TP2
Pandas, data analysis library

ENSAE 2021/2022

Mastère Spécialisé - Data Science

Réda TAWFIKI - Jeffrey VERDIERE

Contents

1 PART 1 : Visualization

1.1 Question 1

To re-order the previous barplot using standard month ordering, we use the `pd.Categorical()` function of pandas which represents a categorical variable in statistics. It takes on a fixed and limited number of possible values, like months. In the case of categorical variables, logical order isn't the same as categorical data but the sorting of these variables uses logical order. The parameter *val* takes for value dataframe's column "month". We put the standard month ordering in *categories*, and we put *ordered* as true so that the categorical is treated as ordered. Then, the function will return the categorical variable using the standard month ordering that we wanted.

1.2 Question 2

There is a much higher cancellation rate in the City Hotel than in the Resort Hotel. Indeed, the cancellation rate for the City Hotel is around 40% while it is around 15-30% for the Resort Hotel. Concerning the City Hotel, the most cancellations seem to happen in spring (April, May, June and October) while for the Resort Hotel, it seems to be during summer (June, July, August and September). The months with the lowest cancellation rates in the City Hotel are January, February, March and November, while for the Resort Hotel they are January, March, November, December, therefore usually during winter for both hotels.

City Hotel's cancelation rate seems to be quite consistent over the year, always around 40%. The season / months doesn't seem to have a significant impact on it.

However, for the Resort Hotel, the cancelation rate is higher during Spring Summer (around 30%) than during Autumn and Winter (15-20%).

1.3 Question 3

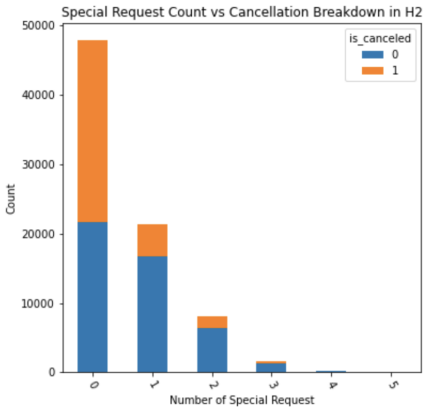
We created two datasets, for each hotel. `.groupby("country")` groups observations by the country of origin and `.count()` counts the amount of reservations for each country of origin. Then, we sort the values by descending order. The most and the second most common country of origin for reservations of City Hotel are Portugal and France while for Resort Hotel Portugal and Great Britain.

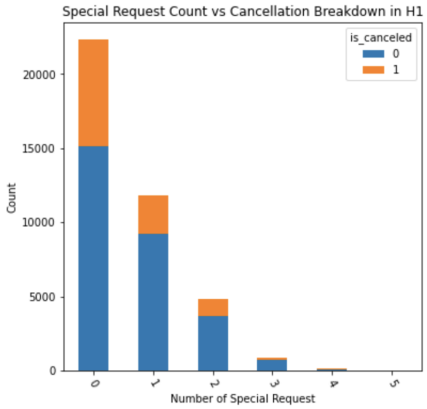
1.4 Question 4

In order to show the counts of observations in each categorical bin (repeated guest or not) using bars, we used the `seaborn.countplot()` method. We can see that most guests in these two hotels are not repeated and that the repeated guests are less likely to cancel.

1.5 Question 5

Most of the reservations in the City Hotel (H2) have no special requests and the cancellation rate in this case is almost 50%. However, when special requests are made, the cancellation rate is significantly lower.
Concerning the Resort Hotel (H1), the same observations are made. Indeed, most of the reservations have no special requests and in this case the cancellation rate is around 30%. However, as for the City Hotel, when special requests are made, the cancellation rate is significantly lower.





2 PART 2 : Machine Learning

2.1 Question 1

In some datasets, we can encounter variables that contain numbers with no specific order. We might confuse our model into thinking that a column has data with some kind of order or hierarchy, when we don't have it. To avoid this, `OneHotEncoder` allowed us to encode categorical features as a one-hot numeric array. One hot encoding takes a column which has categorical data and then splits it into multiple columns. The values are replaced by 1s and 0s, depending on which column has which value.

Human-Readable → Machine-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

In our case, if the feature is categorical and is missing, we chose to define a new category and call it `Not defined`. We created a pipeline to sequentially apply the list of transforms indicated in the parameter *stepslist* on our categorical data. Firstly, we apply `SimpleImputer` which is an imputation transformer for completing missing values by `Not defined`. Secondly, we apply `OneHotEncoder` to encode our categorical features as explained.

2.2 Question 2

We know thanks to the first TP that proper normalization can resolve the convergence problem. Indeed, when an optimization algorithm does not converge, it is usually because the problem isn't well conditioned due to a poor scaling of the decision variables.
We chose to insert a normalization step in the pipeline by using `StandardScaler()` to normalize the numerical data so that the problem gets better conditioned. This can speed up convergence. `StandardScaler()` scales data by removing the mean and scaling to unit variance. It assumes the data is normally distributed. The formula is $x_{scaled} = \frac{x-\mu}{\sigma}$. This solution resolved the warning.

2.3 Question 3

We modified the code by inserting a Random Forest algorithm, our favorite Machine Learning method. (cf. code in the notebook and appendix for more explanation).

3 PART 3 : The Homework

3.1 Our approach

To solve this problem, we use the feature `required_car_parking_spaces` which is the number of car parking spots required by the customer to create our target variable. Indeed, we created a binary variable `homework_target` that expresses the need or not to have a parking spot. For each observation, if `required_car_parking_spaces` is equal to 0, we set `homework_target` to 0; otherwise we set it to 1 no matter how many spots the customer wants:

$$\texttt{homework\_target} = \begin{cases} 1 & \text{if} \quad \texttt{required\_car\_parking\_spaces} \neq 0 \\ 0 & \text{if} \quad \texttt{required\_car\_parking\_spaces} = 0 \end{cases}$$

Then, we created a binary classification problem where the goal is to predict if a customer will need a parking spot or not. We will study the available features to create a classification model which can answer the problem.

3.2 Data exploration & feature engineering

3.2.1 Descriptive statistics

First of all, we checked some basic information such as the number of observations (119390) and the number of features (32) and their type. Then, we displayed some basic statistics about each feature (see table 1). We also checked the number of NaN values per feature which goes as follow:

Variable	Number of missing values
`children`	4
`country`	488
`agent`	16340
`company`	112593

Table 1: Features with amount of corresponding missing values

We created the date type variable `arrival_date` which combines the variables `arrival_date_day_of_month`, `arrival_date_month` and `arrival_date_year`. This variable simply gives us the date when the customer is supposed to arrive at the hotel (the date when the booking starts).
We also created the variable `reserved_assigned_room_type` which synthesizes the information contained in the features `reserved_room_type` and `assigned_room_type` as follow:

$$\texttt{reserved\_assigned\_room\_type} = \begin{cases} 1 & \text{if} \quad \texttt{reserved\_room\_type} = \texttt{assigned\_room\_type} \\ 0 & \text{otherwise} \end{cases}$$

Finally, we decided to drop the following features :

is_canceled
lead_time
arrival_date_year
arrival_date_month
arrival_date_week_number
arrival_date_day_of_month
country
agent
company
reserved_room_type
assigned_room_type
required_car_parking_spaces
reservation_status
reservation_status_date

since we assumed them unknown to train our classification algorithm or no longer useful to our analysis because of the new variables we created. Indeed, we assumed that we wanted to send a SMS notification to customer in need of a parking spot before they arrive at the hotel. The best timing would probably be to send it at the time the customer receives his confirmation message and suggest him to change his reservation right away to include a parking spot. Thus some information provided by this data set is unknown at the time we want to target customers likely in need of a parking spot.

