MASTER 2 : ISF

Matière : Machine Learning

Année : 2025-2026

Prédiction de risque de Crédit

Réalisé par :

Saad BENJELLOUN

Clément ROUGERON

Audrey NGOLOHO

Table des matières

I- Contexte du Projet

Ce projet vise à développer un modèle de prédiction du risque de crédit pour des demandes de prêt. L'objectif principal est de construire un système capable d'estimer le score de risque d'un emprunteur potentiel en se basant sur ses caractéristiques financières et démographiques. Cette prédiction permet aux institutions financières de mieux évaluer les risques associés à l'octroi de crédit et d'optimiser leurs décisions d'acceptation ou de refus de prêt.

Le jeu de données initial contient des informations détaillées sur les emprunteurs, incluant leurs revenus, leur historique de crédit, leurs actifs financiers, ainsi que les caractéristiques des prêts demandés. La variable cible à prédire est le **RiskScore**, un indicateur numérique du niveau de risque associé à chaque demandeur.

II- Présentation et analyse des données

Dans cette partie nous nous intéressons à l'exploration des données. C'est l'une des parties les plus importantes d'un projet de Machine Learning puisqu'elle permet de comprendre qualitativement les features et la target, ainsi que les interactions entre icelles. Cela va nous aider à faire les meilleurs choix possibles lors de la sélection de variables et de modèles.

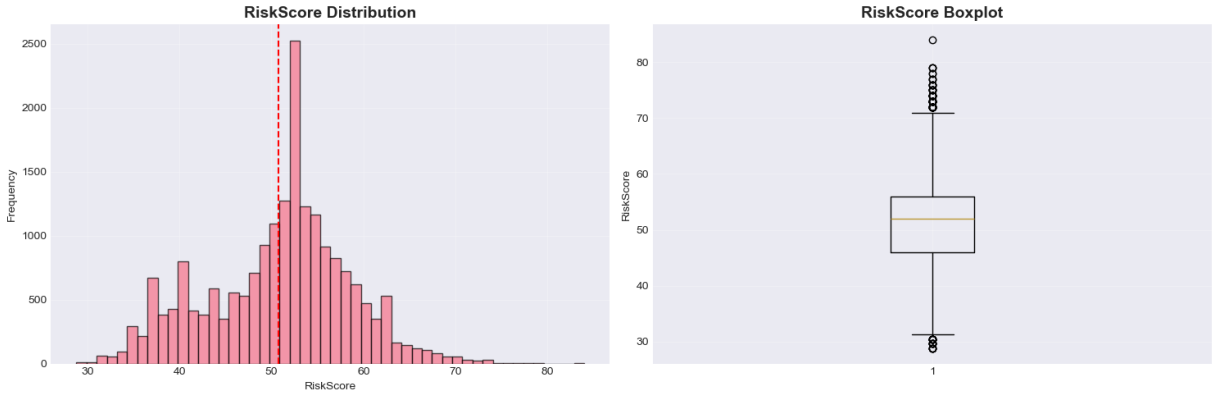
Le jeu de données que nous avons est un tableau de taille 20 001 x 36 dans lequel la première ligne sert à nommer les colonnes et les 20 000 dernières lignes sont 20 000 demandes de prêt bancaire. Les 36 colonnes sont 36 informations sur les demandes de prêt. Ce sont 36 variables que nous devons analyser dans le but de prédire le RiskScore. Il y a 5 variables catégorielles et 31 numériques, dont le RiskScore. Nous avons donc à faire à un problème de régression.

Les 20 000 demandes de prêt sont toutes complètes, nous n'avons aucune donnée manquante. De plus, nous n'avons aucune ligne en doublon.

1- Présentation de la target

Dans cette sous partie nous allons présenter les 36 variables.

Voici la variable target (ou variable cible) : **RiskScore**



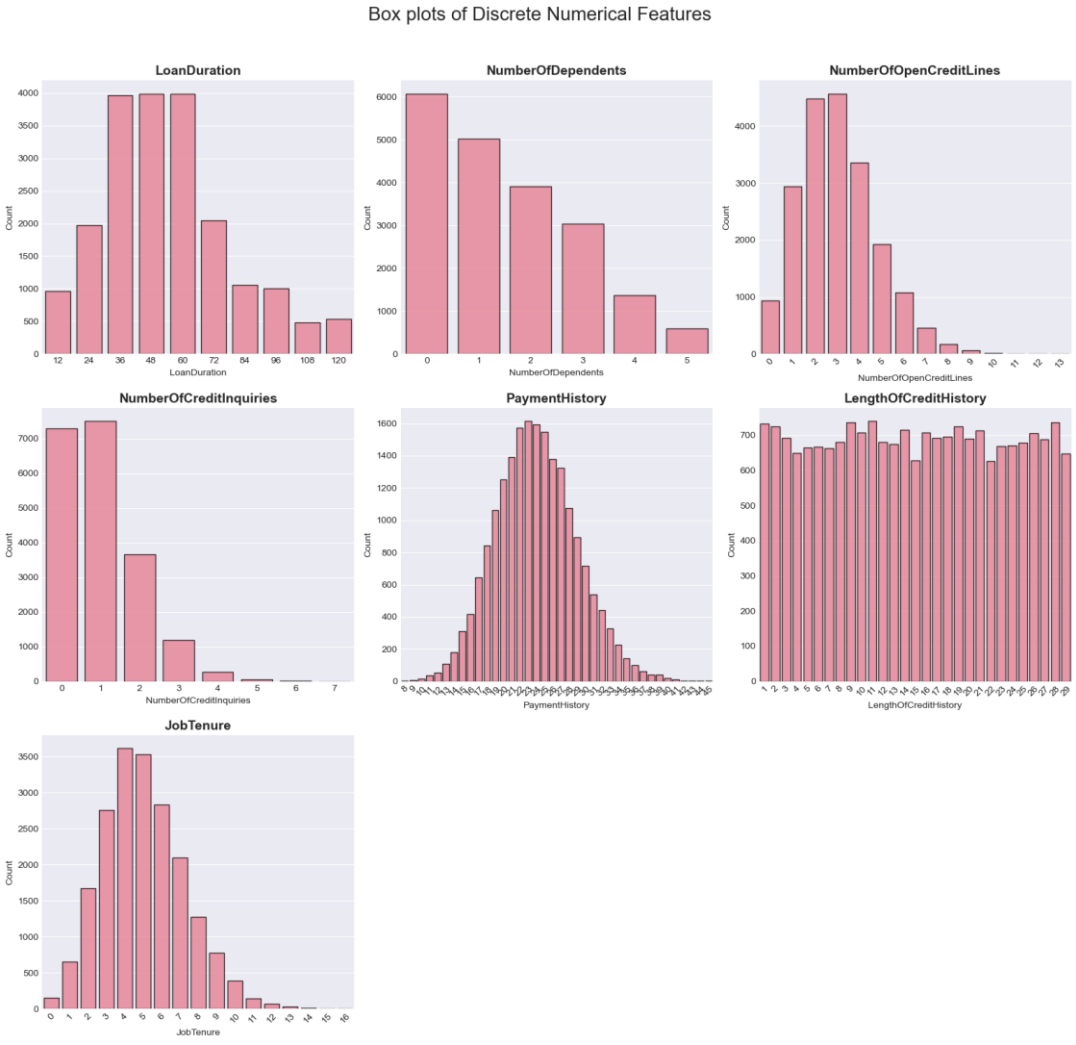
Statistiques de RiskScore : Moyenne: 50.77 ; Médiane: 52 ; Écart-type: 7.78 ; Min: 28.8 ; Max : 84

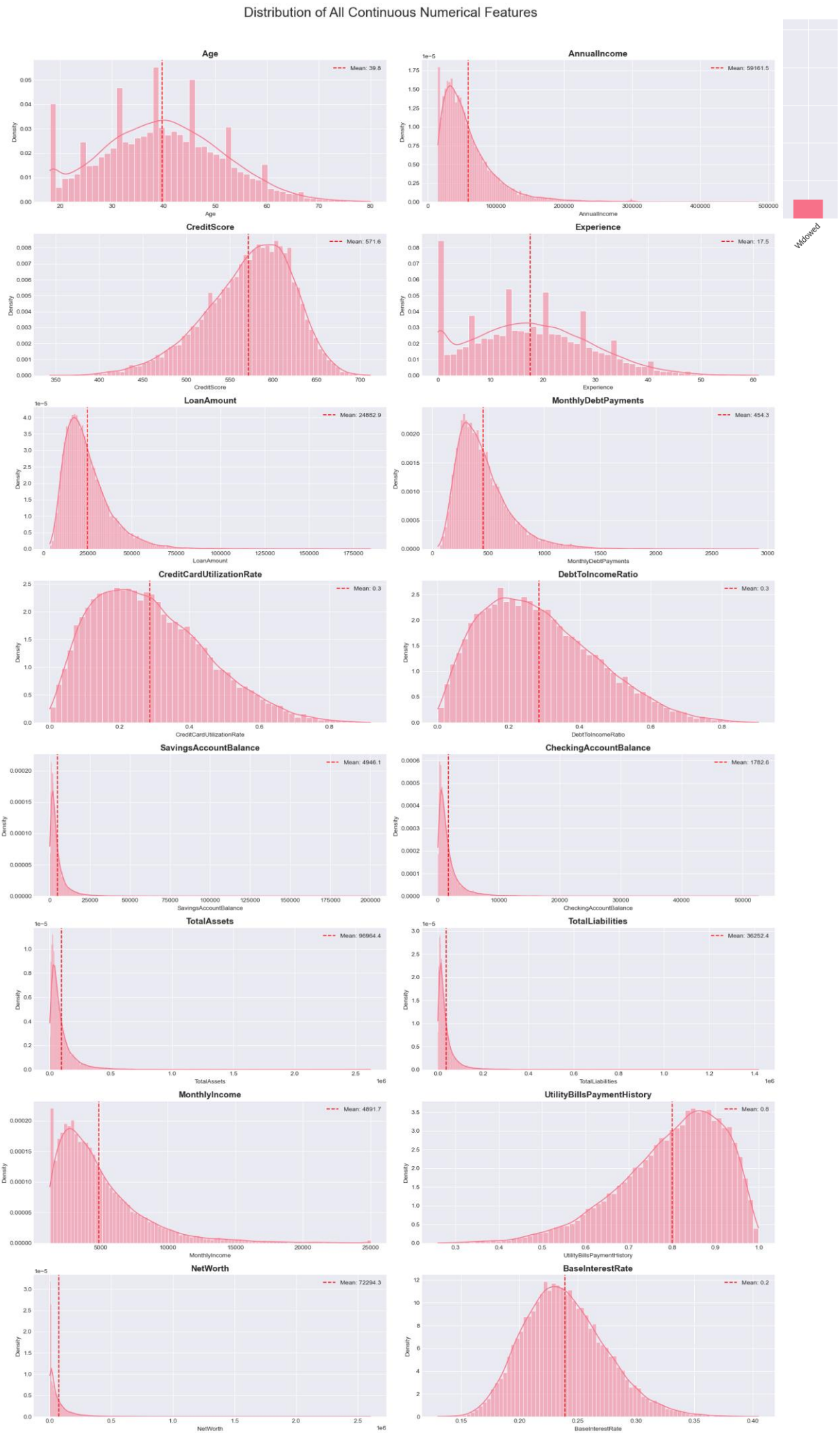
La ligne rouge pointillée verticale du graphique de droite correspond à la moyenne.

