RAPPORT INTRODUCTION A L'APPRENTISSAGE STATISTIQUE - PROJET

Martin Berthier, Quentin Bertrand, Janethe Ferhoune, Fanoa Razafimbelo, Yohann Blackburn

Référent : Kirian Guiller

Nous allons à travers ce projet, prédire si des passagers d'une compagnie aérienne sont satisfaits ou non à partir d'un dataset. Pour cela, nous allons tout d'abord explorer les données, puis faire le preprocessing du dataset avant d'entraîner les modèles choisis sur ces données.

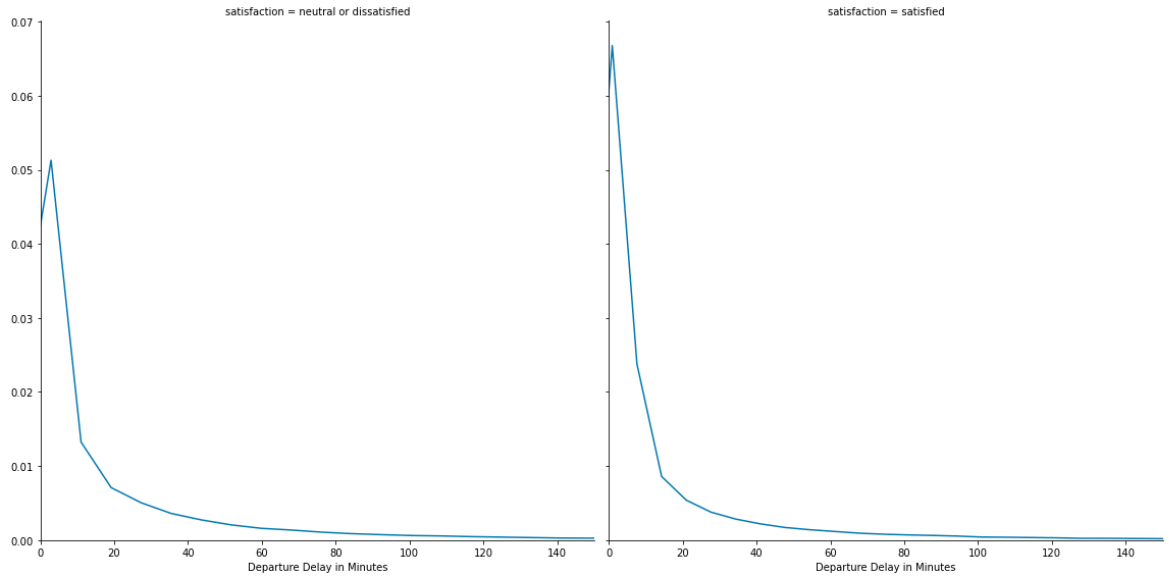
Présentation du dataset:

Il s'agit d'un dataset obtenu à l'aide d'une enquête réalisée à partir des caractéristiques d'un vol comme le temps de retard et la distance de vol et le ressenti des passagers vis à vis de leur expérience chez cette compagnie aérienne, à savoir diverses notes concernants les services offerts avec, entre autre, la qualité de la nourriture, la propreté et le confort général...
Le dataset est constitué d'environ 103 000 lignes et 24 labels, dont la satisfaction qui est encodée comme une variable qui prend 2 valeurs : "satisfait" ou "neutre ou pas satisfait" et qui sera au coeur de notre analyse. Les features du dataset sont en grande partie catégorielles. Par ailleurs, les notations varient entre 0 et 5.

Analyse du dataset:

Le temps de retard au décollage semble être une variable assez intéressante dans notre problème dans la mesure où un long retard devrait pénaliser la satisfaction générale d'un individu interrogé. On s'intéresse à la distribution du temps de retard conditionnellement à la satisfaction finale d'un individu pris au hasard dans les passagers.

