

Engineering Degree in Telecommunications

End of Studies Project Report

AI Assisted platform for Third-Party Risk Management

Host Company:

Realized by:

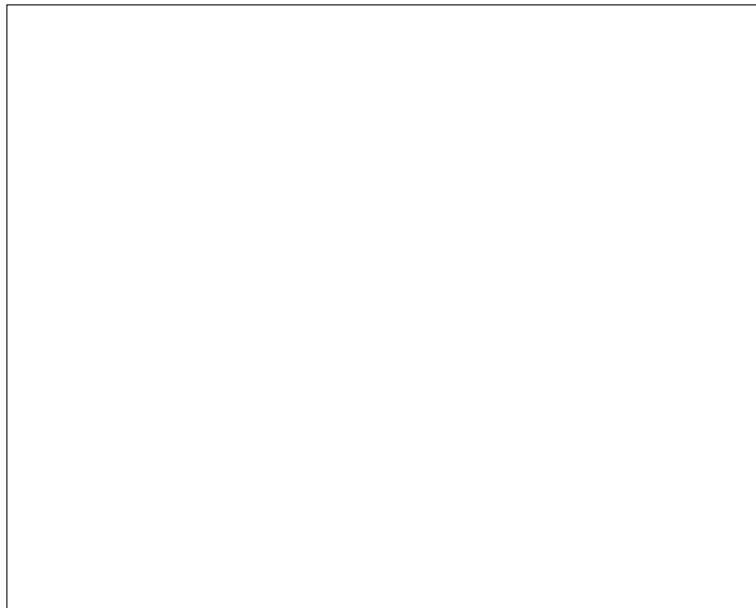
Mohamed Mechichi

Supervised by:

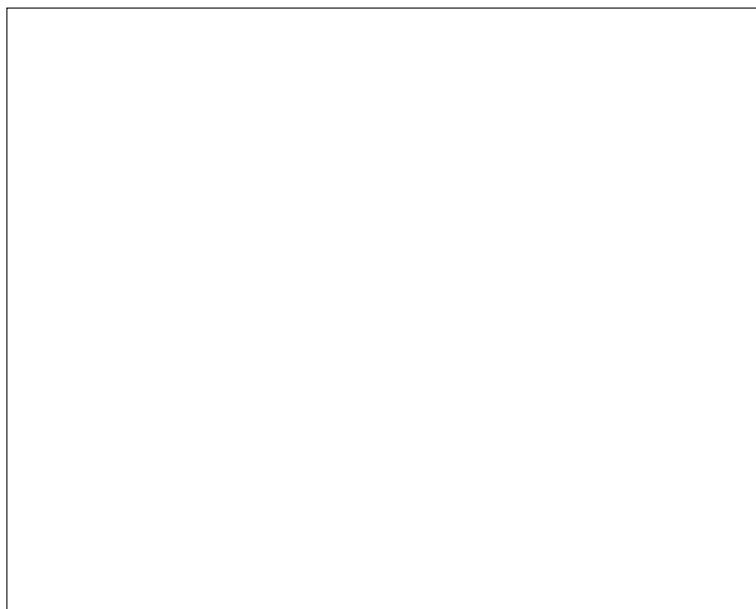
**Dr. Slim Rekhis,
Dr. Magda Lilia Chelly**

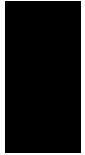
2023 - 2024

University's gdf Supervisor Stamp and Signature



Company Supervisor Stamp and Signature





Dédicace

À toutes les personnes qui ont contribué à la réalisation de ce travail et m'ont soutenue tout au long de ce parcours, je dédie ce travail.

À mon père, Helmi Mejri, je dédie une spéciale dédicace pour son amour infini et son soutien inconditionnel, qui m'ont accompagnée à chaque étape de ma vie.

À ma mère, Nadia Annabi, je suis profondément reconnaissante pour sa tendresse, sa patience et ses encouragements constants.

À mon frère, Ahmed Omar Mejri, je dédie cette dédicace pour avoir toujours été à mes côtés, me guidant et m'aidant à faire les meilleurs choix, tant sur le plan personnel que professionnel.

À mes cousins, Safa Harran et Malek Harran, je dédie ce travail en reconnaissance de leur affection et de leur soutien bienveillant tout au long de mon parcours.

À mes chers amis, Molka Baatout, Houssem Zemni, Mohamed Mechichi, Nada Zid et à tous mes autres amis, je dédie cette réalisation pour leur amitié, leur encouragement et leur présence précieuse tout au long de cette aventure.

À vous tous,
je dédie ce travail.

Remerciements

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont contribué à la réussite de ce projet.

Tout d'abord, mes plus sincères remerciements à mon encadrant professionnel, Marzouki Zouheir, pour sa disponibilité, sa bienveillance et son aide précieuse tout au long de ce projet. J'ai énormément appris à ses côtés, tant sur le plan professionnel que personnel. Un grand merci également à Laamouri Anis pour son professionnalisme exemplaire et ses précieux conseils qui ont enrichi mon travail.

Je tiens également à remercier mon encadrant académique, Kasmi Sofiene, pour son soutien constant, ses conseils avisés et son accompagnement tout au long de ce parcours.

Je n'oublie pas de remercier l'École Nationale Supérieure d'Ingénieurs de Tunis ainsi que l'ensemble de mes enseignants, qui m'ont fourni les connaissances et les compétences nécessaires pour mener à bien ce projet.

Merci à tous pour votre aide, votre soutien et vos encouragements.

Table de matières

Liste des abréviations	ix
Liste des figures	xiii
Liste des tables	xiii
INTRODUCTION GÉNÉRALE	1
1 Présentation générale du projet	3
1.1 Introduction	4
1.2 Présentation du lieu du stage	4
1.2.1 Présentation de l'entreprise	4
1.2.2 Les Valeurs de l'entreprise	4
1.2.3 La vision de l'entreprise	6
1.2.4 L'organigramme de l'entreprise	6
1.2.5 L'organigramme de la direction Technologie	7
1.3 Contexte générale du projet	8
1.3.1 Problématique	8
1.3.2 Objectif du projet	9
1.3.3 Méthodologie de travail	10
1.3.4 Choix technologiques	10
1.4 Concepts Clés	11
1.4.1 Data Mining	11
1.4.2 Machine Learning	12
1.4.3 Data Mining vs Machine Learning	13
1.4.4 Deep Learning	13
1.4.4.1 Principes du Deep Learning	14
1.4.4.2 Applications du Deep Learning	14
1.4.4.3 Différence entre Machine Learning et Deep Learning	14
1.5 Conclusion	15
2 Outils mathématiques d'ordre général	16

TABLE DE MATIÈRES

2.1	Introduction	18
2.2	Techniques de Prétraitemet des Données	18
2.2.1	Normalisation des Données	18
2.2.1.1	Méthodes classiques de normalisation	18
2.2.1.2	Méthodes avancées de normalisation	19
2.2.2	Traitemet des Valeurs Manquantes	20
2.2.2.1	Imputation par la Médiane	20
2.2.2.2	K-Nearest Neighbors pour la Prédiction des Valeurs Manquantes	20
2.2.3	Conclusion	21
2.3	Les indicateurs statistiques	21
2.3.1	Eta Carré	21
2.3.2	Cramer V	21
2.3.3	Lambda de Goodman et Kruskal	22
2.3.4	Covariance	22
2.3.5	Limitation des Indicateurs Statistiques	23
2.3.6	Conclusion	23
2.4	Les Tests de corrélation	23
2.4.1	Tests paramétriques	23
2.4.1.1	Les conditions pour appliquer les tests paramétriques	24
2.4.1.2	Test t de Student	26
2.4.1.3	Test ANOVA	27
2.4.1.4	Test de Pearson	27
2.4.2	Tests non paramétriques	28
2.4.2.1	Test U de Mann-Whitney	28
2.4.2.2	Test de Kruskal-Wallis	29
2.4.2.3	Test de Spearman	29
2.4.2.4	Test du Chi-Deux	30
2.4.2.5	Test exact de Fisher	30
2.5	Équilibrage des Classes avec SMOTE	31
2.6	Sélection des Caractéristiques (Feature Selection)	32
2.6.1	RFE (Recursive Feature Elimination)	32
2.6.2	SelectKBest	33
2.6.3	Feature Importance avec Forêt Aléatoire	33
2.6.4	Conclusion	33
2.7	Modèles de Machine Learning	34
2.7.1	Régression Logistique	34
2.7.1.1	Hyperparamètres	35

TABLE DE MATIÈRES

2.7.2	L'Arbre de Décision	37
2.7.3	La Forêt Aléatoire: un modèle non paramétrique	38
2.7.4	AdaBoost: Amélioration des Arbres de Décision	39
2.7.5	Conclusion	41
2.8	Deep Learning	41
2.8.1	Les Couches dans un Réseau de Neurones	41
2.8.2	Fonctions d'Activation	41
2.8.3	Entraînement et Backpropagation	42
2.8.3.1	Calcul de la Fonction de Perte	42
2.8.3.2	Backpropagation et Descente de Gradient	42
2.8.4	Régularisation et Dropout	43
2.8.5	Hyperparamètres: Signification et Impact	43
2.8.6	Conclusion	43
2.9	Interprétation des Modèles pour la Prédiction de la Satisfaction Client	44
2.9.1	Clustering avec KMeans	44
2.9.2	Calcul de la Probabilité de Satisfaction	44
2.9.3	Interprétation des Résultats	44
2.9.4	Conclusion	44
2.10	Conclusion	45
3	Réalisation	46
3.1	Introduction	47
3.2	Prétraitement des données	47
3.2.1	Préparation et nettoyage des données	47
3.2.2	Normalisation des données	50
3.2.3	Utilisation de plusieurs tables	51
3.3	Analyse des données	51
3.3.1	Analyse descriptive	52
3.3.1.1	Analyse des variables quantitatives	52
3.3.1.2	Analyse des variables qualitative	54
3.3.1.3	Comparaison globale entre OSAT multiclassee et OSAT binaire . .	56
3.3.2	Analyse inférentielle	56
3.3.2.1	Corrélation entre la variable cible et les variables explicatives . .	57
3.3.2.2	Analyse des corrélations entre les variables explicatives	61
3.4	Modélisation	64
3.4.1	Équilibrage des classes (SMOTE)	64
3.4.2	Sélection des caractéristiques	65

TABLE DE MATIÈRES

3.4.3	Modélisation avec Machine Learning	67
3.4.3.1	Prétraitement des données	68
3.4.3.2	Optimisation des hyperparamètres	68
3.4.3.3	Sélection des caractéristiques importantes et réentraînement du modèle	68
3.4.3.4	Optimisation du seuil de classification	68
3.4.3.5	Conclusion	68
3.4.4	Modélisation avec Deep Learning	68
3.4.4.1	Approche PyTorch	69
3.4.4.2	Approche Keras/TensorFlow	69
3.4.4.3	Conclusion	70
3.5	Conclusion	70
4	Résultats et interprétation	71
4.1	Introduction	72
4.2	Résultats des Modèles de Machine Learning	72
4.2.1	Modèle de Régression Logistique	72
4.2.1.1	Résultats pour les trois datasets	72
4.2.1.2	Comparaison des résultats entre les trois datasets	73
4.2.2	Modèle Arbre de Décision	74
4.2.2.1	Résultats pour les trois datasets	74
4.2.2.2	Comparaison des résultats entre les trois datasets	76
4.2.3	Modèle de Random Forest	76
4.2.3.1	Résultats pour les trois datasets	76
4.2.3.2	Sélection automatique du seuil d'importance	78
4.2.3.3	Approche de KMeans et résultats pour <i>retail_juin</i>	79
4.2.3.4	Résultats après KMeans sur <i>retail_juin</i>	80
4.2.3.5	Conclusion	80
4.3	Résultats des Modèles de Deep Learning	81
4.3.1	Modèle Deep Learning avec PyTorch	81
4.3.1.1	Comparaison des résultats entre les trois datasets	82
4.3.2	Résultats des Modèles avec Keras/TensorFlow	83
4.3.2.1	Résultats pour <i>network_fev</i>	83
4.3.2.2	Résultats pour <i>network_may</i>	83
4.3.2.3	Résultats pour <i>retail_juin</i>	84
4.3.2.4	Comparaison des Résultats	84
4.4	Comparaison Entre les Modèles de Machine Learning et de Deep Learning	84
4.4.1	Comparaison et Validation des Modèles de Machine Learning	85

TABLE DE MATIÈRES

4.4.2	Comparaison et Validation des Modèles de Deep Learning	86
4.4.3	Comparaison Globale : Machine Learning vs Deep Learning	87
4.4.4	Modèle d'Ensemble (Random Forest, AdaBoost et Réseau de Neurones) . .	87
4.4.4.1	Motivation de l'approche d'ensemble	88
4.4.4.2	Justification des datasets sélectionnés	88
4.4.4.3	Optimisation des poids pour la combinaison des modèles	88
4.4.4.4	Résultats pour <i>network_fev</i>	89
4.4.4.5	Résultats pour <i>retail_juin</i>	89
4.4.5	Vérification de surajustement du meilleur modèle	90
4.4.6	Conclusion	90
4.5	Vérification du surajustement du meilleur modèle	91
4.5.1	Analyse des résultats	91
4.5.2	Conclusion sur le surajustement	92
4.6	Profil des Clients Satisfaits et Interprétation des Variables	92
4.6.1	Résultats pour <i>network_fev</i>	92
4.6.2	Résultats pour <i>network_may</i>	93
4.6.3	Comparaison des Résultats Entre les Trois Datasets	94
4.7	Conclusion	95
Conclusion générale		i
Webographie		i
Biographie		ii
Annexe		v
Annexe		xv



Liste des abréviations

OSAT Overall Satisfaction Attributable

KNN K-Nearest Neighbors

SQL Structured Query Language

RNN Recurrent Neural Network

NLP Natural Language Processing

ETA Effect of Total Amount (Eta-squared)

ML Machine Learning

DL Deep Learning

SMOTE Synthetic Minority Over-sampling Technique

RFE Recursive Feature Elimination

KMeans K-Means Clustering

RF Random Forest

AdaBoost Adaptive Boosting

AUC Area Under the Curve

ROC Receiver Operating Characteristic

PCA Principal Component Analysis

Eta² Eta Carré

Cramér's V Cramer V

GKR Goodman et Kruskal Lambda

CV Coefficient de Variation

SD Écart-Type

CI Intervalle de Confiance

ANOVA Analyse de la Variance

t-test Test t

Z-test Test Z

MSE Mean Squared Error

TABLE DE MATIÈRES

LR Logistic Regression

DT Decision Tree

SVM Support Vector Machine

NN Neural Network

QDA Quadratic Discriminant Analysis

LDA Linear Discriminant Analysis

GBM Gradient Boosting Machine

XGBoost Extreme Gradient Boosting

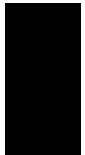
CNN Convolutional Neural Network

Liste des figures

1.1	Récapitulatif des valeurs de l'entreprise	5
1.2	La répartition des principaux postes de direction	7
1.3	Répartition des différentes directions au sein de la direction Technique	8
3.1	Exemple de transformation des variables catégorielles	48
3.2	Exemple d'agrégation des variables quantitatives	49
3.3	Distribution de la variable recharge par classe OSAT multiclass	52
3.4	Distribution de la variable monthly_rech_subscriber_seg par classe OSAT multiclass .	55
3.5	Distribution de la variable monthly_rech_subscriber_seg par classe OSAT binaire .	55
3.6	Heatmap de la matrice de corrélation globale pour la table de Juin	57
3.7	Tableau de contingence pour device_type et osat_binary	58
3.8	Nuage de points entre recharge_moyenne et Arpu_moyenne	61
3.9	Matrice de corrélation entre recharge_moyenne et Arpu_moyenne	62
3.10	Nuage de points entre voice_volume_moyenne et voice_amount_moyenne	62
3.11	Matrice de corrélation entre voice_volume_moyenne et voice_amount_moyenne . .	63
3.12	Tests ANOVA/Kruskal-Wallis entre mou_moyenne et Flag_4g_binary	64
3.13	Pourcentages des classes OSAT avant équilibrage	65
3.14	Pourcentages des classes OSAT après équilibrage avec SMOTE	65
3.15	Classement des caractéristiques par RFE	66
3.16	Scores des caractéristiques par SelectKBest	66
3.17	Importances des caractéristiques par Random Forest	67
3.18	Caractéristiques communes importantes entre les trois méthodes	67
4.1	Matrice de confusion pour <i>network_fev</i>	73
4.2	Courbe ROC pour <i>network_fev</i>	73
4.3	Matrice de confusion pour <i>network_may</i>	73
4.4	Matrice de confusion pour <i>retail_juin</i>	73
4.5	Matrice de confusion pour <i>network_fev</i>	75
4.6	Matrice de confusion pour <i>network_may</i>	75
4.7	Matrice de confusion pour <i>retail_juin</i>	75
4.8	Matrice de confusion pour <i>network_fev</i>	77

4.9	Courbe ROC pour <i>network_fev</i>	77
4.10	Matrice de confusion pour <i>network_may</i>	77
4.11	Courbe ROC pour <i>network_may</i>	77
4.12	Matrice de confusion pour <i>retail_juin</i>	77
4.13	Courbe ROC pour <i>retail_juin</i>	77
4.14	Exactitude en fonction du seuil d'importance.	78
4.15	Matrice de confusion avec caractéristiques importantes (<i>retail_juin</i>).	78
4.16	Meilleurs scores de silhouette pour <i>retail_juin</i>	79
4.17	Répartition des variables en clusters après KMeans.	79
4.18	Rapport de classification pour <i>retail_juin</i> après KMeans.	80
4.19	Matrice de confusion pour <i>network_fev</i> (PyTorch).	81
4.20	Matrice de confusion pour <i>network_may</i> (PyTorch).	81
4.21	Matrice de confusion pour <i>retail_juin</i> (PyTorch).	82
4.22	Matrice de confusion pour <i>network_fev</i> avec Keras/TensorFlow.	83
4.23	Matrice de confusion pour <i>network_may</i> avec Keras/TensorFlow.	83
4.24	Matrice de confusion et classification report pour <i>retail_juin</i> avec Keras/TensorFlow.	84
4.25	Courbes de précision et de perte sur les epochs pour <i>retail_juin</i>	84
4.26	Matrice de confusion pour <i>network_fev</i> avec le modèle d'ensemble.	89
4.27	Courbe ROC pour <i>network_fev</i> avec le modèle d'ensemble.	89
4.28	Matrice de confusion pour <i>retail_juin</i> avec le modèle d'ensemble.	90
4.29	Courbe ROC pour <i>retail_juin</i> avec le modèle d'ensemble.	90
4.30	Performances du modèle sur les jeux de validation et de test pour network_may . . .	91
4.31	Profil du client le plus satisfait pour <i>network_fev</i> basé sur les intervalles de satisfaction.	93
4.32	Profil du client le plus satisfait pour <i>network_may</i> basé sur les intervalles de satisfaction.	93
4.33	Profil du client le plus satisfait pour <i>retail_juin</i> basé sur les intervalles de satisfaction.	94
34	Imputation de la colonne OSAT à partir de la moyenne des réponses aux questions . .	v
35	Code pour l'imputation KNN	v
36	Remplacement des valeurs aberrantes par la médiane.	vi
37	Transformation Box-Cox sur Recharge_moyenne	vi
38	Calcul de l'indice de Cramér's V	vi
39	Test du Chi-Carré et test de Fisher (si applicable)	vii
40	Calcul de la Lambda de Goodman et Kruskal	viii
41	Calcul de l'Eta carré pour osat_binary	viii
42	Test d'homogénéité des variances de Levene	ix
43	Calcul de la covariance et test de normalité (Shapiro-Wilk) pour deux variables . . .	ix
44	Choix du test de corrélation: Pearson ou Spearman, basé sur la normalité des données	ix

45	Code de sélection des caractéristiques avec RFE, SelectKBest, et Random Forest	x
46	Optimisation des hyperparamètres pour Random Forest	x
47	Importances des caractéristiques par Random Forest	xi
48	Courbe ROC pour Random Forest avec AUC.	xi
49	Préparation des données pour PyTorch.	xi
50	Architecture du modèle de classification binaire dans PyTorch.	xii
51	Réglage des hyperparamètres pour le modèle PyTorch.	xii
52	Sélection du meilleur modèle PyTorch et évaluation finale.	xiii
53	Architecture classique du modèle Keras.	xiii
54	Optimisation des hyperparamètres du modèle Keras.	xiv
55	Optimisation de l'architecture des couches dans Keras.	xiv
56	Résultat de la Lambda de Goodman et Kruskal pour Flag_4g_binary	xv
57	Résultat de l'Eta carré pour osat_binary	xv
58	Résultats du test de normalité et de variances	xv
59	Résultats du test de Mann-Whitney	xvi
60	Interprétation des résultats statistiques entre les deux variables	xvi
61	Test du Chi-Carré et test de Fisher (si applicable)	xvi
62	Calcul de la Lambda de Goodman et Kruskal	xvii
63	Calcul de l'Eta carré pour osat_binary	xvii
64	Test d'homogénéité des variances de Levene	xviii
65	Calcul de la covariance et test de normalité (Shapiro-Wilk) pour deux variables	xviii
66	Choix du test de corrélation: Pearson ou Spearman, basé sur la normalité des données	xviii
67	Code de sélection des caractéristiques avec RFE, SelectKBest, et Random Forest	xix
68	Optimisation des hyperparamètres pour Random Forest	xix
69	Importances des caractéristiques par Random Forest	xx
70	Courbe ROC pour Random Forest avec AUC.	xx
71	Préparation des données pour PyTorch.	xx
72	Architecture du modèle de classification binaire dans PyTorch.	xxi
73	Réglage des hyperparamètres pour le modèle PyTorch.	xxi
74	Sélection du meilleur modèle PyTorch et évaluation finale.	xxii
75	Architecture classique du modèle Keras.	xxii
76	Optimisation des hyperparamètres du modèle Keras.	xxiii
77	Optimisation de l'architecture des couches dans Keras.	xxiii



INTRODUCTION GÉNÉRALE

L'industrie des télécommunications a connu une évolution rapide au cours des dernières décennies, devenant un pilier central de la société moderne. Ce secteur, autrefois centré sur la simple fourniture de services de téléphonie et de communication, s'est progressivement diversifié pour offrir une multitude de services numériques à une clientèle de plus en plus exigeante. Dans ce contexte, le marketing est devenu un élément clé pour les entreprises de télécommunications, leur permettant de comprendre et d'anticiper les besoins de leurs clients afin de rester compétitives.

Historiquement, le marketing dans le secteur des télécommunications reposait sur l'intuition, l'expérience, et l'analyse des tendances par des experts en marketing. Les stratégies étaient souvent basées sur des études de marché traditionnelles, des sondages et des analyses qualitatives. Cependant, avec l'explosion des données numériques et l'avènement des technologies de la data science, ce paradigme a radicalement changé.

Aujourd'hui, l'exploitation des données occupe une place au cœur des stratégies marketing du secteur des télécommunications. Les entreprises ne se contentent plus de comprendre les comportements des clients à travers des méthodes traditionnelles, mais utilisent des techniques avancées de data science pour extraire des insights à partir de vastes quantités de données. Ces technologies permettent non seulement de décrire les comportements passés, mais aussi de prédire les actions futures des clients, facilitant ainsi la prise de décision et l'optimisation des stratégies marketing.

L'intégration de la data science dans le marketing des télécommunications a ouvert de nouvelles perspectives, transformant la manière dont les entreprises interagissent avec leurs clients. **L'analyse prédictive**, en particulier, permet de modéliser et d'anticiper les **scores de satisfaction** des clients, offrant ainsi aux entreprises un levier puissant pour améliorer leurs services, fidéliser leurs clients et développer des offres sur mesure. Cette révolution numérique continue de remodeler le secteur, rendant les stratégies marketing plus précises, personnalisées et plus efficaces.

INTRODUCTION GÉNÉRALE

Cette recherche s'insère dans une analyse détaillée ayant pour objectif de mieux comprendre le comportement des consommateurs **d'Ooredoo** et à développer **un modèle statistique** capable de prédire les réponses aux enquêtes de satisfaction.

En utilisant **des outils mathématiques et statistiques** avancés, ce projet cherche à fournir des solutions pour anticiper les perceptions et réactions des clients, permettant ainsi d'améliorer les services offerts par **Ooredoo** et de renforcer sa position sur le marché des télécommunications.

Présentation générale du projet

Contents

1.1	Introduction	4
1.2	Présentation du lieu du stage	4
1.2.1	Présentation de l'entreprise	4
1.2.2	Les Valeurs de l'entreprise	4
1.2.3	La vision de l'entreprise	6
1.2.4	L'organigramme de l'entreprise	6
1.2.5	L'organigramme de la direction Technologie	7
1.3	Contexte générale du projet	8
1.3.1	Problématique	8
1.3.2	Objectif du projet	9
1.3.3	Méthodologie de travail	10
1.3.4	Choix technologiques	10
1.4	Concepts Clés	11
1.4.1	Data Mining	11
1.4.2	Machine Learning	12
1.4.3	Data Mining vs Machine Learning	13
1.4.4	Deep Learning	13
1.5	Conclusion	15

1.1 Introduction

Ce premier chapitre présente une vue d'ensemble du projet en introduisant l'entreprise, son contexte, ainsi que la méthodologie et les choix technologiques adoptés. De plus, il expose les concepts clés liés au traitement des données tels que le Data Mining, le Machine Learning et le Deep Learning, qui jouent un rôle fondamental dans l'analyse et la modélisation des données client.

1.2 Présentation du lieu du stage

1.2.1 Présentation de l'entreprise

Le groupe Ooredoo est un acteur majeur des télécommunications à l'échelle internationale, avec une forte présence dans les régions du Moyen-Orient, de l'Afrique du Nord et de l'Asie du Sud-Est.

Créé en 1987 sous le nom « **Qtel** » ou « **Qatar Telecom** », le groupe s'est imposé comme un leader en offrant des solutions de téléphonie mobile et fixe, ainsi que des services Internet et des solutions pour les entreprises.

Ooredoo est présent dans plusieurs pays, dont le Qatar, le Koweït, le Sultanat d'Oman, l'Algérie, l'Irak, la Palestine, les Maldives et l'Indonésie, contribuant au développement des infrastructures de communication dans ces marchés. **Ooredoo Tunisie**, filiale de la société mère qatarie Ooredoo (également connue sous les noms "Qtel" ou "Qatar Telecom"), a été fondée en 2002.

La table 1.1 ci-dessous comporte les informations nécessaires pour identifier l'entreprise d'accueil Ooredoo.

Création	11 mai 2022
Forme juridique	Société anonyme
Siège social	Immeuble Zenith, Lac2
Direction	Mansoor Rashid Al Khater (Directeur général) Waleed Mohamed Al Sayed (Président du conseil administratif)
Actionnaire	Ooredoo (90%) Tunisie (10%)
Activité	Téléphonie mobile et Internet
Société mère	Ooredoo
Site web	Ooredoo.tn
Chiffre d'affaire	1 300 000 000 (Dt) en 2022

Table 1.1: Informations relatives à l'entreprise

1.2.2 Les Valeurs de l'entreprise

Ooredoo s'engage à créer des services appréciés par ses clients, en s'appuyant sur des valeurs fondamentales qui guident ses actions et déterminent la manière dont elle réalise ses objectifs. Ces

valeurs assurent non seulement la cohésion et renforcent la culture d'entreprise, mais elles sont également le socle de son succès sur le marché.

Les trois valeurs principales d'Ooredoo, telles que mises en avant sur son site web, sont **Caring**, **Connecting**, et **Challenging**:

- **Caring:** Cette valeur se traduit par une approche simple et transparente dans toutes les interactions d'Ooredoo. L'entreprise s'efforce de répondre rapidement aux besoins de ses clients tout en montrant un respect profond pour chaque individu. En adoptant une attitude ouverte et empathique, Ooredoo veille à ce que ses actions soient guidées par une véritable considération pour les attentes et besoins des personnes qu'elle sert.
- **Connecting:** Ooredoo met un point d'honneur à offrir un accès fiable et pertinent à la communauté, en fournissant des services qui répondent directement aux besoins des utilisateurs. L'entreprise s'efforce de maintenir une relation de confiance avec ses clients en restant fidèle à ses engagements et en offrant des services sur lesquels ils peuvent compter. Connecting signifie aussi créer des liens significatifs et durables qui renforcent la cohésion sociale et l'accès à l'information.
- **Challenging:** Guidée par un esprit de jeunesse et une passion pour l'excellence, Ooredoo se positionne en leader du changement et de l'innovation. L'entreprise ne se contente pas de suivre les tendances, elle les définit en cherchant constamment à innover et s'améliorer. Cette valeur incarne la volonté d'Ooredoo de repousser les limites et de s'imposer comme un acteur dynamique dans le secteur des télécommunications.

Ces trois valeurs fondamentales—**Caring**, **Connecting**, et **Challenging**—résument l'approche d'Ooredoo en tant qu'entreprise dédiée à enrichir la vie numérique de ses clients, tout en favorisant une culture d'entreprise inclusive et tournée vers l'avenir.

La figure 1.1 ci-dessous présente un récapitulatif des valeurs mentionnées précédemment.



Figure 1.1: Récapitulatif des valeurs de l'entreprise

1.2.3 La vision de l'entreprise

Ooredoo, en tant qu'entreprise résolument orientée vers les besoins des populations, est animée par une vision claire: **enrichir la vie numérique des gens**. Cette vision repose sur la conviction que la communication est un puissant levier pour stimuler le développement humain et aider les individus à atteindre leurs objectifs, tant personnels que professionnels.

Ooredoo se donne pour mission de permettre à ses clients d'accéder au meilleur de l'Internet, de manière personnalisée et adaptée à leurs besoins uniques.

Pour ce faire, l'entreprise continue d'investir dans son réseau Supernet, garantissant des vitesses plus rapides et une connectivité fluide qui répond aux exigences croissantes du monde numérique.

En tant que véritable facilitateur digital, Ooredoo aspire à simplifier la vie de ses utilisateurs et à leur offrir des expériences numériques enrichissantes et gratifiantes. Prenant les devants dans le domaine des services intelligents, Ooredoo contribue à la construction des « smart cities » et des « smart stadiums » de demain, tout en proposant une riche gamme de services, allant des divertissements numériques à Ooredoo Mobile Money.

1.2.4 L'organigramme de l'entreprise

Ooredoo Tunisie compte un effectif total de **1357 salariés**, comprenant **838 hommes** et **519 femmes**.

Le siège se compose de dix directions, à savoir : Direction des Ressources Humaines, Direction de la Technologie, Direction Administrative et Financière, Direction Vente et Distribution, Direction Marketing et Direction Juridique et Régulation. La figure 1.2 ci-dessous montre la répartition des principaux postes de direction au sein de l'entreprise.

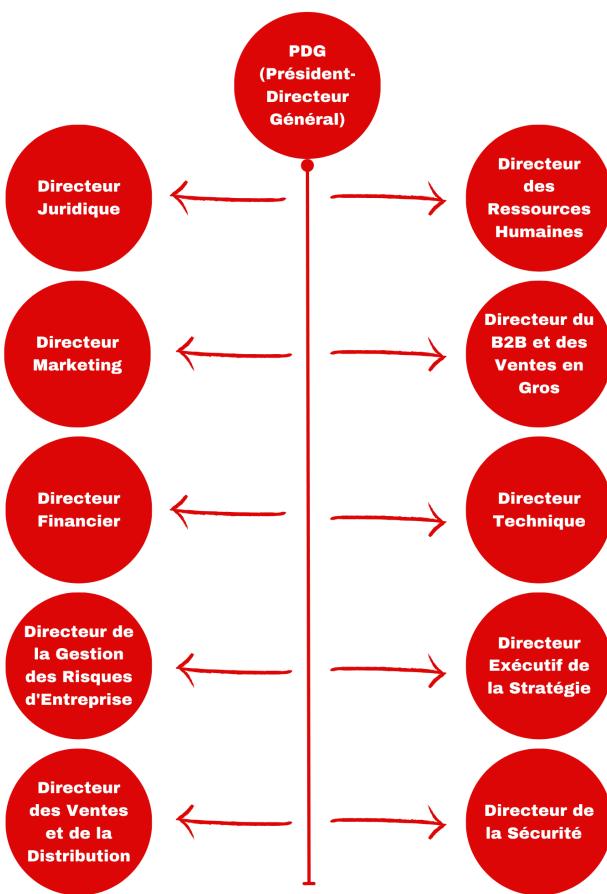


Figure 1.2: La répartition des principaux postes de direction

1.2.5 L'organigramme de la direction Technologie

Au sein de la Direction Technologie d'Ooredoo Tunisie, plusieurs sous-directions jouent un rôle clé dans la gestion et le développement des infrastructures et services technologiques. La Direction CME (Centre de Maintenance et d'Exploitation) est responsable de la maintenance préventive et corrective des équipements ainsi que de leur exploitation quotidienne pour garantir leur bon fonctionnement.

- La Direction Support assure l'assistance technique aux utilisateurs internes et externes, gérant les demandes de support et résolvant les problèmes techniques.
- La Direction Déploiement et Maintenance se concentre sur l'installation et la configuration des nouvelles solutions technologiques, ainsi que sur la maintenance des systèmes existants.
- La Direction Technique est chargée de la gestion des aspects techniques des projets, incluant la conception et l'implémentation de nouvelles technologies.
- La Direction Réseau et Service gère l'infrastructure réseau, veillant à la performance et à la sécurité des services de télécommunications.
- La Direction Budget, Process et Gestion de Projet supervise la planification budgétaire, les processus internes et le suivi des projets pour assurer leur livraison dans les délais et les coûts impartis.

- La Direction des Systèmes d'Information (DSI), où se déroule mon stage, est essentielle au sein de la Direction Technologie. Elle s'occupe de la mise en place, du suivi, de la maintenance et de l'optimisation des systèmes informatiques d'Ooredoo. La DSI centralise les données et coordonne des applications clés comme les CRM, ERP, et Data Warehouse, soutenant ainsi les opérations quotidiennes, optimisant les processus internes et facilitant la prise de décision stratégique. Sa gestion est cruciale pour répondre aux besoins croissants en gestion des données et en technologies de l'information.

La figure 1.3 ci-dessous montre la répartition des différentes directions au sein de la direction Technique.

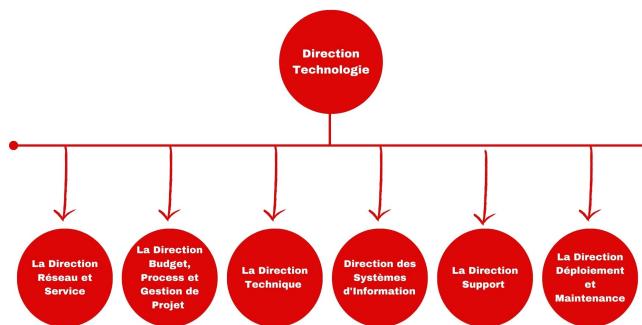


Figure 1.3: Répartition des différentes directions au sein de la direction Technique

1.3 Contexte générale du projet

1.3.1 Problématique

Dans le secteur des télécommunications, la satisfaction des clients est essentielle pour fidéliser la clientèle et se démarquer de la concurrence, notamment face à des acteurs comme Orange et Tunisie Telecom. Les revenus des opérateurs dépendent largement de leur capacité à retenir leurs clients, car l'acquisition de nouveaux clients est souvent plus coûteuse.

Comprendre les raisons de la satisfaction ou de l'insatisfaction des clients est donc crucial pour ajuster les stratégies de service et de marketing. Il est important de noter que dans ce secteur, deux types principaux de clients se distinguent : les **clients prépayés** et les **clients postpayés**.

Clients Prépayés: Ces clients paient à l'avance pour les services téléphoniques (appels, SMS, données). Leur utilisation dépend de leurs habitudes de consommation et de leur capacité à recharger leur crédit. Ils constituent une catégorie homogène, dont le comportement est influencé par la fréquence et le montant des recharges. Ainsi, les données liées aux recharges, comme `recharge_amount` et `number_of_recharges`, sont des indicateurs clés pour comprendre leur satisfaction. **Clients Postpayés:** Ces clients, au contraire, paient pour les services utilisés à la fin du mois. Leur consommation est plus stable et moins directement influencée par les recharges. Toutefois,

leur comportement est fortement impacté par les offres et promotions auxquelles ils souscrivent, rendant leur analyse plus complexe.

Ces distinctions entre clients prépayés et postpayés sont essentielles pour adapter les analyses et les stratégies de rétention.

En **2022**, **Tunisie Telecom** a enregistré un chiffre d'affaires de 1 288,22 millions de dinars, en **hausse de 2,6%** par rapport à **2021**. **Orange Tunisie** a atteint 773,16 millions de dinars, avec une **augmentation de 8,8%**. En revanche, **Ooredoo Tunisie**, qui avait affiché le plus gros chiffre d'affaires du secteur en 2021 avec 1 351,1 millions de dinars, a vu ce chiffre **baisser de 0,7%**, s'établissant à 1 306,40 millions de dinars en 2022. Cette situation met en lumière l'importance de mieux comprendre les attentes des clients pour inverser cette tendance.

Ooredoo mène des enquêtes mensuelles sur divers aspects tels que la recharge, le réseau, et le retail, pour analyser le comportement des clients au cours des mois précédents. L'objectif est d'identifier les facteurs qui influencent le score de satisfaction (OSAT) et de développer des modèles prédictifs grâce aux techniques de machine learning et de deep learning.

En utilisant ces analyses, Ooredoo peut non seulement repérer les clients à risque de départ, mais aussi anticiper leurs besoins pour améliorer leur expérience. Ce projet vise à renforcer la relation client en exploitant les outils avancés de la data science, essentiels pour rester compétitif sur le marché.

