Openclassrooms.com

2022

**Rapport de projet**

OpenClassrooms, Project 8 (project libre)

Construire des modèles prédictifs pour les événements sportifs MMA

Mykola Mykhailytskyi

# REMERCIEMENTS

Je tiens à remercier tous ceux qui m'ont soutenu pendant ce long voyage. OpenClassrooms pour avoir créé la plateforme d'apprentissage elle-même, mes pairs qui ont toujours été là pour m'aider, et mes mentors. Jean Savary, Valérie Tiedrez et surtout Iselin Martial Naoussi Kuitche.

Ce fut une expérience précieuse tant dans mon nouveau domaine professionnel de l'analyse de données qu'en français. Malgré le fait d'étudier une langue étrangère et d'échouer souvent, tout le monde m'a soutenu et aidé. Dans cet esprit, je voudrais m'excuser à l'avance pour toutes les erreurs, les fautes d'orthographe et les phrases non naturelles qui pourraient apparaître dans ce rapport. L'apprentissage du français est toujours un projet en cours pour moi. Un jour, j'espère parler couramment le français et l'analyse de données.

# SOMMAIRE

[**REMERCIEMENTS**](#_nql5rhg3lvg) **2**

[**SOMMAIRE**](#_65nnlobwikkj) **3**

[**CONTEXTE**](#_iy9lroca5mus) **4**

[**PROBLÉMATIQUE**](#_m6rofviln31m) **5**

[**MISSIONS CONFIÉES**](#_wsb2l5w5rgud) **6**

[Analyse exploratoire des données (EDA)](#_fjk29gir5hh9) 6

[Nettoyage des données](#_cmej4yvw7zm2) 9

[Construction de modèles prédictifs (avec normalisation)](#_3vl7d13ocz5q) 12

[Construction de modèles prédictifs (pas de normalisation)](#_59syzlakhp2g) 14

[Poids des caractéristiques individuelles](#_vg1qrrd0ldor) 16

[**BILAN**](#_7royqa9y9dto) **19**

# CONTEXTE

Je suis un ancien rédacteur et support client qui a décidé de changer de carrière. Après avoir analysé mes points forts, ce que j'aime faire et ce que je fais le mieux, mon choix s'est porté sur le domaine de l'analyse de données. Étant l'un des meilleurs agents du service clientèle de Blizzard Entertainment, je me suis toujours appuyé sur les données qui m'étaient accessibles. J'ai creusé dans les statistiques de mon travail - nombre de réponses par catégorie, par période de la journée, types de problèmes, de quel jeu, si nos joueurs étaient satisfaits de la réponse elle-même, si le problème a été résolu avec mon conseil ou non, etc. Lentement mais sûrement, j'ai pu trouver des réponses à mes propres questions, comme par exemple comment donner des réponses captivantes, informatives et précieuses aux problèmes que les gens rencontraient avec nos jeux.

Lorsque le moment est venu de passer à autre chose, j'ai relevé le défi d'apprendre de nouvelles choses afin de devenir analyste de données. Une partie du plan d'apprentissage consistait à rejoindre la plateforme OpenClassrooms et à suivre leur cours d'analyste de données.

Le programme d'OpenClassrooms demande aux étudiants de réaliser un projet dans un domaine professionnel qui les intéresse. Comme mon parcours professionnel est ancré dans l'industrie du jeu, je suis généralement une personne compétitive qui aime le sport et les jeux. Mon projet libre consisterait à prédire l'issue des combats d'arts martiaux mixtes (souvent abrégés en MMA). L'idée est de créer un produit (modèle d'apprentissage automatique) pour une société de paris ou dans le cadre de notre propre startup. Nous aimerions construire un modèle qui nous permettrait de prédire l'issue des combats de MMA, sur la base de données historiques. Le résultat idéal est un modèle qui peut à la fois prédire le vainqueur et donner une probabilité suffisamment élevée (>65-70%) pour ce résultat. De cette façon, nous pourrions également tirer parti des facteurs de risque lorsque nous décidons de parier selon le modèle ou contre lui.

