RAPPORT DE PROJET DE FIN D'ETUDES

Promotion 2021

# Démission de clients dans la banque et l'assurance

*Etudiants :*
Antoine BOULINGUEZ
Alexis GASSMANN
Alex GEORGET

*Encadrants :*
Kaichen MA
Lamiae MKADMI

# Sommaire

**1 Introduction** **2**

**2 Théorie** **3**
2.1 Démarche . . . 3
2.2 Modèles . . . 3
2.3 Métriques . . . 5
2.4 Traitement des données textuelles . . . 6

**3 Etude des données** **8**
3.1 Les jeux de données . . . 8
3.2 Analyse descriptive . . . 9
3.3 Modélisation et résultats pour les données Kaggle . . . 12
3.4 Feature importance . . . 13
3.5 Feature engineering . . . 16
3.6 Résultats pour les données textuelles . . . 16

**4 Application de présentation des résultats** **18**

**5 Conclusion** **19**

**Bibliographie** **20**

**Annexes** **21**

# 1 Introduction

L'étude de la démission, aussi appelée analyse de churn (de l'anglais "churn analysis") est un domaine qui a pour objectif d'évaluer la perte de clients d'une entreprise, d'en comprendre les causes et éventuellement d'effectuer des prédictions, afin notamment de prendre des décisions stratégiques, d'adapter son offre et d'améliorer ses résultats. L'analyse de churn est très utilisée dans le domaine bancaire, le domaine de l'assurance ou encore la télécommunication car ces activités récoltent par définition un grand nombre de données relativement précises sur leurs clients afin de leur proposer des produits et services adaptés, peu risqués, et rentables pour l'entreprise. De nombreuses recherches ont été faites dans le domaine des télécommunications (cf [ [5], [3]]). Dans notre cas nous nous intéresserons au cas bancaire.
La concurrence est rude et il est aujourd'hui simple de changer de banque ou de contrat d'assurance rapidement sur internet. Il est donc intéressant pour ces entreprises de s'intéresser à la démission de leurs clients grâce aux données dont elles disposent et à des méthodes statistiques et algorithmiques pour aider à la description et à la prédiction.

L'objectif principal est bien sûr de retenir un maximum de clients. Toutefois, l'analyse de churn en général a de multiples avantages : gain de connaissances pour améliorer l'entreprise, meilleure compréhension des clients, réduction du risque et construction d'un avantage concurrentiel.

L'analyse de churn peut prendre plusieurs formes, qui dépendent évidemment des données à disposition. L'idéal est d'avoir des données régulièrement actualisées et riches en informations, mais ce n'est pas toujours possible. Par exemple, on peut s'intéresser à la raison du départ d'un client (passage chez un concurrent, plus besoin du produit/service, etc.) ou encore à des facteurs caractérisant le client (engagement dans l'entreprise, fidélité, rentabilité, etc.). Pour une bonne analyse de la démission,, comme pour toute étude statistique, il est donc important d'avoir des données qualitatives et variées : pour cela, les entreprises ont plusieurs outils à leur disposition comme le questionnaire de satisfaction, aujourd'hui systématiquement proposé à un client lorsqu'il décide de mettre fin à un contrat (QCM, texte libre).

Dans le cadre de ce projet, nous avons décidé de nous concentrer sur l'aspect modélisation et prédiction, avec une approche machine learning de la démission traitée de façon binaire. Dans un premier temps nous travaillerons avec des données numériques en libre accès concernant une base de clients de banque, puis dans un deuxième temps avec des données textuelles (commentaires des clients). Ainsi, l'objectif du projet sera de décrire les données étudiées, de comparer des modèles d'apprentissage supervisé pour la classification binaire et de présenter ces résultats. Pour la partir analyse textuelle, des modèles spécifiques au NLP seront utilisés.

# 2 Théorie

## 2.1 Démarche

Notre objectif est de comparer différents modèles d'apprentissage supervisé appliqués à des données du domaine bancaire, afin de prédire de façon binaire si un client va quitter l'établissement ou y rester. Pour cela nous utiliserons 2 jeux de données : un jeu de données présent sur Kaggle contenant des information sur des clients d'une banque sous forme de tableau (qui est notre jeu de données principal), et un jeu de données scrappées que nous utiliserons pour de l'analyse textuelle.

En ce qui concerne le jeu de données issues de Kaggle :

Le jeu de données ne présente que des variables numériques continues, discrètes ou des variables catégorielles. Il ne nécessite pas énormément de traitement particulier. On effectuera principalement du one-hot encoding des variables catégorielles ainsi qu'une normalisation des données.

Pour la séparation entraînement/test, on sépare le jeu de données en 2 parties (80% pour l'entraînement et 20% pour le test). Le jeu d'entraînement pourra être utilisé en validation croisée (5 blocs) de type Stratified K-fold : ainsi on conservera le ratio positif/négatif (churn/non-churn) dans chaque sous-échantillon pour représenter au mieux l'échantillon général.

Avant de s'intéresser au modèles d'apprentissage, on effectuera une analyse descriptive des données (comparaison à l'aide de graphiques, corrélation entre variables, etc) afin de mieux répondre à la problématique. Cela pourra éventuellement nous donner des idées de feature engineering à tester lors de l'apprentissage de nos modèles.

Pour chaque modèle on effectue les étapes suivantes :

1. On recherche les paramètres optimaux à l'aide d'un GridSearch par validation croisée (5-fold), puis on retient les paramètres jugés les meilleurs, c'est-à-dire ceux qui donnent les meilleurs résultats au regard d'une certaine métrique (ici le F1-score).
2. On effectue ensuite une validation croisée 5-fold sur chacun des modèles avec ses paramètres optimisés.
3. Pour finir on effectue les prédictions sur le jeu de test.

A la fin, on peut donc comparer les différents modèles en temps de calcul, selon les différentes métriques, en validation comme en test. On pourra également s'intéresser au feature importance afin de compléter l'analyse descriptive et de mieux comprendre la construction des modèles sur nos données.

L'analyse de données textuelles reprend à peu près la même démarche (traitement, entraînement , comparaison des modèles) avec des spécificités liées au NLP qui seront précisées à part dans le rapport.

## 2.2 Modèles

### Régression Logistique

La régression logistique est un modèle linéaire généralisé qui peut s'écrire sous la forme suivante :

$\forall i \in \{1, ..., n\}; \sigma_{w,c}(X_i) = \frac{1}{1+exp(-X_i^T w+c)} = P(Y_i = 1|X_i, w, c)$
où X est la matrice des covariables, Y le vecteur réponse (binaire) et $w$ et $c$ les paramètres.

C'est un modèle souvent utilisé pour la classification binaire. Nous utiliserons ici la régression logistique dans un cadre de machine learning. Ce type de modèle a pour avantage d'être simple à entraîner et à comprendre, mais n'est pertinent que dans les cas où il y a une certaine linéarité entre la réponse et les covariables.

On cherchera à répondre au problème d'optimisation suivant (minimisation de la log loss) :

$\min_{w,c} \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$

Il est également possible d'obtenir une version régularisée du problème, permettant de limiter le sur-apprentissage :

$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$ ; pour la régularisation $l_2$
$\min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$ ; pour la régularisation $l_1$
Le paramètre $C$ permet d'ajuster le niveau de régularisation.

Ces problèmes peuvent être résolus par descente de gradient, ou par d'autres algorithmes d'optimisation (présents dans les bibliothèques classiques de machine learning).

Dans notre cas, nous utiliserons la régression logistique de scikit-learn, avec l'optimiseur SAGA qui permet de gérer toutes les méthodes de régularisation. On équilibrera également les poids de classes. Les autres paramètres (C et le type de régularisation) sont optimisés par validation croisée.

**RandomForest**

L'algorithme de forêts d'arbres décisionnels (random forest classifier) est un algorithme d'apprentissage automatique utilisant la méthode du bagging. Il permet de gérer des problèmes de régression comme de classification (même si à l'origine il a été conçu en classifcation binaire) Son principe est d'agréger des arbres binaires faibles (par exemple des CART) dans le but de réduire la variance de l'estimateur. Pour cela, la méthode RandomForest repose sur plusieurs étapes :

1. On choisit un nombre B d'arbres à agréger.
2. Pour chaque arbre décisionnel, on utilise un échantillon d'entraînement bootstrap obtenu à partir de l'ensemble des observations (principe du bagging).
3. De plus, on peut sélectionner aléatoirement un nombre restreint $m$ de covariables parmi les $p$ covariables d'origine (souvent tel que $m < \sqrt{p}$).
4. Après avoir entraîné les B arbres, on peut obtenir la prédiction finale pour chaque observation à l'aide d'un vote à la majorité.