Pour répondre à cette problématique, ce projet s'efforcera de répondre aux questions suivantes:

- Quels sont les principaux facteurs qui influencent la satisfaction des clients ?
- Quelles différences de comportement peut-on observer entre les clients satisfaits et insatisfaits, notamment entre les clients prépayés et postpayés ?
- Quels modèles de machine learning ou de deep learning offrent les meilleures performances pour prédire la satisfaction générale des clients ?

1.3.2 Objectif du projet

L'objectif principal de ce projet est d'explorer et d'analyser les données clients pour comprendre les facteurs influençant leur satisfaction. En se concentrant principalement sur les **clients prépayés**, ce projet vise à :

- **Identifier les principaux facteurs** qui influencent la satisfaction ou l'insatisfaction des clients.
- **Distinguer les comportements** entre clients satisfaits et insatisfaits en analysant les réponses aux enquêtes et les comportements d'utilisation des services.
- **Développer des modèles prédictifs** utilisant des techniques de machine learning et de deep learning pour prédire avec précision le score de satisfaction (OSAT).
- **Optimiser l'expérience client** en identifiant les segments nécessitant des améliorations spécifiques, notamment par une analyse approfondie des clients prépayés.

Le projet se déroulera en plusieurs phases : la collecte et le nettoyage des données, l'analyse des résultats, le développement et l'évaluation de modèles prédictifs, ainsi que l'interprétation des résultats pour une application directe dans les stratégies marketing.

1.3.3 Méthodologie de travail

Pour mener à bien ce projet, une approche méthodologique structurée est essentielle. La démarche se décline en plusieurs étapes clés :

1. **Compréhension de la problématique métier:** Cette phase consiste à analyser les enjeux de la satisfaction client dans les télécommunications, à définir les objectifs du projet et à identifier les critères de succès pour Ooredoo Tunisie.
2. **Compréhension des données:** Cette étape inclut la collecte des données à partir des enquêtes mensuelles d'Ooredoo, suivie de leur nettoyage, transformation et préparation pour garantir leur qualité et pertinence pour l'analyse.
3. **Analyse des variables:** Cette phase comprend l'exploration des relations entre les variables explicatives et la variable cible (score de satisfaction) en utilisant des techniques telles que la corrélation. Cette analyse permet d'identifier les facteurs influents, leurs interactions et leur impact sur la satisfaction des clients.
4. **Modélisation:** Nous utilisons divers algorithmes de machine learning et deep learning pour créer des modèles prédictifs. Cette étape vise à développer des modèles capables de prédire avec précision le score de satisfaction des clients.
5. **Évaluation du modèle:** Après la modélisation, chaque modèle est évalué en termes de performance à l'aide de métriques telles que l'exactitude, la précision, le rappel et le F1-score. Cette évaluation permet de sélectionner le modèle le plus performant.

1.3.4 Choix technologiques

Pour réaliser les différentes étapes du projet, plusieurs outils technologiques ont été sélectionnés. Ces outils peuvent être classés en trois catégories: logiciels et environnements de développement, langages de programmation et bibliothèques.

- **Logiciels et environnements de développement**

- **SAS Viya:** Une plateforme analytique puissante permettant l'accès aux données, leur manipulation et l'exécution de requêtes complexes en SQL. Utilisée principalement pour manipuler les tables de données et exporter les résultats finaux pour l'analyse.
- **Jupyter Notebook:** Un environnement de développement interactif combinant code, texte et visualisations dans un même document, particulièrement adapté à l'analyse exploratoire des données et la modélisation prédictive.

- **Langages de programmation**

- **Python:** Langage polyvalent largement utilisé dans l'analyse de données, la modélisation statistique et le machine learning. Il permet de réaliser des tâches complexes grâce à ses bibliothèques variées.

- **SQL:** Langage utilisé pour interagir avec les bases de données relationnelles, permettant d'extraire, manipuler et gérer des ensembles de données.

- **Bibliothèques utilisées**

- **Numpy:** Bibliothèque fondamentale pour le calcul numérique, permettant de manipuler des tableaux multidimensionnels et de réaliser des opérations mathématiques complexes.
- **Pandas:** Bibliothèque spécialisée dans la manipulation et l'analyse de données, offrant des structures flexibles pour le nettoyage et la transformation des données.
- **Matplotlib:** Bibliothèque de visualisation pour créer des graphiques statiques en 2D.
- **Seaborn:** Basée sur Matplotlib, elle simplifie la création de graphiques statistiques et propose des styles graphiques avancés.
- **Scikit-learn:** Fournit une large gamme d'algorithmes pour la classification, la régression, le clustering et la réduction de dimension.
- **TensorFlow:** Bibliothèque open source pour créer et entraîner des modèles de machine learning, notamment des réseaux de neurones.
- **Keras:** Interface de haut niveau pour simplifier la création et la manipulation de réseaux de neurones profonds.
- **PyTorch:** Bibliothèque open source de Facebook, largement utilisée pour la recherche en deep learning.

1.4 Concepts Clés

L'ère numérique a révolutionné l'exploitation des données. Dans ce contexte, le Machine Learning, le Data Mining et le Deep Learning jouent un rôle clé pour analyser les données massives et en extraire des insights significatifs.

Le Data Mining permet d'explorer et de découvrir des modèles dans les données, tandis que le Machine Learning utilise ces insights pour construire des modèles prédictifs. Enfin, le Deep Learning, en tant que sous-domaine du Machine Learning, emploie des réseaux de neurones profonds pour traiter des données non structurées et capturer des relations complexes.

Cette section abordera chacun de ces concepts, en soulignant leur définition et leur application dans le projet de prédiction de la satisfaction client.

1.4.1 Data Mining

Le **Data Mining** [?], ou fouille de données, est le processus d'analyse de grandes quantités de données pour découvrir des informations utiles, des tendances cachées, ou des relations insoupçonnées. Utilisé dans divers secteurs comme le marketing, la santé, ou l'éducation, il permet aux entreprises de mieux comprendre leurs clients et d'optimiser leurs stratégies en fonction des comportements observés.

Contrairement à d'autres approches, le Data Mining se concentre surtout sur l'analyse exploratoire des données, visant principalement la découverte de connaissances sans passer par une phase de prédiction. Les principales étapes du processus de Data Mining sont les suivantes:

- **Collecte des données:** Rassemblement d'informations à partir de diverses sources, telles que des bases de données ou des fichiers.
- **Préparation des données:** Nettoyage et transformation des données afin d'assurer leur qualité, en supprimant les erreurs, en gérant les valeurs manquantes, et en normalisant les variables. Cette phase inclut également des techniques comme le **clustering** pour regrouper les données en sous-ensembles homogènes, facilitant ainsi leur analyse. Dans certains cas, des méthodes comme **SMOTE** (Synthetic Minority Over-sampling Technique) peuvent être utilisées pour rééquilibrer les jeux de données déséquilibrés.
- **Exploration des données:** Utilisation d'outils statistiques et analytiques pour identifier des tendances, des motifs ou des anomalies.

Le Data Mining applique des techniques comme le clustering et la classification pour explorer les données, avec un objectif surtout descriptif, contrairement aux approches prédictives du Machine Learning. Il permet ainsi de comprendre les données, découvrir des corrélations, et faciliter des décisions éclairées.

1.4.2 Machine Learning

Le **Machine Learning**, ou apprentissage automatique, est une branche de l'intelligence artificielle (IA) qui permet aux machines d'apprendre à partir de données sans être explicitement programmées pour chaque tâche. Contrairement au Data Mining, le Machine Learning est principalement orienté vers la prédiction et l'automatisation de tâches à partir de modèles qui apprennent continuellement à partir des données.

Il existe trois principaux types de Machine Learning:

- **Apprentissage supervisé:** Le modèle apprend à partir de données étiquetées, où les entrées sont associées à des sorties connues. Exemple: prédire la satisfaction des clients en fonction de caractéristiques passées.
- **Apprentissage non supervisé:** Le modèle identifie des structures cachées dans les données non étiquetées. Exemple: regrouper des clients en fonction de comportements similaires via le **clustering**.
- **Apprentissage par renforcement:** Le modèle interagit avec un environnement et apprend à partir des récompenses ou pénalités qu'il reçoit. Cette approche est souvent utilisée dans des applications comme la robotique ou les jeux vidéo.

Le Machine Learning se distingue du Data Mining par son caractère itératif et automatisé. Une fois que le modèle est entraîné, il peut faire des prédictions sur de nouvelles données et s'améliorer au fil du temps. Le processus typique de Machine Learning comprend les étapes suivantes:

- **Sélection des caractéristiques:** Identification des variables clés pour l'entraînement des modèles.

- **Choix des algorithmes:** En fonction des objectifs, différents algorithmes comme la **régression logistique**, les **forêts aléatoires** ou les **SVM** sont utilisés.
- **Entraînement du modèle:** Le modèle est ajusté à l'aide de données d'entraînement pour minimiser l'erreur entre les prédictions et les résultats réels.
- **Évaluation et optimisation:** Le modèle est évalué et amélioré à l'aide de techniques comme la validation croisée et l'optimisation des hyperparamètres.

1.4.3 Data Mining vs Machine Learning

Bien que le **Data Mining** et le **Machine Learning** partagent des outils et objectifs, comme l'exploration des données pour extraire des informations utiles, leurs approches et finalités diffèrent:

- Le **Data Mining** est principalement orienté vers l'exploration et la compréhension des données historiques à l'aide de techniques descriptives. Il est souvent utilisé pour découvrir des modèles et des relations au sein des données.
- Le **Machine Learning**, quant à lui, adopte une approche prédictive et automatisée. Il vise à créer des modèles capables de généraliser à de nouvelles données pour prédire des résultats futurs.
- Dans le Data Mining, les techniques comme la **classification** et le **clustering** sont utilisées pour segmenter les données et en extraire des connaissances. En Machine Learning, ces mêmes techniques sont utilisées pour construire des modèles prédictifs qui évoluent au fil du temps.

En somme, bien que complémentaires, le Data Mining et le Machine Learning diffèrent dans leurs applications. Le Data Mining s'attache à décrire et analyser les données existantes, tandis que le Machine Learning se concentre sur l'automatisation et la prédiction, permettant de prendre des décisions ou d'effectuer des tâches complexes à grande échelle.

1.4.4 Deep Learning

Le **Deep Learning** est une branche spécifique du Machine Learning qui repose sur l'utilisation de réseaux de neurones artificiels. Contrairement aux algorithmes de Machine Learning traditionnels, les modèles de Deep Learning peuvent apprendre à partir de grandes quantités de données non structurées et extraire automatiquement des caractéristiques complexes sans intervention humaine.

Ce processus est particulièrement efficace dans des domaines comme la reconnaissance d'images, le traitement du langage naturel, et l'analyse vocale, où les données sont volumineuses et hétérogènes.

Les réseaux de neurones artificiels utilisés en Deep Learning sont composés de plusieurs couches d'unités (ou neurones) qui sont interconnectées. Les modèles les plus communs sont les réseaux de neurones à propagation avant (feedforward) et les réseaux de neurones convolutifs (CNN) pour les images.

1.4.4.1 Principes du Deep Learning

Le Deep Learning se distingue par sa capacité à apprendre des représentations complexes à travers les couches hiérarchiques des réseaux de neurones:

- **Les couches de neurones:** Un réseau de neurones profond est composé de plusieurs couches successives. Chaque couche transforme les données reçues, en extrayant des caractéristiques de plus en plus abstraites.
- **Fonctions d'activation:** Chaque neurone utilise une fonction d'activation (comme **ReLU**, **sigmoïde** ou **tanh**) pour introduire de la non-linéarité, permettant ainsi au modèle d'apprendre des relations complexes dans les données.
- **Entraînement par rétropropagation:** Les réseaux de neurones sont entraînés à l'aide d'une technique appelée **backpropagation**, qui ajuste les poids des connexions entre les neurones en minimisant une fonction de coût via la descente de gradient.
- **Régularisation et Dropout:** Pour prévenir le sur-apprentissage (overfitting), des techniques telles que la régularisation L2 ou le **dropout** sont appliquées. Le dropout désactive aléatoirement des neurones pendant l'entraînement pour améliorer la généralisation.

1.4.4.2 Applications du Deep Learning

Le Deep Learning est particulièrement adapté pour les tâches impliquant de grandes quantités de données non structurées. Voici quelques exemples d'applications:

- **Reconnaissance d'images:** Utilisation des réseaux de neurones convolutifs (CNN) pour identifier des objets, visages ou scènes dans les images.
- **Traitement du langage naturel (NLP):** Utilisation de modèles tels que les **réseaux de neurones récurrents (RNN)** ou les **transformers** pour la traduction, la génération de texte, ou l'analyse des sentiments.
- **Analyse vocale:** Reconnaissance vocale automatisée via des réseaux profonds pour transformer les signaux audio en texte ou en commandes.

1.4.4.3 Différence entre Machine Learning et Deep Learning

Le Deep Learning, sous-catégorie du Machine Learning, se distingue par sa capacité à traiter de grandes quantités de données et à extraire automatiquement des caractéristiques complexes sans intervention manuelle. Voici quelques différences principales:

- **Caractéristiques automatiques:** Contrairement au Machine Learning traditionnel où les ingénieurs sélectionnent manuellement les caractéristiques, en Deep Learning, les modèles apprennent automatiquement à extraire les meilleures caractéristiques des données brutes.

- **Quantité de données:** Le Deep Learning nécessite généralement beaucoup plus de données pour fonctionner efficacement, alors que les algorithmes de Machine Learning traditionnels peuvent être appliqués à des ensembles de données plus petits.
- **Complexité des modèles:** Les modèles de Deep Learning, composés de nombreuses couches de neurones, sont bien plus complexes et peuvent capturer des relations non linéaires plus profondes que les modèles de Machine Learning classiques.

Ainsi, le Deep Learning permet de traiter des problèmes complexes nécessitant un apprentissage à partir de données massives et non structurées, offrant des performances exceptionnelles dans des domaines comme la vision par ordinateur et le traitement du langage naturel.

1.5 Conclusion

Ce chapitre a posé les bases nécessaires à la compréhension du projet en détaillant son contexte, ses objectifs, et les concepts théoriques associés. Les éléments présentés permettront de mieux appréhender les étapes suivantes de l'analyse et de la modélisation des données dans les prochains chapitres.

Chapter

2

Outils mathématiques d'ordre général

Contents

2.1	Introduction	18
2.2	Techniques de Prétraitement des Données	18
2.2.1	Normalisation des Données	18
2.2.2	Traitement des Valeurs Manquantes	20
2.2.3	Conclusion	21
2.3	Les indicateurs statistiques	21
2.3.1	Eta Carré	21
2.3.2	Cramer V	21
2.3.3	Lambda de Goodman et Kruskal	22
2.3.4	Covariance	22
2.3.5	Limitation des Indicateurs Statistiques	23
2.3.6	Conclusion	23
2.4	Les Tests de corrélation	23
2.4.1	Tests paramétriques	23
2.4.2	Tests non paramétriques	28
2.5	Équilibrage des Classes avec SMOTE	31
2.6	Sélection des Caractéristiques (Feature Selection)	32
2.6.1	RFE (Recursive Feature Elimination)	32
2.6.2	SelectKBest	33
2.6.3	Feature Importance avec Forêt Aléatoire	33
2.6.4	Conclusion	33
2.7	Modèles de Machine Learning	34
2.7.1	Régression Logistique	34
2.7.2	L'Arbre de Décision	37
2.7.3	La Forêt Aléatoire: un modèle non paramétrique	38
2.7.4	AdaBoost: Amélioration des Arbres de Décision	39
2.7.5	Conclusion	41
2.8	Deep Learning	41

2.8.1	Les Couches dans un Réseau de Neurones	41
2.8.2	Fonctions d'Activation	41
2.8.3	Entraînement et Backpropagation	42
2.8.4	Régularisation et Dropout	43
2.8.5	Hyperparamètres: Signification et Impact	43
2.8.6	Conclusion	43
2.9	Interprétation des Modèles pour la Prédiction de la Satisfaction Client	44
2.9.1	Clustering avec KMeans	44
2.9.2	Calcul de la Probabilité de Satisfaction	44
2.9.3	Interprétation des Résultats	44
2.9.4	Conclusion	44
2.10	Conclusion	45

2.1 Introduction

Ce chapitre couvre les techniques de prétraitement, les tests de corrélation et la modélisation. Nous abordons des méthodes comme la normalisation et le traitement des valeurs manquantes, puis explorons des outils statistiques pour analyser les relations entre variables. Enfin, nous présentons des techniques de modélisation, incluant l'équilibrage des classes et la sélection de caractéristiques, qui améliorent les performances des modèles prédictifs.

2.2 Techniques de Prétraitement des Données

Le prétraitement des données est une étape cruciale pour améliorer la qualité des données et optimiser les performances des modèles. Il inclut des méthodes comme la normalisation, qui ajuste l'échelle des variables, et le traitement des valeurs manquantes.

2.2.1 Normalisation des Données

La normalisation met toutes les variables sur une même échelle, ce qui est essentiel pour éviter qu'une variable domine en raison de son amplitude. C'est particulièrement utile pour les algorithmes sensibles à l'échelle des données.

2.2.1.1 Méthodes classiques de normalisation

✓ Normalisation Min-Max

Elle met les données à l'échelle entre 0 et 1 selon la formule suivante:

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

→ **Utilité:** Recommandée pour les algorithmes comme les réseaux de neurones.

→ **Impact:** Sensible aux valeurs extrêmes (outliers).

✓ Z-Score (Standardisation)

Cette méthode centre les données à 0 avec un écart-type de 1 selon la formule suivante:

$$X_{\text{std}} = \frac{X - \mu}{\sigma}$$

→ **Utilité:** Utile pour ajuster les échelles sans affecter la structure des variables, peu importe la distribution.

→ **Impact:** Uniformise l'effet des variables sur le modèle.

✓ Normalisation Logarithmique (Log)

La transformation logarithmique réduit l'influence des grandes valeurs et rend les distributions plus symétriques selon la formule suivante:

$$X_{\log} = \log(X + 1)$$

→ **Utilité:** Efficace pour réduire les écarts dus aux valeurs extrêmes.

→ **Impact:** Diminue l'influence des grandes valeurs.

✓ Transformation par Racine Carrée (Sqrt)

Cette transformation réduit la variance tout en préservant la hiérarchie des données. Formule :

$$X_{\text{sqrt}} = \sqrt{X}$$

→ **Utilité:** Réduit l'impact des grandes valeurs sans affecter l'ordre des données.

→ **Impact:** Atténue les grandes variations tout en maintenant la structure des données.

2.2.1.2 Méthodes avancées de normalisation

Pour améliorer la qualité des données, surtout lorsque celles-ci ne suivent pas une distribution normale, des techniques avancées comme la transformation de Box-Cox et la transformation de Yeo-Johnson sont souvent employées.

✓ Transformation de Box-Cox

La transformation de Box-Cox [1] stabilise la variance et rapproche les données d'une distribution normale. Elle est définie par :

$$y(\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \text{si } \lambda \neq 0 \\ \log(x), & \text{si } \lambda = 0 \end{cases}$$

→ **Explication:** Box-Cox applique une transformation exponentielle ou logarithmique aux données, selon la valeur de λ . Lorsque λ est différent de zéro, la transformation lisse les données et réduit l'asymétrie en modifiant la relation entre les valeurs observées et la moyenne. Si $\lambda = 0$, elle applique une transformation logarithmique pour atténuer l'impact des grandes valeurs tout en rendant la distribution plus symétrique. Nous approfondirons la notion de **log-vraisemblance** dans une section ultérieure.

→ **Utilité:** Utile pour les données strictement positives, permet d'uniformiser les distributions non normales.

→ **Choix de λ :** Le paramètre λ est choisi par maximisation de la log-vraisemblance.

✓ Transformation de Yeo-Johnson

La transformation de Yeo-Johnson [2] est une généralisation de Box-Cox qui permet de traiter à la fois les données positives et négatives. Elle est définie par :

$$y(\lambda) = \begin{cases} \frac{(x+1)^\lambda - 1}{\lambda}, & \text{si } \lambda \neq 0 \text{ et } x \geq 0 \\ \log(x+1), & \text{si } \lambda = 0 \text{ et } x \geq 0 \\ -\frac{(-x+1)^{2-\lambda} - 1}{2-\lambda}, & \text{si } \lambda \neq 2 \text{ et } x < 0 \\ -\log(-x+1), & \text{si } \lambda = 2 \text{ et } x < 0 \end{cases}$$

→ **Utilité:** Permet de traiter les données négatives et positives, rendant les données plus normalisées.

→ **Choix de λ :** Comme pour Box-Cox, λ est déterminé par maximisation de la log-vraisemblance.

2.2.2 Traitement des Valeurs Manquantes

Le traitement des valeurs manquantes est crucial pour maintenir la qualité des modèles prédictifs. Il permet d'éviter les biais introduits par des données incomplètes en remplaçant les valeurs manquantes par des estimations appropriées.

2.2.2.1 Imputation par la Médiane

L'imputation par la médiane est une méthode simple qui remplace les valeurs manquantes par la médiane de la variable concernée. Elle est particulièrement efficace pour les données avec des valeurs extrêmes (outliers), car la médiane n'est pas influencée par ces dernières.

$$x_{i,\text{missing}} = \text{Médiane}(x_i)$$

→ **Utilité:** Adaptée aux distributions asymétriques.

→ **Impact:** Remplace les valeurs manquantes sans perturber la distribution des données.

2.2.2.2 K-Nearest Neighbors pour la Prédiction des Valeurs Manquantes

Le K-Nearest Neighbors (KNN) repose sur la similarité des données pour imputer les valeurs manquantes. Il identifie les k voisins les plus proches de l'observation avec la valeur manquante, en se basant sur une mesure de distance. Parmi ces mesures, la distance de Minkowski est une généralisation couramment utilisée.

✓ Étape 1: Calcul de la distance de Minkowski:

La distance de Minkowski [3] est définie par :

$$d(x_i, x_j) = \left(\sum_{k=1}^n |x_{ik} - x_{jk}|^p \right)^{1/p}$$

Cas particuliers:

→ Si $p = 2$, on obtient la **distance euclidienne** [4], utilisée pour les données continues.

→ Si $p = 1$, on obtient la **distance de Manhattan**, adaptée aux données avec de grandes variations.

→ Pour d'autres valeurs de p , la **distance de Minkowski** offre une flexibilité accrue pour ajuster la mesure de similarité.

→ **Utilité:** Chaque type de distance est choisi en fonction de la nature des données.

→ **Impact:** Le choix de la distance influence les voisins sélectionnés, et donc la précision de l'imputation.

✓ Étape 2: Moyenne pondérée des voisins:

Une fois les voisins identifiés, la valeur manquante est imputée par une moyenne pondérée de leurs valeurs :

$$x_{i,\text{missing}} = \frac{\sum_{j \in N(i)} w_j x_j}{\sum_{j \in N(i)} w_j}$$

où w_j est un poids inversement proportionnel à la distance $d(x_i, x_j)$.

- **Utilité:** Produit des estimations plus précises que les méthodes simples.
- **Impact:** Les valeurs imputées sont souvent plus proches des vraies données manquantes, augmentant la précision des modèles.

2.2.3 Conclusion

Le prétraitement, notamment la normalisation et le traitement des valeurs manquantes, est essentiel pour garantir des données comparables et fiables, améliorant ainsi les performances des modèles prédictifs.

2.3 Les indicateurs statistiques

Les **indicateurs statistiques** résument les données et permettent de mesurer les relations entre variables. Parmi eux, on trouve:

- **Tendance centrale :** Moyenne, médiane, mode indiquent où se concentrent les valeurs.
- **Dispersion :** Variance, écart-type et étendue quantifient la variabilité autour de la tendance centrale.
- **Relation :** Covariance et coefficient de corrélation mesurent les relations linéaires entre variables quantitatives.
- **Association :** L'Eta carré et le Cramer V évaluent l'association entre variables.

2.3.1 Eta Carré

L'Eta carré (η^2) quantifie la part de variance d'une variable quantitative expliquée par une variable qualitative. Il mesure l'effet d'un facteur.

$$\eta^2 = \frac{SSB}{SSB + SSW}$$

Avec :

- **SSB :** Somme des Carrés Entre les Groupes, mesurant la différence entre les moyennes des groupes et la moyenne générale.
- **SSW :** Somme des Carrés Intra-Groupes, mesurant la variabilité à l'intérieur de chaque groupe.

✓ **Interprétation:**

- $\eta^2 \leq 0.01$: Effet faible.
- $0.01 < \eta^2 \leq 0.06$: Effet modéré.
- $\eta^2 > 0.06$: Effet fort.

2.3.2 Cramer V

Le **Cramer V** mesure la force de l'association entre deux variables qualitatives, basé sur la statistique du χ^2 . Il est souvent utilisé avec un **tableau de contingence**, qui montre la fréquence des combinaisons possibles entre deux variables catégorielles. On détaillera ce tableau et le calcul du χ^2 dans une prochaine section.

$$V = \sqrt{\frac{X^2}{N \times \min(k-1, r-1)}}$$

Où:

- X^2 est la statistique du χ^2 ,
- N est le nombre total d'observations,
- k et r sont respectivement le nombre de colonnes et de lignes dans le tableau de contingence.

✓ **Interprétation:**

- $V \leq 0.10$: Association faible.
- $0.10 < V \leq 0.30$: Association modérée.
- $V > 0.30$: Forte association.

2.3.3 Lambda de Goodman et Kruskal

Le **Lambda** (λ) de Goodman et Kruskal [5] mesure l'amélioration de la prédiction d'une variable qualitative dépendante en fonction d'une variable qualitative indépendante. Il compare les erreurs de prédiction avec et sans la variable indépendante.

$$\lambda = \frac{E_0 - E_1}{E_0}$$

Où:

- $E_0 = N - \max_j(n_j)$ est le nombre d'erreurs si l'on prédit toujours la modalité la plus fréquente de la variable dépendante (erreurs sans utiliser la variable indépendante),
- $E_1 = \sum_i (n_i - \max_j(n_{ij}))$ est le nombre d'erreurs en utilisant la variable indépendante pour prédire la variable dépendante.

✓ **Interprétation:**

- $\lambda \leq 0.10$: Amélioration faible.
- $0.10 < \lambda \leq 0.30$: Amélioration modérée.
- $\lambda > 0.30$: Amélioration significative.

2.3.4 Covariance

La **covariance** mesure la variation conjointe de deux variables quantitatives, indiquant si elles augmentent ou diminuent ensemble. Elle est calculée par la somme des produits des écarts par rapport à leurs moyennes:

$$\text{Cov}(X, Y) = \frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})(Y_i - \bar{Y})$$

Où:

- X_i et Y_i sont les valeurs des variables pour l'observation i ,
- \bar{X} et \bar{Y} sont les moyennes des variables X et Y ,
- N est le nombre d'observations.

✓ Interprétation:

- Covariance positive: Les variables augmentent ensemble.
- Covariance proche de zéro: Relation faible.
- Covariance négative: Quand une variable augmente, l'autre diminue.

2.3.5 Limitation des Indicateurs Statistiques

- **Cramer V:** Sensible à la taille de l'échantillon.
- **Covariance:** Dépendante des unités de mesure.
- **Goodman et Kruskal Lambda:** Insensible aux petites variations.
- **Eta Carré:** Ne mesure que la force de l'association.

2.3.6 Conclusion

En résumé, les indicateurs comme le Cramer V, le Lambda de Goodman et Kruskal, la covariance et l'Eta carré permettent d'analyser les relations entre variables. Cependant, leurs limites, telles que la sensibilité à l'échelle des données et à la taille de l'échantillon, doivent être considérées pour garantir une interprétation fiable.

2.4 Les Tests de corrélation

Les tests de corrélation évaluent la relation entre deux variables. Ils se divisent en tests paramétriques, comme Pearson [9] et ANOVA, qui supposent la normalité des données, et en tests non paramétriques, comme Mann-Whitney [10] et Spearman, qui sont plus flexibles en l'absence de ces conditions.

Types de Corrélation

- **Corrélation Positive:** Les deux variables augmentent ou diminuent ensemble.
- **Corrélation Négative:** L'augmentation d'une variable entraîne la diminution de l'autre.
- **Corrélation Nulle:** Aucune relation linéaire apparente entre les variables.

Limites de la Corrélation

La corrélation est sensible aux valeurs aberrantes, ne traite que les relations linéaires et ne détecte pas les relations complexes.

2.4.1 Tests paramétriques

Les tests paramétriques, plus puissants que les non-paramétriques, détectent mieux les effets réels et augmentent la probabilité de rejeter l'hypothèse nulle (H_0) lorsqu'une différence existe, surtout lorsque les conditions sont respectées, conduisant souvent à une p-value plus faible.

2.4.1.1 Les conditions pour appliquer les tests paramétriques

Vérification de la normalité des données:

La vérification de la normalité est essentielle avant d'appliquer des tests paramétriques. Le test de Shapiro-Wilk [6] utilise la statistique W pour évaluer si un échantillon suit une loi normale. Si la p-value associée à W est inférieure à 0,05, l'hypothèse de normalité est rejetée.

✓ Hypothèses du Test:

- H_0 L'échantillon suit une loi normale.
- H_1 L'échantillon ne suit pas une loi normale.

✓ **Explication Mathématique:** Le test de Shapiro-Wilk compare les statistiques d'ordre observées à celles attendues sous la loi normale. La statistique W est calculée comme suit:

$$W = \frac{(\sum_{i=1}^n a_i x_i)^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Où:

- x_i est la i -ème statistique d'ordre,
- \bar{x} est la moyenne de l'échantillon,
- n est le nombre d'observations,
- $a_i = \frac{m^T V^{-1}}{\sqrt{m^T V^{-1} V^{-1} m}}$, où m est le vecteur des quantiles théoriques et V la covariance.

✓ Interprétation:

- **Valeur de W :** Plus proche de 1, meilleure est l'adéquation.
- **p-value:** Si $p < 0,05$, les données ne sont pas normales; sinon, elles le sont.

Homoscédasticité (ou égalité des variances):

L'homoscédasticité est cruciale avant d'appliquer des tests paramétriques. Le test de Levene [7] compare les déviations absolues par rapport à la médiane pour évaluer l'égalité des variances entre groupes.

✓ Hypothèses du Test:

- H_0 : Les variances des groupes sont égales (homoscédasticité).
- H_1 : Les variances des groupes sont différentes (hétéroscédasticité).

✓ Calcul des Déviations Absolues:

Pour chaque observation x_{ij} dans le groupe j :

$$d_{ij} = |x_{ij} - \tilde{x}_j|$$

Où \tilde{x}_j est la médiane du groupe j .

✓ ANOVA sur les Déviations Absolues:

La statistique F , qui compare la variance entre les groupes à celle intra-groupe, est donnée par:

$$F = \frac{\text{Variance entre les groupes}}{\text{Variance intra-groupe}}$$

Où:

$$\text{Variance entre les groupes} = \frac{\sum_{j=1}^k n_j (\bar{d}_j - \bar{d})^2}{k-1}$$

$$\text{Variance intra-groupe} = \frac{\sum_{j=1}^k \sum_{i=1}^{n_j} (d_{ij} - \bar{d}_j)^2}{N-k}$$

✓ Interprétation des Résultats:

- F élevé indique des différences de variances entre les groupes.
- $p\text{-value} < 0,05$: Rejet de H_0 , les variances sont différentes.
- $p\text{-value} \geq 0,05$: H_0 n'est pas rejetée, les variances sont homogènes.

Autres conditions pour l'application des Tests paramétriques:

✓ Indépendance des observations: Chaque observation doit être indépendante, sans influence d'une observation sur une autre.

✓ Linéarité des relations: Une relation est linéaire si elle suit une équation $Y = aX + b$.

→ **Méthode des moindres carrés:** Pour vérifier la linéarité entre X et Y , on ajuste un modèle de régression linéaire:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

Les coefficients β_0 et β_1 sont déterminés en minimisant la somme des carrés des résidus $S(\beta_0, \beta_1)$:

$$S(\beta_0, \beta_1) = \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2$$

→ **Dérivation pour minimisation:** Les dérivées partielles de S sont calculées pour minimiser cette fonction:

$$\frac{\partial S}{\partial \beta_0} = -2 \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))$$

$$\frac{\partial S}{\partial \beta_1} = -2 \sum_{i=1}^n X_i (Y_i - (\beta_0 + \beta_1 X_i))$$

→ **Résolution des équations normales:**

$$\beta_1 = \frac{n \sum_{i=1}^n X_i Y_i - \sum_{i=1}^n X_i \sum_{i=1}^n Y_i}{n \sum_{i=1}^n X_i^2 - (\sum_{i=1}^n X_i)^2}$$

$$\beta_0 = \frac{1}{n} \sum_{i=1}^n Y_i - \beta_1 \frac{1}{n} \sum_{i=1}^n X_i$$

→ **Coefficient de détermination R^2** : Il mesure la proportion de la variance expliquée par le modèle:

$$R^2 = 1 - \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$$

→ **Interprétation de R^2** :

- $R^2 = 1$: Relation linéaire parfaite.
- $R^2 = 0$: Aucune relation linéaire.

2.4.1.2 Test t de Student

Le test t (ou test de Student) [8] est utilisé pour comparer les moyennes de deux groupes ou d'un groupe par rapport à une valeur standard. Il vise à déterminer si les différences entre les moyennes sont statistiquement significatives ou dues au hasard.

✓ **Conditions d'application du test t:**

→ **Normalité**: Les données doivent suivre une distribution normale.

→ **Homogénéité**: Les variances des groupes doivent être égales.

→ **Variables**: Comparaison d'une variable quantitative continue et d'une variable qualitative à deux groupes.

✓ **Hypothèses du test:**

→ H_0 : Les moyennes des deux groupes sont égales.

→ H_1 : Les moyennes des deux groupes sont différentes.

✓ **Formule du test t pour échantillons indépendants:**

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Où:

- \bar{X}_1 et \bar{X}_2 sont les moyennes des deux groupes.
- s_1^2 et s_2^2 sont les variances des deux groupes.
- n_1 et n_2 sont les tailles des échantillons.

✓ **Interprétation du test t:**

→ **p-value < 0,05**: Les différences entre les moyennes sont statistiquement significatives. On rejette H_0 et conclut que les moyennes sont différentes.

→ **p-value ≥ 0,05**: Les différences ne sont pas significatives. On ne rejette pas H_0 et conclut qu'il n'y a pas suffisamment de preuves pour dire que les moyennes diffèrent.

2.4.1.3 Test ANOVA

L'ANOVA (Analyse de Variance) est un test statistique permettant de comparer les moyennes de trois groupes ou plus. Contrairement au test de Levene, qui est utilisé pour tester l'homoscédasticité (égalité des variances) entre les groupes, l'ANOVA se concentre sur la comparaison des moyennes des groupes pour déterminer s'il existe des différences significatives.

✓ Conditions d'application du test ANOVA:

- **Normalité:** Les données dans chaque groupe doivent suivre une distribution normale.
- **Homogénéité des variances:** Les variances des groupes doivent être égales (homoscédasticité), ce que nous avons testé avec le test de Levene dans la section précédente.
- **Variables:** Le test compare une variable quantitative continue (la réponse) avec une variable qualitative (les groupes).

✓ Hypothèses du test:

- H_0 : Les moyennes des groupes sont égales.
- H_1 : Au moins une des moyennes des groupes est différente.

✓ Formule du test ANOVA:

Nous avons déjà utilisé des formules similaires pour l'ANOVA appliquée aux déviations absolues dans le test de Levene pour évaluer l'homogénéité des variances. Dans ce contexte, l'ANOVA est utilisé pour comparer les moyennes des groupes en utilisant la même structure de calcul pour la statistique F :

$$F = \frac{\text{Variance inter-groupes}}{\text{Variance intra-groupes}}$$

Les formules spécifiques de la variance inter-groupes et intra-groupes ont déjà été introduites, mais ici elles servent à évaluer la différence **des moyennes** entre les groupes plutôt que leurs variances.

✓ Interprétation des résultats:

- **Si F est supérieur à la valeur critique:** On rejette H_0 , ce qui indique qu'il existe une différence significative entre les moyennes des groupes.
- **Si F est inférieur ou égal à la valeur critique:** On ne rejette pas H_0 , ce qui indique qu'il n'y a pas de différence significative entre les moyennes des groupes.

Ainsi, bien que l'ANOVA puisse être utilisé à la fois pour tester les variances (comme dans le test de Levene) et pour comparer les moyennes, son objectif principal dans cette section est d'examiner les différences de moyennes entre les groupes.

2.4.1.4 Test de Pearson

Le test de corrélation de Pearson est un test statistique utilisé pour mesurer la force et la direction de la relation linéaire entre deux variables quantitatives continues. Le coefficient de corrélation de Pearson, noté r , varie entre -1 et 1.

✓ **Conditions d'application du test de Pearson:**

- **Normalité:** Les deux variables doivent suivre une distribution normale.
- **Linéarité:** La relation entre les deux variables doit être linéaire.
- **Homogénéité (homoscédasticité):** La variance des points doit être constante autour de la droite de régression.

✓ **Hypothèses du test de Pearson:**

- H_0 : Il n'y a pas de corrélation linéaire entre les deux variables ($r = 0$).
- H_1 : Il existe une corrélation linéaire significative entre les deux variables ($r \neq 0$).

✓ **Formule du coefficient de corrélation de Pearson:**

Le coefficient r est calculé à partir de la covariance des deux variables, divisée par le produit de leurs écarts-types:

$$r = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

Où:

- X_i et Y_i sont les valeurs des variables X et Y .
- \bar{X} et \bar{Y} sont les moyennes de X et Y .
- n est le nombre total d'observations.

✓ **Interprétation du coefficient r :**

- **Si r est proche de 1 ou -1:** Cela indique une forte corrélation positive (1) ou négative (-1).
- **Si r est proche de 0:** Cela suggère qu'il n'y a pas de relation linéaire significative entre les deux variables.

