Prédiction de la durée de matchs de tennis

Léo Beuque

Projet supervisé par Pierre-Yves DALLA LIBERA

Octobre-Novembre 2022 Parcours SDI

Remerciements

Je tiens à remercier mon tuteur, Pierre-Yves DALLA LIBERA, pour m'avoir accompagné de très près au cours de ce projet au format peu conventionnel et pour s'être toujours bien assuré du bon déroulement de celui-ci et de la définition d'échéances courtes et régulières, tout cela malgré la distance. Son adaptabilité et son sens des priorités m'ont permis de progresser de manière régulière et concrète dans le projet et m'ont finalement permis de mener à bien celui-ci.

Enfin, je tiens à remercier l'ensemble du corps enseignant de l'École Centrale de Lille pour m'avoir permis d'acquérir les compétences et connaissances nécessaires au métier d'ingénieur en science des données.

Table des matières

Introduction

Le tennis est l'un des sport les plus populaires dans le monde. A titre d'exemple, le tournoi de Roland-Garros accueille chaque année plus de trois cent mille spectateurs et plusieurs millions de télespectateurs. Bien que très incertain, le déroulement de matchs de tennis a donc fait l'objet de nombreux travaux cherchant à le prédire, principalement dans le but de déterminer à l'avance le vainqueur, dans le contexte des paris sportifs en ligne par exemple. Mais le résultat du match n'est pas la seule variable d'intérêt. En effet, la prédiction de leur durée serait également très précieuse pour différents acteurs comme les agences publicitaires, les organisateurs de tournois ou même les joueurs. Le but de ce projet est ainsi d'améliorer un premier modèle de prédiction des durées de matchs de tennis qui avait été mis au point l'année dernière. Ainsi, l'objectif de ce projet est d'améliorer un premier modèle de prédiction de la durée des matchs de tennis qui avait été développé dans le cadre d'un autre projet l'année dernière. Parmi les pistes initialement évoquées pour la mise en place de cette amélioration, on retrouve notamment la construction d'un réseau de neurones et la prise en compte d'autres sources de données exogènes. Une étude approfondie du premier modèle mis en place et de la bibliographie a cependant permis de mettre en évidence d'autres aspects prioritaires à améliorer et vers lesquels ce projet s'est réorienté. Nous développerons ceux-ci en détail dans la partie dédiée (4).

Le projet précédent ne faisait pas mention d'une quelconque recherche bibliographique hormis la citation du répertoire github de Jeff-Sackmann duquel provient les données. Cependant, le tennis étant un sport populaire, de nombreux auteurs [1][2][3][4] ont cherché à prédire le résultat de matchs dans le contexte de la compétitivité propre au milieu des paris sportifs en ligne. Bien qu'aucun de ces travaux ne semble directement porter sur la prédiction des durées de matchs de tennis, il est possible de s'en inspirer pour notre modélisation, les données étant soit très semblables, soit les mêmes.

L'article [5] porte quant à lui sur une mise en application directe de modèles de régression dans le but de prédire la durée de matchs de criquets, ce qui peut être intéressant pour comparer nos modélisations et nos résultats.

1 Contexte du problème

1.1 Provenance des données

Le but du projet est donc d'être capable de prédire les durées d'un match à partir d'informations sur celui-ci et sur les deux joueurs impliqués. Les données sont issues des matchs disputés lors de l'ATP Tour qui est le principal circuit international de tennis masculin lors duquel les joueurs professionnels s'affrontent dans de nombreux tournois différents. En fonction du classement du joueur à un tournoi et de l'importance de ce tournoi, des points ATP lui sont attribués pour une durée de 52 semaines. C'est sur la base de ces points qu'est construit le classement mondial de l'ATP qui est utilisé pour la qualification aux prochains tournois. Les données utilisées, provenant du répertoire GitHub de Jeff-Sackmann, comportent des informations sur les conditions et la nature du du match (date, surface, nombre de sets gagnants, etc.), sur des caractéristiques des

joueurs (taille, points et rangs ATP, nom du joueur, etc.), sur les performances de chaque joueur lors du match (nombre de ace, de doubles fautes, etc.) et évidemment, sur sa durée. Enfin, afin de mieux comprendre la nature des données, voici une courte description du déroulement d'un match de tennis.

1.2 Déroulement d'un match de tennis

Un joueur gagne le match lorsqu'il remporte 2 ou 3 sets, en fonction du format du match (best of 3 ou best of 5). Chaque set est constitué de 6 jeux remportés, sauf lorsque chacun des joueurs a remporté au moins 5 jeux, auquel cas le joueur remportant le set est le premier qui parvient à remporter 7 jeux. Un jeu est remporté par le premier joueur qui marque quatre points (0/15/30/40/jeu), si l'autre joueur n'a pas marqué 3 points. Sinon, un joueur remporte le jeu dès qu'il a deux points d'avance sur son adversaire. Enfin, chaque échange détermine l'issue d'un point.

Ces règles peuvent varier dans certains cas particuliers en fonction du tournoi ou même de l'époque. Notamment, le match le plus long jamais disputé a duré 11 heures et 5 minutes, en raison d'un set qui s'est terminé avec 70-68 : un joueur ne pouvait remporter le set qu'à condition d'avoir deux jeux d'avance sur son adversaire. La règle du tie-break qui s'est répandue plus tard dans différents tournois évite ceci.

2 Analyse des données

2.1 Une structure de données complexe

Une structure de données complexe, dûe à de fortes interdépendances , dont la dépendance au temps.

2.1.1 Dépendance temporelle des données

Il apparait assez rapidement que la date à laquelle un match se déroule a une importance toute particulière. En effet, les performances des joueurs évoluent significativement au cours du temps en raison de leur âge, de leur expérience ou encore des méthodologies d'entraînement. De même, même si ces changements sont très rares, les règles de différents tournois peuvent évoluer au cours du temps, comme c'était le cas pour la mise en place du tie-break dans certains tournois, ou pour la suppression de la surface "Carpet" de l'ATP Tour en 2009. Il semble donc que la date d'un match fait significativement évoluer la relation entre nos données et la cible. Autrement dit, plus un match s'est déroulé il y a longtemps, moins il est susceptible d'être pertinent pour aider le modèle à déterminer cette relation, et donc à prédire la durée du match. Ainsi, bien qu'une première solution soit de se débarrasser des données les plus anciennes, nous avons choisi d'utiliser des poids calculés en fonction de la date du match, à l'instar de plusieurs auteurs. Ces coefficients sont calculées de la façon suivante :

$$recency\_weights = \left(\frac{1}{2}\right)^{\frac{t}{\lambda}}$$

où t est correspond au nombre de jours entre le match et le jour le plus récent présent dans la base de données et λ où est appelé "demi-vie".
La détermination de ce coefficient λ sera décrite plus en détail dans 4.4.3.

Une conséquence importante de cette dépendance des données au temps est l'ajout d'une dimension supplémentaire importante à l'espace des données. Autrement dit, il semble que tout se passe comme si le nombre de données était réduit : Si λ vaut 365 et que l'on a autant de données chaque année, la somme des poids des données de l'année 2021/2022 est plus grande que la somme des poids du reste des données (somme géométrique de raison 1/2). Ainsi, c'est "un peu comme si" on ne se retrouvait qu'avec l'équivalent de 2 années de données, au lieu de 30 années.

2.1.2 Regroupement des données par joueur

Il pourrait être intéressant de garder le nom de chaque joueur comme attribut pour la prédiction, mais le nombre de joueurs est trop important et ces joueurs changent au cours du temps. En effet, on dénombre plus de 2300 joueurs ayant participé à l'ATP Tour depuis 1990, ce qui correspondrait à 2300 classes, ce qui n'est pas compatible avec de l'encodage one-hot et avec le nombre de données (de l'ordre de 80000 matchs), en raison de la malédiction de la dimension.

Il peut également être tentant de regrouper des matchs similaires ensemble(s) (matchs disputés par un même joueur, par deux mêmes joueurs, contre un même joueur). Ceci a d'ailleurs été fait d'une certaine manière dans 4.5 par le biais de la création de nouveaux attributs à l'aide de données historiques. Cependant, le nombre de matchs joués par chaque joueur dans l'ATP Tour depuis 1990 est en pratique très limité. En effet, bien que Roger Federer détient le plus grand nombre de matchs joués depuis 1990, ce qui correspond à 1424 matchs, le nombre moyen des matchs joués par chaque joueur est de 97 de tandis que son nombre médian est de 9. Il ne paraît donc pas raisonnable de construire un modèle de machine learning par joueur, ceux-ci nécessitant généralement un nombre important de données. C'est pourtant cette approche qui a été choisie dans le projet réalisé l'année dernière, dont nous évaluerons les résultats dans la partie dédiée (3.1).

2.1.3 Choix de la représentation des informations sur les joueurs

Il n'est pas évident de choisir la représentation de données que l'on souhaite fournir au modèle. En effet, pour chaque match, plusieurs attributs sont associés à chacun des deux joueurs. Ainsi, si on n'altère pas la façon dont sont initialement représentées les données d'un match, cette représentation dépendra de l'ordre a priori arbitraire dans lequel sont choisis les joueurs.

