

Projet Big Data

Prédiction de la place finale d'un joueur sur le jeu vidéo PUBG



Année scolaire 2019/2020

Achraf XX
Clément XX
Quentin RAUFFLET

Professeurs : Mme XX
Mr XX

Table des matières

Contenu

1. Introduction.....	3
2. Présentation et étude préalable des données	4
A) Listing des variables.....	4
3. État des lieux du jeu de données	5
a) Des échelles de kills très disparates	5
b) Les modes d'équipe privilégiés	6
c) Bien se soigner	8
d) Gérer ses déplacements.....	9
4. Traitement et sélection des variables	10
A) Analyse des corrélations.....	10
B) Variables supplémentaires	10
1) Variables sur la partie.....	10
2) Variables sur l'équipe	10
C) Classification du jeu de données.....	11
5. Modélisation	14
1) Choix du modèle.....	14
2) Exécution du modèle.....	14
3) Application du modèle	15
6. Conclusion	16
7. Annexes.....	17
1) Annexe 1 : glossaire.....	17
2) Annexe 2 : mode de classement Elo.....	17
3) Annexe 3 : Matrice de corrélation.....	18
4) Annexe 4 : Méthode des k-means.....	19
5) Annexe 5 : Retour d'expérience	19

1. Introduction

PlayerUnknown's Battlegrounds (PUBG) est un jeu vidéo édité par Sony Interactive Entertainment, et disponible sur les différentes consoles de jeux vidéo de salon de huitième génération (PS4, Xbox One...).

Le concept du jeu est le suivant : 100 joueurs sont lâchés sur une île les mains vides, avec pour mission d'explorer, de ratisser la zone, et d'éliminer les autres joueurs dans le but d'être le dernier survivant, alors que la zone de jeu rétrécit constamment au fur et à mesure de la partie.

Depuis sa sortie, le jeu a acquis une popularité croissante, jusqu'à atteindre les 50 millions de ventes et devenir le 5^{ème} jeu le plus vendu de l'histoire.

L'équipe du jeu a rendu disponible au public l'ensemble des données officielles du jeu, dont a bénéficié Kaggle grâce à l'interface de programmation PUBG Developer.

Ces derniers nous donnent l'accès aux données de joueurs anonymisés sur un ensemble de 65000 parties, divisées en un jeu d'apprentissage et un jeu de test.

Le but du concours est de prédire le classement final d'un joueur en fonction des statistiques finales du joueur après la partie.

En premier lieu, nous dresserons un état des lieux de la base de données pour prendre en main le matériel dont nous disposons, à travers la description des variables et les statistiques descriptives.

Ensuite, nous procéderons, si besoin il y a, à une sélection des variables les plus pertinentes pour notre analyse, à partir de différentes méthodes.

Enfin, nous mettrons en place une modélisation permettant la prédiction du classement final, tout en minimisant autant que faire se peut le taux d'erreur.

2. Présentation et étude préalable des données

A) Listing des variables

Voici la liste des variables mises originellement à disposition des participants, ainsi que leur description. La description des mots en gras se trouve dans le glossaire (annexe 1).

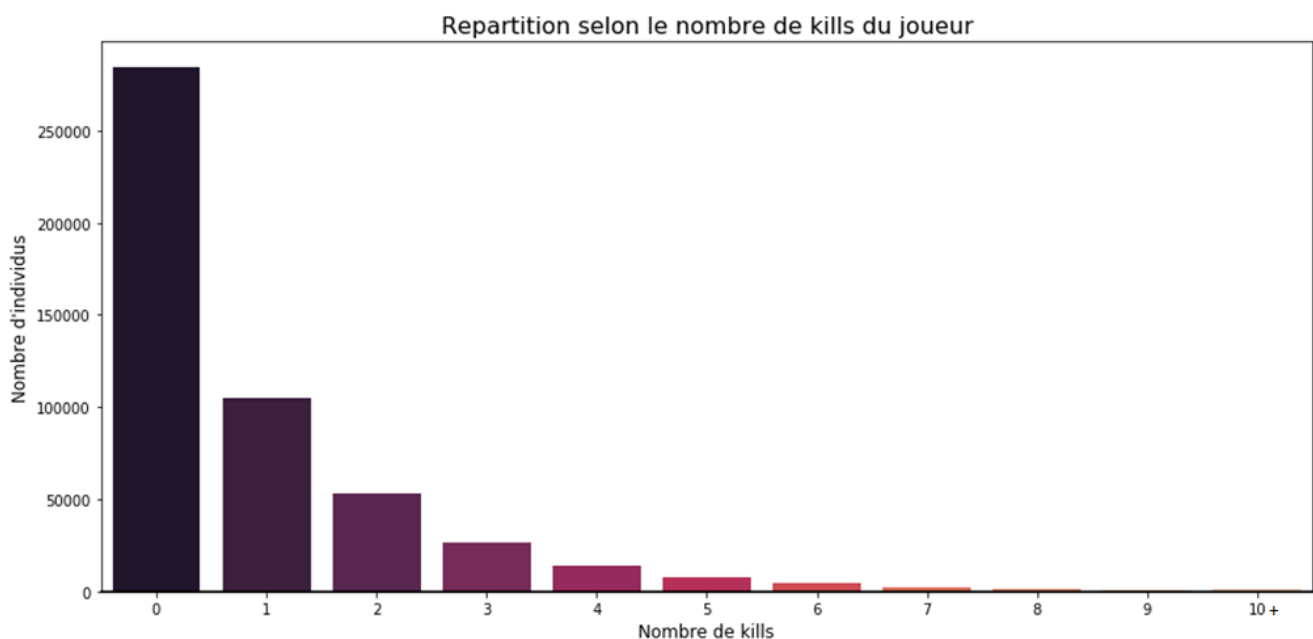
Variable	Description
DBNOs	Nombre d'ennemis assommés
assists	Nombre d'ennemis d'abord endommagés par le joueur, puis tués par des coéquipiers
boosts	Nombre de boosts utilisés
damageDealt	Total des dommages infligés (dommages infligés à soi-même exclus)
headshotKills	Nombre d'ennemis tués par headshot
heals	Nombre d'items de guérison utilisés
Id	Identifiant du joueur
killPlace	Classement dans la partie au nombre d'ennemis tués
killPoints	Classement Elo basé sur le nombre d'ennemis tués
killStreaks	Plus grand nombre d'ennemis tués sur une courte période donnée
kills	Nombre d'ennemis tués
longestKill	Plus grande distance entre le joueur et un joueur tué au moment de la mort
matchDuration	Durée de la partie en secondes
matchId	Identifiant de la partie (pas d'identifiant communs entre apprentissage et test)
matchType	Identification du mode de jeu : solo, duo, équipe...
rankPoints	Classement du joueur basé sur un modèle de calcul ELO (voir annexe 1)
revives	Nombre de fois qu'un joueur a réanimé un coéquipier
rideDistance	Distance totale parcourue à bord de véhicules, mesurée en mètres
roadKills	Nombre de victimes tuées à bord d'un véhicule
swimDistance	Distance totale parcourue à la nage, mesurée en mètres
teamKills	Nombre de fois où le joueur a tué un coéquipier
vehicleDestroys	Nombre de véhicules détruits
walkDistance	Distance totale parcourue à pied, mesurée en mètres
weaponsAcquired	Nombre d'armes récoltées
winPoints	Classement Elo basé sur le nombre de victoires du joueur
groupId	Identifiant d'un groupe dans une partie. Si un même groupe de joueurs joue des parties différentes, ils auront un identifiant différent
numGroups	Nombre de groupes dont les données sont disponibles pour chaque partie
maxPlace	Pire emplacement dont les données sont disponibles pour la partie
winPlacePerc	Classement du joueur en pourcentage : 1 correspond à la première place, 0 correspond à la dernière place

