UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Master 2 Traitement de l'information et data-science en entreprise (TIDE)

Analyse en grande dimension

Projet groupe INN Hotels : prédiction des annulations

GROUPE :

FIX Emma (Analyse / *Boosting*)

HURTADO DIAZ Ezequiel (Analyse / Régression Logistique)

MADI Tamim (Analyse / *Random Forest*)

Année universitaire 2024/2025

Introduction

Un grand nombre de réservations d'hôtels sont annulées ou non honorées en raison de changements de programme ou de conflits d'agenda. Cette situation est facilitée par les options d'annulation gratuite ou à faible coût avantageuses pour les clients mais problématiques pour les hôtels, notamment en cas d'annulations de dernière minute.

De plus, les plateformes de réservation en ligne ont modifié le comportement des clients rendant la gestion des annulations plus complexe pour les hôtels qui subissent des pertes de revenus, des coûts supplémentaires et une baisse de leur marge bénéficiaire.

Face à l'augmentation des annulations, une solution basée sur le *Machine Learning* est nécessaire pour prédire quelles réservations risquent d'être annulées.

L'objectif de cette étude sera donc d' aider le groupe INN Hotels, qui possède une chaîne d'hôtels au Portugal et qui est confronté à un taux élevé d'annulations, à trouver une solution basée sur les données des réservations dans leurs hôtels.

Pour réaliser cela, nous analyserons les données afin d'identifier les facteurs influençant les annulations et construirons un modèle prédictif capable d'anticiper une majorité des annulations. A partir de nos résultats, nous pourrons proposer au groupe INN Hotels des stratégies et des politiques rentables pour gérer les annulations et les remboursements.

I. Compréhension et exploration des données

1. Description de la base d'étude

Les données fournies par le groupe INN Hotels contiennent les informations détaillées des réservations clients avec 19 variables et 36 275 observations.
Nous avons 5 variables qualitatives et 14 variables quantitatives dont 3 temporelles.

Description des variables :

- *Booking_ID* : Identifiant unique de chaque réservation
- *no_of_adults / no_of_children* : Nombre d'adultes et d'enfants
- *no_of_weekend_nights / no_of_week_nights* : Nombre de nuits réservées le week-end ou en semaine
- *type_of_meal_plan* : Type de repas réservé (aucun, petit-déjeuner, demi-pension, pension complète)
- *required_car_parking_space* : Besoin d'une place de parking (0 : Non, 1 : Oui)
- *room_type_reserved* : Type de chambre réservée (7 modalités)
- *lead_time* : Nombre de jours entre la réservation et l'arrivée
- *arrival_year / arrival_month / arrival_date* : Date d'arrivée (année, mois, jour)
- *market_segment_type* : Segment de marché de la réservation (5 modalités)
- *repeated_guest* : Client régulier (0 : Non, 1 : Oui)
- *no_of_previous_cancellations / no_of_previous_bookings_not_canceled* : Nombre de réservations précédentes annulées ou non
- *avg_price_per_room* : Prix moyen par nuit de la réservation (en euros)
- *no_of_special_requests* : Nombre total de demandes spéciales
- *booking_status* : Indique si la réservation a été annulée ou non

2. Analyses univariées

Lors de l'import de nos données, nous avons pris la décision de supprimer deux variables.
La première est *Booking_ID* car cette dernière n'est en rien pertinente pour le modèle, et de plus c'est une variable qualitative qui pourrait nous gêner dans nos analyses.
La seconde est *arrival_year* car nous avons considéré qu'elle pourrait biaiser le modèle. En effet, lors d'analyses réalisées postérieurement, nous avons vu que cette dernière semblait être très pertinente pour le modèle (Test d'indépendance), pourtant il ne s'agissait pas réellement de cela, la répartition des valeurs entre 2017 et 2018 était juste très déséquilibrée dans la table car nous avions très peu de données pour 2017 et ainsi influencer faussement le modèle.

Voici un sommaire du dataframe que nous obtenons :

skimpy summary

Data Summary

Dataframe	Values
Number of rows	36275
Number of columns	18

Data Types

Column Type	Count
int64	13
string	4
float64	1

number

column	NA	NA %	mean	sd	p0	p25	p50	p75	p100	hist
no_of_adults	0	0	1.845	0.5187	0	2	2	2	4	
no_of_children	0	0	0.1053	0.4026	0	0	0	0	10	
no_of_weekend_nights	0	0	0.8107	0.8706	0	0	1	2	7	
no_of_week_nights	0	0	2.204	1.411	0	1	2	3	17	
required_car_parking_space	0	0	0.03099	0.1733	0	0	0	0	1	
lead_time	0	0	85.23	85.93	0	17	57	126	443	
arrival_year	0	0	2018	0.3838	2017	2018	2018	2018	2018	
arrival_month	0	0	7.424	3.07	1	5	8	10	12	
arrival_date	0	0	15.6	8.74	1	8	16	23	31	
repeated_guest	0	0	0.02564	0.1581	0	0	0	0	1	
no_of_previous_cancellations	0	0	0.02335	0.3683	0	0	0	0	13	
no_of_previous_bookings_not_canceled	0	0	0.1534	1.754	0	0	0	0	58	
avg_price_per_room	0	0	103.4	35.09	0	80.3	99.45	120	540	
no_of_special_requests	0	0	0.6197	0.7862	0	0	0	1	5	

string

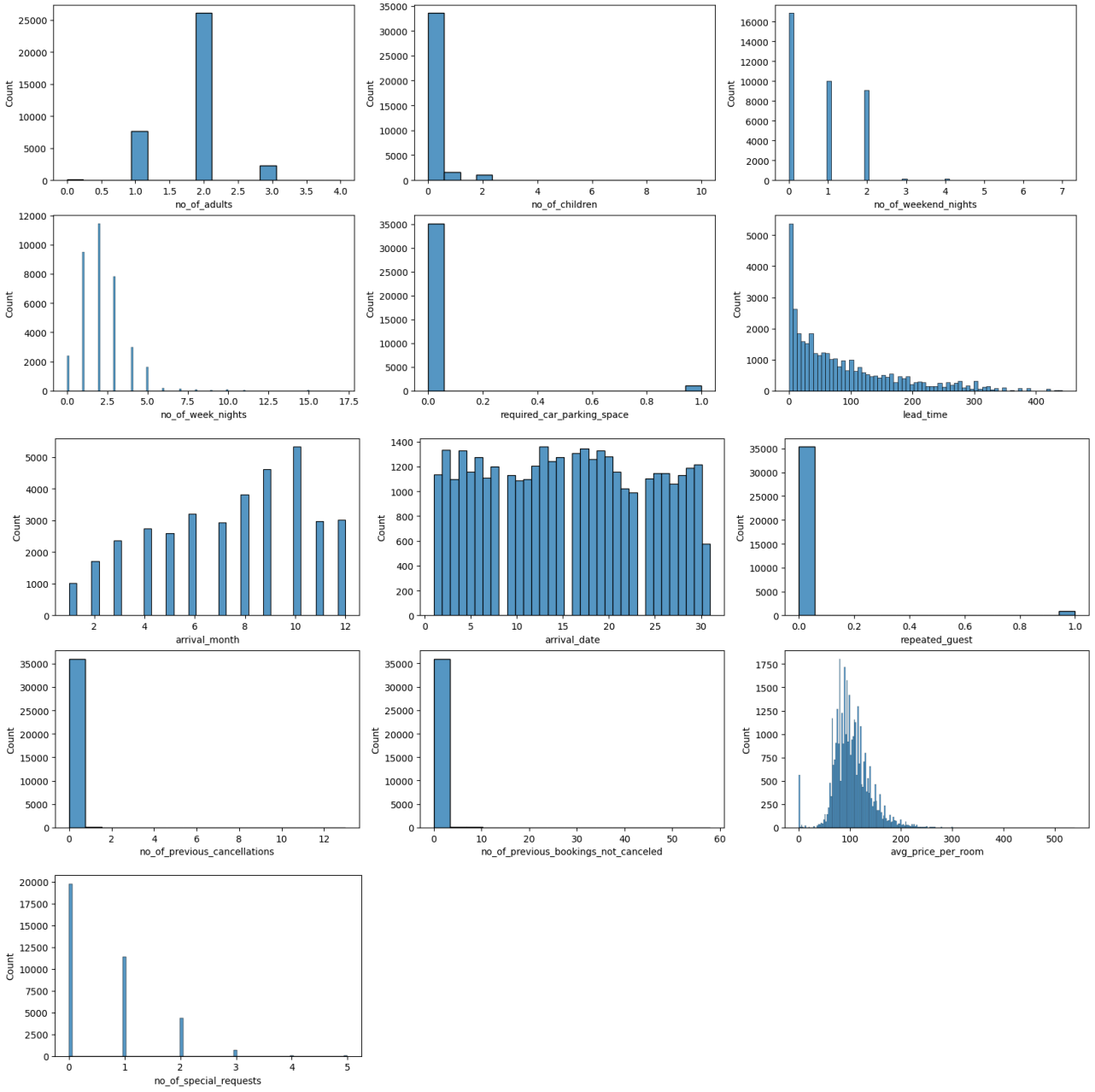
column	NA	NA %	shortest	longest	min	max	chars per row	words per row	total words
type_of_meal_plan	0	0	Meal Plan 1	Not Selected	Meal Plan 1	Not Selected	11.1	2.9	103695
room_type_reserved	0	0	Room_Type 1	Room_Type 1	Room_Type 1	Room_Type 7	11	2	72550
market_segment_type	0	0	Online	Complementary	Aviation	Online	6.54	1	36275
booking_status	0	0	Canceled	Not_Canceled	Canceled	Not_Canceled	10.7	1	36275

Regardons en détail certaines variables intéressantes.

