



DataScientest • com

# Rapport Technique d'Évaluation

## **Equipe :**

Philippe ARMANI  
Francis MAHANAM  
Noman GHULAM

## **Mentor :**

Mounir, DataScientest

**Projet : Goodtipster**

# 1 Table des matières

<b>1 Table des matières</b>	2
<b>2 Contexte</b>	4
<b>3 Définition de la problématique et objectifs</b>	6
3.1 Best Practices et Méthodologie suivie	6
3.2 Réponse à la problématique	7
3.3 Etudes préliminaires	7
<b>4 Présentation du dataset et preprocessing</b>	9
4.1 Structure du dataset et dataviz	9
Répartition des matchs par années	10
Top 10 des joueurs ayant remporté/perdu le plus de matchs	11
Nombre de tournois par surface	12
Top 10 des joueurs sur chaque type de surface	12
Taux de bonnes prédictions du bookmaker Bet365	13
Distribution des variables	13
Relations entre les variables	15
Identification des composantes principales	17
4.2 Preprocessing	17
Traitement des NaN	17
Suppression des colonnes et des lignes	18
Encodage des variables catégorielles	18
<b>5 Modélisation</b>	19
5.1 Transformation du dataset	19
Elaboration de la target	19
5.2 Projet et Modèles	19
RandomForest avec critère Gini	19
KNN	21
AdaBoost	22
Bagging	24
5.3 Scénario alternatif – Utilisation des Moyennes roulantes	25
5.4 Choix de la métrique et récapitulatif de la modélisation	26
<b>6 Simulation Pari Sportif</b>	29
<b>7 Conclusion</b>	32
Objectifs	32
Déroulement du projet et modèles choisis	32
Difficultés rencontrées	32

Bilan & Suite du projet.....	33
Pistes d'améliorations .....	33

## 2 Contexte

### 2.1 Contexte et présentation du projet

Ce projet a été réalisé dans le cadre de la formation de Data Analyst en format bootcamp. Au vu du temps imparti, ce fut un projet intense et complexe à mener étant donné le profil novice de chacun des membres. Le choix des paris sportifs a été motivé par l'intérêt des membres concernant ce milieu où les statistiques sont omniprésentes et où on a pu observer une forte émergence des plateformes de paris ces dernières années.

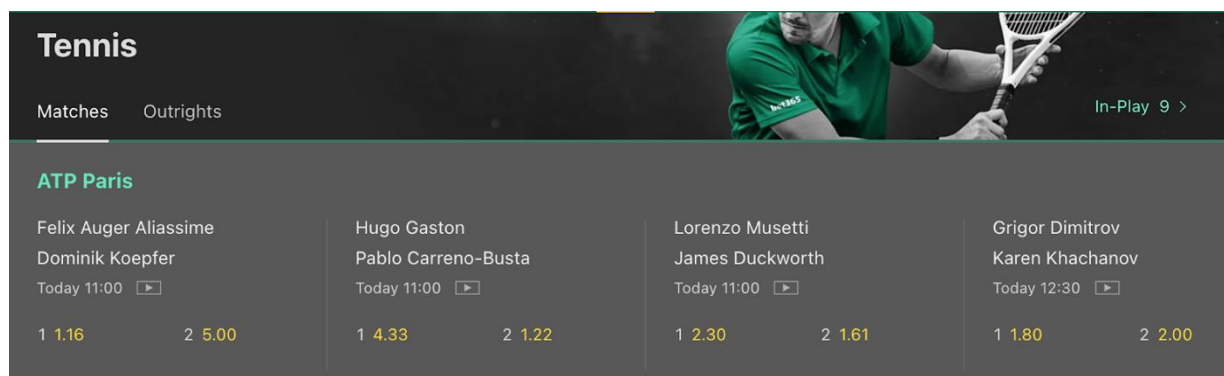
Au niveau technique, ce projet permet une approche conciliant l'industrie du divertissement à des notions de data science comme le machine learning ou encore la business intelligence.

Le dataset se focalise sur les rencontres du circuit ATP sur les deux dernières décennies.

### 2.2 Les paris sportifs

Le principe des paris sportifs est assez simple à appréhender. Un parieur va miser sur la réalisation d'un événement lors d'une rencontre sportive. Les possibilités sont multiples mais ici nous nous attarderons uniquement sur l'issue d'un match de tennis. Il s'agit donc de parier sur la victoire d'un joueur. Pour ce faire, le parieur va devoir analyser les cotes proposées par les sites de paris en ligne. La cote permet de savoir, en fonction du montant que l'on va miser, le gain potentiel que le parieur peut réaliser.

Exemple pris sur Bet365 pour le premier match :



The screenshot shows the Bet365 website interface for Tennis. At the top, there's a header with 'Tennis' and 'In-Play 9 >'. Below the header, there are tabs for 'Matches' and 'Outrights'. The main content area is titled 'ATP Paris' and displays four match cards. Each card lists the players, the start time, and the odds for both players.

Match	Player 1	Player 2	Time	1	2
Match 1	Felix Auger Aliassime	Dominik Koepfer	Today 11:00	1.16	5.00
Match 2	Hugo Gaston	Pablo Carreno-Busta	Today 11:00	4.33	1.22
Match 3	Lorenzo Musetti	James Duckworth	Today 11:00	2.30	1.61
Match 4	Grigor Dimitrov	Karen Khachanov	Today 12:30	1.80	2.00

En misant 100 € sur Felix Auger Aliassime coté à 1.16, on peut :

- en cas de victoire, gagner  $100 \times 1.16 = 116 - 100 = 16$  €
- en cas de défaite, perdre 100€

Le bookmaker fixe les cotes de manière à se garder une marge qui est calculée selon la formule suivante :

Marge =  $(1/1,16) \times 100 + (1/5 \times 100) = 86,2 + 20 = 106,2$  soit une marge de 6,2 %.

Pour une mise **m** et une cote **c**, un joueur peut avoir un profit **P** :

- profit si le pari est réalisé :  $(cm - m)$  avec une probabilité **p**
- dans le cas contraire :  $-m$  avec une probabilité  $(1 - p)$

L'espérance de gain est définie par la relation suivante :

$$E[P] = p(cm - m) - (1 - p)m = (pc - 1)m$$

En gardant les mêmes cotes et la même mise que précédemment :

- si le parieur perd le pari, le bookmaker empoche 100 €
- si le parieur gagne le pari, le bookmaker perd 16 €

Le gain du bookmaker lorsque le parieur perd sera 6,25 fois plus important que le gain du parieur si celui-ci gagne (100/16)

La cote agit comme un multiplicateur de la mise. Plus la cote est élevée, moins la réalisation d'un événement est probable. A l'inverse, plus la cote est faible, plus ce dernier est susceptible de se produire.

### 3 Définition de la problématique et objectifs

En analysant les performances de chaque joueur et de leur adversaire, il est possible d'émettre des hypothèses concernant l'issue d'une rencontre, notamment via l'utilisation des statistiques. Cette méthode nécessite d'être constamment informé, ce qui représente déjà une première difficulté pour un parieur. Par ailleurs, l'interprétation et l'utilisation de la data peut aussi être erronée ou biaisée par un parieur. La cote des bookmakers est un bon indicateur mais elle n'est pas suffisante et on peut penser qu'elle joue en faveur du bookmaker dont l'objectif est de maximiser ses gains, cela au détriment des parieurs.

Afin de répondre à ce besoin de neutralité, une solution est de construire un outil permettant d'évaluer les résultats des matchs et d'évaluer, selon une stratégie de paris, les gains potentiels en utilisant les cotes des bookmakers.

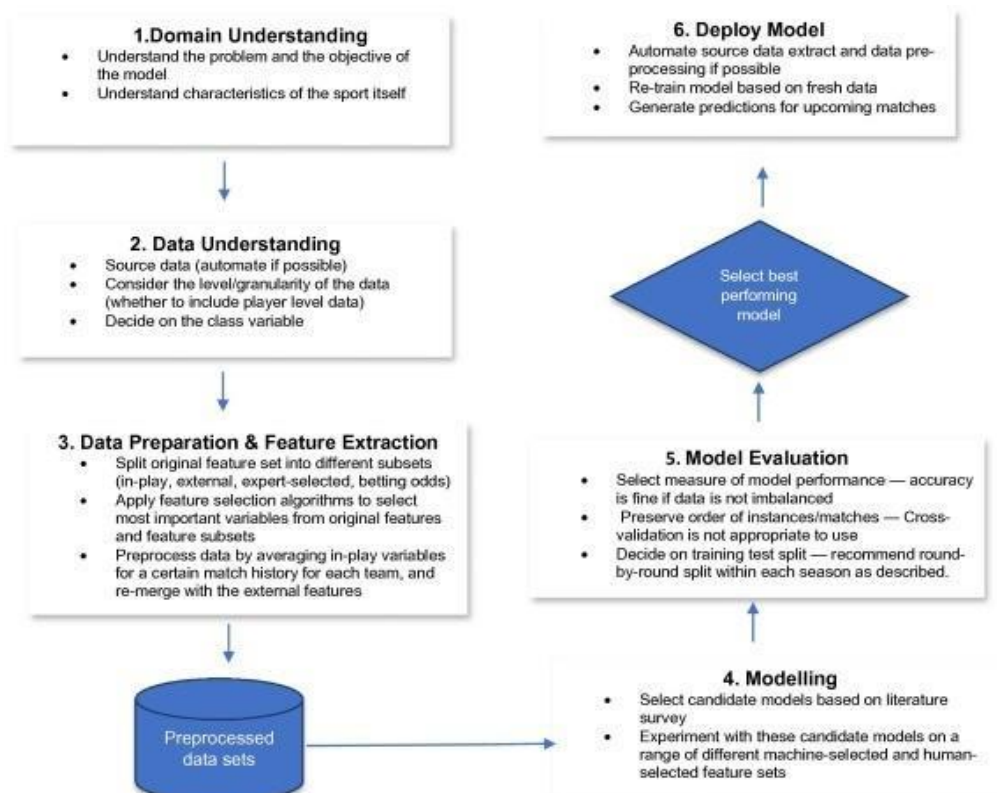
Pour réaliser cet outil, il est nécessaire d'avoir un historique qui permette de créer un modèle prédictif.

