

Projet de Machine Learning : Competition Spaceship Titanic

Valentin Hesters

Sara Firoud

Lina El haddaj

Équipe Kaggle: M1 Valentin Sara Lina

4 avril 2024

Sommaire

1	Analyse des données	2
2	Pré-traitement des données	5
3	Entrainement des modèles	7
3.1	Régression logistique	7
3.2	SVM	9
3.3	Random Forest	10
3.4	Boosting	11
4	Résultat et score sur Kaggle	12
5	Conclusion	13
6	Code	13

Nous avons un jeu de données composé de passagers du Spaceship Titanic.

Dans celui-ci, de nombreuses informations personnelles sur le passager ainsi que sur son activité à bord sont connues.

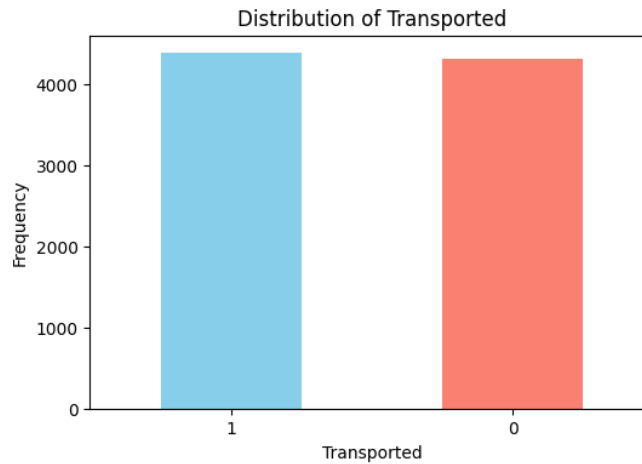
Grâce à toutes ces informations récoltées, nous allons tenter de répondre à la problématique suivante :

Comment prédire avec précision quels passagers du Spaceship Titanic ont été transportés vers une dimension alternative après la collision avec une anomalie spatio-temporelle, afin d'optimiser les opérations de sauvetage ?

1 Analyse des données

Les documents qui nous sont fournis sont:

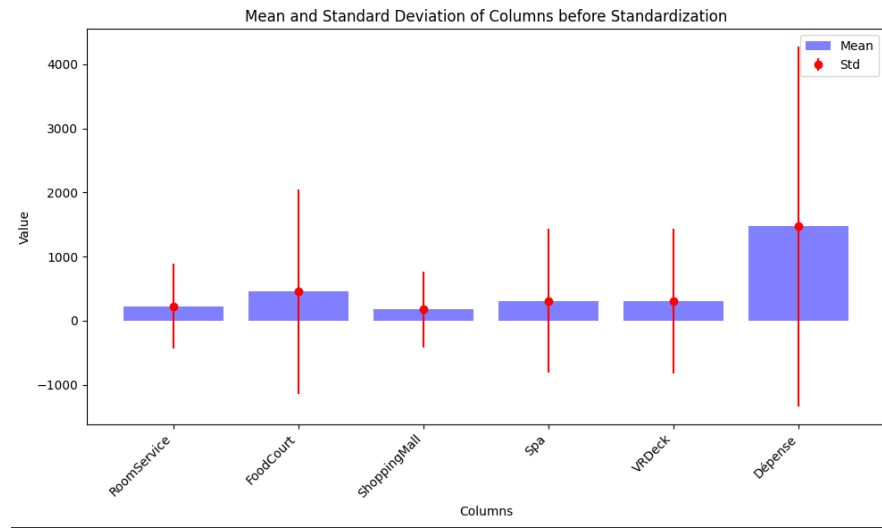
- "sample_submission.csv" : un modèle de la forme que doivent avoir tous fichiers de soumission sur kaggle pour cette compétition ;
- "train.csv" : la base d'entraînement des données sur lesquelles nous effectuerons notre travail d'analyse de données. Elle contient les valeurs de Transported pour les deux tiers des passagers évalués et nous permettra donc d'entraîner notre modèle ;
- "test.csv" : le fichier sur lequel nous devons prédire les valeurs de Transported des passagers du tiers restant. Il nous sera notamment utile pour l'obtention du score final, via la soumission sur kaggle, des modèles testés.