Nous remarquons que la variable *lead_time* varie de 0 à 443 jours, ce qui peut être un facteur clé des annulations, elle reste cependant concentrée en dessous de trois mois.
Le prix moyen par chambre, *avg_price_per_room*, atteint 540 euros, avec une moyenne de 103 euros. Nous trouvons dans les données des valeurs à 0 ce qui peut paraître être une

valeur aberrante, mais qui peut être dû à des chambres offertes pour une raison quelconque, des offres sur des sites…
Le maximum de *no_of_children* est à 10, cette valeur semble être une exception selon le contexte de réservation de la chambre, plutôt qu'une valeur aberrante. La médiane quant-à elle est à 0, donc la plupart des réservations sont faites sans enfants.
Le minimum de *no_of_adults* est à 0, ce qui peut être dû à des chambres mitoyennes, une pour les parents et une pour les enfants.
Enfin, en ce qui concerne les variables *repeated_guest* et *no_of_previous_cancellations*, peu de clients ont un historique de réservation, il semble donc difficile d'utiliser les réservations précédentes pour prévoir les annulations.

Nous traçons les graphiques en barre des variables quantitatives afin de voir la distribution des variables et leur boxplot pour mieux visualiser les éventuelles valeurs aberrantes.



Pour les variables binaires (0,1) comme *required_car_parking_space*, c'est normal d'avoir des *outliers* à 1, il ne s'agit pas de valeurs aberrantes.
Nous voyons avec les distributions que les dates d'arrivée sont réparties sur toute l'année, avec une tendance à la hausse entre août et octobre.
La variable *avg_price_per_room* semble suivre une distribution normale (mis à par les valeurs à 0).
La variable *no_of_previous_cancellations* se concentre à 0, malgré cela il y a tout de même des valeurs montant jusqu'à 13 ce qui sera sans doute un bon indicateur d'une possible annulation.

3. Analyse multivariée

Afin de voir la corrélation entre les variables quantitatives nous traçons la matrice de corrélation.

Les variables des données fournies par le groupe INN Hotels sont peu corrélées, les coefficient le plus élevé étant à 0.54, 0.39 et 0.47 entre *repeated_guest*, *no_of_previous_booking_not_cancelled* et *no_of_previous_cancellations* ce qui semble tout à fait logique. En effet, qu'une réservation soit faite par une personne qui en avait déjà fait précédemment augmente positivement la probabilité que cette personne ait déjà annulé ou non une précédente réservation.

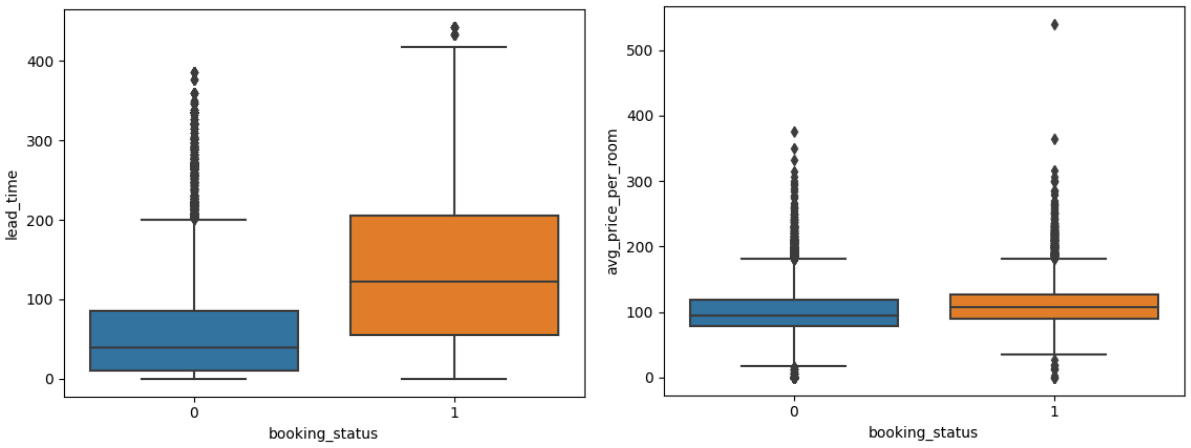
II. Etude du lien entre la variable cible et les données

A partir de là, nous définissons la variable cible comme étant *booking_status*.
Nous transformons la variable cible en variable binaire, avec pour valeur 0 si on avait pour valeur "*not_canceled*" dans les données d'origine et 1 pour "*canceled*".
Nous pouvons ainsi étudier le lien entre cette dernière et les autres variables afin de déterminer les facteurs influençant les annulations.

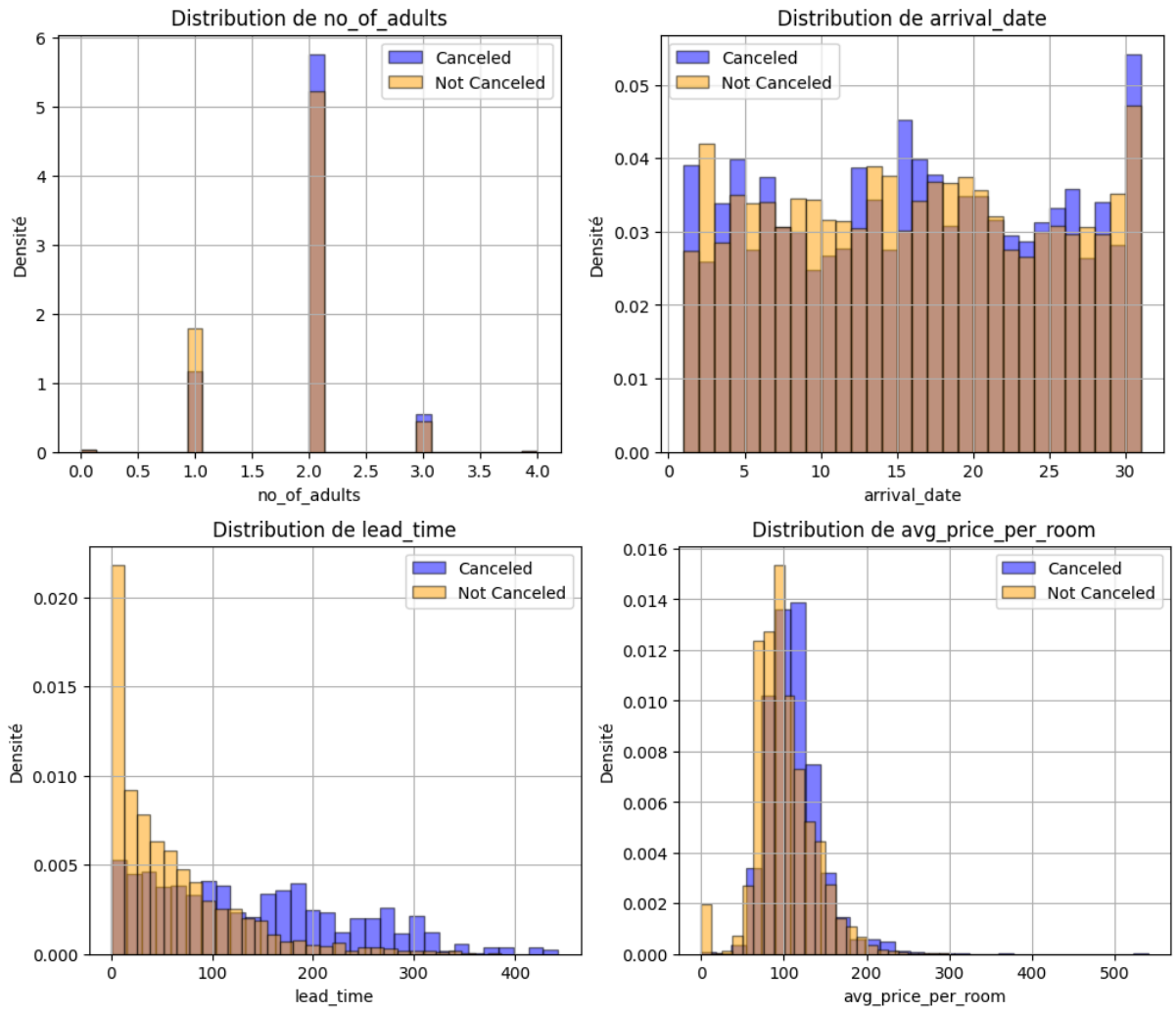
1. Analyse visuelle

En affichant la distribution de la variable cible nous pouvons voir qu'environ 32.8% des réservations sont annulées, ce qui est un taux significatif. Cela justifie l'intérêt d'un modèle prédictif pour anticiper ces annulations.

Ensuite nous traçons la répartition de certaines variables jugées pertinentes par rapport à la valeur de la variable cible. Nous avons observé deux graphiques intéressants : *lead_time* et *avg_price_per_room*.



En effet, dans ces graphiques, nous observons que la distribution varie en fonction de la variable cible contrairement à toutes les autres variables comme *no_of_adults* ou *arrival_date* (ci-dessous).



Pour la variable *lead_time*, elle a tendance a avoir des valeurs bien plus élevées lorsque la variable cible vaut 1 (annulation). Il en va de même pour la variable *avg_price_per_room* qui a une légère tendance à la hausse lorsque la variable cible vaut 1.
Nous pouvons donc émettre des premières hypothèses : les réservations ayant été effectuées 150 jours avant le séjour ainsi que les réservations concernant les chambres les plus chères sont très susceptibles d'être annulées.

2. Analyse statistique

Afin d'avoir une idée plus sûre de quelles sont les variables clés, nous réalisons des tests statistiques.

- Test du Chi2 pour les variables qualitatives

Avec pour hypothèse H0 : la variable cible est indépendante de la variable, H1 : la variable cible n'est pas indépendante de la variable.

```
type_of_meal_plan: p-value test chi 2 = 4.951915406087789e-60
room_type_reserved: p-value test chi 2 = 1.5627796772447422e-10
market_segment_type: p-value test chi 2 = 6.748763024557236e-175
```

Toutes les *p-values* sont inférieures à 0.05, et de loin, donc on peut rejeter l'hypothèse H0 et en déduire que toutes les variables qualitatives sont très pertinentes pour le modèle, *market_segment_type* étant la plus intéressante.