— Représentations des matchs dans la littérature pour la prédiction du résultat du match :

  Une première façon de procéder, utilisée à plusieurs reprises dans des travaux ([2][4]) dont le but est de prédire le vainqueur , est de se baser sur un attribut fortement corrélé au niveau du joueur, comme son rang ATP par exemple, pour

définir quel joueur est le favori et quel joueur est l'"underdog" afin que le choix de l'ordre ne soit plus arbitraire. Malheureusement, il n'est pas clair que le meilleur des attributs que l'on puisse choisir soit suffisant pour proposer un ordre fiable. En effet, deux couples de joueurs peuvent très bien correspondre à un niveau global presque égal (donné par l'attribut précédemment décrit) mais se différencier sur d'autres attributs très différents les uns des autres. Une autre méthode souvent utilisée et combinée à la première dans certains travaux dont le but est à nouveau de prédire le vainqueur ([2][4]) consiste à réduire l'ensemble des attributs associés aux joueurs à des attributs symétriques par rapport à 0, en remplaçant par exemple les attributs par les différences des attributs entre les deux joueurs. Cette approche semble très pertinente lorsque le but du modèle est de comparer les performances des joueurs afin de prédire l'issue du match. Cependant, elle conduit à une perte d'information potentiellement précieuse dans notre cas, où l'on cherche à prédire la durée des matchs et non leur issue. En effet, il est tout à fait possible par exemple que l'âge des joueurs soit davantage corrélé à la durée que la différence d'âge entre les deux joueurs. De plus, on peut imaginer que les modèles cherchant à prédire le vainqueur doivent être symétrique par rapport à 0, ce qui n'est clairement pas le cas de notre modélisation dont le but est la prédiction de durées. Enfin, il peut malgré tout être intéressant de ne pas remplacer les attributs mais plutôt de les compléter à l'aide de différences, bien que ceci conduise à l'augmentation de la dimension des données et expose ainsi nos modèles aux problèmes associés.

— Représentations utilisées dans ce projet :

Deux autres approches ont donc été utilisées. Une troisième consiste à dupliquer les données en interchangeant les joueurs. Ainsi, cela permet de ne pas perdre d'information et de ne pas exposer le modèle à une représentation dépendant d'un choix contestable d'attribut discriminant. En contrepartie, tout modèle symétrique ne faisant pas intervenir d'interdépendances entre les attributs (comme une régression logistique) sera incapable d'exploiter les attributs résultant de différences (attributs symétriques en fonction de l'ordre des joueurs), étant donné que les coefficients associés à ces attributs devraient être nuls en raison de la symétrie des données induite par ce choix de représentation.
Enfin une quatrième solution, étudiée, correspond à l'introduction de valeurs absolues permettant de résoudre ce problème de symétrie. Cependant, on perd alors la possibilité de comparer les valeurs relatives de ces attributs, ce qui correspond à une perte d'information. A titre d'exemple, on pourrait imaginer qu'un joueur mieux classé et qui soit meilleur aux services que son adversaire puisse écraser très rapidement celui-ci, tandis qu'un joueur mieux classé mais moins bon que son adversaire aux services mène un match serré et long.
En pratique, des tests ont montré que la troisième approche donnait de meilleurs résultats que la quatrième lorsqu'elle était utilisée avec le meilleur des modèles testés au moment des tests, c'est-à-dire avec une forêt aléatoire peu profonde, malgré la symétrie de l'ensemble d'entraînement.

2.2 Analyse statistique

Cette partie a pour but de donner une idée de la répartition des données et de mettre en évidence des corrélations, ainsi que leur degré, entre certains attributs et la cible, c'est-à-dire la durée du match. Ainsi, la figure 1 permet par exemple d'observer l'importante et naturelle différence de distribution des durées des matchs en fonction du nombre de sets gagnants, ainsi que de constater que les durées des matchs sont très étalées, ce participe à mettre en exergue la difficulté du problème.

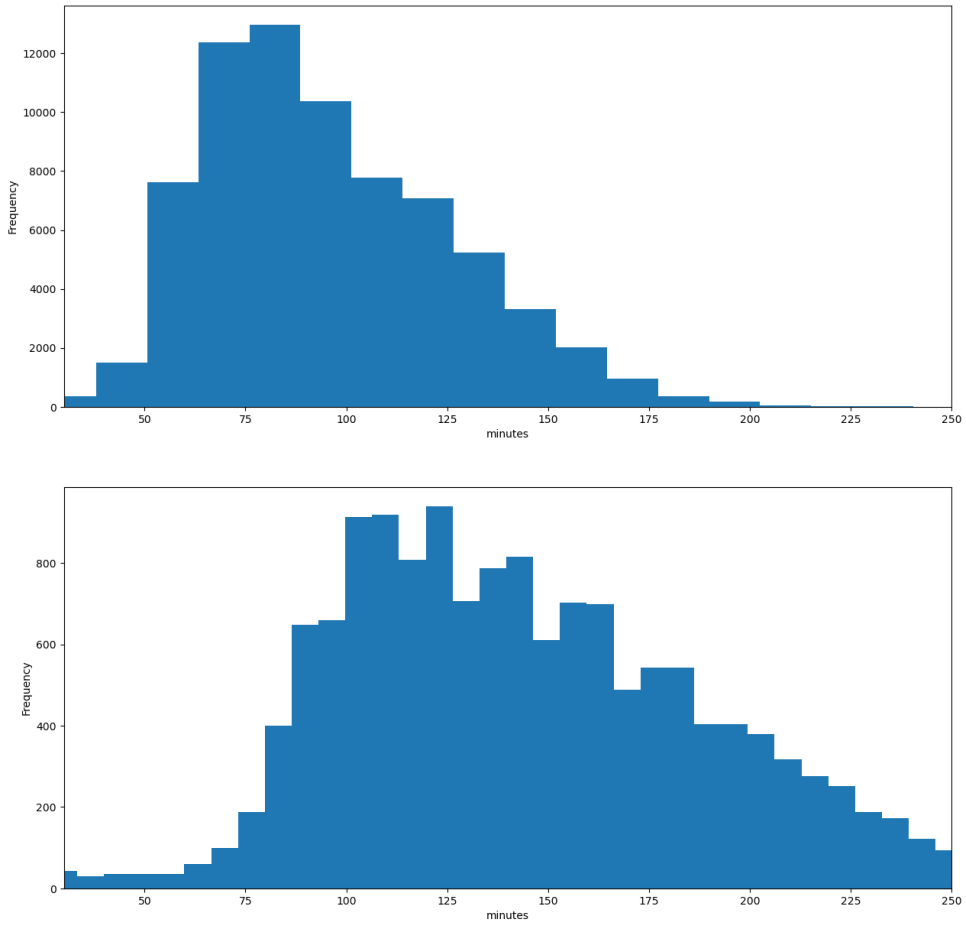

FIGURE 1 – Distribution des durées des matchs en fonction de l'attribut "best-of"

Enfin, les figures 2 et 3 permettent de constater que les corrélations entre les différents attributs et la cible semblent très faibles. En particulier, alors que l'on verra plus tard que l'attribut "rank_diff" est l'un de ceux qui contribuent le plus à la détermination des durées prédites, on observe sur 3 que la corrélation entre cet attribut et les durées des matchs semble être déjà très faible. Ainsi, on espère que de fortes interdépendances entre les différents attributs permettront d'apporter des informations supplémentaires quant à la durée des matchs, et que le nombre de données permettra aux différents modèles de mettre celles-ci en évidence.

