

**LE JONCOUR Jérémie**

18 Mai 2022

---

# DOSSIER

## Titre Professionnel

**Niveau 6.** Développeur en Intelligence Artificielle  
Enregistré au RNCP sous le n°34757

---



**ISEN**

ALL IS DIGITAL!

**BREST**



**SIMPLON**  
.CO



Le dossier présente les différents travaux liés à l'épreuve de certification du parcours développeur en intelligence artificielle. Il est constitué de deux parties nécessaires pour l'obtention du titre professionnel :

- **Cas pratique réf. E1** : Le rapport du Projet Chef d'œuvre et sa veille technologique associée de réf. E3 (Pages 3 à 83). Le PCO porte sur la création d'une application IA fonctionnelle. Le sujet choisi fait référence à la détection des phénomènes d'Attrition au sein d'un service de *streaming* de musique.
- **Cas pratique réf. E2** : Le rapport sur l'amélioration de l'IA (Au-delà de la page 84). Le sujet donné à pour but d'intégrer une IA de classification en complément d'un modèle de détection d'objet (détection des portes) entraîné en amont.

Pour chaque épreuve, un lien GitHub menant sur les répertoires est disponible, où notebooks, scripts et informations complémentaires sont disponibles.



# LE JONCOUR Jérémie

18 Mai 2022

---

# Cas Pratique E1

## Projet Chef-d'œuvre

Conception d'une application pour  
la détection de l'attrition

---

## Titre Professionnel de Niveau 6

Développeur en Intelligence Artificielle  
Enregistré au RNCP sous le n°34757

SIMPLON<sup>+</sup>.CO



# SOMMAIRE

---

<b>INTRODUCTION.....</b>	<b>6</b>
<b>PARTIE 1 : Etat de l'art et Objectifs.....</b>	<b>7</b>
1. Contexte .....	7
2. Modèles utilisés dans la prédition de l'attrition.....	9
3. Objectifs du Projet.....	13
<b>PARTIE 2 : Conception du projet.....</b>	<b>14</b>
1. Stratégie appliquée .....	14
2. Analyses des données et construction du Dataset .....	14
2.1. Description des données .....	14
2.2. Outils d'analyse .....	15
2.3. Démarche de construction du Dataset final.....	15
2.4. Résumé de l'analyse des données.....	16
2.4.1. Informations récoltées sur <i>User Logs</i> .....	17
2.4.2. Informations récoltées sur <i>Transactions</i> .....	18
2.4.3. Informations récoltées sur <i>Members</i> .....	19
2.4.4. Analyses sur le Dataset final et <i>Features engineering</i> .....	20
3. Modélisation.....	21
3.1. Que cherche-t-on comme Intelligence Artificielle ?.....	21
3.2. Stratégies utilisées dans l'apprentissage machine.....	22
3.2.1. Outils de modélisation .....	22
3.2.2. Une « parenthèse » sur les métriques d'évaluation utilisées .....	22
3.3. Démarche de sélection.....	24
3.3.1. Traitement du Dataset et segmentation .....	24
3.3.2. Sélection du modèle .....	25
3.3.3. Amélioration du modèle sélectionné .....	27
3.3.4. Autres axes étudiés .....	30
3.4. Bilan sur la modélisation .....	32
4. Construction de la base de données relationnelle.....	32
4.1. Base de données relationnelle analytique originale .....	33
4.1.1. Echantillonnage de démonstration .....	33

4.1.2. Exportation des données en BDD.....	33
4.2. La base de données relationnelle intégrée à l'application.....	34
5. Test BDD et Prédiction de l'IA sur Notebook .....	35
5.1. Exportation des données depuis la BDD d'application .....	35
5.2. Preprocessing sur le Dataset résultant.....	36
5.3. Préparation à l'utilisation du modèle IA.....	36
5.4. Prédiction obtenues.....	37
<b>PARTIE 3 : Application du projet.....</b>	<b>39</b>
1. Développement d'une application d'IA.....	39
2. Maquette de l'application .....	39
3. Exécution générale de l'application .....	40
4. Architecture globale et présentation .....	41
4.1. Page de connexion .....	42
4.2. Page principale .....	43
4.3. Pages de résultats.....	43
4.5. Page de performance IA .....	44
4.4. Page de Simulation et Informations .....	45
4.5. Monitoring (avec <i>Flask Monitoring Dashboard</i> ) .....	45
4.6. Caractérisation des points critiques principaux .....	46
<b>PARTIE 4 : Gestion du Projet.....</b>	<b>48</b>
1. Compréhension et caractérisation des besoins .....	48
2. Planification du Projet .....	49
3. Difficultés rencontrées et Résolution.....	50
<b>CONCLUSION &amp; PERSPECTIVES .....</b>	<b>51</b>
1. Discussion et axes d'amélioration .....	51
2. Bilan .....	52
<b>BIBLIOGRAPHIE .....</b>	<b>54</b>
<b>ANNEXES.....</b>	<b>56</b>

# INTRODUCTION

---

Cette épreuve s'intègre dans l'obtention du Titre Professionnel de Développeur en Intelligence Artificielle. Formant la première partie pratique, le projet a pour but de valider 15 compétences techniques. Elles font références aux capacités d'analyses et traitements de données, d'intégration de protocoles de modélisation IA, au développement du *Back-end* et *Front-end* d'une application, de la gestion des bases de données relationnelles ou encore la planification du présent projet.

Le sujet s'inspire des travaux effectués durant ma période d'alternance et portant sur les phénomènes d'attrition, soit la détection précoce de clients risquant de se désabonner ou quitter un service. Etant un cas typique de la data-science, à savoir un problème de classification binaire, de nombreux jeux de données sont disponibles.

Le choix s'est porté sur un Dataset conséquent fournis par une entreprise réelle (KKBOX) sur la plateforme *Kaggle*. Ainsi, ce projet permettra d'appliquer et d'approfondir les compétences acquises durant l'année de formation.

Afin de répondre aux objectifs, une première partie fera mention du contexte même du projet, répertoriant les différents existants et détaillant la problématique. Une deuxième partie portera sur la mise en œuvre du projet, de l'analyse des données à la création des bases résultantes, en passant par la modélisation IA. La partie suivante regroupera l'ensemble des informations sur le développement de l'application. La gestion du projet sera présentée dans une dernière partie avant de conclure et d'évoquer certaines améliorations potentielles de l'application IA.

# PARTIE 1 : Etat de l'art et Objectifs

## 1. Contexte

Depuis 2010, les industries ont développé de nouvelles technologies en vue de la demande croissante des consommateurs de *streaming*, ce processus par lequel la musique multimédia est accessible depuis internet. A ce jour, le *streaming* représente plus de la moitié des revenus de l'industrie musicale mondiale (Site Web UK. Parliament. 2020).

Des plateformes diverses telles que *YouTube Music*, *Apple Music*, *Amazon Music* ou encore *Spotify* ont ainsi réussi à s'étendre à travers le monde et à se populariser (Site Web Frandroid. 2021). Ces services proposent un catalogue de musiques divers et varié, ainsi que d'autres fonctionnalités en rapport (importation de fichiers MP3 des particuliers, écoutes hors-connexion...). Le nombre croissant de services de *streaming* d'écoute sur cette dernière décennie a élevé le niveau de la concurrence, notamment depuis la démocratisation des smartphones. Afin de maintenir leur compétitivité sur ces marchés concurrentiels, les entreprises (fournisseur de biens ou de services) mettent en application plusieurs stratégies :

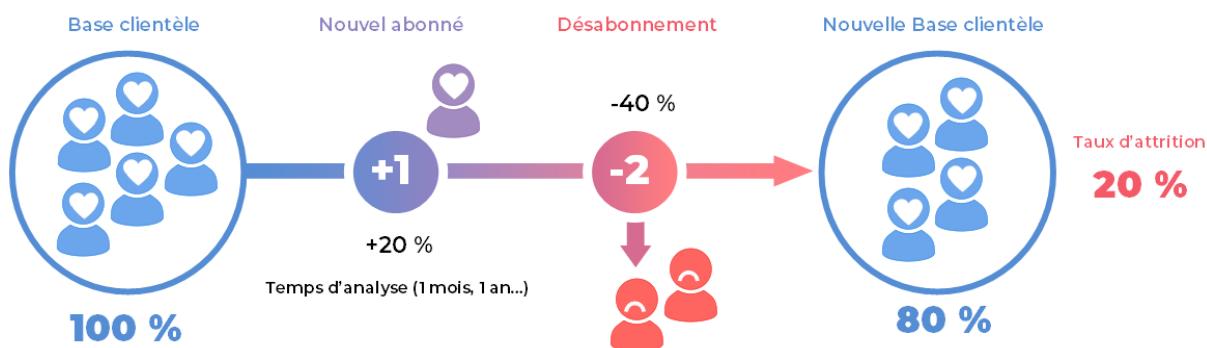
- Acquérir davantage de nouveaux clients
- Ventes incitatives pour les clients existants
- Augmenter la période de rétention des clients.

L'analyse de la valeur du retour sur investissement a montré que la dernière stratégie semblait la plus rentable et la plus simple à mettre en place (E. Ascarza *et al.* 2016). Cela montre que fidéliser un client existant coûte bien moins cher que d'en acquérir de nouveaux (SA. Qureshii *et al.* 2013). Ainsi, les entreprises doivent en général diminuer l'attrition de leurs clients.

Pour une entreprise, le taux d'attrition (communément évoqué sous le terme *churn rate*) représente le pourcentage de clients perdus sur une période donnée (généralement un an ou un mois) par rapport au nombre total de clients se trouvant dans la base clientèle au début de cette période. A l'inverse, le taux de rétention est la capacité de l'entreprise à garder ses clients durant cette même période donnée. Le taux d'attrition est majoritairement calculé sur une segmentation particulière de la clientèle (par rapport à l'âge, segmentation marketing...).

En d'autres termes, le taux d'attrition résume la capacité de l'entreprise à retenir le client, en le fidélisant grâce à une offre pertinente et une gamme de produits ou de services conçue à cet effet. Il est considéré comme un indicateur de performance clé dans les bonnes décisions stratégiques (Site Web Optimove. 2020). Par extension, il indique la satisfaction client auprès du service. La fidélisation du client est donc cruciale pour la pérennité de l'entreprise. Attirer de nouveaux clients est bien plus coûteux que de conserver les clients actuels. En effet, les utilisateurs habitués aux services que propose l'entreprise sont beaucoup plus susceptibles d'acheter des produits que les nouveaux clients. Une autre étude (Site Web Invespcro. 2021) montrait que la probabilité de vendre un produit à un client actuel s'élève à près de 70 % contre seulement 5 à 20 % pour un nouveau client, renforçant ainsi le choix des entreprises à privilégier des mesures réduisant les phénomènes d'attrition.

Par simplification, il existe deux types de désabonnement. Le premier est involontaire, par circonstances indépendantes de leur volonté et entraînant un arrêt du service. Par exemple, cela peut être une expiration de la carte bancaire associée à l'offre de l'entreprise. Le deuxième type, le plus fréquent, est le désabonnement actif.



**Figure 1.** Phénomène d'Attrition, en bleu les clients abonnés à une offre particulière, en rouge, les clients désabonnés. Le taux d'attrition est la résultante de la comparaison du nombre d'utilisateurs totaux avant et après l'analyse.

Trois formes de désabonnement volontaire peuvent être caractérisées et concernent un changement d'intérêt de l'utilisateur pour le service :

- Le client, par perte d'intérêt du type de service ou produit, résilie/se désabonne
- Le client insatisfait passe chez l'entreprise concurrente
- Le client s'abonne à une autre offre commercialisée par la même entreprise, répondant mieux à ses besoins. C'est un phénomène de phagocytage ou cannibalisation des offres.

Pour le bon maintien de leurs services, les entreprises ont généralement recours à un CRM (système de gestion de la relation client). Ce système joue notamment un rôle central dans le développement de la satisfaction client, de la fidélité et de l'interface principale pour interagir avec ses clients. Les CRM (et leurs solutions SaaS/Cloud) peuvent regrouper un ensemble d'informations sur le client et être utilisées afin de contrer les phénomènes d'attrition et assurer leur fidélité par exemple (Site Web *Eudonet*. 2018).

A travers les données provenant de CRM, les entreprises peuvent mettre en place plusieurs stratégies. On retrouve notamment le « *win back* », une opération marketing qui consiste à reconquérir un client présentant une forte probabilité de partir (Site Web *Définitions marketing*. 2019). Cela peut se faire par appel téléphonique et surtout dans les secteurs d'activités commercialisant.

En amont, la détection de l'attrition est quant à elle réalisée à partir d'un *scoring* prédictif. Chaque client se voit attribuer un score en fonction de l'offre choisie et de son comportement vis-à-vis du service utilisé.

Généralement, ces scores sont définis par les services marketing par une analyse de données plus ou moins complexe, comparant les clients satisfaits du service et des clients déjà désabonnés afin d'en ressortir les aspects défaillants de l'offre et le profil des utilisateurs résilients. Dans cette optique, des processus de modélisation via des algorithmes de *Machine Learning* (apprentissages supervisés

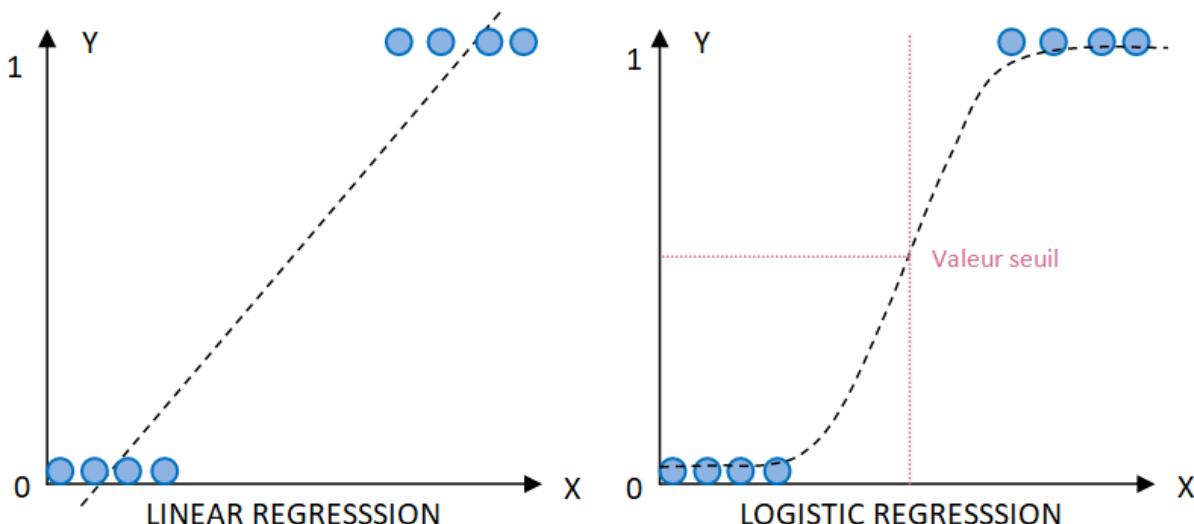
notamment) permettent une détection plus affinée des utilisateurs à fort potentiel d'attrition. La prédiction de ce phénomène est l'un des cas d'usages les plus fréquents de la data-science.

S'appliquant à la plupart des entreprises traitant du désabonnement des clients, de nombreuses solutions ont vu le jour par l'émergence d'outils facilitant la mise en place d'algorithmes performants. De nombreuses plateformes basées sur l'IA telles que *Akkio*, *Churnly* ou encore *Prevision.io*, proposent leurs services afin que les entreprises puissent contrôler, dans ce cas précis, les départs clients grâce à des interfaces utilisateurs simples reliées généralement aux bases CRM.

## 2. Modèles utilisés dans la prédiction de l'attrition

Plusieurs travaux de recherche se sont orientés sur l'étude d'algorithme d'apprentissage supervisé afin de concevoir des modèles de prédiction performants. Les entreprises disposant maintenant d'une grande quantité de données, l'utilisation d'algorithme de *Machine Learning* ou de *Deep Learning* sont envisageables. La détection de l'attrition est en général un problème de classification binaire, comprenant une classe de clients abonnés et de clients désabonnés, le but étant de créer un modèle capable de distinguer les deux types d'utilisateurs dans ce cas spécifique. En les différenciant à travers des variables associées aux clients, les algorithmes entraînés sont capables de prédire sur de nouveaux utilisateurs, leur comportement et leur potentiel désistement de tels services.

Plusieurs algorithmes de classification sont retrouvés dans la littérature pour répondre à ce projet de détection d'attrition. Par exemple, l'équipe de SA. Neslin *et al.* (2006) présentait dans leur publication diverses approches de modélisation comprenant des méthodes statistiques traditionnelles telles que la régression logistique. La **régression logistique** est un modèle statistique linéaire généralisé utilisant une fonction de lien logistique. Elle établit avant tout la probabilité de réalisation d'une des modalités à prédire (abonnés = 0; désabonnés = 1). Cette probabilité est modélisée par une courbe sigmoïde d'intervalle 0 à 1.

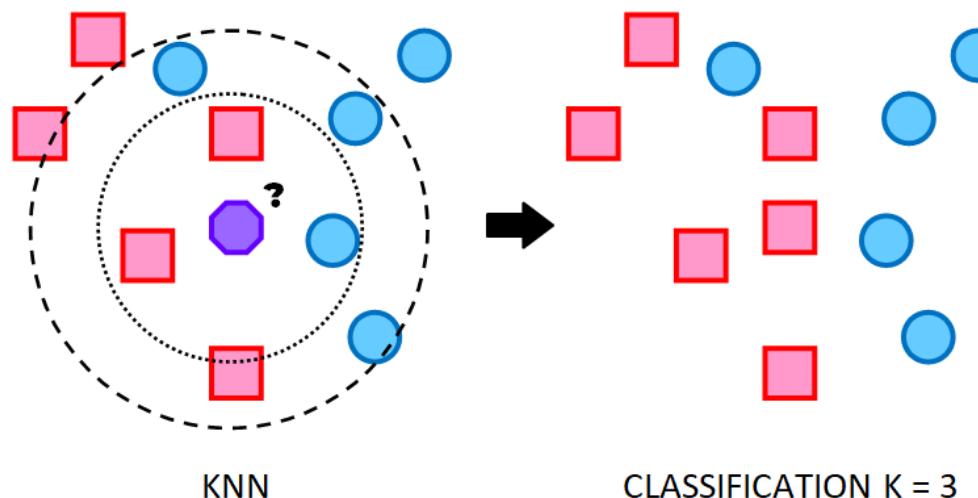


**Figure 2.** Comparaison de Fonction Linéaire et Sigmoïde sur la régression. La régression logistique repose sur l'optimisation des paramètres  $\Theta$  réalisée lors de l'apprentissage machine.

Les algorithmes **Naives Bayes**, basés sur le Théorème de Bayes (fondé sur les probabilités conditionnelles), peuvent aussi être employés dans ce genre de problématique (K.K. Mohbey. 2020). Ces ensembles d'algorithmes traitent les caractéristiques indépendamment de l'ensemble des variables à disposition. Cela implique que chaque fonctionnalité soit indépendante, ce qui n'est pas toujours le cas, et en fait sa principale faiblesse (Site Web *Mrmint*. 2017).

Les modèles de **support vector machine** (SVM) ont été envisagés dans des travaux de modélisation sur la détection de l'attrition. L'article de ZY. Chen *et al.* (2012) effectue un état des lieux sur ces types d'algorithmes et suggère des améliorations probantes. Les SVM ont pour principe de générer une frontière séparant les données de différentes catégories. Dans des cas majoritairement non-linéaire, les SVM utilisent des fonctions (« noyaux ») permettant de projeter les données sur une *feature space*, séparées par un hyperplan. Bien que robuste, les entraînements des SVM sont longs sur des jeux de données volumineux.

On y trouve aussi des modèles statistiques non paramétriques comme l'algorithme des **K plus proches voisins** (*K-nearest neighbors*), des modèles polyvalents qui peuvent être utilisés dans divers secteurs de métier comme dans le cas de prédiction des clients susceptibles de se désabonner d'un service (D. Ruta *et al.* 2006). L'algorithme de *KNN* est l'un des plus simples utilisé dans le *Machine Learning*. Dans la classification, l'algorithme se charge ainsi d'étiqueter le vecteur par rapport aux échantillons d'entraînement les plus proches du point à prédire.

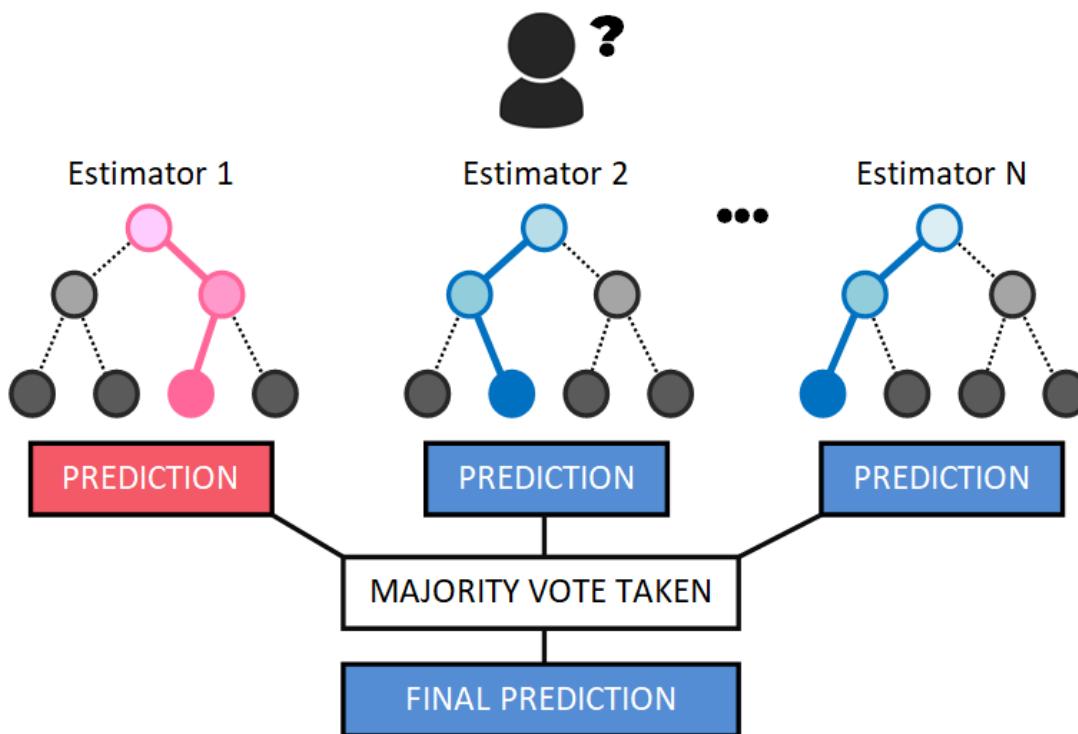


**Figure 3.** Processus de classification KNN. La valeur non étiquetée (en mauve) sera classée en fonction du nombre K voisins les plus proches. La valeur sera classée en rouge si K = 3, ou en bleu si K = 7. K est défini en amont par méthode de *Cross-validation*.

Des modèles décisionnels peuvent être élaborés pour répondre aux projets luttant contre les phénomènes d'attrition comme les algorithmes simples **d'arbre de décision**. Ces algorithmes ont montré une certaine fiabilité dans ce genre de classification binaire (S. Höppner *et al.* 2020). Ils ont pour principe d'effectuer un filtrage de variables par décision. Ces algorithmes sont souvent utilisés sous des méthodes d'ensemble, présentant de meilleures performances dans la classification binaire (K. Coussemont, K.W. De Bock. 2013).

Ces **méthodes d'ensemble** permettent d'obtenir un modèle de prédiction (dans ce cas-ci) résultant des prédictions d'algorithmes simples (*weak learner* ou apprenants faibles à forte variance)

confectionnés en amont. L'objectif de tels modèles vise à corriger plusieurs inconvénients comme la complexité et le surapprentissage des arbres décisionnels par exemple. On y trouve notamment les **Forêts d'Arbres décisionnels** (*Random Forest Classifier*) dans la catégorie des modèles *Bagging* (ou *Bootstrap Aggregating*). Les algorithmes *Bagging* créés des estimateurs entraînés en parallèle pour en ressortir une tendance générale.

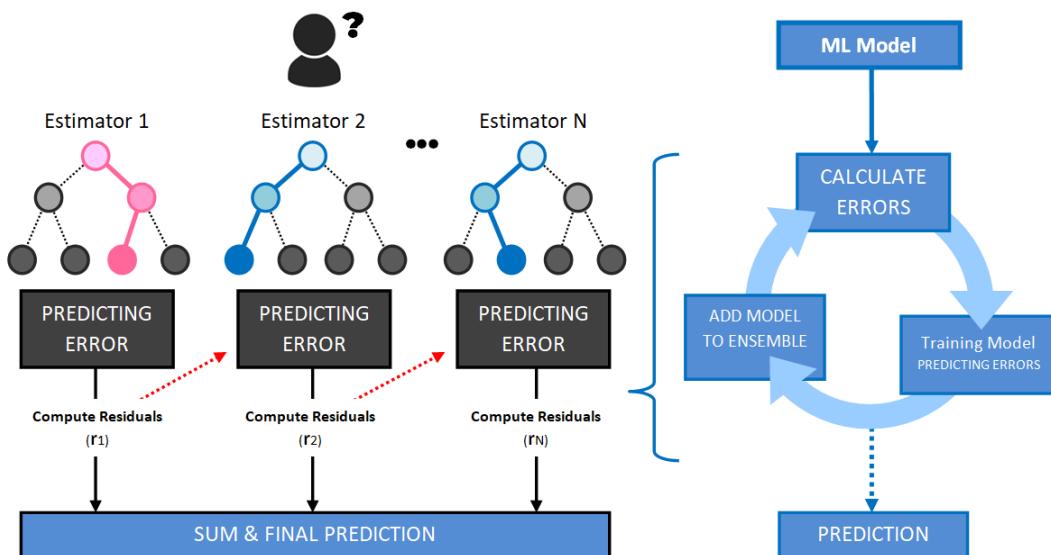


**Figure 4.** Schéma du fonctionnement des Forêts d'arbres décisionnels. Les estimateurs peuvent être des Arbres de Décision, s'entraînant sur des sous-ensembles de données. Chaque arbre émet une prédition et un vote à la majorité est réalisé avant de donner la prédition finale.

Une autre catégorie d'ensemble, nommée *Boosting* (comme le modèle *AdaBoost*), se présente par une succession d'estimateurs qui cherchent à minimiser l'erreur du précédent. Le principe même du *Boosting* est d'accorder une importance (ou poids) sur les estimateurs ayant un taux d'erreur faible. Ces algorithmes *Boosting* adopte ainsi une approche itérative.

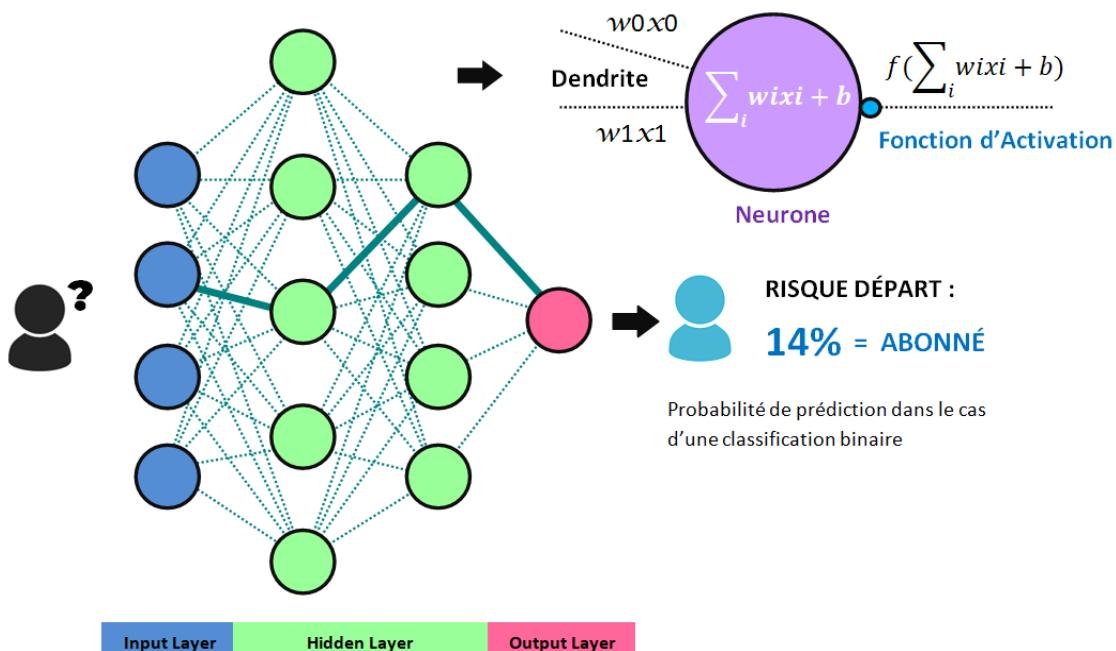
Les modèles de **Gradient Boosting** et leurs versions alternatives tels que *XGBoost* (A.K. Ahmad *et al.* 2019), *LightGBM* ou *CatBoost* par exemple, connaissent une certaine popularité dans de nombreuses compétitions *Kaggle* pour leur efficacité, leur rapidité et leurs performances (Site Web *Datascientest*. 2020).

Au niveau de l'apprentissage supervisé profond (*Deep Learning*), plusieurs publications comme celle de l'équipe de Y. Khan *et al.* (2019) détaillent l'utilisation de **réseaux de neurones artificiels** dans la prédiction des phénomènes d'attritions (notamment dans les entreprises Télécoms). Dans le domaine du *Deep Learning*, les réseaux de neurones (ANN) s'inspirent de l'infrastructure du système nerveux central humain. Différents types de réseaux existent, spécifiques pour chaque besoin. On y trouve notamment les Perceptrons Multicouches (proactifs ou rétroactifs), ou encore les réseaux Convolutifs (traitant les images).



**Figure 5.** Schéma du fonctionnement général d'un *Gradient Boosting* comme *XGBoost*. A la différence des modèles *Bagging*, les estimateurs "renvoient" l'erreur résiduelle de leur prédiction à l'estimateur suivant.

Généralement, les réseaux sont composés d'au moins deux couches de neurones (entrée et sortie). Ainsi, plus le problème à résoudre est complexe, plus le nombre de couches augmente. A chaque neurone composant ces couches est attribué un “poids synaptique” modulé par des règles d’apprentissage. Il en résulte des connexions neurales plus importantes qui vont orienter la prédiction finale du modèle. Ces modèles peuvent présenter des performances plus élevées que les algorithmes de *Machine Learning* sur certaines problématiques courantes notamment sur les données séquentielles (C. Gary Mena *et al.* 2019).



**Figure 6.** A gauche, schéma de réseau de neurone artificiel avec les différentes couches et un exemple de sortie (prédiction). En haut à droite, schéma d'un neurone artificielle et sa représentation mathématique où  $x$  est l’axone (vecteur) du précédent neurone,  $w$  un paramètre de la force synaptique (poids). Le neurone représente la somme de ces vecteurs et leur poids.  $b$  est une constante. Une fonction d’activation est présente afin de transmettre les résultats de l’équation aux neurones de la couche suivante.

Enfin, d'autres recherches se sont penchées sur des systèmes hybrides (principe du **Stacking**), combinant différents algorithmes de *Machine Learning* ou différents modèles ANN pour les rendre collectivement plus performants (Tianpei Xu *et al.* 2021). La stratégie d'agrégation est réalisée à partir des données en construisant un métamodèle. Ce dernier permet d'attribuer des poids « optimaux » aux différents modèles entraînés en amont.

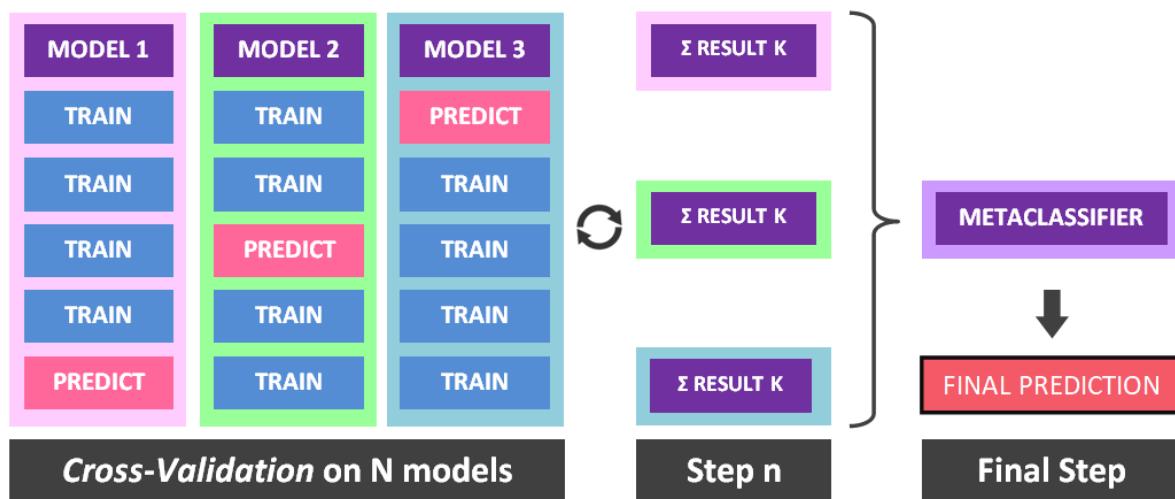


Figure 7. Processus de prédition par un métamodèle. A gauche l'étape de validation croisée K Fold. Une fois les prédictions des modèles préliminaires effectuées, les résultats sont dirigés vers un estimateur global qui émet la prédition finale.

### 3. Objectifs du Projet

Pour récapituler, les phénomènes d'attrition des clients jouent un rôle crucial dans la pérennité de l'entreprise. À travers différentes méthodes de segmentation client, associées aux données des CRM, les services marketing peuvent déterminer la clientèle prédisposée à se désabonner aux services par la mise en place de scores. L'IA peut jouer un rôle important dans ce genre de problématique, notamment par la multitude de modèles existant, et pouvant être adaptés sur les données de l'entreprise mises à disposition.

Le projet de ce présent rapport se propose de construire des modèles prédictifs sur le comportement des utilisateurs (détection des clients à haut potentiel d'attrition) du service de *streaming* d'écoute KKBOX. Il est le premier service de *streaming* musical en Asie, détenant la bibliothèque musicale Asia-Pop la plus complète au monde avec plus de 30 millions de pistes. Ils offrent une version illimitée de leur service à des millions de personnes, soutenue notamment par de la publicité et des abonnements payants. Actuellement, la société utilise des techniques d'analyse de survie pour déterminer la durée de vie résiduelle de chaque abonné.

L'objectif est donc de construire un algorithme qui prédit si un utilisateur se désabonne après l'expiration de son abonnement, à partir des données fournies par KKBOX. Similaire aux services existants de *Churn Detection*, la future application doit contenir un *Front* affichant un *dashboard* récapitulatif des données utilisateurs (provenant d'une base de données faisant office de CRM). Un tableau répertoriant les clients les plus susceptibles de partir sera généré grâce aux résultats fournis par le modèle IA.

# PARTIE 2 : Conception du projet

## 1. Stratégie appliquée

Pour mener à bien ce projet d'application reportant la détection des clients susceptibles de se désabonner des services de KKBOX, 3 étapes ont été définis. Premièrement, un traitement et une analyse des données mises à disposition sont effectués afin d'en relever un profil type préliminaire des clients, ou encore pour créer de nouvelles variables.