L'histogramme révèle une distribution multimodale, suggérant la présence de sous-populations au sein de la base d'emprunteurs (des pics sont visibles à 38, 42, 52 et 62). La distribution est légèrement asymétrique vers la droite, la concentration la plus élevée d'emprunteurs se situe près de la moyenne, mais avec une longue queue contenant des valeurs aberrantes de risque s'étendant vers 80.

Le Boxplot révèle une distribution légèrement asymétrique vers la droite pour le RiskScore, avec une valeur médiane de 52. Alors que les 50 % centraux des données sont étroitement regroupés entre les scores de 46 et 56, la présence de nombreuses valeurs aberrantes dans la partie supérieure (scores > 70) indique un sous-ensemble de prêts présentant des caractéristiques de risque nettement supérieures à la norme.

2- Présentation des variables




Distribution of All Continuous Numerical Features
Age
Mean: 39.8
Density
Age
AnnualIncome
Mean: 59161.5
1e−5
AnnualIncome
CreditScore
Mean: 571.6
CreditScore
Experience
Mean: 17.5
Experience
LoanAmount
Mean: 24882.9
1e−5
LoanAmount
MonthlyDebtPayments
Mean: 454.3
MonthlyDebtPayments
CreditCardUtilizationRate
Mean: 0.3
CreditCardUtilizationRate
DebtToIncomeRatio
Mean: 0.3
DebtToIncomeRatio
SavingsAccountBalance
Mean: 4946.1
SavingsAccountBalance
CheckingAccountBalance
Mean: 1782.6
CheckingAccountBalance
TotalAssets
Mean: 96964.4
1e−5
1e6
TotalAssets
TotalLiabilities
Mean: 36252.4
1e−5
1e6
TotalLiabilities
MonthlyIncome
Mean: 4891.7
MonthlyIncome
UtilityBillsPaymentHistory
Mean: 0.8
UtilityBillsPaymentHistory
NetWorth
Mean: 72294.3
1e−5
1e6
NetWorth
BaseInterestRate
Mean: 0.2
BaseInterestRate
Widowed

Age : La variable Age donne l'âge de la personne demandant le prêt. La distribution des âges présente une distribution assez étalée avec une moyenne autour de 40 ans, bien que l'on observe plusieurs pics locaux, on parle de distribution multimodale. Il y a quand même beaucoup de jeunes actifs et de pré-retraités, aucune catégorie n'est omise.

AnnualIncome : Cette variable donne le revenu annuel du demandeur du prêt. La distribution de la variable montre une forte asymétrie positive. Le pic de fréquence est à gauche, suivi d'une longue "queue" vers la droite représentant les hauts salaires. C'est une distribution classique en finance que l'on appelle loi de Pareto. La moyenne est tirée vers le haut par quelques clients très riches. Pour la modélisation, il sera impératif d'appliquer une transformation logarithmique sur ces variables pour réduire l'impact des valeurs extrêmes.

CreditScore : La variable CreditScore donne le score de crédit du demandeur de prêt. C'est un nombre à trois chiffres qui est calculé à partir de toutes les opérations effectuées sur une carte de crédit. Un score de crédit élevé signifie que la personne gère bien ses crédits. La distribution de cette variable présente une asymétrie négative. La masse des données est concentrée sur la droite (scores élevés), avec une queue qui s'étire vers la gauche (scores faibles). La majorité des clients de cette base de données sont de "bons" payeurs avec des scores. Les clients situés dans la queue gauche (scores < 450) représentent le segment à haut risque.

EmployementStatus : Cette variable renvoie si le demandeur est salarié ("Employed"), indépendant ("Self-Employed") ou sans emploi ("Unemployed"). La distribution est extrêmement déséquilibrée. La catégorie "Employed" écrase totalement les autres, représentant la très grande majorité des emprunteurs (plus de 80% des individus). Les catégories "Self-Employed" et "Unemployed" sont marginales. Cette sur-représentation des salariés indique que la base de données est biaisée vers des profils aux revenus probablement stables. Cela suggère que le modèle prédictif sera très performant pour les salariés classiques, mais pourrait manquer de précision pour évaluer le risque des indépendants ou des personnes sans emploi, faute de données suffisantes.

EducationLevel : Cette variable donne le plus haut diplôme académique obtenu par demandeur du prêt. Il y a 5 catégories Lycée ("High School"), Bac+2 ("Associate"), Licence ("Bachelor"), Master ("Master") et Doctorat ("Doctorate"). Les niveaux "Bachelor" (Licence) et "High School" sont les plus représentés, avec environ 30% des individus chacun. Les niveaux supérieurs "Master", "Doctorate" sont moins fréquents, ces diplômes étants étant les moins délivrés. Le portefeuille de prêts couvre une large gamme de niveaux éducatifs, mais cible principalement la classe moyenne (niveau bac à bac+3).

LoanAmount : Cette variable donne le montant du prêt demandé. La distribution de ce graphique est similaire à celle de AnnualIncome avec une forte asymétrie positive. Le pic de fréquence est à gauche, suivi d'une longue "queue" vers la droite représentant les très gros prêts. La moyenne est tirée vers le haut par quelques clients très riches. Là encore, pour la modélisation, il sera impératif d'appliquer une transformation logarithmique sur cette variable pour réduire l'impact des valeurs extrêmes.

LoanDuration : Cette variable exprime la durée du prêt, en mois. La distribution est donc discrète avec des pics très nets sur les multiples de 12 (le remboursement d'un prêt se fait sur un nombre entier d'années). Les durées intermédiaires ou très longues, au-delà de 96 mois, sont beaucoup moins fréquentes. Cela reflète la standardisation des produits financiers. La majorité des contrats se situent entre 3 et 5 ans, ce qui est typique pour des prêts à la consommation ou automobiles.

MaritalStatus : Cette variable dit si le demandeur est célibataire ("Single"), marié ("Married"), divorcé ("Divorced") ou veuf ("Widowed"). Les demandeurs mariés sont dominants dans ce jeu de données, avec environ 50% des individus, suivi par les célibataires, avec environ 30% des individus. Les divorcés et les veufs représentent une minorité de la base de données. La prédominance des emprunteurs mariés peut suggérer une certaine stabilité financière, par exemple le fait d'avoir deux revenus par foyer, ce qui est généralement un facteur positif pour l'évaluation du risque de crédit.

NumberOfDependents : Cette variable donne le nombre de personne à charge dans le foyer, c'est-à-dire le nombre de personne du foyer qui n'ont pas de revenus. Généralement ce sont les enfants du demandeur mais cela peut aussi être des parents retraités ou bien des proches avec des problèmes de santé (handicap, ...). On observe une décroissance quasi-linéaire. La catégorie la plus représentée est "0 dépendant" (plus de 6 000 cas), et le nombre d'emprunteurs diminue à mesure que le nombre de personnes à charge augmente. La base de données est majoritairement composée de personnes sans charge de famille directe (célibataires ou couples sans enfants), ce qui peut théoriquement laisser un "reste à vivre" plus important pour le remboursement, toutes choses égales par ailleurs.

HomeOwnershipStatus : Cette variable dit si le demandeur du prêt est locataire ("Rent") ou propriétaire ("Own") de son logement, s'il a un crédit immobilier en cours ("Mortgage") ou s'il est dans une autre situation ("Other"), par exemple s'il est hébergé chez un proche. La majorité des emprunteurs ont un crédit immobilier en cours ou sont locataires (respectivement 40% et 30%). Ceux qui sont pleinement propriétaires représentent un groupe plus restreint (environ 20%). Cela signifie que la grande majorité des clients ont déjà une charge financière mensuelle importante liée au logement (paiement loyer ou remboursement hypothécaire). C'est un point de vigilance important pour le calcul du ratio dette/revenu.

MonthlyDebtPayments : Cette variable donne le montant mensuel que le demandeur paye tous les mois pour rembourser ses crédits déjà contractés (prêt immobilier, étudiant, automobile ...). La distribution de cette variable est asymétrique avec une queue vers la droite. Cela signifie que beaucoup de personnes ont peu de remboursements mensuels et qu'un petit nombre fait augmenter la moyenne totale avec des sommes plus élevées.

