1. Data preparation and model fitting

The purpose of this project was to create a model that would predict whether a flight would arrive on time or be delayed based on choice of airline and how busy the departure and arrival airports are on an annual basis.

The data used as part of this was found on the Kaggle website and extracted from data that I had previously compiled during my normal course of work. Information about the Kaggle data can be found in the “Project Concept.docx” Microsoft Word document in this folder. The relevant data that I provided consisted of airport operations count information that was a direct measure of how busy each airport is.

The “Flight Delay Model.ipynb” Jupyter notebook file is where the data wrangling logic can be found. Flight, airport, and airline information was loaded. Canceled and diverted flight information was removed. Rows with NULL data, missing data, and invalid Airport identifier information were removed.

After the initial cleaning, the airport, airline and operations data sets were merged. Counts were calculated for the various airlines and one-hot encoding was performed for the airlines with the higher late flight counts. This was used to help decide which columns to use for the model.

A logistic regression model was employed, fitted, and a pickle file was created to serialize this model to disk. The pickle file, named FlightDelayLogRegModel.pik, is included in this directory.

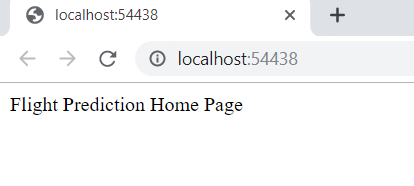
The inputs to this model consist of the origin airport annual operation count, the destination airport annual operation count, and which of the 6 “tardiest” airlines was selected for the respective flight.

1. Flask Deployment

The Flask environment within the Visual Studio 2019 IDE was used to emulate deployment.

The runserver.py file contained in this directory houses the logic used to create the ‘predict’ endpoint used to facilitate a web-based REST method to use the model previously serialized into a pickle file to predict whether a flight will be delayed or not. It also provides a ‘health’ endpoint that can be used to ensure the interface is active by the end user.

To run the project, start the project in Visual Studio 2019. This will display a main page shown below:



Next, open a command window (if in a Windows environment). Below is a sample curl command to call the prediction endpoint and the response from the model contained in the pickle file. In this example, the prediction returned ‘0’, indicating that the flight would not be late (i.e., on time) and the prediction probabilities.

curl -d "{\"OpsOrigin\":\"30000\", \"OpsDest\":\"50000\", \"AA\":\"1\", \"DL\":\"0\", \"EV\":\"0\", \"OO\":\"0\", \"UA\":\"0\", \"WN\":\"0\"}" -H "Content-Type: application/json" -X POST http://localhost:54438/predict