Notre jeu de données est très propre : sur **4 446 966** individus, nous n'avons qu'une valeur manquante pour la variable winPlacePerc. Étant donné le poids dans la base, il est sans conséquence et judicieux de supprimer la ligne directement.

Le jeu de données sur lequel sera effectué le travail sera donc composé de **4 446 965** individus, pour **29** variables.

3. État des lieux du jeu de données

a) Des échelles de kills très disparates

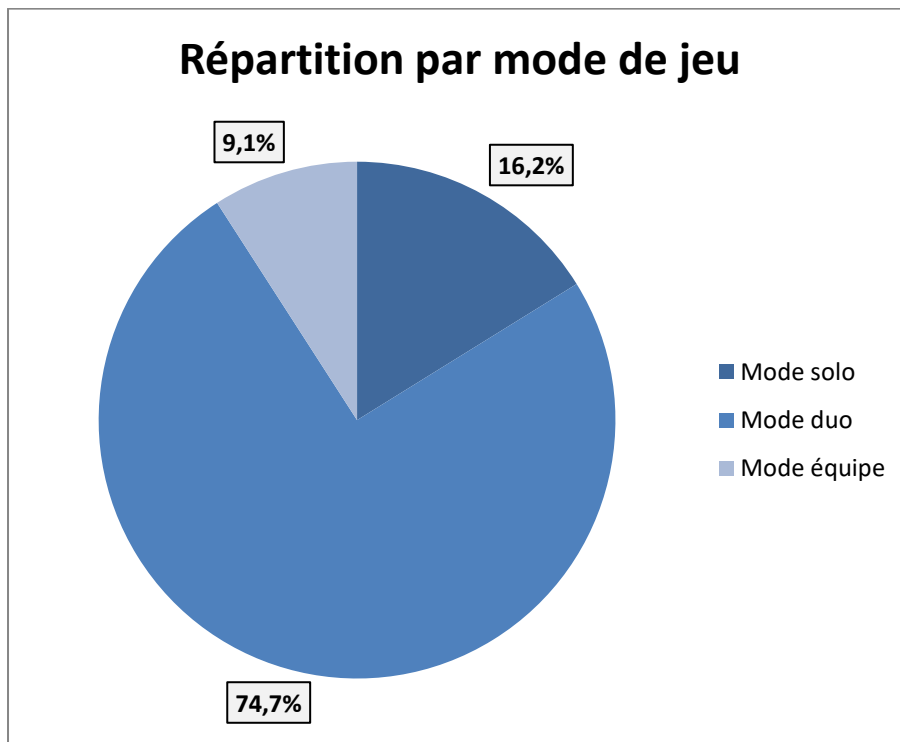


De manière globale, le nombre de kills est très dispersé. Au final, une très grande majorité des joueurs finit sa partie avec un score vierge... Très peu de dominants et beaucoup de dominés ?

On le voit assez nettement, ceux qui réalisent plus de 5 kills font partie d'une « élite » de joueurs assez restreinte.

Cette surreprésentation des « 0 kill » explique que la moyenne soit tirée vers le bas, puisque le nombre de kills moyen par joueur sur une partie est de 1.

