



UNIVERSITÉ DE
SHERBROOKE

FACULTÉ DES SCIENCES

Rapport du projet de session du cours de Techniques d'apprentissage

<i>Auteur</i>	<i>Email</i>	<i>Id</i>
Zo Lalaina Yannick RAHARIJAONA	rahz1802@usherbrooke.ca	18 138 837
Kendadi Mohammed	kenm2001@usherbrooke.ca	19 145 177
Nabil BENJAA	benn2613@USherbrooke.ca	19 148 469

Présenté à :
M.Pierre Marc Jodoin

12 décembre 2019

Table des matières

1	Introduction :	2
2	Formulation du problème :	2
2.1	Description du problème :	2
2.2	Démarche scientifique :	2
3	Expérimentations :	2
3.1	Compréhension du dataset :	2
3.2	Choix des modèles :	4
3.3	Première approche :	4
3.3.1	Strict minimum de pré-traitement de donnée :	4
3.3.2	Entraînement des modèles :	5
3.3.3	Cross validation :	5
3.3.4	Comparaison des modèles et soumission des résultats dans kaggle : .	5
3.4	Deuxième approche :	5
3.4.1	Pré-traitement des données :	5
3.4.2	Entraînement des modèles :	9
3.4.3	Cross validation :	9
3.4.4	Comparaison des modèles et soumission des résultats dans kaggle : .	9
4	Résultats et interprétations :	9
4.1	Approche sans pré-traitement des données :	9
4.2	Approche avec pré-traitement des données :	12
5	Conclusions :	15
A	Annexe :	15
A.1	Organisation du code :	15
A.2	Gestion du projet :	16

Table des figures

1	Balancement du dataset en fonction de la classe cible "survival"	3
2	Nombre de valeur nulle par variable	4
3	Distribution de la probabilité de survie sachant la variable "Sex"	6
4	Distribution de la fréquence de survie en fonction de la variable "Pclass" . . .	7
5	Distribution de la fréquence de survie en fonction de la variable "Parch" . . .	8
6	Distribution de la probabilité de survie en sachant de la variable "SibSp" . .	8
7	Comparaison des Classifieurs dans la première approche	10
8	Matrice de confusion de Logistic Regression après validation	11
9	Courbe ROC de Logistic Regression après validation	11
10	Resultat de la premiere approche sur kaggle	12
11	Comparaison des Classifier dans la seconde approche	13
12	Matrice de confusion de Logistic Regression après validation	14
13	Courbe ROC de Logistic Regression après validation	14
14	Resultat de la seconde approche sur kaggle	15
15	Organisation du code	16

Liste des tableaux

1	Dictionnaire de donnée	3
2	Résultats de la première approche	10
3	Résultats de la deuxième approche	12

1 Introduction :

Dans un cadre de projet de session, on nous demande d'expérimenter des méthodes de classification sur un dataset choisi dans Kaggle. En outre, un outil collaboratif comme git est sollicité pour favoriser le travail en équipe.

2 Formulation du problème :

2.1 Description du problème :

Dans ce projet, on expérimente plusieurs modèles de classification sur le dataset Titanic pour sélectionner le ou les meilleurs modèles qui généralisent mieux sur les données de test. Pour ce faire, on implémentera sept modèles variés (quelques modèles à faible capacité et des modèles à forte capacité). Sans oublier de tenir compte des étapes nécessaires à savoir le pré-traitement des données et les validation croisées ainsi que les autres démarches scientifiques pour obtenir de bons modèles.

2.2 Démarche scientifique :

Pour résoudre le problème, on va faire recours à deux approches :

- Premièrement, on ne fait que **le strict minimum de pré-traitement des données** afin d'utiliser directement les modèles de classification. On procède après à la "cross-validation" pour avoir des métriques pour chaque modèles.
Le meilleur parmi ces modèles sera utilisé pour générer le fichier à soumettre dans Kaggle. Ainsi, le résultat retourné par kaggle va nous servir de "**benchmark**" que l'on va essayer d'améliorer par la suite.
- Ensuite, on procède au **pré-traitement des données** suivi de la "cross-validation". Une fois ces étapes terminées, on compare les modèles. Le classifieur le plus performant sera utilisé pour générer le fichier de soumission à remettre dans Kaggle.

3 Expérimentations :

3.1 Compréhension du dataset :

Avant toute chose, il est nécessaire de voir en profondeur le dataset pour essayer de le comprendre. Il est toujours bon de connaître chacune des "variables" du dataset. Le tableau suivant décrit une par une les variables du dataset.

