Samar El Ayed APST 1

Valentine Chéruel 2024

# Rapport Data Challenge - Football : Qui va gagner ?

# Introduction

Le domaine des sciences des données connaît une expansion rapide, notamment dans les sports professionnels, où les analyses de données influencent désormais les décisions stratégiques. Ce rapport s'inscrit dans le cadre du **QRT Data Challenge 2024**, une compétition axée sur la prédiction des résultats de matchs de football en utilisant des données historiques riches et variées.

Dans ce défi, les participants exploitent des statistiques détaillées sur les équipes et les joueurs, couvrant plusieurs ligues de football à travers le monde. Ces données, fournies par **Sportmonks**, incluent des performances passées, des agrégats statistiques et des indicateurs de match. L'objectif principal est de développer des modèles prédictifs capables de généraliser efficacement à différents contextes de compétition.

Le rapport présentera en détail :

- **La compréhension du challenge**, incluant les objectifs, les contraintes et la structure des données.
- **Les méthodologies employées**, notamment les approches d'ingénierie des caractéristiques et de modélisation.
- **Les résultats obtenus**, analysés en comparaison aux benchmarks fournis.
- **Les conclusions et perspectives**, discutant des limites et améliorations potentielles.

Ce document reflète une démarche rigoureuse visant à intégrer des approches avancées de modélisation tout en assurant une interprétabilité et une généralisation robustes.

# Explication du challenge

Le défi proposé consiste à résoudre un problème de **classification multiclasse**, où l'objectif est de prédire le résultat d'un match de football en se basant sur des données historiques détaillées. Les trois classes possibles sont : **victoire de l'équipe à domicile**, **match nul**, ou **victoire de l'équipe à l'extérieur**.

Pour relever ce défi, deux ensembles de données distincts sont fournis : l'un pour les **équipes**, et l'autre pour les **joueurs**, comprenant diverses statistiques collectées sur plusieurs matchs précédents. L'objectif est de construire des modèles prédictifs robustes et généralisables, capables de fonctionner indépendamment des ligues ou des régions. En exploitant au mieux les données disponibles, les modèles doivent fournir des prédictions précises pour ce problème multiclasse complexe.

# Les données utilisées

Pour relever ce défi, nous disposons de deux ensembles de données principaux : les données des **équipes** et des **joueurs**, chacun offrant une vue complémentaire sur les performances des équipes de football.

1. **Données des équipes** : Ces données incluent 25 statistiques calculées à partir des cinq derniers matchs et des performances cumulées de la saison. Elles regroupent des métriques telles que les **tirs cadrés**, les **fautes commises**, les **passes réussies** ou encore les **cartons rouges et jaunes**.
2. **Données des joueurs** : Plus granulaires, elles incluent 52 statistiques par poste et joueur, offrant une vue détaillée des contributions individuelles à la performance de l'équipe. Ces statistiques couvrent des éléments comme les **attaques**, les **tirs**, les **passes réussies**, et d'autres indicateurs spécifiques.

Les deux ensembles sont structurés pour les équipes à domicile (*home team*) et à l'extérieur (*away team*), et on a standarisé les variables pour une comparaison facile. L'objectif est d'utiliser ces informations pour maximiser la précision des modèles prédictifs.

# Méthodes utilisées :

Pour évaluer les performances des différentes méthodes utilisées, nous avons calculé la précision (accuracy) pour chaque modèle. L'accuracy mesure la proportion de prédictions correctes parmi l'ensemble des prédictions effectuées et est définie par la formule suivante :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Cette métrique a été utilisée comme critère principal pour comparer les modèles, car elle offre une vue globale de leur capacité à classer correctement les résultats dans ce problème de classification multiclasse déséquilibrée. Chaque modèle a été évalué en termes d'accuracy sur les ensembles d'entraînement, de validation et de test pour identifier celui qui s'adapte le mieux aux données et fournit les prédictions les plus robustes.

**Gradient Boosting Classifier :** Un modèle d'ensemble qui construit des arbres décisionnels de manière séquentielle, chaque arbre corrigeant les erreurs de l'arbre précédent. Il est particulièrement puissant pour les problèmes de classification et peut gérer des données complexes.

**XGBClassifier :** Un algorithme de gradient boosting optimisé (XGBoost) qui utilise un modèle d'arbres décisionnels comme base. Il est conçu pour être rapide et performant, et il inclut des techniques comme le contrôle du sur-apprentissage et la régularisation.

**Random Forest Classifier :** Un modèle d'ensemble basé sur des arbres décisionnels, où plusieurs arbres sont formés sur des sous-échantillons aléatoires de données et leurs prédictions sont moyennées pour obtenir un résultat plus robuste et précis.

**Logistic Regression :** Un modèle simple de régression utilisé pour des problèmes de classification binaire ou multiclasse, qui prédit la probabilité qu'une observation appartienne à une classe spécifique en utilisant une fonction logistique.

**Neural Networks :** Des modèles d'apprentissage profonds composés de couches de neurones, capables de capturer des relations complexes dans les données. Utilisés pour des tâches comme la reconnaissance d'images ou la prédiction de séries temporelles.

**Ensemble Methods :** Ces méthodes combinent plusieurs modèles de base (comme les arbres décisionnels) pour améliorer la précision des prédictions. Par exemple, le stacking et le bagging combinent les forces de modèles individuels.

