Titanic
Machine Learning from Disaster

Projet de *Algos ML*

Code disponible sur Github : *ici*

Remerciement à M Hamara pour l'enseignement de cette UE

Biéchy Lucas

0 Choix des données

Le choix de mon jeu de données s'est porté sur "Titanic - Machine Learning from Disaster", disponible sur Kaggle. Deux raisons ont motivé cette décision. Tout d'abord, les données sont relativement simples à appréhender, puisqu'elles se présentent sous forme de tableaux de valeurs train et test, à l'instar de celles que nous utilisons en cours. De plus, la plateforme Kaggle offre un environnement propice à l'apprentissage pour les étudiants en intelligence artificielle, ce qui a renforcé mon choix.

Le jeu de données train se compose d'un tableau de 891 lignes et 11 colonnes portant les noms suivants : **Survived** (la variable à prédire, indiquant si la personne a survécu ou non, avec 1 pour "survécu" et 0 pour "non survécu"), **Pclass** (classe du ticket), **Name**, **Sex** (sexe de la personne), **Age**, **SibSp** (nombre de frères/soeurs ou conjoints à bord), **Parch** (nombre de parents ou d'enfants à bord), **Ticket** (numéro du ticket), **Fare** (prix du billet), **Cabin** (numéro de cabine) et **Embarked** (port d'embarquement). Chaque passager est identifié par un identifiant unique. Le jeu de données test, quant à lui, est constitué de 418 lignes pour les mêmes 11 colonnes, à l'exception de la colonne **Survived** qui est logiquement absente.

1 Exploratory Data Analysis

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

TABLE 1 – Cinq premières valeurs du dataset train

Dans le tableau 1, on constate la présence de valeurs manquantes (`NaN`) dans le jeu de données. Il est donc nécessaire de vérifier le pourcentage de valeurs manquantes par variables :

Variable	**Taux de valeurs manquantes (%)**
Survived	0.00%
Pclass	0.00%
Name	0.00%
Sex	0.00%
Age	19.87%
SibSp	0.00%
Parch	0.00%
Ticket	0.00%
Fare	0.00%
Cabin	77.10%
Embarked	0.22%

TABLE 2 – Pourcentage de `NaN` par variables

En examinant le pourcentage de `NaN` par variables présents dans le tableau 2, on constate que **Cabin** a un taux de 77% de valeurs manquantes, ce qui est considérablement élevé pour qu'elle soit représentative, donc nous allons la supprimer pour la suite de l'analyse.

Les autres variables ont des taux de `NaN` moins importants. Nous pouvons donc effectuer une première visualisation des variables numériques en fonction de la survie de l'individu.