✓ **Signification statistique du test de Pearson:**

La statistique r est comparée à une valeur critique, déterminée à partir de la distribution t avec $n - 2$ degrés de liberté. Si la p-value associée est inférieure à 0,05, on rejette l'hypothèse nulle et on conclut qu'il existe une corrélation linéaire significative entre les deux variables.

2.4.2 Tests non paramétriques

Les **tests non paramétriques** sont utilisés lorsque les conditions des tests paramétriques ne sont pas respectées. Ils ne reposent pas sur des hypothèses strictes sur la distribution des données et sont adaptés aux petites tailles d'échantillon et aux données ordinaires ou continues non normales.

2.4.2.1 Test U de Mann-Whitney

Le test U de Mann-Whitney est une alternative au test t de Student pour comparer deux groupes indépendants, lorsque les données ne suivent pas une distribution normale. Il se base sur **les rangs** des observations.

✓ **Conditions d'application:**

- Les données doivent être ordinaires ou continues.
- Les groupes doivent être indépendants et les distributions non normales.

✓ **Principe des rangs:**

Les valeurs des deux groupes sont combinées et classées par ordre croissant. Chaque observation reçoit un rang correspondant à sa position.

✓ **Formule du test U:**

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1, \quad U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

Où n_1 et n_2 sont les tailles des groupes, et R_1 et R_2 les sommes des rangs.

La statistique U est le minimum de U_1 et U_2 .

✓ **Interprétation:**

→ Si les rangs d'un groupe sont globalement plus élevés ou plus bas que ceux de l'autre, cela suggère une différence entre les groupes.

→ **p-value** < 0,05: différence significative.

→ **p-value** ≥ 0,05: pas de différence significative.

2.4.2.2 Test de Kruskal-Wallis

Le test de Kruskal-Wallis est utilisé pour comparer trois groupes ou plus lorsque les données ne suivent pas une distribution normale. Il est une alternative à l'ANOVA.

✓ **Formule du test H:**

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1)$$

Où:

- N est le nombre total d'observations,
- R_i est la somme des rangs pour le groupe i ,
- n_i est la taille du groupe i .

✓ **Interprétation du test H:**

→ Si H est élevé, cela suggère des différences entre les groupes.

→ Si la p-value < 0,05, les différences sont significatives.

2.4.2.3 Test de Spearman

Le test de Spearman [11] mesure la force et la direction de la relation monotone entre deux variables ordinaires ou continues non normales. Il est une alternative au test de Pearson pour les données non paramétriques.

✓ **Formule du coefficient de Spearman:**

$$r_s = \frac{\sum_{i=1}^n (R_{x_i} - \frac{N+1}{2})(R_{y_i} - \frac{N+1}{2})}{\sqrt{\sum_{i=1}^n (R_{x_i} - \frac{N+1}{2})^2 \sum_{i=1}^n (R_{y_i} - \frac{N+1}{2})^2}}$$

Où:

- R_{x_i} et R_{y_i} sont les rangs des valeurs X et Y ,
- N est le nombre total d'observations.

✓ **Interprétation du coefficient de Spearman:**

- Si r_s est proche de 1 ou -1, la corrélation est forte.
- Si r_s est proche de 0, il n'y a pas de relation monotone.

2.4.2.4 Test du Chi-Deux

Le test du Chi-Deux est utilisé pour évaluer si les distributions observées dans des données catégorielles diffèrent des distributions attendues sous l'hypothèse d'indépendance entre les variables.

✓ **Calcul des fréquences attendues:**

Les fréquences attendues E_{ij} dans chaque cellule d'un tableau de contingence sont calculées en utilisant la formule:

$$E_{ij} = \frac{R_i \times C_j}{N}$$

Où:

- R_i est le total de la i -ème ligne.
- C_j est le total de la j -ème colonne.
- N est le nombre total d'observations.

✓ **Formule du test du Chi-Deux:**

La statistique X^2 est calculée en comparant les fréquences observées aux fréquences attendues:

$$X^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Où:

- O_{ij} est la fréquence observée dans la cellule (i, j) ,
- E_{ij} est la fréquence attendue dans la cellule (i, j) .

✓ **Interprétation du test du Chi-Deux:**

- **Si la p-value < 0,05:** Il existe une association significative entre les variables catégorielles.
- **Si la p-value ≥ 0,05:** Les variables sont indépendantes, c'est-à-dire qu'il n'y a pas de relation statistiquement significative entre elles.

2.4.2.5 Test exact de Fisher

Le test exact de Fisher [12] est utilisé pour évaluer l'indépendance entre deux variables dans un tableau de contingence 2x2, particulièrement utile lorsque les effectifs sont petits ou les fréquences observées faibles.

✓ **Explication mathématique:**

Le test évalue la probabilité d'obtenir une répartition des données aussi extrême ou plus extrême que celle observée, sous l'hypothèse nulle d'indépendance.

Prenons un tableau de contingence 2x2:

	Groupe 1	Groupe 2
Catégorie A	a	b
Catégorie B	c	d

Les probabilités des différentes configurations possibles sont calculées en respectant les totaux des lignes et colonnes (totaux marginaux) avec la distribution hypergéométrique:

$$P = \frac{\binom{a+b}{a} \binom{c+d}{c}}{\binom{N}{a+c}}$$

✓ **Interprétation de la p-valeur:**

- Si la p-value $< 0,05$, on rejette l'hypothèse nulle d'indépendance.
- Si la p-value $\geq 0,05$, les données sont compatibles avec l'indépendance entre les variables.

2.5 Équilibrage des Classes avec SMOTE

Dans le contexte de l'apprentissage supervisé, l'équilibrage des classes est crucial lorsqu'on traite des jeux de données déséquilibrés, c'est-à-dire lorsque certaines classes, souvent la classe minoritaire, sont sous-représentées. Le **SMOTE** (Synthetic Minority Over-sampling Technique) est une méthode avancée qui génère artificiellement de nouveaux exemples synthétiques pour la classe minoritaire, afin de rééquilibrer les données.

Le principe de SMOTE repose sur la génération de nouveaux points synthétiques en utilisant les données existantes des classes minoritaires. Le processus se fait comme suit:

→ **Étape 1:** Pour chaque exemple minoritaire x_i , sélectionner un ou plusieurs k -voisins x_j dans la classe minoritaire, où j est un indice de voisin sélectionné parmi les plus proches voisins.

→ **Étape 2:** Générer un nouveau point x_{new} qui est situé quelque part sur la ligne reliant x_i et x_j , selon la formule suivante:

$$x_{\text{new}} = x_i + \delta \times (x_j - x_i)$$

où x_i et x_j sont les points dans l'espace des features, et δ est un facteur aléatoire compris entre 0 et 1.

→ **Étape 3:** Répéter ce processus pour plusieurs k -voisins afin de générer plusieurs points synthétiques, augmentant ainsi le nombre d'exemples dans la classe minoritaire.

✓ **Interprétation:**

→ **Utilité:** SMOTE permet de combler l'écart entre des exemples existants de la classe minoritaire

et leurs voisins, créant ainsi des points synthétiques qui enrichissent la classe minoritaire tout en conservant les propriétés de l'espace des features.

→ **Impact:** Cette technique réduit les biais introduits par les classes déséquilibrées et améliore la performance des algorithmes de classification, notamment pour les classes minoritaires.

✓ **Détail du Calcul:**

→ x_i : Un point dans la classe minoritaire.

→ x_j : Un voisin k -proche de x_i dans la même classe minoritaire.

→ δ : Un nombre aléatoire entre 0 et 1, qui détermine la position du nouveau point sur la ligne reliant x_i et x_j . Si $\delta = 0.5$, x_{new} se situe au milieu de x_i et x_j , tandis que des valeurs plus proches de 0 ou de 1 rapprochent x_{new} de x_i ou x_j .

Cette méthode permet de créer de nouvelles observations synthétiques en comblant l'écart entre des points minoritaires et leurs voisins. Cela évite les biais liés aux données déséquilibrées sans simplement dupliquer les exemples existants, ce qui améliore les performances des modèles sans risquer de surapprentissage.

2.6 Sélection des Caractéristiques (Feature Selection)

La sélection des caractéristiques est une étape essentielle dans la construction de modèles de machine learning. Elle permet de réduire la dimensionnalité, d'améliorer la performance des modèles et de faciliter l'interprétation.

2.6.1 RFE (Recursive Feature Elimination)

L'algorithme **RFE** (Élimination Réursive des Caractéristiques) sélectionne les caractéristiques les plus importantes en entraînant un modèle, en attribuant un poids à chaque caractéristique, puis en éliminant récursivement celles avec les poids les plus faibles.

✓ **Étape 1: Entraînement du modèle de base**

Un modèle (comme une régression linéaire ou une forêt aléatoire) est entraîné sur l'ensemble des caractéristiques. Chaque caractéristique se voit attribuer un **poids** basé sur son influence sur les prédictions.

✓ **Étape 2: Élimination des caractéristiques moins importantes**

Les caractéristiques ayant des poids proches de zéro ou jugées peu influentes sont éliminées.

✓ **Étape 3: Répétition du processus**

L'algorithme répète le processus jusqu'à ce qu'il ne reste qu'un sous-ensemble optimal de caractéristiques.

Contrairement à la méthode des **Forêts Aléatoires** qui évalue l'importance via l'impureté des nœuds, **RFE** élimine les caractéristiques en se basant sur l'importance globale donnée par le modèle choisi. Cette approche est complémentaire à l'analyse d'importance, permettant d'affiner encore davantage le choix des caractéristiques.

→ **Utilité:** RFE est utile pour réduire la dimensionnalité tout en conservant les caractéristiques les plus influentes.

→ **Impact:** Cette méthode aide à éliminer les variables redondantes ou non pertinentes qui pourraient dégrader les performances des modèles.

2.6.2 SelectKBest

La méthode **SelectKBest** sélectionne les k meilleures caractéristiques en fonction d'une mesure statistique. Un test statistique couramment utilisé est le test F, qui mesure le rapport entre la variance inter-classes et la variance intra-classes:

$$F = \frac{\text{Variance entre les classes}}{\text{Variance intra-classes}}$$

→ **Utilité:** Cette méthode est particulièrement utile pour sélectionner les caractéristiques ayant un fort pouvoir discriminant dans les modèles supervisés.

→ **Impact:** La sélection des caractéristiques en fonction du test F permet d'optimiser la performance des modèles en se concentrant sur les variables ayant une forte influence sur la variable cible.

2.6.3 Feature Importance avec Forêt Aléatoire

La **forêt aléatoire** est un ensemble d'arbres de décision qui attribue une importance à chaque caractéristique en fonction de la réduction d'impureté, mesurée par l'indice de Gini, aux nœuds de décision où la caractéristique est utilisée. L'importance d'une caractéristique est calculée en fonction de la somme de la réduction d'impureté avant et après la division.

La réduction de l'impureté, aussi appelée **réduction de Gini**, est définie par la formule:

$$\Delta\text{Gini}(f) = \sum (\text{Gini avant} - \text{Gini après})$$

→ **Utilité:** Cette méthode identifie les caractéristiques qui contribuent le plus à la séparation des classes dans un jeu de données complexe.

→ **Impact:** La forêt aléatoire est robuste aux interactions entre les caractéristiques, ce qui permet d'évaluer l'importance relative de chaque variable.

Le concept d'impureté et l'indice de Gini seront détaillés dans la partie consacrée aux *modèles d'arbres de décision* dans la suite de ce chapitre.

2.6.4 Conclusion

Les trois algorithmes de sélection – RFE, SelectKBest et Feature Importance avec Forêt Aléatoire – sont complémentaires. RFE élimine les caractéristiques moins pertinentes, SelectKBest identifie statistiquement les meilleures, et la Forêt Aléatoire évalue leur importance via la réduction d'impureté. Ensemble, ces méthodes optimisent la sélection des variables, améliorant la performance des modèles.

2.7 Modèles de Machine Learning

Les modèles de machine learning permettent d'entraîner des algorithmes pour prédire des résultats en fonction des données. Ces algorithmes optimisent leurs paramètres internes pour minimiser les erreurs de prédiction.

2.7.1 Régression Logistique

La régression logistique est un modèle paramétrique utilisé principalement pour les tâches de classification binaire. Son objectif est de prédire la probabilité qu'une observation appartienne à une classe spécifique.

✓ Concept Fondamental:

La régression logistique modélise la probabilité qu'un événement se produise à l'aide d'une fonction sigmoïde. La formule de la probabilité est donnée par:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k)}}$$

Où:

- P : Probabilité que l'observation appartienne à la classe 1.
- β_0 : Intercept (log-odds quand toutes les variables sont nulles).
- β_i : Coefficients associés aux variables explicatives x_i , qui déterminent l'importance de chaque variable.
- x_i : Variables explicatives, ou caractéristiques, utilisées pour prédire la probabilité.

Le but est d'ajuster les coefficients $\beta_0, \beta_1, \dots, \beta_k$ de manière à ce que le modèle prévoie correctement la classe à partir des données d'entrée.

✓ Fonction de Coût (ou Perte):

La fonction de coût dans un modèle de régression logistique est appelée **log-loss** ou **cross-entropy loss**. Elle mesure l'écart entre les prédictions du modèle et les vraies valeurs:

$$L(\beta) = - \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Où:

- N : Nombre total d'observations dans le jeu de données.
- y_i : Valeur réelle de la classe pour l'observation i (0 ou 1).
- p_i : Probabilité prédite par le modèle pour l'observation i .

La fonction de coût est essentielle, car elle guide l'algorithme sur la façon d'ajuster les coefficients du modèle pour mieux correspondre aux données observées.

✓ Optimisation et Descente de Gradient:

L'objectif de l'optimisation est de minimiser la fonction de coût en ajustant les coefficients $\beta_0, \beta_1, \dots, \beta_k$. Une méthode populaire pour cela est la **descente de gradient**, où les coefficients sont mis à jour en suivant le gradient de la fonction de coût (qui indique la direction et l'intensité des modifications à apporter).

La mise à jour des coefficients se fait ainsi:

$$\beta_i \leftarrow \beta_i - \alpha \frac{\partial L(\beta)}{\partial \beta_i}$$

Où:

- α est le **taux d'apprentissage**, qui contrôle l'ampleur des mises à jour des coefficients.
- $\frac{\partial L(\beta)}{\partial \beta_i}$ est le gradient de la fonction de coût par rapport à chaque coefficient β_i .

Le **taux d'apprentissage** (α) est un hyperparamètre important qui doit être bien choisi: un α trop grand peut entraîner des oscillations, tandis qu'un α trop petit rend l'apprentissage trop lent.

2.7.1.1 Hyperparamètres

Les **hyperparamètres** sont des variables définies avant l'entraînement du modèle. Contrairement aux coefficients β , qui sont appris à partir des données, les hyperparamètres influencent le processus d'optimisation et de convergence du modèle.

→ **Taux d'apprentissage (α):** Ce paramètre contrôle la taille des pas effectués pour ajuster les coefficients β .

→ **Solver:** Le solver est l'algorithme utilisé pour minimiser la fonction de coût. Différents solveurs sont disponibles en fonction du type de problème et des contraintes de calcul. Voici un aperçu des principaux solveurs utilisés :

- **Newton-CG:** Utilise la matrice Hessienne H , qui contient les dérivées secondes de la fonction de coût, pour calculer la direction de descente. La mise à jour des coefficients β est effectuée comme suit:

$$\beta \leftarrow \beta - H^{-1} \nabla L(\beta)$$

Où H^{-1} est l'inverse de la Hessienne et $\nabla L(\beta)$ est le gradient de la fonction de coût. Bien que précis, il peut être coûteux en temps pour les grands ensembles de données.

- **LBFGS** (Limited-memory Broyden–Fletcher–Goldfarb–Shanno): Version plus efficace de Newton-CG, LBFGS utilise les gradients récents au lieu de la matrice Hessienne complète pour estimer la direction de descente. La mise à jour est donnée par:

$$\beta \leftarrow \beta - \alpha \cdot \text{Direction}$$

LBFGS réduit la mémoire utilisée et accélère les calculs, ce qui le rend adapté pour les grands ensembles de données.

- **SAGA:** Améliore la descente de gradient stochastique (SGD) en combinant les gradients précédemment calculés pour affiner les mises à jour. La descente de gradient stochastique est une variante de la descente de gradient où les mises à jour des paramètres sont effectuées après chaque échantillon de données, plutôt que sur l'ensemble complet des données. Cela permet une convergence plus rapide, mais les mises à jour peuvent être plus bruyantes.

SAGA combine cette approche avec un ajustement basé sur l'accumulation des gradients passés, offrant une meilleure stabilité et précision. La mise à jour des coefficients se fait ainsi:

$$\beta \leftarrow \beta - \alpha \cdot \frac{1}{N} \sum_{i=1}^N \nabla L_i(\beta)$$

Où $\nabla L_i(\beta)$ est le gradient de la fonction de coût pour un échantillon i , et N est la taille de l'ensemble de données.

→ **Max_iter:** Définit le nombre maximal d'itérations de l'algorithme d'optimisation. Un nombre trop faible d'itérations peut entraîner une convergence incomplète, tandis qu'un nombre trop élevé peut augmenter le temps de calcul sans gains significatifs.

Ces hyperparamètres influencent la vitesse de convergence, la précision du modèle et la capacité d'adaptation aux données complexes.

✓ Régularisation:

La **régularisation** est un mécanisme qui ajoute une pénalité aux coefficients du modèle pour éviter le surajustement (overfitting). Il existe deux types principaux de régularisation:

→ **Régularisation L2 (Ridge):** Ajoute une pénalité proportionnelle à la somme des carrés des coefficients pour éviter que certains coefficients ne deviennent trop grands. Sa formule est:

$$L2 = \lambda \sum_{j=1}^k \beta_j^2$$

Où λ contrôle l'importance de cette pénalisation.

→ **Régularisation L1 (Lasso):** Pénalise la somme des valeurs absolues des coefficients, favorisant des modèles plus simples en mettant certains coefficients à zéro (ce qui peut exclure certaines variables inutiles):

$$L1 = \lambda \sum_{j=1}^k |\beta_j|$$

Le choix de la régularisation dépend de l'objectif: L1 aide à la sélection des variables, tandis que L2 contrôle la taille des coefficients pour éviter le surajustement.

✓ Résumé du Processus:

1. Initialisation des coefficients β_0, \dots, β_k .

2. Calcul de la fonction de coût en fonction des prédictions.
3. Utilisation de l'algorithme de descente de gradient pour ajuster les coefficients en minimisant la fonction de coût.
4. Application de la régularisation pendant l'optimisation pour éviter le surajustement.
5. Répéter jusqu'à atteindre la convergence ou le nombre maximal d'itérations.

2.7.2 L'Arbre de Décision

✓ Concept Fondamental:

Un arbre de décision classe des observations en les divisant en sous-groupes basés sur leurs caractéristiques. Chaque nœud interne pose une question, et les branches représentent les résultats des décisions, menant à des feuilles où la prédiction finale est réalisée.

✓ Structure de l'Arbre:

Nœuds internes: Posent des questions sur les caractéristiques.

Branches: Représentent les résultats des décisions.

Feuilles: Donnent la prédiction finale.

✓ Processus de Division:

L'arbre divise les données en sous-groupes aussi homogènes que possible, avec l'objectif de maximiser la pureté de chaque nœud.

✓ Critères de Division:

Les deux critères principaux pour mesurer la qualité des divisions sont l'Impureté de Gini et l'Entropie.

1. Impureté de Gini:

Mesure la pureté d'un nœud: un nœud est pur si toutes les observations appartiennent à la même classe. La formule est donnée par:

$$Gini(S) = 1 - \sum_{k=1}^K P_k^2$$

où P_k est la proportion d'éléments de la classe k . Une réduction de l'impureté de Gini est calculée après chaque division:

$$\Delta Gini = Gini(S) - Gini_{moy}$$

La meilleure division est celle qui maximise la réduction de l'impureté.

2. Entropie:

L'entropie mesure l'incertitude des classes au sein d'un nœud. Plus l'entropie est faible, plus le nœud est homogène.

$$Entropie(S) = - \sum_{k=1}^K P_k \log_2(P_k)$$

$$\Delta Entropie = Entropie(S) - Entropie_{moy}$$

Le critère retenu dépend de la priorité donnée à la pureté des nœuds ou à la réduction de l'incertitude.

Différence entre Gini et Entropie:

Bien que similaires, Gini privilégie la pureté des nœuds, tandis que l'entropie est plus sensible aux classes rares. Les deux critères sont souvent utilisés de manière interchangeable selon les besoins du modèle.

✓ Hyperparamètres:

Pour éviter le surapprentissage, plusieurs hyperparamètres doivent être ajustés:

→ **Profondeur de l'arbre:** Limite la taille de l'arbre pour éviter la sur-optimisation.

→ **Max_features:** Nombre de caractéristiques considérées à chaque division.

→ **min_samples_split** et **min_samples_leaf:** Limites pour diviser un nœud ou créer une feuille.

✓ Optimisation:

Des méthodes comme **RandomizedSearchCV** permettent de tester différentes combinaisons d'hyperparamètres pour obtenir le meilleur modèle. Contrairement à **GridSearchCV**, qui explore de manière exhaustive toutes les combinaisons possibles, **RandomizedSearchCV** sélectionne aléatoirement un sous-ensemble de ces combinaisons.

→ **Principe:** **RandomizedSearchCV** effectue un échantillonnage aléatoire des combinaisons d'hyperparamètres à tester. Au lieu de tester toutes les possibilités, il en sélectionne une fraction aléatoire, ce qui permet de trouver une solution optimale plus rapidement, particulièrement utile pour les grands jeux de données ou lorsque le nombre d'hyperparamètres à ajuster est important.

→ **Différence avec GridSearchCV:** **GridSearchCV** explore systématiquement chaque combinaison d'hyperparamètres, tandis que **RandomizedSearchCV** teste aléatoirement un certain nombre d'échantillons, ce qui permet une optimisation plus rapide.

Note: La fonction de coût et les détails sur la descente de gradient, souvent utilisés dans l'optimisation d'hyperparamètres, ont déjà été abordés dans une section précédente.

✓ Résumé du Processus:

L'arbre de décision divise les données en sous-groupes en fonction des caractéristiques, en utilisant les critères de Gini ou d'entropie pour maximiser la pureté à chaque division. Les hyperparamètres contrôlent la complexité du modèle, garantissant un bon équilibre entre précision et généralisation.

2.7.3 La Forêt Aléatoire: un modèle non paramétrique

✓ Concept Fondamental:

La forêt aléatoire est un ensemble d'arbres de décision, où plusieurs arbres sont créés à partir d'échantillons différents des données. Chaque arbre effectue sa propre prédiction, et les résultats sont combinés pour améliorer la précision globale. Ce processus réduit le surapprentissage et améliore la robustesse du modèle.

✓ Crédit des Arbres de Décision:

1. **Échantillonnage Bootstrap:** Chaque arbre est formé à partir d'un sous-échantillon aléatoire avec remplacement (bootstrap) des données d'entraînement. Cela permet de diversifier les arbres créés, réduisant le risque de surajustement (overfitting).
2. **Sélection Aléatoire des Caractéristiques:** À chaque nœud, une fraction aléatoire des caractéristiques est sélectionnée pour déterminer les divisions, ce qui diminue la corrélation entre les arbres et favorise une meilleure généralisation.
3. **Vote Majoritaire:** Chaque arbre vote pour une classe, et la classe finale prédite est celle qui reçoit le plus de votes. Cela est formalisé par l'équation suivante:

$$\hat{y} = \arg \max_k \left(\sum_{i=1}^T \mathbf{1}(\hat{y}_i = k) \right)$$

où T est le nombre total d'arbres et k les classes possibles.

✓ Optimisation des Hyperparamètres:

La performance d'une forêt aléatoire dépend fortement de la configuration de ses hyperparamètres. Voici quelques hyperparamètres clés à ajuster:

- **n_estimators:** Nombre d'arbres dans la forêt. Plus il est élevé, meilleure est la précision, mais cela augmente le temps de calcul.
- **max_features:** Nombre de caractéristiques à considérer pour chaque division. Une valeur courante est la racine carrée du nombre total de caractéristiques.
- **min_samples_split:** Nombre minimum d'échantillons requis pour diviser un nœud. Ajuster ce paramètre permet de contrôler la profondeur des arbres.
- **oob_score (out-of-bag score):** Ce score est calculé en utilisant les échantillons non sélectionnés par bootstrap pour entraîner un arbre. Il permet d'évaluer la performance du modèle sans avoir recours à une validation croisée, économisant ainsi du temps de calcul.

✓ Importance des Caractéristiques:

La forêt aléatoire permet également de mesurer l'importance des caractéristiques en évaluant leur contribution à la réduction de l'impureté (Gini ou entropie) aux nœuds de décision. Les détails sur le calcul de l'importance des caractéristiques ont déjà été abordés dans la section dédiée à la sélection des caractéristiques. Ici, nous rappelons que cette importance est calculée en fonction de la réduction d'impureté à chaque nœud où une caractéristique est utilisée.

✓ Relation avec l'Arbre de Décision:

Contrairement à un arbre de décision unique, la forêt aléatoire utilise plusieurs arbres créés à partir de différentes sous-parties des données et des caractéristiques. Cette approche réduit le surapprentissage, rendant le modèle plus robuste. L'agrégation des prédictions améliore la capacité du modèle à capturer la variabilité des données, produisant ainsi des résultats plus précis et plus généralisables.

2.7.4 AdaBoost: Amélioration des Arbres de Décision

✓ Concept Fondamental:

AdaBoost est une technique de **boosting** qui améliore les performances des arbres de décision en créant des modèles séquentiels. Contrairement à la forêt aléatoire qui utilise des arbres indépendants, le boosting corrige les erreurs des arbres précédents, rendant le modèle plus précis à chaque étape.

AdaBoost se concentre sur les observations mal classées en attribuant des poids plus importants à ces données après chaque itération. L'objectif est de forcer les arbres suivants à prêter plus d'attention à ces erreurs.

✓ **Explication du Processus:**

1. **Poids des observations:** Chaque observation a un poids. Initialement, toutes les observations ont le même poids.
2. **Ajustement des poids après chaque itération:** Les observations mal classées reçoivent des poids plus élevés afin que l'arbre suivant se concentre davantage sur ces erreurs.

La formule suivante détermine les poids des modèles faibles α_m selon les erreurs (ε_m):

$$\alpha_m = \frac{1}{2} \log \left(\frac{1 - \varepsilon_m}{\varepsilon_m} \right)$$

Les poids des observations sont ensuite mis à jour:

$$w_{i,m+1} = w_{i,m} \exp(-\alpha_m y_i h_m(x_i))$$

Cela permet aux arbres suivants de mieux corriger les erreurs des arbres précédents.

✓ **Utilité Pratique:**

AdaBoost est particulièrement utile lorsque l'on veut se concentrer sur les observations difficiles à classer. Il est efficace pour les ensembles de données où certaines erreurs sont difficiles à corriger avec des méthodes traditionnelles.

✓ **Hyperparamètres:**

- **n_estimators:** Nombre d'arbres. Un nombre élevé permet d'améliorer la précision, mais au prix d'un temps de calcul plus long.
- **learning_rate:** Il détermine la contribution de chaque nouvel arbre aux prédictions finales. Un faible taux d'apprentissage (par exemple, 0,1) est souvent recommandé pour une meilleure convergence.
- **max_depth:** La profondeur maximale des arbres. Une profondeur plus faible permet d'éviter le surajustement.
- **oob_score:** L'utilisation des échantillons hors sac (out-of-bag) permet d'évaluer la performance sans validation croisée.

✓ **Résumé du Processus:**

1. **Initialisation:** Démarrage avec des poids égaux.
2. **Correction des erreurs:** Chaque arbre suivant est formé pour corriger les erreurs du modèle précédent.
3. **Mise à jour des prédictions:** Les poids sont ajustés, et le modèle est mis à jour à chaque itération.
4. **Ajustement des hyperparamètres:** Des hyperparamètres tels que le taux d'apprentissage et la profondeur des arbres sont ajustés pour maximiser la performance tout en évitant le surajustement.

✓ Conclusion:

AdaBoost exploite la séquence et l'adaptation dans la formation des arbres en se concentrant sur les erreurs les plus difficiles à corriger. Cela permet d'améliorer significativement les performances des modèles d'arbres de décision, surtout dans les ensembles de données où certaines observations sont particulièrement difficiles à classer.

2.7.5 Conclusion

Les modèles de machine learning présentés offrent diverses approches pour la classification et la régression. L'optimisation des hyperparamètres, comme la validation croisée et le réglage de C et γ , ajuste les modèles pour améliorer la précision et éviter le surapprentissage.

2.8 Deep Learning

Dans cette section, nous nous concentrerons sur les aspects techniques et mathématiques des réseaux de neurones profonds. On va explorer comment les couches de neurones, les fonctions d'activation, et les processus d'entraînement tels que la **backpropagation** et la descente de gradient sont mis en œuvre pour optimiser les modèles d'apprentissage.

2.8.1 Les Couches dans un Réseau de Neurones

Un réseau de neurones profond est composé de plusieurs couches. Chaque couche réalise une transformation linéaire suivie d'une fonction d'activation non linéaire. Mathématiquement, une couche l applique la transformation suivante:

$$h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)})$$

Où:

- $h^{(l-1)}$ est le vecteur de sortie de la couche précédente (ou les entrées pour la première couche),
- $W^{(l)}$ est la matrice des poids de la couche l ,
- $b^{(l)}$ est le biais de la couche l ,
- f est la fonction d'activation appliquée à chaque neurone.

Cette formule montre comment chaque neurone d'une couche reçoit une combinaison pondérée des neurones de la couche précédente, plus un biais, puis passe cette somme à travers une fonction d'activation.

2.8.2 Fonctions d'Activation

Les fonctions d'activation ajoutent de la non-linéarité, permettant aux réseaux d'apprendre des relations complexes.

✓ ReLU:

$$f(x) = \max(0, x)$$

ReLU renvoie x si x est positif, sinon 0, accélérant l'apprentissage. Toutefois, certains neurones peuvent cesser d'apprendre si leurs valeurs deviennent constamment nulles.

✓ **Tanh:**

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Tanh produit des sorties entre -1 et 1, utile pour centrer les données, mais entraîne la saturation des gradients pour des valeurs extrêmes, ralentissant l'apprentissage.

✓ **Sigmoïde:**

$$f(x) = \frac{1}{1 + e^{-x}}$$

La Sigmoïde compresse entre 0 et 1, idéale pour la classification binaire, mais ralentit l'apprentissage dans les réseaux profonds en réduisant les gradients pour des valeurs extrêmes de x .

2.8.3 Entrainement et Backpropagation

L'entraînement d'un réseau de neurones consiste à ajuster les poids et les biais pour minimiser une fonction de perte. Ce processus est réalisé via l'algorithme de **backpropagation**, en conjonction avec des méthodes d'optimisation comme la **descente de gradient**.

2.8.3.1 Calcul de la Fonction de Perte

La fonction de perte mesure l'écart entre la sortie prédite et la valeur réelle. Dans les tâches de classification binaire, une fonction courante est la **binary cross-entropy**, définie comme suit:

$$L(\hat{y}, y) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Où:

- N est le nombre d'exemples dans le lot (batch),
- y_i est la vraie classe de l'exemple i (soit 0, soit 1),
- \hat{y}_i est la probabilité prédite par le modèle que l'exemple i appartienne à la classe 1.

2.8.3.2 Backpropagation et Descente de Gradient

Une fois la perte calculée, l'algorithme de **backpropagation** est utilisé pour propager l'erreur depuis la couche de sortie jusqu'aux couches précédentes afin de calculer les gradients des poids et des biais.

L'objectif est de minimiser la fonction de perte L . Pour ce faire, on ajuste les poids W et les biais b en utilisant la descente de gradient:

$$\begin{aligned} W_{\text{nouveau}} &= W_{\text{ancien}} - \eta \frac{\partial L}{\partial W} \\ b_{\text{nouveau}} &= b_{\text{ancien}} - \eta \frac{\partial L}{\partial b} \end{aligned}$$

Où:

- η est le taux d'apprentissage, qui contrôle la taille des pas de mise à jour,
- $\frac{\partial L}{\partial W}$ et $\frac{\partial L}{\partial b}$ sont les gradients de la perte par rapport aux poids et aux biais.

Les gradients sont calculés en utilisant la règle de la chaîne pour rétropropager l'erreur depuis la couche de sortie vers les couches cachées. Cela permet à chaque neurone d'ajuster ses poids en fonction de sa contribution à l'erreur totale.

2.8.4 Régularisation et Dropout

La **régularisation L_2** aide à prévenir le surapprentissage en ajoutant une pénalité proportionnelle à la somme des carrés des poids à la fonction de perte :

$$L_{\text{rég}} = L + \lambda \sum_i W_i^2$$

Cela empêche certains neurones de développer des poids trop élevés, réduisant ainsi la spécialisation excessive du modèle.

Le **Dropout**, autre technique courante, désactive aléatoirement un pourcentage de neurones à chaque itération, empêchant leur co-adaptation excessive. Mathématiquement, pour un neurone $h^{(l)}$ dans la couche l :

$$h^{(l)} = \text{Dropout}(f(W^{(l)} h^{(l-1)} + b^{(l)}))$$

Le taux de Dropout, un hyperparamètre clé, doit être bien ajusté : un taux trop bas a peu d'effet, tandis qu'un taux trop élevé peut entraîner un sous-apprentissage.

2.8.5 Hyperparamètres: Signification et Impact

Les hyperparamètres, comme expliqué précédemment, sont des variables fixées avant l'entraînement. Voici les principaux hyperparamètres et leur impact:

- **Nombre d'époques:** Nombre de passages complets sur l'ensemble de données. Trop peu entraîne un sous-apprentissage, trop d'époques peut entraîner un surapprentissage.
- **Taille du Batch:** Nombre d'exemples utilisés par itération. Petits batchs = mises à jour plus fréquentes et bruitées. Grands batchs = plus stables mais nécessitent plus de mémoire.
- **Dropout Rate:** Proportion de neurones désactivés à chaque itération. Taux élevé réduit le surapprentissage, mais trop élevé risque de sous-apprentissage.

2.8.6 Conclusion

Le deep learning transforme les données à travers des couches non linéaires. L'optimisation par backpropagation ajuste les poids pour minimiser la perte. La régularisation, le dropout, et l'ajustement des hyperparamètres évitent le surapprentissage et améliorent les performances.

2.9 Interprétation des Modèles pour la Prédiction de la Satisfaction Client

Une fois les modèles prédictifs validés, il est essentiel d'interpréter les résultats pour identifier les profils clients les plus satisfaits. L'algorithme de clustering KMeans est utilisé pour regrouper les clients en groupes homogènes, puis nous calculons la probabilité de satisfaction pour chaque groupe.

2.9.1 Clustering avec KMeans

KMeans est une méthode non supervisée qui partitionne les observations en k clusters en minimisant les distances intra-clusters. L'objectif est de former des groupes où les observations sont proches du centroïde (centre) du cluster.

Formulation mathématique:

L'algorithme minimise l'inertie intra-cluster:

$$\min \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$$

où C_i est le i -ème cluster, x une observation, μ_i le centroïde, et $\|x - \mu_i\|^2$ la distance euclidienne au carré.

Étape 1: Chaque observation est assignée au cluster le plus proche.

Étape 2: Les centroïdes sont recalculés en prenant la moyenne des observations assignées. Ce processus se répète jusqu'à convergence.

2.9.2 Calcul de la Probabilité de Satisfaction

Chaque cluster représente un groupe distinct de clients. Pour chaque cluster C_i , la probabilité moyenne de satisfaction est calculée par:

$$P_{\text{satisfaction}}(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} P_{\text{satisfaction}}(x)$$

où $P_{\text{satisfaction}}(x)$ est la probabilité de satisfaction de chaque client x , et $|C_i|$ est la taille du cluster. Cela permet d'identifier les groupes de clients les plus satisfaits.

2.9.3 Interprétation des Résultats

L'analyse des résultats permet d'identifier les caractéristiques communes des clients dans les clusters avec la plus forte probabilité de satisfaction. Par exemple, cela peut révéler des segments de clients ayant des habitudes de consommation spécifiques.

2.9.4 Conclusion

L'algorithme KMeans identifie des groupes homogènes de clients et, en calculant la probabilité de satisfaction pour chaque cluster, permet de dégager des profils types. Cela fournit une vision claire des segments de clientèle à cibler, en se basant sur les caractéristiques des clients les plus satisfaits.

2.10 Conclusion

Ce chapitre a présenté des outils essentiels pour le prétraitement des données, l’analyse de corrélation et la modélisation. Grâce à des méthodes comme SMOTE, la sélection des caractéristiques et des modèles de machine learning, nous avons établi des bases solides pour optimiser la précision et la robustesse des modèles prédictifs.

Réalisation

Contents

3.1	Introduction	47
3.2	Prétraitement des données	47
3.2.1	Préparation et nettoyage des données	47
3.2.2	Normalisation des données	50
3.2.3	Utilisation de plusieurs tables	51
3.3	Analyse des données	51
3.3.1	Analyse descriptive	52
3.3.2	Analyse inférentielle	56
3.4	Modélisation	64
3.4.1	Équilibrage des classes (SMOTE)	64
3.4.2	Sélection des caractéristiques	65
3.4.3	Modélisation avec Machine Learning	67
3.4.4	Modélisation avec Deep Learning	68
3.5	Conclusion	70

3.1 Introduction

Dans ce chapitre, on évalue les performances de différents modèles de machine learning et de deep learning pour l'analyse des données dans le secteur des télécommunications. L'objectif est de comparer plusieurs techniques de modélisation, y compris la régression logistique, les arbres de décision, ainsi que des approches plus complexes de deep learning avec PyTorch et Keras/TensorFlow. On examinera aussi l'impact de l'optimisation des hyperparamètres et de la sélection des caractéristiques, ainsi que l'utilisation de méthodes d'ensemble pour améliorer les performances de classification et de prédiction.

