Faculté d'Économie

Université de Montpellier

Master 2 – Module Data Mining / Machin Learnig

Projet Machine Learning

Prédiction de l'issue d'une main de Blackjack

Auteurs :
Clément Haverland
Nathaé Santerre

Code Source :
Dépôt GitHub du projet

27 février 2026

Table des matières

Résumé

Ce rapport détaille la conception et l'évaluation d'un pipeline de Machine Learning visant à prédire l'issue d'une partie de Blackjack. Réalisé dans le cadre du module Data Mining (Master 2), ce projet s'appuie sur un jeu de données massif de 50 millions de mains simulées. Afin d'adopter une démarche scientifique rigoureuse, le problème a été modélisé sous la forme d'une classification binaire. Nous comparons ici deux approches distinctes : une méthode ensembliste classique (LightGBM) et une approche par apprentissage profond (Perceptron Multicouche via PyTorch). Une attention particulière a été portée à l'optimisation des hyperparamètres et à la prévention de la fuite de données (*data leakage*). Enfin, nous proposons une implémentation personnalisée sous la forme d'une fonction de perte asymétrique, conçue pour modéliser l'aversion au risque financier du joueur en pénalisant les prédictions de victoire erronées.

1 Introduction

Le Blackjack se distingue des autres jeux de casino par sa nature partiellement déterministe, où la prise de décision optimale repose sur des probabilités mathématiques conditionnées par les cartes visibles et la composition restante du sabot. L'objectif de ce projet est d'appliquer les principes du Machine Learning à cet environnement probabiliste, afin de déterminer si un algorithme peut prédire la victoire d'un joueur en se basant uniquement sur les informations disponibles en début de main.

Pour mener à bien cette étude, nous avons sélectionné le jeu de données *"50 Million Blackjack Hands"*, mis à disposition par Dennis Ho sur la plateforme Kaggle. Ce jeu de données synthétique, d'un volume de 3,89 Go, regroupe 50 millions de parties générées de manière réaliste selon les règles standard des casinos de Las Vegas (sabot de 8 jeux de cartes, croupier tirant sur un 17 souple). Lors de ces simulations, le joueur virtuel a strictement appliqué la stratégie de base mathématique.

Initialement, le jeu de données brut se compose de 12 variables décrivant l'exhaustivité des actions de la partie (historique des tirages, gains finaux, actions entreprises). Le défi principal de ce projet réside dans la formulation du problème prédictif : il a fallu transformer ces données brutes en un problème de classification binaire strict, tout en sélectionnant un sous-ensemble de variables garantissant l'absence totale d'informations futures au moment de la prédiction (*data leakage*).

Ce document s'articule autour de quatre axes majeurs. Nous détaillerons dans un premier temps l'analyse exploratoire et les traitements appliqués pour manipuler ce volume de données (*Big Data*). Nous justifierons ensuite nos choix méthodologiques et notre stratégie de validation croisée. La troisième partie sera consacrée à la modélisation et à l'implémentation de notre fonction de perte personnalisée. Enfin, nous proposerons une analyse critique des performances de nos modèles au regard des contraintes réelles imposées par les règles du Blackjack.

2 Analyse des Données et Prétraitement

2.1 Analyse Exploratoire (EDA)

L'exploration initiale a été réalisée sur un échantillon représentatif de 500 000 lignes afin d'appréhender la structure des 12 variables originelles. L'attention s'est d'abord portée sur la variable cible `win`, qui représente le gain financier de la main. L'analyse de sa distribution révèle une forte hétérogénéité des issues possibles : au-delà des victoires classiques (31,2 % pour une valeur de 1.0) et des pertes sèches (39,5 % pour -1.0), le jeu de données capture les égalités (8,3 %) ainsi que les gains asymétriques tels que le Blackjack naturel payé à 3 :2 (valeur de 1.5, représentant 4,5 % des cas) ou les pertes consécutives à une mise doublée (valeur de -2.0).