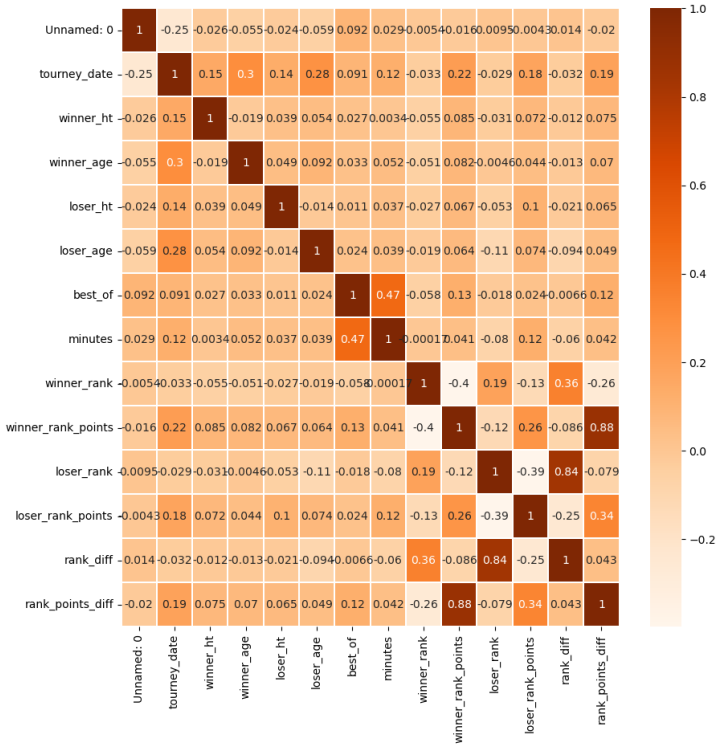

FIGURE 2 – Matrice de corrélation associée aux attributs numériques

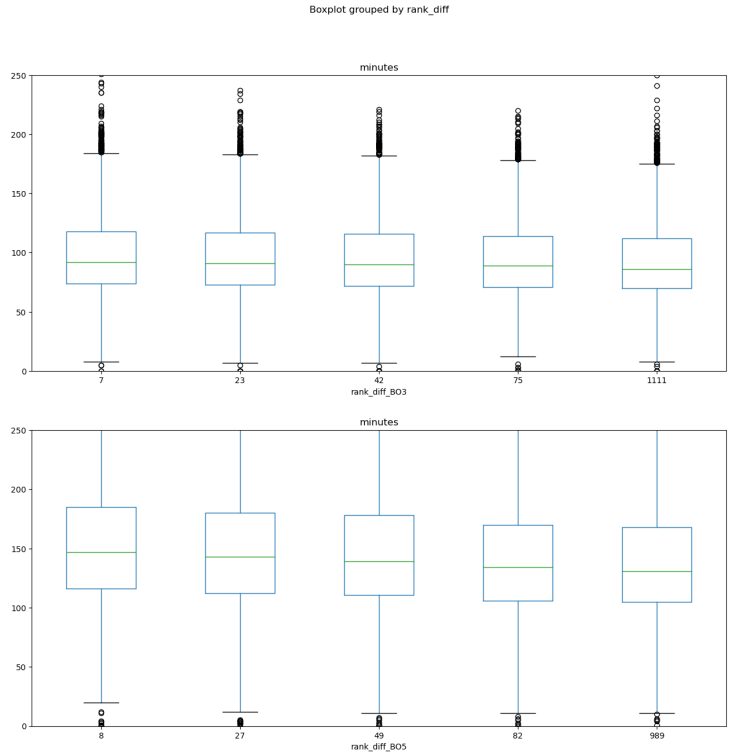

FIGURE 3 – Mise en évidence à l'aide de boîtes à moustache de la corrélation a priori faible entre la différence de rang et la durée des matchs

3 Point de départ du projet : l'approche du premier projet

3.1 Résumé de l'approche du premier projet

L'objectif de cette partie est de décrire rapidement ce qui a été fait dans le projet précédent (dont le répertoire github est accessible ici) afin de mettre en avant les améliorations apportées par mon projet, le but de ce dernier étant justement de reprendre et chercher à améliorer la première approche. Dans un premier temps, une analyse des données, que j'ai par la suite approfondie, a été réalisée afin de mettre en évidence quels ensembles de données étaient utilisables et d'étudier les corrélations entre certains des attributs et la variable cible. Quant à la modélisation, les éléments notables sont l'utilisation d'un modèle et d'un ensemble de données par joueur, ce qui permet de conserver l'identification de celui-ci, mais réduit drastiquement le nombre de données et nécessite l'entrainement d'un modèle par joueur (ce qui est trop coûteux lorsque ceci est combiné à une optimisation des hyperparamètres par modèle), et l'implémentation manuelle d'une forêt aléatoire à partir de la classe DecisionTree de Sklearn sans bootstrap. D'autres éléments récurrents dans les pipelines d'apprentissage automatique ont pu être réutilisés comme la normalisation des données, une partie de la sélection d'attributs, l'ensemble des hyperparamètres associés aux arbres (et non à la forêt) , la tentative d'estimation de l'erreur du modèle, ou même certains éléments qui m'ont aidé à mettre en place le changement de représentation des données. **En dehors de ces éléments, tout le reste provient d'une nouvelle implémentation propre à mon projet**.

Enfin, ce premier projet mettait à disposition un lien MyBinder vers une application en ligne générée à l'aide de la librairie python ipywidgets et d'un répertoire github public. Cependant, la prédiction de la durée d'un match dont on pouvait entrer les informations à la main nécessitait le réentraînement complet d'un modèle, ce qui pouvait être très long et ne semble par utiliser correctement le potentiel de l'apprentissage automatique.

3.2 Evaluation du modèle du premier projet

Aucun ensemble de test n'étant mis à disposition, j'ai utilisé pour évaluer son modèle mon premier ensemble de test que j'ai partitionné en différents sous-ensembles pour chaque joueur. Il était trop couteûx de chercher à optimiser chaque modèle pour chaque joueur. J'ai donc sélectionné les hyperparamètres correspondant à la meilleure forêt aléatoire obtenue au moment du test et entraînée à l'aide d'un ensemble d'entrainement utilisant les données de tous les matchs de tous les joueurs, joués avant ceux de l'ensemble de test.

Les résultats obtenus étaient très mauvais, le R2-score obtenu étant négatif. Je n'ai donc pas cherché à conserver sa modélisation et ai plutôt fait le choix de conserver l'ensemble des données non segmenté par joueur pour la suite.

4 Amélioration de l'ensemble du processus, des données au modèle

4.1 Discussion sur la méthodologie adoptée

4.1.1 Une utilisation de la bibliographie non optimale ?

Comme énoncé plus tôt, le projet précédent n'utilisait aucune référence bibliographique. Bien que mon expérience m'ait naturellement conduit à chercher rapidement des problèmes similaires au notre dans la littérature, mes premières lectures étaient superficielles car je cherchais seulement alors à trouver des travaux correspondant exactement à notre problème de prédiction de durées de matchs de tennis. N'ayant trouvé que des travaux dont le but était de prédire l'issue du match et non sa durée, nous avons préféré nous plonger directement dans le code mis à disposition par le projet de l'année dernière, avant d'avoir approfondi la lecture des différents articles. L'approche choisie consistait alors à consulter ces travaux (ainsi que d'autres) seulement lorsque des choix cruciaux devaient être faits (hyperparamètres à tester, représentation des données des matchs, attributs à sélectionner), et la lecture s'interrompait dès qu'une solution au problème traité sur le moment était identifiée. L'avantage de cette approche est que celle-ci permet de ne pas perdre de temps à consulter des sous-parties sans aucun lien avec le problème général traité, ce qui est loin d'être négligeable dans un projet réalisé sur une période de temps aussi courte.

Cependant, du temps aurait en pratique pu être économisé à l'aide d'une lecture au moins légèrement plus approfondie des différents travaux portant sur le tennis. En effet, l'approche choisie nous a concrètement fait perdre du temps sur un problème de surapprentissage dont l'origine n'a pas été facile à identifier. Finalement, il s'est avéré que le problème provenait majoritairement d'un mauvais choix de méthode de validation croisée, inadapté à la dépendance temporelle des données, alors que la méthode de validation croisée finalement sélectionnée (4.2.3) apparaissait déjà dans l'un des articles [2] étudié plus tard. Cependant, les tests réalisés permettent malgré tout de confirmer la pertinence des solutions proposées dans les autres travaux et appliquées à notre problème particulier, et le temps passé à les implémenter n'est donc probablement pas perdu.

Finalement, il semble qu'il aurait quand bien même pu être pertinent d'adopter un niveau de lecture des travaux comportant des données similaires aux nôtres légèrement plus élevé que celui initialement choisi.

4.1.2 Ordre et Priorisation des tests

Dans tout projet de science des données, de nombreux choix doivent être faits concernant la représentation des données, leur traitement, le modèle à sélectionner, son architecture éventuelle, les hyperparamètres du modèle et du traitement de données, etc. L'espace constitué par l'ensemble des modélisations possibles est très vaste, et en pratique impossible à explorer intégralement dans la majorité des projets. L'idéal serait donc que les différentes composantes de cet espace soit indépendantes vis-à-vis de l'attribut cible, ce qui permettrait d'optimiser nos choix un par un mais ce n'est pas nécessairement le cas

non plus, sans oublier qu'il peut être long de tester chaque possibilité. Une mission fondamentale du data scientist est alors de déterminer quels choix peuvent être faits en se basant uniquement sur ses connaissances, expériences et intuitions, et quels choix doivent être testés, ce qui est souvent loin d'être évident, en fonction du temps imparti. En pratique, la réflexion précédente a été formulée seulement à l'issue du projet et non en amont. Ainsi, j'ai adopté dans ce projet une approche dans laquelle j'ai réalisé un nombre très important de tests, sans prendre en compte le fait que cela était très coûteux en temps, ce qui m'a certes permis de m'assurer que les choix effectués étaient les plus pertinents parmi ceux testés mais m'a en contrepartie empêché d'attribuer le temps nécessaires à l'exploration des pistes initialement proposées en début de projet, c'est-à-dire à la mise en place d'un réseau de neurones adapté au problème et à la prise en compte de données exogènes. Il est possible d'argumenter que les tests effectués étaient de toute façon nécessaires à l'amélioration de la modélisation, mais il est d'avantage ici question d'une priorisation de ces tests nécessaires qui n'a pas été effectuée en début de projet avec le degré de recul suffisant, bien qu'il ne soit pas évident qu'une telle priorisation pertinente puisse être mise en place. Ainsi, il aurait par exemple été possible de réutiliser directement des éléments de la bibliographie transposables à notre problème sans les tester, comme l'hyperparamètre λ optimisé ici 4.4.3, mais pas comme la représentation de données symétrique, parfaitement adaptée à la prédiction des résultats du match mais moins à celle des durées (2.1.3).

Enfin, l'ordre des prochaines sous-parties rend justement en partie compte de l'ordre chronologique dans lequel les tests et implémentations ont été effectués.

4.2 Métriques et ensembles utilisés pour les tests

Le reste de cette partie fait intervenir une succession des tests précédemment décrits, dont le but était donc de déterminer la meilleure modélisation pour notre problème de régression.