#### Incertitudes associées au projet

Une difficulté du projet réside dans la qualité des données. Il est certain que la qualité des données est déterminante dans l'affinement d'un modèle prédictif ainsi que de sa performance.

#### 3.1 Best Practices et Méthodologie suivie

Dans l'élaboration de notre projet, nous avons suivi une méthode classique pour un projet en data science. Cette dernière est détaillée dans le schéma ci-dessous :



### 3.2 Réponse à la problématique

La première étape consistait à réaliser un outil pour essayer de prédire l'issue d'un match, c'est-à-dire de déterminer le vainqueur et le perdant d'une rencontre. La cible étant le résultat du match.

Dans la seconde étape, nous avons créé un outil de simulation de gains. Au préalable, nous avons adopté une stratégie de paris. On parle de stratégie à partir du moment où un parieur est amené à miser sur plusieurs rencontres. Plusieurs critères peuvent influencer cette stratégie :

#### La maximisation des gains :

Dans cette stratégie, les mises sont faites sur les joueurs avec une cote importante. Cette stratégie est risquée car ce sont les joueurs les moins performants et donc avec très peu de probabilité de gagner. Le risque de pertes des mises est important

#### La minimisation des pertes :

A l'inverse de la stratégie précédente, les mises sont faites sur les joueurs avec une cote faible. Si elle semble moins risquée à première vue, cette stratégie est néanmoins risquée car les gains sont faibles. On constate qu'il suffit de quelques défaites pour avoir un bilan nul ou négatif.

Le dilemme d'un parieur est de maximiser ses gains en minimisant ses risques de perte. Il existe différentes manières de parier et il serait intéressant de voir s'il existe une stratégie optimale de pari. Dans le cadre de ce projet, faute de temps, nous ne ferons pas d'optimisation sur cette partie, le but premier étant d'avoir les meilleures prédictions possibles.

Par la suite nous ferons une simulation de paris tirée de l'utilisation d'un modèle prédictif.

### 3.3 Etudes préliminaires

Nous testons deux scénarios. Le premier prend pour cible le rang du joueur et le deuxième le résultat des matchs. Pour le scénario 1, nous testons des modèles de régression tandis que pour le scénario 2, nous testons des modèles de classification. La cible dans le scénario 1 est une variable quantitative d'où le choix d'un modèle de régression contrairement au scénario 2 où la cible est une variable catégorielle.



### Pour le scénario 1 :

Nous utilisons d'abord un modèle RidgeCV qui est une version régularisée de la Régression Linéaire où un terme de régularisation est ajouté à la fonction de coût. Le score (accuracy) est de 0.24 ce qui est très faible. En appliquant un gridsearch sur ce modèle, le score passe à 0.25 on observe donc un changement négligeable malgré l'optimisation des hyperparamètres.

Ensuite, nous testons le modèle Lasso. Le score est de 0.24 aussi et ce malgré l'utilisation des hyperparamètres optimaux.

Au vu des résultats obtenus, le scénario 1 n'est pas retenu.

### Pour le scénario 2 :

Le scénario 2 a pour but de prédire l'issue d'un match, c'est-à-dire de déterminer le vainqueur et le perdant d'une rencontre, la cible étant le résultat d'une rencontre.

Pour ce scénario, nous allons avoir recours à des modèles de classification, tous issus de la bibliothèque scikit-learn. Le premier modèle testé pour ce scénario est le RandomForest dont les atouts sont :

- la prise de décision avec plusieurs arbres de décisions
- l'entraînement de chaque arbre sur un sous ensemble aléatoire de données
- les prédictions évaluées selon un système de vote
- la non standardisation des variables numériques.

Nous avons ensuite testé KNN qui a l'avantage :

- d'être polyvalent
- d'être utile pour de grands ensembles de données changeant fréquemment

Puis nous avons testé Bagging qui a pour avantage de :

- corriger l'instabilité des arbres de décision

Nous avons enfin testé AdaBoost dont les avantages sont :

- de corriger de manière itérative les erreurs du classificateur faible et améliore la précision en combinant les apprenants faibles.
- AdaBoost n'est pas sujet au surajustement.

Nous avons donc retenu le scénario 2 qui se traduit par un problème de classification.



## 4 Présentation du dataset et preprocessing

### 4.1 Structure du dataset et dataviz

Dataset contenant les résultats sportifs de l'ATP pour les années 2000 à 2018 et également les cotes de deux bookmakers, Pinnacle Sports et Bet365.

Chaque ligne du dataset correspond à un match opposant un joueur à un autre. Il s'agit uniquement de jeu en simple.

Le dataset contient 23 colonnes dont les contenus sont :

- colonne 0 ATP : Nombre de tournoi(homme)
- colonne 1 Location : Lieu du match
- colonne 2 Tournoi : Nom du Tournoi
- colonne 3 Date : date du match
- colonne 4 Series : Nom de la série ATP
- colonne 5 Court : extérieur ou intérieur
- colonne 6 Surface : type de surface
- colonne 7 Round : tour d'un match
- colonne 8 Best of : nombre de sets maximum dans un match 3 ou 5
- colonne 9 Winner : nom du gagnant
- colonne 10 Loser : nom du perdant
- colonne 11 WRank : classement ATP mondial du gagnant
- colonne 12 LRank : classement ATP mondial du perdant
- colonne 13 Wsets : nombre de sets gagnés pour le gagnant
- colonne 14 Lsets : nombre de sets gagnés pour le perdant
- colonne 15 Comment : issue du match
- colonne 16 PSW : cote PINNACLE du gagnant
- colonne 17 PSL : cote PINNACLE du perdant
- colonne 18 B365W : cote B365 du gagnant
- colonne 19 B365L : cote +B365 du perdant
- colonne 20 elo\_winner : Elo avant match du joueur gagnant
- colonne 21 elo\_loser : Elo avant match du joueur perdant
- colonne 22 proba\_elo :  $1 / (1 + 10^{((elo\_loser - elo\_winner) / 400)})$

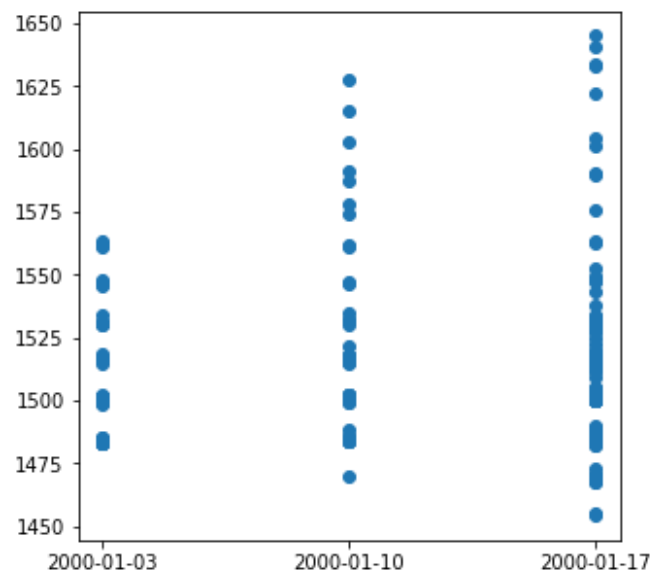
Les colonnes elo sont calculées ainsi :

- étape 1 : attribution d'un elo de 1500 à tous les joueurs initialement.

puis pour chaque ligne

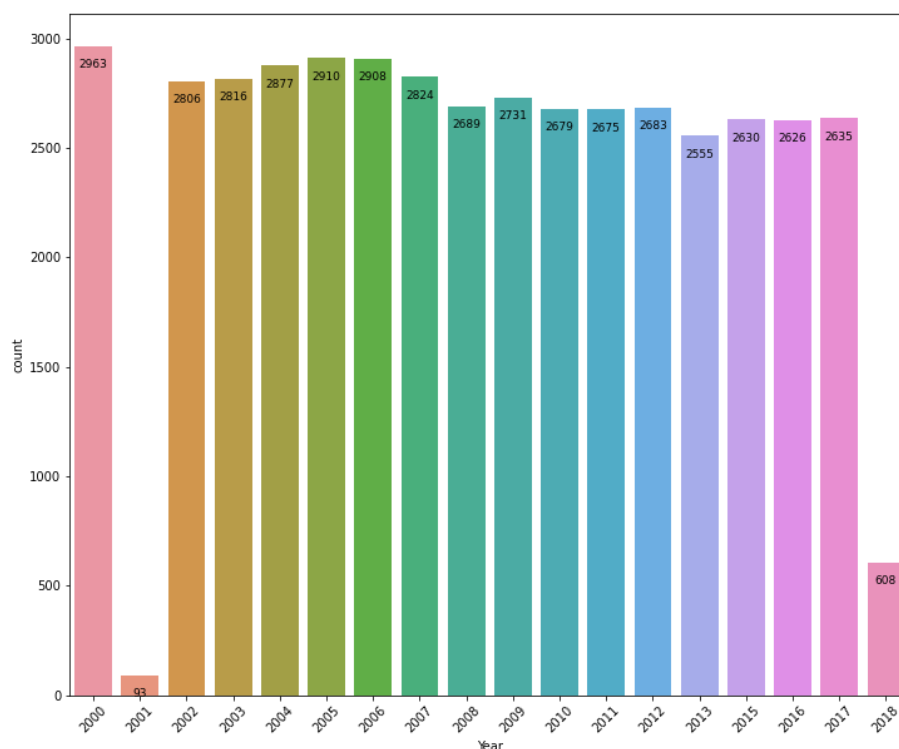
- étape 2 calcul de la proba\_elo :  $1 / (1 + 10^{((elo\_loser - elo\_winner) / 400)})$
- étape 3 calcul du elo\_winner :  $elo\_winner + 32 * (1 - proba\_elo)$
- étape 4 calcul du elo\_loser :  $elo\_loser - 32 * (1 - proba\_elo)$

Avec cette méthode, nous comprenons que les différences s'accroissent avec le nombre de matchs joués. A priori la méthode peut présenter un décalage par rapport aux vraies elos des joueurs. Cependant l'observation montre que, sur une échelle de temps, la dispersion (valeur max valeur min) se fait très rapidement. Ces valeurs représentent donc un bon indicateur sur le potentiel des joueurs.



