TieBreaker AI - Cahier des charges

IA de prédiction de matchs de tennis (ATP/WTA)

09 Octobre 2025

Par Edouard DUPUCH & Valentin ROUSSEAU

1. Introduction

1.1 Contexte et ambition

TieBreaker IA vise à prédire, avant un match professionnel, avec un ensemble complet de statistiques : probabilité de victoire des joueurs, chances de gagner chaque set, probabilité de tenir le service (hold) ou de breaker, total de jeux, nombre attendu d'aces et volume de doubles fautes.

Pour le projet TieBreaker IA, le but est de concevoir une IA qui prédit et restitue des statistiques entre deux joueurs de tennis fournis en entrée.

L'équipe se compose de deux développeurs (Edouard DUPUCH et Valentin ROUSSEAU), tous deux orientés IA, avec Valentin en tant que développeur Lead.

Le projet se déroule sur 4–5 mois à raison de deux jours de travail par semaine (≈ 1 mois plein).

Le langage de référence est Python, choisi pour son écosystème IA riche. Une première brique de modélisation repose sur un DecisionTreeClassifier et Random Forest ,avec un objectif d'itérer ensuite vers XGBoost/LightGBM.

1.2 Hypothèse de travail et limites

TieBreaker est pensé :

IA : traitement de la data, entrainement et amélioration de la prédiction.

Front : Affichage type "UserFriendly", maximum de données sur les résultats des prédictions ainsi que des matchs.

Back : Requête Get a des API afin d'améliorer la base de données et de permettre un affichage en temps réel pour l'utilisateur.

Le But n'est pas de « prédire l'avenir » de manière déterministe, mais de fournir des probabilités bien calibrées qui aident à la prise de décision et à l'analyse avant rencontre entre les sets et a posteriori pour le débriefing analytique.

1.3 Base de données

Nous considérons que les facteurs majeurs expliquant l'issue d'un match sont capturés par des variables contextuelles et historiques : surface, forme récente, affrontements passés, fatigue, style de service et qualité de retour.

Certaines composantes non observées (état psychologique, micro-blessures non déclarées) resteront des sources d'incertitude incompressibles.

Notre démarche vise donc la robustesse statistique et la calibration plutôt que la sur-optimisation sur un jeu de test fixe.

1.4 Valeur métier et scénarios d'usage détaillés

La base inclut, pour chaque match, des métadonnées normalisées (tournoi, tour, date, surface, indoor/outdoor, pays), les identifiants joueurs, et un ensemble de statistiques de performance courantes.

Les classements hebdomadaires et les points associés sont historisés afin d'estimer des courbes de forme et des ratings spécifiques à la surface.

Un protocole d'alignement et de dédoublonnage des joueurs est appliqué pour garantir la cohérence inter-saisons.

Une base de données consolidée constitue le socle du projet. Elle recense l'ensemble des matchs officiels et les classements ATP/WTA depuis le début de l'ère Open, en 1968.

Cette couverture longue permet de construire des indicateurs robustes par surface, d'agréger des historiques Head-to-Head sur le temps long, et de modéliser des effets structurels (évolution du matériel, rythmes de jeu, transitions de surfaces) tout en maîtrisant les biais de période historiques.

1.5 Qualité des données et traçabilité

Pour un coach, ces éléments orientent la préparation tactique et la gestion de l'effort.

Pour un analyste, ils offrent un cadre quantifié et traçable, tandis qu'un public enthousiaste dispose d'une lecture probabiliste claire et respectueuse de l'incertitude.

Avant le match, TieBreaker IA sert à obtenir un diagnostic synthétique : probabilité de victoire, chance de gagner le premier set, probabilités conditionnelles de hold et de break et métriques dérivées comme le total de jeux attendu.

Après match, l'outil permet de réaliser une comparaison entre la prediction et le résultat du match.

2. Objectifs et périmètre

2.1 Objectifs d’usage

Le système charge un match (joueurs, tournoi, surface, tour) et produit des prédictions pré-match.

Les sorties couvrent la probabilité de victoire du match, la probabilité de gain de set, les chances de hold/break pour chaque joueur et des indicateurs agrégés (total de jeux, aces attendus, doubles fautes, probabilité de tie-break).