Deuxièmement, au vu de la veille technique, plusieurs algorithmes d'apprentissage machine ont été évalués sur le jeu de données résultant afin de sélectionner le modèle de classification le plus performant en se basant sur des métriques spécifiques.

Finalement, deux bases de données relationnelles ont été construites. Une base pour l'application, contient des données non traitées par le modèle IA (données *TEST*). Une autre pour la validation des compétences associées, comporte un échantillon de données clients n'ayant pas subi de traitement, et correspondant aux fichiers CSV.

## 2. Analyses des données et construction du Dataset

Pour obtenir un modèle de classification binaire viable, la première étape était d'étudier les données mises à disposition. Ces analyses permettent de comprendre les profils clients utilisant les services de KKBOX. Le principal objectif était de former un Dataset global permettant l'insertion des données dans une base relationnelle. Cette première étape a été effectuée sur des fichiers Jupyter Notebook, idéal pour le traitement de données et de data-visualisation.

### 2.1. Description des données

Les jeux de données proviennent d'une compétition *Kaggle* disponible à cette [adresse](#) qui présente de nombreux fichiers CSV d'entraînement et de soumissions.

*Une première analyse avait été réalisée sur la Version 1 des CSV avec un ensemble de données beaucoup plus conséquent (près de 420 millions d'instances). L'analyse s'était d'ailleurs portée sur les 8 millions premières lignes d'*user\_logs.csv* et *transactions.csv*, des données antérieures à Mars 2017. Du fait de l'importation en base de données, l'étude s'est concentrée uniquement sur l'analyse des données de Mars 2017.*

Quatre fichiers CSV ont été retenus, rassemblant près de 23 variables différentes :

- ***user\_logs\_v2.csv*** : Un fichier comportant des données sur l'activité du client sur la plateforme comme le temps d'écoute, le nombre de musiques écoutés par jour.

- ***transactions\_v3.csv*** : Regroupe les données sur le type d'abonnement choisi, la méthode de paiement, le prix de l'abonnement, des informations sur son renouvellement ou annulation d'abonnement.
- ***members\_v3.csv*** : Comporte des informations sur le client comme son âge, sa localisation, la méthode d'enregistrement de son compte KKBOX.
- ***train\_v2.csv*** : Comprenant la variable cible nommée *is\_churn* associée pour chaque ID client *msno*. Cette variable binaire indique donc si le client est désabonné ou non.

Le détail de chaque fichier CSV est disponible en annexes (page 56).

## 2.2. Outils d'analyse

Pour mener à bien l'analyse générale des données mises à disposition et appréhender les différents profils utilisateurs, plusieurs librairies Python prévues à cet effet ont été utilisées :



### NumPy

Utilisé dans le traitement de données et la manipulation matricielle à N dimensions. Les concepts de vectorisation, d'indexation et de diffusion de NumPy sont aujourd'hui les standards *de facto* de l'informatique matricielle.



### Pandas

Un outil d'analyse et de manipulation de données *open source* rapide, puissant, flexible et facile à utiliser, construit sur le langage de programmation Python. Il est idéal pour analyser des Dataframes.



### SciPy

Une bibliothèque fournissant des algorithmes pour l'optimisation, l'intégration, l'interpolation, les problèmes de valeurs propres, les équations algébriques, les équations différentielles et les statistiques.



### Matplotlib & Seaborn

Des bibliothèques Python de visualisation de données. Seaborn, basé sur Matplotlib, fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs.

## 2.3. Démarche de construction du Dataset final

Avant de former le Dataset Final, servant de matière première pour la construction des modèles IA, chaque fichier a été traité séparément :

- Les dimensions de chaque Dataframe ont été répertoriées.
- Le type de chaque *Feature* a été vérifié.
- Les valeurs manquantes ont été recherchées.

- Les valeurs aberrantes ont été détectées, en mettant en place différentes stratégies de modification.
- Des graphes *Countplots* ont été réalisés pour appréhender la proportion d'individus dans chaque catégorie.
- Des matrices de corrélation de Pearson ont été construites pour surtout en tirer des points de connexion entre les *Features*.

Une fois les Datasets nettoyés, chacun a été fusionné grâce à la méthode *Pandas.merge* avec le Dataframe *Train*. En reliant les données par utilisateur, le processus a été réalisé sur la variable *msno*.

Les informations des utilisateurs ont été regroupées par un *Groupby* sur l'ID client *msno* afin d'avoir une ligne pour chacun des clients. Sur *User\_logs*, les données de chaque variable ont été reportées sur le mois en générant de nouvelles *Features* comptabilisant la somme, la moyenne et le nombre de jours d'écoute. Sur *Transactions*, un filtre est appliqué en prenant la dernière date de transaction effectuée pour chaque client. Une variable *transaction\_count* est créée pour conserver le nombre de transactions effectuées sur le mois.

Une analyse est effectuée pour les deux types d'utilisateurs (abonnés et désabonnés) sur chaque variable. Des tests statistiques ont été réalisés pour vérifier si des différences significatives entre les deux types de « population » d'individu existaient.

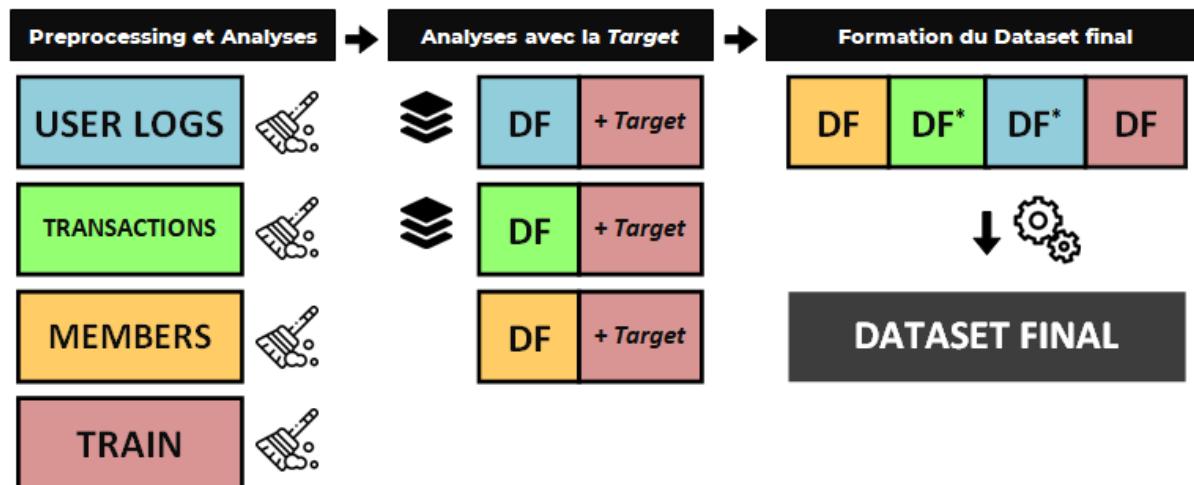


Figure 8. Démarche de l'analyse et création du Dataset Final.

Après réduction de dimension, en appliquant une ligne pour un utilisateur unique (par méthode *Groupby*), les quatre Dataframes ont été fusionnés. De nouvelles variables ont été créées en conséquence (*Features Engineering*), dans le but d'affiner potentiellement les prédictions du modèle d'Intelligence Artificielle.

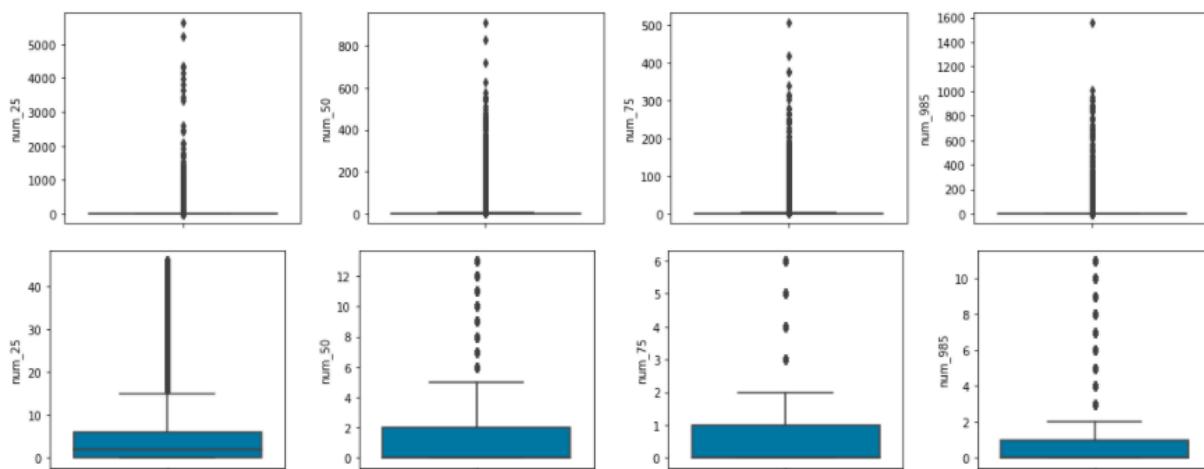
## 2.4. Résumé de l'analyse des données

*Les jeux de données étant conséquents, et possédant de nombreuses variables, l'étude de ces derniers a été cruciale dans la stratégie de modélisation. Cette partie forme un récapitulatif du travail réalisé en amont de la formation du Dataset Final. L'étude complète est disponible sur [Github](#).*

## 2.4.1. Informations récoltées sur User Logs

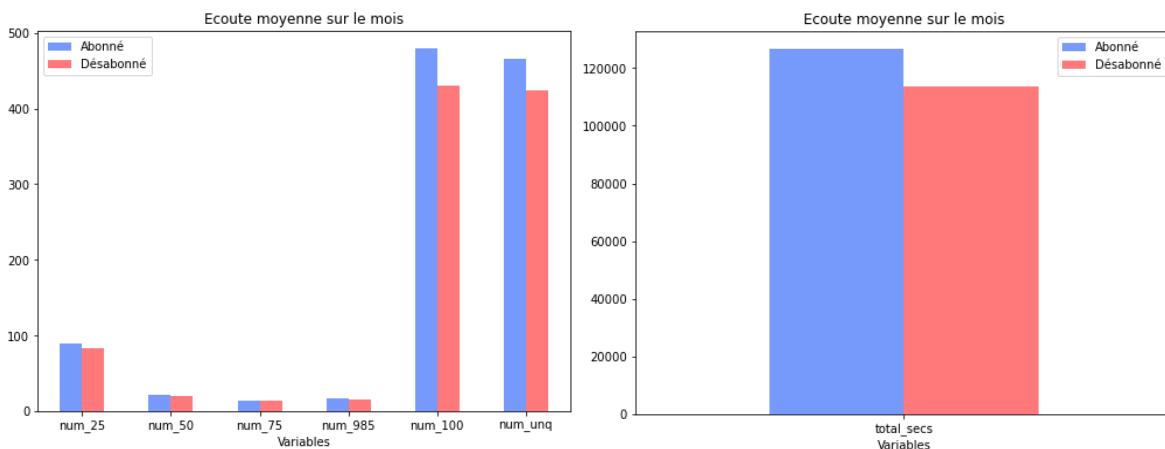
Ce Dataset ne contient aucune valeur manquante. Il est constitué de plus de 18 millions d'entrées. Sans compter la variable *msno* (Identifiant utilisateur de type *String*), les autres variables contiennent des données de type *Integer* et *Float*.

Après vérification des valeurs minimales et maximales, certaines valeurs aberrantes ont été décelées. Une analyse de la quantité de ces valeurs a donc été envisagée. L'étude s'est orientée sur la définition d'un seuil grâce au Z-Score (nombre d'écart-types fixé par rapport à l'estimation moyenne). Bien que la part de valeurs aberrantes soit restreint (environ 1% sur l'ensemble des données), la fonction de réduction des valeurs aberrantes a tout de même été utilisée.



**Figure 9.** Ensemble de Boxplots présentant la réduction de valeurs aberrantes sur des exemples de variables. En haut la part de valeurs aberrantes avant modification, en bas après réduction des valeurs aberrantes sur le Dataset Log de chaque variable.

Les données de temps d'écoute et du nombre de musiques écoutées par jour pour chaque utilisateur ont été reportées sur la totalité du mois de Mars. Après ajout de la variable cible *is\_churn*, des tests ont montré des différences significatives entre les individus (moyennes des distributions) ayant stoppé leur abonnement et les individus encore abonnés.

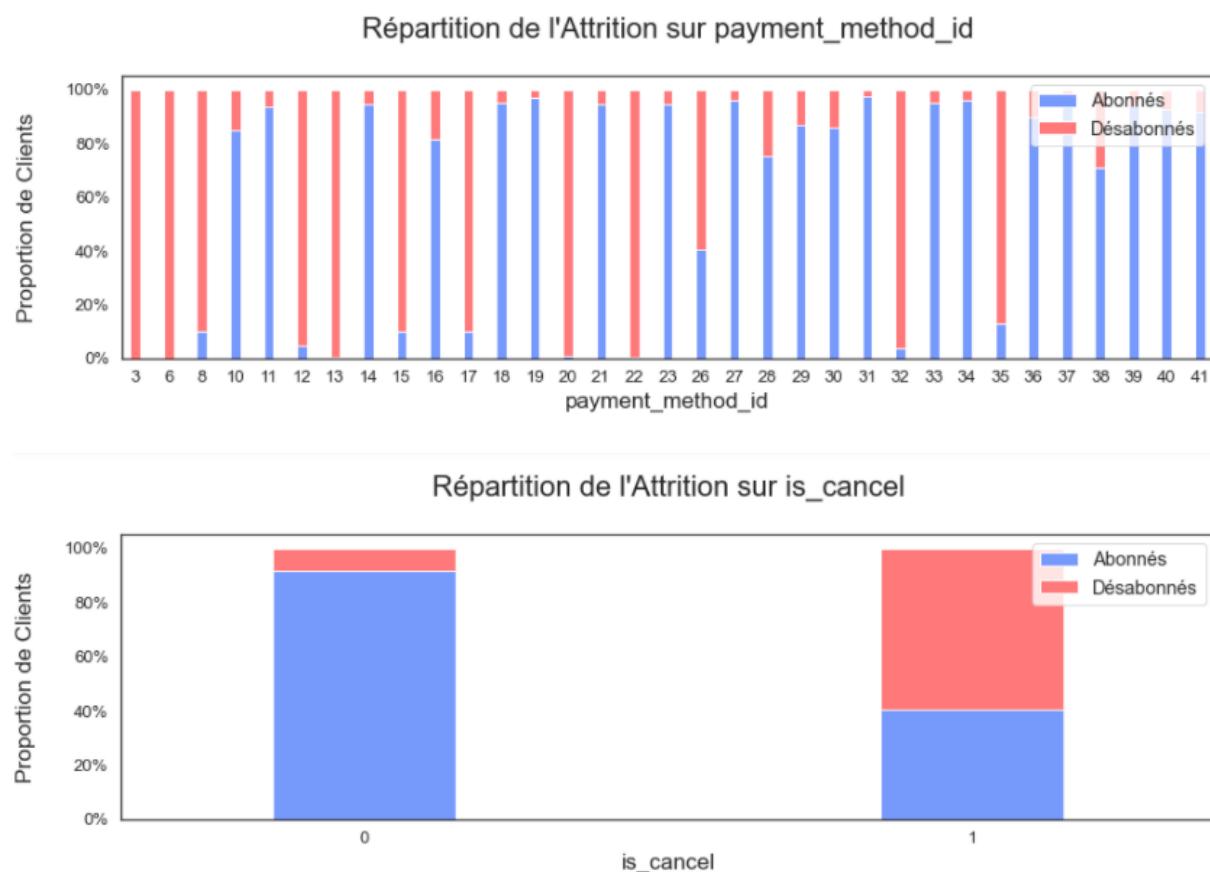


**Figure 10.** Proportion d'individus désabonnés et abonnés en fonction du nombre de chansons moyen à x% écoutés et temps d'écoute moyen sur le mois de Mars 2017.

Globalement, les individus désabonnés écoutaient moins de musique et moins longtemps sur le mois. A ce stade de l'étude, l'hypothèse émise serait que ces variables peuvent être des marqueurs de choix dans la détermination des futurs membres désabonnés.

#### 2.4.2. Informations récoltées sur *Transactions*

Le Dataset du fichier *transactions.csv* ne présente pas de variable quantitative. Bien que ces catégories contiennent des données de type *Integer*, elles représentent des variables de nature temporelle, ordinale, binaire et nominale. Après vérification, ce jeu de données ne présentait aucune valeur manquante. La proportion d'utilisateur a été étudiée sur chaque variable et pour chaque type de cible.



**Figure 11.** Exemple d'analyses sur la proportion d'individus en fonction des variables *Transactions*. Ces visualisations permettent de remarquer la grande variabilité au sein des deux catégories d'individus.

En résumé :

- **Payment\_method\_id** : Cette catégorie montre une certaine variabilité dans la proportion d'utilisateurs quittant les services KKBOX. Les clients qui utilisent certaines méthodes de paiement sont plus sujet à partir.
- **Plan\_list\_price** : La proportion d'individus en fonction de cette catégorie sur *is\_churn* présente une grande variabilité.

- **Actual\_amount\_paid** : Comme pour *plan\_list\_price*, cette catégorie présente une forte variabilité de proportion entre utilisateurs restant et sortant. Les abonnements payés par la plupart des utilisateurs s'élèvent à 149 NTD et 99 NTD.
- **Auto\_renew et Is\_cancel** : Ces variables présentent une corrélation notable respectivement négative et positive sur la propension à ce qu'un utilisateur arrête son abonnement.
- **Transaction\_count** : Cette catégorie, créée à partir de la fonction *count*, présente une grande variabilité.

Il existe de nombreuses ambiguïtés sur le Dataset *Transactions*. Certains utilisateurs présentent une date d'expiration d'abonnement très avancée dans le temps mais ils sont tout de même caractérisés comme désabonnés. De plus le *payment\_plan\_days* de 30 jours et la date d'expiration de certains utilisateurs présentent des anomalies (cumule de jours d'abonnement et différence entre la dernière date de transaction et la date d'expiration ne corroborant pas).

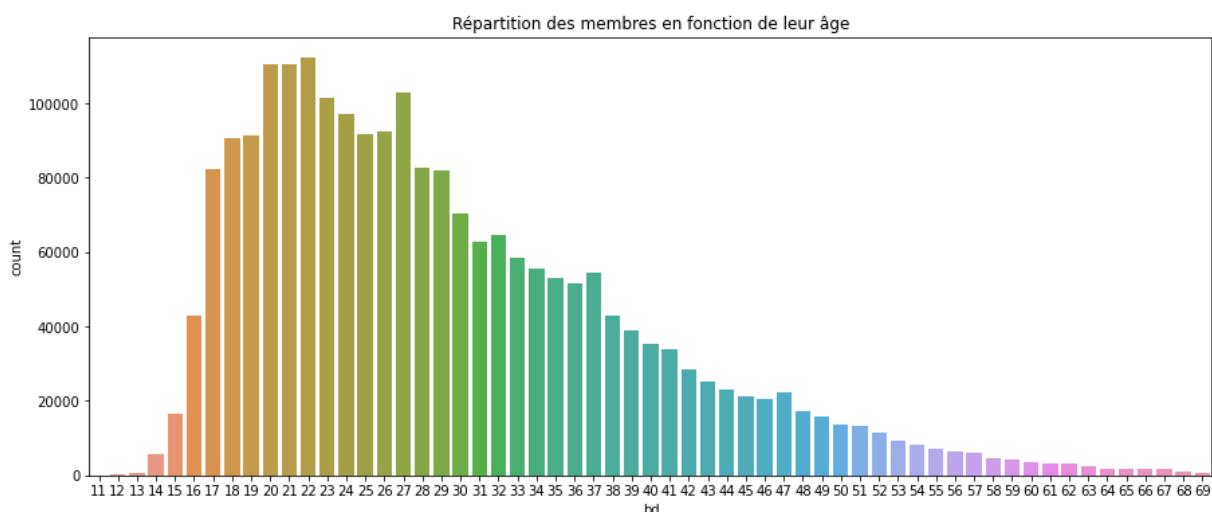
Aussi, selon les auteurs, des données de transactions manquent au-delà du mois de Mars 2017. Les dates de transactions n'avaient que peu d'intérêt dans la conception du modèle pour éviter tout biais.

#### 2.4.3. Informations récoltées sur *Members*

Le Dataset *Members* comprend des données sur l'âge (*bd* : seule variable quantitative), le sexe (*gender*), la méthode d'enregistrement de compte (*registered\_via*) (variables nominales) et la date d'enregistrement (*registration\_init\_time*).

Des valeurs manquantes ont été relevées sur la variable *gender* (plus de 65%). Elles ont été remplacées par le statut « *Inconnu* » pour la suite de l'analyse (cette *Feature* ne sera pas prise en compte dans le cadre de la modélisation). Sur la description des données, *bd* présente de nombreuses valeurs aberrantes (valeur minimale : -7168 ans; valeur maximale : 2016 ans).

De façon arbitraire, les valeurs considérées comme aberrantes sont celles situées hors d'un intervalle convenable de 10 - 70 ans. Les valeurs aberrantes en dehors de l'intervalle arbitraire ont été notées « -1 » pour la suite de l'étude. Ainsi la part de ces valeurs modifiées atteint près de 67.3%.



### Répartition de l'Attrition sur bd

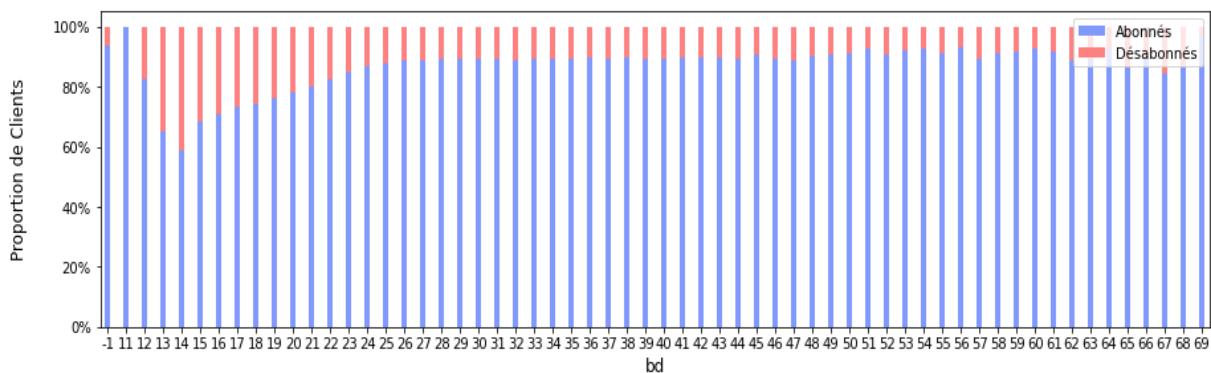


Figure 12. Proportion des individus en fonction de leur âge et en fonction de la catégorie cible *is\_churn*.

Pour faire le bilan sur ce Dataset :

- **Bd** : La proportion d'utilisateurs ayant une plus forte propension à quitter ces services sont âgés de 11 à 25 ans. La proportion de membres se désabonnant pour plus de 30 jours est similaire dans les autres catégories d'âge.
- **City** : La répartition des utilisateurs abonnés/désabonnés, est équivalente en fonction des villes. Dans cette première analyse, la localisation de l'utilisateur ne doit pas influencer sur son statut d'abonnement.
- **Gender** : La part d'utilisateur qu'ils soient Homme ou Femme quittant les services de *streaming* est la même. Dans cette première analyse, le genre ne doit pas influencer sur ces phénomènes d'attrition.
- **Register\_via** : La proportion d'individus utilisant la méthode d'enregistrement 7 (qui plus est, majoritaire) est moins encline à partir. A l'inverse, elle est plus importante sur la méthode d'enregistrement 4 (6% des utilisateurs).

#### 2.4.4. Analyses sur le Dataset final et Features engineering

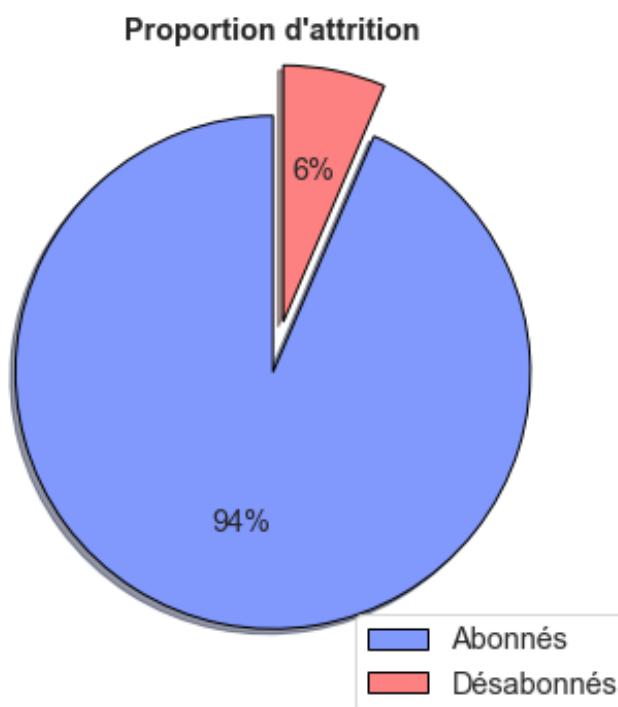
Après concaténation des 4 Datasets, une analyse générale a été réalisée. Le Dataset final comprend plus de 725000 instances/utilisateurs uniques. La part des catégories de la variable cible *is\_churn* montre que près de 6% des utilisateurs n'ont pas renouvelé leur abonnement. Ainsi, 94% des clients sont toujours abonnés durant la période d'analyse. Cette nouvelle information a été prise en considération pour la construction du modèle IA.

Deux nouvelles *Features* ont été créées en conséquence, générées à partir de plusieurs variables existantes :

- **Rapport Prix Abonnement et Temps d'abonnement** : Ce service étant disponible dans de nombreux pays, le prix / jour peut différer et est peut-être une variable intéressante à prendre en compte, un rapport du prix de l'abonnement sur une journée à donc été réalisé.

- **Temps de fidélité** *Membership\_expire\_date* et *Registration\_init\_time* : *Fidelity\_days* : A partir d'une conversion des données temporelles *string to datetime*, une nouvelle variable *Fidelity\_days* a été créée. Les utilisateurs désabonnés ont en moyenne été abonnés plus longtemps que ceux étant encore abonnés. Cette étude est contre-intuitive mais pourrait s'expliquer par un désintérêt croissant des utilisateurs désabonnés au service de la plateforme. Elle peut aussi s'expliquer par des arrêts discontinus de l'abonnement, espaçant davantage l'utilisation des services proposés par la plateforme de *streaming*.

Le Dataset Final a enfin été enregistré sous un format CSV, et prêt à être utilisé pour développer les modèles IA.



**Figure 13.** Proportion d'utilisateurs désabonnés et abonnés sur le Dataset fusionné. Sur le mois, cela revient à un taux d'attrition de 6%.

## 3. Modélisation

### 3.1. Que cherche-t-on comme Intelligence Artificielle ?

Dans le cadre de ce projet, le modèle IA idéal est capable de détecter à la perfection, avec une confiance de prédiction élevée, les individus susceptibles de quitter les services de la plateforme de *streaming* KKBOX. Toutefois, la littérature a montré que les modèles basées sur de la classification à partir de classes déséquilibrées présentent certaines contraintes (Bartosz Krawczyk, 2016).

Ce déséquilibre des classes est d'ailleurs constaté dans le Dataset construit en amont, présentant un ratio de 1:15. Ainsi, la proportion de Faux-Négatifs et Faux-Positifs peut être élevée dans ce genre de cas. La plupart des algorithmes de *Machine Learning* étant performants sur des classes équilibrées, cette notion d'inégalité est à prendre en compte.

En terme général, le modèle doit donc être capable de minimiser les erreurs de prédiction, tout en émettant une probabilité élevée de cette dernière (basée sur le *Log Loss* et l'*AUC*). De plus, le modèle doit minimiser la part de Faux-Négatifs (individus classés comme restant abonnés alors qu'ils sont partis) tout en modérant la part d'individus jugés comme désabonnés mais restant fidèles. Dans sa finalité, le modèle doit retourner une probabilité de risque au lieu d'émettre simplement si le client part ou non.

Evidemment, le modèle entraîné doit être capable de garder un sens de la généralisation. Son entraînement doit donc être contrôlé par une méthode de validation croisée dans le cas des algorithmes de *Machine Learning*.

Une seconde étude s'est toutefois portée sur la création de modèles de *Machine Learning* entraînés sur des classes équilibrées où l'échantillonnage s'était réalisé par différentes méthodes.

## 3.2. Stratégies utilisées dans l'apprentissage machine

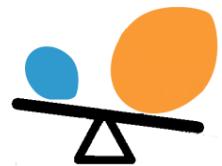
### 3.2.1. Outils de modélisation

Divers outils ont été utilisés afin de construire les différents modèles d'intelligence artificielle. De plus, les algorithmes d'évaluation ont été nécessaires afin de vérifier les performances de ces dernières :



#### Scikit Learn

Une bibliothèque Python développée par de nombreux contributeurs d'instituts français de l'enseignement supérieur. Elle met à disposition des algorithmes d'apprentissage automatique, rapide à prendre en main.



#### Imbalanced Learn

Une bibliothèque open-source et sous licence MIT s'appuyant sur *Scikit Learn*. Elle fournit des algorithmes contrôlés de rééquilibrage de classe ainsi que des algorithmes de *Machine Learning* prévu lors de déséquilibres de classe avéré.



#### Tensorflow Keras

Développé par Google Brain, *Tensorflow Keras* est l'API open-source permettant de créer et d'entraîner des modèles de *Deep Learning*. Il est facile à prendre en main et est considéré comme l'un des outils les plus utilisés en IA.

### 3.2.2. Une « parenthèse » sur les métriques d'évaluation utilisées

Se basant sur de la classification binaire, les métriques d'évaluation de modèle sont :

- ***Log Loss*** (*Logistic Loss*, *Cross-entropy Loss*) : La mesure de classification la plus importante dans le cadre de ce projet, basée sur les probabilités. Cette métrique est celle suggérée pour l'évaluation des modèles dans la compétition *Kaggle*. Les prédictions du modèle évalué sont

davantage robustes si le score est proche de 0. *Log Loss* est une variante de la fonction de vraisemblance.

$$L_{\log}(y, p) = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p))$$

**Figure 14.** Le score représente le produit des probabilités prédictives. La fonction de perte Logistique est définie sous l'équation ci-dessus où  $y$  est la Target 0 ou 1,  $p$  l'estimation du modèle.

- **Recall** (Sensibilité, Taux de Vrai Positifs) & **Precision** (Spécificité) : Permettent de mieux comprendre la part d'erreur de classification (Faux Négatif, et Faux positif) du modèle évalué. *Recall* représente la part de Vrais-Positif (True Positive Rate TPR). *Precision* correspond à la part de Vrais Positifs réels sur l'ensemble des valeurs jugées positives par le modèle et réel.
- **Score F1** : résume la performance de classification du modèle. Ces scores sont tout aussi pertinents pour en vérifier les prédictions du modèle sur l'ensemble des données *Valid* et *Test* (Chang Cao *et al*, 2020).
- **Matthews Correlation Coefficient (MCC)** : Similaire au *score F1*, *MCC* est plus sensible sur les classes déséquilibrées. Le *MCC* fournit une mesure équilibrée qui peut être utilisée même si les classes ont des tailles différentes. Un coefficient de +1 indique une prédiction parfaite ; 0 signifie que la prédiction n'est pas meilleure qu'une prédiction aléatoire ; -1 indique la pire prédiction possible. Les chercheurs en apprentissage automatique et en biostatistiques ont largement utilisé le *MCC* comme mesure de performance pour évaluer les classificateurs binaires pour des ensembles de données équilibrés et déséquilibrés (Chang Cao *et al*, 2020).

$$\text{Recall} = \text{True positive rate (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

$$MCC = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}}$$

**Figure 15.** Equation des différentes métriques utilisées dans ce projet, couramment utilisées dans la vérification des performances d'IA basées sur des classes déséquilibrées (d'après Chang Cao *et al*, 2020).

- **AUC** (Aire Sous la Courbe ROC, Fonction d'Efficacité du Récepteur) : Cette métrique mesure toute la zone bidimensionnelle sous toute la courbe ROC (intégral) de (0,0) à (1,1). L'AUC est comprise entre 0 et 1. Un modèle dont les prédictions sont fausses à 100 % a une AUC de 0 ; celui dont les prédictions sont correctes à 100 % a une AUC de 1. Cette métrique est pertinente pour les cas de distributions très déséquilibrées.

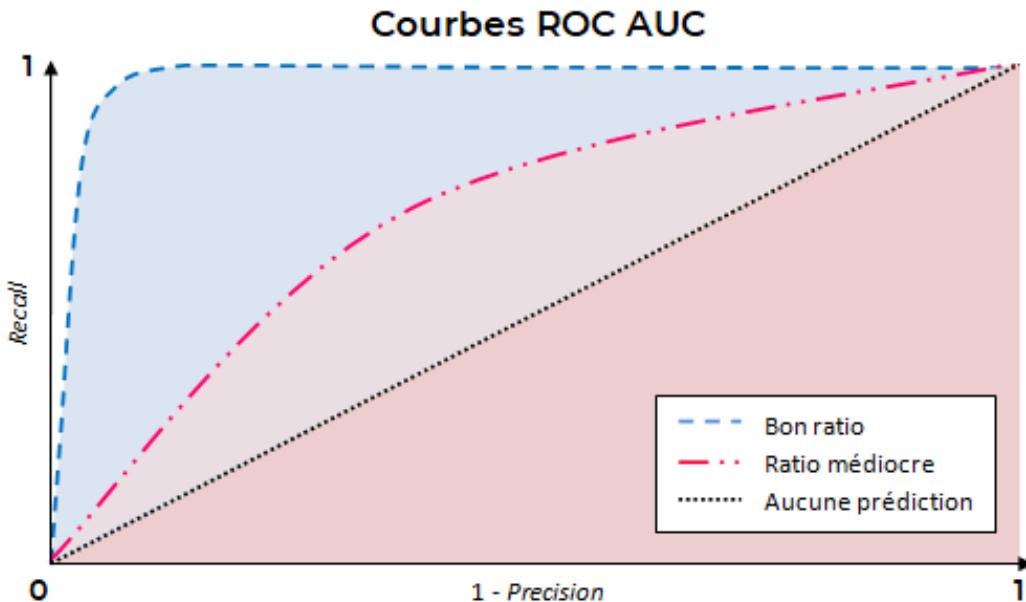


Figure 16. Présentation schématique des courbes ROC AUC.

### 3.3. Démarche de sélection

Dans cette étape, plusieurs modèles ont été construits à partir des données mises à disposition. En se référant à la littérature, plusieurs stratégies ont été appliquées. La modélisation a été réalisée sur Google Collaboratory (Colab Notebook), permettant d'exécuter du code Python via un navigateur Web, adapté au *Machine Learning* et *Deep Learning*.

*L'étape de modélisation complète est disponible en annexe et sur [GitHub](#).*

#### 3.3.1. Traitement du Dataset et segmentation

Le Dataset résultant de l'analyse a subi un encodage sur les variables catégorielles et temporelles afin que les algorithmes puissent les traiter durant la phase d'apprentissage. La fonction `get_dummies` de *Pandas* convertit les données catégorielles en variables indicatrices. Des variables d'ordre binaire sont donc créées. Pour les dates, celles-ci ont été converties en *Integer*. Il en résulte au total 90 variables pour 725722 instances (725722 clients uniques).