La figure ci-dessus qui représente l'amplitude des elos des winners pour le mois de janvier 2000 (premier mois du dataset). Nous voyons que l'amplitude qui est de 80 pour le 3/1/2000 passe à 157 le 10/1/2000 et à 190 le 17/1/2000

## Répartition des matchs par années

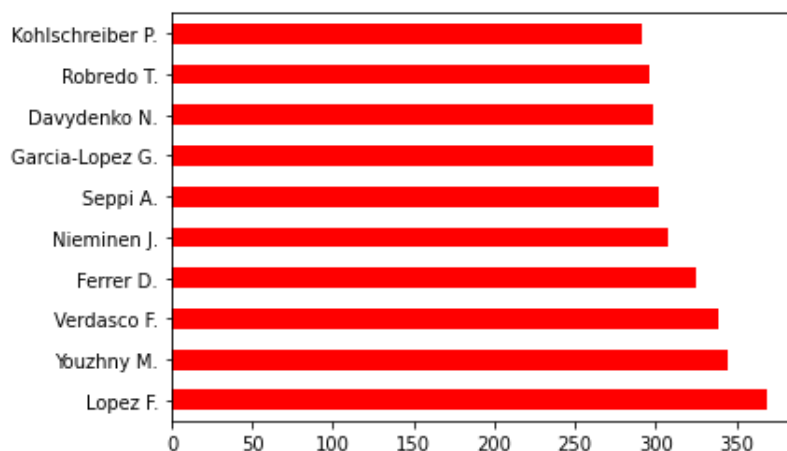
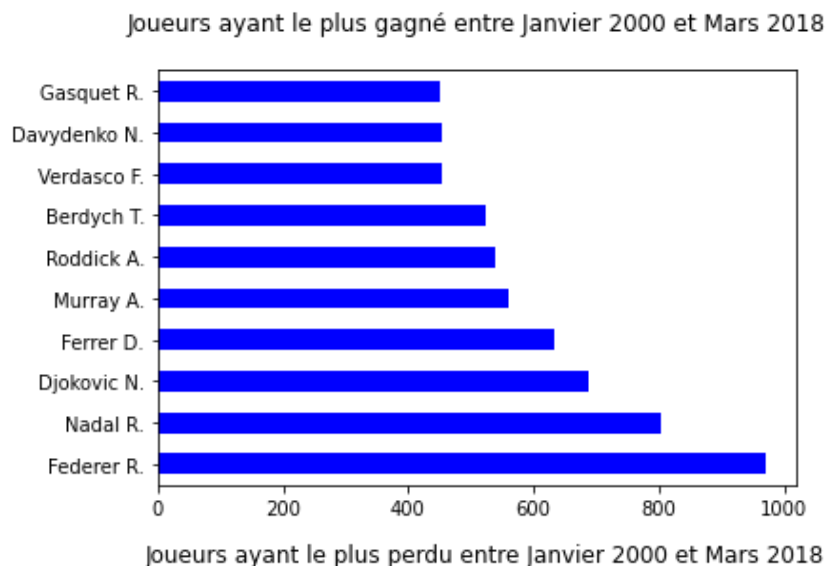


Nous constatons que les années 2001 et 2018 sont incomplètes. Par ailleurs, le nombre de matchs n'est pas toujours égal selon les années. Nous avons deux hypothèses pour expliquer les variations du nombre de matchs par année.

Hypothèse 1 : Certains tournois peuvent être supprimés ou ajoutés selon les années par les organisateurs.

Hypothèse 2 : Dans le traitement des données réalisé en amont, certains matchs ont été supprimés ou non traités.

## Top 10 des joueurs ayant remporté/perdu le plus de matchs



On retrouve sur les deux graphes Davydenko et Ferrer. Davydenko a gagné environ 450 matchs et perdu 300 matchs Ferrer a gagné environ 650 matchs et perdu 325 matchs. Si on retrouve ces deux joueurs dans les gagnants et les perdants, c'est parce qu'ils jouent plus de matchs que les autres du fait que les tournois font une sélection d'entrée. On peut constater que les joueurs qui perdent le plus ne correspondent pas aux joueurs ayant les plus mauvaises performances, au contraire ils s'identifient à des joueurs ayant des résultats très supérieurs à la moyenne. En regardant la distribution des matchs et des joueurs on observe que sur près de 1400 joueurs, plus de la moitié des joueurs disputent moins de 20 matchs sur la période 2000-2018. Ainsi un mauvais profil s'apparente davantage à un joueur qui joue peu et qui reste peu longtemps sur le circuit.

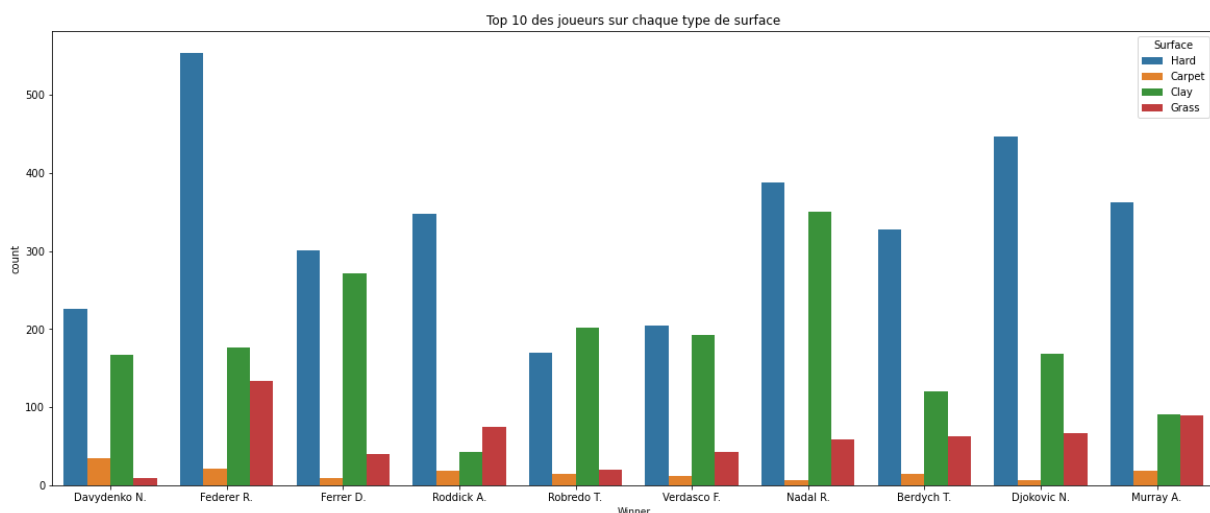
## Nombre de tournois par surface

Tournement	
Surface	
Carpet	11
Clay	80
Grass	19
Hard	107

Les surfaces prédominantes sont le dur puis la terre battue. Cette analyse nous amènera à comparer les performances des joueurs pour une même surface. On en déduit que la grande majorité des matchs se déroulent sur surface dure et sur terre battue mais très peu sur gazon ou bien sur une surface moquette.

Hypothèse 3 : On peut penser que la surface a une influence sur les performances des joueurs mais aussi le choix du tournoi.

## Top 10 des joueurs sur chaque type de surface



Sur le dur, Federer domine les autres joueurs alors que sur terre battue, Nadal domine le top 10. Chaque joueur a des préférences de surface, Federer est nettement meilleur sur le dur, alors que Nadal est pratiquement aussi bon sur le dur que la terre battue.

## Taux de bonnes prédictions du bookmaker Bet365

	accuracy_PS	accuracy_B365	dataset_size
<b>full dataset</b>	0.514002	0.602107	1.000000
<b>dropna</b>	0.702184	0.692927	0.720028
<b>2003 to 2018</b>	0.591567	0.672811	0.868883

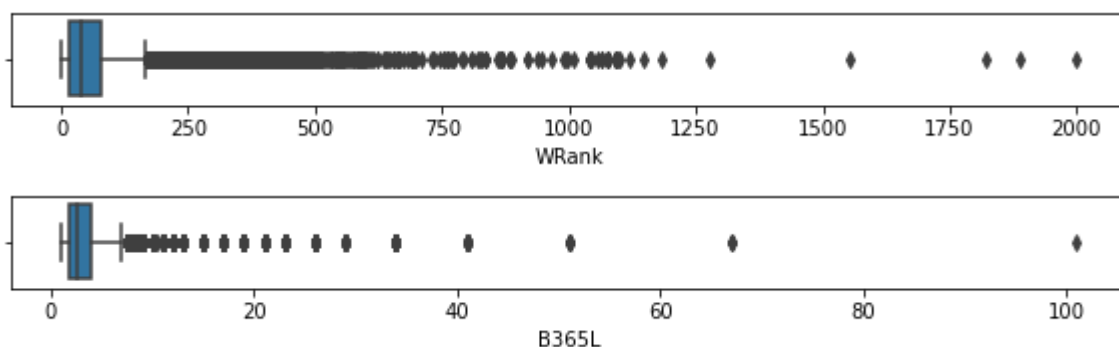
Une « bonne prédiction » doit être entendue au sens où le bookmaker prédit la victoire d'un joueur (favori) en lui attribuant une cote plus faible que son adversaire (challenger). En traduisant cela par rapport à notre dataset, lorsque la cote B365W est plus faible que B365L, le bookmaker effectue une bonne prédiction.