CreditCardUtilizationRate : Cette variable donne le taux, entre 0 et 1, d'utilisation de la carte bancaire du demandeur. Un faible score indique que le demandeur n'utilise pas sa carte compulsivement. La distribution suit une loi Beta légèrement asymétrique, centrée autour de 0.3. Les valeurs extrêmes (0.0 ou >0.8) sont plus rares.

NumberOfOpenCreditLines : Cette variable donne le nombre de crédits contractés par le demandeur. Avoir 2 à 4 crédits est un comportement financier standard (par exemple une carte de crédit et un prêt auto). Ici, la distribution est asymétrique vers la droite. La grande majorité des emprunteurs possède entre 2 et 4 lignes de crédit actives. Il est rare d'avoir plus de 8 lignes ouvertes simultanément. Cependant, la "queue" de distribution vers la droite, plus de 10 crédits, identifie des profils potentiellement surendettés qu'il faudra surveiller de près.

NumberOfCreditInquiries : Cette variable donne le nombre de demandes de crédits effectuées par le demandeur. Un nombre élevé de demandes de crédit sur une courte période est souvent un signe de détresse financière (le client cherche de l'argent partout). Ici, la distribution chute drastiquement. La vaste majorité des clients a fait 0 ou 1 seule demande de crédit récemment. Les profils ayant fait 3 demandes ou plus sont très minoritaires.

DebtToIncomeRatio : Cette variable donne le ratio, entre 0 et 1, entre les dettes à rembourser du demandeur et son salaire. La distribution de cette variable est très similaire à **CreditCardUtilizationRate.**

BankruptcyHistory : Cette variable vaut 1 si le demandeur a déjà fait faillite et 0 sinon. Il y a environ 95% de personnes qui n'ont jamais fait faillite. Cela posera donc un problème pour entrainer les modèles sur des personnes qui ont fait défaut.

LoanPurpose : Cette variable donne la raison de la demande de prêt. Il y a les catégories immobilier ("Home"), automobiles ("Auto"), études ("Education"), consolidation de dettes ("Debt Consolidation") et autres ("Other"). Les motifs sont dominés par l'immobilier et la consolidation de dettes, avec respectivement 30% et 25% des demandes. Les prêts auto et éducation suivent ensuite. Il y a deux dynamiques distinctes : les prêts d'investissement (Maison, Éducation) et les prêts de gestion de passif (Consolidation de dettes). Le risque associé à la consolidation de dettes est souvent différent de celui d'un achat immobilier, car il peut signaler des difficultés financières préexistantes.

PreviousLoanDefaults : Cette variable vaut 1 si le demandeur a déjà fait défaut sur un prêt antérieur et 0 sinon. Environ 10% de ce jeu de données a déjà fait défaut. Les classes sont donc déséquilibrées et cela peut avoir pour conséquence de mauvaises prédictions sur les personnes ayant déjà fait défaut.

PaymentHistory : Cette variable donne l'historique des paiements passés. La distribution de cette variable est une loi Normale quasi parfaite. La plupart des clients ont un comportement de paiement "moyen" ou standard. Les deux extrémités de la courbe permettront de segmenter facilement les très mauvais payeurs, à gauche, des clients "premium", à droite.

LengthOfCreditHistory : Cette variable donne la durée de l'historique de crédit. Contrairement aux autres variables, celle-ci présente une distribution uniforme. Il y a autant de clients avec 2 ans d'historique qu'avec 25 ou 30 ans. Cette uniformité est atypique pour des données réelles car on s'attendrait généralement à voir moins de personnes avec 30 ans d'historique qu'avec 5 ans. Cela suggère qu'il s'agit de données synthétiques.

SavingsAccountBalance : Cette variable donne le montant de l'épargne du demandeur. C'est l'une des distributions les plus asymétriques. Énormément de personnes ont très peu d'épargne mais la moyenne est haute à cause de quelques individus qui ont beaucoup d'épargne.

CheckingAccountBalance : Cette variable donne le solde du compte courant du demandeur. Comme pour les comptes d'épargne, il y a une très forte asymétrie et il faudra surement utiliser le logarithme de cette variable pour prédire le RiskScore.

TotalAssets : Cette variable donne la valeur des actifs du demandeur. Elle est distribuée comme les deux précédentes avec un pic en 0 et une queue très fine.

TotalLiabilities : Cette variable donne la valeur des dettes (passif) du demandeur. Comme les trois variables ci-dessus, elle est extrêmement asymétrique.

NetWorth : Valeur nette totale du demandeur. C'est la différence entre les actifs et les passifs. Ainsi la distribution de cette variable est autant asymétrique que les quatre du dessus.

MonthlyIncome : Revenus mensuels du demandeur. La distribution de cette variable est similaire à celle de AnnualIncome car elles sont multiples l'une de l'autre. Il faudra retirer l'une des deux variables lors de la sélection de variables.

UtilityBillsPaymentHistory : Cette variable est un nombre entre 0 et 1 qui traduit le paiement des factures. Plus le nombre est élevé et plus les factures ont bien été réglées. La majorité des demandeurs de ce jeu de données sont des bons payeurs.

JobTenure : Cette variable donne l'ancienneté du demandeur dans son poste actuel. La distribution montre un pic autour de 4 à 6 ans d'ancienneté, avec une baisse progressive au-delà de 7 ans. Les très faibles anciennetés (0-1 an) sont moins fréquentes que les anciennetés moyennes. La stabilité professionnelle semble correcte. Le fait qu'il y ait peu de profils très précaires (0 an) est rassurant pour le risque de défaut lié à la perte d'emploi.

BaseInterestRate : Cette variable donne le taux d'intérêt de base du prêt demandé. Cette distribution quasi-Gaussienne, bien symétrique autour de sa moyenne. Cela suggère que la politique de tarification de la banque est standardisée et suit une logique cohérente, probablement indexée sur le marché, avec peu de dérogations extrêmes.

InterestRate : Cette variable donne le taux d'intérêt servi du prêt demandé. Elle est distribuée comme BaseInterestRate.

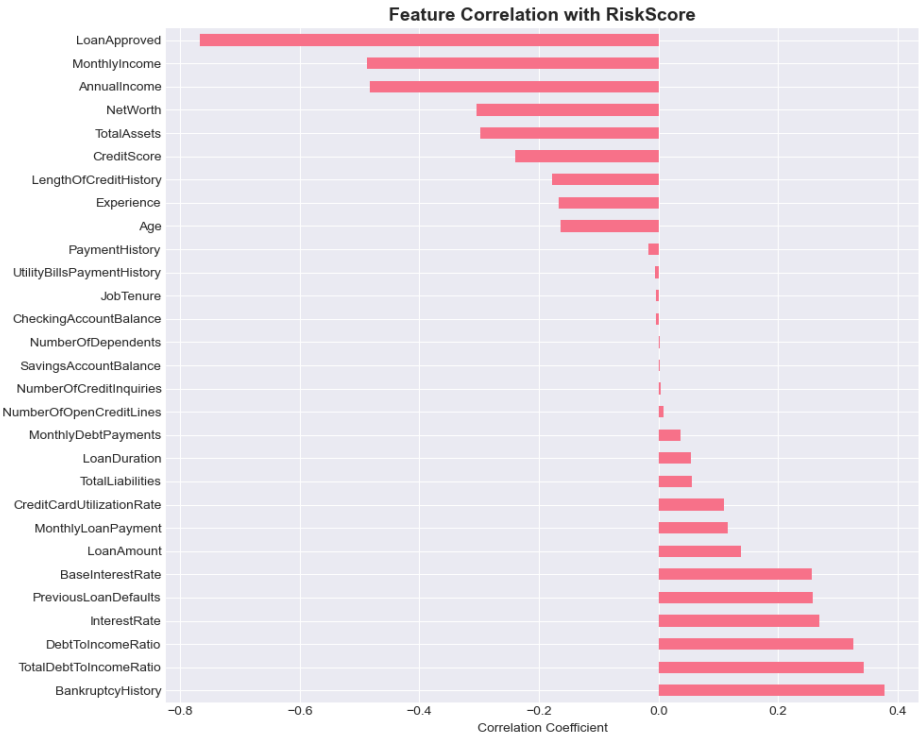
MonthlyLoanPayment : Cette variable donne le montant des mensualités du prêt demandé. Elle est distribuée comme LoanAmount.

TotalDebtToIncomeRatio : Cette varibale donne le ratio dettes / revenus en incluant les dettes du prêt demandé.

