Predicting Air Passengers
— Report —

Hugo Ehlinger and Maxime Redstone Leclerc
MAP536 Python for Data Science, Ecole Polytechnique

3rd December 2020

1 Introduction

The COVID-19 pandemic has highly impacted the air traffic industry and led to a dramatic plunge in demand leaving air flight companies in turmoil. Models used to price flights were very much challenged while faced with this black swan event. This highlights the importance of having models able to accurately predict the number of passengers on a given flight.

This study aims to predict the number of passengers on a given flight for a ticket reservation company and find data that can be correlated to the number of passengers travelling on a specific flight on a given day. An exploratory data analysis phase was conducted at first to understand the provided dataset. Various external data were used in an attempt to improve the predictions. Finally, different models were evaluated to provide accurate predictions while ensuring reasonable execution times.

2 Exploratory Data Analysis

The provided dataset was composed of five columns: DateOfDeparture, Departure, Arrival, WeeksToDeparture, std_wtd and Passengers where Passengers was the variable to predict. The flights occured between 01/09/2011 and 05/03/2013 and relate to 20 airports in the USA.

Our approach was to leverage information about the different types of routes and the date of travel. Figure 1 shows the average number of passengers for all routes available in our dataset and clearly highlights inherent differences between them. Moreover, looking at Figure 2, we see most routes are gaussian-like which comforted us in the idea that the mean per route was a feature that could be exploited by our model. Figure 3 clearly displays a monthly seasonality in terms of number of passengers traveling. This seasonality was also observed at the week-scale and day-scale.

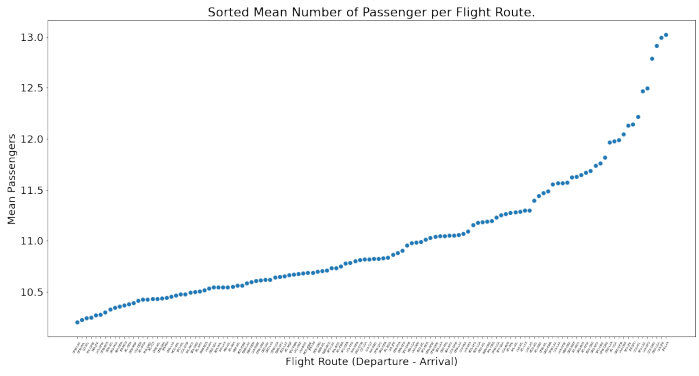

Figure 1: Mean Number of Passengers per Flight Route in USA

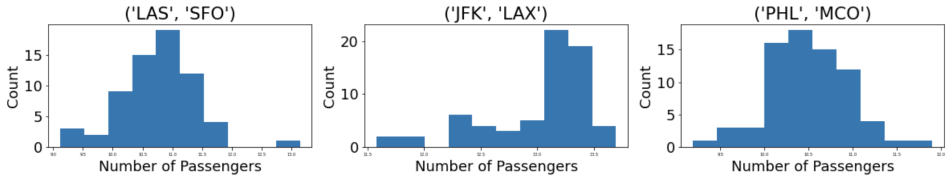

Figure 2: Histograms of Number of Passengers for Different Routes.

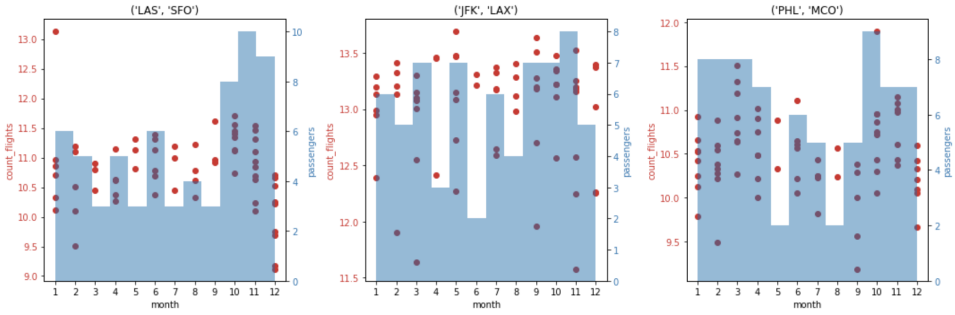

Figure 3: Histograms of Number of Passengers and Flights per month for Different Routes.

