



Participez à une compétition Kaggle !

*Spaceship Titanic*

*Predict which passengers are transported to an alternate dimension*

Soutenance du projet n°8 : Parcours « Ingénieur Machine Learning »

Luke Duthoit

## Plan de la présentation :

- Présentation de la problématique, de son interprétation et des déductions effectuées quant aux pistes de recherche possibles.
- Présentation du cleaning effectué, du feature engineering et de l'exploration.
- Présentation des différents modèles testés ainsi que des améliorations effectuées.
- Présentation du modèle final sélectionné ainsi que des performances et améliorations effectuées.

# La problématique

- *Dataset* :  
Données fictives issues de la compétition Kaggle « *Spaceship Titanic* », inspirées du jeu de données des passagers du *Titanic*, séparées en *train/test set*.
- Paramètres :  
Renseignements multiples au sujet des passagers d'un vaisseau spatial (*identifiants et features*).  
Booléen d'état de transport vers un dimension parallèle (*target*).
- Objectif :  
Entraîner un algorithme à prédire si le passager a été transporté ou pas.
  - Maximiser le score de précision.
  - Prédire l'état du booléen pour les passagers du *test set*.

# Interprétation

- Prédire la valeur d'un booléen.  
⇒ PB de classification binaire.
- Booléen seulement connu sur le *train set*.  
⇒ Précision impossible à calculer sur le jeu de test.  
⇒ Validation croisée pour obtenir une prédiction réaliste (éviter sur-apprentissage).  
⇒ Comparer la prédiction sur test set avec les proportions Transported sur train set.
- Une seule mesurable demandée : précision (*accuracy*) de la prédiction.  
⇒ Un peu réducteur...  
⇒ {précision, rappel, f1} serait + pertinent...  
⇒ Par soucis de temps et d'efficacité, on s'est tenu à la précision.

# Pistes de recherches

- Notebook d'inspiration : «  Spaceship Titanic -  EDA + 27 different models  »<sup>1</sup>
  - + : EDA assez complète, stratégies d'encodages et de valeurs manquantes simples, revue exhaustive d'algorithme de classification.
  - - : Pas optimisation des hyper-paramètres, pas de création de *features*, EDA avec recherche de corrélations linéaires entre paramètres.
- Discussions relatives à la compétition :
  - Sur le *features engineering* (quels paramètres sont à décomposer).<sup>2</sup>
  - Sur les types de corrélations entre *features* à explorer.<sup>3</sup>
  - Sur les relations entre *features* (EDA multi-variée).<sup>4</sup>
- Recherches d'informations sur le Titanic de 1912 (en particulier : positions des ponts dans le navire + lien entre pont et classes sociales).

1 : <https://www.kaggle.com/code/odins0n/spaceship-titanic-eda-27-different-models>

2 : <https://www.kaggle.com/competitions/spaceship-titanic/discussion/30969>

3 : <https://www.kaggle.com/competitions/spaceship-titanic/discussion/310574>

4 : <https://www.kaggle.com/competitions/spaceship-titanic/discussion/309513>

## Déductions

Pour apporter un peu d'originalité lors de mon travail :

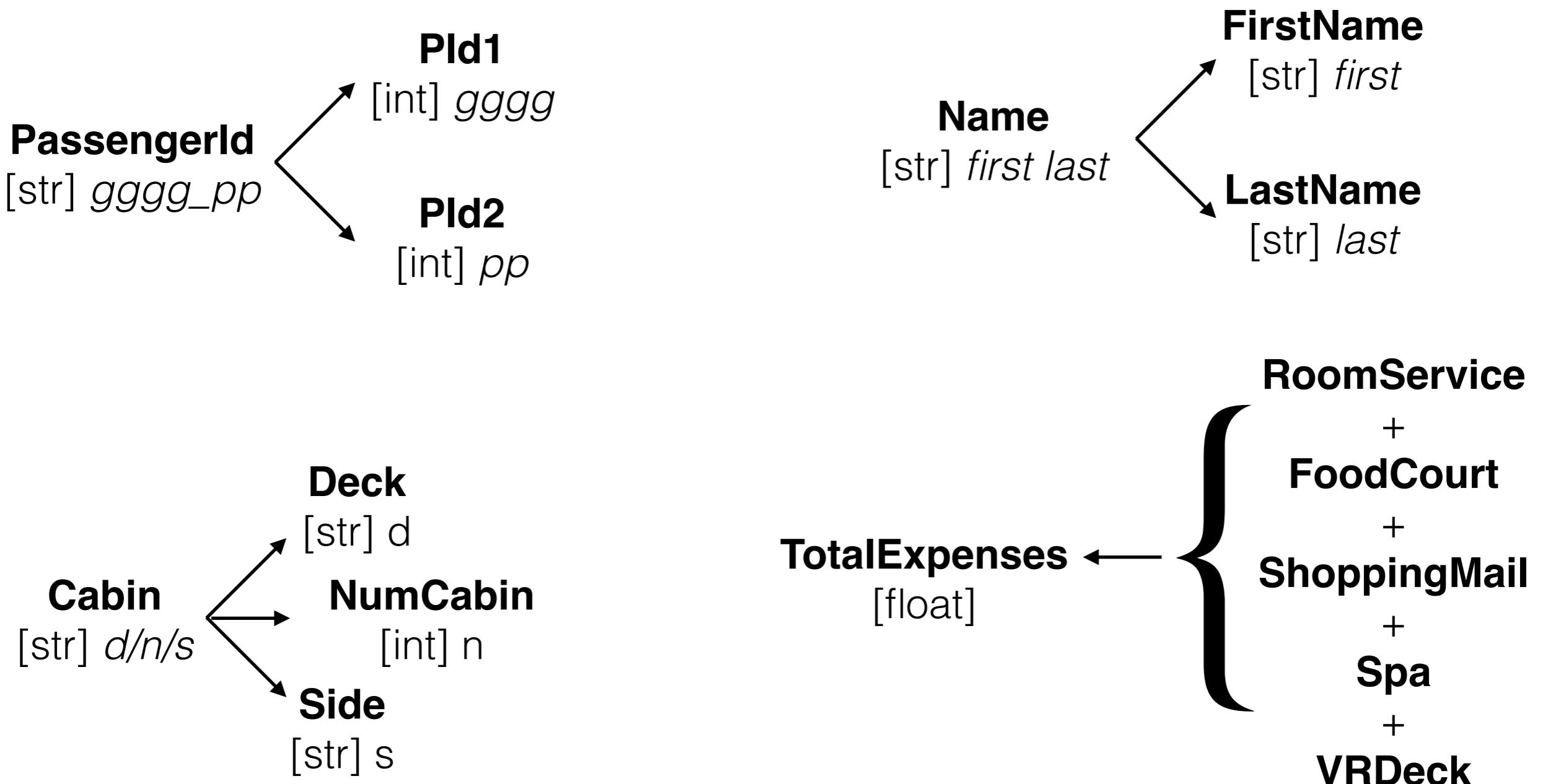
- Créer de nouvelles features + pertinentes.
- Calculer des coefficients de corrélation non linéaire.
- Remplir les valeurs manquantes de façons plus complexe.
- Tester moins d'algorithmes, mais en faisant l'optimisation des hyper-paramètres.

## Plan de la présentation :

- Présentation de la problématique, de son interprétation et des déductions effectuées quant aux pistes de recherche possibles.
- Présentation du *cleaning* effectué, du *features engineering* et de l'exploration.
- Présentation des différents modèles testés ainsi que des améliorations effectuées.
- Présentation du modèle final sélectionné ainsi que des performances et améliorations effectuées

# Feature engineering initial

D'emblée on crée de nouveaux paramètre en remplacement des originaux :



# 1<sup>er</sup> tri entre identifiants et *features* potentielles

- Concaténation des *train* et *test sets* pour faciliter le travail ⇒ création du booléen *TrainOrTest* pour retrouver appartenance à l'un des deux jeux.

