Projet IAS

Réservation d'hôtel

M’hamed KHOUBABA
Ridouane OUSMANE DOUDOU
Henintsoa ANDRIAMAHADIMBY

Introduction

Lien du dataset:
https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset

Nous avons un dataset qui représente les différentes caractéristiques d’une réservation d’hôtel. Notre projet consiste à prédire si une personne va honorer ou pas sa réservation d'hôtel.

C’est une tâche supervisée de classification binaire, en effet nous avons deux classes à prédire: “cancelled” et “not cancelled”. Pour ce faire, nous avons décidé de comparer deux modèles différents: le Perceptron à une couche et le Random Forest Classifier.

Visualisation des données

Caractéristiques générales du dataset

Le jeu de données est composé de 36275 exemples et de 19 caractéristiques (features). Aucune valeur nulle n'est observée dans le jeu de données. Les données sont de deux types : 14 valeurs numériques (int/float) et 5 chaînes de caractères (string).

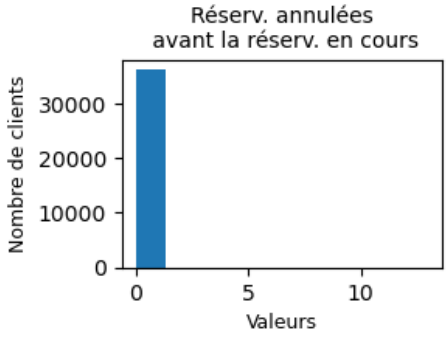
Distribution et visualisation des données

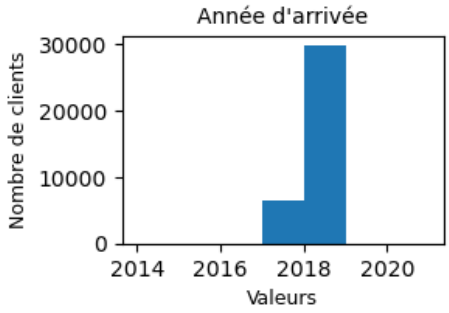
```
data.describe()
```

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	required_car_parking_space	lead_time	arrival_year
count	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000
mean	1.844962	0.105279	0.810724	2.204300	0.030986	85.232557	2017.820427
std	0.518715	0.402648	0.870644	1.410905	0.173281	85.930817	0.383836
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2017.000000
25%	2.000000	0.000000	0.000000	1.000000	0.000000	17.000000	2018.000000
50%	2.000000	0.000000	1.000000	2.000000	0.000000	57.000000	2018.000000
75%	2.000000	0.000000	2.000000	3.000000	0.000000	126.000000	2018.000000
max	4.000000	10.000000	7.000000	17.000000	1.000000	443.000000	2018.000000

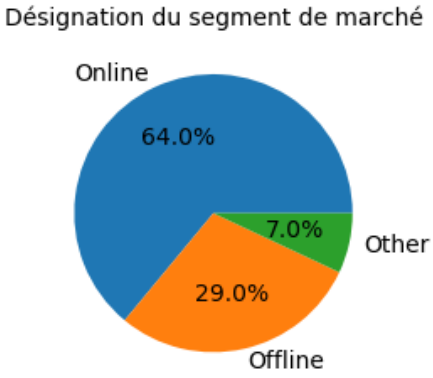
D'après la description du jeu de données, on observe que les valeurs de chaque colonne sont bien réparties à l'exception de certaines pour lesquelles l'écart-type est plus élevé que

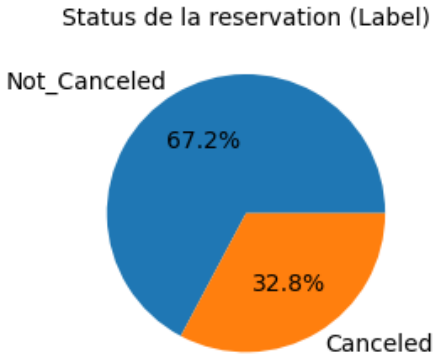
la moyenne. La colonne "arrival_year", quant à elle, présente un écart-type extrêmement faible par rapport à sa moyenne.





