Machine Learning avec Python : Titanic ML from Disaster

MOUNIER TEBAS Teodoro

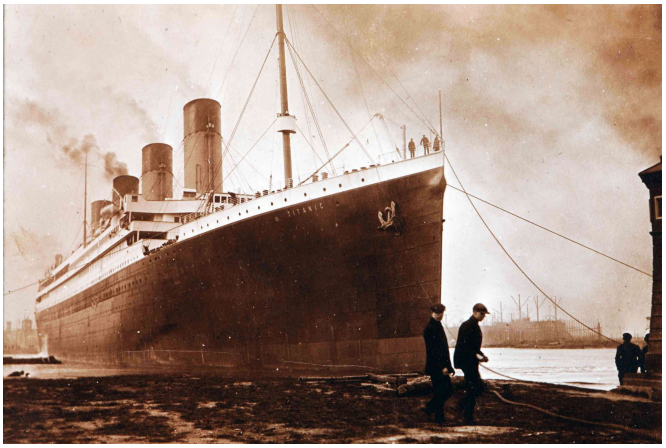
Université de Nantes

Encadré par GUICHARD Camille

Abstract

Le développement suivant résume un projet réalisé pour la matière 'Machine Learning sous Python'. L'objectif de ce projet est de prédire les survivants du Titanic à partir de la base de données publique présente sur Kaggle. Bien qu'il n'y ait pas un grand intérêt à prédire les survivants du Titanic, c'est un passage presque obligé pour tout Data scientist (ou plutôt future Data Scientist). En effet, c'est un sujet ou les données nous ont permis de mettre en pratique la théorie vue en cours. Nous allons donc expliquer la démarche suivie pour prédir le nombre de survivants du naufrage ainsi que les codes Python utilisés.

Keywords: Kaggle, Titanic, Machine Learning, RandomForest.



1. Introduction

Si nous reprenons l'introduction du Kaggle Challenge [1], le naufrage du Titanic est le naufrage le plus célèbre de l'histoire. Le 15 avril 1912, lors de son voyage inaugural, le RMS Titanic, considéré comme «insubmersible», coula après avoir heurté un iceberg. Malheureusement, il n'y avait pas assez de canots de sauvetage pour tout le monde à bord, ce qui a entraîné la mort de 1502 passagers et membres d'équipage sur 2224.

Bien que chaque passager avait une chance de survivre, il semble que certains groupes de personnes étaient plus susceptibles de survivre que d'autres.

Le défi de ce challenge est donc de construire un modèle prédictif qui répond à la question : «**Quels types de personnes étaient plus susceptibles de survivre ?**» en utilisant les données sur les passagers (le nom, l'âge, le sexe, la classe socio-économique, etc.).

Pour réaliser ce projet nous avons utilisé le langage **Python** avec comme interface un **Jupyter notebook** afin de pouvoir commenter et conclure chaque code et sortie avec du text (Markdown). Ce résumé ne fait que reprendre ce notebook en détaillant la démarche et les résultats obtenus.

2. Import et manipulations des données

Avant de commencer quoi que soit, nous avons importé les librairies nécessaires à la réalisation du projet. La librairie **Pandas** [2] qui est la librairie incontournable pour l'analyse de donnée. Avec celle-ci il faut rajouter les librairies **numpy**, **scipy**, **matplotlib** et **seaborn** pour la manipulation de données, le calcul scientifique et la visualisation des données. En ce qui concerne les modélisations, c'est avec la librairie **scikit-learn** que nous allons faire l'apprentissage statistique. Pour charger une librairie dans l'environnement python, cela se fait de la manière suivante :

```
# Data Analyse
import pandas as pd
import numpy as np
```

1. https://www.kaggle.com/c/titanic

2. **Pan**el **Da**ta **S**ystem

```
import scipy.stats as stats

# Data Viz
import matplotlib.pyplot as plt
import seaborn as sns

# Data Learn
from sklearn.(...) import (...)
```

2.1. Import des bases

Les données sont composés de deux fichiers, *Train* et *Test* que nous avons téléchargés à partir de Kaggle. Le fichier *Train* est composé de 891 lignes et 12 colonnes et le fichier *Test* de 418 lignes et 11 colonnes. La colonne manquante étant, bien entendu, la variable à prédire **Survived**. Nous avons stocké les deux fichiers *CSV* dans notre espace de travail et les avons chargés avec la librairie **pandas** de la manière suivante :

```
# Import train
train=pd.read_csv(path+"train.csv", sep=",")
# Import test
test=pd.read_csv(path+"test.csv", sep=",")
```

2.2. Compréhension des données

Nous avons commencé par indexer le *Data.Frame* avec la variable **PassengerID** ce qui nous permet d'avoir accès aux variables directement en utilisant la commande *train.columns*. Ensuite grâce à la fonction *train.info()* nous avons vérifié la composition de notre base de données ainsi que la nature des variables.

