Partie 4 : De la fouille de donnée à l'AutoML

Shaina Boutebba

December 2025

1 Introduction

Après des études manuelles effectuées dans les parties précédentes pour la prédiction et le clustering, nous avons à présent automatisé la recherche du meilleur modèle. L'objectif est de rechercher les meilleurs modèles et hyperparamètres pour la prédiction et le clustering. Dans un premier temps, nous avons utilisé le module **TPOT** de *Python* pour rechercher le meilleur modèle pour la classification de la cible **DValue**. Ensuite, nous avons utilisé la bibliothèque **hyperopt** de *Python* pour optimiser les hyperparamètres de l'algorithme **K-means** pour le clustering.

2 AutoML pour la Prédiction

2.1 Données utilisées

Pour cette phase d'automatisation de la prédiction, nous avons réutilisé les données prétraitées dans la partie 3. En effet, ces données ont déjà subi l'ensemble des transformations nécessaires : nettoyage, traitement des valeurs manquantes, et encodage des variables catégorielles. Cela nous permet de nous concentrer directement sur l'optimisation des modèles sans avoir à répéter les étapes de préparation.

La variable cible de notre prédiction est **value_cat**, une variable catégorielle qui représente différents niveaux de la valeur marchande des joueurs. Comme le montre la figure 1 de répartition des classes, cette variable présente une distribution uniforme entre ses différentes catégories (low, very low, medium, high, elite). Cette répartition équilibrée est un atout considérable pour notre tâche de classification, car elle évite les problèmes liés aux classes déséquilibrées qui peuvent biaiser les modèles vers les classes majoritaires.

Grâce à cette distribution uniforme, nous avons pu choisir des métriques d'évaluation simples et directes pour mesurer la performance de nos modèles. Nous avons opté pour l'**accuracy** (précision globale) comme métrique principale, car elle reflète fidèlement la qualité de la prédiction lorsque les classes sont équilibrées. En complément, nous utilisons également le score **ROC AUC** (Area Under the Receiver Operating Characteristic Curve), qui permet d'évaluer

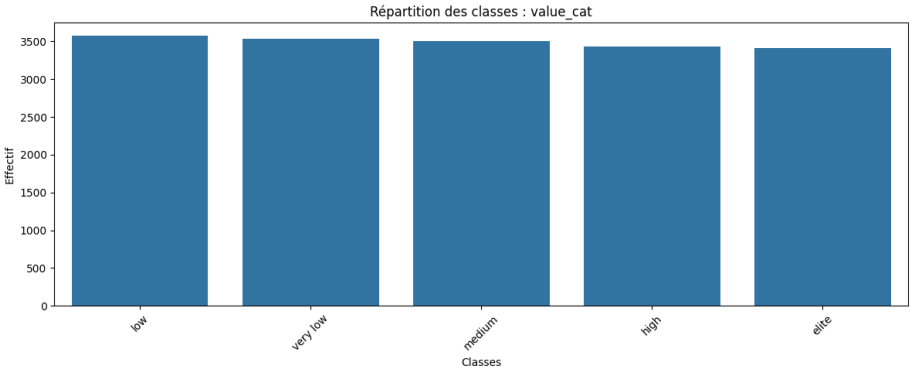

Figure 1: Répartition des classes

la capacité du modèle à discriminer entre les différentes classes, indépendamment du seuil de classification choisi.

2.2 Définition de l'espace de recherche

L'espace de recherche définit l'ensemble des algorithmes et des hyperparamètres que TPOT va explorer pour identifier la meilleure configuration. Compte tenu des contraintes de ressources computationnelles, l'exécution se faisant sur une machine personnelle avec des capacités limitées, nous avons dû restreindre significativement cet espace de recherche.