- Sont exclus des *features* :

- *PId1* (cardinalité trop hautes)
- *First/LastName* (cardinalité haute + str)

- *TotalExpenses* conservé (malgré forte cardinalité) car complément des 5 autres paramètres de dépenses.

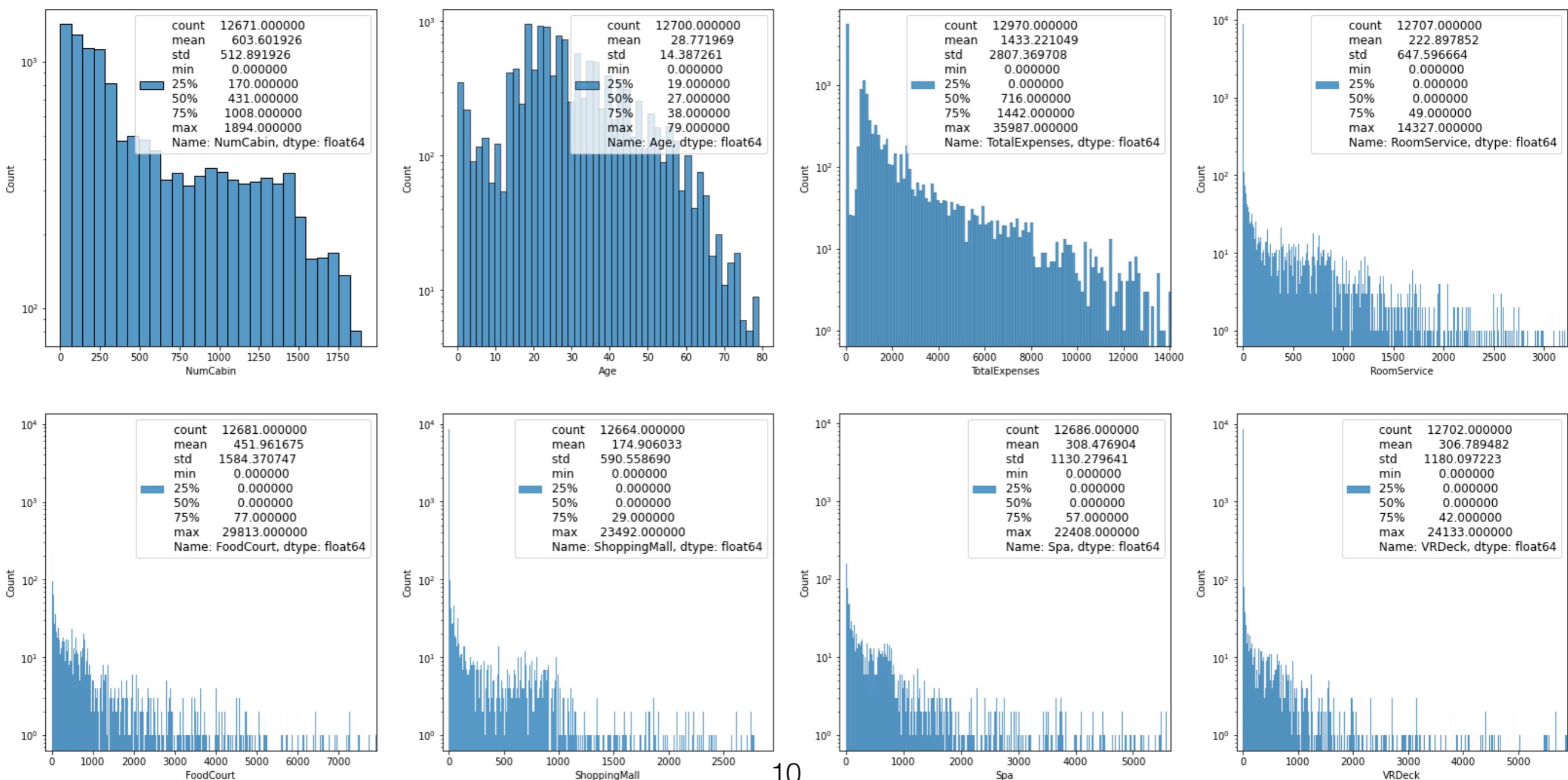
- *PId2* (int) envoyé avec les *features* qualitatives car faible cardinalité.

- *Features* issues de Cabin séparés entre *features* qualitatives (*Deck*, *Side*) et quantitatives (*NumCabin*).

| Column               | Number of unique values |
|----------------------|-------------------------|
| <i>PId1</i>          | 9280                    |
| <i>PId2</i>          | 8                       |
| <i>HomePlanet</i>    | 3                       |
| <i>CryoSleep</i>     | 2                       |
| <i>Deck</i>          | 8                       |
| <i>NumCabin</i>      | 1894                    |
| <i>Side</i>          | 2                       |
| <i>Destination</i>   | 3                       |
| <i>Age</i>           | 80                      |
| <i>VIP</i>           | 2                       |
| <i>RoomService</i>   | 1578                    |
| <i>FoodCourt</i>     | 1953                    |
| <i>ShoppingMall</i>  | 1367                    |
| <i>Spa</i>           | 1679                    |
| <i>VRDeck</i>        | 1642                    |
| <i>TotalExpenses</i> | 2980                    |
| <i>FirstName</i>     | 2883                    |
| <i>LastName</i>      | 2406                    |
| <i>Transported</i>   | 2                       |
| <i>TrainOrTest</i>   | 2                       |

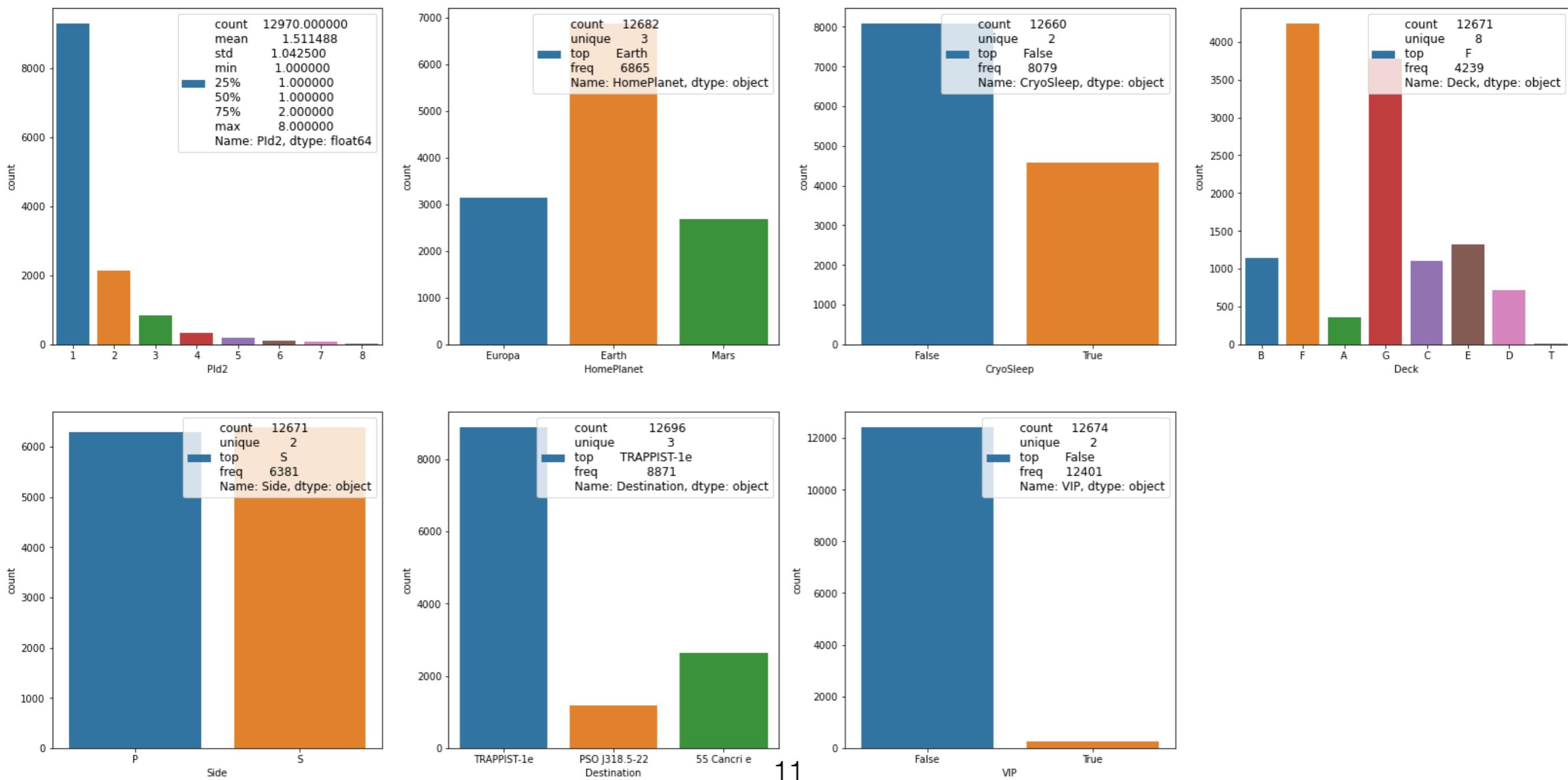
# EDA uni-variée : *features ordinaires/quantitatives*