Les paris sportifs représentent une industrie de plusieurs milliards de dollars. Sa popularité au niveau des marchés régionaux et mondiaux en fait une activité très attractive. Divers rapports prévoient une croissance deux à trois fois supérieure à celle de l'industrie en l'espace de 10 ans. La lutte contre le cancer est l'une des principales activités de ce secteur.

L'un des sujets les plus intéressants et les plus complexes que j'ai découverts en analyse de données est l'apprentissage automatique et les algorithmes prédictifs. Alors que l'objectif du projet est de construire un modèle prédictif réussi, mes objectifs d'apprentissage sont les suivants :

* Approfondir le domaine de l'apprentissage automatique
* S'entraîner à résoudre des problèmes de la vie réelle avec l'aide de l'AD et de l'apprentissage automatique.
* Expérimenter différentes approches des problèmes de ML
* Comprendre ce qui fait que le projet ML fonctionne ou non, ce qui peut être amélioré.

Un autre objectif de ce projet pourrait être pour les professionnels du MMA eux-mêmes. Par exemple, les entraîneurs et les combattants, qui veulent comprendre ce qui est plus ou moins important (mesures clés) et orienter l'ensemble de leur entraînement. Ou même construire un entraînement spécialisé orienté vers la défaite d'adversaires spécifiques.

# PROBLÉMATIQUE

Est-il possible de construire un algorithme d'apprentissage automatique fiable pour prédire le résultat d'un événement sportif ?

# MISSIONS CONFIÉES

## Analyse exploratoire des données (EDA)

Ce type de projet étant nouveau pour moi, j'ai dû commencer par élaborer un plan, afin de pouvoir m'appuyer sur quelque chose. Mon plan initial s'est avéré très similaire aux précédents projets d'analyse de données réalisés pour OpenClassrooms. Le plan initial ressemblait à ceci :

* Récupérer le jeu de données MMA sur Kaggle (récupéré sur le site Web de l'UFC).
* Charger, explorer, nettoyer et préparer les données pour nos modèles.
* Appliquer 3-5 différents types de modèles d'apprentissage automatique et voir lesquels sont les plus performants.
* Essayez et ajustez les performances de ces modèles
* Donnez à notre (nos) meilleur(s) modèle(s) des données complètement nouvelles et voyez comment il(s) se comporte(nt).

Après avoir chargé les données brutes dans le cadre de données Pandas, je me suis rapidement rendu compte que le nombre de colonnes (caractéristiques pour les modèles d'apprentissage automatique) est très élevé. Au cours de projets antérieurs, j'ai eu l'occasion de mettre en œuvre quelques modèles ML simples, et la théorie et la documentation me rappelaient qu'un nombre très faible ou élevé de colonnes dans vos données pouvait entraîner des problèmes pour tous les modèles existants.Nos données brutes sont constituées de 5144 lignes, chaque ligne représentant un seul combat. Il y a 145 colonnes qui donnent diverses descriptions statistiques ou littérales de ces combats. La majorité des données sont numériques, par exemple l'historique de chaque combattant (victoires, défaites, séries de victoires et de défaites, combien de fois il a combattu pour le titre, taille, poids, portée de bras, etc.) et les statistiques de chaque combat (combien de fois le combattant rouge a tenté de donner un coup de pied avec sa jambe, combien de fois il a réussi à donner un coup de pied, combien de rounds ont été nécessaires pour déterminer le vainqueur, etc.) Certaines des données sont stockées sous forme de chaînes de caractères et d'objets. Il s'agit principalement d'informations lisibles par l'homme, comme le nom des combattants, l'arbitre, le lieu, la catégorie de poids, etc. Avec cela à l'esprit, je prévoyais déjà de changer les données de chaînes et d'objets en dummies ou d'appliquer LabelEncoder, puisque les modèles ML ne seraient pas capables de lire les types de données originaux de ces colonnes.