Additionally, we noticed that when less passengers travel, less planes fly. The flight company therefore has already optimized the number of planes flying on these specific days and routes in order to maximise each flight's occupancy. This means we are aiming at predicting the optimisation error. Indeed, if their model predicted the number of passengers with 100% accuracy then planes would always be full.

Computing the correlation matrix between the features available to us, we found *Passengers* was not correlated with any of the original features (*weekday* (-0.27), *n_days* (-0.1) and *years* (-0.08)). These low correlations can be explained by the fact the seasonality does not follow a linear trend across months, weeks, day of the year, etc. To efficiently grasp the seasonality of flights and the dependence on the route, we calculated the conditional mean of passengers travelling per route on the one hand and date (month, week, week day and day of the year) on the other hand. We took the encoded date as a series of categorical features and ranked their modalities according to past data in order to transform them into numerical features. Therefore, the model could easily create rules (for decision trees) that would not be based on the initial feature (month number 1-12) but on a linear engineered feature. We believe this was a) pertinent as taking month number (resp. week number or day number) did not follow a linear trend with number of passengers and b) more efficient than one-hot-encoding each of these columns as "categorical feature" as it did not require adding a great number of columns.

3 External Data Search

In our attempt to find other predictors to try and explain the variations of *Passengers* per flight, we assessed the following variables and separated them as seen in our database in Figure 4.

- Weather data - daily weather information per airport (Max TemperatureC, Mean TemperatureC, Min TemperatureC, Dew PointC, MeanDew PointC, Min DewpointC, Max Humidity, ...).
- Economic data - daily oil price, S&P500 stock price and volume of stocks traded, American Airlines stock price and volume of stocks traded, yearly state GDP per capita, state holidays, state unemployment rate, and monthly nation-wide inflation rate.
- Airport data - Longitude, Latitude (used to calculate distance for each flight), population of the airport's closest city (feature embedding to distinguish between airports, more efficient than one-hot-encoding), daily waiting time, customs' booths and flights, monthly load factor (% of full capacity) of passengers on flights.

- Web searches data (Google Trends US) - For each airport, we tracked the search intensity (between 0 and 100) of the keywords *closest city + flight* for the period of interest. We merged this feature both on the date of departure of the flight and on the average date at which passengers booked their tickets.