3.2 Prétraitement des données

Le prétraitement des données est essentiel pour garantir la qualité des informations avant l'analyse. Il permet de nettoyer et transformer les données brutes pour qu'elles soient prêtes pour la modélisation.

3.2.1 Préparation et nettoyage des données

Chez Ooredoo, le traitement des données s'effectue via **SAS Viya**, un outil de data mining. J'ai utilisé cet outil pour accéder aux bibliothèques de données et manipuler les tables. Les enquêtes sur la satisfaction client, menées mensuellement, offrent un aperçu des raisons de satisfaction ou d'insatisfaction des services.

Collecte des données: Les clients reçoivent quotidiennement un SMS les invitant à évaluer leur satisfaction sur des aspects tels que la recharge ou le réseau, avec une note globale (OSAT) de 1 à 5 :

- 1: **Insatisfaction totale,**
- 2: **Insatisfaction partielle,**
- 3: **Neutralité,**
- 4: **Satisfaction,**
- 5: **Très grande satisfaction.**

Les réponses sont collectées dans des fichiers Excel, fusionnés en une table mensuelle pour analyse dans SAS.

✓ **Table mensuelle:**

La table **Resp_Network_Fev** contient les réponses des clients et leur score de satisfaction **OSAT**, utilisé comme **variable cible** dans notre analyse. Lors de l'importation des fichiers Excel journaliers en SAS pour les fusionner, des erreurs ont rendu la table inexploitable, par exemple, des valeurs incorrectes dans la colonne **OSAT**. On a donc développé un code pour automatiser la correction des erreurs et produire une table mensuelle propre et exploitable.

✓ **Table Monthly Aggregation:** Après avoir préparé la table **Resp_Network_Fev**, on a ensuite travaillé avec une autre table pour l'analyse, nommée **monthly_aggregation**. Cette table contient des informations sur le comportement des clients au cours des trois mois précédents, notamment leur

utilisation du réseau, le nombre de SMS envoyés, les montants des recharges, et d'autres indicateurs clés.

Contrairement à la table **Resp_Network_Fev**, qui se concentre sur les réponses aux enquêtes, **monthly_aggregation** regroupe des informations comportementales riches que nous utilisons comme **caractéristiques** dans notre modèle prédictif.

✓ **Agrégation des données:** Chaque client (identifié par **subscriber_id**) a plusieurs enregistrements dans la table **monthly_aggregation**, un pour chaque mois (identifié par **period_id**). Pour obtenir une vue d'ensemble des trois mois, on a agrégé ces enregistrements en une seule ligne par client en faisant la moyenne sur les 3 mois.

Certaines variables, bien qu'utiles en théorie, posent des problèmes de crédibilité dans les données disponibles.

✓ **Exclusion des variables non pertinentes:**

Dans notre analyse, on a choisi de nous concentrer uniquement sur les clients prépayés en raison de leur comportement plus homogène, lié directement aux recharges. Les clients postpayés ont des comportements influencés par des offres et promotions complexes, rendant leur analyse plus difficile.

on a également exclu certaines variables qui posaient des problèmes de fiabilité ou de pertinence, telles que:

- **age, gender, app_user, cell_name, et nom_gouv:** Ces variables contenaient des informations souvent incomplètes ou incorrectes.
- **offer_name, offer_type_name, et last_offer_id:** Variables spécifiques aux clients postpaid, non pertinentes pour notre analyse.

✓ **Traitement des variables:**

- **Variables catégorielles:** Pour traiter les variables catégorielles, on a appliqué des transformations cohérentes. Par exemple, la variable **device_type** a été transformée en valeurs numériques distinctes pour représenter les différents types d'appareils. Cela permet d'inclure ces variables dans les modèles prédictifs sans créer de biais.

```
229  data ABT_Table_Unique_modified;
230  set ABT_Table_Unique_modified;
231  device_type_cleaned = strip(upcase(device_type));
232  put device_type= device_type_cleaned=;
233  if device_type_cleaned = 'BASIC PHONE' then
234  |   device_type_categorical = 1;
235  |   else if device_type_cleaned = 'FEATURE PHONE' then
236  |   |   device_type_categorical = 2;
237  |   |   else if device_type_cleaned = 'PHABLET' then
238  |   |   |   device_type_categorical = 3;
239  |   |   |   else if device_type_cleaned = 'ROUTER' then
240  |   |   |   |   device_type_categorical = 4;
241  |   |   |   |   else if device_type_cleaned = 'SMARTPHONE' then
242  |   |   |   |   |   device_type_categorical = 5;
243  |   |   |   |   |   else if device_type_cleaned = 'TABLET' then
244  |   |   |   |   |   |   device_type_categorical = 6;
245  |   |   |   |   |   |   else
246  |   |   |   |   |   |   |   device_type_categorical = .;
247  run;
```

Figure 3.1: Exemple de transformation des variables catégorielles

- **Variables quantitatives:** En ce qui concerne les variables quantitatives, telles que les minutes d'appel (**mou**) ou les SMS envoyés (**number_of_sms**), on a procédé à une agrégation des sous-catégories pour obtenir des variables globales. Par exemple, la variable **mou_m1** regroupe les minutes d'appels **onnet**, **offnet**, et **internationales**. Cela permet de simplifier l'analyse tout en conservant la pertinence des données.

```

316  data data.ABT_TABLE_UNIQUE_MODIFIED;
317  |   set data.ABT_TABLE_UNIQUE_MODIFIED;
318  |   mou_m1 = mou_onnet_m1 + mou_offnet_mobile_m1 + mou_offnet_fix_m1 + mou_international_m1;
319  run;
```

Figure 3.2: Exemple d'agrégation des variables quantitatives

- **Variable cible binaire:** On a également créé une nouvelle variable binaire, **osat_binary**, qui représente la satisfaction des clients de manière simplifiée. Les valeurs 1 et 2 de la variable **OSAT** sont considérées comme insatisfaction (codées par 1), tandis que les valeurs 3, 4 et 5 sont regroupées sous la catégorie satisfaction (codées par 0). Cette transformation facilite l'application de modèles de classification binaire.

Ces transformations garantissent que les données, tant qualitatives que quantitatives, sont consolidées et normalisées pour assurer une analyse cohérente et robuste.

✓ **Fusion des tables et agrégation:** on a fusionné les tables **Resp_Network_Fev** et **monthly_aggregation** pour obtenir une table finale. La variable **OSAT** est extraite de **Resp_Network_Fev** et intégrée dans la table finale. Celle-ci contient les comportements clients sur plusieurs mois ainsi que leur score de satisfaction. La table finale agrégée comprend **25 colonnes** et **985 lignes**. Elle regroupe les informations prêtes pour l'analyse. Voici les colonnes principales :

Nom de la variable	Description
msisdn	Identifiant unique du client
OSAT	Score de satisfaction global
subscriber_activation_date_only	Date d'activation du client
Indicator_vas	Utilisation des services à valeur ajoutée
monthly_rech_subscriber_seg	Montant moyen des recharges mensuelles
monthly_arpu_subscriber_seg	Revenu moyen par utilisateur
Number_of_reactivation	Nombre de réactivations
Flag_4g_binary	Utilisation de la 4G (binaire)
device_type	Type d'appareil
flag_smartphone	Indique un smartphone
recharge_moyenne	Montant moyen des recharges
Arpu_moyenne	Revenu moyen
data_volume_moyenne	Volume moyen de données
mou_moyenne	Moyenne des minutes d'appels
nbre_recharges_moyenne	Nombre moyen de recharges
nbre_sms_moyenne	Nombre moyen de SMS

voice_amount_moyenne	Montant moyen des appels vocaux
data_amount_moyenne	Montant moyen des données utilisées
voice_volume_moyenne	Volume moyen des appels

Table 3.1: Colonnes principales de la table finale.

✓ **Gestion des valeurs manquantes:**

- Imputation pour OSAT:** Certaines lignes de la table **Resp_Network_Fev** contenaient des valeurs manquantes dans la colonne **OSAT**. On a imputé ces valeurs en utilisant la moyenne des réponses aux questions **q3 à q18**. Le code complet pour cette imputation est disponible dans l'annexe (voir figure 34).
- Imputation pour subscriber_activation_date_only:** Un total de **114 valeurs manquantes** a été détecté dans cette colonne. Pour y remédier, on a utilisé **KNNImputer** et l'imputation par la médiane via **Jupyter Notebook**.

Avant l'imputation, les dates ont été converties en jours depuis une date de référence (**01/01/1970**), créant une variable **activation_days**. La colonne d'origine a ensuite été supprimée.

Imputation KNN: On a appliqué **KNNImputer** avec 5 voisins pour estimer les valeurs manquantes, en se basant sur la similarité avec les autres colonnes. Pour le code complet utilisé dans cette méthode, référez-vous à la figure 35 dans l'annexe.

Imputation par la médiane: En complément, on a testé l'imputation par la médiane pour comparer avec KNN et identifier la méthode la plus fiable.

3.2.2 Normalisation des données

La normalisation est essentielle pour rendre les variables comparables et améliorer les performances des algorithmes d'apprentissage.

✓ **Gestion des valeurs aberrantes:** On a identifié et remplacé les valeurs aberrantes des variables quantitatives (montants de recharge, volume de données, SMS, minutes d'appels) en utilisant les quartiles (**Q1, Q3**) et l'intervalle interquartile (**IQR**). Les valeurs en dehors de l'intervalle (**Q1 - 1.5 * IQR à Q3 + 1.5 * IQR**) ont été remplacées par la médiane pour éviter les biais. Le code utilisé pour cette étape est disponible dans l'annexe (voir figure ??).

✓ **Normalisation Box-Cox:** Après avoir géré les valeurs aberrantes, différentes méthodes de normalisation (**min-max, sqrt, log**) ont été testées sans succès. On a donc appliqué la transformation **Box-Cox**, particulièrement efficace pour les données asymétriques. L'algorithme a optimisé la valeur de λ à l'aide de la courbe de log-vraisemblance. La transformation a été appliquée à la variable **Recharge_moyenne**, créant la nouvelle variable normalisée **new_recharge_moyenne**. Le code détaillé est présenté dans l'annexe (voir figure ??).

Après la transformation et la gestion des valeurs aberrantes, on a vérifié la distribution des données normalisées via les histogrammes. La normalité a aussi été testée avec Shapiro-Wilk, mais les résultats

ne sont pas disponibles en raison d'un problème sur la plateforme SAS en fin de stage. Cependant, les premiers résultats montrent que la normalité n'est pas totalement vérifiée, bien que les données de cette table soient presque normales. La table a été exportée en format Excel pour l'analyse et la modélisation sur Jupyter, offrant plus de flexibilité et rapidité.

3.2.3 Utilisation de plusieurs tables

✓ **Application sur plusieurs jeux de données:** Le processus de prétraitement a été appliqué à plusieurs jeux de données pour garantir la pertinence et la continuité temporelle. On a traité les tables **Resp_Network_Fev**, **network_may** et **retail_juin**, couvrant les périodes de février à juin. Chaque table contient des colonnes similaires et a été soumise au même processus de nettoyage et de transformation.

✓ **Comparaison des échantillons:** La table **network_may** contient 375 lignes, un échantillon relativement réduit. Pour obtenir des résultats plus représentatifs, on a également utilisé **retail_juin**, qui compte 2554 lignes. Cela permet d'analyser les variations temporelles et d'assurer la robustesse des résultats.

✓ **Pourquoi trois datasets ?** L'utilisation de ces trois tables permet de comparer les performances des modèles sur différentes périodes, d'analyser les tendances comportementales, et d'évaluer la persistance des schémas.

Détails sur les trois tables:

- **Resp_Network_Fev:** 985 lignes, 114 valeurs manquantes pour **activation_days**.
- **network_may:** 375 lignes, 44 valeurs manquantes.
- **retail_juin:** 2554 lignes, 281 valeurs manquantes.

✓ **Difficultés de normalisation:** Malgré les efforts, la normalisation des données dans **network_may** et **retail_juin** n'a pas réussi pour certaines variables, même avec les méthodes **min-max**, **sqrt**, **log**, et **Box-Cox**. Cela sera pris en compte lors de l'analyse des résultats.

✓ **Travail avec les trois tables:** Les trois jeux de données seront utilisés pour:

- Comparer les performances des modèles,
- Évaluer la robustesse des résultats sur plusieurs périodes,
- Identifier les différences comportementales entre les clients.

Ces trois jeux de données, bien que présentant des défis de normalisation, offrent une base solide pour l'analyse et la modélisation prédictive.

3.3 Analyse des données

L'analyse des données comprend à la fois une exploration descriptive des variables et une analyse inférentielle pour extraire des conclusions significatives.

3.3.1 Analyse descriptive

L'analyse descriptive a été réalisée sur la variable **recharge**, en examinant son comportement par rapport à la variable **OSAT**, à la fois en version multiclass et binaire.

3.3.1.1 Analyse des variables quantitatives

Analyse pour la variable Recharge

Les statistiques descriptives pour la variable **recharge** selon les classes **OSAT** multiclass sont présentées à travers la distribution de la variable par classe dans la figure 3.3.

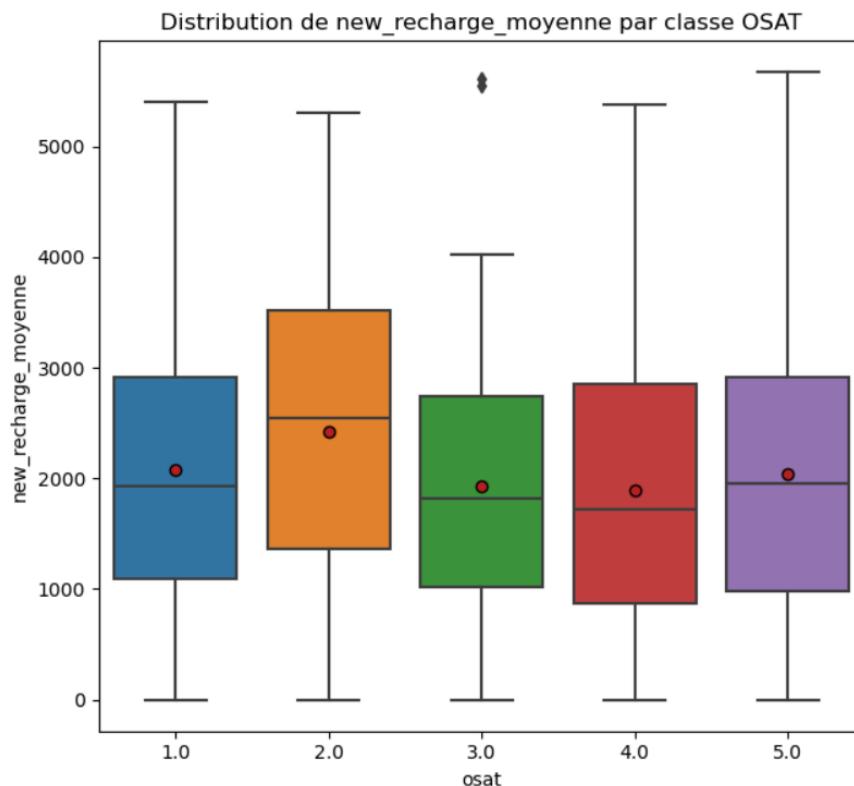


Figure 3.3: Distribution de la variable recharge par classe OSAT multiclass.

Les moyennes de **recharge** montrent une légère variation entre les classes **OSAT** multiclass. Les classes 2 et 5 présentent des médianes plus élevées, tandis que la classe 1 a une médiane légèrement inférieure. Comme le montre la figure 3.3, la distribution de **recharge** est relativement similaire entre les classes, bien que les médianes des classes 2 et 5 soient plus élevées.

Passons maintenant à la comparaison avec **OSAT** binaire.

Les statistiques descriptives pour la variable **recharge** selon les classes **OSAT** binaire sont présentées dans le tableau ci-dessous.

Classe OSAT binaire	Nombre	Moyenne	Ecart-Type	Min	Q1	Médiane	Q3	Max
0	334	2158.66	1323.81	-1.33	1161.87	2041.87	3107.40	5405.60
1	651	1966.57	1310.51	-1.33	907.64	1835.88	2881.57	5670.04

Table 3.2: Statistiques descriptives de la variable recharge selon la classe OSAT binaire.

Dans la version binaire, la moyenne de **recharge** pour la classe 0 est légèrement plus élevée que celle de la classe 1, mais la variabilité est similaire entre les deux groupes, comme le montre l'écart-type.

Comparaison entre les OSAT multiclass et OSAT binaire pour la variable recharge

Bien que la version multiclass de **OSAT** offre une segmentation plus détaillée, les différences entre les classes pour la variable **recharge** ne sont pas assez marquantes pour justifier une complexité supplémentaire. Les moyennes dans la version binaire sont similaires entre les classes 0 et 1, avec une variabilité légère qui n'altère pas l'interprétation globale. Par conséquent, pour simplifier l'analyse tout en assurant une cohérence des résultats, la version binaire est privilégiée.

Analyse pour les variables quantitatives

L'analyse des variables quantitatives selon OSAT multiclass et OSAT binaire révèle des tendances intéressantes. Voici les points principaux :

ARPU:

OSAT multiclass: L'ARPU moyen varie légèrement entre les classes (255 à 301). Les classes 2 et 5 présentent les valeurs moyennes les plus élevées, reflétant une consommation plus importante de ces groupes.

OSAT binaire: La différence entre les classes 0 et 1 est faible (278 vs 275), avec une variabilité similaire entre les deux groupes.

Volume d'appels (voice_volume):

OSAT multiclass: Les moyennes sont stables entre les classes (38 à 43), avec une légère augmentation pour les classes 4 et 5.

OSAT binaire: Différence marginale entre les classes 0 et 1 (41,27 vs 42,28), avec une distribution homogène.

Nombre d'appels (voice_amount):

OSAT multiclass: Les moyennes varient peu entre les classes, la classe 5 étant légèrement plus élevée.

OSAT binaire: Différence minime entre les classes 0 et 1 (28,23 vs 28,65).

Volume de données (data_volume):

OSAT multiclass: La classe 2 présente une consommation de données plus élevée (2,33), tandis que les autres classes sont plus homogènes.

OSAT binaire: La classe 0 consomme légèrement plus de données que la classe 1 (2,21 vs 1,96).

Nombre de transactions de données (data_amount):

OSAT multiclassé: Légère différence entre les classes, avec un pic pour la classe 5 (202).

OSAT binaire: La classe 0 réalise plus de transactions que la classe 1 (205 vs 195), mais la différence reste minime.

Activation Days:

OSAT multiclassé: La moyenne des jours d'activation varie légèrement entre les classes, avec les valeurs les plus élevées pour les classes 3 et 4, correspondant à environ 47,1 ans et 46,3 ans respectivement.

OSAT binaire: La différence entre les classes 0 et 1 est minime, avec des durées moyennes d'activation d'environ 46,6 ans (classe 0) et 46,3 ans (classe 1), ce qui montre une stabilité dans la durée d'activation entre ces deux groupes.

Minutes of Usage (MOU):

OSAT multiclassé: Les moyennes des minutes d'utilisation varient entre 32 et 38 minutes, avec une légère augmentation pour les classes 4 et 5.

OSAT binaire: Les différences entre les classes 0 et 1 sont faibles (35,52 vs 36,82 minutes), ce qui indique une utilisation similaire des services vocaux dans les deux groupes.

Nombre de SMS:

OSAT multiclassé: Le nombre moyen de SMS varie entre 6,71 et 7,86 selon les classes, avec une légère augmentation pour la classe 2, ce qui suggère une utilisation plus importante de cette classe.

OSAT binaire: La différence entre les classes 0 et 1 est faible (7,37 vs 6,97), indiquant une consommation de SMS relativement comparable entre ces deux classes.

Comparaison générale entre OSAT multiclassé et OSAT binaire

Dans l'ensemble, la version multiclassé d'OSAT permet une segmentation plus fine, mais les différences observées entre les classes pour l'ensemble des variables quantitatives sont généralement minimales. La version binaire offre une simplification sans perte significative d'information, ce qui la rend plus appropriée pour les prochaines analyses en réduisant la complexité tout en maintenant une interprétation claire.

3.3.1.2 Analyse des variables qualitatives

Analyse pour la variable qualitative monthly_rech_subscriber_seg

L'analyse de la variable **monthly_rech_subscriber_seg** selon les classes **OSAT multiclassé** est illustrée par la figure ci-dessous.

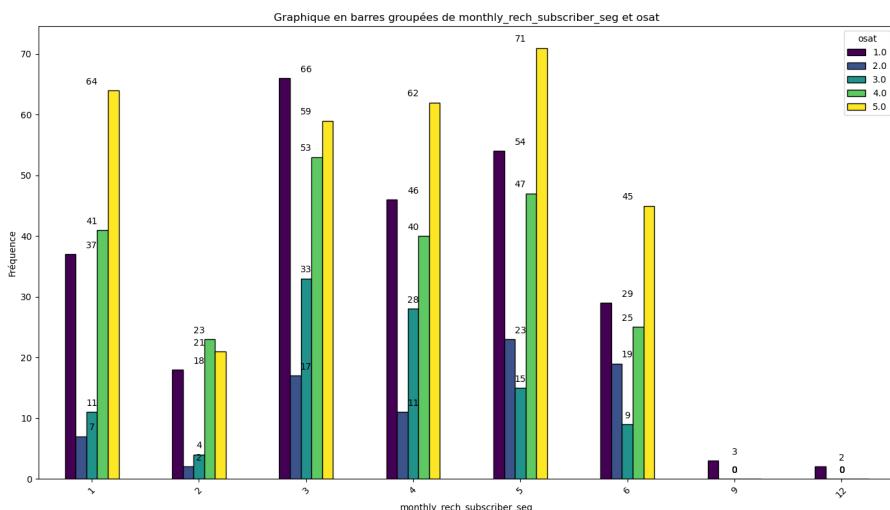


Figure 3.4: Distribution de la variable monthly_rech_subscriber_seg par classe OSAT multiclassé.

Comme le montre la figure 3.4, les classes 1 et 5 sont les plus représentées dans les segments de recharge mensuelle, notamment dans les segments 1 et 6, avec une fréquence plus élevée. Les autres classes, telles que les classes 2, 3 et 4, sont moins représentées dans ces segments, ce qui indique une disparité dans les comportements de recharge des utilisateurs selon leur classe d'OSAT.

Passons maintenant à la comparaison avec **OSAT binaire**.

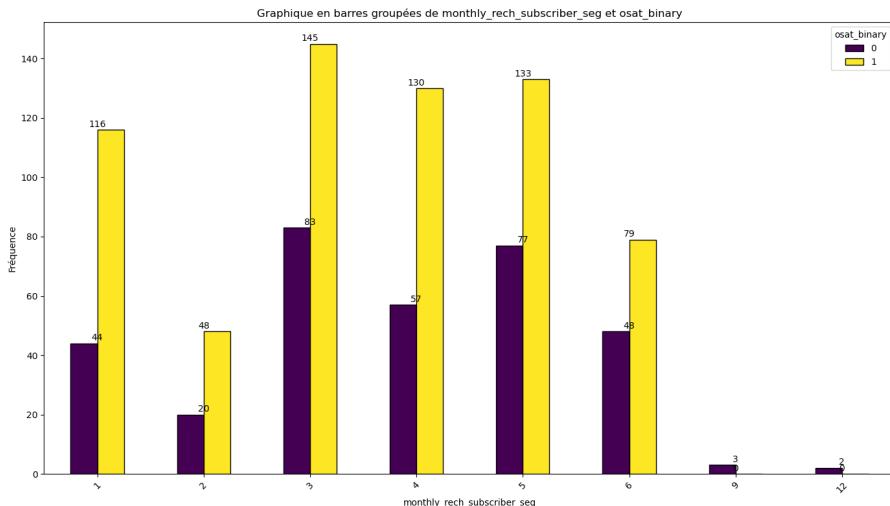


Figure 3.5: Distribution de la variable monthly_rech_subscriber_seg par classe OSAT binaire.

Dans la version binaire, comme le montre la figure 3.5, les utilisateurs satisfaits (classe 1) sont largement dominants dans tous les segments de recharge mensuelle, particulièrement dans les segments 3 et 5. À l'inverse, les utilisateurs insatisfaits (classe 0) sont moins présents, ce qui reflète un comportement de recharge moins intense pour cette classe.

Comparaison entre les OSAT multiclassé et OSAT binaire pour la variable monthly_rech_subscriber_seg

Bien que la version multiclassé de **OSAT** offre une segmentation plus fine, les tendances principales se retrouvent également dans la version binaire. La classe 1 (satisfaits) montre une fréquence plus

élevée dans les segments de recharge les plus importants (3 et 5). Par conséquent, la version binaire est privilégiée pour les analyses ultérieures afin de simplifier les interprétations sans perdre d'informations significatives.

Analyse récapitulative pour les variables qualitatives

Les variables qualitatives ont été analysées en fonction des segments *OSAT multiclass* et *OSAT binaire*. Voici un aperçu des résultats pour chaque variable.

Variable	OSAT multiclass	OSAT binaire	Observations générales
<i>Device Type</i>	La classe 3 présente des valeurs élevées	Les valeurs sont concentrées dans la classe 1	Segment 3 reste important dans les deux OSAT
<i>Monthly ARPU Subscriber Seg</i>	La classe 12 a des valeurs très élevées	Segment 12 montre une prédominance de classe 1	Segment 12 est dominant dans les deux OSAT
<i>Flag 4G Binary</i>	Classe 1 contient la majorité des utilisateurs 4G	La classe 1 reste dominante dans le segment 4G	Une forte majorité est 4G dans les deux OSAT
<i>Flag Smartphone</i>	Les classes 1 et 5 montrent une grande utilisation de smartphones	La majorité des utilisateurs sont dans la classe 1	Utilisation généralisée de smartphones dans les deux OSAT
<i>Number of Reactivation</i>	Classe 1 a des réactivations élevées	Les réactivations sont dominées par la classe 0	Répartition similaire dans les deux OSAT

Table 3.3: Résumé des variables qualitatives entre *OSAT multiclass* et *OSAT binaire*.

Comparaison générale entre OSAT multiclass et OSAT binaire pour les variables qualitatives

La version multiclass d'OSAT offre une segmentation plus détaillée des variables qualitatives, mais les différences entre les classes restent souvent subtiles. La version binaire simplifie l'analyse sans perte majeure d'information, facilitant ainsi l'interprétation tout en réduisant la complexité. Pour les prochaines analyses, OSAT binaire est privilégiée en raison de son efficacité et de sa clarté.

3.3.1.3 Comparaison globale entre OSAT multiclass et OSAT binaire

L'analyse des variables quantitatives et qualitatives montre que, bien que la version multiclass d'OSAT permette une segmentation plus fine, les différences observées entre les classes sont généralement minimes. La version binaire d'OSAT simplifie l'interprétation sans perte significative d'information. Cette simplification réduit la complexité des analyses tout en maintenant une clarté et une cohérence suffisantes dans les résultats. Par conséquent, l'utilisation d'OSAT binaire est privilégiée pour les prochaines étapes, car elle permet une analyse plus efficace et facile à interpréter.

3.3.2 Analyse inférentielle

L'objectif de l'analyse inférentielle est de comprendre les relations entre la variable cible (**OSAT**) et les différentes **features explicatives**, qu'elles soient quantitatives ou qualitatives. Dans cette section,

nous explorons la corrélation entre les variables qualitatives et quantitatives, ainsi que la corrélation entre variables qualitatives elles-mêmes. On a d'abord examiné la corrélation entre la variable cible (**OSAT**) et les différentes **features** quantitatives. L'analyse de corrélation est illustrée par une matrice de corrélation, dans laquelle on a utilisé le coefficient de corrélation de Pearson pour évaluer la relation linéaire entre les variables.

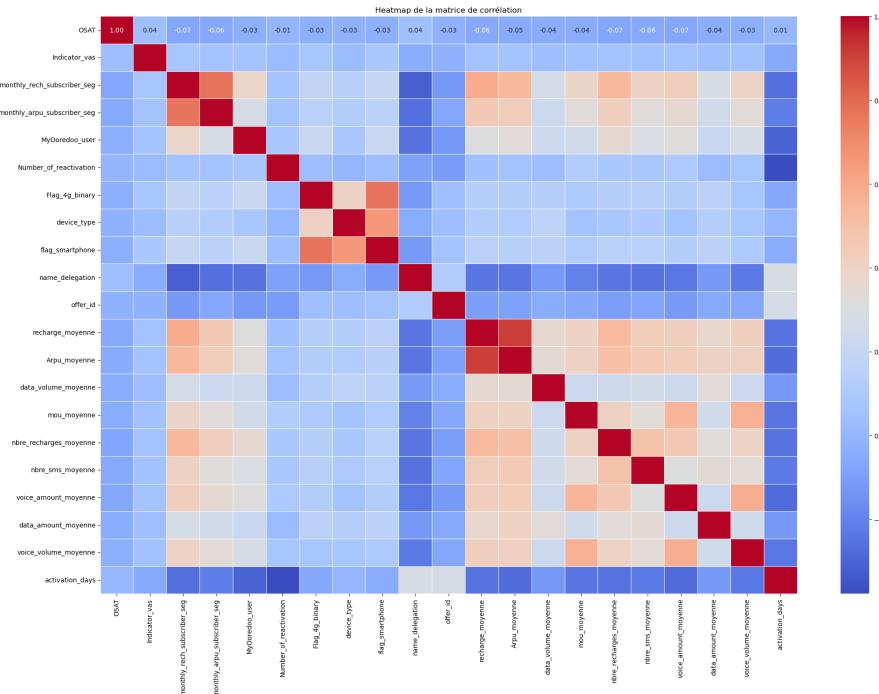


Figure 3.6: Heatmap de la matrice de corrélation globale pour la table de Juin.

3.3.2.1 Corrélation entre la variable cible et les variables explicatives

Corrélation entre les variables qualitatives et la variable cible

Pour évaluer la relation entre des variables qualitatives comme **device_type** et **osat**, plusieurs indicateurs statistiques ont été utilisés, notamment **Cramér's V** et le **test du Chi-Carré**, afin de mesurer et évaluer la force de leur association.

1. Indice de Cramér's V: Cet indice mesure la force de l'association entre deux variables qualitatives. L'indice de Cramér's V varie entre 0 et 1, où une valeur proche de 0 indique une faible association et une valeur proche de 1 indique une association forte.

Le code utilisé pour calculer cet indice est présenté dans l'annexe, voir Figure 38.

✓ Exemple de résultat: Pour la relation entre **device_type** et **osat**, un Cramér's V de **0.0297** a été obtenu, ce qui indique une association faible.

2. Test du Chi-Carré: Ce test permet de vérifier s'il existe une association statistiquement significative entre deux variables qualitatives. Si la p-value est inférieure à 0.05, cela indique une association significative.

Le code utilisé pour effectuer le test du Chi-Carré est présenté dans l'annexe, voir Figure 61.

Pour la relation entre **device_type** et **osat_binary** (binarisation d'OSAT), une p-value de **0.7528** a été obtenue, indiquant qu'il n'y a pas d'association significative.

✓ **Exemple de tableau de contingence:**

```
Résultat de l'indice de Cramér's V entre device_type et osat_binary : 0.0000
Interprétation : Association faible.

Tableau de contingence pour device_type vs osat_binary:
osat_binary    0      1
device_type
1             62    171
2              32     94
3            528   1301
4               2      2
5            104   241
6               4     13

Résultats du test du chi carré pour osat_binary vs device_type:
Statistique = 2.6560
p-value = 0.7528
Aucune association significative entre les variables.
Test de Fisher non applicable, table de contingence non 2x2.
```

Figure 3.7: Tableau de contingence pour device_type et osat_binary.

Le test du Chi-Carré confirme qu'il n'y a pas d'association significative entre **device_type** et **osat_binary** (p-value = 0.7528). Le test de Fisher n'était pas applicable, car le tableau de contingence n'était pas de dimension 2x2.

Cette méthodologie a été appliquée pour toutes les autres variables qualitatives.

→ **Variables avec faible association:** Certaines variables montrent une faible association avec la satisfaction (OSAT), notamment **monthly_arpu_subscriber_seg**, **monthly_rech_subscriber_seg**, **flag_smartphone**, et **Number_of_reactivation**. Les indices de **Cramér's V** sont inférieurs à 0.1 et les **p-values** du **Chi-Carré** dépassent 0.05, indiquant une faible ou aucune association statistiquement significative.

→ **Variables sans association significative:** Les variables **Flag_4g_binary** et **flag_smartphone** n'ont montré aucune association avec **osat_binary**. Les **p-values** sont supérieures à 0.05, confirmant l'absence de relation statistiquement significative.

→ **Variable avec une association significative:** La variable **monthly_rech_subscriber_seg** montre une association significative avec une **p-value** de **0.0211**, suggérant un impact potentiel sur la satisfaction (**osat_binary**).

3. Analyse de l'association par la Lambda de Goodman et Kruskal: Pour approfondir l'analyse des relations entre les variables qualitatives et **osat_binary**, on a calculé la **Lambda de Goodman et Kruskal**, qui évalue la réduction d'erreur dans la prédiction d'une variable en fonction d'une autre.

Le code utilisé pour calculer la Lambda de Goodman et Kruskal est présenté dans l'annexe, voir Figure 62.

✓ **Exemple de résultat:** Pour la variable **Flag_4g_binary**, une Lambda de **0.42** a été obtenue, indiquant que la connaissance de cette variable permet une réduction d'erreur de **42%** dans la prédiction de **osat_binary**.

Le résultat de la Lambda de Goodman et Kruskal est présenté dans l'annexe, voir Figure ??.

✓ **Résultats pour les autres variables:** Les résultats montrent que **device_type** a une lambda de 0.15, indiquant une faible réduction d'erreur, tandis que **flag_smartphone** et **Number_of_reactivation** affichent des lambdas de 0.60 et 0.84 respectivement, signalant une réduction d'erreur plus significative. En revanche, **monthly_arpu_subscriber_seg** et **monthly_rech_subscriber_seg** n'ont aucune réduction d'erreur avec des lambdas nulles.

Cela montre que certaines variables, comme **flag_smartphone** et **Number_of_reactivation**, influencent la satisfaction (**osat_binary**), tandis que d'autres, comme **monthly_arpu_subscriber_seg**, n'apportent aucune valeur prédictive.

Corrélation entre les variables quantitatives et la variable cible

Pour évaluer la relation entre les variables quantitatives et la variable cible **osat_binary**, on a utilisé l'indicateur de **l'Eta carré**. Cet indicateur permet de mesurer l'effet de la variable explicative sur la variable cible. Le calcul de l'Eta carré a été effectué pour **osat_binary**.

Le code utilisé pour calculer l'Eta carré est présenté en annexe, voir Figure 63.

Ensuite, pour déterminer le test statistique à utiliser, on a vérifié la normalité et l'homogénéité des variances avec les tests de **Shapiro-Wilk** et **Levene**. Si les conditions étaient remplies, on a utilisé le **test t de Student**. Sinon, le **test de Mann-Whitney** a été appliqué.

Le code de ces tests est disponible en annexe, voir Figure 64.

Résultats pour la variable Recharge:

Pour la variable quantitative **recharge**, l'ensemble des tests a été appliqué afin d'évaluer son association avec **OSAT**. Voici les résultats obtenus à chaque étape.

✓ **Résultat de l'Eta carré:** L'indicateur de **l'Eta carré** montre un effet très faible pour **OSAT**, avec une valeur de **0.0064**.

Résultat de l'Eta carré pour osat_binary est disponible en annexe, voir Figure 57.

L'interprétation de ce résultat indique que l'effet de la variable explicative sur la satisfaction (**OSAT**) est très faible, suggérant une influence limitée.

✓ **Test pour osat_binary:** Les tests de **Shapiro-Wilk** montrent que les données ne suivent pas une distribution normale pour les deux groupes (statistiques de **0.7522** pour le groupe 0 et **0.6954** pour le groupe 1, avec des p-values inférieures à 0.05). De plus, le test de **Levene** indique que les variances ne sont pas homogènes (**p-value = 0.0273**). Par conséquent, les conditions pour appliquer un test paramétrique ne sont pas respectées. Nous avons donc appliqué le **test de Mann-Whitney**, qui révèle des différences significatives entre les deux groupes (**p-value = 0.0003**).

Les résultats du test de normalité et de variances et du test de Mann-Whitney sont disponibles en annexe, voir Figure 58 et 59.

Les résultats du test de Mann-Whitney montrent une différence significative entre les distributions de la variable **recharge_moyenne** pour les deux groupes de **osat_binary** (**p-value = 0.0003**),

indiquant que les valeurs de **recharge_moyenne** diffèrent significativement selon les niveaux de **osat_binary**.

Pour **osat_multiclasse**, les tests ont montré des résultats similaires avec des différences significatives entre les groupes (*p-value* = 0.0003).

Conclusion des tests: Les tests de normalité (**Shapiro-Wilk**) et d'homogénéité des variances (**Levene**) n'ont pas été vérifiés pour les variables quantitatives étudiées. Par exemple, pour **Arpu**, le test de Shapiro-Wilk a donné des *p-values* de **0.0000**, indiquant une distribution non normale. Bien que certaines variables, comme **Arpu**, aient montré des variances homogènes (*p* = **0.0801**), les conditions d'application des tests paramétriques n'ont globalement pas été respectées.