Par le passé, d'autre recherches dans le churn prediction ont usé du modèle RandomForest comme dans [2]. Dans notre cas on utilisera RandomForestClassifier de scikit-learn. Nous optimiserons par validation croisée le nombre d'arbres, la profondeur maximale des arbres, le nombre minimum d'observations par noeud et le nombre minimum d'observations par échantillon bootstrap.

Le RandomForest, comme la plupart des algorithmes d'ensemble learning fondés sur des arbres, il a pour avantage de fonctionner en régression et en classification, de limiter le sur-apprentissage et de rester assez interprétable grâce à l'importance des variables. Toutefois, il reste assez "boîte noire" comme beaucoup d'algorithmes de machine learning et peut être assez lent.

**XGBoost, CatBoost et LightGBM**

Certains travaux [1] ont déjà été mené sur le churn prediction avec des modèle de boosting. Le boosting est une méthode d'apprentissage supervisé qui consiste à agréger des règles faibles (comme des arbres de décisions) pour minimiser le biais et éventuellement la variance. Il donne des modèles à forte capacité de prédiction mais avec une tendance à sur-apprendre. Il fonctionne itérativement en entraînant des règles faibles et en modifiant à chaque fois les poids des observations en fonction de la performance de la prédiction obtenue. Ainsi, les observations qui sont mal prédites prennent plus de poids dans la construction des règles. Dans le cas du Gradient Boosting, la règle finale est la somme pondérée des règles faibles, obtenues itérativement en cherchant pour chaque nouvelle règle ajoutée à réduire l'erreur de prédiction.

XGBoost, CatBoost et LightGBM sont des frameworks permettant l'optimisation fine d'algorithmes de type gradient boosting en utilisant au mieux les capacités de la machine et les principes du machine learning (optimisation d'hyperparamètres notamment). Les trois frameworks sont reconnus dans le milieu du machine learning et utilisés régulièrement dans le cadre de data challenges. Les défauts inhérents aux modèles issus de ces framework sont, tous comme bon nombre de modèles de machine learning, la lenteur d'apprentissage (en partie corrigée par le travail d'optimisation du framework) et l'aspect boîte noire, mais qui est ici amplifié par la surcouche du framework. Pour palier à cela, certain outils ont été développés comme le package SHAP qui permet d'obtenir des informations sur l'importance de chaque feature dans l'apprentissage d'un modèle de Tree Boosting. Toutefois ces défauts sont contrebalancés par le côté pratique des frameworks (facilité d'utilisation, optimisation, versatilité) qui permet d'entraîner des modèles complexes sans devoir écrire tout le code à la main.

#### Réseau de neurones

De manière simple, un réseau de neurones est un algorithme d'apprentissage automatique (supervisé dans la plupart des cas) qui peut être considéré comme un empilement de couches plus ou moins complexes de neurones artificiels dont les paramètres sont optimisés successivement pour réduire la fonction de coût (grâce à une descente de gradient ou une autre méthode d'optimisation). Pour cela il est nécessaire à chaque étape d'entraînement de calculer le perte (feed forward) puis de mettre à jour les paramètres du modèle grâce au résultat obtenu (back propagation).

Dans notre étude, nous utiliserons un modèle simple constitué d'une couche de batch normalisation (pour la stabilité), de 1 à 3 couches de type "fully connected" (i.e. une couche simple de neurones) avec la fonction d'activation ReLU ($f : x \mapsto max(0, x)$) et d'une couche de dropout (pour limiter le sur-apprentissage) puis d'une couche finale à un neurone avec la fonction d'activation sigmoïde ($f : x \mapsto \frac{1}{1+exp(-x)}$). Le nombre de couches, leur nombre de neurones et le taux de dropout ainsi que le learning rate sont optimisés par un GridSearch en validation croisée. On utilisera l'entropie croisée binaire comme fonction de perte.

Les réseaux de neurones ont pour avantage de pouvoir donner de très bon résultats, notamment lorsqu'on a beaucoup de données d'entraînement. Cependant ils sont longs à entraîner, sujets au sur-apprentissage et ont aussi un aspect "boîte noire".

#### Stacking

Le stacking est une technique d'ensemble learning qui consiste à utiliser les prédictions de modèles simples (idéalement peu corrélées) comme ensemble d'apprentissage d'un méta-modèle qui donnera les prédictions finales.

#### Blend Model (mean model)

Notre 'Blend" modèle d'ensemble learning consiste à utiliser la moyenne des prédiction de modèles simples (ici les modèles précédents sauf le stacking) pour obtenir la prédiction binaire finale (en utilisant un niveau, par défaut à 0.5). L'objectif est de réunir les avantages de tous les modèles, sans dépendre des caractéristiques inhérentes à un méta-modèle comme pour le stacking précédent.

### 2.3 Métriques

Afin d'entraîner et de sélectionner nos modèles il est nécessaire de comparer leur performance au regard de métriques. La métrique la plus classique est l'accuracy, qui représente simplement le pourcentage de bien classés.

Cependant elle n'est pas forcément la plus intéressante dans une problématique d'analyse de la démission. En effet, dans si on cherche à prédire la démission d'un client, on cherchera principalement à ne pas "rater" des clients qui risquent de quitter la banque. Pour cela on pourra utiliser le rappel (ou recall en anglais).

Le recall est le nombre de vrais positifs divisé par le nombre total de positifs dans la base. Cela est donc bien adapté à notre soucis de prédiction de churn (si on considère qu'un client démissionnaire est positif).

Toutefois, afin de ne pas totalement omettre les faux positifs (non-churn qui sont prédits churn) dans notre calcul, il peut être intéressant de combiner cette métrique avec la précision (nombre de vrais positifs sur nombre de prédictions positives). Le F1-score permet cela : c'est la moyenne harmonique du recall et de la precision.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

C'est cette métrique que nous chercherons à optimiser en priorité car c'est un "juste milieu" entre recall et precision.

La courbe ROC peut aussi être un outil intéressant pour comparer les modèles. Cela permet de comparer plusieurs modèles face au modèle idéal et au modèle aléatoire selon le taux de faux positifs et vrais positifs. L'aire sous la courbe (AUC) est une métrique dérivée de la courbe ROC.

Les métriques accuracy et AUC auront surtout un rôle indicatif.

## 2.4 Traitement des données textuelles

### Pré-traitement des données textuelles

Le prétraitement des données permet de réduire le vocabulaire du corpus pour faciliter l'interprétation du texte par le modèle de classification.
Une première étape consiste à supprimer toutes les ponctuations par des espaces et supprimer les majuscules afin de faciliter le traitement des données.
Ensuite, avec l'aide du package SPACY, nous pouvons appliquer différents prétraitements. Le POS-tagging permet de prédire la classe grammaticale de chacun des mots et donc de sélectionner seulement les mots de certaines classes grammaticales tels que verbe, nom commun, ou adjectif. La méthode supprimant les stop-words français consiste à supprimer les mots appartenant à une liste prédéfinie des mots fréquemment utilisés dans la langue française (exemple : 'a'). La lemmatisation est une méthode permettant de réduire les mots à leur lemme (radical) afin de réunir les mots avec un même sens et de réduire le vocabulaire du corpus (exemple : 'comptes' réduit à 'compte').

### Modèle Bag-of-word

Le modèle Bag-of-word est une manière de représenter un corpus de documents sous forme matricielle, documents x mots du vocabulaire. La représentation la plus simple est la matrice TF représentant la fréquence des termes dans les documents :

$$TF(t,d) = \frac{\text{nombre de fois le terme t apparaît dans le document d}}{\text{nombre total de termes dans le document d}}$$

Une autre représentation Bag-of-word est la matrice TF-IDF. La matrice TF-IDF est une matrice non négative définie pour un terme t dans un document d défini par :

$$TF - IDF(t,d) = TF(t,d) * IDF(t) \text{ où } IDF(t) = \frac{\text{nombre total de documents dans le corpus}}{\text{nombre de document contenant le terme t}}$$

La matrice TF ou TF-IDF obtenue peut ensuite être utilisée comme variable d'entrée à des modèles de classifications adaptés à la distribution des données textuelles : Naive Bayes classification, régression linéaire ou bien SVM.