To sum up, we have 21 features left in our dataset: 11 are numerical, 9 are categorical and 1 has a date type (`arrival_date`). A definition of each of them is available in the appendix.

Numerical variables	Categorical variables
stays_in_weekend_nights	hotel
stays_in_week_nights	meal
adults	market_segment
children	distribution_channel
babies	is_repeated_guest
previous_cancellations	deposit_type
previous_bookings_not_canceled	customer_type
booking_changes	reserved_assigned_room_type
days_in_waiting_list	homework_target
adr	
total_of_special_requests	

Table 2: Types of variables

3.2.2 Univariate analysis

For the univariate analysis we first plotted a pie chart of the occurrences of each label of the target variable. It clearly appears on figure 2 that in almost 94% of the case a customer won't need a parking spot which shows that our data set in really unbalanced. We will tackle this issue by performing an under-sampling as explained later. Then we plotted the histogram of each numerical variable to see their distribution and similarly we plotted horizontal bar charts of the categorical variables.

3.2.3 Bivariate analysis

We checked the correlation between our numerical features by computing a correlation matrix. The figure 22 shows us the results and we will assume no colinearity between our features according to these results even

though the variable `stays_in_weekend_nights` and `stays_in_week_nights` are correlated at a 50% level. We also computed Chi-square test between each categorical variable and the target variable. The Khi 2 independence test can be written as follow:

H_0 : The 2 variables aren't independent $\quad H_1$: The 2 variables are independent

X_i	χ^2	Degrees of freedom	pvalue
`hotel`	5808.20	1	0.00
`meal`	332.49	4	0.00
`market_segment`	4087.27	7	0.00
`distribution_channel`	2757.93	4	0.00
`is_repeated_guest`	674.98	1	0.00
`deposit_type`	1102.67	2	0.00
`customer_type`	590.13	3	0.00
`reserved_assigned_room_type`	785.60	1	0.00

Table 3: Khi 2 statistics

Since our p-values are always null we reject H_0, thus our explanatory variables are all correlated to our target variable.

Finally, we looked at the evolution of our target variable trough time and it appears on the figure 23 that there might be some seasonality in our data. We won't dig further into these results.

3.3 Data Processing & data cleaning

3.3.1 Missing values

Thanks to the feature engineering we did in the previous part we deleted 3 of the 4 variables with missing values which are: `country`, `agent` and `company`. We considered that the variable `country` didn't bring any valuable information to our model since it wasn't correlated to the target variable and the variables `agent` and `company` were highly correlated to the variables `market_segment` and `distribution_channel` so we dropped them from the data set to avoid introducing endogeneity in our model.
Regarding our remaining variable with missing values, we assumed that the 4 N/A in the `children` variable could be considered as 0 since only customers with no children would be likely not to fill in this information.

3.3.2 Normalization of numerical features

To normalize the numerical features, we decided to scale each feature by its maximum absolute value. We used `MaxAbsScaler` because this estimator scales and translates each feature individually such that the maximal absolute value of each feature in the training set will be 1, that means that the scaled values are mapped in the range [0, 1]. The formula is $x_{scaled} = \frac{x}{max(abs(x)}$. This scaler works better for cases where the distribution isn't Gaussian. It does not shift/center the data, and thus does not destroy any sparsity.

3.3.3 Discretization of categorical features

We know that features that contain numbers can confuse a model into thinking that these features have data with some kind of order or hierarchy so we decided to discretize our categorical data by using `OneHotEncoder`. `OneHotEncoder` allowed us to encode categorical features as a one-hot numeric array as explained earlier in the question about `OneHotEncoder`.

3.3.4 Balancing the training set

Data imbalance usually reflects an unequal distribution of classes within a dataset. In our case only 6% of the customers need a parking spot (`homework_target` = 1) and almost 94% that don't need it (

`homework_target` = 0). Due to the disparity of classes in the target variables, the algorithm will tend to classify our observations into the class with more occurrences (the majority class here is obviously 0), while at the same time giving the false sense of a highly accurate model. Both the inability to predict rare events, the minority class, and the misleading accuracy will detract from the predictive models we build. Because we have a lot of data (around 120 000 observations), we decided to resample the dataset by using undersampling. Undersampling is the process where you randomly delete some of the observations from the majority class in order to match the numbers with the minority class. After this step, we have a ratio of 50:50 between our two classes.

3.4 Model selection

In this section we will compare 4 different models to choose the best one. We will compare a k-NN Classifier, a Logistic Regression, a Support Vector Classifier, and a Random Forest Classifier. For each model, an explanation and a mathematical definition is given in the appendix. Moreover, all the plots (ROC, Confusion Matrix,...) are also available in appendix.

3.4.1 k-NN

Let's first have a look at the results of the kNN Classifier. After running a GridSearchCV, we obtain 5 as the optimal value for the *n_neighbors* parameter. The confusion matrix gives us the distribution of our predictions. We can see that 19991 instances of the negative class and 1406 of the positive class were correctly classified (TN and TP respectively). However, 8021 instances of the negative class were mis-classified as "1" (FP, also called type 1 error), while 430 instances of the positive class were mis-classified as "0" (FN, also called type 2 error). Our model performs slightly better on the positive class (Recall = 0.77 vs Specificity = 0.71), but overall its performance is not bad with a Balanced Accuracy Score of 0.74. We also plot the ROC Curve, another metric to evaluate the performance of classification models, and obtain an Area Under Curve of 0.81, which confirms the not to bad performance of our model. This curve created by plotting the TP Rate ($TPR = \frac{Sensitivity}{Recall}$) vs. the FP Rate ($FPR = \frac{FP}{FP+TN}$) at different classification thresholds. The blue line represents the ROC Curve of our model. A good model stays as close to the upper left corner as possible, creating thus the largest possible Area Under Curve (values between 0 and 1).

3.4.2 Logistic Regression

Now let's have a look at the results of the second model: the Logistic Regression. Following a GridSearchCV, we obtain 0.25 for the optimal `C` value that is the regularization parameter that tells how much we want to avoid misclassifying each training example. The strength of the regularization is inversely proportional to C. If we look at the confusion matrix, we can see that we have less FP than before, but also more FN. This results in a better Specificity of 0.75 (how the model performs on the negative class) and a slightly weaker Recall of 0.75 (how the model performs on the positive class). This shows that our model performs equally good on both classes. Overall, the performance of our Logistic Regression is very slightly better than the k-NN, with a Balanced Accuracy Score of 0.75 (vs 0.74). The AUC Score of 0.82 (vs 0.81 for k-NN) confirms this very small improvement.