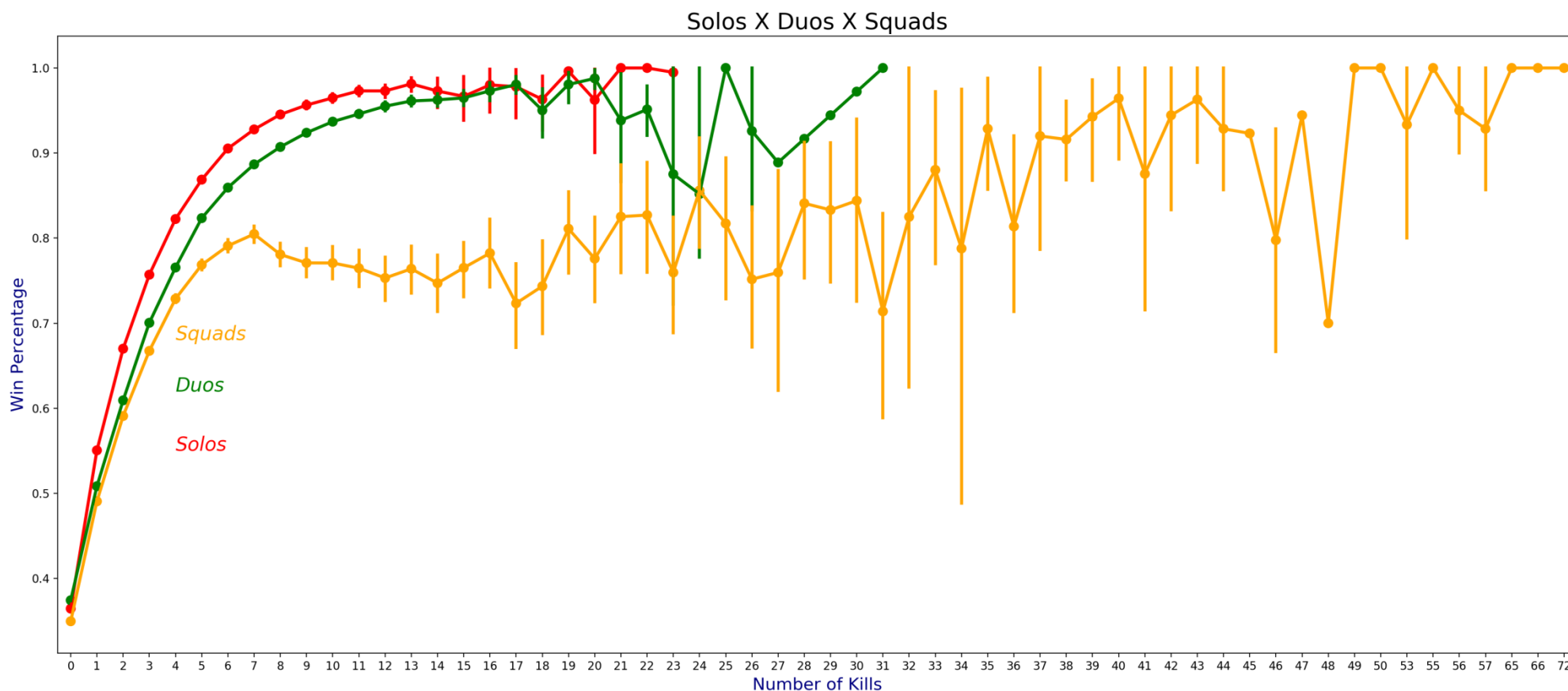
b) Les modes d'équipe privilégiés



Selon le mode de jeu, la façon d'aborder une partie et son déroulement changent. Nous en avons donc étudié la répartition. Près de 3 parties sur 4 sont jouées en mode duo, c'est le mode qui prédomine. Le mode solo est le deuxième le plus joué, alors que le mode équipe est celui qui concerne le moins de parties, avec moins de 10% de notre base.