**Modèle FastText**

Des modèles de Deep Learning ont été créés pour capturer les relations sémantiques des termes et vectoriser des termes respectant une notion de similarité. C'est un point qu'on ne retrouve pas avec la matrice TF-IDF qui se repose essentiellement sur la fréquence des mots.
Le modèle FastText est basé sur un réseau de neurones permettant de former des vecteurs pour des lettres n-grams. Un mot est représenté par un ensemble de lettres n-grams. L'approche est basée sur le modèle skip-gram, pour chaque lettre n-grams, le modèle prédit les lettres n-grams entourant la lettre n-grams.
L'apprentissage du modèle apprend sous une forme de couche cachée une matrice de dimension V x H avec H la dimension de la couche cachée et V la taille du vocabulaire du corpus. Le résultat de cette matrice permet de donner un vecteur de taille H à chaque n-lettre. La représentation d'une n-lettre après entraînement est unique et ne dépend pas du contexte de la phrase.

**Mécanisme Attention**

Le modèle FastText permet d'obtenir une représentation vectorielle pour chaque mot. Pour construire l'entrée d'un modèle de classification, la première idée serait de sommer les vecteurs des mots pour obtenir un seul vecteur représentant le document. Cependant, les prépositions ou déterminants peuvent prendre le dessus dans l'addition et donc cacher les mots qui pourraient avoir une importance dans la classification. Le mécanisme Attention a pour but de montrer où est l'information pertinente en ajoutant un poids à chaque mot du document.

Le vecteur du document $d_i$ dépend donc des sorties de la couche précédente $(h_1, ..., h_m)$ et des poids du mécanisme attention $\alpha_i$ :

$d_i = \sum_{j=1}^{m} \alpha_{ij} h_i$ avec $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{m} \exp(e_{ik})}$ et $e_{ij} = g(W, h_j)$

Où $W$ est la matrice des poids de la couche Attention et $e_{ij}$ mesure la relation de la sortie $h_j$ avec classification du document $d_i$. $\alpha_{ij}$ représente en tant que poids l'importance de la sortie $h_j$ (mot $j$) pour la classification de $d_i$.

**CamemBERT**

Le modèle CamemBERT repose sur l'architecture du modèle RoBERTa pré-entraîné sur le corpus français de OSCAR. RoBERTa repose essentiellement sur le mécanisme Attention dont l'avantage est d'apporter une importance des mots suivant la tâche. Le modèle est constitué d'une série d'encodeur (un encodeur est constitué d'une couche Attention et d'un réseau de neurones à propagation avant) et la tâche du modèle, sous forme semi-supervisé, consiste à prédire des mots du documents qui ont été masqués en entrée (15% des mots). De ce fait, certains mots auront plus d'influences que d'autres pour prédire les mots masqués d'où l'utilisation du mécanisme Attention.
Pour le modèle de classification, nous supprimons la couche pour prédire les mots masqués pour obtenir un vecteur pour chaque mot du document. L'avantage de ce modèle est de pouvoir obtenir une représentation vectorielle pour un même mot différent selon le contexte de la phrase.

# 3 Etude des données

## 3.1 Les jeux de données

### Données bancaires numériques

La mise à disposition de données ouvertes dans le domaine bancaire étant rare pour des raisons évidentes de confidentialité, nous n'avions pas beaucoup de choix à notre disposition. Notre attention s'est portée sur un jeu de données disponible sur Kaggle, qui est bien adapté à l'apprentissage supervisé pour la classification binaire.

Le dataset comprend 10000 obeservations. Chaque observation est associée à un client unique, représenté par un identifiant. L'identifiant n'étant pas intéressant, on ne l'utilisera pas dans la modélisation.

La variable à expliquer est la variable binaire *Exited*, qui prend pour valeur 0 si le client est resté dans la banque et 1 s'il a quitté la banque.

Les variables explicatives sont au nombre de 10 :

- CreditScore : méthode de notation du client (en fonction du salaire, nombre d'enfant, job, âge...)
- Tenure : Nombre d'années d'ancienneté du client dans la banque, à valeurs dans {0, 1,..., 10}
- Balance : méthode de notation (facteurs : comptes et dernier relevé bancaire), le but est de définir l'écart entre les recettes et les dépenses
- NumOfProducts : nombre de produit que le client a acheté à la banque, à valeurs dans {1,2,3,4}
- HasCrCard : possède (1) ou non (0) une carte de crédit
- IsActiveMember : membre actif (1) ou non (0) de la banque
- Geography : indique la nationalité du client (France, Germany, Spain)
- Gender : genre du client (Male, Female)
- Age : âge du client
- EstimatedSalary : salaire estimé du client

### Données textuelles

Nous avons récupéré des commentaires français concernant différentes banques sur le site TrustPilot. Dans l'objectif de la classification binaire, nous considérons un commentaire à sentiment positif s'il possède 5 ou 4 étoiles sinon nous l'associons à un sentiment négatif.

Nous avons choisis 3 banques : OrangeBank avec 628 commentaires et 46.5% de commentaires positifs, Floabank avec 2114 commentaires et 29.0% commentaires positifs, Cofidis avec 2441 commentaires et 91.2% commentaires positifs.

Exemples de commentaires :
"J'étais bien accueilli. Personne très souriante et accueillante c'était super" (positif)
"Je suis déçu du système d'orange bank et de son acharnement à me réclamer une somme indue." (négatif)

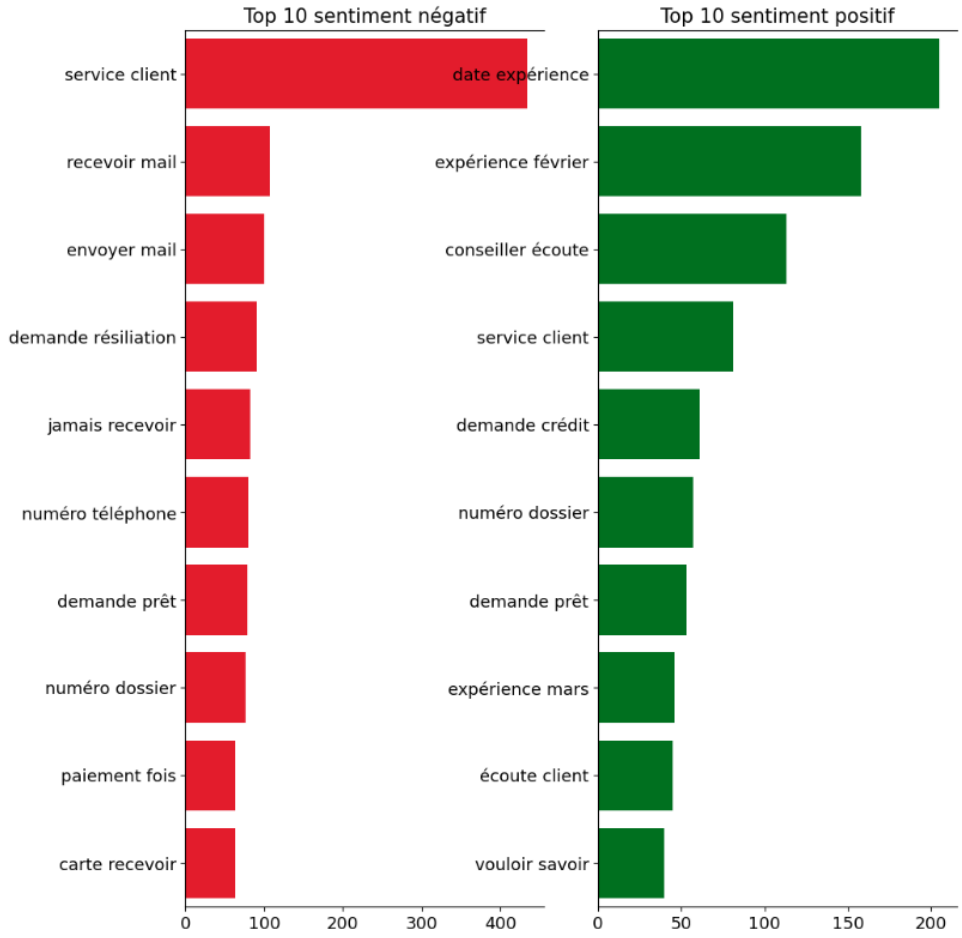


FIGURE 1 – Top bigramme après application des méthodes POS-tagging, stop-words et lemmatisation

### 3.2 Analyse descriptive

En analysant les données, une des premières remarques qu'on peut faire est le déséquilibre sur la variable *Exited* (20.4% de churn, et 79.6% de non-churn) ce qui n'est pas surprenant pour une analyse de la démission. On devra donc adapter les méthodes utilisées en tenant compte de ce déséquilibre de classes, par exemple en utilisant des poids de classes.