Nous avions la variable **Survived**, la variable à prédire qui indique la mort ou survie du passager avec 1 pour "survie" et 0 pour "mort". Ensuite vient la variable **Pclass** qui indique la classe des chambres du navire à trois niveaux avec 1 pour les "meilleures" cabines et 3 pour les cabines "économiques", 2 se retrouvant entre les deux. La variable **Name** donne le nom de la personne.

Les variable **Sex** et **Age** qui correspondent au sexe du passager "Male" ou "Female" et à son âge. Viennent ensuite les variables **SibSp** (Sibling and Spouse) et **Parch** (Parent and Child) qui correspondent au nombre de membres de la famille du passager de type frère, soeur, époux, épouse pour la première variable et de type père, mère, fils, filles pour la seconde.

Les variables **Ticket** et **Fare** nous donnent le numéro du ticket ainsi que son prix. Enfin il nous reste la variable **Cabin** avec le numéro de Cabin du passager et **Embarked**, le port d'embarquement du passager qui est une variable catégorielle à trois modalités : "C" (*Cherbourg*), "Q"(*Queenstown*) et "S" pour (*Southampton*).

2.3. Manipulations des données

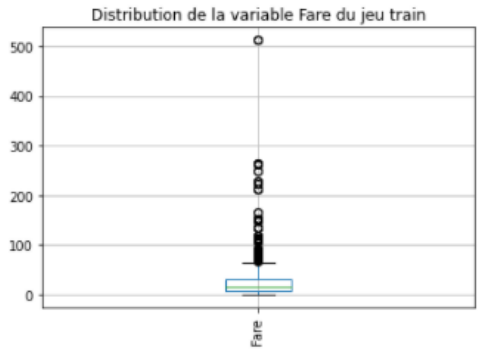
Avant les modélisations il est nécessaire de vérifier si la base comporte des **valeurs manquantes** (NA) qu'il faudrait soit, retiré de la base s'il y en a peu, soit procéder à une imputation c'est à dire remplacer les NA par la moyenne ou la médiane de la variable concernée ou encore par une prédiction de celle-ci. Nous devons vérifier aussi la présence de **points atypiques** qui, si nous ne les retirons pas, biaiseraient les estimations.

Afin d'observer le nombre de NA par variables, nous appliquons le code suivant à notre base :

```
train.isna().sum()/train.shape[0]*100
```

La variable la plus problématique est la variable **Cabin** avec 77% de NA. Cette variable est tout de même importante car elle nous indique sur quel 'pont' se trouvaient les passagers et donc s'ils étaient proches ou pas des canots de sauvetage ainsi que le 'statut' social du passager en question. Les valeurs manquantes empêchent l'utilisation de cette variable pour la modélisation, nous avons décidé de remplacer les NA en -1 pour qu'ils fassent partie d'une catégorie à part entière et ne gênent pas les méthodes d'estimation. De même la variable **Age** comporte 20% de NA, cette fois-ci, étant une variable numérique, nous avons décidé d'imputer aux valeurs manquantes la médiane pour ne pas fausser la distribution. Enfin, pour la variable **Embarked** qui n'a que 2 valeurs manquantes nous avons fait le choix de retirer les observations concernées de la base.

Pour la détection des outliers nous avons choisi de représenter graphiquement la distribution des variables quantitatives et retirer les observations vraiment extrêmes. On observe pour la variable **Fare** du jeu *train* 2 points vraiment atypiques à retirer qui correspondent à des billets de plus de 500$ tandis que la plupart des billets coûtaient une bouchée de pain.