Les résultats sont exportables en plusieurs format type JSON, MD, PDF.

2.2 Cas d’usage

Analyste/Coach : recherche des insights explicables et actionnables pour préparer un match.

Étudiant : explore les variables et compare plusieurs modèles de prédiction.

Fan : souhaite un score de probabilité clair et lisible pour appréhender l’issue probable.

3. Données & qualité

3.1 Sources & couverture

Sources historiques ouvertes couvrant ATP, WTA, Challenger et Grands Chelems sur plusieurs saisons et années.

Contexte : surface, indoor/outdoor, météo , jours de repos, phase du tournoi.

3.2 Nettoyage & gouvernance

Unification des noms et identifiants joueurs ; gestion des doublons.

Prise en compte des matchs incomplets/abandons.

Traçabilité des versions (CSV).

Respect RGPD aucune donnée personnelle sensible n’est traitée.

4. Schéma minimal & feature engineering

4.1 Schéma minimal

Match : id, tournoi, tour, surface, pays, score par set, durée, date.

Joueurs : id, latéralité, âge, classement, points, forme récente.

Statistiques : aces, doubles fautes, points gagnés 1ère/2ème, balles de break.

4.2 Fonctionnalités attendues

Base de données complète avec actualisation des matchs termine.

Résultats rapides via DecisionTreeClassifier, Random Forest.

Perfectionnement via XGBoost/LightGBM.

Front Flutter (iOS/Android/Web) pour une diffusion multiplateforme.

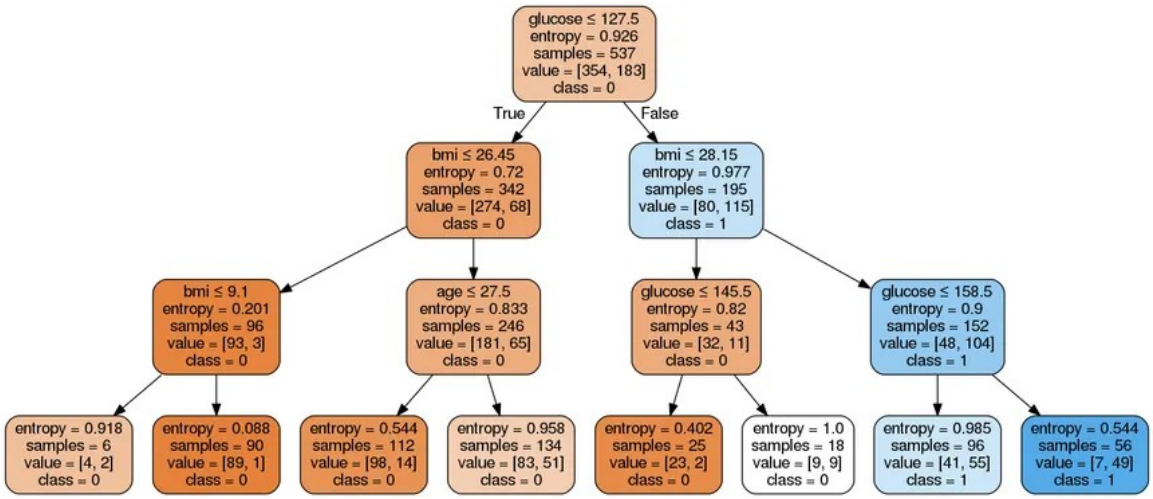
5. Modélisation & calibration

5.1 Définition et concept (arbre de décision)

Un arbre de décision est un outil d'aide à la décision représentant un ensemble de choix sous forme d'arbre.

C'est une approche intuitive et visuelle qui modélise des processus complexes de manière explicable.

EXEMPLES :



5.2 Justification du choix (DTC → XGB/LGBM)

Peu de données au départ → DTC fournit une baseline simple et robuste.

Facilité d'implémentation et vitesse d'apprentissage par rapport à un réseau de neurones. XGBoost/LightGBM améliore les performances.

5.3 Outils et technologies utilisés

Pandas : manipulation, nettoyage, pré-processing des jeux de données.

Numpy : Librairie mathématique indispensables.

DecisionTreeClassifier : implémentations des arbres de classification.

Scikit-learn : implémentations des arbres, random forest, calibration et évaluation.