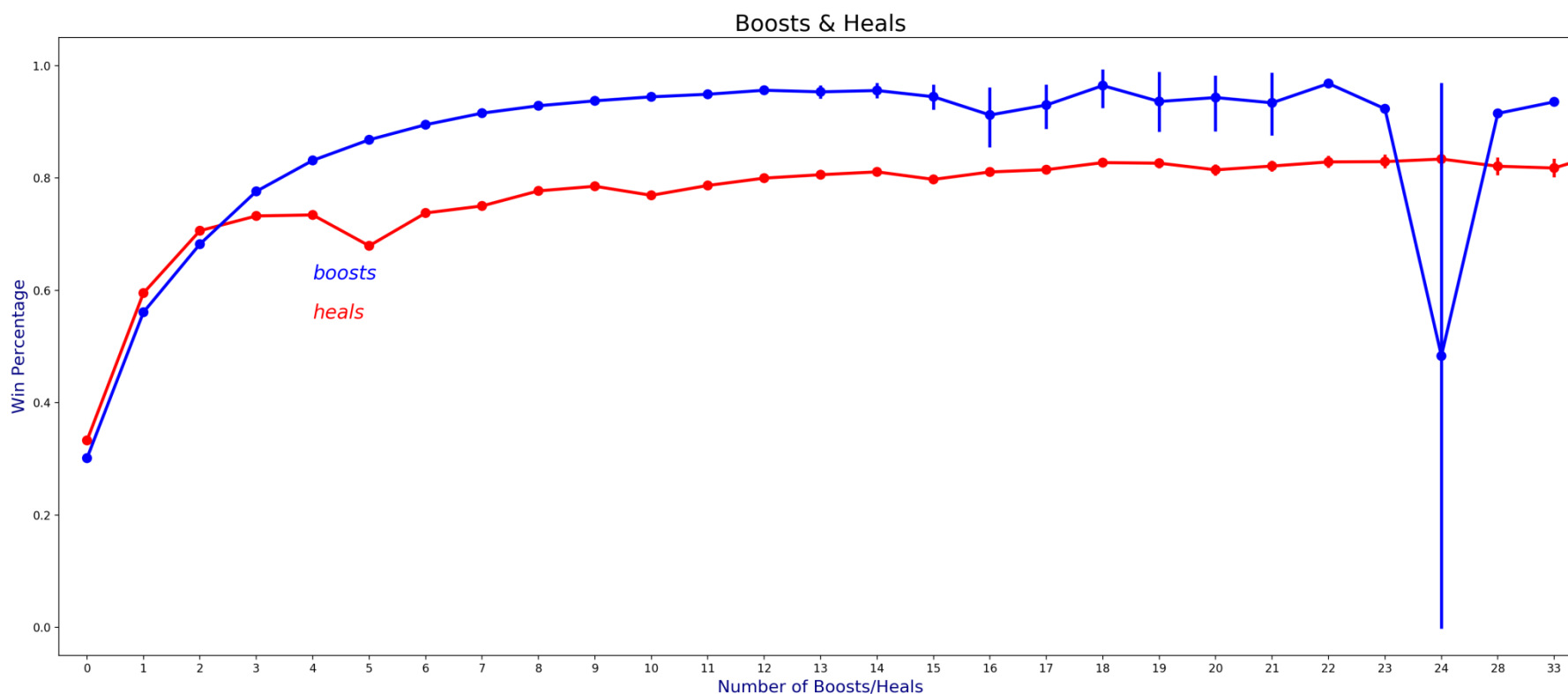
Pour mieux analyser cette différence entre les modes, nous nous sommes intéressés au pourcentage de victoire en fonction du nombre de kills, par mode de jeu. Si les trois modes se tiennent lorsqu'il y a un faible nombre de kills, l'écart se creuse au-dessus de 4 : l'impact est plus fort sur le pourcentage de victoire dans les modes de jeu solo et duo.

On peut également observer pour chaque mode une certaine proportionnalité entre le nombre d'adversaires tués et le pourcentage de victoires, mais ce seulement jusqu'à un certain point, notamment pour le mode squad. Cela est aussi dû au fait que les parties avec beaucoup de kills sont rares, et donc les données moins nombreuses sur ce niveau de kills (d'autant plus que le mode squad est le moins représenté).



c) Bien se soigner

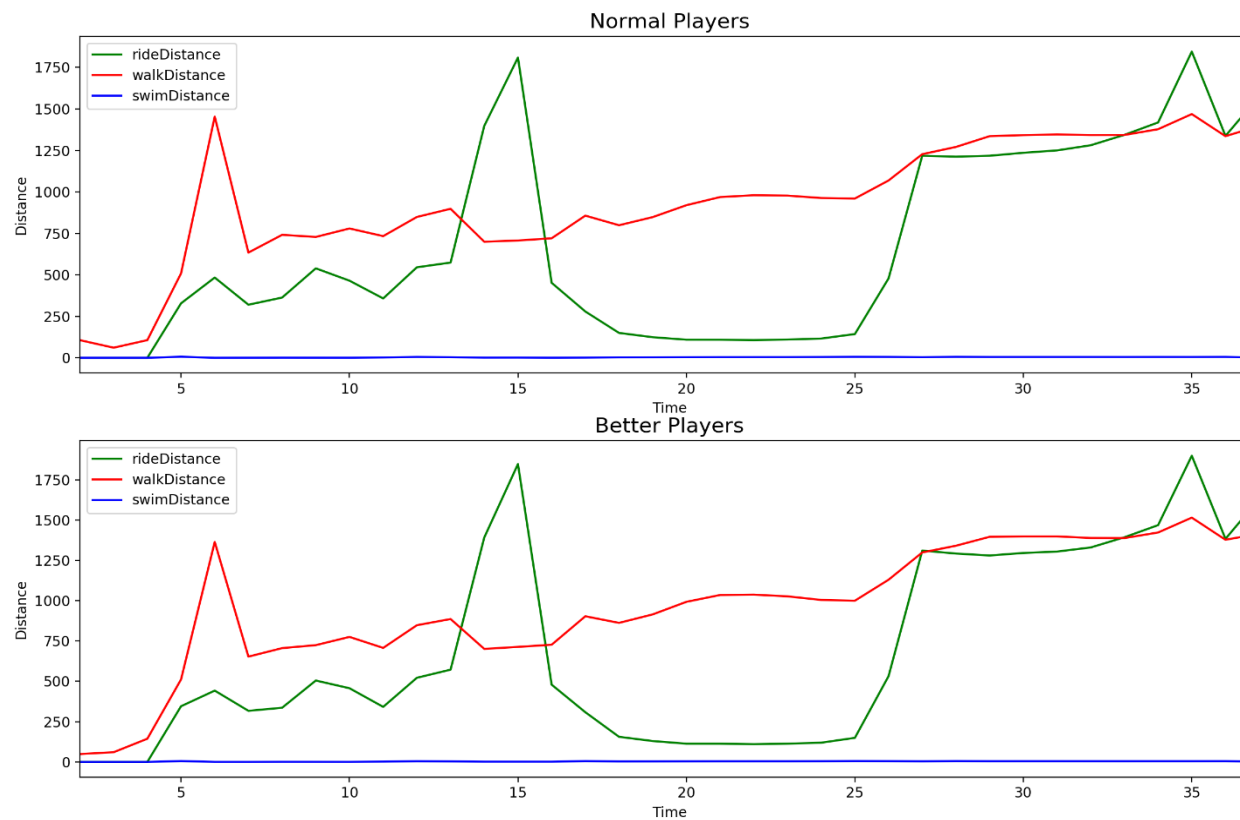
Le graphique ci-dessous témoigne de l'importance –logique– de conserver son énergie durant une partie. On constate que les boosts (qui servent à momentanément doper les compétences du joueur) sont plus valorisés que les heals, servant à récupérer de l'énergie après avoir subi des dommages. En d'autres termes, mieux vaut augmenter ses caractéristiques que conserver son énergie. Dans les deux cas, une certaine proportionnalité se démarque : plus on consomme de ces items, plus les chances d'atteindre la victoire se multiplient. Toutefois, le « breakpoint » (nombre d'items après lequel en consommer plus n'apporte pas d'avantage supplémentaire) apparaît plus tôt pour les boosts que les heals. Consommer plus de heals servira toujours à augmenter ces chances de gagner (même à la marge), alors qu'au-delà de 13 boosts, il n'est plus forcément bénéfique d'en rechercher.



