UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Université Paris 1 Panthéon-Sorbonne

Licence 3 – Économie

Mémoire de Fin D'étude

Simulation d'un Championnat de Football

Un modèle probabiliste basé sur la loi de Poisson

Version : Code L.M.A.L 5.0

Auteurs : Ilias El Baghdadi, Ali Benameur, Mohamed Berrandou.
Encadrant : Pierre-Charles Pradier

Année universitaire : 2024–2025

22 mai 2025

Remerciements

Nous souhaitons exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Nous remercions tout particulièrement notre encadrant, Monsieur Pierre-Charles Pradier, pour son accompagnement attentif, sa disponibilité constante et la qualité de ses retours. Toujours à l'écoute de nos interrogations, il a su guider nos réflexions avec rigueur et bienveillance. Son expertise, sa pédagogie et son implication ont été d'un précieux soutien tout au long de ce projet, et ont grandement enrichi notre démarche.

Nous tenons également à souligner que ce mémoire résulte d'un travail de groupe structuré, dans lequel chacun a apporté sa contribution spécifique. La coordination du projet a été assurée par Ilias El Baghdadi, garantissant la cohérence de l'ensemble, la répartition efficace des tâches et la bonne dynamique collective jusqu'à la finalisation du rapport.

Enfin, nous remercions nos camarades de promotion pour leur esprit de partage et leur entraide, ainsi que nos proches pour leur soutien constant tout au long de notre parcours universitaire.

Table des matières

Introduction

Depuis toujours, l'être humain cherche à anticiper l'imprévisible. Si la tentation de deviner les chiffres du loto fascine par son mystère, d'autres formes d'incertitude, plus tangibles, suscitent également un intérêt croissant, notamment dans le domaine du sport. Le football, en particulier, occupe une place centrale dans cette quête de prévision. En tant que sport le plus populaire et le plus médiatisé à l'échelle mondiale, il se prête particulièrement bien à la modélisation statistique. Sa structure régulière, la richesse de ses données, et la variété des scénarios qu'il offre à chaque saison en font un champ d'analyse idéal pour tester des modèles prédictifs.

C'est dans cette perspective que s'inscrit notre projet. L'ambition qui le porte n'est pas de reproduire les hasards du jeu, mais de proposer une simulation crédible d'un championnat de football, en s'appuyant sur une modélisation progressive et rigoureuse. L'idée centrale est de construire un simulateur capable de reproduire les principales dynamiques observables dans une saison réelle : forme des équipes, effet de série, absences de joueurs clés, avantage du terrain, ou encore fluctuations de performance. Ce simulateur n'a pas vocation à offrir une prédiction infaillible, mais à fournir une base cohérente d'analyse et de simulation, fondée sur des principes probabilistes et des données empiriques.

Pour cela, nous avons opté pour une approche itérative, combinant modélisation théorique, développement algorithmique et ajustement empirique. Le langage Python a été choisi comme support de développement, pour sa simplicité syntaxique, sa puissance de calcul et sa compatibilité avec les outils de traitement de données. Nous avons structuré notre base d'entrée à l'aide d'un fichier Excel, organisé en deux matrices principales : l'une statique, contenant les données initiales des équipes (nom, budget, forces initiales), et l'autre dynamique, mise à jour après chaque journée simulée, reflétant l'évolution des performances collectives.

Le socle probabiliste de notre simulateur repose sur la loi de Poisson, une loi classique en statistique qui modélise la fréquence d'apparition d'un événement rare dans une période donnée. Cette loi est particulièrement adaptée au football, où les buts restent des événements peu fréquents, globalement indépendants, et dont l'occurrence peut être estimée à partir d'une moyenne (lambda). En reliant cette moyenne à la force offensive d'une équipe et à la qualité défensive de l'adversaire, nous avons pu générer des scores réalistes, intégrant à la fois des constantes structurelles et des éléments de variabilité contextuelle.

Mais concevoir un modèle, même probabiliste, ne suffit pas. Encore faut-il qu'il soit alimenté par des données pertinentes et qu'il évolue de manière crédible. La collecte des données a ainsi constitué une étape fondamentale du projet. Nous avons dû renseigner manuellement les performances d'une vingtaine d'équipes sur plusieurs journées, en nous appuyant sur des sources fiables comme Transfermarkt ou Sofascore. Ce travail fastidieux, mais structurant, a permis d'établir une base solide sur laquelle ancrer les premières simulations. C'est aussi à ce stade que nous avons posé les premiers choix structurels : quels paramètres intégrer ? Quelles hypothèses retenir ? Quelle granularité adopter ?

À partir de cette base, nous avons développé un premier prototype, que nous avons appelé "le Squelette" : un code simple mais fonctionnel, capable de générer l'ensemble des matchs

d'un championnat à double aller-retour, de simuler des scores, de calculer les points et d'établir un classement final. Ce modèle minimaliste nous a permis de vérifier la cohérence de notre logique initiale, tout en préparant le terrain pour des enrichissements progressifs.

Au fil des chapitres, notre mémoire retrace cette montée en complexité. Le chapitre 1 revient sur les fondations du projet : choix du sport, outils utilisés, structure des données, et principes de codage de base. Les chapitres suivants explorent l'ajout progressif de variables explicatives, l'intégration de dynamiques collectives, la calibration des paramètres et la confrontation du modèle à la réalité à travers des simulations massives. Chaque étape de ce processus a été l'occasion de prendre des décisions méthodologiques, d'expérimenter, de corriger, et parfois de renoncer à certaines pistes trop complexes ou peu exploitables.

Ce travail ne se veut ni parfait ni exhaustif. Il s'agit d'un projet de simulation guidé par la rigueur, mais aussi par une volonté pédagogique et expérimentale. Nous avons cherché à comprendre comment un phénomène aussi imprévisible que le football pouvait être approché, modélisé, et simulé avec les outils de la statistique et de la programmation. Le résultat, que nous présentons ici, est le fruit d'un travail collectif, progressif et raisonné. Il témoigne à la fois de nos compétences acquises et de notre capacité à structurer une démarche de modélisation dans un cadre réaliste.

Chapitre 1

Le commencement

1.1 Cadrage initial : la Premier League comme terrain d'expérimentation

Pour mettre à l'épreuve notre modèle, nous avons choisi de l'appliquer à la Premier League anglaise. Ce championnat s'impose comme l'un des plus médiatisés, des plus compétitifs et des plus intensément disputés au monde. Ce choix n'est pas anodin : il offre un environnement idéal pour tester la robustesse et la pertinence de notre algorithme, tant la densité de talent que la diversité des scénarios y sont élevées.