Les données de type "string" comportent plusieurs valeurs et l'on remarque que deux d'entre elles dominent. Nous avons donc regroupé les valeurs minoritaires dans une classe "Other".





Preprocessing

Nous allons tout d'abord supprimer la colonne Booking_ID qui représente les identifiants de chaque réservation. Nous supprimerons également les colonnes dont les distributions ne sont pas pertinentes pour l'apprentissage: no_of_children, required_car_parking_space, arrival_year, repeated_guest, no_of_previous_cancellations, et no_of_previous_bookings_not_canceled.

Ensuite, pour le label, nous allons remplacer les valeurs "not_cancelled" par 0 et "cancelled" par 1.

Nous allons ensuite encoder les données catégorielles en vecteurs One-Hot. Étant donné que ces colonnes sont représentées par trois classes (principale_1, principale_2, other), nous allons transformer chaque colonne d'origine en deux colonnes dont le nom sera la valeur des deux principales classes. Pour représenter la classe "other", nous mettrons 0 comme valeur pour les deux colonnes principales.

Enfin, nous allons normaliser les données afin que les valeurs soient comprises entre 0 et 1, pour améliorer la performance et la stabilité du modèle.

Train, Validation et Test

Nous allons commencer par séparer les données en deux ensembles : 80% pour l'entraînement (train) et 20% pour les tests (test). Ensuite, nous allons effectuer une validation croisée à 5 plis sur les 80% de données d'entraînement afin d'optimiser les hyperparamètres du modèle. Nous utiliserons l'accuracy et la MSE pour évaluer la performance des modèles.

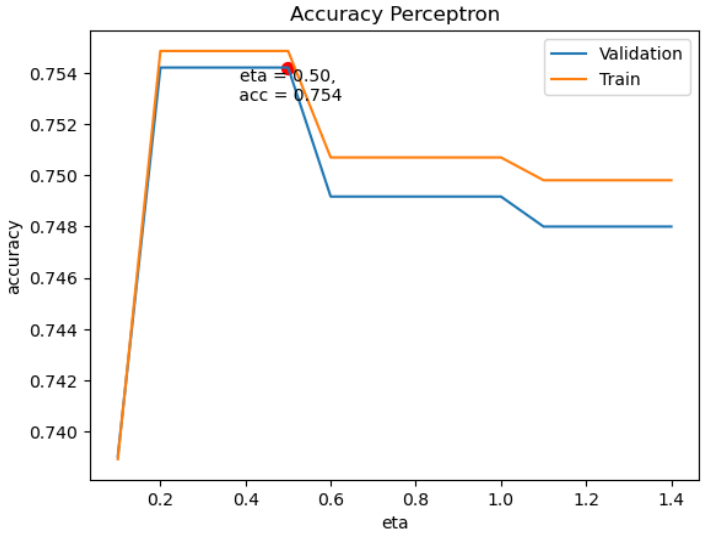
Perceptron

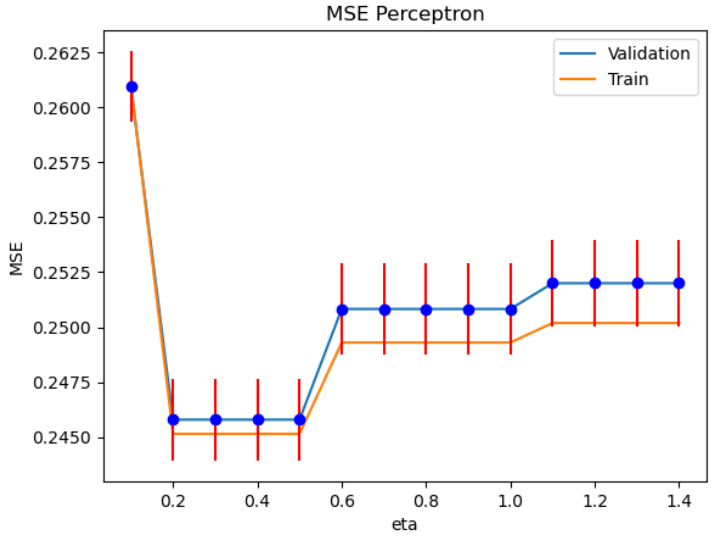
Le perceptron est un algorithme d'apprentissage supervisé de classification binaire. Il utilise une fonction linéaire pour combiner les entrées pondérées et les seuils pour produire une sortie binaire. L'algorithme d'apprentissage du perceptron consiste à ajuster les poids et le biais en fonction des erreurs de classification jusqu'à ce que le modèle puisse correctement classer tous les exemples d'apprentissage.

