Université de Sherbrooke

DÉPARTEMENT D'INFORMATIQUE

Projet de forage de données

Titanic - Forage de données à partir d'une catastrophe

Partie 1 : Données

Auteurs:

Membres de l'équipe:
AIT LHAJ, Walid
BOUHAMIDI EL ALAOUI, Kaoutar
RAZAFINDRAMISA, Andrianihary
THOMAS, Eliott

CIP:
aitw2501
bouk1001
raza3902
thoe2303

Superviseur:

Nadia Tahiri

Hiver 2022

Table des Matières

1 Introduction

1.1 Contexte général : Domaine d'étude

Le naufrage du paquebot Titanic dans l'océan Atlantique Nord est l'un des accidents le plus célèbre de l'histoire. Le 15 avril 1912, lors de son premier voyage, le Titanic coula après être entré en collision avec un iceberg. Malheureusement, il n'y avait pas assez de canots de sauvetage pour tout le monde à bord, ce qui entraîna la mort de 1502 des 2224 passagers et membres d'équipage. Cet évènement fut l'une des plus grandes catastrophes maritimes de l'époque.

Cette tragédie choqua la communauté internationale et mit en évidence l'insuffisance des règles de sécurité de l'époque. Elle mena les différentes parties prenantes dans le domaine de transport touristique et maritime à instaurer de meilleures mesures de régulations pour leurs navires, notamment dans les procédures d'évacuation d'urgence.

1.2 Description du projet et problématique

La tragédie du Titanic causa beaucoup de morts et entraîna des modifications des règles de voyages maritimes. Bien qu'il y ait eu un élément de chance dans la survie, il semble que certains groupes de personnes étaient plus susceptibles de survivre que d'autres. C'est dans ce cadre que se présente notre projet de session de forage de données.

Le but du projet est de faire la classification des personnes qui étaient à bord du paquebot pour savoir si elles ont survécu ou non en fonction de leurs données socio-économiques, ceci en utilisant des modèles de classification. Pour ce faire, notre projet est divisé en trois phases principales: une phase de manipulation des données (transformation, nettoyage), une phase de mise en œuvre d'algorithmes et une phase d'interprétations des résultats.

Notre problématique s'inscrit donc comme suit: "Quelles sont les catégories de personnes les plus susceptibles de survivre à la catastrophe du Titanic?"

1.3 Importance du sujet et motivations

Les motivations qui nous ont poussés à choisir ce sujet ont été les suivantes :

- Walid : Je n'ai pas encore regardé Titanic. C'est donc une occasion de savoir ce qui s'est passé de manière scientifique.
- Kaoutar: L'histoire du film 'Titanic' a éveillé ma curiosité sur le processus de sauvetage appliqué à l'époque, ainsi que sur les critères adoptés quant au choix des personnes installées sur les canaux de sauvetage.
- Andrianihary : Cette compétition est légendaire et il s'agit du premier défi à effectuer avant de commencer les compétitions d'apprentissage automatique sur Kaggle. C'est un excellent point de départ dans la science des données et, je trouve, qu'elle doit être faite par tous ceux qui veulent devenir scientifique des données.
- Eliott : Je trouve que ce sujet est l'équivalent du 'Hello World' dans le site de Kaggle et dans le domaine du forage de données en particulier. Il n'en demeure pas moins intéressant et challengeant de par sa grande popularité.

2 Description des données

Les données contiennent des informations sur les passagers comme leur nom, leur classe socio-économique, leur genre, leur âge ou leur port d'embarquement. Ces données sont divisées en deux groupes :

- L'ensemble "train.csv" pour créer les modèles ;
- Le jeu de test "test.csv" pour voir dans quelle mesure le modèle fonctionne sur la prédiction de survie de passagers non classés.

Feature	***Valeur***	***Type***
PassengerId	1-891	Entier
Survived	0,1	Entier
Pclass	1-3	Entier
Name	Nom des passagers	Objet
Sex	Male, female	Objet
Age	0-80	Réel
SibSp	0-8	Entier
Parch	0-9	Entier
Ticket	numéro du ticket	Objet
Fare	0-512	Réel
Cabin	numéro de cabine	Objet
Embarkment	S, C, Q	Objet

Table 1: Description des Données du Dataset

2.1 Description des attributs

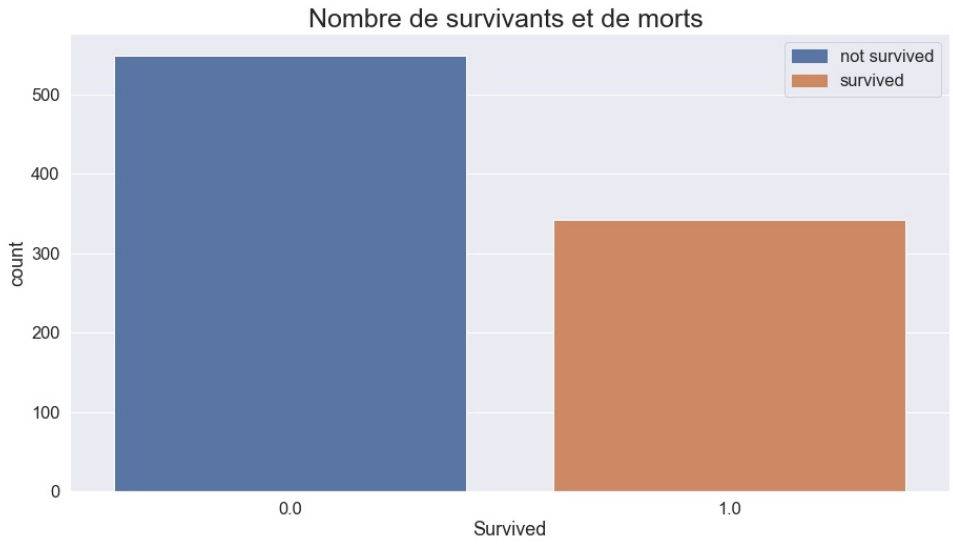
Nous allons à présent détailler les caractéristiques de l'ensemble de données d'entraînement et de test combinées, leurs statistiques et leur visualisation :

2.1.1 PassengerId

Cette caractéristique donne l'ID unique du passager. Elle va de 1 à 891.

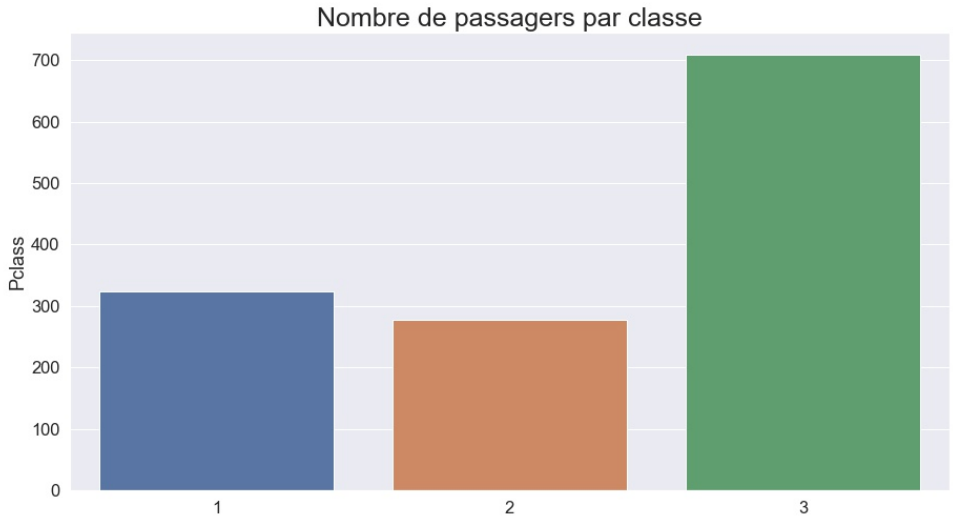
2.1.2 Survived

Cette caractéristique précise si le passager a survécu ou non. "0" signifie que le passager est mort et "1" signifie qu'il a survécu. Sur tous les passagers des données d'entraînement, 549 sont morts et 342 ont survécu. C'est ce que représente la figure suivante .