Nous nous sommes limités à deux algorithmes de classification reconnus pour leur efficacité :

- **XGBoost** : un algorithme de boosting par gradient très performant pour les tâches de classification. Les hyperparamètres explorés sont :
  - `max_depth` : profondeur maximale des arbres, variant entre 2 et 15
  - `n_estimators` : nombre d'arbres dans l'ensemble, entre 500 et 1000
  - `eta` : taux d'apprentissage, entre 0.03 et 0.3
- **RandomForest** : un algorithme d'ensemble basé sur des forêts d'arbres de décision. Les hyperparamètres explorés sont :
  - `n_estimators` : nombre d'arbres dans la forêt, entre 300 et 1000
  - `max_depth` : profondeur maximale des arbres, entre 2 et 15
  - `criterion` : critère de division des nœuds (gini, entropy, log_loss)

Il est important de noter que cette limitation à deux algorithmes et à un nombre restreint d'hyperparamètres est une contrainte imposée par les ressources disponibles. Idéalement, un espace de recherche plus large incluant d'autres algorithmes (SVM, réseaux de neurones, etc.) et davantage de valeurs pour chaque

hyperparamètre permettrait potentiellement d'obtenir de meilleurs résultats. Cependant, cela nécessiterait des ressources computationnelles considérablement plus importantes et un temps d'exécution beaucoup plus long, ce qui n'était pas envisageable dans notre contexte.

2.3 Configuration de TPOT

Pour l'optimisation automatisée, nous avons configuré TPOT avec les paramètres suivants :

- **generations=10** : TPOT évoluera les pipelines sur 10 générations successives.
- **population_size=15** : À chaque génération, une population de 15 individus (pipelines) sera maintenue. Cette taille de population relativement modeste est un compromis entre l'exploration de l'espace de recherche et le temps de calcul.
- **cv=5** : Validation croisée à 5 plis pour évaluer chaque pipeline. Cela garantit une estimation robuste des performances en testant chaque modèle sur différentes partitions des données.
- **scorers=['accuracy', 'roc_auc_ovr']** : Les deux métriques définies précédemment sont utilisées pour évaluer les pipelines, avec une pondération égale (1/2, 1/2).
- **max_time_mins=60** : L'exécution est limitée à 1 heure maximum. Cette contrainte temporelle stricte est imposée par nos limitations en ressources computationnelles.
- **random_state=42** : Graine aléatoire fixée pour assurer la reproductibilité des résultats.

Cette configuration représente un équilibre entre exhaustivité de la recherche et contraintes pratiques. Avec 10 générations et une population de 15 individus, TPOT évaluera au maximum 150 pipelines différents (en réalité moins, car certains pipelines peuvent être réutilisés entre générations). Le temps d'exécution observé a été d'environ 60 minutes, atteignant la limite temporelle fixée.

2.4 Résultats de l'optimisation

2.4.1 Meilleur modèle identifié

À l'issue du processus d'optimisation, TPOT a identifié un modèle **XGBClassifier** comme étant le plus performant. Les hyperparamètres optimaux découverts sont :

- `n_estimators=577` : nombre d'arbres de décision

- `max_depth=4` : profondeur maximale des arbres
- `eta=0.1046981249386` : taux d'apprentissage (learning rate)

Il est intéressant de noter que TPOT a convergé vers une valeur de `max_depth` relativement faible (4), ce qui suggère que des arbres peu profonds sont suffisants pour capturer les relations dans nos données, tout en évitant le surapprentissage. Le taux d'apprentissage modéré (environ 0.10) permet une convergence stable du modèle.

2.4.2 Performances sur l'ensemble de test

Le meilleur modèle a obtenu les performances suivantes sur l'ensemble de test :

- **Accuracy** : 0.9571 (95.71%)
- **ROC AUC Score (OvR)** : 0.9982 (99.82%)

Ces résultats indiquent une très bonne capacité de prédiction globale. L'accuracy de 95.71% signifie que le modèle classe correctement environ 96% des joueurs dans la bonne catégorie de valeur. Le score ROC AUC quasi-parfait (99.82%) démontre que le modèle possède une excellente capacité à discriminer entre les différentes classes.