d) Gérer ses déplacements

Les deux graphiques suivants présentent une comparaison de la distance parcourue en fonction du temps, pour les deux groupes de joueurs « Bons joueurs » et « Joueurs classiques ». Même si la différence n'est pas exacerbée, on constate que les meilleurs joueurs parcourent moins de distance. Savoir gérer ses déplacements est donc un atout pour augmenter les chances de victoire lors d'une partie, même si cela n'est pas une garantie.

Mean distance travelled at different match durations



4. Traitement et sélection des variables

A) Analyse des corrélations

Nous disposons d'un grand nombre de variables dans notre jeu de données, ce qui peut être négatif dans l'établissement du modèle si elles ne sont pas toutes significatives. Voilà pourquoi nous avons essayé de sélectionner les plus pertinentes par rapport à notre variable cible.

Dans un premier temps, nous avons utilisé la méthode répandue consistant à observer les variables explicatives extrêmement corrélées entre elles, et de ne retenir à chaque fois celle la plus corrélée à la variable cible.

C'est ainsi que nous avons choisi de supprimer les variables suivantes :

- maxPlace
- winPoints
- DBNOs

B) Variables supplémentaires

1) Variables sur la partie

Nous avons décidé de doubler les variables en rajoutant pour chacune d'elle la moyenne de la partie. En effet, chaque partie est différente, et il convient de situer le joueur à l'intérieur de la partie plutôt que sur l'entièreté du jeu. Pour chaque variable, nous avons donc introduit la **moyenne** et le **maximum** sur la partie.

Par exemple, si un joueur en tue 6 autres, ça n'a pas la même signification si le maximum et la moyenne de la partie sont bien plus élevés ou si justement le nombre de personnes maximum tuées est de 8.

2) Variables sur l'équipe

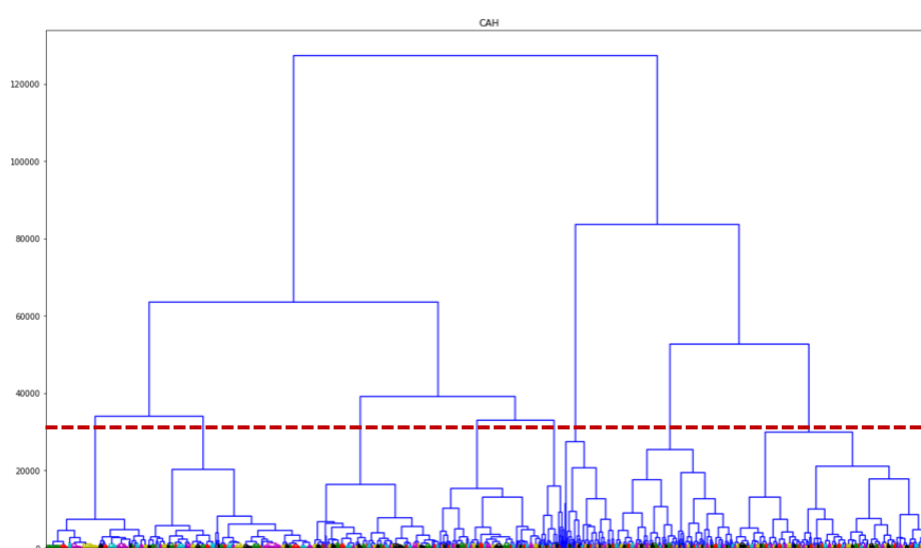
Un peu de la même manière que chaque partie est différente, chaque équipe impacte les joueurs : un joueur très mauvais en termes de caractéristiques peut se retrouver dans le haut du classement car bénéficiant du soutien de ses coéquipiers. Il est donc important de pouvoir situer le joueur dans son équipe.

C'est pourquoi, en ce qui concerne les parties jouées en équipe, nous avons rajoutée comme variables la moyenne et le maximum de l'équipe.