Une exploration plus approfondie de l'ensemble de données a montré que les enregistrements ont commencé en novembre 1993 et qu'actuellement, les dernières données de combat dont nous disposons datent de juin 2019.

Puisque nous voulons prédire qui a gagné le combat (Rouge ou Bleu), nous savons que notre variable dépendante sera la colonne "Gagnant". Nous regardons la distribution dans cette colonne et nous voyons qu'il y a une grande disproportion.La variable dépendante contient 3470 gagnants rouges (Red), 1591 gagnants bleus (Blue) et 83 tirages (Draw). En nous rappelant les conseils que nous avons reçus dans le cadre du projet "détecter les faux billets de banque", nous notons ici que pendant la division des données en test et train, nous devons nous assurer que les distributions sont égales. Cela permet d'obtenir de meilleurs modèles et, par conséquent, de meilleures prédictions.

Voyons combien de tours il faut pour déterminer le vainqueur d'un match particulier :



Étonnamment, la majorité absolue des combats ont tendance à durer 3 rounds. Très peu de combats durent 1, 2 ou 4 rounds. La deuxième durée la plus populaire est celle de 5 rounds.

Le MMA, comme de nombreux sports de combat, comporte plusieurs catégories de poids. Afin de trouver une approche visuelle de ces classes, nous convertissons les livres américaines de l'ensemble de données en kilogrammes et traçons ce qui suit :Maintenant, il serait intéressant de voir si la classe de poids a un effet sur la durée du combat. Pour obtenir cette information, nous utilisons un graphique en forme de violon. Ce type de graphique fonctionne très bien ici:

Compte tenu de nos précédentes conclusions sur la durée des combats, il n'est pas surprenant que la majorité des catégories de poids mettent en moyenne trois rounds pour désigner un vainqueur. Mais grâce à cette visualisation, nous pouvons voir quelques exceptions. Il s'agit des poids ouverts (inférieurs à la moyenne) et des poids plumes féminins (supérieurs à la moyenne).

## Nettoyage des données

Ensuite, nous commençons à préparer notre ensemble de données pour l'objectif du projet. En commençant par les valeurs manquantes, nous constatons qu'une quantité importante de données est manquante. Notamment ~25% des données dans les colonnes qui correspondent aux moyennes des combattants bleus et ~12,64% des données qui correspondent aux moyennes des combattants rouges. D'autres colonnes ont très peu de valeurs manquantes, ce qui ne justifie pas de s'inquiéter à ce stade.

Notre objectif est de pouvoir déterminer entre deux résultats. Comme nous l'avons vu précédemment, la variable dépendante contient 3 issues, mais la 3ème a très peu de points de données. Sur cette base, nous décidons de supprimer l'issue Draw de l'ensemble de données. Dans le même temps, nous supprimons la colonne date et tous les combats (lignes) qui contiennent des valeurs NaN. Cela réduit le nombre de lignes dans notre jeu de données de 5144 à 3151.

Nous devons maintenant nous occuper de 145 colonnes (caractéristiques). Après avoir consulté la documentation Scikit-learn et les didacticiels en ligne, nous décidons que la meilleure solution consiste à supprimer les colonnes qui ne sont pas pertinentes.

Il y a plusieurs façons de procéder, mais la solution optimale à ce stade est de ne laisser que les colonnes critiques (title\_bout, weight\_class, etc.) et celles qui contiennent des données qui devraient toujours exister avant qu'un combat réel ait lieu. Des colonnes telles que la taille, le poids, les victoires et les défaites, les séries, la portée, le nombre total de rounds combattus, les combats pour le titre, etc. Ceci devrait être un bon point de départ pour notre objectif. Si nécessaire, nous pouvons modifier notre approche ou réintroduire certaines colonnes à l'avenir.