- Test de Student pour les variables quantitatives

Avec pour hypothèse H0 : la distribution de la variable sachant cible =1 est la même que la distribution de la variable sachant cible=0, H1 : les distributions sont différentes.

```
no_of_adults: p-value test Student = 2.1251958404620702e-65
no_of_children: p-value test Student = 1.9746021531473454e-09
no_of_weekend_nights: p-value test Student = 4.838631407811381e-30
no_of_week_nights: p-value test Student = 5.450111353334159e-62
required_car_parking_space: p-value test Student = 3.368068076687889e-92
lead_time: p-value test Student = 0.0
arrival_month: p-value test Student = 0.021879873347925648
arrival_date: p-value test Student = 0.043471227593661385
repeated_guest: p-value test Student = 9.739644968354598e-178
no_of_previous_cancellations: p-value test Student = 2.323826982890173e-14
no_of_previous_bookings_not_canceled: p-value test Student = 3.867328299826672e-60
avg_price_per_room: p-value test Student = 2.3570548204656775e-175
no_of_special_requests: p-value test Student = 0.0
```

Nous avons les mêmes résultats que pour les variables qualitatives. Toutes les p-values sont inférieures à 0.05, ainsi la distribution de chaque variable quantitative n'est pas la même en fonction de la valeur de la variable cible.
Malgré ça nous pouvons voir une grande marge entre certaines valeurs des p-values.

Voyons en détail les résultats :

- Pour les variables *no_of_adults, no_of_children*, *no_of_weekend_nights, no_of_week_nights, required_car_parking_space* : il y a une association forte entre ces variables et l'annulation. Certaines configurations, réservations solo où en groupe, famille avec enfants, voyages touristiques, client d'affaires influencent probablement l'annulation.

- *lead_time* et *no_of_special_requests* : la distribution de la variable change radicalement selon la valeur de la variable cible. Plus la réservation est faite longtemps à l'avance, plus le risque d'annulation est élevé, ce que nous avions déjà vu graphiquement. De plus, les clients avec des demandes spéciales semblent moins susceptibles d'annuler.

- *arrival_month* et *arrival_date* : on a une *p-value* significative mais beaucoup plus élevée que les autres, ainsi, l'annulation dépend légèrement du mois et du jour d'arrivée, ce qui peut être lié à la saisonnalité ou à une plus forte tendance à la réservation le weekend par exemple.

- Pour les variables *repeated_guest* et *avg_price_per_room* : on a un impact énorme. Les clients récurrents annulent beaucoup moins souvent, ce qui est logique, ils sont fidèles à l'hôtel. Quant au prix de la chambre, plus elle est chère, plus il y aura une tendance à l'annulation, ce que nous avions déjà vu graphiquement.

- *no_of_previous_cancellations et no_of_previous_bookings_not_cancelled* : ces deux variables ont une distribution qui varie en fonction de la variable cible mais pas autant que ce à quoi l'on pourrait s'attendre.

Pour les deux variables avec des *p-values* arrondies à 0 nous avons vérifié les moyennes et écart-types afin d'être sûr de nos résultats.

Ces analyses statistiques viennent corroborer notre hypothèse précédente faite après l'analyse visuelle : les variables *lead_time,* et *avg_price_per_room* sont très pertinentes pour le modèle. Elles permettent aussi d'ajouter à ces dernières les variables *repeated_guest* et *no_of_special_requests.*

Cependant toutes les variables de la table restent pertinentes à leur échelle.

III. Préparation des données

Nous allons maintenant préparer les donner au LASSO que nous réaliserons ultérieurement et à la modélisation.
Pour commencer, nous avons vérifié que la table ne contient aucune donnée manquante. Etant donné que c'est le cas, nous allons pouvoir passer à l'encodage de nos variables qualitatives, puis séparer les données en set d'entraînement et de test.

1. Encodage

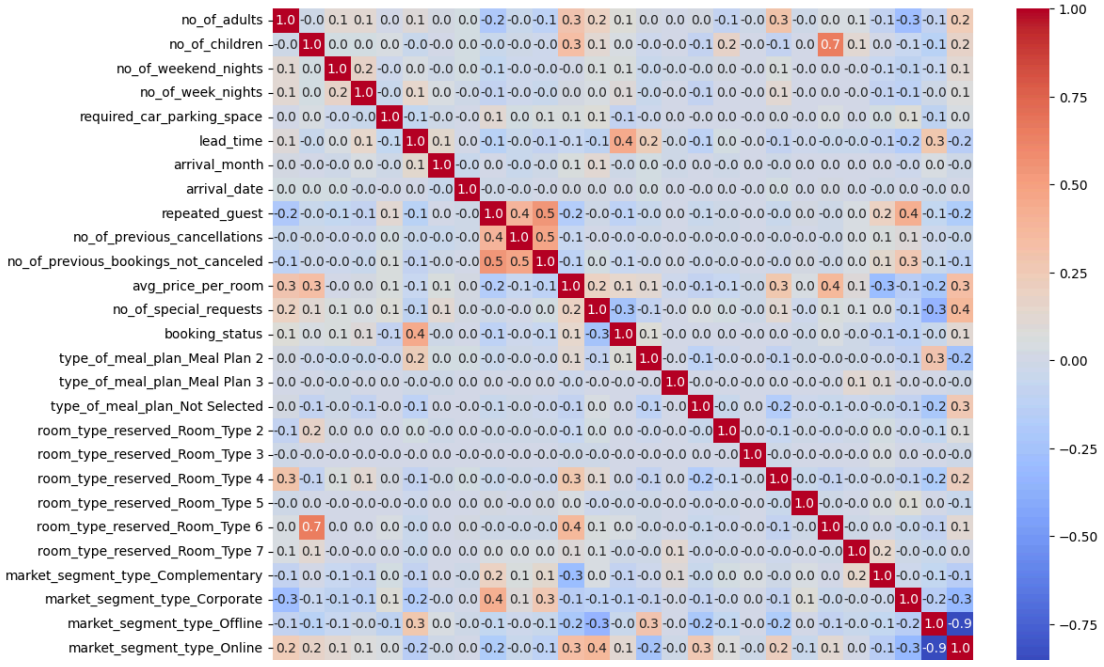
Nous avons 3 variables qualitatives ayant jusqu'à 7 modalités.

```
['Meal Plan 1' 'Not Selected' 'Meal Plan 2' 'Meal Plan 3']
['Room_Type 1' 'Room_Type 4' 'Room_Type 2' 'Room_Type 6' 'Room_Type 5'
 'Room_Type 7' 'Room_Type 3']
['Offline' 'Online' 'Corporate' 'Aviation' 'Complementary']
```

Nous décidons d'utiliser le *One Hot Encoding* pour encoder ces trois variables car c'est l'encodage le plus compatible avec la plupart des modèles. De plus, cela permettra qu'il n'y ait pas de relation d'ordre introduite indirectement entre les valeurs des variables.

Nous utilisons la fonction *get_dummies* de pandas avec l'option *drop_first=True* afin d'éliminer directement la première modalité, ce qui permet d'éviter d'introduire de la corrélation entre variable dans notre nouvelle table encodée.

Afin de vérifier qu'aucune corrélation n'a été introduite dans notre table, nous retraçons la matrice de corrélation des variable quantitatives (toute la table ici sauf la variable cible).



Nous retrouvons le carré de coefficients élevés vu précédemment et repérons deux autres coefficients à considérer à cause de leur valeur extrêmement élevée.
Le premier est le coefficient valant 0.7 entre *room_type_reserved_room_type_6* et *no_of_children*, ce qui doit signifier que le fait de prendre cette chambre augmentent très positivement avec le nombre d'enfants annoncé dans la réservation.
Le deuxième, vaut -0.9 et est entre *market_segment_type_offline* et *market_segment type_online*. Il est tout à fait logique d'avoir un très coefficient élevé et négatif entre ces variables car si la réservation a été faite "en ligne" alors elle n'aura pas été faite "hors ligne" et inversement.
Nous supprimons donc la variable *market_segment_type_offline* de la table car la variable *market_segment type_online* porte l'information des deux variables à elle seule.

2. Standardisation

Nous créons une table standardisée (copie de la table d'origine) que nous utiliserons pour le LASSO. En effet, ce dernier pénalise les coefficients des variables, ainsi, à échelles différentes, les grandes valeurs domineront la régularisation.

Nous pourrons aussi utiliser cette table standardisée pour la régression logistique car cette dernière est basée sur des coefficients linéaires, donc comme pour le LASSO, une échelle différente entre les variables peut fausser l'interprétation des coefficients.

En revanche, pour le modèle de *Random Forest* comme pour celui de *Boosting*, il ne sera pas nécessaire d'utiliser cette table standardisée. Les forêts aléatoires sont basées sur des arbres de décision qui ne sont pas affectés par l'échelle des variables. De même pour *XGBoost* qui fonctionne avec des seuils (et utilise des arbres de décision) et n'est pas sensible aux échelles.