C) Classification du jeu de données

Pour tenter d'observer des caractéristiques différenciantes entre tous les individus, nous avons procédé à une classification. Le but étant de créer des groupes les plus différents possibles les uns des autres, dont les joueurs sont proches entre eux à l'intérieur de ces groupes.

Quand on a beaucoup d'observations et que l'on ne sait pas combien de classes créer, une méthode possible est de créer énormément de classes (2 000 dans notre cas), et de rentrer la moyenne des variables par classe dans un algorithme de classification ascendante hiérarchique (CAH) qui nous donne à travers un graphique appelé dendrogramme (voir ci-dessous) le nombre de classes optimal à mettre en place : **7 classes**, dans notre exemple.



Ensuite, nous avons utilisé la méthode des k-means (annexe 4), pour déterminer le contenu de ces classes.

Celles-ci ont chacune leurs particularités, que nous retrouvons dans la description suivante.

++ très discriminant

++ peu discriminant

Classe 1 : *rapides et mauvais*, 1 105 237 individus (25%)

Caractéristiques de la classe	++
Mauvais classement (0,29 en moyenne)	++
Très peu d'assistés (0,11 en moyenne)	++
Peu de boost	++
Partie courte	++

Classe 2 : rapides et mauvais, mais dans leur coin, 1 610 492 individus (36%)

Caractéristiques de la classe	++
Mauvais classement	++
Très peu d'assists (0,11 en moyenne)	++
Peu de boost	++
Partie courte	++
Privilège les parties solos	++

Classe 3 : moyens dans la durée, 333 270 individus (7%)

Caractéristiques de la classe	++
Classement moyen	++
Beaucoup de heals	++
Moyennement bon en nombre de kills	++
Parties longues	++
Distance en voiture très importante	++

Classe 4 : les sprinteurs, 639 429 individus (14%)

Caractéristiques de la classe	++
Classement bon	++
Parties courtes	++
Distance à pied importante	++

Classe 5 : les sprinteurs de relai, 478 952 individus (11%)

Caractéristiques de la classe	++
Classement bon	++
Parties courtes	++
Distance à pied importante	++
Proéminence des parties squad	++

Classe 6 : les as du volant, 141 637 individus (3%)

Caractéristiques de la classe	++
Classement bon	++
Beaucoup de heals	++
Performance faible en kills	++
Parties longues	++
Distance en voiture très importante	++

Classe 7 : les tops joueurs, 137 947 individus (3%)

Caractéristiques de la classe	++
Classement très bon	++
Beaucoup d'assists	++
Beaucoup de boosts récupérés	++
Beaucoup de heal	++
Parties longues	++
Distance en voiture moyenne	++
Distance à la nage importante	++
Distance à pied importante	++

Cette classification a pour principal objectif de mettre en exergue des variables discriminantes, de nous donner des idées de variables à créer, et d'avoir un regard plus pragmatique sur celles qui sont les plus importantes.

Les variables fortement clivantes sont les suivantes : la longueur de la partie, la distance parcourue, qu'elle soit à pied, en véhicule, ou à la nage, le nombre d'assists, de kills, de boosts et de heals.

5. Modélisation

La modélisation a été, avec la prise en main des variables, la phase la plus fastidieuse du projet. En effet, il a été difficile de trouver une solution pour générer un modèle sur plus de 4 millions d'individus, et 150 variables explicatives.

1) Choix du modèle

La première étape a été de tester différents types de méthodes sur une portion réduite de notre base, pour observer lesquelles pourraient être les plus efficaces. Parmi celles testées : régression, réseau de neurones, random forest et gradient boosting. C'est cette dernière que nous avons retenue car les résultats trouvés étaient sans équivoque toujours meilleurs.

Le boosting consiste à travailler de manière séquentielle. A partir d'un premier modèle construit, l'individu est pondéré en fonction de la performance de la prédiction. Grâce à cette étape, toutes les observations seront pondérées en fonction de cette performance, pour attribuer lors de la construction du modèle suivant un poids supérieur aux individus dont la valeur a été mal prédite. Cela permet d'ajuster les poids au fur et à mesure des modèles pour mieux prédire les valeurs difficiles.

Le gradient boosting n'est autre qu'un boosting qui utilise le gradient de la fonction de perte à chaque étape de calcul de poids des individus.

Comme nous l'avons vu lors de la première partie, les comportements diffèrent fortement d'un mode de jeu à l'autre. Il semblait donc essentiel de bien les distinguer aussi lors de cette étape charnière, et donc d'adapter les modèles en fonction, ce qui donne potentiellement 3 modèles.

2) Exécution du modèle

Le principal problème était donc de créer ces modèles sur l'ensemble de la base. Inutile de penser à réaliser l'algorithme en local sur les machines, mais il était également trop lourd pour la RAM sur la machine adaptée au Big Data, Odin. De plus nous n'avions pas les retours nous permettant de savoir à quel moment le script plantait, car il était généralement lancé en fin de séance, étant donné le temps potentiellement très long de calcul.

A ce problème, une solution : nous avons utilisé une méthode que l'on appelle **méthode incrémentale**.

Elle consiste à diviser le jeu d'apprentissage en 10 échantillons (le nombre d'échantillons pouvant être changé). Le modèle est d'abord appliqué au premier échantillon. Puis au deuxième, en prenant en compte les lacunes du modèle précédent, ce qui consolide et améliore les résultats du deuxième modèle. Cette étape est répétée dix fois pour obtenir lors de la dernière itération le modèle le plus qualitatif possible.