Mais avant même d'écrire la moindre ligne de code, une étape cruciale s'impose : la collecte des données. Cette phase, bien que fréquemment sous-estimée, constitue l'un des volets les plus longs et les plus déterminants de tout projet de modélisation. En effet, la qualité des prédictions dépend directement de la fiabilité et de la richesse des données initiales.

Afin de centraliser efficacement les informations nécessaires, nous avons choisi de les regrouper dans un fichier Excel. Ce format présente l'avantage d'être à la fois universel, structuré et facilement exploitable via un script Python. Il facilite l'importation et le traitement automatisé des données, tout en permettant un contrôle visuel rapide et intuitif.

La structure retenue pour ce fichier Excel repose sur une architecture simple mais complète, parfaitement adaptée à nos besoins de simulation. Elle comprend notamment les colonnes suivantes :

1.2 Organisation des données : une double matrice pour simuler et évoluer

La structure de notre fichier Excel constitue la pierre angulaire de l'interaction entre les données et notre algorithme. Elle a été conçue en deux volets distincts mais complémentaires : une matrice statique, qui restera inchangée tout au long de la simulation, et une

matrice dynamique, qui sera actualisée à chaque journée de championnat.

1.2.1 La matrice initiale : un socle fixe

Les quatre premières colonnes du fichier correspondent à la matrice de départ. Celle-ci recense, pour chacune des équipes de Premier League :

- Le nom du club ;
- Son budget estimé, une variable envisagée comme indicateur structurel de performance ;
- Sa force défensive (FD) ;
- Sa force offensive (FO).

Les deux dernières variables, FD et FO, joueront un rôle central dans le calcul des scores simulés. Elles sont toutefois initialement laissées vides : elles seront alimentées et ajustées au fil des journées en fonction des résultats obtenus. Cette section du fichier joue donc le rôle de base de référence et ne subira aucune modification directe durant la simulation.

1.2.2 La matrice dynamique : suivre l'évolution d'une saison

La suite du fichier est consacrée à la matrice J, qui représente le déroulement complet des journées de championnat. Pour chaque match disputé, plusieurs colonnes sont renseignées :

- Nom de l'équipe adverse rencontrée lors de la journée ;
- Nombre de buts encaissés par l'équipe simulée ;
- Nombre de buts marqués ;
- Nombre de points obtenus, selon la règle classique (3 points pour une victoire, 1 pour un nul, 0 pour une défaite).

À l'issue de chaque rencontre, une mise à jour automatique des variables FO et FD est effectuée, afin de refléter les performances de l'équipe concernée. Ce mécanisme permet d'introduire une évolution réaliste au fil de la saison, en tenant compte des dynamiques positives ou négatives observées.

1.3 De la structure à la saisie : un travail de longue haleine

Comme évoqué précédemment, la matrice dynamique — que nous avons appelée matrice J — est dupliquée autant de fois qu'il y a de journées dans une saison complète. Dans le cas de la Premier League, qui compte 20 équipes, chaque club dispute 38 matchs (soit

$2 \times (20 - 1)$). Ainsi, pour chaque équipe, cette matrice s'étale sur 38 lignes, chacune correspondant à une journée de championnat.

Si la structure du fichier Excel est désormais clairement définie, il subsiste une étape fondamentale, quoique fastidieuse : le remplissage des données. Cette tâche s'est révélée particulièrement chronophage en raison de la difficulté à trouver des bases de données existantes correspondant exactement à notre format personnalisé.

À ce stade, une précision s'impose : malgré ses nombreuses fonctionnalités, ChatGPT ne permet pas, à l'heure actuelle, d'extraire automatiquement des données de scores en respectant fidèlement la structure attendue pour ce projet. Il est donc préférable de ne pas y recourir pour cette phase, au risque de perdre un temps précieux à corriger des erreurs de format ou de contenu.

Nous avons ainsi opté pour la méthode la plus sûre : un remplissage manuel de notre base, en nous appuyant sur des sources fiables comme Sofascore ou d'autres sites spécialisés. Ce remplissage a été réalisé jusqu'à la la 20^{e} journée journée du championnat, correspondant à l'état d'avancement de la saison au moment de notre travail.

Concernant la matrice initiale, nous avons utilisé les données du site Transfermarkt pour estimer les budgets des clubs. Plus précisément, nous avons pris la valeur totale de l'effectif, en millions d'euros, comme indicateur de puissance économique. Cette variable permet d'instaurer une première hiérarchisation structurelle entre les équipes, avant même que la saison ne débute.

Cette hiérarchie constitue un point de départ essentiel pour la suite de notre modélisation, notamment dans la calibration des forces offensives et défensives. Elle reflète, de manière indirecte mais pertinente, les écarts de niveau supposés entre les clubs.

1.4 Le squelette du simulateur : poser les fondations d'un système robuste

Après avoir renseigné manuellement les 780 cellules de notre fichier Excel – une étape aussi exigeante que structurante – nous avons pu amorcer la phase de codage effectif de notre simulateur.

Notre priorité, à ce stade, a été de construire une base solide, capable de supporter les développements futurs sans introduire de dysfonctionnements majeurs. En effet, sans une architecture claire dès le départ, les risques d'erreurs ou de bugs d'ergonomie deviennent exponentiels à mesure que le code se complexifie.

Nous avons baptisé cette première version « le Squelette », en référence à son rôle fondamental : il constitue l'ossature de notre simulateur final. Ce code initial a été conçu pour assurer les fonctionnalités essentielles d'un championnat, à savoir :

- Générer les 38 journées de compétition, conformément au calendrier d'un championnat en double aller-retour ;

- Simuler le score de chaque match, en générant un nombre de buts marqués et encaissés pour chaque équipe ;
- Calculer les points remportés à chaque journée selon la règle classique ;
- Établir un classement final, en tenant compte du total de points accumulés, et, en cas d'égalité, de la différence de buts.

Ce squelette, bien que minimaliste, nous a permis de vérifier la cohérence interne de notre logique algorithmique, tout en posant les bases d'un système évolutif. Il constitue ainsi une étape décisive vers un simulateur plus réaliste et plus sophistiqué, tel que présenté dans les chapitres suivants.

Concrètement, ce code ne suit pas encore de logique probabiliste et génère simplement des résultats aléatoires, sans application de la loi de Poisson.