Distributions des paramètres de dépenses problématiques :  
abondance de 0 ;  $\sigma \gg mean$  ;  $max \gg mean\dots$



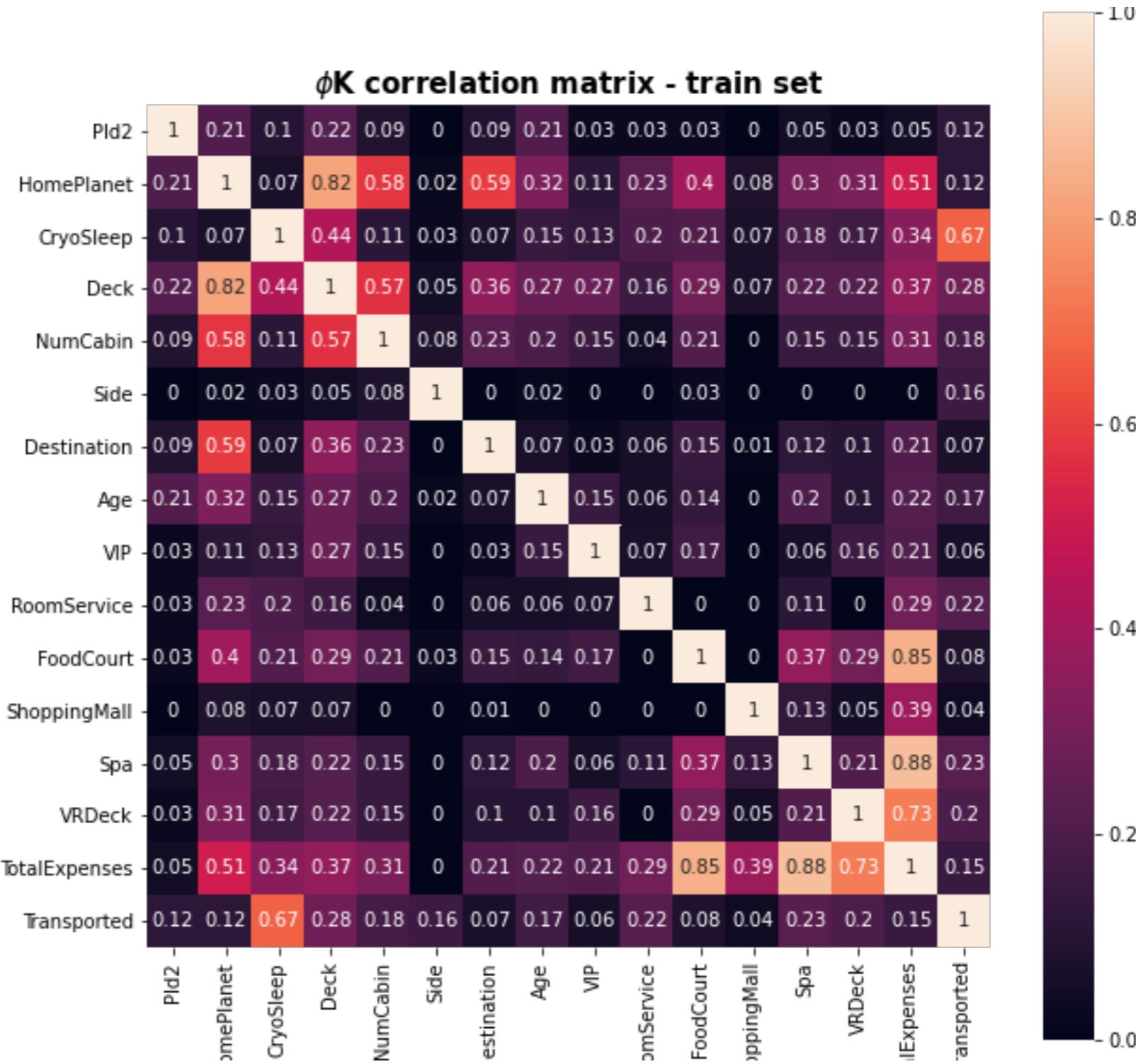
# EDA uni-variée : *features catégorielles/qualitatives*

Grande diversités de représentations des différentes catégories selon les *features*.