Variable	Définition	Type
survival	La classe cible, elle indique si la personne survie ou non à l'accident	Catégorique
pclass	Classe de billet	Catégorique
sex	Sexe de la personne	Catégorique
age	Age de la personne	Continue
sibsp	Nombre de frères et sœurs ou conjoint(s) à bord du Titanic	Continue
parch	Nombre de parents ou enfants à bord du Titanic	Continue
ticket	Numéro de billet	Continue
fare	Tarif du billet	Continue
cabin	Numero de cabin	Catégorique
embarked	Port d'Embarquement	Catégorique

Tableau 1: Dictionnaire de donnée

Une fois qu'on comprend le dataset, il est temps de voir le balancement des étiquettes de classe. Ici, notre classe prend juste une valeur binaire donc on a un problème de classification binaire.

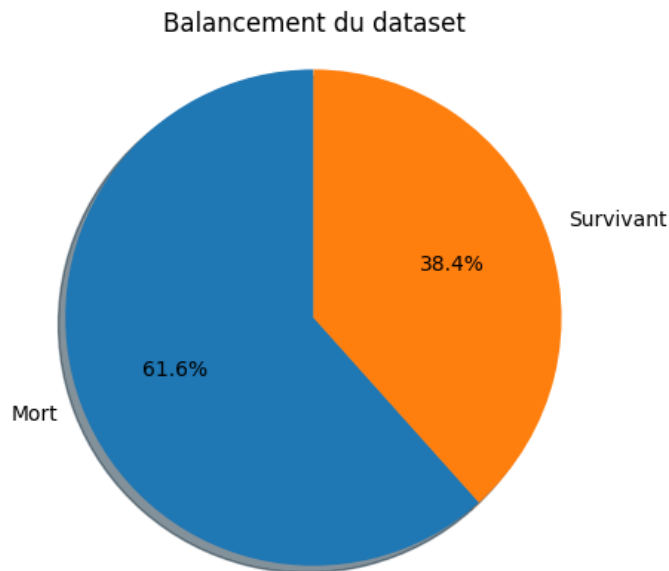


FIGURE 1 – Balancement du dataset en fonction de la classe cible "survival"

D'après la figure 1, on a un dataset un peu debalancé de presque 60 contre 40 pourcent.

3.2 Choix des modèles :

Pour avoir de bon résultat, on a choisi d'utiliser des modèles variés. Autant de classifieur qui sont linéaire que de classifieur non linéaire. Ainsi que des modèle combiné comme le "Bagging classifier" et "Adaboost classifier". Les sept modèles utilisés sont les suivants :

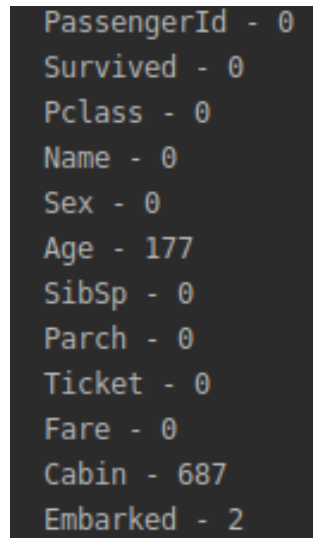
- Adaboost Classifier
- Bagging Classifier
- Decision tree Classifier
- Fully connected
- Logistic regression
- Random Forest
- SVM

3.3 Première approche :

Comme cité précédemment dans la sous-section 2.2, la première approche consiste à faire le strict minimum de pré-traitement du dataset, faire la cross-validation puis sélectionner le modèle le plus performant afin de l'évaluer dans Kaggle.

3.3.1 Strict minimum de pré-traitement de donnée :

On ne peut pas appliquer directement les algorithmes de classification sur des datasets qui contient des valeurs nulles ou des valeurs comme les chaîne de caractère. La figure 2 illustre les valeurs nulles groupées par variable.



```
PassengerId - 0
Survived - 0
Pclass - 0
Name - 0
Sex - 0
Age - 177
SibSp - 0
Parch - 0
Ticket - 0
Fare - 0
Cabin - 687
Embarked - 2
```

FIGURE 2 – Nombre de valeur nulle par variable

Telle que illustrée par la figure 2, "Age", "Cabin" et "Embarked" contiennent des valeurs nulles. La plus simple façon de remplir ces valeurs nuls est de les remplir par les moyennes de chaque colonne. Après remplissage des valeurs, il est évident de supprimer les colonnes comme "Name" et "PassengerId" ainsi que de changer les variables catégoriques qui prend des valeurs comme les caractères par des valeurs entiers numériques.

3.3.2 Entraînement des modèles :

Une fois l'étape précédente terminée, on peut procéder directement à l'entraînement des modèles. Juste dans le but d'avoir des premières métriques. Certes, cette étape d'entraînement permet de sélectionner les bons paramètres de chaque algorithme de classification mais chaque modèle a des "hyperparameter" à choisir astucieusement.

3.3.3 Cross validation :

La cross-validation permet de sélectionner les bons "hyperparameter" qui vont de pair avec chaque modèle de classification. Comme stratégie de cross-validation, on a utilisé la "k fold cross validation" que l'on peut faire facilement à partir de la méthode "gridsearchcv" (qui prend comme paramètre "stratifiedkfold") dans la bibliothèque "sklearn".