2.4.3 Analyse du rapport de classification

Le rapport de classification détaillé par classe révèle les performances suivantes :

Classe	Precision	Recall	F1-Score	Support
0 (elite)	0.97	0.98	0.98	1103
1 (very low)	0.98	0.97	0.98	1147
2 (high)	0.94	0.95	0.95	1153
3 (medium)	0.94	0.93	0.93	1165
4 (low)	0.95	0.95	0.95	1195

L'analyse de ces métriques par classe révèle plusieurs observations importantes :

- Les classes 0 (elite) et 1 (very low) présentent les meilleures performances avec des F1-scores de 0.98, suggérant que le modèle distingue particulièrement bien les joueurs aux extrêmes du spectre de valeur : les joueurs d'élite et ceux de très faible valeur.
- Les classes intermédiaires (2-high, 3-medium, 4-low) affichent des performances légèrement inférieures mais toujours excellentes (F1-scores entre 0.93 et 0.95), indiquant une difficulté modérée à différencier les nuances entre les différents niveaux de joueurs moyens à bons.

- La précision (precision) et le rappel (recall) sont bien équilibrés pour toutes les classes, sans biais significatif vers les faux positifs ou les faux négatifs.
- Les macro et weighted averages sont tous deux à 0.96, confirmant que les performances sont homogènes à travers toutes les classes, ce qui est cohérent avec notre distribution équilibrée des données.

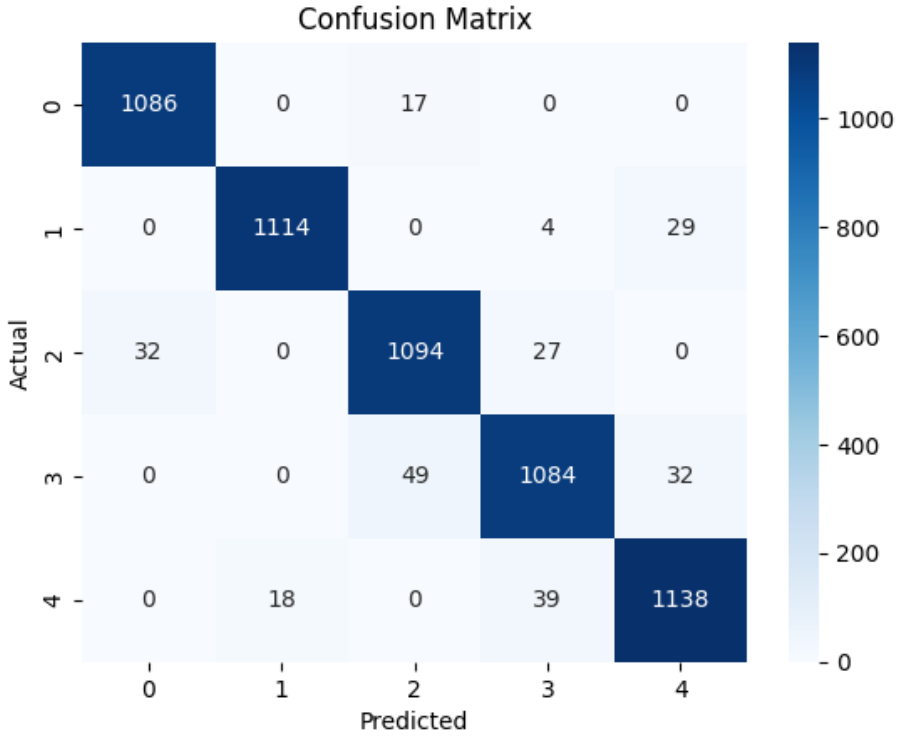

Figure 2: Matrice de confusion du meilleur modèle sur l'ensemble de test

La matrice de confusion (Figure 2) illustre visuellement les performances du modèle. On observe que :

- La diagonale principale est fortement dominante, avec des valeurs élevées (1086, 1114, 1094, 1084, 1138), confirmant que la majorité des prédictions sont correctes.
- Les erreurs de classification sont principalement concentrées entre classes adjacentes ou proches. Par exemple, la classe 0 (elite) est parfois confondue avec la classe 2 (high, 17 cas), ce qui est compréhensible étant donné que ces catégories représentent des joueurs de haut niveau.