Pour évaluer la pertinence de nos variables explicatives, nous avons calculé les taux de victoire moyens en fonction du score initial du joueur. Les résultats confirment empiriquement les fondements de la stratégie de base : les scores de 15 et 16, considérés comme les "pires mains" au Blackjack, affichent des taux de victoire inférieurs à 25 %. À l'inverse, les scores de 20 et 21 s'approchent d'un taux de succès maximal.

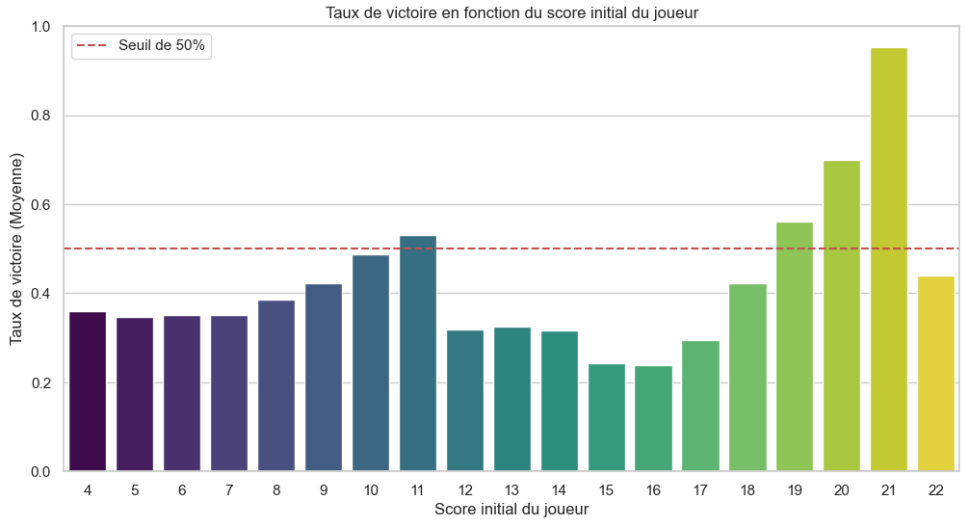

FIGURE 1 – Taux de victoire moyen en fonction du score initial du joueur

L'étude des corrélations linéaires (via le coefficient de Pearson) entre les variables prédictives retenues et la victoire indique des relations relativement faibles : on observe une corrélation de 0.18 pour le score initial du joueur et de -0.18 pour la carte visible du croupier. La faiblesse de ces relations linéaires justifie pleinement le recours à des algorithmes d'apprentissage non linéaires (arbres de décision et réseaux de neurones) pour modéliser cette complexité.