**CatBoost :** Un modèle de gradient boosting développé pour traiter efficacement des données catégorielles. Il est optimisé pour être rapide et nécessite moins de prétraitement des données.

**KNN (K-Nearest Neighbors) :** Un modèle basé sur la proximité où chaque point est classé selon la classe majoritaire de ses voisins les plus proches dans l'espace des caractéristiques. Simple mais efficace pour des jeux de données non linéaires**.**

**LightGBM :** Un modèle de boosting basé sur des arbres qui se distingue par sa rapidité et son efficacité, particulièrement sur des jeux de données volumineux. Il utilise des histogrammes pour accélérer l'apprentissage tout en conservant de bonnes performances.

Les notebooks dédiés au traitement des données des équipes (*team data*) et des joueurs (*player data*) sont disponibles dans le dépôt GitHub suivant : GuessTheWinner-DataChallenge.

# Team data :

## Analyse des données :

Dans cette étape, une analyse approfondie des données a été réalisée sur les ensembles d'entraînement pour les équipes à domicile et à l'extérieur, ainsi que sur l'ensemble des données combinées. On a d'abord examiné les dimensions des différents jeux de données, qui contiennent **12303** entrées et **140** colonnes pour chaque équipe (à domicile et à l'extérieur), et un total de **280** colonnes après la combinaison des deux ensembles de données.

Ensuite, on a analysé les **données manquantes** dans les colonnes et les lignes des jeux de données. Il a été observé qu'aucune colonne n'avait de données manquantes, mais certaines lignes contenaient une quantité significative de valeurs manquantes (plus de **50%**). Cependant, on a constaté qu'une colonne avait environ **27%** de valeurs manquantes, ce qui est relativement élevé. Malgré cela, aucune ligne ni colonne n'a été supprimée pour éviter de perdre des informations potentiellement importantes.

En raison de la présence de données manquantes, une **imputation** des valeurs manquantes a été planifiée. On a estimé qu'il y aurait suffisamment de données pour effectuer cette imputation sans introduire de biais significatifs, et cette opération sera réalisée au moment de l'entraînement du modèle pour garantir que les données soient prêtes à être utilisées efficacement.

## Entraînement de modèles :

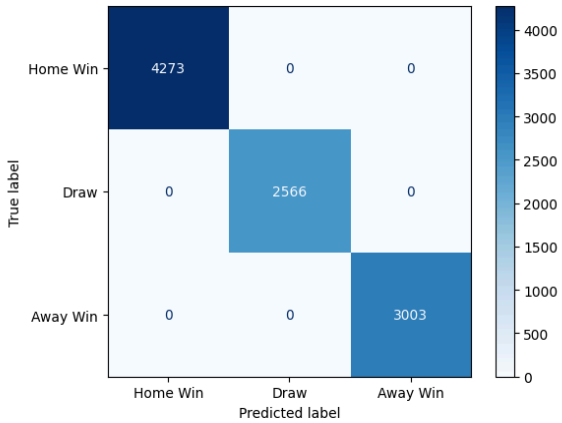
On a préparé la variable cible en combinant les colonnes HOME_WINS, DRAW et AWAY_WINS, représentant les résultats des matchs, et on a utilisé la fonction idxmax pour attribuer à chaque ligne la catégorie ayant la valeur maximale parmi ces trois colonnes. Ensuite, on a mappé ces catégories en valeurs numériques : HOME_WINS a été codé par 0, DRAW par 1, et AWAY_WINS par 2, créant ainsi une cible unique adaptée pour la modélisation.

Les données ont été divisées en un ensemble d'entraînement (80 %) et un ensemble de test (20 %) pour garantir une évaluation réaliste et indépendante des performances du modèle. Étant donné que les classes cibles pouvaient être déséquilibrées, on a calculé les poids des classes pour chaque catégorie. Cette méthode, utilisant l'option balanced de la fonction compute_class_weight, permet d'attribuer un poids plus élevé aux classes sous-représentées, évitant ainsi que le modèle soit biaisé vers les classes majoritaires.

Les poids des classes calculés étaient les suivants :

- Classe 0 (HOME_WINS) : 0.7677
- Classe 1 (DRAW) : 1.2785
- Classe 2 (AWAY_WINS) : 1.0925

Ces valeurs montrent que la classe DRAW a reçu le poids le plus élevé (1.2785) car elle était probablement sous-représentée dans les données, tandis que la classe HOME_WINS a reçu un poids plus faible (0.7677), indiquant qu'elle était plus présente. Le poids de la classe AWAY_WINS est intermédiaire, à 1.0925, reflétant une distribution des classes relativement équilibrée, mais pas autant que celle des égalités.




Avant d'entraîner le modèle, les valeurs manquantes ont été imputées en utilisant la médiane de chaque colonne, puis les données ont été standardisées à l'aide de StandardScaler afin d'assurer une échelle uniforme pour toutes les variables. Étant donné qu'il s'agit d'un problème de classification multiclasse déséquilibrée avec une forte dimensionalité, des techniques ont été appliquées pour améliorer les performances : la réduction de dimensionnalité par PCA et le suréchantillonnage SMOTE-Tomek Links pour équilibrer les classes.