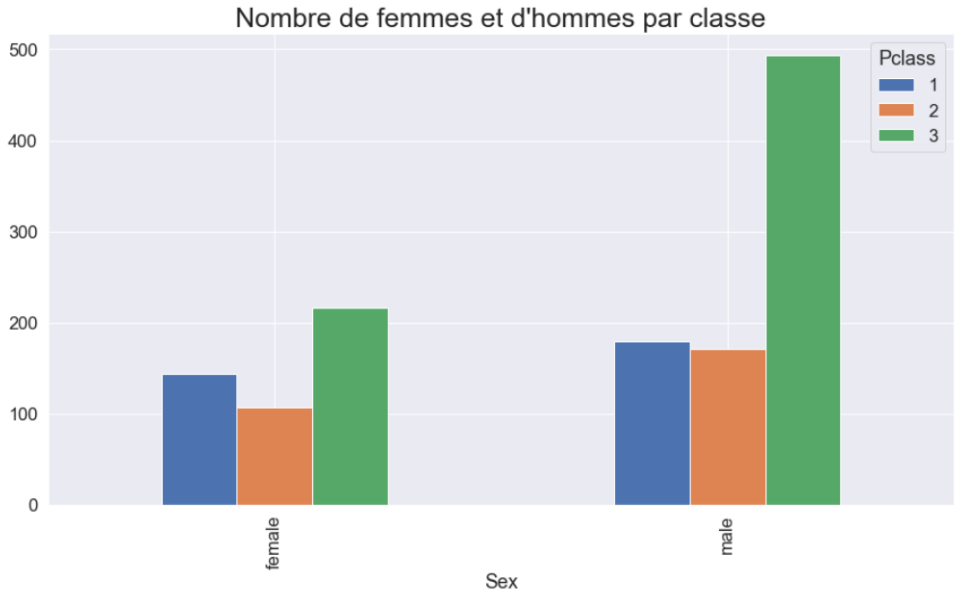


2.1.3 Pclass

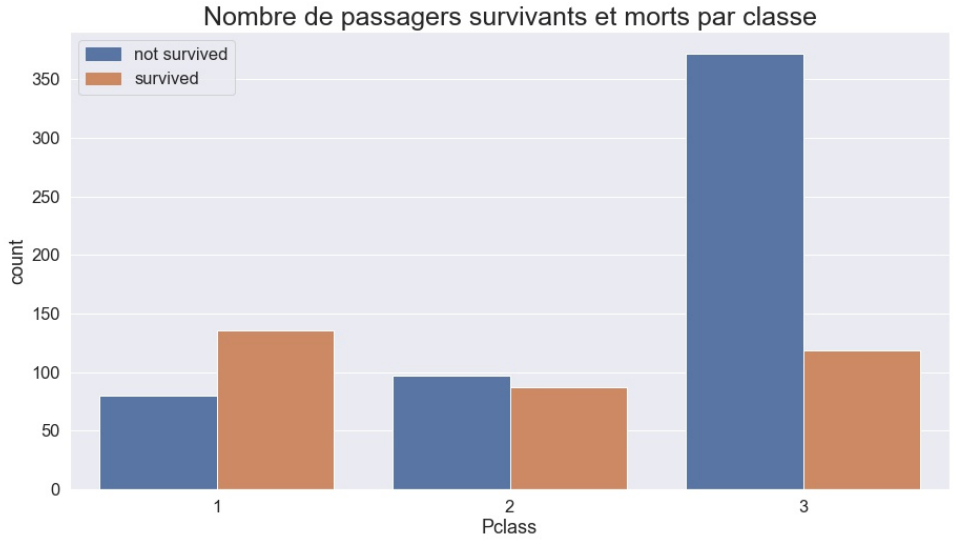
Cette caractéristique donne la classe du passager. "1" signifie que le passager était en 1ère classe (323 passagers), "2" en 2e classe (277 passagers) et "3" en 3e classe (709 passagers). C'est un indicateur du statut socio-économique du passager. La figure suivante représente le nombre de passagers par classe.



On représente également le nombre de femmes et d'hommes par classe dans la figure qui suit.



Les passagers avec le taux de survie le plus élevé sont les passagers les plus riches c'est-à-dire ceux de la première classe. C'est ce que représente la figure ci-dessous.

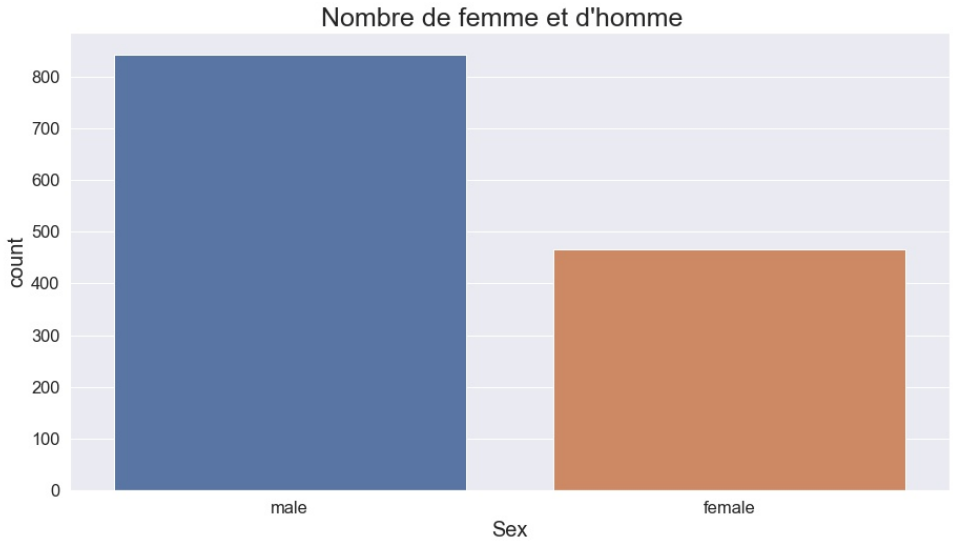


2.1.4 Name

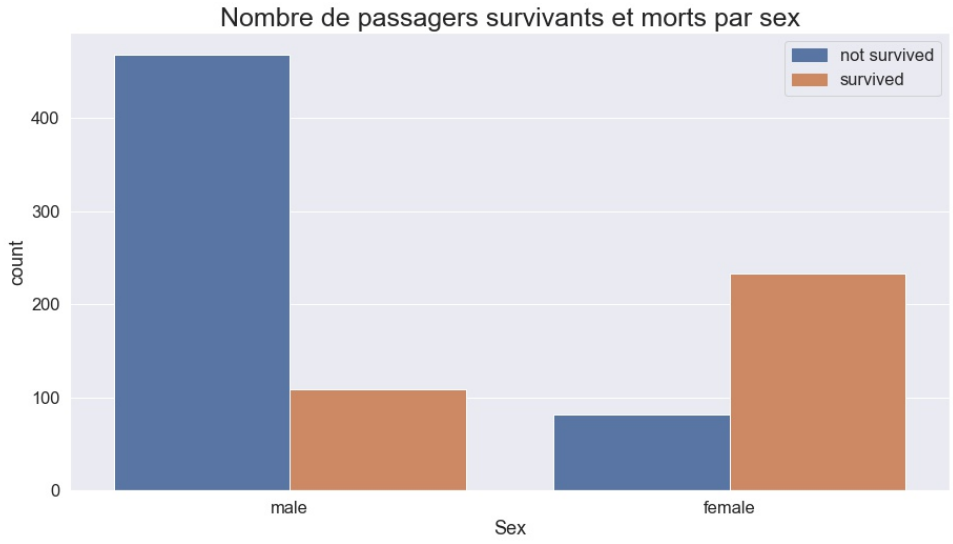
Cette caractéristique donne le nom de chaque passager. Ce nom est unique.

2.1.5 Sex

Cette caractéristique donne le genre du passager. Il y a deux catégories "Male" et "Female". 466 passagers sont des femmes (35,59 % des passagers) et 843 sont des hommes (64,40 % des passagers).



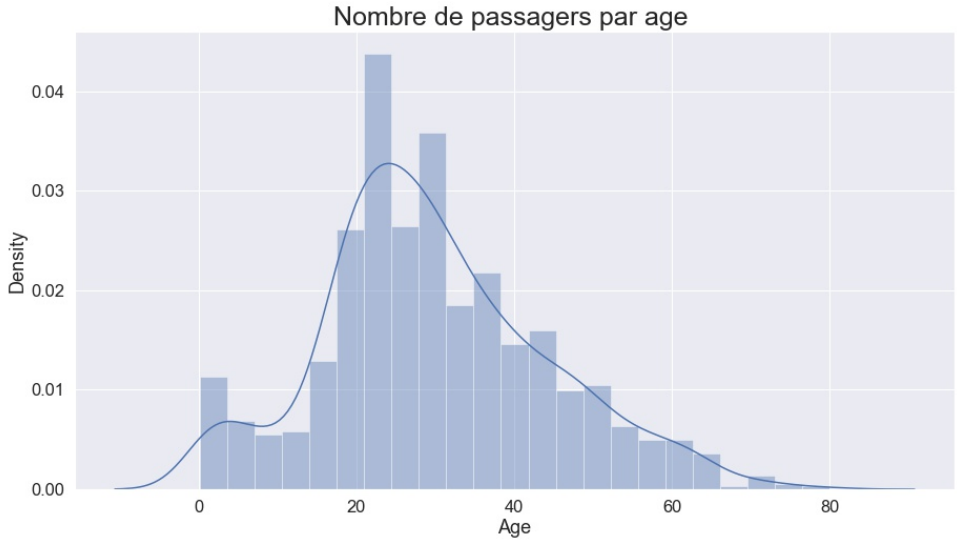
Cela nous permet d'obtenir le taux de survie pour les données d'entraînement. 50% des femmes ont survécu (233 passagères), contre seulement 12.9% des hommes (109 passagers).



2.1.6 Age

Cette caractéristique précise l'âge du passager. L'âge des passagers va de 0 à 80 ans et l'âge moyen est de 29.88 ans. La majorité des passagers ont entre 20 et 40 ans. C'est ce que représente la boîte à moustaches en dessous.

On visualise également le nombre de passagers par âge et on confirme que la majorité des passagers avaient entre 20 et 40 ans comme le montre le pic de la courbe.