La distribution des valeurs dans la colonne cible Transported est équitable. La valeur de score obtenue par nos modèle aura donc du sens si elle est meilleure qu'un modèle qui prédirait cette valeur complètement aléatoirement et qui aurait donc un score de 0.50.

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

Cette description des données numériques du dataset nous apprend que les passagers sont relativement jeunes. Les variables de dépenses sont intéressantes car elles montrent une disparité financière chez les passagers : en effet, l'écart-type est très grand et suggère qu'il y a plusieurs niveaux de richesse des passagers au sein du bateau. Nous standardiserons donc ces variables lors du pré-traitement des données pour les rendre comparables et donc éviter qu'il y en ait qui soient plus importantes que d'autres dans nos modèles. Par exemple FoodCourt, qui a une moyenne et un écart-type importants, par rapport à ShoppingMall.

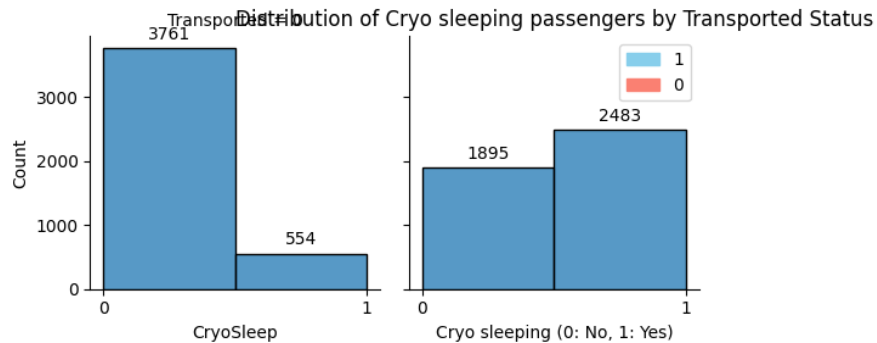


	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	VIP	Name
count	8693	8492	8476	8494	8511	8490	8493
unique	8693	3	2	6560	3	2	8473
top	0001_01	Earth	False	G/734/S	TRAPPIST-1e	False	Gollux Reedall
freq	1	4602	5439	8	5915	8291	2

On remarque que les colonnes PassengerId et Name ne contiennent que des valeurs uniques et ne sont donc pas intéressantes à garder dans la mesure où elles ne vont pas aider le modèle à trouver des schémas qui mènent à un état vrai de Transported. En revanche, PassengerId contient aussi un décompte des groupes présents dans le bateau, ce qui sera vital dans le pré-traitement des données.

La variable VIP confirme quant à elle l'hypothèse des différentes classes de richesses chez les passagers.

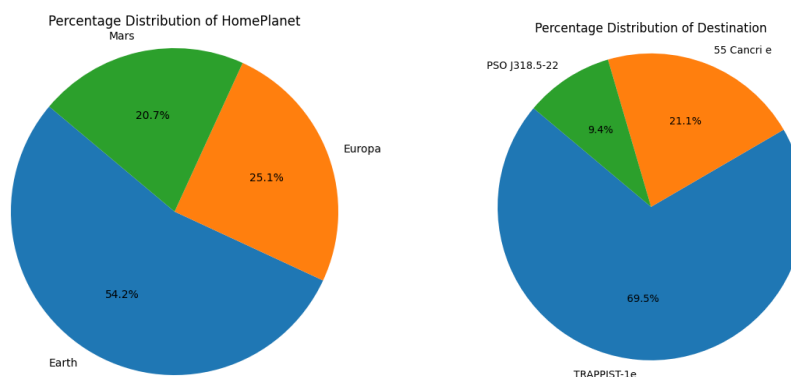
Intuitivement, la variable booléenne CryoSleep, qui indique si un passager est cryogénisé pour la durée du voyage et est donc confiné dans sa cabine, semble être intéressante pour prédire la valeur de Transported. Ceci est d'ailleurs confirmé par le graphique suivant:



On remarque que chez les personnes qui ne sont pas transportées, le rapport entre le nombre

de passagers cryogénisés et non cryogénisés est faible. Ce rapport est inversé chez les personnes transportées, ce qui suggère donc que cette variable est importante pour les prédictions futures. De leur côté, Homeplanete et Destination sont liées.