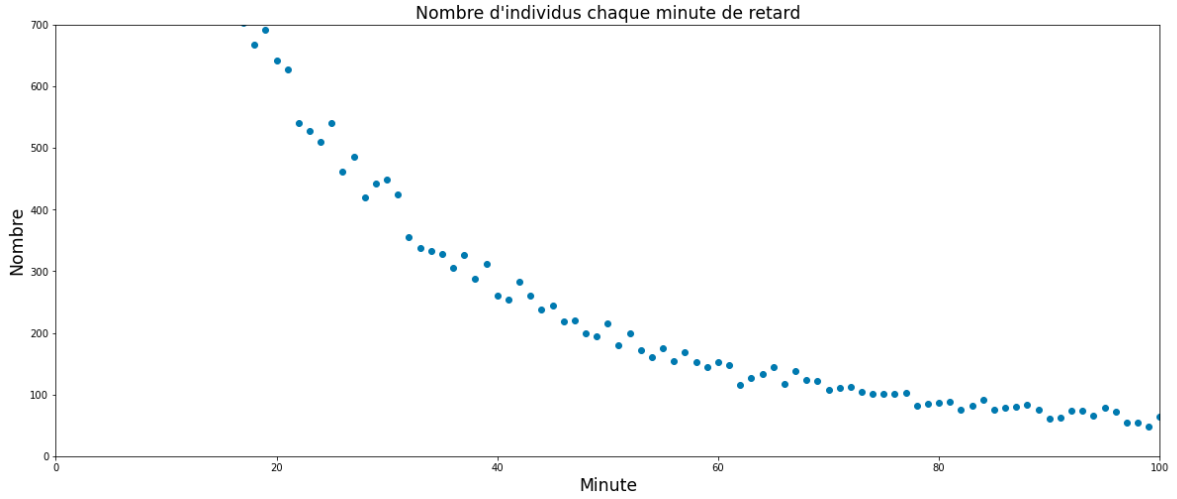
Graphique présentant la distribution de non satisfaction et satisfaction des passagers en fonction du temps de retard au décollage:



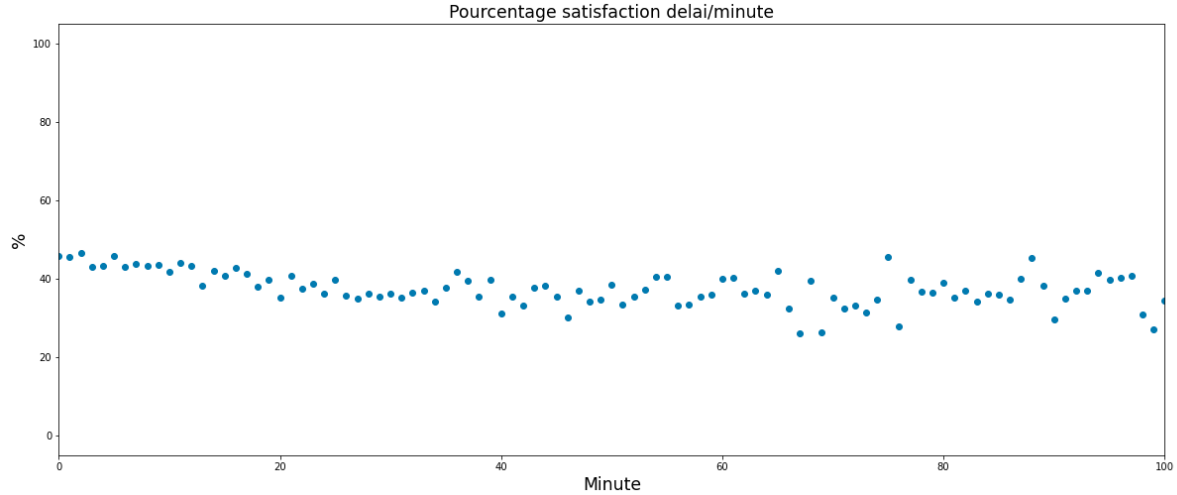
Une personne, qui sera, au final satisfaite de son vol, le sera plus souvent qu'une personne insatisfaite, associée à un temps de retard proche de 0.
Cela se voit à la masse au voisinage de 0 qui est perdue entre le profil satisfait et non satisfait.
Curieusement, une personne qui est satisfaite peut être associée à un temps de retard au début du vol de 40 minutes.