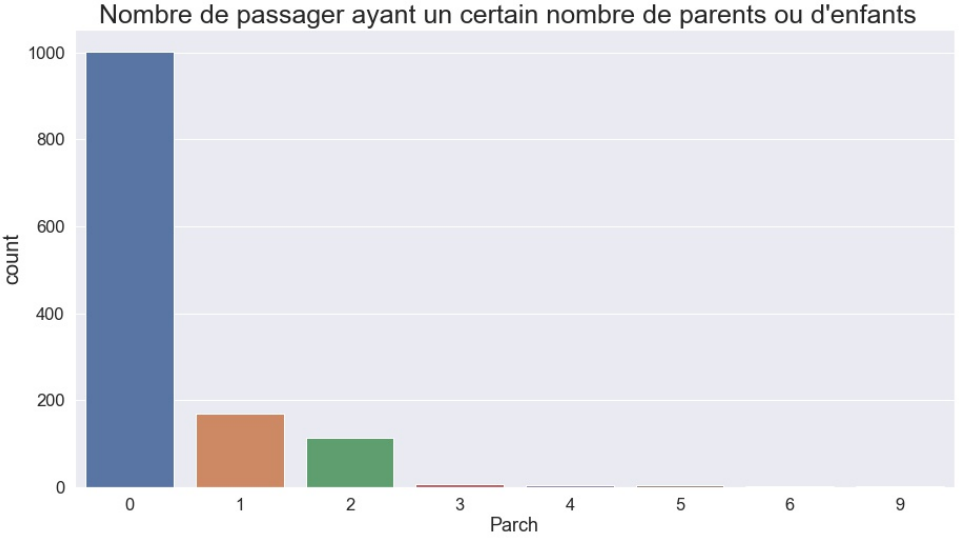


2.1.7 SibSp

Cette caractéristique donne le nombre de passagers ayant un certain nombre de frères et sœurs (les demi-frères et demi-sœurs sont pris en compte) et de conjoints (les fiancés sont ignorés) à bord du Titanic. Elle va de 0 à 8. 891 passagers n'ont pas de frères et sœurs ou de conjoint à bord, et 418 en ont au moins un. Ceci est montré dans la figure ci-dessous.

2.1.8 Parch

Cette caractéristique donne le nombre de parents/enfants du passager à bord, les belles-filles et les beaux-fils sont compris dans ce nombre. Ce nombre va de 0 à 9. 1002 passagers n'ont pas de parents ou d'enfants à bord.



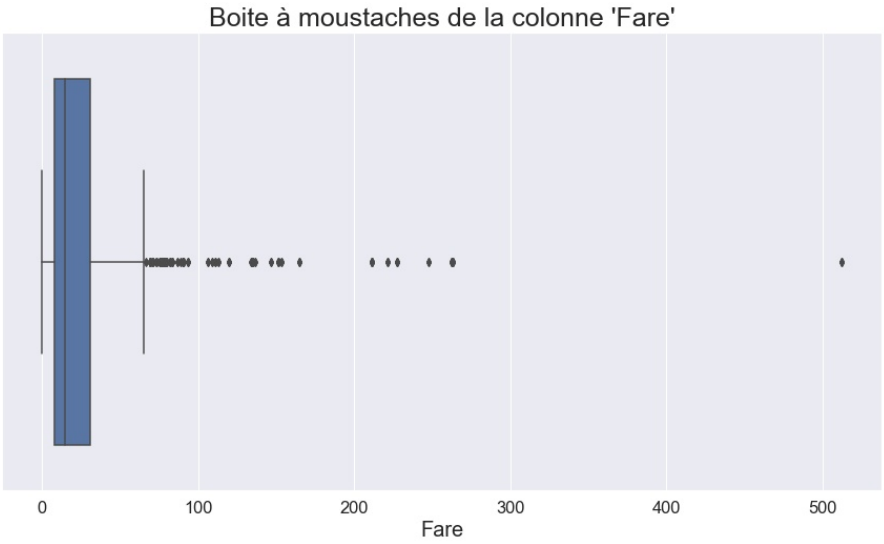
2.1.9 Ticket

Cette caractéristique précise le numéro de billet du passager. Ticket a 681 valeurs uniques.

2.1.10 Fare

Cette caractéristique précise le prix payé par chaque passager pour son ticket et ce montant est compris entre 0 et 512. La majorité des passagers ont payé entre 7.89 et 31.27.

C'est ce que la boîte à moustaches en dessous montre.

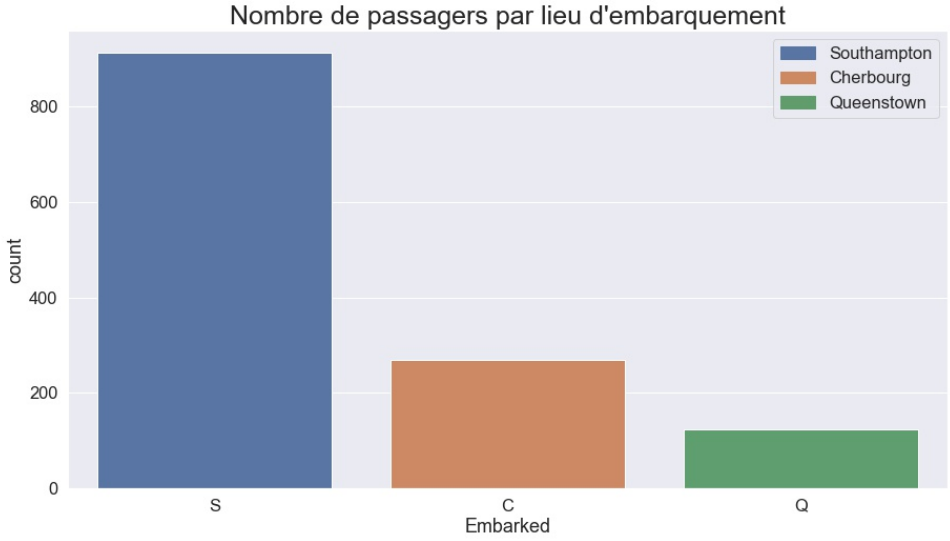


2.1.11 Cabin

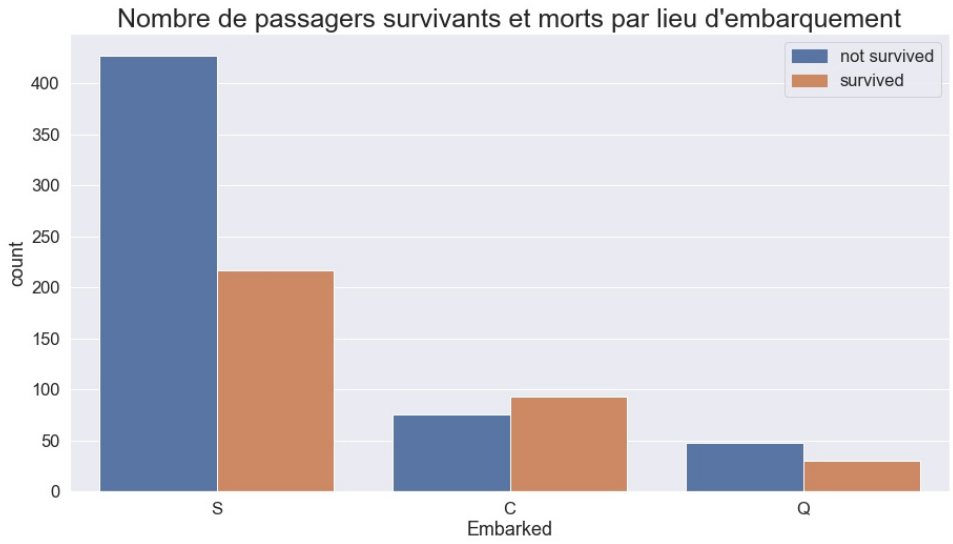
Cette caractéristique donne le numéro de cabine du passager. On ne connaît pas ce numéro pour 1014 passagers.

2.1.12 Embarked

Cette caractéristique donne le port d'embarquement du passager. "C" signifie Cherbourg, "S" signifie Southampton et "Q" signifie Queenstown. 914 des passagers ont embarqué à Southampton contre 270 à Cherbourg et 123 à Queenstown .