	HomePlanet	Transported		Destination	Transported
1	Europa	0.658846	0	55 Cancr e	0.610000
2	Mars	0.523024	1	PSO J318.5-22	0.503769
0	Earth	0.423946	2	TRAPPIST-1e	0.471175



Les valeurs de HomePlanet et Destination sont intéressante et doivent être gardées car elles montrent une disparité dans les valeurs de Transported, par exemple une personne venant de Europa a 66% de chances de finir transportée contre 42% si elle vient de Earth. De même, une personne allant à 55 Cancr e a 61% de chance de l'être, contre 47% si elle se rend sur TRAPPIST-1e.

2 Pré-traitement des données

	Missing Values	Percentage (%)
CryoSleep	217	2.496261
ShoppingMall	208	2.392730
VIP	203	2.335212
HomePlanet	201	2.312205
Name	200	2.300702
Cabin	199	2.289198
VRDeck	188	2.162660
FoodCourt	183	2.105142
Spa	183	2.105142
Destination	182	2.093639
RoomService	181	2.082135
Age	179	2.059128

Ce dataset contient environ 2% de valeurs manquantes par colonne et il est indispensable de remplir de façon adéquate les cases vides du dataset. Dans un premier temps, nous avons décidé de les remplir de manière simple, c'est-à-dire de mettre la valeur moyenne pour les valeurs numériques et la valeur la plus fréquente de la colonne pour les variables catégorielles. Afin d'optimiser les scores de nos modèles, nous avons changé notre manière de faire en remplissant ces cases de façon plus adéquate.

La variable PassengerId nous servira ici à regrouper les passagers par groupes:

```
def detecter_groupe(data_train):  
    data_train['Group'] = data_train['PassengerId'].str[:4]  
    occurrences_groupes = data_train['Group'].value_counts()  
    data_train['DansGroupe'] = data_train['Group'].map(occurrences_groupes) > 1
```

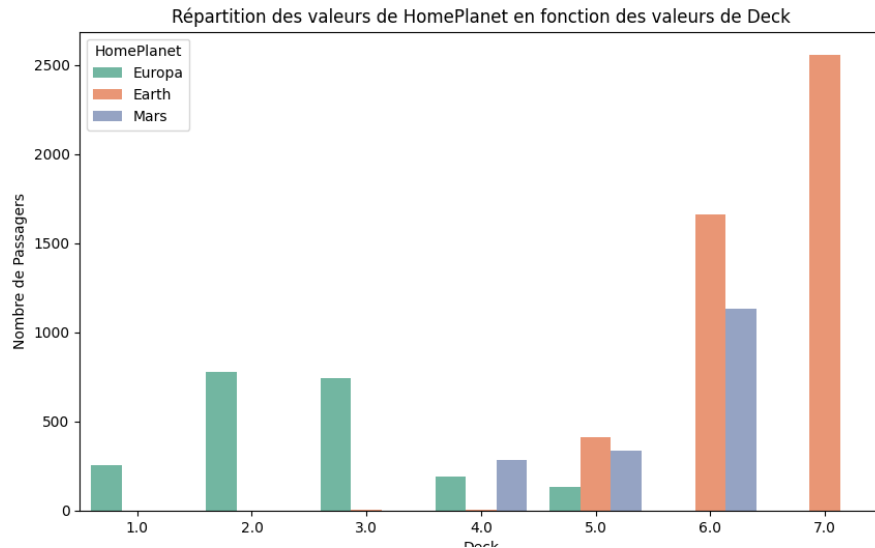
Ainsi, pour les variables HomePlanet, Destination, VIP et CryoSleep, nous remplissons les cases vides par la valeur la plus fréquente du groupe auquel ils appartiennent. S'ils n'appartiennent à aucun groupe, alors la case sera remplie par la valeur la plus fréquente du dataset.