3. Séparation des données en entraînement et test

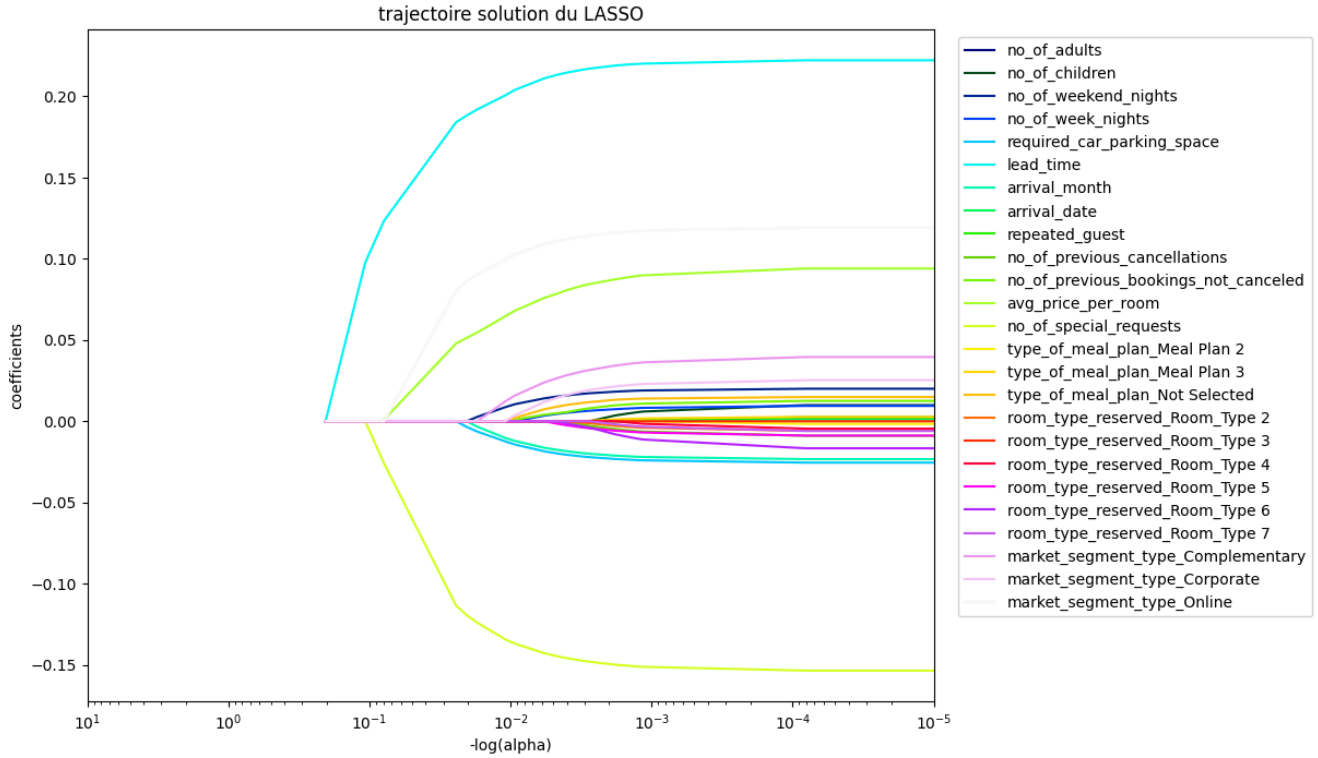
La variable cible est déséquilibrée : 32.8 % de personnes annulent leur réservation. Ainsi, nous devons respecter cette répartition dans le jeu d'entraînement et de test. Nous allons donc utiliser un échantillon stratifié pour avoir la même répartition dans les deux jeux de données.

Nous séparons les *dataframes* standardisés et non standardisés en différents set d'entraînement (80%) et de test (20%) pour les différentes parties où nous utilisons l'un ou l'autre.

IV. Sélection des variables : LASSO

Nous réalisons après la préparation de nos données une sélection des variables grâce au LASSO, afin de préparer au mieux notre table pour la modélisation. En effet, le LASSO est utile pour éliminer les variables inutiles, ce qui permet de réduire la complexité du modèle, moins il y a de variables plus le modèle est interprétable. Il permet aussi d'éviter le surajustement en supprimant les variables non pertinentes.

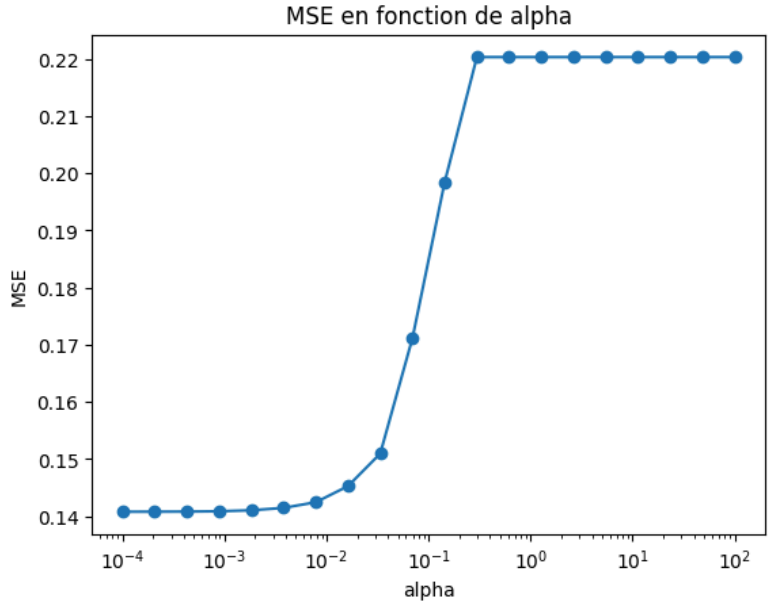
A partir de nos données d'entraînement standardisées nous traçons la trajectoire du LASSO afin d'avoir une représentation visuelle des valeurs des coefficients en fonction de *alpha*.



Nous pouvons voir que le premier coefficient qui prend une valeur non négative est celui de la variable *lead_time*, viennent ensuite ceux des variables *no_of_special_requests* et *avg_price_per_room* ce qui semble tout à fait logique car nous avons déjà établie la pertinence de ces variables précédemment.

Ensuite nous traçons la valeurs de la *mean squared error* en fonction de 20 valeurs de *alpha* entre 10^{-4} et 10^{2}, afin d'avoir une idée de l'évolution de celle-ci. Nous faisons aussi ressortir la valeur optimale selon l'algorithme du LASSO de *alpha* avec une validation croisée 5 découpages et la *MSE* associée à cette valeur.

```
alpha optimum :  0.00033387607703134635
MSE avec le alpha optimum:  0.1407800693135113
```



Nous pouvons remarquer que la valeur de *alpha* choisie par le LASSO ($\simeq 3x10^{-4}$) ne fait pas de sélection de variables. En effet, on peut voir avec la trajectoire du LASSO, qu'à cette valeur de *alpha*, aucun coefficient n'est nul.

Enfin, afin de confirmer cela, nous faisons ressortir les valeurs des coefficients que le LASSO a associé à chaque variable pour le *alpha* optimal.

```
lead_time: 0.2208
no_of_special_requests: -0.1526
market_segment_type_Online: 0.1181
avg_price_per_room: 0.0924
market_segment_type_Complementary: 0.0385
required_car_parking_space: -0.0262
market_segment_type_Corporate: 0.0252
arrival_month: -0.0227
no_of_weekend_nights: 0.0185
room_type_reserved_Room_Type 6: -0.0159
type_of_meal_plan_Not Selected: 0.0157
no_of_previous_bookings_not_canceled: 0.0106
no_of_week_nights: 0.0106
no_of_children: 0.0093
room_type_reserved_Room_Type 5: -0.0070
room_type_reserved_Room_Type 7: -0.0064
no_of_previous_cancellations: -0.0051
room_type_reserved_Room_Type 2: -0.0048
room_type_reserved_Room_Type 4: -0.0040
no_of_adults: 0.0024
type_of_meal_plan_Meal Plan 3: 0.0019
room_type_reserved_Room_Type 3: 0.0012
arrival_date: 0.0007
type_of_meal_plan_Meal Plan 2: -0.0007
repeated_guest: 0.0003
```

On peut voir qu'avec la valeur optimale de *alpha* choisit par le modèle, aucun coefficients n'est nuls, ce qui vient confirmer les résultats des tests d'indépendance réalisés dans notre analyse, le LASSO n'exclue aucune variable du modèle, elles sont toutes plus ou moins utiles.
Par ailleurs, les valeurs des différents coefficients viennent soutenir l'idée que nous avions, du fort impact de certaines variables sur les futures prédictions.
Pour rappel, si $\beta > 0$ alors la variable a un effet positif sur la cible (augmente la probabilité 1 de l'événement cible), alors que, si $\beta < 0$ la variable a un effet négatif sur la cible (diminue la probabilité 1 de l'événement cible).

Lorsque nous entraîneront nos modèles par la suite, garderons la sélection du LASSO.

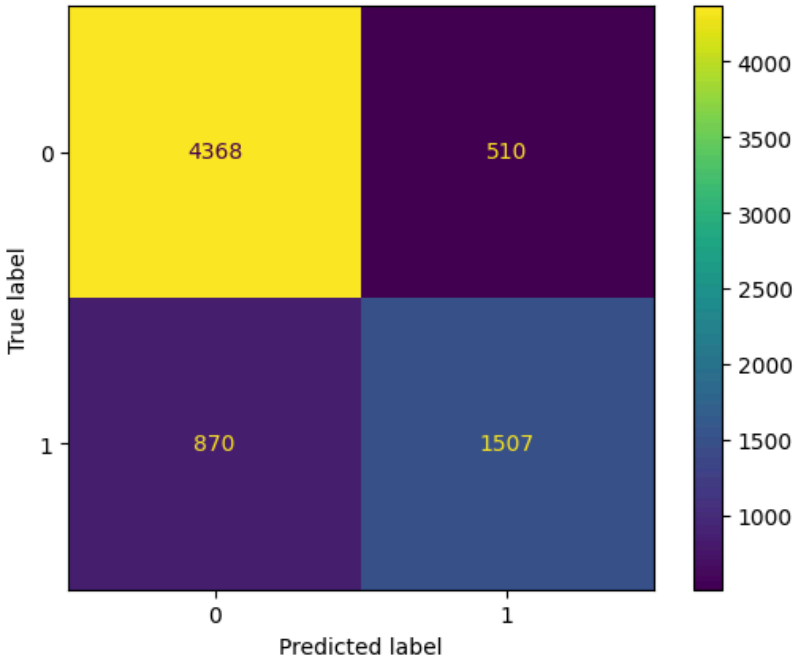
IV. Modélisation

1. Régression logistique

Sachant que la variable que nous cherchons à prédire est binaire (après encodage de cette dernière), le modèle de régression logistique paraît être la méthode appropriée pour commencer notre modélisation. C'est en effet un modèle qui nous permet de poser une base pour ensuite continuer avec des modèles plus complexes, jusqu'à arriver à une bonne prédiction avec le modèle le plus simple possible.

Nous avons modélisé la régression logistique grâce au package *Sklearn Logistic Regression*. Aucune pénalisation n'a été utilisée dans cette étape. Une fois le modèle entraîné, nous avons réalisé des prédictions et nous avons comparé les résultats avec notre échantillon de validation.

Nous avons obtenu une *Mean Square Error* de 0.19 ce qui est tout à fait correcte pour un premier modèle très simple et avons procédé par la matrice de confusion pour voir les spécificités de nos prédictions.