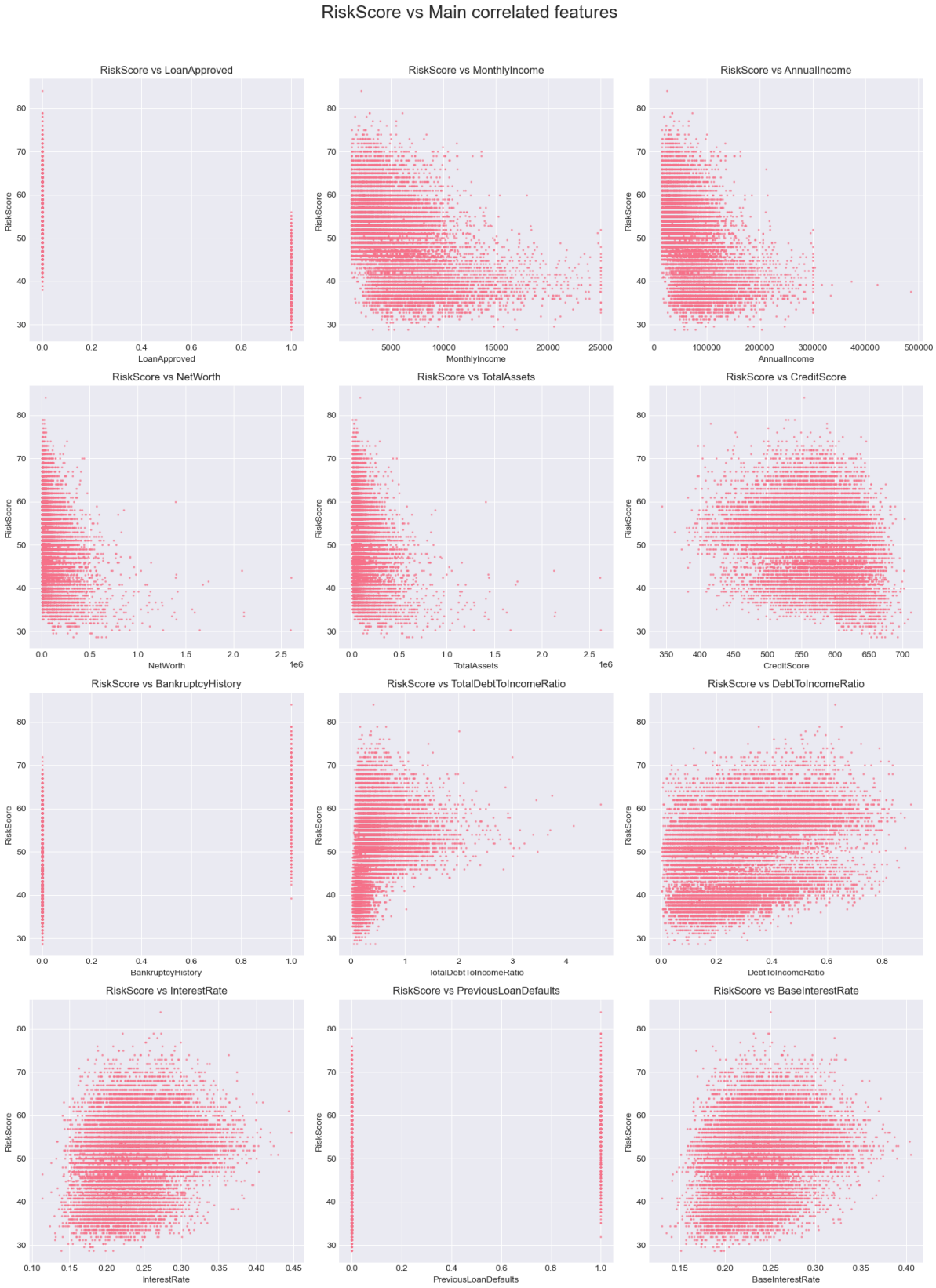
LoanApproved : Cette variable vaut 0 si le prêt est réfusé et 1 s'il est accepté. Seulement un quart des prêts sont acceptés.

3- Analyse des données et corrélation des variables avec la target

Nous allons maintenant analyser les données. Une première approche est de comparer les corrélations entre RiskScore et les autres variables. C'est l'objet du prochain graphique.



Maintenant que l'on sait quelles variables sont corrélées à la target nous pouvons tracer le RiskScore en fonction de ces variables corrélées.



LoanApproved : Ce graphique est le plus révélateur. Pour la valeur 0, le *RiskScore* couvre toute l'échelle (de 30 à plus de80). En revanche, pour la valeur 1, le nuage de points est "coupé" net vers le haut : il n'y a quasiment aucun prêt approuvé avec un *RiskScore* supérieur à 55. Cela confirme la cohérence de la base de données. Le *RiskScore* est effectivement utilisé comme critère d'éligibilité. La banque a un "appétit au risque" limité et rejette systématiquement les scores trop élevés.

Les Taux d'Intérêt (InterestRate, BaseInterestRate) : Le nuage de points forme une diagonale montante assez nette : plus le taux est élevé, plus le *RiskScore* est élevé. C'est la confirmation du principe de "Risk-Based Pricing" (tarification au risque). La banque facture plus cher les clients risqués pour compenser la probabilité de défaut plus élevée.

Les Indicateurs de Richesse (Income, Assets, NetWorth) : *MonthlyIncome*, *AnnualIncome*, *TotalAssets* et *NetWorth*, on observe une forme en "L" (ou une concentration en bas à gauche qui s'étire vers la droite). Les points situés loin sur l'axe X (les clients riches) sont presque toujours situés très bas sur l'axe Y (faible risque). Il existe une corrélation négative : plus on est riche, moins on est risqué. Cependant, la relation n'est pas linéaire sur toute la ligne : une fois un certain seuil de richesse atteint, le risque atteint un plancher et ne descend plus.

CreditScore : Le nuage de points montre une tendance descendante diffuse. Les *RiskScores* élevés, i.e. supérieurs à 60, se trouvent majoritairement chez les gens ayant un *CreditScore* moyen ou faible, inférieur à 550. À l'inverse, un très bon *CreditScore*, c'est-à-dire au-dessus de 650, garantit presque toujours un *RiskScore* contenu sous les 50. Le *CreditScore* est une composante majeure du *RiskScore* interne, mais la dispersion du nuage suggère que le modèle que l'on doit construire devra ajouter d'autres facteurs pour affiner le jugement.

Les Ratios d'Endettement (TotalDebtToIncomeRatio, DebtToIncomeRatio) : On discerne une légère pente ascendante. Les ratios élevés sont associés à des scores de risque plus élevés, mais la dispersion reste très importante. L'endettement impacte le risque, mais de manière moins "mécanique" que le taux d'intérêt. Un client peut avoir un ratio élevé mais être peu risqué s'il a des actifs importants à côté.

Bien que nous ayons désormais une liste des variables corrélées à la target, rien ne garantit que les variables ne soient pas corrélées entre elles. Ces retraitements sont l'objet de la prochaine section.

III- Retraitements Appliqués

1- Ingénierie des Variables

La 2e étape du projet a consisté à enrichir le jeu de données initial par la création de nouvelles variables susceptibles d'améliorer la capacité prédictive des modèles. Cette phase d'ingénierie des variables s'est articulée autour de plusieurs axes stratégiques.

- Indicateurs temporels : Nous avons extrait des informations temporelles à partir de la variable **ApplicationDate.** Ces variables incluent le mois de demande, le jour de la semaine, le trimestre et un indicateur de fin de mois. Ces éléments permettent de

capturer d'éventuelles variations saisonnières dans les comportements d'emprunt et les politiques de crédit.

- Indicateurs de stabilité financière et de solvabilité : Le **ratio de stabilité des revenus** mesure le revenu annuel divisé par l'ancienneté dans l'emploi. Le **ratio d'épargne** sur revenu compare le solde du compte d'épargne au revenu annuel. Ces ratios donnent une vision plus nuancée de la capacité de l'emprunteur à gérer ses finances et à faire face à ses engagements.
- Indicateurs du comportement de crédit : **L'indice de stress du crédit** combine le taux d'utilisation des cartes de crédit avec le nombre de lignes de crédit ouvertes. Le taux de demandes récentes rapporte le nombre de demandes de crédit à la durée de l'historique de crédit. Un indicateur binaire signale les emprunteurs dont le taux d'utilisation des cartes dépasse 30%, seuil généralement considéré comme risqué.
- Indicateurs de la structure du prêt : Concernant la structure du prêt, nous avons calculé le **ratio prêt sur revenu** et **le ratio prêt sur actifs** pour évaluer la proportionnalité de la demande. L'écart d'intérêt entre le taux proposé et le taux de base permet d'identifier les prêts à risque plus élevé. Le coût total du prêt et le fardeau de remboursement mensuel donnent une vision de la charge financière totale.
- Des variables d'interaction ont été créées pour capturer les effets combinés de certaines caractéristiques. Par exemple, **l'interaction entre l'âge et le score de crédit**, ou entre le **montant du prêt et le taux d'intérêt**. Ces variables permettent aux modèles de mieux saisir les relations complexes entre les différents facteurs.

Au total, cette phase d'ingénierie a permis de passer de 36 variables initiales à plus de 70 variables enrichies, offrant ainsi une représentation plus complète du profil de risque des emprunteurs. Le feature engineering est dans le notebook 2, on peut s'y référer pour avoir une liste de toutes les variables qui ont été créées.

2- Prétraitement des Données

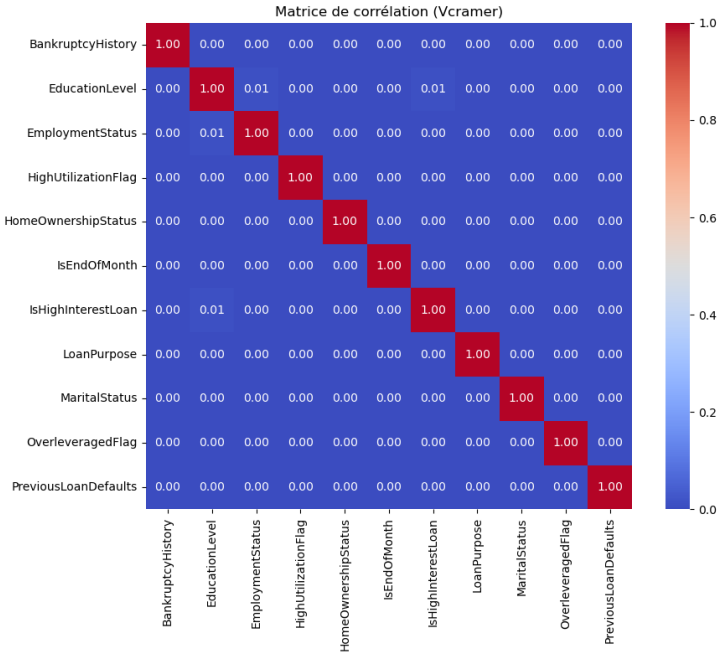
A- Prétraitement général

Après l'enrichissement des variables, nous avons procédé à une phase de prétraitement pour préparer les données à la modélisation. Cette étape a nécessité plusieurs décisions importantes concernant le traitement des variables et la gestion de leurs corrélations.