4.2.1 Métriques utilisées

C'est le R2-score qui a été choisi afin de comparer les résultats des différents tests. Il s'agit d'une métrique couramment utilisée dans les problèmes de régression et calculée de la façon suivante :

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

où n est le nombre de matchs, y_i la valeur de la durée associée au match numéro i, $\hat{y}_i$ la valeur prédite correspondante et $\bar{y}$ la moyenne des durées.

4.2.2 Choix des différents ensembles

Afin que les résultats des tests représentent fidèlement les performances réelles du modèle, il faut que les ensembles testés soient représentatifs de la distribution des données qu'on voudra finalement tester. Pour cela, il faut que l'ensemble testé soit suffisamment grand. En pratique, plusieurs ensembles de test ont été utilisés et seul le dernier mis

en place répondait aux exigences précédemment définies. Celui-ci correspond donc aux matchs les plus récents, sur une période d'une année entière.

Cependant, cet ensemble de test ne doit pas être trop sollicité lors de la recherche des meilleurs hyperparamètres. En effet, à l'inverse des ensembles de validation généralement utilisés pour cette recherche, l'ensemble de test doit seulement servir à vérifier l'exigence définie précédemment, qui ne peut plus être vérifiée si les hyperparamètres ont été choisis sur la base des résultats obtenus sur cet ensemble particulier, qui ne représenterait plus alors la distribution sous-jacente des données mais seulement un échantillon particulier. C'est pourquoi des ensembles de validation, construits de la même façon que les ensembles de test, sont utilisés dans le but de déterminer ces hyperparamètres optimaux. On utilise même généralement une validation croisée pour diminuer le risque de surapprentissage des hyperparamètres sur un couple ensemble d'entraînement/ensemble de validation spécifique, et ainsi obtenir une estimation plus robuste des performances du modèles.

En pratique, l'ensemble de test a malgré tout été utilisé lorsque le nombre de tests à faire était minime (lorsqu'on a du choisir entre deux représentations de données par exemple). En pratique, cela s'est avéré être un mauvais choix en raison de la démultiplication des tests utilisant l'ensemble de test (plus d'une vingtaine).

4.2.3 Ensemble de validation et validation croisée

A l'instar de l'ensemble de test, la méthode de génération de la validation croisée a évoluée au cours du projet. En effet, la première stratégie consistant à utiliser des échantillons aléatoires des données d'entraînement pour générer des "splits" utilisés pour créer les ensembles de validation s'est avérée infructueuse, menant à un surapprentissage dont l'identification de la cause n'a pas été aisée. En effet, celle-ci ne fonctionne que dans le cas où les données sont indépendantes et identiquement distribuées, ce qui n'est pas le cas de données dépendant du temps. C'est pourquoi la méthode TimeSeriesSplit de Sklearn a finalement été utilisée pour générer les ensembles de validation, dont on peut voir la représentation en figure 4. On peut d'ailleurs noter que l'ensemble d'entraînement ne varie pas significativement d'un ensemble à l'autre. Enfin, il s'est avéré grâce à une lecture approfondie tardive de certains articles que cette méthode avait déjà été sélectionnée dans le cadre d'autres travaux [2].

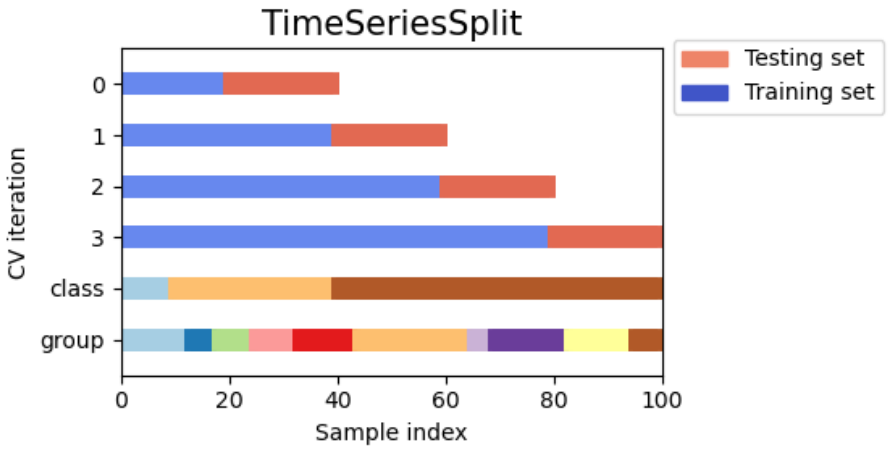

FIGURE 4 – Comportement de la méthode TimeSeriesSplit de Sklearn. Source

Les ensembles de validation ainsi utilisés dans la validation croisée correspondaient tous à des échantillons de la même taille que l'ensemble de test, correspondant à une succession d'années entières. En pratique, nous avons parfois fait le choix de ne pas utiliser de validation croisée mais plutôt un seul ensemble de validation afin de réduire le temps de calcul. En effet, les ensembles d'entraînements générés par la méthode TimeSeriesSplit étant très proches les uns des autres, il s'agissait a priori d'une concession raisonnable.

4.3 Pré-traitement et Représentation des données

4.3.1 Elimination de données

Après une rapide analyse de la distribution de la durée des matchs, nous avons choisi de ne conserver que les matchs de durées supérieures à 35 minutes et inférieures à 665 minutes de l'ensemble d'entraînement. De plus, nous avons supprimé de l'ensemble des données complet les matchs dont les scores faisaient apparaître des mentions identifiées comme anormales ("w/o", "ret" et "def") et ceux, minoritaires, dont les données étaient incomplètes (valeurs manquantes).

4.3.2 Choix des attributs

Concernant la quantité des attributs, on cherche généralement à ne sélectionner que les attributs les plus pertinents, c'est-à-dire ceux qui permettront de minimiser le mieux possible la fonction de coût finale, et à se débarrasser d'un certains nombre d'attributs moins intéressants afin d'échapper à la malédiction de la dimension, qui tend à augmenter la distance entre les points de l'espace et à rendre le problème de régression moins évident à résoudre. D'autres techniques de réduction de dimensions comme l'analyse de composantes principales ou l'analyse discriminante linéaire peuvent également être utilisées. Cependant, dans le cas des forêts aléatoires, comme expliqué dans ici, une sélection

"greedy" est généralement utilisée dans les algorithmes d'arbres de décision pour choisir l'attribut utilisé pour séparer un noeud en plusieurs noeuds. Ainsi, tous les attributs sont testés indépendamment à chaque noeud, ce qui augmente certes linéairement le temps de calcul en fonction du nombre d'attributs testés mais semble moins sensible à la malédiction de la dimension que d'autres modèles. Ainsi, peu d'attention à été portée dans ce projet à la sélection d'attributs, et presque tous ont été conservés. De même, nous n'avons pas envisagé l'utilisation de techniques de réduction de dimension, de nombreux attributs étant catégoriels.

A posteriori, cela n'est probablement pas idéal car cela ne se généralise absolument pas à la majorité des modèles existants. De plus, la sélection "greedy" ne cherchant pas à minimiser la fonction de coût associée à l'arbre entier mais seulement celle calculée à partir de celles des noeuds fils, on cherche parfois à limiter le nombre d'attributs testés lors de chaque sélection à l'aide de l'hyperparamètre "max_features" afin d'augmenter la diversité des arbres de la forêt. Alors, il est important que chaque attribut testé soit potentiellement intéressant, et que tous les attributs jugés comme "inutiles" aient été filtrés.

En ce qui concerne la qualité des attributs sélectionnés, il peut être intéressant de créer de nouveaux attributs qui pourraient apporter des informations supplémentaires, comme c'est le cas des différences de certains attributs associés à deux joueurs différents, comme le rang ou les points ATP. Il est également essentiel de ne pas oublier de se débarasser de tout attribut ou de toute information obtenue à l'issue du match, car ces informations ne seraient pas disponible au moment de la prédiction sur l'ensemble de test. Pourtant, ces informations comme le nombre de services gagnants, le nombre de doubles fautes etc. peuvent être prise en compte à l'aide de la création d'attributs basés sur des données historiques seulement, que nous décrirons plus en détail dans la sous-partie dédiée 4.5.

4.3.3 Découpage des ensembles d'entraînement

Plusieurs travaux [1] [3] considèrent que les données forment des groupes qu'il est préférable de ne pas comparer entre eux. Ainsi, le nombre de sets gagnants répartit les matchs en deux groupes (best-of-3 et best-of-5) et il est également possible de séparer les données par surface jouée, sous prétexte que les joueurs n'ont pas le même niveau d'une surface à l'autre. Cependant, les arbres de décisions sont capables de recréer ces séparations eux-mêmes, et il résulte du test 6 du tableau fourni en annexe 1 qu'il n'est pas préférable d'utiliser un modèle par nombre set gagnant dans le cas des forêts aléatoires, probablement car cette approche diminue le nombre de données utilisées par modèle.
Cette segmentation des données a cependant été utilisée lors du tests d'autres modèles, comme nous le verrons plus tard 4.4.2, lorsqu'il n'était pas évident que le modèle effectuerait la séparation correctement lui-même. Il semble cependant dommage de devoir réaliser cette séparation manuellement car le rôle des modèles d'apprentissage automatique est également d'apprendre à effectuer celles-ci en autonomie.