Le Dataset a ensuite été segmenté en 3 parties avec l'outil `train_test_split` de *Scikit Learn*:

- *TRAIN* est composé de 70% du jeu de données de départ, utilisé dans la phase d'apprentissage des modèles.
- Les 30% restants sont partagés entre le Dataset *VALID* (80%) et *TEST* (20%). *VALID* est utilisé dans l'évaluation des modèles entraînés, afin d'en vérifier leurs performances à travers les différentes métriques précédemment évoquées. Le Dataset *TEST* est incorporé dans une base de données relationnelle, et sera utilisé uniquement dans l'application du projet.

Du fait que les classes soient déséquilibrées, le paramètre *stratify* a été défini sur la variable cible *is\_churn*. Pour conserver éventuellement la partition des données, un *random\_state* a été appliqué. Les variables *msno* et *is\_churn* ont été enlevées de *X\_train* et *X\_valid*.

Les données *X\_train* et *X\_valid* ont ensuite été remises à l'échelle par une standardisation (*Standarscaler*). La configuration de la standardisation a été sauvegardée dans un fichier *Scaler.Joblib* pour l'appliquer sur de prochaines nouvelles données (dans le cas des données *Test* ici).

### 3.3.2. Sélection du modèle

Un schéma détaillé est disponible en annexes (annexe 10, page 63).

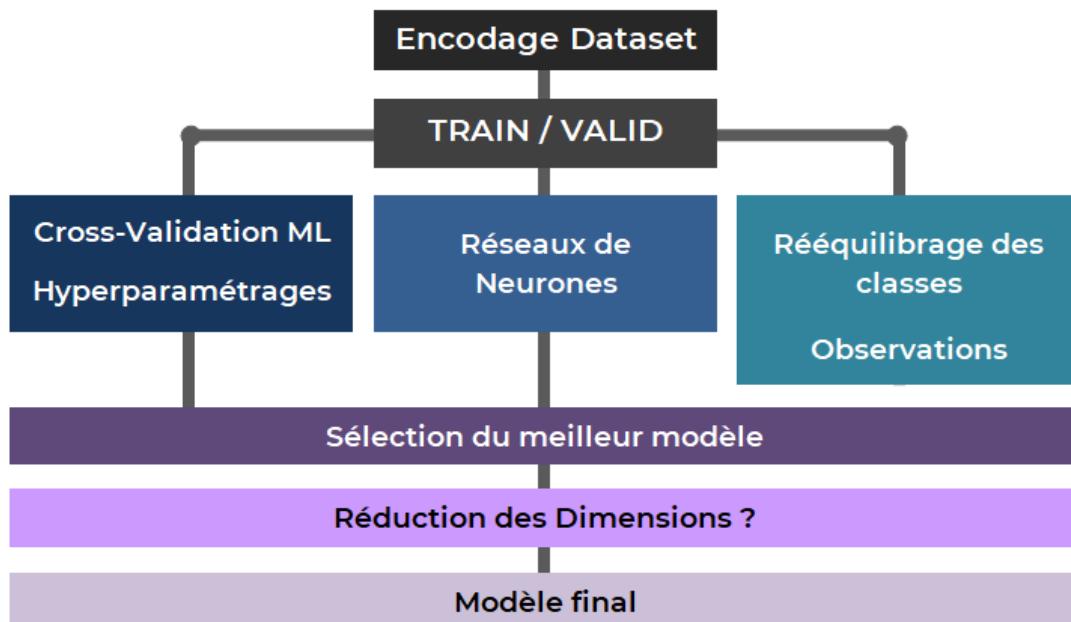


Figure 17. Schéma de la démarche où 3 stratégies de modélisation ont été réalisées.

#### 3.3.2.1. Sélection de Modèle ML par *Cross-Validation / Hyperparamétrages*

Huit modèles ML ont été évalués par méthode de *Cross-Validation*. La validation croisée permet de déterminer de façon rigoureuse si les modèles peuvent généraliser ce qu'ils ont appris ou remarquer s'ils sont sous/sur-ajustés. Par méthode *K-Fold*, les données *Train* sont partitionnées en plusieurs plis (sous-échantillons) qui seront fournies aux modèles. La moyenne et la variabilité des performances de chaque modèle seront ainsi étudiées. Un modèle ayant une faible variabilité et une moyenne élevée dans ses scores de performances sera plus enclin à être sélectionné.

Parmi ces algorithmes, six proviennent de la bibliothèque *Scikit Learn*, les deux autres sont des modèles de *Gradient Boosting* (Light GBM et XGBoost Classifier) provenant d'autres bibliothèques.

Au vu de cette phase d'entraînement par validation croisée, ces deux derniers modèles montrent les meilleures performances et ont été retenus pour une optimisation de leurs hyperparamètres et évalués sur le set *Valid*.

### Comparaison Score MCC après Cross-validation

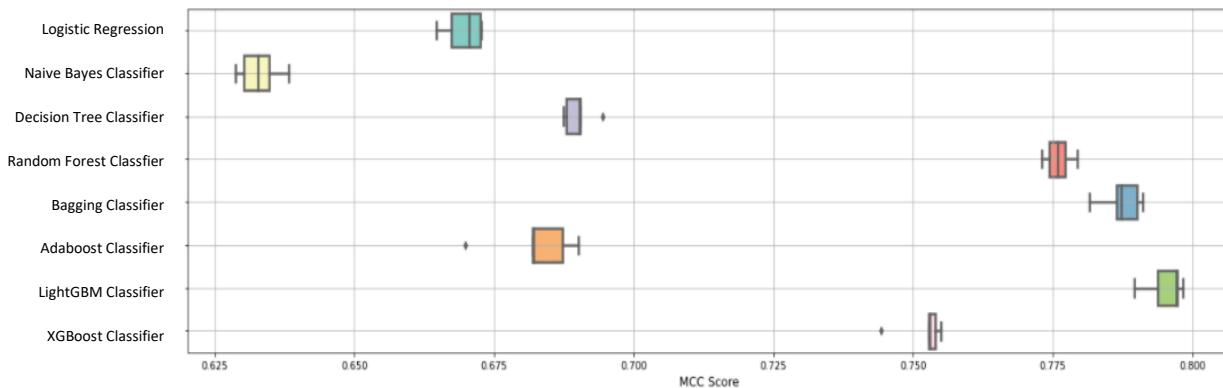


Figure 18. Comparaison des modèles ML en fonction du score MCC.

L'entraînement et la recherche des meilleurs paramètres prennent un certain temps au vu de la quantité des données d'entrée. La méthode *HalvingGridSearchCV* (annexe 13 page 65) permet de gagner du temps sur l'hyperparamétrage. Elle est plus rapide que *GridSearchCV* et reste tout aussi efficace.

Pour la démonstration, les paramètres calibrés sont :

- **N\_estimators** : Nombres d'estimateurs
- **Max\_Depth** : Profondeur de l'estimateur

Après l'insertion des meilleurs paramètres (avec la méthode *Model.best\_params\_*) pour chaque modèle, des matrices de confusions ont été réalisées.

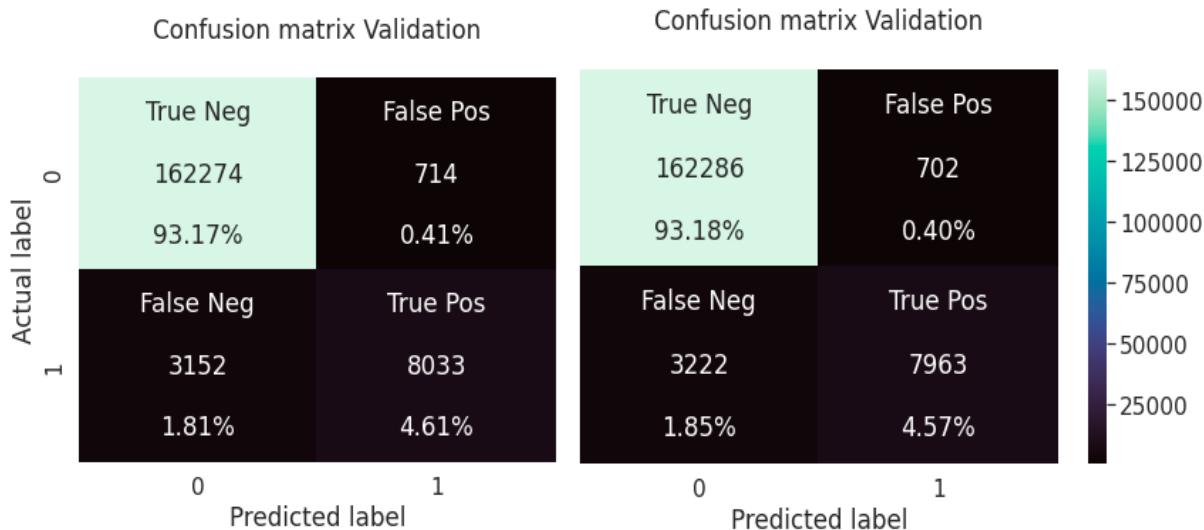


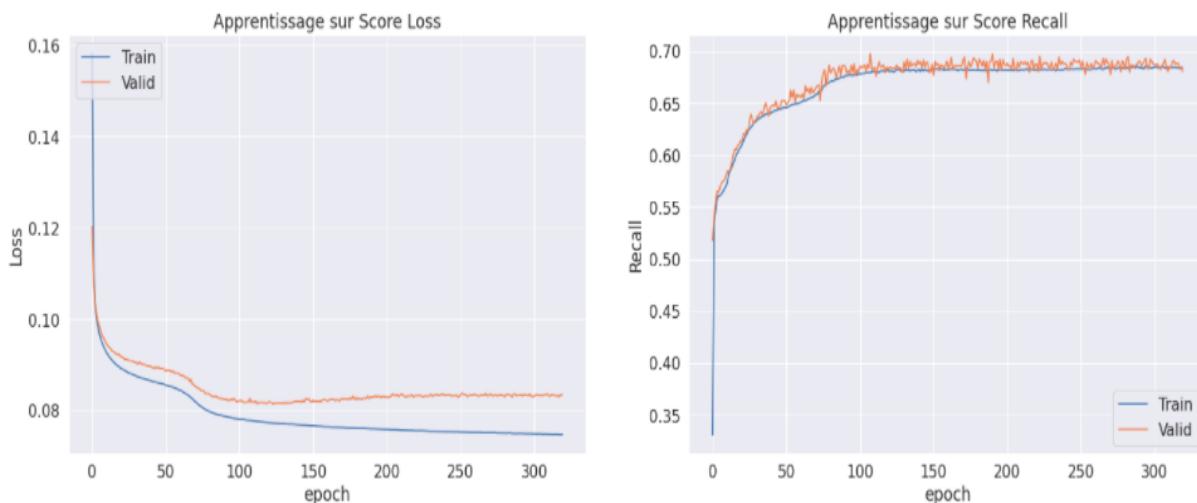
Figure 19. Matrice de confusion des modèles optimisés XGBoost (à gauche) et Light GBM (à droite).

#### 3.3.2.2. Conception de Réseaux de Neurones Artificiels

Trois réseaux de neurones ont été construits sous *Tensorflow.keras* et possèdent le même nombre de couches. Les performances ont été mesurées sur les mêmes métriques que pour les modèles de

*Machine Learning.* Ces modèles diffèrent sur le nombre de neurones dans chaque couche ou encore sur le *learning\_rate*.

L'architecture de chaque réseau reste globalement similaire, composée d'une couche d'entrée avec un *input\_shape* de même dimension que les *Features* (88,) constituant la matrice *X\_train*. Une fonction d'activation reLU (activation linéaire standard) a été définie en sortie de chacun de ces neurones. Deux couches intermédiaires sont intégrées au réseau. Enfin une couche de sortie est appliquée et possède une fonction d'activation *Softmax*. L'*optimizer* (descente de gradient) choisi est *Adamax*, une extension de la version Adam, conçue pour accélérer le processus d'optimisation. La fonction de perte associée (*Loss*) est *binary\_crossentropy* prévu pour les problèmes de classification binaire. Le nombre d'*Epoch* est fixé à 320 pour un *batch\_size* de 256.



**Figure 20.** Learning Curve sur le MultiLayer Perceptron 1. A gauche les courbes de progression sur *Loss*, à droite les courbes de progression sur la métrique *Recall*. En bleu, les courbes d'apprentissage sur le set *Train*, en orange, les courbes de validation (set *Valid*).

Après l'apprentissage, le modèle le plus performant semble être *MLPerceptron Model 1* (*Density input* : 20, *Density hidden1* : 15, *Density hidden2* : 10, *learning rate* :  $1.10^{-3}$ ).

### 3.3.2.3. Quelques tests sur les modèles *Stacking* avec SKLearn

Trois modèles *Stacking* ont été construits en utilisant, en guise d'estimateurs primaires, les meilleurs modèles ML conçus en amont durant la sélection par validation croisée. Ainsi, les algorithmes XGBoost et LightGBM Classifier optimisés ou non ont été incorporés. Les estimateurs finaux sont Logistic Regression ou LightGBM.

Il s'avère que le métamodèle composé des estimateurs primaires XGBoost et LightGBM optimisés avec un estimateur final Logistic Regression soit le plus performant des trois.

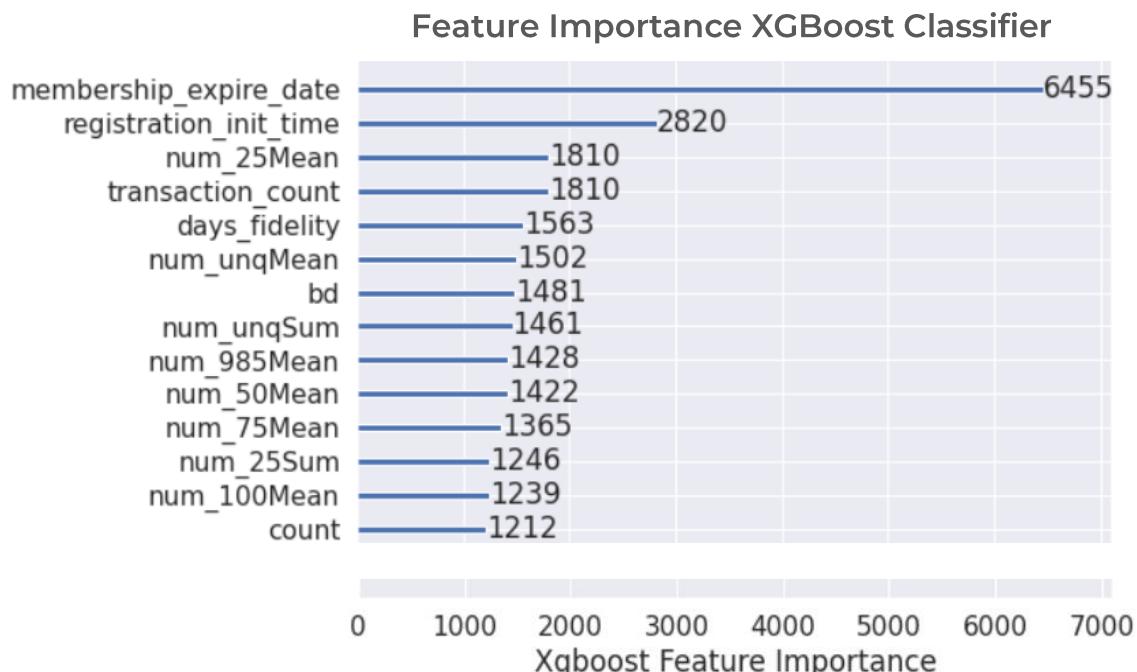
### 3.3.3. Amélioration du modèle sélectionné

Ce sont ainsi près de 16 modèles qui ont été entraînés et évalués. Le meilleur modèle est un XGBoost Classifier optimisé avec les paramètres *learning\_rate* 0.05, *max\_depth* 9 et *n\_estimators* à 200. Il montre de bonnes performances sur le set *VALID* pour la majorité des métriques d'évaluation.

**Tableau 1.** Ensemble des performances des 16 modèles sur le set *Valid* en fonction des métriques.

Modèles	ROC AUC	Log Loss	F1	Precision	Recall	MCC
Optimal XGBC	0,980	0,068	0,806	0,918	0,718	0,801
LightGBM Classifier	0,980	0,069	0,804	0,920	0,714	0,800
Optimal LGBMC	0,980	0,069	0,802	0,919	0,712	0,798
MLPerceptron Model 1	0,969	0,080	0,773	0,912	0,671	0,770
Stacking Model 2	0,980	0,082	0,807	0,918	0,720	0,802
XGBoost Classifier	0,970	0,083	0,757	0,936	0,635	0,759
Stacking Model 1	0,980	0,083	0,802	0,912	0,716	0,797
MLPerceptron Model 3	0,966	0,084	0,770	0,903	0,671	0,766
MLPerceptron Model 2	0,965	0,085	0,767	0,906	0,664	0,764
Random Forest Classifier	0,970	0,100	0,785	0,936	0,675	0,784
Stacking Model 3	0,979	0,100	0,801	0,899	0,723	0,794
Logistic Regression	0,927	0,125	0,661	0,886	0,527	0,668
Bagging Classifier	0,941	0,267	0,797	0,905	0,713	0,792
AdaBoost Classifier	0,962	0,647	0,684	0,863	0,567	0,684
Decision Tree Classifier	0,853	1,309	0,712	0,696	0,728	0,691
Naive Bayes Classifier	0,853	1,327	0,631	0,816	0,515	0,630

XGBoost possède une méthode *Features\_importance*, permettant de déterminer les variables ayant le plus de « poids » sur les prédictions du modèle. Les informations résultantes peuvent être utilisées pour effectuer une réduction de dimension du Dataset. Ce processus permet au modèle de garder une certaine généralité et réduit les problèmes de bruit que certaines variables non cruciales peuvent provoquer.

**Figure 21.** Graphe généré par *XGBoost.Feature\_importance*.

*Feature\_selection.SelectFromModel* permet d'appliquer un seuil de sélection de *Features*. Pour chaque seuil fixé (en enlevant la variable la moins importante à chaque fois), le modèle a été entraîné. Il s'avère que le modèle restait tout aussi performant en ne retenant que les 30 variables les plus importantes.

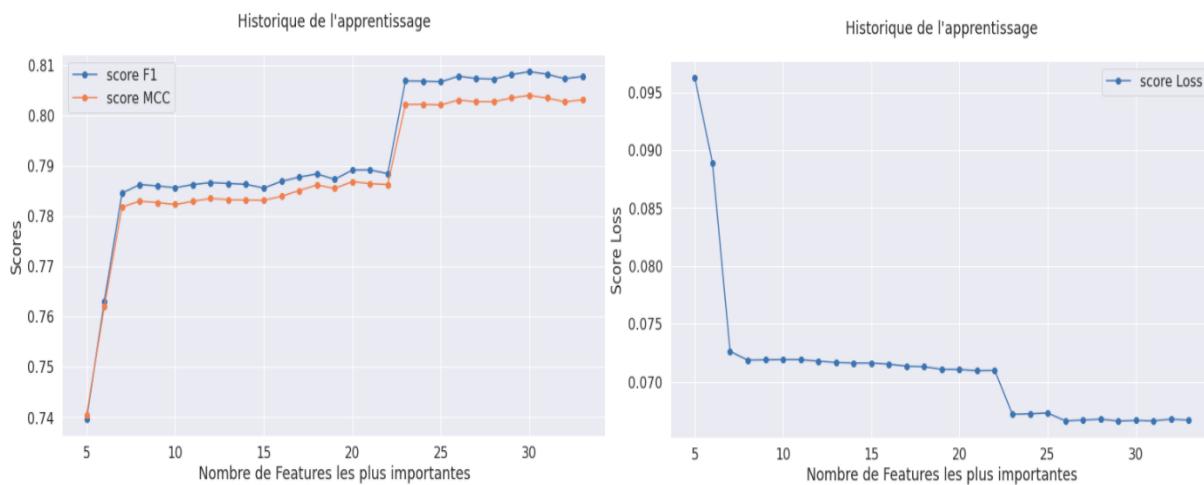


Figure 22. Performance du modèle XGBoost Optimisé en fonction du nombre de variables les plus importantes.

Une comparaison des modèles XGBoost non optimisés, optimisés et avec sélection de variable a été réalisée en étudiant la *Precision/Recall curve*, la *ROC AUC curve* ou encore les matrices de confusions associées (annexe 15 page 67). Ce dernier modèle semble être légèrement plus performant (moins de Faux-Négatifs, plus de Vrais-Positifs).

### Part de Faux Négatifs et Vrais Positifs VALID en fonction des probabilités de prédiction

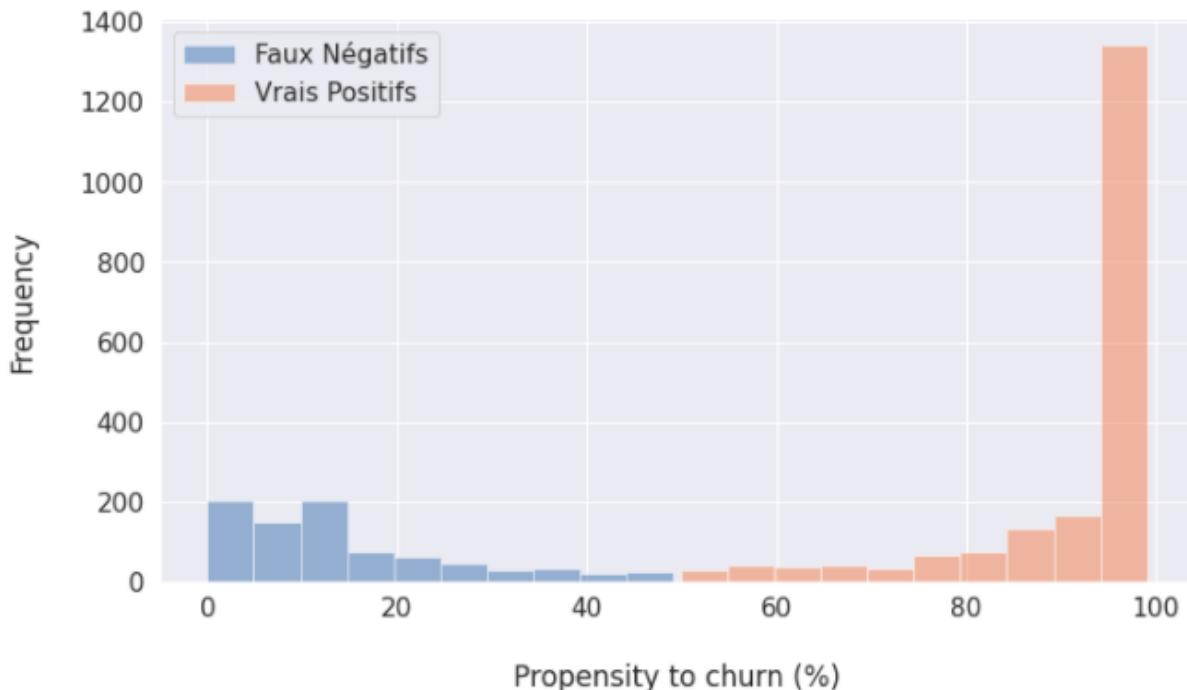


Figure 23. Proportion des clients désabonnés réels en fonction du score de risque généré par le modèle IA. En bleu les Faux Négatifs (inférieur à 50% de probabilité de partir), en orange les Vrais Positifs (supérieur à 50% de probabilité de partir).

### 3.3.4. Autres axes étudiés

Dans le cadre de classifications binaires déséquilibrées, d'autres stratégies peuvent être appliquées en fonction des performances recherchées par l'intermédiaire de certaines métriques. Ces axes se concentrent sur l'augmentation du score *Recall* (maximiser le taux de Vrais-Positifs). Bien que le principal objectif fût d'obtenir un modèle ayant un *Log Loss* minimisé, il était intéressant de voir d'autres possibilités.

Quatre études ont été réalisées de façon succincte :

- Echantillonnage permettant l'équilibrage des classes avec les outils *Imblearn*.
- Utilisation d'algorithmes prévus pour le traitement de classes déséquilibrées.
- Modification du paramètre *Scale\_pos\_weight* d'un modèle XGBoost Classifier.
- Application d'un seuil de probabilité par l'étude du *MCC/F1 curve*.

#### 3.3.4.1. Echantillonnage des données d'entraînement

Plusieurs méthodes d'*undersampling*, *oversampling* et hybridation ont été utilisées pour rééquilibrer les classes de la variable *is\_churn*. Pour le sous-échantillonnage, des techniques comme *NearMiss-3*, *Condensed Nearest Neighbor* ou *TomekLinks* peuvent être utilisées pour sélectionner les données de la classe majoritaire.

Des méthodes de sur-échantillonnage de la classe minoritaire ont aussi été envisagées. La première est de tout simplement découpler les données pour atteindre un ratio équilibré. Une autre option est de créer des données synthétiques de la classe minoritaire par des algorithmes tels que SMOTE (et ses alternatifs) ou encore ADASYN.

Les entraînements de modèles ont été effectués par validation croisée. Les résultats montrent une augmentation globale des scores moyens *F1* et *Recall* au détriment des scores *AUC*, *Log Loss*, *Precision* et *MCC*. Les scores intermédiaires de chaque pli présentent toutefois une grande disparité d'un échantillon à l'autre (notamment pour la méthode de sous-échantillonnage).

Comparaison Score MCC avec *NearMiss-3*

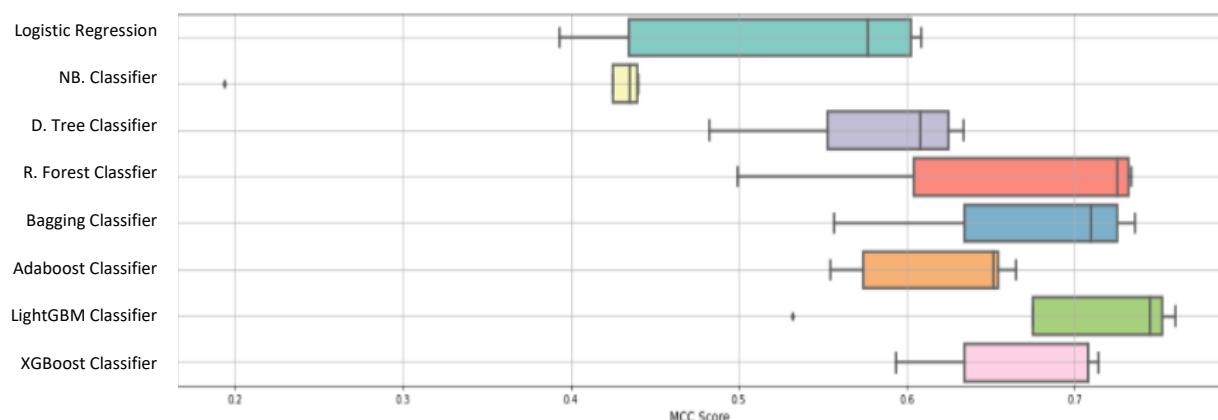


Figure 24. Comparaison après validation croisée sur le score MCC pour chaque modèle. Un exemple de résultats à partir d'un sous-échantillonnage *NearMiss-3* de la classe majoritaire.

### 3.3.4.2. Modèles *Imblearn*

*Imblearn* propose divers modèles spécialisés contrant ces phénomènes de déséquilibre de classes. Deux modèles avec des paramétrages à défaut (*Ensemble* et *Boosting*) sont proposés. Au vu des matrices de confusion générées en comparaison des prédictions avec les valeurs cibles réelles sur set *VALID*, la part de Faux-Positifs reste trop importante malgré une augmentation de la part de Vrais-Positifs (annexe 18 page 69).

### 3.3.4.3. Etude préliminaire sur *Scale\_pos\_weight* XGBoost Classifier

Une calibration du paramètre *Scale\_pos\_weight* de XGBoost peut être utilisée afin d'ajouter, comme son nom l'indique, plus de « poids » sur les données de la classe minoritaire. Ce paramètre est surtout utile si la stratégie de l'entreprise est d'obtenir une IA de détection de l'attrition minimisant le taux de Faux-Négatifs. Cependant, la métrique *Log Loss* indiquera une augmentation générale des erreurs et donc une baisse de performance.

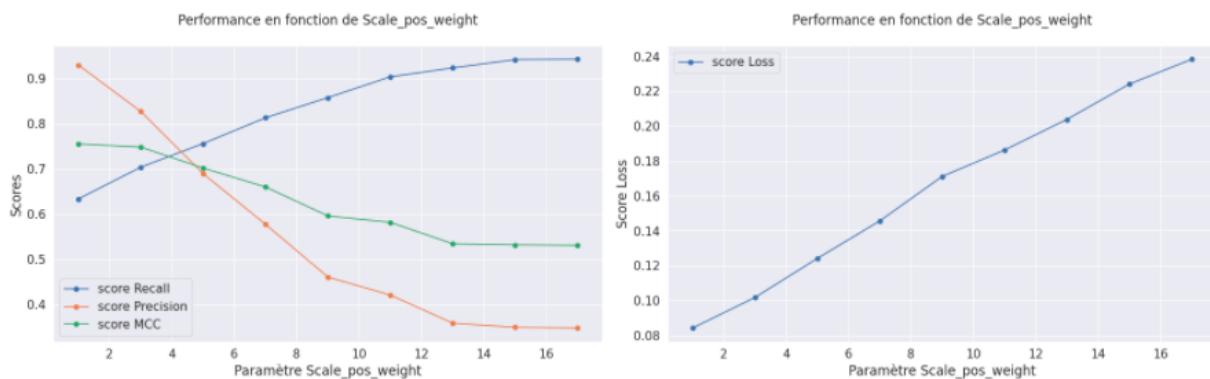


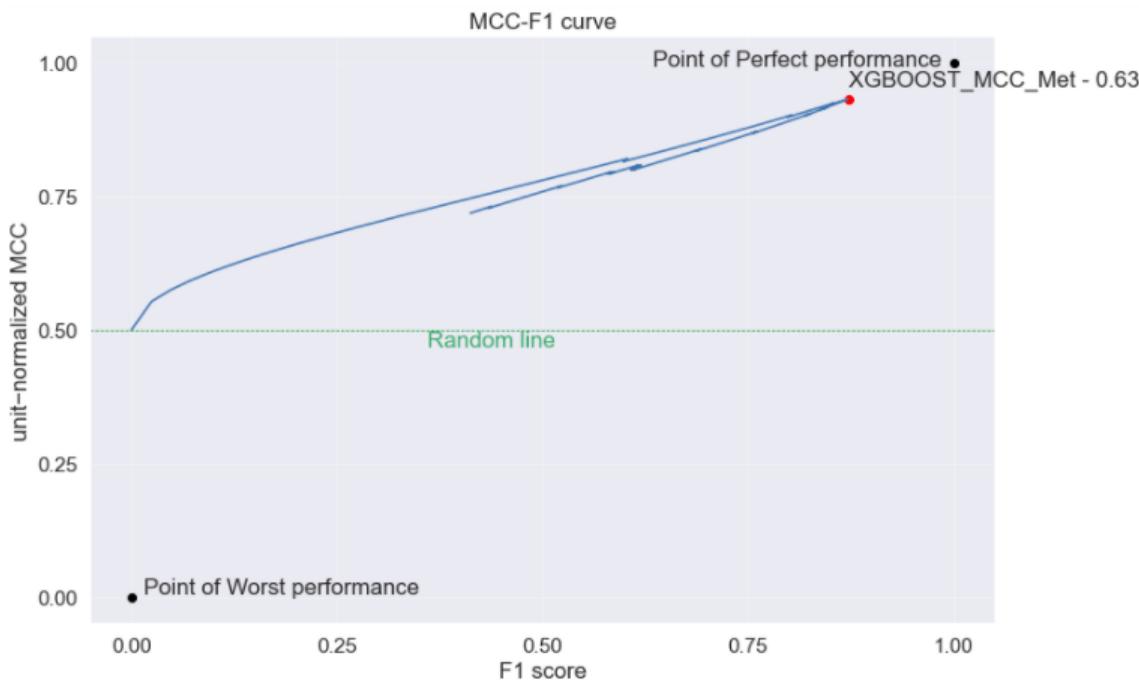
Figure 25. Performances de XGBoost Classifier sur les métriques *Recall*, *Precision*, *MCC* et *Log Loss* en fonction du paramètre *Scale\_pos\_weight*.

### 3.3.4.4. Définition du seuil de décision *MCC/F1 curve*

Les probabilités de prédiction permettent d'appréhender le risque qu'un client se désabonne ou reste abonné aux services de KKBOX. En terme général, si la probabilité de prédiction est supérieure à 0.5 (50%) pour un client, celui-ci est classé dans la catégorie des désabonnés, à l'inverse, il sera placé dans l'autre catégorie. Ce seuil peut être modifié en analysant par exemple la courbe *MCC/F1*. Le seuil sera fixé de sorte d'optimiser le score *MCC* du modèle en minimisant les parts de Faux-Positifs et Faux-Négatifs.

En s'inspirant de l'étude de Cao, Chicco, & Hoffman de 2020, le meilleur seuil a été relevé. Sur une portion du set *VALID*, ce seuil serait placé à 0.34. Ainsi, les individus ayant un score de risque supérieur à 34% seront considérés comme désabonnés. En comparaison, les scores *MCC* et *Recall* passe respectivement de 0.80 et 0.72 (pour un seuil à 0.5) à 0.85 et 0.80 (pour un seuil à 0.34).

Dans le cadre de ce projet, seule la probabilité de prédiction est retenue mais cette étude reste pertinente si on se focalise sur les scores *MCC* et *F1*.



**Figure 26.** MCC/F1 curve généré sur la portion du set *Valid* avec 10000 instances et gardant le même ratio entre les classes (d'après de Cao, Chicco, & Hoffman. 2020). MCC-F1-Met est le rapport de la distance moyenne du score *MCC-F1* au point de performance parfaite (1,1). Il s'agit d'une métrique pour comparer les performances du modèle.

### 3.4. Bilan sur la modélisation

Il existe donc de nombreuses possibilités pour répondre aux objectifs du projet. À travers les différentes stratégies menées, le modèle final sélectionné est le XGBoost Classifier optimisé traitant 30 variables. Ses scores de performance (en privilégiant principalement le *Log Loss*) sont les plus élevés parmi les différents modèles évalués. Pour le développement de l'application, le modèle et la liste des variables retenues ont été enregistrés sous *Joblib*.

## 4. Construction de la base de données relationnelle

Deux bases de données relationnelles MySQL ont été construites pour la validation du projet. La première BDD a été créée directement à partir des fichiers CSV mis à disposition sur le site *Kaggle*. Le nombre de données étant conséquent, cette BDD "échantillon" est réalisée pour la validation des compétences associées.

La deuxième BDD relationnelle contient des données prétraitées du Dataset Final construit en fin d'analyse. Cet ensemble de données correspondent d'ailleurs au set *TEST* créé en amont de la modélisation et n'ont pas été utilisées pour l'entraînement ou l'évaluation du modèle.

Deux autres BDD non relationnelles ont aussi été construites afin d'envoyer des requêtes pour l'insertion et mise à jour de données pour les besoins de l'application. Ces autres bases seront sommairement détaillées dans la 3<sup>ème</sup> Partie présentant l'application.

*Des scripts de création de BDD et insertion de données sont disponibles en annexes et sur [GitHub](#).*

## 4.1. Base de données relationnelle analytique originale

Cette étape présente la marche à suivre pour l'intégration des CSV dans une base de données relationnelle.

