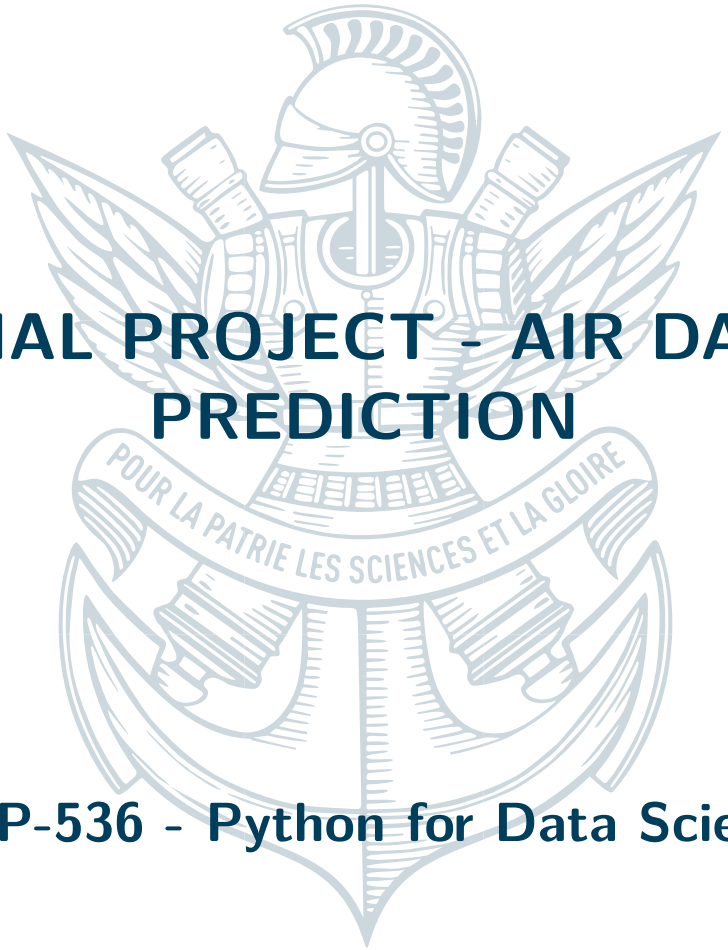


FINAL PROJECT - AIR DATA PREDICTION

MAP-536 - Python for Data Science

December 4, 2019

Leonardo Natale & Guillaume Le Fur

1 Introduction

The objective of this project was to predict a quantity related to the number of passengers on a given flight, on a given date. The data we were provided with initially was the following :

Feature	Description
DateOfDeparture	Date of departure
Departure/Arrival	Departure and arrival airport
WeeksToDeparture	Average number of weeks before departure when the ticket is booked
log_PAX	Variable related to the number of passengers
std_wtd	Standard deviation of WeeksToDeparture

2 External Data and Data Preprocessing

2.1 External Data

We have added the following data:

Feature	Description
jet_fuel	Daily jet fuel price
coordinates	Airport geographical coordinates
gdp	Departure and Arrival GDP
passengers	Monthly flow of passengers between airports
holiday	holiday data in the U.S.

First, we decided to take a step back to really be able to understand the problem, rather than jumping into coding. What is the main driver when purchasing a plane ticket? We would say price. What influences ticket price? Jet fuel costs are an example. On the other hand, people living in cities with a higher GDP per capita are more likely to fly. Holidays are also a main factor when deciding whether to leave your home. At the same time, the overall passenger flow is a meaningful indicator when predicting daily flows.