3.3.4 Comparaison des modèles et soumission des résultats dans kaggle :

Après que l'on ait fini de sélectionner des bons "hyperparameter", on peut procéder à la prédiction des données de test et l'enregistrer dans un fichier de soumission que l'on va soumettre directement dans Kaggle.

3.4 Deuxième approche :

3.4.1 Pré-traitement des données :

C'est cette section qui diffère la deuxième approche à la première. Au cours de notre réalisation, nous avons pris le temps de nettoyer les données, transformer des variables, créer de nouvelle variable, voir la corrélation qui existe entre chaque variable et la cible puis supprimer les variables qui semblent ne pas être très significatives.

- Analyse de corrélation en pivot :

On a émis quelques hypothèses sur quelques variables comme le "Sex", "Pclass", "SibSp", "Parch" qui peuvent être en corrélation avec la cible. Ces hypothèses sont émit juste par intuition.

La figure 3 montre la corrélation qui existe entre la variable 'Sex' et la classe cible.

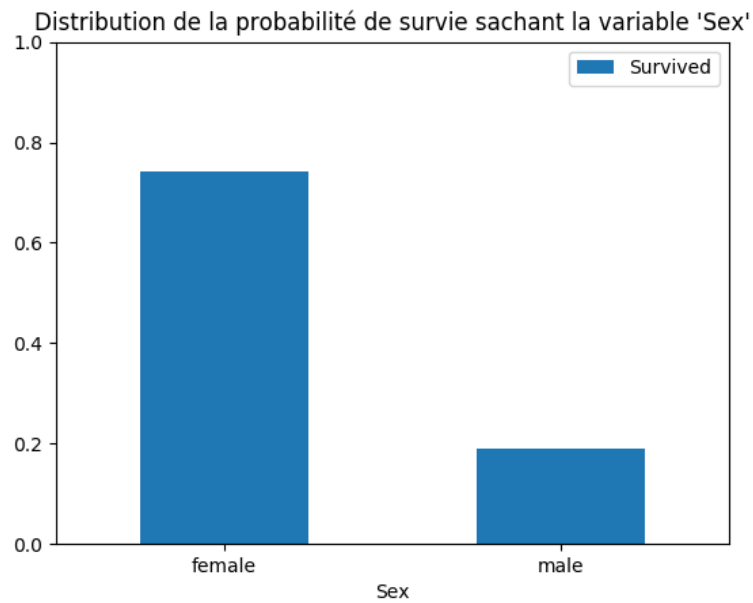


FIGURE 3 – Distribution de la probabilité de survie sachant la variable "Sex"

On peut affirmer l'hypothèse que le sexe influe beaucoup sur la "survie". Une femme a plus de probabilité de survivre qu'un homme pendant l'accident.

La figure ci-dessous illustre la corrélation qui existe entre la variable 'Pclass' et la classe cible.

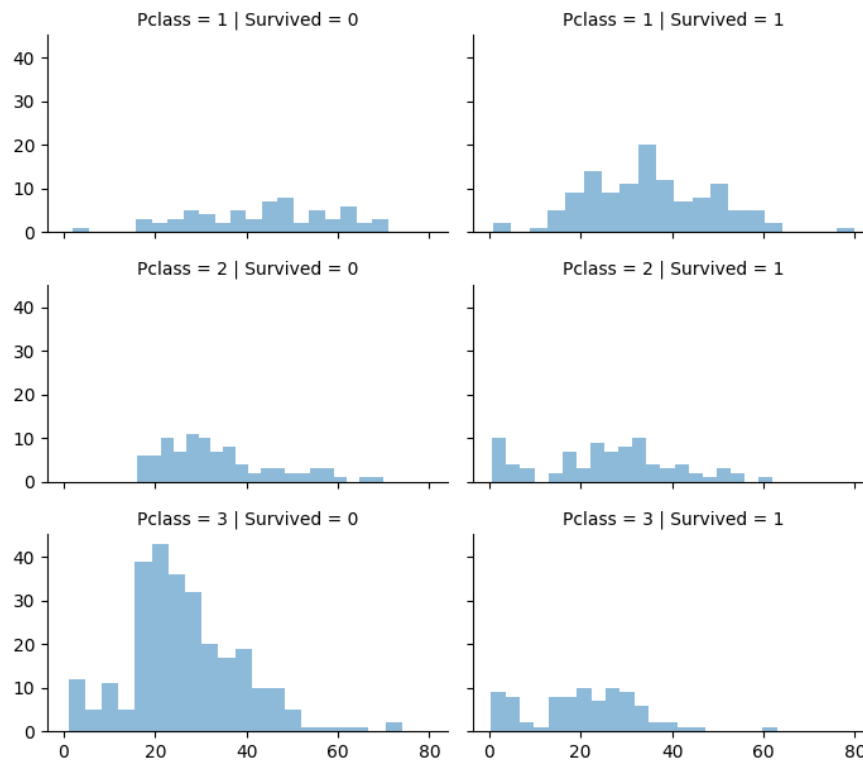


FIGURE 4 – Distribution de la fréquence de survie en fonction de la variable "Pclass"

On peut déduire que les personnes de la classe 1 ont de chances de survie et que les personnes de la classe 3 ont moins de chances de survivre après l'accident.

