Visualization of Massive Data

Module présentée par Nicolas Lehir

Document récapitulatif réalisé par Nicolas Deviers

Nicolas Deviers
Nathanael Melouki
Antoine La Mache

Choix du dataset :

https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results

Ce dataset rassemble de nombreuses données sur tous les participants aux Jeux Olympiques depuis l'édition de Rio de 1896.
Il contient les champs suivants :

1. **ID** - Unique number for each athlete
2. **Name** - Athlete's name
3. **Sex** - M or F
4. **Age** - Integer
5. **Height** - In centimeters
6. **Weight** - In kilograms
7. **Team** - Team name
8. **NOC** - National Olympic Committee 3-letter code
9. **Games** - Year and season
10. **Year** - Integer
11. **Season** - Summer or Winter
12. **City** - Host city
13. **Sport** - Sport
14. **Event** - Event
15. **Medal** - Gold, Silver, Bronze, or NA

Il est intéressant car très fourni, et offre un large potentiel d'informations que l'on pourrait en extraire en recoupant les différents champs.

Le dataset est situé dans l'archive archipe.zip, dans le dossier dataset. Il faut l'extraire afin d'exécuter les scripts.

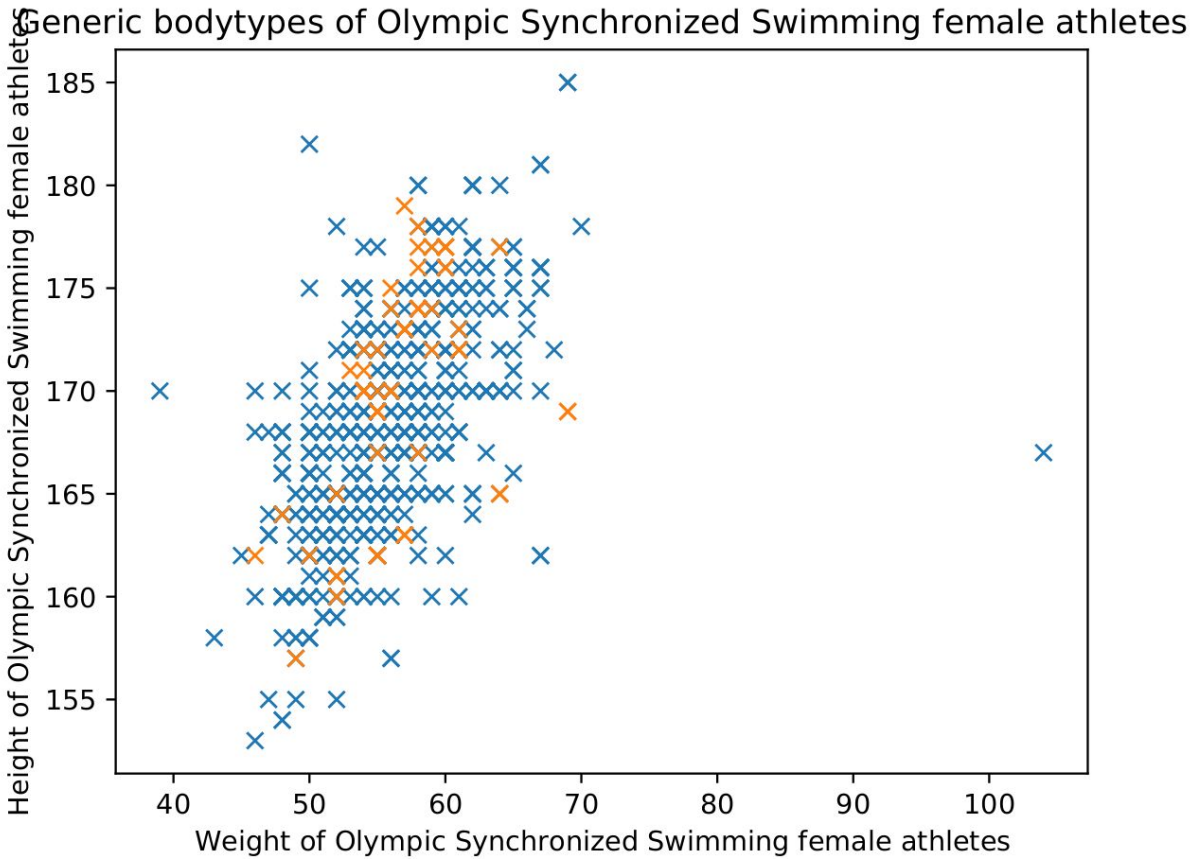
Visualization :