Concernant les variables de dépenses, nous les remplissons d'une autre manière: les personnes jeunes, c'est-à-dire les moins de 14ans et/ou les personnes ayant été cryogénisées (CryoSleep = 1) auront des valeurs de dépense automatiquement mises à 0:

	Variable	Pourcentage_0	Pourcentage_non_0
0	RoomService	90.967742	9.032258
1	FoodCourt	92.165899	7.834101
2	ShoppingMall	91.059908	8.940092
3	Spa	91.981567	8.018433
4	VRDeck	91.981567	8.018433

Pour les autres personnes, nous avons choisi un remplissage en fonction de la moyenne des valeurs de la colonne correspondante car les dépenses d'un passager ne semblent pas être liées au groupe auquel il appartient.

Nous avons extrait les détails de la colonne cabine, à savoir le pont (deck), le numéro (num) et le côté (side). Pour encoder les variables HomePlanet, Destination, Side et Deck nous utilisons le One-Hot Encoding à l'aide de la méthode *pd.get_dummies()*. Ainsi ces colonnes peuvent être interprétées par nos modèles, avec la même valeur d'importance pour toutes, ce qui n'aurait pas été le cas si nous avions choisi de simplement affecter un chiffre par valeur dans ces colonnes.



On constate qu'en fonction de leur planète de départ, les passages ont des valeurs de Deck réparties différemment. Nous prenons cette différence en compte lors du remplissage des valeurs manquantes de Deck : par exemple, un passager partant de Earth aura une valeur de Deck vraie pour Deck_5.0, Deck_6.0 ou Deck_7.0. Enfin, la colonne Num est supprimée et le dataset d'entraînement final est prêt à être entraîné.

3 Entraînement des modèles

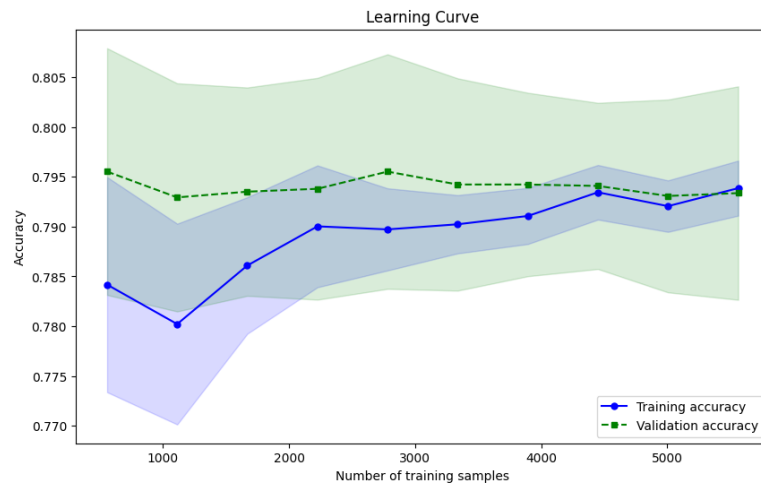
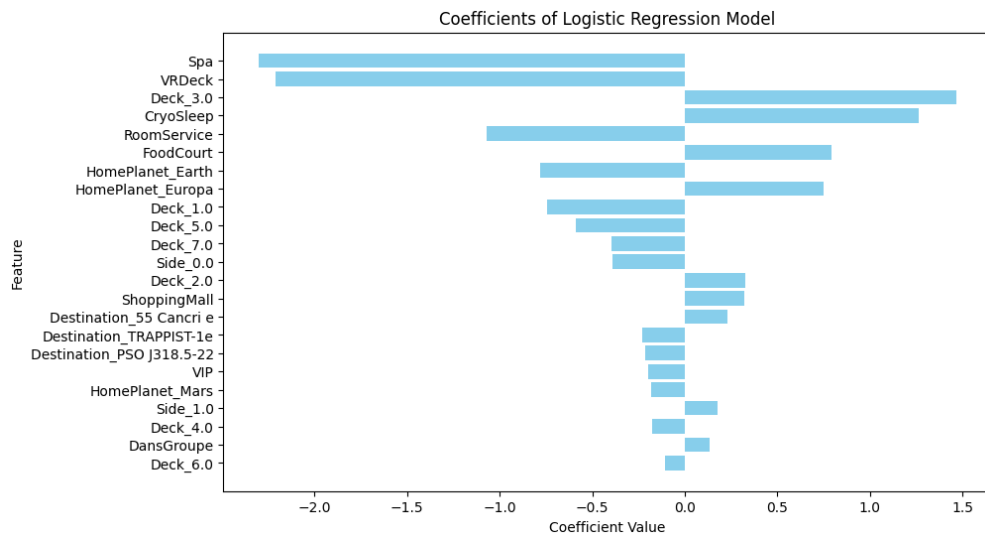
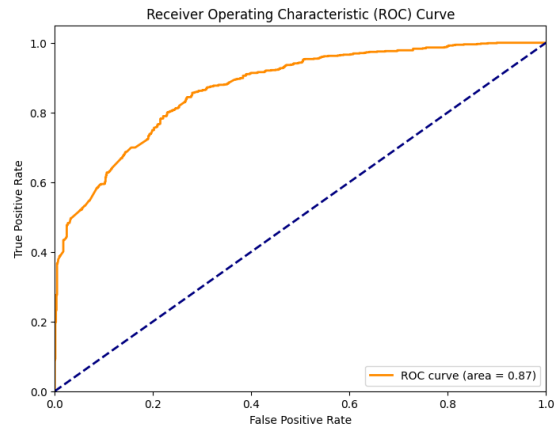
3.1 Régression logistique