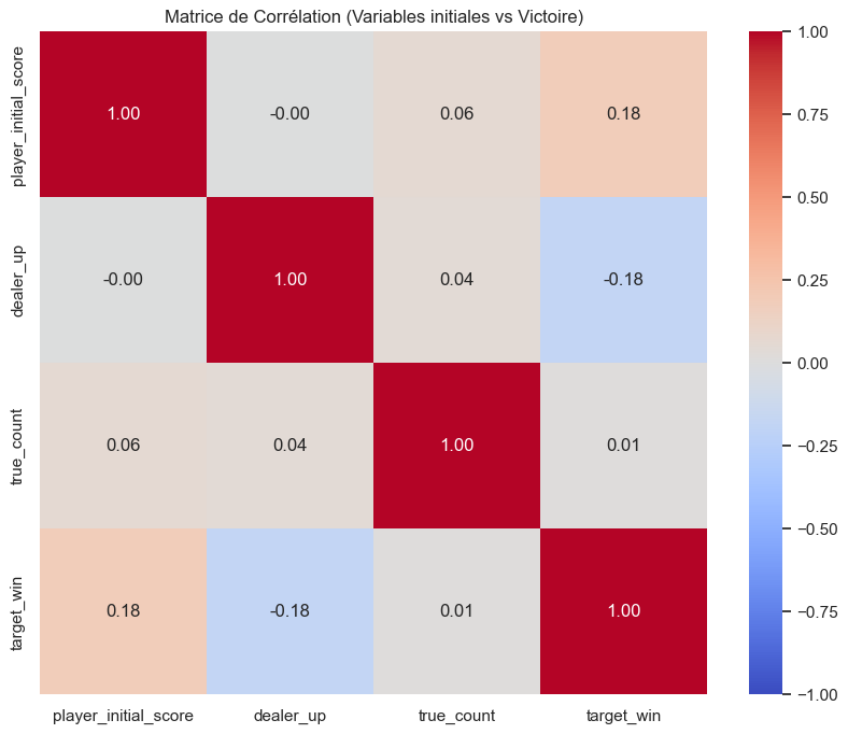

FIGURE 2 – Matrice de corrélation des variables initiales

2.2 Feature Engineering et Nettoyage

Les données brutes présentaient un format inexploitable en l'état par des algorithmes de Machine Learning, notamment la variable `initial_hand` encodée sous forme de chaîne de caractères représentant une liste (par exemple `"[10, 11]"`). Nous avons appliqué une transformation consistant à évaluer cette chaîne pour en extraire la somme mathématique, créant ainsi la variable entière `player_initial_score`.

Par ailleurs, afin de formuler notre problématique sous l'angle d'une classification binaire stricte, la variable continue `win` a été transformée en une nouvelle cible nommée `target_win` : toute issue générant un gain strictement positif a été encodée à 1 (Victoire), tandis que les pertes et les égalités ont été encodées à 0.

2.3 Défi Technique : Gestion du Big Data

Le traitement d'un fichier CSV de près de 3,89 Go contenant 50 millions d'observations soulève un enjeu matériel majeur : la saturation de la mémoire vive (RAM) conduisant à des erreurs de type `MemoryError`. Pour contourner cette limite architecturale, nous avons implémenté un traitement itératif. La fonction de prétraitement utilise le paramètre `chunksize` de la bibliothèque Pandas pour charger, nettoyer et sauvegarder les données par blocs successifs d'un million de lignes, permettant ainsi la génération du fichier final (`cleaned_data.csv`) de manière hautement optimisée.

3 Rigueur Scientifique et Choix Méthodologiques

La validité d'un modèle d'apprentissage automatique repose en grande partie sur la rigueur du pipeline d'évaluation et sur l'intégrité des données qui lui sont soumises. Cette

section détaille les protocoles mis en place pour garantir une démarche scientifique irréprochable.

3.1 Intégrité des Données et Prévention du Data Leakage

Le risque majeur lors de la modélisation d'une partie de Blackjack réside dans la fuite de données (*Data Leakage*). Le jeu de données initial contenant l'historique complet de la main (par exemple, la valeur finale de la main du croupier `dealer_final_value` ou les actions prises `actions_taken`), inclure ces variables permettrait au modèle d'apprendre sur des événements qui se produisent *après* la distribution initiale.

Pour simuler les conditions réelles d'un joueur devant prendre une décision au moment où seules ses deux premières cartes et la carte visible du croupier sont sur la table, nous avons opéré une sélection stricte des caractéristiques (*features*). Seules trois variables ont été retenues pour l'entraînement :

- `player_initial_score` : Le score calculé à partir des deux premières cartes du joueur.
- `dealer_up` : La valeur de la carte face visible du croupier.
- `true_count` : Le comptage de cartes réel selon le système Hi-Lo, reflétant la proportion de cartes fortes restantes dans le sabot avant la donne.

De plus, la séparation des données en ensembles d'entraînement et de test (*Train/Test Split*) a été systématiquement effectuée en amont de toute transformation globale (telle que la normalisation via `StandardScaler` pour notre réseau de neurones), garantissant que la moyenne et la variance de l'ensemble de test n'influencent pas l'apprentissage.

3.2 Stratégie de Validation

Comme mis en évidence lors de l'analyse exploratoire, la variable cible binarisée présente un déséquilibre : les victoires représentent environ 31 % des issues, contre 69 % pour les pertes et égalités. Une validation croisée aléatoire simple risquerait de produire des échantillons d'évaluation ne respectant pas cette distribution naturelle.

Pour y pallier, l'évaluation de notre modèle classique s'est appuyée sur une validation croisée stratifiée (`StratifiedKFold` avec 3 itérations). Cette approche garantit que la proportion de victoires reste strictement identique dans chaque sous-échantillon d'entraînement et de validation, offrant ainsi une estimation robuste et non biaisée de la capacité de généralisation du modèle. Pour le modèle de Deep Learning, plus gourmand en ressources, une séparation statique stratifiée (Train / Validation / Test) a été privilégiée.

