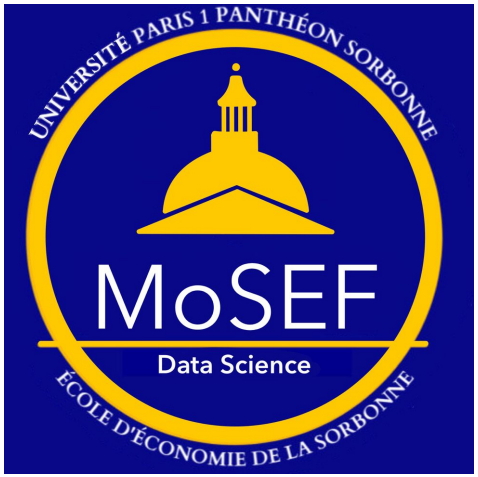
MOSEF Bank Churn Prediction

Data mining

Author

Sharon Chemma
Lina Benzemma



Introduction

1. Challenge Kaggle
2. Visualisation des données
3. Preprocessing
4. Choix du modèle et optimisation

Challenge Kaggle

Introduction

Contexte :

- Les banques perdent des revenus lorsque des clients quittent leurs services.
- Identifier les clients à risque est crucial pour agir en amont.

Objectif du projet :

- Construire un modèle prédictif robuste pour détecter les clients susceptibles de quitter.
- Maximiser la performance en atteignant une **AUC optimale**.

Impact attendu : Anticiper les départs et optimiser les stratégies de fidélisation client.

kaggle

Visualisation des données

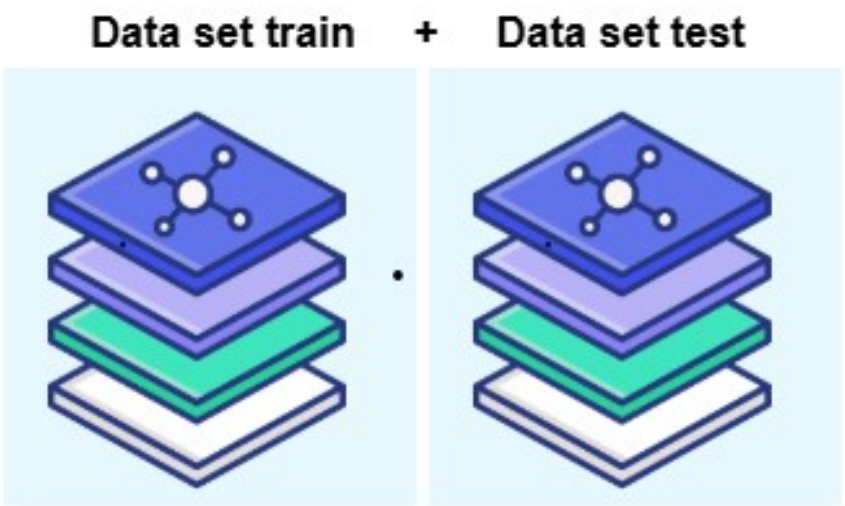
Présentation des Datasets

Dimensions des Datasets :

- **Train DATA SET :** 15,000 lignes, 14 colonnes
- **Test DATA SET :** 10,000 lignes, 13 colonnes