De façon prévisible, un individu insatisfait peut également être associé à un temps de retard également supérieur à 40 min.
Cependant, de façon générale, on observe les mêmes profils de distribution, ce qui laisse supposer que le temps de départ n'a pas tant d'influence que cela dans l'analyse. Nous allons continuer à observer cette variable afin de conforter cette intuition.

On s'intéresse maintenant à la représentation de la loi de probabilité de la satisfaction conditionnellement, cette fois, au temps de retard au décollage. Dans ce cas-ci, on choisit d'interpréter le temps de retard comme une variable discrète à valeur dans les entiers naturels. Cette vision des choses va nous permettre d'interpréter la loi conditionnelle étudiée ici comme une loi discrète et donc de faire appel à un diagramme en bâtons pour sa représentation. Pour que celle-ci soit possible, on doit disposer d'assez d'individus pour chaque minute. Heureusement, du fait de la taille de notre dataset, cela est réalisé comme le montre le graphe suivant:



Il y a une quantité d'individus supérieure à 80 pour chaque minute entre 0 et 100, ce qui sera suffisant pour estimer la probabilité conditionnelle de façon fiable. Par contre, après 100, nous ne disposons pas d'assez d'individus pour que l'estimation soit fiable. On se focalisera donc sur les temps de retard inférieurs à 100 minutes pour la représentation.



(C'est le même graphe que le graphe avec les mêmes intitulés , mais zoomé et avec des intervalles de taille 1)

Un individu ayant attendu un nombre de minutes entre 0 et 40 min sera assez pareillement enclin à être satisfait de son vol. Néanmoins, on peut relever un très léger mécontentement croissant d'un individu en fonction du nombre de minutes qu'il a eu à attendre toujours entre 0 et 40 min. En effet, la satisfaction d'un individu conditionnellement au temps de retard au décollage est très légèrement décroissante entre 0 et 40 min où elle varie entre 45% et 40%. Après 40, cela est difficile à dire en raison des fluctuations de la probabilité d'être satisfait. Ces fluctuations sont dues au manque d'individus ayant attendu un nombre de minutes donné. On peut par ailleurs remarquer que curieusement, la moyenne de ces probabilités a l'air de tourner autour de 40%.

Cela a de nouveau l'air de conforter l'idée que la variable de retard au départ ne joue pas un rôle prépondérant dans l'évaluation de la satisfaction.

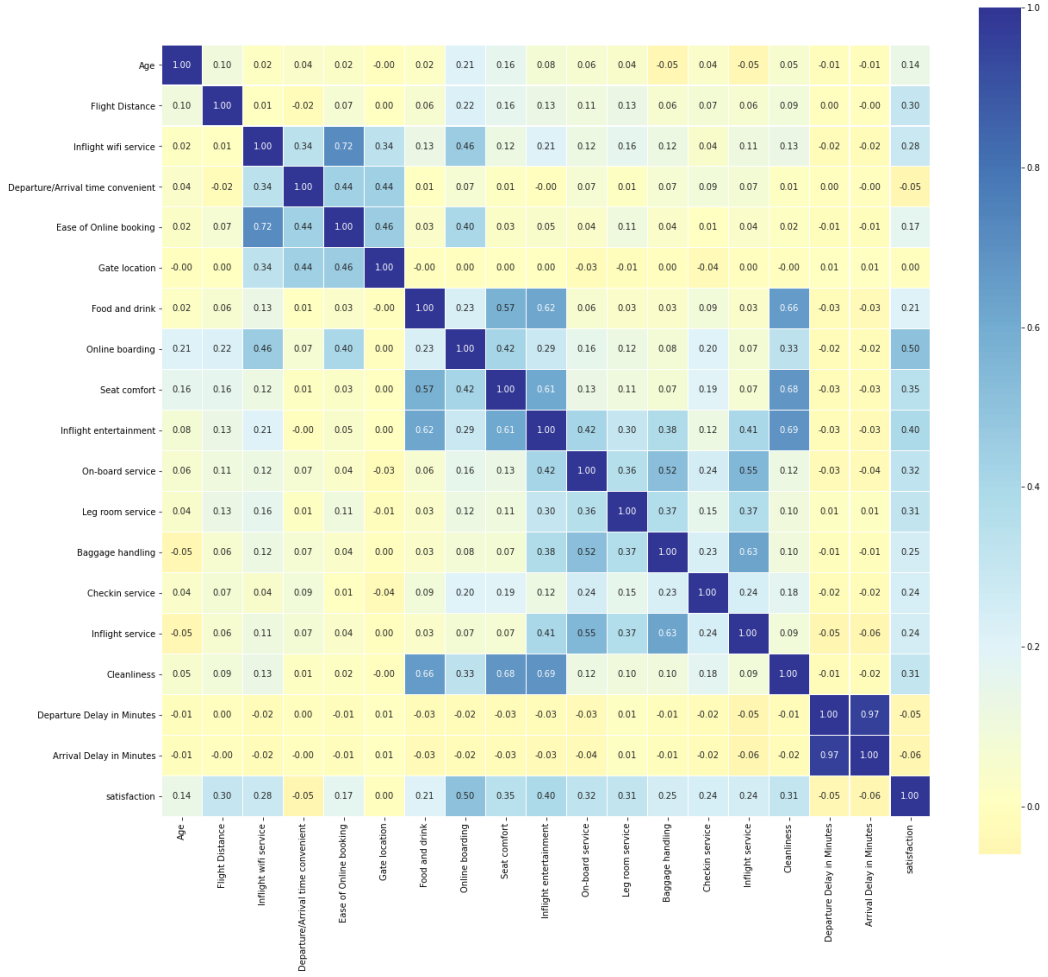
Preprocessing:

La partie preprocessing consiste à préparer aux mieux nos données afin d'optimiser les performances de l'algorithme, c'est la partie la plus importante en machine learning.

Feature selection:

Dans un premier temps, nous avons voulu voir s'il y avait une certaine corrélation entre chaque feature de nos données et la satisfaction que nous cherchons à prédire. Cela va aussi nous permettre de possiblement éliminer celles qui n'ont absolument aucun impact sur la satisfaction des clients, cela nous permettra de réduire la dimensionnalité de nos données aussi, par exemple nous pouvons déjà éliminer la feature 'id' vu que celle-ci n'aura aucun impact sur la satisfaction, ce n'est qu'un identifiant pour les clients. Cependant il est possible que nos features ne soient pas linéairement corrélées à la satisfaction, mais pour simplifier les choses nous supposons que c'est le cas. Pour cela nous avons donc calculer la matrice de corrélation de notre dataset, avec les méthodes fournies par la librairie panda, que nous allons ensuite visualiser grâce à une heatmap.

Heatmap présentant la corrélation des features dans le dataset :



On peut voir qu'il n'y a pas de corrélation entre la satisfaction des clients et les features : "Departure/Arrival time convenient", "Gate location", "Departure Delay in Minutes", "Arrival Delay in Minutes". Nous pouvons donc les enlever.

Binarisation :

Après avoir effectué une feature selection, certaines de nos features ont besoin d'être encodées afin de pouvoir être exploitées par la suite. Notamment, la valeur de satisfaction que nous voulons prédire à la fin et d'autres features tel que le type de voyage ou le type de client. Ces features sont encodées binairement représentées par un “1” ou “0” (voire même “2” dans certains cas).
Cette transformation des données est possible par le biais de la bibliothèque sklearn qui fournit la classe LabelEncoder permettant d’encoder les données.

Une fois nos données nettoyées, nous les avons séparées en 2 ensembles: l'entraînement et la validation pour pouvoir ensuite entraîner le modèle avec (test set déjà défini dans les fichiers .cvs).

Entraînement des modèles:

Choix du modèle :

La tâche de notre modèle sera d'effectuer la classification de nos données, nous pouvons déjà éliminer tous les modèles de régression non logistiques. Ensuite, vu que nous avons décidé de ne pas normaliser nos données, étape de preprocessing que certains modèles ont besoin, et que nous avons une quantité de données assez élevée (~103 000), il est hors de question que nous utilisions les modèles qui retiennent en mémoire la totalité du dataset (K-neighbors classifier). Enfin vu que nos données ont une dimensionnalité assez importante et que nous ne voulons pas perdre de précision, nous avons opté pour les modèles qui fonctionnent avec des arbres de décisions notamment DecisionTreeClassifier et RandomForestClassifier.