## Méthodes utilisées :

## Gradient Boosting

Dans la section **Gradient Boosting**, un modèle a d'abord été entraîné avec des paramètres de base, obtenant une accuracy de **49.6%** . Une optimisation des hyperparamètres via **GridSearchCV** a permis d'améliorer légèrement l'accuracy à **49.7%**.

Ensuite, une analyse de l'**importance des caractéristiques** a révélé que certaines variables, comme AWAY_TEAM_BALL_POSSESSION et HOME_TEAM_SHOTS_TOTAL, étaient les plus influentes. Un modèle avec **poids de classes** a été testé pour gérer les déséquilibres, mais a diminué la précision à **45.4%**.

Finalement, une **réduction de la dimensionnalité** via PCA a confirmé qu'il y avait peu de redondance dans les caractéristiques, et une sélection des variables les plus importantes a conduit à une précision de **49.4%**. En résumé, bien que des ajustements aient amélioré le modèle, la précision reste proche de **50%**.

## Random Forest :

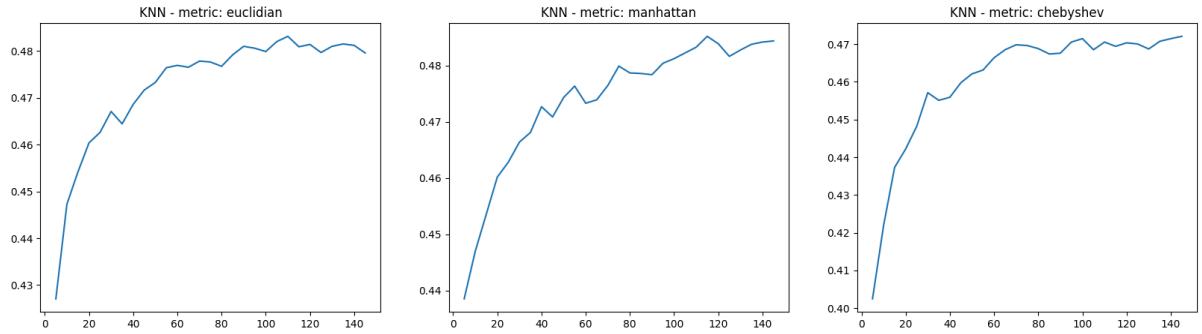
Dans cette section, un modèle **Random Forest** a été utilisé pour prédire les résultats des matchs. Le modèle initial a donné une précision d'environ **47.5%**. Ensuite, une **recherche par grille (GridSearchCV)** a été effectuée pour optimiser les hyperparamètres, ce qui a amélioré la précision à **48.7%**.

Une analyse de l'**importance des caractéristiques** a permis de mettre en évidence les variables les plus influentes, bien que toutes aient une importance inférieure à 1%. Une **réduction de la redondance** a été réalisée en utilisant une analyse de corrélation, et des **caractéristiques sélectionnées** ont été retenues pour simplifier le modèle. En désactivant l'option **bootstrap**, le modèle a été réajusté, obtenant une précision finale de **50.2%**.

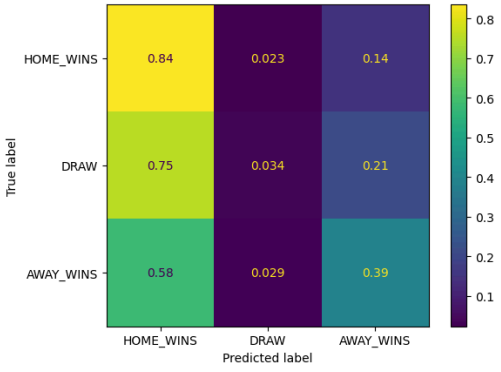
En résumé, après l'optimisation des paramètres et la sélection des caractéristiques, la précision du modèle a légèrement augmenté, mais reste autour de **50%**, suggérant qu'il peut y avoir encore des possibilités d'amélioration.

## K-Nearest Neighbors (KNN) :

Dans la section **K-Nearest Neighbors (KNN)**, un **grid search** a été effectué pour optimiser les hyperparamètres du modèle, notamment le nombre de voisins (n_neighbors) et la distance utilisée (euclidean, manhattan, chebyshev). Après l'exécution du **GridSearchCV**, le meilleur modèle a été trouvé avec n_neighbors = 115 et la métrique manhattan, donnant une précision d'environ **49.7%** sur les données de test.



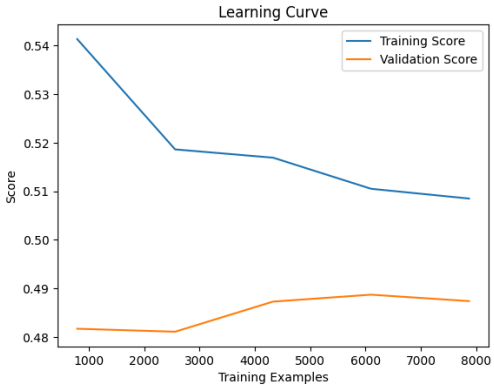

Les courbes de performance pour chaque distance ont montré l'évolution du score en fonction du nombre de voisins. On a observé un **overfitting** lorsque le nombre de voisins était faible, ce qui signifie que le modèle était trop spécifique aux données d'entraînement. Cependant, après 100 voisins, la courbe a montré une **stabilisation** du score, suggérant que le modèle atteint un équilibre entre sous-apprentissage et sur-apprentissage.