1 - Scatter (Nicolas Deviers)

Deux versions de scatter ont été réalisées, qui sont les suivantes :

La première version compare la taille et le poids des athlètes pour chaque discipline olympique où ces données sont fournies. Un fichier est généré par sexe aussi, les mensurations étant différentes. Sur le graphique, les athlètes ayant obtenu une médaille d'or ont leur point coloré d'une couleur distincte. Cela permet de voir s'il existe un gabarit optimal pour un sport. Puisque la professionnalisation du sport est assez récente (pouvoir vivre de son sport sans travailler à côté et donc s'entraîner à fond sans contrainte), je n'utilise que les données datant d'après 1980.

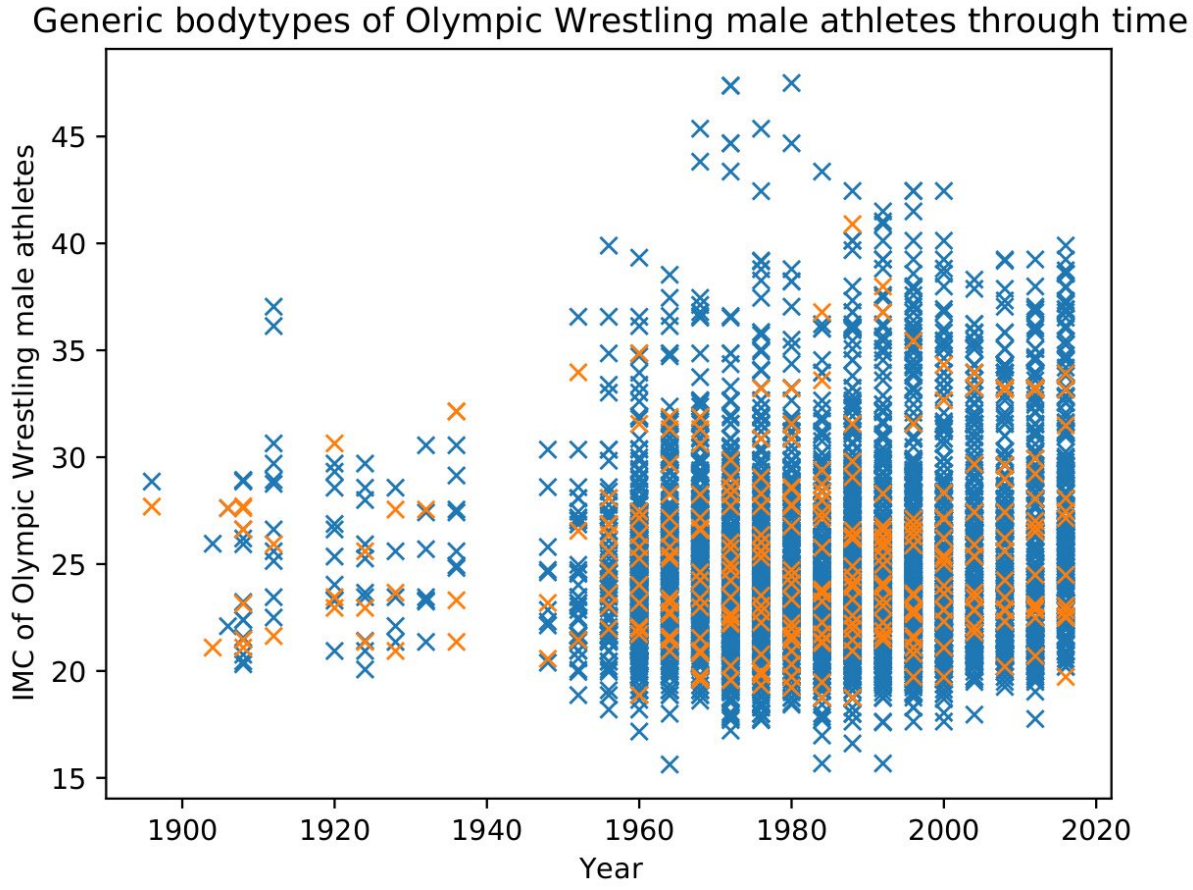
Exemple :