- La classe 3 (medium) présente des confusions notables avec la classe 2 (high, 49 cas) et la classe 4 (low, 39 cas), ce qui est logique car les joueurs de niveau moyen se situent à la frontière entre différents niveaux de performance.
- Il y a très peu de confusions entre classes éloignées (par exemple, très peu de joueurs "elite" classés comme "low" et vice-versa), ce qui démontre que le modèle capture bien les différences fondamentales entre les niveaux de valeur extrêmes.
- Les erreurs les plus significatives concernent les classes centrales du spectre (medium et high), suggérant que la frontière entre ces niveaux intermédiaires est la plus difficile à déterminer pour le modèle, probablement en raison de la subtilité des différences entre ces catégories de joueurs.

2.5 Comparaison avec les résultats de la Partie 3

Lorsque nous comparons ces résultats avec ceux obtenus manuellement dans la partie 3, nous constatons **qu'aucune amélioration significative n'est observée**. Les performances obtenues par AutoML sont similaires, voire légèrement inférieures dans certains cas, à celles obtenues par la sélection et l'optimisation manuelles des modèles.

Cette absence d'amélioration s'explique principalement par les limitations de notre approche AutoML :

- **Espace de recherche trop restreint** : En limitant notre exploration à seulement deux algorithmes (XGBoost et RandomForest) avec des plages d'hyperparamètres réduites, nous avons considérablement limité la capacité de TPOT à découvrir des configurations potentiellement meilleures. D'autres algorithmes (SVM, réseaux de neurones, ensembles plus complexes) auraient pu offrir de meilleures performances.
- **Nombre limité de générations et taille de population réduite** : Avec seulement 10 générations et 15 individus par génération, l'algorithme n'a pas eu suffisamment d'opportunités pour explorer exhaustivement même l'espace restreint que nous avions défini.
- **Contrainte temporelle stricte** : La limite d'une heure d'exécution a forcé TPOT à terminer prématurément, potentiellement avant d'avoir convergé vers une solution optimale.
- **Complexité limitée des pipelines** : TPOT est capable de construire des pipelines complexes avec plusieurs étapes de prétraitement et d'ingénierie de caractéristiques. En restreignant notre espace de recherche, nous n'avons pas exploité pleinement cette capacité.

Ces résultats suggèrent qu'avec des ressources computationnelles plus importantes, permettant d'explorer un espace de recherche beaucoup plus large (davantage d'algorithmes, plus de valeurs d'hyperparamètres, plus de générations),

nous pourrions potentiellement observer des améliorations significatives par rapport aux résultats manuels. L'AutoML montre tout son potentiel lorsqu'il peut explorer exhaustivement un vaste espace de solutions, ce qui n'était pas possible dans notre contexte contraint.

Néanmoins, ces résultats confirment la qualité du travail manuel effectué dans la partie 3 : le fait que l'AutoML, même avec ses limitations, n'ait pas pu surpasser significativement nos choix manuels valide la pertinence de notre approche méthodologique initiale.

3 AutoML pour le Clustering

3.1 Préparation des données

Pour cette phase d'optimisation automatisée du clustering, nous avons réutilisé les données prétraitées et nettoyées de la partie 2. Ces données ont déjà subi toutes les transformations nécessaires : normalisation des variables numériques et exclusion des colonnes (Player Value, Wage et Overall Rating). Cette approche nous permet de nous concentrer directement sur l'optimisation des hyperparamètres de l'algorithme de clustering sans avoir à répéter les étapes de préparation des données.

3.2 Choix de l'outil d'optimisation : Hyperopt

Initialement, nous avions prévu d'utiliser les outils spécialisés pour l'AutoML en clustering suggérés dans le sujet du projet. Cependant, nous avons rencontré des problèmes techniques significatifs lors de leur installation et configuration, notamment des incompatibilités de dépendances et des limitations du système d'exploitation. Ces difficultés nous ont contraints à adopter une solution alternative.