Une matrice de confusion normalisée a été générée pour évaluer les performances du modèle, permettant d'observer les prédictions pour chaque classe. Elle montre que le modèle prédit bien les victoires à domicile avec une taux de 84 %, mais confond souvent les victoires à l'extérieur (39 % correctes) et les matchs nuls (seulement 3,4 % corrects) avec d'autres classes. Les erreurs les plus fréquentes sont la classification des matchs nuls et des victoires à l'extérieur comme des victoires à domicile.

## Régression Logistique :

Dans la section **Régression Logistique**, un modèle de régression logistique a d'abord été entraîné sur les données avec une accuracy de **48.4%**. Ensuite, une recherche par grille a été effectuée pour optimiser les hyperparamètres, tels que le paramètre de régularisation C, le solveur, et la pénalité. Le meilleur modèle trouvé a été configuré avec C = 0.001, solver = 'saga', et penalty = 'l2', ce qui a légèrement amélioré la précision à **49.6%**.




Une matrice de confusion normalisée a été utilisée pour évaluer les performances du modèle, et une courbe d'apprentissage a été tracée pour observer l'évolution des scores d'entraînement et de validation. Cette courbe a montré que le modèle s'améliorait à mesure que le nombre d'exemples d'entraînement augmentait, indiquant une bonne capacité de généralisation.

En résumé, la recherche par grille a permis d'améliorer légèrement la précision du modèle, et l'analyse de la courbe d'apprentissage a montré une amélioration progressive des performances avec l'augmentation des données d'entraînement.

## XGBoost :

On a d'abord entraîné un modèle **XGBoost** avec des paramètres de base, obtenant une précision de **49.7%** sur les données de test. Une **recherche par grille** a permis d'optimiser certains hyperparamètres, notamment la profondeur des arbres, et les régularisations **alpha** et **lambda**. Le meilleur modèle obtenu avait les paramètres max_depth=3, reg_alpha=0.1 et reg_lambda=0.1, avec une précision de **49.7%**.

On a ensuite effectué une deuxième recherche par grille pour optimiser le paramètre **eta** (taux d'apprentissage), ce qui a permis d'obtenir une précision légèrement améliorée de **49.1%** avec **eta=0.06**.

L'**importance des caractéristiques** a été analysée, et seules les variables ayant une importance supérieure à **1%** ont été conservées pour entraîner un nouveau modèle. Ce modèle, utilisant uniquement ces caractéristiques, a donné une précision de **49.7%**.

Finalement, on a utilisé **Optuna** pour effectuer une optimisation automatisée des hyperparamètres, ajustant des paramètres comme max_depth, learning_rate, subsample, et colsample_bytree. Le meilleur modèle obtenu grâce à **Optuna** a donné une précision de **49.5%** sur les données de validation, avec des paramètres comme max_depth=6, learning_rate=0.0329, et des régularisations alpha=4.42 et lambda=3.50.

Malgré plusieurs optimisations des paramètres, la précision du modèle XGBoost reste autour de **49.5%**.

## CatBoost :

On a d'abord entraîné un modèle **CatBoost** en utilisant une **recherche par grille** pour optimiser les hyperparamètres, tels que le nombre d'itérations, le taux d'apprentissage, la profondeur des arbres, et les régularisations **L2**. Le meilleur modèle trouvé avait les paramètres bagging_temperature=0.2, depth=6, iterations=1000, l2_leaf_reg=7, et learning_rate=0.01.

Ensuite, on a effectué une évaluation robuste avec une **validation croisée** en utilisant une **Stratified K-Fold** avec 5 plis. La précision moyenne obtenue après la validation croisée était de **46.26%**. Après cela, on a entraîné un modèle final sur l'ensemble des données d'entraînement, ce qui a donné une précision finale de **44.49%**.

On a également analysé l'**importance des caractéristiques**, en sélectionnant celles ayant une importance supérieure à **1%**. Cela a permis de réduire le nombre de caractéristiques à conserver, mais l'impact sur la précision était limité.

Enfin, on a essayé d'ajouter des **poids de classe** pour compenser les déséquilibres dans les données, mais cela n'a pas amélioré la précision du modèle.

Bien que plusieurs optimisations aient été réalisées, la précision du modèle CatBoost est restée autour de **44.5%**, indiquant que des ajustements supplémentaires pourraient être nécessaires pour améliorer les performances.

### LightGBM :

On a d'abord entraîné un modèle **LightGBM** avec des paramètres de base : n_estimators=300, learning_rate=0.05, max_depth=7, et num_leaves=31. Le modèle a donné une précision de **47.9%** sur les données de test.

Ensuite, on a optimisé les hyperparamètres du modèle en utilisant **GridSearchCV**. Les paramètres ajustés incluaient le taux d'apprentissage, le nombre de feuilles, la profondeur des arbres et le nombre d'estimateurs. Le meilleur modèle obtenu après optimisation avait les paramètres : learning_rate=0.01, max_depth=5, n_estimators=300, et num_leaves=50. Cependant, la précision du modèle amélioré était de **45.3%**, ce qui est inférieur à la précision initiale.