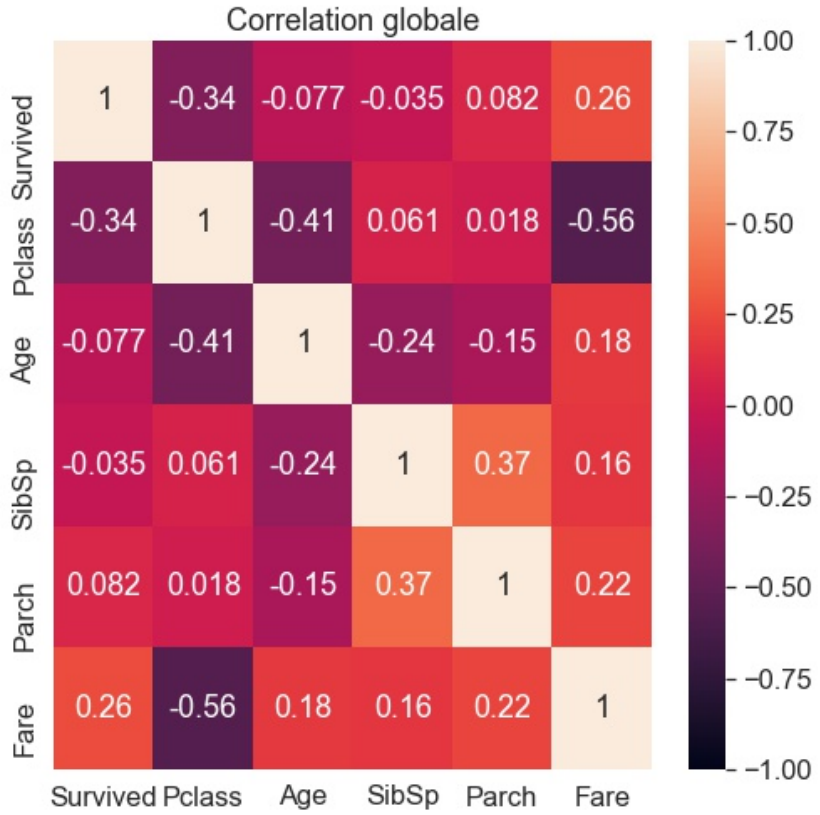


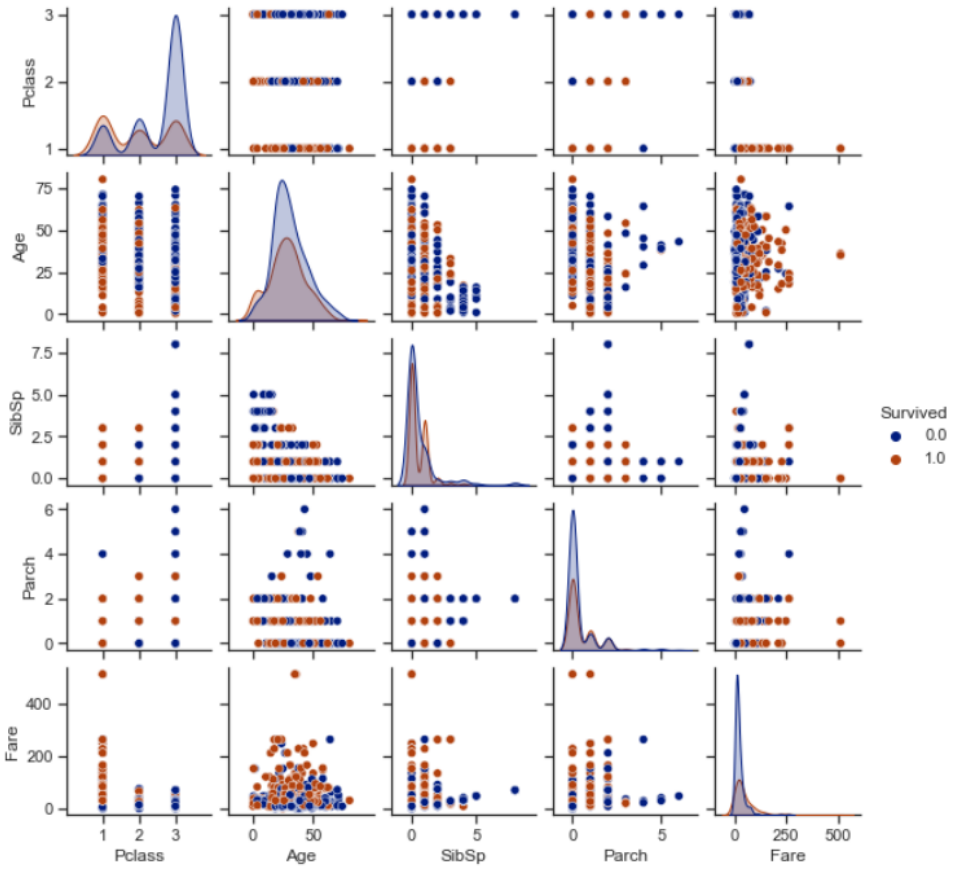
La figure suivante montre que les personnes qui ont embarqué à Cherbourg ont le taux de survie le plus élevé.



2.2 Visualisation Globale

Pour finir, nous allons vous présenter une visualisation globale des données avec une matrice de corrélation.





Grâce au premier graphique, nous serons en mesure de savoir quels attributs privilégier dans la partie 2 par rapport aux autres. Typiquement *Pclass* semble avoir un impact bien plus important sur la survie que *SibSp*.

Le second graphique, le pair plot, permet également (entre autres) de mettre en lumière l'importance de la classe du passager dans sa survie. En effet, la colonne "classe 3" est majoritairement de couleur bleue, indiquant que les personnes moins favorisées économiquement étaient également moins susceptibles de survivre.

3 Historique des travaux et des développements faits en rapport avec le sujet

L'ensemble de données Titanic est un data set classique des sciences de données. On peut s'en rendre compte facilement en regardant le nombre de résultats fournis par Google Scholar avec '*dataset Titanic*' : environ 10000 ! Nous avons isolé trois papiers de recherche sur le sujet que vous pourrez retrouver dans la section Références :

- [2] qui applique 14 techniques d'apprentissage automatique à l'ensemble de données.
- [3] qui utilise des outils d'exploration de données modernes (Weka) pour trouver une relation convaincante entre la survie des passagers et leurs caractéristiques.
- [4] qui est un récent article de 2020 reprenant les techniques classiques de ML et visant une fois de plus à prédire le sort des passagers basé sur leurs caractéristiques.

Nous allons vous présenter une liste non exhaustive des techniques et algorithmes présentés dans les articles ci-dessus. Pour chacune d'entre elles nous feront une brève description explicative ainsi que la liste des articles mentionnant cette technique.

- Régression logistique : C'est un algorithme de classification qui fonctionne sur des données discrètes. Il est basé sur l'équation de la sigmoïde :

  $$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

  Cette approche est utilisée par [4]

- Machine à Support de Vecteur (SVM) : C'est un algorithme qui permet de séparer les données grâce à un hyperplan. Cette technique fonctionne mieux sur les petits ensembles de données. Cette approche est utilisée par [2] et [4]

- Arbre décisionnel : génère un arbre de décision suggérant un chemin de classification et donc les caractéristiques les plus significatives en ce qui concerne la survie. L'intérêt majeur de cette approche est qu'elle a une très bonne interprétabilité. Cette approche est utilisée par [2], [3] et [4].

- Simple K Means Cluster Analysis : regroupe les données à partir d'associations simples et permet d'analyser visuellement les groupes au sein de l'ensemble de données. Cette approche est utilisée par [3].

- K plus proches voisins (KNN) : un des algorithmes de classification les plus courants, utilise les métriques de distance pour mesurer la proximité entre les échantillons d'apprentissage et l'échantillon de test. Il attribue à l'échantillon de test la classe de ses k échantillons d'apprentissage les plus proches. Il est principalement basé sur la distance euclidienne. Cette approche est utilisée par [2] et [4].

- Bayes Naïf (NB) : basé sur le théorème de Bayes en supposant que toutes les caractéristiques sont indépendantes compte tenu de la valeur de la variable de classe, NB permet une classification efficace et rapide. NB fonctionne bien sur des ensembles de données complexes de grande dimension. Cette approche est utilisée par [2].

- Bagging : technique pour créer un ensemble de classifieurs, améliore la précision en ré-échantillonnant l'ensemble d'apprentissage. Un classifieur unique de base est appliqué aux ensembles d'apprentissage générés puis les modèles de classification générés sont combinés en fonction du vote à la majorité. Cela permet de réduire la variance et le biais. Cette approche est utilisée par [2].

- AdaBoost : est une méthode d'apprentissage d'ensemble. La notion de base est qu'un classifieur fort peut être créé en combinant linéairement un certain nombre de classifieurs faibles. AdaBoost augmente le poids des points de données mal classés tout en diminuant les poids des points de données correctement classés ce qui permet de pondérer à nouveau toutes les données d'entraînement à chaque itération. Les classifieurs faibles sont appliquées en série, puis les modèles de classification générés sont combinés en fonction du vote à la majorité pondérée. Cette approche est utilisée par [2].

- Extra Trees : méthode de classification d'ensemble d'arbres de décision basée sur la randomisation. Pour chaque nœud de l'arbre, des règles de fractionnement sont tirées au hasard, puis la règle la plus performante est associée à ce nœud. Cette approche est utilisée par [2].

- Forêt aléatoire : algorithme de classification qui utilise un ensemble de prédicteurs d'arbres, chaque arbre est construit en amorçant les données d'apprentissage et, pour chaque fractionnement, un sous-ensemble de caractéristiques sélectionné au hasard est utilisé. Cette méthode estime les données manquantes tout en conservant l'exactitude. Cette approche est utilisée par [2].

- Gradient Boosting : Cette méthode utilise le boosting pour estimer des fonctions. Cette approche est utilisée par [2].

- ANN et MLP. Les réseaux de neurones artificiels (ANN) et les perceptrons multicouches (MLP) permettent de résoudre des problèmes de classification non linéaires avec une très bonne précision. Cette approche est utilisée par [2].