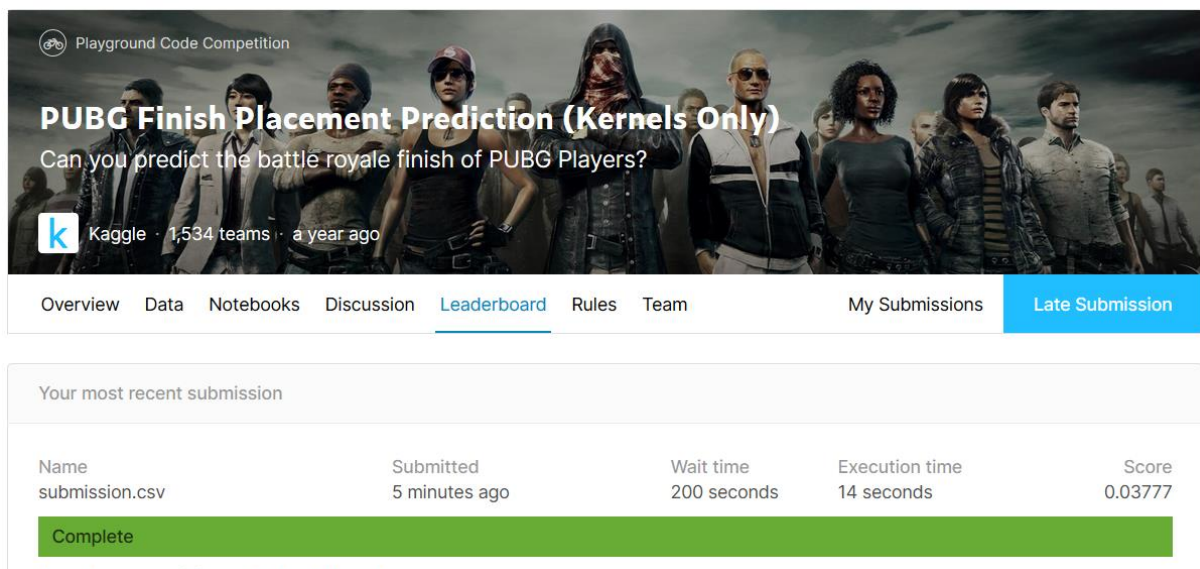
Cette méthode permet donc de diviser par au moins 10 la quantité de données stockée en RAM lors de la modélisation, et les modèles sont calculés de manière beaucoup plus rapide.

3) Application du modèle

Après avoir trouvé les paramètres qui donnaient le meilleur résultat, il a suffi de l'appliquer au jeu de test pour juger de sa capacité à prédire sur des jeux inconnus.

Nous avons commencé par réaliser la modélisation en ne séparant pas les types de match, puis nous avons réalisé un modèle par type de match, afin de voir si la deuxième méthode était plus efficace ou non.

Le taux d'erreur absolu moyen obtenu avec le modèle comportant toutes les données est de 3,777 %, alors que le taux avec le deuxième modèle est de 3,793 %. C'est donc le premier modèle qui obtient le meilleur score. Ce taux d'erreur nous permet d'être classé 637^{ème} sur un peu plus de 1500 participants, ce qui est très correct.



Playground Code Competition

PUBG Finish Placement Prediction (Kernels Only)

Can you predict the battle royale finish of PUBG Players?

Kaggle · 1,534 teams · a year ago

Overview Data Notebooks Discussion **Leaderboard** Rules Team My Submissions Late Submission

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	5 minutes ago	200 seconds	14 seconds	0.03777

Complete

6. Conclusion

Finalement, nous avons respecté le cheminement initialement prévu : après avoir pris en main les variables de notre jeu de données, nous avons pu sélectionner celles qui nous paraissaient, - d'un point de vue pragmatique et à travers des méthodes statistiques-, les plus pertinentes.

Pour finir, nous avons mis au point un modèle, basé sur une méthode incrémentale de gradient boosting, qui nous a permis de traiter notre nombre très volumineux de données. Il nous a donné un taux d'erreur de 3,777 % sur le jeu de test, qui est le résultat le plus satisfaisant parmi toutes les situations essayées lors du processus.

Nous finissons à une place de **637** sur plus de 1 500 participants.

7. Annexes

1) Annexe 1 : glossaire

Boost : Différents items que les joueurs peuvent récolter pour améliorer leurs compétences de manière momentanée.

Headshot : Balle tirée dans la tête provoquant une morte instantanée.

Kills : action de tuer un joueur adverse, ce qui l'élimine et termine sa partie.

Assist : action de blesser un individu, qui se fait ensuite tuer par un coéquipier. Cette action n'est donc réalisable qu'en mode duo/squad, puisqu'il faut des coéquipiers pour cela.

Heal : item permettant de récupérer de l'énergie.

2) Annexe 2 : mode de classement Elo

A l'origine, le classement Elo (inventé par Arpad Elo) est un système d'évaluation comparative utilisé pour le niveau des joueurs d'échec ou d'autres jeux de un contre un.

Son calcul se repose sur la fonction : $E_{n+1} = E_n + K[W - p(D)]$, en utilisant l'écart entre le résultat attendu ($p(D)$), le résultat de la partie (W) ; un coefficient de développement (K), et l'ancien Elo (E_n).

Tout l'intérêt de ce classement réside donc dans sa capacité à évoluer et à prendre compte tant des performances passées que des présentes et futures.

3) Annexe 3 : Matrice de corrélation

Pour savoir quelles variables sont les plus intéressantes à conserver, nous avons construit une matrice de corrélation, en mettant en avant les variables les plus corrélées entre elles.