La figure ci-dessous illustre la corrélation qui existe entre la variable 'Parch' et la classe cible.

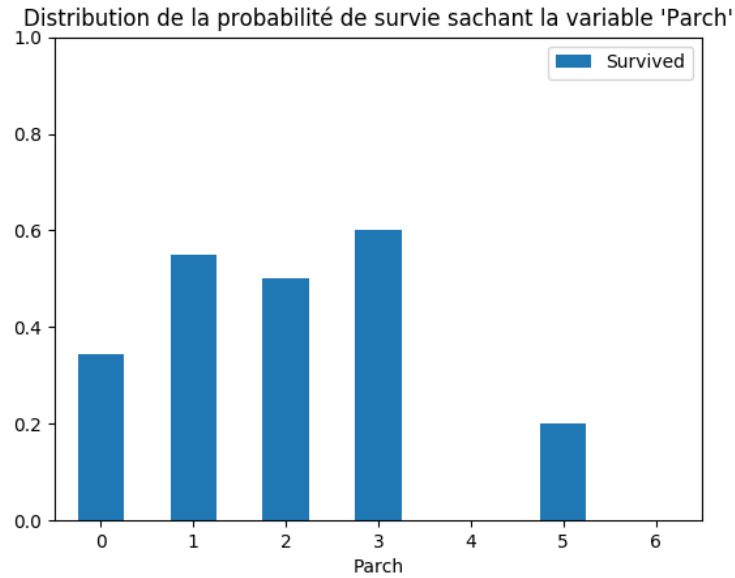


FIGURE 5 – Distribution de la fréquence de survie en fonction de la variable "Parch"

La figure ci-dessous illustre la corrélation qui existe entre la variable 'Sibsp' et la classe cible.

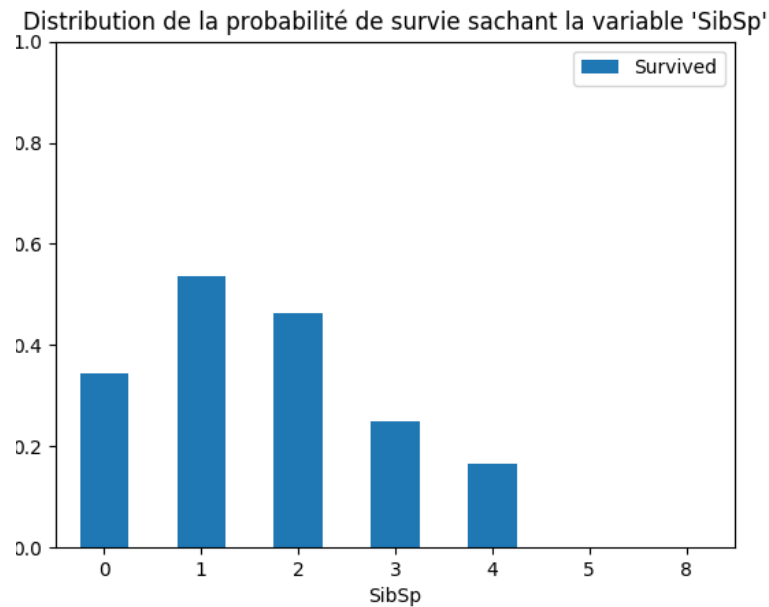


FIGURE 6 – Distribution de la probabilité de survie en sachant de la variable "SibSp"

La figure 5 et 6 montrent en générale que plus on est accompagné plus on risque de ne pas survivre. Mais quand même il existe quelques exceptions cela est due peut être à l'emplacement de la personne à bord du bateau ou de son âge.

— Transformation des variables :

On a transformé par exemple la variable âge en intervalle qui sera codé numériquement

- pour avoir des catégories d'âge comme enfant, adulte et vieux.
- Création de nouvelle variable :
Comme l'on a vu précédemment "SibSp" et "Parch" influe beaucoup sur la variable la classe de prédiction donc on peu combiner les deux pour créer une variable "FamilySize" pour voir le nombre des membres de famille qui voyage avec chaque personne.
- Suppression des variables :
On peu supprimer maintenant les variables comme le "Embarkment" ainsi que les variables utilisées pour former les nouvelle variables comme "SibSp" "Parch" et "Age".
- Après transformation des données, les données sont à peu près de même ordre de grandeur (vue que ce sont tous des variables discrètes) donc transformer la distribution des données en centrée réduite est jugée comme pas très nécessaire.