En conséquence, nous avons réduit le nombre de colonnes (caractéristiques) de 145 à 30.Ensuite, nous faisons appel à la méthode .info() de pandas et notons les colonnes dont le type de données doit être modifié pour s'adapter aux modèles ML. Les colonnes 'Referee', 'R\_fighter', 'B\_fighter', 'R\_Stance', 'B\_Stance', 'Winner' et 'weight\_class' sont modifiées au format chaîne de caractères approprié. Juste après, nous appliquons LabelEncoder fit\_transform afin de transformer les chaînes en valeurs numériques, lisibles par ML. Essentiellement, nous donnons à chaque arbitre, combattant, position, etc. un numéro d'identification unique.

Après le premier passage de LabelEncoder sur notre variable dépendante, il était difficile de comprendre quelle valeur numérique correspondait à la Red\Blue précédente. Nous avons pu contourner cette confusion en utilisant l'approche suivante :

Nous avons maintenant une confirmation visuelle et un rappel que LabelEncoder a changé tous les gagnants rouges en 1, et tous les gagnants bleus en 0. Cela sera important à l'avenir.

À ce stade, il était évident que nous aurions besoin d'un moyen de tester notre ou nos modèles sur des données totalement nouvelles et inédites. En d'autres termes, nous avions besoin de simuler l'application réelle de notre ou nos modèles. Cette étape peut être réalisée par le biais d'APIs et de web-scraping, mais étant donné les contraintes de temps, notre mentor Martial a conseillé d'utiliser des données existantes. Merci Martial pour cette idée !



Ce que nous avons fait ici, c'est sélectionner au hasard 10 combats (rangées), puis supprimer ces combats exacts de l'ensemble de données. De cette façon, validation\_sample conserve le même format que notre ensemble de données, mais ne sera jamais vu par les modèles jusqu'à ce que nous essayions de les tester.

Puisque nous avons fait beaucoup de manipulations avec notre ensemble de données, nous aimerions voir la distribution actuelle de notre variable dépendante :



La situation s'est améliorée, mais nous pouvons dire avec certitude qu'elle est encore loin d'être parfaite. Un équilibre de 36 à 64 signifie que les points de données sont encore inégaux et que nous devrons procéder à une stratification ou à un équilibrage des poids.

## Construction de modèles prédictifs (avec normalisation)

Il nous faut maintenant sélectionner les modèles qui peuvent répondre à notre objectif. Ayant très peu d'expérience, nous nous appuyons fortement sur la documentation et l'expérience personnelle des autres. Nous avons appliqué un algorithme pour trouver les meilleurs modèles pour une tâche spécifique à partir de scikit-learn https://scikit-learn.org/stable/tutorial/machine\_learning\_map/index.html ainsi qu'en prenant des notes de notre mentor.

En suivant les étapes ci-dessus, nous déclarons que notre échantillon a plus de 50 échantillons, nous voulons prédire une catégorie (0 ou 1), nos données sont étiquetées et la quantité de données est inférieure à 100k. Par conséquent, nous nous retrouvons avec des modèles de type classification qui nous conviendront le mieux. Cela signifie que nous allons essayer d'utiliser les modèles suivants :

* Naive Bayes gaussien
* KNN (classificateur K Nearest Neighbors)
* SVC
* Classificateur Random Forest
* Classificateur Decision Trees

Nous pouvons également essayer d'appliquer d'autres types de modèles, à des fins d'expérimentation.

Notre première approche pour construire des modèles ML est de normaliser nos données après le train\_test\_split, afin d'éviter tout bruit provenant des données elles-mêmes. Après avoir écrit 6 différents types de modèles (5 mentionnés précédemment plus la régression logistique), nous notons leurs scores de précision :

* Régression logistique Score moyen 0.6549
* Score moyen de Naive Bayes gaussien 0,56034
* Arbre de décision Score moyen 0,5760
* Score moyen de KNN 0,6112
* Random Forest Score moyen 0,6454
* Classification par vecteur de support Score moyen 0,6682

A en juger par le score moyen de validation croisée, les modèles les plus performants sont la régression logistique, la forêt aléatoire et le svc. Nous testons ces 3 modèles en stockant leurs différents scores dans un tableau séparé. Nos résultats finaux sont les suivants :



Le score de précision est le nombre d'étiquettes que les modèles prédisent exactement. Le modèle le plus performant selon ce score est le SVC.