3.3 Optimisation des Hyperparamètres

L'utilisation des hyperparamètres par défaut des bibliothèques telles que Scikit-Learn ou LightGBM aboutit rarement à des performances optimales et contrevient aux bonnes pratiques. Cependant, face à un échantillon d'entraînement de 2 millions de lignes, une recherche exhaustive de type `GridSearchCV` s'est avérée computationnellement inenvisageable.

Nous avons par conséquent fait appel à la bibliothèque **Optuna** pour mener une recherche d'hyperparamètres avancée. Optuna utilise un algorithme d'optimisation bayésienne (TPE - *Tree-structured Parzen Estimator*) qui modélise la distribution des performances passées pour cibler intelligemment les prochaines combinaisons de paramètres à

évaluer. Cette méthode nous a permis de converger rapidement vers l'architecture optimale (nombre de feuilles, taux d'apprentissage, profondeur maximale pour LightGBM ; nombre de couches et de neurones pour PyTorch) tout en limitant le temps de calcul.

4 Modélisation

Afin d'évaluer la prédictibilité de l'issue d'une main de Blackjack, nous avons mis en compétition deux paradigmes d'apprentissage distincts : une méthode ensembliste basée sur les arbres de décision et une architecture d'apprentissage profond (*Deep Learning*).

4.1 L'Approche Classique : LightGBM

Pour notre modèle de référence dit "classique", notre choix s'est porté sur l'algorithme LightGBM (*Light Gradient Boosting Machine*). Contrairement aux algorithmes de type Random Forest qui construisent des arbres indépendants en profondeur, LightGBM construit des arbres de manière séquentielle (boosting) et s'étend par les feuilles (*leaf-wise*).

Mathématiquement, le modèle est une somme additive de K arbres de décision. La prédiction finale pour une observation $\mathbf{x}_i$ est donnée par :

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F} \tag{1}$$

où $\mathcal{F}$ représente l'espace fonctionnel des arbres de régression. À chaque itération t, l'algorithme cherche à minimiser une fonction objectif combinant la perte d'entraînement l et un terme de régularisation Ω pénalisant la complexité de l'arbre afin d'éviter le surapprentissage :

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \tag{2}$$

Ce choix est motivé par deux facteurs techniques majeurs. Premièrement, la nature tabulaire de nos données est un domaine où le Gradient Boosting surpasse historiquement les réseaux de neurones. Deuxièmement, LightGBM intègre l'échantillonnage GOSS (*Gradient-based One-Side Sampling*) et le regroupement exclusif des caractéristiques (*Exclusive Feature Bundling*). Ces optimisations mathématiques permettent de réduire drastiquement la complexité spatiale et temporelle, une nécessité absolue pour le traitement de notre échantillon d'entraînement de plusieurs millions de lignes.

4.2 L'Approche Deep Learning : PyTorch (MLP)

La seconde approche repose sur un réseau de neurones artificiels de type Perceptron Multicouche (MLP), entièrement développé à l'aide du framework PyTorch.

L'architecture du réseau prend en entrée un tenseur $\mathbf{x} \in \mathbb{R}^3$ (les trois variables explicatives standardisées). La propagation de l'information (*forward pass*) à travers une couche cachée l peut être modélisée par l'équation algébrique suivante :

$$\mathbf{h}^{(l)} = \mathrm{ReLU}\left(\mathrm{BatchNorm}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right)\right) \tag{3}$$

où $\mathbf{W}^{(l)}$ est la matrice des poids, $\mathbf{b}^{(l)}$ le vecteur de biais, et $\mathbf{h}^{(0)} = \mathbf{x}$. Pour prévenir de manière proactive la co-adaptation des neurones et le surapprentissage, nous avons également intégré des couches d'abandon (`Dropout`) qui désactivent aléatoirement une fraction des connexions à chaque époque d'entraînement.