- Vote : On peut agréger ensemble de nombreux modèles différents pour obtenir une classification de meilleure qualité. On peut faire voter les modèles uniformément. On peut également pondérer l'impact de certains modèles si on juge l'importance de ces derniers non équilibrée. Cette approche est utilisée par [2].

4 Pré-traitement des données

4.1 Traitement des données manquantes

Afin de procéder au traitement des données manquantes on commence d'abord par voir le nombre d'éléments manquants par colonne. Nous remarquons grâce au tableau ci-dessous que les colonnes Cabin et Age ont le plus de données manquantes, alors que Fare et Embarked ont très peu de valeurs de moins.

	Types	Nb manquants	Ratio manquants%
Pclass	int64	0	0.000000
Name	object	0	0.000000
Sex	object	0	0.000000
SibSp	int64	0	0.000000
Parch	int64	0	0.000000
Ticket	object	0	0.000000
Fare	float64	1	0.000764
Embarked	object	2	0.001528
Age	float64	263	0.200917
Cabin	object	1014	0.774637

Figure 1: Type et ratio de données manquantes des colonnes

Pour "Embarked", on remplace les deux valeurs manquantes par la valeur la plus fréquente. Dans notre cas 'S' est présente 914 fois, C est présent 270 fois et Q est présente 123 fois. On va donc assigner la valeur 'S' pour les valeurs manquantes.

Pour la colonne "Fare", on remplace les valeurs manquantes par la médiane de "Fare" pour la classe 3 vue que la personne qui n'a pas de "Fare" appartient à cette classe et les colonnes 'Pclass' et 'Fare' sont corrélées.

Pour la colonne des "Age", on remplace les valeurs manquantes par la médiane des âges par classe vue que ces deux colonnes sont assez bien corrélées.

Cependant pour la colonne des cabines, on ne peut malheureusement pas faire de traitements des valeurs manquantes car la cabine dépend de chaque passager.

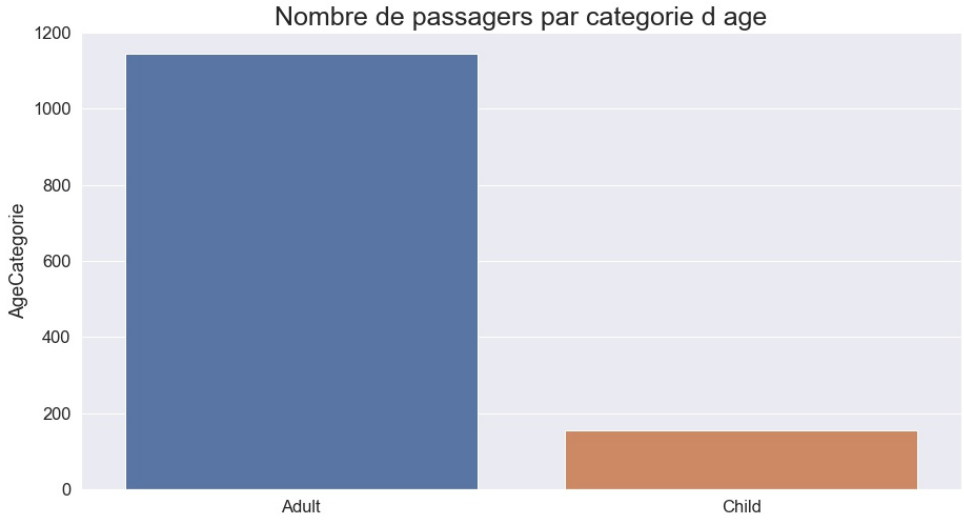
4.2 Traitement des valeurs aberrantes

On décide de supprimer les valeurs aberrantes de la colonne des âges , c'est-à-dire les valeurs supérieures à (1.9 IQR + Q3). Ceci nous permet d'éliminer les personnes de plus de 65 ans. Cependant pour la colonne 'Fare' on ne va pas supprimer les quatre valeurs qui peuvent être vues comme des valeurs aberrantes car ce sont manifestement des vraies valeurs.

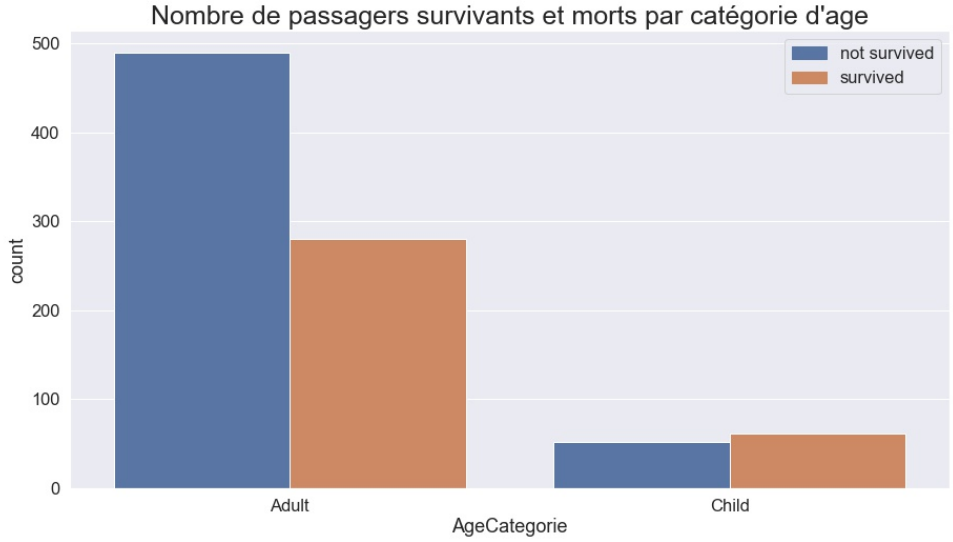
4.3 Création de nouvelles colonnes

On remarque que, pour les colonnes des âges et des prix de ticket payés par passager, les valeurs sont assez diversifiées.

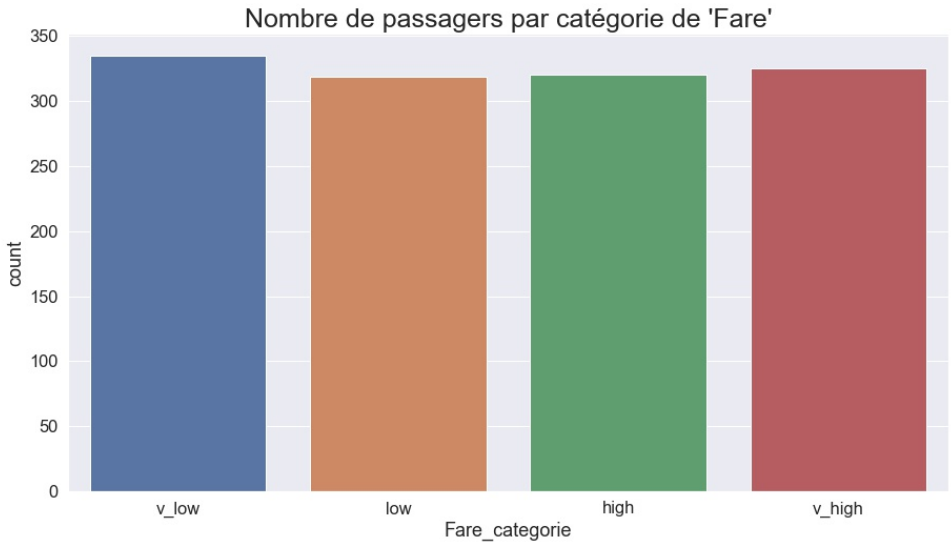
Pour la colonne des âges, on choisit de créer une catégorie 'Child' pour les personnes de moins de 18 ans et une catégorie 'Adult' pour le reste des passagers. On peut voir le nombre de personnes par catégorie d'âge dans la figure ci-dessous. On remarque que la majorité des passagers du Titanic étaient des adultes.



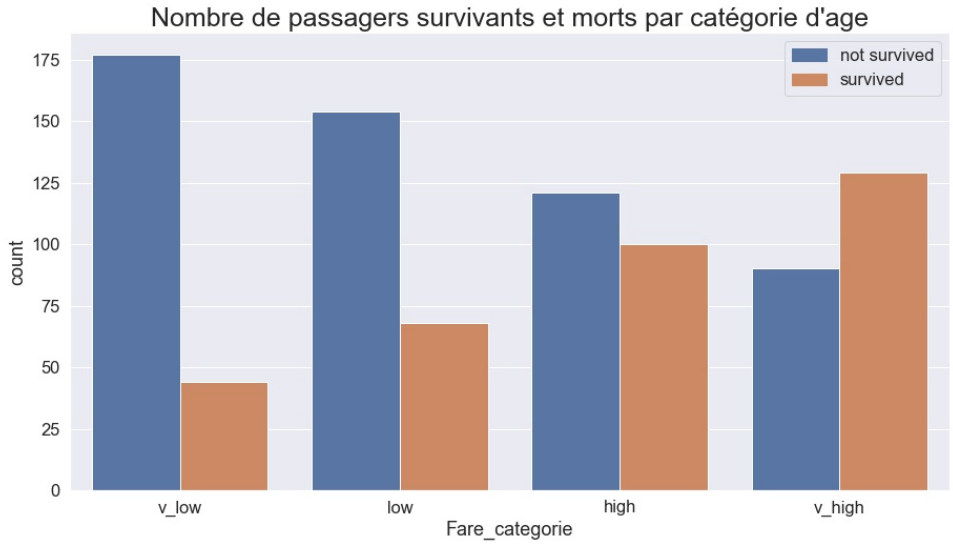
On s'intéresse également au taux de survie dans les deux catégories. On voit donc que celui-ci est plus élevé que le taux de mortalité chez les enfants. Cependant c'est l'inverse pour les adultes. C'est ce que montre la figure ci-dessous.