3.4.3 Support Vector Classifier

Our 3rd model is a Support Vector Classifier. LinearSVC is like the SVC class but it's a (faster) implementation for the case of a linear kernel. Like the Logistic Regression, after a cross-validation, we obtain 0.25 for the optimal `C` value. Let's look at the results of the SVC. They are actually almost exactly the same than the ones of the Logistic Regression (all evaluation metrics have the same values), which we could expect given the similar nature of the 2 models and the fact that the data is quite linearly separable.

3.4.4 Random Forest

Finally, our 4th and final model is a Random Forest Classifier. Following a GridSearchCV, we obtain 12 for the optimal `max_depth` value and 90 for the optimal `n_estimators`. If we look at the confusion matrix and

the classification metrics, we can see that the results are all better than for the previous models. Indeed, with a Recall of 0.8, the Random Forest is the model performing the best on the positive class. And for the negative class, it performs as good as previous models with a specificity of 0.75. Overall, we have a Balanced Accuracy Score of 0.77, which confirms that this model is the best one (vs 0.75 and 0.74). The AUC value of 0.85 (vs 0.82 and 0.81) confirms that fact.

4 Appendix

4.1 Homework features

4.1.1 Definition of each feature we removed

- `is_canceled`: Value indicating if the booking was canceled (1) or not (0)
- `lead_time`: Number of days that elapsed between the entering date of the booking into the PMS and the arrival date
- `arrival_date_year`: Year of arrival date
- `arrival_date_month`: Month of arrival date
- `arrival_date_week_number`: Week number of year for arrival date
- `arrival_date_day_of_month`: Day of arrival date
- `country`: Country of origin (categories are represented in the ISO 3155–3 : 2013 format)
- `agent`: ID of the travel agency that made the booking
- `company`: ID of the company/entity that made the booking or responsible for paying the booking (ID is presented instead of designation for anonymity reasons)
- `reserved_room_type`: Code of room type reserved (code is presented instead of designation for anonymity reasons)
- `assigned_room_type`: Code for the type of room assigned to the booking; sometimes the assigned room type differs from the reserved room type due to hotel operation reasons (e.g. overbooking) or by customer request (code is presented instead of designation for anonymity reasons)
- `required_car_parking_spaces`: Number of car parking spaces required by the customer
- `reservation_status`: Reservation last status, assuming one of three categories:
  - Canceled = booking was canceled by the customer
  - Check-Out = customer has checked in but already departed
  - No-Show = customer did not check-in and did inform the hotel of the reason why
- `reservation_status_date`: Date at which the last status was set

4.1.2 Definition of each feature used in the model

The numerical variables are :

- `stays_in_weekend_nights`: Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
- `stays_in_week_nights`: Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- `adults`: Number of adults
- `children`: Number of children
- `babies`: Number of babies
- `previous_cancellations`: Number of previous bookings that were cancelled by the customer prior to the current booking

- `previous_bookings_not_canceled`: Number of previous bookings not cancelled by the customer prior to the current booking
- `booking_changes`: Number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation
- `days_in_waiting_list`: Number of days the booking was in the waiting list before it was confirmed to the customer
- `adr`: Average Daily Rate as defined by dividing the sum of all lodging transactions by the total number of staying nights
- `total_of_special_requests`: Number of special requests made by the customer (e.g. twin bed or high floor)

The categorical variables are:

- `hotel`: Hotel (H1 = Resort Hotel or H2 = City Hotel)
- `meal`: Type of meal booked. Categories are presented in standard hospitality meal packages:
  - Undefined/SC = no meal package
  - BB = Bed & Breakfast
  - HB = Half board (breakfast and one other meal – usually dinner)
  - FB = Full board (breakfast, lunch and dinner)
- `market_segment`: Market segment designation.
  - The term "TA" means "Travel Agents"
  - The term "TO" means "Tour Operators"
- `distribution_channel`: Booking distribution channel.
  - The term "TA" means "Travel Agents"
  - The term "TO" means "Tour Operators"
- `is_repeated_guest`: Value indicating if the booking name was from a repeated guest (1) or not (0)
- `deposit_type`: Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories:
  - No Deposit = no deposit was made
  - Non Refund = a deposit was made in the value of the total stay cost
  - Refundable = a deposit was made with a value under the total cost of stay
- `customer_type`: Type of booking, assuming one of four categories:
  - Contract = when the booking has an allotment or other type of contract associated to it
  - Group = when the booking is associated to a group
  - Transient = when the booking is not part of a group or contract, and is not associated to other transient booking
  - Transient-party = when the booking is transient, but is associated to at least other transient booking
- $\texttt{reserved\_assigned\_room\_type} = \begin{cases} 1 & \text{if} \quad \texttt{reserved\_room\_type} = \texttt{assigned\_room\_type} \\ 0 & \text{otherwise} \end{cases}$
- $\texttt{homework\_target} = \begin{cases} 1 & \text{if} \quad \texttt{required\_car\_parking\_spaces} \neq 0 \\ 0 & \text{if} \quad \texttt{required\_car\_parking\_spaces} = 0 \end{cases}$

4.2 Tables & figures : data visualization, processing and cleaning

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
hotel	119390	2	City Hotel	79330	NaN	NaN	NaN	NaN	NaN	NaN	NaN
is_canceled	119390	NaN	NaN	NaN	0.370416	0.482918	0	0	0	1	1
lead_time	119390	NaN	NaN	NaN	104.011	106.863	0	18	69	160	737
arrival_date_year	119390	NaN	NaN	NaN	2016.16	0.707476	2015	2016	2016	2017	2017
arrival_date_month	119390	12	August	13877	NaN	NaN	NaN	NaN	NaN	NaN	NaN
arrival_date_week_number	119390	NaN	NaN	NaN	27.1652	13.6051	1	16	28	38	53
arrival_date_day_of_month	119390	NaN	NaN	NaN	15.7982	8.78083	1	8	16	23	31
stays_in_weekend_nights	119390	NaN	NaN	NaN	0.927599	0.998613	0	0	1	2	19
stays_in_week_nights	119390	NaN	NaN	NaN	2.5003	1.90829	0	1	2	3	50
adults	119390	NaN	NaN	NaN	1.8564	0.579261	0	2	2	2	55
children	119386	NaN	NaN	NaN	0.10389	0.398561	0	0	0	0	10
babies	119390	NaN	NaN	NaN	0.00794874	0.0974362	0	0	0	0	10
meal	119390	5	BB	92310	NaN	NaN	NaN	NaN	NaN	NaN	NaN
country	118902	177	PRT	48590	NaN	NaN	NaN	NaN	NaN	NaN	NaN
market_segment	119390	8	Online TA	56477	NaN	NaN	NaN	NaN	NaN	NaN	NaN
distribution_channel	119390	5	TA/TO	97870	NaN	NaN	NaN	NaN	NaN	NaN	NaN
is_repeated_guest	119390	NaN	NaN	NaN	0.0319122	0.175767	0	0	0	0	1
previous_cancellations	119390	NaN	NaN	NaN	0.0871178	0.844336	0	0	0	0	26
previous_bookings_not_canceled	119390	NaN	NaN	NaN	0.137097	1.49744	0	0	0	0	72
reserved_room_type	119390	10	A	85994	NaN	NaN	NaN	NaN	NaN	NaN	NaN
assigned_room_type	119390	12	A	74053	NaN	NaN	NaN	NaN	NaN	NaN	NaN
booking_changes	119390	NaN	NaN	NaN	0.221124	0.652306	0	0	0	0	21
deposit_type	119390	3	No Deposit	104641	NaN	NaN	NaN	NaN	NaN	NaN	NaN
agent	103050	NaN	NaN	NaN	86.6934	110.775	1	9	14	229	535
company	6797	NaN	NaN	NaN	189.267	131.655	6	62	179	270	543
days_in_waiting_list	119390	NaN	NaN	NaN	2.32115	17.5947	0	0	0	0	391
customer_type	119390	4	Transient	89613	NaN	NaN	NaN	NaN	NaN	NaN	NaN
adr	119390	NaN	NaN	NaN	101.831	50.5358	-6.38	69.29	94.575	126	5400
required_car_parking_spaces	119390	NaN	NaN	NaN	0.0625178	0.245291	0	0	0	0	8
total_of_special_requests	119390	NaN	NaN	NaN	0.571363	0.792798	0	0	0	1	5
reservation_status	119390	3	Check-Out	75166	NaN	NaN	NaN	NaN	NaN	NaN	NaN
reservation_status_date	119390	926	2015-10-21	1461	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 1: Basic statistics