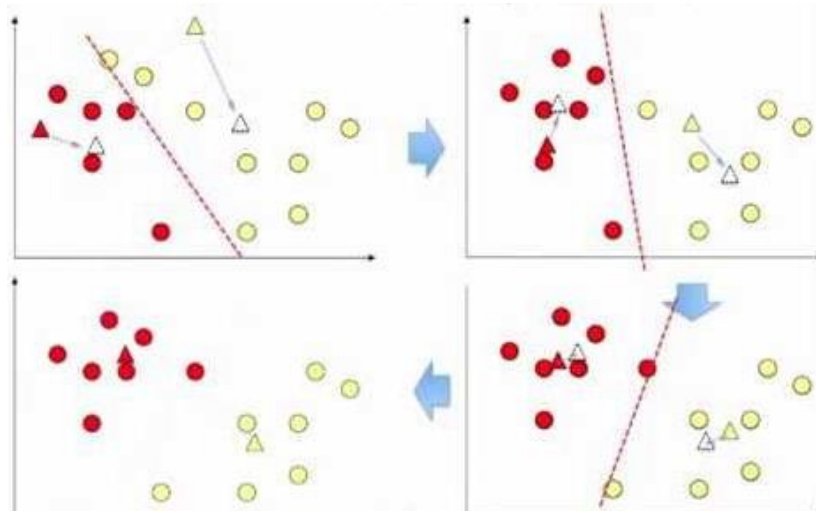
Notre méthode de suppression a été la suivante :

- Regarder les variables corrélées à plus de 80% ensemble
- Pour chacune d'elle, observer le taux de corrélation avec la variable à expliquer
- Supprimer la variable la moins corrélée à la variable à expliquer

	assists	boosts	damage Dealt	headshot Kills	heals	kill Place	kill Points	kills	longest Kill	match Duration	num Groups	revives	ride Distance	road Kills	swim Distance	team Kills	vehicle Destroys	walk Distance	weapons Acquired	win Place Perc
assists	100%	31%	41%	20%	23%	-29%	4%	32%	26%	-2%	-15%	20%	11%	1%	2%	1%	6%	29%	24%	30%
boosts		100%	52%	33%	54%	-55%	1%	50%	42%	7%	-1%	25%	33%	4%	11%	1%	9%	64%	41%	63%
damageDealt			100%	61%	34%	-68%	5%	89%	56%	-1%	-4%	26%	14%	5%	4%	2%	8%	40%	35%	44%
headshotKills				100%	20%	-47%	2%	67%	45%	-2%	1%	15%	8%	1%	3%	1%	4%	25%	22%	28%
heals					100%	-39%	0%	31%	26%	11%	-6%	24%	30%	2%	8%	4%	6%	43%	31%	43%
killPlace						100%	-2%	-73%	-54%	-1%	2%	-27%	-24%	-6%	-9%	-4%	-7%	-59%	-49%	-72%
killPoints							100%	4%	1%	-10%	-4%	1%	-3%	0%	0%	-4%	-1%	0%	0%	1%
kills								100%	60%	-3%	0%	25%	11%	6%	4%	2%	8%	37%	34%	42%
longestKill									100%	6%	-2%	17%	19%	3%	5%	1%	8%	41%	28%	41%
matchDuration										100%	4%	2%	37%	2%	1%	5%	6%	15%	13%	-1%
numGroups											100%	-16%	-5%	1%	1%	-4%	-2%	-8%	-4%	4%
revives												100%	11%	1%	2%	3%	4%	24%	17%	24%
rideDistance													100%	10%	5%	5%	12%	31%	28%	34%
roadKills														100%	0%	1%	3%	2%	2%	3%
swimDistance															100%	1%	1%	17%	8%	15%
teamKills																100%	11%	2%	4%	2%
vehicleDestroys																	100%	8%	6%	7%
walkDistance																		100%	54%	81%
weaponsAcquired																			100%	58%
winPlacePerc																				100%

Sur la matrice ci-dessus, à ce stade les variables suivantes ont été supprimées via la méthode expliquée ci-dessus : **maxPlace**, **winpoints**, **DBNOs**, **rankpoints** et **killStreaks**. Certaines variables sont encore très corrélées entre elles, mais nous avons ajouté à notre process une analyse pragmatique. Selon nous, ces variables (kills, damageDealt...) sont trop importantes dans la détermination de la variable à expliquer pour les supprimer.

4) Annexe 4 : Méthode des k-means



Pour résumer de manière simple, la méthode des k-means tient en la suite d'opérations suivantes :

- Définir k centres aléatoirement
- Chaque observation est affectée à la classe du noyau le plus proche
- Un nouveau centre de gravité est calculé grâce à la composition de chaque groupe
- Les étapes 2 et 3 sont répétées jusqu'à atteindre un état stable

5) Annexe 5 : Retour d'expérience

« Cette étude fut particulièrement enrichissante, de par son organisation –travailler en équipe avec des partenaires aux compétences parfois complètement différentes- et son contenu.

Travailler sur ce sujet nous a permis d'approfondir nos compétences en termes de méthodes, puisque nous avons dû parfois creuser pour trouver la solution la mieux adaptée, et de nous familiariser à la plateforme Odin, et donc à la gestion du Big Data et ses avantages/contraintes.

La principale barrière fut les ralentissements, voire impossibilité d'exécuter certains codes en raison du volume très conséquent de notre base et la difficulté des calculs exigés. »