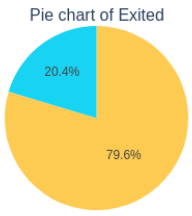


FIGURE 2

La visualisation du dataset complet (graphique complet en annexes) ne permet pas d'obtenir énormément d'informations intéressantes, si ce n'est la grande proportion de Français dans la base par rapport aux Allemands et aux Espagnols. En outre, on remarque que la variable Balance est souvent nulle : il s'agit en fait d'observations pour lesquelles la Balance n'a pas été remplie ; on peut considérer cela comme une valeur manquante, mais en réalité cela ne pénalise pas les résultats de nos modèles.

Le graphique des corrélations entre variables peut donner des premières pistes d'analyses. On remarque que les variables explicatives sont peu corrélées entre elles, sauf Balance avec Geography et NumOfProducts. En outre, la variable réponse est corrélée le plus fortement avec l'âge (0.29), mais aussi avec IsActiveMember (-0.16), Geography (0.15), Balance (0.12) et Gender (0.11). Ainsi, on pourrait déjà penser que les clients qui quittent la banque sont plus souvent des personnes âgées, peu actives au sein de la banque et qui ont une balance assez élevée.

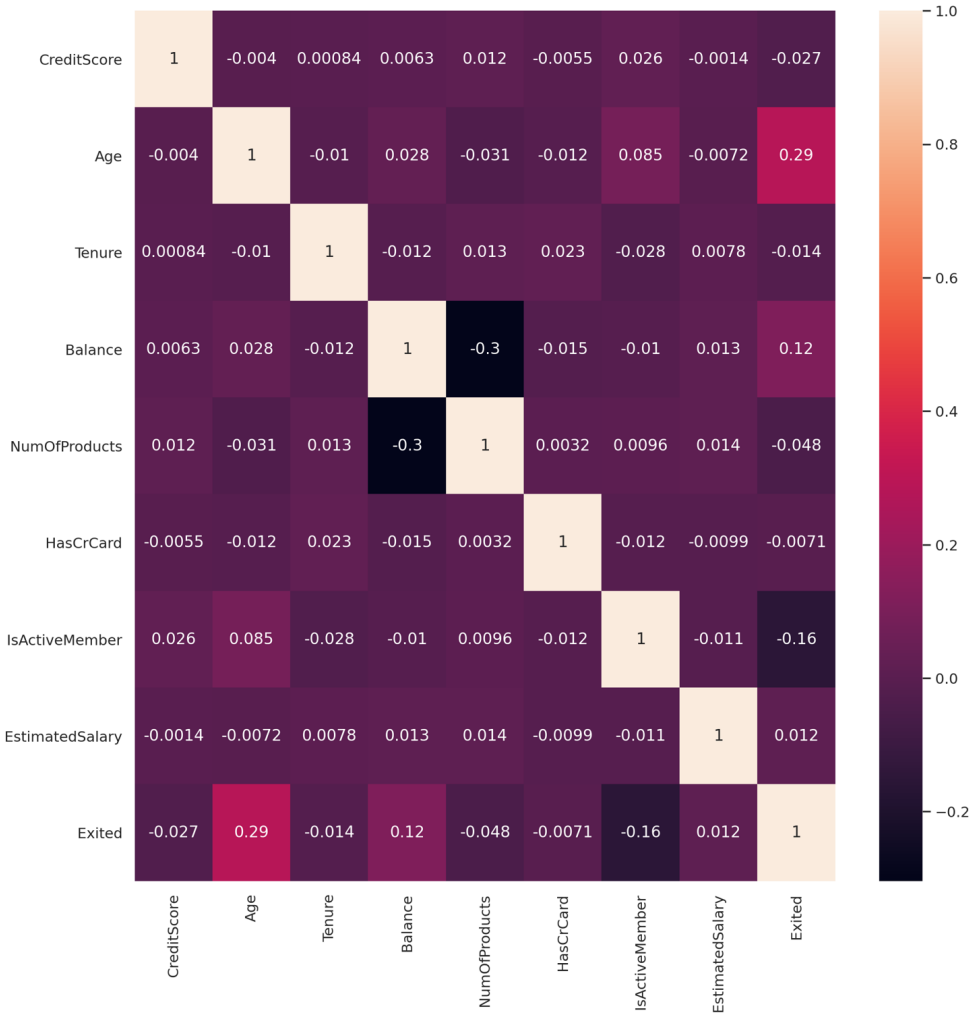


FIGURE 3 – Graphique de corrélations entre variables

Comparons maintenant les tableaux de données restreints aux populations respectivement churn et non-churn.

La comparaison des deux populations sur certaines variables qui nous intéressent confirme pour le moment les observations précédentes. On remarque aussi que les clients qui quittent la banque sont plus souvent âgés, Allemands, des femmes et des clients qui ont uniquement 1 produit ou alors 3 ou 4 produits, mais rarement 2 produits. Cependant, on ne remarque pas de différence de distribution notable pour la variable Balance.

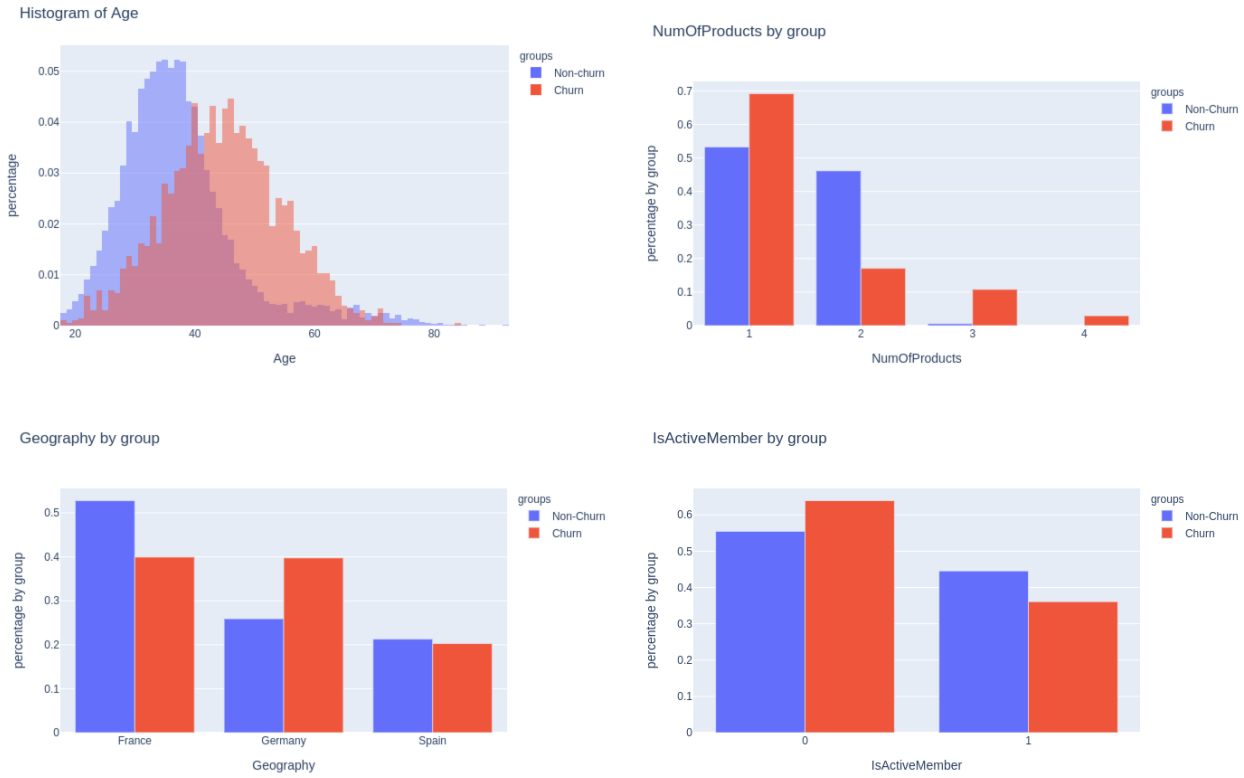


FIGURE 4 – Comparaison par groupe

En plus de l'analyse graphique, on effectue un test de Mann-Whitney pour confirmer la différence de distribution entre les deux groupes sur la variable Age. Pour l'hypothèse nulle, les deux groupes ont la même distribution alors que l'hypothèse alternative admet que les deux groupes n'ont pas la même distribution. Le résultat du test réalisé avec le package SciPy donne une *p-value* de $1.6 * 10^{-230}$ donc on peut bien rejeter l'hypothèse nulle au niveau 5%. On a donc un intérêt à comparer l'âge pour les deux groupes. Dans le cas des autres variables continues comme CreditScore et Balance, même si l'analyse graphique le suggérait déjà, le test donne une $p - value$ supérieure à 0.05 qui ne permet pas de rejeter l'hypothèse nulle, on peut donc raisonnablement considérer que leur intérêt est limité dans notre étude.