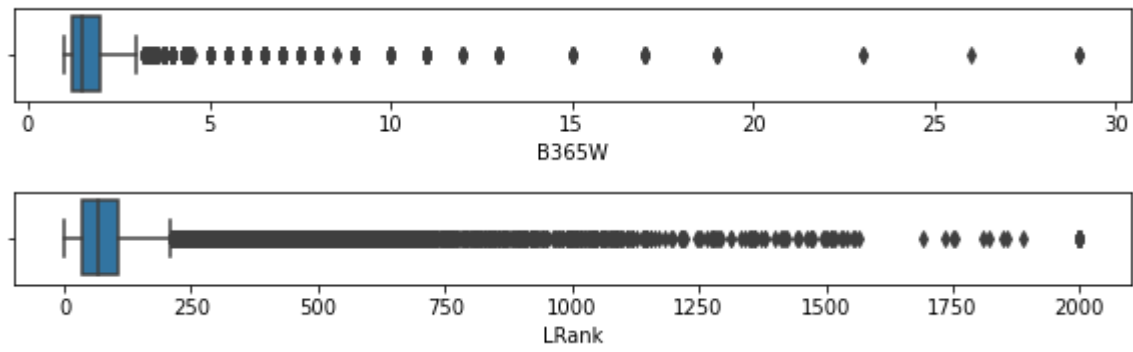
- La ligne “full dataset” représente le taux de bonnes prédictions sur l'ensemble du dataset sans suppression de lignes. Ce qui veut dire que les NaN sont comprises, notamment avec les années 2000 à 2002 où beaucoup de cotes sont manquantes. On peut voir que cela fausse en partie le score des bookmakers.
- La ligne “dropna” représente le taux de bonnes prédictions sur le dataset diminué des lignes contenant les NaN.
- La ligne 2003 à 2018 représente le taux de bonnes prédictions pour le dataset diminué des années contenant le plus de NaN, c'est à dire pour les années 2000 à 2002.

Sans les NaN, le taux de bonnes prédictions est d'environ 70% pour les deux bookmakers. Enfin la suppression des années 2000 à 2002 montre une différence conséquente entre le taux de bonnes prédictions de PS et B365. On peut l'expliquer principalement par les cotes manquantes PSW et PSL pour les années 2003 et 2009.

## Distribution des variables

Nous avons représenté plusieurs distributions de variables en utilisant les boîtes à moustaches. Nous nous sommes attardés sur la distribution des cotes notamment car elles présentent de nombreuses valeurs dites outliers.





On peut constater qu'il existe des valeurs extrêmes notamment au niveau des cotes. Nous en supprimons une partie pour ne pas fausser la modélisation.

## Relations entre les variables

### Matrice de corrélations



Nous constatons que les valeurs entre PSW et B365W sont très proches, c'est également le cas entre PSL et B365L.

Hypothèse 4 : Cela nous confirme que les valeurs d'un seul bookmaker pourraient suffire. Nous choisirons les cotes de Bet 365 car ces dernières possèdent moins de NaN que les cotes de Pinnacle Sports sur l'ensemble du dataset. Nous constatons que les corrélations entre la colonne ATP et les autres variables est très faible. Ce qui implique que cette colonne n'a pas de liens avec les autres. Par ailleurs, elle n'apporte pas d'information pertinente sur les matches.



Hypothèse 5 : Nous pouvons supprimer la colonne ATP

La valeur que renvoie la colonne ATP ne donne pas d'information interprétable.

Après exploration du jeu de données nous observons qu'elle ne renvoie ni à un tournoi spécifique, ni à un événement déterminant ou influençant l'issue du match.

En prenant les valeurs les plus fortes de chaque variables, nous constatons que la variable Best of a une forte corrélation (0.82) avec la variable Wsets, la variable WRank a une corrélation moyenne (-0.5) avec la variable elo\_winner, la variable LRank a une corrélation faible (-0;39) avec la variable elo\_loser

### Test ANOVA

Nous avons fait une étude de corrélation des variables numériques qui concernent les joueurs. A partir de p-value, il apparaît qu'elles sont toutes liées.

	df	sum_sq	mean_sq	F	PR(>F)
<b>B365L</b>	1.0	1.481349e+07	1.481349e+07	4138.387991	0.000000e+00
<b>B365W</b>	1.0	8.967566e+06	8.967566e+06	2505.234814	0.000000e+00
<b>LRank</b>	1.0	8.104476e+06	8.104476e+06	2264.116666	0.000000e+00
<b>Wsets</b>	1.0	7.428650e+04	7.428650e+04	20.753136	5.240471e-06
<b>Lsets</b>	1.0	1.885381e+03	1.885381e+03	0.526712	4.679970e-01
<b>elo_winner</b>	1.0	3.263193e+07	3.263193e+07	9116.258474	0.000000e+00
<b>elo_loser</b>	1.0	2.643743e+05	2.643743e+05	73.857238	8.705564e-18
<b>proba_elo</b>	1.0	3.191018e+06	3.191018e+06	891.462523	1.099505e-193
<b>Residual</b>	38833.0	1.390039e+08	3.579531e+03	NaN	NaN

Nous allons maintenant nous intéresser aux relations entre variables catégorielles en procédant à un test du chi2

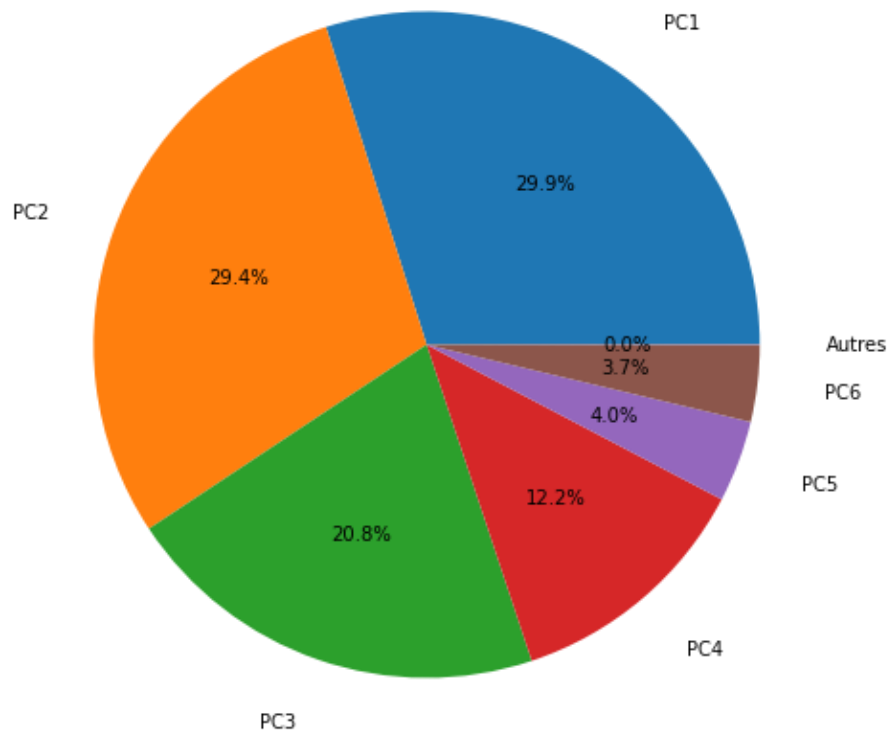
### Test du Chi2

	Location	Tournament	Date	Series	Court	Surface	Round	Winner	Loser	Comment
<b>Location</b>	V/V	L	L	L	L	L	L	L	L	L
<b>Tournament</b>	L	V/V	L	L	L	L	L	L	L	L
<b>Date</b>	L	L	V/V	L	L	L	L	L	L	L
<b>Series</b>	L	L	L	V/V	L	L	L	L	L	L
<b>Court</b>	L	L	L	L	V/V	L	L	L	L	NL
<b>Surface</b>	L	L	L	L	L	V/V	L	L	L	L
<b>Round</b>	L	L	L	L	L	L	V/V	L	L	L
<b>Winner</b>	L	L	L	L	L	L	L	V/V	L	L
<b>Loser</b>	L	L	L	L	L	L	L	L	V/V	L
<b>Comment</b>	L	L	L	L	NL	L	L	L	L	V/V

V/V : valeur sur valeur L : variables liées car  $p\_value < 0.05$  NL: variables non liées car  $p\_value > 0.05$

Au vu des tableaux ci-dessus nous décidons de conserver l'ensemble des colonnes à l'exception des colonnes Comment, et Localisation.

### Identification des composantes principales



L'étude montre qu'avec trois composantes on atteint 80,1% de représentativité, ce qui souligne l'importance de chaque colonne dans le dataset. Nous décidons de conserver la majorité des colonnes présentes.

L'analyse en composantes principales est un outil de synthèse de l'information, qui s'applique particulièrement lorsqu'on est en présence d'une somme importante de données quantitatives à traiter et interpréter. Dans notre cas, nous n'avons pas eu de problème de réduction du nombre de colonnes.

## 4.2 Preprocessing

### Traitement des NaN

Le dataset contient 12517 lignes avec des NaN sur 44708 soit 28% des lignes. Ne pouvant pas supprimer toutes ces lignes car trop nombreuses, nous avons étudié la possibilité de trouver une relation entre les colonnes elos et les cotes pour pouvoir générer les cotes manquantes.

La présence des NaN se fait principalement sur les colonnes des côtes (PSW,PSL,B365W,B365L) notamment sur les premières années du dataset. Une solution serait de reconstituer les côtes en fonction des variables "elo" qui indiquent le "rang" des joueurs. Le système des cotes fait que plus un joueur est susceptible de remporter un match, plus sa cote est faible. Ainsi on peut supposer que plus le elo entre les joueurs est important plus l'écart des cotes le serait aussi. On voudrait ici calculer la cote des joueurs en se basant sur l'écart des elo.

Cette solution n'a pas été adoptée car on observe une disparité trop importante au niveau de l'écart des elo par rapport à l'écart entre cote. Il semble difficile de reconstituer les cotes uniquement par rapport aux elos et d'obtenir des résultats cohérents avec l'ensemble du dataset. Cette solution n'a donc pas été retenue.

Une autre solution serait de compléter le jeu de données avec un autre dataset mais nos recherches ont montré qu'il est difficile d'obtenir les cotes manquantes, ces dernières concernent principalement les premières années du dataset.

### **Suppression des colonnes et des lignes**

Nous choisissons donc de supprimer certaines lignes du dataset : en gardant un seul bookmaker B365, nous pouvons supprimer les années 2000, 2001 et 2002 qui entraînent une suppression de 5862 lignes soit 13% du dataset.

