Number of Flight Passengers Prediction

DST Guild

Marouane NAJID **Asmae SOUGHOU**

The goal of the challenge is to predict the number of passengers per plane on some flights in the US. The data is provided to us by a single company. This is a supervised regression problem.

From the company point of view, the interest of this challenge is to be able to evaluate the percentage of no-show reservations, to properly calibrate overbooking.

Some passengers make reservation but do not show up on the flight, leading to empty seats in the plane. Estimating the number of passengers effectively boarding the plane is thus important for the company. The left-out data has dates that come after the training data, so a time series approach is possible.

I. Introduction

The flight passengers competition organized by Outcoder aims to predict the number of passengers per plane for some US flights, based on the following training dataset:

	flight_date	from	to	avg_weeks	target	std_weeks
0	2012-06-19	ORD	DFW	12.875000	12.331296	9.812647
1	2012-09-10	LAS	DEN	14.285714	10.775182	9.466734
2	2012-10-05	DEN	LAX	10.863636	11.083177	9.035883
3	2011-10-09	ATL	ORD	11.480000	11.169268	7.990202
4	2012-02-21	DEN	SFO	11.450000	11.269364	9.517159

```
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   flight_date  8896 non-null   object
 1   from         8896 non-null   object
 2   to           8896 non-null   object
 3   avg_weeks    8896 non-null   float64
 4   target       8896 non-null   float64
 5   std_weeks    8896 non-null   float64
```

The dataset above contains basically 5 features and the target:

- Flight_date: an object variable that represents the date of the flight in the format YYYY-MM-DD with no missing values. Its type will be converted to datetime afterwards to perform feature engineering on it

- From: an object variable that represents the code of the departure airport and that has no missing values
- To: an object variable that represents the code of the destination airport and that has no missing values
- Avg_weeks: a float variable of the average number of weeks between booking and flight date, across passengers. This column has no missing values
- Std_weeks: a float variable of the standard deviation of number of weeks between booking and flight date, across passengers. This column has no missing values as well
- Target: a float variable representing a transformation of the number of passengers boarding the plane, with no missing values

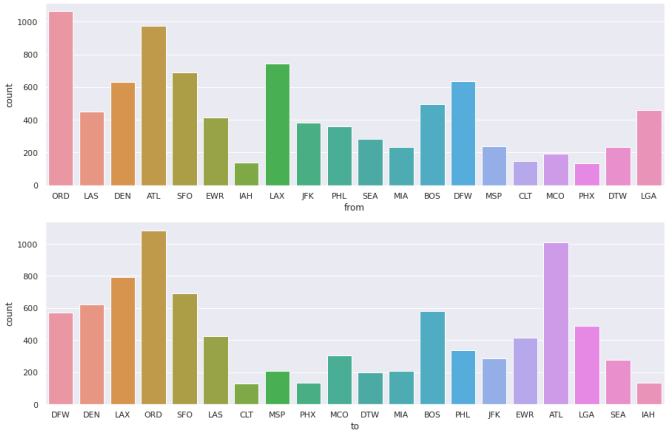
As we can see, the dataset has few variables and needs a genuine feature engineering to enrich it and help perform the machine learning models, but before jumping to the feature engineering, let's enjoy the exploratory data analysis and better discover what we have.

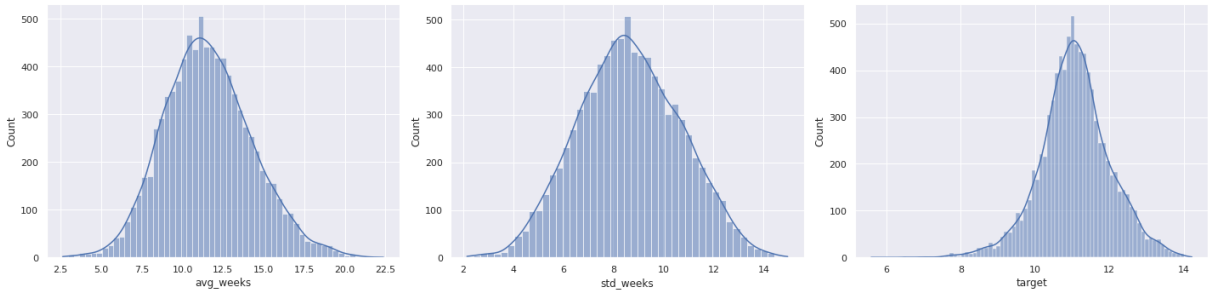
NB: the execution of the models Notebook took more than 18 minutes 😉.

