**Ramp Competition: Number of Air Passengers**

Project Report

**By Paul-Antoine Girard & Adrien Toulouse**

**Username: toulouse\_girard**

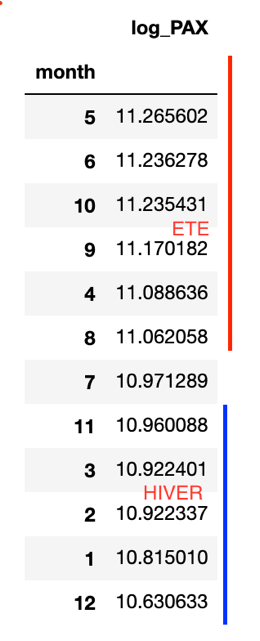
**Final submission name: ???**

*The aim of this report is to narrate our workflow, to detail everything we tried and explain the choices we made to build our best model and predict the number of air passengers.*

## Starting point, descriptive statistics and data preprocessing

Understanding the task and how to proceed was our first challenge, as it was for both of us our first data science competition.

We started by taking a look at the data we were given. We tried to find patterns, to see the correlations between variables, to detect missing values and to see which variables could help us predict the number of air passengers.

Une image contenant capture d’écran

Description générée automatiquement

For example, we started to look closely at the date after the one-hot encoding.

Our analysis showed us that there was a difference in the number of air passengers for the different weekdays and the different months. Therefore, we built a variable ‘*weekend’* and a variable ‘*season’* to better capture this information. These variables improved our score when we used specific regressors such as SVM. However, they didn’t improve our score in our final gradient boosting model, so we didn’t keep them in our final submission.

We also remarked that the variables which represent the date (‘*month’*, ‘*day’*…) weren’t continuous. In fact, if we look at the variable ‘*month*’, December corresponds to 12 and January to 1. So the algorithm doesn’t know that January follows December. Therefore, we designed a model in which we transform the data into two dimensions using cosines and sinus functions to represent the data continuously.

Then, we concentrated our work on how to encode the airports. The first method was to use a one-hot encoding. However, it leads to 40 columns (two columns per airport) and we wanted to reduce the number of columns. Therefore, our first idea was to create a column per airport and put the value 1 if the airplane leaves the airport or -1 if it arrives. The number of columns was reduced by 2. We also searched for tips on the Internet to add even more information to our variables. We found the method of the mean encoding, and we decided to add this information to our external dataset and to use it in our airport encoding.

Finally, by focusing on the reservation variables (‘*WeeksToDeparture’* and ‘*std\_wtd’*), we observed that there is a strong correlation between these two. We tried different methods to reduce the information contained in these two columns. We replaced these two by a column that represents the product. We also tried to use a PCA algorithm with one principal component, but we decided to drop the variable ‘*std\_wtd’* and to create different categories for the variables ‘*WeeksToDeparture’*. We created four groups of equal sizes by ordering the variables ‘*WeeksToDeparture’*, and we encoded these groups from 0 to 3.

## Adding and preprocessing external data

We started to look at the external weather dataset. We tried to find a correlation between variables and the number of air passengers. However, after numerous tries:

* regrouping variables using a PCA algorithm with a smaller number of principal components,
* deleting some redundant information (minimum and maximum of a variable),
* doing operations between them (like multiplying humidity and temperature),

We ended up only keeping the variable ‘*Events’* which we thought could have an influence on the number of passengers*.* As this variable was categorical, we first replaced missing values by 0 and we encoded into three numerical values:

* 0 when there were no particular events,
* 1 for common events such as rain or fog,
* 2 for more important events such as thunderstorms.

Finally, we thought that the weather data wasn’t necessarily the best thing to really improve our model. In fact, the weather on the day of arrival won’t impact the weather during a week of holiday for example.

As the external data we initially had didn’t allow us to explain enough the number of air passengers, we decided to add new external data.

Overall, we added the following four variables:

* Distance in kilometer between the airports: ‘Distance’,
* Number of inhabitants per city: ‘Population’,
* Revenue per state: ‘Revenue’,
* Crude of oil price: ‘Oil\_Price’.