- Etape 1 : Nous avons d'abord retiré deux variables essentielles qui ne devaient pas être utilisées dans les modèles. La variable **ApplicationDate** a été supprimée car une date brute ne peut pas servir directement de prédicteur. Nous avions déjà extrait ses composantes temporelles utiles lors de la phase précédente. La variable **LoanApproved** a également été retirée car il s'agit d'une information postérieure à la décision de crédit. L'inclure dans le modèle créerait une fuite d'information et biaiserait complètement les prédictions.
- Etape 2 : L'analyse de variance des variables numériques a révélé que certaines variables présentaient une très faible variance. Par exemple, le ratio **CheckingToIncomeRatio** montrait une variance inférieure à 0.01, indiquant que la plupart des emprunteurs ont des ratios similaires. De plus, les variables liées aux taux d'intérêt comme **BaseInterestRate**, **InterestRate**, **InterestSpread** possédaient également une variance faible. Cela pouvant

s'expliquer par le fait que l'ordre de grandeur de ces dernières est relativement faible. Cependant, nous avons décidé de conserver ces variables car elles peuvent quand même contenir des signaux utiles, même si faibles. Cette décision pourrait être révisée ultérieurement si les modèles indiquent que ces variables n'apportent rien.

B- Corrélation entre les variables catégorielles



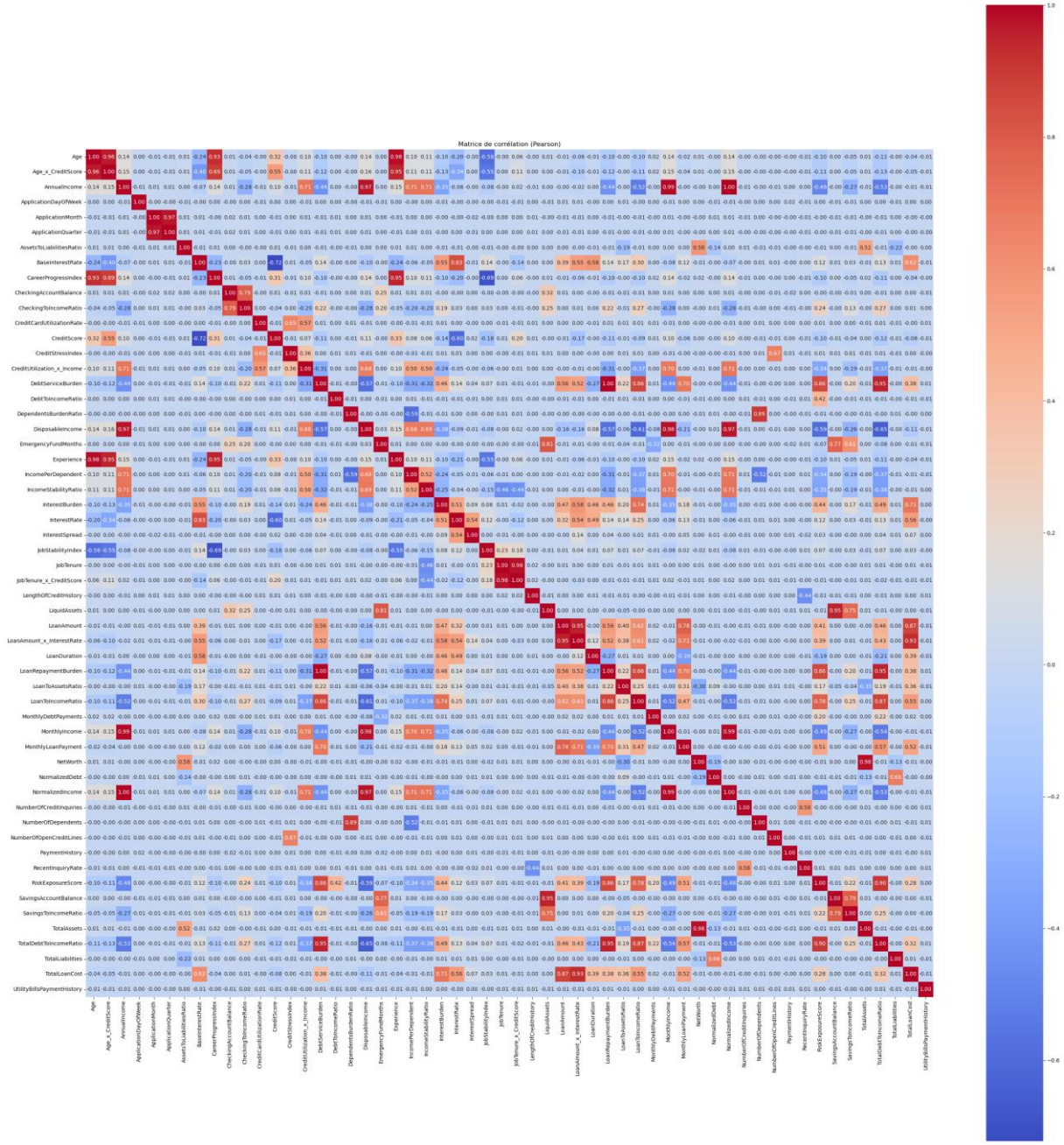
On peut remarquer que les variables catégorielles sont pas du tout corrélées entre elles. On peut les laisser inchangées (on ne retire aucune variable). Toutefois, on devra les transformer en format exploitable par nos différents algorithmes de machine learning (one hot encoding ou ordered encoding).

C- Transformation des variables catégorielles

Nous avons décidé de ne pas traiter toutes ces variables de la même manière, mais de choisir un encoding pertinent par rapport au type de cette dernière. Nous avions différents types :

- Variables binaires : Ces dernières restent inchangées. On y retrouve par exemple : **BankruptcyHistory**, **HighUtilizationFlag**, **IsEndOfMonth** ...
- Variables catégorielles sans ordre : Celles-ci doivent être transformée en one hot encoding. On peut citer : **EmploymentStatus**, **HomeOwnershipStatus**, **LoanPurpose...**
- Variables catégorielles avec ordre : Cela ne concerne que la variable **EducationLevel** qui a pour modalités ['Master' 'Associate' 'Bachelor' 'High School' 'Doctorate']. On peut clairement voir la notion d'ordre parmi ces dernières. Nous avons donc choisi un ordered encoding afin de conserver l'information implicite contenue dans cette la variable.

D- Analyse de la corrélation entre les variables numériques



Voici une heatmap de la corrélation entre les variables numériques, nous pouvons zoomer lorsque le document est en format numérique pour avoir la valeur exacte. Sinon nous pouvons nous référer uniquement aux couleurs. On peut également se référer au notebook 3.

Constat : L'analyse de corrélation entre variables numériques a mis en évidence un nombre important de paires de variables fortement corrélées. Cette situation est normale car plusieurs variables mesurent des aspects similaires de la situation financière. Par exemple, **DebtToIncomeRatio** et **TotalDebtToIncomeRatio** sont naturellement corrélés car ils mesurent tous deux le niveau d'endettement. De même, les variables dérivées du feature engineering sont corrélées avec les variables dont elles sont issues.

Problème : La multicolinéarité pose un problème spécifique pour les modèles linéaires comme Ridge, Lasso ou ElasticNet. Ces modèles peuvent avoir des difficultés à distinguer l'effet individuel de variables corrélées, ce qui peut rendre les coefficients instables et difficiles à interpréter. En revanche, les modèles basés sur des arbres de décision ne sont pas sensibles à ce phénomène car ils sélectionnent les variables de manière séquentielle et peuvent naturellement gérer les redondances.

Solution : Nous avons décidé d'étudier la corrélation entre les variables 2 à 2. Si cette corrélation est supérieure à 0.8 (valeur arbitraire), nous étudions la corrélation de chacune des variables de la paire avec la target **(RiskScore).** Nous gardons la variable de la paire la plus corrélée avec la target, et nous supprimons l'autre. Nous avons procédé ainsi car énormément de variables étaient corrélées entre elles et une réduction du dataset était nécessaire afin d'alléger les calculs. Les variables hautement corrélées, gardées et supprimées peuvent être retrouvée dans le notebook 3.

E- Préparation des différents datasets

Face à cette situation, nous avons adopté une stratégie de préparation de données différenciée. Nous avons créé trois jeux de données distincts adaptés aux différents types de modèles

1- Le premier jeu de données conserve toutes les variables (y compris les variables issues du feature engineering) sans traitement de la multicolinéarité. Il est destiné aux modèles basés sur des arbres comme Random Forest, XGBoost, LightGBM. Ces modèles peuvent exploiter toutes les informations disponibles sans être pénalisés par les corrélations.

2- Le deuxième jeu de données applique une réduction des variables corrélées pour les modèles linéaires. Pour chaque paire de variables dont la corrélation dépasse 0.8 en valeur absolue, nous avons conservé celle qui présente la plus forte corrélation avec la variable cible **RiskScore**. Cette approche permet d'éliminer les redondances tout en maximisant le contenu informatif des variables retenues. Environ 15 variables ont été supprimées lors de ce processus, réduisant le jeu de données à environ 38 variables.
On peut se référer au notebook 3 pour avoir la liste des variables extrêmement corrélées et la corrélation de chacune avec la target.

3- Le troisième jeu de données concerne spécifiquement CatBoost, un algorithme qui peut gérer nativement les variables catégorielles sans encodage préalable. Pour ce modèle, nous avons conservé les variables catégorielles dans leur format original et fourni l'information sur leurs indices au modèle.

Cette approche multi-pipelines permet d'optimiser les performances de chaque type de modèle en adaptant les données à leurs spécificités algorithmiques, plutôt que d'appliquer un prétraitement unique qui pourrait pénaliser certains modèles.