Decision Tree :

Le premier modèle que nous avons utilisé est le DecisionTreeClassifier de la bibliothèque sklearn.

Dans un premier temps, ce modèle a été entraîné en définissant ses hyperparamètres aléatoirement, ce qui nous a donné un score de 0.91 sur le validation set.

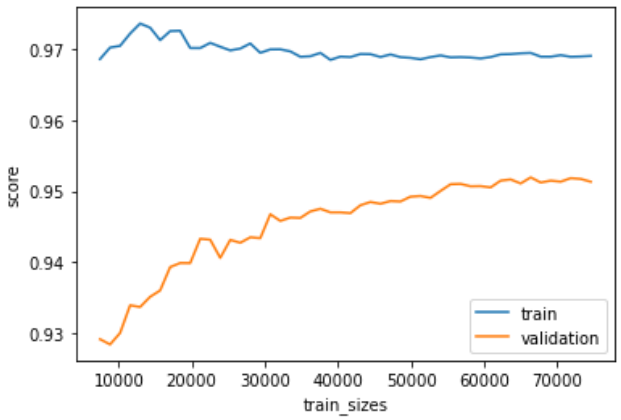
❖ Optimisation du Decision Tree :

Par la suite, nous avons cherché à définir les meilleures hyperparamètres possible afin de maximiser le score obtenu sur ce modèle par le biais de la classe GridSearchCV de la bibliothèque sklearn.

En définissant une range d'hyperparamètres à tester pour le modèle, notamment les hyperparamètres : criterion, max_depth, et min_sample_leaf, nous avons obtenues les valeurs optimales suivantes : 16 pour le max_depth et 4 pour le min_sample_leaf. Ce qui nous permet d'obtenir un meilleur score de 0.95

❖ Visualisation :

Graphique présentant le score en fonction de la taille des données pour le train set :



On remarque que la courbe du score sur le validation set a tendance à stagner après ~80 000 données d'entraînement mais augmente en général, peut-être qu'avec un peu plus de données, notre modèle pourrait obtenir un meilleur score.

Random Forest :

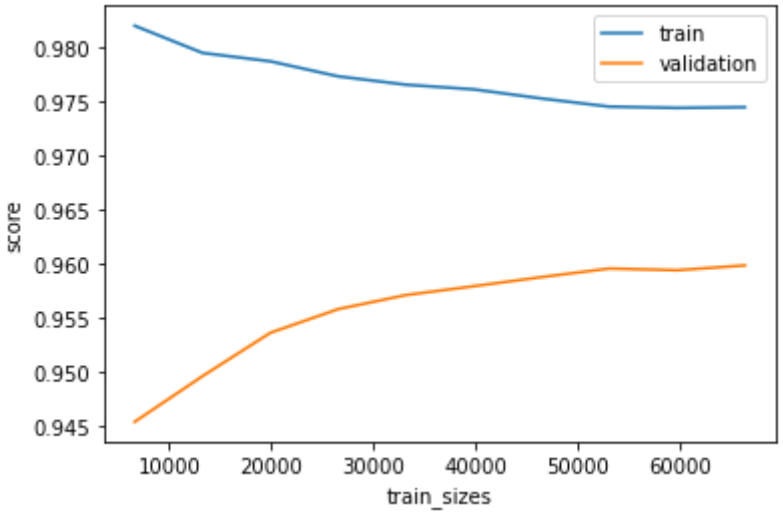
Le deuxième modèle utilisé est le RandomForestClassifier. Ce dernier est un modèle qui est composé de plusieurs Decision Tree, ce qui facilite l'optimisation plus tard. Nous avons donc initialisé notre modèle avec les hyperparamètres optimals de notre modèle précédent. Le seul hyperparamètre que nous n'avons pas touché est 'n_estimators' qui a une valeur à 100 par défaut, ce dernier permet de définir combien d'arbres il y a dans notre forêt. Ce modèle aura naturellement un meilleur score que le DecisionTreeClassifier de par son fonctionnement mais il est difficile de visualiser le procédé de classification, de plus, ce dernier aura un plus gros temps de calcul.