Le score de précision est le rapport tp / (tp + fp) où tp est le nombre de vrais positifs et fp le nombre de faux positifs. La précision est intuitivement la capacité du classificateur à ne pas étiqueter comme positif un échantillon qui est négatif. La meilleure valeur est 1 et la pire valeur est 0. Le modèle le plus performant par ce score est la régression logistique.

Le score de rappel est le rapport tp / (tp + fn) où tp est le nombre de vrais positifs et fn le nombre de faux négatifs. Le rappel est intuitivement la capacité du classificateur à trouver tous les échantillons positifs. La meilleure valeur est 1 et la pire valeur est 0. Le modèle le plus performant par ce score est le SVC.

Globalement, le SVC montre les meilleurs résultats, mais la régression logistique n'est que légèrement derrière.

A ce stade, nous essayons de "nourrir" les modèles les plus performants avec de nouvelles données, celles que nous avons stockées dans validation\_sample. Les résultats sont les suivants:

**Logistic Regression** predicts the following winners: [0 0 1 1 1 0 1 1 0 0]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for Logistic Regression on new data is 0.5

Precision score for Logistic Regression on new data is 0.6

Recall score for Logistic Regression on new data is 0.5

**RandomForest** predicts the following winners: [0 0 1 1 0 1 0 1 0 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for RandomForest on new data is 0.9

Precision score for RandomForest on new data is 1.0

Recall score for RandomForest on new data is 0.8333333333333334

**SVC** predicts the following winners: [1 1 1 1 1 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for SVC on new data is 0.6

Precision score for SVC on new data is 0.6

Recall score for SVC on new data is 1.0

Conclusions sur les modèles avec des données standardisées :

* Aucun des 3 modèles les plus performants ne semble bon (sur la base de plusieurs exécutions).
* les 3 sont inconsistants, avec une précision de 0,0 à 0,9. Une moyenne de 0,5 (basée sur de multiples exécutions).
* les modèles actuels ne nous permettent pas de les utiliser pour prédire les combats. Nous devrions essayer d'entraîner les mêmes modèles sur des données originales, non standardisées.

## Construction de modèles prédictifs (pas de normalisation)

La deuxième approche consiste à entraîner nos modèles sur des données originales, non standardisées. Après avoir écrit les mêmes types de modèles, nous notons leurs scores de précision :

* Score moyen de la régression logistique 0,6582 (légèrement augmenté par rapport à 0,6549).
* Score moyen de Naive Bayes gaussien 0,56034 (est resté le même, 0,56034)
* Arbre de décision Score moyen 0,5798 (en légère augmentation par rapport à 0,5760)
* KNN Score moyen 0,5541 (en baisse par rapport à 0,6112)
* Random Forest Score moyen 0,6473 (en légère hausse par rapport à 0,6454)
* Classification par vecteur de support Score moyen 0,6397 (en baisse par rapport à 0,6682)

Une fois encore, les modèles les plus performants sont la régression logistique, la forêt aléatoire et la classification par vecteur de support. Nous testons donc ces 3 modèles en stockant leurs différents scores dans un tableau séparé. Nos résultats finaux sont les suivants :



Il n'y a pas de vainqueur clair, mais nos expériences ont prouvé que les résultats sont ici incohérents. Parfois, les modèles formés sur des données standardisées sont légèrement plus performants, parfois les modèles formés sur des données originales sont légèrement plus performants. L'observation la plus cohérente à ce stade est que les deux modèles les plus performants sont la régression logistique et le classificateur à vecteur de support.