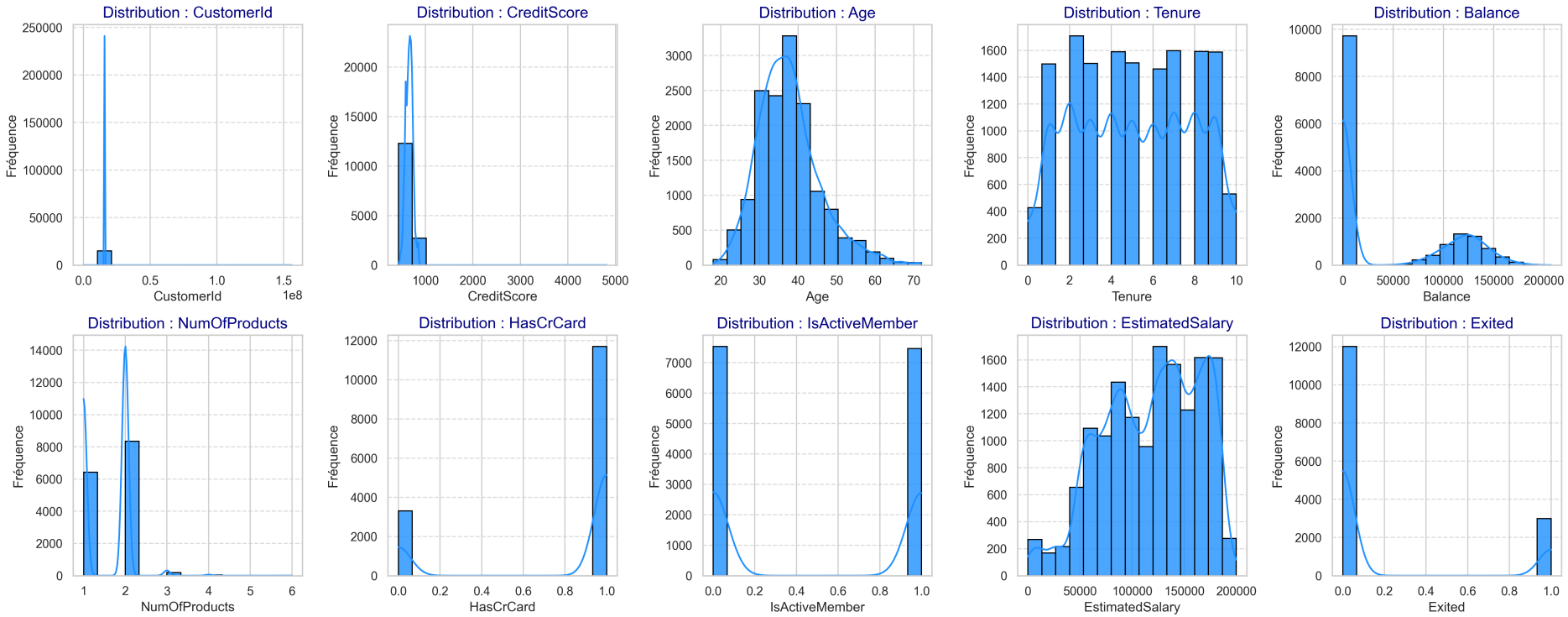
Types de Variables :

- **Catégoriques :** Geography, Gender, Surname
- **Numériques :** CreditScore, Age, Tenure, Balance, NumOfProducts, etc.
- **Variable cible :** Exited (1 = Churn, 0 = Non-Churn)



Distributions des Variables Numériques

- Les distributions montrent une variabilité significative entre les variables.
- La variable `Balance` montre une concentration élevée autour de 0 avec quelques valeurs extrêmes.
- La variable cible `Exited` est déséquilibrée : les clients qui ne quittent pas la banque sont largement dominants.



Preprocessing

Préparation des Données

Étapes de Prétraitement :

- **Suppression des colonnes inutiles :**
  - Les colonnes `id`, `CustomerId` et `Surname` sont supprimées car elles n'apportent pas d'information pertinente pour la prédiction.
- **Encodage des variables catégoriques :**
  - Les colonnes `Geography` et `Gender` sont encodées avec la méthode **One-Hot Encoding**.
  - L'option `drop_first=True` a été utilisée pour éviter la colinéarité (suppression de la première modalité de chaque catégorie).
- **Pas de valeurs manquantes.**