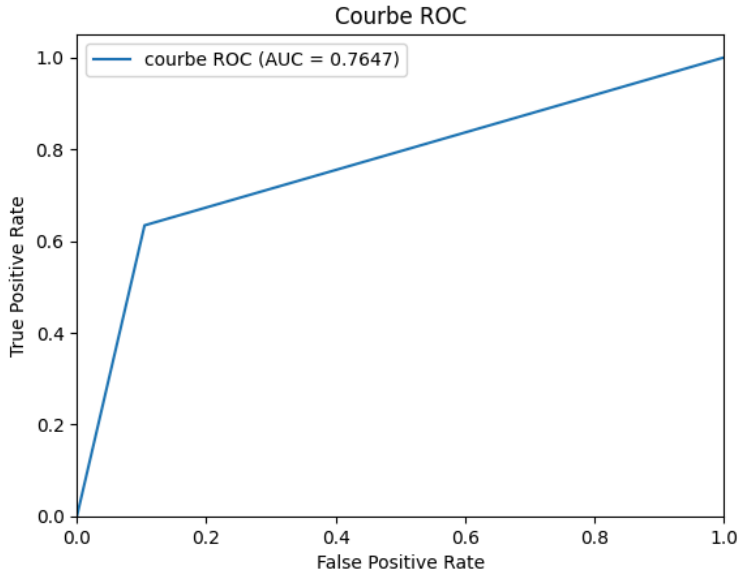


Sachant que "*Not_Cancelled*" correspond à l'encodage 0, nous pouvons dire que la régression logistique, bien qu'elle soit une méthode très simpliste à bien réussi à prédire les annulations des réservations dans 1507 cas soit 63% des cas.

Pour analyser les résultats nous avons utilisé plusieurs scores comme l'AUC, le *recall*, la précision et le F1-Score. Nous ne nous servirons pas de l'*accuracy* du fait que les classes sont déséquilibrées.

	precision	recall	f1-score	support
0	0.83	0.90	0.86	4878
1	0.75	0.63	0.69	2377
accuracy			0.81	7255
macro avg	0.79	0.76	0.77	7255
weighted avg	0.81	0.81	0.81	7255

Nous remarquons que la classe 0 est mieux prédite que la classe 1, et ceci peu importe le score. Dans l'ensemble, les scores sont acceptables mais visiblement améliorables. Il est important de mettre l'accent sur la grande différence en termes de qualité parmi la prédiction des classes.



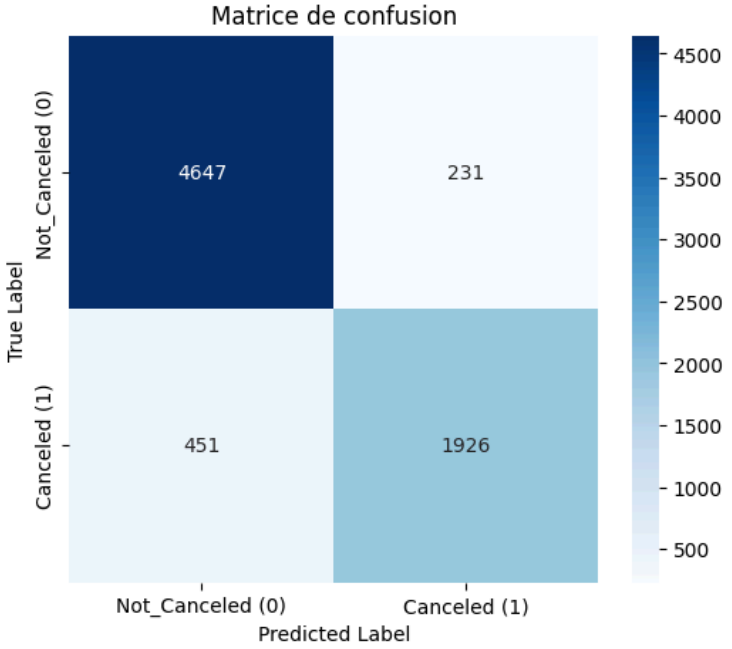
Finalement, nous analysons l'AUC qui vaut 0.765 et visualisons la courbe ROC. Cela nous montre que le modèle est meilleur que le hasard lorsqu'il s'agit de prédire les classes. Cette performance peut être considérée comme modérée, nous utiliserons donc d'autres approches afin de développer un modèle plus performant.

2. Random Forest

Après avoir observé les résultats avec une régression logistique, nous voulions améliorer les résultats notamment sur les annulations non prédites par le modèle.

Pour cela, la classification avec *Random Forest* nous paraissait être une solution idéale. En effet, le modèle gère très bien la non-linéarité des variables contrairement à la régression logistique, il ne nécessite pas de normaliser les données et est robuste en ce qui concerne les valeurs aberrantes. De plus, l'outil *feature_importances_* de Scikit-learn nous permet d'essayer une sélection de variable autre que celle du LASSO.

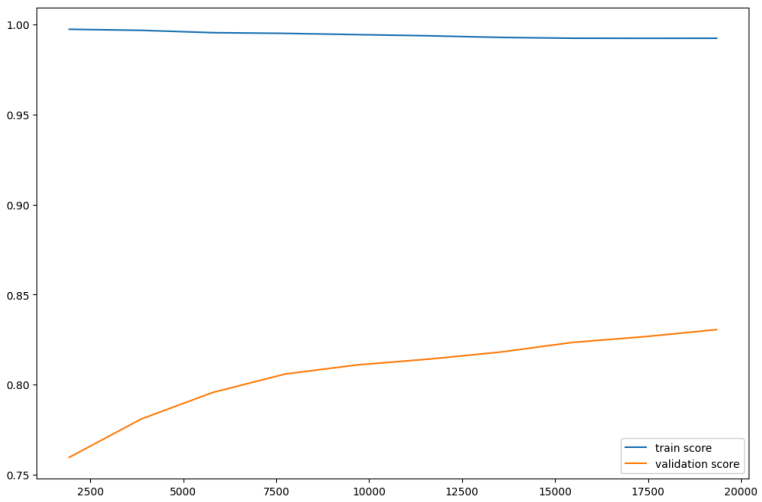
Voici la matrice de confusion et les différents scores du modèles :



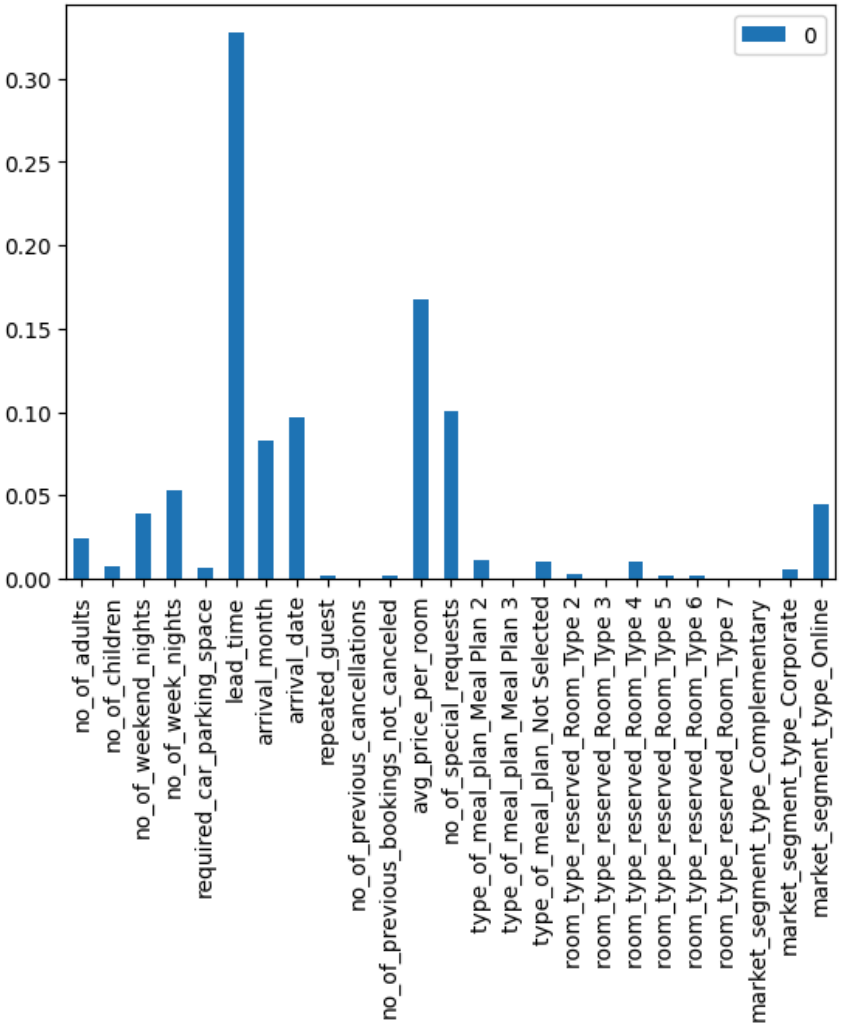
	precision	recall	f1-score	support
0	0.91	0.95	0.93	4878
1	0.89	0.81	0.85	2377
accuracy			0.91	7255
macro avg	0.90	0.88	0.89	7255
weighted avg	0.91	0.91	0.90	7255

Au vu des résultats, nous pouvons affirmer que nous avons de meilleurs résultats que précédemment. La prédiction des classes 0 et 1 sont plus homogènes avec un score de précision plus élevé d'après le rapport de classification.

Voyons maintenant ce qu'il en est du surapprentissage. Pour cela nous traçons la *learning curve*.



La *learning curve* nous montre dans un premier temps que plus il y a de données plus le score de validation augmente. Néanmoins, dans un second temps, on observe qu'il y a un écart considérable entre les scores d'entraînement et de validation, ce qui suppose que le modèle est peut être victime d'un surapprentissage. Ajouter des données améliorera sûrement le modèle car le score d'entraînement diminue petit à petit au contraire du score de validation qui lui augmente et ne semble pas se stabiliser.



A l'aide de *feature_importances_* nous pu voir quel ordre d'importance attribuait le modèle Random Forest aux variables. Ce dernier est quasiment similaire à celui du LASSO : *lead_time* est la variable la plus influente du modèle suivie de la variable *avg_price_per_room* et *no_of_special_requests.*

En utilisant dans le modèle les variables avec des valeurs supérieurs à 0.01, nous obtenons les mêmes résultats que sans la sélection de variable et cela reste meilleur que la régression logistique.