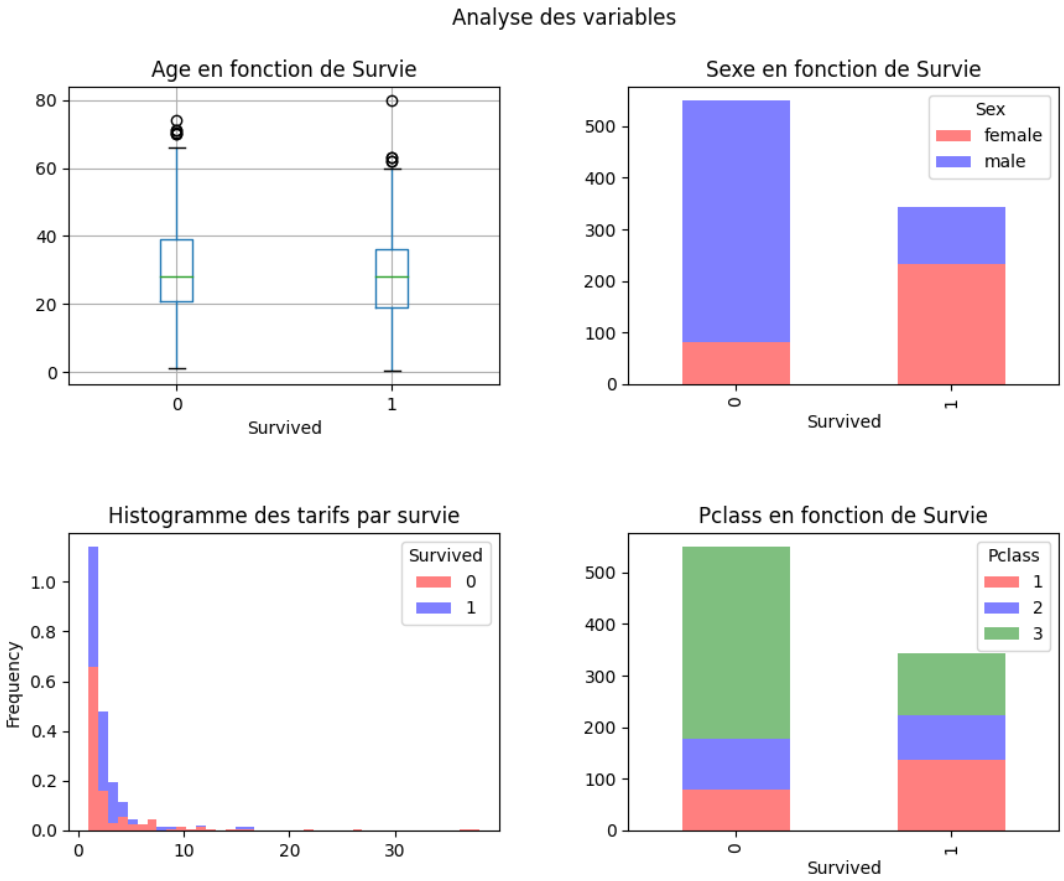

FIGURE 1.1 – Analyse des variables

Les graphiques de la figure 1.1 montrent que la survie des passagers du Titanic semble être influencée par plusieurs facteurs. En effet, le ratio de décès par rapport aux survivants est nettement inférieur à 1 pour les femmes, tandis qu'il est supérieur à 1 pour les hommes. De même, on observe que la proportion de survivants est plus élevée dans la classe 1 que dans la classe 3, tandis que la classe 2 se situe à peu près à égalité entre les deux. Cependant, il n'est pas clair pour le moment que la survie soit directement corrélée au prix du billet, comme l'indique la répartition de l'histogramme. De même, l'âge moyen des survivants ne diffère pas beaucoup de celui des victimes, étant d'environ 30 ans dans les deux cas. Examinons la matrice de covariance pour essayer de trouver des liens linéaires entre **Survived** et d'autres variables.

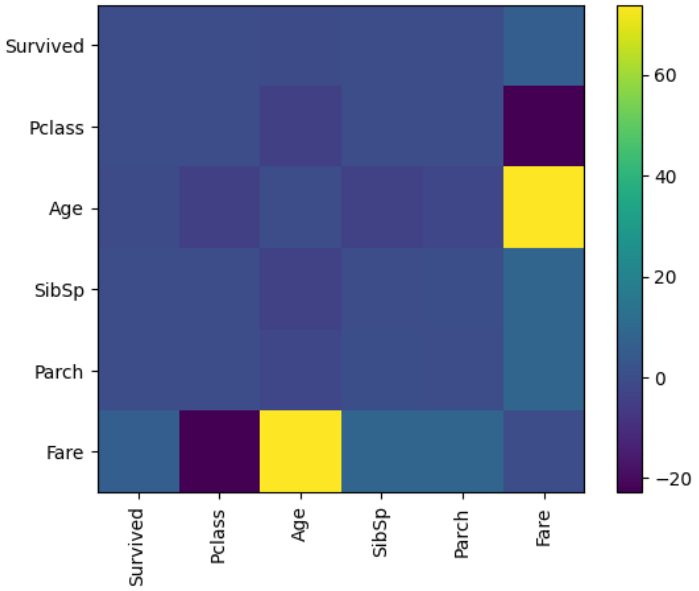

FIGURE 1.2 – Matrice de covariance

Dans cette figure 1.2 représentant la matrice de covariance, il ne semble n'y avoir aucune covariance significative entre le paramètre **Survived** et les autres. On retrouve par contre que plus une personne est vieille plus elle achète son billet cher et que plus la classe du billet augmente plus sa valeur diminue. Nous avons exploré les variables de manière non exhaustive à travers différentes visualisations, telles que les histogrammes, les boxplots et les matrices de covariance. Bien que nous n'ayons pas trouvé de relations linéaires significatives entre la variable **Survived** et les autres variables, nous avons observé des tendances intéressantes, telles que la corrélation entre l'âge et le tarif du billet ainsi que la relation inverse entre la classe du billet et la valeur du tarif. Nous pouvons maintenant passer à la prochaine partie de notre analyse.

2 Features Engineering

Cette section permet l'application des méthodes de Feature Engineering vues en cours. Une copie de l'ensemble d'entraînement est créée afin de pouvoir comparer les résultats obtenus avec et sans l'utilisation de ces méthodes.

De ce fait, le tarif du billet sera normalisé car on voit bien qu'à travers l'histogramme de la figure 1.1, cette variable ne suit pas une loi normale mais est plus proche d'une loi de poisson ou de chi-2. Les valeurs extrêmes peuvent donc influencé dans certains modèles qu'on utilisera (tous hors RandomForest). On remplace aussi les `NaN` par la moyenne pour **Age** ou par la valeur la plus fréquente pour **Embarked** car certains modèles n'acceptent pas de `NaN` en entrés. Enfin, ne savant pas faire de features engineering sur des strings aussi complexes que les noms et les tickets, les variables **Name** et **Ticket** seront arbitrairement supprimées.

De plus, il semble approprié d'appliquer un encodage one-hot aux variables **Embarked** et **Pclass** car elles ont un nombre fini et restreint de valeurs possibles. En revanche, la variable **Sex** sera encodée binairement car elle n'a que deux valeurs possibles (homme ou femme) comme les variables **Survived**, **SibSp** et **Parch**. On supprime aussi une des colonnes créée par chaque one-hot encoding, cela ne conduit pas à une perte d'informations et permet de garder la matrice identifiable. Le tableau 3 montre les premières valeurs du dataset train qui sera utilisé lors de l'entraînement des modèles.

PassengerId	Survived	Age	SibSp	Parch	Fare	Sex	Pclass 1	Pclass 3	Embarked Q	Embarked S
1	0	-0.530005	1	0	-0.502163	1	0	1	0	1
2	1	0.571430	1	0	0.786404	0	1	0	0	0
3	1	-0.254646	0	0	-0.488580	0	0	1	0	1
4	1	0.364911	1	0	0.420494	0	1	0	0	1
5	0	0.364911	0	0	-0.486064	1	0	1	0	1

TABLE 3 – Cinq premières valeurs du dataset train après features engineering

3 Benchmerks

La procédure suivie pour chaque section de modèle est la même :

0. Si nécessaire, une recherche des hyperparamètres optimaux du modèle pour la prédiction est effectuée par validation croisée (dans ce cas, le modèle est annoté avec `cv` et les précisions sont disponibles dans le code).
1. Le modèle est ajusté aux données d'entraînement.
2. Les prédictions sont effectuées sur les données d'entraînement en utilisant le modèle ajusté.
3. Les résidus $predictions - targets$ et le taux de mal classés sont calculés.
4. Les résidus sont représentés graphiquement en fonction de l'identifiant du passager (index).
5. La matrice de confusion est affichée graphiquement.
6. La courbe ROC est tracée et affichée graphiquement avec en légende l'aire sous la courbe.