Ce schéma nous montre les mensurations des athlètes de nage synchronisée féminine depuis 1980. On y remarque que tous les points sont relativement concentrés dans un rectangle entre 1m60 et 1m78 de hauteur et entre 48 kg et 65 kg de masse. Cependant, la majorité des médailles d'or sont rassemblées entre 1m68 et et 1m78 de hauteur et entre 51 kg et 61 kg de masse

La seconde version compare l'IMC des mêmes athlètes au cours des années, et ce depuis 1896 pour toutes les disciplines. On pourra y voir à quel point cette même professionnalisation du sport à pu impacter le physique des athlètes au travers d'un siècle d'amélioration constante des techniques d'entraînement.
La formule de l'IMC est le poids divisé par le carré de la taille.

Exemple :



Ce schéma nous démontre l'évolution de l'IMC des lutteurs masculins depuis 1896. On y remarque tout d'abord une augmentation importante du nombre de participants - représenté par le nombre de croix - après la seconde guerre mondiale accompagné un élargissement impressionnant des variations d'IMC. La ou celui-ci était confiné à une fourchette contenue entre 20 et 30 avant la guerre, il se retrouve disséminé sur un véritable rateau oscillant entre 18 jusqu'à 40 ! Cependant la majorité des champions Olympiques restent confinés entre 20 et 30.

Les graphes générés sont stockés dans le dossier scatterImage/ a côté des scripts. Les fichiers des graphes générés par la seconde version sont reconnaissables par leur nom commençant par 'ZZ_'.
Une aide à l'utilisation du script est incorporée dans celui-ci avec l'argument '-h'.

L’exécution du programme étant assez longue à la taille du dataset et au nombre de disciplines, l’utilisateur peut décider de ne l’exécuter que sur une seule discipline en la précisant comme argument au programme à son exécution. La liste des disciplines disponibles est disponible avec l’argument -h.

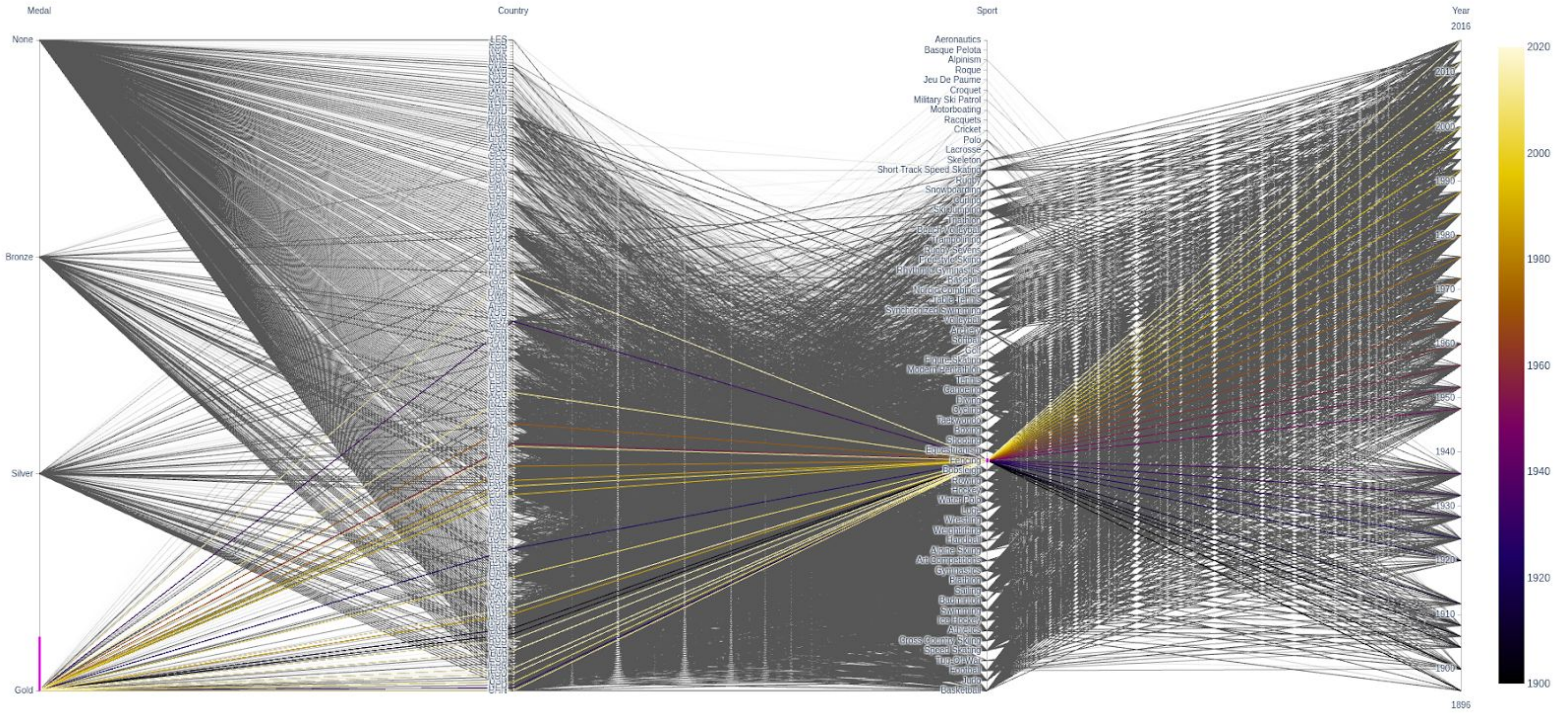
2 - Parallel Coordinates (Nicolas Deviers)