4.3.4 Tests de différentes représentations des informations sur les joueurs

Chronologiquement, les tests décrits ici ont été réalisés après le choix du modèle de régression, qui faisait alors intervenir la représentation des données avec duplication des matchs décrite dans 2.1.3.

Conformément à ce qui a été introduit dans 2.1.3, le but de cette partie est de sélectionner l'une des deux représentations candidates des données. Le constat initial est que la symétrie de l'ensemble des données apparaissant lors de la duplication de celui-ci nécessite l'utilisation d'assymétrie dans le modèle utilisé afin de pouvoir prendre en compte les attributs symétriques, comme ceux issus de différences entre les joueurs. Ce constat a été mis en évidence à l'aide des shaps plots associés à la représentation utilisant une duplication des matchs mais sans utilisation de valeurs absolues. Ces figures permettent de mettre en évidence la contribution des différents attributs dans le calcul de la durée prédite. On peut ainsi y observer que les attributs résultant de différences peuvent correspondre à des contributions opposées pour une même valeur, comme on peut le voir sur la figure 8 présente plus loin dans le document. Afin d'éviter cela, nous avons cherché à introduire de l'assymétrie directement via une transformation des attributs symétriques, en l'occurence par le biais de l'utilisation de valeurs absolues. L'idée était que cela permettrait de réduire le nombre d'arbres ainsi que leur profondeur, qui sont nécessaires pour gérer ce problème de symétrie. Cependant, cela induit une perte d'information. En effet, si l'on considère uniquement par exemple l'attribut associé à la valeur absolue de la différence des rangs, celui-ci ne permet plus d'identifier lequel des joueurs possédait le rang le plus important. Il n'est donc pas évident de choisir entre une représentation utilisant des valeurs absolues et une autre sans valeurs absolues, d'où la nécessité des tests suivants.

Pour évaluer les modèles associés à ces tests, un ensemble de validation d'une période de quatre années a été utilisé, s'étendant du 29/08/2017 au 29/08/2021. Tous les tests sont réalisés pour une forêt aléatoire peu profonde avec une profondeur maximale de 7 et un nombre d'élément minimal par feuille de 51, qui correspond au modèle sélectionné. Il n'est donc absolument pas garanti qu'elle soit adaptée à d'autres modèles très différents.

Le tableau 4.3.4 présente ainsi les résultats de ces tests. Finalement, la représentation sélectionnée est celle qui conserve le plus d'informations, c'est-à-dire sans valeurs absolues. Il est à noter que c'est celle-ci avait été utilisée dès le début du projet, d'où l'absence de répercussion de ces tests dans le tableau 1 de l'annexe 1.

Description de la représentation testée	Train score	Weighted train score	Test score
Duplication des matchs, pas de valeur absolue	0.243	0.38	0.2429
Duplication des matchs et valeur absolue des attributs issus de différences	0.245	0.41	0.2418
Valeur absolue des attributs issus de de différence sans duplication des matchs. A la place, on remplace les attributs associés aux joueurs par les minima et maxima associés. Ce test a été sélectionné parmi un ensemble d'autres tests correspondant à des représentations de données sans duplication et avec valeurs absolues, car c'est celui qui obtenu les meilleurs résultats parmi ces derniers.	0.240	0.43	0.2408

Tableau présentant les tests de 3 représentations de données différentes et les R2-scores associés.

4.4 Choix du modèle, entrainement et optimisation des hyperparamètres

4.4.1 Voir si la description de la cross-validation plus haut suffit à parler du problème d'overfitting rencontré

4.4.2 Choix du modèle et optimisation des hyperparamètres

Le modèle le plus utilisé tout au long de ce projet est le modèle de forêt aléatoire. Dans un premier temps, cela était dû au choix réalisé lors du projet précédant et dans un second temps, aux résultats des tests que nous décrirons dans cette partie. Enfin, il s'est avéré par la suite qu'un article [5] dans lequel l'auteur cherche à prédire la durée de matchs de cricket faisait également usage d'une forêt aléatoire pour obtenir leurs meilleurs résultats.

Les résultats d'un modèle de forêt aléatoire peu profonde puis profonde apparaissent pour la première fois lors des tests d'indice 1 et 5 du tableau 1. Ces résultats sont comparés à ceux obtenus pour un SVM (Machine à Vecteurs de Support) dont les hyperparamètres ont été optimisés, et à ceux obtenus pour un Gradient Boosting Regressor de sklearn, avec des hyperparamètres par défaut. Ce dernier était utilisé en premier lieu pour chercher à estimer la qualité des prédictions fournies, ce sur quoi nous reviendrons plus tard (5.3). Pour chacun de ces deux derniers modèles, la base de données a été séparée en deux sous-ensembles, conformément à ce qui a été vu en 4.3.3, et les attributs ont été normalisés (sauf pour les attributs encodés en one-hot). En revanche, les résultats obtenus pour le SVM démontraient un fort surapprentissage dont nous avons rapidement abandonné l'identification de la cause, bien que la démultiplication de choix de modélisation adaptés spécifiquement aux forêts aléatoire en soit probablement la cause. Ces tests ne permettent donc pas d'éliminer ces modèles d'un point de vue scientifique, mais nous ont en pratique

orienté vers le maintien de l'utilisation de forêts aléatoires. Ce choix est également justifié par le fait que les améliorations suivantes (poids de vieillissement, attributs additionnels) sont a priori utilisables pour d'autres modèles que les forêts aléatoires.

4.4.3 Prise en compte du vieillissement des données

On a vu plus tôt (2.1.1) qu'il était important de différencier les données en fonction de leur date, ce que l'on a choisi de faire par le biais de poids associés aux différents matchs. Ces poids de vieillissement sont calculés à partir de l'hyperparamètre λ. Or, bien qu'il soit possible de réutiliser directement des valeurs issues de la bibliographie, la valeur optimale de cet hyperparamètre peut dépendre des autres hyperparamètres du modèle (Voir aussi 4.1.1 sur la pertinence de ces tests). Ainsi, on a cherché à déterminer λ pour deux des meilleures combinaisons d'hyperparamètres obtenues pour le randomForestRegressor, qui représentent une forêt "profonde" et une forêt "peu profonde". La figure 5 décrit les résultats des tests obtenus.

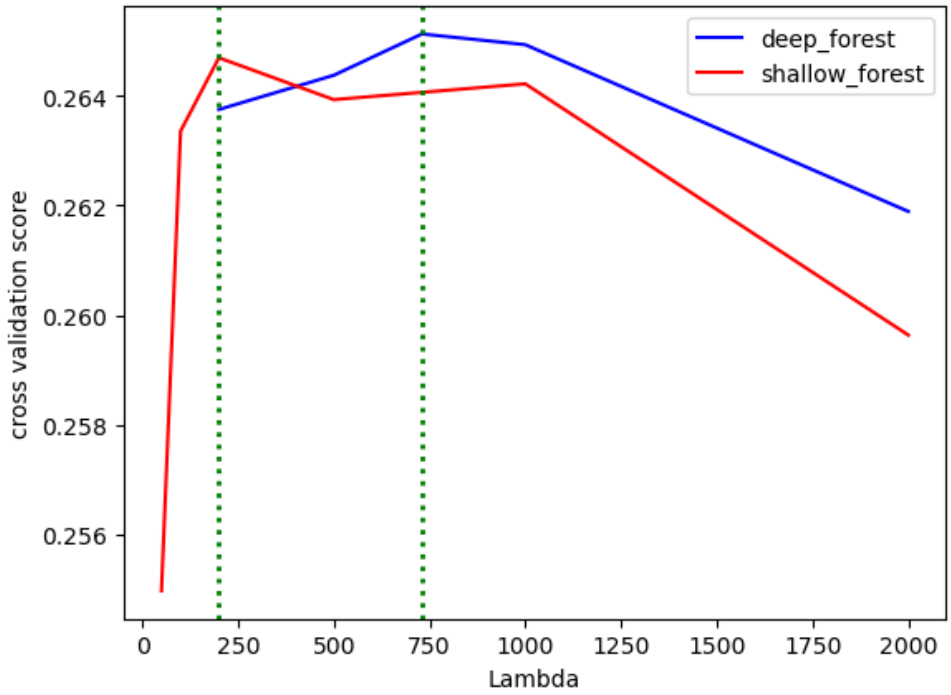

FIGURE 5 – R2-scores sur un ensemble de validation de 4 années de deux modèles RandomForestRegressor, en fonction du poids attribué aux données en fonction de leur vieillissement

On observe que l'hyperparamètre optimal λ dépend du modèle testé. Conformément aux travaux de [6]("using a form decay with a half-life of 240 days"), nous avons finalement choisi une valeur de 250 pour le modèle peu profond, et de 730 (=2*365) pour le modèle profond. Il semble ainsi qu'un modèle faisant intervenir un plus grand nombre de paramètres nécessite un nombre plus important de données pour l'entraînement pour éviter le surapprentissage. Or, comme expliqué en 2.1.1, une valeur plus grande de λ semble conduire à un nombre de données virtuellement plus grand, ce qui pourrait expliquer qu'une valeur de λ plus grande donne de meilleurs résultats pour le modèle profond.

En revanche, bien que le nombre de données soit virtuellement plus important, elles sont a priori de moins bonne qualité car la moyenne pondérée de leur date est plus ancienne.