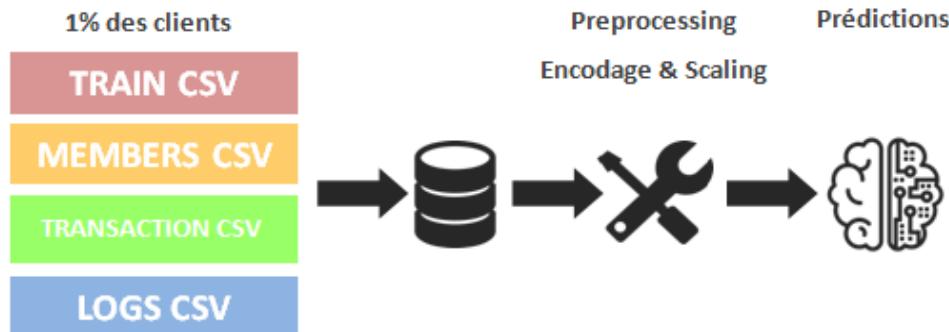


Figure 27. Schéma du processus d'intégration des données dans la BDD et traitement par le modèle en aval.

Le nombre d'instance étant conséquent, 1% des données clients sont introduites dans la BDD pour présenter les requêtes de création et d'introduction de la donnée. L'application souhaitant démontrer l'envoie de requêtes instantanées à travers différentes rubriques, le chargement prendrait un certain temps. La seconde base de données a été créée avec des données prétraitées (réduisant le temps d'exécution et d'affichage des résultats dans l'application).

### 4.1.1. Echantillonnage de démonstration

La méthode *Train\_Test\_Split* est réalisée pour récupérer 1% des données sur le fichier *Train\_v2.csv*, contenant des ID clients uniques (*msno*). La séparation est faite de telle sorte que le ratio de la variable cible *is\_churn* soit conservé. Les variables temporelles (format *string*) observées en amont sont converties en *datetime* pour leur future insertion en BDD.

Pour faire la connexion entre les différentes tables, un ID de type *Integer* est attribué pour chaque identifiant *msno* unique. La nouvelle variable est donc rajoutée aux Dataframes *logs*, *transactions* et *members*.

### 4.1.2. Exportation des données en BDD

La base de données a été initialisée par un script Python avec *mysql.connector*. Il permet de créer la base de données et les tables correspondantes. Par l'intermédiaire de requêtes SQL, les données peuvent être exportées vers la BDD (annexe 22 page 73, avec la requête *INSERT INTO*).

#### 4.1.2.1. Création de la BDD

Ces tables reprennent la structure des 4 fichiers CSV. Une table *User* comprend les identifiants *msno* et ID auto-incrémenté de la table nommé *User\_ID* qui fait le lien avec les autres tables. Les 3 autres tables comprennent respectivement les données de *logs*, *transactions* et *members*. Cette dernière

table contient aussi la cible *is\_churn*. Des tables complémentaires sont reliées pour *City*, *Register*, *Payment\_method*. Enfin une table *Time\_save* fait référence aux données traitées (ici Mars 2017).

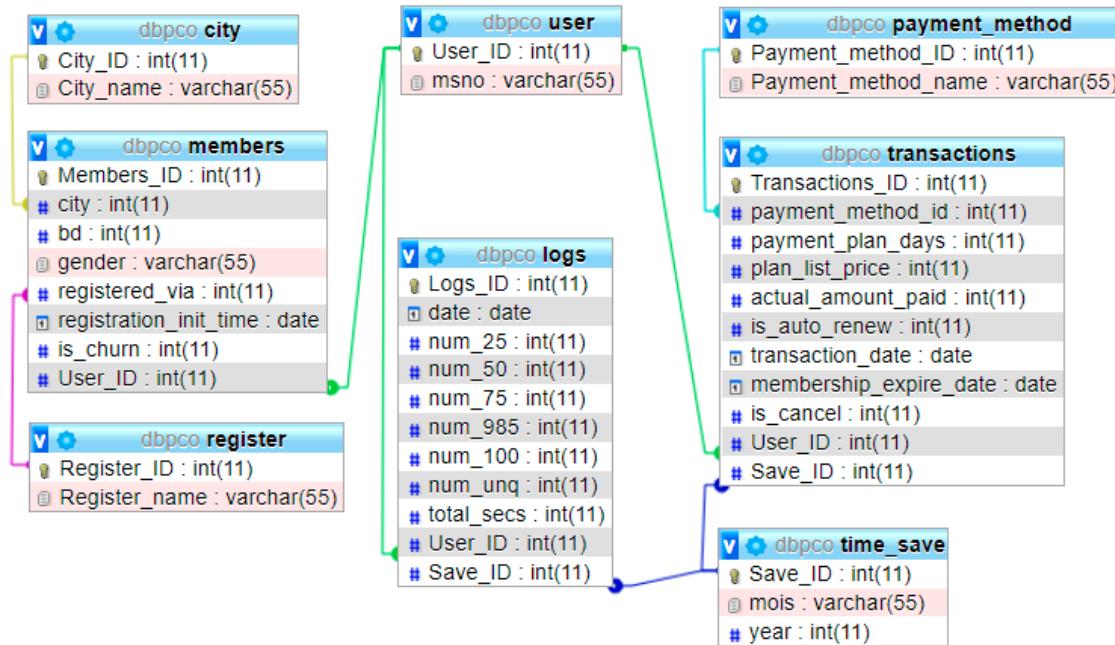


Figure 28. Architecture de la base de données relationnelle originale DBPCO sur l'interface *PhpMyAdmin*.

#### 4.1.2.2. Exportation des données vers la BDD

Un ensemble de requêtes restitue les données de chaque Dataset dans leur table respective par l'intermédiaire de scripts Python. Une fois dans la BDD, les données peuvent être ainsi exportées et prêtes pour l'analyse initiale si toutefois la totalité des clients se trouvaient dans la BDD.

## 4.2. La base de données relationnelle intégrée à l'application

Les données intégrées dans cette base de données reliée à l'application proviennent du set *TEST* du Dataset Final. Les données ont donc subi un traitement durant l'analyse effectuée en amont. En reprenant le même principe que pour la BDD relationnelle originale, 4 tables ont été créées en s'inspirant des fichiers CSV.



Figure 29. Schéma du processus d'intégration des données dans la BDD prévue pour l'application et traitement par le modèle en aval.

Cette BDD DATABASEKKBOX comprend elle aussi des tables *User*, *Transactions*, *Logs*, *Members*. Une fois les données correctement importées, elles peuvent être disponibles par requêtes afin de les convertir en Dataframe. En comparaison avec la base DBPCO, les tables complémentaires *Time\_save*, *Payment\_method*, *City* et *Register* n'ont pas été ajoutées car non nécessaires pour la modélisation.

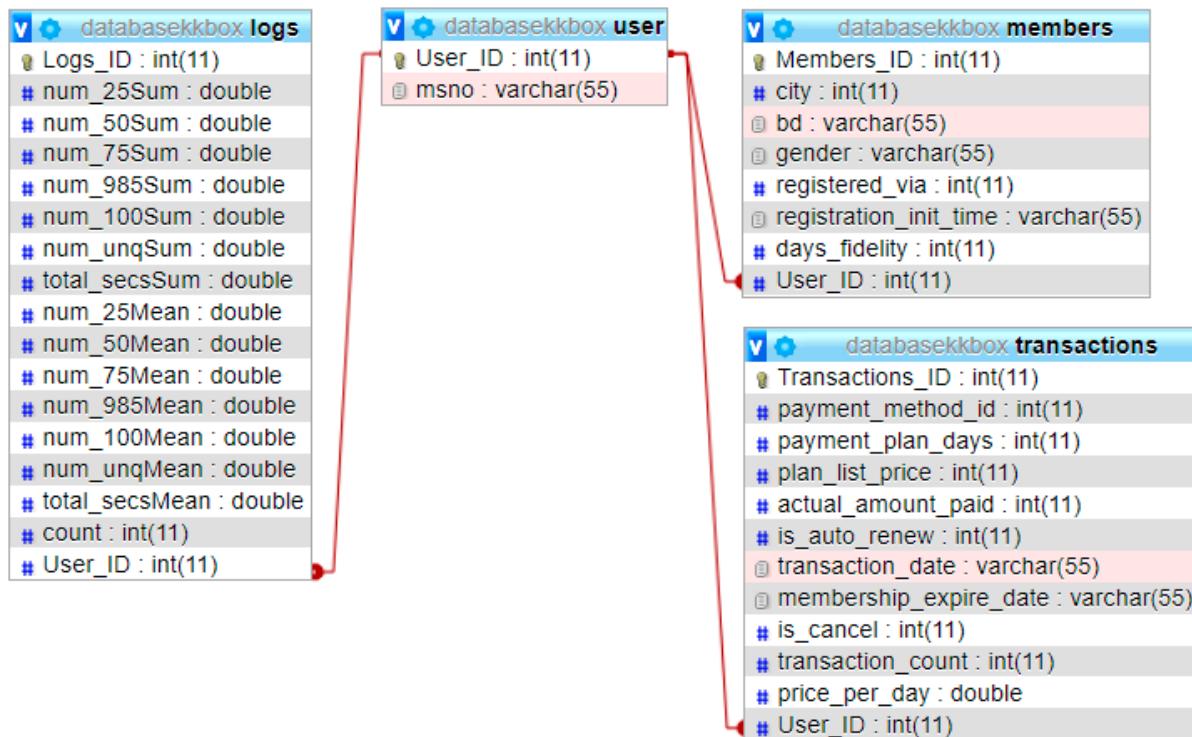


Figure 30. Architecture finale de la base de données relationnelle DATABASEKKBOX pour la démonstration de l'application sur l'interface PhpMyAdmin.

## 5. Test BDD et Prédiction de l'IA sur Notebook

Afin de s'assurer que les requêtes permettent la formation d'un Dataset de même configuration que le Dataset Final créé en amont, des tests ont été réalisés. Ce jeu de données pourra ensuite être traité par le modèle pour émettre des prédictions de départ utilisateur.

### 5.1. Exportation des données depuis la BDD d'application

L'application donne la possibilité de sélectionner les clients selon leur date d'expiration d'abonnement en précisant le mois et l'année. Dans cet exemple, les scores de risque générés par le modèle pour des clients ayant une date d'expiration d'abonnement d'Avril 2017 sont affichés dans un tableau. Bien entendu, l'identifiant des clients y est associé afin que des mesures personnalisées puissent être prises afin d'éviter leur départ.

Les tables sont exportées sous forme de Dataframes. *Members*, *Transactions* et *Logs* sont joins à la table *User* pour afficher l'identifiant *msno*. Aussi, la requête associée à l'exportation des tables *Transactions/User* comprend des conditions (*WHERE*) où le mois et l'année d'expiration d'abonnement sont définis sur Avril 2017 dans cet exemple.

Les Dataframes sont ensuite rassemblés en un seul Dataset avec la fonction *Pandas.merge* sur *msno*. Une dernière vérification s'assure que le nouveau jeu de données ne contient pas de valeur manquante.

```

SELECT
    Transactions.payment_method_id,
    Transactions.payment_plan_days,
    Transactions.plan_list_price,
    Transactions.actual_amount_paid,
    Transactions.is_auto_renew,
    Transactions.transaction_date,
    Transactions.membership_expire_date,
    Transactions.is_cancel,
    Transactions.transaction_count,
    Transactions.price_per_day,
    User.msno

FROM Transactions
JOIN User on Transactions.User_ID = User.User_ID
WHERE
    MONTH(str_to_date(Transactions.membership_expire_date, '%Y-%m-%d')) = %s
    AND
    YEAR(str_to_date(Transactions.membership_expire_date, '%Y-%m-%d')) = %s

```

Figure 31. Requête SQL pour la formation du Dataframe *Transactions/User* avec conditions sur la variable date d'expiration d'abonnement. Les variables mois et année sont représentées par l'annotation %s (04 et 2017).

## 5.2. Preprocessing sur le Dataset résultant

Pour que le jeu de données puisse être traité par le modèle IA, le Dataset généré subit un traitement au niveau de l'emplacement des colonnes. Le fichier *Joblib* comprenant une liste des colonnes du Dataset Final enregistré en amont, permet de disposer les variables de façon similaire.

Un filtre a aussi été appliqué sur le Dataset Final sur la variable *membership\_expire\_date* pour obtenir uniquement les clients ayant une expiration d'abonnement sur le mois d'Avril 2017. Les deux Datasets sont ensuite comparés avec la fonction *Pandas.equal* qui retourne *True* s'ils sont strictement similaires. Cette vérification permet de s'assurer que la requête exportait correctement le jeu de données souhaité. Un processus similaire d'exportation de la BDD originale a aussi été réalisé en parallèle pour vérifier que les requêtes étaient cohérentes.

## 5.3. Préparation à l'utilisation du modèle IA

Le Dataset construit depuis la BDD de l'application subit un encodage sur les variables catégorielles (*payment\_method\_id*, *ID registered\_via*, *ID city*) comme lors de l'entraînement du modèle. Un filtre étant appliqué sur le Dataset, plusieurs variables encodées, nécessaires pour la standardisation,

peuvent être absentes du jeu de données extrait de la BDD. De nouvelles colonnes ont donc été générées et ajoutées au Dataset avec des valeurs nulles. Les données ont été remises à la même échelle que celles utilisées pour l'entraînement et la validation du modèle avec le fichier *Scaler*.

```
# Importation des Fichiers de Modèles et Scaler :
scaler = joblib.load('ProcessModel/ScalerXGBC_BF.joblib')
columnscaler = joblib.load('ProcessModel/ColonnesForScale.joblib')
columnsmodel = joblib.load('ProcessModel/ColonnesXGBC_BF.joblib')
classifier = joblib.load('ProcessModel/XGBC_BF.joblib')

# Dataset généré après requêtes dans la BDD & merge :
df = pd.merge(Transactions_sql, Logs_sql, on='msno', how='inner')
df = pd.merge(df, Members_sql, on='msno', how='inner')

# Calibration avec le Dataset de départ ayant servi à la modélisation :
for feature in columnscaler:
    if feature not in df.columns:
        df[feature] = 0

df = df[columnscaler]

# Standardisation des données Test avec la variable scaler :
Identity = df['msno']
X_test = df.drop(['msno', 'is_churn'], 1)
X_test2 = pd.DataFrame(scaler.transform(X_test.values))
X_test2.columns = X_test.columns.values
X_test2.index = X_test.index.values
X_test = X_test2

# Sélection des variables traitées par le modèle :
X_test = X_test[columnsmodel]
```

Figure 32. Script de traitement et standardisation des données avant passage dans le modèle.

Le modèle effectuant des prédictions à partir des 30 « meilleures » variables, un fichier *Joblib* répertoriant ces variables dans une liste permet de réduire les dimensions sur *X\_test*.

## 5.4. Prédictions obtenues

Le risque du départ de chaque client est finalement indiqué dans un tableau. Les résultats proviennent de la probabilité de prédiction obtenue par le *classifier*. Les clients ayant un risque élevé sont affichés en haut du tableau. L'ID de chaque client sauvegardé en amont est affiché dans la table des résultats avec la variable *membership\_expire\_date* qui assure que le filtre a été correctement appliqué sur Avril 2017.

```
# Prédictions de la classe et probabilité de prédiction :
y_pred = classifier.predict(X_test)
probability = classifier.predict_proba(X_test)
probability = probability[:,1]

final_results['propensity_to_churn(%)'] = probability
```

Figure 33. Script des probabilités de prédiction sur  $X_{test}$ .

Tableau 2. Tableau de résultats (échantillon sur 20 clients) indiquant l'identifiant utilisateur, la date de fin d'abonnement et le score de risque généré par le modèle XGBoost.

msno	membership_expire_date	propensity_to_churn(%)
vD8lxPe7PJeg	2017-04-03	97
pBYTvH5BBPPZ	2017-04-01	92
M2cEruZfBjrs	2017-04-17	88
uF7mYLP6xBtw	2017-04-06	81
ufdsaZ2WLTfK	2017-04-16	76
fVlkYixHk3Sr	2017-04-02	71
xw2CUoHu8XT8	2017-04-12	68
43gsEFNWGVqa	2017-04-12	65
RfqkWhA44gRW	2017-04-12	59
TEjlJmPQ5Jr	2017-04-25	56
RsZoIElwFrMr	2017-04-09	48
QwRW4AwW7j9s	2017-04-16	43
PEd6AZVALcTG	2017-04-02	37
mUHCMhZavKhW	2017-04-02	31
WsrNuY73Ec4g	2017-04-02	27
o7DFqQoy7hbT	2017-04-07	22
Zxw82f+WQzbo	2017-04-12	12
hrdtrI2CLPa0	2017-04-13	11
9ed3oYZzgsvw	2017-04-02	1
3l/OKBhs7eaV	2017-04-13	1

Les tests ne retournant aucune erreur, l'intégration des scripts Python au *Back-End* de l'application peut être effectuée. Bien entendu, d'autres tests ont été réalisés avec *Pytest* (annexe 25 page 83) par exemple, en s'assurant que chaque fonction construite ne provoque aucune anomalie lors du développement de l'application. A cette fin, une dernière partie est consacrée à la construction de l'application qui affiche entre-autres, les résultats sur une interface graphique.

# PARTIE 3 : Application du projet

## 1. Développement d'une application d'IA

L'application a été confectionnée sous divers langages et Frameworks. Le *Front-end* a été construit avec HTML5, CSS3 et le Framework Bootstrap 4. Pour rendre le contenu dynamique, du JavaScript a aussi été incorporé, notamment en intégrant les graphiques conçus sous *HighCharts*. Evidemment, la gestion de la donnée, stockée sur MySQL, a été réalisée sous Python avec le Framework *Flask*, permettant de faire le lien entre *Front* et *Back-end* (API). L'ensemble des scripts ont été développés sur Visual Studio Code.

### Flask



Un micro-Framework open-source de développement écrit en Python. Ce Framework, très léger, a pour objectif de garder un noyau simple mais extensible. De nombreuses extensions permettent d'ajouter facilement des fonctionnalités (*Sessions, Jinja*). L'application n'ayant pour vocation principale que de fournir une interface et une base de données simple, ce Framework est suffisant pour répondre au besoin du projet.

### JavaScript AOS



Contient une dépendance JS, permettant de dynamiser les éléments contenus en scrollant les pages. La bibliothèque open-source permet de personnaliser les animations directement dans les fichiers HTML.

### HighCharts



Contient une dépendance JS, permettant d'incorporer facilement des graphiques dynamiques et de bonnes qualités à partir des données générées en Python. La variable data inscrite dans le code JS peut-être présentée sous forme de Liste ou Dictionnaire. Elle doit donc être convertie dans le script Python de l'application.

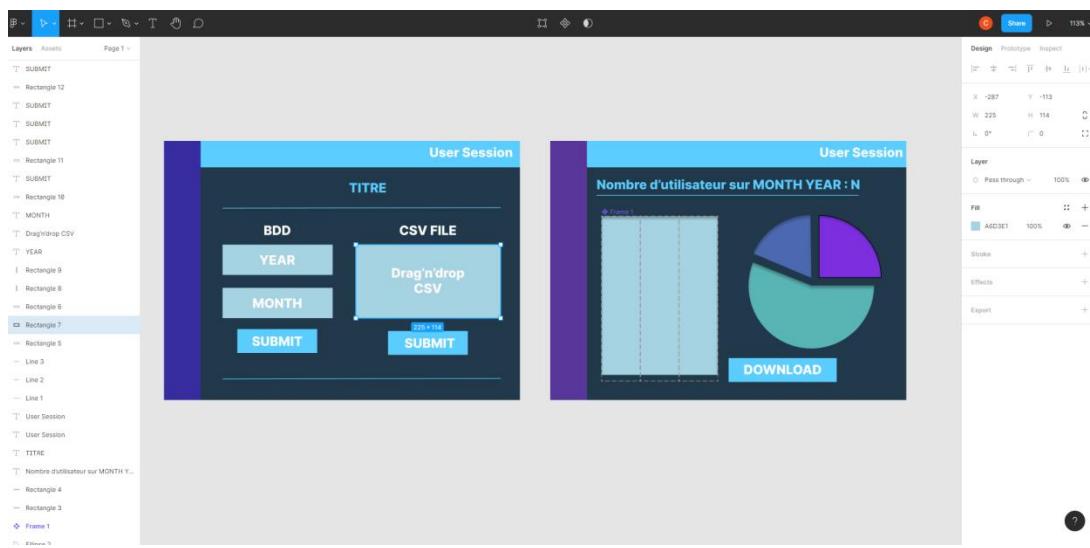
### Bootstrap 4



Un Framework CSS, qui permet d'incorporer des classes HTML préformatées CSS, maintenant une cohérence visuelle. Plusieurs *templates* peuvent être utilisés notamment sur les formulaires, facilitant leur intégration.

## 2. Maquette de l'application

Afin de mener à bien le projet, une maquette sommaire de l'interface graphique a été réalisée dans le but de garder une ligne directrice. La maquette correspond à la principale fonctionnalité de l'application IA. L'utilisateur peut indiquer un mois et une année pour envoyer une requête en base de données, ou ajouter directement un nouveau fichier CSV traité.



**Figure 34.** Ebauches des maquettes du Front de l'application IA, à gauche le formulaire ou zone de chargement de fichier, à droite les résultats fournis par l'IA. Les maquettes ont été réalisées en début de projet sur Figma.

L'utilisateur possède une session de connexion à l'application. Sur la page principale, il peut soit utiliser la BDD ou importer un fichier CSV comportant de nouvelles données. Les résultats comportant les clients et leur risque de désabonnement sont affichés sous la forme d'un tableau téléchargeable. Un graphique présente la proportion des clients en fonction de leur score de risque potentiel.

*Des captures d'écran du prototype et de l'application finale sont disponibles en annexes.*

### 3. Exécution générale de l'application

Le schéma ci-dessous montre le processus de l'application et comment il interagit avec l'utilisateur lors de son utilisation principale :



**Figure 35.** Schéma du processus d'affichage des résultats IA sur l'application.

- 1 L'utilisateur rentre les informations souhaitées dans le formulaire, il s'agit ici de la date d'expiration d'abonnement des clients. Le mois et l'année peuvent être sélectionnés.
- 2 Les informations sont traitées via *Flask* et *Python* pour construire les requêtes conditionnées vers la base de données DATABASEKKBOX. Le processus est similaire au test effectué dans le notebook.
- 3 Les requêtes SQL sont envoyées via Python *mysql.connector*. Elles demandent de récupérer les données clients ayant une date d'expiration d'abonnement contenant le mois et l'année choisis par l'utilisateur.
- 4 Les données sont récoltées et traitées via Python. Un Dataframe global est créé en conséquence. Après traitement des données et standardisation, le jeu de données peut être traité par l'IA.
- 5 Le modèle IA enregistré en amont sous format *Joblib* traite les données d'entrée et renvoie les probabilités de prédictions. Python met en forme les résultats sous forme d'un tableau.
- 6 Les résultats sont renvoyés au *Front*, accessible par l'API.
- 7 Les dashboards correctement générés via JS et *HighCharts* sont affichés sur le support HTML/CSS. Les résultats peuvent être ensuite téléchargés sous format CSV si l'utilisateur le souhaite en cliquant sur le bouton prévu à cet effet.

## 4. Architecture globale et présentation

Pour respecter les compétences associées au projet tout en maintenant la fonctionnalité principale, de nombreux ajouts ont été développés. La structure de l'application est représentée comme telle en répertoriant les utilisations de l'IA et des bases de données :

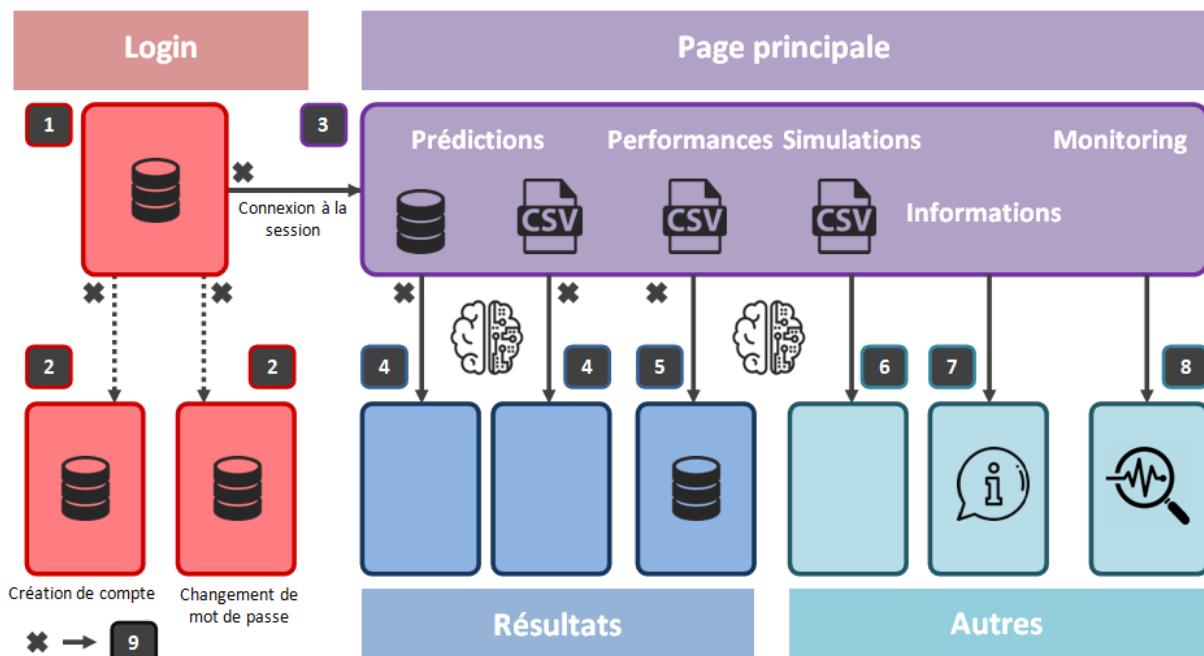


Figure 36. Architecture globale de l'application IA.

## 4.1. Page de connexion

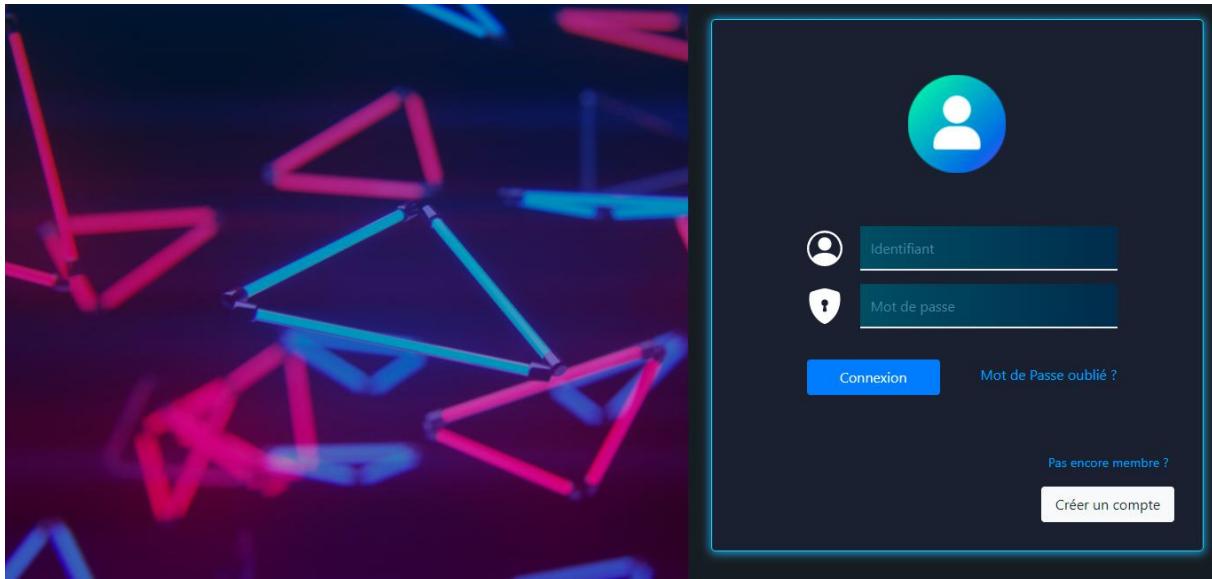


Figure 37. Screen Page de connexion de l'application.

1

Après exécution de l'application, une page Web est retournée où l'utilisateur peut entrer ses identifiants pour accéder à la page principale. Le formulaire, construit à partir d'un *template* Bootstrap, renvoie les informations à l'application :

- POST *login, password*
- Requête vers la base de données AUTHENTICATOR avec *Mysql.connector*
- Script python vérifiant si l'identifiant et le mot de passe associé existe

```
SELECT User_ID, Password FROM login WHERE login.User_ID = {login} AND login.Password = {password}
```

Accès à la page principale /user après ouverture de la Session *Flask* si la requête aboutit, sinon, l'utilisateur est invité à retenter une connexion.

2

Si l'utilisateur ne possède pas de compte, il peut accéder à la page de création. Ses nouveaux identifiants seront insérés dans la base de données prévue à cet effet. Un script Python permet de vérifier si l'identifiant n'existe pas déjà dans la base de donnée AUTHENTICATOR Si l'utilisateur oublie son mot de passe, il peut le changer en entrant son identifiant. Le nouveau mot de passe confirmé remplacera l'ancien mot de passe dans la base de données par une requête *UPDATE*.

```
SELECT User_ID FROM login WHERE login.User_ID = {login}
```

```
UPDATE login SET Password = {password} WHERE User_ID = {login}
```

## 4.2. Page principale

**Figure 38.** Screen Page principale de l'application proposant d'accéder aux prédictions via base de données ou par l'import d'un fichier CSV.

3

Une fois connecté, l'utilisateur arrive sur sa session, sur une page principale expliquant en détail le phénomène d'attrition. Différentes rubriques y sont présentées.

- Prédiction sur données Clients
- Performance du modèle IA
- Simulation démonstrative

## 4.3. Pages de résultats

Identifier	Expiration Abonnement	Risque de Partir (%)
Z7/g62LmWgZz	2017-06-07	99
G4SY3lcPZQwG	2017-06-22	99
t4e0rhKdk1st	2017-06-27	99
0/2b3hOXdrd7	2017-06-13	99
8RAfXANzI2yx	2017-06-21	99
k3/wCaFQDCvd	2017-06-02	99
dl/ibmionaii	2017-06-17	99
fgrNDiljdTcP	2017-06-02	99
VlVodUGoJKBp	2017-06-19	99

**Figure 39.** Screen Page résultats de l'application proposant d'accéder aux prédictions sous forme de tableau téléchargeable et présentant le graphique de proportion de risque sur l'ensemble des clients filtrés sur mois/année.

4

L'utilisateur peut accéder aux prédictions générées par l'IA (*Endpoint* : /database). Soit il peut effectuer une requête en base de données en entrant une date d'expiration d'abonnement, soit en chargeant un CSV conforme comme mentionné précédemment. En reprenant la structure de la maquette, les pages résultantes affichent un tableau récapitulatif téléchargeable et un graphique de la proportion des clients en fonction de leur score de risque d'attrition (en dessous de 25% de risque, entre 25 et 50%, 50-75%, 75%-90%, au dessus de 90%).

L'utilisateur a la possibilité de télécharger les résultats sous format CSV, que ce soit le tableau complet ou le graphique associé.

## 4.5. Page de performance IA



Figure 40. Screen Page performance IA de l'application proposant d'évaluer l'IA sur un nouveau jeu de données sous format CSV.

5

En chargeant un CSV conforme, l'utilisateur peut accéder à une page récapitulant les performances de l'IA sur le Dataset entré (*Endpoint* : /performances). Elle affiche les scores de différentes métriques, une matrice de confusion, une *Precision/Recall* curve, et un histogramme de la répartition du nombre de clients jugés Faux-Négatifs ou Vrais-Positifs en fonction de leurs scores de risque. Lorsque la page est générée, une requête INSERT introduit les nouveaux scores dans une BDD non-relationnelle nommée LOG\_MODEL. Un graphe affichant l'historique des performances du modèle IA est aussi présenté en appelant cette BDD. Elle permet de vérifier que l'IA reste performante dans le temps.

```
INSERT INTO log_model (Date, Log_loss, Score AUC, Score F1, Score MCC, Recall, Precision, Instances, ID) VALUES (Date, loss, auc, scoreF, scoremcc, recall, precision, Total, NULL)

SELECT * FROM log_model
```

## 4.4. Page de Simulation et Informations



Figure 41. Screen Page de simulation de l'application proposant d'accéder aux prédictions de client unique sélectionné aléatoirement à partir d'un jeu de données chargé.

6

A titre démonstratif, l'application se propose d'émettre une prédiction sur un client aléatoire et de la comparer au statut réel de ce dernier (*Endpoint* : /user/simulation). Les données proviennent d'un fichier CSV chargé automatiquement dans l'application (temps de requête inférieur que si le client provient de la base de données).

L'interface montre le score de risque généré par le modèle et compare le résultat avec le statut réel du client. Une icône indique si le modèle émet une bonne prédiction ou non en fonction du client choisi aléatoirement.

7

Une page statique regroupant les informations sur les données d'origine, le type de modèle utilisé, est proposée dans la barre de navigation.

## 4.5. Monitoring (avec *Flask Monitoring Dashboard*)

8

Pour les besoins du projet, un système de monitoring doit être associé à l'application. Une session générée par *Flask Monitoring Dashboard* accessible par l'administrateur (*Endpoint* : /dashboard). Utilisant *Flask*, ce système de monitoring est simple à mettre en place et propose de nombreuses fonctionnalités comme le report du nombre et le temps de requêtes effectuées sur chaque *Endpoint*, le report d'erreur sur ces derniers.

De nombreux graphes sont générés en conséquence afin d'avoir une vision de «l'état de santé» de chaque page de l'application IA. Les données proviennent d'un fichier *flask\_monitoringdashboard.db* généré par ce module de monitoring. Après caractérisation des points critiques de l'application, le niveau de monitoring est défini sur chaque *Endpoint*. Ainsi, les accès les plus sensibles aux erreurs seront davantage contrôlés.

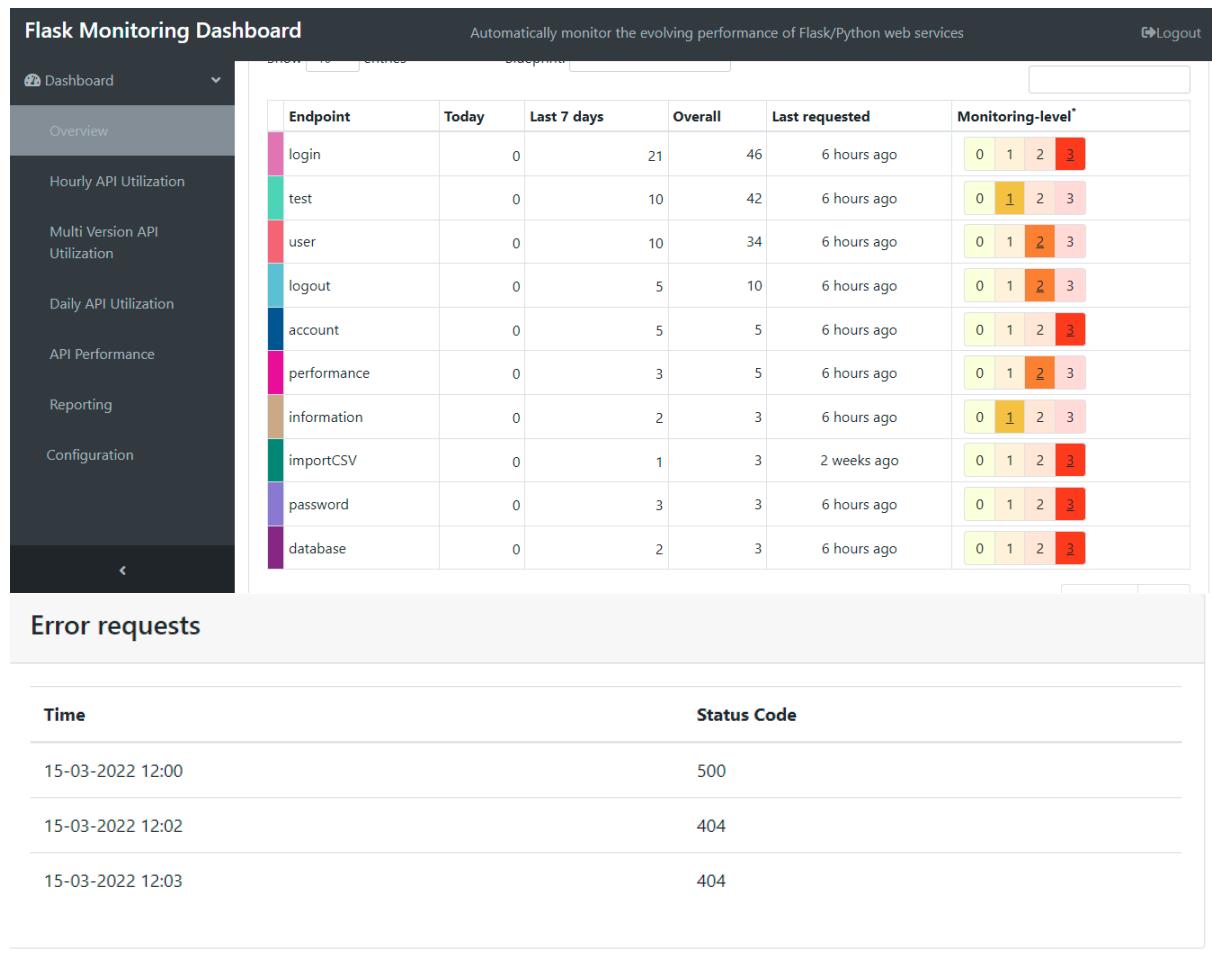


Figure 42. Screens Pages de monitoring de l'application proposant d'accéder aux fonctionnalités sur les performances API et erreurs rencontrées durant la navigation.