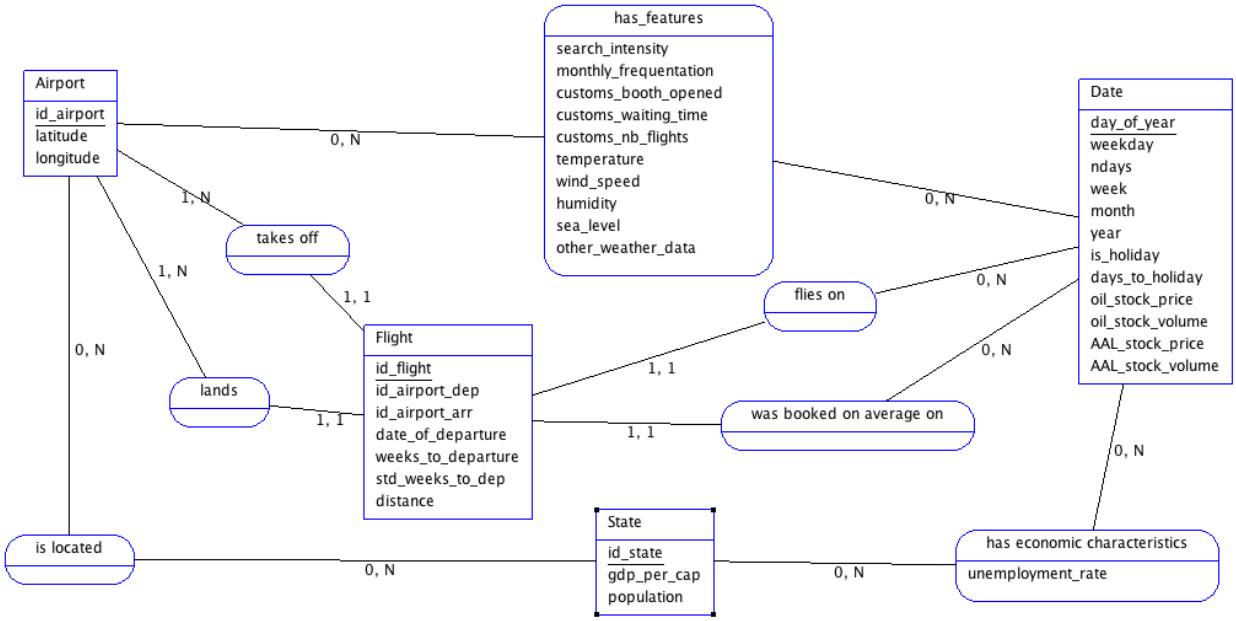

Figure 4: Entity Relationship Diagram for Database.

4 Model Selection

We attempted to use different models with parts or all of the external data throughout the project. We started with linear regression but quickly realised that a lot of features did not have a linear relationship with *Passengers*. We then turned to decision trees which are able to deal with non linearity and allow for better predictions when the data contains a lot of noise. To make sure our predictions were general enough we specifically choose the Random Forest ensemble algorithm. One advantage of using this algorithm was that many parameters can be fined tuned using a grid search approach in order to improve predictions.

To boost the forests capability, we tuned an XGBoost Regressor using the readily available wrapper for Sklearn. We took particular care to evaluate the following hyper parameters: colsample_bytree which specifies the number of features used to construct each tree, learning_rate which ensure more accurate convergence but increases execution time, n_estimators which specifies the size of the forest (allow for more general predictions), max_depth which specifies the maximum depth of each tree (prevents overfitting), min_child_weight which tells the tree when to a leaf is reached and subsample which indicates the fraction of observations used for each tree.

5 Project Architecture

Environment Setup. Github was used as a version control system (VCS) throughout the project and was organised as follows: the *data* folder containing all external data in csv format, the *src* folder containing codes, the *submission* folder where different attempts were logged and a *config* folder used to set up the virtual environment and required libraries for this project. The repository can be found at: `https://github.com/hehlinge42/air_traffic_project`.

Data Management. An artificial database was devised to facilitate the creation of the *external data* file required for the submission pipeline. An Entity Relation diagram is shown in Figure 4. Newly available data was first created in the database (*create_db.py*) before creating the external data file (*gen_ext_data.py*). We defined a MergerTransformer class that could be used on any merger taking the merging parameters as input.

Submissions. Results shown on RAMP have been submitted by team *MaximeRedstone* or *hehlinge42* and are named *claque_au_sol_#* from 1 to 14.

6 Outcomes

This open-ended challenge gave us flexibility in the way to tackle the problem. We focused our efforts on finding data that could be accurate predictors of the number of passengers given the initial dataset. Analysing feature importance plots clearly showed some features were much more useful to the XGBoost forest than others. These include: Weeks_to_departure, route_mean, day_of_year_mean, search_intensity_date_booked, wind_dir_deg_departure, airport_customs_nb_booths. We found that adding the whole set of additional features led to a more accurate prediction although this naturally increased the execution time of our model. As can be seen on our RAMP submissions, *claque_au_sol_13* yields the best RMSE (0.264) but involves a long training and validation time (order of 1000's and 100's respectively) while *claque_au_sol_10* computes a slightly higher RMSE (0.294) but is much more efficient with a training and validation time of the order of 10's and 1's respectively).

Reaching a sound trade-off depends on the business problem at hand. In a context of constantly incoming new data, which could be the case for air travel companies, training/retraining the model has to be done regularly which emphasizes the need for a efficient training process. In a context where predictions accuracy is the main objective, a more complex model could be prefered.