Malgré tout ce qui a été avancé, on peut néanmoins argumenter que si les règles du tennis ne changent pas, que les conditions des matchs restent les mêmes et que les styles de jeu et méthodes d'entrainement n'évoluent pas (ces hypothèses sont bien sûr non vérifiées), ainsi que d'autres facteurs plus ou moins cachés, on ne voit pas pourquoi la répartition dans l'espace des niveaux et des performances des joueurs varierait au cours du temps : Statistiquement, il y aurait tout à parier que des profils de jeunes joueurs similaires viendraient à remplacer ceux des anciens joueurs chaque année ou tous les 10 ans. Ainsi, comme les modèles utilisés dans l'optimisation de l'hyperparamètre λ ne prenaient alors pas en compte d'informations nécessitant l'identification des joueurs (ces modèles n'utilisent pas leur nom ni de données additionnelles relatives à leur historique, mais seulement des attributs comme leur taille, leur rang, leur age), on pourrait s'attendre à ce que λ tende vers l'infini et à ce que les poids associés tendent vers 1 dans le cas où les hypothèses citées plus haut seraient vérifiées. Ce serait alors plutôt la non vérification des hypothèses citées plus haut qui justifierait la nécessité de l'utilisation de poids de vieillissement, contrairement à notre première intuition qui avait été évoquée dans 2.1.1.

4.5 Création d'attributs additionnels

La dernière principale étape du projet abordée correspond à la création d'attributs basée sur l'historique des matchs de chaque joueur. En effet, les données utilisées font intervenir beaucoup d'informations relatives aux services. Il semble donc intéressant de créer de nouveaux attributs basés sur les données des matchs passés pour chaque joueur. En pratique, malgré la charge de travail importante représentée par le calcul de ces attributs, nous avons décidé de les implémenter en raison de leur importance significative dans plusieurs travaux [2] [1].
Bien que certains auteurs [2] proposent d'utiliser directement des moyennes calculées sur l'ensemble des matchs disputés par chaque joueur, cette méthode de calcul ne semble pas fiable. En effet, comme Michal Sipko le fait remarquer dans son mémoire [1], très utilisé dans cette sous-partie, il ne semble pas pertinent de comparer le pourcentage d'"aces" effectués par un joueur affrontant majoritairement des joueurs du top niveau et celui correspondant à un joueur affrontant des adversaires en moyenne largement moins forts. De plus, il est important de prendre à nouveau en compte les poids calculés dans 4.4.3. Pour résoudre ce problème, Sipko et nous-mêmes utilisons la stratégie des "adversaires en commun" : pour chaque match, on détermine les adversaires qui ont affronté les deux joueurs de ce matchs au moins une fois dans le passé, et on se base sur ces affrontements uniquement pour obtenir créer des attributs comparables, issus de différences en pratique. Cependant, il apparaît que les attributs calculés à partir d'un seul affrontement, ancien, par joueur seront beaucoup moins fiables que ceux calculés à partir de plusieurs affrontements récents. C'est pourquoi [1] introduit le calcul d'incertitudes associées à ces attributs calculés, qui seront utilisées plus tard dans la sélection des données d'entraînements. Ces incertitudes sont définies dans [1] comme suit :

$$U = \frac{1}{\sum_j S_1(C_j) \cdot S_2(C_j)}$$

où

$$S_i(C_j) = \sum_{m \in P_i(C_j)} W(m)$$

où $W(m)$ est la somme des poids associés au match m et $P_i(C_j)$ est l'ensemble des matchs disputés entre le joueur i et l'adversaire C_j.

Ensuite, concernant le choix des attributs calculés, nous nous sommes basés sur les travaux de [2] qui explicitaient les calculs et donnaient des informations sur les attributs les plus utilisés (ce dernier point étant aussi assuré par [1]).

Enfin, il s'agit d'intégrer ces nouveaux attributs dans la base de données, de trier les matchs par incertitudes croissantes et de ne garder qu'un pourcentage des données d'entraînement correspondant aux incertitudes les plus faibles, le reste étant considéré comme étant du bruit, l'incertitude étant trop grande. Ce pourcentage doit être déterminé à l'aide de tests et ne peut pas être récupéré depuis la bibliographie car il dépend du paramètre λ choisi pour le calcul des attributs, du modèle choisi (ainsi que du paramètre λ associé au modèle). Voici donc les résultats de ces tests sur un ensemble de validation d'une durée d'une année, l'année précédant celle de l'ensemble de test : 6.
Il est à noter que le calcul des attributs additionnels étant très long, l'ensemble des données d'entrainement a été restreint en se basant sur l'évanescence des poids utilisés : on n'a considéré que les données à partir de 2010 pour une valeur de λ de 250, et les données à partir de 2005 pour une valeur de λ de 730. Il est également à noter que les ensembles d'entraînements n'étant pas de la même taille pour différentes valeurs de λ, les seuils ne sont pas comparables d'une valeur de λ à l'autre.

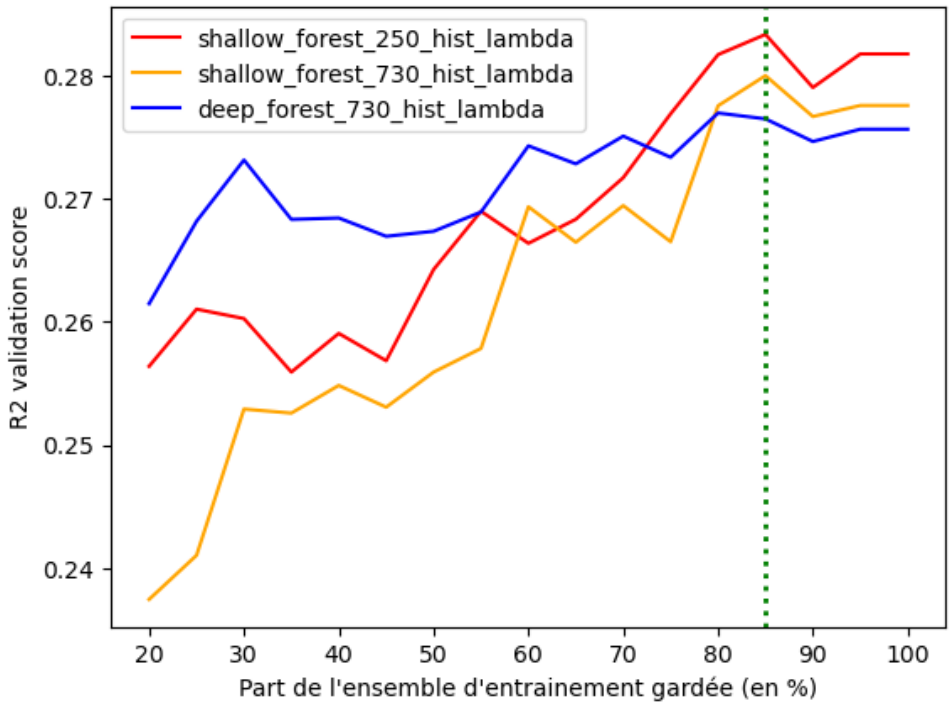

FIGURE 6 – R2-scores sur un ensemble de validation d'une année de deux modèles RandomForestRegressor, en fonction de la part de l'ensemble d'entrainement conservée et de l'hyperparamètre λ utilisé pour construire les attributs additionnels

On observe que même sur l'ensemble calculé à partir d'une valeur de λ de 730, la forêt peu profonde obtient de meilleurs résultats sur l'ensemble de validation (on garde bien une valeur de λ de 250 pour l'entraînement du modèle). Les meilleurs résultats correspondent finalement au modèle peu profond, pour une double utilisation d'une valeur de λ de 250, et pour un seuil correspondant à une conservation de 85% des données de l'ensemble d'entraînement.

4.6 amélioration du livrable

Initialement, un des objectifs du projet correspondait à l'amélioration du livrable, et principalement à la mise en place d'un mot de passe sur l'application ainsi que d'une fonction pour mettre à jour les données automatiquement. La fonction de mise à jour automatique des données a été implémentée sur python et permet de récupérer automatiquement les données mises à jour sur le github de Jeff-Sackmann. Quant au mot de passe, il s'est avéré que la page web MyBinder initialement utilisée dans le projet précédent pour créer relativement facilement une interface Web n'est compatible qu'avec des répertoires publics, ce qui est incompatible avec la mise en place de mots de passe. Nous avons donc finalement décidé de se concentrer sur d'autres aspects du projet et de ne pas chercher à mettre en place une application Web de A à Z.

5 Résultats pour un problème difficile

5.1 Du bruit difficile à quantifier

L'idéal serait que la connaissance parfaite des données d'un match permette de déterminer son issue ou sa durée. Cependant, il est clair qu'à partir des attributs constituant la base de données utilisée, il est impossible de prédire la durée de manière exacte. En effet, différents éléments ayant un impact majeur sur le déroulement d'un match comme la motivation du joueur ou même son épuisement physique et mental sont absents de la base de données et bien qu'il soit possible de chercher à les quantifier et à les estimer, cela reste une tâche loin d'être évidente. Ces données manquantes semblent donc être une source importante de bruit, qui peut rendre très difficile notre tâche qui consiste à estimer la relation entre la durée des matchs et les attributs décrivant ces matchs et qui sont disponibles.
Une estimation de l'importance de ce bruit (i.e. de la part de la durée du match qui est inexplicable à l'aide de nos données) pourrait être intéressante pour pouvoir estimer l'erreur de prédiction provenant de notre modélisation en opposition à celle provenant des données utilisées. En effet, il pourrait apparaître qu'aucune modélisation ne permet de valider les exigences concernant la qualité des prédictions à fournir (exigences qui n'ont par ailleurs pas été définies dans ce projet, le but étant d'atteindre les meilleurs résultats possibles). Cependant, il m'a semblé difficile d'estimer l'importance de ce bruit au vu de la taille de l'espace des données (Une idée relativement simple avec un espace très réduit consisterait par exemple à considérer la répartition de la durée des matchs pour différents clusters de données, ou même pour différentes données identiques pour un espace très réduit) et j'ai préféré me concentrer sur l'amélioration du modèle.