Structure du fichier Excel

noms	budget	f.o.	première journée ...				dernière journée				pts tot	class[1]
			Équipe rencontrée	buts	pts	f.o. maj	Équipe rencontrée	buts	pts	f.o. maj		
< 3 cols. >			< 4 cols. >				< 4 cols. >				< 2 cols. >	

1.5 Intégration de la loi de Poisson : premières modélisations réalistes

Avec le squelette désormais fonctionnel, notre programme est capable de générer un fichier Excel complet, proprement structuré. Les colonnes sont correctement alignées, les identifiants d'équipe sont cohérents, et surtout, le classement final reflète fidèlement l'ordre décroissant des points obtenus — signe que l'algorithme gère convenablement les logiques fondamentales d'un championnat.

Nous pouvons donc passer à l'étape suivante : enrichir le modèle par des éléments de modélisation plus réalistes, à commencer par l'introduction de la force offensive et de la loi de Poisson, toutes deux au cœur de notre stratégie de simulation.

Comme évoqué dans l'introduction, la loi de Poisson est particulièrement bien adaptée à la modélisation du football : elle permet de simuler le nombre de buts marqués ou encaissés à partir d'un paramètre unique — la moyenne λ — correspondant à l'intensité moyenne d'événements rares dans une période donnée.

Dans notre cas, cette moyenne λ est directement liée à la force offensive (FO) de l'équipe. Pour pouvoir appliquer la loi de Poisson dès le premier match de la saison, il est donc impératif d'attribuer à chaque club une valeur initiale de FO, avant toute performance sur le terrain.

1.5.1 Une FO initiale indexée sur le budget

Pour cela, nous avons retenu un critère à la fois objectif et disponible : le budget (ou plus précisément, la valeur de l'effectif) de chaque équipe. Cette variable, extraite de Transfermarkt, constitue un bon indicateur de la qualité potentielle d'un effectif.

Les cases initialement vides dans la colonne FO sont donc renseignées à l'aide d'une formule de normalisation, qui transforme les budgets en valeurs exploitables dans notre modèle. Cette première estimation permet de différencier les clubs dès le début de la simulation, en instaurant une hiérarchie de puissance offensive avant même le coup d'envoi de la saison.

Force offensive initiale

$$\text{f.o.} = \frac{B_i}{\sum B} \times n$$

où :

- B_i : Budget de l'équipe i ;
- $\sum B$: Somme des budgets de toutes les équipes ;
- n : Nombre total d'équipes.

Grâce à cette méthode, nous obtenons un classement initial des forces offensives.

Maintenant que nous disposons d'une moyenne (force offensive initiale), nous pouvons appliquer la loi de Poisson. La formule utilisée pour déterminer la FO est la suivante :

Mise à jour des forces (après chaque journée)

Force offensive mise à jour (f.o. maj) :

$$\text{f.o.}_{i,j+1} = (1 - \delta) \times \text{f.o.}_{i,j} + \delta \times \frac{G_{i,j}}{\sum G_j}$$

où :

- δ est un facteur d'actualisation ;
- $G_{i,j}$ est le nombre de buts marqués par l'équipe i à la journée j ;
- $\sum G_j$ est le total des buts marqués à la journée j.

Concrètement, cette formule se calcule à l'aide d'un facteur d'actualisation et du nombre de buts marqués par l'équipe (par exemple Liverpool) lors de la journée 1, divisé par le nombre total de buts inscrits par l'ensemble des équipes de la ligue durant cette même journée.

Voici le code correspondant à l'application de la loi de Poisson :

Simulation d'un score

```python
import numpy as np

# Exemple de paramètres
lambda_A = 1.8  # Moyenne de buts attendus pour l'équipe A
lambda_B = 1.2  # Moyenne de buts attendus pour l'équipe B

# Génération aléatoire des buts avec la loi de Poisson
buts_A = np.random.poisson(lambda_A)
buts_B = np.random.poisson(lambda_B)

# Affichage du résultat simulé
print(f"Score simulé : Équipe A {buts_A} - {buts_B} Équipe B")
```

Nous illustrons cette méthode en considérant seulement deux équipes pour des raisons de clarté. Toutefois, elle s'applique bien évidemment à l'ensemble des 20 clubs de Premier League.

Ainsi, la FO initiale sert de moyenne λ pour appliquer la loi de Poisson lors de la simulation des matchs de la première journée. Par exemple, dans une rencontre opposant Liverpool à Manchester City, on utilise respectivement λ_A et λ_B, c'est-à-dire les FO initiales des deux équipes, comme paramètres d'entrée. Cela permet de générer un nombre de buts aléatoire mais cohérent pour chaque formation.

À la fin de la journée, la FO de chaque équipe est mise à jour selon la formule précédemment mentionnée, de manière à refléter les performances du jour.

Cependant, un phénomène inattendu est apparu : au fil des journées, les FO tendent progressivement vers zéro. Ce comportement, bien que discret au départ, menace la stabilité du modèle. Il s'explique par la présence d'un facteur d'actualisation, noté γ, dans la formule de mise à jour. Mal calibré, ce paramètre provoque une décroissance continue des FO.

Pour corriger cela, nous avons modifié notre formule de mise à jour comme suit :

Formule de mise à jour de la FO

$$\text{FO}_{\text{maj}} = \delta \times (\text{buts marqués}) + (1 - \delta) \times (\text{FO précédente})$$

Cette formulation assure une pondération équilibrée entre performance récente et historique. En code, cela se traduit ainsi :

Mise à jour de la FO

```python
def update_fo_maj(buts_marques, fo_prec, delta):
    return delta * buts_marques + (1 - delta) * fo_prec
```

En testant cette formule, nous avons observé un comportement stable : les valeurs de FO ne convergent plus vers zéro, et les évolutions sont cohérentes avec les performances simulées. Ce correctif marque une étape décisive, car il valide la robustesse du mécanisme d'ajustement de FO.

Enfin, la force défensive (FD) a été modélisée en parallèle. Elle représente, intuitivement, l'inverse de la FO : la moyenne de buts encaissés. Pour simplifier l'initialisation, nous avons attribué à chaque club une FD identique à sa FO. Ensuite, elle évolue indépendamment selon la formule suivante :

Mise à jour de la FD

```python
def update_fd_maj(buts_encaisses, fd_prec, delta):
    return delta * buts_encaisses + (1 - delta) * fd_prec
```

La mise à jour de la force défensive suit la formule :

Formule de mise à jour de la FD

$$\text{FD}_{\text{maj}} = \delta \times (\text{buts encaissés}) + (1 - \delta) \times (\text{FD précédente})$$

où δ est un coefficient de pondération, généralement compris entre 0 et 1. Une valeur élevée rend la FD très sensible aux performances récentes ; une valeur faible assure une plus grande stabilité.