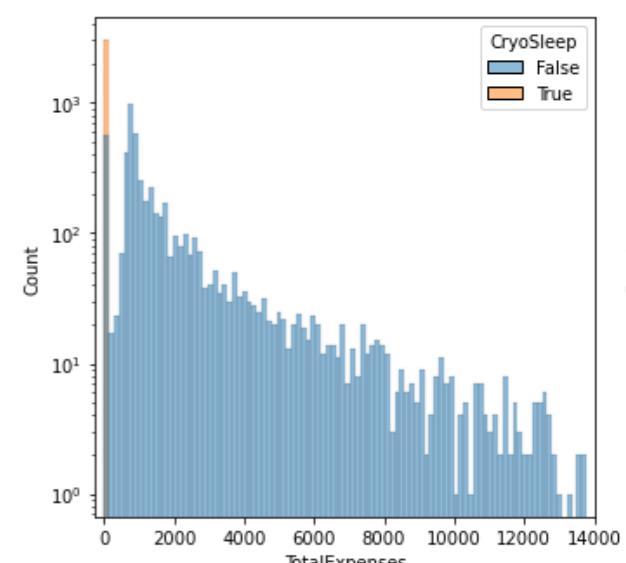
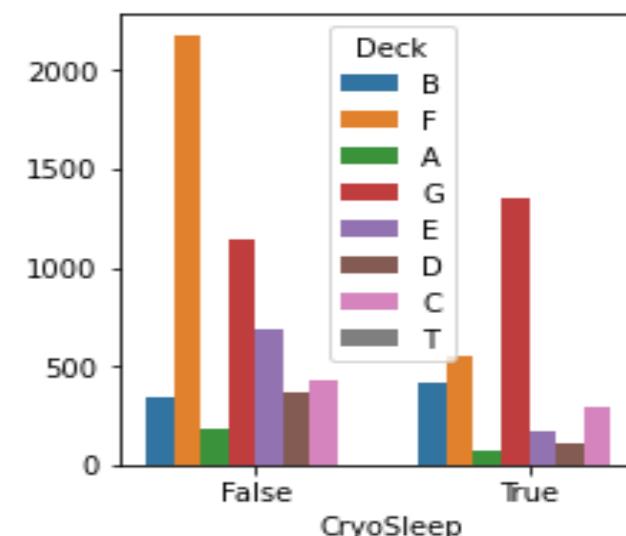
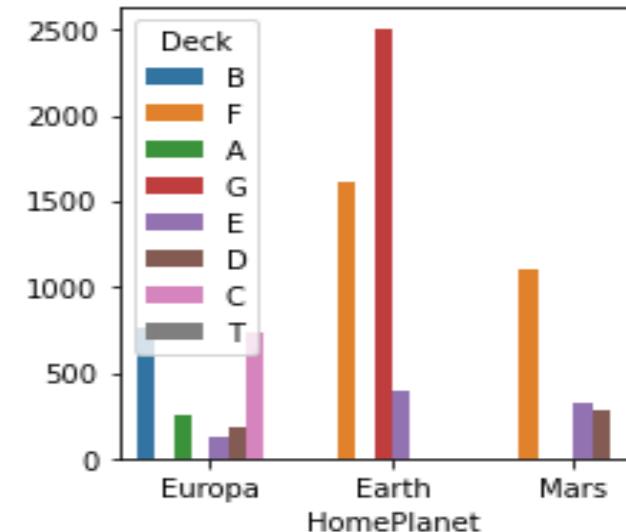
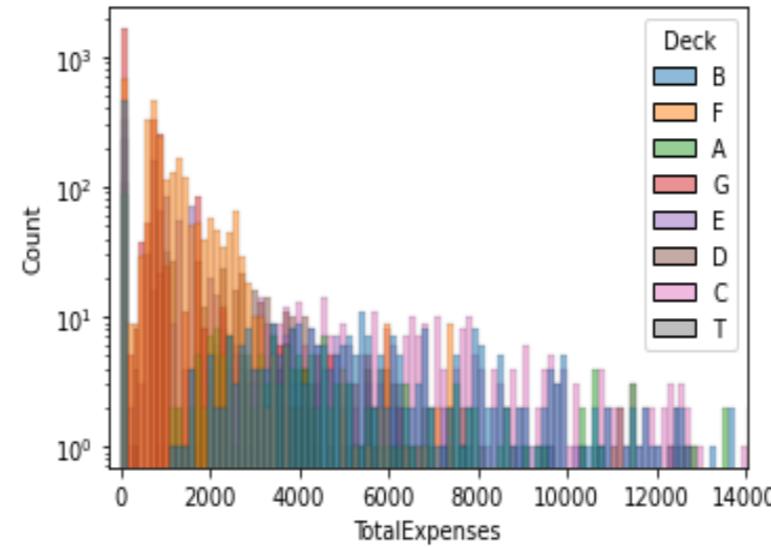
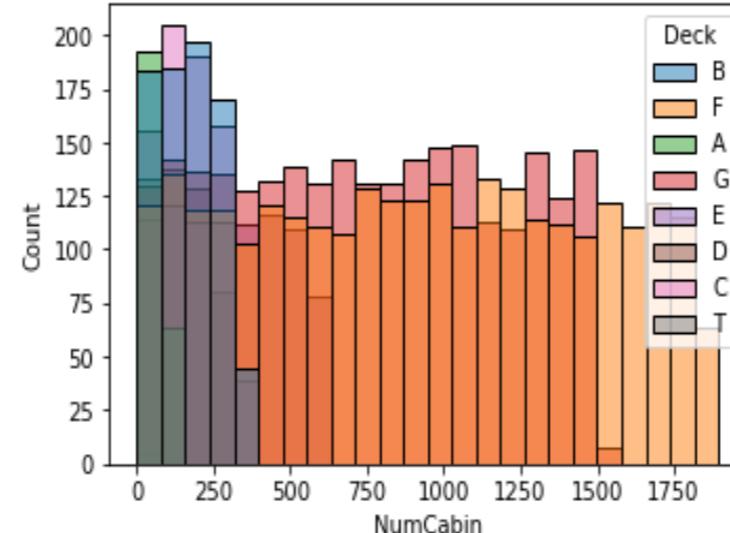
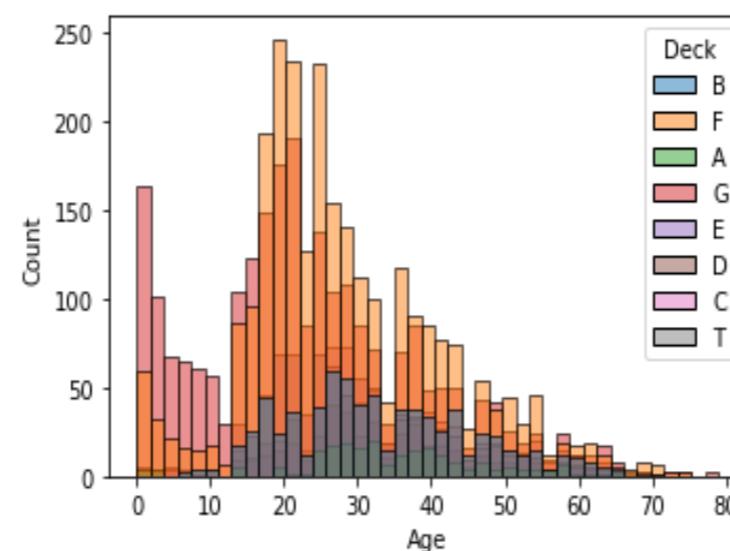
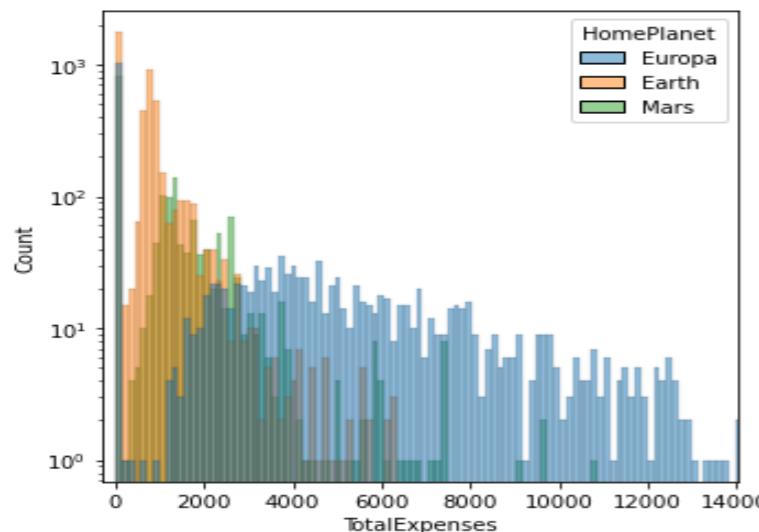
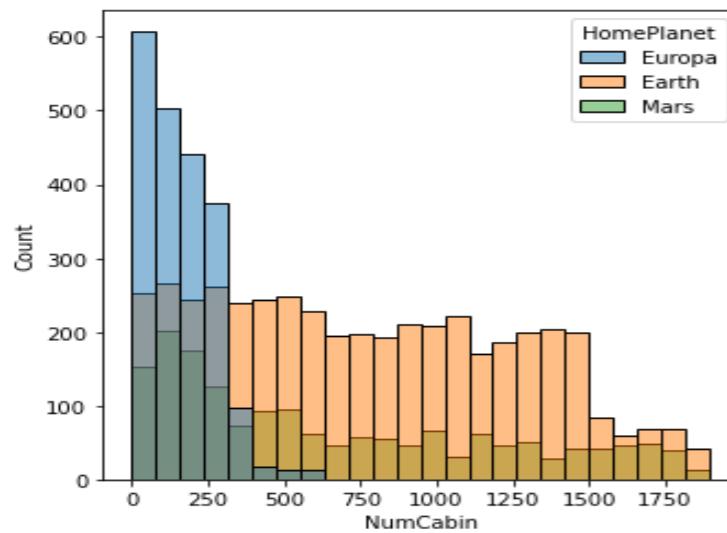
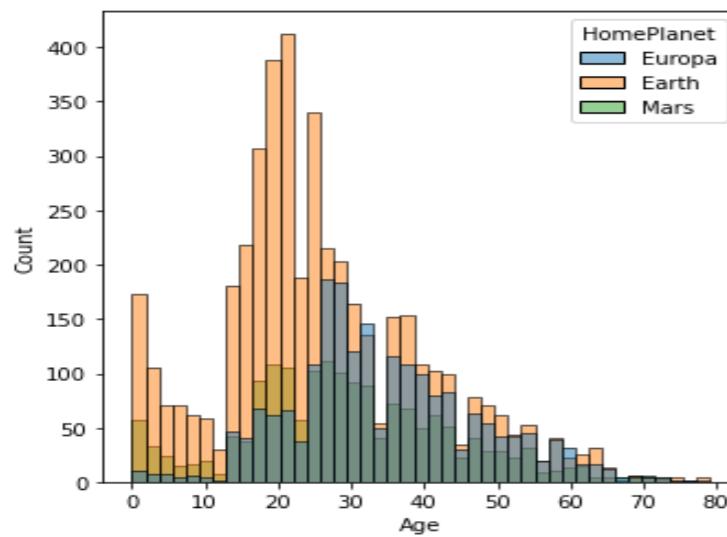