Nous avons choisi d'optimiser le paramètre **eta** afin d'avoir un bon score et le moins de Loss possible. L'hyper-paramètre eta contrôle la taille des ajustements de poids et de biais effectués lors de la mise à jour du modèle. Si l'hyper-paramètre eta est trop petit, le modèle peut avoir besoin de nombreuses itérations pour converger, ce qui peut rendre l'apprentissage lent. Si l'hyper-paramètre eta est trop grand, les poids et les biais peuvent osciller et ne pas converger vers une solution stable.

Nous allons donc réaliser une cross-validation pour toutes les valeurs de eta comprises entre 0,1 et 1,5, avec un pas de 0,1 à chaque itération. D'après la figure ci-dessous, nous obtenons la meilleure accuracy avec un eta compris entre 0,2 et 0,5. Nous préférons cependant choisir un eta de 0,5 afin d'atteindre rapidement le minimum.

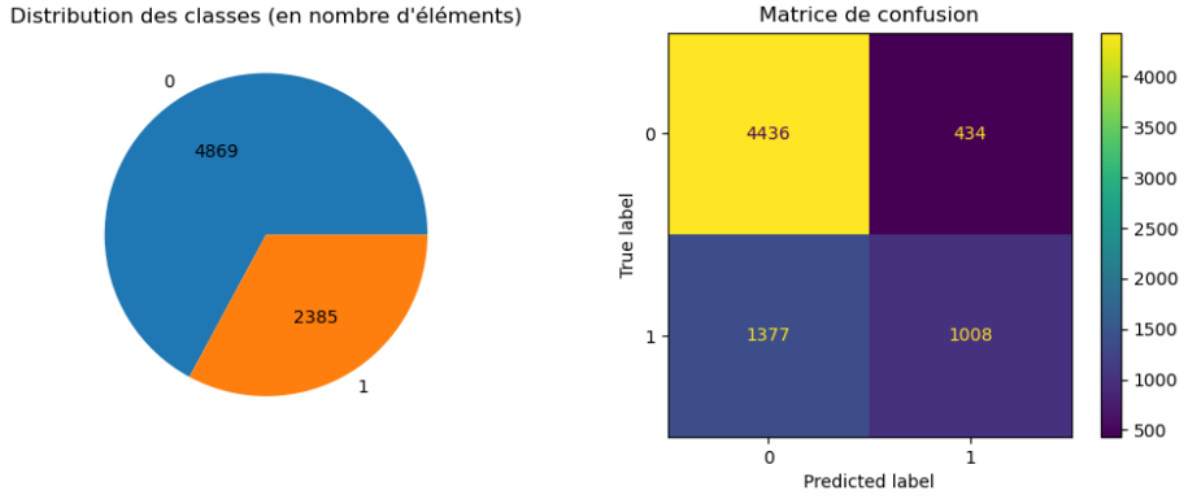
En ce qui concerne la MSE, le choix d'un eta de 0,5 reste pertinent. En effet, la variance de l'erreur est assez faible et constante à tous les niveaux, ce qui assure la fiabilité du modèle.





En effectuant la prédiction, nous obtenons effectivement une précision de 75%. Cependant, en examinant la matrice de confusion, nous constatons que le modèle éprouve des difficultés à prédire la classe 1, qui correspond à la classe "cancelled".

```
Temps de fit : 0.03 secondes
Score: 0.7503790489317712
```



Random Forest Classifier

Le Random Forest Classifier est un algorithme d'apprentissage automatique pour la classification supervisée. Le principe est de construire un grand nombre d'arbres de décision, chaque arbre étant construit sur un sous-ensemble aléatoire des données d'entraînement et sur un sous-ensemble aléatoire des variables d'entrée. Chaque arbre vote pour la classe à laquelle il assigne chaque exemple, et la classe majoritaire est choisie comme la classe prédite par le modèle.