En premier lieu, nous avons entraîné notre jeu de valeurs sur un modèle de régression logistique. Les premiers tests ce sont tous faits sur le dataset d'entraînement : data_train issu du fichier train.csv.

L'échantillonnage est de la forme suivante :

- 80% de valeurs d'entraînement ;
- 20% de valeurs de test.

Cette régression logistique nous a donné une valeur d'accuracy de 0.78263, et nous pouvons en sortir ces informations:



La courbe ROC de ce modèle nous apprend que ce dernier est efficace pour déduire les valeurs positives de Transported des passagers. En effet, l'aire sous la courbe, AUC, est de 0.87 ce qui est bien meilleur qu'un modèle complètement aléatoire.

Le graphique des coefficients des variables de notre modèle de régression logistique nous indique une importance de toutes les variables : elles sont soit positivement, soit négativement corrélées à la variable cible, mais dans tout les cas, elles sont importantes à l'entraînement du modèle.

Ce graphique a été réalisé avec notre dataset final, mais nous avons dans un premier temps des variables avec un coefficient proches de 0, comme Name ou Age que nous avons supprimées, augmentant ainsi notre score d'accuracy.

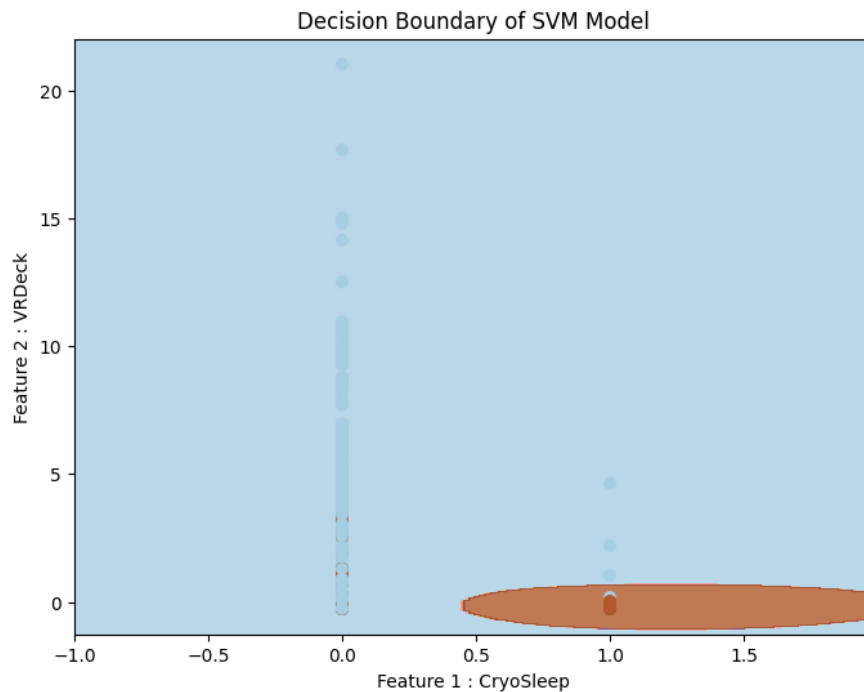
De plus, nous avons également sommé les variables de dépense dans une seule et même colonne, ce qui a eu pour effet de réduire considérablement les capacités de la régression logistique.