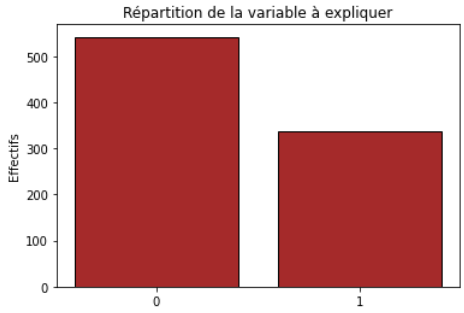


Après détection et suppression de tous les points atypiques nous nous retrouvons avec une base de 879 observations pour la base *train*.

3. Statistiques descriptives

3.1. La variable à expliquer : "Survived"

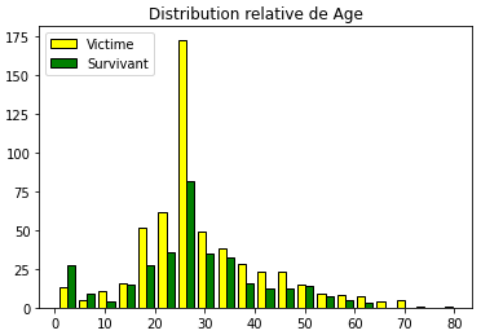
Avant toute analyse il est important de regarder les répartitions des individus au sein de chaque modalité des variables explicatives pour vérifier la répartition. Comme nous pouvons le voir par le diagramme à barres ci-dessous, les personnes ayant survécu sont au nombre de 337 ce qui représente ainsi 38.3% de la base.



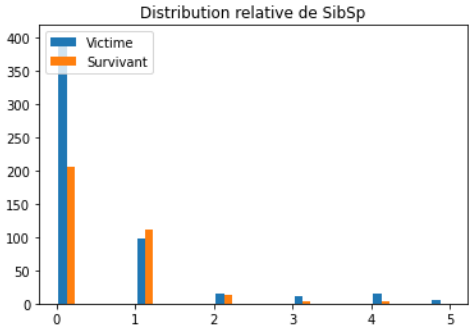
Maintenant, nous allons faire de même avec les variables explicatives avant de les inclure dans nos modèles. Nous allons observer la distribution de chaque variable en fonction de celle à expliquer. Nous rajouterons pour les variables quantitatives l'étude des corrélations.

3.2. Les variables explicatives quantitatives

- **Age** : À partir du plot ci-dessous nous pouvons observer que la majorité des passagers du Titanic avait entre 20 et 35 ans et c'est cette même tranche d'âge qui a été la plus touchée par le naufrage. En regardant attentivement le graphique nous nous sommes rendu compte qu'il y avait beaucoup de survivants parmi les enfants ce qui nous amènera à la création de la variable **is_child** ultérieurement.

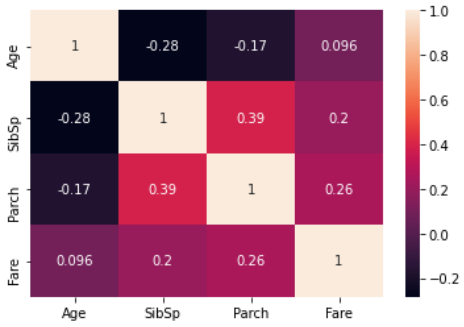


- **SibSp/Parch** : Les distributions de **SibSp** et **Parch** sont quasiment identiques et nous pouvons constater que la majorité des passagers voyageaient seuls.



- **Fare** : On peut constater que les personnes ayant payé le moins cher leur traversée sont aussi celles qui ont été les principales victimes du naufrage. Cette variable confirme le triste constat d'une inégalité face aux chances de survie pour les classes les plus pauvres.

En guise de conclusion des variables quantitatives, se sont, d'une manière générale les plus jeunes (en proportion) qui ont davantage survécu ainsi que les plus riches ou ceux ayant beaucoup de proches à bord, ce qui n'est pas surprenant tout compte fait.

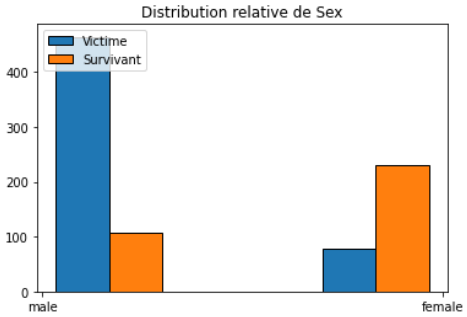


Avant de passer aux variables qualitatives nous pouvons jeter un coup d'oeil à la **matrice de corrélations** présente ci-dessus. Nous avons la confirmation de notre intuition d'une corrélation assez forte entre **Parch** et **SibSp** égale à 0.39. Cependant, la corrélation étant inférieure à 0.5 nous n'allons pas devoir sélectionner une seule des deux variables pour pouvoir l'estimer via une régression logistique. Cette contrainte ne s'applique pas pour les algorithmes de Machine Learning.