F- Split train/test et Scaling

Avant d'enregistrer chacun des datasets, nous les avons divisés en 2 parties : Train 80% et Test 20%. Nous nous sommes bien assurés que le scaling se faisait en fonction des paramètres du Train (la moyenne du train et la variance du train) afin d'éviter tout biais de data leakage de notre test vers notre train.

Remarque : Les variables catégorielles ne subissent pas de scaling car elles sont binaires après l'encodage. Toutefois, la variable **EducationLevel** ayant subi un ordered encoding au lieu d'un one-hot encoding a dû être scalée au même titre que les variables numériques.

IV- Modélisation 1 : Modèles Linéaires Régularisés (Ridge, Lasso, ElasticNet)

1- Présentation Générale

Les modèles de régression linéaire régularisée constituent notre première famille de modèles. Ces algorithmes s'appuient sur des relations linéaires entre les variables et la cible, avec des mécanismes de régularisation pour contrôler le surapprentissage. Trois variantes ont été testées : Ridge (régularisation L2), Lasso (régularisation L1) et ElasticNet (combinaison L1 et L2).

Ces modèles partagent plusieurs caractéristiques communes. Ils utilisent tous le jeu de données prétraité pour modèles linéaires, où les variables fortement corrélées ont été réduites. La répartition entre ensemble d'entraînement et test est identique : 80/20, soit environ 8000 observations pour l'entraînement et 2000 pour le test.

2- Stratégie d'Hyperparamétrage pour le Grid Search

Pour Ridge, nous avons exploré 11 valeurs d'alpha (0.001 à 1000.0) et 7 solveurs différents, soit 77 combinaisons testées via validation croisée à 5 plis (385 modèles entraînés). Le paramètre alpha contrôle l'intensité de la régularisation L2.

Pour Lasso, 9 valeurs d'alpha ont été testées avec 2 méthodes de sélection (cyclic/random), donnant 18 combinaisons et 90 modèles entraînés. La régularisation L1 de Lasso permet d'annuler complètement certains coefficients, effectuant une sélection automatique de variables.

Pour ElasticNet, nous avons testé 8 valeurs d'alpha, 7 valeurs de l1_ratio (ratio entre régularisation L1 et L2) et 2 méthodes de sélection, soit 112 combinaisons et 560 modèles entraînés.

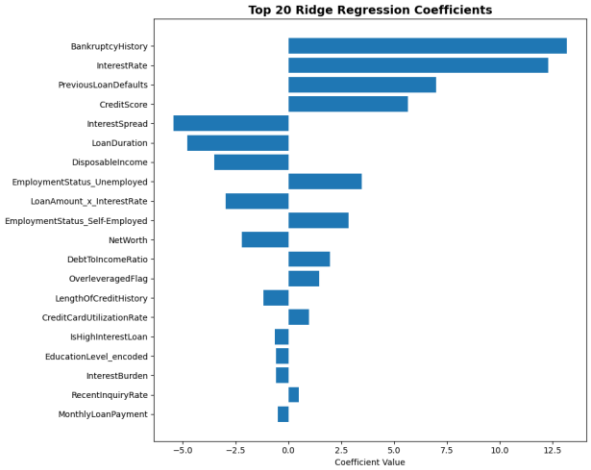
La métrique d'évaluation était la RMSE en validation croisée, choisie car elle pénalise fortement les erreurs importantes, ce qui est souhaitable dans un contexte de risque de crédit.

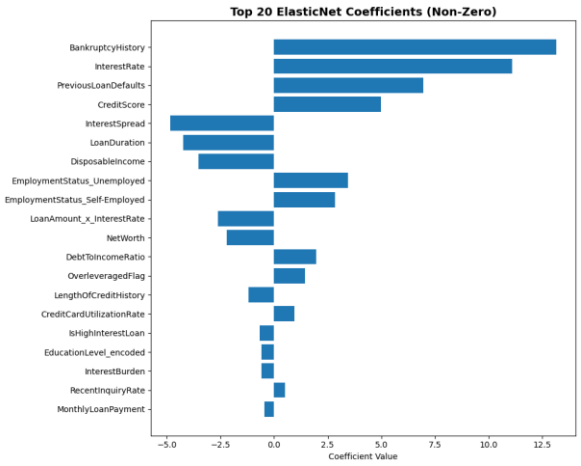
L'ensemble des résultats peuvent être retrouvés dans le notebook 4.

3- Résultats Comparatifs

Les trois modèles ont obtenu des performances remarquablement similaires :

Modèle	Hyperparamètres	RMSE (Train)	MAE (Train)	R^2 (Train)	RMSE (Test)	MAE (Test)	R^2 (Test)	Nombre de variables
Ridge	α = 10.0, solveur = cholesky	3.43	2.69	0.80	3.65	2.81	0.79	65
Lasso	α = 0.1, sélection = cyclic	3.43	2.69	0.80	3.65	2.81	0.79	~45
ElasticNet	α = 0.1, l1_ratio = 0.5, sélection = cyclic	3.43	2.69	0.80	3.65	2.81	0.79	~50





4- Analyse Globale des Modèles Linéaires

Les performances quasi-identiques des trois modèles (R^2 de 0.79) suggèrent que pour ce problème, la structure linéaire capture une bonne partie de la relation entre variables et cible.

L'écart minimal entre entraînement et test confirme que les mécanismes de régularisation fonctionnent efficacement pour éviter le surapprentissage.

Les différences se situent principalement au niveau de la sélection de variables. Ridge conserve toutes les variables mais réduit leur amplitude. Lasso effectue une sélection agressive en éliminant 20 variables. ElasticNet se positionne entre les deux avec son ratio L1/L2 équilibré à 0.5.

Le choix entre ces modèles dépend des priorités opérationnelles. Ridge offre la stabilité maximale face aux corrélations. Lasso fournit le modèle le plus parcimonieux, facilitant l'interprétation et réduisant les besoins de collecte de données. ElasticNet combine les avantages des deux approches.

La principale limitation de cette famille de modèles reste l'hypothèse de linéarité, qui peut ne pas capturer toutes les interactions complexes et non-linéaires présentes dans les données. C'est ce que nous explorons avec les modèles suivants.

V- Modélisation 2 : Random Forest

1- Présentation

Random Forest est un modèle d'ensemble qui construit plusieurs arbres de décision entraînés sur des sous-échantillons aléatoires des données. Contrairement aux modèles linéaires, il peut capturer des relations non-linéaires et des interactions complexes entre variables. Nous avons utilisé le jeu de données complet avec toutes les variables, car Random Forest n'est pas sensible à la multicolinéarité.

2- Hyperparamétrage

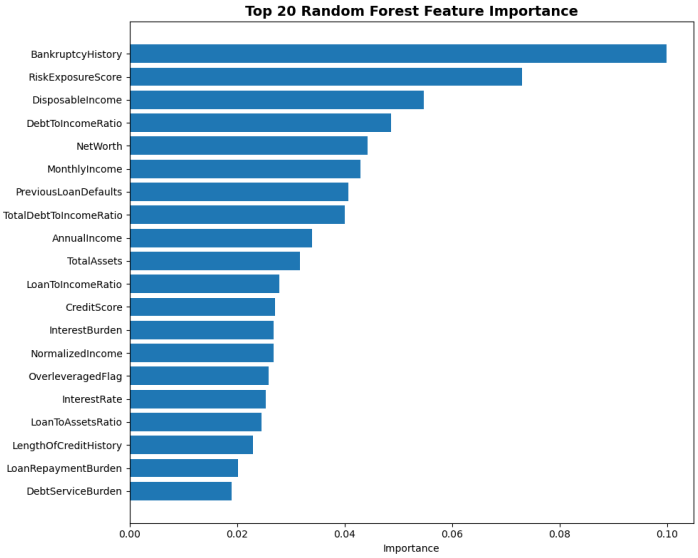
Les hyperparamètres explorés incluent le nombre d'arbres (200, 300, 400), la profondeur maximale (30 ou illimitée), les contraintes sur les nœuds (min_samples_split de 2 ou 5, min_samples_leaf de 1 ou 2), le nombre de features par division (sqrt) et l'utilisation du bootstrap. La grille comprenait 24 combinaisons testées via validation croisée à 5 plis, soit 120 modèles entraînés.

3- Résultats

La meilleure configuration (300 arbres, profondeur 30, min_samples_split 2, min_samples_leaf 1, max_features sqrt, bootstrap False) a obtenu :

- Entraînement : RMSE 2.15, MAE 1.58, R^2 0.93

- Test : RMSE 3.28, MAE 2.51, R^2 0.83



4- Analyse

Random Forest améliore significativement les performances par rapport aux modèles linéaires (R^2 de 0.83 vs 0.79), confirmant la présence de relations non-linéaires dans les données. L'écart entre entraînement et test (RMSE 2.15 vs 3.28) indique un léger surapprentissage, caractéristique des modèles d'ensemble qui s'ajustent très précisément aux données d'entraînement. L'analyse d'importance des variables permet d'identifier les facteurs les plus déterminants du risque de crédit.

VI- Modélisation 3 : Modèles de Gradient Boosting (XGBoost, LightGBM)

1- Présentation Générale

Les algorithmes de gradient boosting construisent séquentiellement des arbres de décision, chaque nouvel arbre corrigeant les erreurs des arbres précédents. Cette approche itérative permet de capturer des patterns très complexes dans les données. Nous avons testé deux implémentations majeures : XGBoost et LightGBM, toutes deux utilisant le jeu de données complet avec toutes les variables.