3.4.2 Entraînement des modèles :

Comme dans la première approche après le pré-traitement des données on peut procéder à l'entraînement pour avoir des modèles permettant de généraliser le dataset.

3.4.3 Cross validation :

Après avoir entraîner les modèles, il est convenable de choisir les "hyperparameter" adéquats en gardant la même stratégie de cross validation utilisée lors de la première approche.

3.4.4 Comparaison des modèles et soumission des résultats dans kaggle :

Après que l'on ait fini de sélectionner les bons "hyperparameter", on fait la prédiction sur les données de test pour régénérer le fichier à soumettre dans Kaggle.

4 Résultats et interprétations :

Nos résultats se reposent sur trois metriques : l'accuracy, la matrice de confusion, ainsi que l'AUC ROC. Les résultats des deux approches seront affichés dans ce rapport sous forme de tableau. Seul les deux modèles performants pour les approches respectifs seront affichés sous forme de graphes par soucis d'encombrement.

4.1 Approche sans pré-traitement des données :

Le tableau 2 résume les scores de Train, Test et Validation Test (Score de chaque model cross-validées sur les données de tests).

Model	Train		Test		Validation Test	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Logistic Regression	80.7	86.0	93.8	98.0	93.0	98.0
Random Forest	98.09	100.0	81.6	89.0	88.0	95.0
SVM	95.85	97.0	64.1	71.0	90.0	93.0
Fully Connected	81.9	87.0	92.8	98.0	91.0	93.0
Adaboost	83.8	90.0	89.7	92.0	81.0	83.0
Decision Tree	98.0	100.0	77.2	77.0	90.0	91.0
Bagging Classifier	92.6	98.0	86.4	94.0	88.0	96.0

Tableau 2: Résultats de la première approche

Si l'on compare les résultats de Test et de la Validation Test (Test à partir d'un model validé) on remarque directement l'importance de la "cross-validation". Car les tests faits à partir des model validés sont plus crédibles.

La figure 7 résume la performance des différents "Classifieur" utilisés. Elle illustre bien que "Logistic Regression" est le "Classifieur" le plus performant.

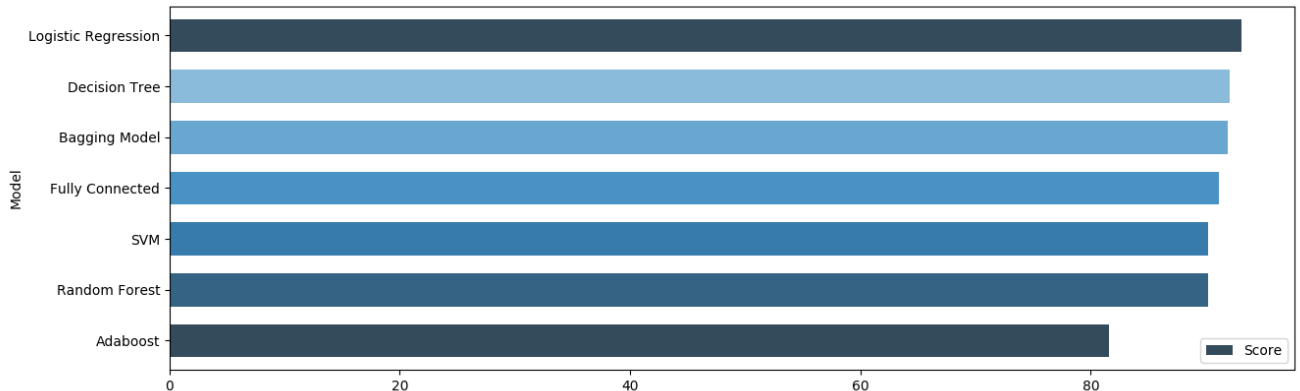


FIGURE 7 – Comparaison des Classifieurs dans la première approche

Si l'on compare chaque "model" dans la figure 7, les "model" à faible capacité qui permettent la séparation linéaire des données performe mieux que les modèles à forte capacité. Plus précisément, Logistic Regression, Decision-Tree performent mieux que SVM, Fully Connected, Random Forest et Adaboost.

Il se peut que les données soient alors linéairement séparable (c'est juste une hypothèse qu'on émet à première vue). Mais l'on ne peut encore rien conclure car ces résultats sont des résultats "naïves" sur des données non pré-traités. Pour tirer conclusion, on doit pré-traités les données.

Voyons en profondeur d'autres métriques pour évaluer "Logistic Regression" après validation croisée pour assurer la fiabilité de nos résultats.