## 4.6. Caractérisation des points critiques principaux

9

Une page d'erreur est retournée si une anomalie est présente lors du chargement de l'une des pages dont la flèche de connexion est associée d'une croix. Certaines erreurs ont été identifiées :

- Création d'alerte sur la page de connexion pour identifiant ou mot de passe incorrect.
- Création d'alerte si l'identifiant existe déjà dans la BDD lors de la création d'un compte.
- Problèmes avec la connexion au serveur local.
- Erreur de requête vers la base SQL DATABASEKKBOX (Mois/Année indisponible dans la BDD), le filtre retourne None.
- Fichier importé au format incorrect.
- Structure du Dataset du fichier CSV incorrect.

Chaque retournement à la page d'erreur (*error.html*) envoie une notification par SMS. Ce message est généré par l'API *Twilio*, qui comprend de nombreuses fonctionnalités.

The screenshot shows the Twilio Dashboard interface. On the left, there's a sidebar with categories like 'Journaux', 'les erreurs', 'Journaux d'erreurs', 'Webhooks et alertes', 'Messagerie', 'Appels', 'Conférences', 'Enregistrements d'appels', 'Documents et assistance', and a 'Body' section. The 'Body' section contains the message content: "Sent from your Twilio trial account - Alerte Application : Bad Request. Route : /performance". The main panel is titled 'Properties' and displays the following details:

Message SID	SMd59a6e4	Direction	Outgoing API
Messaging Service	—	API Endpoint	—
Created At	8:27:26 UTC 2022-04-11	Message Segments	1
From	(redacted)	Encoding	GSM
To	(redacted)	Cost	(redacted)
<b>Body</b>			
Sent from your Twilio trial account - Alerte Application : Bad Request. Route : /performance			

**Figure 43.** Screen Dashboard Twilio sur les notifications reçues par SMS. Il indique la route API n'ayant pas aboutit, l'heure d'envoie, et le message détaillé.

The image contains two side-by-side screenshots. The left screenshot is a mobile phone screen showing a list of four received SMS messages. The messages are:

- lundi 11 avril 2022 10:00: Sent from your Twilio trial account - Alerte Application : Le serveur SQL est inaccessible.
- lundi 11 avril 2022 10:24: Sent from your Twilio trial account - Alerte Application : Le serveur SQL est inaccessible. Route : /login
- lundi 11 avril 2022 10:27: Sent from your Twilio trial account - Alerte Application : Bad Request. Route : /importCSV
- lundi 11 avril 2022 10:27: Sent from your Twilio trial account - Alerte Application : Bad Request. Route : /performance

The right screenshot is a code editor window displaying a Python script. The script uses the Twilio REST API to send SMS messages. The code is as follows:

```

20 from twilio.rest import Client
21 from dotenv import load_dotenv
22
23 # Initialisation Twilio Client Alerte
24 load_dotenv()
25 twilio_client = Client()
26
27 # Fonction Alerte SMS
28 def alert(alerte):
29     twilio_client.messages.create(
30         body=f"Alerte Application : {alerte}",
31         from_='+.....',
32         to='+.....')
33
34

```

**Figure 44.** A gauche, Screen des SMS reçus et générés par l'API Twilio. Quatre messages d'alertes ont été envoyés lors de la création d'erreurs (Fermeture serveur SQL, envoie de fichiers non adaptés au traitement etc...). A droite la fonction utilisée dans l'application qui envoie un message personnalisé en fonction de l'erreur rencontrée.

# PARTIE 4 : Gestion du Projet

## 1. Compréhension et caractérisation des besoins

La caractérisation des objectifs techniques a permis de mener à bien le présent projet. Un suivi fréquent des étapes de développement était nécessaire afin de ne pas perdre de vue le but premier de l'application, à savoir détecter des clients susceptibles de se désabonner des services de la plateforme de *streaming* KKBOX. Comme laisse suggérer le plan de ce présent document, la construction de l'application s'est déroulée comme suit :

- La recherche d'un jeu de données portant sur les phénomènes d'attrition.
- Une veille technologique afin de s'imprégner du sujet et appréhender les différents outils existants pour répondre à la problématique.
- Une analyse de la donnée mise à disposition afin de la nettoyer en vue de l'entraînement de modèle IA et définir un profil type des clients par des statistiques et data-visualisation.
- Un travail conséquent sur la modélisation, afin de sélectionner le modèle capable de classer au mieux les clients pour un usage sur des données futures.
- La construction de bases de données relationnelles afin de stocker les données comme le ferait une telle entreprise.
- La réalisation de l'interface en appliquant différents outils de développement.
- La mise en place d'un système de monitoring sur l'IA et de l'application en général.

Un *Trello* a ainsi été confectionné en début de parcours, en renseignant les différentes compétences associées à la certification. Ces compétences sont regroupées en fonction du type de travaux à effectuer (préparation au projet, analyse, modélisation, BDD, développement de l'application). Ceci permet de segmenter et faciliter les tâches à réaliser.

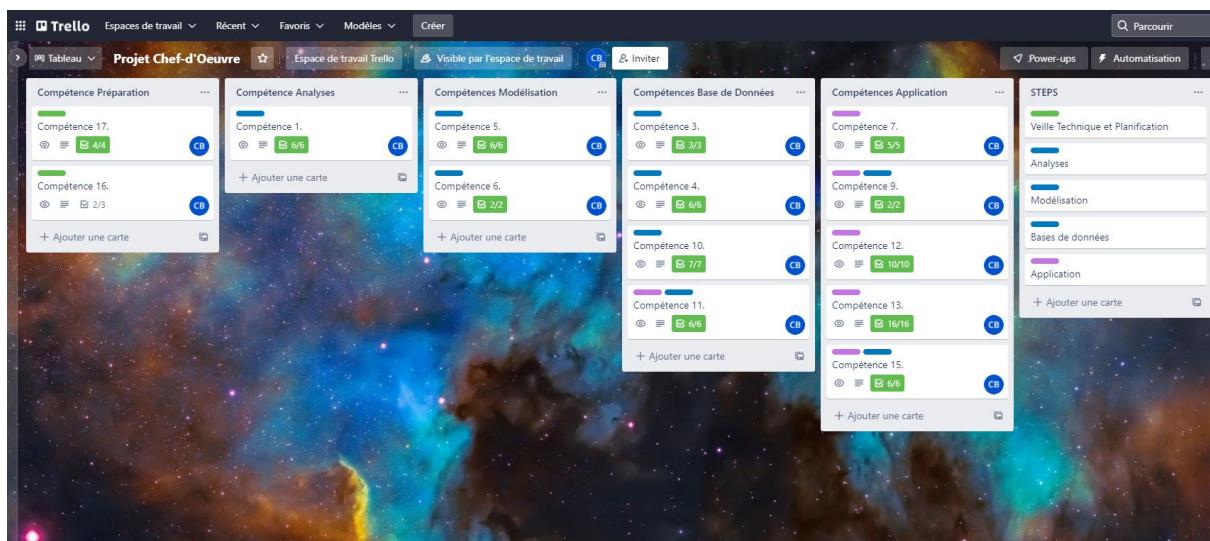


Figure 45. Suivi de projet en caractérisant les compétences sur l'interface Trello.

Les *Users Stories* ont été définis en conséquence ainsi que leur priorisation. Les tâches associées à chaque compétence ont pu être caractérisées afin de répondre aux sous-objectifs. Bien évidemment, au fur et à mesure que le projet avançait, plusieurs tâches ont été ajoutées, correspondant à de nouveaux axes d'analyses, certains tests pour la validation de diverses compétences ou encore de nouvelles fonctionnalités intégrées dans l'application IA.

The screenshot shows a Trello card for 'Compétence 13'. The card has a title 'Compétence 13.' and a subtitle 'Dans la liste Compétences Application'. It includes sections for 'Membres' (CB), 'Étiquettes' (purple), and 'Dates' (1 déc. 2021 - 1 févr. à 18:45, respectée). A 'Description' section contains the text: 'Développer le front-end de l'application d'intelligence artificielle à partir de maquettes et du parcours utilisateur-rice, dans le respect des objectifs visés et des bonnes pratiques du domaine.' To the right, there's a sidebar with options like 'Ajouter à la carte', 'Membres', 'Étiquettes', 'Checklist', 'Dates', 'Pièce jointe', 'Image de couvert...', 'Champs personna...', 'Commencer l'essai gratuit', 'Power-Ups', 'Automatisation', and 'Ajouter un bouton'. A large checklist on the left lists tasks such as 'Architecture de l'interface', 'Construction des maquettes', 'Création HTML-CSS', etc., each with a progress bar indicating completion.

**Figure 46.** Suivi de la Compétence 13. Les éléments sont cochés au fur et à mesure de leur intégration à l'application IA. Des tâches peuvent être ajoutées durant le développement.

## 2. Planification du Projet

Pour chaque segment, leur charge de travail a été définie. Le projet a été finalisé en 7 sprints d'environ 2 semaines chacun. Certains sous-objectifs ont demandé plus de temps au vu de la quantité de données mise à disposition, le nombre de modèles IA évalués ou encore par l'ajout de fonctionnalités qui n'était pas prévu à la base. Un tableau répertorie le nombre de jours réel effectué pour chaque sprint ou actions.

Après obtention des jeux de données et de s'être imprégné correctement du sujet, toutes les informations étaient à disposition pour débuter la construction du projet. Une première analyse et un entraînement de modèle IA sommaires ont été réalisés. Une ébauche d'interface a été développée après la construction des maquettes. Le prototype fonctionnant correctement, les étapes suivantes ont consisté à approfondir l'analyse pour récolter davantage d'informations. Des algorithmes plus performants ont été validés pour répondre aux attentes du projet. Les bases de données ont été construites à l'occasion et l'application finale a été développée. Une marge a permis d'apporter des ajouts de fonctionnalités et certaines modifications.

**Tableau 3.** Tableau de planification simplifié du projet répertoriant le nombre de jours réalisé sur chaque étape de construction. Le code couleur fait référence aux parties du présent rapport.

Mois	Nombre de jours	Temps / Sprint	Avance / Retard	Sprints	Actions
Avril	7				Veille Technique
Mai	3	12	+ 3	Sprint 1	Recherche des jeux de données
Juin	2				Construction du Dataset et analyse rapide
Juillet	5	15	0	Sprint 2	Conception d'un modèle IA basique
Aout	10				Construction de la maquette
Septembre	15	15	0	Sprint 3	Analyses des données
Octobre	20	20	- 5	Sprint 4	Modélisations et Observations
Novembre	15	15	0	Sprint 5	Construction des BDD relationnelles
Décembre	10	20	- 5	Sprint 6	Développement de l'application
Janvier	10				
Février	5	8	+ 7	Sprint 7	Besoin et ajout de nouvelles fonctionnalités
Mars	3				
<b>Total :</b>		<b>105</b>			

### 3. Difficultés rencontrées et Résolution

Bien que le projet se base sur un modèle de classification binaire, la volonté de travailler sur un jeu de données « réel » a permis de soulever plusieurs points bloquants qui peuvent être rencontrés sur des projets professionnels. Le Dataset étant composé de nombreuses variables, l'analyse fut laborieuse et fut finalisée en le segmentant en sous-partie. Le traitement des données, se trouvant en grande quantité, incita à utiliser des échantillons afin de créer les scripts de *preprocessing* qui seraient ensuite utilisés sur l'ensemble des données.

Pour répondre aux compétences liées à la gestion des BDD, seulement une portion des données traitées durant l'analyse a été exportée. Encore une fois, il fut nécessaire d'échantillonner le jeu de départ, de créer des scripts propres et de les exporter vers une seconde base spécifique pour simuler une récupération de la donnée en respectant les compétences associées.

L'intégration du système de monitoring fut laborieuse avec un premier choix qui s'était porté sur *Airbrake*. Du fait d'une incompatibilité de la machine (malgré la mise en place d'un environnement propre au projet), un système tiers de *Flask* a permis de proposer une solution convenable pour les besoins du projet.

# CONCLUSION & PERSPECTIVES

## 1. Discussion et axes d'amélioration

Les phénomènes d'Attrition sont l'un des cas courants de la Data-Science qui porte sur l'optimisation de la réussite client pour des entreprises tertiaires. Dans ce projet basé sur la détection des individus à haut-potentiel d'attrition, le modèle IA est un classificateur binaire entraîné sur des classes aux proportions déséquilibrées, des individus abonnés et désabonnés. Dépendant des métriques souhaitant vérifier, plusieurs stratégies ont pu être appliquées afin de répondre au besoin de l'entreprise :

- Un modèle avec une classification performante émettant de fortes probabilités de prédiction par la vérification des métriques *Log Loss* et *AUC*.
- Un modèle détectant au maximum les utilisateurs se désabonnant au détriment de l'augmentation de Faux-Positifs par l'intermédiaire d'un rééquilibrage des classes. Le modèle est alors évalué principalement avec des métriques telles que *Recall* et *Precision*.

Plusieurs modèles de *Machine Learning* et de *Deep Learning* ont été entraînés afin de trouver le meilleur algorithme possible. XGBoost a été choisi par cette méthode et présente un *Log Loss* de 0.07 et un score *MCC* et *F1* de près de 0.81. Ce modèle présente donc des scores satisfaisants en comparaison de ceux créés durant la compétition *Kaggle* (Scores *Log Loss* entre 0.1 et 0.07 sur les données de soumission).

Toutefois, le modèle reste encore perfectible :

- Le modèle n'a été entraîné que sur le dernier mois des données d'entraînement. Le Dataset fournis durant la compétition *Kaggle* comportait plusieurs mois/années de données (420 Millions d'instances). La tâche aurait consisté à utiliser une solution globale de stockage avec le Framework open-source de calcul distribué *Apache Spark* par exemple. Cet outil est prévu pour l'analyse à grande échelle de la donnée, afin d'avoir une vision plus large sur le comportement des utilisateurs de la plateforme.
- De ce fait, des tendances pourraient en ressortir, amenant à la création de nouvelles variables pertinentes. Une analyse sur plusieurs mois permettrait de garder un historique des informations clients et ainsi d'anticiper leur éventuel désabonnement sur le temps.
- Ainsi, il serait intéressant d'entrainer le modèle sur les données du mois précédent (Février) de chaque client avec la variable cible de Mars et comparer les résultats.
- En outre, l'IA réalisée dans le cadre de ce projet pourrait être utilisée sur des données futures pour s'assurer de la bonne cohésion des prédictions fournies par notre modèle XGBoost Classifier et la réalité.

- De plus, une optimisation des hyperparamètres plus poussée pourrait être concevable. Seulement 3 paramètres ont été testés durant la modélisation des algorithmes de ML. Aussi, d'autres réseaux de neurones devraient être construits. Ce processus permettrait un affinage des performances du/des modèles sélectionnés.
- Enfin, il serait intéressant d'effectuer une analyse de l'écosystème et l'apport des bénéfices. En créant une nouvelle métrique, le modèle pourrait différer, mais encore une fois, cela serait dépendant des attentes de l'entreprise.

Au niveau du développement de l'application, l'interface répond aux attentes du projet, à savoir un affichage de données par requêtes API qui filtrent les données depuis une base de données relationnelle. Les résultats sont affichés rapidement sous la forme d'un tableau récapitulatif et possiblement téléchargeable dans son entier. L'interface peut aussi permettre la prédiction sur des données hors de la BDD en important un fichier CSV. L'application est accessible en se connectant via un système de *login* simple. Un système de monitoring, *Flask Monitoring Dashboard*, est intégré afin de maintenir l'application en vérifiant les éventuelles erreurs que pourrait rencontrer un utilisateur.

Bien sûr, plusieurs améliorations sont aussi envisageables sur cette partie du projet :

- Développée avec un script Python simple, plusieurs problèmes peuvent survenir au niveau de la sécurité de l'application. Un système robuste de droit d'accès devrait être implémenté en conséquence afin que certains modules proposés par l'application soient disponibles pour des utilisateurs spécifiques et pas à d'autres.
- Il serait intéressant de regrouper les méthodes de stockage de la donnée. De nombreuses bases de données de différentes natures sont reliées à l'application, pouvant compliquer les processus de maintenance. C'est notamment le cas pour la base de données créée par le module de monitoring de *Flask*.
- Le système de monitoring pourrait provenir d'un service tiers comme *Airbrake* par exemple, qui apporte de nombreuses solutions de maintenances avancées d'application.
- Le développement de l'interface a été coûteux en temps du fait des outils graphiques utilisés comme *HighCharts*. D'autres outils de développement IHM existent comme *Streamlit* qui semble être une bonne solution et donnant la possibilité d'intégrer facilement un ensemble de graphiques construit sous différentes librairies Python comme *Plotly* par exemple.
- Enfin, l'application pourrait être introduite dans des services tiers type Cloud (*Azure*, *AWS*, *Heroku*...) par exemple qui la rendrait disponible. Une solution de containerisation est aussi envisageable en utilisant *Docker*.

## 2. Bilan

Ce projet a finalement permis de couvrir un large panel de compétences et répondre à la problématique initiale.

La veille technologique a été utile dans le choix de la stratégie de modélisation IA qui présentait plusieurs solutions pertinentes en fonction des besoins. L'analyse fut l'occasion d'effectuer certaines études statistiques et de la data-visualisation, utiles pour la compréhension du jeu de données. Il fut aussi l'opportunité de relier toutes ces compétences en un produit cohérent en appliquant des outils complets de *Back-end* et *Front-end*.

Globalement, j'ai beaucoup apprécié l'exercice, notamment sur les parties d'analyse de données et de modélisation, en mettant au point un protocole structuré et pouvoir expérimenter de nouveaux outils. En approfondissant le sujet, j'ai remarqué les nombreuses possibilités d'obtenir le meilleur modèle possible pour ce jeu de données, notamment en effectuant une veille sur des publications de recherche. Cependant, comme mentionné précédemment, l'analyse et l'entraînement des modèles IA furent chronophages.

Quelques difficultés ont été rencontrées sur la partie construction des bases de données afin que celles-ci puissent coïncider avec la validation des compétences. Un travail a dû être effectué plus tard, accumulant un retard sur les prévisions faites lors de la planification. Sur la partie Application, j'ai préféré réaliser l'incorporation de différents éléments *from scratch*, développés en JavaScript ou en HTML/CSS afin de garder un certain contrôle sur le comportement du programme. Cela m'a permis de maintenir une charte graphique durant tout le développement du projet. Bien qu'ici aussi, l'utilisation de *Flask* pour intégrer de nombreux graphiques *HighCharts* alourdissait le code de l'application et n'était peut-être pas la meilleure solution. Enfin, il est clair que ce projet peut encore être amélioré, en utilisant des outils beaucoup plus adaptés mais dont j'avais pris connaissance que bien plus tard.

Finalement, cette expérience me conforte dans mon choix de reconversion professionnelle et je suis sûr que les connaissances que j'ai acquises me permettront d'être opérationnel sur des projets de divers secteurs.

# BIBLIOGRAPHIE

---

**A.K.Ahmad, A.Jafar & K.Aljoumaa.** 2019. *Customer churn prediction in telecom using machine learning in big data platform.*

**A.Votava.** 2021. *Churn prediction model.*

**B.Krawczyk.** 2016. *Learning from imbalanced data: open challenges and future directions.*

**C.Cao et al.** 2020. *The MCC-F1 curve: a performance evaluation technique for binary classification.*

**C.Schmidt.** 2020. *Approaching Unbalanced Datasets Using Data Augmentation.*

**D.Ruta, D.Nauck & B.Azvine.** 2006. *K Nearest Sequence Method and Its Application to Churn Prediction.*

**Définitions marketing.** 2019. *Win Back.*

**E.Ascarza, R.Iyengar & M.Schleicher.** 2016. *The perils of proactive churn prevention using plan recommendations: evidence from a field experiment.*

**Eudonet.** 2018. *Qu'est ce qu'un CRM ?*

**Flask.** 2010. *Testing Flask Applications.*

**Fandroid.** 2021. *Quel est le meilleur service de streaming de musique ?*

**I.Kuznetsov.** 2019. *Metrics for Imbalanced Classification.*

**Imbalanced-learn.** *Ensemble of samplers.*

**Invespcro.** 2020. *Customer Acquisition Vs.Retention Costs – Statistics And Trends.*

**J. Zhang & I. Mani.** 2003. *KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction.*

**J.Brownlee.** 2020. *8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset.*

**J.Brownlee.** 2021. *Random Oversampling and Undersampling for Imbalanced Classification.*

**J.Brownlee.** 2021. *Tour of Evaluation Metrics for Imbalanced Classification.*

**J.Brownlee.** 2021. *Undersampling Algorithms for Imbalanced Classification.*

**J.Czakon.** 2022. *24 Evaluation Metrics for Binary Classification (And When to Use Them).*

**J.Than.** 2020. *How to deal with imbalanced data in Python.*

**K.Coussement & K.W.De Bock.** 2013. *Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning.*

**K.K.Mohbey.** 2020. *Employee's Attrition Prediction Using Machine Learning Approaches.*

**K.S.V.Muralidhar.** 2021. *The right way of using SMOTE with Cross-validation.*

**L.Chen.** 2020. *From imbalanced datasets to boosting algorithms.*

**M. Alam.** 2020. *Z-score for anomaly detection.*

**Mr. Mint.** 2017. *Naive Bayes Classifier pour Machine Learning.*

**Optimove.** 2020. *Attrition de la clientèle.*

**Perceptive Analytics.** 2019. *How to Predict Customer Churn Predict churning customers and take proactive action.*

**S.A.Neslin, et al.** 2006. *Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models.*

**S.A.Qureshii et al.** 2013. *Telecommunication subscribers' churn prediction model using machine learning.*

**S.Höppner et al.** 2020. *Profit driven decision trees for churn prediction.*

**T.Bex.** 2021. *11 Times Faster Hyperparameter Tuning with HalvingGridSearch.*

**UK Parliament.** 2021. *Music streaming must modernise. Is anybody listening ?*

**Z.Y.Chen, Z.P.Fan & M.Sun.** 2012. *A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data.*



## Lien GitHub du Projet

<https://github.com/JeremyLeJoncour/E1-PCO>



## ANNEXES : ANALYSES DES DONNEES

Regroupent des informations complémentaires sur le jeu de données et son analyse, la présentation de différents graphiques et interprétations des résultats.

## **transactions\_v3.csv**

- msno: utilisateur id
- payment\_method\_id: mode de paiement
- payment\_plan\_days: durée de l'abonnement en jours
- plan\_list\_price: en nouveau dollar de Taïwan (NTD)
- actual\_amount\_paid: en nouveau dollar de Taïwan (NTD)
- is\_auto\_renew
- transaction\_date: format %Y%m%d. Dernière transaction au 31/03/2017
- membership\_expire\_date: format %Y%m%d. Dates d'expiration de l'abonnement
- is\_cancel: si l'utilisateur a annulé ou non l'adhésion à cette transaction

## **user\_logs\_v2.csv**

- msno: utilisateur id
- date: format %Y%m%d. Données collectées jusqu'au 31/03/2017.
- num\_25: nombre de musiques jouées moins de 25 %
- num\_50: nombre de musiques jouées entre 25 % et 50 %
- num\_75: nombre de musiques jouées entre 50 % et 75 %
- num\_985: nombre de musiques jouées entre 75 % et 98.5 %
- num\_100: nombre de musiques jouées sur 98,5% de la durée
- num\_unq: nombre de musiques uniques jouées
- total\_secs: nombre total de secondes écouté

## **members\_v3.csv**

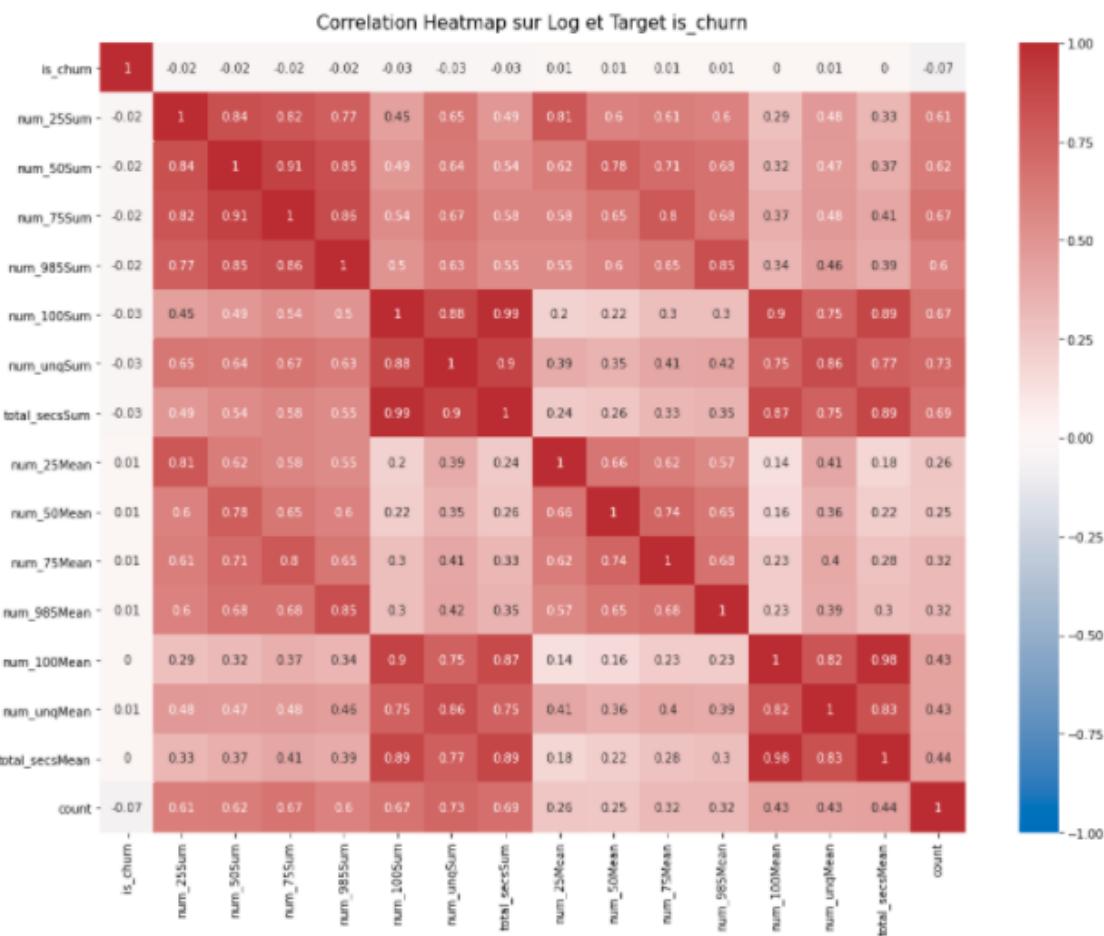
- msno: utilisateur id
- city: ville id
- bd: Age. Remarque : cette colonne contient des valeurs aberrantes allant de -7000 à 2015.
- gender : Genre de l'individu
- registered\_via: Méthode d'enregistrement
- registration\_init\_time: Date d'enregistrement
- expiration\_date: Date d'expiration d'abonnement, pris comme un instantané auquel le member.csv est extrait. Ne représentant pas le comportement de désabonnement réel.

## **train\_v2.csv**

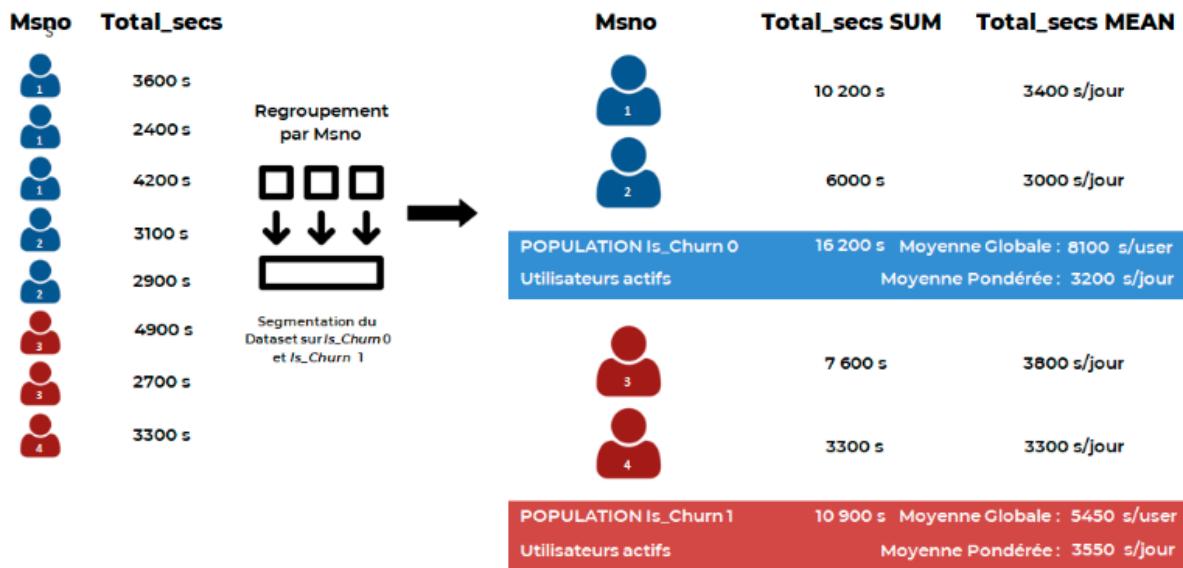
Contenant les identifiants des utilisateurs et s'ils ont quitté les services du site.

- msno: utilisateur id
- is\_churn: La variable cible.

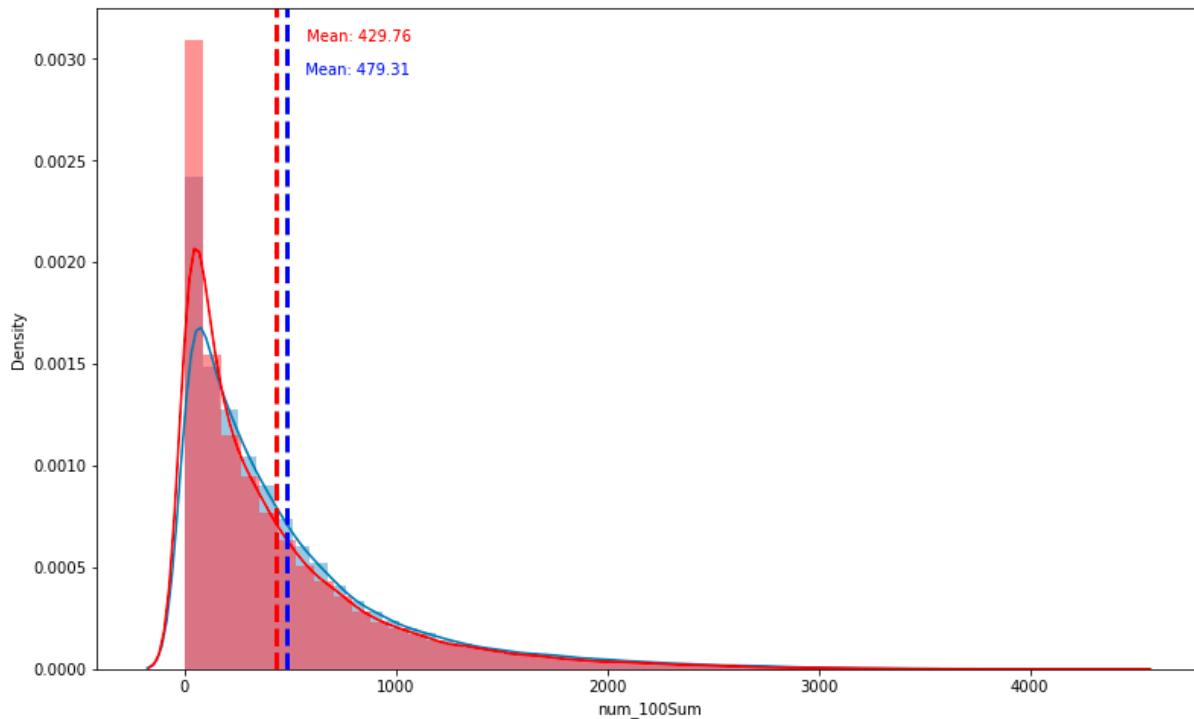
[Annexe 1. Informations complémentaires des variables de chaque fichier CSV.](#)



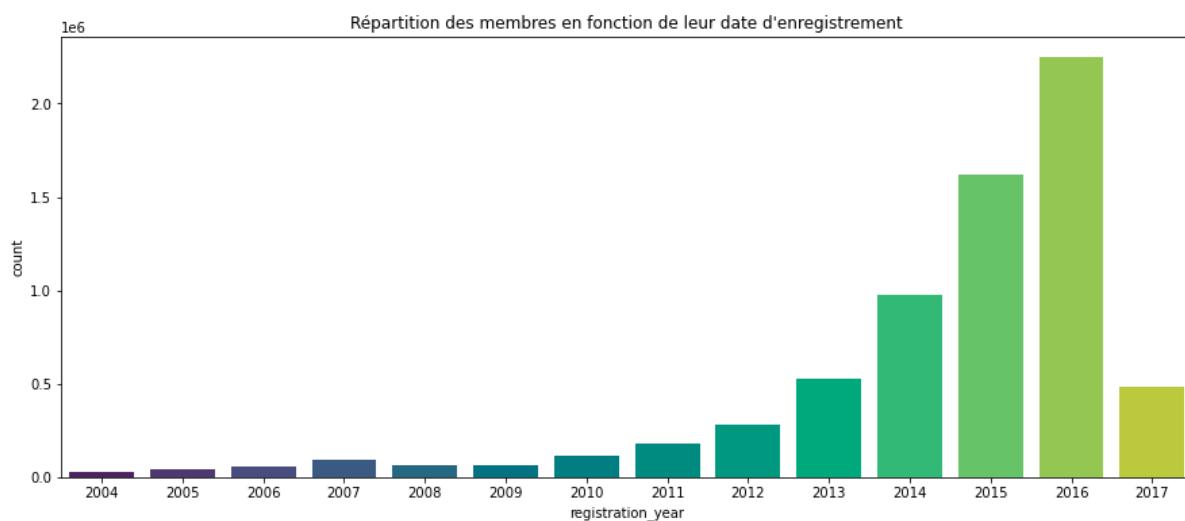
Annexe 2. Matrice de Corrélation avec Dataset Log.