Nous avons donc pu développer un modèle assez robuste avec des scores de précision assez élevés.

Malgré les très bons résultats du modèle de *Random Forest*, nous souhaitions améliorer notre modèle par rapport aux problèmes importants du manque de généralisation et la tendance à trop s'adapter aux données d'entraînement (sur apprentissage). Ce problème ne se résout pas avec une éventuelle sélection de variable, nous avons donc choisi, pour essayer d'y pallier, de développer un modèle de *Boosting*.

3. Boosting

L'objectif ici est de prédire les annulations de réservation du groupe INN Hotels en utilisant un modèle XGBoost ; un algorithme de gradient boosting basé sur les arbres de décision.

Premièrement, nous effectuons une optimisation bayésienne des hyperparamètres avec *BayesSearchCV* pour améliorer la performance du modèle.

L'optimisation des hyperparamètres a permis d'obtenir ces meilleurs paramètres :

- Arbre de décision comme estimateur de base (*base_estimator*)
- Modèle basé sur des arbres de décision (*booster*)
- Taux d'apprentissage plutôt bas à 0.35 (*learning_rate*)

Le taux d'apprentissage contrôle la vitesse à laquelle le modèle ajuste ses poids à chaque itération. Si le taux est élevé, le modèle est plus rapide mais il y a un risque de surajustement. Si le taux est bas le modèle est plus lent mais fait une meilleure généralisation (ce que nous recherchons). Ainsi, après que nous ayons essayé plusieurs intervalles pour ajuster au mieux le taux d'apprentissage, l'optimisation a déterminé un taux d'apprentissage de 0.35, ce qui permet un apprentissage relativement rapide, mais sans surapprentissage.

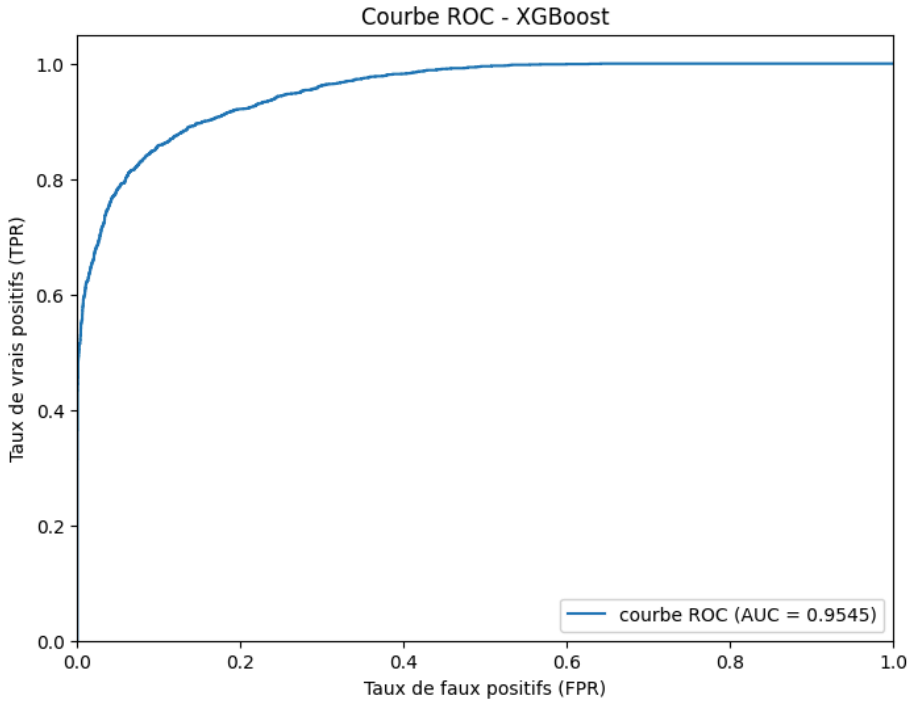
Avec ces paramètres, l'AUC vaut 0.948 sur le set d'entraînement.

Voici un tableau détaillé des résultats du modèle sur le set de test selon plusieurs métriques.

	precision	recall	f1-score	support
0	0.91	0.94	0.92	4878
1	0.87	0.80	0.83	2377
accuracy			0.89	7255
macro avg	0.89	0.87	0.88	7255
weighted avg	0.89	0.89	0.89	7255

Nous pouvons voir que les "non annulation" (classe 0) sont très bien prédite avec 91% de précision et 94% de rappel. En revanche, les "annulation" (classe 1) sont moins bien prédite avec un rappel de 80%, ce qui signifie que 20% des annulations ne sont pas détectées. Cela peut poser un problème quant-à la qualité du modèle car c'est ce que nous cherchons à prédire.

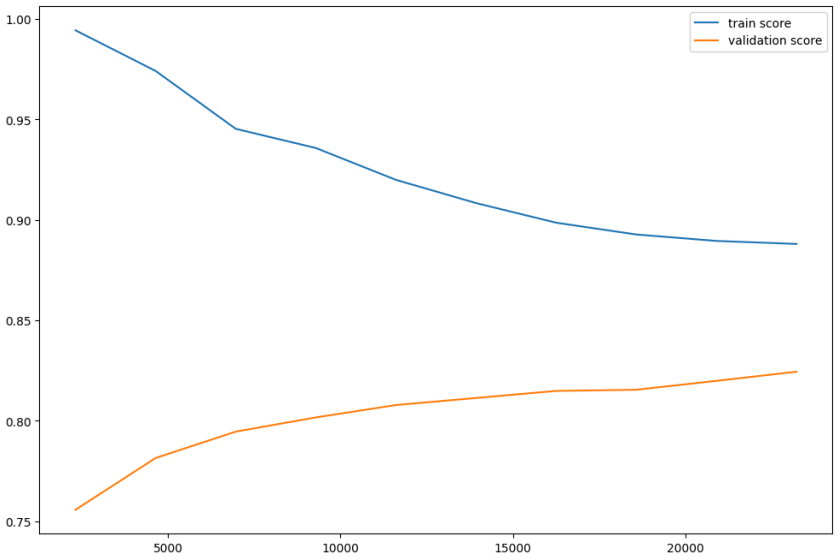
La courbe ROC montre un très bon compromis entre faux positifs et vrais positifs. En effet, plus la courbe se rapproche du coin en haut à gauche, plus le compromis est bon, ce qui est le cas ici.



De plus, nous avons obtenu un score AUC de 0.9545, c'est un score excellent, qui indique une très bonne séparation entre les réservations annulées et non annulées.

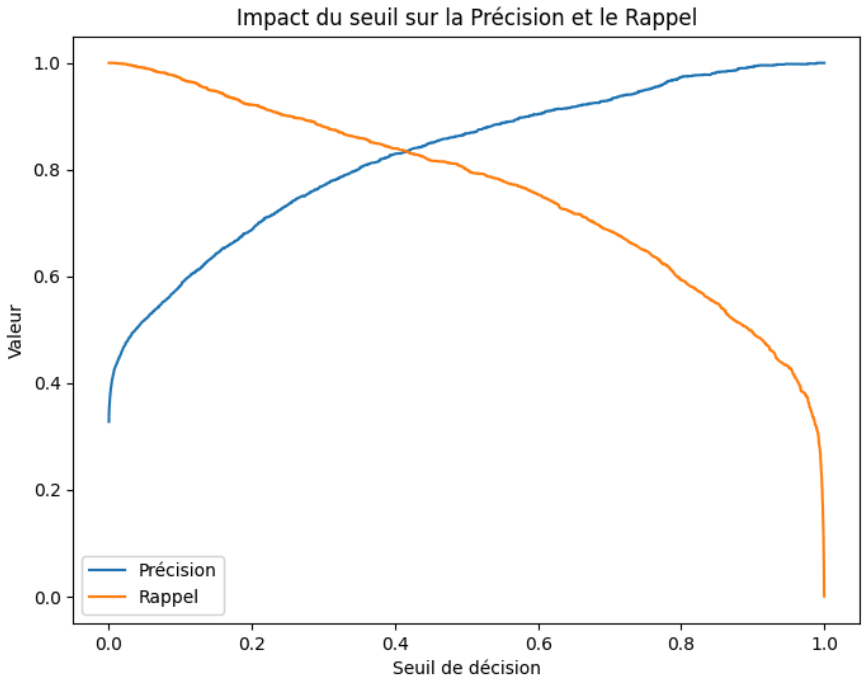
Globalement, le modèle est très bon mais éventuellement non suffisant pour notre problématique et moins bon que celui de Random Forest pour la prédiction.

En revanche, ce modèle basé sur le boosting permet de résoudre le problème de surapprentissage :



La *learning curve* nous apprend que plus l'on a de données, plus les scores des set d'apprentissage et de test convergent l'un vers l'autre, ce qui est une très bonne chose.

Enfin, il aurait été bien de connaître le seuil qu'aurait aimé fixer le groupe INN Hotels pour pouvoir améliorer notre modèle au mieux.



Nous pouvons observer ici l'impact sur seuil sur la précision et le rappel, et ainsi voir que celui-ci est très important.

Conclusion

Face au fléau des annulations de réservations, nous proposerons comme modèle de *Machine Learning* le modèle généré à partir d'un algorithme de *Boosting*. Ce dernier permettra, bien qu'il ai un score légèrement moins bon, d'éviter le sur-apprentissage sur les données d'entraînement. Une alternative qui se serait inscrite dans la suite de notre projet mais qui n'a pu être réalisée par manque de temps aurait été de combiner ces deux modèles afin d'optimiser la capacité de détection des annulations tout en limitant l sur-apprentissage.
Malgré cela, le modèle que nous proposons sera un outil d'aide à la décision qui permettra de prédire de façon pertinente les réservations susceptibles d'être l'objet d'une annulation. Grâce à cela, le groupe INN Hôtels sera en capacité de prendre des mesures lui permettant de diminuer les pertes et manques à gagner liés aux annulations.

En effet, plusieurs solutions s'offrent à cette entreprise.