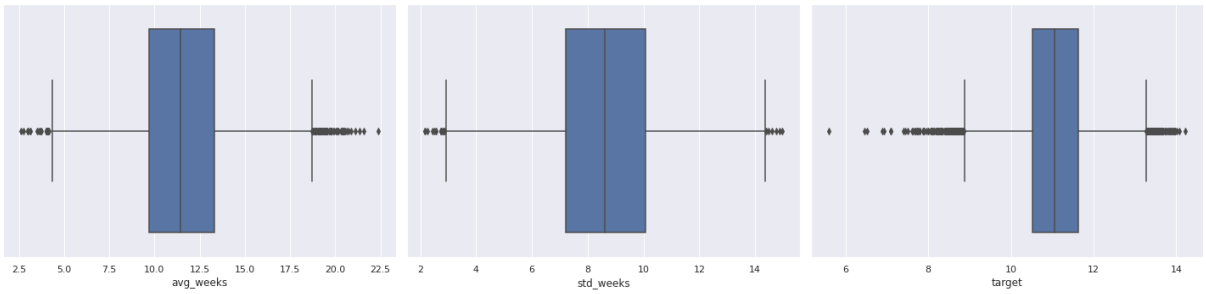
II. Exploratory Data Analysis

1. Distribution of the features

The following graphs shows that the categorical variables "**from**" and "**to**" are well distributed, with no neglectable part of any categories. Concerning numerical variables, we can see that they are in range of 0-20 with no outliers, so no need to scale the data.