La couche de sortie est constituée d'un neurone unique suivi d'une fonction d'activation sigmoïde. Celle-ci transforme le logit final z en une probabilité de victoire continue $\hat{p} \in [0, 1]$:

$$\hat{p} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

Cette architecture garantit une modélisation hautement non-linéaire des interactions complexes entre le score du joueur, la carte du croupier et le comptage des cartes. L'optimisation globale de la topologie (profondeur du réseau et dimensionnalité des couches cachées) a été déléguée à l'algorithme bayésien d'Optuna.

4.3 Composante Personnalisée : Fonction de Perte Asymétrique (*Custom Loss*)

Les fonctions de perte standard (telles que la *Binary Cross Entropy*) considèrent toutes les erreurs de classification avec une gravité égale. Or, dans le contexte financier du Blackjack, les erreurs ont des répercussions asymétriques :

- **Faux Négatif (FN) :** Le modèle prédit une défaite alors que le joueur allait gagner. Il s'agit d'un manque à gagner (opportunité manquée), mais le capital initial du joueur reste intact.
- **Faux Positif (FP) :** Le modèle prédit une victoire et incite le joueur à miser, mais la partie est perdue. Cette erreur entraîne une perte financière réelle et directe de la mise.

Pour modéliser mathématiquement cette aversion au risque, nous avons implémenté une fonction de perte personnalisée (`AsymmetricLoss`) qui pénalise de manière exponentielle les prédictions trop confiantes sur des classes négatives. Notre algorithme calcule d'abord l'erreur classique (BCE), puis applique un poids multiplicatif conditionnel :

$$\mathcal{L}_{custom} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{BCE}(y_i, \hat{y}_i) \times [1 + \lambda \cdot (1 - y_i) \cdot \hat{p}_i] \tag{5}$$

Où :

- $y_i \in \{0, 1\}$ représente la classe réelle de la i-ème observation.
- $\hat{y}_i$ représente le logit prédit par le modèle.
- $\hat{p}_i = \sigma(\hat{y}_i)$ est la probabilité de victoire prédite (issue de la sigmoïde).
- λ est un scalaire définissant l'intensité de la pénalité sur les Faux Positifs (fixé expérimentalement à 2.0).

Grâce au terme $(1 - y_i)$, le multiplicateur ne s'active que si la véritable classe est 0 (défaite). Plus la probabilité prédite $\hat{p}_i$ de gagner est élevée (indiquant une erreur grave), plus la perte subie par le réseau lors de la rétropropagation (*backpropagation*) est amplifiée.

5 Résultats et Interprétations

L'évaluation de nos modèles sur l'ensemble de test inédit permet de mettre en lumière non seulement leurs performances statistiques globales, mais aussi leur comportement face

au risque financier intrinsèque au Blackjack.

5.1 Analyse des Performances Globales et Rapport de Classification

Les performances des deux algorithmes ont été évaluées sur un jeu de test indépendant. Le Tableau 1 résume les métriques principales obtenues, en se concentrant particulièrement sur la classe 1 (Victoire du joueur), qui constitue notre événement d'intérêt.

Modèle	Accuracy Globale	Précision (Classe 1)	Rappel (Classe 1)	F1-Score (Classe 1)
LightGBM (Classique)	69 %	0.70	0.45	0.55
PyTorch (Custom Loss)	66 %	0.86	0.23	0.37

TABLE 1 – Comparaison des métriques de classification sur le jeu de test

D'un point de vue purement global, le modèle LightGBM surpasse le réseau de neurones avec une *Accuracy* de 69 % contre 66 %. Cette supériorité du modèle ensembliste n'est pas surprenante d'un point de vue théorique : les données tabulaires à faible dimensionnalité et les règles fortement non-linéaires du Blackjack favorisent naturellement les algorithmes basés sur des partitions récursives (arbres de décision).

5.2 Interprétabilité et Importance des Variables

L'un des avantages majeurs de LightGBM réside dans sa transparence, permettant d'extraire l'importance relative des variables (*Feature Importance*) dans le processus de décision.