CONFUSION MATRIX VISUALIZATION AFTER VALIDATION

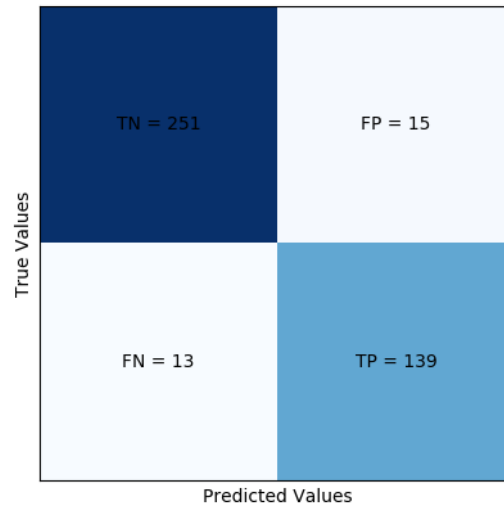


FIGURE 8 – Matrice de confusion de Logistic Regression après validation

On remarque directement que le nombre de "TN" et "TP" sont très élevés par rapport aux nombre de "FP" et "FN". Ce qui est aussi une bonne indication que "Logistic Regression" généralise bien les données.

La figure 9 illustre la courbe ROC pour evaluer "Logistic Regression".

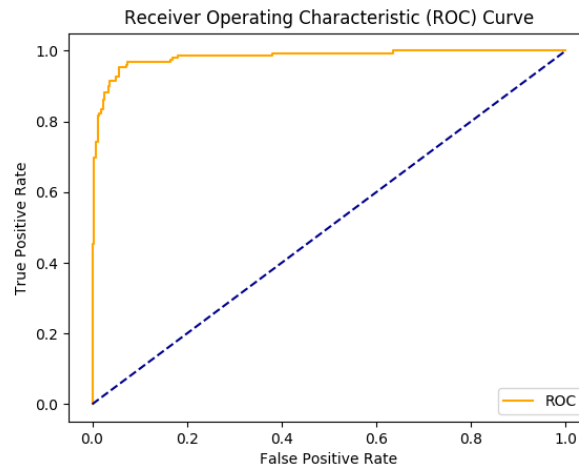


FIGURE 9 – Courbe ROC de Logistic Regression après validation

L'aire en dessous de la courbe est très développés, elle montre bien que le taux de vrai positive est très élevés. Ce qui est logique car le Taux de TP est largement supérieur au taux de FP dans la matrice de confusion. Ce aussi démontre la bonne performance de "Logistic Regression".

La figure 10 montre le résultat sur Kaggle de la première approche.

13674	RAHARIJAONAZoLalainaYan...		0.76076	1	-10s
-------	----------------------------	-------------------------------------------------------------------------------------	---------	---	------

FIGURE 10 – Resultat de la premiere approche sur kaggle

Comme prévu, on va prendre comme "benchmark" à améliorer le résultat du Classifier qui performe le plus. Dans notre cas, les résultats de "Logistic Regression" sont prises comme mesure de référence à améliorer. On s'attend que ce "benchmark" s'améliore après avoir fait les pré-traitement des données.

4.2 Approche avec pré-traitement des données :

Dans cette seconde approche, on vise à améliorer le "benchmark" obtenu lors de la première approche. Dans un premier temps, le tableau 3 récapitule les résultats lors de l'expérimentation.

Model	Train		Test		Validation Test	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Logistic Regression	78.1	86.0	98.3	99.0	98.0	99.0
Random Forest	86.8	93.0	84.7	93.0	88.0	96.0
SVM	86.3	88.0	86.7	89.0	83.	94.0
Fully Connected	81.7	87.0	93.4	98.0	91.0	96.0
Adaboost	81.7	87.0	90.1	96.0	84.0	88.0
Decision Tree	86.7	94.0	84.0	86.0	87.0	95.0
Bagging Classifier	84.51	91.0	88.52	95.0	89.0	97.0

Tableau 3: Résultats de la deuxième approche

Si l'on compare les résultats de la validation de "Logistic Regression" dans cette approche par rapport à la première, on voit qu'elle s'améliore. Sauf que les données de Train et Test diminuent beaucoup parce-que dans la première approche il existe des variables comme "Parch" qui n'ont pas de même ordre de grandeur que les autres qui influe beaucoup sur le résultat. Et que d'autre variable significative que l'on a supprimer aussi peu fausser les résultats lors de la première approche. Les pré-traitements faites sur les données joue beaucoup sur les résultats de l'entraînement et de prédiction.

La figure 10, montre encore que "Logistic Regression" performe mieux que les autres méthodes de classification.