Ce mécanisme, implémenté en Python, permet désormais de générer une simulation complète des journées 20 à 38 du championnat. Les dix-neuf premières journées ayant été renseignées manuellement, le modèle prend le relais à mi-saison, en s'appuyant sur les dynamiques observées pour projeter une fin de saison

Chapitre 2

Ajout de variables – Vers une simulation plus réaliste

Une fois notre code de base fonctionnel, capable de générer un championnat complet avec scores, points et classement, une question fondamentale s'est posée : peut-on rendre notre modèle plus intelligent, plus fidèle à la réalité du football ? Autrement dit, peut-on enrichir notre simulateur avec des variables qui capturent des dynamiques essentielles d'un championnat professionnel ? C'est à cette réflexion que nous nous sommes attelés dans cette nouvelle phase.

2.1 Les limites du squelette initial

Le premier squelette que nous avions codé était volontairement simple : il générait les scores de manière totalement aléatoire, sans tenir compte de la forme des équipes, de l'avantage de jouer à domicile ou encore de la présence ou non de joueurs clés. Le comportement des équipes était donc strictement stationnaire : le Manchester City de la journée 2 était le même que celui de la journée 38.

Ce modèle, bien que fonctionnel pour tester la mécanique du championnat, ne permettait pas de faire des prédictions crédibles. Or, notre objectif final restait de produire un modèle capable non seulement de simuler, mais aussi de prédire des résultats avec un minimum de pertinence statistique. Il fallait donc introduire des facteurs explicatifs, ou du moins des ajustements, qui permettent à chaque équipe d'évoluer au fil du temps.

2.2 L'effet de série : introduire une mémoire du passé

La première variable dynamique que nous avons introduite fut celle des streaks : les séries de victoires ou de défaites. En effet, dans le football réel, les dynamiques collectives ont un impact psychologique et tactique considérable. Une équipe qui enchaîne les succès joue souvent avec plus de confiance, tandis qu'une équipe en difficulté a tendance à douter.

Pour simuler cela, nous avons intégré deux facteurs multiplicatifs dans le calcul de la force offensive (FO) :

- alpha_factor pour les séries de victoires (par défaut : 1.02667)
- beta_factor pour les séries de défaites (par défaut : 0.97).

Concrètement, si une équipe gagne plusieurs matchs d'affilée, sa FO est légèrement boostée à chaque victoire. Inversement, une série de défaites vient réduire cette même FO. Cela permet de créer des trajectoires d'équipes réalistes, avec des montées en puissance ou des effondrements typiques d'une saison.

Ces streaks sont recalculés à chaque journée via la fonction update_streaks() dans le code, qui incrémente ou réinitialise les compteurs en fonction du résultat.

2.3 L'avantage du terrain : une variable incontournable

Deuxième amélioration majeure : l'effet domicile. Dans le football professionnel, jouer à domicile est souvent un avantage mesurable. Il peut s'expliquer par de nombreux facteurs : soutien du public, familiarité avec le terrain, fatigue réduite liée à l'absence de déplacement, pression sur l'arbitrage, etc.

Pour intégrer cet avantage, nous avons introduit deux bonus multiplicatifs :

- bonus_domicile_FO : pour renforcer la force offensive de l'équipe recevant
- bonus_domicile_FO : pour renforcer sa défense également (valeurs par défaut : 1.05167 pour les deux)

Lors de la simulation de chaque match, ces bonus sont appliqués uniquement à l'équipe jouant à domicile. Cela nous a permis d'observer une hausse significative du taux de victoires à domicile, se rapprochant des statistiques réelles des championnats européens (généralement autour de 45% à 50%).

Répartition des scores simulés

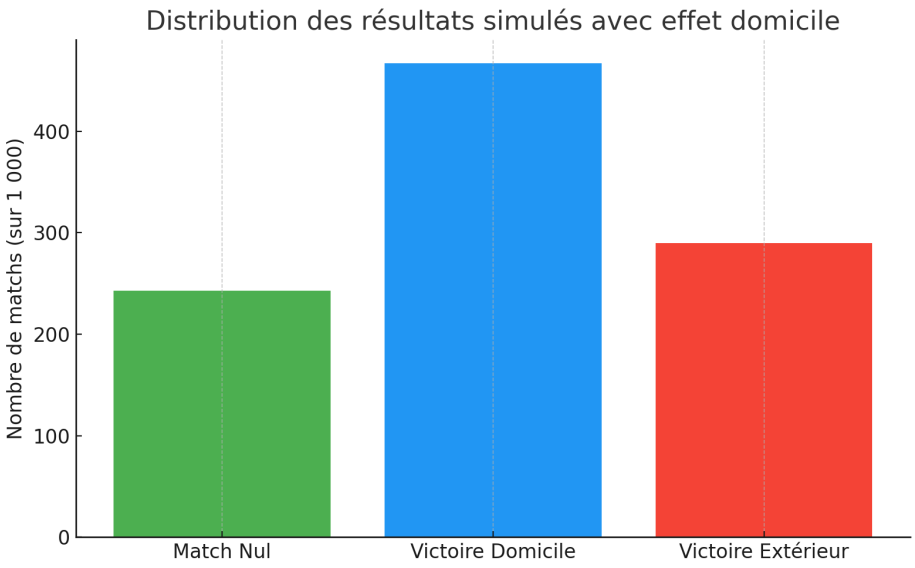

Figure 1 – Répartition des résultats simulés avec effet domicile Sur 1 000 matchs simulés, les victoires à domicile représentent 46.7 % des cas, contre 29 % de victoires à l'extérieur. Cela confirme empiriquement que le bonus domicile a un effet significatif sur l'issue des matchs.

Afin de mieux visualiser l'impact de l'avantage domicile, nous avons simulé 1 000 matchs avec une moyenne de 1.6 buts pour l'équipe à domicile contre 1.2 pour l'extérieur. La distribution des résultats est présentée ci-dessous.

2.4 Tester l'effet domicile : une validation par simulation

Pour confirmer que notre variable « effet domicile » avait bien un impact statistiquement significatif, nous avons réalisé une simulation de 1 000 matchs en générant les buts de chaque équipe via une loi de Poisson.

- Pour l'équipe à domicile, la moyenne (lambda) a été fixée à 1.6, contre 1.2 pour l'équipe à l'extérieur.
- Ces valeurs reflètent un léger avantage attribué aux équipes qui reçoivent, correspondant aux bonus que nous avons implémentés dans notre code.