Malgré l'optimisation des hyperparamètres, la précision du modèle LightGBM n'a pas significativement augmenté, restant autour de **45.3%**.

### Stacking :

On a d'abord entraîné un modèle de **Stacking** en utilisant trois modèles de base : **XGBoost**, **CatBoost**, et **LightGBM**, avec un **Logistic Regression** comme modèle final. Ce modèle a donné une précision de **50.1%** sur les données de test.

Ensuite, on a optimisé les hyperparamètres du modèle de **Logistic Regression** dans le modèle de **Stacking** en utilisant **GridSearchCV**. Les paramètres ajustés comprenaient le paramètre **C** (pour la régularisation) et le **solver**. Le modèle amélioré a donné une précision similaire de **50.1%** sur les données de test, avec les meilleurs paramètres trouvés pour **C = 1** et **solver = 'liblinear'**.

En résumé, bien que l'optimisation ait permis d'ajuster les paramètres du modèle final, la précision du modèle de **Stacking** est restée autour de **50.1%**, ce qui montre que cette approche n'a pas apporté d'amélioration significative par rapport aux modèles de base.

### Analyse des scores :

L'**analyse des scores** montre les performances des différents modèles utilisés dans ce projet, triées par ordre décroissant de précision. Le modèle **Random Forest avec sélection de caractéristiques** a obtenu la meilleure précision, avec **50.18%**, suivi de **Stacking** avec **50.14%**, et de **Stacking amélioré** à **50.10%**. Ces trois modèles sont légèrement au-dessus de **50%** de précision, ce qui indique une performance modérée dans cette tâche.

Les autres modèles, comme **K-Nearest Neighbors (KNN)**, **XGBoost avec variables sélectionnées** et **Gradient Boosting**, ont obtenu des scores autour de **49.6%**, tandis que les modèles de régression logistique, qu'ils soient de base ou optimisés, ont montré des performances légèrement inférieures (environ **49%**).

Les modèles **LightGBM** ont donné des résultats relativement plus faibles, avec des précisions de **47.9%** pour le modèle de base et **45.3%** pour le modèle optimisé.

Enfin, le modèle **CatBoost** a obtenu la précision la plus basse avec **44.5%**. Le modèle **Neural Network (NN)** a également montré des performances faibles, avec une précision de **41.6%** pour le modèle de base et **48.8%** pour le modèle standardisé, indiquant une amélioration

| | score |
|---|---|
| score_rf_selected | 0.501829 |
| improved_stacking_accuracy | 0.501422 |
| stacking_accuracy | 0.501016 |
| KNN Score | 0.496546 |
| score_xgb_var_imp | 0.496546 |
| best_model_gb_accuracy | 0.496140 |
| Reg Logistique Opt Hyparam | 0.495733 |
| xgb_optuna_accuracy | 0.495327 |
| score_gb_selected | 0.494514 |
| nn_accuracy_standardised | 0.488013 |
| rf_best_model_accuracy | 0.487200 |
| score_rf_nb | 0.486794 |
| Reg Logistique Score | 0.483950 |
| lightGBM_accuracy | 0.479074 |
| lightGBM_accuracy_improved | 0.453474 |
| final_accuracy catboost | 0.444941 |
| nn_accuracy | 0.415685 |

par la standardisation des données, mais sans atteindre les performances des autres modèles.

En résumé, les meilleurs résultats ont été obtenus avec **Random Forest**, **Stacking**, et **XGBoost**, tandis que **CatBoost**, **LightGBM**, et les réseaux de neurones ont montré des performances inférieures.

### Résultats de soumissions :

En comparant les résultats des modèles obtenus lors de l'analyse des scores et des soumissions sur les données de test, on constate des écarts modérés. Les modèles **Random Forest** et **Stacking Ensemble** ont montré des performances similaires entre l'analyse et les soumissions, avec des différences marginales autour de **50%** de précision. En revanche, certains modèles comme **XGBoost** et **Neural Network** ont enregistré des précisions plus faibles sur les données de test par rapport à l'analyse, suggérant une légère perte de généralisation. Par exemple, le **Neural Network** est passé de **48.8%** en analyse à **46.82%** dans la soumission. Les modèles **LightGBM** et **KNN** ont également montré des performances inférieures sur les données de test, avec des précisions respectives de **46.26%** et **46.82%**, indiquant que l'optimisation n'a pas permis d'améliorer significativement leurs performances.

Le modèle **CatBoost** a présenté un comportement différent. Bien qu'il ait obtenu une précision modérée sur les données d'entraînement, il a atteint une précision plus élevée de **48.8%** lors de la soumission. Cette amélioration peut être attribuée à la nature des données de test, qui sont probablement **plus équilibrées** en termes de distribution des classes. Cela a permis au modèle de mieux prédire les classes minoritaires, un point fort de CatBoost grâce à sa gestion efficace des déséquilibres de classe. En résumé, bien que la plupart des modèles aient rencontré des difficultés à se généraliser, **CatBoost** s'est distingué par une meilleure précision sur les données de test.