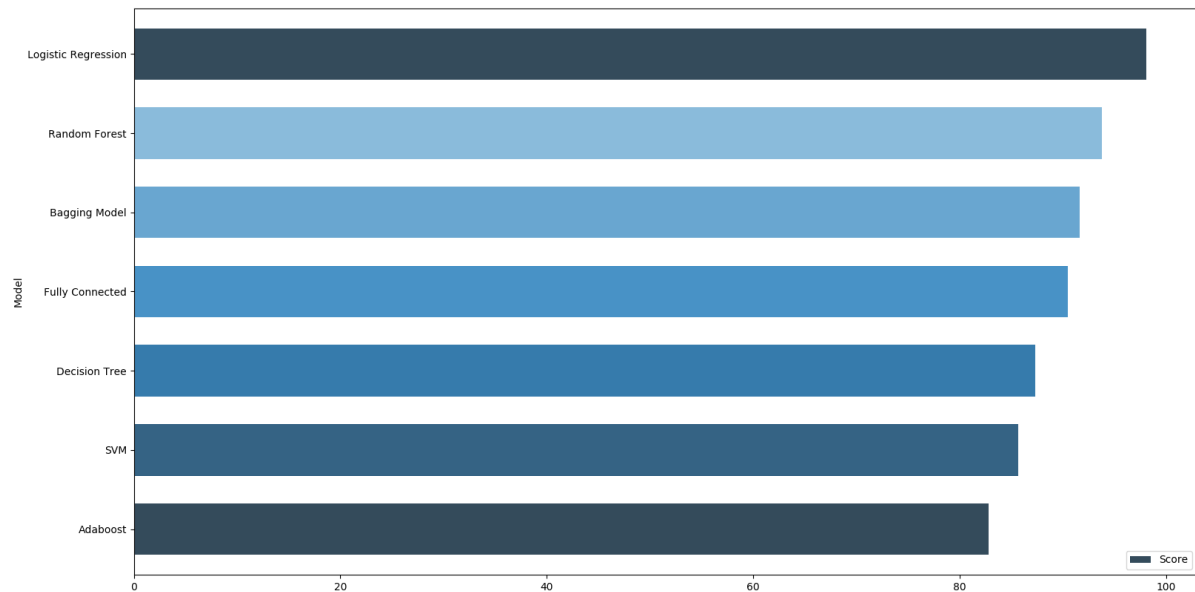


FIGURE 11 – Comparaison des Classifier dans la seconde approche

Maintenant, on peut affirmer que les données sont plus ou moins linéairement séparable car des model simple comme "Logistic Regression" permet de généraliser les données. Il se peut donc que le pré-traitement des données ont permis de mieux séparés les données linéairement. Ainsi les model à fortes capacités comme SVM, Fully-connected, ainsi que les models combinés ont du mal à performer car ils ont de forte variance car ils tendent à chercher des "décision boundary" beaucoup plus complexe.

Pour souligner nos résultats, voyons les résultats des matrices de confusion et de la courbe ROC.

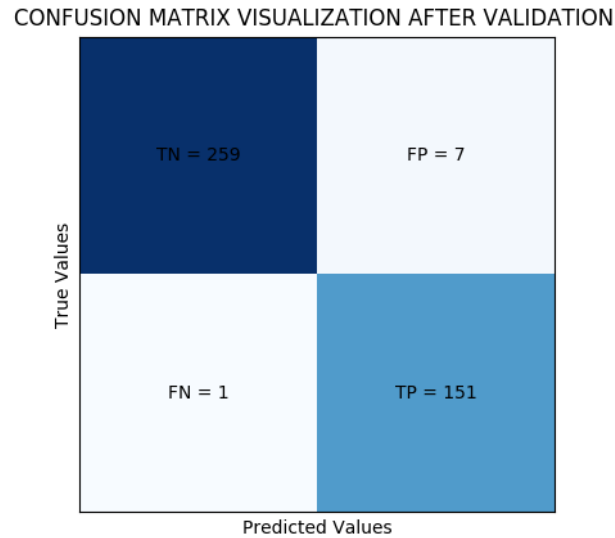


FIGURE 12 – Matrice de confusion de Logistic Regression après validation

Si l'on regarde très attentivement la matrice de confusion et l'on compare avec la matrice de confusion dans la première approche, on voit bien que nos résultats se sont améliorés car le nombre de TN et TP ont augmentés. Une forte taux de TP et de TN explique bien la forte "accuracy" que nous avons.

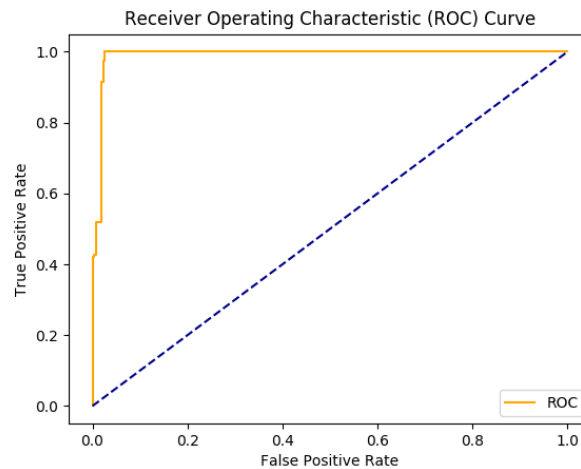


FIGURE 13 – Courbe ROC de Logistic Regression après validation

Comme le nombre de TP est beaucoup plus grand que le nombre de FN or :

$$TPR = TP / (TP + FN) \quad (1)$$

ce qui implique que le TPR est **élevé**.