# Recours à la corrélation « φK »

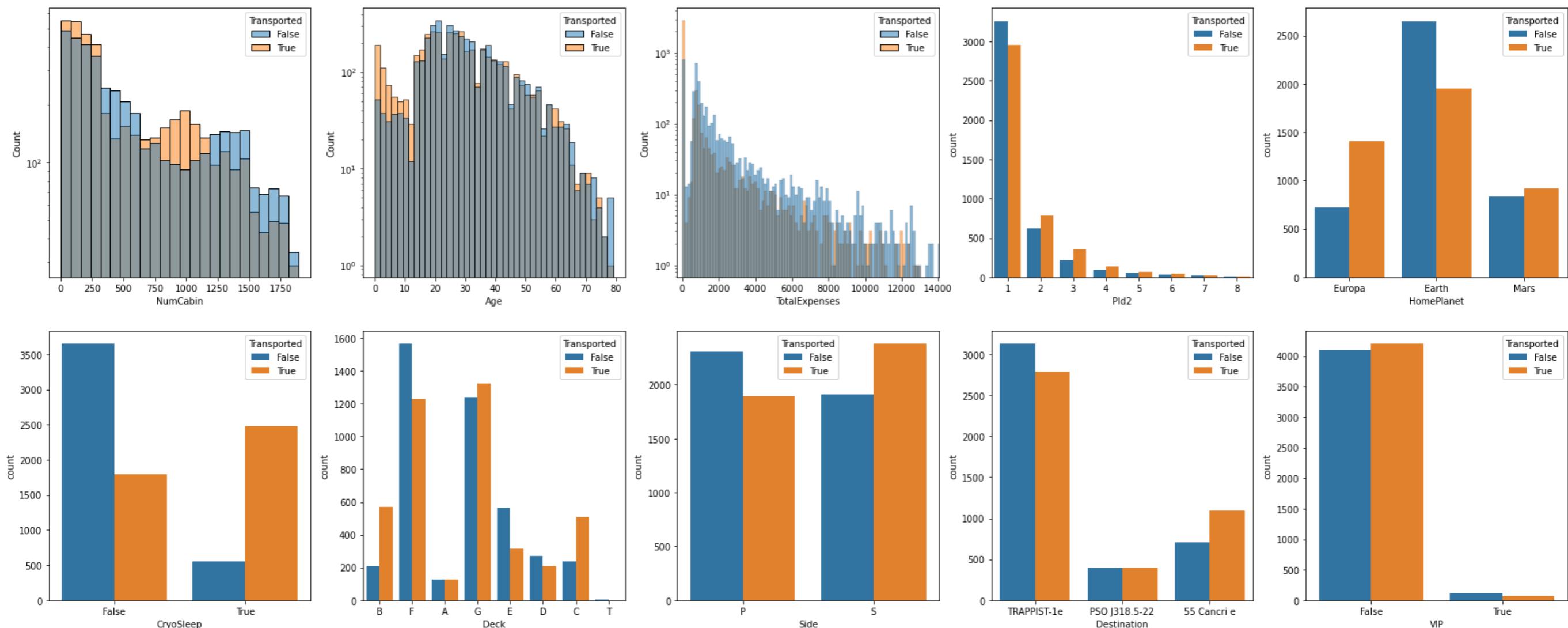
- La corrélation φK :
  - Déetecte aussi corrélations non linéaire.
  - Résistante au bruit.
  - Permet la comparaison entre features de ≠ types.
  - Coefficient en valeur absolue.
- Matrice des coeff. de corr. révèle de fortes corrélations entre :
  - certains *features* et la *target*,
  - entre certains *features* (CryoSleep, Deck, HomePlanet, NumCabin, TotalExpenses).