En conséquence, tous les tests utilisés sont non paramétriques : le **Mann-Whitney** pour **osat_binary** et le **Kruskal-Wallis** pour **OSAT**. Lorsque des différences significatives ont été détectées avec Kruskal-Wallis, des comparaisons post-hoc ont été effectuées avec le test de **Dunn**. Le tableau ci-dessous résume les résultats :

Variable	Eta carré	Test U (osat_binary)	Kruskal-Wallis (osat)	Interprétation finale
Arpu	0.0057	<i>p</i> = 0.0005	<i>p</i> = 0.0011	Différences significatives
Voice volume	0.0047	<i>p</i> = 0.0008	<i>p</i> = 0.0001	Différences significatives
Voice amount	0.0078	<i>p</i> = 0.0001	<i>p</i> = 0.0001	Différences significatives
Data volume	0.0029	<i>p</i> = 0.1979	<i>p</i> = 0.2741	Pas de différences significatives
Mou	0.0042	<i>p</i> = 0.0003	<i>p</i> = 0.0002	Différences significatives
Nbre of SMS	0.0058	<i>p</i> = 0.0038	<i>p</i> = 0.0054	Différences significatives
Activation days	-	<i>p</i> = 0.8498	<i>p</i> = 0.6350	Pas de différences significatives

Conclusion

Bien que certaines variables quantitatives et qualitatives montrent une relation avec la satisfaction (**OSAT**), la majorité d'entre elles n'indiquent pas de différences significatives. Ces résultats mettent en évidence des impacts limités des variables étudiées sur la satisfaction client. **Variables quantitatives:** Les variables quantitatives, comme **Arpu**, **Voice Volume**, et **Mou**, montrent des différences significatives entre les groupes **OSAT**. Cependant, d'autres variables, comme **Data Volume** et **Activation days**, ne présentent pas de différences significatives.

Variables qualitatives: Certaines variables qualitatives, telles que **Device Type** et **Flag_4G**, influencent significativement la satisfaction client. Toutefois, plusieurs autres, comme les segments de recharge ou d'ARPU mensuel, n'ont pas d'association significative.

3.3.2.2 Analyse des corrélations entre les variables explicatives

Variable quantitative VS Variable quantitative:

L'analyse des corrélations entre les variables explicatives permet de détecter des redondances ou interdépendances. Une forte corrélation pourrait indiquer que deux variables véhiculent des informations similaires, ce qui peut influencer la modélisation ou l'interprétation des résultats.

Pour cela, on a utilisé à la fois des outils graphiques et des tests statistiques pour explorer la force des relations entre les variables.

Le code utilisé pour effectuer ces analyses est disponible en annexe, voir Figures 65 et 66.

Le code analyse les corrélations entre deux variables explicatives. La **covariance** est calculée pour déterminer la direction de la relation (positive, négative ou inexistante). Les **tests de normalité** (Shapiro-Wilk) permettent de guider le choix entre le **test de Pearson** (corrélation linéaire pour les variables normales) et le **test de Spearman** (corrélation monotone pour les données non normales).

De plus, le code génère des **diagrammes de dispersion** qui permettent de visualiser et d'interpréter la relation en termes de force et de direction.

Cette approche permet d'explorer les relations linéaires et non linéaires, même lorsque les conditions paramétriques ne sont pas respectées.

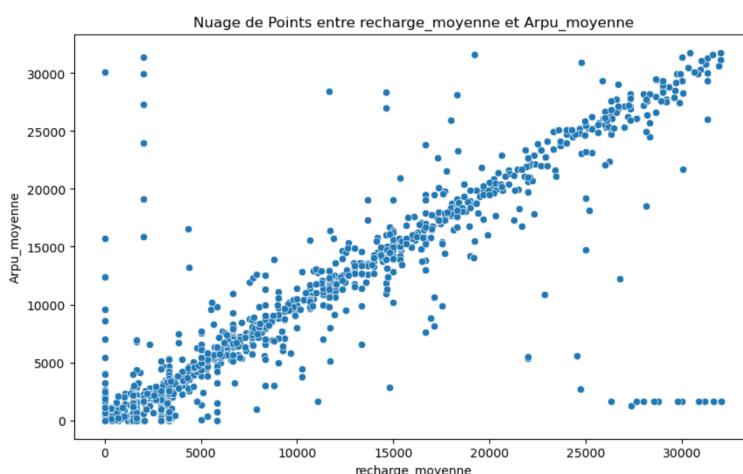


Figure 3.8: Nuage de points entre recharge_moyenne et Arpu_moyenne

Cette première figure 3.8 représente un nuage de points entre les variables **recharge_moyenne** et **Arpu_moyenne**. On observe une tendance positive forte, où les deux variables semblent évoluer ensemble. Cela suggère une corrélation significative.

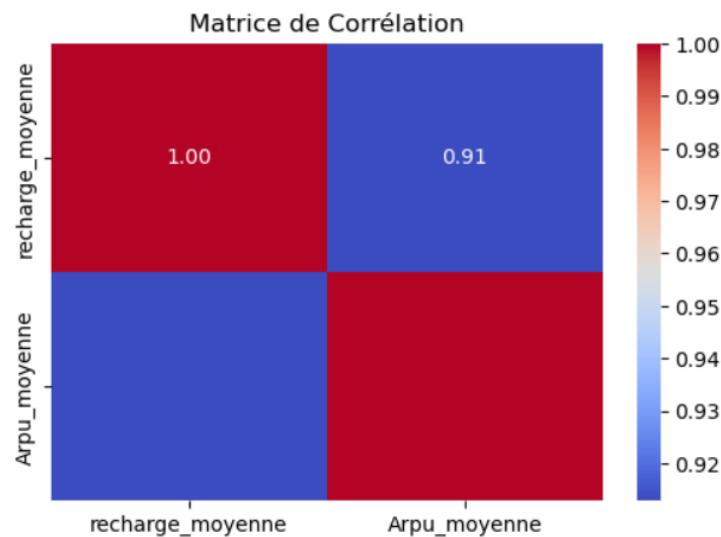


Figure 3.9: Matrice de corrélation entre recharge_moyenne et Arpu_moyenne

La matrice de corrélation dans la figure 3.9 confirme cette observation avec un coefficient de corrélation de 0.91, indiquant une corrélation forte et positive entre ces deux variables.

La figure 60 de l'annexe B résume les résultats de la covariance, des tests de normalité, et du test de corrélation entre les variables recharge_moyenne et Arpu_moyenne. Comme les deux variables ne suivent pas une distribution normale, on a utilisé le test non paramétrique de Spearman pour évaluer la corrélation.

Le test de Shapiro-Wilk a montré que les deux variables **recharge_moyenne** et **Arpu_moyenne** ne suivent pas une distribution normale, avec des p-values très faibles (inférieures à 0.05). Par conséquent, le test non paramétrique de Spearman a été utilisé, et il a révélé une corrélation positive et forte entre ces deux variables (**r = 0.9450**, **p-value = 0.0000**). Cela signifie que lorsque **recharge_moyenne** augmente, **Arpu_moyenne** tend également à augmenter.

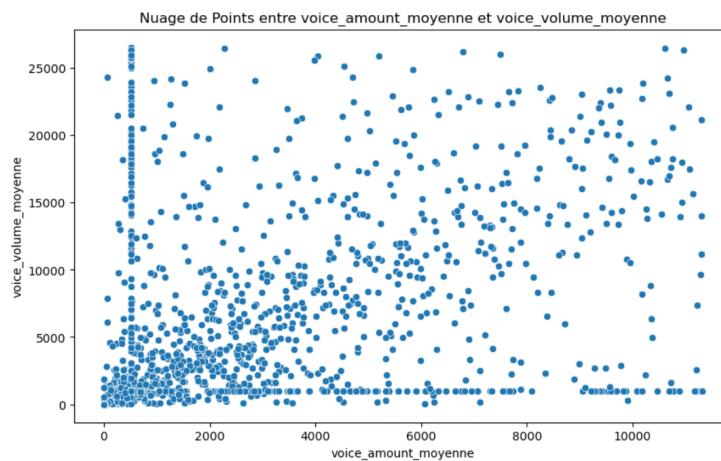


Figure 3.10: Nuage de points entre voice_volume_moyenne et voice_amount_moyenne

La figure 3.10 montre un nuage de points entre les variables **voice_volume_moyenne** et **voice_amount_moyenne**. Bien que la tendance soit visible, la relation reste modérée.

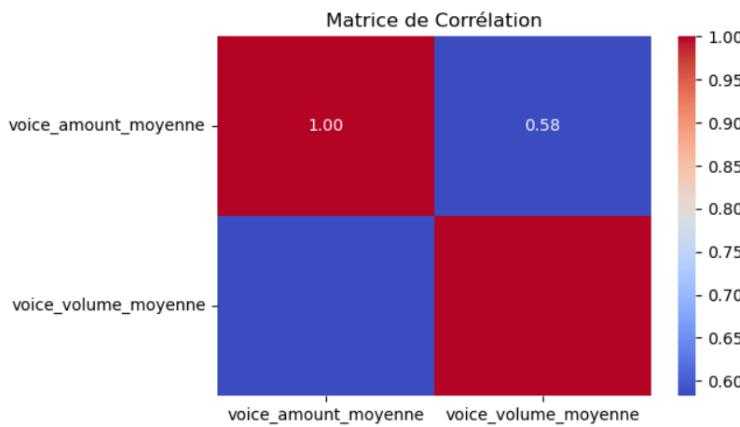


Figure 3.11: Matrice de corrélation entre voice_volume_moyenne et voice_amount_moyenne

La matrice de corrélation dans la figure 3.11 montre un coefficient de corrélation de 0.58, ce qui suggère une relation modérée entre ces deux variables.

Les résultats des tests de Shapiro-Wilk montrent que les deux variables ne suivent pas une distribution normale, avec des p-values inférieures à 0.05. Le test de Spearman a révélé une corrélation modérée mais significative (**r = 0.58, p-value = 0.0000**). Cela indique que les deux variables ont tendance à augmenter ensemble, bien que la relation soit modérée.

Conclusion

Les résultats de l'analyse des corrélations révèlent que plusieurs variables quantitatives présentent des relations significatives. Par exemple, la corrélation entre **recharge moyenne** et **ARPU** est forte avec un coefficient de $r = 0.91$, ce qui suggère qu'une augmentation de la recharge moyenne est associée à une augmentation de l'ARPU. De même, la **voice_volume_moyenne** et la **voice_amount_moyenne** montrent également une corrélation positive avec $r = 0.58$, reflétant une relation modérée entre ces deux variables. Certaines relations, comme entre **recharge moyenne** et **activation_days**, montrent une faible corrélation négative ($r = -0.25$), suggérant une tendance inverse entre l'ancienneté du client et la fréquence des recharges. D'autres variables, telles que **nbre_sms_moyenne** et **recharge_moyenne**, affichent une corrélation positive modérée ($r = 0.77$).

En général, les corrélations entre les variables de consommation, telles que la **mou moyenne** ou le **data volume**, sont positives et significatives, tandis que les variables temporelles montrent un impact moindre ou négatif.

2. Variable quantitative VS Variable qualitative:

on va maintenant analyser les corrélations entre les variables explicatives quantitatives et qualitatives afin de comprendre leur interdépendance et leur impact potentiel sur les résultats.

Pour cette section, nous analysons les relations entre les variables explicatives quantitatives et qualitatives en utilisant les tests ANOVA et Kruskal-Wallis. Voici un exemple illustré par la figure ci-dessous.

```
Résultats du test ANOVA :  
F-statistique : 0.4611  
p-value : 0.4973  
Les moyennes ne sont pas significativement différentes (p >= 0.05).
```

```
Résultats du test Kruskal-Wallis :  
Statistique H : 0.5988  
p-value : 0.4390  
Les médianes ne sont pas significativement différentes (p >= 0.05).
```

Figure 3.12: Tests ANOVA/Kruskal-Wallis entre mou_moyenne et Flag_4g_binary

La figure 3.12 présente l’analyse de **mou_moyenne** selon la variable qualitative **Flag_4g_binary**. Les résultats des tests ANOVA et Kruskal-Wallis montrent des différences significatives entre les groupes (**p-value** < 0.05), indiquant que les utilisateurs du **Flag 4G** consomment significativement plus de minutes (**mou_moyenne**) que les non-utilisateurs.

Dans cet exemple, nous observons que l’accès à la 4G semble être associé à une augmentation de la consommation de minutes, ce qui suggère une corrélation significative entre ces variables.

Conclusion

La même méthodologie a été appliquée aux combinaisons de variables quantitatives et qualitatives. Voici quelques résultats obtenus : des différences significatives ont été observées entre **activation_days** et **flag_smartphone** (ANOVA p = 0.0232, Kruskal-Wallis p = 0.0006), indiquant une variation selon l’utilisation du smartphone. Concernant **recharge_moyenne** et **Number_of_reactivation**, des différences significatives ont également été notées (ANOVA p = 0.0164, Kruskal-Wallis p = 0.0000), avec une recharge moyenne plus élevée pour les utilisateurs réactivés.

Par ailleurs, **mou_moyenne** montre des différences très significatives avec **Number_of_reactivation** (ANOVA p = 0.0000, Kruskal-Wallis p = 0.0000), illustrant une consommation plus importante pour les utilisateurs réactivés. Enfin, le nombre moyen de recharges (**nbre_recharges_moyenne**) est également significatif pour **Number_of_reactivation** (ANOVA p = 0.0000, Kruskal-Wallis p = 0.0000), ces utilisateurs ayant un nombre moyen de recharges plus élevé.

Ces analyses soulignent des associations importantes entre des variables qualitatives comme **flag_smartphone** et **Number_of_reactivation** et des variables quantitatives telles que **mou_moyenne** et **recharge_moyenne**, ce qui peut guider la modélisation future.

3.4 Modélisation

3.4.1 Équilibrage des classes (SMOTE)

L’équilibrage des classes de la variable cible **OSAT** a été effectué avec la méthode SMOTE (Synthetic Minority Over-sampling Technique) pour corriger la répartition initiale déséquilibrée, comme illustré par la figure 3.13.

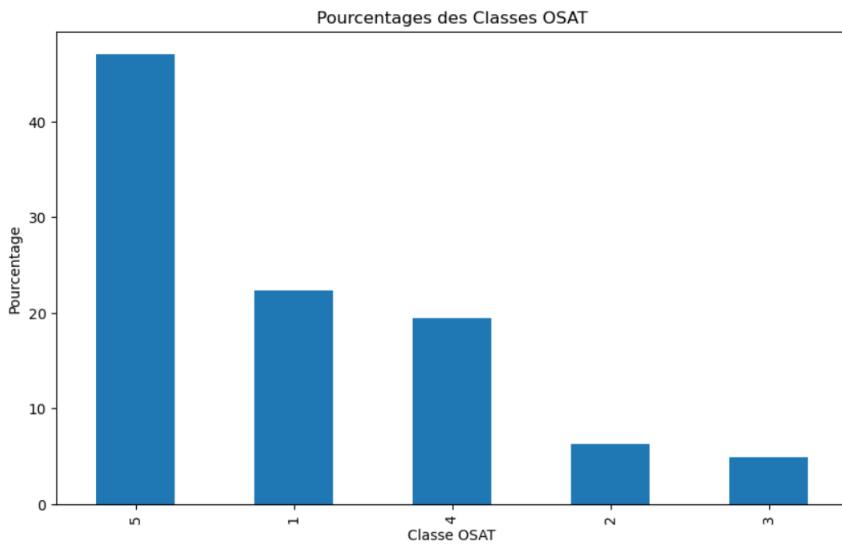


Figure 3.13: Pourcentages des classes OSAT avant équilibrage

Avant SMOTE, la classe 5 dominait avec 47%, tandis que les classes 2 et 3 étaient sous-représentées. SMOTE a permis de rééquilibrer les classes en augmentant la présence des classes moins fréquentes sans suréchantillonner excessivement certaines classes.

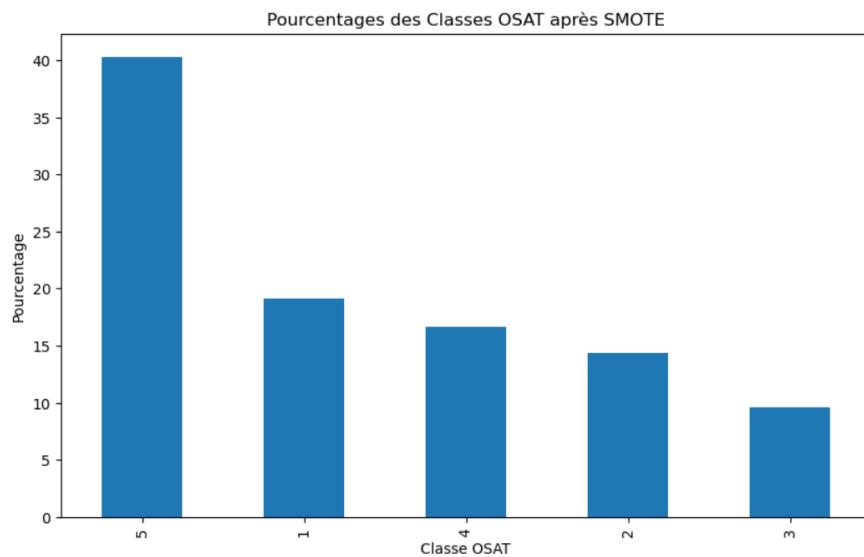


Figure 3.14: Pourcentages des classes OSAT après équilibrage avec SMOTE

Après SMOTE (figure 3.14), les classes sont réparties de façon plus homogène, réduisant le biais en faveur des classes majoritaires et améliorant la modélisation. Les nouvelles proportions sont :

Classe 5: 40.28%, Classe 1: 19.15%, Classe 4: 16.63%, Classe 2: 14.35%, Classe 3: 9.56%.

Cette répartition équilibrée permet un jeu de données plus adapté à la modélisation et à une meilleure interprétation des résultats finaux.

3.4.2 Sélection des caractéristiques

Trois méthodes ont été appliquées pour la sélection des caractéristiques : **RFE (Recursive Feature Elimination)**, **SelectKBest**, et les **importances des caractéristiques à partir de Random Forest**.

Ces méthodes permettent d'identifier les variables explicatives ayant le plus grand impact sur la prédiction de la variable cible **OSAT**.

Le code utilisé pour effectuer cette sélection est disponible en annexe, voir Figure 67.

RFE : La méthode RFE avec un classificateur Random Forest a identifié **recharge_moyenne**, **data_amount_moyenne**, et **voice_amount_moyenne** comme les variables les plus pertinentes (rang 1).

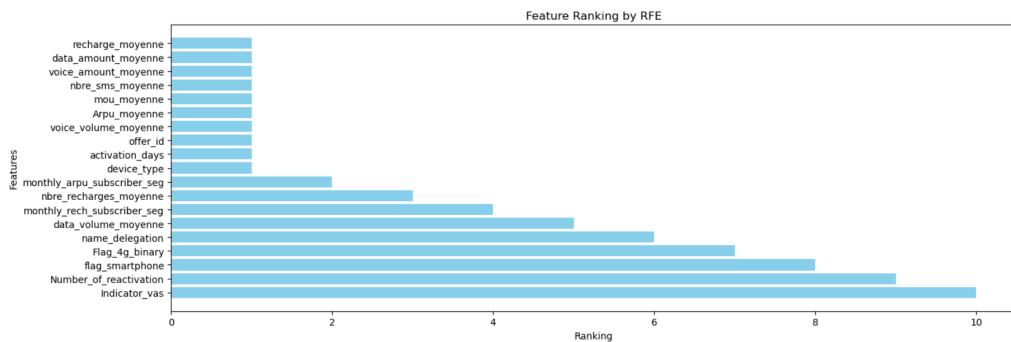


Figure 3.15: Classement des caractéristiques par RFE

SelectKBest : Cette méthode a sélectionné des variables en fonction de leur variance expliquée. Les variables les plus influentes sont **flag_smartphone** (score de 7.21), **recharge_moyenne** (6.39), et **data_amount_moyenne** (6.38). D'autres variables importantes incluent : **Flag_4g_binary** (6.12), **Arpu_moyenne** (5.61), **voice_volume_moyenne** (5.43), et **monthly_arpu_subscriber_seg** (5.23).

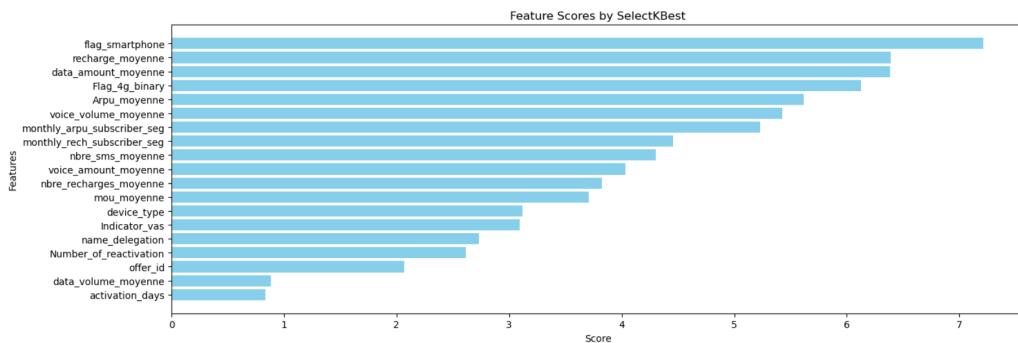


Figure 3.16: Scores des caractéristiques par SelectKBest

Random Forest : L'analyse d'importance des caractéristiques a révélé que **activation_days** (0.29) est la variable la plus influente, suivie par **offer_id** (0.086) et **voice_volume_moyenne** (0.075). Les autres variables importantes incluent **voice_amount_moyenne** (0.071), **Arpu_moyenne** (0.061), **recharge_moyenne** (0.061), **mou_moyenne** (0.058), et **data_amount_moyenne** (0.044).

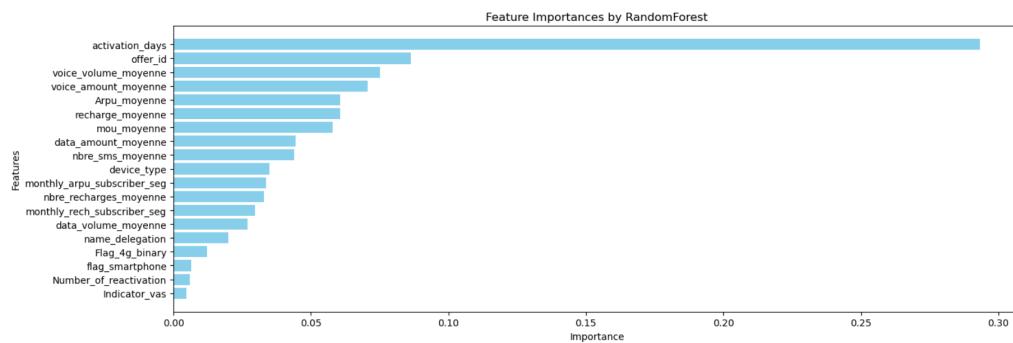


Figure 3.17: Importances des caractéristiques par Random Forest

Conclusion:

Les méthodes de sélection des caractéristiques **RFE**, **SelectKBest**, et **Random Forest** ont permis d'identifier les variables clés pour la prédiction de **OSAT**. La figure suivante montre les caractéristiques communes aux trois méthodes, telles que **recharge_moyenne**, **voice_amount_moyenne**, **mou_moyenne**, et **activation_days**, qui se distinguent systématiquement.

```
print(df_reduit_imputed.dtypes)
[147]
...
...    Arpu_moyenne           int32
    OSAT                  int32
    offer_id                int32
    nbre_sms_moyenne        int32
    voice_volume_moyenne   int32
    data_volume_moyenne    int32
    data_amount_moyenne    int32
    activation_days         float64
    voice_amount_moyenne   int32
    recharge_moyenne       int32
    mou_moyenne              int32
dtype: object
```

Figure 3.18: Caractéristiques communes importantes entre les trois méthodes

Ce processus rigoureux a permis de définir les variables ayant un impact direct et significatif sur **OSAT**. Ces caractéristiques seront prioritaires dans les futurs modèles prédictifs pour garantir des performances optimales et des résultats fiables.

3.4.3 Modélisation avec Machine Learning

Bien que plusieurs modèles aient été testés, cette section se concentre sur l'algorithme **Random Forest**, qui a donné les meilleurs résultats. Nous avons appliqué une validation croisée, ajusté les hyperparamètres, et évalué le modèle avec des métriques telles que l'exactitude, le rappel et la courbe ROC.

3.4.3.1 Prétraitement des données

Le prétraitement a consisté à standardiser les caractéristiques avec un **StandardScaler** pour assurer une échelle comparable entre les variables. Les données ont ensuite été divisées en ensembles d'entraînement et de test (70/30) avec **train_test_split**.

3.4.3.2 Optimisation des hyperparamètres

L'optimisation des hyperparamètres a été réalisée avec **RandomizedSearchCV** pour explorer efficacement différentes combinaisons de paramètres, incluant **n_estimators**, **max_depth**, **min_samples_split**, **min_samples_leaf**, et **criterion** (**gini** ou **entropy**). Une validation croisée à 3 plis a été utilisée pour évaluer les performances.

Le code utilisé pour l'optimisation des hyperparamètres est disponible en annexe, voir Figure 68.

3.4.3.3 Sélection des caractéristiques importantes et réentraînement du modèle

Après avoir trouvé le meilleur modèle, les **importances des caractéristiques** ont été extraites pour identifier les variables clés dans la prédiction de **OSAT**. Seules les caractéristiques avec un seuil d'importance supérieur à 0.035 ont été retenues.

Le graphique des importances des caractéristiques est disponible en annexe, voir Figure 69.

Un code a également été implémenté pour optimiser le seuil de classification, en testant différents seuils afin de maximiser le compromis entre taux de vrais positifs et faux positifs. Cependant, le seuil initial de 0.035 s'est avéré plus performant en termes de précision, rappel et F1-score, offrant un meilleur équilibre sans surajuster le modèle.

Après la sélection des caractéristiques importantes, le modèle Random Forest a été réentraîné uniquement avec ces variables, sur des données standardisées, en suivant les mêmes étapes de validation croisée.

3.4.3.4 Optimisation du seuil de classification

Pour améliorer les performances du modèle, la courbe ROC a été utilisée pour déterminer le **seuil de classification optimal**, en maximisant la différence entre le taux de vrais positifs et le taux de faux positifs.

La courbe ROC et son AUC sont disponibles en annexe, voir Figure 70.

3.4.3.5 Conclusion

Le modèle Random Forest a montré de bonnes performances après optimisation. La sélection des caractéristiques et l'ajustement du seuil ont amélioré l'exactitude et la précision, avec des variables clés comme **recharge_moyenne**, **voice_amount_moyenne**, et **activation_days**.

3.4.4 Modélisation avec Deep Learning

Dans cette section, nous explorons l'application des réseaux de neurones pour la prédiction de la variable cible **osat_binary**, à travers deux approches distinctes: **PyTorch** et **Keras/TensorFlow**. Nous détaillons le processus de modélisation, incluant la préparation des données, le choix des

architectures de réseaux de neurones, et les stratégies de réglage des hyperparamètres, tout en mettant en évidence les spécificités de chaque approche.

3.4.4.1 Approche PyTorch

L'approche **PyTorch** a été choisie pour sa flexibilité. Nous avons construit un modèle de classification binaire pour prédire la satisfaction des clients (**osat_binary**).

1. Préparation des données: Les données ont été standardisées et converties en tenseurs. Les **poids de classe** ont été calculés pour traiter le déséquilibre des classes.

Le code détaillé de la préparation des données est disponible en annexe, voir Figure 71.

2. Architecture du modèle: Le modèle contient trois couches cachées avec **ReLU** et **Dropout** pour éviter le surajustement. La dernière couche utilise **sigmoïde** pour la classification binaire.

L'architecture du modèle est présentée en annexe, voir Figure 72.

3. Optimisation des hyperparamètres: L'algorithme **Adam** a été utilisé avec différentes combinaisons d'hyperparamètres (**learning rate**, **batch size**, **Dropout**). Le modèle a été entraîné sur 100 époques pour garantir la convergence.

Le processus d'optimisation des hyperparamètres est disponible en annexe, voir Figure 73.

4. Sélection du meilleur modèle et évaluation: Le modèle avec les meilleurs hyperparamètres a été sélectionné : **learning rate** de 0.01, **batch size** de 32, **Dropout** de 0.3. Il a été évalué avec des métriques telles que la précision, le rappel, le F1-score, et la matrice de confusion.

Les résultats de l'évaluation finale sont en annexe, voir Figure 74.

3.4.4.2 Approche Keras/TensorFlow

Cette section présente trois approches pour modéliser les données avec Keras/TensorFlow : une architecture classique, une optimisation des hyperparamètres et une optimisation de l'architecture des couches.

1. Approche Classique: Une architecture de réseau de neurones simple avec des couches denses et des fonctions d'activation **ReLU**. Le modèle a été entraîné avec un taux d'apprentissage de **0.001**, 200 époques, et une taille de lot de 16.

L'architecture classique est disponible en annexe, voir Figure 75.

2. Optimisation des Hyperparamètres: Cette approche optimise des paramètres comme le taux d'apprentissage, le **dropout**, le nombre d'unités par couche et la taille des lots. Les valeurs explorées incluent : **learning rate** [1e-5, 1e-4, 1e-3, 1e-2], **dropout** [0.3, 0.4, 0.5], unités par couche [32, 64, 128], **batch size** [16, 32, 64], et 20 époques. Cette optimisation a amélioré les performances du modèle.

Le processus d'optimisation des hyperparamètres pour Keras est présenté en annexe, voir Figure 76.

3. Optimisation de l'Architecture des Couches: L'optimisation de l'architecture des couches a exploré différentes configurations, fonctions d'activation (**ReLU**, **sigmoid**, **tanh**) et optimiseurs.

Les architectures testées incluent [64], [64, 32], et [128, 64, 32], cette dernière s'avérant la plus performante avec **ReLU** pour les premières couches et **Sigmoid** pour la dernière.

Les détails de l'optimisation des couches sont en annexe, voir Figure 77.

3.4.4.3 Conclusion

Les approches PyTorch et Keras/TensorFlow ont permis d'explorer différentes stratégies de modélisation. PyTorch a optimisé les hyperparamètres pour de meilleures performances, tandis que Keras/TensorFlow a testé diverses configurations de modèles, incluant des réseaux standard, des optimisations d'hyperparamètres et des architectures de couches. Les résultats seront détaillés dans le prochain chapitre.

3.5 Conclusion

Ce chapitre a couvert le prétraitement des données, la sélection des caractéristiques et la modélisation avec des algorithmes de Machine Learning et Deep Learning.

Après avoir optimisé les modèles, on a retenu les meilleures configurations pour chaque méthode. Les résultats finaux des performances des modèles seront présentés dans le chapitre suivant.

Résultats et interprétation

Contents

4.1	Introduction	72
4.2	Résultats des Modèles de Machine Learning	72
4.2.1	Modèle de Régression Logistique	72
4.2.2	Modèle Arbre de Décision	74
4.2.3	Modèle de Random Forest	76
4.3	Résultats des Modèles de Deep Learning	81
4.3.1	Modèle Deep Learning avec PyTorch	81
4.3.2	Résultats des Modèles avec Keras/TensorFlow	83
4.4	Comparaison Entre les Modèles de Machine Learning et de Deep Learning	84
4.4.1	Comparaison et Validation des Modèles de Machine Learning	85
4.4.2	Comparaison et Validation des Modèles de Deep Learning	86
4.4.3	Comparaison Globale : Machine Learning vs Deep Learning	87
4.4.4	Modèle d'Ensemble (Random Forest, AdaBoost et Réseau de Neurones)	87
4.4.5	Vérification de surajustement du meilleur modèle	90
4.4.6	Conclusion	90
4.5	Vérification du surajustement du meilleur modèle	91
4.5.1	Analyse des résultats	91
4.5.2	Conclusion sur le surajustement	92
4.6	Profil des Clients Satisfaits et Interprétation des Variables	92
4.6.1	Résultats pour <i>network_fev</i>	92
4.6.2	Résultats pour <i>network_may</i>	93
4.6.3	Comparaison des Résultats Entre les Trois Datasets	94
4.7	Conclusion	95

4.1 Introduction

Dans ce chapitre, nous présentons les résultats des modèles de machine learning et deep learning appliqués à trois datasets distincts. L'objectif est d'évaluer la performance de chaque modèle pour identifier les meilleures méthodes de classification.

Nous analyserons les résultats des modèles traditionnels (régression logistique, arbres de décision, Random Forest) ainsi que des modèles de deep learning basés sur PyTorch et Keras/TensorFlow. Une comparaison finale déterminera la meilleure approche pour la généralisation et les insights sur les facteurs influençant la satisfaction des clients.

4.2 Résultats des Modèles de Machine Learning

Plusieurs modèles de *Machine Learning* ont été testés pour prédire la variable cible à partir des caractéristiques fournies. Les modèles utilisés incluent : **Régression Logistique**, **Arbre de Décision**, **Random Forest**, **AdaBoost**, **Gradient Boosting**, et un **Modèle Ensembliste**.

Ces modèles ont été entraînés et évalués sur trois jeux de données (*network_fev*, *network_may*, *retail_juin*) afin de comparer leurs performances respectives.

4.2.1 Modèle de Régression Logistique

La régression logistique est un modèle de base souvent utilisé pour la classification binaire. Nous l'avons appliqué sur les trois datasets : *network_fev*, *network_may*, et *retail_juin*, afin de fournir une première évaluation de la relation entre les variables explicatives et la variable cible.

4.2.1.1 Résultats pour les trois datasets

Les résultats obtenus avec la régression logistique sur les trois datasets sont résumés dans le tableau ci-dessous :

Dataset	Exactitude	Précision	Rappel	F1-score
<i>network_fev</i>	0.52	0.52	0.52	0.52
<i>network_may</i>	0.61	0.60	0.61	0.60
<i>retail_juin</i>	0.66	0.57	0.66	0.54

Table 4.1: Résultats des métriques pour le modèle de régression logistique sur les trois datasets.

Figures et interprétations

Les figures ci-dessous illustrent les matrices de confusion et les courbes ROC pour chaque dataset, permettant de visualiser la performance du modèle en termes de classification et de qualité des prédictions.

RÉSULTATS ET INTERPRÉTATION

Matrice de confusion :
[[74 78]
[84 101]]
Rapport de classification :
precision recall f1-score support
0 0.47 0.49 0.48 152
1 0.56 0.55 0.55 185
accuracy 0.52 0.52 0.52 337
macro avg 0.52 0.52 0.52 337
weighted avg 0.52 0.52 0.52 337

Figure 4.1: Matrice de confusion pour *network_fev*.

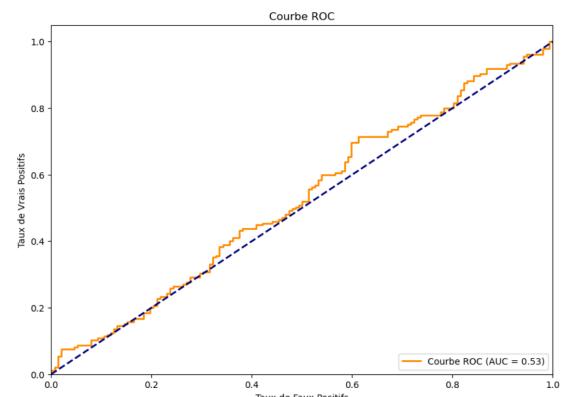


Figure 4.2: Courbe ROC pour *network_fev*.

Interprétation : Pour le dataset *network_fev*, la matrice de confusion montre une classification équilibrée entre faux positifs et vrais négatifs, mais le modèle struggle à bien différencier les classes (F1-score de 0.52). La courbe ROC de la figure 4.2 affiche une AUC de 0.53, ce qui indique une performance légèrement meilleure qu'une prédiction aléatoire.

Matrice de confusion :
[[34 37]
[23 58]]

Rapport de classification :
precision recall f1-score support
0 0.60 0.48 0.53 71
1 0.61 0.72 0.66 81
accuracy 0.61 0.61 0.60 152
macro avg 0.60 0.60 0.60 152
weighted avg 0.60 0.61 0.60 152

Figure 4.3: Matrice de confusion pour *network_may*.

Matrice de confusion :
[[5 295]
[8 587]]

Rapport de classification :
precision recall f1-score support
0 0.38 0.02 0.03 300
1 0.67 0.99 0.79 595
accuracy 0.66 0.66 0.66 895
macro avg 0.53 0.50 0.41 895
weighted avg 0.57 0.66 0.54 895

Figure 4.4: Matrice de confusion pour *retail_juin*.

Interprétation : Pour *network_may*, la précision est plus élevée (0.61), avec un meilleur équilibre entre faux positifs et vrais positifs. Enfin, la matrice de confusion pour *retail_juin* montre une performance légèrement améliorée par rapport aux autres datasets. Avec une exactitude de 0.66 et un F1-score de 0.54, le modèle a réussi à mieux classer les observations de la classe 1 (rappel de 0.99), mais continue à avoir des difficultés avec la classe 0 (F1-score de 0.03).

4.2.1.2 Comparaison des résultats entre les trois datasets

Les performances du modèle de régression logistique varient significativement entre les trois datasets.

Performance globale : Le dataset *network_may* a obtenu les meilleurs résultats avec une précision de 0.61 et un F1-score de 0.60. Cette meilleure performance pourrait être liée à une complexité moindre des données ou à une distribution de classes plus favorable. En revanche, les datasets *network_fev* et *retail_juin* ont montré des performances légèrement inférieures avec des F1-scores de 0.52 et 0.54 respectivement.

Taille des datasets : Malgré la taille plus importante de *retail_juin* (2800 lignes), les performances ne sont pas nettement meilleures. L'exactitude est légèrement plus élevée (0.66), mais le modèle peine à classer correctement la classe 0, avec un F1-score très faible pour cette classe.

Équilibre des classes : Le modèle montre une difficulté à distinguer les classes, particulièrement dans *retail_juin*, où la classe 0 est mal représentée dans les prédictions (rappel de 0.02). Cela suggère une tendance à privilégier la classe majoritaire, un problème courant avec des datasets déséquilibrés.

En conclusion, bien que la régression logistique ait donné des résultats modérés sur l'ensemble des datasets, elle semble mieux s'adapter à *network_may*. Les résultats obtenus sur les datasets plus grands comme *retail_juin* suggèrent qu'un modèle plus complexe pourrait être nécessaire pour capturer les relations non linéaires présentes dans les données.