5.2 Des résultats peu satisfaisants

Malgré toutes les optimisations réalisées, le R2-score du meilleur modèle sur l'ensemble de test reste relativement faible, étant de 0.273, ce qui parait loin de l'objectif de 1. Il a aussi été mis en évidence que pour ce modèle, seules 20.1% des durées des matchs sont correctement prédites à 10 minutes près, et 40.2% à 20 minutes près, ce qui ne semble pas non plus satisfaisant.

Ces écarts entre les durées réelles et prédites apparaissent notamment sur la figure 7 permettant de comparer visuellement la différence entre les durées prédites par le modèle et les durées réelles. On peut déduire de cette figure que le modèle parvient difficilement à faire contribuer d'autres attributs que le "best-of" à la prédiction des durées, ce qui se confirme d'ailleurs sur le shap plot associé à ce modèle 8.

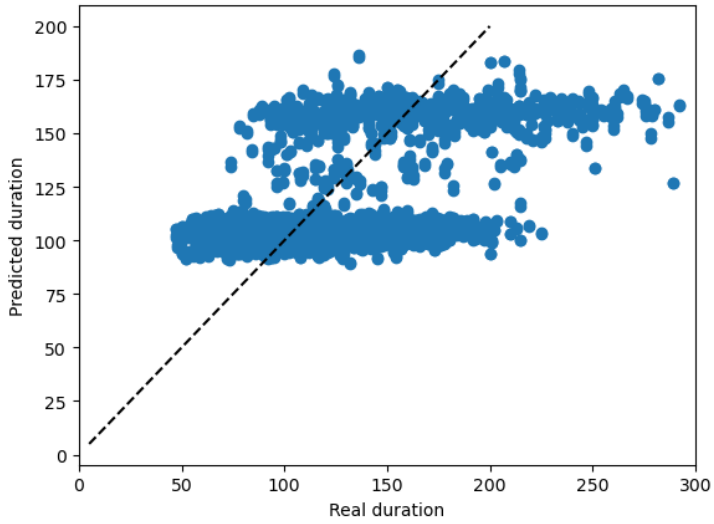

FIGURE 7 – Distribution des durées prédites en fonction des durées réelles

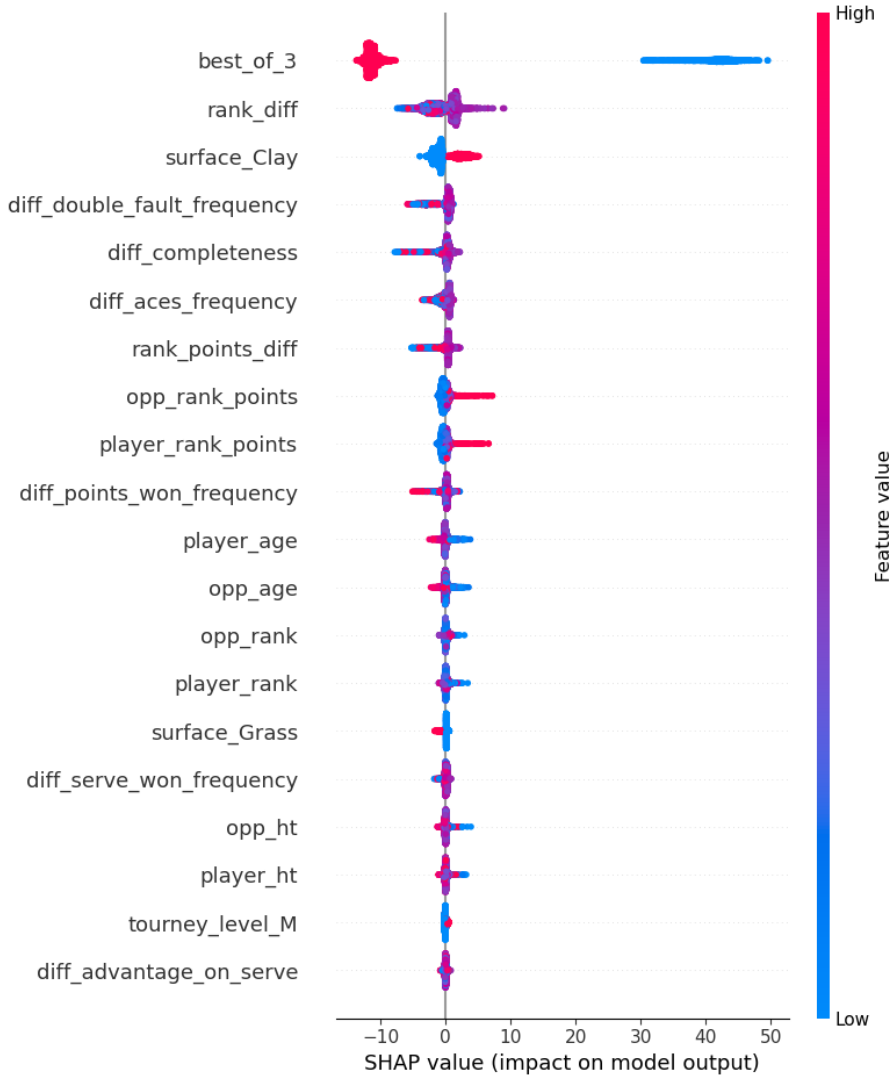

FIGURE 8 – shap plot associé au meilleur modèle. On peut y observer la contribution de chaque attribut à la durée prédite.

Ces résultats peu satisfaisants proviennent probablement en grande partie de la nature très incertaine du déroulement des matchs de tennis. En effet, d'après [4], mêmes les bookmakers, qui peuvent également prendre en compte des éléments additionnels comme les blessures des joueurs ou même des éléments de leur vie personnelle, ne parviennent qu'à prédire l'issue du match que dans 78% des cas, 50% étant le pire résultat possible.

5.3 Tentatives d'estimation de la fiabilité des prédictions

Il était également question dans ce projet d'estimer la qualité des prédictions fournies. Cela est possible à l'aide de modèles permettant de faire de la régressions de quantiles, comme le Gradient Boosting Regressor utilisé. Les quantiles prédits ont la forme suivante 9. On peut cependant noter (et même observer) qu'au vu du score obtenu pour ce modèle sur l'ensemble de test, les quantiles estimés sont loin d'être fiables et suffisants pour garantir la qualité des prédictions fournies.

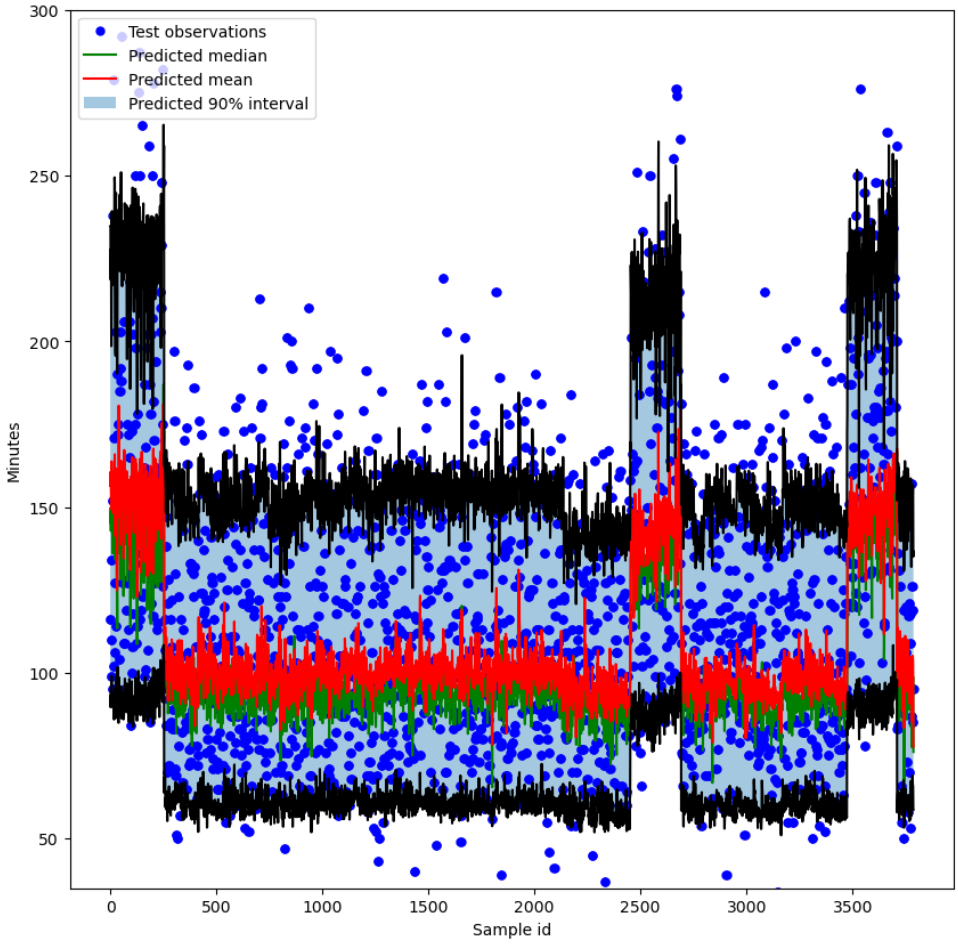

FIGURE 9 – Estimation de quantiles qui permettraient de donner des indications quant à la qualité des prédictions fournies

