MAP-536 - PYTHON FOR DATA SCIENCE

Final Project - Air data prediction

December 4, 2019

Leonardo NATALE & Guillaume LE FUR

ÉCOLE POLYTECHNIQUE

INSTITUT POLYTECHNIQUE DE PARIS

scikit learn

1 External Data and Data Preprocessing

1.1 Getting External Data

The original data consists of :

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

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 is the U.S.

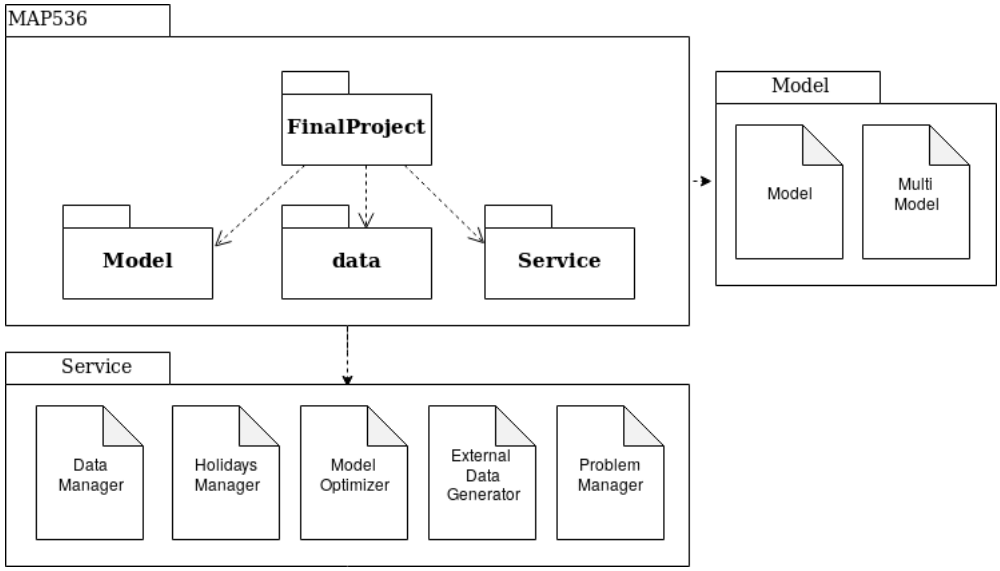
1.2 Feature Engineering

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

- Number of days to the closest holiday.
- Distance in kilometers between departure and arrival airport.
- Airport Dimension
- Monthly log_PAX per airport, both arrival and departure.

2 Structure

(Adjust the size and position, maybe explanation at the right)



- **Data** contains all the external data files that we use to create our additional features.
- **ExtrnalDataGenerator** merges 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.
- **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.
- **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. The main role of this class, when used locally, is to compare the quality of the fit of a model before and after optimization of the hyper-parameters. When used on RAMP, it's 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. When creating an instance, one can pass multiple models and hyper-parameters list. Then all the models are going to be optimized. The predict method can be used in two different ways : either returning separate predictions for each model or either making the average of the models, which can allow to test models that are combinations of several models.
- **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.)

3 Models and Tuning

3.1 Models

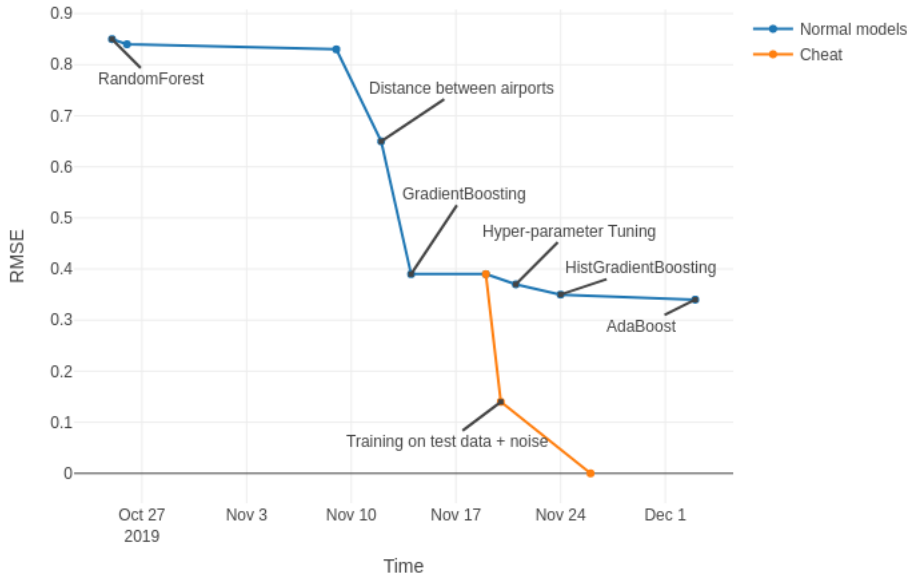
Add a table with comparison between different models.

	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)	0.07	0.35	14.8
AdaBoostRegressor	0.19	0.35	107

The Model we chose in the end is **HistGradientBoostingRegressor**, mostly because it's a good compromise between accuracy and training time. It's also the model that gives the best results without overfitting too much. Indeed, after a certain point ($RMSE_{test} \approx 0.37$), making the RMSE better by tuning hyper-parameters resulted in a very low training error ($RMSE_{train} \approx 0.07$).

3.2 Hyper-parameter Tuning vs Feature Importance

During the project, we tried several models, made some 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, along with the reasons of the main changes.



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.

4 Conclusion

4.1 Model Interpretability

Feature importance graph and comments.

4.2 Evaluation of Uncertainty in Predictions

Because we don't have much data, we cannot say that our predictions are extremely accurate. It's all the more problematic that the values of log_PAX are not spread a lot (mostly between 9 and 12)

4.3 Final Comments and Possible Improvements

We didn't 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.