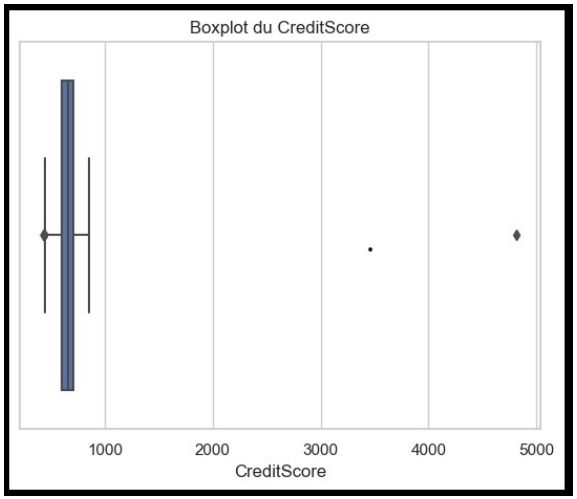
Analyse des Outliers dans `CreditScore`

Étapes pour identifier les outliers :

- Calcul des bornes inférieure et supérieure en utilisant l'**IQR** (Interquartile Range).
- Identification des valeurs en dehors des bornes : 16
- Ajout d'une variable binaire pour capturer les outliers.

Proportions de churn :

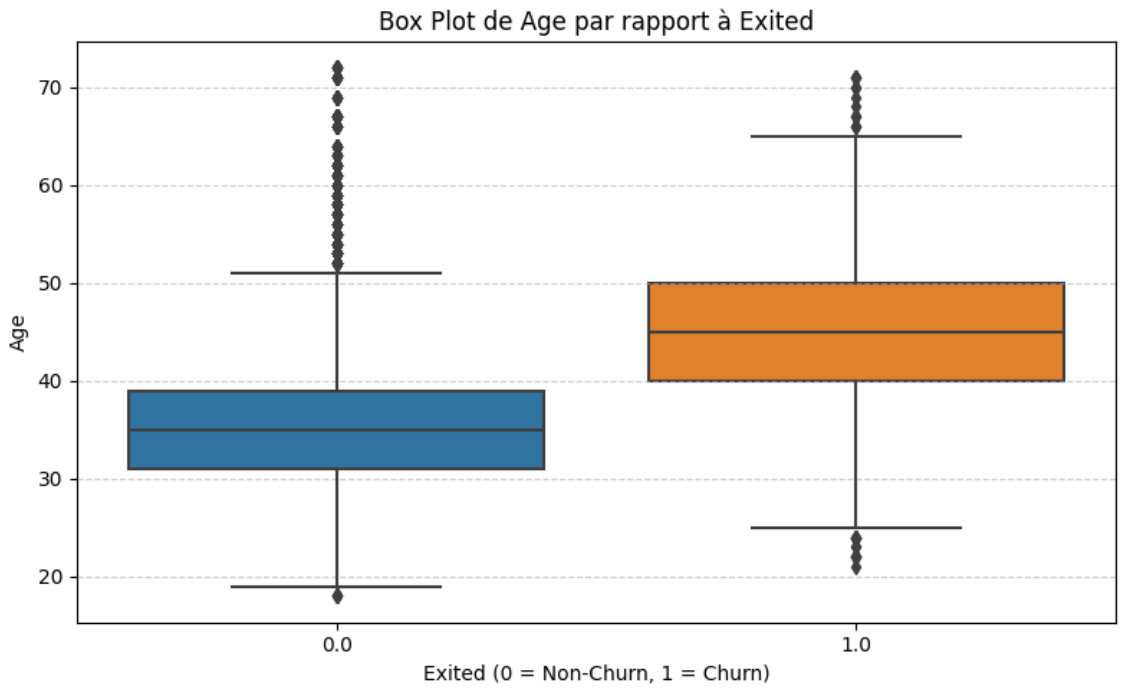
- **Pour les outliers et les non-outliers :**
  - 62.5% contre 80.08% des clients n'ont pas churné.
  - 37.5% contre 19.91% des clients ont churné.