En analysant cette fois le churn rate pour chaque modalité de variable (ou par tranche pour les variables continue), on remarque que pour l'âge, on a une tranche particulièrement sensible à la démission : les 45-65 ans. Cela pourrait être expliqué par le fait que ce sont ceux qui ont le plus de pouvoir d'achat (beaucoup d'économies et bon salaire) et donc le plus d'intéret à chercher les contrats qui les avantagent ou encore à investir.

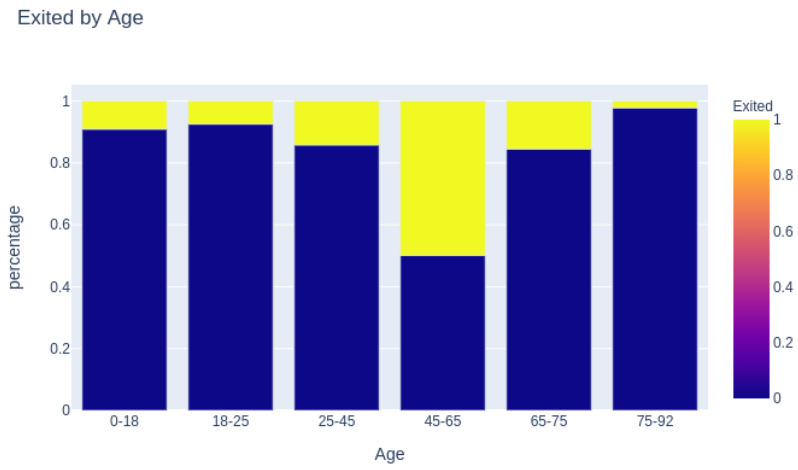


FIGURE 5 – Churn rate par tranche d'âge

### 3.3 Modélisation et résultats pour les données Kaggle

On va chercher à entraîner des modèles sur nos données préparées de la façon spécifiée précédemment (séparation 80-20 pour entraînement/test et cross-validation 5-fold en entraînement). D'abord, on entraînera et comparera 6 modèles :

- régression logistique
- RandomForest
- XGBoost
- CatBoost
- LightGBM
- réseau de neurones

Les hyperparamètres de chaque modèle sont optimisés grâce à un GridSearch qui permet de sélectionner le modèle avec les paramètres correspondant au modèle qui donne le meilleur résultat au regard du F1-score en validation.

Quelques précisions sur l'optimisation des modèles :

| Modèle | bibliothèque | paramètres optimisés |
|---|---|---|
| Régression logistique | sklearn.linear_model | C, type de pénalité |
| RandomForest | sklearn.ensemble | nombre d'arbres, profondeur maximale, nombre d'observations min pour diviser un noeud, nombre d'observations max par estimateur |
| XGBoost | xgboost | nombre d'estimateurs, profondeur maximale, learning rate |
| CatBoost | catboost | nombre d'estimateurs, profondeur maximale, learning rate |
| LightGBM | lightgbm | nombre d'estimateurs, nombre de feuilles, learning rate |
| Réseau de neurones | TensorFlow Keras | nombre de couches cachées, nombre de neurones par couche, learning rate, dropout rate |

De plus, on utilisera les paramètres class_weight='balanced' si celui-ci est disponible ou alors on attribuera des poids aux classes nous-même pour palier au déséquilibre entre nos deux classes dans le jeu de données.

Après l'entraînement de nos modèles, on obtient les résultats suivant sur données de test :

| Modèle | Accuracy | Recall | Precision | F1-score | AUC |
|---|---|---|---|---|---|
| RandomForest | 0.8335 | 0.6658 | 0.5791 | 0.6194 | 0.8496 |
| LightGBM | 0.8130 | 0.6757 | 0.5319 | 0.5952 | 0.8504 |
| XGBoost | 0.8615 | 0.4889 | 0.7425 | 0.5896 | 0.8551 |
| CatBoost | 0.7885 | 0.7322 | 0.4869 | 0.5849 | 0.8581 |
| Réseau de neurones | 0.7965 | 0.6732 | 0.5 | 0.5738 | 0.8356 |
| Régression logistique | 0.6990 | 0.6855 | 0.3705 | 0.4810 | 0.7572 |

TABLE 1 – Résultats sur données de test

Au regard des métriques qui nous intéressent (F1-score, recall) le RandomForest offre le meilleur F1-score, donc il s'agit du meilleur choix si on souhaite obtenir un bon compromis entre Recall et Precision. Toutefois, ce n'est pas le modèle offrant le meilleur résultat pour le Recall : en effet le CatBoost offre le meilleur recall, et également un F1-score relativement bon car sa precision est correcte. Ainsi selon notre objectif, on préférera RandomForest ou CatBoost. Les autres modèles ne présentent pas d'avantages par rapport à RandomForest et CatBoost : ils ont trop faible Recall ou une trop faible Precision. On remarque aussi la mauvaise performance de la régression logistique, ce qui suggère que les relations entre variables ne peuvent pas être exprimées simplement de manière linéaire.

Afin d'essayer d'obtenir de meilleurs résultats, i.e. un modèle encore plus performant, on peut essayer des méthodes d'ensemble learning : c'est-à-dire qu'on se sert des prédictions obtenues par les modèles présentés précédemment pour les combiner et en obtenir de nouvelles (qui sont supposées être encore meilleures). Les deux méthodes que nous allons utiliser sont le stacking et un modèle de moyenne (appelé blend par la suite). Pour le stacking on utilisera les prédictions obtenues par nos 6 modèles comme covariables pour l'entraînement d'un RandomForest. Ce RandomForest donnera à son tour des prédictions qui seront les prédictions finales pour notre classification binaire. En ce qui concerne le blend, on prendra simplement la moyenne des prédictions (en probabilité) obtenues par nos 6 modèles pour avoir la prédiction finale. De manière intuitive, le stacking sera moins efficace si les prédictions de nos modèles initiaux sont trop corrélées, ce qui est le cas (voir annexes).

| Modèle | Accuracy | Recall | Precision | F1-score | AUC |
|---|---|---|---|---|---|
| Blend | 0.8335 | 0.6536 | 0.5808 | 0.6150 | 0.8541 |
| Stacking | 0.8595 | 0.4963 | 0.7266 | 0.5898 | 0.8455 |

TABLE 2 – Résultats sur données de test

Comme on peut le remarquer, le stacking ne donne pas de résultats intéressants, ce qui était prévisible. Toutefois le Blend donne un bon résultat sur le F1-score qu'on cherche à optimiser. Cela n'est pas surprenant car le F1-score est une moyenne du Recall et de la Precision et le modèle Blend réalise une moyenne des modèles initiaux.

Pour finir, si on cherche le meilleur F1-score, on devrait choisir RandomForest ou le Blend. Toutefois, il est nécessaire pour le modèle Blend d'apprendre les modèles précédents, il s'agit donc d'une méthode assez coûteuse en temps et en ressources. Le RandomForest est donc plus adapté en général.

### 3.4 Feature importance

En complément de l'analyse descriptive, nous avons décidé de faire la variable importance afin d'affiner nos connaissances sur les données. Pour se faire nous avons en grande partie eu recours au package SHAP mais aussi à des options spécifiques de sklearn. La variable importance se fait sur un modèle déjà entraîné, nous avons donc utilisé une random forest avec les paramètres optimaux que nous avons trouvé avec gridsearch et le score qui nous intéresse est le F1-score.