Un premier point important est de relever que l'hôtel pourrait cesser d'ouvrir ses réservations plus de 150 jours avant le jour du séjour. Faire cela permettrait d'atténuer de manière importante les annulations car nous avons vu à de multiples reprises que le temps de réservation avant le séjour est le facteur le plus influent sur l'annulation de la réservation.
Dans un second temps, nous avons pensé qu'il serait possible pour le groupe d'apposer plusieurs tarifications. Une tarification de base (moins chère) ne permettant pas d'annuler, où le paiement serait dû même en cas de non présence de la personne ou à des frais supplémentaires permettant d'annuler la réservation ce qui permettrait à l'établissement de pouvoir limiter la perte potentielle.
Une autre solution serait de survendre l'hôtel. Cette pratique est largement présente dans le domaine du transport des passagers, notamment dans le domaine aérien. L'idée serait de créer un système permettant de vendre autant de chambres que disponibles en ajoutant celles qui sont susceptibles d'être libérées à cause d'annulations. En revanche, cela pourrait poser problème si des clients se retrouvent sans chambre, car ça pourrait nuire à la réputation du groupe.

En somme, plusieurs opportunités s'offrent à cette entreprise. Aussi, une discussion sur le seuil (balance précision/recall) d'annulation que le groupe souhaite détecter serait pertinente afin de rendre le modèle encore plus efficace et personnalisé pour le groupe. Il est aussi possible de continuer à améliorer le modèle, comme cité plus haut, ou selon la solution que le groupe INN Hotels souhaite entreprendre.
Par exemple, il serait intéressant de créer un outil permettant de créer une tarification dynamique tenant compte du risque d'annulation.

D'un point de vue académique, ce projet a été enrichissant tant du côté technique que du côté "business".
L'utilisation de plusieurs méthodes abordées en cours nous a permis de les tester dans un cadre réel et de jauger leur intérêt à chacune en fonction du contexte de notre étude.

Comme mentionné précédemment, nous avons procédé à des recherches afin de connaître quelles étaient les solutions apportées dans d'autres domaines qui subissent le même genre de problème.
De plus, ce sujet peut être lié à des sujets d'importance majeur comme la place du sur-tourisme dans les économies modernes et ainsi amener à une réflexion plus poussée.

Annexe

1 Packages

[ ]:
```
#package
import pandas as pd
import seaborn as sns
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
#import skimpy
import warnings
warnings.filterwarnings("ignore")
import sklearn.preprocessing as pre
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

#lasso
from sklearn.linear_model import lars_path, Lasso, LassoCV, LogisticRegression

#reg log
import sklearn.model_selection as ms
import sklearn.linear_model as lm
import sklearn.metrics as mt

#boosting
import xgboost as xgb
from sklearn.ensemble import StackingClassifier
from skopt import BayesSearchCV
from skopt.space import Real, Categorical
from sklearn.tree import DecisionTreeClassifier
from timeit import default_timer as timer

#RF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

#évaluation de modèle
```

```
from sklearn.metrics import accuracy_score, mean_squared_error, roc_auc_score,␣
 ↪classification_report, roc_curve, auc, precision_recall_curve, f1_score,␣
 ↪confusion_matrix
from sklearn.model_selection import learning_curve
```

2 Import des données et informations sur le Dataframe

[ ]:
```
#import
Hotels = pd.read_csv("/kaggle/input/innhotelsgroup/INNHotelsGroup.csv", sep=",")
Hotels = Hotels.drop(columns=['Booking_ID','arrival_year'])

#informations
Hotels.info()
Hotels.head()
```

3 Compréhension et visualisation des données

[ ]:
```
#séparation des variables qualitatives et quantitatives
Hotels_quant = Hotels.select_dtypes(exclude='object')
Hotels_qual = Hotels.select_dtypes(include='object').
 ↪drop(columns='booking_status')
```

3.1 Analyse univariée

[ ]:
```
#statistiques decriptives
skimpy.skim(Hotels)

#vérification pour les valeurs manquantes
print(Hotels.isna().sum().sum())
```

[ ]:
```
#modalités des variables qualitatives
print(Hotels['type_of_meal_plan'].unique())
print(Hotels['room_type_reserved'].unique())
print(Hotels['market_segment_type'].unique())
print(Hotels['booking_status'].unique())
```

[ ]:
```
#visualisation des variables quantitatives
fig, ax = plt.subplots(5,3, figsize = (20,20))
for i, subplots in zip(Hotels_quant.columns, ax.flatten()) :
    sns.histplot(Hotels_quant[i], ax = subplots)
fig.show()
```

[ ]:
```
fig, ax = plt.subplots(5,3, figsize = (20,20))
for i, subplots in zip(Hotels_quant.columns, ax.flatten()) :
```

```
    sns.boxplot(Hotels_quant[i], ax = subplots)
fig.show()
```

3.2 Analyse multivariée

[ ]:
```
Hotels_corr = Hotels_quant.corr()
plt.figure(figsize=(10,8))
sns.heatmap(Hotels_corr, fmt='.2f', annot= True)
plt.show()
```

4 Analyse du lien entre la variable cible et les données

4.1 Analyse visuelle

[ ]:
```
# définir la variable cible
target = 'booking_status'
Hotels[target]=  np.where(Hotels[target] == "Not_Canceled",0,1) # On passe la␣
 ↪variable en 0/1
sns.catplot(x=target,kind="count",data=Hotels)

print(Hotels[target].mean())
```

[ ]:
```
# répartition d'une variable en fonction de la variable cible
sns.boxplot(x=target,y="lead_time",data=Hotels)
```

[ ]:
```
# répartition d'une variable en fonction de la variable cible
sns.boxplot(x=target,y="no_of_special_requests",data=Hotels)
```

[ ]:
```
# répartition d'une variable en fonction de la variable cible
sns.boxplot(x=target,y="avg_price_per_room",data=Hotels)
```

[ ]:
```
numeric_cols = Hotels[['no_of_adults', 'arrival_date', 'lead_time',␣
 ↪'avg_price_per_room']]
all_cols = list(numeric_cols)

fig, axes = plt.subplots(2,2, figsize=(12, 10))
axes = axes.flatten()

for i, col in enumerate(numeric_cols):
    axes[i].hist(Hotels.loc[Hotels['booking_status'] == 1, col], bins=30,␣
 ↪alpha=0.5, color='blue', label='Canceled', edgecolor='black', density=True)
    axes[i].hist(Hotels.loc[Hotels['booking_status'] == 0, col], bins=30,␣
 ↪alpha=0.5, color='orange', label='Not Canceled', edgecolor='black',␣
 ↪density=True)

    # Titres et légendes
```

```
    axes[i].set_title(f"Distribution de {col}")
    axes[i].set_xlabel(col)
    axes[i].set_ylabel("Densité")
    axes[i].legend()
    axes[i].grid(True)

plt.show()
```

4.2 Analyse statistique

Pour les variables catégorielles, utilisation d'un test du Chi2.

Hypothèse H0 : Les variables sont indépendantes

Hypothèse H1 : Les variables ne sont pas indépendantes

Si la valeur de la pvalue est inférieure à 5% on rejette l'hypothèse H0 avec un risque de 5%. On considère que les deux ne variables ne sont pas indépendantes

[ ]:
```
for v in Hotels_qual.columns.tolist():
    if v!=target:
        cont = Hotels[[v, target]].pivot_table(index=v, columns=target,␣
 ↪aggfunc=len).fillna(0).copy().astype(int)
        st_chi2, st_p, st_dof, st_exp = st.chi2_contingency(cont)
        print(v + ": p-value test chi 2 = " + str(st_p))
```

Pour les variables quantitatives, utilisation d'un test du Student.

Hypothèse H0 : La distribution de la variable quantitative sachant target = 1 est la même que la distribution de la variable quantitative sachant target = 0

Hypothèse H1 : Les distributions sont différentes

Si la valeur de la p-value est inférieure à 5% on rejette l'hypothèse H0 avec un risque de 5%. On considère que les distributions ne sont pas identiques

[ ]:
```
for v in Hotels_quant.columns.tolist():
    if v!= target:
        a=list(Hotels[Hotels[target]==0][v])
        b=list(Hotels[Hotels[target]==1][v])
        st_test, st_p = st.ttest_ind(a, b, axis=0, equal_var=False)
        print(v + ": p-value test Student = " + str(st_p))
```

–> toutes plus ou moins pertinentes

[ ]:
```
#vérif pour les p-value arrondie à 0
print(Hotels.groupby(target)[['lead_time','no_of_special_requests']].mean())
print(Hotels.groupby(target)[['lead_time','no_of_special_requests']].std())
```

–> on a bien des moyenne différentes et des écart significatifs entre chaque groupe de la variable cible pour les deux variables

5 Préparation des données

5.1 Encodage

[ ]:
```
# One hot encoding avec get_dummies
Hotels = pd.get_dummies(Hotels, columns=Hotels_qual.columns.tolist(),␣
 ↪drop_first=True)
Hotels.head()
```

[ ]:
```
#vérif qu'elle est similaire à la précédente
corr_mat2 = Hotels.corr(numeric_only=True)
plt.figure(figsize=(12,9))
sns.heatmap(corr_mat2, fmt='.1f', annot=True, cmap='coolwarm') #--> ok
plt.show()
```

[ ]:
```
Hotels = Hotels.drop(columns='market_segment_type_Offline')
```

5.2 Standardisation

[ ]:
```
#nouveau dataframe pour garder l'ancien
scaler = StandardScaler()

# on standardise toutes les colonnes sauf la variable cible
Hotels_cr = Hotels.copy()
Hotels_cr[[col for col in Hotels.columns if col != 'booking_status']] = scaler.
 ↪fit_transform(Hotels[[col for col in Hotels.columns if col !=␣
 ↪'booking_status']])

Hotels_cr.head()
```

5.3 Séparation des données en entraînement et test

[ ]:
```
#séparation de la table en entrainement et test en utilisant les données␣
 ↪standardisées
#pour le lasso et la reg log
X_cr = Hotels_cr.drop(columns=[target])
Y_cr = Hotels_cr[target]

X_train_L, X_test_L, Y_train_L, Y_test_L = train_test_split(X_cr, Y_cr,␣
 ↪test_size=0.2, stratify=Y_cr, random_state=42)
```