3.1 Linear Model cv

Il est probable que les résultats obtenus ne soient pas optimaux car, comme le montre la figure suivante, il n'y a pas de corrélation linéaire évidente entre **Survived** et certaines de ses variables, ce qui est normale car nous sommes dans un cas de classification. Cela peut poser problème pour les modèles de régression linéaire qui sont sensibles à ce type de relation.

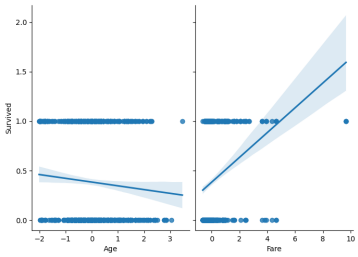

FIGURE 3.3 – Régression linéaire de **Survived** en fonction de **Age** et **Fare**

Taux de mal classés : 18.41 %

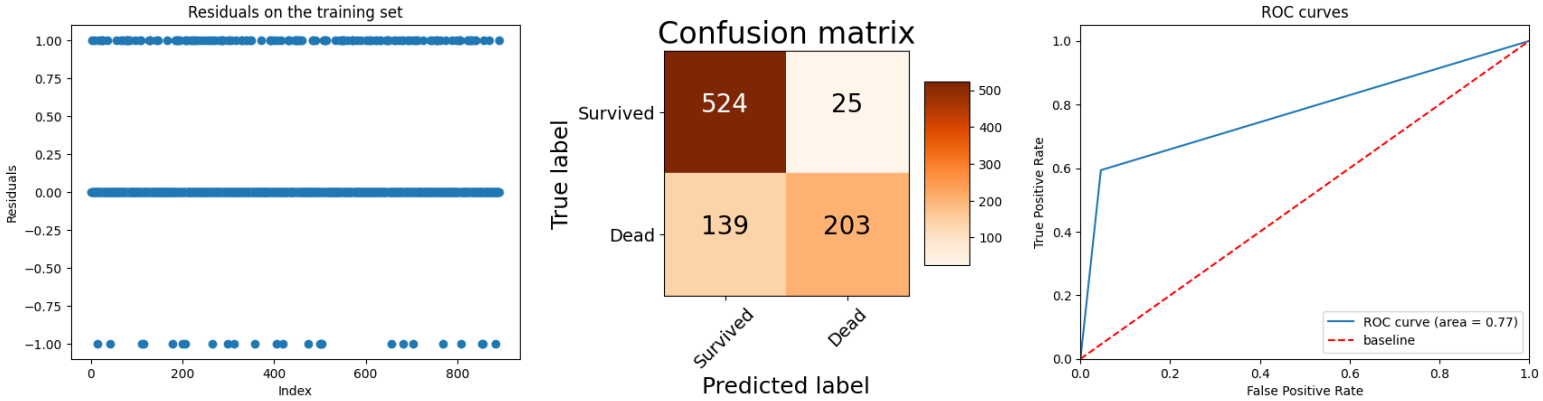

FIGURE 3.4 – Analyse de la régression linéaire

Le taux de mal classés de 18% est étonnamment satisfaisant. Cette performance pourra servir de référence de performance minimale pour les autres modèles. Cependant on voit que notre modèle est biaisé car les faux négatifs (139) sont beaucoup plus nombreux que les faux positifs (25). Il est donc important d'explorer d'autres types de modèles plus adaptés à cette situation.