Le graphique des coefficients nous a permis de nous rendre compte de cette erreur car une fois séparées, on se rend compte que les variables de dépense ont des coefficient complètement différents : ils peuvent être positifs comme négatifs. Mais cette différence s'est estompée lorsqu'on les a sommées entre elles.

Le troisième graphique, "Learning Curve", représente le score obtenu lors de l'entraînement et lors de la validation en fonction du nombre d'exemples étudiés. On remarque que ces 2 courbes se rejoignent au fur et à mesure que le nombre d'exemples augmente, ce qui suggère que le modèle apprend correctement et ne fait pas de sur-apprentissage.

3.2 SVM

Le score obtenu via le modèle SVM est de 0.79523 après la cross-validation. Nous avons utilisé un noyau gaussien étant donné notre nombre faible de variable, et moyen d'exemples d'apprentissage : c'est aussi le noyau qui nous a donné les meilleurs résultats en pratique.



Le graphique ci-dessus indique la frontière de décision effectuée par le modèle SVM dans un espace bi-dimensionnel, pour simplifier la visualisation entre les variables CryoSleep et VRDeck, qui sont fortement corrélées à la variable cible Transported. L'objectif du SVM est de séparer au mieux les différentes classes, et ce graphique nous apporte la preuve que le modèle y arrive lors de son entraînement sur le dataset car la distinction entre les deux variables est nette. La recherche d'hyperparamètres a été effectuée via une recherche bayésienne. Les meilleurs hyperparamètres trouvés sont :

- $C = 4.985621981018506$;
- $\gamma = 0.06624098850466889$.

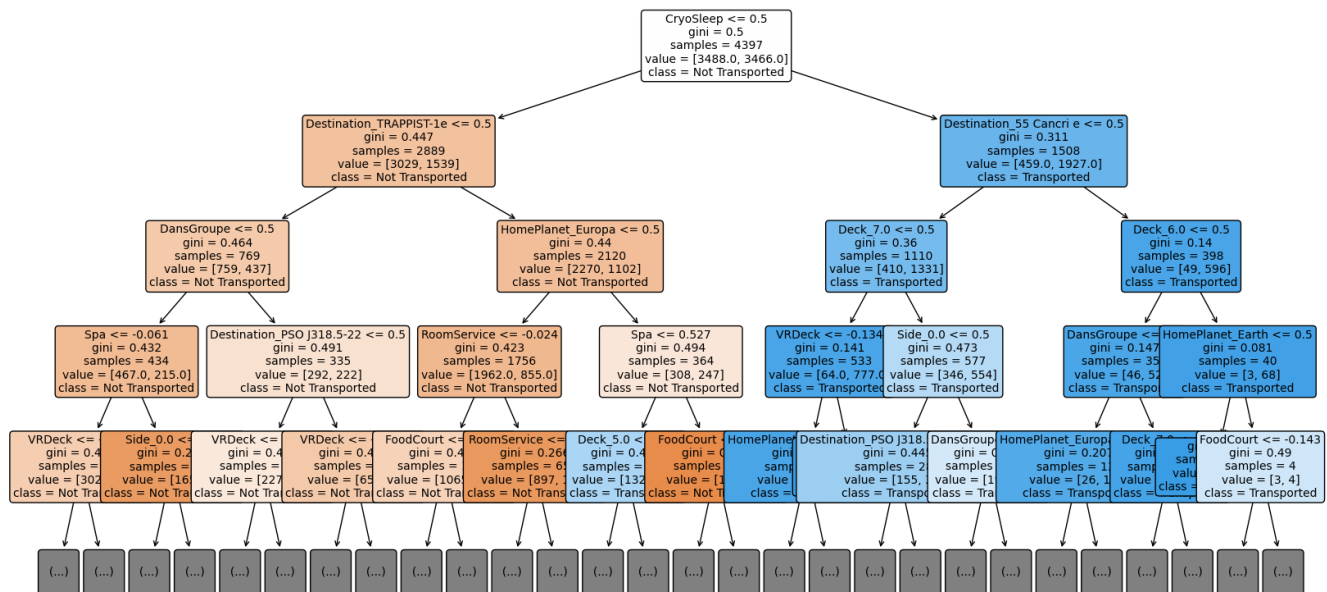
On remarque que le paramètre C est grand et que notre modèle n'est donc pas tolérant aux erreurs de classification. γ est relativement faible : le modèle n'accorde donc pas d'importance qu'au points proches de l'hyperplan, mais il considère aussi les points plus éloignés.