Analyse des Variables Numériques

Observations clés :

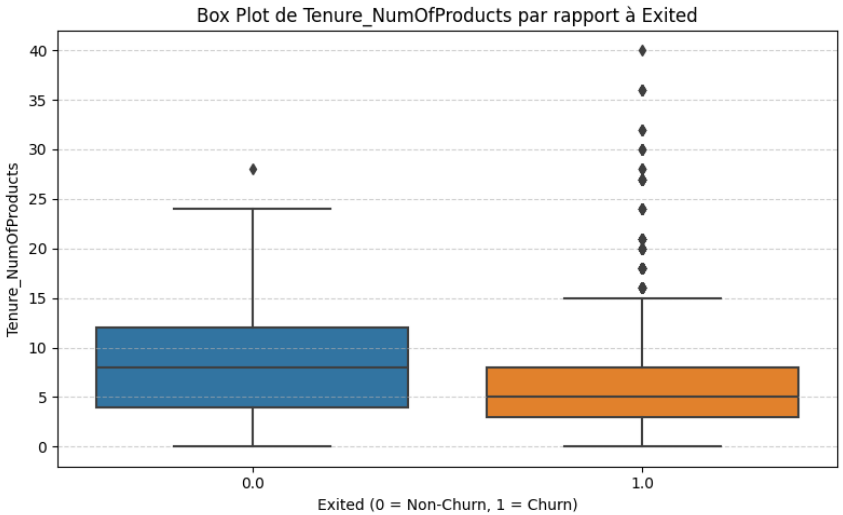
- Les clients churnants ont des caractéristiques distinctes pour certaines variables.
- Ces distributions offrent des pistes pour créer de nouvelles variables afin de mieux segmenter les clients.

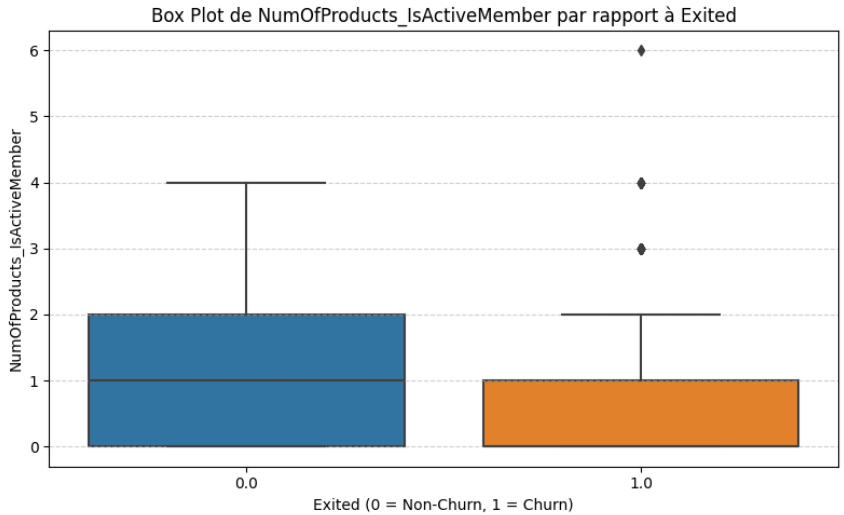


Analyse des Variables Numériques

Explorations supplémentaires :

- `Tenure_NumOfProducts` :
  - Les clients churnants ont souvent des combinaisons spécifiques d'ancienneté (`Tenure`) et de produits détenus.
- `NumOfProducts_IsActiveMember` :
  - Les clients churnants ont généralement un engagement moindre (`IsActiveMember`) malgré un certain nombre de produits.





Transformations et Création de Variables

- **Logarithmique Transformation :**
  - La variable `Balance` a été transformée en `Log_Balance` avec la fonction $\log(1 + x)$.
- **Création de Groupes d'Âge :**
  - La variable `Age` a été catégorisée en groupes :
    - Groupes définis par les intervalles : [0-25), [25-40), [40-60), [60-100).
    - Étiquettes associées : 1, 2, 3, 4.
  - La variable `Age` a été supprimée après transformation.

Observations :

- **Logarithmique Transformation (`Log_Balance`) :**
  - La transformation de `Balance` en `Log_Balance` a amélioré l'AUC des modèles.
  - Cette transformation a permis de réduire l'effet des valeurs extrêmes dans `Balance`.
- **Binning de l'Âge (`Age_Group`) :**
  - La catégorisation de l'âge en groupes n'a pas eu d'impact significatif sur l'AUC.