6 Conclusion et perspectives

Ce projet, qui m'avait paru simple dans un court premier temps, s'est avéré être un vrai défi en raison de la nature finalement complexe des données utilisées et particulièrement en raison de leur obsolescence progressive. De très nombreuses optimisations ont ainsi été réalisées afin de mieux représenter les données, en m'inspirant à la fois de mon expérience, de mes connaissances, mais également en grande partie de différents travaux de la bibliographie. Bien que les prédictions finalement obtenues ne soient pas à la hauteur de nos espérances, les résultats ont été légèrement améliorés et les travaux réalisés pourront être réutilisés dans un projet ultérieur.
Ainsi, ayant souhaité aller le plus loin possible dans le projet, il a été convenu avec l'encadrant qu'afin de préparer la passation du projet, le travail de mise en forme du code et du répertoire et de mise à jour de l'application sera réalisé peu après la soutenance comme c'était le cas pour le projet de l'année dernière. Enfin, les traitement de données s'étant sans cesse multiplié, plusieurs pistes n'ont pas pu être explorées lors de ce projet. Parmi celles-ci, on retrouve l'exploitation de données exogènes, comme celles des bookmakers, dont l'accès semble pourtant restreint, ou encore la mise en place de réseaux de neurones qui seraient adaptés à cette tâche de prédiction des durées. De plus, l'utilisation de bases de données plus récentes pourraient également permettre l'implémentation d'attributs estimant l'état psychologique des joueurs, leur fatigue, l'inertie de leurs derniers matchs. Enfin, une piste également prometteuse pourrait correspondre à la prise en compte de la structure hiérarchique du système de points au tennis, en cherchant par exemple à estimer la durée d'un point ainsi que le nombre de points disputés.

A Principales modélisations testées

Dans la table suivante, seuls les tests donnant de la matière pour la justification des choix de modélisation ont été sélectionnés, dans un soucis de concision. Il n'est probablement pas d'usage de faire apparaître ce genre de table. Cependant, j'ai malgré tout choisi de la présenter ici afin de rendre compte en toute transparence de l'évolution de la modélisation et de justifier les choix qui ont été faits.
En ce qui concerne les noms de colonnes non explicites, "From" fait référence à l'"Id" de la modélisation utilisée comme point de départ de la modélisation associée au test en question, "Weighted train score" correspond au R2-score calculé en prenant en compte les poids associés aux données d'entrainement, et "Ensemble testé" correspond à l'évolution qui aurait de préférence dûe être évitée de l'ensemble de test (1 : ensemble de matchs choisis au hasard dans l'ensemble d'entrainement, 2 : ensemble des matchs disputés entre Janvier et Août 2022, 3 : ensemble des matchs disputés entre Août 2021 et Août 2022, 4 : Ensemble 3 avec suppression d'outliers pour le calcul des attributs additionnels).

Table 1: Table décrivant les principales étapes de l'évolution de la modélisation et les R2-scores associés sur différents ensembles.

Id	From	Evolutions du modèle testé	Commentaires généraux et remarques	Train score	Weighted train score	Test score	Ensemble testé	Conclusion tirée
1	/	RandomForest Regressor (RFr), best_params = {'max_depth' : 7, 'min_samples_leaf' : 51}, entraînement sur l'ensemble des matchs complet.	Peu d'hyperparamètres ont été testés, par GridSearch, et ceux-ci correspondaient à des arbres peu profonds. La méthode de validation croisée utilisée était inadéquate.	0.26	/	0.255	1	
2	/	Un modèle RFr entraîné par joueur : l'ensemble d'entraînement correspond uniquement aux matchs disputés par le joueur en question.	54 des 312 noms testés n'apparaissent pas dans l'ensemble d'entraînement, ce qui peut justifier le score anormalement bas.	0.18	/	-7.32	1	Il semble que le nombre de données utilisé dans les ensembles d'entraînement est trop faible
3	1	Ajouts des poids de vieillissement des données, représentation des données avec duplication des matchs, prise en compte d'attributs calculés à partir de différences.	Il s'agit du dernier test réalisé sans ensemble proprement défini.	0.24	0.27	0.278	1	Chacune des améliorations citées contribue indépendamment à l'amélioration du score.
4	3	Elimination d'outliers	Ensemble de test clairement défini et récent.	0.27	0.31	0.218	2	Sert de référence. On constate du surapprentissage, probablement lié au vieillissement des données.

5	4	Nouvelle optimisation du RFr. Paramètres sélectionnés : {'n_estimators' : 400, 'min_samples_split' : 100, 'max_features' : 'sqrt', 'max_depth' : 60}	L'ensemble des hyperparamètres étant cette fois très large, RandomSearchCV a ici été utilisé. Première utilisation de TimeSeriesSplit pour la validation croisée.	0.36	0.47	0.215	2	Un modèle profond apprend davantage sur l'ensemble d'entraînement, mais les résultats sur l'ensemble testé restent comparables à ceux du modèle peu profond.
6	4	Optimisation du RFr sur deux bases de données différentes, une pour les « best-of-3 » et une pour les « best-of-5 ».		?	?	0.209	2	Il est plus pertinent de conserver l'ensemble des données complet pour un modèle capable d'effectuer la séparation lui-même, comme un RFr.
7	4	Gradient boosting Regressor avec quantiles. learning_rate :0.05 n_estimators :200 max_depth :2 min_smpl_leaf :9 min_smpl_split :9		0.31	/	0.19	2	Moins performant que la RFr, mais permet d'estimer des quantiles.
8	4	Optimisation d'un SVM sur 2 sous-ensembles : {'gamma' : 10.0, 'C' : 1000.0} pour le best-of-5 et {'gamma' : 10.0, 'C' : 100.0} pour le best-of-3	R2-scores négatifs sur les ensembles de validation et de test	0.63	/	0.153	2	On observe un très grand surapprentissage malgré l'optimisation des hyperparamètres.

9	4	Test après optimisation du poids de vieillissement pour le RFr peu profond		0.19	0.37	0.244	2	Résultats significativement meilleurs.
10	9	Modification de l'ensemble de test		0.20	0.37	0.250	3	Sert de référence
11	5	Optimisation du poids de vieillissement pour le RFr profond et modification de l'ensemble de test	On passait de 0.215 à 0.223 sur l'ensemble de test avant sa modification	0.34	0.51	0.231	3	Surapprentissage, scores moins bons.
12	9	Modifications de l'ensemble de test : élimination d'outliers pour le calcul des attributs additionnels		0.20	0.40	0.263	4	Sert de référence
13	12	Modification de l'ensemble d'entraînement (seuil de 0.8) 4.5		0.30	0.35	0.270	4	Sert de référence. Amélioration des résultats liée au filtre sur l'ensemble d'entraînement.
14	13	Prise en compte des attributs additionnels	optimisation des hyperparamètres : {'n_estimators' : 400, 'min_samples_leaf' : 40, 'max_depth' : 7}	0.30	0.37	0.273	4	Très faibles améliorations par rapport au modèle sans attributs additionnels, ce qui est plutôt décevant relativement aux efforts investis pour mettre en place ceux-ci.

B Répertoire github

Voici le lien vers le répertoire github associé à mon projet : https ://github.com/leobeuque/PDI-Tennis . En particulier, seuls les fichiers regression-tree-leo.ipynb (grande majorité de mes implémentations), regression-tree.ipynb (test de la modélisation du projet précédent),

key-variable-analysis_leo.ipynb (analyse de l'ensemble des données) et atp_cat.csv correspondent à mes travaux. Le reste est issu du projet précédent, et est récupérable directement depuis https ://github.com/uporito/PDI-Tennis.

Références

[1] M. Sipko and W. Knottenbelt, "Machine learning for the prediction of professional tennis matches," *MEng computing-final year project, Imperial College London*, 2015.

[2] V. Candila and L. Palazzo, "Neural networks and betting strategies for tennis," *Risks*, vol. 8, no. 3, p. 68, 2020.

[3] J. Peters, "Predicting the outcomes of professional tennis matches," 2017.

[4] F. Lisi and G. Zanella, "Tennis betting : Can statistics beat bookmakers ?," *Electronic Journal of Applied Statistical Analysis*, vol. 00, pp. 1–35, 09 2013.

[5] S. Tyagi, R. Kumari, S. C. Makkena, S. S. Mishra, and V. S. Pendyala, "Enhanced predictive modeling of cricket game duration using multiple machine learning algorithms," in *2020 international conference on data science and engineering (ICDSE)*, pp. 1–9, IEEE, 2020.

[6] I. McHale and A. Morton, "A bradley-terry type model for forecasting tennis match results," *International Journal of Forecasting*, vol. 27, no. 2, pp. 619–630, 2011.