[ ]:
```
#séparation de la table en entrainement et test en utilisant les données non␣
 ↪standardisées
#pour le boosting et le rf
X = Hotels.drop(columns=[target])
Y = Hotels[target]
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪stratify=Y, random_state=42)
```

6 LASSO

[ ]:
```
#trajectoire du LASSO
X_train_np = X_train_L.astype(float).to_numpy()
Y_train_np = Y_train_L.astype(float).to_numpy()

alpha, active, coefs = lars_path(X_train_np, Y_train_np, method='lasso')

plt.figure(figsize=(10, 8))
colors = plt.cm.get_cmap('gist_ncar', coefs.shape[0])

for i in range(coefs.shape[0]):
    plt.plot(alpha, coefs[i, :], label=X_train_L.columns[i], color=colors(i))

plt.xlabel('-log(alpha)')
plt.ylabel('coefficients')
plt.title('trajectoire solution du LASSO')
plt.xscale('log')
plt.gca().invert_xaxis()
plt.xlim(10, 10**(-5))
plt.legend(loc='upper left', bbox_to_anchor=(1.02, 1))
plt.show()
```

[ ]:
```
#mse en fonction de alpha
#en faisant varier alpha entre entre 10^-4 et 10^2 et on trace la MSE en␣
 ↪fonction des -log(alpha)
alphas = np.logspace(-4, 2, 20) #MSE dépend de alpha
mse_values = []

for a in alphas:
    lasso_model = Lasso(alpha=a)
    lasso_model.fit(X_train_np, Y_train_L)
    Y_pred = lasso_model.predict(X_test_L)
    mse_values.append(mean_squared_error(Y_test_L, Y_pred))

plt.plot(alphas, mse_values, marker='o')
plt.xscale('log')
plt.xlabel('alpha')
plt.ylabel('MSE')
plt.title('MSE en fonction de alpha')

#choix du alpha optimum avec la cross validation
lasso_cv_model = LassoCV(cv=5)  # validation croisée 5 découpage
```

```
lasso_cv_model.fit(X_train_np, Y_train_L)

alpha_opti = lasso_cv_model.alpha_
print("alpha optimum : ", alpha_opti)

Y_pred_cv = lasso_cv_model.predict(X_test_L)

mse_cv = mean_squared_error(Y_test_L, Y_pred_cv)
print("MSE avec le alpha optimum: ", mse_cv)
```

[ ]:
```
# Trier les variables par importance absolue des coefficients
var = X_train_L.columns
coef_opti = lasso_cv_model.coef_
sort_coeffs = sorted(zip(var, coef_opti), key=lambda x: abs(x[1]), reverse=True)

# Afficher les résultats triés
for name, coef in sort_coeffs:
    print(f"{name}: {coef:.4f}")
```

7 Modélisation

7.1 Régression logistique

[ ]:
```
#dataset avec variables choisies
X_train_Lr = X_train_L.iloc[:,lasso_cv_model.coef_ !=0]
X_test_Lr = X_test_L.iloc[:,lasso_cv_model.coef_ !=0]
```

[ ]:
```
#modele de regression logistique
logistic_model_lasso = lm.LogisticRegression(fit_intercept=True)
logistic_model_lasso.fit(X_train_Lr, Y_train_L)
```

[ ]:
```
#prediction et evaluation au sense de la MSE
Y_pred_rl = logistic_model_lasso.predict(X_test_Lr)
mt.mean_squared_error(Y_test_L,Y_pred_rl)
```

[ ]:
```
# Evaluation au sens de la matrice de confusion.
rl_confMatrix = mt.confusion_matrix(Y_test_L,Y_pred_rl)
cmPlot = mt.ConfusionMatrixDisplay(rl_confMatrix)
cmPlot.plot()
```

[ ]:
```
print(mt.classification_report(Y_test_L,Y_pred_rl))
```

[ ]:
```
fpr, tpr, seuils =  mt.roc_curve(Y_test_L, Y_pred_rl)
plt.figure()
plt.plot(fpr, tpr, label=f'courbe ROC (AUC = {mt.roc_auc_score(Y_test_L,␣
↪Y_pred_rl):.4f})')
```

```
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Courbe ROC')
plt.legend()
plt.show()
```

7.2 Random Forest

[ ]:
```
model = RandomForestClassifier(random_state=42)
```

[ ]:
```
def evaluation(model):

    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)

    cm = confusion_matrix(Y_test, Y_pred)
    labels = ['Not_Canceled (0)', 'Canceled (1)']

    #matrice de confusion
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,␣
 ↪yticklabels=labels)
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.title('Matrice de confusion')
    plt.show()

    print(classification_report(Y_test, Y_pred))

    N, train_score, val_score = learning_curve(model, X_train, Y_train, cv=3,␣
 ↪scoring = 'f1', train_sizes = np.linspace(0.1, 1, 10))

    plt.figure(figsize=(12, 8))
    plt.plot(N, train_score.mean(axis=1), label = 'train score')
    plt.plot(N, val_score.mean(axis=1), label = 'validation score')
    plt.legend()
```

[ ]:
```
evaluation(model)
```

[ ]:
```
importance_df = pd.DataFrame(model.feature_importances_, index=X_train.columns)

importances = model.feature_importances_
feature_names = X_train.columns
```

```
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
})

important_features = importance_df[importance_df['Importance'] > 0.01]

print(important_features['Feature'].tolist())
```

[ ]:
```
pd.DataFrame(model.feature_importances_, index=X_train.columns).plot.bar()
```

7.3 Boosting : XGBoost

[ ]:
```
# variables choisies par le LASSO
X_train_r = X_train.iloc[:,lasso_cv_model.coef_ != 0]
X_test_r = X_test.iloc[:,lasso_cv_model.coef_ != 0]
```

[ ]:
```
#HYPERPARAMETRES
#déf des hyperparamètres pour XGBoost
param_dict_XGB = {
    'objective':Categorical(['binary:logistic']), #objectif : effectue une␣
 ↪classification binaire
    'booster':Categorical(['gbtree']), #type de booster : arbre de décision
    'base_estimator' :␣
 ↪Categorical([LogisticRegression(),DecisionTreeClassifier()]), #estimateurs␣
 ↪de base
    'learning_rate' : Real(0.01,0.4, prior='uniform') #taux d'apprentissage
}

#optimisation des hyperparamètres
mod = BayesSearchCV(xgb.XGBClassifier(),param_dict_XGB,n_iter=5,cv=5,scoring =␣
 ↪'roc_auc')
#utilisation de XGBoost pour la classification + on cherche les meilleurs␣
 ↪hyperparamètres + nb iter = 5 + v-fold=5 + on optimise le critère AUC-ROC

#entraînement du modèle
mod.fit(X_train_r, Y_train)

#résultats
best_param_opti_bayes =mod.best_params_ #meilleurs hyperparamètres trouvés
best_score_opti_bayes = mod.best_score_ #meilleur score trouvé
#all_result_opti_bayes = mod.cv_results_ #tous les essais réalisés
#print("\n Essais réalisés : \n", all_result_opti_bayes)
print("\n Paramètres optimaux : \n", best_param_opti_bayes)
print("\n Résultats  : \n", best_score_opti_bayes)
```

```
#entraînement du modèle XGBoost optimisé
xgb_clf = xgb.XGBClassifier(**mod.best_params_) #on applique les meilleurs␣
 ↪hyperparamètres trouvés
xgb_clf.fit(X_train_r,Y_train)
```

[ ]:
```
#PREDICTION ET EVALUATION DU MODELE
#rapport detaillé
Y_pred = xgb_clf.predict(X_test_r)
print(classification_report(Y_test, Y_pred))

#courbe auc-roc
Y_prob = xgb_clf.predict_proba(X_test_r)[:, 1]  # Prédictions des probabilités␣
 ↪pour la classe positive
#roc_auc = roc_auc_score(Y_test, Y_prob) # Calcul du score AUC
#print(f"\n AUC-ROC sur le set de test : {roc_auc:.4f}")
fpr, tpr, thresholds = roc_curve(Y_test, Y_prob) # Calculer la courbe ROC
auc_score = auc(fpr, tpr)  # Calcul de l'AUC

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'courbe ROC (AUC = {auc_score:.4f})')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs (FPR)')
plt.ylabel('Taux de vrais positifs (TPR)')
plt.title('Courbe ROC - XGBoost')
plt.legend(loc='lower right')
plt.show()
```

[ ]:
```
N, train_score, val_score = learning_curve(xgb_clf, X_train_r, Y_train, cv=5,␣
 ↪scoring = 'f1', train_sizes = np.linspace(0.1, 1, 10))
plt.figure(figsize=(12, 8))
plt.plot(N, train_score.mean(axis=1), label = 'train score')
plt.plot(N, val_score.mean(axis=1), label = 'validation score')
plt.legend()
```

[ ]:
```
#IMPACT DE DIFFERENTS SEUILS
precisions, recalls, thresholds = precision_recall_curve(Y_test, Y_prob)

# Tracer la courbe Précision-Rappel
plt.figure(figsize=(8, 6))
plt.plot(thresholds, precisions[:-1], label="Précision")
plt.plot(thresholds, recalls[:-1], label="Rappel")
plt.xlabel("Seuil de décision")
plt.ylabel("Valeur")
plt.title("Impact du seuil sur la Précision et le Rappel")
plt.legend()
plt.show()
```

```
# Définir un nouveau seuil
seuil = 0.4
Y_pred_adjusted = (Y_prob >= seuil).astype(int)

# Afficher les nouvelles performances
print(f"Classification report avec un seuil de {seuil} :")
print(classification_report(Y_test, Y_pred_adjusted))

#courbe auc-roc
fpr, tpr, thresholds = roc_curve(Y_test, Y_pred_adjusted) # Calculer la courbe␣
↪ROC
auc_score = auc(fpr, tpr)  # Calcul de l'AUC

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'courbe ROC (AUC = {auc_score:.4f})')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs (FPR)')
plt.ylabel('Taux de vrais positifs (TPR)')
plt.title('Courbe ROC - XGBoost')
plt.legend(loc='lower right')
plt.show()
```