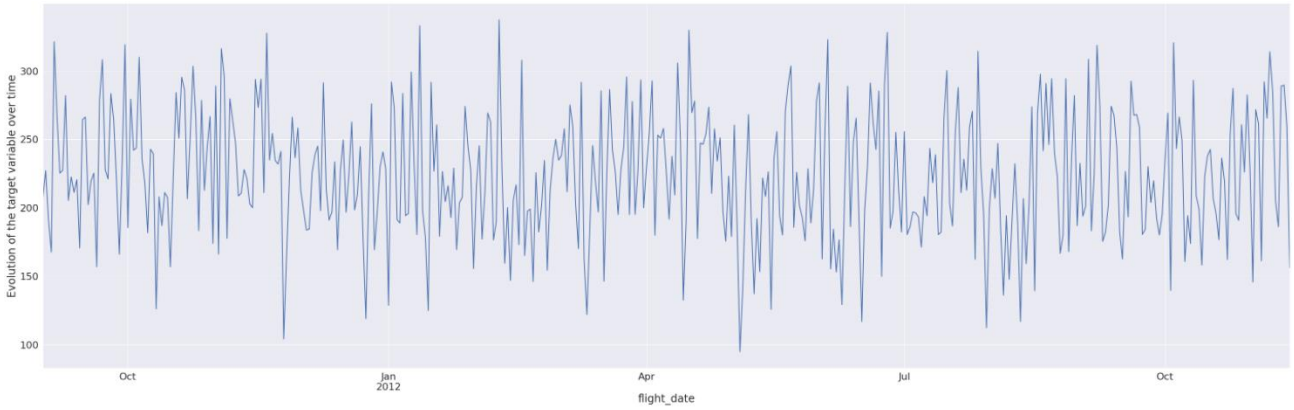


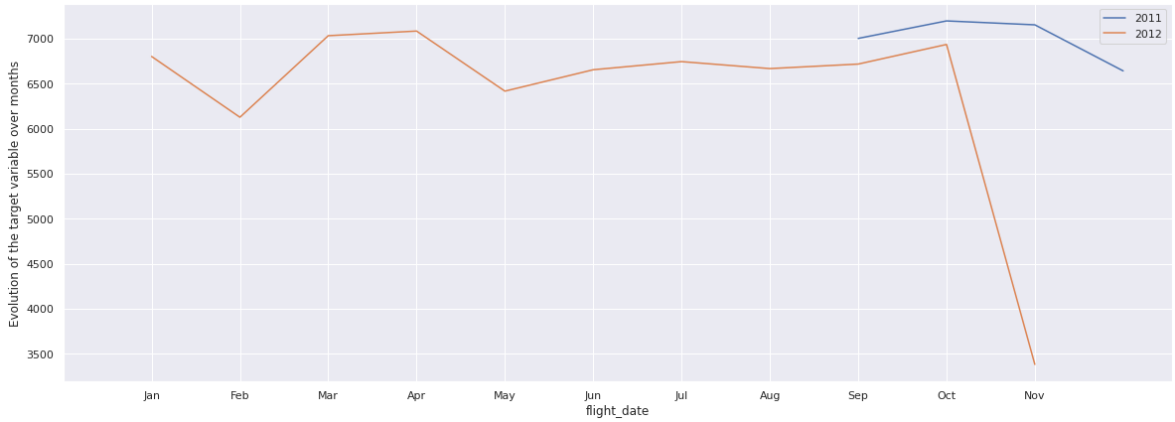




2. Evolution of the target variable over time:

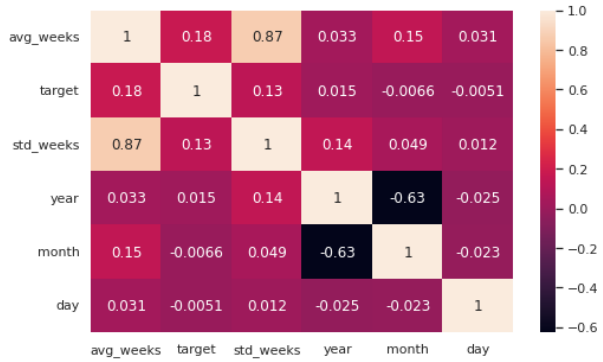
The evolution of the target variable over time represented on the graph below shows continuous fluctuations of the target variable which makes it hard to notice a clear trend of its evolution. However, there are some similarities in the curve that let us make hypothesis about an eventual correlation between the flight date and the target.





The curve above shows that the target variable reaches its maximum in April for the year of 2012, but we can't judge the year of 2011 due to the lack of data.

3. Correlation



We can observe that our target variable is best correlated with **"avg_weeks"** and **"std_weeks".** However, we don't see any significant correlation with the year, month, or day variables, which led us to think of adding new variables related to the flight date.

II. Feature Engineering

The main difficulty of the challenge is the limited amount of information per flight (the data is "thin", it has few columns). In order to improve our score, we will look for new variables that correlate with our target.

1. Flight date

The first approach is that we can treat our data as a Time Series to see the impact of 'flight_date'. After a correlation and an autocorrelation test, we can see a pseudo seasonality in our data. To get benefits from this ascertainment, we decided to do some feature engineering on this variable by creating many new variables: we split our date into **"year"**, **"month"** and **"day"** (the year variable isn't that important because our train data has only two years, 2011 and 2012, but our test has other ones, 2012 and 2013). Also, we generate other variables like, **"julian"** (number of days between '2010-01-01' and the given date), "**quarter**", **"week"** and **"dayofweek"**, this last one turned to be very interesting for our model.

2. Distance

Taking into consideration that the airline does not refund passengers that doesn't show, we can make a hypothesis that expensive trips have a low no-show rate, in other way there is a dependency between no-show rate and ticket fares. After many attempts, we didn't find the mean price for a specific trip and for a specific date, so we decided to use another approach. In a way or other there is a dependency between the ticket price and the distance of the trip. So, we used an API to determine the GPS coordinate ("**from_longitude**", "**from_latitude**", "**to_longitude**" and "**to_latitude**") of each IATA and then calculate the distance ("**distance**").