Maintenant, nous "alimentons" à nouveau les modèles les plus performants avec de nouvelles données, celles que nous avons stockées dans validation\_sample. Les résultats sont ci-dessous :

Logistic Regression predicts the following winners: [1 1 1 0 0 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for Logistic Regression on new data is 0.6

Precision score for Logistic Regression on new data is 0.625

Recall score for Logistic Regression on new data is 0.8333333333333334

RandomForest predicts the following winners: [1 1 0 1 1 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for RandomForest on new data is 0.5

Precision score for RandomForest on new data is 0.5555555555555556

Recall score for RandomForest on new data is 0.8333333333333334

SVC predicts the following winners: [1 1 1 1 1 1 1 1 1 1]

Actual fight results: [0 1 1 1 0 1 0 1 0 1]

Accuracy score for SVC on new data is 0.6

Precision score for SVC on new data is 0.6

Recall score for SVC on new data is 1.0

Conclusions sur les modèles avec des données non standardisées :

* la différence entre les scores est très faible, mais aussi incohérente. Les scores des modèles basés sur les données originales sont parfois légèrement supérieurs, parfois légèrement inférieurs (sur la base de plusieurs exécutions).
* il semble que l'entraînement sur les données originales donne des prédictions plus cohérentes (sur la base de plusieurs exécutions)
* nous pourrions avoir besoin d'évaluer la capacité à voir quelle caractéristique (colonne) a plus de valeur pour le modèle, par rapport aux autres.

## Poids des caractéristiques individuelles

Afin de trouver les caractéristiques importantes (celles qui, selon nos modèles, ont le plus de valeur lorsqu'on essaie de prédire l'issue du combat), nous utilisons la bibliothèque SHAP pour Python : https://shap-lrjball.readthedocs.io/en/latest/index.html.

SHAP (SHapley Additive exPlanations) est une approche de la théorie des jeux qui permet d'expliquer les résultats de tout modèle d'apprentissage automatique. Elle relie l'allocation optimale de crédits aux explications locales en utilisant les valeurs de Shapley classiques de la théorie des jeux et leurs extensions connexes.

Avec l'aide de cette bibliothèque, nous pouvons calculer les valeurs d'importance de chaque caractéristique et les tracer sur le graphique. Nos résultats pour le modèle de régression logistique sont les suivants :



Les caractéristiques sont triées en fonction de leur impact, du plus élevé (en haut) au plus faible (en bas). Il s'agit de l'axe Y. L'axe X correspond aux valeurs SHAP. Chaque point est une observation distincte. Le code couleur aide à comprendre la signification de la caractéristique correspondante. Le bleu est faible, le rouge est élevé (il n'y a pas de lien entre ce code couleur et notre variable fiable du gagnant rouge-bleu). Par exemple :

* Si la caractéristique B\_total\_rounds\_fought est plus élevée, il est fort probable que le combattant bleu gagne.
* Si la caractéristique R\_total\_title\_bouts devient plus élevée, le combattant rouge a de fortes chances de gagner.
* Si B\_reach\_cms devient élevé, le combattant bleu a plus de chances de gagner ; s'il devient faible, le combattant rouge a plus de chances de gagner.

En plus des poids de la bibliothèque SHAP, nous pouvons utiliser les coefficients de la régression logistique elle-même et comparer les deux.

En comparant les deux, nous pouvons constater à la fois des différences radicales et des pondérations similaires. Par exemple, SHAP quantifie la caractéristique B\_total\_rounds\_fought comme la plus importante, mais les coefficients du modèle lui-même ont une valeur opposée, la plaçant plus bas que la 20ème place. En revanche, la caractéristique B\_losses est dans le top 3 pour les deux méthodes.

Sur la base de la documentation fournie pour SHAP et scikit-learn, nous voyons que leurs calculs sont différents. Cela signifie que nous ne devrions pas faire aveuglément confiance à une seule méthode d'évaluation du poids des caractéristiques. Ce serait une bonne idée d'expérimenter l'ingénierie des caractéristiques basée sur les coefficients des deux sources (ou même de trouver d'autres moyens de quantifier les poids).