**Permutation importance :** Nous nous sommes d'abord intéressés à la permutation importance de sklearn. Le principe de la permutation importance est, pour chaque variable, regarder la différence de performance entre les données telles quelles et les données où on permute entre elles les valeurs de la variables choisies.

| Weight | Feature |
|---|---|
| 0.1452 ± 0.0313 | Age |
| 0.1097 ± 0.0172 | NumOfProducts |
| 0.0295 ± 0.0146 | IsActiveMember |
| 0.0195 ± 0.0186 | Balance |
| 0.0127 ± 0.0122 | Germany |
| 0.0076 ± 0.0091 | Gender |
| 0.0001 ± 0.0082 | CreditScore |
| -0.0008 ± 0.0047 | EstimatedSalary |
| -0.0008 ± 0.0028 | Tenure |
| -0.0011 ± 0.0032 | Spain |
| -0.0025 ± 0.0049 | HasCrCard |
| -0.0058 ± 0.0089 | France |

FIGURE 6 – Permutation importance

Sur ces résultats nous pouvons remarquer que les variables les plus importantes sont l'âge, le nombre de produits, le fait d'être un membre actif, et ainsi de suite. Il est à noter le gap des poids entre les deux premières variables et le reste. En effet, mélanger la colonne de l'âge modifierait le F1-score de 0.14 alors que mélanger la colonne "IsActiveMember" modifierait le F1-score de 0.03 soit 5 presque 5 fois moins. De plus, nous pouvons observer des valeurs négatives. Ceci arrive quand la permutation des valeurs d'une variables a engendré un score meilleur. Nous pouvons en conclure que ces variables ne sont pas importantes mais l'aléatoire à fait que les prédictions étaient plus précises.

**Partial plot** Suite à ça, pour se faire une meilleure idée de l'impact des variables sur les prédictions nous avons regardé les partial plots de quelques variables importantes déterminées précédemment. Le principe est, pour une variable spécifique, de regarder les prédictions pour plusieurs valeurs prisent de la variable. Voici les résultats obtenus pour l'âge et le nombre de produits :

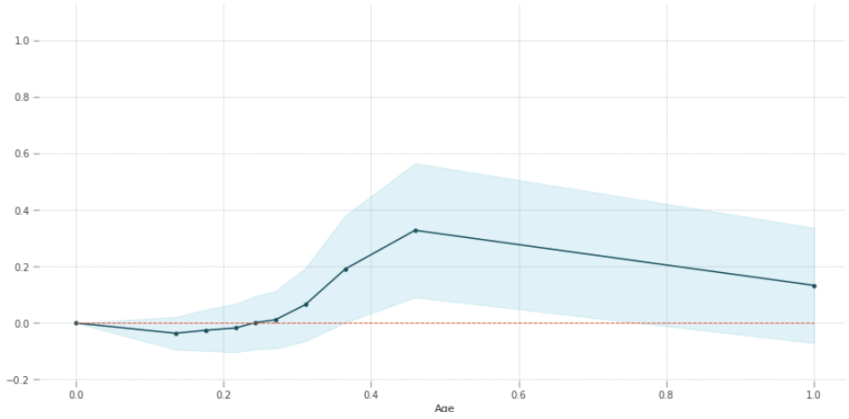


FIGURE 7 – Partial plot de l'âge

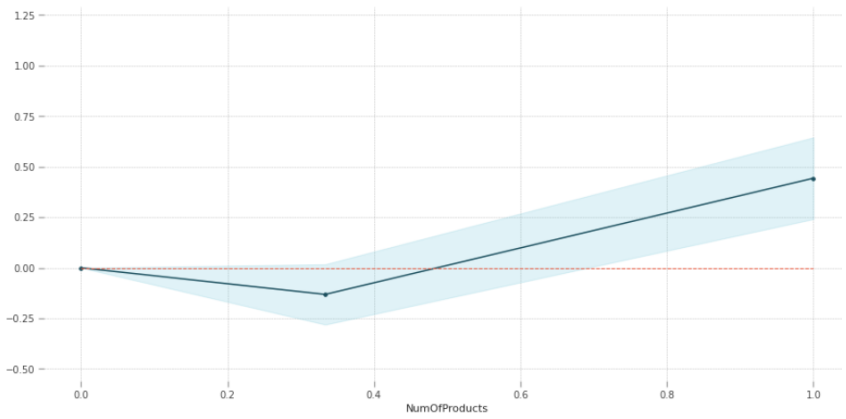


FIGURE 8 – Partial plot du nombre de produits

Vis-à-vis de l'âge nous avons un résultat intéressant qui rappelle une intuition que nous avions au vue de l'analyse descriptive. Il y aurait en effet une zone d'âge autour de laquelle les clients seraient plus susceptibles de quitter la banque. Concernant le nombre de produits, nous avons un résultat plutôt contre intuitif car il semble que plus un client est de produits plus il sera susceptible de quitter la banque.

**SHAP values** Enfin, pour terminer l'analyse de la variable importance nous nous sommes intéressés aux SHAP values. Tout d'abord un summary plot pour comprendre l'impact des variables dans leur globalité et ensuite nous nous intéresserons à la façon dont chaque variables influent sur les prédictions.

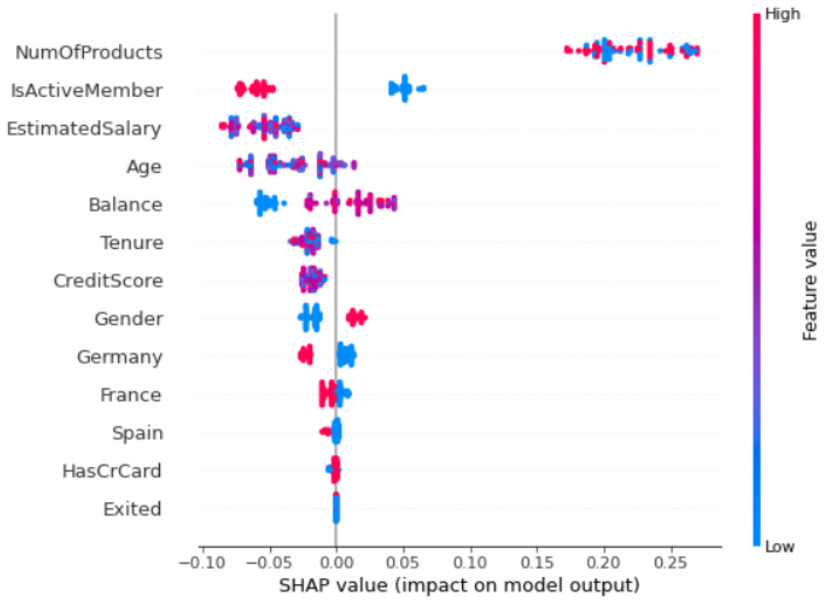


FIGURE 9 – SHAP summary plot

Sur ce summary plot, nous retrouver des résultats vus dans les paragraphes précédent mais aussi d'autre résultats intéressants. En effet on remarque que la variable "NumOfProducts", qui est la deuxième variable avec la plus importante, a systématique une prédiction plus élevée. De même, la variable "Age" qui est la variable la plus importante, ne semble pas avoir différent effet sur les prédiction selon la valeur prise par la variable.

### 3.5 Feature engineering

Après avoir accumulé des connaissances sur les données nous avons décidé de faire du feature engineering pour obtenir de meilleurs score. Tout d'abord nous avons décidé de donner un effet négatif à certaines variables binaires. Nous avons remplacé, pour les variables "IsActiveMember" et "HasCrCard", les 0 en -1. Ensuite nous avons créé de nouvelles variables : "BalanceSalaryRatio" qui est le rapport entre la "Balance" et "EstimatedSalary", "CreditScoreBalanceRatio" qui est le rapport entre la "CreditScore" et "Balance" et "TenureByAge" qui est le rapport entre la "Tenure" et "Age".

Après avoir modifié et crée quelques variables nous avons fait quelques expériences. Au début nous avons juste utilisé les modifications apportées aux variables "IsActiveMember" et "HasCrCard", puis nous avons ajouté au fur et à mesure les nouvelles variables.
Malheureusement, nous n'avons jamais réussi à améliorer notre meilleure score, RandomForest avec le dataset de base, mais nous avons quand même eux quelques résultats intéressant. Les modèles "simples" comme LinearRegression et RandomForest ont toujours eux des résultats plus faibles avec les modifications des variables mais certains modèles plus "complexes", notamment les modèles de boosting, ont eux eu de meilleurs résultats qu'auparavant mais aucun d'eux n'a dépassé les résultats de la RandomForest.

### 3.6 Résultats pour les données textuelles

L'analyse churn pour données textuelles repose sur le même principe que l'analyse pour données numériques (recherche des meilleurs paramètres des modèles, prédiction par validation croisée et prédiction sur données test).