Pour la colonne des prix payés 'Fare', on choisit de créer quatre catégories 'v_low', 'low', 'high', 'v_high' pour répartir les valeurs des prix. La figure ci-dessous représente la répartition des passagers par catégorie de 'Fare'. On a séparé les données en 4 parties égales grâce à la méthode *pandas.qcut*. On observe que la répartition n'est pas exactement de 4 fois 25% ; les personnes ayant payé le même prix se retrouvent dans la même catégorie.



On souhaite connaître l'influence de la catégorie du prix payé sur la survie. On voit donc que le taux de survie augmente avec l'augmentation des prix payés, c'est ce qu'on voit sur la figure ci-dessous.



Après avoir regardé l'architecture du bateau qui est représentée ci-dessous. On remarque que pour la colonne des cabines les valeurs commencent par une lettre suivie d'un numéro.

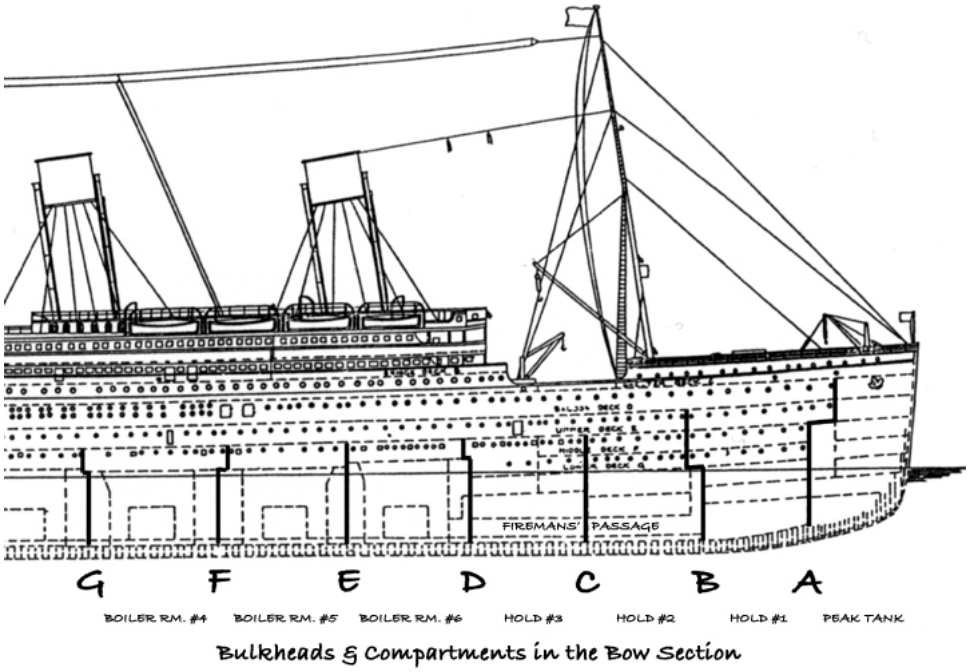
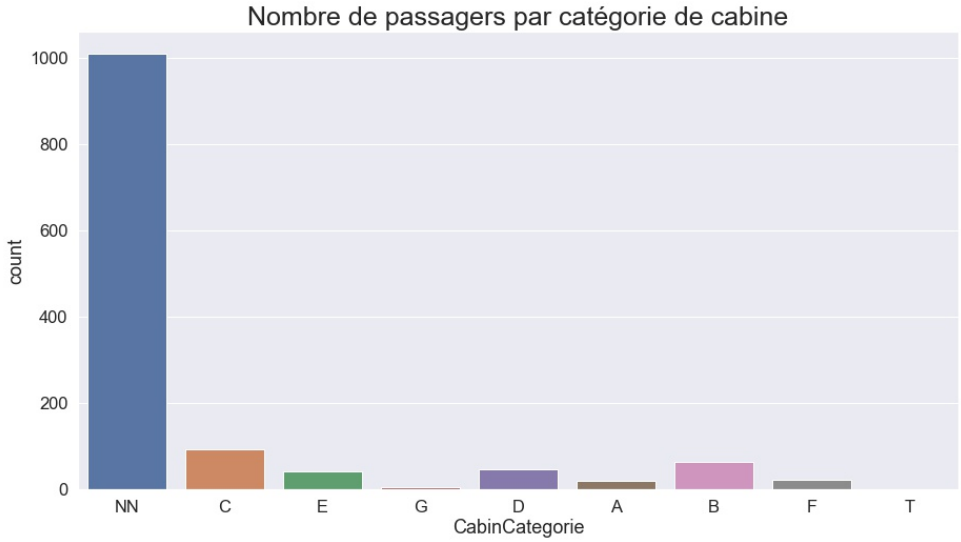


Figure 2: Architecture du bateau

On s'intéresse donc ainsi à garder la lettre et on l'utilise pour créer la colonne *cabin_categ*. On obtient ainsi la répartition des passagers par catégorie de cabine comme le représente la figure ci-dessous.



On remarque aussi qu'il y a une catégorie 'T' qui n'est présente que pour une seule personne. On estime donc qu'il s'agit d'une erreur. On décide, par conséquent, de lui assigner la valeur 'NN'.

Pour la colonne du sexe, on choisit d'en faire une colonne 'female' de données binaires. On assigne la valeur de 1 si le passager est bien une femme et 0 si non.

Par la suite, on utilise la fonction *get_dummies* de pandas sur les colonnes 'CabinCat-

egorie', 'Pclass', '*Fare_categorie*', 'Embarked', 'AgeCategorie' afin de créer les one-hot vecteurs. En d'autres termes, on crée une colonne par catégorie, on supprime les colonnes 'Cabin', 'Name', 'Ticket', 'Sex', 'Fare'.

5 Mise en œuvre

Après cette succession de pré-traitements sur les données de test et de train réunies, on les sépare et on crée les ensembles de données de train et de test afin de commencer l'entraînement de nos modèles. De plus, on divise nos données d'entraînement en données de train (80%) et en données de validation (20%). Les données de train vont servir dans un premier lieu à entraîner les classifieurs avec les paramètres par défaut puis à la recherche d'hyperparamètres grâce à un Grid Search CV. Les données de validation vont servir à tester le meilleur modèle retenu puis à obtenir la valeur de la justesse et du F1-score correspondant. On choisit comme classifieurs: la régression logistique, les Forêt aléatoire, AdaBoost, le SVM, les réseaux de neurones (Multilayer Perceptron), l'arbre décisionnel, le gradient boosting, le KNN et le vote.

5.1 Entraînement

Afin d'entraîner nos classifieurs, on utilise la classe Grid Search CV avec plusieurs valeurs des hyperparamètres. Et pour éviter le sur-apprentissage, nous utilisons la validation croisée à cinq divisions : 4 divisions sont utilisées pour l'entraînement et le 5ème fold sert de validation. Cette recherche d'hyperparamètres nous permettra de trouver, pour chaque classifieur, une combinaison à utiliser pour entraîner notre modèle afin d'obtenir la meilleure justesse.

5.1.1 Les hyperparamètres des classifieurs

Pour chaque classifieur, on a choisi des hyperparamètres à optimiser en utilisant la validation croisée.

- Régression logistique:
  - C: L'inverse de la régularisation. Plus sa valeur est petite, plus la régularisation est forte.
  - solver: Algorithme à utiliser pour l'optimisation.
  - penalty: La pénalité utilisée (Aucune ou l2).
  - tol: La tolérance pour les critères d'arrêt.
- Random Forest:
  - n_estimators : Nombre d'arbres dans la forêt. Plus il est grand, meilleure est l'approximation mais plus le fit est long à calculer.
  - max_depth : La profondeur maximale des arbres. Plus il est grand, plus on a de sur-apprentissage et plus de temps à calculer.
  - min_samples_leaf : Nombre d'éléments minimum nécessaires dans un nœud.
  - min_samples_split : Nombre d'éléments dans un nœud pour pouvoir le séparer.
  - max_features : Nombre de caractéristiques considérées pour séparer un nœud.

  - criterion: Fonction de mesurer de la qualité de division. ("gini" pour les impuretés Gini ou "entropy" pour les informations d'entropie).

- AdaBoost:
  - n_estimators : Nombre de modèles à entraîner de manière itérative.
  - learning_rate : Contribution de chaque modèle aux pondérations.

- SVM:
  - kernel : Choix des noyaux.
  - gamma : Paramètre pour les hyperplans non linéaires. Plus il est grand, plus la machine essaie de s'ajuster exactement sur l'ensemble des données d'apprentissage.