Every time we added a new variable, our technique was to also to challenge the other variables. We didn’t want to keep information if they weren’t relevant. Therefore, each time we added data to our model, we tested to remove others to see if they continue to impact our model or if the accuracy is still the same without them.

* 1. **‘Distance’**

Our first idea was to add the distance between the airports. We had the intuition that longer flights would carry more passenger and we thought that this information could help our algorithm to make a difference between long flights with more passengers and shorter flights.

We therefore found the geographical longitude and latitude of each airport and used a function to calculate the distance between each of the twenty airports.

This information consequently improved our score, so we decided to keep this variable.

We tried at some point to regroup the distance between airports into three categories of flights that we could encode: long flights, medium flights and short flights. However, this encoding did not allow us to improve or score and we lost a bit of precision, so we decided to stick with the distance variable.

* 1. **‘Population’**

Our second idea was that the number of passengers on a flight could be explained by the population of the city of arrival and that flights going to big cities would carry more passengers. Therefore, we created a column with the number of passengers in the city of arrival. We kept it during a long period of our tests, but in the end, we saw that it didn’t explain a high part of our model, so we finally didn’t use it.

* 1. **‘Revenue’’**

Related to the number of inhabitants of a city, we also thought that the per Capita income could represent the resources of a state. The more a state is rich, the more its infrastructure is developed, and the more travelers and workers are coming. However, the score was the same with and without this variable. So, we tried to use it differently and, we created 3 categories. We affected a 0 to the states with a low income, a 1 to the states with a standard income and a 2 for rich states. Badly, this didn’t help our algorithm, and we also forgot this variable.

* 1. **‘Oil\_price’**

Lastly, by thinking at a larger scale, we watched the expenses of an airline company. Most of them are related to the airplanes and their maintenance, the crews, and oil. Since this last expense has a cost that varies enormously, its variations may be reflected in air tickets’ prices. So, we added this variable to our external dataset and tried different techniques to use it.

When we first added this variable, it wasn’t relevant. In truth, the oil price of the day of the flight doesn’t influential on the tickets’ prices since they are established a long time before.

So, we tried another approach and we associated with a flight the oil price three months before it. However, after different try on the number of months before the flight, it wasn't relevant. So, we decided to forget this variable.

* 1. **‘Nb\_flight’**

Add the number of flights per day going to this city

## Choosing a machine learning algorithm: ‘Honey badger don’t give a crap’

After comparing the results from several regressors among which Random forest, SVM and Logistic regression, we decided to go for a gradient boosting algorithm. More particularly, we opted for the *XGBRegressor* as it fitted the best to our data and gave us our best score.

The advantages of the algorithm are:

* f

After several tries, we wanted to find another algorithm that keeps our accuracy but goes faster. We tried to use the *LGBMRegressor*, which is another gradient boosting regressor that is fast in theory. This new algorithm was performant and a lot faster when we used it with the local test (RAMP\_submission\_test). However, when we submitted our work using this regressor, we didn’t get results as good as with the *XGBRegressor*, and the time wasn’t reduced. It went only 0.5 seconds faster. Therefore, we dropped *LGBMRegressor* and kept *XGBRegressor*.

## Parameters tuning

Every time we tried a new model or tried an existing model on new data, we tuned and adapted the parameters of our regressor to fit the data as well as possible. To do so, we used a cross-validation GridSearch based on minimizing the RMSE.

For our best model, we used the following parameters:

* Max\_leaf:
* Describe most important parameters and their effects

## Score obtained, lessons learnt and conclusion

***In the end, our model obtained a RMSE of ??? and ranked ??? in the RAMP competition.***

Our good score was a combination of all the things we tried during these several weeks of competition: adding new useful variables to our dataset, preprocessing all our variables, choosing the best machine learning algorithm and its best parameters to fit our data.

Our workflow was very circular: every time we added new data or did relevant feature preprocessing, we would challenge other variables in the actual model, and analyze how our score was impacted. We also tried to choose the best regressor and adapt it so that it would fit the dataset.

We tried to improve our score at every step of our workflow and would continuously go back and forth all the steps.

Overall, this project was particularly interesting to understand the importance of data preparation and the particularities of the different regressors. It was also fun to look for new external data and see how they could improve our model. Undoubtedly our model could still be improved by adding more data and going deeper into parameter tuning