# BILAN

L'une des premières leçons importantes apprises au cours du projet est qu'un grand nombre de colonnes (caractéristiques) ne se traduit pas nécessairement par de meilleurs modèles d'apprentissage automatique. Cela peut introduire du "bruit dans les données", compliquer excessivement l'apprentissage du modèle (plus nous avons de données, plus il faudra de temps de traitement à un PC pour effectuer tous les calculs) et prendre beaucoup de temps.

Lors de la planification des modèles à utiliser, nous nous sommes largement appuyés sur la documentation existante et l'expérience de nos collègues. Mais étonnamment, le modèle le plus performant en termes de score s'est avéré être la régression logistique, qui n'a pas été proposée par l'algorithme scikit-learn. Il est suivi de très près par Random Forest et SVC. Ce qu'il faut retenir ici, c'est que chaque problème n'a pas de solution "parfaite" ni de "meilleur" modèle. Parfois, nous devons expérimenter et trouver ce qui fonctionne le mieux pour notre cas particulier, notre ensemble de données et notre problème.

En outre, il a été surprenant de constater que les résultats des trois modèles les plus performants sont incohérents. Alors que nous avons parfois vu une précision de ~65-67%, ce score n'était pas assez cohérent. Cela signifie que même si la régression logistique ou le SVC ont produit une précision de 0,9 sur une exécution particulière, la suivante pourrait produire une précision de 0,3 ou 0,5. Malheureusement, cela n'était pas bon pour l'objectif de notre projet de faire des prédictions fiables. En d'autres termes, il serait trop risqué de prédire à la fois selon nos modèles et contre eux.

Les améliorations possibles pour ce projet sont, entre autres, les suivantes :

1. Donner plus de poids aux données récentes

Notre idée ici est que même si l'ensemble de données commence à décrire des combats de 1993, ces vieilles statistiques peuvent avoir une qualité ou une importance moindre que de nos jours. Après tout, tout évolue et s'améliore avec le temps. Si nous ne spécifions pas cela à nos modèles, ils pourraient traiter les anciennes données avec la même importance, ce qui fausserait les résultats.

1. Essayez d'utiliser toutes les colonnes/caractéristiques existantes.

L'une des améliorations possibles pour notre projet est d'augmenter le nombre de caractéristiques. Nous avons supprimé les statistiques de chaque combat individuel (combien de coups de poing ou de coups de pied ont été donnés par chaque combattant, combien ont été réussis, etc.) Nous pourrions les conserver et en déduire divers paramètres historiques pour chaque combattant qui pourraient aider aux prédictions, par exemple le % d'attaques réussies pour chaque combattant. Cela pourrait aider avec les prédictions.

1. Egaliser les échantillons de validation pour avoir 50% de gagnants rouges et 50% de gagnants bleus.

Notre échantillon de validation était basé sur un échantillon aléatoire de 10 combats. Bien que nous ayons pu équilibrer les distributions dans l'ensemble de données d'entraînement, nous ne l'avons pas fait pour les échantillons de validation.

1. Essayez d'autres types de modèles (XGBoost, etc.).
2. Construire une meilleure représentation des prédictions.

Nous pourrions améliorer l'expérience de l'utilisateur en donnant un code couleur aux résultats, en donnant des noms aux gagnants, etc.

1. Construire une meilleure façon de montrer la confiance du modèle dans une prédiction spécifique.

En plus des gagnants prédits, nous pourrions offrir aux utilisateurs un niveau de confiance pour cette prédiction.

1. Présenter à l'utilisateur un ensemble de facteurs les plus précieux pour la prédiction.

Cela nous ramène à notre tentative de pondération des caractéristiques individuelles. Par exemple, si un modèle donne une prédiction avec un faible niveau de confiance, l'affichage des facteurs de valeur qui ont joué dans cette décision pourrait aider les utilisateurs à décider - parier avec ou contre la prédiction.

1. Créer une interface conviviale (déploiement sur le Web).

La cerise sur le gâteau pour tout projet de ce type serait une application web entièrement développée, qui permettrait à quiconque de l'utiliser facilement depuis n'importe quel appareil.