La colonne ATP ne renvoie pas d'information pertinente par rapport au dataset, elle est donc supprimée. La suppression des colonnes PSW et PSL est due à notre choix de travailler uniquement avec les cotes de Bet365.

### **Encodage des variables catégorielles**

Les variables 'Series', 'Court', 'Tournoiement', 'Surface' et 'Round' sont encodées à l'aide de la fonction LabelEncoder de sklearn. L'encodage de ces variables sera utile pour la suite du projet, notamment dans l'étape de modélisation qui nécessite d'encoder les variables de type objet.

## 5 Modélisation

### 5.1 Transformation du dataset

#### Elaboration de la target

Dans un premier temps, nous attribuons un id à chaque joueur. Le but est de garder une information concernant les joueurs. Comme le dataset contient une colonne Winner et une colonne Loser, il faut regrouper les joueurs, leur affecter un id et remplacer ensuite le nom par l'id.

L'information de la victoire ou de la défaite est indirectement présente dans le dataset par l'intermédiaire des colonnes Winner et Loser. Pour avoir une target qui indique les victoires ou les défaites, nous devons retravailler le dataset afin de pouvoir déterminer le résultat du match par rapport au "premier joueur" ou au "second joueur".

Nous transformons donc le dataset pour avoir une colonne "premier joueur", une colonne "second joueur" et une colonne target. De fait, le dataset est doublé pour avoir à la fois les victoires et les défaites. (Une ligne correspond à la victoire de l'un des joueurs, l'autre ligne correspond à la défaite de l'autre joueur)

La target prend la valeur 1 pour une victoire et 0 pour une défaite. Quand le premier joueur gagne target = 1. A l'inverse, quand le premier joueur perd target = 0, ce qui revient à dire que si le deuxième joueur gagne, target = 0

### 5.2 Projet et Modèles

Dans cette partie, nous testons plusieurs modèles de classification pour ensuite comparer les résultats obtenus. Pour chaque modèle, nous procédons à l'optimisation de leurs hyperparamètres. Nous choisissons de tester les modèles suivants :

- RandomForest
- KNN
- AdaBoost
- Bagging

Pour chacun, nous avons effectué un gridsearch ainsi qu'une cross validation puis affiché la matrice de confusion, le score obtenu et la learning curve ainsi que la courbe ROC.

#### RandomForest avec critère Gini

##### Matrice de confusion

Classe prédite	0.0	1.0
Classe réelle		
0.0	5457	2314
1.0	2291	5475

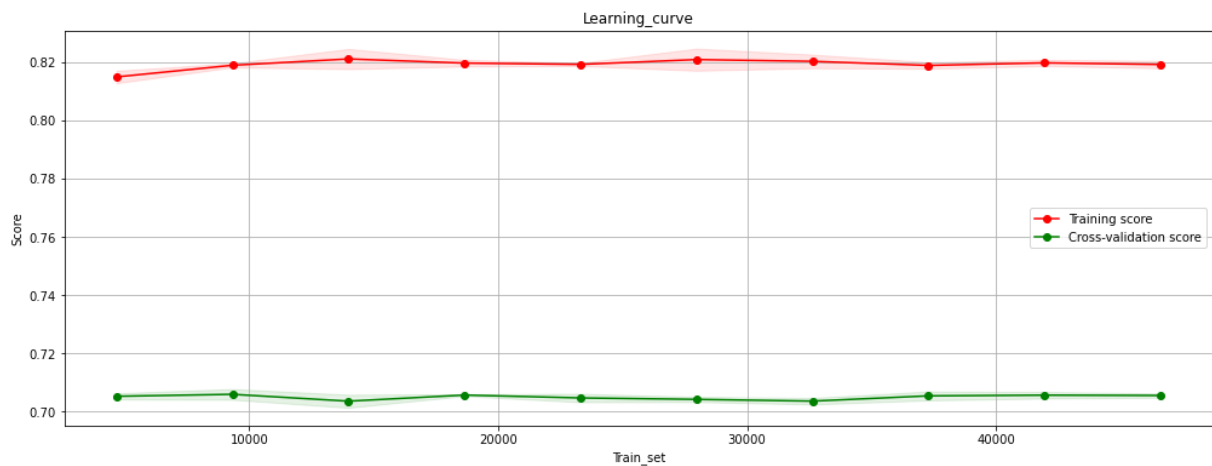
Nous avons à peu près un tiers de mauvaises prédictions aussi bien pour les matchs gagnés que les matchs perdus. Les bonnes prédictions sont équilibrées entre les matchs gagnés et les matchs perdus.

## Classification report

	precision	recall	f1-score	support
0.0	0.70	0.70	0.70	7771
1.0	0.70	0.70	0.70	7766
accuracy			0.70	15537
macro avg	0.70	0.70	0.70	15537
weighted avg	0.70	0.70	0.70	15537

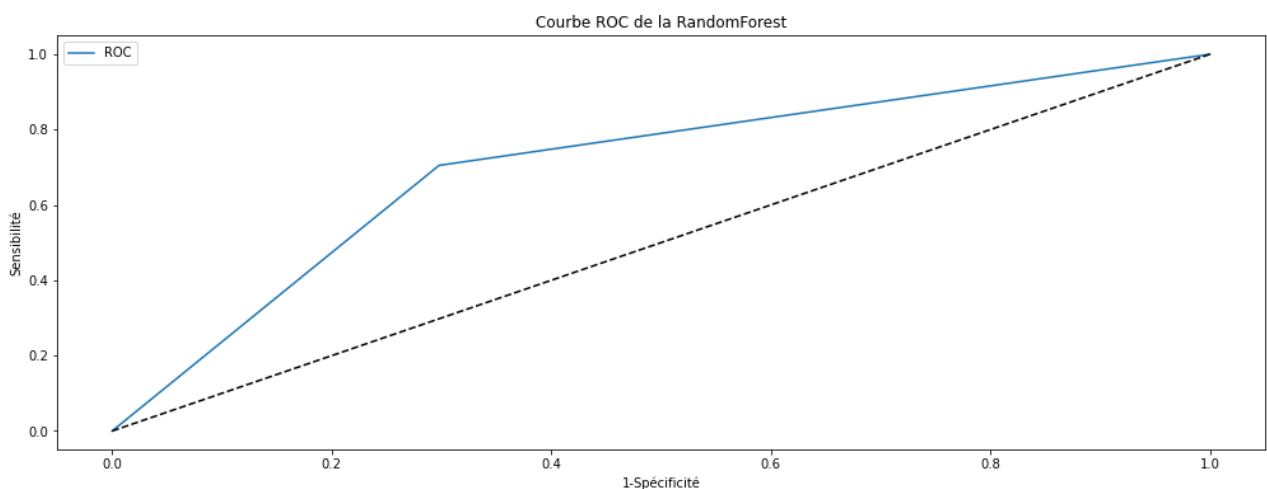
On observe une précision, un recall et un f1-score convenables en correspondance avec la matrice de confusion.

## Learning-curve



A partir d'un train\_set d'environ 15000 valeurs, les résultats restent identiques. Le score du test\_set est moins bon mais reste acceptable. On observe ici un phénomène d'overfitting, le modèle est plus performant sur l'échantillon d'entraînement de près de 10%.

## Courbe ROC



La courbe ROC confirme la qualité du modèle. Elle montre que le modèle est meilleur qu'un tri aléatoire ( $AUC > 0.5$ )

### Explication sur la forme de la courbe ROC

La courbe ROC représente les TFP / TVP avec :

- sensibilité :  $TVP = VP / (VP + FN) \Rightarrow$  qualité à prédire les vrais positifs
- 1-spécificité :  $TFP = FP / (FP + VN) \Rightarrow$  qualité à prédire les faux positifs

TFP : taux faux positifs

TVP : taux vrais positifs

VP : vrais positifs

FN : faux négatifs

Dans un modèle idéal, TVP doit être le plus grand possible et TFP le plus petit possible. Dans notre cas, on choisit le point d'équilibre qui est au sommet du coude de la courbe. On note que la forme de notre courbe ROC est constituée d'un unique point ce qui peut paraître douteux, une explication résiderait dans le fait que :

Le nombre de points dépend du nombre de modalités en entrée :

□ Pour `y_test` le nombre de point est `2 array([0., 1.])`.

Étant donné que le vecteur d'entrée n'a que 2 valeurs uniques, la fonction en sortie ne peut avoir que 2 valeurs, c'est à dire une courbe avec 3 points

□ Pour `fper` le nombre de points est `3 array([0., 0.29024421, 1. ])`

## KNN

### Matrice de confusion

Classe prédite \ Classe réelle	0.0	1.0
0.0	5266	2505
1.0	2616	5150

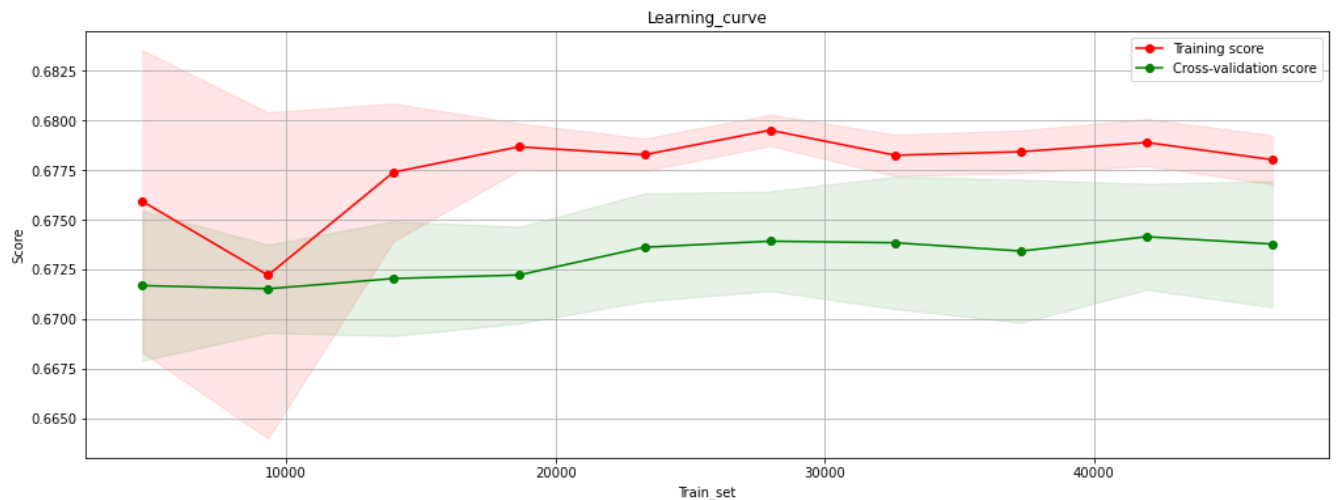
Nous avons un peu plus d'un tiers de mauvaises prédictions aussi bien pour les matchs gagnés que les matchs perdus. La matrice est équilibrée