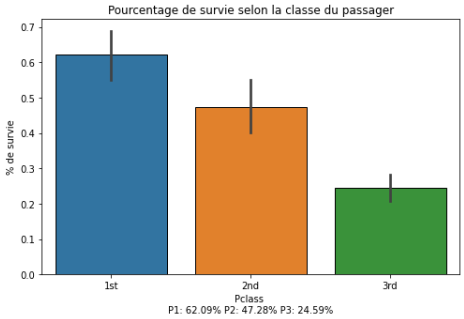
3.3. Les variables explicatives qualitatives

Maintenant, nous allons analyser les variables qualitatives présentes dans les bases de données. Elles sont au nombre de 3, "**Sex**", "**Pclass**" et "**Embarked**". Comme pour la section précédente nous allons observer les effectifs des passagers par modalité pour chaque variable de la base *train*. Commençons par la variable **Sex**.

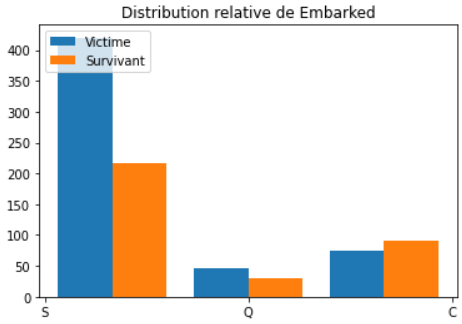
- **Sex** : Cette variable nous informe du genre du passager. À partir du diagramme en barres ci-dessous nous pouvons en retirer deux conclusions. La première c'est qu'il y a presque deux fois plus d'hommes que de femmes à bord. La seconde c'est que ce sont les femmes qui ont principalement survécu.



- **Pclass** : On peut en tirer les informations suivantes, une forte majorité des victimes provient de la troisième classe ; a contrario, les survivants proviennent plutôt de la première classe. Les passagers de la seconde classe sont plus compliqués à classifier correctement car cette classe paraît moins discriminante.



- **Embarked** : Pour terminer sur les variables catégorielles nous avons accès aux données du port d'embarquement du passager. L'histogramme montre que la majorité des passagers ont embarqué à Southampton et ce sont eux aussi qui ont été les principales victimes du naufrage.



3.4. Création de Variables

Certaines variables de la base de données nous ont posé quelques difficultés car nous ne savions pas comment les prendre en compte pour les estimations. C'était le cas de la variable textuelle **Name** avec les noms des passagers, et où la manipulation de chaînes de caractères était nécessaire pour son traitement avant estimation. Pour la variable **Cabin**, avec 70% de valeurs manquantes nous avions décidé dans un premier temps de la supprimer puis nous avons tenté d'en tirer de l'information tout de même. Au final nous avons créé trois variables supplémentaires :

- **is_child** à partir de **Age** qui répond à la question "est-ce un enfant ?" ce qui permet de mieux faire parler la variable **Age**. Cette nouvelle variable s'est tout simplement construite à partir du code suivant :

```
train['is_child'] = train.Age < 8
```

- **title** à partir de **Name**. On a pu constater que pour chaque passager, le "titre" de l'individu (Mr, Mrs, Miss etc) est précisé. C'est une information complémentaire aux variables **Sex** et **Pclass**. Nous avons donc créé une nouvelle variable **title** contenant seulement le titre de passager. Pour cela nous avons utilisé la fonction suivante :

```
train['title'] = train['Name']...
.map(lambda x : x.split(',')[1]...
.split('.')[0])
```