3. Fare

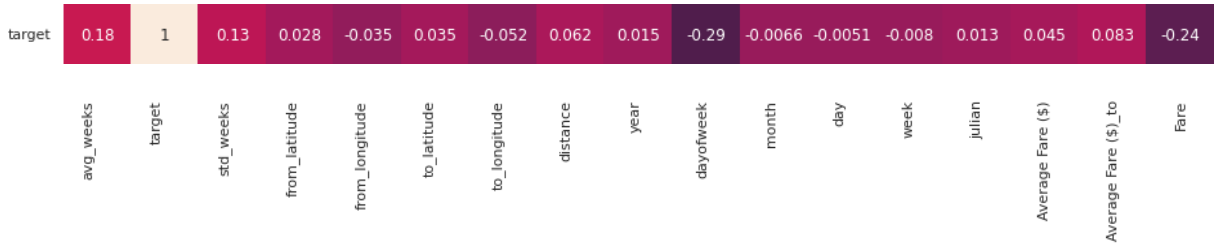
For the same reasons that we considered for the distance variable, we decided to follow another approach to determine the mean price of ticket for trip by quarter, and it's the

"Fare" variable.

The added columns are as the following:

from_latitude	from_longitude	to_latitude	to_longitude	distance	year	dayofweek	month	day	week	julian	Average Fare ($)	Average Fare ($)_to	Fare
41.974163	-87.907320	32.899810	-97.04034	1289.698532	2012	1	6	19	25	900	385.05	430.71	3.842887
36.084000	-115.153740	39.856100	-104.67374	1010.700812	2012	0	9	10	37	983	269.62	320.91	3.956805
39.856100	-104.673740	33.941590	-118.40853	1386.808962	2012	4	10	5	40	1008	320.91	401.84	4.405865
33.640728	-84.427704	41.974163	-87.90732	974.145965	2011	6	10	9	40	646	368.74	378.78	2.401525
39.856100	-104.673740	37.621310	-122.37895	1556.562600	2012	1	2	21	8	781	320.91	408.14	4.479947

The columns "**from**" and "**to**" are converted afterwards to new binary columns using Get_dummies method.



We can notice that the target variable is more correlated with the new variables "**dayofweek**" and "**Fare**". So we'll try with this dataset for the next part.

III. Machine Learning models

For this challenge, we tried three different machine learning model to predict the target. First, we started with Linear Regression, then we moved to Random Forest and finally XGBoost. The two latter models are the ones we focused on as they give the best results. In fact, we were inspired by the *PyCart* library that gives the following results after performing baseline models:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
catboost	CatBoost Regressor	0.2455	0.1096	0.3306	0.8777	0.0288	0.0228	1.7060
xgboost	Extreme Gradient Boosting	0.2576	0.1214	0.3479	0.8645	0.0303	0.0240	0.9840
lightgbm	Light Gradient Boosting Machine	0.2605	0.1242	0.3520	0.8612	0.0305	0.0242	1.0670
rf	Random Forest Regressor	0.2987	0.1761	0.4190	0.8033	0.0362	0.0278	1.3360
et	Extra Trees Regressor	0.3042	0.1871	0.4318	0.7910	0.0374	0.0284	1.3940
gbr	Gradient Boosting Regressor	0.3519	0.2124	0.4603	0.7628	0.0395	0.0325	0.8200
dt	Decision Tree Regressor	0.4227	0.3564	0.5965	0.6001	0.0515	0.0392	0.0500
ada	AdaBoost Regressor	0.5314	0.4549	0.6737	0.4919	0.0563	0.0484	0.4420

To improve the results, we tuned our hyperparameters using the RandomSearchCV method, and we obtained the best submission thanks to XGBoost.

IV. Conclusion

To conclude, we can say that the most important feature in our dataset is the **"flight_date"** that allowed us to add new important features such us **"dayofweek"** and **"fare"**, that both have significantly improved our score.