### Classification report

	precision	recall	f1-score	support
0.0	0.67	0.68	0.67	7771
1.0	0.67	0.66	0.67	7766
accuracy				0.67
macro avg				0.67
weighted avg				0.67

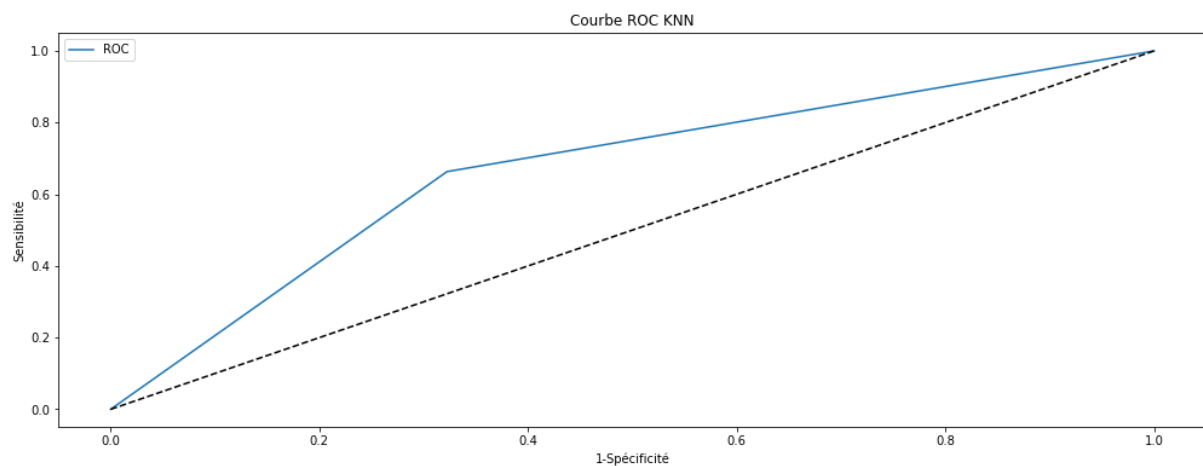
On observe une précision, un recall et un f1-score convenables en correspondance avec la matrice de confusion.

## Learning-curve



A partir d'un train\_set d'environ 25000 valeurs, les résultats restent identiques. Le score du train\_set est bon sans être en overfitting, le score du test\_set est moins bon mais reste acceptable. L'écart entre les deux est proche.

## Courbe ROC



La courbe ROC confirme la qualité du modèle. Elle montre que le modèle est meilleur qu'un tri aléatoire ( $AUC > 0.5$ )

## AdaBoost

### Matrice de confusion

Classe prédite \ Classe réelle	0.0	1.0
0.0	5426	2345
1.0	2180	5586

Nous avons près d'un tiers de mauvaises prédictions aussi bien pour les matchs gagnés que les matchs perdus. La matrice est équilibrée.

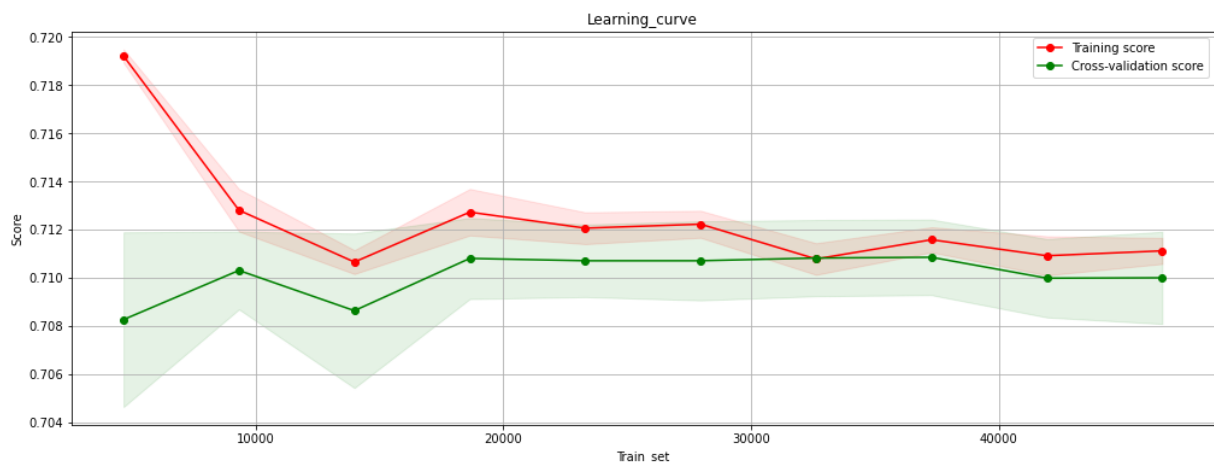


## Classification report

	precision	recall	f1-score	support
0.0	0.71	0.70	0.71	7771
1.0	0.70	0.72	0.71	7766
accuracy			0.71	15537
macro avg	0.71	0.71	0.71	15537
weighted avg	0.71	0.71	0.71	15537

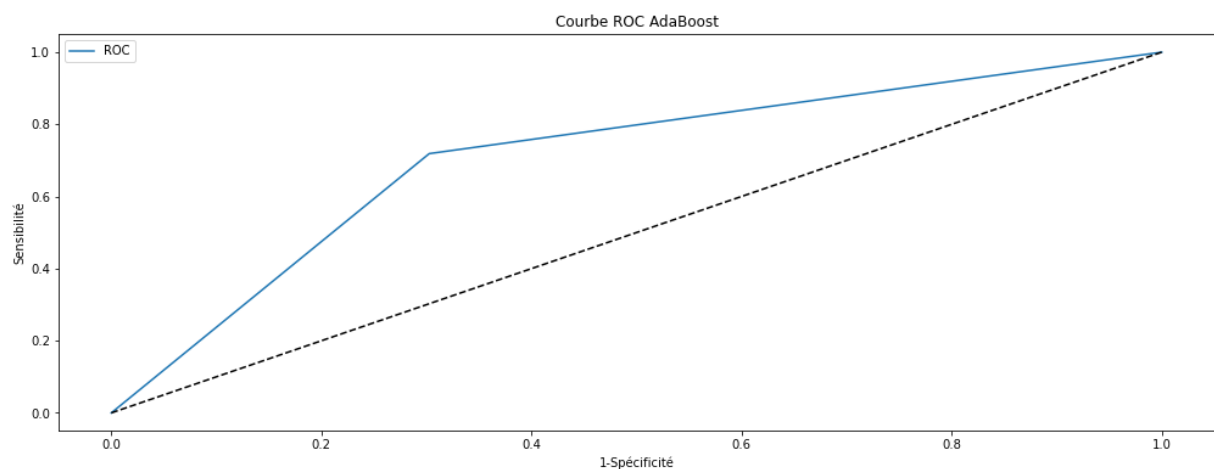
On observe une précision, un recall et un f1-score convenables en correspondance avec la matrice de confusion.

## Learning-curve



A partir d'un train\_set d'environ 18000 valeurs, les résultats restent identiques. Les scores sont bons sans être en overfitting.

## Courbe ROC



La courbe ROC confirme la qualité du modèle. Elle montre que le modèle est meilleur qu'un tri aléatoire ( $AUC > 0.5$ )

## Bagging

### Matrice de confusion

Classe prédite	0.0	1.0
Classe réelle		
0.0	5473	2298
1.0	2235	5531

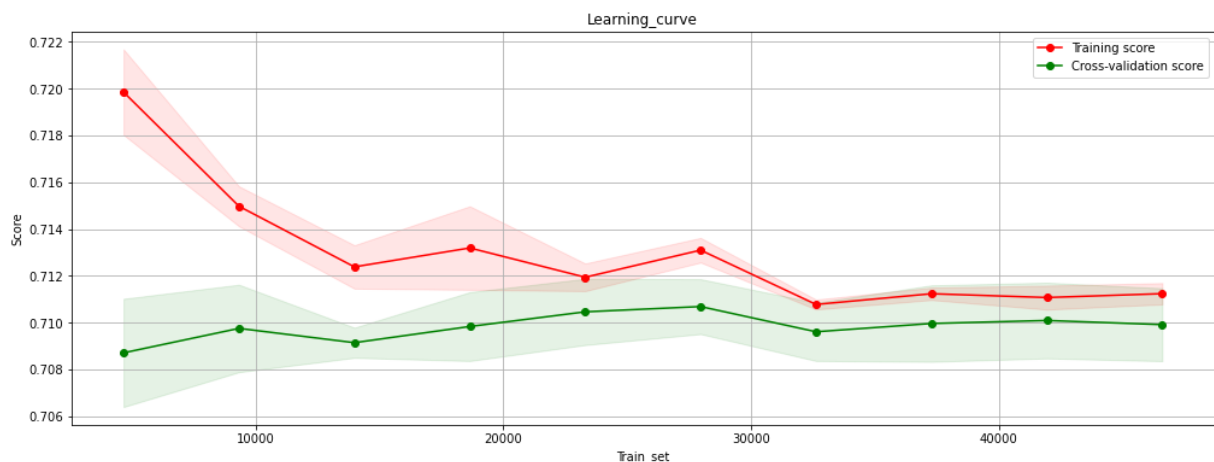
Nous avons à peu près un tiers de mauvaises prédictions aussi bien pour les matchs gagnés que les matchs perdus. La matrice de confusion est équilibrée.