Figure 2: Target variable distribution

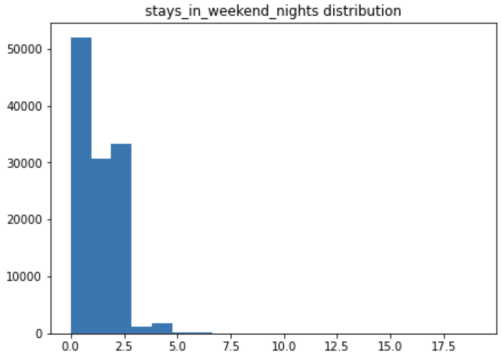


Figure 3: `stays_in_weekend_nights` distribution

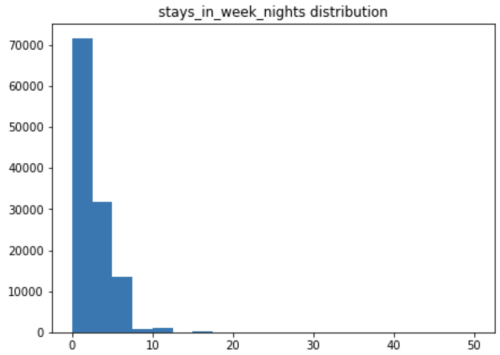


Figure 4: `stays_in_week_nights` distribution

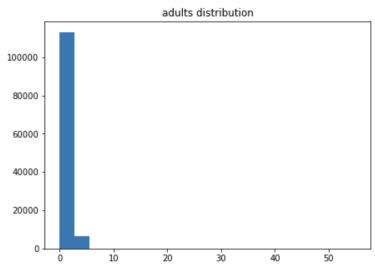


Figure 5: `adults` distribution

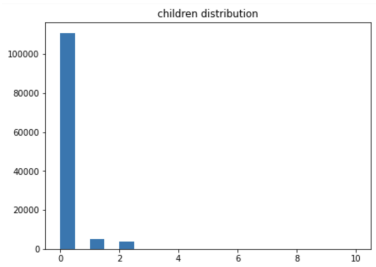


Figure 6: `children` distribution

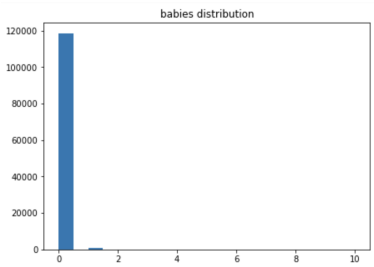


Figure 7: `babies` distribution

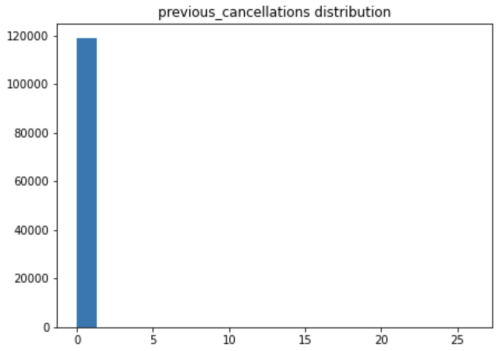


Figure 8: `previous_cancellations` distribution

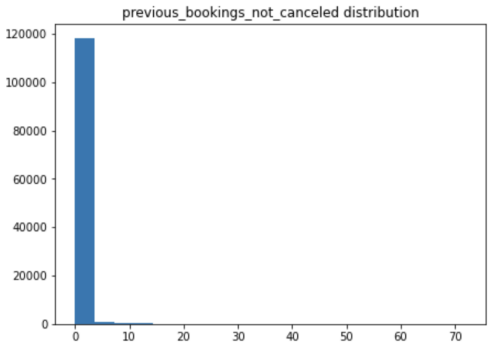


Figure 9: `previous_bookings_not_canceled` distribution

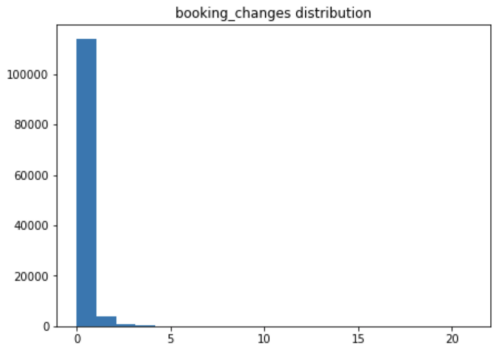


Figure 10: `booking_changes` distribution

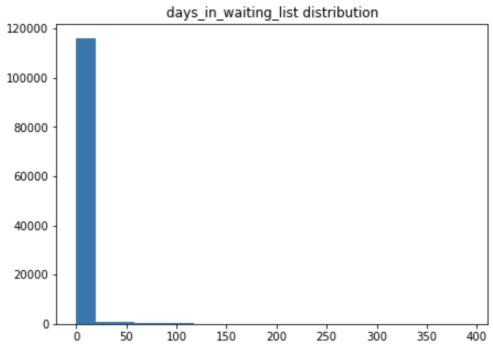


Figure 11: `days_in_waiting_list` distribution

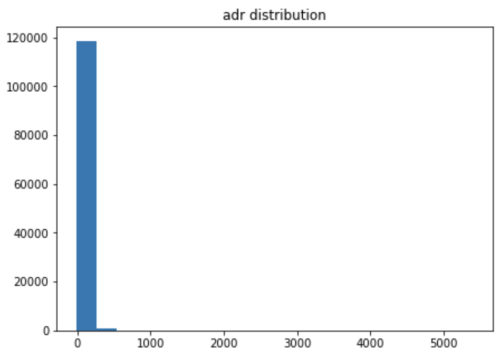


Figure 12: `adr` distribution

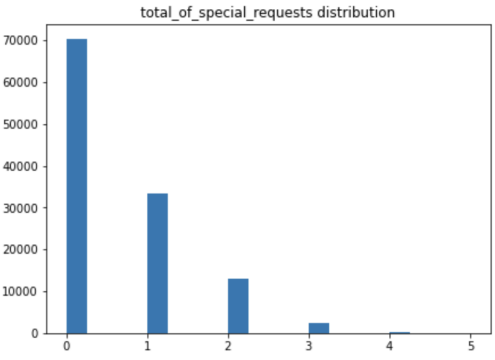


Figure 13: `total_of_special_requests` distribution

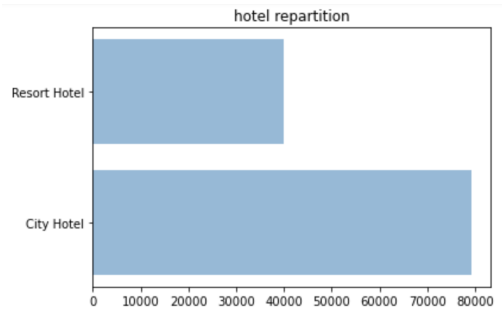


Figure 14: `hotel` distribution

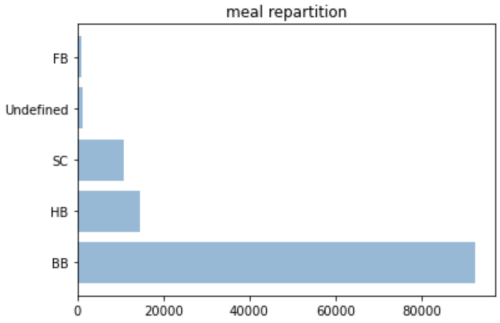


Figure 15: `meal` distribution