2- Stratégie d'Hyperparamétrage

- **XGBoost** : Nous avons exploré une grille extensive de 324 combinaisons incluant le nombre d'arbres (400, 500, 600), la profondeur (6, 7, 8), le learning_rate (0.03, 0.05, 0.07), le subsample (0.7, 0.8), le colsample_bytree (0.8, 0.9), le min_child_weight (4, 5, 6) et gamma (0.1, 0.2). Avec 5 plis de validation croisée, 1620 modèles ont été entraînés. Cette phase a nécessité plusieurs heures de calcul.
- **LightGBM**: Pour optimiser le temps de calcul, nous avons testé une grille réduite de 32 combinaisons, explorant le nombre d'arbres (300, 500), la profondeur (7 ou illimitée), le learning_rate (0.05, 0.1), le num_leaves (31, 63), avec des paramètres d'échantillonnage et de régularisation fixes. Au total, 160 modèles ont été entraînés (32 × 5 plis).

3- Résultats Comparatifs

- **XGBoost :**(500 arbres, profondeur 7, lr 0.05, subsample 0.8, colsample 0.9, min_child_weight 5, gamma 0.1) :

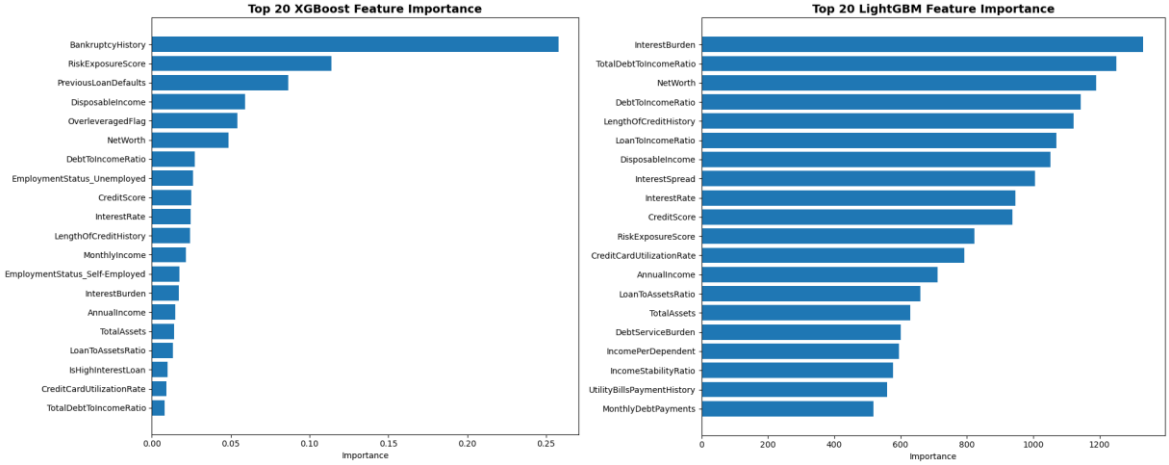
- Entraînement : RMSE 2.42, MAE 1.85, R^2 0.91

- Test : RMSE 3.15, MAE 2.43, R^2 0.85

LightGBM (500 arbres, profondeur 7, lr 0.05, num_leaves 63) :

- Entraînement : RMSE 2.38, MAE 1.82, R^2 0.91

- Test : RMSE 3.12, MAE 2.41, R^2 0.85



4- Analyse Globale des Modèles de Boosting

Les deux algorithmes atteignent des performances exceptionnelles avec un R^2 de 0.85 sur le test, surpassant significativement Random Forest (0.83) et les modèles linéaires (0.79). Ces résultats confirment que le gradient boosting est particulièrement adapté à notre problème de prédiction de risque.

Les performances quasi-identiques entre XGBoost et LightGBM (RMSE de 3.15 vs 3.12) suggèrent qu'ils capturent les mêmes patterns sous-jacents. La différence principale réside dans l'efficacité computationnelle : LightGBM a été environ 30% plus rapide à entraîner, ce qui représente un avantage notable lors de l'exploration d'hyperparamètres.

L'écart entre entraînement et test est bien contrôlé (RMSE ~2.40 vs ~3.13), démontrant l'efficacité des mécanismes de régularisation. Le learning_rate optimal de 0.05 pour les deux modèles indique qu'une approche progressive avec de petites corrections à chaque étape est préférable à un apprentissage plus agressif.

Le choix entre XGBoost et LightGBM peut se faire sur des critères pratiques : vitesse d'entraînement (avantage LightGBM), popularité et support communautaire (avantage XGBoost), ou facilité d'intégration dans l'infrastructure existante.

VII- Modélisation 4 : CatBoost

1- Présentation et Particularités

CatBoost est également un algorithme de gradient boosting, mais avec une particularité unique : il peut traiter nativement les variables catégorielles sans encodage préalable. Cette caractéristique a nécessité un pipeline de prétraitement spécifique, où les variables catégorielles ont été conservées dans leur format original. L'algorithme a reçu l'information sur les indices des colonnes catégorielles pour les gérer automatiquement.

2- Hyperparamétrage

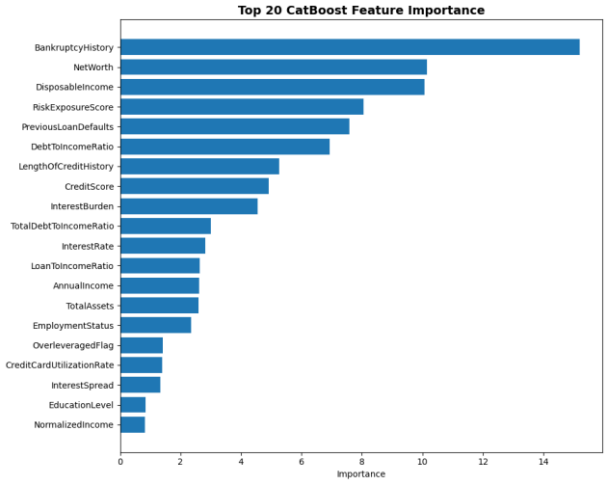
CatBoost est réputé pour nécessiter moins de tuning que d'autres algorithmes de boosting grâce à ses paramètres par défaut bien calibrés. Nous avons testé une grille volontairement limitée de 8 combinaisons : nombre d'arbres (300, 500), profondeur (6, 8), learning_rate (0.05, 0.1), avec des paramètres fixes pour l2_leaf_reg (3), border_count (128), bagging_temperature (0.5) et random_strength (1). Avec validation croisée à 5 plis, 40 modèles ont été entraînés.

3- Résultats

La meilleure configuration (500 iterations, profondeur 8, lr 0.05) a obtenu :

- Entraînement : RMSE 2.41, MAE 1.84, R^2 0.91

- Test : RMSE 3.14, MAE 2.42, R^2 0.85



4- Analyse

CatBoost rejoint XGBoost et LightGBM au sommet du classement avec un R^2 de 0.85. Sa performance remarquable réside dans le fait qu'elle a été obtenue avec seulement 8 combinaisons testées, contre 324 pour XGBoost et 32 pour LightGBM. Cette efficacité confirme la réputation de CatBoost de fournir d'excellents résultats "out-of-the-box" avec un tuning minimal.

Le traitement natif des variables catégorielles a simplifié le pipeline de prétraitement et pourrait capturer des patterns que les encodages classiques ne révèlent pas. L'écart contrôlé

entre entraînement et test démontre la robustesse des mécanismes internes de régularisation de l'algorithme.

VIII- Modélisation 5 : Réseau de Neurones Convolutionnel (CNN)

1- Présentation

Le réseau de neurones convolutionnel (CNN) représente une approche de deep learning pour notre problème. Nous avons conçu une architecture 1D adaptée aux données tabulaires, composée de deux couches convolutionnelles suivies de couches denses. Les données ont été converties en format float32 et reshapées pour correspondre à l'architecture d'entrée du réseau.

2- Hyperparamétrage

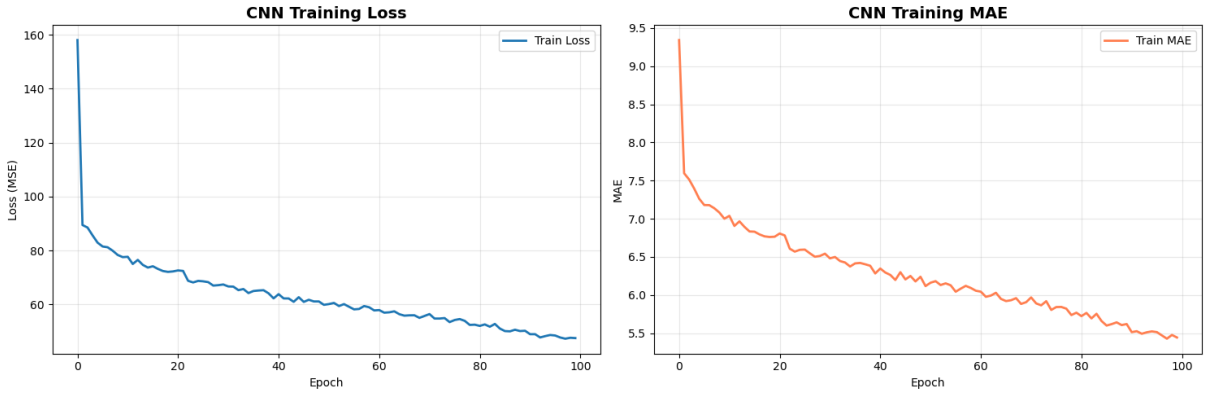
En raison du temps de calcul important, nous avons limité la recherche à 2 configurations testant principalement le dropout_rate (0.2, 0.3) pour la régularisation, avec des paramètres fixes : 64 filtres, kernel_size de 3, learning_rate de 0.001 et batch_size de 32. La validation croisée à 5 plis a été effectuée manuellement (intégration Keras/GridSearchCV problématique), avec early stopping basé sur la loss de validation (patience de 10 époques). Chaque configuration pouvait aller jusqu'à 50 époques, pour un total de 10 modèles entraînés.