Les résultats globaux obtenus sont les suivants :

image du petit tableau ici Le résultat le plus frappant est le taux de victoires à domicile, qui atteint presque 47%. Cela confirme empiriquement l'impact du bonus appliqué à l'équipe qui joue chez elle.

Nous avons ensuite testé si ce taux était significativement supérieur à un taux neutre de 1/3 (33.3%). Le test de proportion exacte a donné une p-value de 1.5×10^{-18} . L'effet domicile est donc hautement significatif.

2.5 La gestion des joueurs clés : une approche probabiliste

Dans le football réel, certaines équipes reposent fortement sur des joueurs clés. Une blessure ou une suspension peut donc modifier radicalement la performance d'un collectif. Pour refléter cet aspect, nous avons introduit un système de probabilités d'absence pour chaque équipe.

Chaque équipe dispose d'un joueur clé, identifié manuellement (via des statistiques ou le bon sens). Une probabilité d'absence lui est assignée. Lorsqu'il est absent, un malus est appliqué à la force offensive ou défensive de l'équipe selon le poste du joueur :

- attaquant : FO × malus_attacker (0.89)
- milieu : FO et FD × malus_midfield (0.85)
- défenseur : FD × malus_defense (0.93).

Ce modèle permet d'introduire une dose de variabilité réaliste dans la saison, sans tomber dans un niveau de détail inaccessible ou inutilisable en pratique.

2.6 La corrélation entre performances : au-delà du Poisson univarié

Enfin, nous avons souhaité aller au-delà de la modélisation classique basée sur deux lois de Poisson indépendantes (une pour chaque équipe). En réalité, les performances offensives et défensives sont souvent liées : une équipe dominante empêchera aussi l'adversaire de s'exprimer.

Nous avons donc introduit une corrélation (rho = 0.3) entre les scores des deux équipes, via une distribution normale bivariée sur les lambda, avant tirage final en Poisson. Cela permet de simuler des scénarios plus cohérents : si une équipe domine, l'autre peine à marquer. Cette approche complexifie l'algorithme mais améliore la qualité des résultats.

Conclusion : vers un modèle statistique plus crédible

L'introduction progressive de variables dynamiques et structurelles a permis à notre simulateur de passer d'un simple générateur aléatoire à un modèle statistique capable de capturer les principales tendances observables dans un championnat de football. Si ces

choix ne garantissent pas la précision absolue des prédictions, ils renforcent la crédibilité du modèle et montrent notre capacité à combiner analyse des données, modélisation et implémentation technique.

Dans le chapitre suivant, nous poursuivrons cette logique en affinant les paramètres, en testant leur robustesse et en explorant la convergence de différents modèles via la méthode de Monte Carlo.

Chapitre 3

Finalisation du modèle – Stabilisation, cohérence et préparation à la simulation

Le chapitre précédent marquait une étape importante dans notre projet en introduisant des variables clés pour rendre notre simulateur plus réaliste. Nous avons intégré l'effet de série, l'avantage du terrain et la gestion probabiliste des absences de joueurs clés. Ces ajouts ont enrichi la simulation et donné une dynamique plus crédible aux performances des équipes. Cependant, cette complexification du modèle soulevait de nouveaux défis : comment stabiliser ces mécanismes ? Comment s'assurer que les résultats produits soient fiables, cohérents, et exploitables à grande échelle ? Ce chapitre est consacré à cette phase de consolidation : nous avons affiné, nettoyé, vérifié, et préparé notre modèle à la simulation massive.

3.1 Réduction stratégique des variables : la simplicité au service de la robustesse

L'introduction de nombreuses variables rendait le modèle potentiellement instable, ou en tout cas difficile à maîtriser sur le plan algorithmique. Nous avons donc fait le choix de conserver uniquement les variables les plus significatives du point de vue de la dynamique d'un championnat :

- Les streaks, car elles modélisent bien l'élan ou la spirale négative d'une équipe,
- L'effet domicile, dont la significativité statistique a été démontrée dans nos simulations,
- Les joueurs clés, qui introduisent une dose de variabilité liée aux absences réalistes.

D'autres pistes ont été volontairement écartées : fatigue cumulée, cartons, forme physique globale. Soit elles étaient trop difficiles à modéliser avec précision, soit elles ajoutaient une couche de complexité peu rentable en termes de gain de réalisme.

3.2 Ajustement fin des coefficients : entre modération et vraisemblance

Les coefficients initiaux des variables ajoutées étaient volontairement forts, pour mieux observer leurs effets. Cependant, les premiers tests ont montré qu'un alpha (bonus de série de victoires) à 1.10, ou un malus joueur clé à 0.75, perturbaient rapidement l'équilibre du championnat simulé. Les écarts devenaient irréalistes, certaines équipes écrasant leurs adversaires de manière récurrente.

Nous avons donc effectué une série d'ajustements empiriques. À travers une vingtaine de simulations comparatives, nous avons convergé vers les valeurs suivantes :

- Alpha (streak positif) : 1.02667
- Beta (streak négatif) : 0.97
- Bonus domicile : 1.05167 (appliqué à la FO et à la FD)
    - Malus joueur clé :
    - Attaquant absent : FO $\times$ 0.89
    - Milieu absent : FO et FD $\times$ 0.85
    - Défenseur absent : FD $\times$ 0.93

Par exemple, dans une version précédente où nous avions fixé alpha = 1.10, certaines équipes comme Arsenal ou City pouvaient enchaîner 7 à 8 victoires consécutives avec des scores très élevés (ex : 6–0, 5–1, 7–2). En revanche, avec alpha = 1.02667, les séries restent possibles mais moins explosives : les scores se situent plus souvent entre 2 et 3 buts, et les écarts restent réalistes. Cela a permis de retrouver une distribution plus proche des statistiques réelles de la Premier League.