Joblib : persistance et rechargement des modèles entraînés

6. Évaluation & critères d'acceptation

Validation via test de match non-ajouté dans la base de données.

7. Architecture technique & API

7.1 Pipeline & entraînement

ETL Python/Pandas vers CSV/Parquet versionnés. Notebooks + scripts CLI.

7.2 API & déploiement

API FastAPI : /predict, /match/{id}, /health ; modèle sérialisé (joblib).

7.3 Spécification d’API (exemple)

POST /predict : { player_a: "Novak Djokovic", player_b: "Carlos Alcaraz", context: { tournament: "Wimbledon", surface: "Grass", round: "Final", indoor: false, rest_days_a: 2, rest_days_b: 1 } }.

Réponse : { p_win_a, p_set_win, p_hold, p_break, expected_games, expected_aces, model: { name, calibration }, ci: {…} }.

8. UX Front (Flutter)

Accueil : recherche joueurs/matchs, derniers événements.

Fiche match : probabilités calibrées, jauges hold/break, total de jeux attendu, export CSV/PNG.

Historique : derniers matchs consultés, export CSV/PNG.

9. Organisation

Valentin ROUSSEAU (Lead/Front) : architecture Flutter, intégration API, revues.

Edouard DUPUCH (Data/MLOps) : pipeline donnée, features, entraînement, évaluations.

10. Planning & jalons (8 sprints de 2 jours / 16 h chacun)

S1 : setup repo, collecte & schéma des données, datasets propres.

S2 : Stabilisation, release v1, démo. Suivi via GitHub Milestones + Trello.

S3 : baseline DTC (notebook + rapport) + recherche documentation librairie.

S4 : Parsing data.

S5 : RandomForest + CSV par saison/surface.

S6 : XGBoost/LightGBM + calibration (courbes de calibration, ECE stabilisée).

S7 : API FastAPI + Docker (endpoint /predict opérationnel, image).

S8 : Front Flutter MVP (Accueil, Fiche match).

S9 : Tests, documentation, monitoring (unit tests, logs, tracing simple).

11. Qualité & conformité

Traçabilité des données. Sécurité API : clés, quotas, logs ; conformité RGPD.

12. Risques & contraintes

12.1 Techniques

Qualité/traitement de la data : résultats faibles si données mal nettoyées → pipeline de nettoyage + tests de qualité.

Poids des applications : les applications Flutter sont souvent plus lourdes qu'une app native équivalente.

12.2 Organisationnelles

Travail 2 jours/semaine en duo : nécessité d'une planification précise et d'un partage des tâches.

12.3 Déploiement

Déploiement iOS potentiellement payant.

13. Livrables finaux

Dataset Propre

Application avec prediction “fiable”

Documentation precise et userfriendly

UI/UX Propre et fonctionnel

RGPD respecter

Requete API fonctionnel, mise a jour de la base de donnees

Annexe A - Glossaire

ETL (Extract – Transform – Load) : exporte les données en CSV et/ou Parquet, avec un système de versionnement.

LogLoss : mesure la qualité des probabilités prédites.

Brier score : erreur quadratique moyenne des probabilités.

ECE (Expected Calibration Error) : mesure de la calibration globale.

Elo / Surface-Elo : ratings de performance globale et spécifique à la surface.

RGPD : Règlement général sur la protection des données

Annexe B - Ressources & documentation consultées

Documentation officielle scikit-learn (DecisionTreeClassifier, RandomForest, calibration).

Tutoriels vidéo (YouTube) sur la théorie/pratique des arbres de décision.

Guides pratiques (ex. DataCamp) pour une mise en œuvre en Python.

Visualisations (diagrammes) illustrant la structure et le fonctionnement des arbres.

Annexe C - Organisation du projet (DTC)

Le code lié à la baseline est organisé dans un dossier « DTC » (Decision Tree Classification).

Cette structure facilite la maintenance et la collaboration, et servira de base aux itérations vers RF et XGB/LGBM.

Annexe D - Justification technologique

Python : écosystème IA (Pandas, scikit-learn, XGBoost/LightGBM), forte communauté, vitesse d'itération.

Flutter : cross-platform (iOS/Android/Web), rapidité de prototypage, UI cohérente sur plateformes.