2.2 Feature Engineering

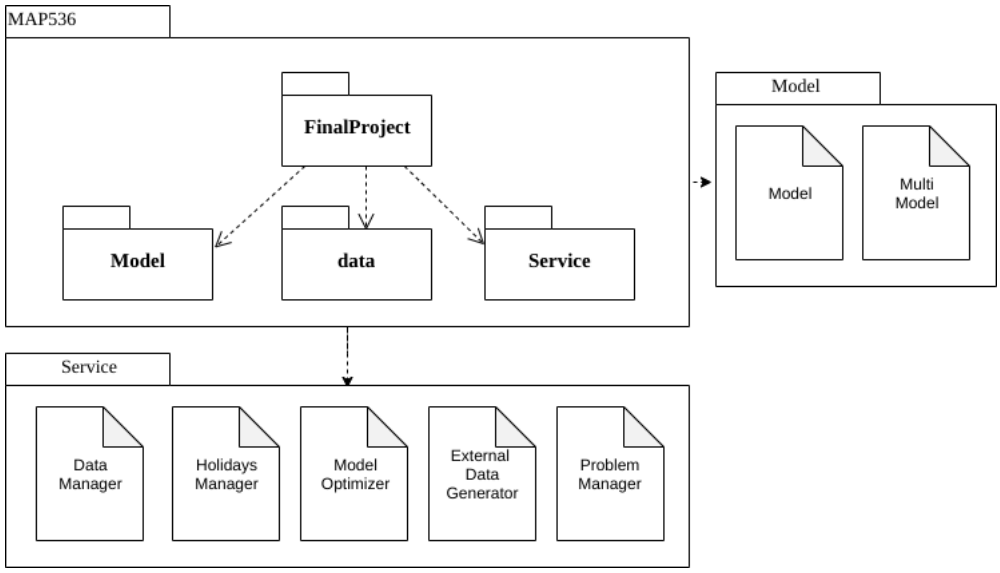
We were able to add the following extra features, by using the data described above:

Feature	Description
closest holiday	Number of days to the closest holiday.
distance	Distance in kilometers between departure and arrival airport.
monthly_logPAX	Monthly log_PAX per airport, both arrival and departure.
weekday_logPAX	log_PAX per airport for every weekday.
avg_wtd	Monthly average on WeeksToDeparture per airport.
avd_std_wtd	Monthly average of std_wtd per airport.

Computing the distance to the closest holiday could help account for seasonality. Kilometer distance between two airports also has a great importance, as we expect airline companies to fill flights that are expensive to operate. LogPAX features are calculated with the target variable from the train data.

3 Infrastructure

In order to test, optimize and integrate our models into the ramp platform, we have come up with our own infrastructure, that enabled us to ease testing and integration of new models. The structure is the following:



The **Data** folder contains all the external data files that we use to create our additional features. The **DataManager** is some kind of an interface between the model and the external data. It takes data as an input (either the train or test data) and merges it with external data, making it ready for fitting. It uses the **ExternalDataGenerator** which groups all our external data into the *external_data.csv* file. It is the class that is responsible for all the feature engineering of our models.
Model is a utility class that stores a sklearn model and two lists : a list of parameters that are to optimize via GridSearchCV and another with parameters that are to be optimized via

RandomSearchCV. When used on RAMP, its role is to contain the model and to fit and predict based on the data passed as an input. **MultiModel** is a class that goal is to simplify the testing of multiple models at the same time. **ModelOptimizer** is an interface, called by Model, that takes care of the RandomSearchCV and GridSearchCV and returns the result to Model. **HolidaysManager** is a utility class used to vectorize operations on dates to determine whether they are holidays or how close they are to a holiday. **ProblemManager** is a class that contains metadata about the problem we intend to solve (columns that are relevant, external data we use, etc.)

4 Models and Tuning

4.1 Models

This table summarizes the train RMSE[1]

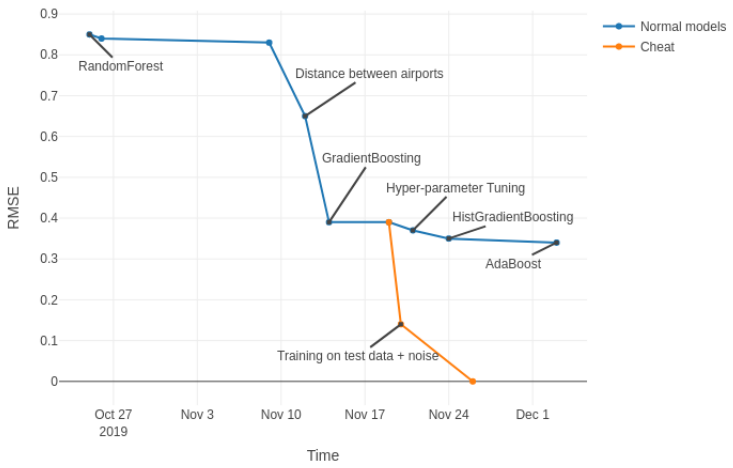
	Train RMSE	Test RMSE	Train time (s)
RandomForest	0.62	0.81	5.6
GradientBoostingRegressor	0.40	0.52	7.67
HistGradientBoostingRegressor	0.27	0.37	5.18
HistGradientBoostingRegressor (Tuned)	**0.11**	**0.35**	**11.8**
AdaBoostRegressor	0.19	0.34	107

