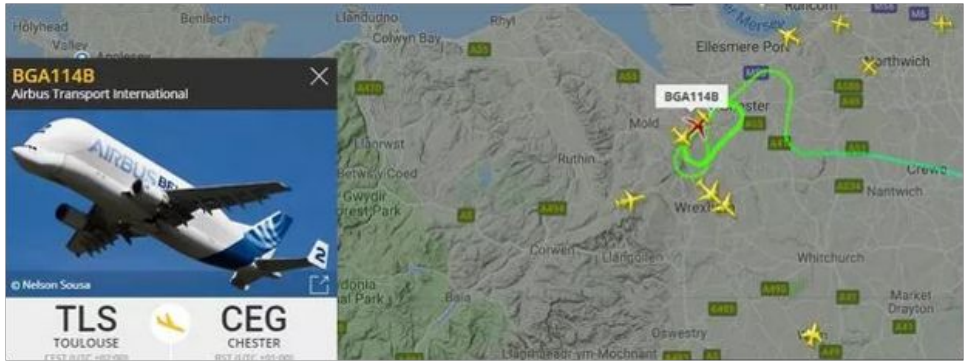
New Method Chosen: ANN

- Why not Decision Tree?
  - Datasets are huge, old, and difficult to clean
  - Kaggle flight performance datasets aren’t always relevant to the flight we are predicting
  - Existing weather data is huge or hard to obtain
- Bespoke model
  - Previous problems are dataset related
  - Obtain live data of a few hundred flights with 6 parameters
  - Creates a small network that is trained in ~90 seconds



Advantage of Live Data

- Accurate portrayal of weather
  - Reflects weather impact on current flights
  - No need for separate weather dataset or source
- Accurately represents systematic airport delay
  - Current flight tracking software just predicts arrival time using interpolation of speed and altitude leading to fluctuating ETA



How the ANN works

- Get data ready by finding the current flight and historical data on it
- Data obtained from API will be JSON objects. Use pandas normalization to convert to DataFrame
- Clean the data to train the model
  - ignore missing data
  - padding
  - tokenization

```
target_flight = flight_number
flightdata = getflight(target_flight)
historical_data = get_historical_flight(target_flight)

historical_flight =
pd.json_normalize(historical_data['flights'])
flight_requested_historical = pd.DataFrame({'operator':
..., 'arrival_delay': ...})
flight_requested = pd.DataFrame({'operator': ...,
'arrival_delay': ...})

...

numeric_cols =
arrivals_cleaned.select_dtypes(include=[int, float])
arrivals_cleaned[numeric_cols.columns] =
numeric_cols.fillna(numeric_cols.mean())
arrivals_cleaned = arrivals_cleaned.fillna('Unknown')

X =
arrivals_cleaned[["operator","departure_delay","aircraft
_type","route_distance","origin.code_icao","destination.
code_icao"]]
y = arrivals_cleaned['arrival_delay']
```

How the ANN works

- After cleaning, we take the following parameters:
  - Operator
  - route(which includes departure and arrival)
  - scheduled departure
  - scheduled arrival
  - departure delay
  - arrival delay
  - route length
  - aircraft type

Note: Operator reflects redundancies in airlines schedules. Route length determine the ability to catch back up in the air. Aircraft type has some relation to do with how many segments they fly a day hence is related to late aircraft delays

How the ANN works

- Finally, the data can be used to train the model
  - 64 input layer
  - 32 hidden layer
  - 1 output (predicted delay in seconds)
- Return the prediction to the user

```
model = Sequential()
model.add(Dense(units=64,
activation='relu', input_dim=6))
model.add(Dense(units=32,
activation='relu'))
model.add(Dense(units=1,
activation='linear'))
model.compile(optimizer='adam',
loss='mean_squared_error')

model.fit(X_train, y_train, epochs=1500,
batch_size=32, validation_data=(X_test,
y_test))

loss = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")

y_pred = model.predict(X_test)
```