3.2 Support Vector Machine

Taux de mal classés : 9.32 %

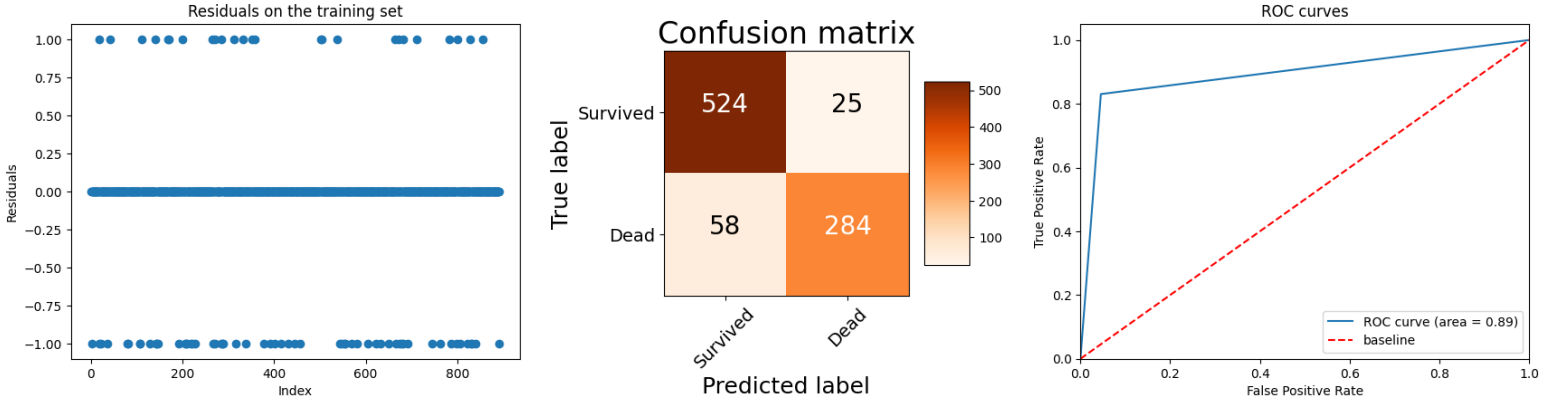

FIGURE 3.5 – Analyse du la classification par Support Vector Machine

Le SVM présente une nette amélioration par rapport à la régression linéaire, avec seulement 25 faux positifs et une réduction du biais, car le nombre de faux négatifs est désormais de 58. Un modèle sans biais est caractérisé par un nombre de faux positifs proche de celui de faux négatifs. La courbe ROC confirme la qualité du modèle avec une aire sous la courbe de 0,89, nettement supérieure à 0,5. Enfin, le taux de mal classés est déjà inférieur à 10%.

3.3 Random Forest

3.3.1 Classique

Taux de mal classés : 1.80 %

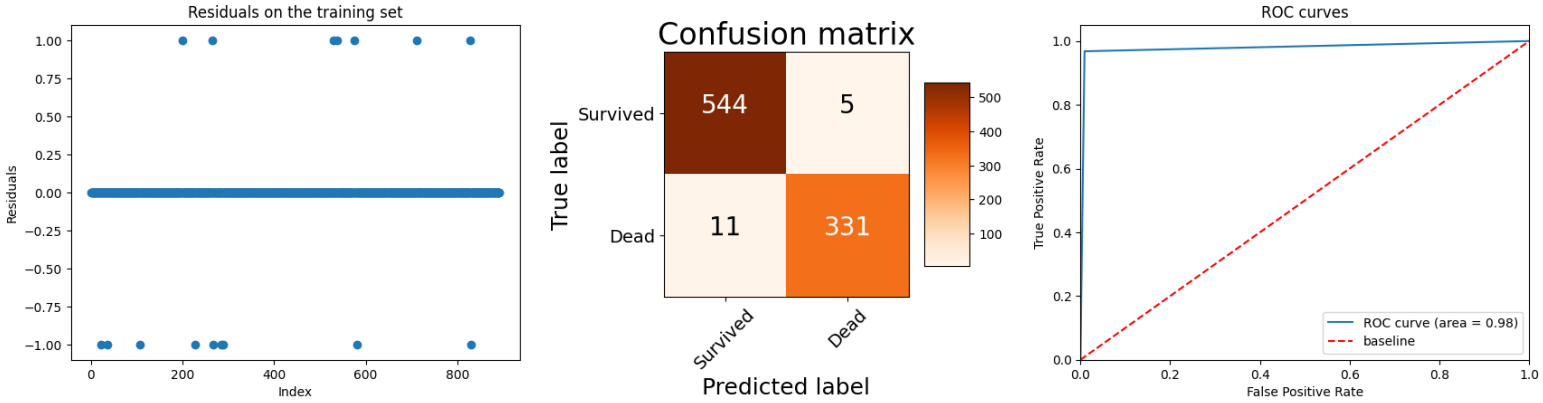

FIGURE 3.6 – Analyse du la classification par Random Forest

Une amélioration significative des performances est observée, la courbe ROC et la matrice de confusion sont presque parfaites avec une aire sous la courbe de 0.98 et un faible écart entre le nombre de faux positifs et de faux négatifs. De plus, il est possible de représenter graphiquement les variables significatives de la Random Forest ainsi que son taux de classification correct en fonction du nombre d'arbres utilisés.

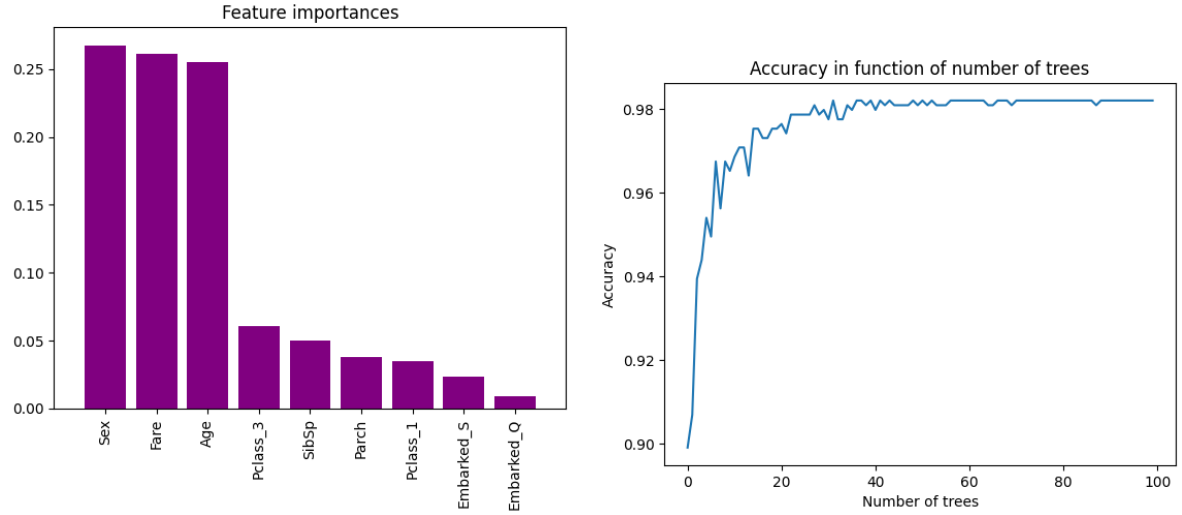

FIGURE 3.7 – Analyse de la RandomForest

Cependant, bien que notre taux de mal classés soit à un niveau record de 1.80%, il est encore possible d'améliorer notre modèle en utilisant une validation croisée.