Cette commande permet de récupérer la chaîne de caractère comprise entre la virgule et le point où se trouve le titre pour chacune des observations. C'est la partie fluotée de la variable **Name** suivante :

```
PassengerId
1                              Braund, Mr. Owen Harris
2    Cumings, Mrs. John Bradley (Florence Briggs Th...
3                               Heikkinen, Miss. Laina
4         Futrelle, Mrs. Jacques Heath (Lily May Peel)
5                             Allen, Mr. William Henry
6                                     Moran, Mr. James
7                              McCarthy, Mr. Timothy J
8                       Palsson, Master. Gosta Leonard
9    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
10                  Nasser, Mrs. Nicholas (Adele Achem)
11                      Sandstrom, Miss. Marguerite Rut
12                             Bonnell, Miss. Elizabeth
13                       Saundercock, Mr. William Henry
14                          Andersson, Mr. Anders Johan
15                  Vestrom, Miss. Hulda Amanda Adolfina
Name: Name, dtype: object
```

- **Cabin** est une variable composée de beaucoup de valeurs manquantes comme nous avions pu le constater précédemment. Cependant, lorsqu'il n'y a pas de NaN, la variable Cabin donne toujours une lettre puis un nombre, la lettre correspondant au 'pont', et le nombre au numéro de cabine. C'est une variable qui nous semblait pertinente si on suppose que les différentes distances entre les cabines et les canots

de sauvetage ont pu déterminer le sort des passagers. Le code nous a permis d'imputer la valeur -1 aux Nan afin de pouvoir rajouter cette variable aux estimations et de récupérer seulement la lettre correspondant au "pont".

```
train['Cabin'] = train.Cabin...
.map(lambda x : x[0] ...
if not pd.isnull(x) else -1)
```

4. Tableau récapitulatif

Variables	V.Quantitative	V.Qualitative	Création
Age	√		
SibSp	√		
Parch	√		
Fare	√		
Sex		√	
Pclass		√	
Embarked		√	
is_child		√	√
title		√	√
Cabin		√	√
ticket	Non incluse		

Table 1: Tableau récapitulatif des variables explicatives

Le tableau précédent récapitule l'ensemble des variables qui vont être incluses dans nos modélisations. Comme vous pouvez le constater, la variable **Ticket** n'a pas été retenue car nous n'avons pas pris le temps de l'étudier suffisamment pour savoir comment l'inclure. En effet, nous ne savions pas comment le numéro du ticket pouvait expliquer les chances de survivre au naufrage.

5. Préparation de la base avant modélisation

La dernière étape est la préparation des données pour les algorithmes d'apprentissage automatique. Cela se fait par la **standardisation** des variables quantitatives et la **dummisation** des variables qualitatives. Ces deux transformations permettent de s'assurer un apprentissage correct pour les méthodes de Machine Learning (DecisionTree/RandomForest/Boosting) et des coefficients interprétables et non-biaisés pour les méthodes de régression (Logistic regression).

5.1. Standardisation des variables quantitatives

La **standardisation** ou *recalibrage des variables* (scaling en anglais) est nécessaire avant l'utilisation des algorithmes d'apprentissage automatique. En effet, les algorithmes ne fonctionnent pas très bien lorsque les variables numériques ont des échelles très différentes. Il existe deux méthodes pour faire cela, la **transformation min-max** afin que toutes les valeurs se retrouvent entre 0 et 1, il suffit pour cela de soustraire par la valeur minimum et diviser par la maximum. La seconde méthode c'est la **normalisation** afin que les distributions de chacune des variables soient de moyenne nulle et de variance égale à 1. Pour cela il faut retrancher la moyenne et diviser par l'écart-type pour chaque observation de la variable. Avec Scikit-Learn nous avons normalisé nos variables numériques de la manière suivante :

```
from sklearn import preprocessing
X_scaled_train = preprocessing.scale(X_quanti)
```

5.2. Dummisation des variables qualitatives

Le traitement des variables catégorielles cette fois se fait en créant une nouvelle variable pour chaque modalité. Ces variables étant composées de 0 ou 1 si la modalité est vérifiée ou non, sont aussi appelées variables *fictives* (**dummy** en anglais). Pour ne pas avoir de problème de multicolinéarité[3], nous retirerons à chaque fois une des variables fictives qui correspondra à la modalité cible. Nous avons appliqué la fonction suivante :

```
...=pd.get_dummies(transform[["Sex","..."]],
drop_first = True)
```

Maintenant que nous avons bien cerné le sujet, les données que nous avions, et appliqué les transformations nécessaires aux variables explicatives, nous pouvons enfin passer aux modélisations.