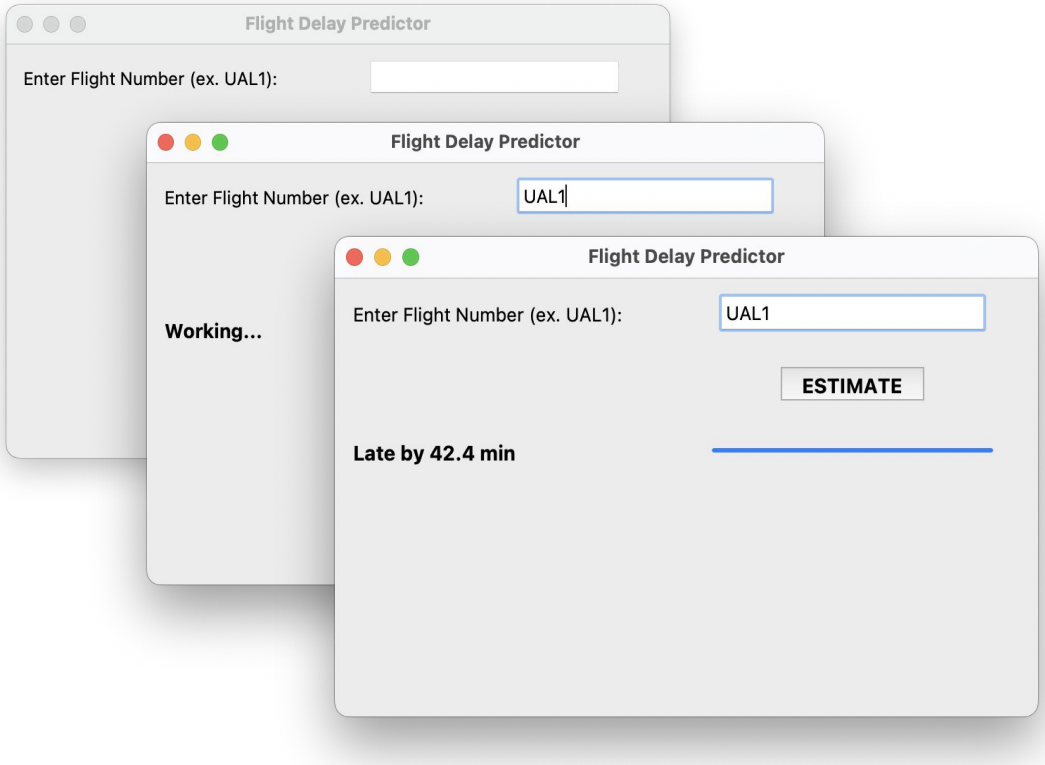
Getting Live Data with FlightAware AeroAPI

- This method requires an API for everything
  - Free
  - Current data
  - Historical data
  - Organized data as JSON
- Using FlightAware API
  - Custom flight data for a lot of parameters like route, aircraft registration, airline, time range, etc., which is ideal for our usage
  - FlightAware is also a flight tracker (Like the more commonly used flightradar24) and you can try it for free

Using the GUI

All you enter is the flight number

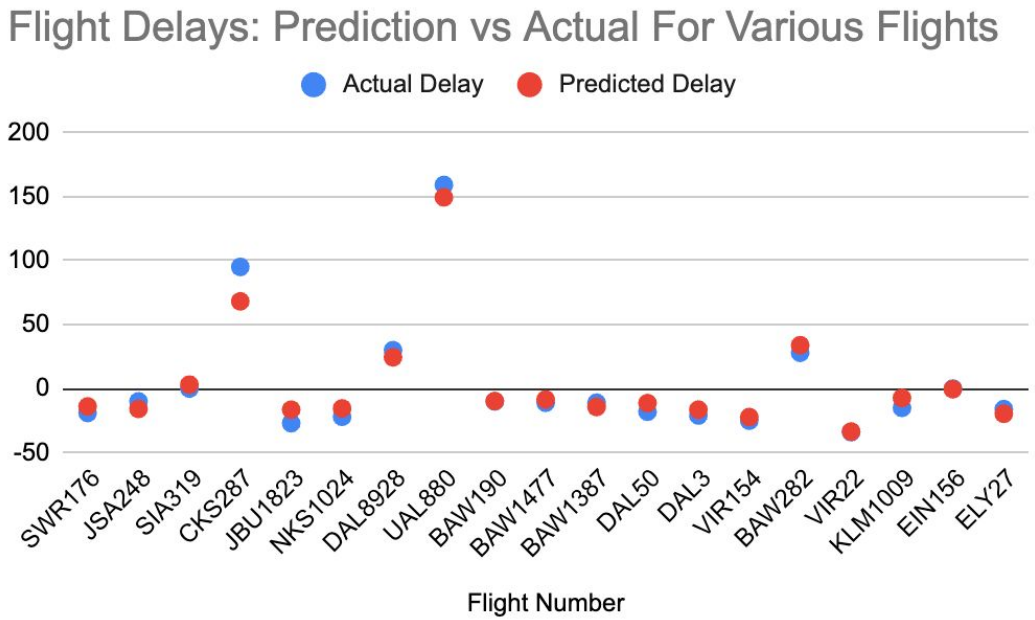
- Don’t need the date
- Trains model and makes delay prediction in about 90 seconds



Prediction Results

Sampled 20 flights from four different arrival airports

- RMSE = 0.7073
- Average accuracy = 80.80%
- Delay or early prediction was always correct



Upcoming

- Further test the performance
- Cache previous models for faster predictions