### Classification report

	precision	recall	f1-score	support
0.0	0.71	0.70	0.71	7771
1.0	0.71	0.71	0.71	7766
accuracy			0.71	15537
macro avg	0.71	0.71	0.71	15537
weighted avg	0.71	0.71	0.71	15537

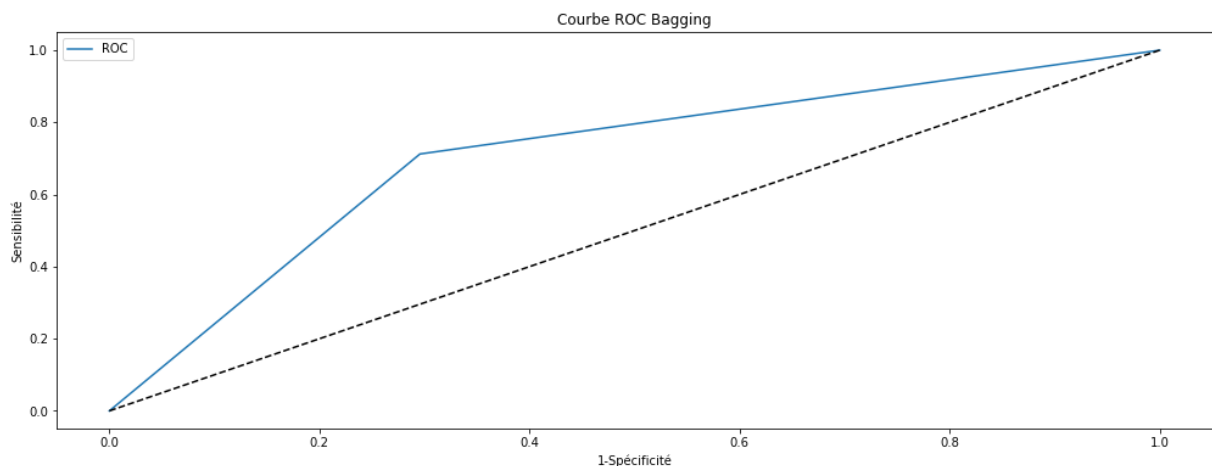
On observe une précision, un recall et un f1-score convenables en correspondance avec la matrice de confusion.

### Learning-curve



Les résultats sont fluctuants mais restent distincts entre le train\_set et le test\_set. Le score du train\_set est bon sans être en overfitting, le score du test\_set est moins bon mais reste acceptable. L'écart entre les deux est réduit.

## Courbe ROC



La courbe ROC confirme la qualité du modèle. Elle montre que le modèle est meilleur qu'un tri aléatoire (AUC>0.5)

## 5.3 Scénario alternatif – Utilisation des Moyennes roulantes

Dans ce scénario, nous supposons que les scores sont dépendant de la performance des joueurs qui peut varier dans le temps, aussi nous remplaçons le nombre de Sets par leurs moyennes roulantes sur les cinq dernières observations de manière à lisser les fluctuations liées aux variations du nombre de sets.

Pour réaliser cette opération, il faut trier le dataset par joueur, puis calculer la moyenne roulante par joueur pour le nombre de d'observations voulues.

### Interprétation des résultats

#### Matrice de confusion

Classe prédite	0.0	1.0
Classe réelle		
0.0	6573	1099
1.0	80	7785

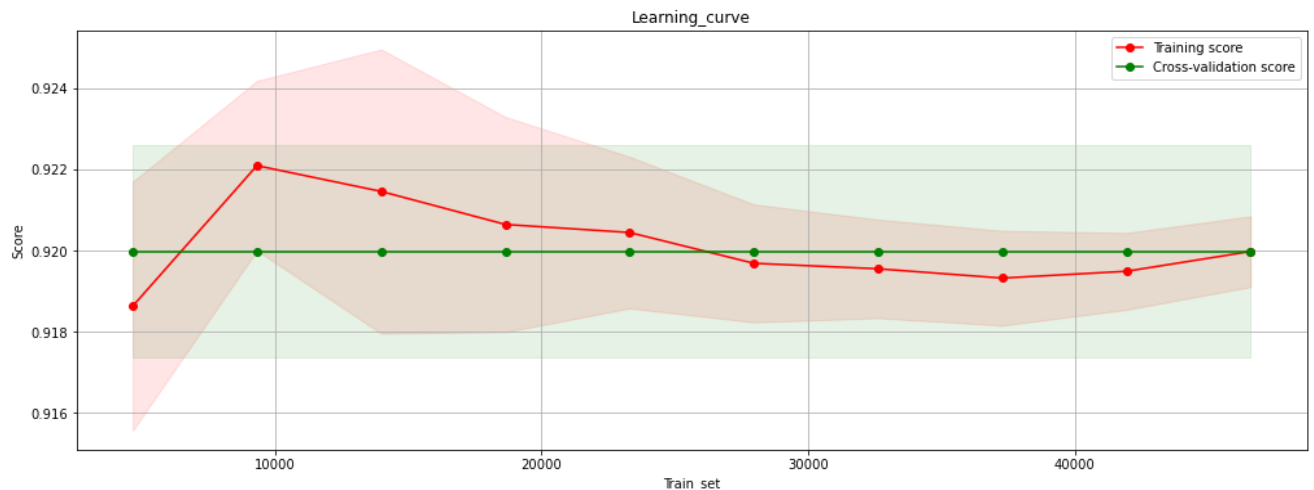
Nous obtenons une matrice très déséquilibrée. Pour les défaites, nous avons 14% de mauvaises prédictions et près de 1% de mauvaises prédictions pour les victoires.

#### Classification report

	precision	recall	f1-score	support
0.0	0.99	0.86	0.92	7672
1.0	0.88	0.99	0.93	7865
accuracy			0.92	15537
macro avg	0.93	0.92	0.92	15537
weighted avg	0.93	0.92	0.92	15537

On observe une précision, un recall et un f1-score très élevé.

## Learning-curve



Le score du train\_set est fluctuant autour de 0.9205, ce qui représente un score très élevé. Le score du test\_set est identique. Il est surprenant d'obtenir des scores aussi élevés et d'avoir la courbe du train\_set qui fluctue alors que celle du test\_set est parfaitement droite. Par ailleurs le croisement entre le training score et le cross-validation score nous semble improbable, cela signifierait que le modèle prédit mieux une donnée jamais rencontrée qu'une donnée déjà vue.

Au vu des résultats décrits ci-dessus, nous n'approfondirons pas ce scénario alternatif.

## 5.4 Choix de la métrique et récapitulatif de la modélisation

Nous avons vu que notre Dataset n'est pas déséquilibré. Sa taille réduite n'implique pas de se préoccuper du temps de traitement ni de la capacité de calcul. Nous nous intéressons uniquement à la précision des algorithmes.

Étant donné que nous utilisons des algorithmes de classification, nous avons choisi les outils suivants :

- la matrice de confusion
- la courbe ROC
- le rapport de classification

### La matrice de confusion, une métrique de classification multi classes :

Elle est particulièrement adaptée dans les problèmes de classification

		Classe réelle	
		0	1
Classe prédite	0	VP (vrais positifs)	FP (faux positifs)
	1	FN (faux négatifs)	VN (vrais négatifs)

Avec  $\text{accuracy} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}$

Nous avons choisi la matrice de confusion pour :

- voir si la classification est équilibrée (VP vs VN, FN vs FP)
- calculer le taux de précision
- voir le nombre de bonnes classifications.

### La courbe ROC, une métrique de classification binaire :

Elle Synthétise les performances d'un classificateur binaire selon les seuils possibles avec

- En abscisse :  $TFP = FP / (FP + VN) = 1 - \text{spécificité}$
- En ordonnée :  $TVP = VP / (VP + FN) = \text{sensibilité}$

TFP = taux de faux positifs

TVP = taux de vrais positifs

Nous avons choisi la courbe ROC pour évaluer l'efficacité de notre modèle et en particulier s'il est plus performant qu'un tri aléatoire

### Le rapport de classification, une métrique multi label :

Il montre les principales métriques de classification :

- La précision : capacité du classificateur à ne pas étiqueter comme positif un échantillon négatif

$$\text{Précision} = VP / (VP + FP)$$

- Le rappel : capacité du classificateur à trouver tous les échantillons positifs

$$\text{Rappel} = VP / (VP + FN)$$

- Le F1-score : moyenne harmonique pondérée de la précision et du rappel

$$\text{F1-score} = 2 \times \text{précision} \times \text{rappel} / (\text{précision} + \text{rappel})$$

### Tableau de synthèse des modèles étudiés :

	Classes	Précision	Rappel	F1-score	Accuracy	Overfitting
RandomForest	0	0,70	0,70	0,70	0,70	Oui
	1	0,70	0,70	0,70		
KNN	0	0,67	0,68	0,67	0,67	Non
	1	0,67	0,66	0,67		
AdaBoost	0	0,71	0,70	0,71	0,71	Non
	1	0,70	0,72	0,71		
Bagging	0	0,71	0,70	0,71	0,71	Non
	1	0,71	0,71	0,71		

Plusieurs remarques concernant la partie modélisation :

- Le RandomForest est en overfitting sur le train\_set. Nous ne choisissons pas ce modèle.
- Le KNN a les moins bonnes performances. Nous ne choisissons pas ce modèle.
- Le Bagging et le Boosting présentent des performances identiques qui sont les meilleures des quatre algorithmes testés.

Dans notre problème, nous privilégions le modèle qui a le meilleur score dans le Rappel, et particulièrement pour la classe 1, c'est-à-dire sa capacité à prédire les matchs gagnés.

En l'occurrence, Adaboost est meilleur. Nous le choisirons pour la partie simulation de paris.

## 6 Simulation Pari Sportif

Nous avons procédé à la suppression des matchs de la dernière année du dataset (2018), afin d'utiliser notre modèle sur des données nouvelles et observer les résultats. Ainsi le modèle est entraîné sur des rencontres entre 2003 et 2017, soit près de 33 000 rencontres.