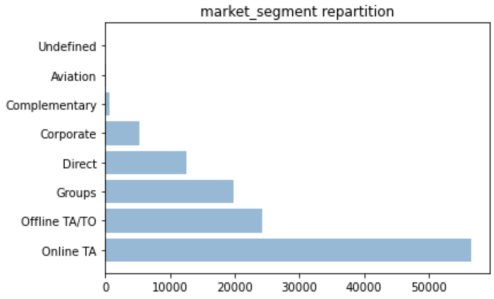


Figure 16: `market_segment` distribution

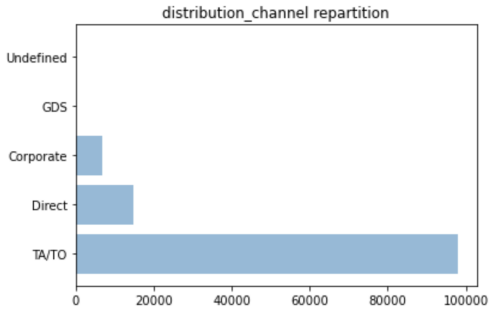


Figure 17: `distribution_channel` distribution

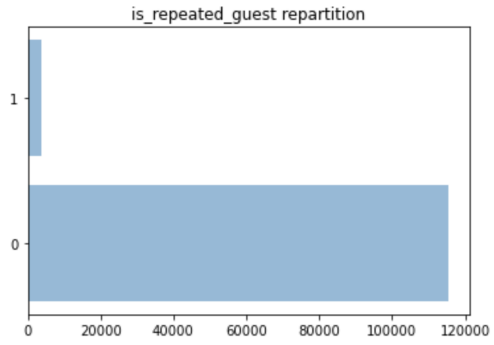


Figure 18: `is_repeated_guest` distribution

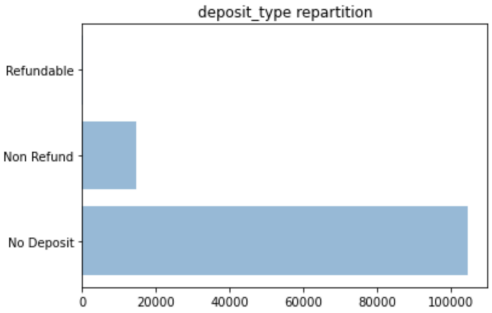


Figure 19: `deposit_type` distribution

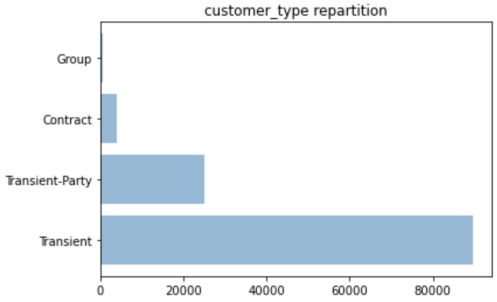


Figure 20: `customer_type` distribution

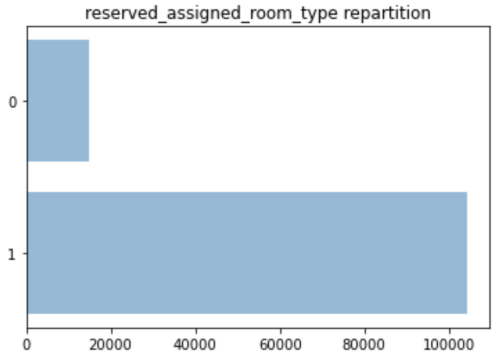


Figure 21: `reserved_assigned_room_type` distribution

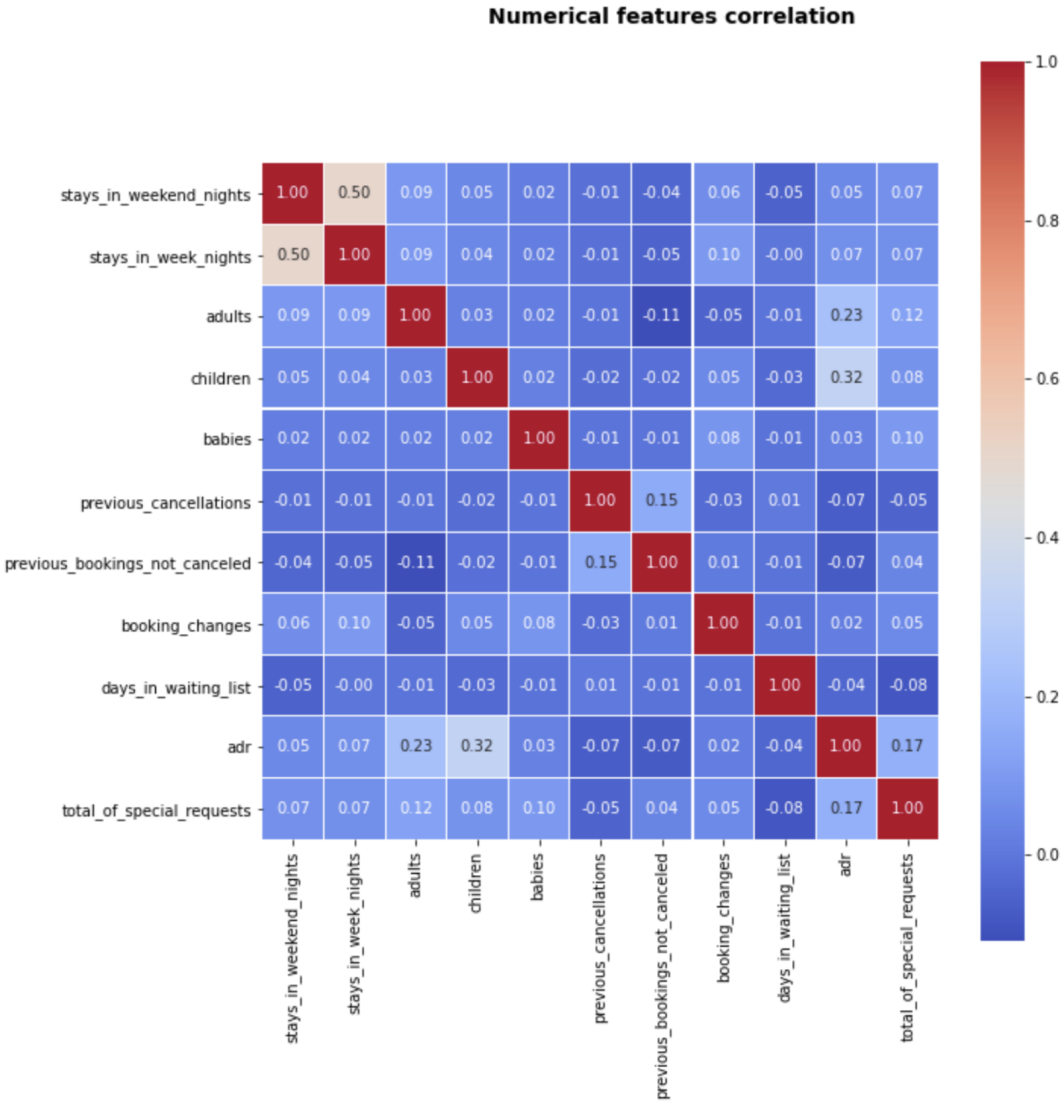


Figure 22: Numerical features correlation

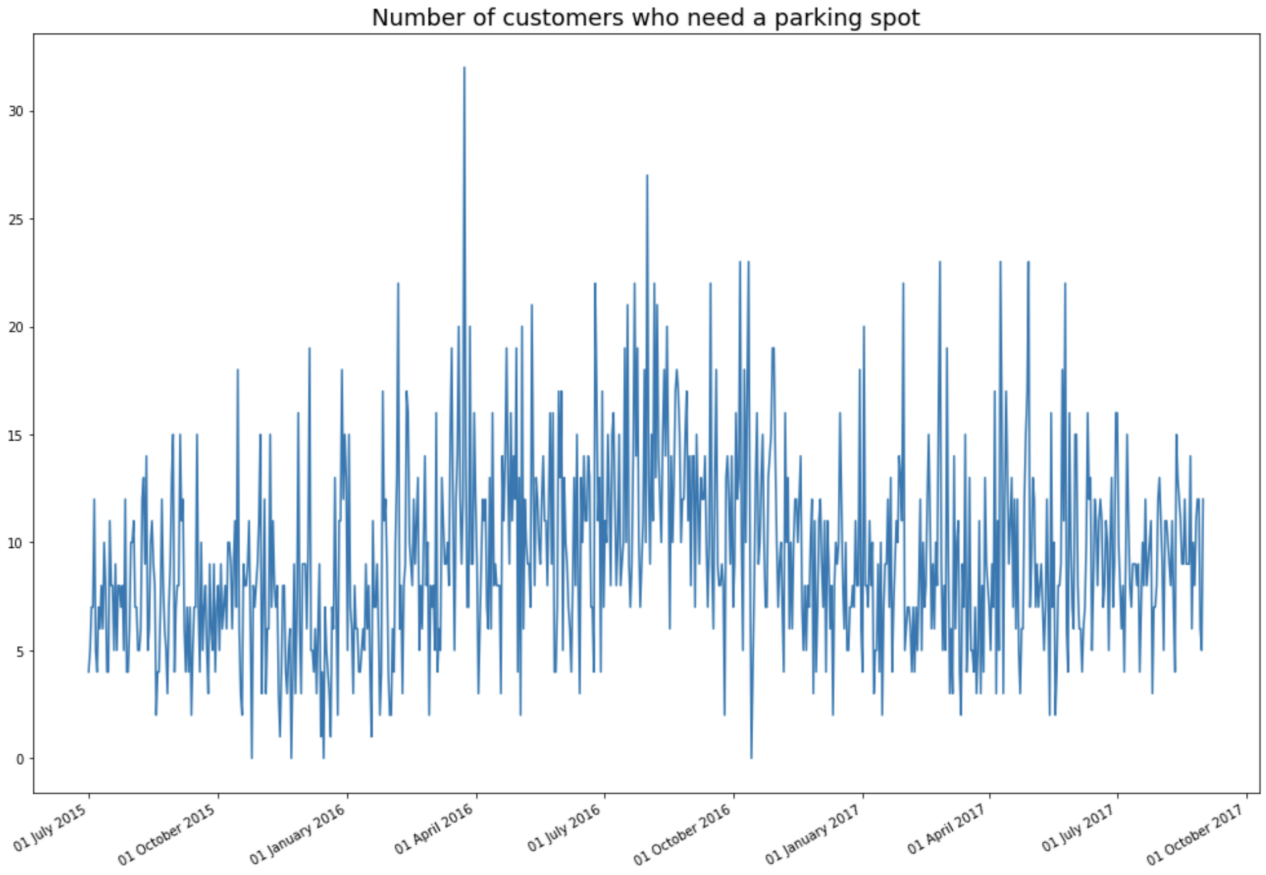


Figure 23: Analysis of the target variable through time

4.3 Models : explanation and mathematical definition

4.3.1 k nearest neighbors

The k nearest neighbors is a plug-in method so in other terms it consists in estimating $\eta = \mathbb{E}(Y|X)$ where $(X, Y) \sim \mathbb{P}$ unknown. Let's denote $d(x_i, x_j)$ a distance between 2 points.

- **Regression:** $\hat{g}(x) = \hat{\eta}(x) = \frac{1}{k} \sum_{i:x_i \in V_k(x)} Y_i \quad$ where $\quad V_k = \{X_{r_1(x)}, ..., X_{r_k(x)}\}$ with r_j the index of the j^{th} closest neighbor.
- **(Binary) Classification:** $\hat{g}(x) = \mathbb{I}_{\hat{\eta}(x) \geq \frac{1}{2}}$

4.3.2 Logistic regression

A logistic model is a model such that given $X = x$, Y is a Bernoulli random variable with parameter:

$$\pi(x) = \mathbb{P}[Y = 1|X = x] = \mathbb{E}(Y|X = x)$$

To understand this model one must first recall the definition of the odds in favor of an event A:

$$Odds(A) = \frac{\mathbb{P}(A)}{1 - \mathbb{P}(A)} \in [0, +\infty[$$

Moreover let's recall the distribution functions of the following distributions:

- $F(u) = logistic(u) = \frac{exp(u)}{1+exp(u)}$

- $F^{-1}(p) = logit(p) = ln\left(\frac{p}{1-p}\right)$

Let's denote $s(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$ the score of an individual. A logistic model satisfies the following equations (which are all implicated by one another):

1. $\pi(x) = F(s(x))$
2. $s(x) = F^{-1}(\pi(x))$
3. $Odds(x) = exp(s(x))$

The logistic regression is a Generalized Linear Model. GLM is an extension of the linear model to accomodate for non gaussian response distribution and transformations to linearity.