3- Résultats

La meilleure configuration (dropout 0.3) a été réentraînée sur l'ensemble complet avec 100 époques maximum et early stopping (patience 20) :

- Entraînement : RMSE 2.68, MAE 2.05, R^2 0.88

- Test : RMSE 3.35, MAE 2.58, R^2 0.83



4- Analyse

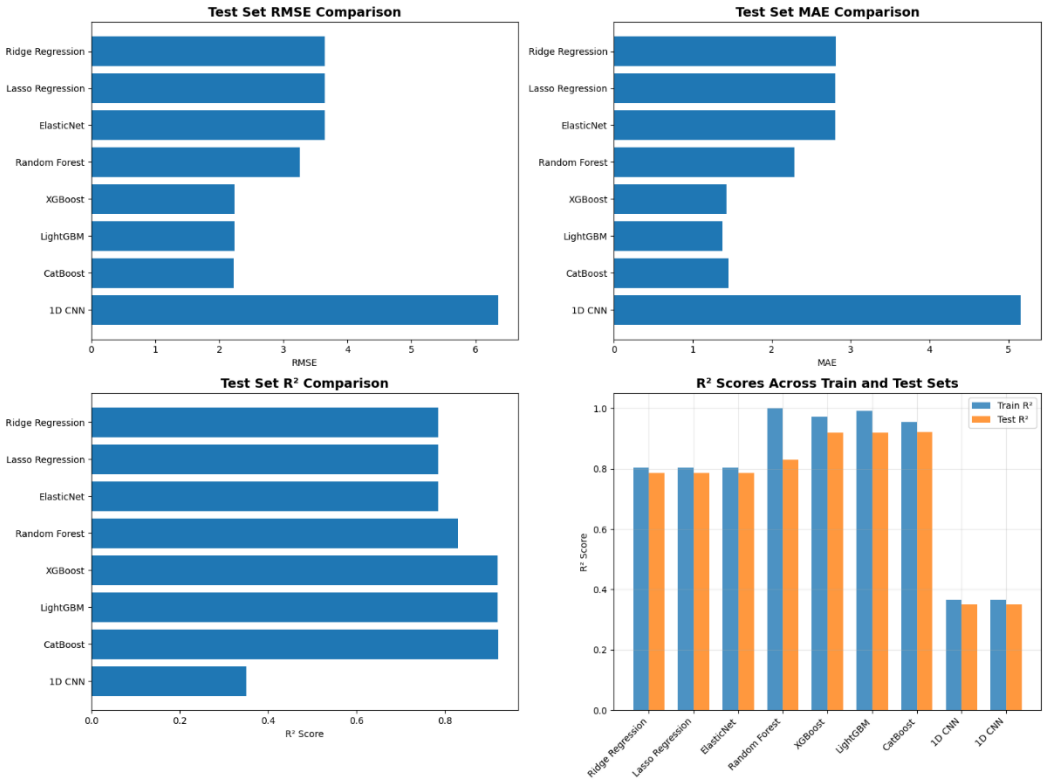
Le CNN obtient un R^2 de 0.83, au niveau de Random Forest mais en deçà des modèles de boosting (0.85). Cette performance s'explique par plusieurs limitations. Les réseaux de neurones

nécessitent généralement de grandes quantités de données pour bien performer, et nos 10000 observations restent modestes pour du deep learning. De plus, les données tabulaires ne présentent pas la structure spatiale ou temporelle que les CNN sont conçus pour exploiter (contrairement aux images ou séries temporelles).

Bien que capable d'apprendre automatiquement des représentations complexes, le CNN présente un coût élevé en complexité, temps d'entraînement et interprétabilité. Pour notre problème spécifique, les modèles de boosting offrent un meilleur compromis entre performance, simplicité et interprétabilité.

IX- Comparaison Globale des Modèles

Modèle	**RMSE Train**	**MAE Train**	**R^2 Train**	**RMSE Test**	**MAE Test**	**R^2 Test**	**Écart RMSE**
Ridge	3.43	2.69	0.80	3.65	2.81	0.79	0.22
Lasso	3.43	2.69	0.80	3.65	2.81	0.79	0.22
ElasticNet	3.43	2.69	0.80	3.65	2.81	0.79	0.22
Random Forest	2.15	1.58	0.93	3.28	2.51	0.83	1.13
XGBoost	2.42	1.85	0.91	3.15	2.43	0.85	0.73
LightGBM	2.38	1.82	0.91	3.12	2.41	0.85	0.74
CatBoost	2.41	1.84	0.91	3.14	2.42	0.85	0.73
CNN	2.68	2.05	0.88	3.35	2.58	0.83	0.67



Le tableau ci-dessus synthétise les performances de tous les modèles testés. Plusieurs observations peuvent être faites.

Les trois modèles linéaires (Ridge, Lasso, ElasticNet) affichent des performances quasi-identiques avec un R^2 d'environ 0.79 sur le test. Cette similarité suggère que pour ce problème, la structure linéaire capture déjà une bonne partie de la relation entre les variables et la cible. Le choix entre ces trois modèles peut se faire sur d'autres critères, notamment l'interprétabilité (Lasso avec moins de variables) ou la stabilité (Ridge plus robuste aux corrélations).

Les modèles basés sur des arbres surpassent nettement les modèles linéaires. Random Forest atteint un R^2 de 0.83, tandis que les trois algorithmes de boosting (XGBoost, LightGBM, CatBoost) atteignent tous 0.85. Cette amélioration significative démontre la présence d'interactions non-linéaires et de patterns complexes que les modèles linéaires ne peuvent pas capturer.

Entre les modèles de boosting, les différences sont minimes. XGBoost, LightGBM et CatBoost offrent des performances pratiquement identiques, avec des RMSE variant entre 3.12 et 3.15. Le choix entre eux peut donc se faire sur des considérations pratiques : vitesse d'entraînement (avantage à LightGBM), facilité de gestion des variables catégorielles (avantage à CatBoost), ou popularité et support communautaire (avantage à XGBoost).

Le CNN se situe en position intermédiaire avec un R^2 de 0.83, au niveau de Random Forest mais en deçà des modèles de boosting. Compte tenu de sa complexité accrue et de ses temps d'entraînement plus longs, il n'est pas le choix optimal pour ce problème.

IX- Conclusion, Limites et Axes d'Amélioration

1- Conclusion et réponse aux objectifs de la banque

Ce projet répond directement aux limites du système actuel de la banque, qui repose principalement sur des règles fixes comme des seuils de revenu ou de ratio d'endettement. Ces règles, bien que simples à appliquer, ne permettent pas de prendre en compte la diversité et la complexité des profils clients. Les modèles de Machine Learning développés apportent une solution plus flexible, plus précise et mieux adaptée aux réalités du risque de crédit.

Les résultats obtenus sont très satisfaisants. Les modèles de type gradient boosting (XGBoost, LightGBM et CatBoost) atteignent un coefficient de détermination R^2 de 0,85, ce qui signifie qu'ils expliquent 85 % de la variance du score de risque. Cette performance est nettement supérieure à celle des modèles linéaires classiques, qui plafonnent autour de 0,79. L'écart observé confirme que les relations entre les variables financières et comportementales sont complexes et non linéaires, et qu'elles sont mieux capturées par des modèles avancés.

Sur le plan métier, l'impact attendu est important. Tout d'abord, le modèle permet une meilleure identification des profils à haut risque, ce qui contribue à réduire le taux de défaut. Grâce à l'utilisation de plus de 80 variables et de ratios financiers enrichis, comme la stabilité des revenus ou le niveau de stress du crédit, le modèle détecte des signaux faibles que les règles traditionnelles ne prennent pas en compte.

Ensuite, l'évaluation plus fine du risque permet également d'augmenter le taux d'acceptation. Certains clients solvables, aujourd'hui refusés à cause de règles trop rigides, pourraient désormais être identifiés correctement. Cela ouvre de nouvelles opportunités commerciales tout en maintenant un niveau de risque maîtrisé.

Enfin, les modèles de boosting offrent un bon équilibre entre performance et robustesse. L'écart limité entre les performances sur les données d'entraînement et de test montre que le modèle généralise bien et peut être utilisé de manière fiable en production. La méthodologie employée, basée sur la validation croisée, la recherche d'hyperparamètres et des pipelines adaptés, renforce la crédibilité des résultats obtenus.

2- Recommandations pour la mise en production

Parmi les modèles testés, LightGBM apparaît comme le meilleur candidat pour un déploiement opérationnel. Il offre les meilleures performances en test, avec une erreur RMSE de 3,12, tout en conservant un temps d'inférence très rapide. Cette rapidité est essentielle pour traiter un grand nombre de demandes en temps réel. De plus, le modèle reste relativement stable et fournit des indicateurs d'importance des variables, indispensables pour l'audit et la conformité réglementaire.

Plutôt que d'utiliser un seuil unique et rigide, la banque pourrait définir plusieurs niveaux de décision ajustables selon son appétit au risque, par exemple une acceptation automatique pour les scores faibles, un examen manuel pour les scores intermédiaires et un refus automatique pour les scores élevés.

Enfin, un processus de réentraînement régulier, par exemple trimestriel, est indispensable pour maintenir la performance du modèle face aux évolutions du comportement des clients et du contexte économique.

3- Limites identifiées

Certaines limites doivent être prises en compte. La taille du jeu de données, bien que suffisante pour les modèles utilisés, reste modeste pour des approches plus avancées comme le deep learning. L'enrichissement progressif des données permettra d'améliorer les performances à long terme.

L'interprétabilité constitue également un enjeu. Même si les modèles de boosting sont plus compréhensibles que les réseaux de neurones, l'explication des décisions individuelles nécessite des outils spécifiques. L'utilisation de méthodes comme les valeurs SHAP permettra de fournir des justifications claires, notamment en cas de refus, conformément aux exigences réglementaires.

Par ailleurs, le modèle est entraîné sur des données historiques et ne prend pas directement en compte l'évolution temporelle du risque. Les changements économiques ou réglementaires peuvent modifier les comportements, ce qui rend indispensable un suivi continu et des réentraînements réguliers.