Nous avons par la suite récupéré un échantillon randomisé de 100 matchs sur ceux ayant eu lieu en 2018 et avons procédé à une simulation de paris. Cela permettra au modèle de prédire l'issue des matchs sur un échantillon qui lui est inconnu.

Nous utiliserons la formule suivante pour calculer notre mise :

$$\text{mise\_minimum} + (\text{mise\_maximum} - \text{mise\_minimum}) \times (\text{probas} - \text{sûreté}) / (1 - \text{sûreté})$$

Sûreté = seuil minimum afin de parier sur une rencontre

mise\_minimum = 100

mise\_maximum = 1000

probas = probabilité de l'évènement d'avoir lieu, ici soit 0 soit 1

La formule de la mise a été tirée d'un module dédié aux paris sportifs de la formation DataScientest.

La fonction mise en place pour la simulation des paris :

- prédit l'issue de chaque rencontre
- affecte une probabilité pour la victoire ou la défaite du joueur 1. Si cette probabilité est supérieure à une certaine valeur dite sûreté (0.8 ici) alors le parieur va miser une somme plus ou moins élevée.
- cette somme est comprise entre 100 et 1000 euros. Plus la probabilité est élevée, plus la mise est importante. A l'inverse, si on atteint seulement la sûreté, alors la mise se limite à 100 euros.

### Remarques :

Le modèle utilisé est celui d'un RandomForest avec n\_estimators = 40 et critère Gini. Nous n'avons pas utilisé Adaboost car nous n'avons pas réussi à faire tourner ce modèle avec la fonction que nous avons construite pour la simulation.



## Résultats obtenus par la simulation avec RandomForest

```

index 6 parier 400€ sur victoire du Joueur 1 - cote à 1.22 - résultat : 88€
index 14 parier 250€ sur victoire du Joueur 1 - cote à 1.08 - résultat : 20€
index 18 parier 250€ sur victoire du Joueur 1 - cote à 1.25 - résultat perte : -250€
index 23 parier 400€ sur victoire du Joueur 1 - cote à 1.07 - résultat : 28€
index 25 parier 400€ sur victoire du Joueur 2 - cote à 1.14 - résultat : 56€
index 29 parier 550€ sur victoire du Joueur 1 - cote à 1.2 - résultat : 110€
index 36 parier 400€ sur victoire du Joueur 2 - cote à 1.12 - résultat : 48€
index 43 parier 550€ sur victoire du Joueur 2 - cote à 1.08 - résultat : 44€
index 56 parier 250€ sur victoire du Joueur 1 - cote à 1.14 - résultat : 35€
index 64 parier 700€ sur victoire du Joueur 1 - cote à 1.03 - résultat : 21€
index 91 parier 250€ sur victoire du Joueur 1 - cote à 1.16 - résultat : 40€
index 96 parier 400€ sur victoire du Joueur 2 - cote à 1.1 - résultat : 40€
La mise totale a été de 4800

```

	index	probas	mise	cote_joueur	gain	cagnotte	mise_totale
0	6	0.900	400	1.22	88	88	400
1	14	0.875	250	1.08	20	108	650
2	18	0.875	250	1.25	-250	-142	900
3	23	0.900	400	1.07	28	-114	1300
4	25	0.900	400	1.14	56	-58	1700
5	29	0.925	550	1.20	110	52	2250
6	36	0.900	400	1.12	48	100	2650
7	43	0.925	550	1.08	44	144	3200
8	56	0.875	250	1.14	35	179	3450
9	64	0.950	700	1.03	21	200	4150
10	91	0.875	250	1.16	40	240	4400
11	96	0.900	400	1.10	40	280	4800

RoI: 0.058

On peut remarquer qu'une seule stratégie a été testée dans cette partie. Avec cette stratégie basée sur la sûreté, fixée à 0.8, nous dégageons un bénéfice en ne pariant que sur un nombre limité (ici 12) parmi les 100 matchs. On remarque que pour cette simulation avec cet échantillon précis, nous obtenons un retour sur investissement positif (RoI) de 5.8%.

Pour obtenir un résultat plus pertinent nous avons réitéré 500 fois cette opération en prenant chaque fois un échantillon de 100 matchs issus de 2018.

Une itération correspond à la prise d'un nouvel échantillon randomisé, sur lequel on effectue la simulation de paris et on obtient à chaque itération un RoI, un résultat et une mise totale.

```

1 resultat_plusieurs_simulations()
2
3
4 #RoI = résultat / mise_totale
5 #Après une simulation sur 500 itérations, on obtient un RoI moyen de 7.5%
6
7
8 #Les tuples se décomposent de la manière suivante (RoI, résultat, mise_totale)

```

(0.094, 711, 7560)  
 (0.015, 121, 8296)  
 (0.111, 803, 7236)  
 (0.19, 1396, 7360)  
 (0.07, 658, 9447)  
 (0.058, 418, 7147)  
 (-0.009, -67, 7461)  
 (0.104, 629, 6021)  
 (0.041, 349, 8588)  
 (0.071, 581, 8236)  
 (0.105, 897, 8556)  
 (0.112, 923, 8238)  
 (0.038, 395, 10459)  
 (0.141, 887, 6273)  
 (0.059, 493, 8335)  
 (0.077, 598, 7809)  
 (0.215, 1682, 7823)  
 (0.046, 406, 8785)  
 (0.153, 1194, 7800)

Nous obtenons ici 500 tuples sous la forme (RoI, résultat, mise\_totale). D'après notre expérience, nous obtenons un RoI moyen d'environ 7.5%. Néanmoins on peut remarquer plusieurs choses :

- Certaines itérations présentent un RoI négatif, ce qui signifie que sur certaines simulations nous pouvons enregistrer des pertes au total.
- La grande majorité des lignes dégagent un RoI positif mais aussi une mise totale importante.
- Le gain moyen obtenu est de 630 euros pour une mise totale moyenne de 8450 euros.
- La performance à chaque itération peut varier considérablement même si en moyenne le parieur réalise des gains.

### Tableau récapitulatif

	RoI	Résultat	Mise_totale
mean	7,50%	629	8431
max	23,00%	1983	12124
min	-8,00%	-635	4835
ecart-type	5,40%	449	1179

## 7 Conclusion

### Objectifs

L'objectif de ce projet était d'essayer de « battre » les algorithmes des bookmakers.

Nous sommes parvenus à prédire l'issue des matchs et à effectuer une simulation de paris, avec pour résultat des gains en tant que parieur. Bien qu'ayant une problématique peu originale, ce projet fut néanmoins challengeant et constructif pour un premier travail dans la data science, notamment en tant que data analyste en formation.

Concernant l'environnement technique, l'initiation à python pour la data science, aux différents modèles ainsi qu'à la pratique de la dataviz ont été des éléments importants tout au long de notre travail. L'utilisation de Github ou encore de Google drive nous ont permis de fluidifier nos échanges au cours du projet. Nous nous sommes partagé les tâches du mieux que possible avec les contraintes techniques liées aux travaux réalisés uniquement en ligne et un temps assez limité pour chaque étape du projet. Cependant le suivi et les conseils prodigués par Mounir, notre chef de projet, nous ont été précieux et nous ont permis d'avancer au fil des sprints.

### Déroulement du projet et modèles choisis

Comme expliqué auparavant, nous avons pu explorer différentes pistes tout au long du projet, notamment des scénarios alternatifs que nous n'avons pas approfondis, faute de résultats non concluants, voire douteux. Ainsi la piste retenue a été la prédiction des rencontres dans l'optique d'effectuer une simulation de paris avec pour objectif un gain pour le parieur. Le modèle retenu après optimisation des hyperparamètres a été celui de l'Adaboost, qui par la suite a confirmé sa capacité à prédire de manière satisfaisante l'issue des matchs.

### Difficultés rencontrées

Plusieurs difficultés ont pu apparaître durant ce projet.

Notre connaissance limitée du milieu des paris sportifs a pu être une limite dans notre approche de la thématique. Plus précisément, le manque d'une vraie connaissance « métier » a pu être pénalisant dans la recherche de la problématique et dans la compréhension des données.

Des difficultés au niveau de l'analyse des données ont aussi pu découler d'un manque de connaissances sur le tennis et notamment du circuit ATP.

Concernant le dataset, la présence de nombreuses valeurs manquantes nous a mis en difficulté. Les tentatives pour compléter le jeu avec des données externes n'ont pas été concluantes tout comme les essais menés pour les compléter via la moyenne ou d'autres variables.

Le fait d'avoir travaillé sur plusieurs scénarios durant une grande partie du projet a montré que nous avons eu du mal à définir la cible et un objectif précis vis-à-vis du sujet.

## Bilan & Suite du projet

Nous estimons que plusieurs objectifs ont pu être atteints via ce projet :

- la compréhension, l'exploration et le nettoyage du jeu de données,
- la familiarisation avec la méthodologie de la data analyse
- une maîtrise des différents outils du data analyste, qui néanmoins reste perfectible.
- l'utilisation de modèles de machine learning adaptés aboutissant à des résultats cohérents et satisfaisants
- une mise en situation de nos travaux répondant à l'objectif de gain pour le parieur

## Pistes d'améliorations

Il est probable que notre code n'est pas optimisé, ni par la structure, ni par le temps de traitement. Nous aurions pu utiliser des pipelines pour 'automatiser' les traitements.

Nous avons découvert tardivement l'utilisation de la fonction VotingClassifier qui permet de créer un meilleur classifieur pour nos modèles en augmentant le score.

Nous avons aussi noté dans nos pistes d'améliorations un traitement plus satisfaisant des NaN. Une approche possible aurait été d'utiliser un modèle pour prédire ces valeurs manquantes. Par exemple, l'imputation par KNN(K-nearest-neighbors) aurait pu être utilisée dans notre situation.

Enfin, l'enrichissement de notre dataset avec des données extérieures aurait probablement amélioré les performances de notre modèle.

Sur la partie simulation de pari, il est dommage que nous n'ayons pas pu explorer d'autres stratégies.

Ce projet nous a permis de sortir du cadre formalisé du contexte d'apprentissage en nous obligeant à déterminer une méthode pour aboutir à l'objectif. En ce sens, elle représente une première expérience appréciable.