6. Modélisations

Dans cette section nous avons réalisé 5 modélisations. Une modélisation économétrique "classique" avec la **régression logistique** et des méthodes de ML d'apprentissage statistique avec un **arbre de décision**, **une forêt aléatoire**, un **bagging** et enfin un **gradient boosting**. Il est nécessaire d'avoir un échantillon d'apprentissage et un échantillon test afin de mesurer la qualité d'ajustement des modèles et pouvoir comparer ainsi les méthodes entre elles. Dans notre cas les jeux de données *train* et *test* venaient déjà de deux fichiers différents, c'est pourquoi nous n'avons pas eu besoin de diviser notre base. Le code python d'application des méthodes d'apprentissage automatique varie très peu. Nous allons donc développer la méthode et les choix des mesures de sélection pour un seul algorithme puis nous résumerons les résultats pour les autres algorithmes d'apprentissage dans un tableau récapitulatif.

3. Lorsqu'une des variables explicatives d'un modèle est une combinaison linéaire d'une ou plusieurs autres variables explicatives introduites dans le même modèle. Il y a donc colinéarité lorsque deux ou plusieurs variables mesurent la "même" chose.

6.1. *Méthodologie*

La **régression logistique**, même si en général moins performante en termes de prédiction, est très intéressante lorsqu'il s'agit de comprendre l'impact des variables explicatives sur la variable **Survived** à expliquer. En effet, la régression logistique estime des coefficients que nous pouvons récupérer pour connaître les effets positifs ou négatifs de chaque variable ainsi que leur poids (importance). L'interprétation des coefficients est très simple :

- un poids positif augmente la probabilité de survivre ;
- à l'inverse, un poids négatif la diminue ;
- Enfin, quand un poids est proche de 0, cela signifie que l'effet de la variable à expliquer est peu discriminant par rapport à la modalité à prédire.

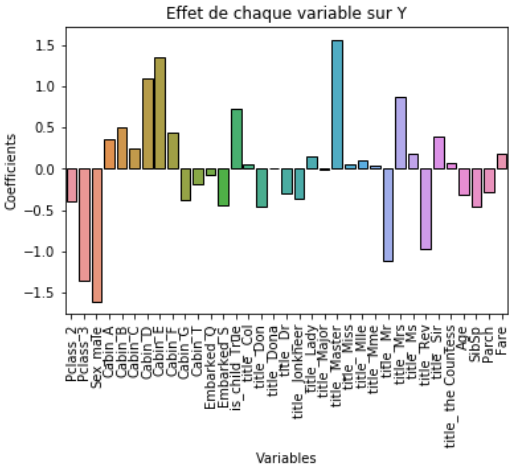
Nous avons donc appliqué une régression logistique aux données d'entraînement avec le code suivant :

```
# On applique la regression logistique
from sklearn.linear_model import LogisticRe-
gression
lr = LogisticRegression()
lr.fit(X_train, Y_train)
```

Il est ensuite important de regarder le taux de bien prédit qui se mesure par la métrique du **F1_score**[4], qui permet d'être moins sensible aux déséquilibres de classe ce qui est le cas ici avec la modalité *victimes* sur-représentée. Nous rajoutons à cela une **validation croisée** par la méthode **K-fold**[5] avec k fixé à 3 pour un modèle plus fiable grâce à l'échantillonnage.

```
from sklearn.model_selection import cross-
_val_score
from sklearn.metrics import f1_score
scores = cross_val_score(lr, X_train,
Y_train, cv=3, scoring='f1_macro')
```

Nous avons donc obtenu un f1_score égal à 0.8271 suite à une prédiction sur les mêmes données à partir desquelles le modèle s'est entraîné. Ce même taux après échantillonage par Cross-Validation est de 0.8108 ce qui est très proche des 0.8271, nous pouvons donc être sûr qu'il n'y a pas de sur-apprentissage, ou **overfitting**[6] en anglais. Nous pouvons donc conclure positivement sur la stabilité de la régression logistique. Nous pouvons observer le poids des variables dans le graphique suivant.