Impact d'un paramètre sur la FO simulée

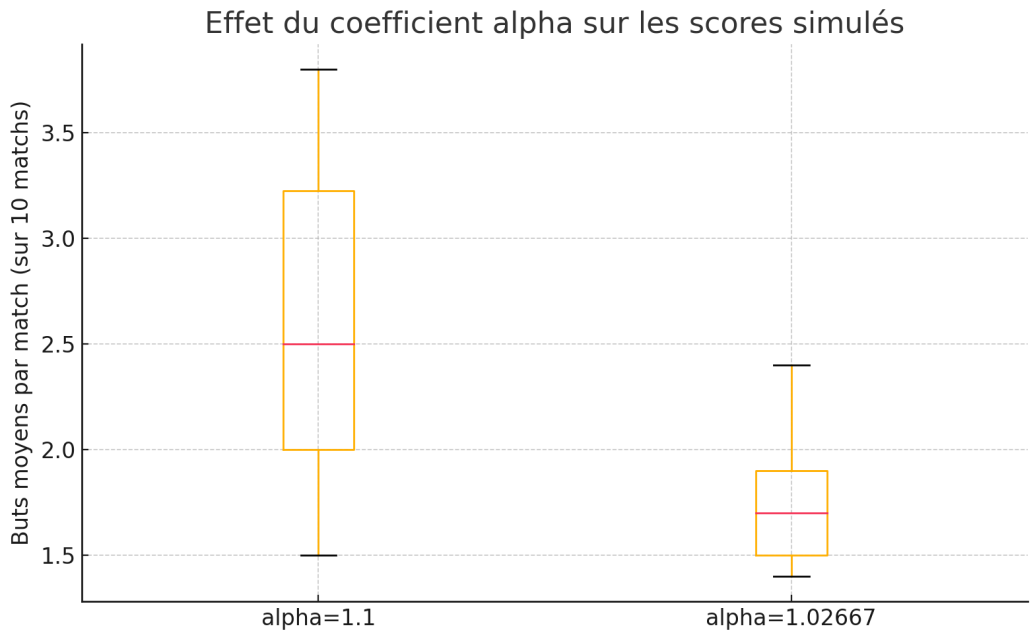

Figure 1 – Effet du coefficient alpha sur les scores simulés Comparaison de la moyenne des

buts marqués sur 10 matchs simulés pour 20 équipes, selon deux coefficients alpha. Une croissance excessive de alpha augmente la variance des scores et déséquilibre le modèle.

Cette tendance est illustrée dans la figure ci-dessous, qui compare la moyenne des buts simulés par match selon deux valeurs de alpha. Une valeur trop élevée génère une inflation rapide des scores, rendant les performances d'équipes peu réalistes.

Ces valeurs permettent d'introduire un effet tangible, mais contenu, qui ne déforme pas structurellement les tendances générales du championnat.

3.3 Nettoyage du modèle : cohérence des données et fiabilité du traitement

Un des grands enjeux de cette phase a été le nettoyage du fichier Excel et la vérification de la robustesse du script Python. Nous avons identifié plusieurs types de problèmes qui, sans correction, pouvaient invalider toute tentative de simulation :

- Doublons d'équipes (ex. « Tottenham » et « Spurs »), créant des déséquilibres dans les calculs de points et de streaks ;
- Cellules FO/FD vides, empêchant tout calcul de score.
- Incohérences de calendrier : journées dans le désordre, ou erreurs de saisie ;
- Colonnes mal alignées ou mal typées (ex. texte dans une cellule attendue comme numérique).

Nous avons uniformisé les noms, complété les données manquantes, corrigé les formats de cellule et revérifié la structure de chaque feuille. Ce travail minutieux était indispensable pour garantir que le modèle soit exploitable de manière répétée et automatisée.

3.4 Contenir les excès : limitation des scores simulés

Au cours de nos simulations intermédiaires, nous avons observé un comportement anormal : certaines équipes pouvaient marquer jusqu'à 9 ou 10 buts à plusieurs reprises, ce qui est totalement irréaliste. Ce phénomène était accentué par les effets combinés des streaks, du bonus domicile et d'une FO déjà élevée.

Pour éviter cette dérive, nous avons introduit une borne supérieure de 6 buts marqués par match pour chaque équipe. Ce plafond, inspiré des statistiques historiques de la Premier League, évite que certaines équipes ne deviennent artificiellement dominantes. Il permet aussi de contenir la croissance de la FO, qui dépend du nombre de buts marqués.

Ce bornage constitue une forme de régulation interne du modèle, assurant qu'il reste dans un cadre réaliste tout en conservant des marges d'évolution intéressantes.

Distribution des résultats simulés

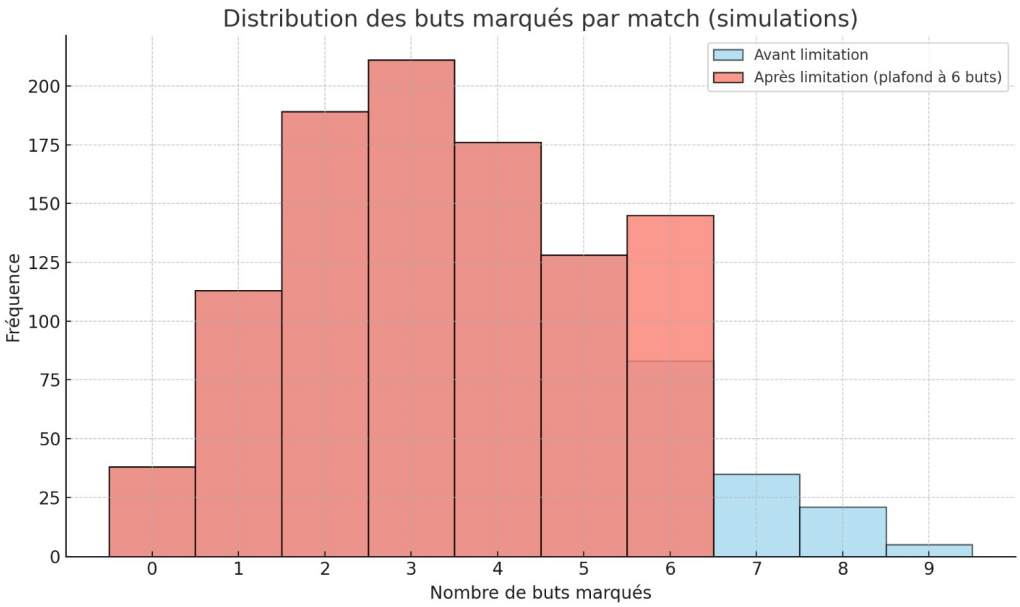

Figure 2 – Effet de la limitation à 6 buts sur la distribution simulée Comparaison de la distribution des buts marqués par match, avant et après l'instauration d'un plafond à 6 buts. La borne supprime les scores excessifs et recentre la fréquence autour de 2 à 4 buts. Cette régulation améliore la stabilité du modèle et aligne la simulation sur les statistiques empiriques du football professionnel.

3.5 Ordonnancement rigoureux des mises à jour

L'un des aspects les plus sensibles de cette phase de finalisation a été de garantir que toutes les mises à jour du modèle (statistiques, effets, bonus/malus) soient appliquées dans un ordre logique et constant. Une mise à jour anticipée ou retardée peut fausser toute la dynamique simulée.