3.3.2 Optimisation de la Random Forest `cv`

Taux de mal classés : 11.56 %

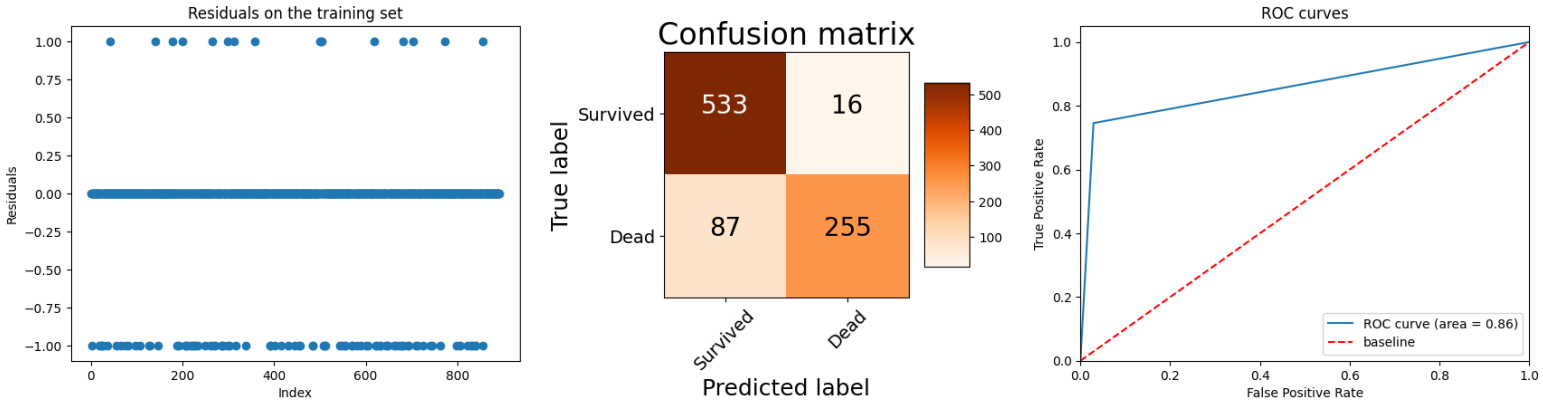

FIGURE 3.8 – Analyse du la classification par Random Forest

Il est étonnant que l'optimisation des hyperparamètres par Cross Validation ait conduit à une diminution de la précision de notre modèle (11.56% de mal classés), d'autant plus que les paramètres initiaux étaient compris dans les plages testées lors de la Cross Validation. Cela pourrait s'expliquer par le fait que la fonction *RandomForestClassifier()* effectue déjà une sorte de Cross Validation sur les paramètres non spécifiés, et le fait de leur donner une fourchette pourrait rendre le modèle moins performant. Il faudrait approfondir cette hypothèse. Le graphique 3.9 nous montre également que la variable **Sexe** est devenue quasi exclusivement déterminante pour la classification. Cette observation est étrange car le bagging est censé éviter ce type de phénomène.