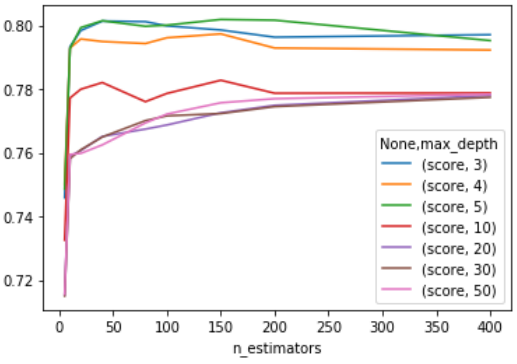
6.2. *Tableau récapitulatif*

Après la régression logistique nous avons appliqué d'autres méthodes mais cette fois-ci d'apprentissage automatique (Arbres/Forêt/Boosting/Bagging). Le tableau suivant récapitule les résultats obtenus pour chaque modélisation.

Modèles	*F1_score*	*CV*
Reg. logistique	82.71%	81.08%
CART	98.3%	75.00%
CART.Opt	77.75%	75.03%
Forêt	96.73%	78.82%
Forêt.Opt	82.47%	79.99%
Bagging	93.34%	80.97%
Boosting	98.31%	79.65%
Boosting.Opt	/	82.35%

TABLE 2: Résultats des modélisations

La meilleure qualité de prévision est donc de 82.35 obtenue par cross-validation avec le **Gradient Boosting optimisé** grâce à la fonction *Grid Search* présente dans *sklearn.model_selection*. Nous pouvons observer comment s'est réalisée la recherche des hyperparmètres optimaux dans le plot suivant :

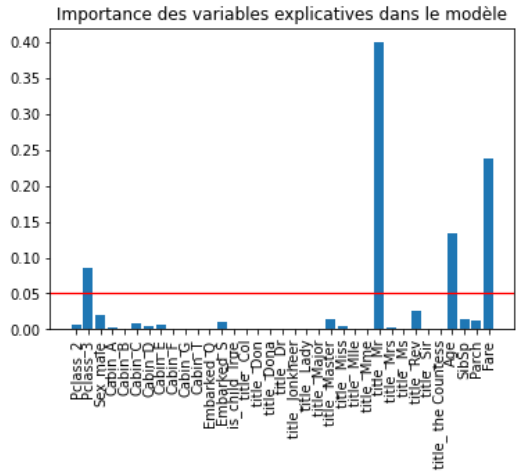


4. Combinaison du **Recall** et de la **Precision**, indicateurs calculés à partir d'une matrice de confusion.

5. Méthode d'échantillonage qui consiste à créer **K-blocs**, où le modèle s'entraîne sur **k-1 blocs** et il se teste sur le bloc restant (**1 bloc**). Cette opération est répétée **k-fois** et le score du modèle est la moyenne des **k-scores** obtenues.

6. Modèle qui s'ajuste trop aux données d'entrées et qui n'est donc pas généralisable (mauvais face à de nouvelles données.

Contrairement à la régression logistique, le gradient boosting est une méthode ensembliste utilisant la descente du gradient pour optimiser une fonction de perte afin d'affecter le juste poids aux observations pour la construction du modèle suivant. Nous ne sommes pas face à une régression donc n'avons pas de coefficients estimés. En revanche, il est possible de mesurer l'importance des variables explicatives dans le modèle. Le diagramme en barres ci-dessous montre quelles ont été les variables les plus significatives dans l'apprentissage du modèle.



7. Conclusion

La réalisation de ce projet amène à la conclusion plutôt évidente que les chances de survie au naufrage du Titanic n'ont pas été identiques pour tout les passagers. Les statistiques descriptives ainsi que les modélisations ont mis en évidence l'inégalité qu'ont eu les passagers face à l'accès au canots de sauvetages. Les variables telles que le prix du billet, l'âge des passagers, le titre ou encore la classe ont été très importantes dans le destin des passagers. Je pense que les statistiques et les modèles sont venus confirmer le triste constat que la richesse à jouer un rôle important dans les chances de survie. Enfin, il est intéressant de constater que face à l'obligation de choisir qui doit survivre ou non, l'importance pour la "survie de la descendance" s'est faite. En effet les femmes et les enfants ont été des privilégiés pour être sauvés.

L'objectif final était tout de même de prédire le nombre de survivants de l'échantillon train et d'obtenir un score sous kaggle. Avec notre meilleur modèle qui correspond au gradient boosting optimisé, nous obtenons un score de bien prédit de 0.7488. Nous pouvons donc conclure que nous avons réussi à partir des données sur les passagers à prédire le funeste destin des 3/4.