3.3 Random Forest

Notre modèle de random forest nous a donné une accuracy de 0.78838. Nous avons utilisé une méthode gridSearch de recherche d'hyperparamètres qui énumère les paramètres et retient les meilleurs. Les paramètres utilisés dans notre modèle sont:

- *'bootstrap'* : *False* ;
- *'max_depth'* : 10 ;
- *'max_features'* : *'log2'* ;
- *'n_estimators'* : 100.

Ce modèle cherche à éviter le sur-ajustement en ne réutilisant pas les variables déjà considérées (*bootstrap = False*) et en limitant la profondeur des arbres. Il limite également le nombre maximal de caractéristiques à considérer par arbre, ce qui permet d'augmenter la différence entre ces derniers et donc les aider à se spécialiser dans une partie différente de l'espace des caractéristiques du jeu de données.



Voici un exemple d'arbre obtenu avec le modèle randomForest. On remarque qu'il commence par séparer les classes du dataset de façon équitable grâce à la variable CryoSleep, ce qui confirme l'importance de cette variable que l'on avait supposée comme particulièrement importante lors de l'analyse des données.

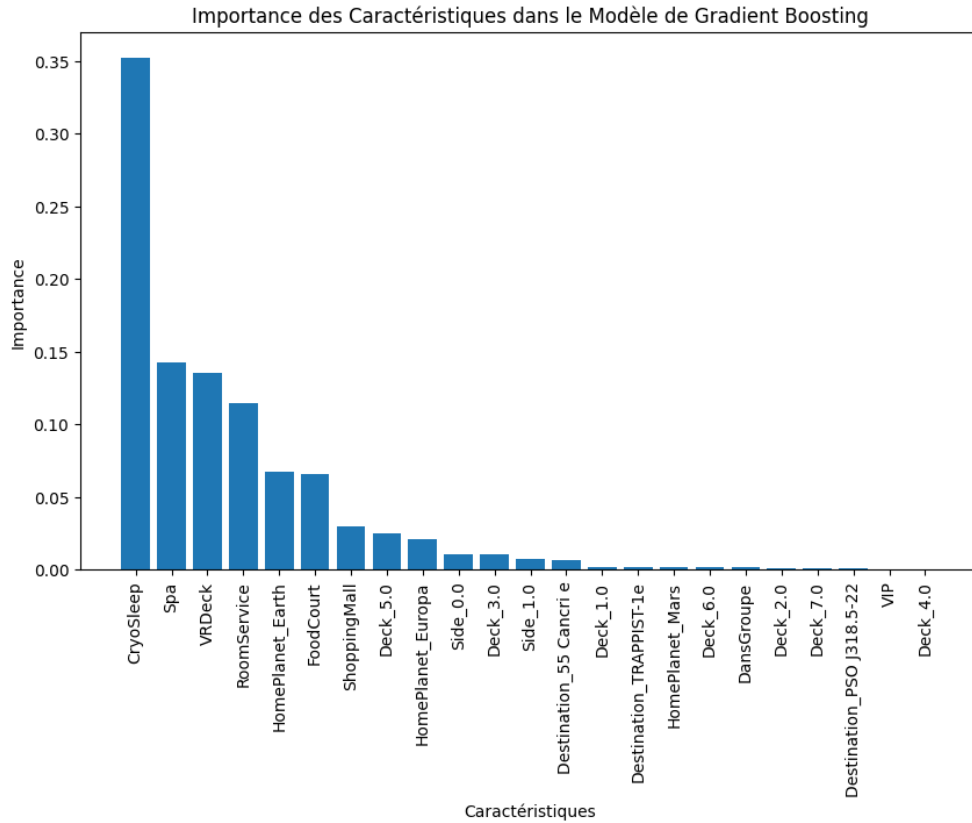
3.4 Boosting

Modèle	Score	Hyperparamètres
AdaBoost	0.792065	learning_rate = 1.0, n_estimators = 150
XGBoost	0.79752	gamma = 0.1, learning_rate = 0.1, max_depth = 3, n_estimators = 150
GradientBoosting	0.80219	loss="expotential"

Les modèles les plus efficaces sont les modèles de Boosting. Le meilleur score obtenu a été réalisé avec le modèle GradientBoosting, qui est un modèle ensembliste qui forme des arbres de décision peu profonds.