# EDA multi-variée : quelques exemples (1)



# EDA multi-variée : quelques exemples (2)



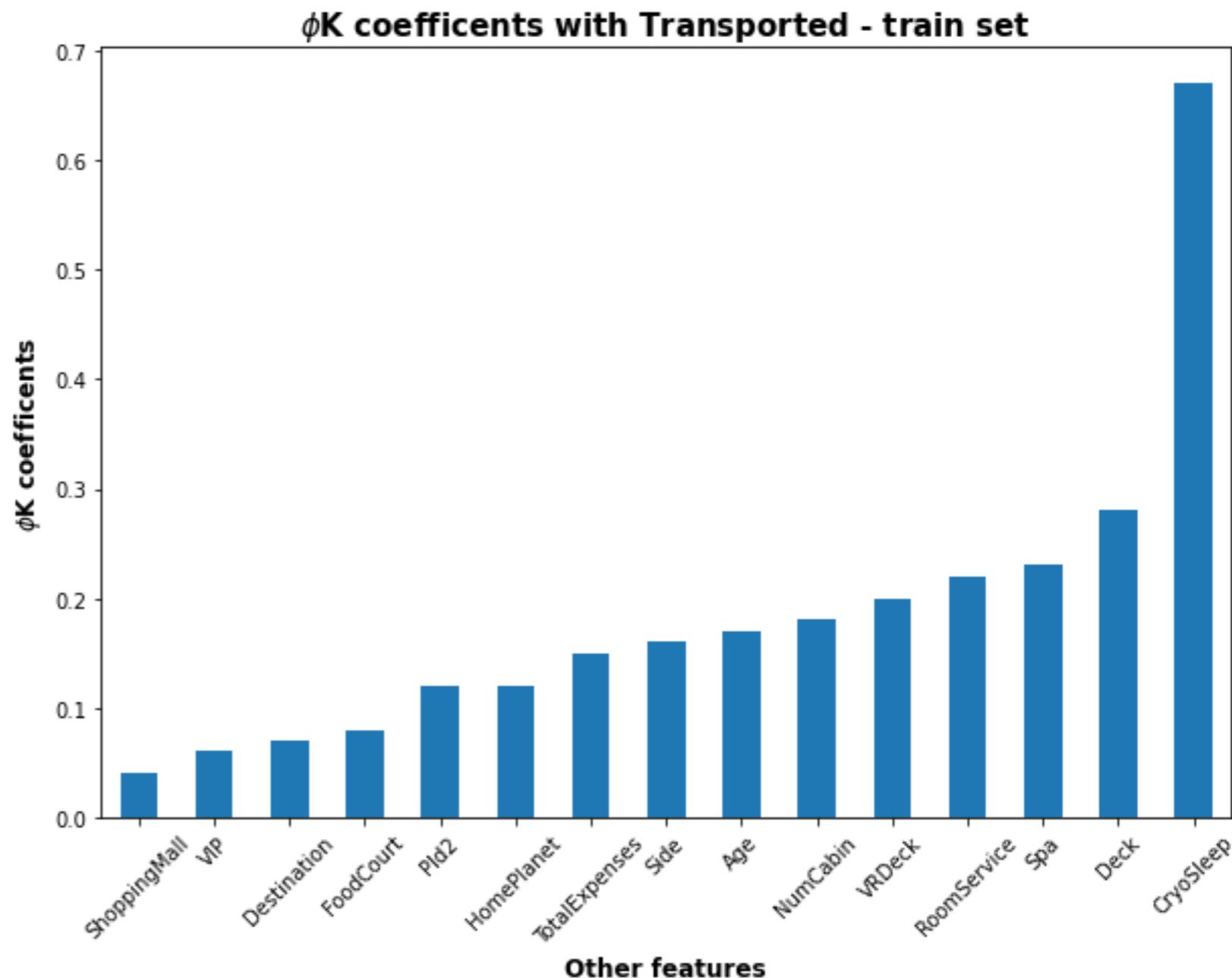
- Tous les *features* n'ont pas la même relation avec la *target* !
- Relation qui n'apparaît pas sur les graphes précédent : entre {*PdI1*, *Lastname*} et les features {*Deck*, *NumCabin*} (cf : notebook, rapport).

## **Plan de la présentation :**

- Présentation de la problématique, de son interprétation et des déductions effectuées quant aux pistes de recherche possibles.
- Présentation du cleaning effectué, du feature engineering et de l'exploration.
- Présentation des différents modèles testés ainsi que des améliorations effectuées.
- Présentation du modèle final sélectionné ainsi que des performances et améliorations effectuées

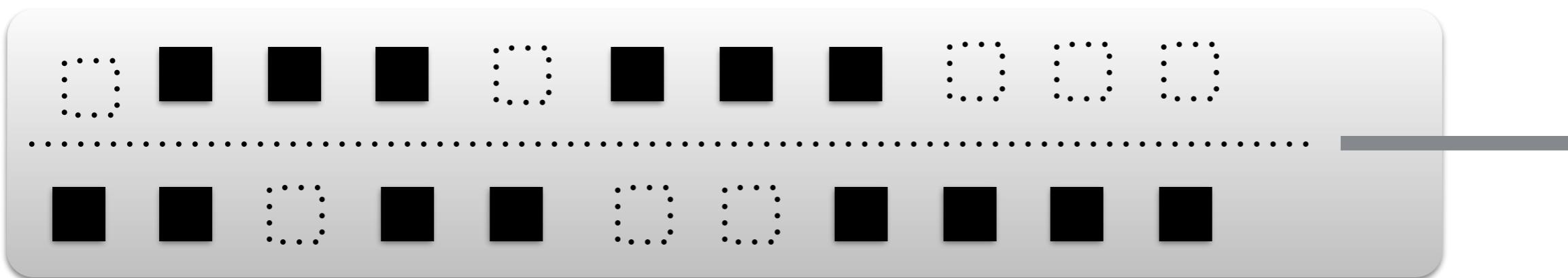
# Remplissage des données : stratégie.

- Stratégie de remplissage basée sur les relations entre features précédentes (autant que possible).
- *Features* remplies dans l'ordre décroissant de leur coeff.  $\phi K$   
⇒ Limite la propagation de l'erreur commise lors du remplissage des *features* plus « importantes ».
- Les features avec un coeff  $\leq 0.1$  sont délaissés.



## Remplissage des données : quelques modèles.

- *CryoSleep* : Rempli grâce à *TotalExpenses* (True si 0, False si >0).
- *Deck / NumCabin / Side* :
  - 1) Recherche de passagers du même groupe *PId1* ⇒ même cabine.
  - 2) Stratégie + complexe :
    - ◆ Estimation de *Deck* le plus probable grâce à *HomePlanet* et *TotalExpenses*.
    - ◆ Attribution d'une cabine « libre » sur ce pont  
(grâce au fait que  $\max(\text{NumCabin}) = f(\text{Deck})$ ).



- Age déterminé par la moyenne sur ses 10 + proches voisins (distance mesurée sur les paramètres de dépenses).

# Modèles d'encodages des données non quantitatives.

- *One Hot Encoding* : *HomePlanet*, *CryoSleep*, *Side* (car 2 ou 3 catégories)
- *Label Encoding* : *Deck*.
  - 8 catégories (risque d'explosion du nombre de *features* avec OHE)
  - Pour le Titanic de 1912, plusieurs relations ordinale entre les ponts :
    - ◆ Niveau de ponts par niveau de profondeur dans le navire.
    - ◆ Niveau de ponts fortement corrélé au niveau de richesses des passagers.

⇒ A → 0, B → 1, ..., G → 6, T → 7.

# Différents modèles de *scaling* testés

- Le *scaling* naturel [en sortie de l'encodage] met en évidence de profondes disparités entre les *features* (maxima, écart-types, etc)
- Pour réduire les écarts d'échelle au sein des features, 2 types de scaling ont été testés :
  - Passage au log+1, puis normalisation de tous les écart-types.
  - Passage par un *QuantileTransformer*.
- Les trois scalings sont testés via une régression logistique (optimisation des HP + validation croisée).

| Type de scaling :                                 | Valeurs des hyper-paramètres optimisés :       | Meilleur taux de précision (en moyenne sur 5 plis) : |
|---|--|--|
| Laissé en l'état à la sortie de l'encodage        | {C : 0.01 ; penalty : L1 ; solver : liblinear} | 0.7799   |
| Passage au log+1 et normalisation de l'écart-type | {C : 0.1 ; penalty : L1 ; solver : liblinear}  | 0.7636   |
| Passage par un <i>QuantileTransformer</i>         | {C : 0.01 ; penalty : L2 ; solver : sage}      | 0.7373   |

## Algorithmes testés

- Deux type d'algo. de classification binaire accessibles avec sklearn :
  - *AdaBoostClassifier* (très bonnes performances dans le notebook d'inspiration)
  - *MLPClassifier* (absent de ce notebook)
- Peu de type d'algo. différents pour passer + de temps sur l'optimisation des hyperparamètre et validation croisée (faites avec GridSearchCV).