Création de Variables Personnalisées

Objectif :

- Capturer des comportements spécifiques ou atypiques des clients pour améliorer la performance des modèles.
- Identifier des groupes de clients susceptibles de churner.

Nouvelles Variables :

- `Inactive_high_balance` :
  - Identifie les clients inactifs (`IsActiveMember` $= 0$) avec un solde élevé (`Balance` $> 100{,}000$).
- `Low_balance_high_products` :
  - Identifie les clients ayant un solde faible (`Balance` $< 5{,}000$) mais utilisant de nombreux produits (`NumOfProducts` > 2).
- `Senior_low_balance` :
  - Identifie les clients seniors (`Age` > 60) ayant un solde faible (`Balance` $< 20{,}000$).
- `High_salary_low_products` :
  - Identifie les clients ayant un salaire élevé (`EstimatedSalary` $> 150{,}000$) mais utilisant peu de produits (`NumOfProducts` $= 1$).

Création de Variables Personnalisées

1. `Balance / EstimatedSalary` :

- Représente le rapport entre le solde bancaire et le salaire estimé d'un client.
- Permet de détecter des comportements financiers atypiques :
  - Un ratio élevé (`> 1`) indique une richesse relative importante par rapport aux revenus.
  - Un ratio faible peut indiquer une faible accumulation par rapport au salaire.

2. `Balance_per_Product_Age` :

- Normalise le solde en tenant compte de la diversification (`NumOfProducts`) et de l'âge.
- Met en évidence des comportements spécifiques :
  - Un ratio faible peut indiquer des clients engagés sur plusieurs produits.
  - Un ratio élevé peut refléter une faible diversification financière.

Choix du modèle et optimisation

Optimisation des Hyperparamètres : Partie 1

Objectif : Ajuster les hyperparamètres pour améliorer la performance du modèle en capturant efficacement les comportements des clients à risque.

Hyperparamètres clés :

- **Nombre d'itérations (`iterations`) :**
  - Définit le nombre total d'arbres générés par le modèle.
  - Augmenté à **1000** pour permettre un entraînement plus approfondi et capturer des interactions complexes.
- **Taux d'apprentissage (`learning_rate`) :**
  - Contrôle la vitesse à laquelle le modèle ajuste les poids à chaque itération.
  - Réduit à **0.03** pour stabiliser la convergence et éviter les oscillations.
- **Profondeur des arbres (`depth`) :**
  - Détermine la profondeur maximale des arbres (c'est-à-dire le nombre de niveaux).
  - Fixée à **6** pour trouver un équilibre entre la capacité d'apprentissage et le risque de surapprentissage.

Comparaison des Modèles

Méthodologie :

- Validation croisée stratifiée (`StratifiedKFold`) avec 5 plis pour assurer une répartition équilibrée de la variable cible dans les jeux d'entraînement et de validation.
- Comparaison basée sur la courbe ROC moyenne et l'AUC moyenne.