Au cours du déroulement de son algorithme, il calcule la somme des erreurs de ses modèles internes, met à jour les prédictions, puis il itère ceci en rajoutant de nouveaux modèles.

En résumé le Gradient Boosting construit un esemble de petits modèles qui se compensent entre eux itérativement.



Le GradientBoosting est le seul modèle à accorder une importance aussi grande et majoritaire à la variable CryoSleep. Le reste du classement d'importance des variables pour l'entraînement du modèle n'est pas surprenant et rejoint ce que les autres modèles ont fait, et ce qui a été supposé lors de l'analyse des données du dataset.

4 Résultat et score sur Kaggle


Tableau des scores des modèles:

Modèle	Entrainement	Test	HyperParamètres
Regression Logistique	0.78263	0.77368	
SVM	0.79523	0.79120	$C = 4.98562198106$ et $\gamma = 0.0662409885046$
RandomForest	0.78838	0.78770	$\text{bootstrap} = \text{False}$, $\text{max_depth} = 10$, $\text{max_features} = \log 2$, $\text{n_estimators} = 100$
AdaBoost	0.792065	0.78921	$\text{learning_rate} = 1.0$, $\text{n_estimators} = 150$
XGBoost	0.79752	0.75917	$\gamma = 0.1$, $\text{learning_rate} = 0.1$, $\text{max_depth} = 3$, $\text{n_estimators} = 150$
GradientBoosting	0.80927	0.80219	$\text{learning_rate} = 0.1$, $\text{max_depth} = 3$, $\text{n_estimators} = 150$

Le score de 0.80219 du GradientBoosting est notre meilleur score et nous place à la 694ème place, le 02/04/2024, du LeaderBoard Kaggle pour la compétition du SpaceShip Titanic.

694


M1 Valentin Sara Lina



0.80219

8

40m



Your Best Entry!
Your submission scored 0.79752, which is not an improvement of your previous score. Keep trying!

Les scores obtenus initialement étaient plus faibles. Nous avons réussi à augmenter l'accuracy de nos modèles principalement en repensant la manière dont nous avons traité le dataset d'entraînement.

En effet, un meilleur remplissage des cases vides, notamment dans la colonne CryoSleep et les colonnes de dépenses qui sont des variables importantes pour les modèles nous ont permis d'augmenter le score de tous nos modèles, qui étaient environ 2 millièmes plus bas. Nous avons eu l'idée de procéder à ce changement principalement en observant les graphiques d'importances et de coefficients des modèles.

5 Conclusion

Pour répondre à notre problématique, notre étude visait à prédire le succès du transport des passagers en utilisant diverses variables démographiques, socio-économiques et sur le transport des passagers. Le modèle GradientBoosting a dépassé nos attentes en termes de précision, confirmé par l'analyse des courbes ROC notamment. Les variables clés, telles que l'état de cryosommeil, ont été identifiées comme des contributeurs significatifs à la réussite du transport selon les différents modèles mis en place. Ces résultats fournissent des perspectives cruciales sur les dynamiques influant sur le transport des passagers.

Bien que les modèles soient des guides, ils offrent des indications pour des interventions ciblées.

En résumé, notre recherche souligne l'importance de comprendre les divers facteurs influant sur le transport des passagers du Spaceship Titanic. Le modèle GradientBoost se positionne comme une méthode prometteuse pour anticiper ces trajectoires, ouvrant ainsi des perspectives pour guider les actions visant à faciliter le sauvetage des survivants.

6 Code

Le code se trouve dans le fichier *SpaceshipTitanic.ipynb* à cette URL : <https://github.com/suprawall/machineLearningprojet>