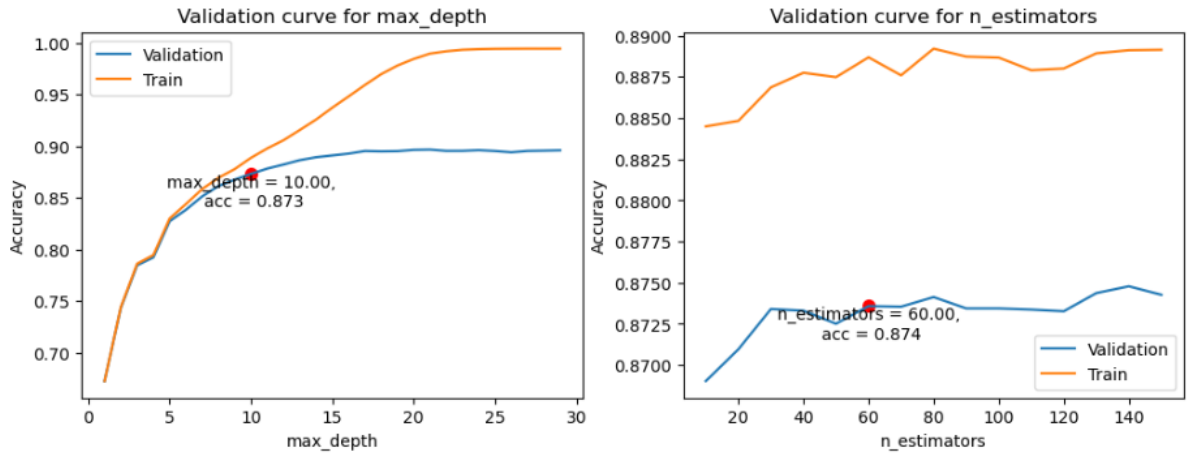
Nous avons choisi d'optimiser les paramètres max_depth et n_estimators qui représente, respectivement, la profondeur maximale de chaque arbre de décision et le nombre d'arbres dans la forêt.

Si max_depth est trop grand, les arbres peuvent sur-apprendre les données d'entraînement et ne pas généraliser correctement aux données de test. Si max_depth est trop petit, les arbres peuvent ne pas être assez profonds pour capturer les relations complexes entre les variables et la variable cible. Le paramètre n_estimators contrôle le nombre d'arbres dans la forêt. En général, un plus grand nombre d'arbres améliore la performance du modèle, mais cela augmente également le temps de calcul et la complexité du modèle.

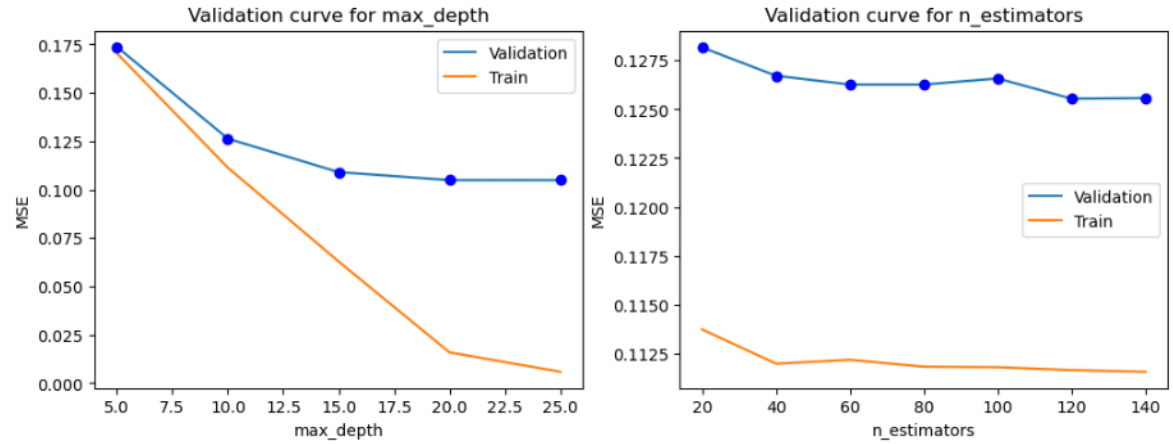
Nous avons commencé par effectuer une Cross-Validation pour optimiser le paramètre max_depth et éviter tout risque de surapprentissage. Nous avons testé les valeurs de max_depth de 0 à 30. Bien que les résultats montrent un très bon taux d'exactitude, nous avons remarqué que le modèle commence à sur-apprendre à partir de max_depth=10. Nous avons donc choisi cette valeur pour obtenir un modèle plus généraliste possible.