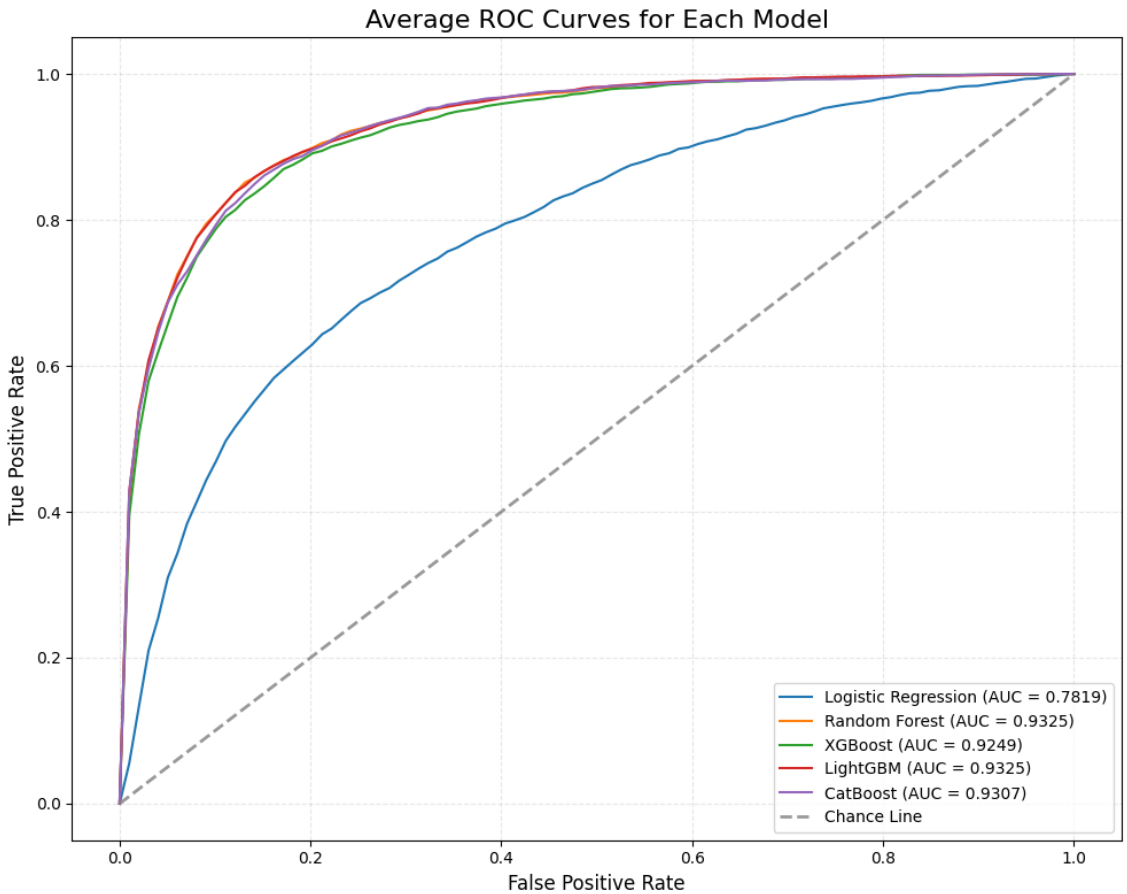
Modèles testés :

- Régression Logistique
- Random Forest
- XGBoost
- LightGBM
- CatBoost

Résultats des Modèles

Courbes ROC Moyennes :

- Chaque modèle a été évalué pour sa capacité à distinguer les classes (churn/non-churn).



Moyenne des AUC par Modèle

Scores AUC moyens :

- Les AUC ont été calculés pour chaque pli, puis moyennés.

Modèle	**AUC Moyenne**
Régression Logistique	0.7819
Random Forest	0.9325
XGBoost	0.9249
LightGBM	0.9325
CatBoost	0.9307

Observations :

- RandomForest et LightGBM sont légèrement meilleur, mais les différences avec CatBoost et XGBoost sont marginales.
- Ces modèles seront optimisés davantage pour maximiser leurs performances.

Optimisation des Hyperparamètres : Partie 2

Objectif : Améliorer la robustesse et l'équilibre du modèle grâce à des réglages précis.