The Model we chose in the end is the **tuned HistGradientBoostingRegressor**, mostly because it is a good compromise between accuracy and training time. It is also the model that gives the best results without overfitting too much. Indeed, after a certain point ($RMSE_{test} \approx 0.35$), making the RMSE better by tuning hyper-parameters resulted in a very low training error ($RMSE_{train} < 0.10$), which we did not consider as acceptable. Furthermore, the training time remains pretty low ($\approx 10s$), which makes our model scalable. Indeed, even though we could achieve better performances with models that were longer to train, we though that an acceptable training time for a model fitted on 10000 rows would be around 10 seconds.

4.2 Hyper-parameter Tuning vs Feature Importance

During the project, we fitted several models, performed hyper-parameter tuning and also added features along the way. We thought it would be interesting to keep track of the evolution of the value of our RMSE over time. The following graphs displays the evolution of our RMSE over time, together with reasons for the main changes.

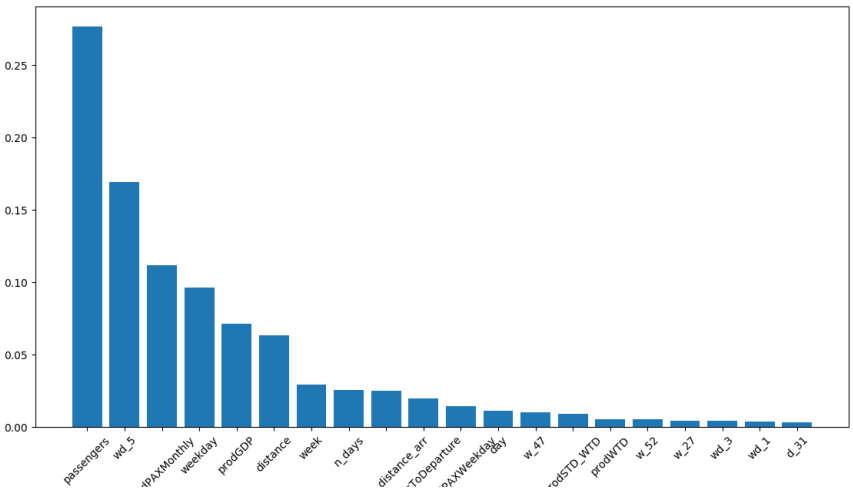
[1]The train and test RMSE values presented here are the one obtained locally as we could not submit the best models before the deadline of the project. Note that the score is often better on the RAMP server.



What we can see is that the main changes were due to adding **new features** to our data, rather than optimizing the hyper-parameters, which often led to overfitting, because of the small amount of data. The methods that were performing best were **tree based methods**.

5 Conclusion

5.1 Model Interpretability



We can extract several relevant information from the above feature importance graph :

- The columns that are an **interaction** between information of the Departure and the Arrival airports are the ones that are the more relevant. For instance, the **distance** between airports is our most relevant predictor. This makes sense as the quantity we are

trying to estimate is the number of passengers going from a place to another, which also integrates the notion of interaction.

- The fact that the date of the flight is a **Friday** is also a really important feature. This can mean that there are many flights on Friday or that the flight habits of passengers change on Fridays.

5.2 Evaluation of Uncertainty in Predictions

Because we only have around ten thousand rows of data, we cannot say that our predictions are extremely accurate. It is all the more problematic that the values of log_PAX are not spread a lot (mostly between 9 and 12). To have a more robust model, we would have needed more data. The lack of data also made it difficult to optimize the hyper-parameters without overfitting.

5.3 Final Comments and Possible Improvements

We did not have time to try the *Time Series* but it would have been interesting to fit a time series on the biggest Departure airports to see how it performed on such data. Apart from the purely Data Science related improvements we could have done, the infrastructure we have designed could be improved to be made more generic and easier to reuse.