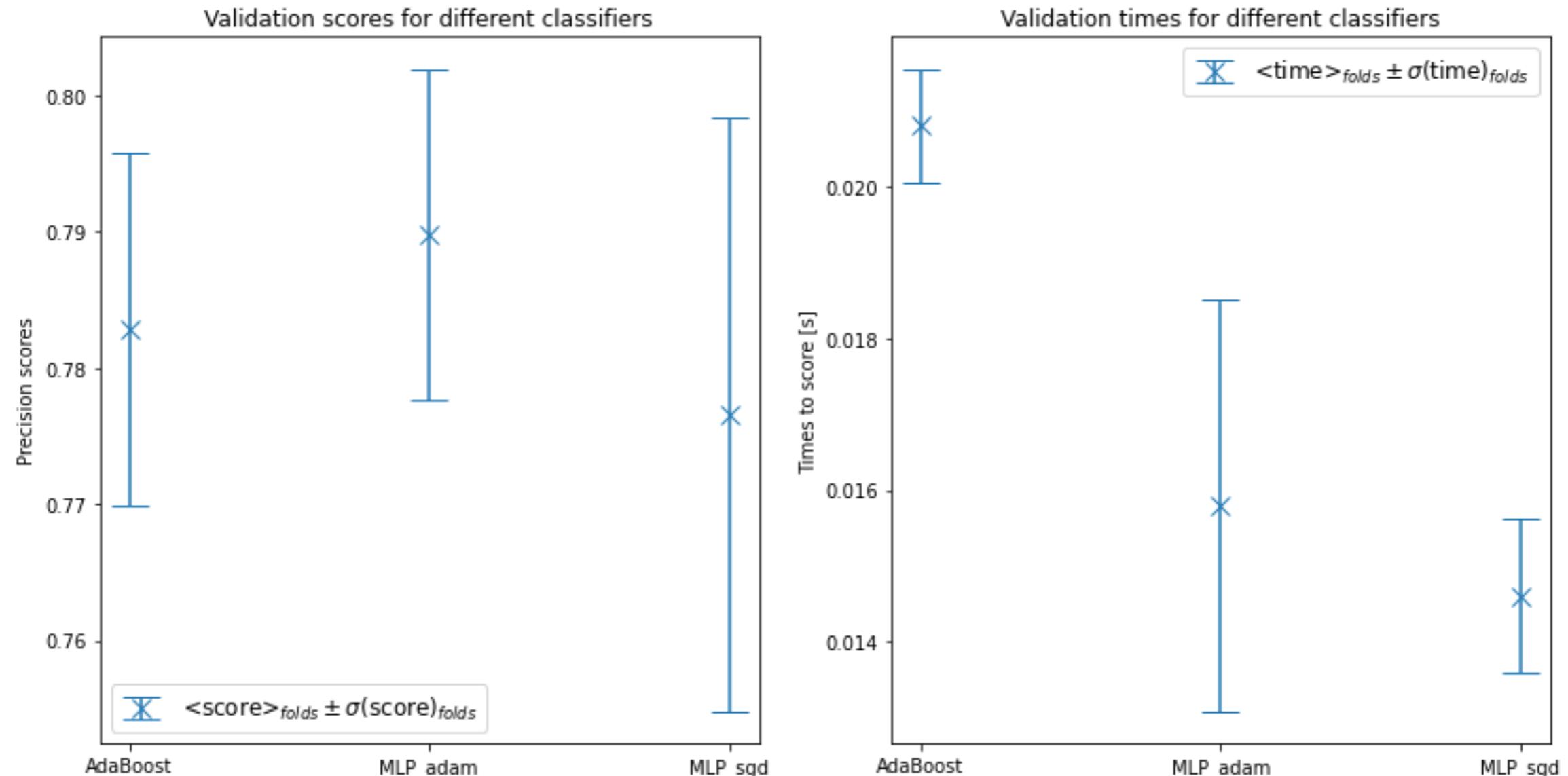
# Optimisation des hyper-paramètres et validations croisée

- On a choisi de se concentrer sur des HP dont on comprenait le mieux la portée.
- Pour *AdaBoostClasifier* :
  - Estimateur de base : arbre de décision.
  - On choisit d'optimiser le *learning rate*, et le nombre d'estimateurs.
- Pour *MLPClassifier* :
  - On utilise un *EarlyStopping* en cas de non progression de l'apprentissage.
  - On se limite à une seule couche cachée.
  - On choisit d'optimiser : le *learning rate*, la taille de la couche cachée, l'algorithme de descente de gradient (*solver*).
  - Autres HP à optimiser selon que *solver* = *Adam* ou *SGD*
- Validation croisée sur 5 plis (*stratified K-fold*)

## **Plan de la présentation :**

- Présentation de la problématique, de son interprétation et des déductions effectuées quant aux pistes de recherche possibles.
- Présentation du cleaning effectué, du feature engineering et de l'exploration.
- Présentation des différents modèles testés ainsi que des améliorations effectuées.
- Présentation du modèle final sélectionné ainsi que des performances et améliorations effectuées

# Performances des différents modèles, et sélection du meilleur



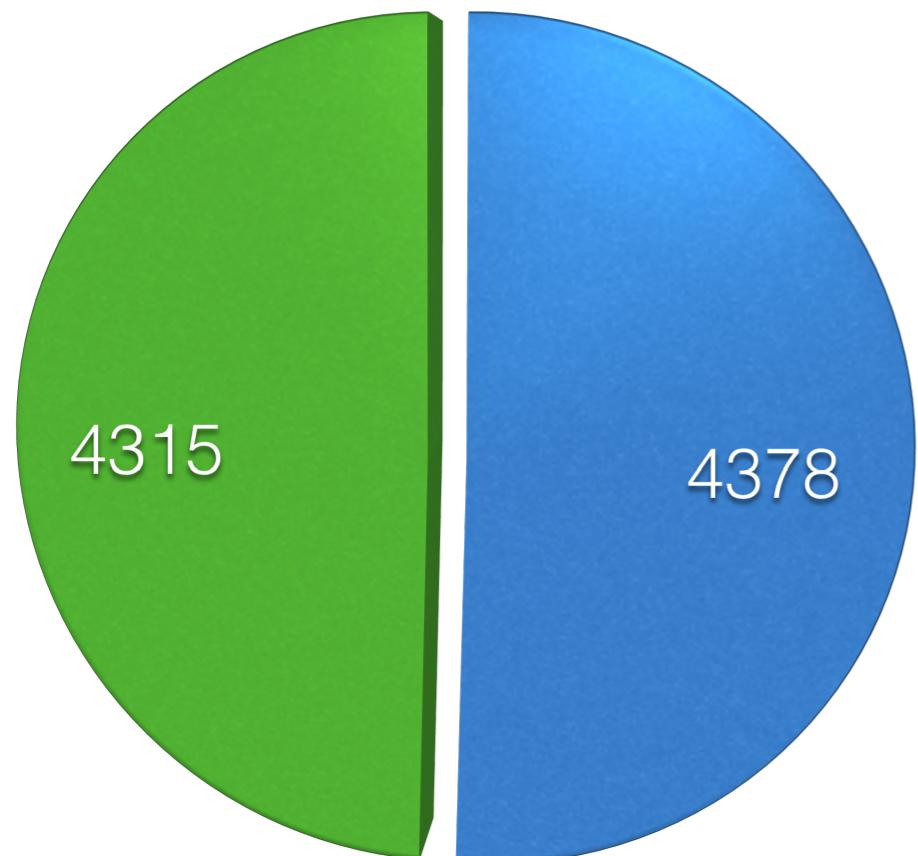
|   | mean_test_score | std_test_score | mean_fit_time [s] | std_fit_time [s] |
|---|-----------------|----------------|-------------------|------------------|
| MLPClassifier(activation='logistic', beta_1=0.0001, beta_2=0.009949874371066206,\nearly_stopping=True,\nlearning_rate='invscaling', max_iter=40,\nrandom_state=420) | 0.790524        | 0.013322       | 1.588758          | 0.172974         |