Annexe 3. Démarche de l'approche données utilisateurs (exemple, données fictives).

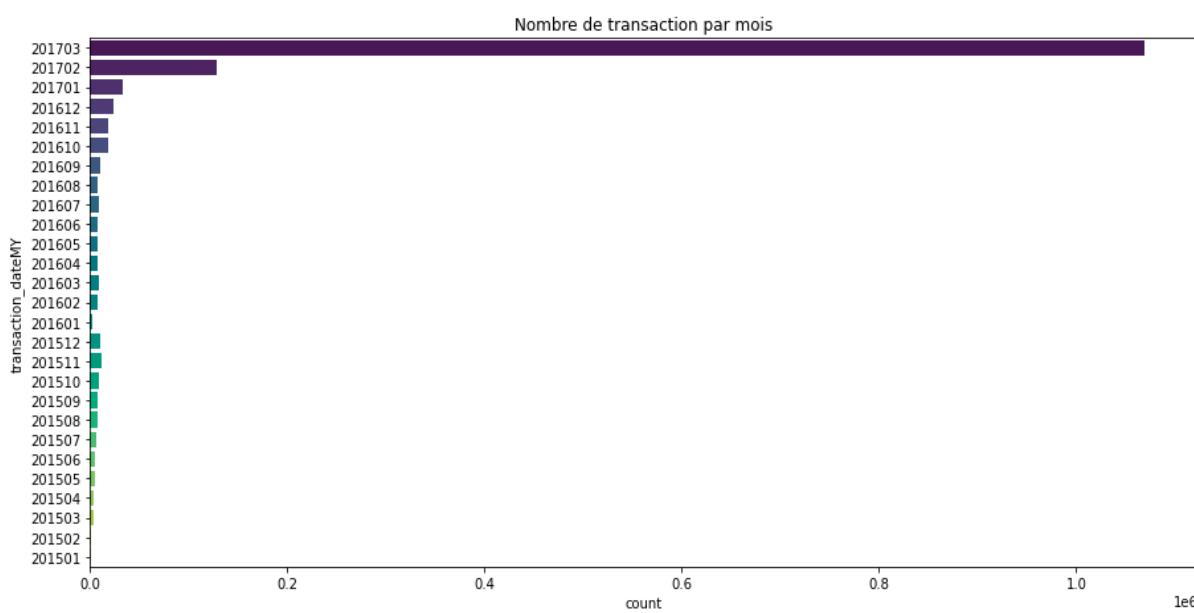


**Annexe 4.** Comparaison de la distribution et moyennes de deux échantillons de tailles égales (en rouge, l'échantillon d'utilisateur désabonnés, en bleu, l'échantillon d'abonné).

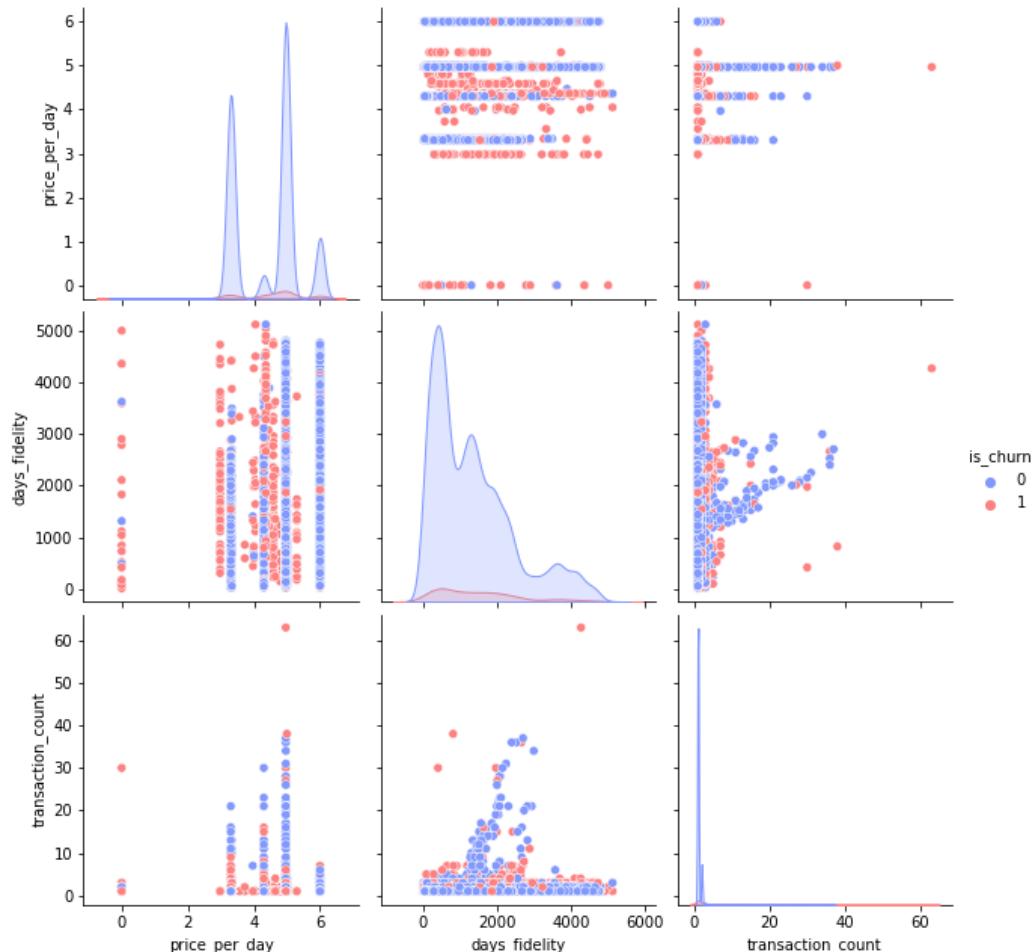


**Annexe 5.** Exemple de proportion des individus en fonction d'une variable, ici Registration\_Year, créée pour l'analyse.

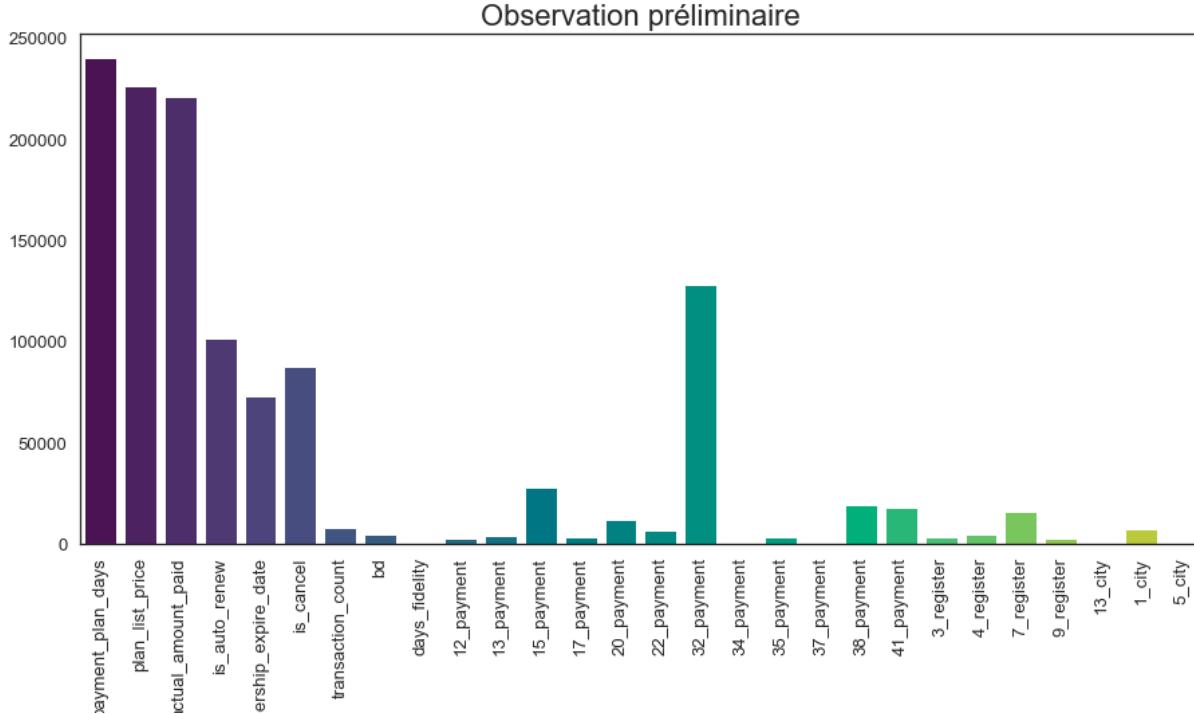
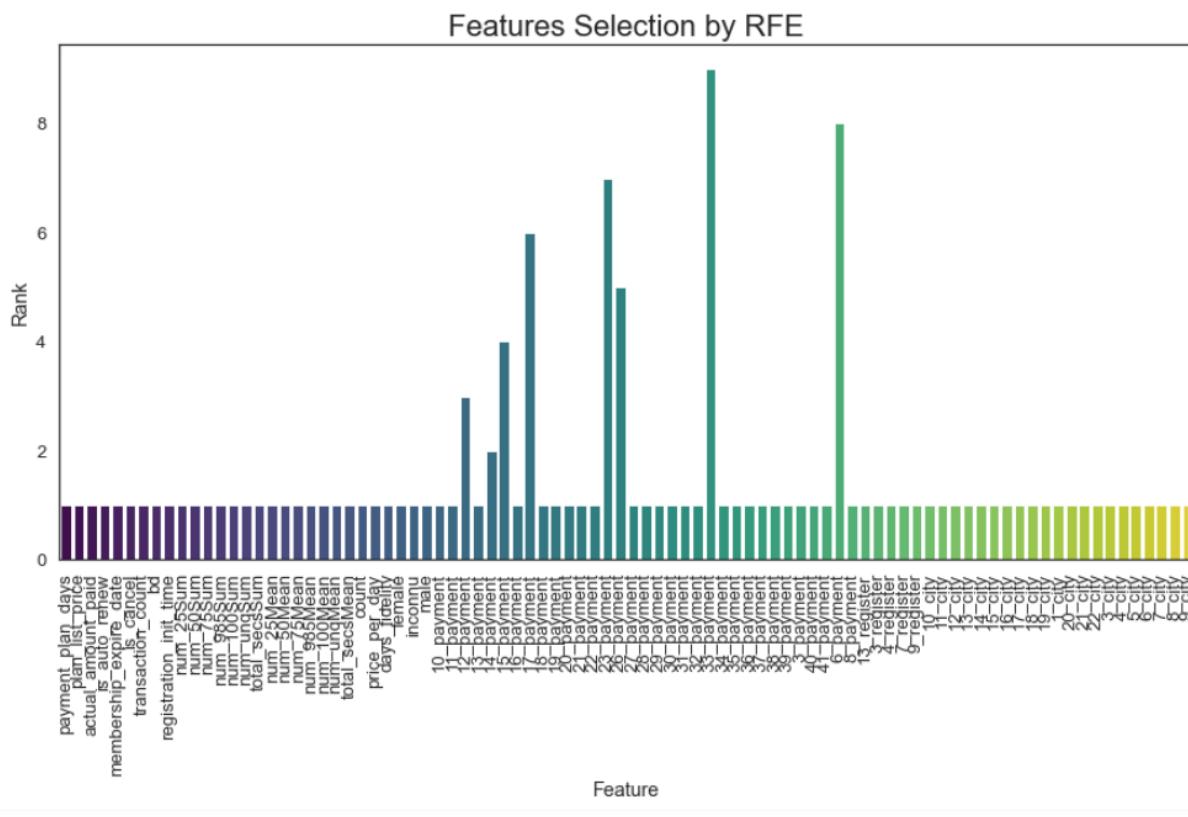
**Constat :** La majorité des transactions ont été effectuées durant le mois de Mars 2017. D'autres transactions comptabilisées ont été réalisées sur l'année précédente pour certains utilisateurs, remontant même à Janvier 2015. Ces données antérieures à l'année 2016 semblent représenter des anomalies lors de l'extraction des jeux de données qui ne devrait remonter seulement l'historique du début de l'année 2016.



**Annexe 6.** Proportion des transactions effectuées sur le mois de Mars 2017, des dates antérieures sont retrouvées, suspectant des anomalies dans le Dataset Transactions.

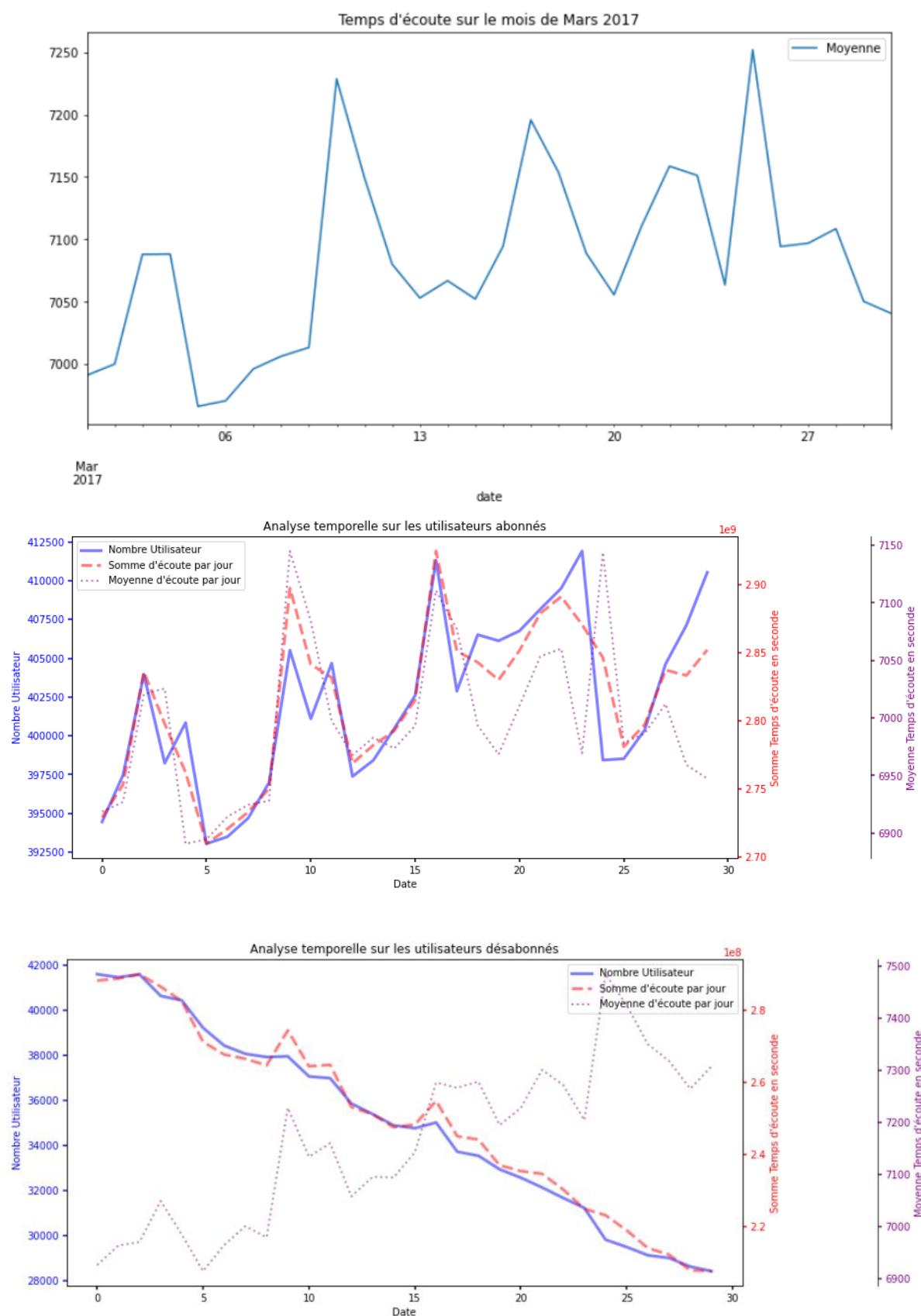


**Annexe 7.** Pairplot sur une part des variables créées lors de l'analyse.



#### Annexe 8. Analyse sur les variables par RFE et sélection univariée.

De nombreux tests statistiques différents peuvent être utilisés afin d'obtenir un aperçu des variables influant le plus sur la *Target*. L'ANOVA F-value est ici utilisée car appropriée pour les entrées numériques et les données nominales encodées.



**Annexe 9.** Analyse temporelle sur le nombre de musiques écoutées par jour, sur le mois de Mars 2017 pour les utilisateurs abonnés et désabonnés.



## ANNEXES : MODELISATIONS

Regroupent des informations complémentaires sur la stratégie de modélisation et les divers résultats obtenus lors de la validation croisée des modèles ML, les courbes d'apprentissage des réseaux de neurones construits etc...

**Analyse des Datasets KKBOX (Téléchargement des CSV Kaggle)**

**LOGS**    **TRANSACTIONS**    **MEMBERS**    **TRAIN**

Analyses sur chaque CSV, Groupby par identifiant pour avoir les données sur le mois de Mars. Suppression des valeurs manquantes, réduction des valeurs aberrantes, tests statistiques entre les populations sortantes et restantes. De la Features Engineering est réalisées. Les CSV sont ensuite Fusionnés pour avoir un Dataset unique, prêt pour la création de modèles IA.

**Création du Dataset**

- Dimension du Fichier : 725722 utilisateurs uniques, 33 Variables
- Dimension après preprocessing : 725722 utilisateurs uniques, 90 Variables

Les données du Dataset sont encodées pour les variables catégorielles et temporelles.

• Target : *is\_churn*    **Abonnés** **94 %**    **Désabonnés** **6 %**

**Segmentation du Dataset**

<b>TRAIN</b>	<b>70 %</b>	<b>VALID</b>	<b>24 %</b>	<b>TEST</b>	<b>6 %</b>
--------------	-------------	--------------	-------------	-------------	------------

Segmentation par le module *Train\_Test\_Split* de SKlearn en s'assurant de garder la même proportion de variable Cible dans chaque set.

**Stratégie Standard**

<b>Abonnés</b>	<b>94 %</b>	<b>Désabonnés</b>	<b>6 %</b>
----------------	-------------	-------------------	------------

**Stratégie Rééquilibrage des classes\***

<b>Abonnés</b>	<b>50 %</b>	<b>Désabonnés</b>	<b>50 %</b>
----------------	-------------	-------------------	-------------

**Sélection des modèles ML par Cross-Validation**

8 modèles sont proposés et comparés sur différentes métriques (*Log\_loss*, *AUC*, *MCC*, *F1*, *Recall*, *Precision*) par cross-validation avec 5 KFolds stratifiés. Les 2 meilleurs modèles seront optimisés dans l'étape d'Hyperparamétrage. Les modèles de classification sont *Logistic Regression*, *Decision Tree*, *Random Forest*, *Bagging*, *AdaBoost*, *LGBM*, *XGBoost*. 3 modèles de *Stacking* sont aussi comparés.

**Sélection des modèles ANN**

3 réseaux de neurones artificiels sont construits et varie en fonction de leur *Learning\_Rate*, et la Densité des couches de neurones. L'Optimizer est *Adamax* et la métrique évaluée est *Binary\_crossentropy*.


**Hyperparamétrages**

Optimisation sur les 3 modèles sélectionnés après entraînement par Cross-Validation : XGBoost, LightGBM, Random Forest Classifier.

**Sélection du meilleur modèle**
**Sélection des Variables**

*Feature\_importance* sur le modèle XGBoost. Les 30 meilleures variables ont été sélectionnées

**Stratégie Rééquilibrage des classes\***
**Sous-échantillonnage**

*Tomek Link*, *NearMiss*, *Random*


**Sur-échantillonnage**

*SMOTE*, *ADASYN*, *Random*


**Combinaisons Samplers**

*SMOTE*, *Random-Under*


**Ensemble Samplers**

*Balanced Random Forest*  
*RUSBoost Classifier*

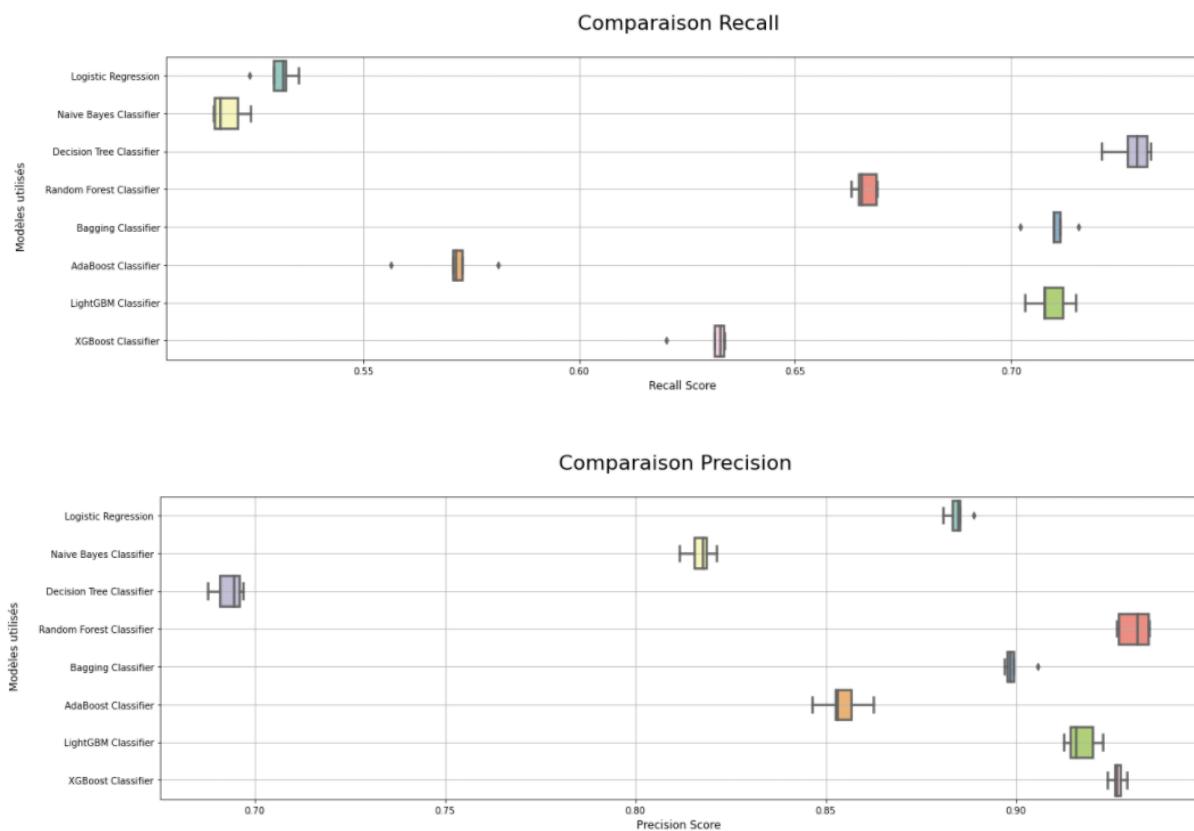
**Annexe 10. Résumé des processus de modélisation.**

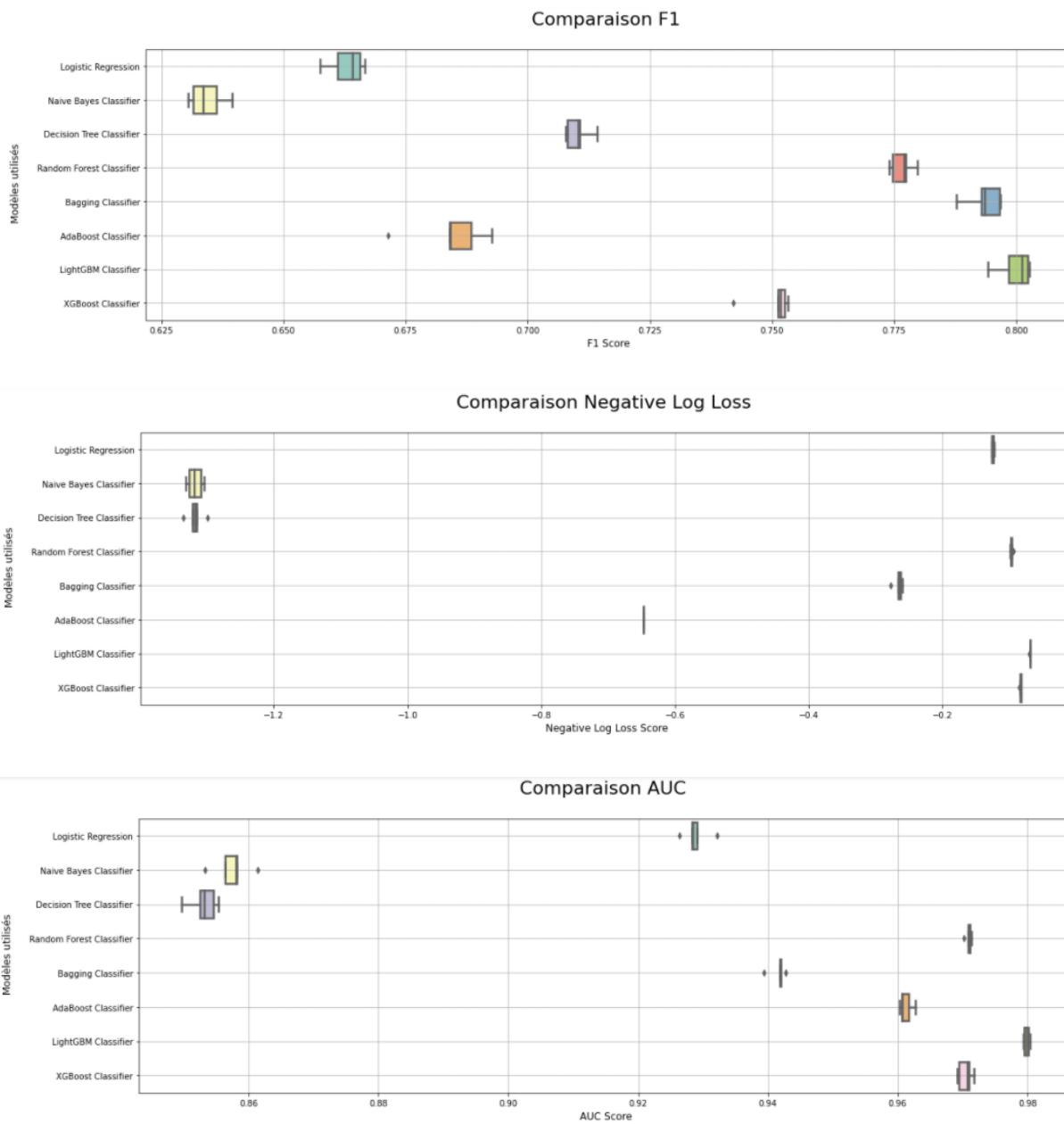
Dimensions Train : (508005, 90)  
 Dimensions Valid : (174173, 90)  
 Dimensions Test : (43544, 90)

Proportion désabonnés sur Train : 6.42 %  
 Proportion désabonnés sur Valid : 6.42 %  
 Proportion désabonnés sur Test : 6.42 %

Dimension X\_train dataset: (508005, 88)  
 Dimension y\_train dataset: (508005, )  
 Dimension X\_valid dataset: (174173, 88)  
 Dimension y\_valid dataset: (174173, )

**Annexe 11. Segmentation du Dataset Final utilisé pour la modélisation.**





Annexe 12. Performances des modèles ML sur Validation croisée avec StratifiedKFold(5) pour chaque métrique.

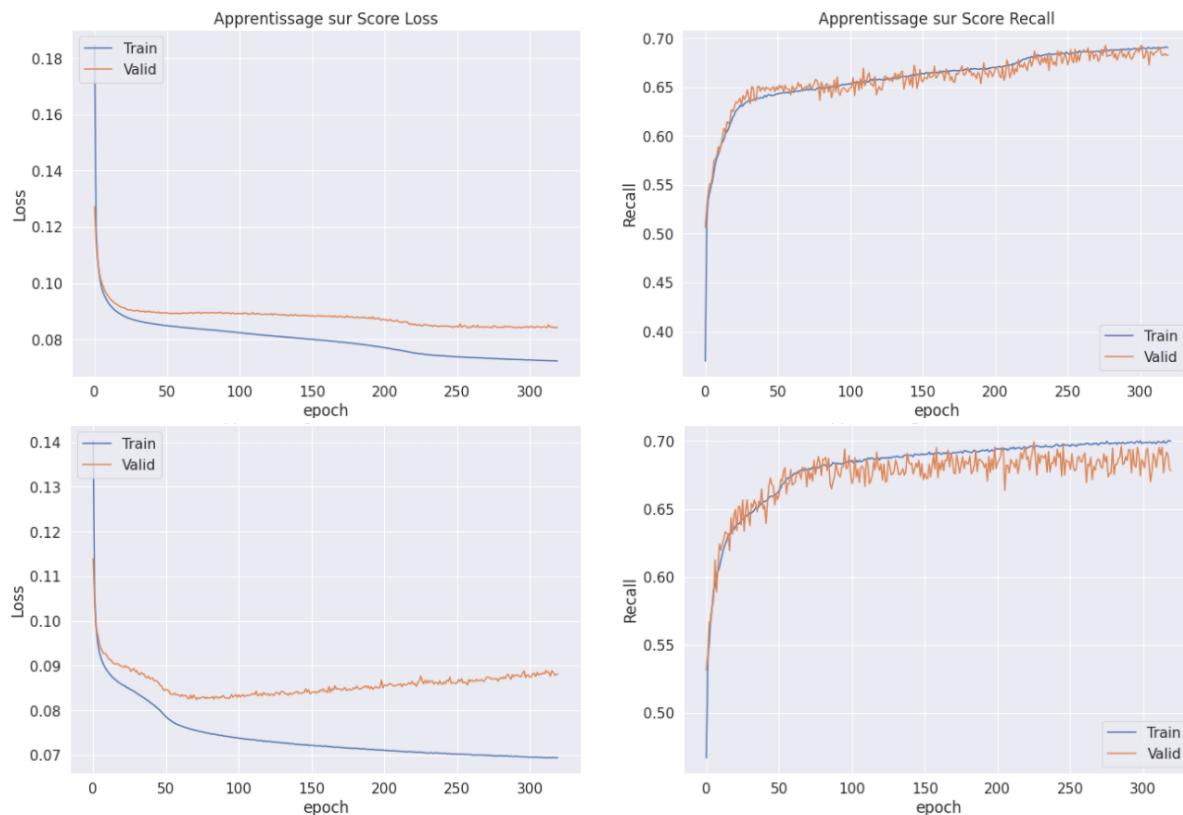
### HalvingGridSearchCV : XGBClassifier (tree\_method : gpu\_hist) 100k entrées sur 4 combinaison

```
{'learning_rate': 0.05, 'max_depth': 6, 'n_estimators': 200, 'subsample': 1}
CPU times: user 38.3 s, sys: 569 ms, total: 38.9 s
Wall time: 7min 56s
```

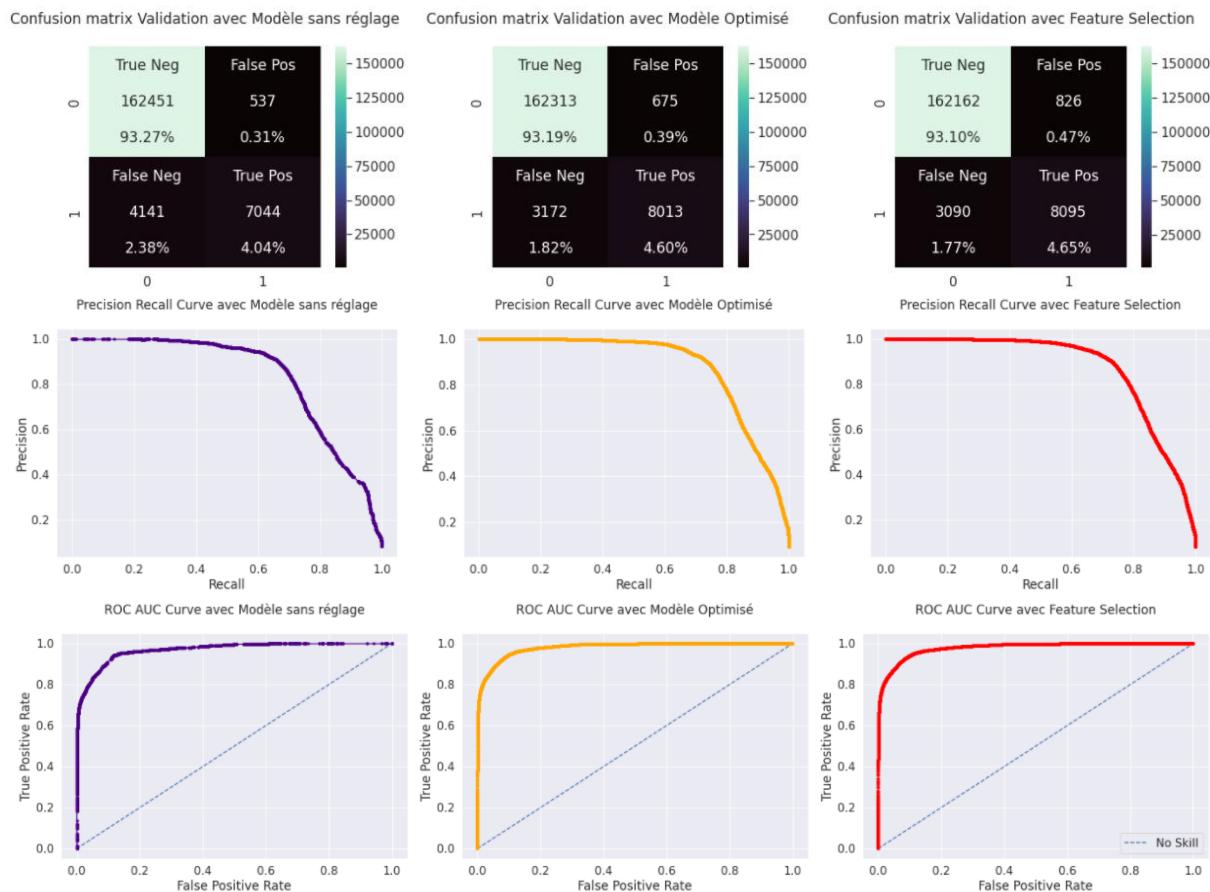
### GridSearchCV : XGBClassifier (tree\_method : gpu\_hist) 100k entrées sur 4 combinaison

```
{'learning_rate': 0.05, 'max_depth': 6, 'n_estimators': 200, 'subsample': 1}
CPU times: user 39.8 s, sys: 752 ms, total: 40.6 s
Wall time: 13min 9s
```

Annexe 13. Test de comparaison de rapidité entre deux méthodes d'hyperparamétrages.



Annexe 14. Learning curve sur Log Loss et Recall pour les modèles Multilayer Perceptron 3 (en haut) et 2 (bas).



Annexe 15. Comparaison sur le modèle XGBoost Classifier.



## ANNEXES : AUTRES AXES

Regroupent des informations complémentaires sur les autres axes d'études comme les méthodes d'équilibrage des classes et résultats obtenus.