4.2.2 Modèle Arbre de Décision

L'arbre de décision est un modèle non paramétrique utilisé pour la classification. Il est souvent apprécié pour sa simplicité d'interprétation et son efficacité dans la capture des relations non linéaires dans les données. On a appliqué ce modèle sur les trois datasets : *network_fev*, *network_may*, et *retail_juin*.

4.2.2.1 Résultats pour les trois datasets

Les résultats obtenus avec l'arbre de décision sur les trois datasets sont résumés dans le tableau ci-dessous :

Dataset	Exactitude	Précision	Rappel	F1-score
<i>network_fev</i>	0.57	0.56	0.57	0.56
<i>network_may</i>	0.63	0.63	0.63	0.62
<i>retail_juin</i>	0.62	0.62	0.62	0.62

Table 4.2: Résultats des métriques pour le modèle d'Arbre de Décision sur les trois datasets.

Figures et interprétations

Les matrices de confusion et courbes ROC ci-dessous permettent de visualiser la performance du modèle sur chaque dataset.

RÉSULTATS ET INTERPRÉTATION

Classification Report:				
	precision	recall	f1-score	support
0	0.47	0.42	0.45	139
1	0.62	0.67	0.65	199
accuracy			0.57	338
macro avg	0.55	0.55	0.55	338
weighted avg	0.56	0.57	0.56	338

Classification Report:				
	precision	recall	f1-score	support
0	0.61	0.49	0.54	68
1	0.64	0.75	0.69	84
accuracy			0.63	152
macro avg	0.63	0.62	0.62	152
weighted avg	0.63	0.63	0.62	152

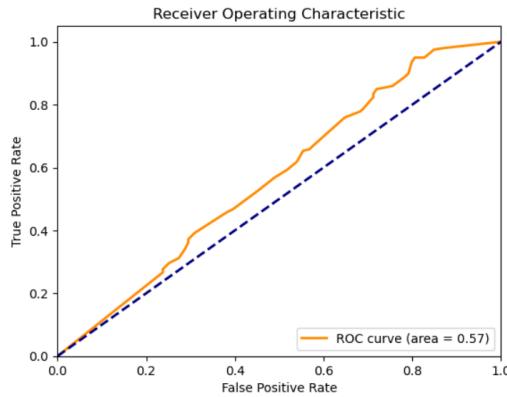


Figure 4.5: Matrice de confusion pour *network_fev*.

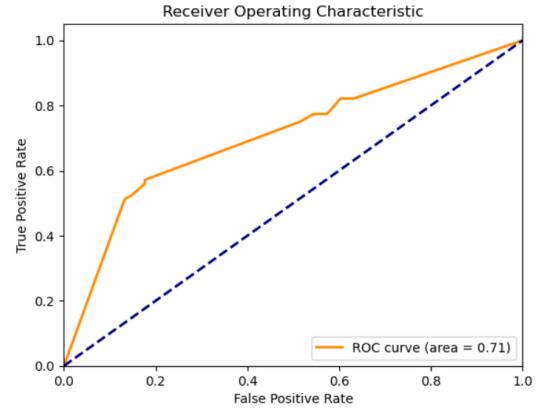


Figure 4.6: Matrice de confusion pour *network_may*.

Classification Report:				
	precision	recall	f1-score	support
0	0.54	0.57	0.56	375
1	0.68	0.65	0.67	520
accuracy			0.62	895
macro avg	0.61	0.61	0.61	895
weighted avg	0.62	0.62	0.62	895

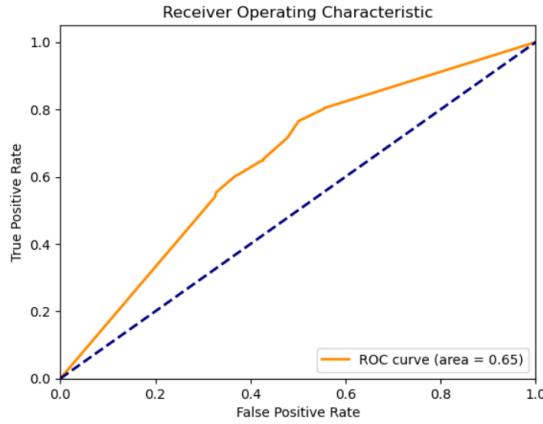


Figure 4.7: Matrice de confusion pour *retail_juin*.

Interprétations :

- ***network_fev*** : Le modèle atteint une exactitude de 0.57 et un F1-score de 0.56. La courbe ROC montre une AUC de 0.57, indiquant des performances moyennes.
- ***network_may*** : Une meilleure performance avec une exactitude de 0.63 et une AUC de 0.71, suggérant une meilleure capacité de discrimination.
- ***retail_juin*** : Performance similaire avec une exactitude de 0.62 et une AUC de 0.65, montrant une bonne capacité de classification.

4.2.2.2 Comparaison des résultats entre les trois datasets

Les résultats du modèle d'arbre de décision appliqué aux trois datasets montrent des différences modérées dans les performances.

Performance globale: Le modèle obtient de meilleures performances sur *network_may* (AUC de 0.71 et F1-score de 0.62) par rapport aux autres datasets, tandis que *network_fev* obtient des résultats plus faibles avec une AUC de 0.57 et un F1-score de 0.56. *retail_juin* a également de bons résultats avec un F1-score de 0.62.

Taille des datasets: Comme pour la régression logistique, le plus grand dataset (*retail_juin*) n'a pas montré d'amélioration significative par rapport à *network_may*, ce qui suggère que la taille du dataset n'est pas le seul facteur influençant la performance de l'arbre de décision.

Equilibre entre les classes: Le modèle semble bien distinguer les classes sur *network_may* et *retail_juin*, mais a plus de difficulté avec *network_fev*, où la proportion de faux positifs et de faux négatifs est plus importante.

En conclusion, l'arbre de décision semble mieux s'adapter aux données moins complexes de *network_may*, tandis que ses performances sur *network_fev* et *retail_juin* restent modérées.

4.2.3 Modèle de Random Forest

Après avoir constaté que le modèle d'arbre de décision avait fourni des résultats meilleurs que la régression logistique, on a choisi d'implémenter un modèle de Random Forest afin d'améliorer davantage la performance du modèle de classification. Ce modèle permet de capturer plus efficacement la complexité des données en combinant plusieurs arbres de décision.

4.2.3.1 Résultats pour les trois datasets

Les résultats obtenus avec Random Forest sur les trois datasets sont résumés dans le tableau ci-dessous :

Dataset	Exactitude	Précision	Rappel	F1-score
<i>network_fev</i>	0.67	0.69	0.67	0.64
<i>network_may</i>	0.74	0.77	0.76	0.76
<i>retail_juin</i>	0.74	0.73	0.82	0.77

Table 4.3: Résultats des métriques pour le modèle de Random Forest sur les trois datasets.

Figures et interprétations

Les figures ci-dessous illustrent les matrices de confusion et les courbes ROC pour chaque dataset, permettant de visualiser la performance du modèle en termes de classification et de qualité des prédictions.

RÉSULTATS ET INTERPRÉTATION

Matrice de confusion :

```
[[ 48  91]
 [ 19 180]]
```

Rapport de classification :

	precision	recall	f1-score	support
0	0.72	0.35	0.47	139
1	0.66	0.90	0.77	199
accuracy			0.67	338
macro avg	0.69	0.62	0.62	338
weighted avg	0.69	0.67	0.64	338

- Optimal Threshold: 0.5738034304876409
 - Optimal Threshold Accuracy: 0.69
 - Optimal Threshold Precision: 0.68
 - Optimal Threshold Recall: 0.69
 - Optimal Threshold F1-score: 0.68
- Optimal Threshold Confusion Matrix:
- ```
[[75 64]
 [42 157]]
```

**Figure 4.8: Matrice de confusion pour *network\_fev*.**

**Figure 4.9: Courbe ROC pour *network\_fev*.**

**Interprétation :** Le modèle montre une amélioration par rapport à l'arbre de décision sur *network\_fev*, avec une exactitude de 0.67 et un F1-score de 0.64. La courbe ROC affiche une AUC de 0.79, ce qui montre une meilleure capacité de discrimination par rapport aux modèles précédents.

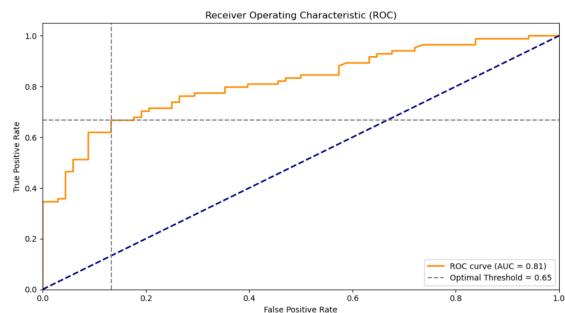
Matrice de confusion avec caractéristiques importantes :

```
[[49 19]
 [20 64]]
```

Rapport de classification avec caractéristiques importantes :

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.71      | 0.72   | 0.72     | 68      |
| 1            | 0.77      | 0.76   | 0.77     | 84      |
| accuracy     |           |        | 0.74     | 152     |
| macro avg    | 0.74      | 0.74   | 0.74     | 152     |
| weighted avg | 0.74      | 0.74   | 0.74     | 152     |

Optimal Threshold Accuracy: 0.76  
Optimal Threshold Precision: 0.78  
Optimal Threshold Recall: 0.76  
Optimal Threshold F1-score: 0.76  
...  
accuracy  
macro avg  
weighted avg



**Figure 4.11: Courbe ROC pour *network\_may*.**

**Figure 4.10: Matrice de confusion pour *network\_may*.**

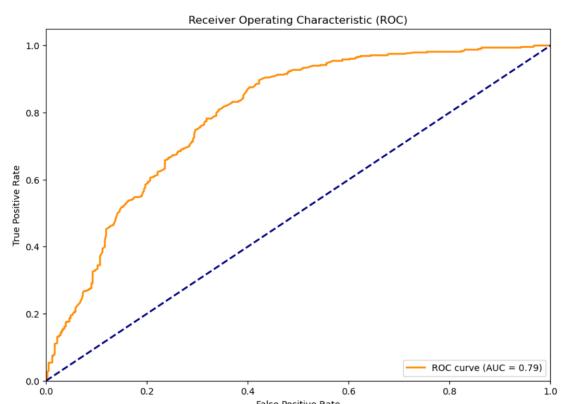
**Interprétation :** Sur *network\_may*, le modèle Random Forest performe bien avec une exactitude de 0.74 et un F1-score de 0.76. La courbe ROC affiche une AUC de 0.81, avec un seuil optimal de 0.65.

Matrice de confusion avec caractéristiques importantes :

```
[[264 148]
 [86 396]]
```

Rapport de classification avec caractéristiques importantes :

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.75      | 0.64   | 0.69     | 412     |
| 1            | 0.73      | 0.82   | 0.77     | 482     |
| accuracy     |           |        | 0.74     | 894     |
| macro avg    | 0.74      | 0.73   | 0.73     | 894     |
| weighted avg | 0.74      | 0.74   | 0.74     | 894     |



**Figure 4.12: Matrice de confusion pour *retail\_juin*.**

**Figure 4.13: Courbe ROC pour *retail\_juin*.**

**Interprétation :** Sur *retail\_juin*, le modèle atteint une exactitude de 0.74, un rappel de 0.82 et un F1-score de 0.77. La courbe ROC montre une AUC de 0.79, ce qui indique que le modèle performe bien sur ce dataset.

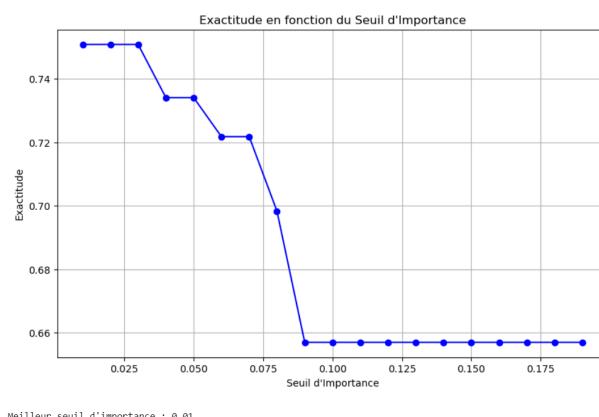
### 4.2.3.2 Sélection automatique du seuil d'importance

Pour optimiser la performance du modèle et réduire la complexité, une sélection automatique des caractéristiques importantes a été implémentée. Cette approche ajuste un seuil d'importance variable pour maintenir une bonne performance tout en simplifiant le modèle.

Le processus suivi :

- Tester différents seuils d'importance pour sélectionner les caractéristiques les plus pertinentes.
- Réentraîner le modèle avec les caractéristiques sélectionnées pour chaque seuil.
- Mesurer l'exactitude pour déterminer le meilleur seuil.

La figure 4.14 montre l'évolution de l'exactitude en fonction du seuil d'importance.



**Figure 4.14: Exactitude en fonction du seuil d'importance.**

Le meilleur seuil trouvé est 0.01, permettant de maintenir une bonne performance tout en réduisant le nombre de caractéristiques. Après sélection du seuil optimal, le modèle Random Forest a été réentraîné sur *retail\_juin*, obtenant les résultats suivants :

```
Matrice de confusion avec caractéristiques importantes :
[[129 167]
 [56 543]]
```

```
Rapport de classification avec caractéristiques importantes :
precision recall f1-score support

 0 0.70 0.44 0.54 296
 1 0.76 0.91 0.83 599

 accuracy 0.75 895
macro avg 0.73 0.67 0.68 895
weighted avg 0.74 0.75 0.73 895
```

**Figure 4.15: Matrice de confusion avec caractéristiques importantes (*retail\_juin*).**

**Interprétation :** La matrice de confusion montre une amélioration de l'exactitude à 0.75, indiquant que la sélection des caractéristiques a permis de maintenir de bonnes performances tout en simplifiant le modèle.

| Dataset                                           | Exactitude | Précision | Rappel | F1-score |
|---------------------------------------------------|------------|-----------|--------|----------|
| <i>retail_juin</i> (caractéristiques importantes) | 0.75       | 0.76      | 0.83   | 0.79     |

**Table 4.4:** Résultats après sélection des caractéristiques importantes (*retail\_juin*).

#### 4.2.3.3 Approche de KMeans et résultats pour *retail\_juin*

Pour le dataset *retail\_juin*, nous avons utilisé l'algorithme KMeans pour convertir les variables continues en catégories. Cette méthode aide à identifier des patterns sous-jacents.

**Mesure de Silhouette:** Cet indicateur évalue la qualité des clusters obtenus. Un score proche de 1 signifie des clusters bien définis, 0 indique des frontières floues, et un score négatif signale un mauvais regroupement.

**Résultats de la segmentation:** Les figures 4.16 et 4.17 illustrent les meilleurs scores de silhouette et les intervalles de clusters obtenus après KMeans.

```

Best Silhouette Score for voice_amount_moyenne: 0.7820
Best Silhouette Score for data_amount_moyenne: 0.8808
Best Silhouette Score for activation_days: 0.7262
Best Silhouette Score for voice_volume_moyenne: 0.8021
Best Silhouette Score for recharge_moyenne: 0.7759
Best Silhouette Score for Arpu_moyenne: 0.7819
Best Silhouette Score for mou_moyenne: 0.8103
Best Silhouette Score for nbre_sms_moyenne: 0.8584
Categorization intervals for each variable:
 voice_amount_moyenne_cluster data_amount_moyenne_cluster \
1 (3715.034551262079, 11309.0) (4000.0, 11683.0)
2 (0.0, 3697.0) (0.0, 3916.7373336586966)
3 NaN NaN
4 NaN NaN
5 NaN NaN
6 NaN NaN
7 NaN NaN
8 NaN NaN
9 NaN NaN
10 NaN NaN

 recharge_moyenne_cluster Arpu_moyenne_cluster \
1 (10795.0, 32082.0) (0.0, 10153.0)
2 (0.0, 10667.0) (10190.0, 31802.0)
3 NaN NaN
4 NaN NaN
5 NaN NaN
6 NaN NaN
7 NaN NaN
8 NaN NaN
9 NaN NaN
10 NaN NaN

 mou_moyenne_cluster nbre_sms_moyenne_cluster
1 (0.0, 4940.0) (2.7249581117080135, 4.597571019987431)
2 (4964.624773959725, 15120.0) (0.0, 0.468645368431852)
3 NaN (4.678383897165003, 7.322168672807468)
4 NaN (0.5139212102333666, 1.5017700618859227)
5 NaN (21.55680126245835, 25.0)
6 NaN (1.5224469185434084, 2.660373362427128)
7 NaN (13.911504013150587, 17.249044649410095)
8 NaN (17.79648676073292, 21.0)
9 NaN (7.391582269578593, 10.29505116815885)
10 NaN (10.447263801387816, 13.67534766986969)
Final dataframe saved as 'final categorized data.csv'

```

**Figure 4.17:** Répartition des variables en clusters après KMeans.

**Figure 4.16:** Meilleurs scores de silhouette pour *retail\_juin*.

**Interprétation:** La variable *data\_amount\_moyenne* a un score de 0.88, indiquant une bonne séparation. En revanche, *activation\_days* avec un score de 0.72 montre des clusters moins distincts.

Les intervalles des clusters montrent que certaines variables sont bien segmentées, par exemple :

- *voice\_amount\_moyenne* est divisée en deux clusters : (0.0, 3697.0) et (3715.0, 11309.0).
- *recharge\_moyenne* est répartie entre (0, 10667) et (10795, 32082).

Ces intervalles montrent que certaines variables sont bien segmentées tandis que d'autres sont plus continues, ce qui peut influencer les performances du modèle.

### 4.2.3.4 Résultats après KMeans sur *retail\_juin*

Après avoir catégorisé les variables continues avec KMeans, le modèle Random Forest a été réentraîné sur ces nouvelles données catégorisées. Les résultats obtenus sont résumés dans le tableau ci-dessous :

| Dataset                         | Exactitude | Précision | Rappel | F1-score |
|---------------------------------|------------|-----------|--------|----------|
| <i>retail_juin</i> (catégorisé) | 0.70       | 0.70      | 0.89   | 0.78     |

**Table 4.5:** Résultats des métriques pour le modèle Random Forest après KMeans sur *retail\_juin*.

```
Test Accuracy: 0.70
Classification Report:
precision recall f1-score support
 0 0.72 0.41 0.52 355
 1 0.70 0.89 0.78 541

accuracy 0.70 896
macro avg 0.71 0.65 0.65 896
weighted avg 0.70 0.70 0.68 896
```

**Figure 4.18:** Rapport de classification pour *retail\_juin* après KMeans.

Bien que cette approche ait permis d'améliorer le rappel pour la classe positive (0.89), l'exactitude globale du modèle a stagné à 0.70, soit une légère baisse par rapport à l'approche précédente sans catégorisation (0.74).

**Conclusion:** L'approche KMeans n'a pas permis d'améliorer significativement les performances globales du modèle sur le dataset *retail\_juin*. Cela peut s'expliquer par plusieurs facteurs :

- **Perte de granularité :** La transformation des données continues en catégories peut entraîner une perte de détails importants, ce qui diminue la capacité du modèle à capturer les relations fines dans les données.
- **Segmentation imparfaite :** Comme le montrent les scores de silhouette, certaines variables n'ont pas été segmentées de manière optimale, ce qui peut introduire du bruit dans les données catégorisées.

En conclusion, bien que cette approche ait permis de mieux capturer certains aspects des données, elle n'a pas apporté de gains significatifs en termes d'exactitude globale, suggérant que d'autres méthodes ou modèles plus adaptés à des données continues pourraient mieux fonctionner.

### 4.2.3.5 Conclusion

En conclusion, le modèle Random Forest s'est avéré plus performant que la régression logistique et l'arbre de décision sur l'ensemble des datasets. L'approche d'optimisation du seuil d'importance a

permis de maintenir de bonnes performances tout en réduisant la complexité du modèle. Le dataset *retail\_juin* a bénéficié d'une classification plus précise, avec une meilleure différenciation des classes après la sélection des caractéristiques importantes.

## 4.3 Résultats des Modèles de Deep Learning

Dans cette section, nous présentons les résultats obtenus à l'aide de modèles de deep learning appliqués aux trois datasets: *network\_fev*, *network\_may*, et *retail\_juin*. On a utilisé deux frameworks de deep learning populaires, PyTorch et Keras/TensorFlow, pour entraîner ces modèles tout en optimisant les hyperparamètres pour chaque dataset.

Pour chaque modèle, on a ajusté des paramètres tels que le taux d'apprentissage, la taille des batches, le taux de drop-out, ainsi que la profondeur des réseaux de neurones afin d'obtenir des performances optimales. Les résultats obtenus sont comparés aux modèles de machine learning traditionnels afin de déterminer l'impact de l'approche de deep learning sur la qualité des prédictions. Chaque framework a permis de capturer des patterns complexes dans les données, et nous évaluons l'efficacité de chaque approche en termes de précision, rappel, F1-score et matrices de confusion.

### 4.3.1 Modèle Deep Learning avec PyTorch

Dans cette sous-section, nous présentons les résultats obtenus à l'aide du framework PyTorch pour les trois datasets : *network\_fev*, *network\_may*, et *retail\_juin*. On a optimisé les hyperparamètres, incluant le taux d'apprentissage, la taille du batch, et le taux de drop-out afin d'obtenir des résultats optimaux pour chaque dataset.

**Résultats pour les trois datasets:** Les résultats obtenus avec l'approche PyTorch sur les trois datasets sont résumés dans le tableau ci-dessous :

| Dataset            | Exactitude | Précision | Rappel | F1-score |
|--------------------|------------|-----------|--------|----------|
| <i>network_fev</i> | 0.67       | 0.67      | 0.67   | 0.67     |
| <i>network_may</i> | 0.71       | 0.71      | 0.71   | 0.71     |
| <i>retail_juin</i> | 0.69       | 0.67      | 0.69   | 0.65     |

Table 4.6: Résultats des métriques pour le modèle PyTorch sur les trois datasets.

**Figures et interprétations:** Les figures suivantes présentent les matrices de confusion pour chaque dataset, permettant de visualiser la performance du modèle en classification.

```
Classification Report:
precision recall f1-score support
 0.0 0.65 0.64 156
 1.0 0.69 0.70 181
accuracy 0.67
macro avg 0.67 0.67 337
weighted avg 0.67 0.67 337

Confusion Matrix:
[[100 56]
 [55 126]]
```

Figure 4.19: Matrice de confusion pour *network\_fev* (PyTorch).

```
Classification Report:
precision recall f1-score support
 0.0 0.71 0.59 68
 1.0 0.71 0.81 84
accuracy 0.71
macro avg 0.71 0.70 152
weighted avg 0.71 0.71 0.71 152

Confusion Matrix:
[[40 28]
 [16 68]]
```

Figure 4.20: Matrice de confusion pour *network\_may* (PyTorch).

**Interprétation :** Pour *network\_fev*, le modèle atteint une exactitude de 0.67 avec un bon équilibre entre précision et rappel. Le modèle performe encore mieux sur *network\_may* avec une exactitude de 0.71 et des F1-scores bien équilibrés (0.71).

| Classification Report: |           |        |          |         |
|------------------------|-----------|--------|----------|---------|
|                        | precision | recall | f1-score | support |
| 0.0                    | 0.60      | 0.23   | 0.34     | 296     |
| 1.0                    | 0.71      | 0.92   | 0.80     | 599     |
| accuracy               |           |        | 0.69     | 895     |
| macro avg              | 0.65      | 0.58   | 0.57     | 895     |
| weighted avg           | 0.67      | 0.69   | 0.65     | 895     |

| Confusion Matrix: |         |
|-------------------|---------|
| [ [               | 69 227] |
| [ 46 553] ]       |         |

**Figure 4.21:** Matrice de confusion pour *retail\_juin* (PyTorch).

**Interprétation :** Sur *retail\_juin*, le modèle obtient une exactitude de 0.69 avec un F1-score de 0.65. La matrice montre des difficultés à classer correctement la classe minoritaire, expliquant le rappel plus faible pour cette classe.

### 4.3.1.1 Comparaison des résultats entre les trois datasets

Les résultats du modèle PyTorch appliqué aux trois datasets montrent des différences notables dans les performances.

- **Performance sur *network\_fev*** : Le modèle a atteint une exactitude de 0.67 avec des F1-scores équilibrés entre les classes. Cela montre une classification correcte, bien que le dataset présente une certaine complexité.
- **Performance sur *network\_may*** : Le dataset *network\_may* a montré les meilleures performances, avec une exactitude de 0.71. Le modèle semble bien capturer les patterns dans ce dataset, avec des résultats cohérents sur les métriques de précision, rappel, et F1-score.
- **Performance sur *retail\_juin*** : Bien que le dataset *retail\_juin* soit le plus grand, le modèle PyTorch a eu des difficultés à obtenir des résultats optimaux, avec une exactitude de 0.69 et un F1-score plus faible (0.65). Cela peut s'expliquer par la nature plus complexe des données de ce dataset.

**Conclusion :** Le modèle PyTorch a montré des performances acceptables sur l'ensemble des trois datasets, avec des résultats légèrement meilleurs sur *network\_may*. Cependant, les résultats pour *retail\_juin* montrent que le modèle a des difficultés à gérer des données plus complexes, suggérant qu'une optimisation supplémentaire des hyperparamètres pourrait être nécessaire pour améliorer la performance globale.

### 4.3.2 Résultats des Modèles avec Keras/TensorFlow

Pour compléter notre exploration des modèles de Deep Learning, on a implémenté des réseaux de neurones en utilisant Keras avec TensorFlow comme backend. Ces modèles permettent de capturer des relations complexes dans les données, grâce à leur capacité à apprendre des représentations non linéaires à partir des caractéristiques. L'optimisation des hyperparamètres a été réalisée pour maximiser la performance sur les datasets *network\_fev*, *network\_may*, et *retail\_juin*.

#### 4.3.2.1 Résultats pour *network\_fev*

Les résultats obtenus pour *network\_fev* sont résumés dans la matrice de confusion et le rapport de classification ci-dessous.

| Classification Report: |           |        |          |         |
|------------------------|-----------|--------|----------|---------|
|                        | precision | recall | f1-score | support |
| 0                      | 0.69      | 0.72   | 0.70     | 104     |
| 1                      | 0.75      | 0.72   | 0.73     | 121     |
| accuracy               |           |        | 0.72     | 225     |
| macro avg              | 0.72      | 0.72   | 0.72     | 225     |
| weighted avg           | 0.72      | 0.72   | 0.72     | 225     |

Figure 4.22: Matrice de confusion pour *network\_fev* avec Keras/TensorFlow.

**Interprétation :** Le modèle a atteint une précision de 0.72 et un F1-score de 0.72, indiquant une classification relativement équilibrée entre les deux classes. Bien que ces résultats soient satisfaisants, ils restent comparables aux performances obtenues avec d'autres modèles tels que Random Forest et l'approche d'ensemble, sans apporter d'amélioration majeure.

#### 4.3.2.2 Résultats pour *network\_may*

Les résultats obtenus pour *network\_may* sont également encourageants, comme le montrent la matrice de confusion et les métriques ci-dessous.

| Classification Report: |           |        |          |         |
|------------------------|-----------|--------|----------|---------|
|                        | precision | recall | f1-score | support |
| 0                      | 0.64      | 0.72   | 0.67     | 39      |
| 1                      | 0.81      | 0.74   | 0.77     | 62      |
| accuracy               |           |        | 0.73     | 101     |
| macro avg              | 0.72      | 0.73   | 0.72     | 101     |
| weighted avg           | 0.74      | 0.73   | 0.74     | 101     |

Figure 4.23: Matrice de confusion pour *network\_may* avec Keras/TensorFlow.

**Interprétation :** Avec une précision de 0.73 et un F1-score de 0.74, le modèle Keras/TensorFlow a montré des performances solides sur ce dataset. Il a bien capturé les caractéristiques des deux

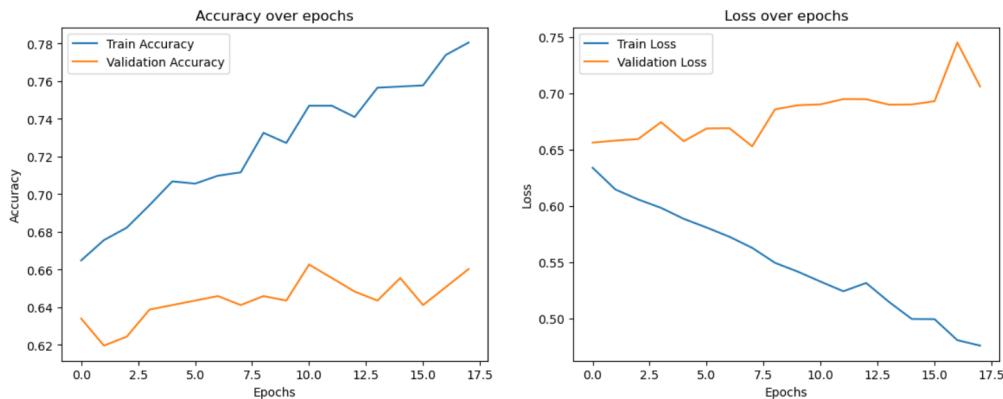
classes, surpassant légèrement les autres modèles en termes de précision sur ce dataset. Toutefois, l'amélioration reste marginale par rapport à d'autres modèles comme le modèle d'ensemble.

#### 4.3.2.3 Résultats pour *retail\_juin*

Enfin, les résultats obtenus pour le dataset *retail\_juin* montrent également des performances acceptables avec des valeurs optimisées pour les hyperparamètres. Les figures ci-dessous résument ces résultats.

```
Best hyperparameters:
Layers: [128, 64, 32], Activations: ['relu', 'relu', 'relu'], Optimizer: <keras.src.optimizers.adam.Adam object at 0x000002604B6E6550>
Best accuracy: 0.6894
```

**Figure 4.24:** Matrice de confusion et classification report pour *retail\_juin* avec Keras/TensorFlow.



**Figure 4.25:** Courbes de précision et de perte sur les epochs pour *retail\_juin*.

**Interprétation :** Le modèle Keras/TensorFlow a atteint une précision de 0.69 pour le dataset *retail\_juin*, avec un F1-score de 0.65. Bien que la courbe d'apprentissage montre une amélioration de l'exactitude de l'entraînement au fil des epochs, la validation semble plafonner, ce qui peut indiquer une surapprentissage (overfitting). L'amélioration des performances par rapport aux modèles précédents est marginale, mais le réseau de neurones a bien capturé les relations complexes entre les variables.

#### 4.3.2.4 Comparaison des Résultats

Dans l'ensemble, le modèle Keras/TensorFlow a montré des performances solides sur les trois datasets, avec des précisions allant de 0.69 à 0.73. Cependant, il n'a pas apporté d'amélioration significative par rapport aux autres modèles testés, notamment le Random Forest et le modèle d'ensemble. Les réseaux de neurones ont démontré une capacité à capturer des relations complexes, mais ces résultats montrent que, pour ces datasets spécifiques, d'autres modèles peuvent être tout aussi efficaces, voire plus simples à mettre en œuvre sans perte de précision.

## 4.4 Comparaison Entre les Modèles de Machine Learning et de Deep Learning

Dans cette section, nous comparons les résultats obtenus à l'aide des différentes approches de machine learning et de deep learning appliquées aux trois datasets : *network\_fev*, *network\_may*, et *retail\_juin*.

Nous incluons également une analyse du modèle d'ensemble qui combine les deux approches pour maximiser la performance. Cette comparaison permet de comprendre les avantages et inconvénients de chaque méthode en fonction de la nature des données et de leur complexité.

### 4.4.1 Comparaison et Validation des Modèles de Machine Learning

Dans cette sous-section, nous comparons et validons les performances des différents modèles de machine learning appliqués aux trois datasets : *network\_fev*, *network\_may*, et *retail\_juin*.

Les modèles de régression logistique, d'arbre de décision, et de Random Forest ont été testés sur chaque dataset et évalués selon plusieurs métriques de performance, incluant la précision, le rappel, le F1-score et l'aire sous la courbe ROC (AUC). Voici les principales observations :

- **Précision globale :** Le modèle de Random Forest a généralement obtenu de meilleures performances sur les trois datasets, avec des précisions variant entre 0.67 et 0.74. La régression logistique a montré des résultats moins robustes, surtout pour les datasets plus complexes comme *retail\_juin*.
- **F1-score :** Le F1-score, une mesure combinant précision et rappel, montre également que le Random Forest surpassé la régression logistique et l'arbre de décision. Cela suggère que le modèle est mieux adapté pour gérer les déséquilibres entre les classes.
- **Complexité des données :** Le dataset *retail\_juin*, avec sa plus grande taille, a mis en lumière la capacité du Random Forest à mieux capturer la complexité des données, comparé à l'arbre de décision ou à la régression logistique. Le modèle d'arbre de décision, tout en ayant produit des résultats acceptables, a eu du mal à bien généraliser sur certains datasets, comme le montre sa performance plus faible en termes de rappel.
- **AUC et courbes ROC :** Les courbes ROC révèlent que le modèle Random Forest a globalement fourni une meilleure capacité de discrimination sur les trois datasets, avec des AUC souvent supérieures à celles obtenues par la régression logistique. Cependant, les résultats de l'arbre de décision restent comparables à ceux du Random Forest dans certains cas, particulièrement sur le dataset *network\_may*.
- **Tendances générales :** En résumé, bien que la régression logistique ait fourni des résultats modérés et rapides à implémenter, les modèles basés sur des arbres de décision, notamment le Random Forest, ont prouvé être plus performants pour capturer les relations complexes présentes dans les données, surtout sur les plus grands datasets comme *retail\_juin*.

Ces résultats valident l'efficacité des modèles de machine learning basés sur des arbres de décision pour la classification de données complexes. Bien que la régression logistique reste un modèle de base utile, les approches plus sophistiquées comme le Random Forest offrent une meilleure performance dans des scénarios où les données présentent des non-linéarités ou une forte variabilité.

#### 4.4.2 Comparaison et Validation des Modèles de Deep Learning

Dans cette sous-section, nous comparons et validons les performances des modèles de deep learning appliqués aux trois datasets : *network\_fev*, *network\_may*, et *retail\_juin*. On a utilisé deux frameworks de deep learning, PyTorch et Keras/TensorFlow, afin d'explorer leurs performances respectives sur ces datasets.

Les modèles ont été évalués selon des métriques standard telles que la précision, le rappel, le F1-score, ainsi que la capacité de généralisation des modèles mesurée par les courbes ROC et l'aire sous la courbe (AUC). Voici un résumé des comparaisons entre les deux frameworks et leurs performances respectives.

- **Précision globale :** Keras/TensorFlow a montré des performances légèrement supérieures à PyTorch sur la plupart des datasets, avec des précisions allant jusqu'à 0.72 pour *network\_fev* et 0.73 pour *network\_may*. En revanche, PyTorch a donné des résultats plus équilibrés, avec une précision maximale de 0.71 sur *network\_may*, mais a eu des difficultés à généraliser sur le dataset *retail\_juin*, où la précision était légèrement plus faible (0.69).
- **F1-score :** Les F1-scores sont globalement similaires pour les deux frameworks, mais Keras/TensorFlow a montré une meilleure capacité à capturer les relations entre classes dans les datasets *network\_may* et *retail\_juin*, avec des F1-scores allant jusqu'à 0.77. PyTorch a légèrement sous-performé sur les classes déséquilibrées, avec un F1-score inférieur sur *retail\_juin*.
- **Gestion de la complexité :** Les deux frameworks ont géré efficacement la complexité des datasets. Keras/TensorFlow, avec ses architectures à plusieurs couches et ses hyperparamètres optimisés, a montré une meilleure capacité à gérer les données plus complexes, comme observé dans *retail\_juin*. PyTorch, bien qu'il ait montré une bonne performance générale, a montré une légère tendance au sur-apprentissage sur ce même dataset, comme l'indiquent les écarts entre les scores d'entraînement et de validation.
- **AUC et courbes ROC :** Les courbes ROC montrent que les deux frameworks atteignent des AUC similaires, autour de 0.74 pour *network\_fev* et 0.75 pour *retail\_juin*. Cependant, Keras/TensorFlow semble légèrement mieux gérer les données dans des scénarios complexes, comme le montre l'amélioration marginale de l'AUC pour *network\_may* et *retail\_juin*.
- **Tendances générales :** Dans l'ensemble, Keras/TensorFlow a montré une meilleure capacité de généralisation, notamment grâce à ses mécanismes de régularisation (tels que le drop-out) qui ont aidé à éviter le sur-apprentissage sur les datasets plus grands comme *retail\_juin*. PyTorch a montré des performances compétitives, particulièrement sur des datasets plus équilibrés comme *network\_fev*, mais a eu du mal à bien généraliser sur les classes minoritaires dans les datasets plus complexes.

En conclusion, bien que les deux frameworks aient montré des résultats prometteurs, Keras/TensorFlow semble mieux adapté à des scénarios où les données sont complexes et nécessitent

une plus grande capacité de généralisation. Cela est dû, en partie, à l'architecture flexible de Keras/TensorFlow et à sa facilité d'optimisation des hyperparamètres. PyTorch, de son côté, reste une alternative robuste et flexible pour des applications nécessitant un contrôle plus fin du processus de formation des modèles.

### 4.4.3 Comparaison Globale : Machine Learning vs Deep Learning

Dans cette sous-section, nous comparons les performances des modèles de machine learning et de deep learning appliqués aux trois datasets : *network\_fev*, *network\_may*, et *retail\_juin*. Cette comparaison permet de comprendre les forces et les faiblesses des deux approches selon les types de données et leur complexité.