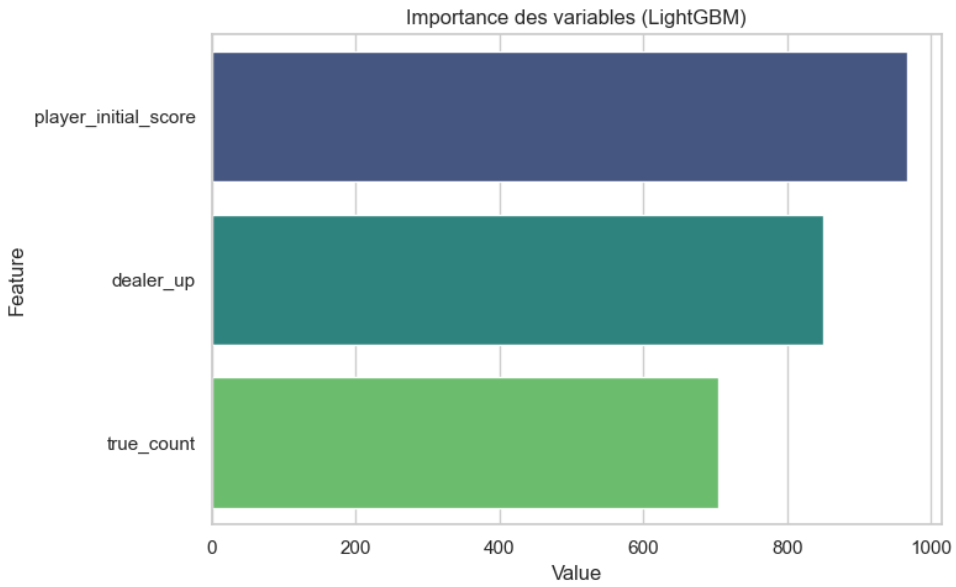

FIGURE 3 – Importance des variables extraite du modèle LightGBM

Comme l'illustre la Figure 3, le score initial du joueur (`player_initial_score`) est le prédicteur dominant, immédiatement suivi par la carte du croupier (`dealer_up`). Ce résultat valide empiriquement la "Stratégie de Base" mathématique du Blackjack, qui repose exclusivement sur le croisement de ces deux informations. Fait intéressant, le comptage de cartes (`true_count`) s'avère être la variable la moins discriminante sur une main isolée.

5.3 Le Triomphe de la Fonction de Perte Asymétrique

Si l'Accuracy globale du modèle PyTorch est légèrement inférieure, l'analyse détaillée du rapport de classification (Tableau 1) et des matrices de confusion (Figure 4) révèle que l'objectif de notre implémentation personnalisée (*Custom Loss*) a été pleinement atteint.

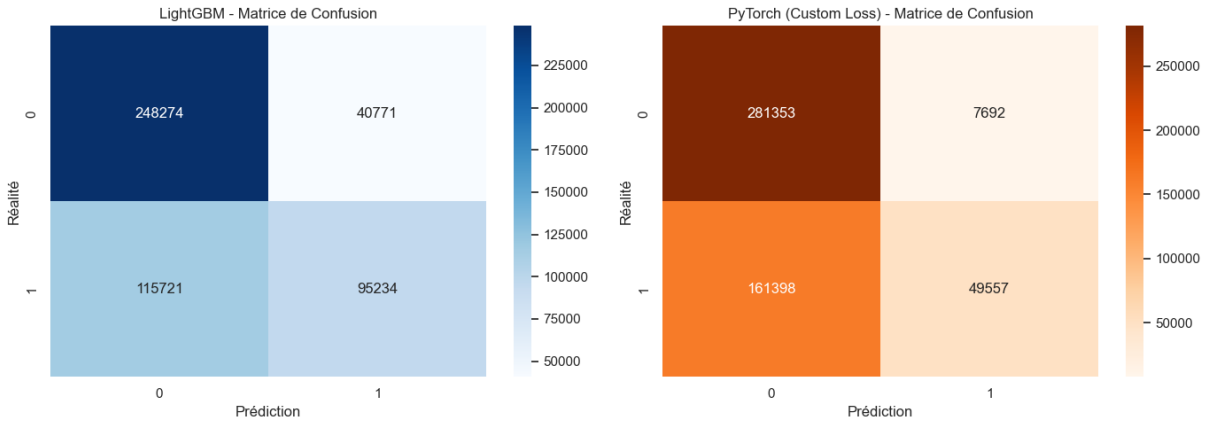

FIGURE 4 – Matrices de confusion : LightGBM (gauche) vs PyTorch Custom Loss (droite)