## Prédictions sur le jeu de test

True

False

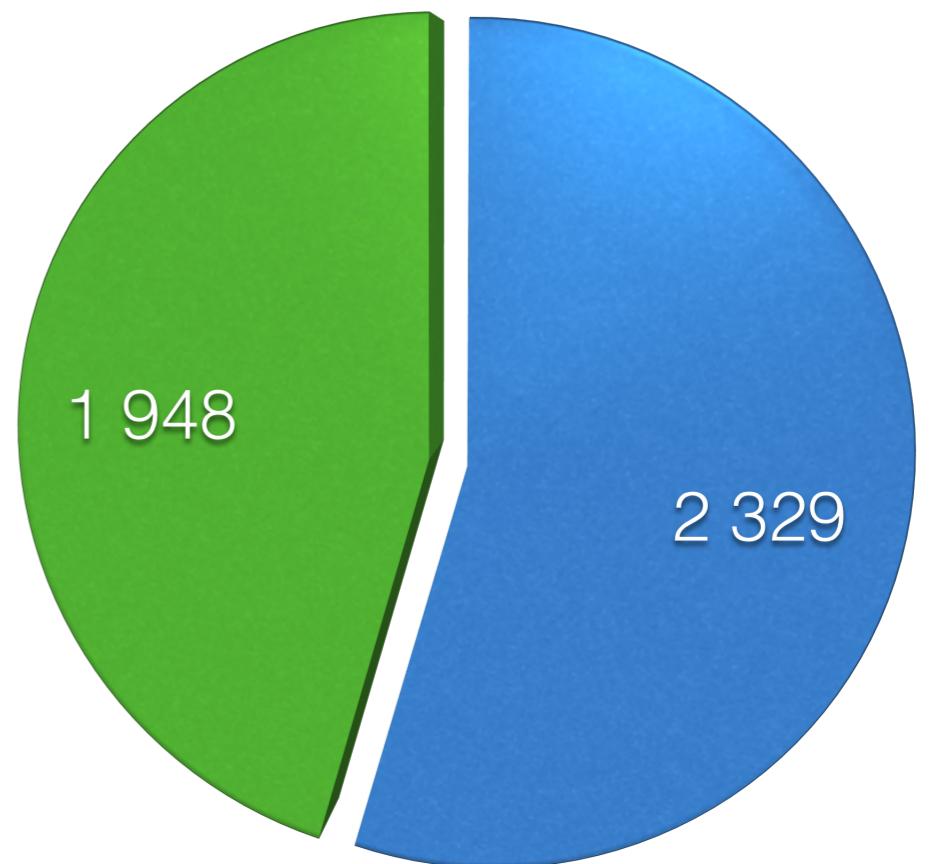
Transported pour le jeu  
d'entraînement



True

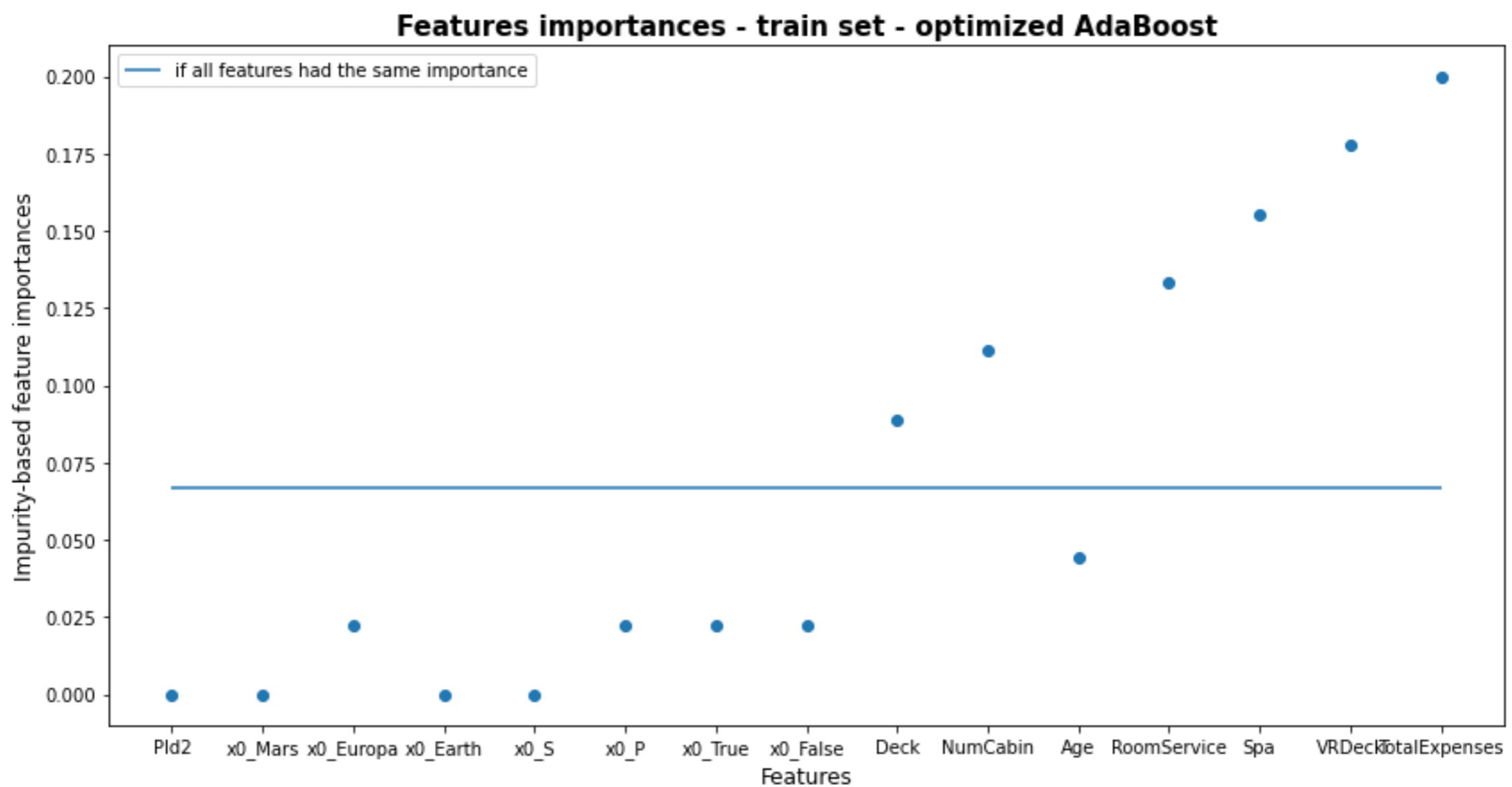
False

Prédictions de Transported  
sur le jeu de test



## *A posteriori, des features pas toujours pertinents*

- On trace l'importance des features grâce à AdaBoostClassifier.
- On voit que les features issues de l'encodage ont une importance faible, voire nulle.



# Conclusions

- On s'est inspiré d'un notebook et de sujets de discussion pour essayer de rendre notre travail un peu original.
- L'EDA a été l'occasion d'utiliser un nouveau type de coefficient de corrélation.
- Gros effort fait sur les étapes de *features engineering* :
  - Remplissage de valeurs manquantes en utilisant les relations entre features.
  - Différents encodages.
  - Différents *rescaling* mis à l'essai.
- Deux algorithmes de classifications aux HP optimisés, et avec validation croisée :
  - AdaBoostClassifier.
  - MLPClassifier, avec deux types de *solver*.
- Le meilleur modèle donne une précision acceptable (près de 79%) et ses prédictions sur *test set* se rapprochent des proportions de l'étiquette sur le *train set*.

# Perspectives

- Utiliser l'importances des *features* pour réduire encore plus le nombre de *features*.
- Ou fusionner les catégories ayant engendré les features les moins importants.
- Se concentrer sur *MLPClassifier* pour chercher à faire un réseau plus profond, avec des méthodes + complexes d'adaptation du *learning rate*.