# Player data :

### Analyse des données

Nous avons préparé les données des joueurs pour l'analyse et la modélisation. Les statistiques des joueurs des équipes à domicile et à l'extérieur ont été importées et nettoyées, en supprimant les colonnes non présentes dans les données de test, telles que TEAM_NAME, LEAGUE et PLAYER_NAME. La colonne POSITION a été transformée en valeurs numériques pour simplifier l'analyse, en attribuant un code à chaque rôle de joueur (1 pour défenseur, 2 pour milieu, 3 pour attaquant, et 4 pour gardien), et les valeurs manquantes ont été remplacées par 0.

Les données des joueurs ont ensuite été agrégées par ID et POSITION, en calculant les moyennes des statistiques pour chaque groupe de joueurs, ce qui a réduit la complexité des données. Les valeurs manquantes résultant de cette agrégation ont été remplacées par 0. Enfin, les données des équipes à domicile et à l'extérieur ont été fusionnées pour permettre une comparaison directe, et des colonnes représentant les différences statistiques entre les deux équipes ont été créées pour enrichir les futurs modèles prédictifs.

Cette préparation a permis de constituer un jeu de données consolidé, prêt à identifier des relations significatives entre les statistiques des joueurs et les résultats des matchs.

### Analyse des données manquantes

Dans l'**analyse des données manquantes**, une inspection approfondie a été réalisée pour identifier les colonnes contenant des valeurs manquantes dans les jeux de données des joueurs. Une grande proportion des colonnes était concernée, avec **99.34%** des colonnes présentant des données manquantes dans les ensembles de données à domicile et à l'extérieur. Les lignes des données comportaient également des valeurs manquantes à **100%**, ce qui suggère que de nombreux joueurs n'avaient pas participé à tous les matchs ou que des erreurs étaient présentes dans la collecte des données.

En raison de cette abondance de données manquantes, certaines colonnes contenant plus de **30%** de valeurs manquantes ont été supprimées, réduisant ainsi la dimension des données de **304** à **271** colonnes. Cependant, la quantité de données

manquantes par ligne était toujours élevée, avec plus de **90%** des données manquantes dans certains cas pour les données de test.

Face à cette situation, une **imputation des données** a été jugée inappropriée, car elle aurait impliqué l'ajout de valeurs manquantes à presque toutes les lignes, ce qui affecterait la qualité des données. Bien que les données des joueurs ne soient pas idéales, nous avons décidé de continuer à les utiliser pour évaluer la précision des prédictions, en dépit de leur inexactitude. **les données des équipes ont été privilégiées car elles offrent une vue plus stable, cohérente et complète du match**, ce qui est essentiel pour créer des modèles de prédiction robustes et performants. C'est pourquoi plus d'efforts ont été consacrés à l'amélioration de ces données, en mettant l'accent sur la réduction des biais et l'optimisation des modèles basés sur les caractéristiques collectives de l'équipe.

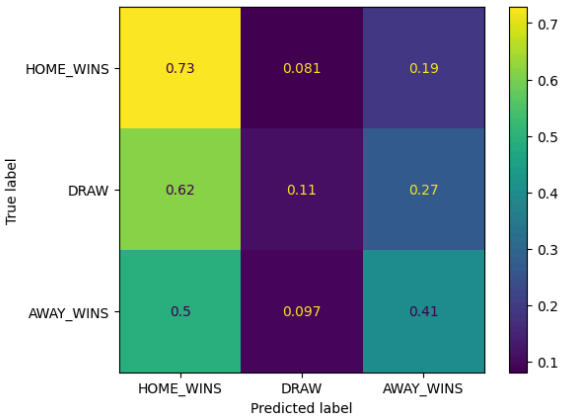
## Méthodes utilisées :

### Gradient Boosting Classifier:

On a utilisé un Gradient Boosting Classifier  pour prédire les résultats des matchs, avec plusieurs optimisations pour améliorer ses performances. Après un entraînement initial avec des paramètres de base, le modèle a obtenu une précision de 28,75% sur les données de validation. Une analyse de l'importance des caractéristiques a identifié les variables les plus influentes, permettant de réduire la dimensionnalité à 25 caractéristiques principales. Des techniques comme la standardisation, la sélection de caractéristiques pertinentes et l'équilibrage des classes via SMOTE ont été appliquées pour affiner le modèle. Enfin, l'ajout d'un early stopping pour éviter le surapprentissage a permis d'atteindre une précision finale de 48,45%.

### XGBClassifier

Pour les données des joueurs, un modèle XGBClassifier a été utilisé pour prédire les résultats des matchs. Une recherche par grille (GridSearchCV) a été réalisée pour optimiser les hyperparamètres clés, notamment la profondeur des arbres (max_depth), le taux d'apprentissage (learning_rate), le nombre d'arbres (n_estimators), ainsi que les fractions de colonnes et d'échantillons utilisées pour chaque arbre (colsample_bytree, subsample). Les meilleurs paramètres trouvés étaient : max_depth=10, learning_rate=0.05, n_estimators=1000, colsample_bytree=0.8, et subsample=1.0. Le modèle entraîné avec ces paramètres a été évalué sur un ensemble de validation, obtenant une accuracyde 47.1%. Une matrice de confusion a été générée pour visualiser la répartition des prédictions par classe (victoires à domicile, égalités, victoires à l'extérieur).