Nous avons donc opté pour **Hyperopt**, une bibliothèque Python d'optimisation d'hyperparamètres. Bien que Hyperopt ne soit pas spécifiquement conçu pour le clustering, il s'agit d'un outil puissant et flexible pour l'optimisation bayésienne d'hyperparamètres dans tout type de problème d'optimisation. Hyperopt utilise l'algorithme Tree-structured Parzen Estimator (TPE) pour explorer intelligemment l'espace de recherche et identifier les configurations d'hyperparamètres les plus prometteuses.

3.3 Définition de la fonction objectif

La fonction objectif est au cœur du processus d'optimisation avec Hyperopt. Elle définit la métrique à optimiser et encapsule l'algorithme de clustering à évaluer. Pour notre cas, nous avons défini la fonction objectif comme suit :

- **Métrique d'évaluation** : Nous utilisons l'**indice de silhouette** (silhouette score) comme métrique de qualité du clustering. Ce choix est motivé par les résultats pertinents obtenus avec cette métrique dans la partie 2,

où l'indice de silhouette nous avait permis d'identifier des configurations de clustering cohérentes et bien séparées.

- **Objectif d'optimisation** : Hyperopt cherche à *minimiser* la fonction objectif. Or, l'indice de silhouette est une métrique à maximiser (des valeurs plus élevées indiquent un meilleur clustering). Nous avons donc défini notre fonction objectif pour retourner l'opposé de l'indice de silhouette (`-silhouette_avg`), transformant ainsi le problème de maximisation en problème de minimisation.

- **Structure de la fonction** : La fonction objectif prend en paramètre un dictionnaire contenant les hyperparamètres à tester, crée et entraîne un modèle K-means avec ces paramètres, calcule l'indice de silhouette sur les données, et retourne un dictionnaire contenant la perte (loss) et le statut de l'évaluation.

3.4 Définition de l'espace de recherche

L'espace de recherche définit l'ensemble des hyperparamètres que Hyperopt va explorer pour l'algorithme K-means. Compte tenu des contraintes techniques liées à Hyperopt, nous nous sommes limités à l'optimisation d'un seul algorithme : **K-means**.

Cette limitation est inhérente au fonctionnement de Hyperopt : bien qu'il soit théoriquement possible d'explorer plusieurs algorithmes différents, cela nécessiterait un niveau de complexité et de "bricolage" important dans la définition de l'espace de recherche. La structure de Hyperopt est optimisée pour l'optimisation d'hyperparamètres d'un algorithme donné plutôt que pour la sélection d'algorithmes. Nous avons donc choisi de nous concentrer sur l'optimisation approfondie de K-means plutôt que sur une exploration superficielle de plusieurs algorithmes.

Les hyperparamètres explorés pour K-means sont :

- **n_clusters** : Nombre de clusters, variant de 2 à 10.

- **init** : Méthode d'initialisation des centroïdes, avec deux options :
  - `k-means++` : méthode d'initialisation intelligente qui sélectionne les centroïdes initiaux de manière à être bien espacés
  - `random` : initialisation aléatoire des centroïdes

- **n_init** : Nombre d'exécutions de l'algorithme avec différentes initialisations, variant entre 10, 20, 30, 40 et 50. Un nombre plus élevé augmente les chances de trouver un optimum global mais accroît également le temps de calcul.

- **max_iter** : Nombre maximum d'itérations pour la convergence de l'algorithme, avec des valeurs de 100, 200, 300, 400 et 500. Ce paramètre contrôle combien de fois l'algorithme peut réajuster les centroïdes avant de s'arrêter.

- **algorithm** : Algorithme de calcul utilisé par K-means, avec trois options :
  - `lloyd`
  - `elkan` : variante optimisée qui utilise l'inégalité triangulaire pour accélérer les calculs

Cette définition de l'espace de recherche représente un compromis entre exhaustivité et faisabilité computationnelle, tout en couvrant les hyperparamètres les plus influents sur les performances de K-means.