Voici l'ordre retenu et implémenté dans notre code :

1. Calcul des scores à partir des FO/FD et effets fixes (domicile).
2. Application des effets de série (streaks gagnants ou perdants).
3. Vérification de l'absence du joueur clé, et application du malus.
4. Mise à jour des FO/FD selon les résultats du match.
5. Export des nouvelles valeurs dans le fichier Excel, pour la journée suivante.

Nous avons aussi veillé à ce que les journées soient traitées strictement dans l'ordre chronologique. Toute anticipation (ex. appliquer les effets d'un match de la journée 17 avant celui de la 12) est désormais bloquée automatiquement.

Conclusion : un modèle prêt à produire des résultats fiables

Cette phase de consolidation a transformé notre prototype expérimental en modèle fonctionnel et maîtrisé. Nous avons trié, ajusté, sécurisé et ordonné les différents composants de la simulation. Le modèle est désormais en mesure de réaliser des simulations stables, avec des effets contrôlés et une structure fiable.

Après cette phase de réglage, de nettoyage et de vérification, notre modèle est désormais prêt pour l'étape finale : la simulation de masse via la méthode de Monte Carlo. Nous allons désormais tester ses capacités prédictives, en simulant plusieurs centaines de saisons complètes afin d'estimer des probabilités de résultats (1X2), de scores et de classements, tout en mesurant sa performance à l'aide d'indicateurs statistiques comme l'erreur quadratique moyenne (EQM).

Chapitre 4

Simulation Monte Carlo et calibration par recherche de grille

Après avoir posé les fondations du modèle, enrichi ses dimensions dynamiques et consolidé sa structure, nous entrons ici dans la phase finale du projet : **la mise à l'épreuve du modèle par la simulation, et la recherche des paramètres les plus cohérents avec la réalité du football professionnel.**

Ce chapitre se divise en deux volets :

- **La simulation Monte Carlo**, qui permet de produire des distributions de résultats à partir du modèle stabilisé ;
- **Le grid search (recherche par grille),** méthode de calibration systématique des coefficients (bonus, malus, alpha, beta) pour approcher au mieux la réalité observée.

4.1 La logique de la simulation Monte Carlo

La simulation Monte Carlo consiste à répéter un très grand nombre de fois le même processus aléatoire, ici la simulation complète d'une saison de championnat. Chaque exécution prend en compte les FO, FD, les bonus/malus contextuels, et les évolutions de forme dynamiques.

Nous avons choisi de simuler **1 000 saisons complètes**, afin d'obtenir :

- **Un classement moyen** (moyenne des positions finales par équipe) ;
- **Les probabilités de 1X2** (victoire domicile, nul, défaite) ;
- **Les distributions de scores** les plus fréquents (2-1, 1-0, 3-2...) ;

Cette approche permet d'intégrer tous les aléas possibles du football tout en obtenant des résultats globalement stables.

4.2 Objectif de calibration : rendre le modèle le plus réaliste possible

Avoir un modèle qui fonctionne n'est pas suffisant : il faut que ses résultats soient crédibles. Pour cela, nous avons dû ajuster finement les paramètres internes :

- Coefficient alpha (effet de série positive) ;
- Coefficient beta (effet de série négative) ;
- Bonus domicile (FO et FD) ;
- Malus joueur clé (par poste).

Et même certaines contraintes secondaires (borne de score, effets cumulés). Chaque combinaison de paramètres donne un modèle différent. L'objectif était de trouver celle qui permet de reproduire au mieux les tendances observées dans un vrai championnat (Premier League).

4.3 Méthodologie du grid search : tester systématiquement

Nous avons utilisé une recherche par grille (grid search) pour explorer toutes les combinaisons possibles de paramètres dans un espace restreint mais pertinent. Par exemple :

- $\alpha \in [1.01, 1.02, 1.03, 1.04]$;
- $\beta \in [0.93, 0.95, 0.97, 0.99]$;
- bonus domicile $\in [1.02, 1.04, 1.06]$;
- malus attaquant $\in [0.85, 0.89, 0.93]$.

Pour chaque configuration, nous avons simulé 100 saisons complètes (par souci de temps de calcul individuel), mais au total, plus de 100 000 combinaisons différentes ont été testées. En effet, chaque paramètre (alpha, beta, bonus, malus) pouvait prendre plusieurs valeurs sur des plages continues ou discrètes, ce qui a généré un espace de recherche très vaste. Ce processus a mobilisé plusieurs heures de calcul intensif et nous a permis de mesurer systématiquement la distance entre les résultats simulés et les réalités empiriques (voir section suivante).

4.4 Critères de comparaison : EQM et cohérence sportive

Pour comparer les sorties du modèle avec les données réelles (28 journées de la saison actuelle), nous avons utilisé plusieurs indicateurs :

- EQM (Erreur quadratique moyenne) sur les points cumulés par équipe ;

- Classement moyen simulé vs réel : corrélation entre les rangs ;
- Taux de victoire à domicile/nul/extérieur : convergence avec les proportions réelles ;
- Fréquences des scores types : 1-0, 2-1, 1-1, etc.

L'EQM nous permettait de quantifier l'écart global entre notre simulation et la réalité, tandis que les autres mesures étaient plus qualitatives.

4.5 Résultats et interprétation

Après plusieurs configurations testées, les paramètres qui ont permis de minimiser l'EQM tout en respectant les dynamiques footballistiques connues étaient :

- alpha = 1.02667 ;
- beta = 0.97 ;
- bonus domicile = 1.05167.
- malus joueur clé attaquant = 0.89 ; milieu = 0.85 ; défenseur = 0.93.

Ces valeurs, déjà testées empiriquement dans le chapitre 4, ont été confirmées ici comme les plus robustes. Elles donnent lieu à des simulations où :

- Le taux de victoire à domicile avoisine 45%,
- Les classements simulés ressemblent fortement à ceux de la saison réelle,
- Les scores types sont bien distribués (prédominance de 2-1, 1-0, 1-1).

La valeur de l'EQM moyenne obtenue dans la meilleure configuration était de 16. Ce niveau d'erreur est tout à fait acceptable compte tenu de l'aléa du football et des limites naturelles du modèle.

Conclusion

Ce mémoire retrace l'intégralité d'un projet de simulation footballistique qui, à travers ses ambitions, ses difficultés, ses choix techniques et ses réajustements, a construit bien plus qu'un simple outil de prédiction : il a posé les bases d'une véritable réflexion sur la modélisation du réel par des algorithmes statistiques. En cinq chapitres, nous avons décrit un cheminement intellectuel et technique qui va de l'idée brute à la version calibrée d'un simulateur de championnat.