La matrice de confusion montre que le modèle XGBClassifier prédit correctement **73% des victoires à domicile**, mais a des difficultés à reconnaître les **matchs nuls** (11% d'accuracy) et les **victoires à l'extérieur** (41% d'accuracy). Les matchs nuls sont souvent confondus avec des victoires à domicile (62%), tandis que les victoires à l'extérieur sont majoritairement mal classées comme victoires à domicile (50%). Cela suggère un biais du modèle en faveur des victoires à domicile, possiblement lié à un déséquilibre des classes ou à des limites dans les données utilisées pour l'entraînement.

Enfin, les valeurs SHAP ont été utilisées pour expliquer les prédictions du modèle, mettant en évidence l'importance relative des caractéristiques, comme la variable POSITION. Cette analyse a permis de mieux comprendre les contributions des différentes caractéristiques au modèle et d'identifier les points potentiels d'amélioration.

### Random Forest Classifier:

Ce modèle est particulièrement adapté pour capturer des relations non linéaires et analyser l'importance des caractéristiques. Initialement, un modèle de base a été construit avec 100 arbres de décision, obtenant une accuracy de validation de 47,55%. Ensuite, une recherche d'hyperparamètres (GridSearchCV) a été réalisée pour optimiser les paramètres clés, notamment la profondeur maximale des arbres (max_depth), le nombre minimum d'échantillons pour diviser un nœud (min_samples_split), et le nombre total d'arbres (n_estimators). Les meilleurs paramètres identifiés étaient : une profondeur maximale de 10, un

minimum de 2 échantillons pour la division, et 100 arbres. En utilisant ces paramètres, un modèle amélioré a été entraîné, atteignant une accuracy accrue de 48,22% sur les données de validation, ce qui démontre l'impact positif de l'optimisation sur les performances du modèle.

**Logistic Regression:**

Ce modèle est connu pour sa simplicité et son interprétabilité, particulièrement adapté lorsque les relations entre les variables explicatives et la cible sont principalement linéaires. Une version multinomiale de la régression logistique a été choisie pour gérer la classification multi-classes, en utilisant le solveur lbfgs pour l'optimisation. Après l'entraînement sur les données, le modèle a atteint une accuracy de 47,92% sur l'ensemble de validation. Bien que son efficacité soit légèrement inférieure à celle des modèles plus complexes, la régression logistique reste un point de référence important en raison de sa robustesse et de sa capacité à fournir des résultats interprétables.

**Neural Networks:**

Un réseau de neurones artificiel (Neural Network) a été utilisé pour capturer les relations complexes et les interactions entre les caractéristiques. Le modèle implémente un MLPClassifier avec deux couches cachées de tailles 64 et 32, et utilise la fonction d'activation ReLU pour introduire la non-linéarité. Ce modèle est particulièrement efficace pour les ensembles de données volumineux et complexes, où il peut détecter des schémas que les modèles traditionnels pourraient manquer. Cependant, dans cette application, le modèle a obtenu une accuracy de 39,43% sur l'ensemble de validation, ce qui est relativement faible. Cela pourrait indiquer que des ajustements supplémentaires des hyperparamètres ou un enrichissement des données d'entraînement sont nécessaires pour améliorer les performances.

**Ensemble Methods**

on a utilisé les méthodes d'ensemble pour améliorer la robustesse et la accuracy des prédictions en combinant les forces de plusieurs modèles. Dans cette approche, un VotingClassifier a été créé en combinant un Random Forest Classifier et un Gradient Boosting Classifier, avec une méthode de vote "soft" qui prend en compte les probabilités prédites par chaque modèle. L'ensemble a été entraîné sur l'ensemble d'entraînement, et ses prédictions ont été évaluées sur l'ensemble de validation. Cette méthode a permis d'obtenir une accuracy de 48,26%, démontrant une légère amélioration par rapport aux modèles individuels pris séparément.

**CatBoost**

On a utilisé le modèle CatBoost pour gérer les données tabulaires, en particulier pour sa capacité à traiter directement les caractéristiques catégorielles sans nécessiter de codage one-hot. Le modèle a été initialisé avec 500 itérations, une profondeur de 6 et un taux d'apprentissage de 0,1. Après l'entraînement sur l'ensemble d'entraînement, les prédictions ont été effectuées sur l'ensemble de validation. Ce modèle a atteint une accuracy de 48,31%, offrant des résultats légèrement supérieurs à ceux des autres modèles. Son efficacité vient de sa capacité à traiter les données catégorielles tout en minimisant le prétraitement nécessaire.

**KNN :**

On a utilisé le modèle K-Nearest Neighbors (KNN) pour la classification en prenant en compte les cinq voisins les plus proches pour déterminer la classe d'une instance. Après l'initialisation du modèle avec n_neighbors=5, il a été entraîné sur l'ensemble d'entraînement. Les prédictions ont ensuite été effectuées sur l'ensemble de validation. Ce modèle a atteint une accuracy de 41,21%. Bien que les performances ne soient pas aussi élevées que celles d'autres modèles, KNN reste une méthode simple et efficace pour les jeux de données où les relations entre les instances sont relativement proches.