3.5 Configuration de l'optimisation

Pour l'optimisation avec Hyperopt, nous avons utilisé les paramètres suivants :

- **algo=tpe.suggest** : Utilisation de l'algorithme Tree-structured Parzen Estimator (TPE), qui construit un modèle probabiliste de la fonction objectif pour guider la recherche vers les régions prometteuses de l'espace d'hyperparamètres.
- **max_evals=100** : L'optimisation évaluera au maximum 100 configurations différentes d'hyperparamètres. Cette limite permet un équilibre raisonnable entre exploration de l'espace de recherche et temps d'exécution.
- **trials=Trials()** : Objet permettant de stocker l'historique de toutes les évaluations effectuées, utile pour analyser le processus d'optimisation a posteriori.

3.6 Résultats de l'optimisation

3.6.1 Hyperparamètres optimaux identifiés

À l'issue du processus d'optimisation avec Hyperopt, après l'évaluation de 100 configurations différentes sur une durée d'environ 7 heures et 57 minutes, l'algorithme a identifié la configuration optimale suivante pour K-means :

- **n_clusters = 4** : nombre de clusters optimal
- **init = 'random'** : initialisation aléatoire des centroïdes
- **n_init = 30** : nombre d'exécutions avec différentes initialisations
- **max_iter = 100** : nombre maximum d'itérations pour la convergence
- **algorithm = 'elkan'** : algorithme de calcul optimisé utilisant l'inégalité triangulaire

Indice de silhouette obtenu : 0.1304 Plusieurs observations importantes peuvent être tirées de ces résultats :

- Le choix de 4 clusters suggère que les données de joueurs se structurent naturellement en quatre groupes distincts, potentiellement correspondant à différents profils ou niveaux de performance.

3.6.2 Visualisation de la solution de clustering

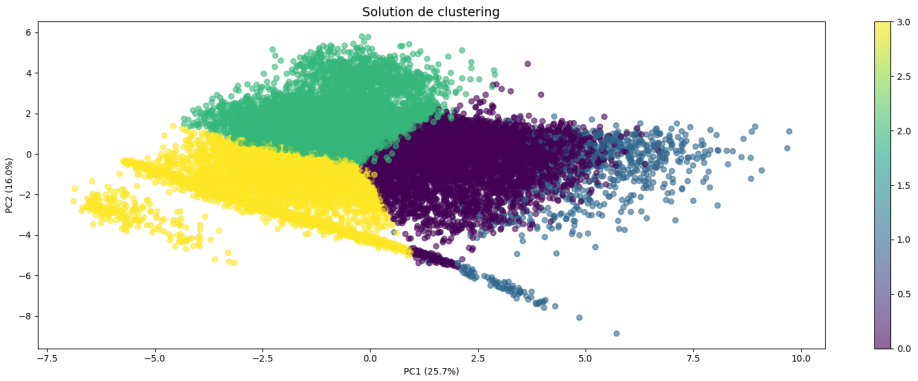

Figure 3: Visualisation de la solution de clustering optimale dans l'espace des composantes principales

La figure 3 présente les résultats du clustering optimal projeté dans l'espace des deux premières composantes principales (PC1 et PC2), qui capturent respectivement environ 25% et 16% de la variance totale des données.

La séparation visuelle entre les clusters est globalement claire, particulièrement entre les clusters jaune et vert. Cependant, on observe des zones de chevauchement, notamment entre les clusters violet et bleu dans la région centrale, ainsi qu'entre le cluster vert et les autres dans certaines zones de transition. Ces chevauchements expliquent en partie l'indice de silhouette modéré obtenu (0.1304), indiquant que si les clusters sont identifiables, les frontières entre eux ne sont pas toujours nettes. Il est important de noter que cette visualisation en deux dimensions ne représente qu'environ 41,7% de la variance totale des données. Les clusters pourraient être plus clairement séparés dans l'espace complet à haute dimension des caractéristiques originales.