- **Optimisation :**

Comme précédemment, l'optimisation du modèle se fait par le biais d'un GridSearchCV mais cette fois ci que sur l'hyperparamètre 'n_estimators'.

- **Visualisation :**

Graphique présentant le score en fonction de la taille des données pour le train set et la validation:

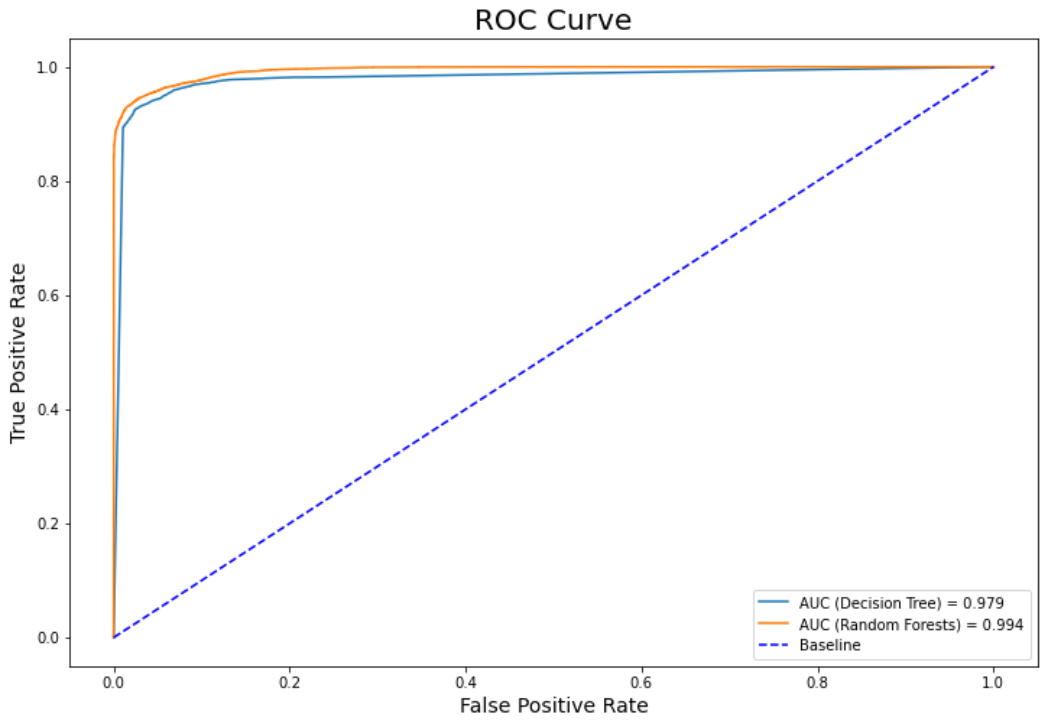


De même que pour le Decision Tree, on remarque que la courbe du score sur le validation set a tendance à augmenter cependant cette dernière stagne après ~60 000 données d'entraînement.

ROC et AUC pour les deux modèles:

Une courbe ROC représente le taux de vrais positifs et de faux positifs pour différents seuils de décision, c'est une autre manière d'évaluer la performance d'un modèle et est très utile lorsqu'il s'agit de comparer plusieurs modèles entre eux.
L'aire sous la courbe ROC, appelée AUC, permet d'évaluer les performances des modèles sur tous les seuils de classification en leur attribuant un score entre 0.0 et 1.0. Ces performances sont évaluées selon la qualité de prédiction des modèles, plus simplement, on peut interpréter l'AUC comme la probabilité que le modèle classe correctement une donnée.

Graphique présentant les courbes ROC et leur AUC associée après entrainement de nos modèles:



D'après le graphique que nous avons obtenu, le modèle Random Forest, qui a un AUC à 0.99 est plus performant que le Decision Tree avec un AUC qui vaut 0.98. C'est un résultat prévisible dans la mesure où un Random Forest est composé de plusieurs Decision Tree ce qui améliore la qualité de la prédiction.