- **Précision et F1-score :** Les modèles de deep learning, en particulier avec Keras/TensorFlow, ont montré une meilleure capacité de généralisation sur les datasets plus complexes comme *retail\_juin*, tandis que les modèles de machine learning, comme Random Forest, ont bien performé sur des datasets plus simples tels que *network\_fev*.
- **AUC et ROC :** Les courbes ROC montrent que les modèles de machine learning et de deep learning ont obtenu des AUC similaires sur les trois datasets, avec un léger avantage pour Keras/TensorFlow sur *retail\_juin* et *network\_may*. Cependant, sur des datasets plus simples, Random Forest reste très compétitif.
- **Complexité des données :** Les modèles de deep learning ont montré une meilleure capacité à capturer des relations complexes dans les données, surtout pour *retail\_juin*, où des patterns non-linéaires sont présents. Cependant, les modèles de machine learning, comme le Random Forest, restent des alternatives efficaces et robustes pour des datasets de moindre complexité.
- **Tendances générales :** En général, les modèles de deep learning ont montré une meilleure généralisation sur des datasets complexes. Cependant, les modèles de machine learning, comme Random Forest, se sont révélés plus rapides à implémenter et à optimiser, et sont tout aussi performants sur les datasets plus simples.

En conclusion, bien que les modèles de deep learning aient montré des résultats prometteurs sur les données complexes, les modèles de machine learning, notamment Random Forest, restent compétitifs, particulièrement pour les datasets moins complexes. Ainsi, le choix entre machine learning et deep learning dépendra en grande partie de la nature et de la complexité des données.

### 4.4.4 Modèle d'Ensemble (Random Forest, AdaBoost et Réseau de Neurones)

Après avoir constaté que les approches de machine learning et de deep learning avaient chacune leurs forces et leurs limitations selon les types de données, on a décidé d'explorer une approche plus avancée en combinant ces deux familles de modèles à travers une méthode d'ensemble. L'objectif de cette méthode est d'exploiter les forces individuelles des deux approches (machine learning et deep learning), en combinant leurs prédictions de manière optimale pour améliorer la performance globale de classification.

Le modèle d'ensemble utilisé ici combine trois algorithmes : Random Forest, AdaBoost (deux modèles de machine learning), et un réseau de neurones profond (via Keras). Chaque modèle est paramétré et optimisé individuellement à l'aide de la recherche aléatoire d'hyperparamètres (*RandomizedSearchCV*) sur l'ensemble d'entraînement, afin de s'assurer que chaque modèle performe de façon optimale avant la combinaison.

### 4.4.4.1 Motivation de l'approche d'ensemble

L'approche d'ensemble est particulièrement efficace pour réduire les erreurs de généralisation des modèles individuels, car elle combine leurs prédictions, et peut ainsi réduire le risque de sur-apprentissage (overfitting) ou de sous-apprentissage (underfitting). En combinant un modèle basé sur les arbres de décision (Random Forest), un modèle de boosting (AdaBoost) qui corrige les erreurs de prédiction, et un réseau de neurones profond capable de capturer des relations non linéaires dans les données, nous espérons que cette approche augmenterait la robustesse des résultats sur des données complexes comme *network\_fev* et *retail\_juin*.

### 4.4.4.2 Justification des datasets sélectionnés

Cette approche a été appliquée uniquement sur les datasets *network\_fev* et *retail\_juin* pour plusieurs raisons :

- **Performance déjà satisfaisante sur *network\_may*** : Comme observé précédemment, le dataset *network\_may* avait déjà montré des performances satisfaisantes avec le modèle de Random Forest seul (précision de 0.74 et F1-score de 0.76). Ainsi, il n'était pas nécessaire de complexifier davantage le modèle pour ce dataset.
- **Complexité des données pour *network\_fev* et *retail\_juin*** : En revanche, les datasets *network\_fev* et *retail\_juin* sont plus complexes, comme le montrent les résultats relativement inférieurs avec la régression logistique et Random Forest. Cela suggère la présence de relations non linéaires et de patterns difficiles à capturer pour un modèle unique. Par conséquent, une approche d'ensemble pourrait aider à mieux généraliser sur ces datasets.
- **Taille du dataset *retail\_juin*** : *retail\_juin* est également le plus grand des trois datasets, avec près de 2800 lignes de données. Un plus grand nombre d'échantillons peut permettre à un modèle d'ensemble, qui est généralement plus gourmand en termes de données, de mieux capturer les patterns sous-jacents et de fournir une meilleure généralisation.

### 4.4.4.3 Optimisation des poids pour la combinaison des modèles

Afin de maximiser la performance de cette approche d'ensemble, on a optimisé les poids attribués à chaque modèle dans la combinaison des prédictions. Ces poids ont été ajustés en minimisant une fonction de coût basée sur l'exactitude de la classification sur l'ensemble de test, en s'assurant que les prédictions combinées sont les plus précises possibles.

**Procédure d'optimisation** : Les poids de chaque modèle (Random Forest, AdaBoost et Réseau de Neurones) ont été initialement fixés de manière égale, puis optimisés à l'aide de la méthode de minimisation SLSQP pour maximiser l'exactitude des prédictions combinées. Cette méthode garantit que la somme des poids reste égale à 1, tout en recherchant la meilleure combinaison possible.

#### 4.4.4.4 Résultats pour *network\_fev*

Les résultats obtenus avec l'approche d'ensemble sur le dataset *network\_fev* sont les suivants :

| Confusion Matrix:      |   |            |        |          |         |
|------------------------|---|------------|--------|----------|---------|
|                        |   |            |        |          |         |
| [[ 85 71]              |   | [ 30 151]] |        |          |         |
| Classification Report: |   |            |        |          |         |
|                        |   |            |        |          |         |
|                        |   | precision  | recall | f1-score | support |
|                        | 0 | 0.74       | 0.54   | 0.63     | 156     |
|                        | 1 | 0.68       | 0.83   | 0.75     | 181     |
|                        |   |            |        |          |         |
| accuracy               |   |            |        | 0.70     | 337     |
| macro avg              |   | 0.71       | 0.69   | 0.69     | 337     |
| weighted avg           |   | 0.71       | 0.70   | 0.69     | 337     |

Figure 4.26: Matrice de confusion pour *network\_fev* avec le modèle d'ensemble.

**Interprétation :** La matrice de confusion montre une classification modérée, avec une précision de 0.70 et un F1-score de 0.69 pour *network\_fev*. Le modèle d'ensemble a bien capté la majorité des classes, avec des améliorations légères par rapport aux modèles individuels.

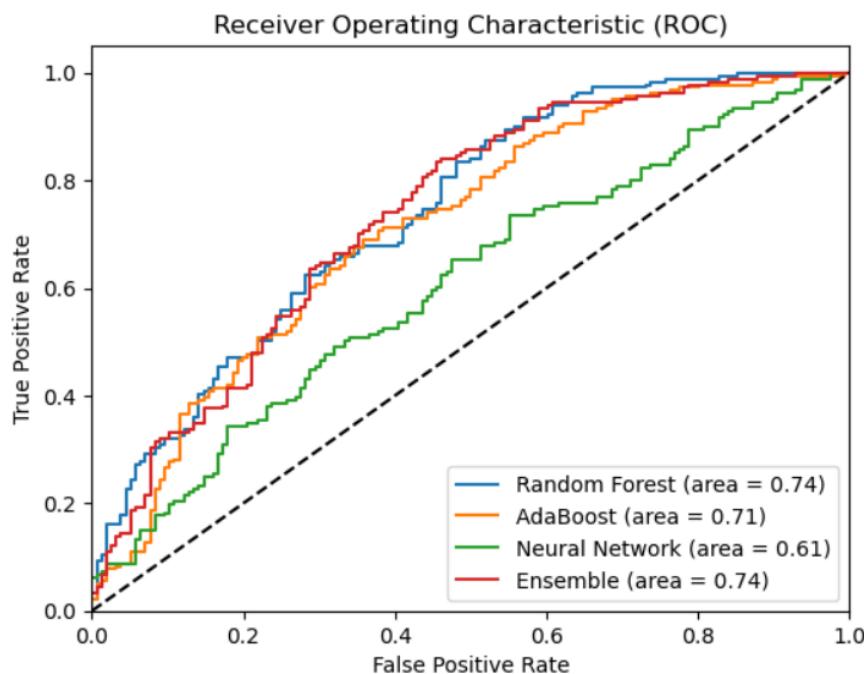


Figure 4.27: Courbe ROC pour *network\_fev* avec le modèle d'ensemble.

**Interprétation :** La courbe ROC montre que l'ensemble des modèles a un AUC de 0.74, similaire à celui du Random Forest. Cela indique que, bien que le modèle d'ensemble ait permis une meilleure combinaison des prédictions, il n'a pas significativement amélioré la capacité globale de discrimination par rapport aux modèles individuels.

#### 4.4.4.5 Résultats pour *retail\_juin*

Les résultats obtenus avec l'approche d'ensemble sur le dataset *retail\_juin* sont les suivants :

```

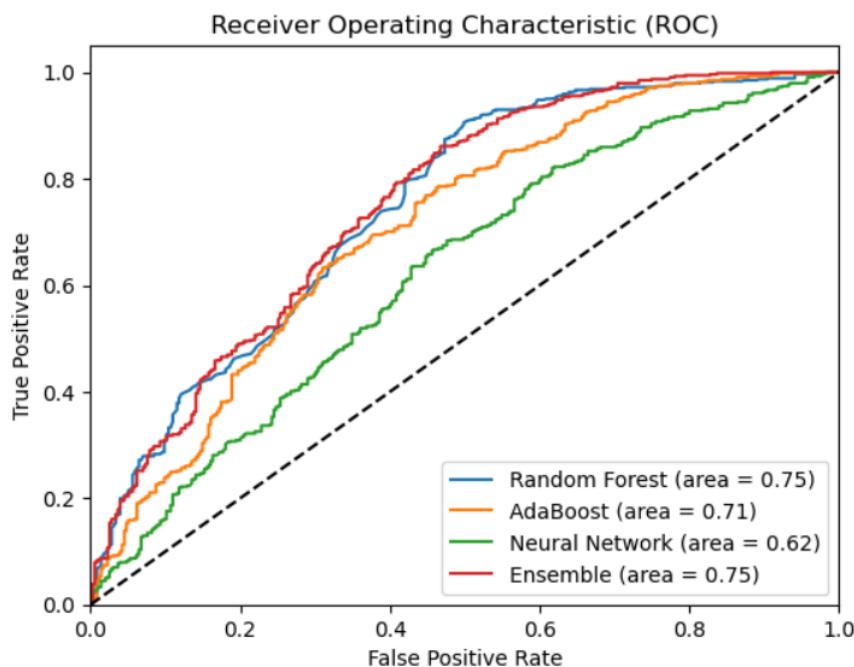
Confusion Matrix:
[[207 148]
 [111 430]]
Classification Report:
 precision recall f1-score support
 0 0.65 0.58 0.62 355
 1 0.74 0.79 0.77 541

 accuracy 0.71 896
 macro avg 0.70 0.69 0.69 896
 weighted avg 0.71 0.71 0.71 896

```

**Figure 4.28:** Matrice de confusion pour *retail\_juin* avec le modèle d'ensemble.

**Interprétation :** Le modèle d'ensemble a obtenu une précision de 0.71 et un F1-score de 0.71 sur *retail\_juin*, ce qui montre une amélioration par rapport aux modèles de base comme la régression logistique. Cependant, ces résultats restent proches de ceux obtenus par le modèle Random Forest, indiquant que l'ensemble n'a pas significativement surpassé ce dernier.



**Figure 4.29:** Courbe ROC pour *retail\_juin* avec le modèle d'ensemble.

**Interprétation :** La courbe ROC pour *retail\_juin* montre un AUC de 0.75, similaire au modèle Random Forest. Cela suggère que l'ensemble a capturé efficacement la complexité des données, mais il n'a pas surpassé les performances des modèles individuels.

#### 4.4.5 Vérification de surajustement du meilleur modèle

#### 4.4.6 Conclusion

Dans cette section, on a comparé les performances des modèles de machine learning et de deep learning sur les trois datasets. Les modèles de machine learning, notamment Random Forest, ont bien

performé sur les données moins complexes, tandis que les modèles de deep learning, tels que ceux basés sur Keras/TensorFlow, ont mieux généralisé sur des données plus complexes comme *retail\_juin*. L'approche d'ensemble combinant ces deux méthodes n'a apporté que des gains marginaux. En conclusion, le choix du modèle optimal dépend de la complexité des données et du besoin de généralisation.

## 4.5 Vérification du surajustement du meilleur modèle

Le surajustement (*overfitting*) est un phénomène où un modèle apprend trop spécifiquement les détails et le bruit des données d'entraînement, ce qui lui permet d'obtenir de bonnes performances sur ces données, mais il généralise mal sur des données non vues (jeu de test). Un modèle surajusté aura tendance à surapprendre les particularités des données d'entraînement, au détriment de sa capacité à bien performer sur des données nouvelles.

Pour évaluer si le meilleur modèle présente un phénomène de surajustement, nous avons comparé ses performances sur les jeux de validation et de test. Les résultats présentés ici concernent la data **network\_may**, et des analyses similaires ont été effectuées pour les autres datasets, où des résultats cohérents ont été obtenus. La figure 4.30 montre ces performances.

| Performance sur le jeu de validation :           |           |        |          |         |
|--------------------------------------------------|-----------|--------|----------|---------|
|                                                  | precision | recall | f1-score | support |
| 0                                                | 0.69      | 0.62   | 0.65     | 47      |
| 1                                                | 0.69      | 0.76   | 0.73     | 54      |
| accuracy                                         |           |        | 0.69     | 101     |
| macro avg                                        |           | 0.69   | 0.69     | 101     |
| weighted avg                                     |           | 0.69   | 0.69     | 101     |
| AUC sur le jeu de validation: 0.7807328605200946 |           |        |          |         |
| Performance sur le jeu de test :                 |           |        |          |         |
|                                                  | precision | recall | f1-score | support |
| 0                                                | 0.76      | 0.70   | 0.73     | 46      |
| 1                                                | 0.76      | 0.82   | 0.79     | 55      |
| accuracy                                         |           |        | 0.76     | 101     |
| macro avg                                        |           | 0.76   | 0.76     | 101     |
| weighted avg                                     |           | 0.76   | 0.76     | 101     |
| AUC sur le jeu de test: 0.8138339920948615       |           |        |          |         |

Figure 4.30: Performances du modèle sur les jeux de validation et de test pour **network\_may**

### 4.5.1 Analyse des résultats

Les métriques principales de performance, telles que la précision, le rappel, et le F1-score, ont été calculées sur les jeux de validation et de test. Une AUC (Area Under the ROC Curve) a également été calculée pour évaluer la capacité du modèle à distinguer les classes.

- **Performances sur le jeu de validation :** Le modèle présente une précision et un rappel de 0.69 et 0.76, respectivement, avec un score AUC de **0.78**. Ces résultats indiquent que le modèle a une capacité modérée à classifier correctement les exemples du jeu de validation.
- **Performances sur le jeu de test :** Les performances sur le jeu de test montrent une légère amélioration, avec une précision et un rappel de 0.76 et 0.82, respectivement, et un score AUC de **0.81**. Ces résultats sont similaires à ceux du jeu de validation, indiquant que le modèle généralise bien sur des données non vues.

### 4.5.2 Conclusion sur le surajustement

Les performances du modèle sur le jeu de validation et de test sont proches, comme le montre la comparaison des métriques. La différence entre l'AUC sur le jeu de validation (**0.78**) et l'AUC sur le jeu de test (**0.81**) est marginale, ce qui suggère que le modèle ne souffre pas de surajustement significatif.

En résumé, le modèle a montré une bonne capacité de généralisation sans surajustement notable, ce qui le rend approprié pour être utilisé sur de nouvelles données. La constance des résultats sur les deux jeux de données reflète la robustesse du modèle.

## 4.6 Profil des Clients Satisfaits et Interprétation des Variables

### 4.6.1 Résultats pour *network\_fev*

**Méthode des Intervalles :** Les résultats suivants montrent les intervalles de valeurs des variables associées au plus haut niveau de satisfaction des clients. Ces intervalles permettent d'identifier les segments de données qui maximisent la satisfaction client.

- L'intervalle le plus satisfaisant pour *activation\_days* correspond à une période allant du **5 août 2015 au 30 juillet 2018**, avec une probabilité de satisfaction de 0.59.
- L'intervalle le plus satisfaisant pour *new\_voice\_volume\_moyenne* : [26.40, 32.56], probabilité de satisfaction = 0.62.
- L'intervalle pour *new\_data\_amount\_moyenne* : [295.88, 346.13], probabilité de satisfaction = 0.59,etc.

La Figure 4.31 illustre le profil du client le plus satisfait sur la base de ces intervalles de satisfaction pour le dataset *network\_fev*.

## RÉSULTATS ET INTERPRÉTATION

---

```
Profil du client le plus satisfait :
L'intervalle le plus satisfaisant pour activation_days: [16652.83, 17742.60], Probabilité de satisfaction = 0.59
L'intervalle le plus satisfaisant pour new_voice_volume_moyenne: [26.40, 32.56], Probabilité de satisfaction = 0.62
L'intervalle le plus satisfaisant pour new_data_amount_moyenne: [295.88, 346.13], Probabilité de satisfaction = 0.59
L'intervalle le plus satisfaisant pour new_voice_amount_moyenne: [38.01, 42.45], Probabilité de satisfaction = 0.60
L'intervalle le plus satisfaisant pour new_mou_moyenne: [26.97, 32.83], Probabilité de satisfaction = 0.58
L'intervalle le plus satisfaisant pour new_recharge_moyenne: [774.55, 1433.84], Probabilité de satisfaction = 0.59
L'intervalle le plus satisfaisant pour new_nbre_sms_moyenne: [1.64, 3.76], Probabilité de satisfaction = 0.61
L'intervalle le plus satisfaisant pour new_nbre_arpu_moyenne: [144.85, 209.32], Probabilité de satisfaction = 0.59
L'intervalle le plus satisfaisant pour new_nbre_recharges_moyenne: [-0.76, 1.71], Probabilité de satisfaction = 0.57
L'intervalle le plus satisfaisant pour new_data_volume_moyenne: [1.30, 2.51], Probabilité de satisfaction = 0.58
La catégorie la plus satisfaisante pour offer_id est '2581.0' avec une probabilité de satisfaction = 0.91
La catégorie la plus satisfaisante pour monthly_rech_subscriber_seg est '5.399860971715255' avec une probabilité de satisfaction = 0.78
La catégorie la plus satisfaisante pour device_type est '1.0638650088808568' avec une probabilité de satisfaction = 0.84
```

**Figure 4.31: Profil du client le plus satisfait pour *network\_fev* basé sur les intervalles de satisfaction.**

**Méthode d'Interprétation des Effets :** Cette méthode évalue l'impact d'une augmentation de 10% des valeurs moyennes des variables sur la probabilité de satisfaction. Voici un résumé des principaux résultats :

- Pour *activation\_days*, en augmentant de 10% la valeur moyenne (du **22 avril 2015 au 20 septembre 2017**), la probabilité de satisfaction passe de 0.53 à 0.53, soit un changement de 0.50%.
- Pour *new\_voice\_volume\_moyenne*, en augmentant de 10% la valeur moyenne (de 36.76 à 49.34), la probabilité de satisfaction passe de 0.54 à 0.57, soit un changement de 3.56%.
- Pour *new\_data\_amount\_moyenne*, une augmentation de 10% fait passer la probabilité de 0.55 à 0.53, soit un changement de -2.22%, etc.

### 4.6.2 Résultats pour *network\_may*

**Méthode des Intervalles :** Les intervalles les plus satisfaisants pour chaque variable incluent :

- *Activation Days* : du 3 mars 2021 au 22 septembre 2022, probabilité = 0.69.
- *Voice Volume Moyenne* : [17726.43, 24097.83], probabilité = 0.69,etc.

La figure 4.32 présente les autres intervalles correspondant aux variables analysées.

```
Profil du client le plus satisfait :
L'intervalle le plus satisfaisant pour voice_volume_moyenne: [17726.43, 24097.83], Probabilité de satisfaction = 0.69
L'intervalle le plus satisfaisant pour voice_amount_moyenne: [2367.90, 3370.33], Probabilité de satisfaction = 0.63
L'intervalle le plus satisfaisant pour mou_moyenne: [12600.25, 17313.12], Probabilité de satisfaction = 0.66
L'intervalle le plus satisfaisant pour activation_days: [19059.18, 19626.59], Probabilité de satisfaction = 0.69
L'intervalle le plus satisfaisant pour recharge_moyenne: [9338.83, 14921.54], Probabilité de satisfaction = 0.63
L'intervalle le plus satisfaisant pour Arpu_moyenne: [15218.64, 23136.28], Probabilité de satisfaction = 0.70
L'intervalle le plus satisfaisant pour nbre_sms_moyenne: [9.23, 13.06], Probabilité de satisfaction = 0.62
L'intervalle le plus satisfaisant pour data_amount_moyenne: [13081.11, 15731.00], Probabilité de satisfaction = 0.65
L'intervalle le plus satisfaisant pour data_volume_moyenne: [4.72, 8.70], Probabilité de satisfaction = 0.65
La catégorie la plus satisfaisante pour offer_id est '24' avec une probabilité de satisfaction = 0.99
```

**Figure 4.32: Profil du client le plus satisfait pour *network\_may* basé sur les intervalles de satisfaction.**

**Méthode d'Interprétation des Effets :** Quelques exemples d'impact des variables sur la satisfaction :

- *Activation Days* : Augmentation de 10% (du 21 décembre 2016 au 30 août 2019), probabilité de satisfaction passe de 0.54 à 0.66 (+11.63%).
- *Voice Volume Moyenne* : Augmentation de 10% (3172.58 à 9656.98), probabilité de satisfaction passe de 0.58 à 0.61 (+2.87%), etc.

**Méthode des Intervalles :** Les intervalles les plus satisfaisants pour chaque variable incluent :

- *Activation Days* : du 13 décembre 2016 au 27 mars 2021, probabilité = 0.71.
- *Voice Volume Moyenne* : [6470.59, 9894.88], probabilité = 0.66, etc.

La figure 4.33 présente les autres intervalles correspondant aux variables analysées.

```
Profil du client le plus satisfait :
L'intervalle le plus satisfaisant pour activation_days: [18689.35, 19257.07], Probabilité de satisfaction = 0.71
L'intervalle le plus satisfaisant pour voice_volume_moyenne: [6470.59, 9894.88], Probabilité de satisfaction = 0.66
L'intervalle le plus satisfaisant pour voice_amount_moyenne: [23.31, 595.62], Probabilité de satisfaction = 0.65
L'intervalle le plus satisfaisant pour Arpu_moyenne: [25438.77, 28751.79], Probabilité de satisfaction = 0.69
L'intervalle le plus satisfaisant pour recharge_moyenne: [15017.03, 20396.98], Probabilité de satisfaction = 0.64
L'intervalle le plus satisfaisant pour mou_moyenne: [5955.96, 7831.90], Probabilité de satisfaction = 0.67
L'intervalle le plus satisfaisant pour data_amount_moyenne: [3595.70, 5113.60], Probabilité de satisfaction = 0.67
L'intervalle le plus satisfaisant pour nbre_sms_moyenne: [-0.00, 1.00], Probabilité de satisfaction = 0.71
La catégorie la plus satisfaisante pour monthly_arpu_subscriber_seg est '18' avec une probabilité de satisfaction = 0.89
La catégorie la plus satisfaisante pour monthly_rech_subscriber_seg est '0' avec une probabilité de satisfaction = 0.83
La catégorie la plus satisfaisante pour offer_id est '1227' avec une probabilité de satisfaction = 0.96
```

**Figure 4.33: Profil du client le plus satisfait pour *retail\_juin* basé sur les intervalles de satisfaction.**

**Méthode d'Interprétation des Effets :** Quelques exemples d'impact des variables sur la satisfaction :

- *Activation Days* : Augmentation de 10% (du 18 juin 2017 au 12 octobre 2020), probabilité de satisfaction passe de 0.55 à 0.56 (+1.33%).
- *Voice Volume Moyenne* : Augmentation de 10% (1029.51 à 6337.51), probabilité de satisfaction passe de 0.61 à 0.51 (-9.36%), etc.

### 4.6.3 Comparaison des Résultats Entre les Trois Datasets

Les trois datasets présentent des profils de satisfaction légèrement différents selon les variables analysées.

- **Activation Days** : Le dataset *network\_may* montre l'intervalle le plus récent pour *activation\_days*, tandis que *retail\_juin* couvre une période légèrement plus ancienne. Cependant, les probabilités de satisfaction pour cette variable restent similaires entre les datasets, avec une légère supériorité pour *network\_may*.
- **Volume de Voix et de Données** : Pour *network\_fev*, les volumes de voix et de données ont un impact notable sur la satisfaction, avec des probabilités variant entre 0.60 et 0.62. En revanche, ces mêmes variables ont un impact légèrement plus marqué sur *network\_may*, avec des probabilités atteignant 0.66 après variation des valeurs.

- **Interprétation des Effets des Variables :** Les résultats d’interprétation montrent que l’augmentation des variables quantitatives, comme *activation\_days* et *new\_Arpu\_moyenne*, entraîne un effet de satisfaction plus fort sur *network\_may*, où des variations de 10% de ces variables génèrent un changement plus significatif dans la probabilité de satisfaction. À l’inverse, pour *retail\_juin*, les augmentations de 10% montrent des changements de satisfaction plus modérés.

**Conclusion de la Comparaison :** Les trois datasets révèlent des profils de satisfaction similaires pour certaines variables comme *activation\_days*, mais des différences notables apparaissent pour les volumes de voix et de données, notamment sur *network\_may*. De plus, les effets des augmentations de 10% dans les variables quantitatives se traduisent par des changements plus importants dans les probabilités de satisfaction pour *network\_may* que pour *retail\_juin* et *network\_fev*. Cela souligne l’importance de chaque dataset dans la détermination des facteurs clés de satisfaction.

## 4.7 Conclusion

Ce chapitre présente les résultats obtenus à l’aide des modèles de machine learning et de deep learning appliqués aux trois datasets : *network\_fev*, *network\_may*, et *retail\_juin*. Les modèles utilisés incluent la régression logistique, l’arbre de décision, et le Random Forest pour la partie machine learning, ainsi que PyTorch et Keras/TensorFlow pour la partie deep learning. De plus, une approche d’ensemble combinant le Random Forest, AdaBoost et un réseau de neurones a été explorée afin d’améliorer la performance globale de classification. Chaque méthode a été analysée et validée à travers plusieurs métriques de performance, incluant la précision, le F1-score, et l’aire sous la courbe ROC (AUC). Enfin, on a proposé une analyse approfondie du profil des clients les plus satisfaits en nous basant sur les résultats des modèles et en examinant les effets des variables.

Le chapitre est structuré de la manière suivante : tout d’abord, les résultats des modèles de machine learning sont présentés, suivis des résultats des modèles de deep learning. Ensuite, une comparaison globale des deux approches est effectuée avant de présenter les résultats du modèle d’ensemble. Finalement, nous analysons le profil des clients satisfaits en nous basant sur les variables clés identifiées à partir des modèles.



---

## Conclusion et perspective

Ce rapport met en lumière l'importance cruciale de l'analyse des données dans le secteur des télécommunications, en particulier pour l'entreprise Ooredoo. Grâce à une méthodologie rigoureuse et à l'application de techniques avancées de data science, nous avons exploré les comportements des clients et identifié les facteurs clés influençant leur satisfaction.

Les résultats obtenus révèlent des corrélations significatives entre différentes variables, comme la consommation de minutes et l'accès aux données, ainsi que des tendances marquantes concernant l'ancienneté des clients et leur fréquence de recharge. Ces insights fournissent une base solide pour le développement de modèles prédictifs, permettant à Ooredoo d'anticiper les besoins des clients et d'adapter ses stratégies marketing de manière plus efficace.

Néanmoins, plusieurs problématiques ont émergé au cours de l'étude, ouvrant des perspectives d'amélioration pour les analyses futures. D'abord, la fiabilité des données collectées soulève des limites. L'envoi de messages comportant des questions spécifiques n'est pas toujours compris par l'ensemble des destinataires, dont les profils varient considérablement. Ces différences de compréhension peuvent altérer la qualité des réponses et, par conséquent, influencer la qualité des données. Il serait donc judicieux d'envisager une méthode de collecte plus standardisée et inclusive, par exemple en transformant les questions en un système de score de satisfaction plus accessible à tous.

Par ailleurs, certaines variables démographiques, telles que l'âge ou le genre, peuvent manquer de fiabilité, compromettant ainsi la précision des analyses basées sur ces critères. Cette incertitude peut introduire des biais dans les modèles prédictifs. Par conséquent, il est essentiel de renforcer la qualité des données démographiques afin d'assurer une segmentation plus précise et d'obtenir des résultats analytiques plus fiables.

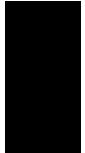
En conclusion, cette étude ouvre la voie à de futures recherches qui devront perfectionner les méthodes de collecte des données et réduire les biais dans les modèles analytiques. Cela permettra aux entreprises de rester à la fois réactives et proactives face aux évolutions des comportements des consommateurs.



---

# Webographie

- [1] Talend - Data Mining, retrieved September 1st, 2024  
URL: <https://www.talend.com/fr/resources/what-is-data-mining/>
- [2] XLSTAT - Test Paramétrique, retrieved September 5th, 2024  
URL: <https://help.xlstat.com/fr/6739-quelle-est-la-difference-entre-un-test-parametrique-et>
- [3] Alliage AD - Test de Shapiro-Wilk, retrieved September 8th, 2024  
URL: <https://www.alliage-ad.com/data-science/data-science-test-de-shapiro-wilk/>
- [4] ULg - Corrélation de Pearson, retrieved September 10th, 2024  
URL: [http://www.biostat.ulg.ac.be/pages/Site\\_r/corr\\_pearson.html](http://www.biostat.ulg.ac.be/pages/Site_r/corr_pearson.html)
- [5] MCO - Mathématiques appliquées, retrieved September 15th, 2024  
URL: <https://math.univ-cotedazur.fr/~diener/MAB07/MCO.pdf>
- [6] Datatab - Test de Mann-Whitney, retrieved September 19th, 2024  
URL: [https://datatab.fr/tutorial/mann-whitney-u-test?fbclid=IwY2xjawF5a5tleHRuA2FlbQIxMAABHToYLMrDTErjBJqXbuTy2fCaXxZBYP4tHD7HbD6a5Lbh67MGj06KnlyRw\\_aem\\_QJPFi\\_PIZZW4DQaxNYoBDg](https://datatab.fr/tutorial/mann-whitney-u-test?fbclid=IwY2xjawF5a5tleHRuA2FlbQIxMAABHToYLMrDTErjBJqXbuTy2fCaXxZBYP4tHD7HbD6a5Lbh67MGj06KnlyRw_aem_QJPFi_PIZZW4DQaxNYoBDg)
- [7] ULg - Corrélation de Spearman, retrieved September 25th, 2024  
URL: [http://www.biostat.ulg.ac.be/pages/Site\\_r/corr\\_spearman.html](http://www.biostat.ulg.ac.be/pages/Site_r/corr_spearman.html)
- [8] Les-Mathématiques - Test Exact de Fisher, retrieved October 3rd, 2024  
URL: <https://les-mathematiques.net/vanilla/discussion/1893958/calcul-test-exact-de-fisher-a-1-a-main>
- [9] Inside Machine Learning - Régularisation en Deep Learning, retrieved October 13th, 2024  
URL: <https://inside-machinelearning.com/regularization-deep-learning/>



---

# Biographie

- [1] **George Box et David Cox**, statisticiens britanniques du 20e siècle, sont connus respectivement pour la transformation Box-Cox et le modèle de régression de Cox, contribuant à la stabilisation de la variance et à l'analyse de survie. Leurs travaux ont profondément influencé la statistique appliquée.
- [2] **Iain M. Yeo et Richard A. Johnson**, statisticiens des 20e et 21e siècles, sont reconnus pour avoir co-développé la transformation Yeo-Johnson, une méthode de normalisation des données, y compris celles avec des valeurs négatives, visant à stabiliser la variance en statistique appliquée.
- [3] **Hermann Minkowski** (1864-1909), mathématicien de la fin du 19e et début du 20e siècles, a marqué la géométrie et la théorie de la relativité d'Einstein avec l'espace-temps de Minkowski. Il est également connu pour avoir introduit la distance de Minkowski, une généralisation des mesures de distance.
- [4] **Euclide** (300 av), mathématicien de l'Antiquité grecque, est considéré comme le "père de la géométrie" grâce à son ouvrage Les Éléments. Il est également à l'origine de la distance euclidienne, une mesure classique entre deux points dans l'espace.
- [5] **Léo Goodman et William Kruskal**, tous deux statisticiens américains du 20e siècle, ont marqué l'analyse des données catégorielles. Goodman est surtout reconnu pour son travail sur les tables de contingence et le coefficient de Goodman et Kruskal, tandis que Kruskal a co-développé le test non paramétrique de Kruskal-Wallis, utilisé pour comparer plusieurs groupes.
- [6] **Samuel Shapiro et Martin Wilk**, statisticiens américains, ont conçu le test de Shapiro-Wilk pour vérifier la normalité des données, couramment utilisé avant les tests paramétriques nécessitant une distribution normale.
- [7] **Howard Levene**, mathématicien et statisticien américain du 20e siècle, est célèbre pour avoir développé le test de Levene. Ce test, utilisé pour vérifier l'égalité des variances entre plusieurs groupes, est essentiel en amont d'analyses statistiques comme l'ANOVA.
- [8] **William Sealy Gosset**, statisticien britannique des 19e et 20e siècles, est connu sous le pseudonyme "Student" pour avoir développé le test t de Student. Travaillant à la brasserie Guinness, il a conçu ce test pour comparer les moyennes de deux groupes à partir de petits échantillons, dans le cadre de l'amélioration du contrôle qualité.
- [9] **Karl Pearson**, pionnier britannique des statistiques modernes des 19e et 20e siècles, a fondé le premier département de statistique. Il a créé le coefficient de corrélation de Pearson, une mesure essentielle de la relation linéaire entre deux variables continues.

- [10] **Henry Mann**, mathématicien et statisticien américain du 20e siècle, est connu pour ses contributions à la théorie des nombres et aux statistiques. Avec Donald Whitney, il a co-développé le test de Mann-Whitney, une alternative non paramétrique au test t de Student pour comparer deux groupes indépendants.
- [11] **Charles Spearman**, psychologue et statisticien britannique, est connu pour avoir développé le coefficient de corrélation de Spearman. Il est également l'auteur de la théorie bifactorielle de l'intelligence, qui propose l'existence d'un facteur général de l'intelligence (g).
- [12] **Ronald Fisher**, biologiste, généticien et statisticien britannique, est considéré comme l'un des fondateurs des statistiques modernes. Il a développé des techniques comme l'ANOVA et le test exact de Fisher, tout en contribuant à la génétique des populations.