3.6.3 Comparaison avec les résultats de la Partie 2

Lorsque nous comparons ces résultats avec ceux obtenus manuellement dans la partie 2 du projet, nous constatons **qu'aucune amélioration significative n'a été observée**. Les performances du clustering automatisé sont essentiellement équivalentes à celles obtenues par l'approche manuelle d'exploration et de sélection des hyperparamètres. Cette absence d'amélioration soulève plusieurs questions et peut s'expliquer par plusieurs facteurs :

1. Optimum potentiellement atteint Il est possible que les résultats obtenus dans la partie 2 représentaient déjà une configuration proche de l'optimum pour l'algorithme K-means sur ces données. Dans ce cas, même

une exploration automatisée plus systématique ne pourrait pas apporter d'amélioration substantielle. L'indice de silhouette modéré (environ 0.13) pourrait refléter une limite intrinsèque liée à la structure naturelle des données, qui ne présentent peut-être pas de clusters extrêmement bien séparés.

2. Limitations algorithmiques de K-means K-means possède des limitations bien connues : il suppose des clusters de forme sphérique, de tailles comparables, et de densités similaires. Si la véritable structure des données ne correspond pas à ces hypothèses, K-means ne pourra jamais produire un clustering optimal, quels que soient les hyperparamètres choisis. D'autres algorithmes de clustering (DBSCAN, clustering hiérarchique, GMM, spectral clustering) pourraient potentiellement révéler des structures plus complexes ou mieux adaptées à la nature de nos données.

3. Contraintes de l'approche d'optimisation Notre utilisation de Hyperopt, bien qu'efficace pour l'optimisation des hyperparamètres de K-means, nous a limités à un seul algorithme. Cette contrainte technique, due à la complexité d'intégrer plusieurs algorithmes différents dans le framework Hyperopt, a empêché une véritable exploration de l'espace des algorithmes de clustering. Une approche AutoML complète pour le clustering aurait idéalement exploré différentes familles d'algorithmes en plus de leurs hyperparamètres respectifs.

4. Limitations des ressources computationnelles Avec 100 évaluations réalisées, nous avons exploré un espace de recherche conséquent mais pas exhaustif. Un nombre plus élevé d'évaluations (par exemple 500 ou 1000) aurait pu révéler des configurations légèrement meilleures, bien que les rendements décroissants suggèrent que les gains potentiels seraient probablement marginaux.

4 Conclusion Générale

Ce projet nous a permis d'explorer de manière exhaustive l'ensemble du processus de fouille de données (data mining), depuis l'analyse exploratoire initiale jusqu'à l'automatisation complète des tâches d'apprentissage automatique. En suivant un cheminement méthodique à travers quatre parties distinctes mais complémentaires, nous avons pu appréhender concrètement chaque étape cruciale du cycle de vie d'un projet de science des données.

4.1 Un parcours complet à travers le data mining

La Partie 1, consacrée à l'analyse statistique et exploratoire des données, a posé les fondations essentielles de notre travail. Cette phase initiale nous a permis de comprendre en profondeur la structure de notre jeu de données, d'identifier les distributions des variables, de détecter les

valeurs aberrantes et de comprendre les relations entre les différentes caractéristiques. Cette étape, bien que parfois perçue comme préliminaire, s'est révélée cruciale pour toutes les phases ultérieures du projet.

La Partie 2, dédiée au clustering, nous a introduits aux techniques d'apprentissage non supervisé. En appliquant différents algorithmes, nous avons cherché à découvrir des structures sous-jacentes dans nos données, c'est-à-dire des groupes naturels de joueurs partageant des caractéristiques similaires. Bien que nous ayons identifié des configurations avec 3 ou 4 clusters, les résultats ont montré des limites en termes de séparation, avec des indices de silhouette modérés (autour de 0,13). Cette performance suggère soit une structure naturellement floue dans les données, soit la nécessité d'explorer d'autres familles d'algorithmes de clustering mieux adaptées à la complexité de nos données.

La Partie 3, axée sur la prédiction, a représenté un contraste marqué avec les résultats du clustering. En développant des modèles de classification pour prédire différentes cibles catégorielles, nous avons obtenu des performances remarquables. Avec des scores d'accuracy dépassant 95% et des scores ROC AUC approchant 99%, nos modèles se sont révélés hautement performants pour cette tâche de prédiction. Cette réussite démontre que, malgré les difficultés rencontrées en clustering, nos données contiennent des patterns discriminants clairs pour la prédiction de la valeur marchande.