Recall 1: An alternative way of writing an OLS model if the 4 following assumptions are verified:

- The model is linear in parameters : $Y = X\beta + U$
- Homoskedasticity : $\mathbb{V}(U|X) = \sigma^2$
- Exogeneity : $\mathbb{E}(U|X) = 0$
- Gaussian distribution of the error term

is to write that $Y \sim \mathcal{N}_n(X\beta, \sigma^2 I_n)$ where $\mathcal{N}_n$ denotes the n-dimensional gaussian distribution and I_n the n-dimensional identity matrix.

Recall 2: If a random variable y belongs to a location-scale exponential family then its density can be written as:

$$f(y|\theta, \phi) = exp\left\{w\frac{y\theta - b(\theta)}{\phi} + c(y, \phi)\right\}$$

where θ is called the canonical parameter (location) and ϕ is the dispersion parameter (scale) (w plays the role of a weight).

A GLM verifies 2 assumptions:

- Distributional assumption: the conditional distribution of the response variable Y given X should be a member of the location-scale exponential family distribution and the different couples responses/covariate are independent with the same ϕ.
- Structural assumption: the link function g (invertible) describes how the mean of the response $\mu = \mathbb{E}(Y|X)$ is related to a linear combination of the predictors $\eta = X\beta = g(\mu) \Leftrightarrow \mu = \mathbb{E}(Y|X) = g^{-1}(X\beta)$.

Now, one can notice that a logistic model is, indeed, a GLM where :

- $g(.)$ is the logit function
- $\phi = 1$, $w = 1$ and $\theta = ln\left(\frac{\pi(x)}{1-\pi(x)}\right)$
- $b(\theta) = -ln(1 - \pi(x))$

4.3.3 Support Vector Machine

The Support Vector Machines (SVM for short below) estimators are classifiers dedicated to the problem of supervised classification to infer the value of a binary variable. The initial SVM algorithm is based on a very natural geometrical idea of separation with hyperplanes : we are interested in finding the optimal hyperplane with optimal margin that separates the two classes of interest when it is possible. Hence, the principle is to be able to split the data set into two parts with an hyperplane that is as far as possible from the observation. The principle of SVM can be roughly speaking described by two main steps :

- First, we will define an "optimal" hyperplane as the solution of a constrained optimization problem.
- Second, we will extend the SVM algorithm to separation with non-linear functions. Such non-linearities will be obtained with the help of kernels. These kernels implicitely modify the original dataset and send the observations in a larger dimensional space.

The linear separation problem: A problem is said to be linearly separable when a linear function f and a decision D exists such that $D(x) = sgn\left(f(x)\right)$ with

$$sgn(u) = \begin{cases} 1 & \text{if} \quad u > 0 \\ -1 & \text{if} \quad u \leq 0 \end{cases} \quad \text{and} \quad f(x) = \langle \beta, X \rangle + \beta_0 = \beta^T X_i + \beta_0$$

So a problem is linearly separable if there exists $(\beta, \beta_0) \in \mathbb{R}^d \times \mathbb{R}$ such that:

$$\begin{cases} \langle \beta, X \rangle + \beta_0 = \beta^T X_i + \beta_0 > 0 & \text{if} \quad Y_i = 1 \\ \langle \beta, X \rangle + \beta_0 = \beta^T X_i + \beta_0 \leq 0 & \text{if} \quad Y_i = -1 \end{cases}$$

Let's now introduce the geometrical margin M such that:

$$M = \min_{i=[1,n]} dist(X_i, \mathcal{L})$$

Therefore the optimal hyperplane is defined as follow: $\mathcal{L} = \{X \in \mathbb{R}^d | \, \langle \beta, X \rangle + \beta_0 = \beta^T X_i + \beta_0 = 0\}$ with

$$(\beta, \beta_0) = \begin{cases} \max_{(\beta, \beta_0): \|\beta\| = 1} M \\ s.c \quad Y_i \left(\beta^T X_i + \beta_0\right) \geq M \end{cases}$$

We will assume that the vector β satisfies $\|\beta\| = 1$ to guarantee uniqueness of the decision rule. But one can get ride of this constraint by rewriting the optimization problem above as follow:

$$(\beta, \beta_0) = \begin{cases} \max_{\beta, \beta_0} M \\ s.c \quad Y_i \left(\beta^T X_i + \beta_0\right) \geq M \|\beta\| \end{cases}$$

We can modify the problem and pose $M = \frac{1}{\|\beta\|}$ which leads us to the following optimization problem:

$$(\beta, \beta_0) = \begin{cases} \max_{\beta, \beta_0} \frac{1}{\|\beta\|} \\ s.c \quad Y_i \left(\beta^T X_i + \beta_0\right) \geq 1 \end{cases}$$

The classical formulation of the SVM is obtained through the minimization of $\|\beta\|$ (instead of maximizing the inverse of the norm of w). We are led to the definition :

$$(\beta, \beta_0) = \begin{cases} \min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 \\ s.c \quad Y_i \left(\beta^T X_i + \beta_0\right) - 1 \geq 0 \end{cases}$$

The Lagrangian of the SVM optimization problem may be written as :

$$\mathcal{L}(\beta, \beta_0, \alpha) = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^{n} \alpha_i \left[Y_i \left(\beta^T X_i + \beta_0\right) - 1\right]$$

In the linearly separable case this problem has a unique solution which is $D(x) = sgn\,(f(x))$ with

$$f(x) = \langle \beta^*, X \rangle + \beta_0^* = \sum_{i \in \mathcal{A}} \langle \alpha_i^* y_i X_i, X \rangle + \beta_0^* = \sum_{i \in \mathcal{A}} \alpha_i^* y_i X_i^T X + \beta_0^*$$

and $\mathcal{A} := \{i \in [1, n] | Y_i \left(\beta^{T*} X_i + \beta_0^*\right) = 1\}$. This results is known a the hard SVM solution.

The non separable case: In this case we have the following constraint

$$\begin{cases} Y_i \left(\beta^T X_i + \beta_0\right) \geq 1 - \xi_i \\ \xi_i > 0 \quad \text{and} \quad \sum_{i=1}^n \xi_i \leq Z \quad \textit{constant} \end{cases}$$

which leads us to the soft SVM solution

$$\min_{\beta, \beta_0, \xi} \left\{ \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.c} \quad \begin{cases} \xi_i \geq 0 \\ \xi_i \geq 1 - Y_i \left(\beta^T X_i + \beta_0\right) \end{cases} \right\}$$

4.3.4 Random forest

Let's first describe the decision tree algorithm. It is based on a recursive binary partitioning of the data space spanned by the predictor variables into increasingly homogeneous groups with respect to the response variable as measured by the Gini index. In practice, computing a "good" partition $R_1, R_2...$ is infeasible (it is a NPcomplete problem). Hence, we consider the following greedy algorithm.