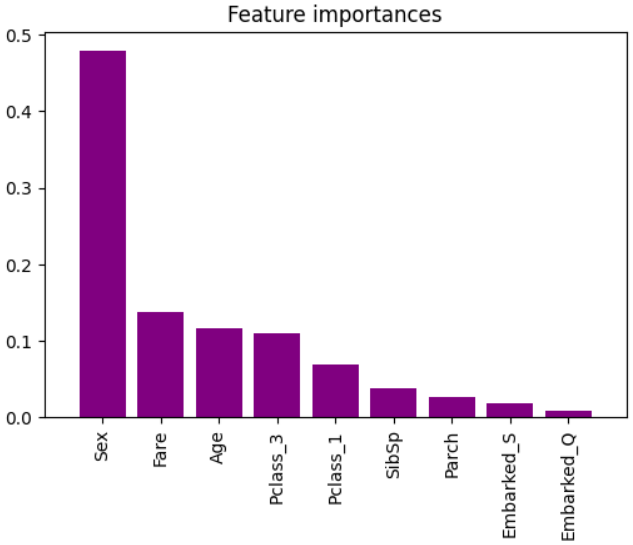

FIGURE 3.9 – Importance des variables selon la Random Forest trouvée par CV

3.4 Boosting

3.4.1 AdaBoost `cv`

Taux de mal classés : 17.96 %

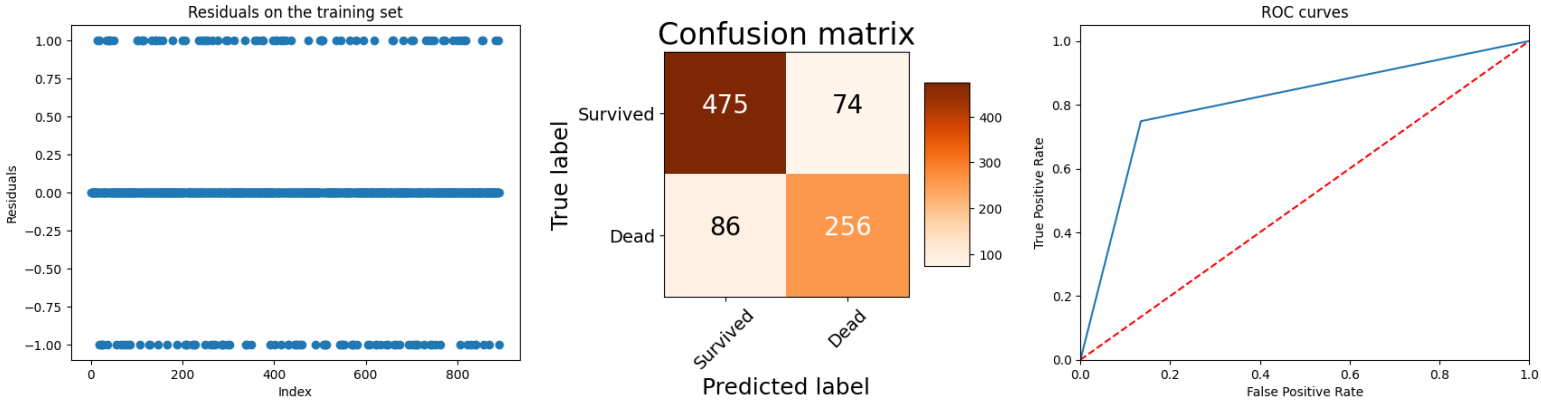

FIGURE 3.10 – Analyse du la classification par Support Vector Machine

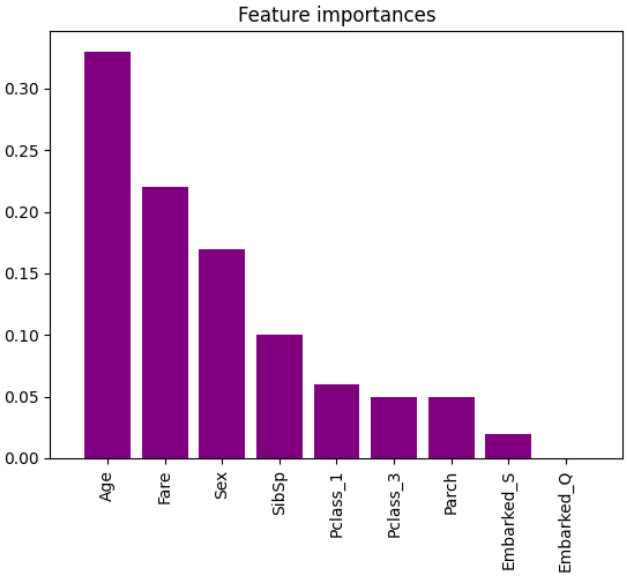

FIGURE 3.11 – Importance des variables selon AdaBoost

Le taux de mal classés obtenu est décevant et comparable à celui obtenu avec la régression linéaire. Ce résultat peut s'expliquer par l'importance accordée à la variable **Age** dans le modèle. En effet, les boxplots présentés dans la figure 1.1 ainsi que les figures 3.7 et 3.9 indiquent que l'âge a moins d'influence sur les résultats que les variables **Fare** et **Sex**. Néanmoins, il est important de noter que ce modèle est l'un des moins biaisés.