Le schéma de Parallel Coordinates est extrêmement complet. Il recoupe les colonnes des années, des sports, des pays et des médailles. On peut y voir en somme toutes les médailles obtenues par toutes les équipes dans tous les sports et ce pour toutes les éditions des jeux Olympiques depuis 1896.

Le schéma est généré par la librairie Plotly et s’ouvre dans le navigateur. Il est particulièrement lourd car il manœuvre 4 colonnes de 271116 données, plus d’un millions de points sont de ce fait chargés. Un bon ordinateur est conseillé pour son utilisation.

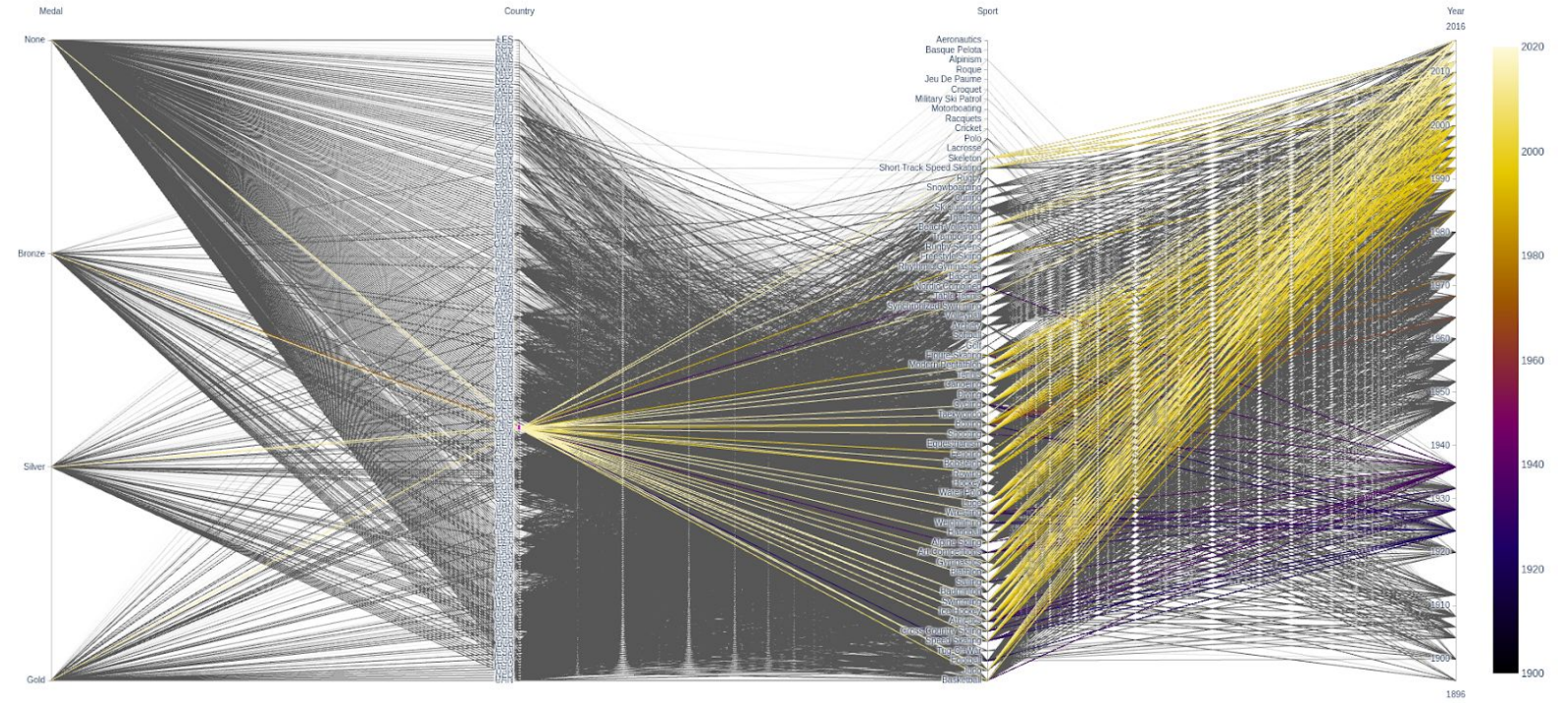
Le format des données du dataset n’étant pas compatible avec le schéma Plotly, celui-la est donc réinterprété afin de n'intégrer que des entiers dans le schéma représentant l’index de la réelle donnée, collée sur les parallèles sous forme de légende de Schéma. Ainsi un écran de grande résolution est recommandé, sinon quoi les indicatifs des noms des pays risquent de se confondre.

Exemple :



Voici un exemple de l'utilisation de ce schéma. En isolant la discipline Escrime et la médaille d'Or, je suis en capacité de retrouver la nationalité de tous les vainqueurs depuis 1896 ainsi que l'année de leur victoire.

Exemple :



Une seconde utilisation du schéma. En ne sélectionnant qu'un pays en particulier, on peut retrouver toutes les disciplines pour lesquelles il participait ou alors les années ou il a pu présenter une délégation.

Quantitative Analysis :

2 - Unsupervised : (Nathanaël Mélouki)