A l'inverse le nombre de TN est beaucoup plus grand que le nombre de FP or :

$$FPR = FP / (FP + TN) \quad (2)$$

ce qui implique que le FPR est **moindre**.

En comparant donc le TPR et le FPR, on en deduit que $TPR \gg FPR$. Ce qui explique la grandeur de l'aire sous la courbe (AUC).

Voyons maintenant notre résultat de soumission sur kaggle.

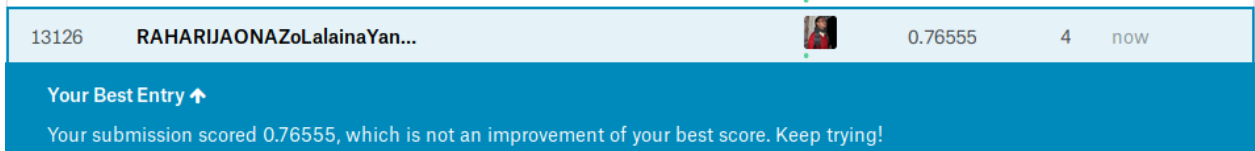


FIGURE 14 – Resultat de la seconde approche sur kaggle

La figure 14 montre que nous sommes passé du score de 0.760 à 0.765 sur le leaderboard de Kaggle. Certes, la plateforme de kaggle ne voit pas cela comme une très grande amélioration mais la soumission faite est notre "best entry". On a atteint ici notre but d'améliorer notre "benchmark". Ce qui souligne que notre démarche scientifique est bonne. D'où, l'efficacité de la seconde approche par rapport à la première.

5 Conclusions :

En conclusion, l'important dans l'élaboration d'un projet de Machine Learning est d'être astucieux que ce soit au niveau des pré-traitement des données ou au niveau de l'élaboration des Algorithmes de classification. Les connaissances statistiques ainsi que la connaissance du domaine sont nécessaires pour pouvoir faire de bon pré-traitement des données. Mais il est tout de fois indispensable de maîtriser les fondements théoriques des algorithmes de Machine Learning qu'on utilise.

A Annexe :

A.1 Organisation du code :

La figure ci dessus montre l'organisation de notre code.

- **classifiers** : dossier contenant les classifieurs utilisés dans le code. Il existe une classe `abstractClassifier` qui est un classifieur parent dont les autres héritent.
- **cross validation** : dossier contenant la classe qui fait la cross validation
- **data utils** : dossier contenant toutes les opérations faites sur le data.
- **datasets** : dossier contenant le dataset.
- **metrics** : dossier contenant les classes de metrics utilisés.
- **model summary** : dossier contenant les fichiers qui resument les modèles.
- **main.py** : fichier central qui exécute indépendamment les différents modèles avec différentes configurations.

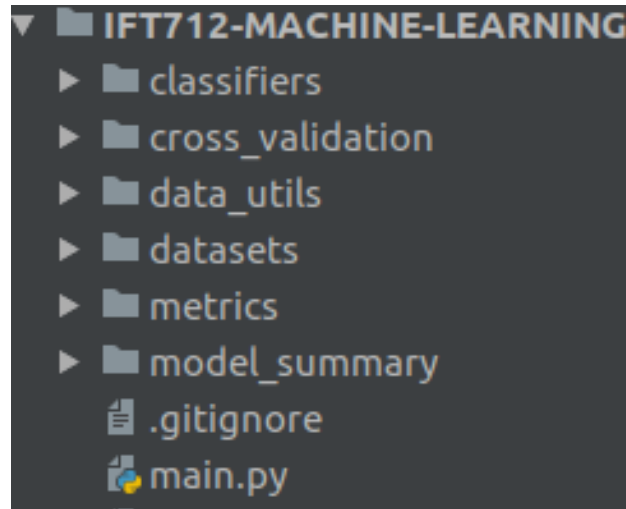


FIGURE 15 – Organisation du code

A.2 Gestion du projet :

On a choisi "git" comme outil collaboratif. Du coup, on a créé un "repository" dans "github" pour stocker le projet. Au cours du développement, on essaie de suivre les "Best practice" imposés par "gitflow" comme la création des branches, faire des "commit" et encore effectuer des "pull request" ainsi que des "merge request". Sans oublier d'utiliser "Issues" pour planifier les tâches à faire.