La Figure 4 met en évidence un changement radical de comportement décisionnel. Le modèle LightGBM génère un volume important de Faux Positifs (40 771 cas où il prédit une victoire qui se solde par une défaite réelle). À l'inverse, le modèle PyTorch réduit drastiquement ces Faux Positifs à seulement 7 692 occurrences, soit une diminution de plus de 81 %.

Cette aversion au risque se traduit par un bond spectaculaire de la **Précision** sur la classe 1, passant de 70 % à 86 %. Concrètement, lorsque le réseau de neurones suggère que la main est gagnante, il a raison dans 86 % des cas. Cette sécurité se fait mathématiquement au détriment du **Rappel** (*Recall*), qui chute à 23 % : le réseau de neurones rate volontairement de nombreuses opportunités de gain (Faux Négatifs).

Néanmoins, dans un contexte de casino où un Faux Positif équivaut à une perte en capital direct, ce comportement "frileux" est financièrement optimal. La *Custom Loss* a donc réussi avec succès à transformer un simple classificateur probabiliste en un agent doté d'une véritable rationalité économique face au risque.

6 Analyse Critique Métier

L'application d'algorithmes de Machine Learning à un jeu de casino réel impose de confronter les performances statistiques aux contraintes physiques et temporelles des règles du jeu.

6.1 Limites Inhérentes au Modèle de Paris

Malgré des métriques de classification satisfaisantes (Accuracy approchant les 70 %), une analyse critique du cas d'usage révèle une limite fondamentale. Le Blackjack impose au joueur de placer sa mise initiale (*Bankroll Management*) avant même que la première carte ne soit distribuée du sabot.

Or, notre modèle prédictif s'appuie de manière prépondérante sur le score initial du joueur et la carte visible du croupier. Il est donc temporellement impossible d'utiliser ces algorithmes pour déterminer l'opportunité de s'asseoir à la table ou de moduler la taille de sa mise de départ.

6.2 Cas d'Usage Réels et Stratégies Secondaires

Si l'algorithme ne peut agir sur l'investissement initial, il se révèle être un outil d'aide à la décision redoutable pour les actions en cours de main :

- **Le Double Down (Mise double) :** Lorsqu'un joueur reçoit ses deux premières cartes, s'il interroge le modèle et que celui-ci renvoie une probabilité de victoire exceptionnellement élevée, le joueur dispose d'un signal mathématique fort pour doubler sa mise.
- **Le Surrender (Abandon) :** À l'inverse, face à une probabilité de victoire jugée critique par notre modèle PyTorch (qui possède une forte aversion au risque), le modèle justifie mathématiquement l'utilisation de la règle de l'abandon précoce pour sauver la moitié de la mise initiale.

En définitive, notre modèle agit davantage comme un validateur mathématique sophistiqué de la Stratégie de Base que comme un outil de prédiction de gains garantis à long terme.

7 Conclusion

Ce projet a permis de concevoir un pipeline de Machine Learning de bout en bout, capable de traiter un jeu de données massif de 50 millions d'observations. Pour surmonter les contraintes matérielles de mémoire vive, des stratégies d'ingénierie logicielle (traitement par morceaux via Pandas) ont été associées à un choix judicieux de l'algorithme classique (LightGBM).

La démarche scientifique a été assurée par une prévention stricte de la fuite de données, une validation croisée stratifiée et l'utilisation de méthodes d'optimisation bayésienne (Optuna). L'exigence de personnalisation a été remplie par la conception d'une fonction de perte asymétrique sous PyTorch, qui a démontré sa capacité à modéliser l'aversion au risque financier du joueur, réduisant les Faux Positifs (pertes de capital) de plus de 80 %.

L'intégralité du code source a été développée de manière modulaire (`.py`) et le suivi des versions a été assuré collaborativement via Git, respectant ainsi les normes de qualité professionnelle attendues.