3.4.2 XGBoost cv

Taux de mal classés : 8.98 %

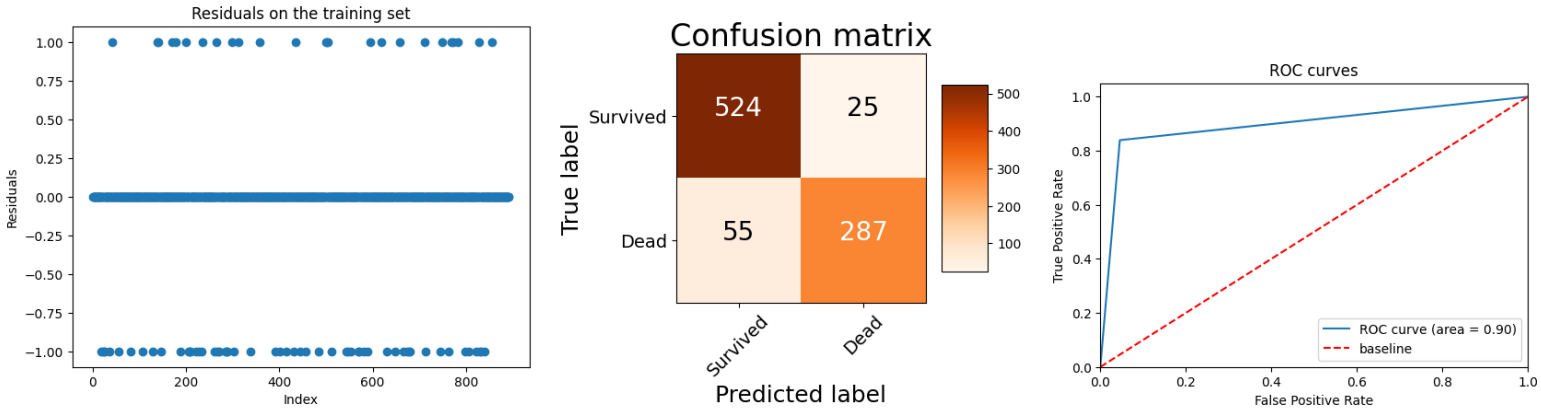

FIGURE 3.12 – Analyse du la classification par XGBoost

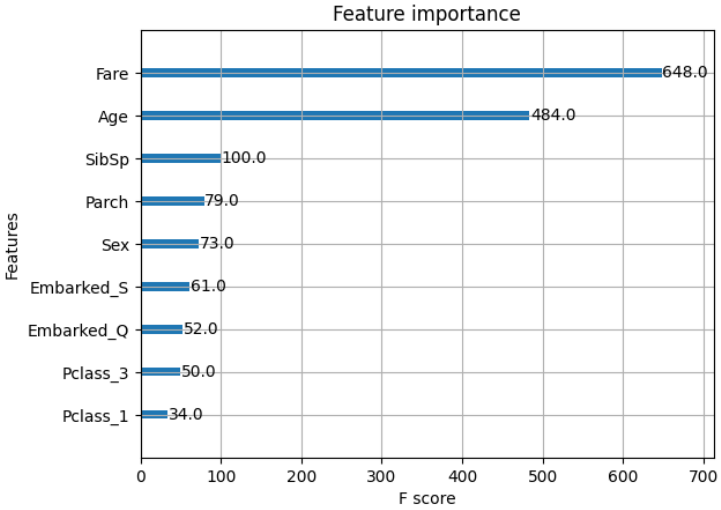

FIGURE 3.13 – Importance des variables selon XGBoost

Le modèle obtient un taux de mal classés très performant ainsi qu'une aire sous la courbe ROC de 0.90, ce qui en fait un bon modèle. Il semble mettre l'accent sur le prix du billet et l'âge des individus pour prédire leur mort.

4 Discussion

4.1 Qualité du Features Engineering

Le but est ici de déterminer si le feature engineering a eu un impact significatif sur les performances de nos modèles. Pour ce faire, nous allons réutiliser le jeu de données initial de train. Seuls les features engineering appliqués à **Embarked** et **Sex** seront conservés, afin de transformer ces variables qualitatives en variables quantitatives utilisables par nos modèles. En outre, les valeurs manquantes pour **Age** seront remplacées par la médiane et les valeurs manquantes pour **Embarked** seront remplacées par la valeur la plus fréquente. On trouve alors la figure 4.14 suivante :

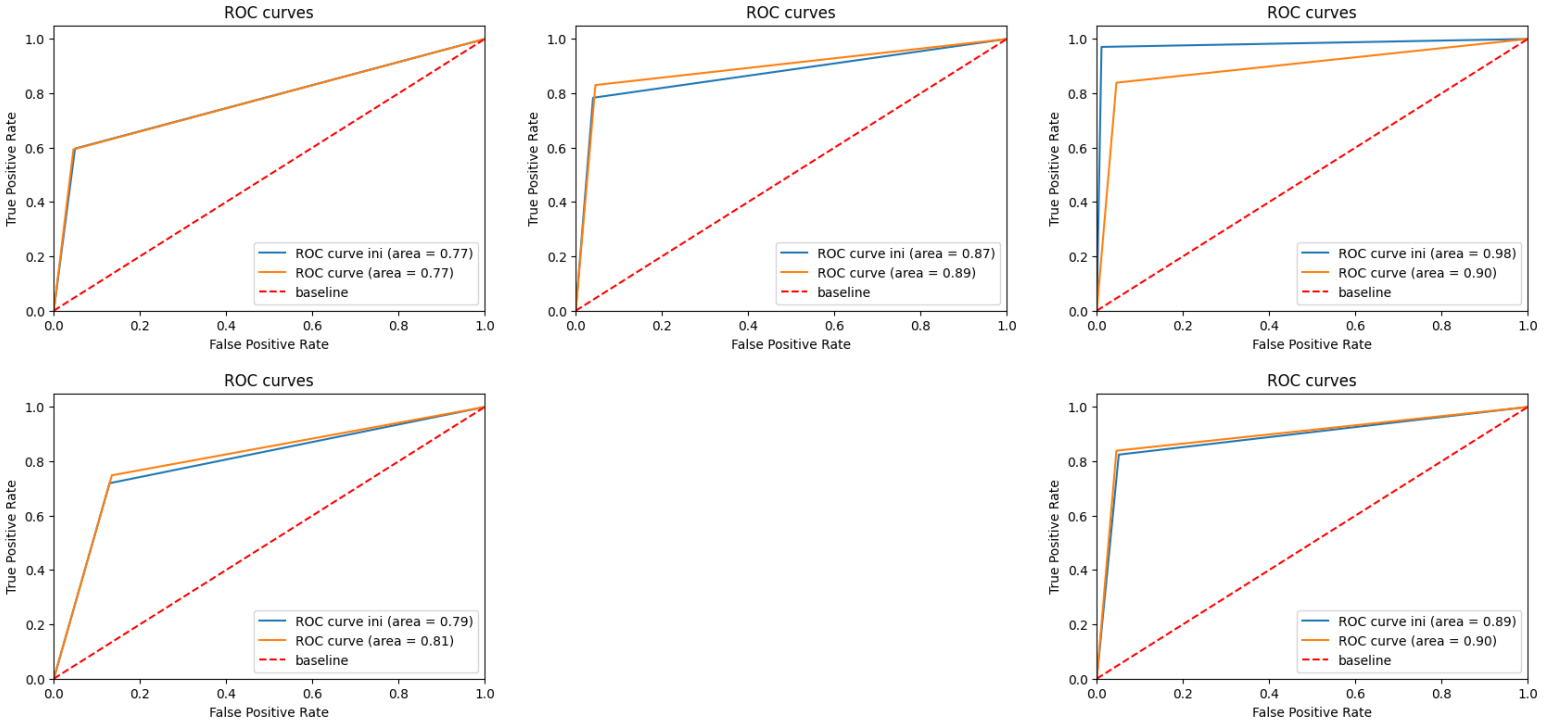

FIGURE 4.14 – Courbe ROC avec et sans (ini) features engineering pour respectivement la régression linéaire, SVM, RandomForest, AdaBoost, XGBoost

Effectivement, les résultats 4.14 montrent une amélioration globale de la performance du modèle avec le feature engineering appliqué. Bien que l'amélioration soit légère avec une baisse d'environ 1% du taux d'erreur de classification, cela reste assez significatif et montre l'utilité du feature engineering même sur des données relativement simples comme celles de Titanic. Cette méthode peut s'avérer particulièrement utile pour des jeux de données plus complexes où les relations entre les variables sont moins évidentes.