- Réseau de neurones MLP:
  - hidden_layer_sizes : Taille des couches cachées du réseau.
  - solver : Méthode d'optimisation des poids W.
  - learning_rate_init : Longueur initiale des pas lors de la descente de gradient.

- Arbre de décision :
  - max_depth : La profondeur maximale de l'arbre.
  - min_samples_split : Nombre minimal d'échantillons requis pour diviser un nœud interne.
  - criterion : Fonction de mesurer de qualité de division. ("gini" pour les impuretés gini ou "entropy" pour les informations d'entropie).

- Gradient boosting :
  - learning_rate : Le pas d'apprentissage divise ici la contribution de chaque estimateur par le taux d'apprentissage.
  - min_samples_leaf : Le nombre d'échantillons minimum que l'on peut retrouver par feuille.
  - min_samples_split : Le nombre d'échantillons minimum nécessaire pour faire un split.
  - max_depth : La profondeur maximale des estimateurs.
  - max_feature : Le nombre d'attributs à considérer pour trouver la meilleure séparation. Fonction pour mesurer la qualité d'une division.

- KNN :
  - n_neighbors : Le nombre de voisins utilisés par défaut pour interroger.
  - leaf_size : La taille des feuilles passées à Ball Tree ou K-D Tree. Cela peut affecter la vitesse de construction et de requête, ainsi que la mémoire nécessaire pour stocker l'arborescence. La valeur optimale dépend de la nature du problème.
  - weights :Comment pondérer l'importance des points pour le KNN. *Uniforme* pour une importance égale ou *Distance* pour donner plus d'importance aux points plus proches.
  - metric : La mesure de distance à utiliser pour l'arbre.

Note : Après avoir trouvé les meilleurs hyperparamètres pour tous les modèles, nous pouvons les utiliser afin de créer nos modèles de Classification par vote majoritaire.

6 Résultats

Afin de comparer la performance de nos modèles, on fait, dans un premier lieu, une recherche d'hyperparamètres pour des données non standardisées puis pour des données standardisées.

6.1 Meilleurs hyperparamètres pour les classifieurs sans standardisation

Nous avons effectué une recherche d'hyperparamètres avec le Grid Search CV. Nous avons commencé par rechercher des valeurs de paramètre proches des valeurs par défaut, puis nous avons concentré nos recherches sur les valeurs prometteuses.

Par exemple pour le *MLP*, pour l'attribut *hidden layer sizes* nous avons testé originalement 10 valeurs de 60 à 150. La meilleure valeur retournée était pour 110. Nous avons ensuite testé tous les entiers de 105 à 115 et cela nous a retourné 110 de nouveau. C'est donc ici que nous arrêtons nos recherches pour cet hyperparamètre.

Nous avons procédé de manière similaire pour tous les hyperparamètres. On représente dans ce qui suit les meilleurs paramètres finaux trouvés pour chaque classifieur.

6.1.1 Régression logistique

- C: 0.1
- penalty: l2
- solver: newton-cg
- tol: 1e-06

6.1.2 Random Forest

- criterion: gini
- max_depth: None
- max_features: log2
- min_samples_leaf: 2
- min_samples_split: 5
- n_estimators: 10

6.1.3 AdaBoost

- learning_rate=1.5
- n_estimators=20

6.1.4 SVM

- kernel: rbf
- gamma: 0.05

6.1.5 Réseau de neurones

- hidden_layer_sizes: 110
- learning_rate_init: 0.10010000000000001
- solver: adam

6.1.6 Decision Tree

- criterion: entropy
- max_depth: 10
- min_samples_leaf: 4

6.1.7 Gradient Boosting

- learning_rate: 0.2
- max_depth: 5
- max_features: None
- min_samples_leaf: 0.1
- min_samples_split: 0.30000000000000004

6.1.8 KNN

- leaf_size=1
- n_neighbors=9
- metric: manhattan
- weights: distance

6.2 Meilleurs hyperparamètres pour les classifieurs avec standardisation

On a également fait la recherche des hyperparamètres avec des données standardisées. Nous présenterons uniquement dans ce qui suit nos résultats de justesse et de f1-score pour des soucis de lisibilité et pour éviter la redondance. Néanmoins, le code nécessaire à la production de ces valeurs peut être trouvé en annexe [1].

6.3 Comparaison des classifieurs et analyse des résultats

Afin de comparer les meilleurs modèles obtenus avec la recherche d'hyperparamètres, on calcule pour chacun d'entre eux la justesse et le f1-score sur les données de validation. Le choix de ces métriques n'est pas arbitraires. D'abord l'accuracy correspond au rapport entre les observations correctement prédites et le nombre total des observations. Cette métrique est utile dans notre cas car on est plus intéressés par les vrais positifs et les vrais négatifs, en d'autres termes ceux qui sont vraiment vivants et ceux qui sont vraiment morts. Le f1-score est une métrique qui tienne compte à la fois de la justesse ainsi que du rappel. Celui-ci est utilisable sur des classes déséquilibrées. C'est le cas de Titanic dont le nombre de morts dépasse celui des survivants.

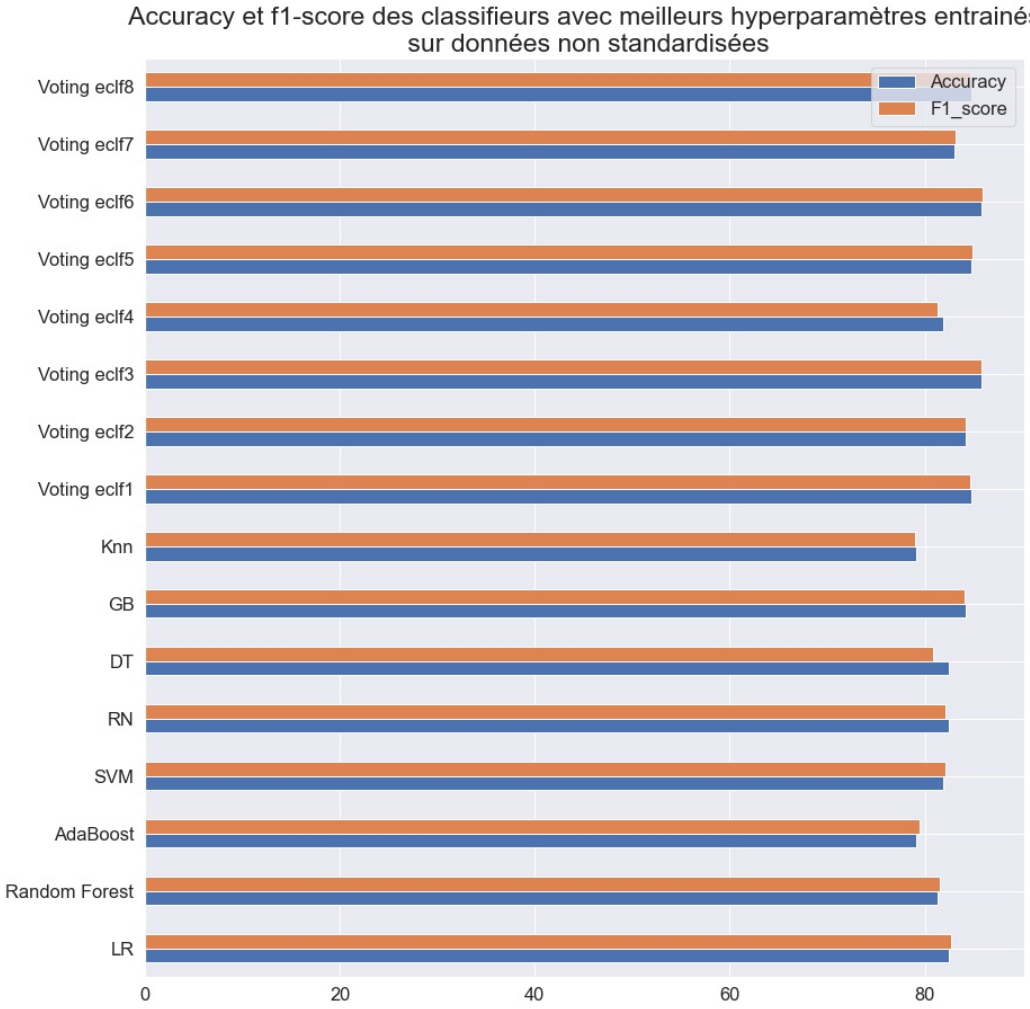
6.3.1 Justesse et F1-score des modèles sur des données non standardisées

Dans le tableau qui suit on présente par ordre décroissant du F1-score nos modèles qui sont entraînés sur des données non standardisées. On remarque que notre meilleur classifieur est le voting de type hard de SVM, Gradient boosting et Decision tree avec une justesse de 85.87% et un f1-score de 85.94%. En second lieu vient le voting de type Soft de KNN, Gradient boosting et le réseau de neurones. Cette combinaison est présente dans l'article [2]. Nous avons obtenu un f1-score (85.94%) meilleur que celui de l'article. Cependant, le KNN est le modèle le moins efficace dans notre cas d'étude avec une justesse de 79.09% et un f1-score de 79.05%.