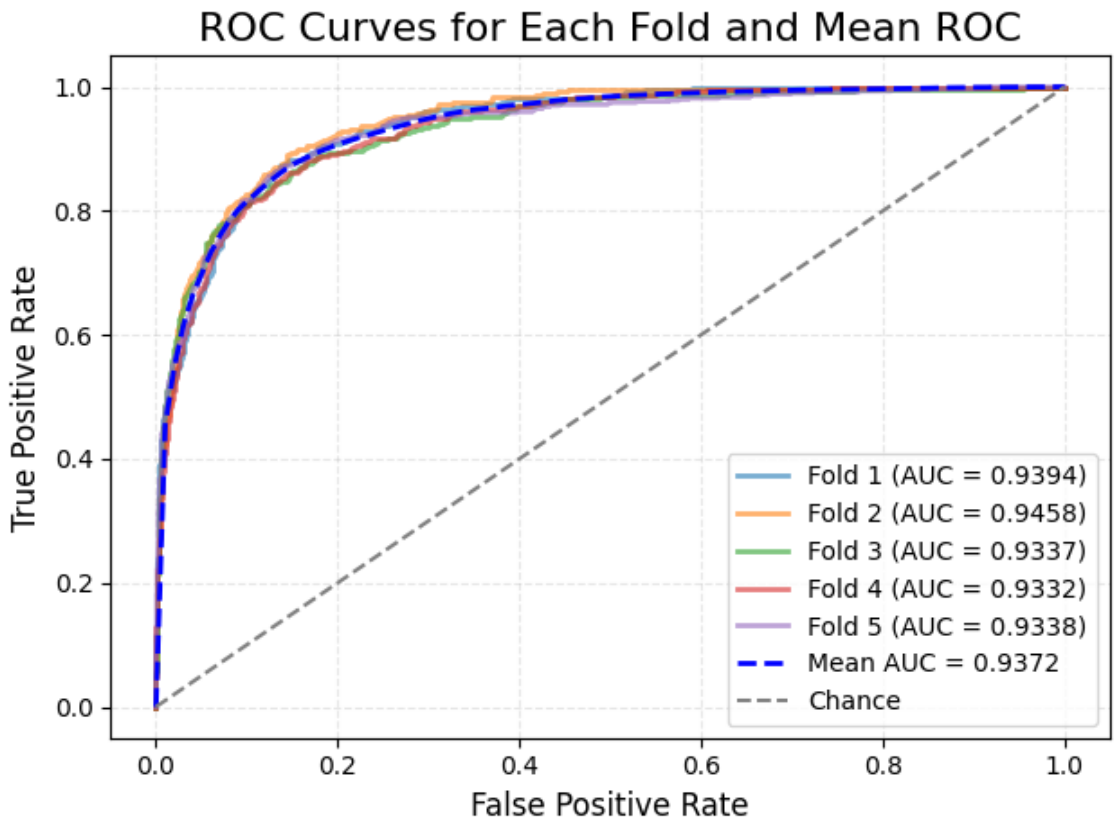
Hyperparamètres avancés :

- **Régularisation L2 (`l2_leaf_reg`) :**
  - Ajoute une pénalité pour les arbres trop complexes afin de prévenir le surapprentissage.
  - Fixée à **3** pour équilibrer la complexité du modèle.
- **Poids des classes (`class_weights`) :**
  - Compense le déséquilibre des classes en accordant plus d'importance à la classe minoritaire.
  - Configuré à [**1, 2.5**] pour renforcer l'apprentissage des clients churnants (`Exited = 1`).
- **Fonction de coût (`loss_function`) :**
  - Utilise `Logloss` pour maximiser la séparation entre les classes.
  - Essentielle pour les tâches de classification binaire.
- **Métrique d'évaluation (`eval_metric`) :**
  - Optimise directement l'AUC, une métrique clé pour les problèmes de classification déséquilibrés.

Impact

Impact :

- Un modèle plus robuste, capable de mieux gérer le déséquilibre des classes.
- Une réduction du surapprentissage grâce à un réglage plus précis des hyperparamètres.



Stacking : Combinaison de Modèles

Objectif : Améliorer les performances en combinant plusieurs modèles de base via un méta-modèle.

Modèles impliqués :

- **Modèles de base :**
  - `CatBoost` : Performant sur les données catégoriques.
  - `XGBoost` : Modèle robuste et bien adapté aux grandes données.
  - `LightGBM` : Rapide et efficace sur les grandes bases déséquilibrées.
- **Méta-modèle :**
  - `XGBoost` avec des hyperparamètres optimisés (taux d'apprentissage à 0.05, profondeur 3).

Stacking : Combinaison de Modèles

Résultats :

- AUC moyenne obtenue : **0.9332**, légèrement inférieure aux modèles individuels optimisés.

Impact :

- Le stacking n'a pas amélioré les performances globales.
- Les modèles individuels, comme `CatBoost`, restent plus performants dans ce contexte.