4.2 Meilleur modèle

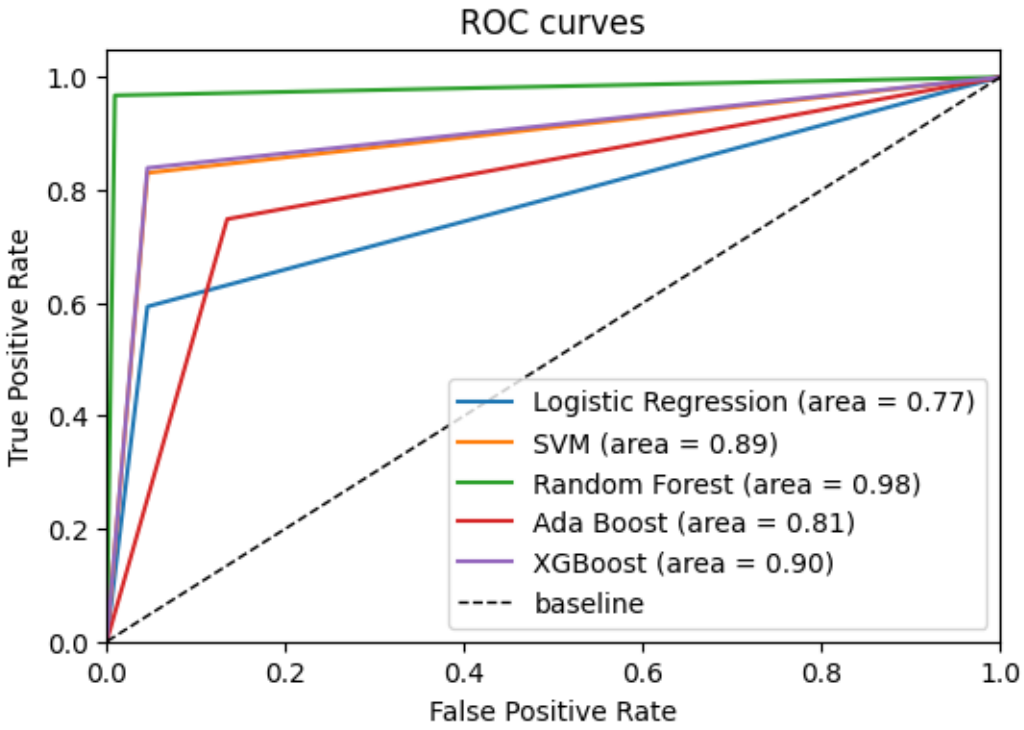

FIGURE 4.15 – Courbes ROC finales

Les courbes ROC 4.15 montrent clairement que le modèle RandomForest est de loin le meilleur pour la prédiction dans ce projet. Cependant, il pourrait être intéressant d'explorer des approches d'agrégation d'experts pour améliorer encore davantage la performance de la prédiction. Cependant, à ma connaissance, cela n'a pas encore été implémenté en Python.

4.3 Score Kaggle

Cette dernière partie concerne l'évaluation du modèle sur Kaggle. Pour être rigoureux, le taux de mal classés aurait dû être remplacé par l'erreur de généralisation, estimée par la moyenne des taux de mal classés obtenus avec une validation croisée k-fold. Cependant, étant donné les résultats de la random forest, il a été décidé de la sélectionner comme meilleur modèle. Par conséquent, le score Kaggle obtenu avec ce choix est de**77%** de bonnes prédictions sur les données test.

Il est regrettable que les colonnes **Name** et **Ticket** n'aient pas été utilisées dans le modèle, car elles pourraient contenir des informations utiles. Par exemple, le nom de famille peut être utilisé pour distinguer les classes sociales et donc, les prix des billets, qui sont une variable importante dans nos modèles. Cependant, la conversion de ces variables de type string en valeurs numériques est un processus complexe et peut nécessiter des techniques avancées telles que la création d'un réseau de neurones pour attribuer une note de "bourgeoisie" à chaque nom, ou l'utilisation de l'embedding de noms.

4.4 Conclusion

Au cours de ce projet, j'ai appris qu'il est important d'explorer plusieurs approches pour résoudre un problème de classification. J'ai pu constater que chaque algorithme de machine learning a ses propres forces et faiblesses, et qu'il est important de choisir le modèle le plus approprié pour le problème en question. Par exemple, la régression linéaire a produit de mauvais résultats pour la classification des données de Titanic, tandis que la Random Forest s'est avérée être le modèle le plus performant. J'ai également appris que l'optimisation des hyperparamètres peut être délicat et qu'il est important de les choisir judicieusement pour obtenir une bonne performance. Enfin, j'ai découvert des algorithmes de boosting tels que Adaboost et XGBoost qui peuvent améliorer les performances des modèles de base en agrégeant des modèles plus simples mais nécéssitent une meilleure optimisation des hyperparamètres.