La Partie 4, consacrée à l'automatisation via l'AutoML, a bouclé la boucle en introduisant des outils modernes d'optimisation automatique. L'utilisation de TPOT pour la prédiction et de Hyperopt pour le clustering nous a permis d'explorer systématiquement l'espace des hyperparamètres. Bien que ces approches automatisées n'aient pas significativement amélioré les résultats obtenus manuellement, elles ont validé la qualité de nos choix méthodologiques et ont révélé les limites imposées par nos contraintes de ressources computationnelles.

4.2 L'importance de chaque étape

L'un des enseignements de ce projet réside dans la compréhension de l'interdépendance et de l'importance de chaque étape du processus de data mining. Chaque phase prépare et conditionne la réussite des étapes suivantes :

- **L'analyse exploratoire** n'est pas une simple formalité : elle permet d'identifier les variables pertinentes, de détecter les problèmes de qualité des données et de guider les choix de prétraitement. Une analyse superficielle ou bâclée à cette étape peut faire manquer des informations qui auraient pu améliorer significativement les performances des modèles ultérieurs.

- **L'ingénierie des features** (feature engineering), souvent sous-estimée, joue un rôle déterminant dans la qualité des résultats. Les transformations, encodages et créations de nouvelles variables effectués lors des phases de prétraitement ont directement influencé les capacités prédictives de nos modèles.
- **Le choix et le réglage des algorithmes**, qu'ils soient de clustering ou de prédiction, nécessitent une compréhension approfondie à la fois des données et des hypothèses sous-jacentes de chaque méthode. L'automatisation peut faciliter ce processus mais ne remplace pas une réflexion méthodologique rigoureuse.
- **L'optimisation des hyperparamètres**, manuelle ou automatisée, représente souvent la différence entre un modèle médiocre et un modèle performant. Cependant, son efficacité dépend intrinsèquement de la qualité de toutes les étapes précédentes.

Ce projet nous a permis de comprendre un principe fondamental du data mining : *négliger ou précipiter une seule étape du processus peut compromettre l'ensemble des résultats.* À l'inverse, un travail rigoureux et attentif à chaque phase crée un effet cumulatif positif qui se reflète dans la qualité finale des modèles.

4.3 Résultats et perspectives

En termes de résultats concrets, ce projet présente un bilan contrasté mais instructif :

Pour le **clustering**, les résultats obtenus demeurent mitigés. Avec des indices de silhouette autour de 0,13 et des visualisations montrant des chevauchements significatifs entre clusters, la séparation identifiée, bien qu'exploitable, reste à la limite du satisfaisant. Cette situation soulève plusieurs questions importantes :

- Les données possèdent-elles réellement une structure de clusters bien définie, ou leur nature est-elle intrinsèquement plus continue et graduelle ?
- L'algorithme K-means, avec son hypothèse de clusters sphériques et de tailles similaires, est-il adapté à la complexité potentielle de ces données ?
- D'autres familles d'algorithmes (DBSCAN pour des formes arbitraires, clustering hiérarchique pour des structures imbriquées, ou GMM pour des distributions plus flexibles) pourraient-elles révéler des structures plus satisfaisantes ?

Ces questions restent ouvertes et suggèrent que le clustering de ces données mérite une exploration plus approfondie avec des approches algorithmiques diversifiées.

Pour la **prédiction**, en revanche, les résultats sont excellents et cohérents à travers les différentes approches (manuelles et automatisées). Avec des accuracies dépassant 95% et des scores ROC AUC supérieurs à 99%, nos modèles démontrent une capacité remarquable à prédire la catégorie de valeur marchande des joueurs. Cette performance élevée s'est maintenue que nous utilisions une optimisation manuelle minutieuse ou des outils d'AutoML, confirmant la robustesse de nos résultats et la qualité de notre prétraitement des données.