## Analyse du Score:

Les performances des différents modèles ont été comparées à l'aide de leur précision (accuracy) sur les données de validation. Le modèle **CatBoost** a obtenu le meilleur score avec une précision de **48,31%**, suivi de près par le **Random Forest** avec **48,22%** et le **Gradient Boosting Classifier** optimisé avec **48,07%**. La **régression logistique** a montré une précision légèrement inférieure, à **47,92%**, tandis que le **K-Nearest Neighbors (KNN)** a obtenu **41,21%**. Enfin, le modèle **XGBoost** a obtenu le score le plus faible, avec une précision de **33,39%**, indiquant qu'il nécessite des ajustements supplémentaires pour être compétitif. Ces résultats

mettent en évidence les forces des modèles d'ensemble comme CatBoost et Random Forest dans ce contexte de classification multiclasse.

### Résultats de soumissions :

Les résultats des soumissions des différents modèles sur les données de test révèlent des variations notables en termes de précision. Le modèle Ensemble Methods, combinant Random Forest et Gradient Boosting, a obtenu la meilleure performance avec une accuracy de 0.40917, suivi de près par le modèle Random Forest avec une accuracy de 0.40586. Ces résultats indiquent que l'approche par ensemble améliore la robustesse des prédictions par rapport aux modèles individuels. En revanche, Logistic Regression a montré une précision similaire à celle de Random Forest, atteignant 0.40523, ce qui montre que, bien que simple et interprétable, elle ne surpasse pas les autres modèles plus complexes. Les Neural Networks et KNN ont obtenu les performances les plus faibles, avec des accuracies respectives de 0.2241 et 0.218779, suggérant qu'ils ne sont pas adaptés aux caractéristiques des données dans ce cas particulier. Enfin, CatBoost a également montré une performance modérée, avec une accuracy de 0.38174, se classant au-dessus des réseaux de neurones et du KNN, mais restant en dessous des meilleurs modèles.

## Conclusion : Comparaison des Résultats de Team Data et Player Data

L'analyse des résultats obtenus à partir des données des équipes (`team data`) et des joueurs (`player data`) met en lumière l'importance du choix des données et des méthodes pour prédire les résultats des matchs de manière optimale.

1. **Performance des Données d'Équipe (Team Data) :** Après avoir soumis les prédictions sur les données de test, le modèle **CatBoost** a donné les meilleurs résultats avec une **accuracy de 48,8%**. Ce modèle a surpassé les autres, notamment grâce à sa capacité à gérer efficacement les données tabulaires et à traiter les caractéristiques catégorielles sans nécessiter de prétraitement complexe. Son efficacité s'explique par la gestion directe des variables catégorielles et son aptitude à capturer des patterns complexes dans les données des équipes. Le **Voting Classifier**, combinant Random Forest et Gradient Boosting, a également montré de bons résultats (48,26%), mais il n'a pas surpassé CatBoost.
2. **Performance des Données de Joueur (Player Data) :** Les modèles utilisant les données des joueurs ont eu une performance globalement plus faible. Le **XGBClassifier**, malgré des réglages optimisés, a obtenu une accuracy de 47,1%, ce qui reste inférieur aux modèles utilisant les données des équipes. L'impact des données manquantes et des biais dans les prédictions (comme la préférence pour les victoires à domicile) a limité la performance des modèles. Les modèles comme **Random Forest** et **Logistic Regression** ont montré des résultats similaires à ceux du XGBClassifier, mais toujours en dessous des meilleures performances observées avec les données d'équipe.

## Comparaison des Meilleures Méthodes :

- **Meilleure Performance avec Team Data :** Le modèle **CatBoost** a obtenu la meilleure performance avec une accuracy de 48,8%, se distinguant par sa gestion des variables catégorielles et sa capacité à minimiser le besoin de prétraitement des données. Bien que le **Voting Classifier** ait également montré une bonne accuracy (48,26%), CatBoost a surclassé les autres modèles en termes de robustesse et d'efficacité.
- **Meilleure Performance avec Player Data :** Le modèle **XGBClassifier** a donné les meilleurs résultats pour les données des joueurs, atteignant une accuracy de 47,1%. Cependant, les résultats étaient nettement inférieurs à ceux obtenus avec les données d'équipe. D'autres modèles comme **Random Forest** et **Logistic Regression** ont également montré des performances similaires, mais avec une précision légèrement inférieure.

Nous avons focalisé nos efforts sur les **données des équipes** pour la prédiction des résultats de matchs, car elles sont plus stables et cohérentes que les données des joueurs. Les performances d'une équipe dépendent de l'interaction collective, tandis que les données des joueurs peuvent être influencées par des absences et des biais individuels. De plus, les données des équipes sont généralement moins sujettes aux valeurs manquantes et permettent une meilleure généralisation à travers

différentes ligues et matchs. Cela a rendu l'utilisation des données des équipes plus efficace pour obtenir des prédictions précises et robustes.

## Conclusion Générale :

Les données des équipes se sont révélées être la meilleure source pour prédire les résultats des matchs, avec une précision notablement plus élevée que celle des données des joueurs. Le modèle CatBoost a été le plus performant avec les données des équipes, atteignant une accuracy de 48,8%, tandis que les modèles basés sur les données des joueurs ont présenté des performances inférieures. En conséquence, il est recommandé d'utiliser les données d'équipe pour obtenir des prédictions plus précises, en particulier avec CatBoost, qui a montré des résultats exceptionnels. Les données des joueurs peuvent ajouter de la valeur dans certains cas, mais leur impact est limité en raison de la qualité des données disponibles et des défis liés aux données manquantes.