**Annexe 16.** Tableaux des performances des modèles entraînés sur échantillonnage avec NearMiss-3 et SMOTE.

Model with NM3	Roc_AUC	Log_loss	F1	Precision	Recall	MCC
LightGBM Classifier	0,9543	0,2240	0,6354	0,5183	0,8208	0,6232
XGBoost Classifier	0,9159	0,3228	0,5530	0,4295	0,7759	0,5398
Random Forest Classifier	0,9086	0,3797	0,4297	0,2936	0,8006	0,4312
Logistic Regression	0,7816	0,5202	0,2275	0,1372	0,6650	0,2002
Bagging Classifier	0,9507	0,5929	0,6290	0,5134	0,8119	0,6160
AdaBoost Classifier	0,8632	0,6828	0,4975	0,3817	0,7143	0,4787
Naive Bayes Classifier	0,7189	1,4069	0,4971	0,9440	0,3374	0,5501
Decision Tree Classifier	0,8585	4,0442	0,4767	0,3343	0,8305	0,4800

Model with SMOTE	Roc_AUC	Log_loss	F1	Precision	Recall	MCC
LightGBM Classifier	0,9741	0,0887	0,7896	0,8538	0,7344	0,7788
Random Forest Classifier	0,9733	0,1009	0,7681	0,7980	0,7405	0,7535
XGBoost Classifier	0,9665	0,1747	0,5703	0,4270	0,8584	0,5697
Bagging Classifier	0,9442	0,2810	0,7928	0,8511	0,7420	0,7817
Logistic Regression	0,9456	0,3363	0,5025	0,3527	0,8735	0,5118
AdaBoost Classifier	0,9508	0,6666	0,5862	0,4737	0,7685	0,5697
Naive Bayes Classifier	0,8591	1,3262	0,6339	0,7993	0,5252	0,6294
Decision Tree Classifier	0,8591	1,3861	0,7041	0,6687	0,7434	0,6837

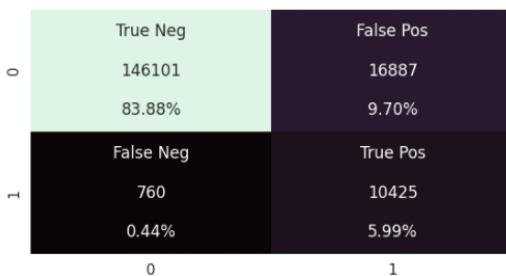
Over : 0.3 Under : 0.7	MCC : 0.7716135130972492	Loss : 0.09144136417549933
Over : 0.3 Under : 0.6	MCC : 0.7811282747366386	Loss : 0.0859181253826907
Over : 0.3 Under : 0.5	MCC : 0.7866776923072185	Loss : 0.0812373946585295
Over : 0.4 Under : 0.7	MCC : 0.7786272769005165	Loss : 0.08551436855990276
Over : 0.4 Under : 0.6	MCC : 0.7843008510489557	Loss : 0.08133373715173944
Over : 0.4 Under : 0.5	MCC : 0.7867265979041622	Loss : 0.07787631981379999
Over : 0.5 Under : 0.7	MCC : 0.780489971984893	Loss : 0.08213662360743794
Over : 0.5 Under : 0.6	MCC : 0.7842613705101497	Loss : 0.07888295907839417
Over : 0.5 Under : 0.5	MCC : 0.784216908028309	Loss : 0.07669685063480552

**Annexe 17.** Vérification des performances d'un modèle LightGBM Classifier par stratégie hybride under/oversampling

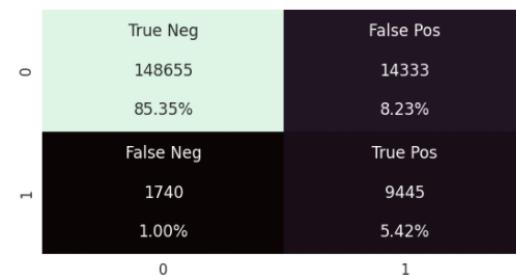
BalancedRandomForestClassifier:  
 LogLoss : 0.2185141678965018      AUC : 0.9739457020890347      F1 : 0.541600644206042  
 Recall : 0.9320518551631649      Precision : 0.38170035149384884      MCC : 0.558506780601123

RUSBoostClassifier:  
 LogLoss : 0.687030437135581      AUC : 0.9600577885349172      F1 : 0.5402854446128765  
 Recall : 0.8444345105051408      Precision : 0.39721591386996385      MCC : 0.5401293913905348

Confusion matrix Validation avec BalancedRandomForestClassifier



Confusion matrix Validation avec RUSBoostClassifier

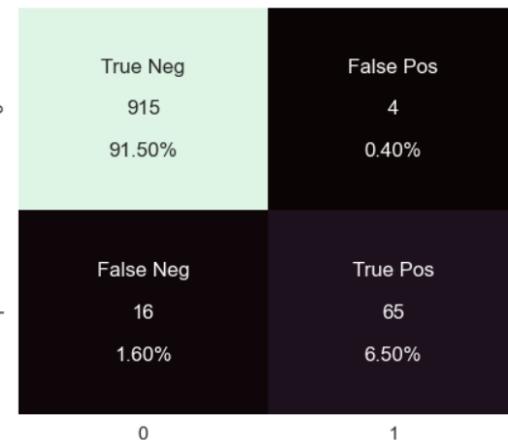


[Annexe 18. Performances sur modèles d'ensemble Sampling de la bibliothèque Imblearn.](#)

Confusion matrix Test avec Modèle enregistré



Confusion matrix Test avec Modèle enregistré



[Annexe 19. Résultat sous forme de matrices de confusion comparées pour un même modèle XGBoost Classifier avec des seuils de prédictions différents. A gauche, un seuil fixé automatiquement à 0.5, a droite, un seuil obtenu par l'étude du MCC/F1 curve et fixé à 0.33. Augmentation des Vrais-Positifs.](#)



## ANNEXES : BASE DE DONNEES

Regroupent des informations complémentaires sur les scripts d'insertion des données en BDD, création de la BDD originale, MCD...

**Annexe 20.** Requêtes de création des tables pour la base de données relationnelle DBPCO constituée de 8 tables, 4 reprenant les variables des fichiers CSV originaux et 4 autres ajoutés mentionnant le nom des villes, des méthodes de paiements ou d'enregistrement. La dernière table Time\_save, fait mention de la date de récolte de la donnée pour 1 mois, ici Mars 2017.

```
CREATE TABLE
    User (
        User_ID INT,
        msno VARCHAR(55),
        PRIMARY KEY (User_ID)
    )

CREATE TABLE
    Time_save (
        Save_ID INT AUTO_INCREMENT,
        mois VARCHAR(55),
        year INT,
        PRIMARY KEY (Save_ID)
    )

CREATE TABLE
    City (
        City_ID INT,
        City_name VARCHAR(55),
        PRIMARY KEY (City_ID)
    )

CREATE TABLE
    Payment_method (
        Payment_method_ID INT,
        Payment_method_name VARCHAR(55),
        PRIMARY KEY (Payment_method_ID)
    )

CREATE TABLE
    Register (
        Register_ID INT,
        Register_name VARCHAR(55),
        PRIMARY KEY (Register_ID)
    )

CREATE TABLE
    Transactions (
        Transactions_ID INT AUTO_INCREMENT,
        payment_method_id INT,
        payment_plan_days INT,
        plan_list_price INT,
        actual_amount_paid INT,
        is_auto_renew INT,
        transaction_date DATE,
        membership_expire_date DATE,
        is_cancel INT,
        User_ID INT,
        Save_ID INT,
        PRIMARY KEY (Transactions_ID),
```

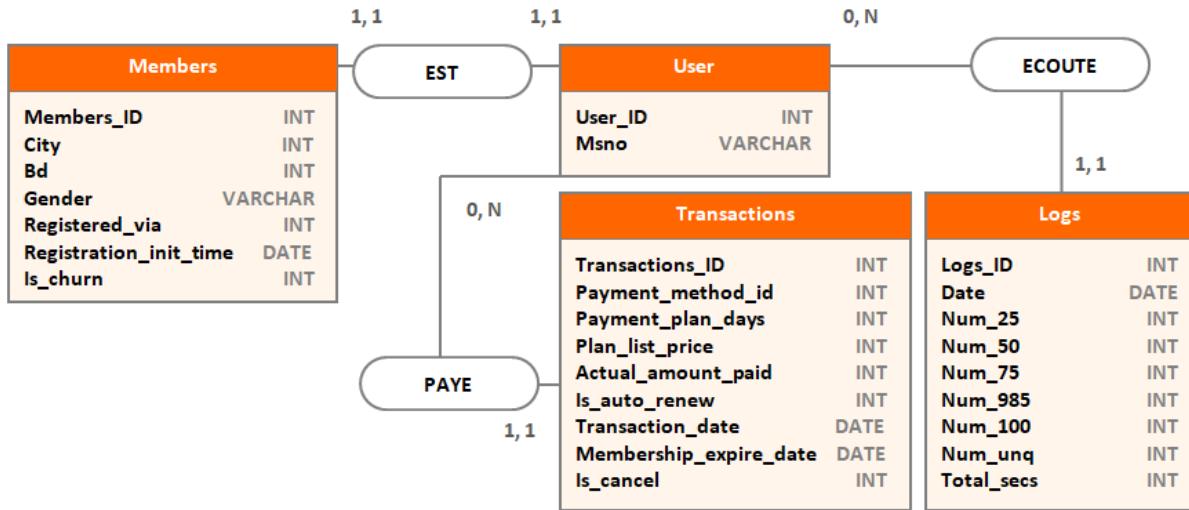
```
FOREIGN KEY fk_user(User_ID)
REFERENCES User(User_ID),  
  
FOREIGN KEY fk_Payment_method(payment_method_id)
REFERENCES Payment_method(Payment_method_ID),  
  
FOREIGN KEY fk_Time_save(Save_ID)
REFERENCES Time_save(Save_ID)
)
```

## CREATE TABLE

```
Members (
    Members_ID INT AUTO_INCREMENT,
    city INT,
    bd INT,
    gender VARCHAR(55),
    registered_via INT,
    registration_init_time DATE,
    is_churn INT,
    User_ID INT,
    PRIMARY KEY (Members_ID),  
  
FOREIGN KEY fk_city(city)
REFERENCES City(City_ID),  
  
FOREIGN KEY fk_Register(registered_via)
REFERENCES Register(Register_ID),  
  
FOREIGN KEY fk_user(User_ID)
REFERENCES User(User_ID)
)
```

## CREATE TABLE

```
Logs (
    Logs_ID INT AUTO_INCREMENT,
    date DATE,
    num_25 INT,
    num_50 INT,
    num_75 INT,
    num_985 INT,
    num_100 INT,
    num_unq INT,
    total_secs INT,
    User_ID INT,
    Save_ID INT,
    PRIMARY KEY (Logs_ID),  
  
FOREIGN KEY fk_User(User_ID)
REFERENCES User(User_ID),  
  
FOREIGN KEY fk_Time_save(Save_ID)
REFERENCES Time_save(Save_ID)
)
```



**Annexe 21.** MCD simplifié sur la base de données relationnelle originale. Un utilisateur peut écouter plusieurs musiques sur le mois de Mars et faire plusieurs transactions.

**Annexe 22.** Scripts d'insertion des données pour la base de données relationnelle DATABASEKKBOX à partir du Dataset Test (6% des clients).

```

# Encodage msno:
le = LabelEncoder()
le.fit(Test['msno'].tolist())
Test['ID_user'] = le.transform(Test['msno'].tolist())
Test['ID_user'] = [i+1 for i in Test['ID_user']]

# Table Users:
for i in range(len(Test)):
    user_id = Test['ID_user'][i]
    msno = Test['msno'][i]

    table = """INSERT INTO User (User_ID, msno) VALUES (%s, %s)"""
    values = (int(user_id), str(msno))

    cursor.execute(table, values)
    link.commit()

# Table Logs:
for i in range(len(Test)):
    num_25Sum = Test['num_25Sum'][i]
    num_50Sum = Test['num_50Sum'][i]
    num_75Sum = Test['num_75Sum'][i]
    num_985Sum = Test['num_985Sum'][i]
    num_100Sum = Test['num_100Sum'][i]
    num_unqSum = Test['num_unqSum'][i]
    total_secsSum = Test['total_secsSum'][i]
    num_25Mean = Test['num_25Mean'][i]
    num_50Mean = Test['num_50Mean'][i]
    num_75Mean = Test['num_75Mean'][i]
    num_985Mean = Test['num_985Mean'][i]
    num_100Mean = Test['num_100Mean'][i]
    num_unqMean = Test['num_unqMean'][i]
    total_secsMean = Test['total_secsMean'][i]
    count = Test['count'][i]
    ID_user = Test['ID_user'][i]

    table = """INSERT INTO Logs (Logs_ID, num_25Sum, num_50Sum, num_75Sum, num_985Sum, num_100Sum, num_unqSum, total_secsSum, num_25Mean, num_50Mean, num_75Mean, num_985Mean, num_100Mean, num_unqMean, total_secsMean, count, User_ID) VALUES (NULL, %s, %s)"""
    values = (float(num_25Sum), float(num_50Sum), float(num_75Sum), float(num_985Sum), float(num_100Sum), float(num_unqSum),
              float(total_secsSum), float(num_25Mean), float(num_50Mean), float(num_75Mean), float(num_985Mean),
              float(num_100Mean), float(num_unqMean), float(total_secsMean), int(count), int(ID_user))

    cursor.execute(table, values)
    link.commit()
    link.close()

```

```

# Table Transactions:
for i in range(len(Test)):
    payment_method_id = Test['payment_method_id'][i]
    payment_plan_days = Test['payment_plan_days'][i]
    plan_list_price = Test['plan_list_price'][i]
    actual_amount_paid = Test['actual_amount_paid'][i]
    is_auto_renew = Test['is_auto_renew'][i]
    transaction_date = Test['transaction_date'][i]
    membership_expire_date = Test['membership_expire_date'][i]
    is_cancel = Test['is_cancel'][i]
    transaction_count = Test['transaction_count'][i]
    price_per_day = Test['price_per_day'][i]
    ID_user = Test['ID_user'][i]

    table = """INSERT INTO transactions (Transactions_ID, payment_method_id, payment_plan_days, plan_list_price,
    actual_amount_paid, is_auto_renew, transaction_date, membership_expire_date, month_expire, year_expire, is_cancel,
    transaction_count, price_per_day, User_ID) VALUES (NULL, %s, %s)"""
    values = (int(payment_method_id), int(payment_plan_days), int(plan_list_price), int(actual_amount_paid),
              int(is_auto_renew), transaction_date, membership_expire_date, int(month_expire), int(year_expire),
              int(is_cancel), int(transaction_count), float(price_per_day), int(ID_user))

    cursor.execute(table, values)
link.commit()

# Table Members:
for i in range(len(Test)):
    city = Test['city'][i]
    bd = Test['bd'][i]
    gender = Test['gender'][i]
    registered_via = Test['registered_via'][i]
    registration_init_time = Test['registration_init_time'][i]
    days_fidelity = Test['days_fidelity'][i]
    is_churn = Test['is_churn'][i]
    ID_user = Test['ID_user'][i]

    table = """INSERT INTO Members (Members_ID, city, bd, gender, registered_via, registration_init_time, days_fidelity,
    is_churn, User_ID) VALUES (NULL, %s, %s, %s, %s, %s, %s, %s, %s)"""
    values = (int(city), str(bd), str(gender), int(registered_via), str(registration_init_time), int(days_fidelity),
              int(is_churn), int(User_ID))

    cursor.execute(table, values)
link.commit()

# Insertion des scores en base de données:
config = {
    'user': 'root',
    'password': 'root',
    'host': '127.0.0.1',
    'port': '3307',
    'database': 'log_model',
    'raise_on_warnings': True,
}

# Création du Cursor
link = mysql.connector.connect(**config)
cursor = link.cursor(buffered=True)

# Requêtes:
query = """
    INSERT INTO log_model
        ('Date', `Log_loss`, `Score AUC`, `Score F1`, `Score MCC`, `Recall`, `Precision`, `Instances`, `ID`)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, NULL)
"""

values = ((Date), float(loss), float(auc), float(scoreF), float(scoremcc), float(recall), float(precision), int(Total))
cursor.execute(query, values)

link.commit()

# Requêtes:
cursor.execute("""SELECT * FROM log_model""")
rows = cursor.fetchall()

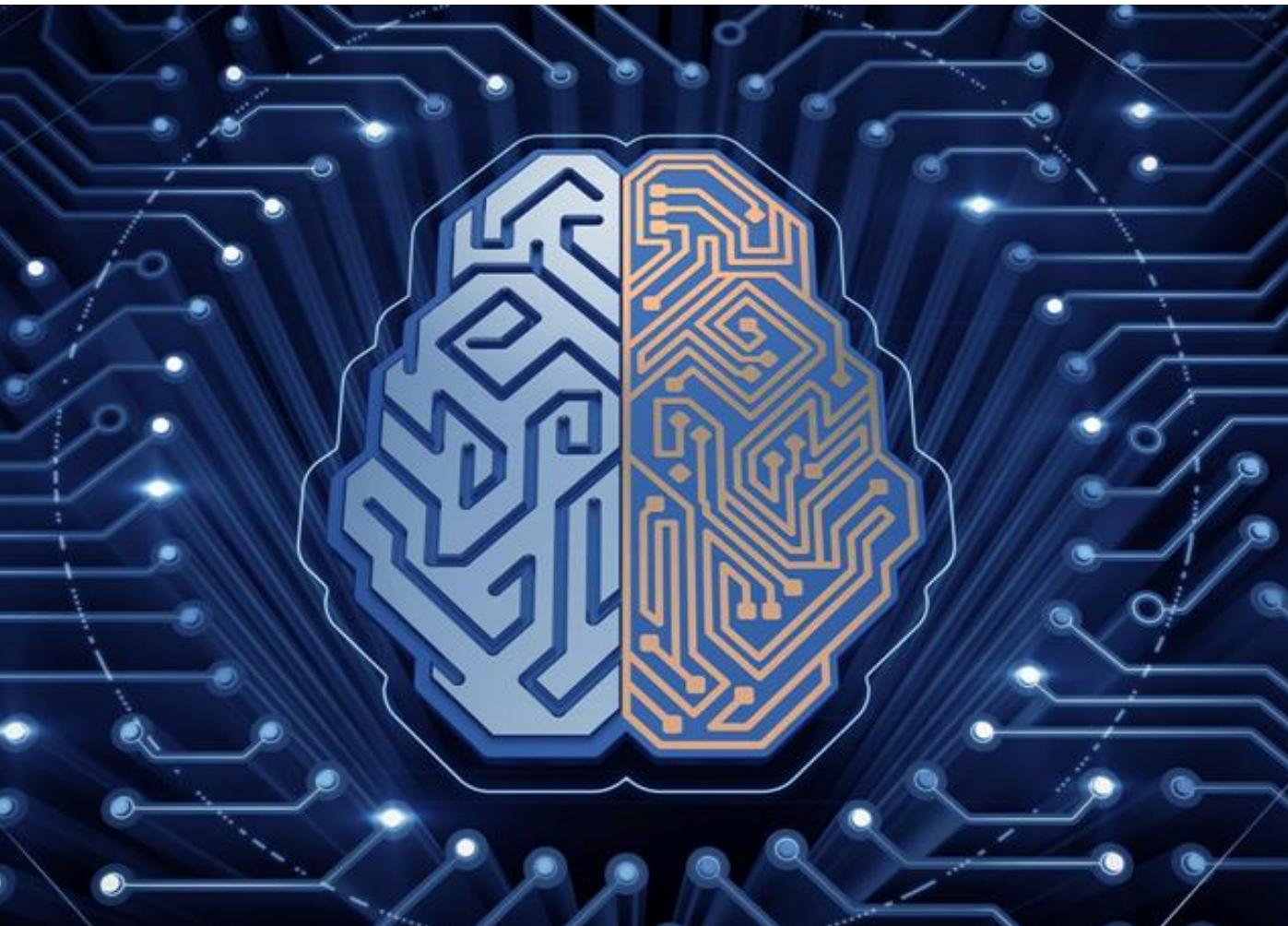
log_model = []
for values in rows :
    log_model.append(list(values))

df_model = pd.DataFrame(log_model, columns=['Date','Log_loss', 'Score AUC', 'Score F', 'Score MCC', 'Recall', 'Precision',
                                             'Instances', 'ID'])

link.close()

```

**Annexe 23.** Script d'insertion des données pour la base de données relationnelle LOG\_MODEL et exportation en Dataframe pour la construction du graphique associé dans l'application.



## ANNEXES : APPLICATION

Regroupent des captures d'écran des différentes pages de l'application, du tableau de bord de *monitoring* ainsi que quelques captures du prototype développé...



**Identifiant**

**Changer de Mot de passe**

**Valider le mot de passe**

**Nouveau mot de passe saisi invalide !**

### Changement de mot de passe

Le Mot de Passe doit contenir au minimum 8 caractères en prenant en compte les conditions suivantes :

- Au moins une Majuscule
- Au moins un caractère numérique (0-9)
- Peut contenir les caractères spéciaux suivant : @#\$%^&+=



**Entrez un Identifiant**

**Entrez un Mot de passe**

**Confirmez votre Mot de passe**

**Confirmer**

**Retour à la page de connexion**

**Compte créé !**

### Conditions de création de compte

L'Identifiant et le Mot de Passe doivent contenir au minimum 8 caractères en prenant en compte les conditions suivantes :

- Au moins une Majuscule
- Au moins un caractère numérique (0-9)
- Peut contenir les caractères spéciaux suivant : @#\$%^&+=



**Churn Target**

**NAVIGATION**

- Accueil
- Informations

**ANALYSES**

- Simulations
- Dashboard

**Prédiction sur les phénomènes d'Attrition chez KKbox**

Projet Chef-d'Oeuvre pour la Certification Développeur IA

**CUSTOMER LOYALTY**

L'attrition est le fait, pour un client (individu, entreprise, etc.) de quitter un fournisseur de biens ou de services, une marque, un produit ou un service. Le taux d'attrition (*«churn rate»* en anglais) se mesure comme le pourcentage de clients perdus, sur une période donnée (en général une année ou un mois) par rapport au nombre total de clients figurant dans la base clientèle au début de cette période.

Il dépend de la capacité de l'entreprise à retenir ses clients, à les fidéliser par une offre pertinente. Elle conduit souvent à l'instauration d'un score prédictif qui sert à déclencher des actions marketings et commerciales.

Généralement, ces scores sont définis par le service marketing par une analyse de données plus ou moins complexe, comparant les clients satisfaits du service et des clients déjà désabonnés afin d'en ressortir les aspects défaillants de l'offre et le profil des utilisateurs résilients. Dans cette optique, des processus de modélisation via des algorithmes de Machine Learning (apprentissages supervisés notamment) permettent une détection plus affinée des utilisateurs à fort potentiel d'attrition. La prédiction de ce phénomène est l'un des cas d'usages les plus fréquents de la data-science.

**Evaluation et Maintenance du Modèle**

Afin que le modèle reste viable, de nouvelles données labelisées doivent être utilisées afin de tester ses performances. Ces données peuvent provenir, dans ce cas, des informations récoltées sur l'utilisateurs à n+1 Mois. En effet, le présent modèle a été entraîné sur des données utilisateurs de la plateforme KKBOX du mois de Mars 2017. Dans un cas réel, les données peuvent être récupérées depuis la base de données pour le mois d'Avril 2017 par exemple. Ainsi, le modèle peut confronter ses prédictions faites à partir des données du mois de Mars sur les données réelles du mois d'Avril.

Dans cette démonstration, les données labelisées proviennent de Mars 2017. Les métriques durant la première évaluation du modèle sont présentées. Une matrice de confusion est générée sur les nouvelles données importées.

**Démonstration sur un CSV**

Glissez et déposez le fichier ici ou...  
IMPORTEZ VOTRE CSV  
Evaluer le modèle

## Analyse sur le Fichier CSV : 2273 Utilisateurs

**Désabonnement Potentiel**

Identifiant	Expiration Abonnement	Risque de Partir (%)
jgOUChQzzMnu	2018-04-20	99
VPyvqYyuwb6R	2017-09-21	99
MmdgJPWqfGlt	2017-09-26	99
mdhgLqYO9quM	2018-03-14	99
I47rD6vrc7Cd	2017-06-03	99
k8oMllrTqKao	2018-05-11	99
8cEmctVOX2II	2017-06-27	99
vFcBM/iWqOro	2018-04-24	99
oCPzTokdScJp	2017-06-01	99

Télécharger le CSV

**Proportion des clients et Probabilité de risque**

Highcharts.com

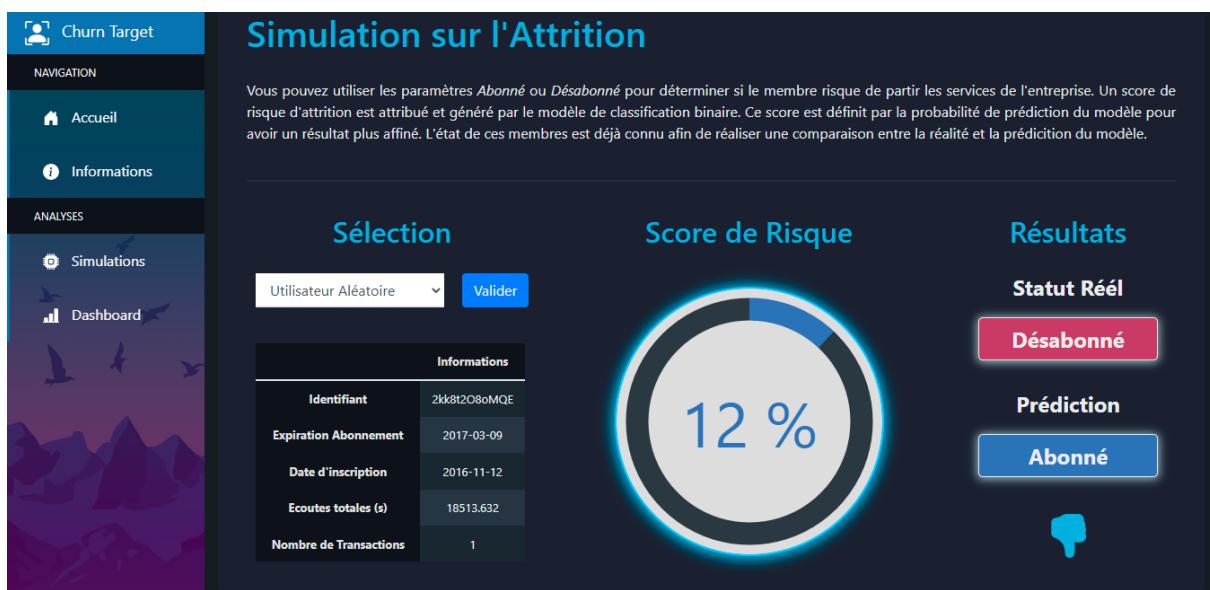
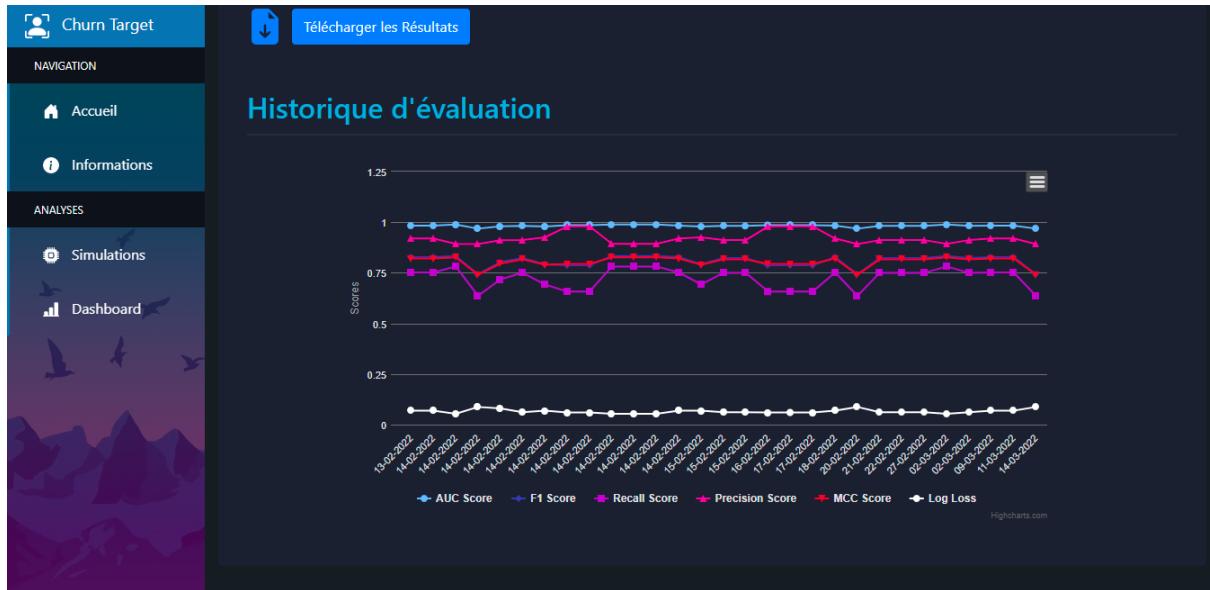
**Rapport Sensibilité/Precision**

Highcharts.com

**Répartition des probabilités**

Faux Negatifs  
Vrais Positifs  
Highcharts.com

Télécharger les Résultats



 Churn Target

 Interrupt Handler Déconnexion

**NAVIGATION**

-  Accueil
-  Informations

**ANALYSES**

-  Simulations
-  Dashboard

# Réalisation du Projet

Projet Chef-d'Oeuvre pour la Certification Développeur IA

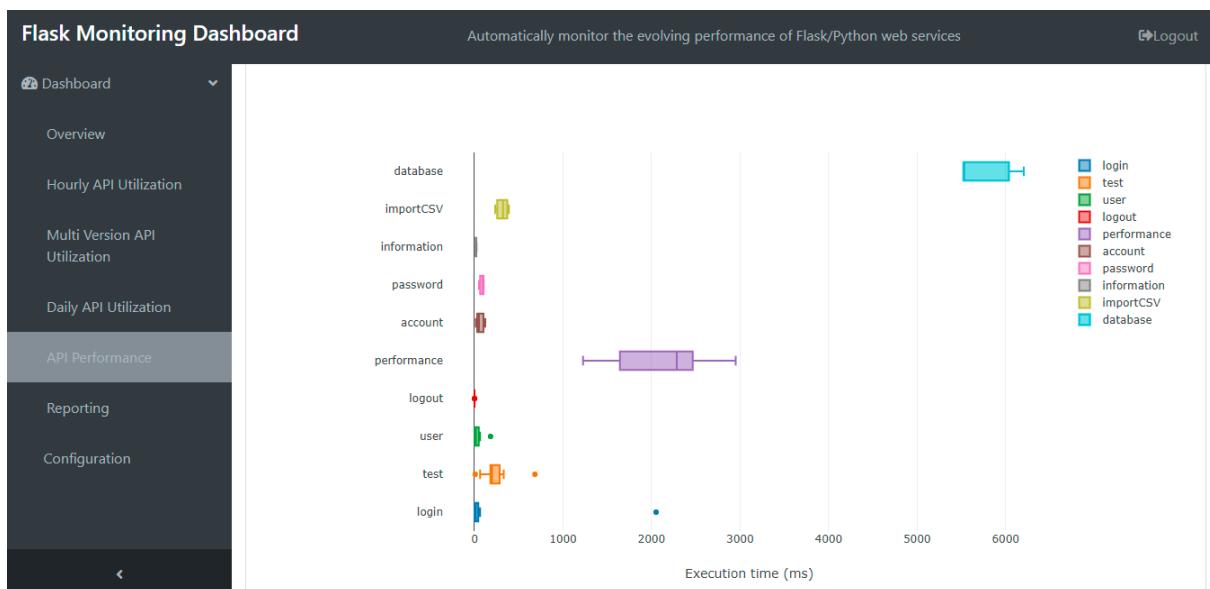
## Analyses et Conception du Dataset

Les jeux de données proviennent d'une compétition Kaggle présentant de nombreux fichiers CSV d'entraînement et fichiers de soumissions. Plusieurs fichiers (Version 2 et 3) sont à disposition comportant des données des utilisateurs jusqu'à 2017. Le but est ici de créer un fichier CSV unique qui sera ensuite intégré dans une base de données relationnelle type MySQL. Le nombre d'utilisateurs uniques diffère d'un CSV à l'autre et une étude approfondie sur un Dataset global apporterait des biais dans la proportion d'individus désabonnés/abonnés.

Ainsi, les études ont été faites individuellement pour chaque CSV *Transactions*, *Membres*, *Logs* avec un merge de *Train*. De plus, le nombre d'utilisateur global sur les CSV *Transactions* et *Logs* est plus élevé que le nombre d'utilisateur unique, indiquant un aspect temporel des informations. Dans le cadre de ce projet, seule la dernière en date pour chaque membre sera pris en compte.

L'analyse complète effectuée en amont de la création de modèle est disponible sur le [ici](#).

## Conception du Modèle



**Flask Monitoring Dashboard**

Automatically monitor the evolving performance of Flask/Python web services

[Logout](#)

Request id: 158      Request date: 15-03-2022 01:17      [Expand all](#)

Code-line	Duration	Percentage
@app.route('/database',methods=['GET','POST'])	2.1 sec	100.0%
def database():	2.1 sec	100.0%
main = connect.DB_dataset(year, month)	2 sec	98.4%
return render_template("error.html")	34.1 ms	1.6%

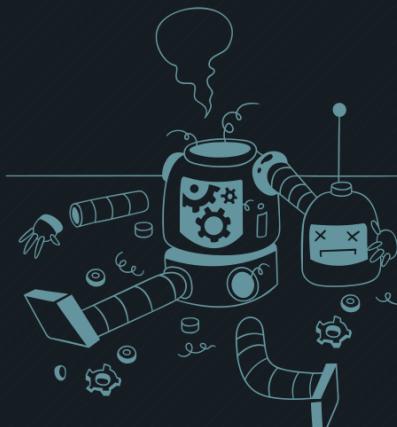
  

Request id: 144      Request date: 14-03-2022 19:20      [Expand all](#)

Code-line	Duration	Percentage
@app.route('/database',methods=['GET','POST'])	5.5 sec	100.0%
def database():	5.5 sec	100.0%
main = connect.DB_dataset(year, month)	5.3 sec	96.2%

## Une erreur est survenue :/

- Le serveur MySQL n'est peut-être pas connecté, impossible d'accéder à la session
- Si vous accédez aux résultats, assurez vous que votre fichier CSV respecte le bon format
- Si vous accédez à la base de données, il se peut que la requête ne retourne aucun résultat



**Churn Target**

**NAVIGATION**

- Accueil
- Informations

**ANALYSES**

- Simulations
- Dashboard

## Prédiction sur le phénomène d'Attrition

L'attrition est le fait, pour un client (individu, entreprise, etc.) de quitter un fournisseur de biens ou de services, une marque, un produit ou un service. Le taux d'attrition (*« churn rate »* en anglais) se mesure comme le pourcentage de clients perdus, sur une période donnée (en général une année ou un mois) par rapport au nombre total de clients figurant dans la base clientèle au début de cette période.

Il dépend de la capacité de l'entreprise à retenir ses clients, à les fidéliser par une offre pertinente. Elle conduit souvent à l'instauration d'un score prédictif qui sert à déclencher des actions marketing et commerciales.



You pouvez accéder aux prédictions des utilisateurs enregistrés en base de données ou en chargeant directement un CSV prévu à cette effet :

Année      Mois      [Valider](#)

[IMPORTEZ VOTRE CSV](#) Import\_demonstration.csv      [Valider](#)

**Churn Target**

**NAVIGATION**

- Accueil
- Informations

**ANALYSES**

- Simulations
- Dashboard

## Simulation sur l'Attrition

Vous pouvez utiliser les différents paramètres pour déterminer si votre nouveau membre risque d'abandonner les services de l'entreprise. Un score de risque d'attrition est attribué et généré par le modèle d'Intelligence Artificielle. Ce score est défini par la probabilité de prédiction du modèle pour avoir un résultat plus affiné.