	Accuracy	F1-score
Voting eclf6	85.875706	85.942428
Voting eclf3	85.875706	85.798732
Voting eclf5	84.745763	84.859937
Voting eclf1	84.745763	84.719306
Voting eclf8	84.745763	84.662630
Voting eclf2	84.180791	84.231856
GB	84.180791	84.124624
Voting eclf7	83.050847	83.199181
LR	82.485876	82.660039
RN	82.485876	82.158954
SVM	81.920904	82.079127
Random Forest	81.355932	81.581700
Voting eclf4	81.920904	81.338091
DT	80.790960	80.908952
AdaBoost	79.096045	79.421789
Knn	79.096045	79.059790

Figure 3: Justesse et f1-score pour données non standardisées

A partir de ce tableau, on affiche la figure ci-dessous regroupant pour chaque modèle son accuracy et le f1-score.

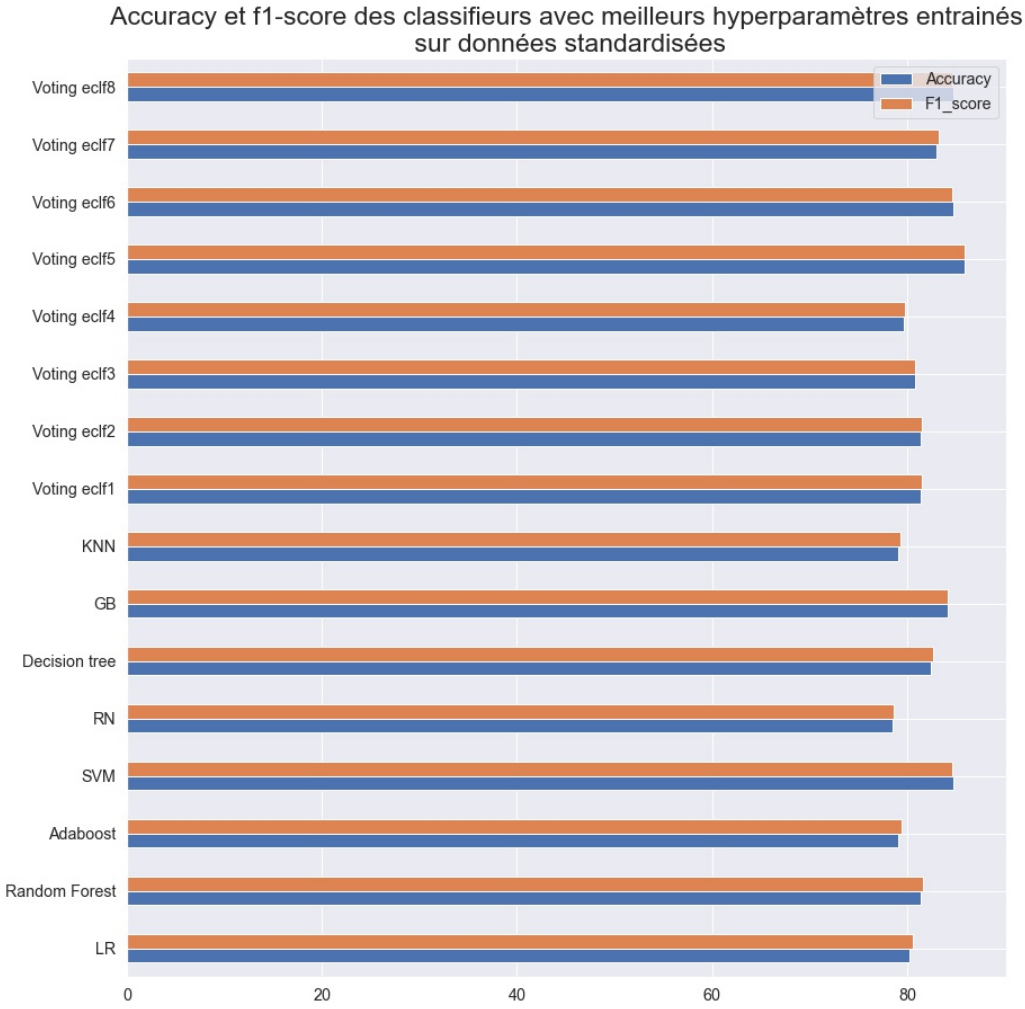


6.3.2 Justesse et F1-score des modèles sur des données standardisées

De même pour les données non standardisées, on représente dans le tableau qui suit nos modèles triés par ordre décroissant du F1-score . On remarque que notre meilleur classifieur devient le voting de type soft combinant le SVM , le Gradient boosting et Decision tree qui sont les trois meilleurs classifieurs avec une justesse de 85.87% et un f1-score de 85.89%. Vient après le SVM. Ceci signifie que la standardisation des données permet d'améliorer son accuracy. Cependant, le perceptron multicouche est le modèle le moins efficace dans notre cas d'étude avec une justesse de 78.53% et un f1-score de 78.60%.

	Accuracy	F1-score
Voting eclf5	85.875706	85.899064
SVM	84.745763	84.662630
Voting eclf6	84.745763	84.662630
Voting eclf8	84.745763	84.662630
GB	84.180791	84.124624
Voting eclf7	83.050847	83.199181
Decision tree	82.485876	82.616965
Random Forest	81.355932	81.581700
Voting eclf1	81.355932	81.541332
Voting eclf2	81.355932	81.495479
Voting eclf3	80.790960	80.852968
LR	80.225989	80.534124
Voting eclf4	79.661017	79.785949
Adaboost	79.096045	79.421789
KNN	79.096045	79.303918
RN	78.531073	78.600376

Figure 4: Justesse et f1-score pour données standardisées



7 Prédictions

Après avoir entraîné nos classifieurs, on va faire les prédictions sur l'ensemble de test, que ce soit pour les modèles entraînés sur les données non standardisées que sur les données standardisées. On se contente pour des soucis de redondance de représenter uniquement certaines prédictions utilisant des données non standardisées (le reste des prédictions se trouve dans le fichier predictions_sur_test.csv voir [1]).

	passenger number 896	passenger number 892	passenger number 1306
Logistic Regression	0	0	1
Random Forest	0	0	1
AdaBoost	1	0	1
SVM	1	0	1
RN	0	0	1
Decision tree	1	0	1
Gradient boosting	0	0	1
KNN	0	0	1
eclf1	0	0	1
eclf2	0	0	1
eclf3	0	0	1
eclf4	0	0	1
eclf5	0	0	1
eclf6	0	0	1
eclf7	0	0	1
eclf8	0	0	1

Figure 5: Justesse et f1-score pour données standardisées

Le tableau précèdent représente les prédictions de tous les classifieurs sur les passagers numéro 896, 892 et 1306. On remarque que pour le passager numéro 896, la majorité des modèles ont prédit qu'il est mort. Cependant SVM, Adaboost et l'arbre de décision ont prédit qu'il a survécu. Ceci est probablement une erreur qui peut être expliquée par leur accuracy qui est plus faible que les autres. Néanmoins, pour le passager numéro 892 et le passager numéro 1306 tous les modèles ont prédit que le premier est mort, alors que le deuxième a survécu.

8 Conclusion

L'objectif de ce projet était de classifier les passagers du Titanic en "survivants" et "morts". La première étape consistait à visualiser et interpréter les données. Par la suite, on a effectué un pré-traitement sur nos ensembles de données d'entraînement et de test afin de traiter les valeurs manquantes et les valeurs aberrantes, puis créer les nouvelles colonnes qui nous ont semblé pertinentes. Ensuite, vient l'étape d'entraînement des classifieurs d'abord avec les hyperparamètres de base pour des données non standardisées et standardisées, puis avec une recherche d'hyperparamètres et une validation croisée de 5-fold pour les deux types de données.

Le classifieur le plus performant qu'on a obtenu était le voting de type hard entraîné sur des données non standardisées de SVM, Gradient boosting et Decision tree avec une justesse de 85.87% et un f1-score de 85.94% sur les données de validation.

Ce projet nous a permis de mettre en exercice ce qu'on a appris dans le cours de forage de données et principalement les méthodes de pré-traitement des données. On a pu également utiliser plusieurs classifieurs de la bibliothèque sklearn afin d'en faire une comparaison et en choisir le plus adapté à notre cas d'étude Titanic.

Que les âmes des défunts reposent en paix !

References

[1] Walid AIT LHAJ, Kaoutar BOUHAMIDI EL ALAOUI, Andrianihary RAZAFINDRAMISA, and Eliott THOMAS. Dépot git du projet. https://github.com/bouhamidi/projet_fdd.

[2] Neytullah Acun Ekin Ekinci, Sevinç İlhan Omurca. A comparative study on machine learning techniques using titanic dataset. https://www.researchgate.net/profile/Neytullah-Acun/publication/324909545_A_Comparative_Study_on_Machine_Learning_Techniques_Using_Titanic_Dataset/links/607533bc299bf1f56d51db20/A-Comparative-Study-on-Machine-Learning-Techniques-Using-Titanic-Dataset.pdf.

[3] Lauren Clarke Shawn Cicoria John Sherlock, Manoj Muniswamaiah. Classification of titanic passenger data and chances of surviving the disaster. https://arxiv.org/ftp/arxiv/papers/1810/1810.09851.pdf.

[4] Lauren Clarke Rajni Sehga Karman Sing, Renuka Nagpa. Exploratory data analysis and machine learning on titanic disaster dataset. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9057955.