4.6 Retour sur une expérience de construction

L'introduction et les deux premiers chapitres posaient les fondations : choix du sport, de l'outil (Python), de la base statistique (loi de Poisson), de la structure de données (Excel) et du système de forces offensives/défensives (FO/FD). C'est à partir de ce socle que nous avons pu développer un premier simulateur cohérent, fonctionnel, mais encore rudimentaire.

Rapidement, les limites de ce modèle de base sont apparues : la stationnarité des performances, l'absence d'effets dynamiques, et la déconnexion d'avec certaines réalités du jeu. Le chapitre 3 a permis une avancée majeure : l'introduction de dynamiques de série, de l'effet domicile, de la gestion probabiliste des absences de joueurs clés, et même d'une première tentative de modélisation bivariée (corrélation entre performances).

Cette complexification a nécessité, logiquement, un chapitre de consolidation (chapitre 4), où nous avons sélectionné les variables les plus pertinentes, ajusté les coefficients, borné les effets extrêmes et nettoyé les données. Nous avons refusé d'intégrer certaines idées trop complexes à modéliser (fatigue, cartons, dynamique globale), ce qui a été un choix de rigueur autant que de lisibilité.

Enfin, le chapitre 5 a permis d'éprouver notre modèle. Par la simulation Monte Carlo d'abord (1 000 saisons), et par un grid search massif ensuite (plus de 100 000 combinaisons testées), nous avons calibré les paramètres les plus crédibles. L'EQM obtenue, la qualité des classements simulés, et la fréquence réaliste des scores valident empiriquement notre architecture.

4.7 Un projet confronté à la réalité : choix, renoncements et arbitrages

Tout au long de ce travail, nous avons dû ajuster nos ambitions à la réalité technique et temporelle. Certaines idées ont été testées puis abandonnées :

- La prise en compte de la fatigue cumulée ;
- L'influence des cartons ou des suspensions complexes ;
- La forme physique globale d'une équipe sur plusieurs journées ;
- Des stratégies d'adaptation à l'adversaire (style de jeu variable).

Ces pistes, bien qu'intéressantes, posaient trop de problèmes en termes de modélisation, de collecte de données fiables, ou d'intégration cohérente dans le pipeline existant. Nous avons donc fait le choix d'un modèle moins complet mais plus robuste, ce qui s'est révélé stratégiquement payant.

Nous avons aussi rencontré des difficultés d'ordre pratique :

- Lenteur des simulations massives sur nos machines personnelles ;
- Erreurs de format dans les données Excel ;
- Bugs de réécriture de FO/FD, variables manquantes, cellules incohérentes ;
- Conflits dans l'ordonnancement des mises à jour (domicile, streak, malus).

Chacune de ces difficultés nous a appris à être plus rigoureux, plus modeste, mais aussi plus créatifs. Le projet ne s'est pas fait en ligne droite. Il a été fait d'essais, d'erreurs, de doutes, de rectifications. Et c'est ce qui le rend précieux : parce qu'il est le fruit d'un vrai processus de travail.

4.8 Ce que nous avons appris

Ce projet nous a formé de manière très complète.

- Générer des classements moyens sur d'autres saisons ou championnats ;
- Produire des probabilités de résultats pour des journées données ;
- L'ajout de couches tactiques ou comportementales ;
- L'implémentation de modèles bayésiens ou d'IA ;
- Comparer l'efficacité prédictive de modèles concurrents (logit, apprentissage automatique).

En modélisation : nous avons compris qu'un bon modèle n'est pas celui qui intègre tout, mais celui qui produit des sorties stables, explicables, et adaptables.

En programmation : nous avons réalisé l'importance d'une structuration claire, de la lisibilité du code, de la réduction des effets secondaires et de la documentation.

En statistique : nous avons appliqué des notions de probabilité (Poisson), de tests d'hypothèse (proportion, EQM), de simulation, et de calibration paramétrique (grid search).

En gestion de projet : répartition des tâches, gestion des versions, arbitrages collectifs, anticipation des contraintes techniques.

Le plus formateur fut sans doute l'art de calibrer : comment choisir des paramètres qui produisent des effets perceptibles, mais pas exagérés ? Comment faire de l'expérimentation dans un cadre empirique rigoureux ? Ce savoir-faire sera utile dans n'importe quel autre projet de données, de simulation ou de recherche.

Enfin, ce travail nous rappelle que, malgré tous les outils statistiques ou probabilistes, le football — et le sport en général — conserve une part irréductible d'imprévisibilité. C'est ce qui en fait un spectacle unique, où tout peut arriver. L'exemple légendaire de Leicester City, sacré champion d'Angleterre en 2016 avec une cote initiale de 1 sur 50 000, incarne cette magie : aucun modèle n'aurait pu l'anticiper, et c'est précisément cela qui rend le football si beau.

4.9 Perspectives et valeur ajoutée

Notre modèle est perfectible, mais il est déjà utilisable. Il peut servir à :

- Générer des classements moyens sur d'autres saisons ou championnats ;
- Produire des probabilités de résultats pour des journées données ;
- L'ajout de couches tactiques ou comportementales ;
- L'implémentation de modèles bayésiens ou d'IA ;
- Comparer l'efficacité prédictive de modèles concurrents (logit, apprentissage automatique).

Il peut aussi être enrichi par la suite par :

- Une interface utilisateur (web ou app) ;
- L'intégration de données temps réel ;
- L'ajout de couches tactiques ou comportementales ;
- L'implémentation de modèles bayésiens ou d'IA.

Mais au-delà du code, ce projet nous a appris à raisonner sur le monde avec des outils quantitatifs. À confronter nos intuitions à des données. À tester, valider, corriger. Et c'est cela que nous retiendrons comme la véritable valeur ajoutée de ce mémoire.

Bibliographie

[1] Transfermarkt, *Données des effectifs et valeurs des clubs.* `https://www.transfermarkt.fr`

[2] Sofascore, *Statistiques de matchs en direct.* `https://www.sofascore.com`

[3] Ross, Sheldon M., *Introduction to Probability Models*, Academic Press, 11$^{\text{th}}$ Edition, 2014.

[4] Kroese, D. P., Brereton, T., Taimre, T., Botev, Z. I., *Why the Monte Carlo method is so important today*, Wiley Interdisciplinary Reviews : Computational Statistics, 2014.

[5] Van Rossum, G., Drake, F. L., *The Python Language Reference Manual*, Python Software Foundation.