Les modèles utilisés sont l'algorithme Naive Bayes, régression logistique et SVM avec un pré-traitement du texte et la matrice TF-IDF comme entrée. Nous avons aussi utilisé le modèle FastText pré-entraîné sur le corpus Wikipédia français avec un ajout d'une couche Attention et ainsi que le modèle CamemBERT. Ces deux derniers modèles n'ont pas besoin d'un pré-traitement car ils ont été entraîné sur des données non pré-traitées. Enfin, nous avons créé BlendModel prenant la moyenne des sorties de tous les modèles.

| Modèle | accuracy | rappel | précision | F1-score | AUC |
|---|---|---|---|---|---|
| CamemBERT | 0.9327 | 0.9335 | 0.9538 | 0.9435 | 0.9325 |
| BlendModel | 0.9269 | 0.9239 | 0.9533 | 0.9384 | 0.9645 |
| tf-idf+Naive Bayes (POS-tag + lem) | 0.9252 | 0.9219 | 0.9524 | 0.9369 | 0.9261 |
| tf-idf+Naive Bayes | 0.9250 | 0.9215 | 0.9524 | 0.9367 | 0.9259 |
| tf-idf+Naive Bayes (POS-tag) | 0.9235 | 0.9203 | 0.9512 | 0.9355 | 0.9244 |
| tf-idf+Logistic Regression (POS-tag) | 0.9199 | 0.9155 | 0.9497 | 0.9323 | 0.9211 |
| tf-idf+SVM | 0.9192 | 0.9167 | 0.9474 | 0.9318 | 0.9198 |
| tf-idf+SVM (POS-tag) | 0.9190 | 0.9151 | 0.9485 | 0.9315 | 0.9200 |
| tf-idf+Logistic Regression | 0.9170 | 0.9191 | 0.9417 | 0.9303 | 0.9165 |
| tf-idf+SVM (POS-tag + lem) | 0.9146 | 0.9111 | 0.9452 | 0.9278 | 0.9155 |
| tf-idf+Logistic Regression (POS-tag + lem) | 0.9129 | 0.9155 | 0.9384 | 0.9268 | 0.9123 |
| Fasttext + couche Attention | 0.9103 | 0.9011 | 0.9474 | 0.9236 | 0.9126 |

TABLE 3 – Résultats par validation croisée (classé par F1-score)
Indication : Pour les modèles utilisant la matrice TF-IDF, la ponctuation et majuscules ont été supprimées, 'POS-tag' indique l'application du pré-traitement POS-tagging en gardant les verbes, noms communs adjectifs et déterminants et 'lem' l'application de la lemmatisation.

D'après Table 1, le modèle CamemBERT obtient le meilleur score de rappel (0.9335) et de précision (0.9538) lors de la validation croisée. Nous choisissons ce modèle pour l'analyse de commentaires étant donné que le F1-score (0.9435) est supérieur de 0.005 du second modèle. La performance du modèle se confirme lors de la prédiction sur les données test. On note une baisse du f1 score (0.008 pour CamemBERT) mais ceci est aussi

observé pour les autres modèles (baisse moyenne de 0.01).

On peut noter que le modèle complexe FastText + couche Attention obtient un score moins intéressant que les modèles utilisant la méthode de bag-of-words pour les résultats par validation croisée et test. Aussi les performances liés au pré-traitements sont très variées, aucune méthode de pré-traitement performe mieux que les autres.

Pour une banque, l'intérêt des commentaires est de connaître l'opinion des clients sur des sujets précis et les causes de départs des clients. Nous pouvons utiliser les modèles précédents et par exemple étudier l'opinion des clients sur le 'service client' pour la banque OrangeBank, en considérant qu'on ne connaît pas le nombre d'étoiles pour chaque commentaire.

Une première méthode pour extraire les commentaires s'exprimant sur le service client est calculer la similarité cosinus entre le vecteur du terme 'service client' et les vecteurs des commentaires. Nous avons choisis de calculer les vecteurs avec le modèle FastText et d'extraire les commentaires avec une similarité avec le terme 'service client' supérieure à 0.4 (limite choisie manuellement en remarquant que les documents avec une similarité inférieure à 0.4 n'ont pas de lien avec 'service client'). Nous obtenons 82 commentaires et en appliquant le modèle CamemBERT, nous constatons que 66% des commentaires ont un sentiments négatifs.

Une autre méthode est d'utiliser un modèle d'extraction de thèmes et de choisir un thème avec des top termes en lien avec 'service client'. Nous avons utilisé la méthode de décomposition NMF avec la matrice TF-IDF comme variable d'entrée et nous avons sélectionné un thème avec pour top termes : service client - opérateur client - réponse service. Nous obtenons 97 commentaires et en appliquant le modèle CamemBERT, nous constatons que 81% des commentaires ont un sentiments négatifs.

Un autre avantage du modèle CamemBERT est de pouvoir extraire les mots qui ont un poids influent dans la dernière couche Attention du modèle :

termes avec les poids les plus élevés du commentaire
top termes du thème sélectionné

Positif : Très satisfait quelques problèmes mais réglé rapidement avec un service client au top

Négatif : Service client injoignable et aucun retour de ma demande de renouvellement de carte défectueuse depuis plusieurs mois , pas top comme service alors que je paie 7'99 / mois pour une carte Prémium qui est défectueuse.

Cette analyse non supervisée nous permet de conclure que plus de 50% des commentaires concernant le service client exprime un sentiment négatif. L'extraction des mots influents nous permettent de visualiser plus rapidement les commentaires et de faire un lien avec l'utilisation de l'importance des variables pour données numériques puisqu'on retrouve l'objectif de comprendre l'impact des variables sur le modèle.

# 4 Application de présentation des résultats

Dans le but de rendre notre travail exploitable d'un point de vue professionnel, nous avons décidé de réaliser un dashboard pour présenter nos résultats. L'objectif est de pourvoir présenter de manière interactive nos résultats (plusieurs onglets, possibilité de sélectionner des options, etc.).

Pour cela nous avons choisi le framework Streamlit pour Python qui a pas mal de succès au sein de la communauté machine learning. Il a pour principal avantage d'être très simple d'utilisation comparé à R-shiny ou Dash et compatible avec les bibliothèques de data science communément utilisées (comme Plotly par exemple).

L'utilisation du dashboard est simple : on exécute au préalable le code permettant de calculer les modèles et d'enregistrer les résultats sous forme de CSV et d'image, puis on lance l'application streamlit qu'on a écrite et qui interprète ces résultats pour les afficher. On divisera l'application en 3 onglets pour les données issues de Kaggle : un onglet traitant de l'EDA, un autre des scores des modèles et un dernier sur la feature importance. Le code du dashboard est facilement adaptable à d'autres résultats d'étude présentés sous le même formats.

De plus, chaque partie a été construite de telle manière qu'il est possible d'avoir la main sur les résultats que nous voulons afficher. Pour l'EDA il est possible d'afficher des résultats pour une variable en particulier et pour la feature importance, il est possible de choisir le modèle sur lequel travailler ainsi que les variables d'intérêt.

Le code permettant de calculer les résultats et d'exécuter l'application Streamlit est disponible sur le dépôt Github suivant : `https://github.com/Tarandro/BinaryML_churn`

# 5 Conclusion

Au cours de notre étude, l'analyse descriptive des données ainsi que la modélisation à l'aide de modèles de machine learning nous ont permis d'atteindre un double objectif : déterminer certaines causes de la démission au sein de la population concernée par ces données ainsi que prédire de manière binaire si un client va quitter la banque. Sur notre jeu de données issues de Kaggle, l'analyse a montré l'influence de certaines covariables sur la variable réponse : l'âge, le nombre de produits achetés et l'activité notamment. En effet, les clients entre 45 et 65 ans sont ceux qui quittent le plus la banque, alors que les jeunes ou les personnes très âgées restent plus souvent. Cela peut s'expliquer par le fait que la tranche d'âge 45-65ans est active et a un haut pouvoir d'âchat. De plus les clients qui ont 2 produits quittent moins souvent que ceux qui ont 1 produit ou plus de 2. Les membres actifs ont aussi tendance à rester.

La modélisation de ces données à l'aide de différents modèles nous a permis de dégager un modèle de type RandomForest pour prédire la variable réponse. En effet, ce modèle est celui qui a obtenu le meilleur F1-score à l'issue de la comparaison. Cette métrique combine les avantages de la précision et du rappel : le choix du modèle a donc pris en compte les faux positifs et les faux négatifs. Le RandomForest a aussi pour avantage d'être un modèle relativement simple puisque c'est un modèle d'arbre, rapide à calculer et qui limite le sur-apprentissage.