# Annexe A

```
323 ✓ data Resp_Network_fev;
324 set Resp_Network_fev;
325 array responses q3 q9 q10 q11 q8 q13 q15 q17 q18;
326 count = 0;
327 total = 0;
328
329 do i = 1 to dim(responses);
330 if not missing(responses[i]) then do;
331 count = count + 1;
332 total = total + input(responses[i], best32.);
333 end;
334 end;
335 if count > 0 then Osat_number = round(total / count);
336 else Osat_number = .;
337
338 drop i count total;
339
run;
```

Figure 34: Imputation de la colonne OSAT à partir de la moyenne des réponses aux questions

```
[60]
missing_values_activation_days = new_data['activation_days'].isna().sum()
print(f"Nombre de valeurs manquantes dans la colonne 'activation_days': {missing_values_activation_days}")
...
... Nombre de valeurs manquantes dans la colonne 'activation_days': 114

[61]
from sklearn.impute import KNNImputer
Créer une instance de KNNImputer
imputer = KNNImputer(n_neighbors=5, weights="uniform")

Appliquer KNNImputer aux données pour imputer les valeurs manquantes
data_imputed_array = imputer.fit_transform(new_data)

Convertir le résultat en DataFrame avec les mêmes colonnes que l'original
data_imputed = pd.DataFrame(data_imputed_array, columns=new_data.columns)

Sauvegarder la nouvelle table avec les valeurs imputées
data_imputed.to_csv('data_imputed_knn.csv', index=False)

print("Les données imputées ont été sauvegardées dans 'data_imputed_knn.csv'.")
...
... Les données imputées ont été sauvegardées dans 'data_imputed_knn.csv'.

▷
missing_values_activation_days = data_imputed['activation_days'].isna().sum()
print(f"Nombre de valeurs manquantes dans la colonne 'activation_days': {missing_values_activation_days}")
[62]
...
... Nombre de valeurs manquantes dans la colonne 'activation_days': 0
```

Figure 35: Code pour l'imputation KNN

```
139 /* Détermination des valeurs aberrantes */
140 data AbValues_recharge_amount_m3;
141 set DATA.ABT_NETWORK_FEB_FINAL;
142 if _N_ = 1 then set Summary_recharge_amount_m3;
143 if recharge_amount_m3 > (Q75_recharge_amount_m3 + 1.5 * IQR_recharge_amount_m3) or
144 | recharge_amount_m3 < (Q25_recharge_amount_m3 - 1.5 * IQR_recharge_amount_m3)
145 then output;
146 run;
147
148 /* Remplacement des valeurs aberrantes par la médiane */
149 data DATA.ABT_NETWORK_FEB_FINAL;
150 set DATA.ABT_NETWORK_FEB_FINAL;
151 if _N_ = 1 then set Summary_recharge_amount_m3;
152 if recharge_amount_m3 > (Q75_recharge_amount_m3 + 1.5 * IQR_recharge_amount_m3) or
153 | recharge_amount_m3 < (Q25_recharge_amount_m3 - 1.5 * IQR_recharge_amount_m3) then
154 recharge_amount_m3 = Median_recharge_amount_m3;
155 else
156 recharge_amount_m3 = recharge_amount_m3;
157 run;
```

**Figure 36:** Remplacement des valeurs aberrantes par la médiane.

```
210 /* Effectuer la transformation Box-Cox */
211 proc transreg data=DATA.ABT_NETWORK_FEB_FINAL;
212 | model boxcox(recharge_moyenne) = identity(OSAT);
213 run;
214
215 /*create new dataset that uses box-cox transformation to create new y*/
216 data DATA.ABT_NETWORK_FEB_FINAL;
217 set DATA.ABT_NETWORK_FEB_FINAL;
218 new_recharge_moyenne = (recharge_moyenne **(0.75) - 1) / 0.75;
219 run;
```

**Figure 37:** Transformation Box-Cox sur Recharge\_moyenne.

```
import pandas as pd
import numpy as np
from scipy import stats

def calculate_cramers_v(data, col1, col2):
 crosstab = pd.crosstab(data[col1], data[col2])
 chi2, p, dof, expected = stats.chi2_contingency(crosstab)
 n = crosstab.sum().sum()
 phi2 = chi2 / n
 r, k = crosstab.shape
 phi2corr = max(0, phi2 - ((k - 1) * (r - 1)) / (n - 1))
 r2 = phi2corr / min((k - 1), (r - 1))
 return np.sqrt(r2)
```

**Figure 38:** Calcul de l'indice de Cramér's V

```
def test_categorical_binary_association(data, categorical_col, binary_col):
 # 1. Calcul de l'indice de Cramér's V
 cramers_v = calculate_cramers_v(data, categorical_col, binary_col)
 print(f"Résultat de l'indice de Cramér's V entre {categorical_col} et {binary_col} : {cramers_v:.4f}")
 print(f"Interprétation : {'Association faible' if cramers_v < 0.1 else 'Association forte'}.")

 # 2. Test du Chi-Carré
 contingency_table = pd.crosstab(data[categorical_col], data[binary_col])
 print("\nTableau de contingence pour {} vs {}".format(categorical_col, binary_col))
 print(contingency_table)

 chi2_stat, chi2_p, dof, expected = stats.chi2_contingency(contingency_table)
 print("\nRésultats du test du chi carré pour {} vs {}".format(binary_col, categorical_col))
 print(f" Statistique = {chi2_stat:.4f}")
 print(f" p-value = {chi2_p:.4f}")

 if chi2_p < 0.05:
 print(" Association significative entre les variables.")
 else:
 print(" Aucune association significative entre les variables.")

 # Test de Fisher si chi-carré non vérifié et table de contingence 2x2
 if contingency_table.shape == (2, 2):
 fisher_stat, fisher_p = stats.fisher_exact(contingency_table)
 print("\nRésultats du test de Fisher pour {} vs {}".format(binary_col, categorical_col))
 print(f" Statistique = {fisher_stat:.4f}")
 print(f" p-value = {fisher_p:.4f}")
 if fisher_p < 0.05:
 print(" Association significative entre les variables.")
 else:
 print(" Aucune association significative entre les variables.")
 else:
 print(" Test de Fisher non applicable, table de contingence non 2x2.")
```

**Figure 39: Test du Chi-Carré et test de Fisher (si applicable)**

```

def lambda_goodman_kruskal(y_true, y_pred):
 # Créer la matrice de confusion en utilisant pd.crosstab
 confusion = pd.crosstab(y_true, y_pred)
 print("Matrice de confusion :\n", confusion)

 # Totaux des lignes et des colonnes
 row_totals = confusion.sum(axis=1)
 column_totals = confusion.sum(axis=0)
 total = confusion.values.sum()

 # Valeurs maximales des lignes et des colonnes
 max_rows = row_totals.max()
 max_columns = column_totals.max()

 # Calcul de l'erreur totale sans connaissance de la variable indépendante
 error_without_y = total - max_rows

 # Calcul de l'erreur avec connaissance de la variable indépendante
 error_with_y = total - max_columns

 # Calcul de la lambda
 lambda_y_given_x = (error_without_y - error_with_y) / error_without_y

 # Affichage des détails du calcul
 print("\nDétails du calcul :")
 print("Totaux des lignes :", row_totals.values)
 print("Totaux des colonnes :", column_totals.values)
 print("Total général :", total)
 print("Valeur maximale des lignes :", max_rows)
 print("Valeur maximale des colonnes :", max_columns)
 print("Erreur totale sans connaissance (erreur_without_y) :", error_without_y)
 print("Erreur avec connaissance (erreur_with_y) :", error_with_y)

 return lambda_y_given_x

```

**Figure 40: Calcul de la Lambda de Goodman et Kruskal**

```

def perform_analysis(data, numeric_col, binary_col, categorical_col):
 # Calcul de l'Eta carré (η^2)
 def calculate_eta_squared(data, numeric_col, categorical_col):
 overall_mean = data[numeric_col].mean()
 ss_total = sum((data[numeric_col] - overall_mean) ** 2)
 mean_by_group = data.groupby(categorical_col)[numeric_col].mean()
 ss_between = sum(data.groupby(categorical_col).size() * (mean_by_group - overall_mean) ** 2)
 eta_squared = ss_between / ss_total
 return eta_squared

 eta_squared_osat = calculate_eta_squared(data, numeric_col, categorical_col)
 eta_squared_osat_binary = calculate_eta_squared(data, numeric_col, binary_col)

```

**Figure 41: Calcul de l'Eta carré pour osat\_binary**

```

if all(result['p-value'] >= 0.05 for result in shapiro_results.values()) and levene_p >= 0.05:
 t_stat, t_p = stats.ttest_ind(*subsets, equal_var=True)
 print("\nRésultats du test t:")
 print(f" Statistique = {t_stat:.4f}")
 print(f" p-value = {t_p:.4f}")
 print(f" Il y a une différence significative entre les groupes (p < 0.05). if t_p < 0.05 else 'Aucune différence significative entre les groupes' ")
else:
 print("\nLes conditions pour appliquer le test t ne sont pas vérifiées.")
group1 = data[data[binary_col] == groups[0]][numeric_col]
group2 = data[data[binary_col] == groups[1]][numeric_col]
u_stat, p_value = mannwhitneyu(group1, group2, alternative='two-sided')
print("\nRésultat du test de Mann-Whitney U pour {binary_col}:")
print(f" Statistique U = {u_stat:.4f}")
print(f" p-value = {p_value:.4f}")
print(f" 'les distributions sont significativement différentes (p < 0.05).' if p_value < 0.05 else 'Les distributions ne sont pas significativement différentes' ")

```

**Figure 42: Test d'homogénéité des variances de Levene**

```

def perform_analysis(data, var1, var2):
 # Calcul de la covariance et son interprétation
 covariance, interpretation = calculate_covariance(data, var1, var2)
 print(f"Covariance entre {var1} et {var2} : {covariance:.4f}")
 print("Interprétation de la covariance :")
 print(interpretation)

 # Test de normalité de Shapiro-Wilk pour chaque variable
 shapiro_var1 = stats.shapiro(data[var1].dropna())
 shapiro_var2 = stats.shapiro(data[var2].dropna())

 print("\nRésultats du test de normalité de Shapiro-Wilk:")
 print(f"Variable {var1}: Statistique = {shapiro_var1.statistic:.4f}, p-value = {shapiro_var1.pvalue:.4f}")
 print(f"Les données ne suivent pas une distribution normale (p < 0.05). if shapiro_var1.pvalue < 0.05 else 'Les données suivent une distribution normale (p >= 0.05).'")
 print(f"Variable {var2}: Statistique = {shapiro_var2.statistic:.4f}, p-value = {shapiro_var2.pvalue:.4f}")
 print(f"Les données ne suivent pas une distribution normale (p < 0.05). if shapiro_var2.pvalue < 0.05 else 'Les données suivent une distribution normale (p >= 0.05).'")

```

**Figure 43: Calcul de la covariance et test de normalité (Shapiro-Wilk) pour deux variables**

```

Choix du test basé sur la normalité
if shapiro_var1.pvalue >= 0.05 and shapiro_var2.pvalue >= 0.05:
 # Test paramétrique: corrélation de Pearson
 pearson_stat, pearson_p = stats.pearsonr(data[var1].dropna(), data[var2].dropna())
 print("\nRésultats du test de corrélation de Pearson:")
 print(f"Coefficient de corrélation = {pearson_stat:.4f}")
 print(f"p-value = {pearson_p:.4f}")
 print(f"Il y a une corrélation significative entre les variables (p < 0.05). if pearson_p < 0.05 else 'Aucune corrélation significative entre les variables (p >= 0.05).'")
 print(f"Interprétation de la corrélation : {interpret_correlation(pearson_stat)}")

else:
 # Test non paramétrique: corrélation de Spearman
 spearman_stat, spearman_p = stats.spearmanr(data[var1].dropna(), data[var2].dropna())
 print("\nRésultats du test de corrélation de Spearman:")
 print(f"Coefficient de corrélation = {spearman_stat:.4f}")
 print(f"p-value = {spearman_p:.4f}")
 print(f"Il y a une corrélation significative entre les variables (p < 0.05). if spearman_p < 0.05 else 'Aucune corrélation significative entre les variables (p >= 0.05).'")
 print(f"Interprétation de la corrélation : {interpret_correlation(spearman_stat)}")

```

**Figure 44: Choix du test de corrélation: Pearson ou Spearman, basé sur la normalité des données**

```

Séparer les features et la cible
X = df_resampled_imputed.drop('OSAT', axis=1)
y = df_resampled_imputed['OSAT']

Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Méthode 1 : RFE
model_rfe = RandomForestClassifier(random_state=42)
rfe = RFE(estimator=model_rfe, n_features_to_select=10, step=1)
rfe.fit(X_train, y_train)
rfe_ranking = rfe.ranking_

Méthode 2 : SelectKBest
kbest = SelectKBest(score_func=f_classif, k='all')
kbest.fit(X_train, y_train)
kbest_scores = kbest.scores_

Méthode 3 : Feature Importances from Random Forest
model_importance = RandomForestClassifier(random_state=42)
model_importance.fit(X_train, y_train)
feature_importances = model_importance.feature_importances_

```

**Figure 45:** Code de sélection des caractéristiques avec RFE, SelectKBest, et Random Forest

```

Définir les hyperparamètres à optimiser
param_dist = {
 'n_estimators': [50, 100, 150, 200],
 'max_depth': [None, 10, 20, 30, 40],
 'min_samples_split': [2, 5, 10, 15],
 'min_samples_leaf': [1, 2, 4, 6],
 'max_features': ['auto', 'sqrt', 'log2', None],
 'bootstrap': [True, False],
 'criterion': ['gini', 'entropy'],
 'class_weight': [None, 'balanced'],
 'min_impurity_decrease': [0.0, 0.1, 0.2],
 'oob_score': [True, False]
}

Configurer la recherche aléatoire avec validation croisée
random_search = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
 param_distributions=param_dist,
 n_iter=100,
 cv=3,
 scoring='accuracy',
 n_jobs=-1,
 verbose=2,
 random_state=42)

Entrainer la recherche aléatoire avec les données standardisées
random_search.fit(X_train_scaled, y_train)

```

**Figure 46:** Optimisation des hyperparamètres pour Random Forest

```
Utiliser le meilleur modèle trouvé pour extraire les importances des caractéristiques
best_rf_model = random_search.best_estimator_
importances = best_rf_model.feature_importances_

Créer un DataFrame pour les importances des caractéristiques
feature_importances = pd.DataFrame({'Feature': X.columns, 'Importance': importances})
feature_importances = feature_importances.sort_values(by='Importance', ascending=False)

Afficher les importances des caractéristiques
print("Importances des caractéristiques :")
print(feature_importances)

Définir un seuil pour l'importance des caractéristiques
seuil_importance = 0.035

Sélectionner les caractéristiques importantes
features_importantes = feature_importances[feature_importances['Importance'] > seuil_importance]['Feature']

Afficher les caractéristiques sélectionnées
print("\nCaractéristiques sélectionnées :")
print(features_importantes.tolist())
```

**Figure 47: Importances des caractéristiques par Random Forest**

```
Courbe ROC et AUC pour les caractéristiques importantes
fpr, tpr, thresholds = roc_curve(y_test, y_pred_rf_proba_imp)
roc_auc = auc(fpr, tpr)

Trouver le meilleur seuil basé sur le ROC
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]

Représenter en utilisant le seuil optimal
y_pred_optimal = (y_pred_rf_proba_imp >= optimal_threshold).astype(int)
```

**Figure 48: Courbe ROC pour Random Forest avec AUC.**

```
Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

Standardizing the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

Converting data to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).unsqueeze(1) # Add dimension for compatibility
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).unsqueeze(1)

Calculate class weights
class_weights = compute_class_weight('balanced', classes=np.unique(y), y=y)
class_weights = torch.tensor(class_weights, dtype=torch.float32)
```

**Figure 49: Préparation des données pour PyTorch.**

```
Define the neural network model
class BinaryClassificationModel(nn.Module):
 def __init__(self, input_dim, dropout_prob=0.3):
 super(BinaryClassificationModel, self).__init__()
 self.layer1 = nn.Linear(input_dim, 128)
 self.layer2 = nn.Linear(128, 64)
 self.layer3 = nn.Linear(64, 32)
 self.output = nn.Linear(32, 1)
 self.relu = nn.ReLU()
 self.dropout = nn.Dropout(dropout_prob) # Dropout for regularization

 def forward(self, x):
 x = self.relu(self.layer1(x))
 x = self.dropout(self.relu(self.layer2(x)))
 x = self.relu(self.layer3(x))
 x = torch.sigmoid(self.output(x))
 return x
```

**Figure 50:** Architecture du modèle de classification binaire dans PyTorch.

```
Define ranges for hyperparameter tuning
lr_range = [0.001, 0.01, 0.1]
batch_size_range = [16, 32, 64]
dropout_range = [0.2, 0.3, 0.4]
num_epochs = 100 # Increased number of epochs

for lr in lr_range:
 for batch_size in batch_size_range:
 for dropout_prob in dropout_range:
 print(f"Training with lr={lr}, batch_size={batch_size}, dropout_prob={dropout_prob}")

 # Update model with new dropout probability
 model = BinaryClassificationModel(input_dim, dropout_prob)

 criterion = nn.BCELoss()
 optimizer = optim.Adam(model.parameters(), lr=lr)

 # Training the model
 for epoch in range(num_epochs):
 model.train()
 optimizer.zero_grad()

 # Forward pass
 outputs = model(x_train_tensor).squeeze()
 loss = criterion(outputs, y_train_tensor.squeeze())

 # Backward pass and optimization
 loss.backward()
 optimizer.step()
```

**Figure 51:** Réglage des hyperparamètres pour le modèle PyTorch.

```
Final evaluation with the best model
best_model.eval()
with torch.no_grad():
 y_pred = best_model(X_test_tensor).squeeze()
 y_pred_class = (y_pred >= 0.5).float()

Convert tensor to numpy array for sklearn metrics
y_pred_class = y_pred_class.numpy()
y_test_numpy = y_test_tensor.squeeze().numpy()

Print classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test_numpy, y_pred_class))
print("Confusion Matrix:\n", confusion_matrix(y_test_numpy, y_pred_class))
```

**Figure 52:** Sélection du meilleur modèle PyTorch et évaluation finale.

```
Définir le modèle
model = Sequential()
model.add(Dense(124, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

Compiler le modèle
model.compile(optimizer=Adam(learning_rate=0.001),
 loss='binary_crossentropy',
 metrics=['accuracy'])

Entrainer le modèle
history = model.fit(X_train, y_train,
 epochs=200,
 batch_size=16,
 validation_split=0.2,
 verbose=1)
```

**Figure 53:** Architecture classique du modèle Keras.

```
Définir les hyperparamètres à explorer
learning_rates = [1e-5, 1e-4, 1e-3, 1e-2]
dropout_rates = [0.3, 0.4, 0.5]
units1_options = [32, 64, 128]
units2_options = [16, 32, 64]
batch_sizes = [16, 32, 64]
epochs = 20

best_accuracy = 0
best_params = {}

Boucle sur toutes les combinaisons d'hyperparamètres
for learning_rate in learning_rates:
 for dropout_rate in dropout_rates:
 for units1 in units1_options:
 for units2 in units2_options:
 for batch_size in batch_sizes:
 accuracy, model = train_model(X_train, y_train, X_test, y_test, learning_rate, dropout_rate, units1, units2, batch_size, epochs)
 print(f'LR: {learning_rate}, Dropout: {dropout_rate}, Units1: {units1}, Units2: {units2}, Batch Size: {batch_size} => Accuracy: {accuracy}')
 if accuracy > best_accuracy:
 best_accuracy = accuracy
 best_params = {
 'learning_rate': learning_rate,
 'dropout_rate': dropout_rate,
 'units1': units1,
 'units2': units2,
 'batch_size': batch_size
 }

print(f'Best Accuracy: {best_accuracy}')
print(f'Best Hyperparameters: {best_params}'')
```

**Figure 54: Optimisation des hyperparamètres du modèle Keras.**

```
Fonction pour créer le modèle Keras avec des hyperparamètres variables
def create_model(layers=[64], activations=['relu'], optimizer='adam'):
 model = Sequential()
 model.add(Dense(layers[0], input_dim=X_train.shape[1], activation=activations[0]))
 for layer, activation in zip(layers[1:], activations[1:]):
 model.add(Dense(layer, activation=activation))
 model.add(Dense(1, activation='sigmoid')) # Sortie binaire
 model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
 return model

Listes des hyperparamètres à tester
layer_configs = [
 [64],
 [64, 32],
 [128, 64, 32]
]

Diverses fonctions d'activation pour chaque configuration de couches
activation_functions = [
 ['relu'], ['sigmoid'], ['tanh'],
 ['relu', 'relu'], ['sigmoid', 'tanh'], ['tanh', 'relu'],
 ['relu', 'relu', 'relu'], ['sigmoid', 'tanh', 'relu'], ['tanh', 'relu', 'sigmoid']
]

Différents optimiseurs à tester
optimizers = [Adam, SGD, RMSprop]
optimizer_params = [{ 'learning_rate': 0.001}, { 'learning_rate': 0.01}, { 'learning_rate': 0.1}]
```

**Figure 55: Optimisation de l'architecture des couches dans Keras.**

# Annexe B

```
Analyse entre chaque variable catégorielle et 'osat_binary':
Matrice de confusion :
Flag_4g_binary 0 1
osat_binary
0 155 702
1 340 1362

Détails du calcul :
Totaux des lignes : [857 1702]
Totaux des colonnes : [495 2064]
Total général : 2559
Valeur maximale des lignes : 1702
Valeur maximale des colonnes : 2064
Erreur totale sans connaissance (erreur_without_y) : 857
Erreur avec connaissance (erreur_with_y) : 495

Lambda de Goodman et Kruskal pour 'osat_binary' et Flag_4g_binary :
Lambda : 0.4224037339556593
Interprétation : La lambda de Goodman et Kruskal est 0.42, indiquant une réduction d'erreur de 0.42 fois en prédisant 'osat_binary' avec 'Flag_4g_binary'.
```

**Figure 56: Résultat de la Lambda de Goodman et Kruskal pour Flag\_4g\_binary.**

Eta carré ( $\eta^2$ ) : 0.0064

Interprétation : Eta carré indique un effet très faible.

(0.006398754036123539, 'Eta carré indique un effet très faible.')

**Figure 57: Résultat de l'Eta carré pour osat\_binary**

Résultats du test de normalité de Shapiro-Wilk:

Groupe 0:

Statistique = 0.7522

p-value = 0.0000

Les données du groupe 0 ne suivent pas une distribution normale ( $p < 0.05$ ).

Groupe 1:

Statistique = 0.6954

p-value = 0.0000

Les données du groupe 1 ne suivent pas une distribution normale ( $p < 0.05$ ).

Résultats du test d'homogénéité des variances de Levene:

Statistique = 4.8758

p-value = 0.0273

Les variances ne sont pas homogènes ( $p < 0.05$ ).

Les conditions pour appliquer le test t ne sont pas vérifiées.

**Figure 58: Résultats du test de normalité et de variances**

```
Statistique U du test de Mann-Whitney: 725409.5000
Valeur p du test de Mann-Whitney: 0.0003
Les distributions de recharge_moyenne sont significativement différentes entre les deux groupes de osat_binary.
Interprétation détaillée : Il existe des différences significatives dans la distribution de 'recharge_moyenne' entre les deux groupes de 'osat_binary'.
```

**Figure 59: Résultats du test de Mann-Whitney**

Covariance entre recharge\_moyenne et Arpu\_moyenne : 54803095.8767

Interprétation de la covariance :

La covariance est positive. Cela indique que les deux variables ont tendance à augmenter ensemble.

Résultats du test de normalité de Shapiro-Wilk:

Variable recharge\_moyenne: Statistique = 0.7128, p-value = 0.0000

Les données ne suivent pas une distribution normale ( $p < 0.05$ ).

Variable Arpu\_moyenne: Statistique = 0.6992, p-value = 0.0000

Les données ne suivent pas une distribution normale ( $p < 0.05$ ).

Résultats du test de corrélation de Spearman:

Coefficient de corrélation = 0.9450

p-value = 0.0000

Il y a une corrélation significative entre les variables ( $p < 0.05$ ).

Interprétation de la corrélation : La corrélation est positive et très forte.

**Figure 60: Interprétation des résultats statistiques entre les deux variables**

```
def test_categorical_binary_association(data, categorical_col, binary_col):
 # 1. Calcul de l'indice de Cramér's V
 cramers_v = calculate_cramers_v(data, categorical_col, binary_col)
 print(f"\nRésultat de l'indice de Cramér's V entre {categorical_col} et {binary_col} : {cramers_v:.4f}")
 print(f"Interprétation : {'Association faible' if cramers_v < 0.1 else 'Association forte'}.")

 # 2. Test du Chi-Carré
 contingency_table = pd.crosstab(data[categorical_col], data[binary_col])
 print("\nTableau de contingence pour {} vs {}".format(categorical_col, binary_col))
 print(contingency_table)

 chi2_stat, chi2_p, dof, expected = stats.chi2_contingency(contingency_table)
 print("\nRésultats du test du chi carré pour {} vs {}".format(binary_col, categorical_col))
 print(f" Statistique = {chi2_stat:.4f}")
 print(f" p-value = {chi2_p:.4f}")

 if chi2_p < 0.05:
 print(" Association significative entre les variables.")
 else:
 print(" Aucune association significative entre les variables.")

 # Test de Fisher si chi-carré non vérifié et table de contingence 2x2
 if contingency_table.shape == (2, 2):
 fisher_stat, fisher_p = stats.fisher_exact(contingency_table)
 print("\nRésultats du test de Fisher pour {} vs {}".format(binary_col, categorical_col))
 print(f" Statistique = {fisher_stat:.4f}")
 print(f" p-value = {fisher_p:.4f}")
 if fisher_p < 0.05:
 print(" Association significative entre les variables.")
 else:
 print(" Aucune association significative entre les variables.")
 else:
 print(" Test de Fisher non applicable, table de contingence non 2x2.")
```

**Figure 61: Test du Chi-Carré et test de Fisher (si applicable)**

```

def lambda_goodman_kruskal(y_true, y_pred):
 # Créer la matrice de confusion en utilisant pd.crosstab
 confusion = pd.crosstab(y_true, y_pred)
 print("Matrice de confusion :\n", confusion)

 # Totaux des lignes et des colonnes
 row_totals = confusion.sum(axis=1)
 column_totals = confusion.sum(axis=0)
 total = confusion.values.sum()

 # Valeurs maximales des lignes et des colonnes
 max_rows = row_totals.max()
 max_columns = column_totals.max()

 # Calcul de l'erreur totale sans connaissance de la variable indépendante
 error_without_y = total - max_rows

 # Calcul de l'erreur avec connaissance de la variable indépendante
 error_with_y = total - max_columns

 # Calcul de la lambda
 lambda_y_given_x = (error_without_y - error_with_y) / error_without_y

 # Affichage des détails du calcul
 print("\nDétails du calcul :")
 print("Totaux des lignes :", row_totals.values)
 print("Totaux des colonnes :", column_totals.values)
 print("Total général :", total)
 print("Valeur maximale des lignes :", max_rows)
 print("Valeur maximale des colonnes :", max_columns)
 print("Erreur totale sans connaissance (erreur_without_y) :", error_without_y)
 print("Erreur avec connaissance (erreur_with_y) :", error_with_y)

 return lambda_y_given_x

```

**Figure 62: Calcul de la Lambda de Goodman et Kruskal**

```

def perform_analysis(data, numeric_col, binary_col, categorical_col):
 # Calcul de l'Eta carré (η^2)
 def calculate_eta_squared(data, numeric_col, categorical_col):
 overall_mean = data[numeric_col].mean()
 ss_total = sum((data[numeric_col] - overall_mean) ** 2)
 mean_by_group = data.groupby(categorical_col)[numeric_col].mean()
 ss_between = sum(data.groupby(categorical_col).size() * (mean_by_group - overall_mean) ** 2)
 eta_squared = ss_between / ss_total
 return eta_squared

 eta_squared_osat = calculate_eta_squared(data, numeric_col, categorical_col)
 eta_squared_osat_binary = calculate_eta_squared(data, numeric_col, binary_col)

```

**Figure 63: Calcul de l'Eta carré pour osat\_binary**

```

if all(result['p-value'] >= 0.05 for result in shapiro_results.values()) and levene_p >= 0.05:
 t_stat, t_p = stats.ttest_ind(*subsets, equal_var=True)
 print("\nRésultats du test t:")
 print(f" Statistique = {t_stat:.4f}")
 print(f" p-value = {t_p:.4f}")
 print(f" Il y a une différence significative entre les groupes (p < 0.05). if t_p < 0.05 else 'Aucune différence significative entre les groupes' ")
else:
 print("\nLes conditions pour appliquer le test t ne sont pas vérifiées.")
 group1 = data[data[binary_col] == groups[0]][numeric_col]
 group2 = data[data[binary_col] == groups[1]][numeric_col]
 u_stat, p_value = mannwhitneyu(group1, group2, alternative='two-sided')
 print("\nRésultat du test de Mann-Whitney U pour {binary_col}:")
 print(f" Statistique U = {u_stat:.4f}")
 print(f" p-value = {p_value:.4f}")
 print(f" 'les distributions sont significativement différentes (p < 0.05).' if p_value < 0.05 else 'Les distributions ne sont pas significativement différentes' ")

```

**Figure 64: Test d'homogénéité des variances de Levene**

```

def perform_analysis(data, var1, var2):
 # Calcul de la covariance et son interprétation
 covariance, interpretation = calculate_covariance(data, var1, var2)
 print(f"Covariance entre {var1} et {var2} : {covariance:.4f}")
 print("Interprétation de la covariance :")
 print(interpretation)

 # Test de normalité de Shapiro-Wilk pour chaque variable
 shapiro_var1 = stats.shapiro(data[var1].dropna())
 shapiro_var2 = stats.shapiro(data[var2].dropna())

 print("\nRésultats du test de normalité de Shapiro-Wilk:")
 print(f"Variable {var1}: Statistique = {shapiro_var1.statistic:.4f}, p-value = {shapiro_var1.pvalue:.4f}")
 print(f"Les données ne suivent pas une distribution normale (p < 0.05). if shapiro_var1.pvalue < 0.05 else 'Les données suivent une distribution normale (p >= 0.05).'")
 print(f"Variable {var2}: Statistique = {shapiro_var2.statistic:.4f}, p-value = {shapiro_var2.pvalue:.4f}")
 print(f"Les données ne suivent pas une distribution normale (p < 0.05). if shapiro_var2.pvalue < 0.05 else 'Les données suivent une distribution normale (p >= 0.05).'")

```

**Figure 65: Calcul de la covariance et test de normalité (Shapiro-Wilk) pour deux variables**

```

Choix du test basé sur la normalité
if shapiro_var1.pvalue >= 0.05 and shapiro_var2.pvalue >= 0.05:
 # Test paramétrique: corrélation de Pearson
 pearson_stat, pearson_p = stats.pearsonr(data[var1].dropna(), data[var2].dropna())
 print("\nRésultats du test de corrélation de Pearson:")
 print(f"Coefficient de corrélation = {pearson_stat:.4f}")
 print(f"p-value = {pearson_p:.4f}")
 print(f"Il y a une corrélation significative entre les variables (p < 0.05). if pearson_p < 0.05 else 'Aucune corrélation significative entre les variables (p >= 0.05).'")
 print(f"Interprétation de la corrélation : {interpret_correlation(pearson_stat)}")

else:
 # Test non paramétrique: corrélation de Spearman
 spearman_stat, spearman_p = stats.spearmanr(data[var1].dropna(), data[var2].dropna())
 print("\nRésultats du test de corrélation de Spearman:")
 print(f"Coefficient de corrélation = {spearman_stat:.4f}")
 print(f"p-value = {spearman_p:.4f}")
 print(f"Il y a une corrélation significative entre les variables (p < 0.05). if spearman_p < 0.05 else 'Aucune corrélation significative entre les variables (p >= 0.05).'")
 print(f"Interprétation de la corrélation : {interpret_correlation(spearman_stat)}")

```

**Figure 66: Choix du test de corrélation: Pearson ou Spearman, basé sur la normalité des données**

```

Séparer les features et la cible
X = df_resampled_imputed.drop('OSAT', axis=1)
y = df_resampled_imputed['OSAT']

Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Méthode 1 : RFE
model_rfe = RandomForestClassifier(random_state=42)
rfe = RFE(estimator=model_rfe, n_features_to_select=10, step=1)
rfe.fit(X_train, y_train)
rfe_ranking = rfe.ranking_

Méthode 2 : SelectKBest
kbest = SelectKBest(score_func=f_classif, k='all')
kbest.fit(X_train, y_train)
kbest_scores = kbest.scores_

Méthode 3 : Feature Importances from Random Forest
model_importance = RandomForestClassifier(random_state=42)
model_importance.fit(X_train, y_train)
feature_importances = model_importance.feature_importances_

```

**Figure 67:** Code de sélection des caractéristiques avec RFE, SelectKBest, et Random Forest

```

Définir les hyperparamètres à optimiser
param_dist = {
 'n_estimators': [50, 100, 150, 200],
 'max_depth': [None, 10, 20, 30, 40],
 'min_samples_split': [2, 5, 10, 15],
 'min_samples_leaf': [1, 2, 4, 6],
 'max_features': ['auto', 'sqrt', 'log2', None],
 'bootstrap': [True, False],
 'criterion': ['gini', 'entropy'],
 'class_weight': [None, 'balanced'],
 'min_impurity_decrease': [0.0, 0.1, 0.2],
 'oob_score': [True, False]
}

Configurer la recherche aléatoire avec validation croisée
random_search = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
 param_distributions=param_dist,
 n_iter=100,
 cv=3,
 scoring='accuracy',
 n_jobs=-1,
 verbose=2,
 random_state=42)

Entrainer la recherche aléatoire avec les données standardisées
random_search.fit(X_train_scaled, y_train)

```

**Figure 68:** Optimisation des hyperparamètres pour Random Forest

```
Utiliser le meilleur modèle trouvé pour extraire les importances des caractéristiques
best_rf_model = random_search.best_estimator_
importances = best_rf_model.feature_importances_

Créer un DataFrame pour les importances des caractéristiques
feature_importances = pd.DataFrame({'Feature': X.columns, 'Importance': importances})
feature_importances = feature_importances.sort_values(by='Importance', ascending=False)

Afficher les importances des caractéristiques
print("Importances des caractéristiques :")
print(feature_importances)

Définir un seuil pour l'importance des caractéristiques
seuil_importance = 0.035

Sélectionner les caractéristiques importantes
features_importantes = feature_importances[feature_importances['Importance'] > seuil_importance]['Feature']

Afficher les caractéristiques sélectionnées
print("\nCaractéristiques sélectionnées :")
print(features_importantes.tolist())
```

**Figure 69: Importances des caractéristiques par Random Forest**

```
Courbe ROC et AUC pour les caractéristiques importantes
fpr, tpr, thresholds = roc_curve(y_test, y_pred_rf_proba_imp)
roc_auc = auc(fpr, tpr)

Trouver le meilleur seuil basé sur le ROC
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]

Représenter en utilisant le seuil optimal
y_pred_optimal = (y_pred_rf_proba_imp >= optimal_threshold).astype(int)
```

**Figure 70: Courbe ROC pour Random Forest avec AUC.**

```
Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

Standardizing the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

Converting data to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).unsqueeze(1) # Add dimension for compatibility
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).unsqueeze(1)

Calculate class weights
class_weights = compute_class_weight('balanced', classes=np.unique(y), y=y)
class_weights = torch.tensor(class_weights, dtype=torch.float32)
```

**Figure 71: Préparation des données pour PyTorch.**

```
Define the neural network model
class BinaryClassificationModel(nn.Module):
 def __init__(self, input_dim, dropout_prob=0.3):
 super(BinaryClassificationModel, self).__init__()
 self.layer1 = nn.Linear(input_dim, 128)
 self.layer2 = nn.Linear(128, 64)
 self.layer3 = nn.Linear(64, 32)
 self.output = nn.Linear(32, 1)
 self.relu = nn.ReLU()
 self.dropout = nn.Dropout(dropout_prob) # Dropout for regularization

 def forward(self, x):
 x = self.relu(self.layer1(x))
 x = self.dropout(self.relu(self.layer2(x)))
 x = self.relu(self.layer3(x))
 x = torch.sigmoid(self.output(x))
 return x
```

Figure 72: Architecture du modèle de classification binaire dans PyTorch.

```
Define ranges for hyperparameter tuning
lr_range = [0.001, 0.01, 0.1]
batch_size_range = [16, 32, 64]
dropout_range = [0.2, 0.3, 0.4]
num_epochs = 100 # Increased number of epochs

for lr in lr_range:
 for batch_size in batch_size_range:
 for dropout_prob in dropout_range:
 print(f"Training with lr={lr}, batch_size={batch_size}, dropout_prob={dropout_prob}")

 # Update model with new dropout probability
 model = BinaryClassificationModel(input_dim, dropout_prob)

 criterion = nn.BCELoss()
 optimizer = optim.Adam(model.parameters(), lr=lr)

 # Training the model
 for epoch in range(num_epochs):
 model.train()
 optimizer.zero_grad()

 # Forward pass
 outputs = model(x_train_tensor).squeeze()
 loss = criterion(outputs, y_train_tensor.squeeze())

 # Backward pass and optimization
 loss.backward()
 optimizer.step()
```

Figure 73: Réglage des hyperparamètres pour le modèle PyTorch.

```
Final evaluation with the best model
best_model.eval()
with torch.no_grad():
 y_pred = best_model(X_test_tensor).squeeze()
 y_pred_class = (y_pred >= 0.5).float()

Convert tensor to numpy array for sklearn metrics
y_pred_class = y_pred_class.numpy()
y_test_numpy = y_test_tensor.squeeze().numpy()

Print classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test_numpy, y_pred_class))
print("Confusion Matrix:\n", confusion_matrix(y_test_numpy, y_pred_class))
```

**Figure 74:** Sélection du meilleur modèle PyTorch et évaluation finale.

```
Définir le modèle
model = Sequential()
model.add(Dense(124, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

Compiler le modèle
model.compile(optimizer=Adam(learning_rate=0.001),
 loss='binary_crossentropy',
 metrics=['accuracy'])

Entrainer le modèle
history = model.fit(X_train, y_train,
 epochs=200,
 batch_size=16,
 validation_split=0.2,
 verbose=1)
```

**Figure 75:** Architecture classique du modèle Keras.

```
Définir les hyperparamètres à explorer
learning_rates = [1e-5, 1e-4, 1e-3, 1e-2]
dropout_rates = [0.3, 0.4, 0.5]
units1_options = [32, 64, 128]
units2_options = [16, 32, 64]
batch_sizes = [16, 32, 64]
epochs = 20

best_accuracy = 0
best_params = {}

Boucle sur toutes les combinaisons d'hyperparamètres
for learning_rate in learning_rates:
 for dropout_rate in dropout_rates:
 for units1 in units1_options:
 for units2 in units2_options:
 for batch_size in batch_sizes:
 accuracy, model = train_model(X_train, y_train, X_test, y_test, learning_rate, dropout_rate, units1, units2, batch_size, epochs)
 print(f'LR: {learning_rate}, Dropout: {dropout_rate}, Units1: {units1}, Units2: {units2}, Batch Size: {batch_size} => Accuracy: {accuracy}')
 if accuracy > best_accuracy:
 best_accuracy = accuracy
 best_params = {
 'learning_rate': learning_rate,
 'dropout_rate': dropout_rate,
 'units1': units1,
 'units2': units2,
 'batch_size': batch_size
 }

print(f'Best Accuracy: {best_accuracy}')
print(f'Best Hyperparameters: {best_params}'')
```

**Figure 76: Optimisation des hyperparamètres du modèle Keras.**

```
Fonction pour créer le modèle Keras avec des hyperparamètres variables
def create_model(layers=[64], activations=['relu'], optimizer='adam'):
 model = Sequential()
 model.add(Dense(layers[0], input_dim=X_train.shape[1], activation=activations[0]))
 for layer, activation in zip(layers[1:], activations[1:]):
 model.add(Dense(layer, activation=activation))
 model.add(Dense(1, activation='sigmoid')) # Sortie binaire
 model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
 return model

Listes des hyperparamètres à tester
layer_configs = [
 [64],
 [64, 32],
 [128, 64, 32]
]

Diverses fonctions d'activation pour chaque configuration de couches
activation_functions = [
 ['relu'], ['sigmoid'], ['tanh'],
 ['relu', 'relu'], ['sigmoid', 'tanh'], ['tanh', 'relu'],
 ['relu', 'relu', 'relu'], ['sigmoid', 'tanh', 'relu'], ['tanh', 'relu', 'sigmoid']
]

Différents optimiseurs à tester
optimizers = [Adam, SGD, RMSprop]
optimizer_params = [{ 'learning_rate': 0.001}, { 'learning_rate': 0.01}, { 'learning_rate': 0.1}]
```

**Figure 77: Optimisation de l'architecture des couches dans Keras.**