Let us describe the first step of this algorithm. For any $j \in \{1, 2, ..., p\}$ (here $p = 41$ after OneHotEncoded the data) and $s \in \mathbb{R}$, we define the pair of half-planes given by splitting the variable x_j at s.

$$R_1(j, s) = x = (x_1, ..., x_p)' \in \mathbb{R}^p \ s.t. \ x_j \leq s$$
$$R_2(j, s) = x = (x_1, ..., x_p)' \in \mathbb{R}^p \ s.t. \ x_j > s$$

These two sets give us a trivial partition of R_p and we can use them to compute the frequencies $\widehat{p}_{1l}(j, s)$ and $\widehat{p}_{2l}(j, s)$ (with $l \in 0, 1$) of each label, respectively in $R_1(j, s)$ and $R_2(j, s)$. Moreover, we can attribute to $R_1(j, s)$ and $R_2(j, s)$ their most represented element.

In order to choose a good index $j \in \{1, ..., p\}$ and split point $s \in R$, we take the values that make the Gini index minimal (it is a measure of the nodes impurity).

$$G = \sum_{l=1}^m \widehat{p}_{kl}(1 - \widehat{p}_{kl}), k \in (1, ..., M)$$

Precisely we have at our disposal $G_1(j, s)$ and $G_2(j, s)$ and the region weights $\overline{\omega}_1(j, s)$ and $\overline{\omega}_2(j, s)$ with $\overline{\omega}_k = \sum_{i \in R_k} \omega_i$ (which corresponds to the sum of the weights attributed to each observation in R_k, this is often equal to the number of observations in R_k since observations are usually not weighted). Thus, we take $j^* \in (1, ..., p)$ and $s^* \in \mathbb{R}$ which reaches the minimal value:

$$\min_{j,s} (\overline{\omega}_1 G_1(j, s) + \overline{\omega}_2 G_2(j, s))$$

The sequel of the algorithm then consists in repeating the same procedure in each region. In such a way, we divide the considered region by two at each step.

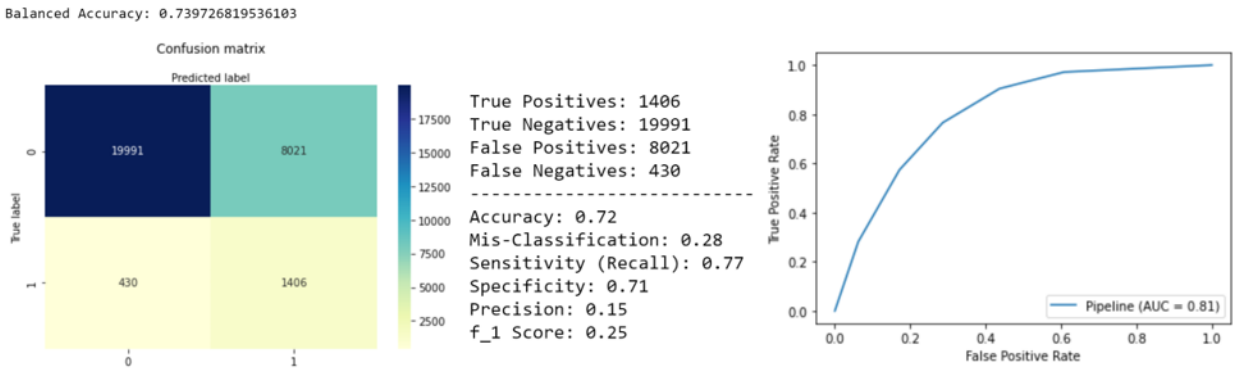
Let's now take a look at random forests models. We have our data set $D_n = (X_1, y_1), ..., (X_n, y_n)$ and the goal is to estimate the regression function $r(x) = \mathbb{E}[Y|X = x]$. A random forest is a classifier based on a collection of randomized base regression trees $r_n(x, \Phi, D_n), m \geq 1$ where $\Phi_1, \Phi_2, ...$ are i.i.d outputs of a randomizing variable Φ. All the random trees are combined to form an aggregated regression estimate:

$$\tilde{r_n}(X, D_n) = \mathbb{E}_\Phi[r_n(X, \Phi, D_n)]$$

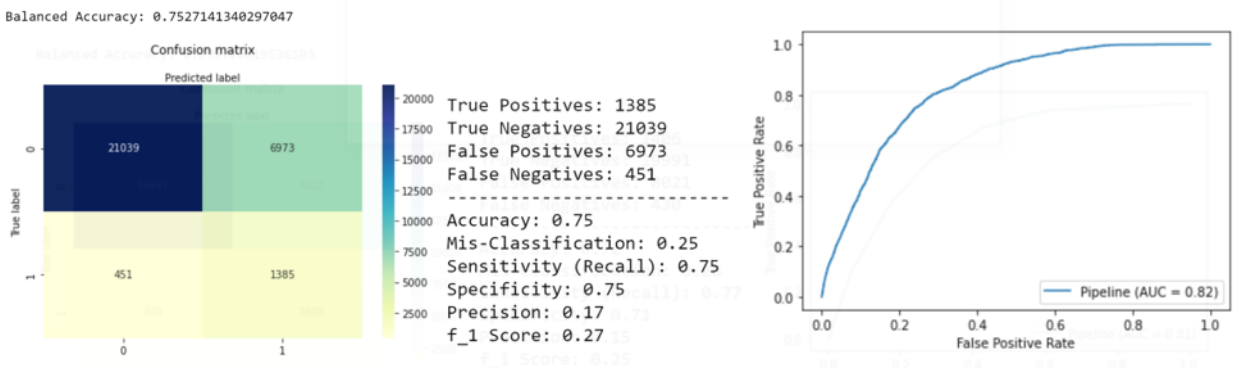
where $\mathbb{E}_\Phi$ is the expectation with respect to the random parameter, conditionally on X and the data set D_n. This expectation is evaluated using Monte Carlo procedure: M random trees are generated and one take the average of the individual outcomes. The randomizing variable Φ is used to select at random and at each node a small group of variables. Then it computes the best split based on these features and cut (as explained previously). The tree grows without pruning. Therefore we aggregate over all the trees.

4.4 Homework results

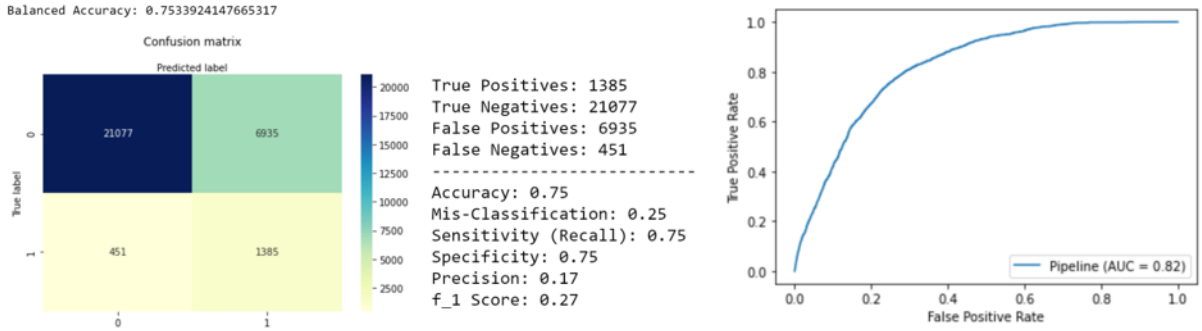
4.4.1 k-NN



4.4.2 Logistic Regression



4.4.3 SVC



4.4.4 Random Forest