Une analyse de feature importance permet en outre d'avoir une certaine idée de l'importance des variables dans la construction de modèles de machine learning ayant un effet "boîte noire". Cette analyse sur nos modèles a confirmé les hypothèses émises lors de l'exploration des données : l'âge, le nombre de produit et l'activité du client sont les variables qui ont le plus d'impact sur les modèles.

Pour finir, notre démarche nous a donc permis de construire un modèle prédictif permettant de déterminer si un client va quitter ou non la banque : un RandomForest avec des paramètres optimisés. Nous avons aussi pu déterminer les principaux facteurs expliquant ce départ au sein de la clientèle.

En ce qui concerne l'analyse de données textuelles, nous avons suivi une démarche de machine learning similaire, adaptée au NLP, pour déterminer si le sentiment dégagé par un commentaire de client est positif ou négatif. Le modèle CamemBERT est le plus efficace au sens du F1-score dans notre cas.

Si l'analyse et les modèles construits ne peuvent pas directement être appliqués à d'autres données sans adaptation, le principe de notre démarche est reproductible pour traiter de problèmes de classification binaire similaires.

## Bibliographie


[1] Abdelrahim Kasem Ahmad, Assef Jafar, and Kadan Aljoumaa. Customer churn prediction in telecom using machine learning and social network analysis in big data platform. *CoRR*, abs/1904.00690, 2019.

[2] Asia Mahdi Naser Alzubaidi and Eman Salih Al-Shamery. Projection pursuit random forest using discriminant feature analysis model for churners prediction in telecom industry, 2019.

[3] Adnan Amin, Feras Al-Obeidat, Babar Shah, Awais Adnan, Jonathan Loo, and Sajid Anwar. Customer churn prediction in telecommunication industry using data certainty. *Journal of Business Research*, 94 :290–301, 2019.

[4] XGBoost developpers. Xgboost documentation. `https://xgboost.readthedocs.io/en/latest/`.

[5] Nabgha Hashmi, Naveed Anwer Butt, and M. Iqbal. Customer churn prediction in telecommunication a decade review and classification. 2013.

[6] Microsoft LightGBM. Lightgbm documentation. `https://lightgbm.readthedocs.io/en/latest/`.

[7] The CatBoost Team. Catboost documentation. `https://catboost.ai/docs`.

# Annexes

FIGURE 10 – Analyse des variables du dataset d'origine

FIGURE 11 – Corrélation entre les prédictions des modèles

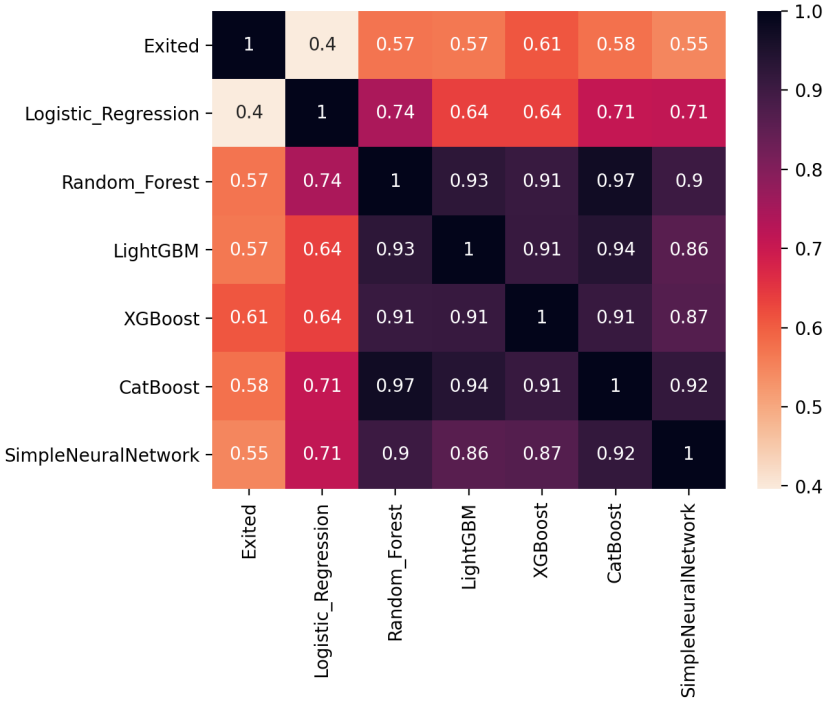
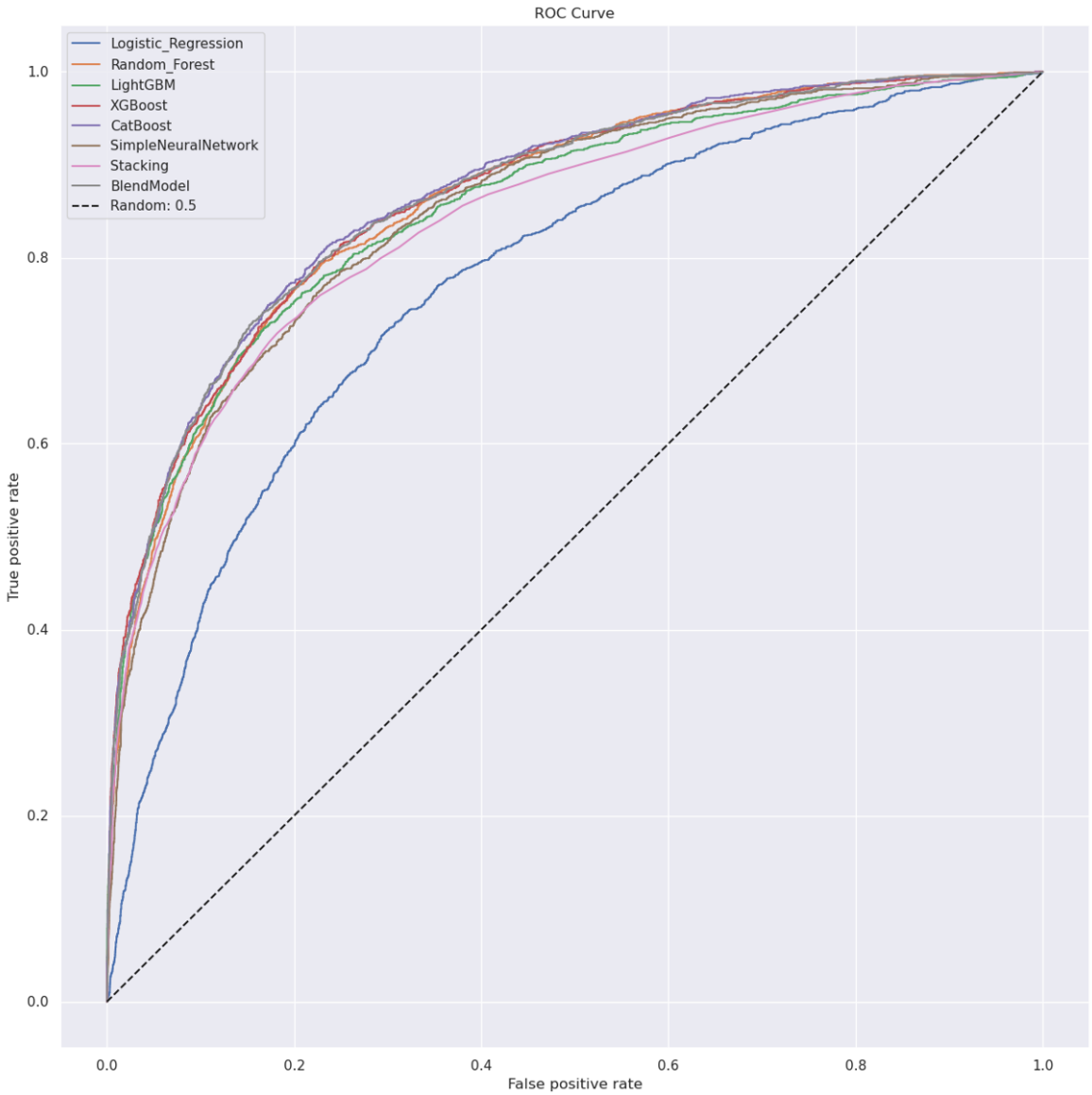
FIGURE 12 – Courbes ROC des modèles