Ensuite, pour réduire le temps de calcul, nous avons effectué une nouvelle Cross-Validation, en fixant max_depth=10, pour optimiser le paramètre n_estimators. Nous avons testé les valeurs de n_estimators de 0 à 150 avec un pas de 10. D'après les résultats obtenus, nous

avons choisi la valeur 60 car elle nous permet d'obtenir un bon taux d'exactitude (87%) avec le moins d'arbres possible.

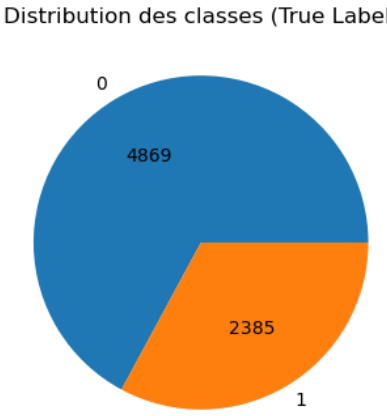


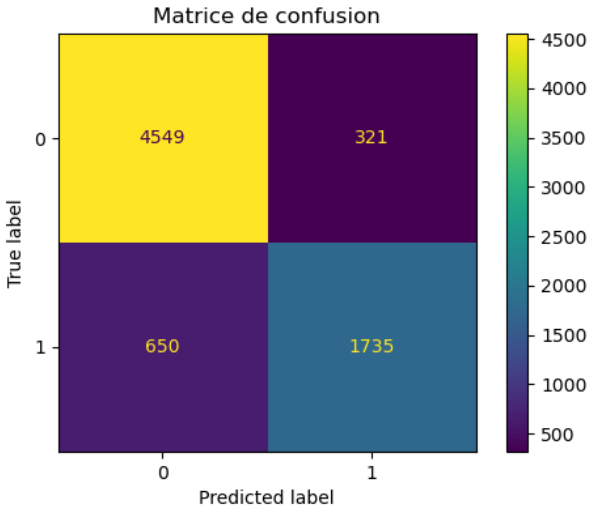
Concernant la Loss, nous pouvons en déduire les mêmes observations que ci-dessus, et nous pouvons également affirmer que la variance de la Loss est extrêmement faible et qu'elle n'est pas visible sur le graphique. Cela signifie que la performance du modèle est cohérente et fiable.



En faisant la prédiction, nous obtenons effectivement la même accuracy de 86%. En observant la matrice de confusion, nous remarquons de bons chiffres sur la diagonale qui représentent les True Positives et les True Negatives, ce qui signifie que le modèle est capable de prédire correctement.

```
Temps de fit : 0.83 secondes
Score: 0.8661612680909717
```





Conclusion

En conclusion, nous pouvons affirmer que le problème de réservation d'hôtel est très intéressant à résoudre en utilisant des techniques de science des données. Notre objectif était de prédire si les clients honoreront ou non leurs réservations d'hôtel. Nous avons réussi à atteindre cet objectif en utilisant deux modèles de classification, à savoir le Random Forest et le Perceptron.

Comparant les deux modèles, nous avons constaté que le Random Forest est meilleur que le Perceptron en termes de score, car il offre une précision supérieure de 10% et peut prédire un nombre plus élevé de True Positif et True Négatif. Cependant, le temps d'apprentissage du Perceptron est 20 fois plus rapide que celui du Random Forest, ce qui est logique étant donné que le Random Forest est un modèle plus complexe.

Cet étude nous a aussi permis d'identifier les caractéristiques qui influencent plus l'état de la prédiction, ce qui peut être intéressant pour les futures prises de décision au sein de l'hôtel.