Méthode de paiement  
Durée d'abonnement  
Prix catalogue  
Montant réel payé  
Autorenouvellement ?  
Annulation ?  
Choisissez une ville  
ID enregistrement

Valider

**Risque d'Attrition**

61 %

Une personne de Wuhan prenant un abonnement de 30 jours et ayant payé la somme de 119 \$ pour les services de l'entreprise, tout en ne renouvelant pas son abonnement et ne l'ayant jamais annulé a un risque modéré de partir.

**Churn Target**

**NAVIGATION**

- Accueil
- Informations

**ANALYSES**

- Simulations
- Dashboard

### PROPORTION D'INDIVIDUS EN FONCTION DU RISQUE DE RUPTURE

id_membre	montant_reel_payé	Risque d'Attrition (%)
3895144	149	100.0
1366803	180	100.0
4915336	149	100.0
5136166	0	100.0
5283216	0	100.0
4513227	0	100.0
5138676	0	100.0
3369934	149	100.0
5280715	149	100.0
4271692	149	100.0

### PROPORTION ATTRITION PAYMENT\_METHOD\_ID

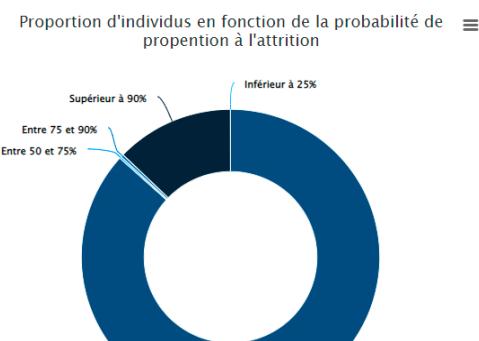
### PROPORTION ATTRITION TEMPS ABOUNNEMENT

Projet Chef d'oeuvre Informations Test Simulation

## Prédictions sur les nouvelles données

Le Modèle classe les nouvelles données d'entrée et émet une probabilité de risque départ de chaque Client. Plus le score est élevé, plus le risque que la personne quitte les services de l'entreprise l'est autant.

	msno	predictions	propensity_to_churn(%)
400	vsUSV1CLgH4+NYUWz9b2TT1/+W63T3DnBRaoYfwII9Y=	1	99.92
66	H5Lh5xiaCc4cxihRLMeBFCNmBV0m+xAbLZQmwpGbgpM=	1	99.92
766	tJ07WK9wgB/+bbg6JVO0zvfhgUSKCDb+8Qj0oGs6keM=	1	99.92
188	gPatjEiN0NtMkcg06yw7oZsppG9ukv79M+mAVJLkj0=	1	99.92
218	IJZ0qXzsItHn+na9tlOvh9/HZufJA0uBq/oSV6iUVWc=	1	99.91
418	1r5hJvwBJSmj8OlnFxuqVqKcaBY1L2ddY1lklYFm7rE=	1	99.91
301	sqNkUx+6BP21i7thGJT41InckNuJBZXP5rTuCAAFBbc=	1	99.91



```

EXPLORATEUR ... app.py
ÉDITEURS OUVERTS app.py
APP PROJET CO - WITH LOGIN M... [+] [-] ⌂
> Ressources
  static
    images
    js
      script_graph2.js
      script_graph3.js
# main.css
templates
  access.html
  base.html
  create_account.html
  database.html
  error.html
  importCSV.html
  index.html
  information.html
  log.html
  password.html
  performance.html
  simulation.html
.env
app.py
connect.py
flask_monitoringdashboard.db

alerte = 'Compte créé !'
alerte_color = '#0c7cbdb5'

else :
    alerte = 'Mots de passe saisis invalides !'
    alerte_color = '#c43939d'

else:
    alerte = 'Identifiant indisponible'
    alerte_color = '#c43939d'

return render_template('create_account.html', alerte = alerte, login = login, password = password)
confirm = confirm, alerte_color = alerte_color)

return render_template('create_account.html', alerte = alerte)

except:
    connect.alert("Le serveur SQL est inaccessible. Route : /create_account")
    return render_template('error.html'), 500

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL
Loading personal and system profiles took 5710ms.
PS C:\App Projet CO - with Login Manager>
Contenu de session restauré à partir de 29/04/2022 à 15:29:36
PowerShell 7.2.2
Copyright (c) Microsoft Corporation.

https://aka.ms/nowershell

```

**Annexe 24.** Infrastructure de l'application sur VS code.

```

PS C:\Users\utilisateur\Desktop\TEST> & C:/Users/utilisateur/anaconda3/python.exe c:/Users/utilisateur/Desktop/TEST/conftest.py
Traceback (most recent call last):
  File "c:/Users/utilisateur/Desktop/TEST/conftest.py", line 13, in <module>
    test_login()
  File "c:/Users/utilisateur/Desktop/TEST/conftest.py", line 11, in test_login
    assert login(input_login, input_password) == (actual_login, actual_password)
AssertionError

```

**Annexe 25.** Exemple de retour d'erreur sur l'assertion de la fonction Login dans le fichier conftest.py.





**LE JONCOUR Jérémie**

18 Mai 2022

---

# Cas Pratique E2

## Amélioration de l'IA

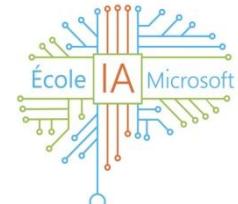
Conception d'une application pour la détection de portes

---

## Titre Professionnel de Niveau 6

Développeur en Intelligence Artificielle  
Enregistré au RNCP sous le n°34757

**SIMPLON**.CO



# SOMMAIRE

---

<b>INTRODUCTION.....</b>	<b>87</b>
<b>PARTIE 1 : Preprocessing .....</b>	<b>88</b>
1. Importation des images.....	88
2. Initialisation .....	88
<b>PARTIE 2 : Modélisation &amp; Test .....</b>	<b>89</b>
1. Architecture du modèle .....	89
2. Evaluation .....	90
3. Test du modèle créé sur image .....	92
<b>PARTIE 3 : Application .....</b>	<b>93</b>
1. Développement de l'application .....	93
2. Résultats .....	94
<b>CONCLUSION.....</b>	<b>95</b>

# INTRODUCTION

---

Cette épreuve s'intègre dans l'obtention du Titre Professionnel de Développeur en Intelligence Artificielle. Formant la deuxième partie pratique, le projet a pour but d'améliorer une IA et de créer une interface graphique. Le sujet propose de créer une IA capable de détecter des portes en reportant leur état d'ouverture (ouverte, entre-ouverte ou fermée). Cette étude peut se rapprocher des recherches récentes sur la détection d'obstacle par exemple et portant surtout sur la mise en place de systèmes intelligents mobiles. Ces robots sont représentés sous divers formes et ayant des tâches spécifiques comme les aspirateurs intelligents, des robots de livraison, des robots de sécurité, des systèmes d'aide-soignant, qui aident les personnes en difficulté dans leurs tâches quotidiennes (Zhihao He & Ming Zhu. 2017). Cette problématique peut aussi se rattacher aux mesures de sécurités et de surveillances des portes où plusieurs méthodes de détection et de différenciation des états d'ouverture sont appliquées (B.Quintana et al. 2018).

*Dans notre cas, une entreprise qui propose des solutions technologiques en caméra de surveillance et sécurité a développé un modèle IA capable de détecter et localiser la présence d'une porte à partir d'une image ou vidéo. Cette entreprise cherche à améliorer et étendre cette application pour classifier si la porte détectée par le premier modèle IA est ouverte, fermée ou entrouverte.*

*Le besoin de cette entreprise est une application Interface Python ou Page Web capable de :*

- *Etape 1 : Charger une image, vidéo ou activer la caméra.*
- *Etape 2 : Exécuter la localisation et encadre la porte.*
- *Etape 3 : Faire revenir le résultat et afficher sur la photo, ou la vidéo le résultat du modèle de classification.*

Afin de répondre aux objectifs, les deux premières parties feront mention du *preprocessing* d'images et de la construction du modèle de classification. La partie suivante regroupera l'ensemble des informations sur l'interface graphique mise en place.



## Lien GitHub du Projet

<https://github.com/JeremyLeJoncour/E2-Amelioration-IA>

Ressources :

Zhihao He & Ming Zhu. 2017. *Real-time Door Detection for Indoor Autonomous Vehicle.*

B.Quintana et al. 2018. *Door detection in 3D coloured point clouds of indoor environments*

JG. Ramôa et al. 2021. *Real-Time 2D-3D Door Detection and Classification on Jetson Nano.*

# PARTIE 1: Preprocessing

## 1. Importation des images

Afin de créer le modèle de classification des portes, un ensemble d'images est proposé dans le cadre de cette épreuve. Le Dataset contient des images en couleur de taille originale (dossier *OriginalSize*) ou rognée (dossier *Cropped*) afin de prendre uniquement en compte la porte.

Chacun de ces dossiers comprend des sous-dossiers prévus pour l'apprentissage du modèle (dossier *Train*, *Valid*, *Test*). Le Dataset est importé sur *Google Colab* afin que ces images soient traitées avant l'entraînement. Une première réflexion s'est portée sur le choix des images à utiliser dans le développement du modèle de classification. Etant donné que le modèle de détection des portes existe déjà, celui-ci génère un cadre (*box*) entourant la porte. Il était donc convenable d'utiliser les images du dossier *Cropped* afin de faire abstraction de l'environnement.

Après définition des *Paths*, les images *Train* et *Valid* peuvent être affichées. Plusieurs vérifications ont été effectuées afin de s'assurer que les données ont bien été importées dans l'environnement *Colab*.

Le Set *Train* comprend 548 images de portes ouvertes, 428 de portes fermées, et 110 de porte entre-ouverte. Le Set *Valid* contient 20 images de chaque catégorie. Afin d'uniformiser la tailles de chacune des images, elles sont redimensionnées avec la méthode *cv2.resize* en 140 par 140 pixels. Ces images ont été intégrées dans des listes intermédiaires. Elles sont aussi associées à leur label respectif.

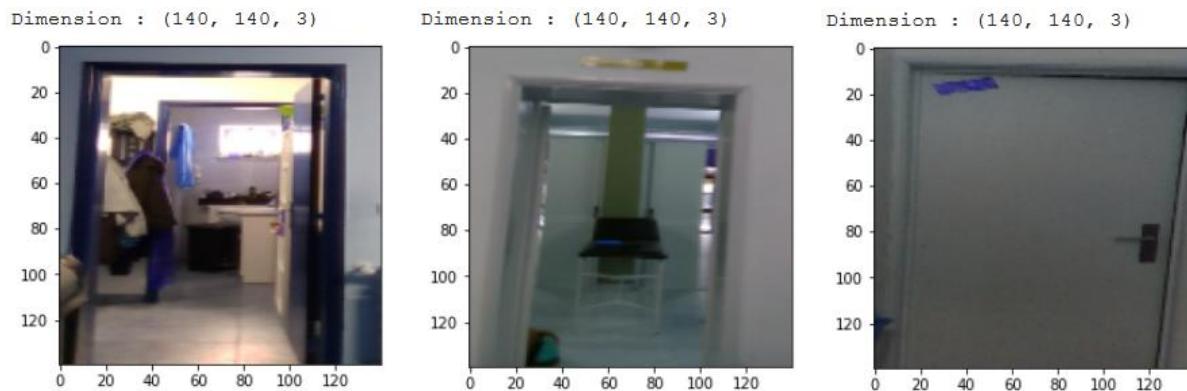


Figure 24. Affichage d'images au hasard (*Train*, *Valid* redimensionné).

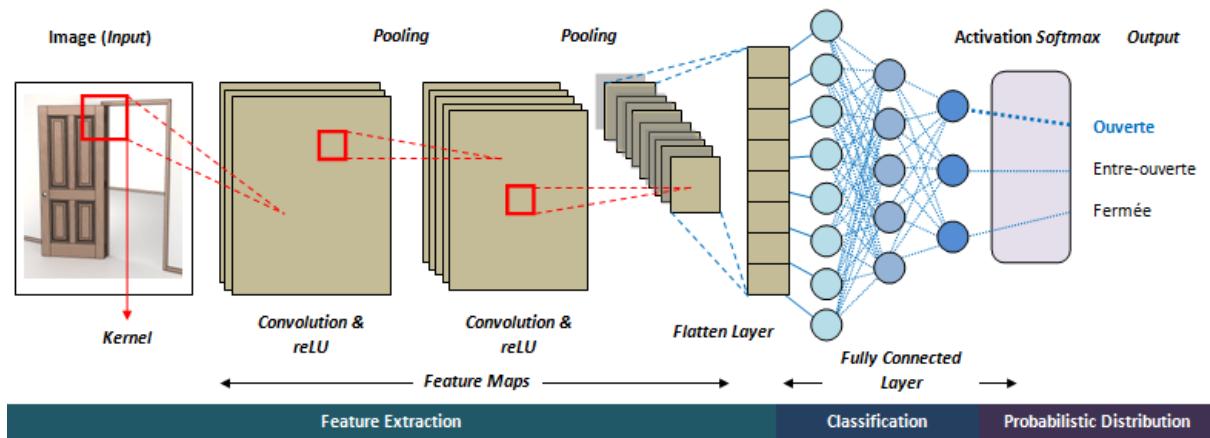
## 2. Initialisation

Les matrices correspondant à chaque image sont ajoutées dans les listes *X\_app* (données d'entraînement) et *X\_val* (données de validation). Leur label sont, quant à eux, ajoutés aux listes *y\_app* et *y\_val*. *X\_app* et *X\_val* sont normalisés avant de les intégrés à l'entraînement du modèle. Les labels sont aussi redimensionnés afin qu'ils puissent être pris en compte dans la dernière couche du modèle de classification. La méthode *keras.utils.to\_categorical* a permis de convertir ces vecteurs simples en matrices.

# PARTIE 2 : Modélisation & Test

## 1. Architecture du modèle

Portant sur de la classification d'image, le modèle est un réseau de neurones convolutifs (*Convolutional Neural Networks CNN*). Ces CNN s'inspirent du cortex visuel des animaux. L'ensemble des neurones les constituant permettent de traiter individuellement des portions d'informations contenues dans les images afin d'en ressortir une prédiction et en soi, une classification des images.



**Figure 25.** Schéma du principe du CNN. Des cartes de caractéristiques sont générées, poolées et vectorisées avant de passer dans le réseau de neurones qui classera l'image d'entrée en fonction de la probabilité de prédiction.

L'architecture du modèle de ce présent projet s'inspire des réseaux de classification d'images classiques réalisés durant la formation (panneaux de signalisation ou masques). Le réseau est construit sous Tensorflow Keras.

Pour l'initialisation de ce modèle, les couches *Conv2D* sont utilisées sur les traitements d'objets bidimensionnels. Le nombre de filtres (ou canaux de convolution) d'entrée commence à 32 (et double à chaque couche intermédiaire), avec une définition de *kernel* de dimension (3,3) idéale pour des images ayant une taille moyenne de 150x150 pixels.

L'*input\_shape* reprend les dimensions des images fournies au modèle (140x140 pixels en 3 couleurs RGB). Chaque couche est reliée à la suivante par une fonction d'activation *ReLU*, l'activation linéaire standard qui permet de fixer les valeurs négatives de ces matrices à 0. *MaxPooling2D* réduit les dimensions des images injectées et conserve les traits principaux.

Les dernières couches *Flatten* et *Dense* font la liaison entre les couches précédentes et convertissent les données en matrice à 1 dimension. L'activation *Softmax* retrouvée sur la dernière couche, performant pour la multiclassification, est finalement utilisée.

L'*optimizer* (descente de gradient) choisi est *Adam*. La fonction de perte associée (*Loss*) est *categorical\_crossentropy* prévu pour les problèmes de classification multiple. Le nombre d'*Epochs* est fixé à 50 pour un *batch\_size* de 16.

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_4 (Conv2D)	(None, 138, 138, 32)	896
conv2d_5 (Conv2D)	(None, 136, 136, 64)	18496
max_pooling2d_2 (MaxPooling 2D)	(None, 68, 68, 64)	0
conv2d_6 (Conv2D)	(None, 66, 66, 128)	73856
conv2d_7 (Conv2D)	(None, 64, 64, 256)	295168
max_pooling2d_3 (MaxPooling 2D)	(None, 32, 32, 256)	0
flatten_1 (Flatten)	(None, 262144)	0
dense_2 (Dense)	(None, 64)	16777280
dense_3 (Dense)	(None, 3)	195
<hr/>		
Total params:	17,165,891	
Trainable params:	17,165,891	
Non-trainable params:	0	

Figure 26. Summary de l'architecture du CNN construit.

Du fait du nombre restreint d'images, de la data-augmentation (*ImageDataGenerator*) a été effectuée, permettant de créer des copies modifiées de chaque image *Train*. Ces exemplaires peuvent être zoomés, rognés, décalés sur un axe de rotation.

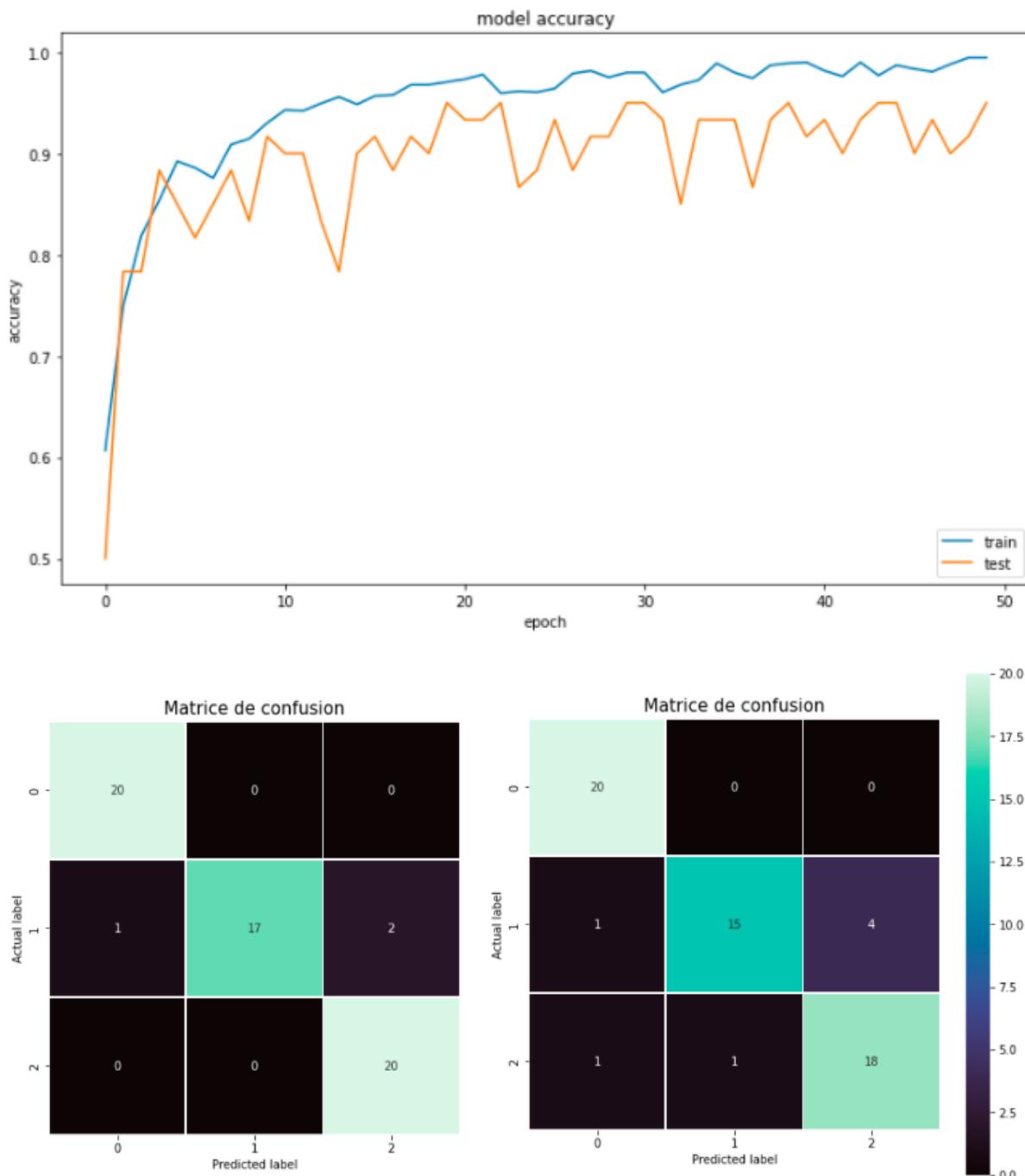
Afin de sauvegarder les performances maximales (les « poids ») du modèle durant l'entraînement, un *Callback* est initié par la méthode *keras.callbacks.ModelCheckpoint*. Les poids sont enregistrés dans un fichier *h5*.

A la fin de l'entraînement, le modèle est enregistré lui aussi dans un fichier *h5* qui pourra être chargé dans l'éventuelle application.

## 2. Evaluation

Les courbes d'apprentissage sur les métriques *Accuracy* (métrique d'exactitude) et *Loss* (par la fonction de perte) ont été affichées afin de vérifier l'évolution de l'entraînement du modèle. Un plateau est atteint à la fin de cette phase d'apprentissage. Grâce aux *checkpoints*, le modèle aux poids optimaux possède une *Accuracy* de 0.95 pour un *Loss* de 0.26 sur les données de validation.

Pour évaluer la répartition des images du set *Valid* classé par le modèle, une matrice de confusion a aussi été affichée. On rencontre des erreurs de classification sur la catégorie *Semi*. Le modèle prédit des images représentant des portes closes ou ouvertes alors qu'elles sont entre-ouvertes.



**Figure 27.** En haut, Courbe d'apprentissage sur la métrique *Accuracy* sur 50 Epochs. En bas à gauche, la matrice de confusion sur *Valid* et à droite sur *Test* (0 : *Closed*, 1 : *Semi*, 2 : *Open*).

Dans le cas de l'évaluation sur les images de *Test* (inutilisées durant l'apprentissage et la validation du modèle), il y a davantage d'erreurs de prédiction et d'autant plus dans la catégorie *Semi*. Toutefois, la majorité des images sont classées correctement dans leur catégorie respective.

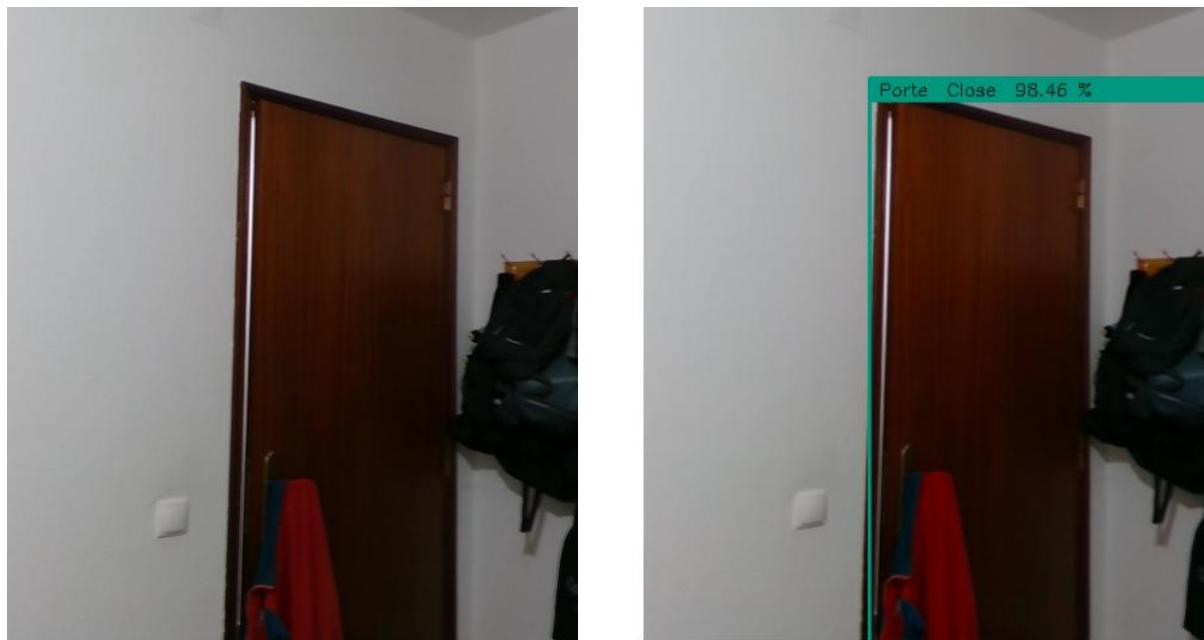
Un dernier test, appliquant le modèle de détection de porte est réalisé afin de visualiser ce que pourrait donner les résultats sur images avant d'intégrer ces deux IA dans une application.

### 3. Test du modèle créé sur image

Le modèle de détection de porte a été réalisé en amont et utilise un algorithme YOLO. Ce système de reconnaissance d'objet en temps réel prédit les objets d'une image et les signale sous la forme de cadre (*boxes*). Les images représentant des portes ont été labellisées dans un premier temps avant d'entraîner le modèle de détection spécifique. YOLO à l'avantage inhérent d'effectuer rapidement ses prédictions en une seule itération, contrairement à d'autres modèles qui détectent les régions d'intérêt séparément de la reconnaissance d'objet.

Un script permet d'afficher le cadre autour de l'objet détecté sur une image. Ses coordonnées peuvent donc être utilisées pour récupérer l'image contenue à l'intérieur et le passer au modèle de classification créé ultérieurement (d'où l'utilisation d'images du dossier *Cropped*). Cette nouvelle image est traitée en conséquence comme ce fut le cas pour les images *Train* et *Valid*.

Finalement, la porte est détectée par le premier modèle. Son statut d'ouverture est indiqué par la prédiction (et sa probabilité) grâce au deuxième modèle de classification.



**Figure 28.** Exemple de résultat sur image. A gauche l'image *Test* originale (*door0000528.png*), à droite l'image traitée par les deux modèles IA indiquant que la porte a bien été détectée et qu'elle est fermée (pour une probabilité de 98.5%).

# PARTIE 3 : Application

## 1. Développement de l'application



Pour répondre au projet, une application sommaire a été développée avec le Framework *open-source Streamlit*. Utilisant le langage Python, il a pour but de construire des interfaces convenables et facilement sans utiliser de langages *Front-end* spécifiques. Il a notamment été conçu pour présenter des projets de *Machine Learning* et de data-science.

Cette application propose d'utiliser les IA modélisées en amont sur différents fichiers, que ce soit des images, des vidéos ou encore permettant la possibilité d'utiliser directement la webcam. Le programme a donc été développé de telle sorte qu'il puisse répondre aux attentes de l'entreprise.

Les modèles de détection et de classification sont chargés, ainsi que les poids optimaux. Une liste contenant les labels des classes disposés dans le même ordre que lors de la création du *classifier* est aussi déclarée. Deux fonctions sont créées :

- Une fonction *cache\_video* permet de garder la capture vidéo en cache et permettant l'incorporation simultanée des traitements générés par les modèles IA. Cette fonction est utilisée pour la vidéo et l'utilisation de la caméra.
- Une fonction *image\_traitement* traitant l'image d'entrée et en ressortant la même image avec les cadres de détection de porte et le label de classification (et probabilité).

Finalement, un menu permet de naviguer et d'utiliser les différentes fonctionnalités. Pour les images, il y a la possibilité de les déposer directement sur l'interface. Pour l'utilisation de la caméra et vidéo, des boutons permettent de démarrer ou stopper la lecture :

The screenshot shows a dark-themed Streamlit interface. At the top, there's a title icon (a door) and the title "Interface de détection de portes". Below the title, a note says: "Vous pouvez sélectionner une image ou une vidéo afin d'utiliser les modèles IA de détection, vous pouvez aussi détecter les portes en temps réel en activant la webcam." A dropdown menu labeled "Choisissez :" is open, showing "Image" as the selected option. Below it, a text input field says "Choisissez une image". At the bottom, there's a file upload section with a "Drag and drop file here" button and a "Browse files" button. The "Limit 200MB per file" text is visible next to the upload area.

Figure 29. Interface *Streamlit* utilisée pour le développement de l'application IA.

## 2. Résultats

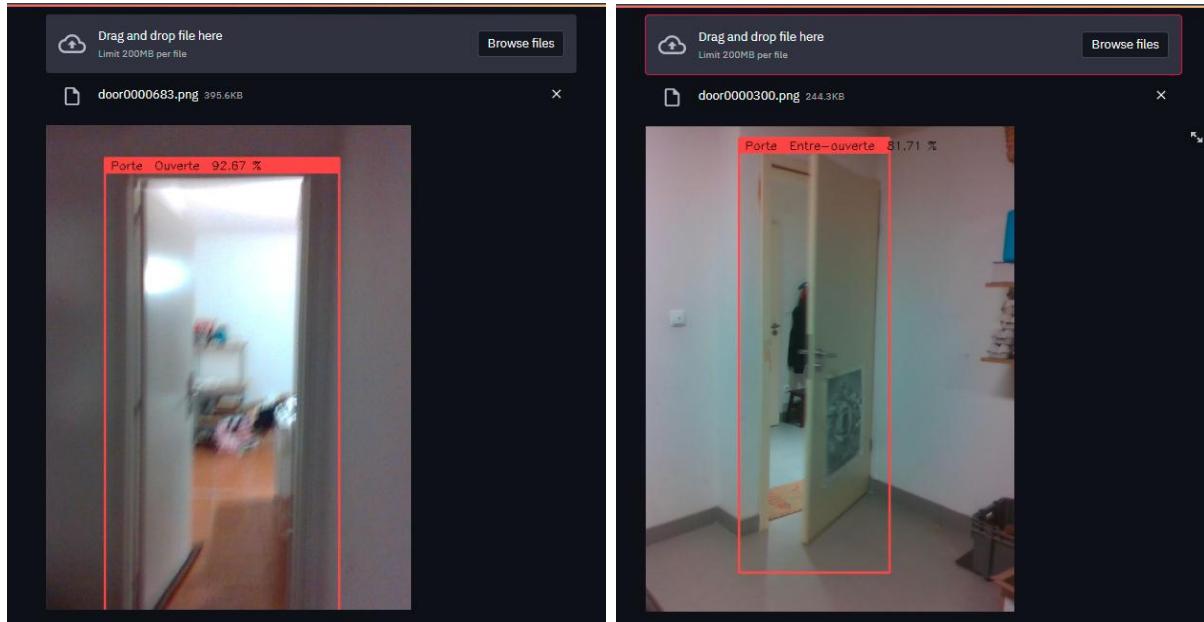


Figure 30. Images d'exemple de détection et classification de porte sur l'interface.

La majorité des images *Test* de tailles originales sont détectées par le premier modèle. Leur classification, affichée simultanément, reportent peu d'erreur.

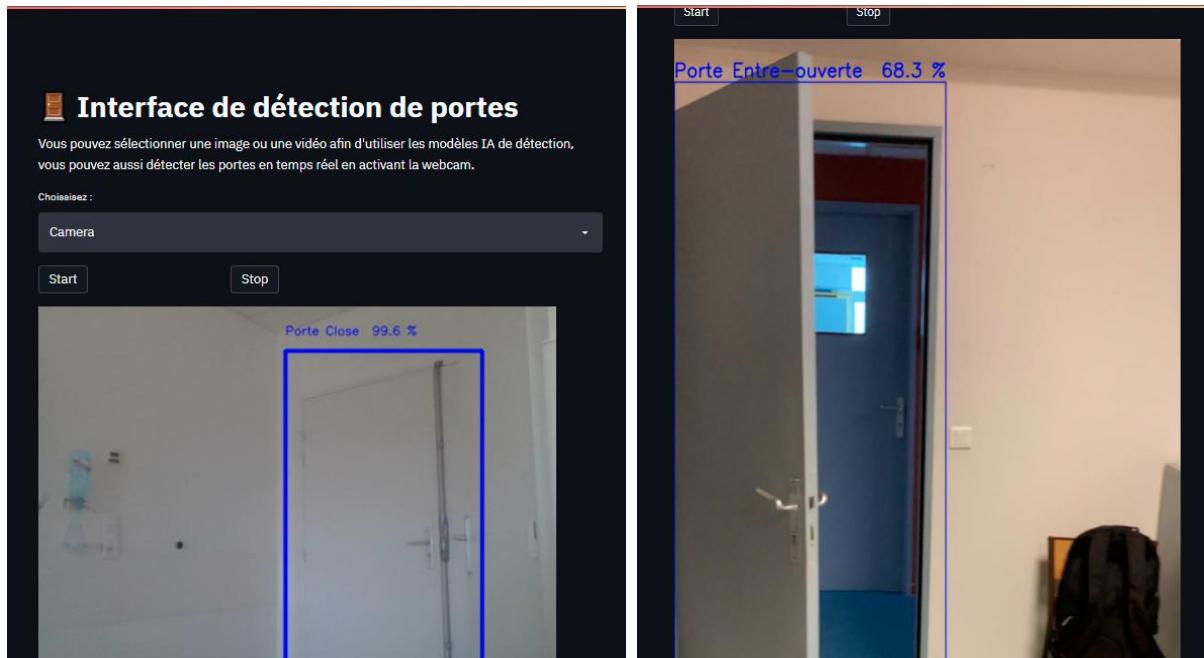


Figure 31. Test de l'interface sur Caméra et Vidéo d'exemple.

Les tests sur caméra et vidéo se sont bien déroulés. Les portes sont généralement bien détectées et classées. Cependant, des imprécisions ont été remarquées lors de la détection en temps réel. Des fenêtres, vitres ou autres éléments de forme rectangulaire sont considérés comme des portes par le modèle de détection.

# CONCLUSION

---

Dans le cadre de cette épreuve, l'amélioration de l'IA portait sur un ajout d'un modèle de classification différenciant l'état d'ouverture des portes. Quelques réseaux CNN, dont l'architecture s'inspire d'autres modèles existant, ont été construits. Le modèle final a été évalué sur deux métriques et présentait de bons résultats, notamment sur les matrices de confusion. Cependant, les courbes d'apprentissage présentent une variabilité de progression conséquente.

Lors de la phase de test sur l'interface graphique, construit sous *Streamlit*, la plupart des portes présentées dans les photos du set *Test* sont détectées par le modèle YOLO et correctement classées. Toutefois, sur la détection en temps réel, il existe quelques problèmes de détection et de classification. Le modèle de détection confond certains éléments et les considère comme des portes. Au niveau de la classification, le modèle prévu à cet effet présente quelques difficultés, majoritairement sur les portes entrouvertes.

Durant la phase d'entraînement, certaines images mises à dispositions étaient similaires entre les sets *Train*, *Valid* et *Test*, biaisant les véritables performances du modèle CNN de classification. Bien que la data-augmentation ait pu sans doute réduire l'écart entre les performances théoriques et les tests en temps réel, celle-ci n'est pas encore suffisante pour obtenir un modèle robuste et généralisant la classification des portes.

Par la suite, il serait intéressant d'évaluer le modèle de détection afin de fixer un seuil de probabilité optimal permettant de cibler uniquement les portes. Pour le modèle de classification, augmenter le nombre d'images et éviter de répartir les mêmes portes dans chaque set permettra d'avoir une vision des performances réelles du modèle en fonction de l'architecture adoptée. Plusieurs autres paramètres peuvent être aussi à modifier comme la taille des images d'entrée, l'utilisation d'un plus grand *Kernel* d'entrée, ajouter des *Dropout* pour éviter le sous/sur-ajustement, ajuster le *batch size*, *learning rate*...

Aussi, il peut être envisageable d'appliquer le principe de *transfert learning* afin d'associer un modèle pré-entraîné potentiellement plus robuste.