Afin de pouvoir de fit un modèle non supervisé sur les données de ce dataset, il est d'abord nécessaire de manipuler et mettre les données en ordre. Un réseau de neurones ne peut pas traiter des données nulles ou non numériques.

On doit donc repérer toutes les données 'NaN' du dataset. Dans le cas des médailles, il s'agit du label (médaille gagnée = 1, pas de médaille = 0), on remplace donc toutes les médailles gagnées par 1 (qu'elles soit bronze, argent ou or) et les NaN par 0.

On vérifie le nombre de données manquantes :

```
Sex            0
Age         9474
Height     60171
Weight     62875
Team           0
NOC            0
Year           0
Season         0
City           0
Sport          0
Event          0
Medal          0
dtype: int64
```

On peut remplacer les valeurs manquantes par la valeur moyenne. Il faut cependant prendre en compte le fait que homme et femme ont des poids moyen et taille moyenne différente.

		Age	Height	Weight	Year
Medal	**Sex**				
0	**F**	23	167	59	1992
	M	26	178	75	1974
1	**F**	24	170	63	1992
	M	26	181	79	1966

On constate qu'il y a une différence marquante entre homme et femme.
On remplace donc les valeurs manquantes des hommes par la valeur moyenne du poids masculin ou de la taille masculine, de même pour les femmes.

Une fois cela fait, on a cependant encore un jeu de données avec beaucoup de données catégoriques. On retire les colonnes non pertinentes pour prédire le gain du médaille ou non : ID, Nom et lieu des jeux olympiques (ID, Names, Games).

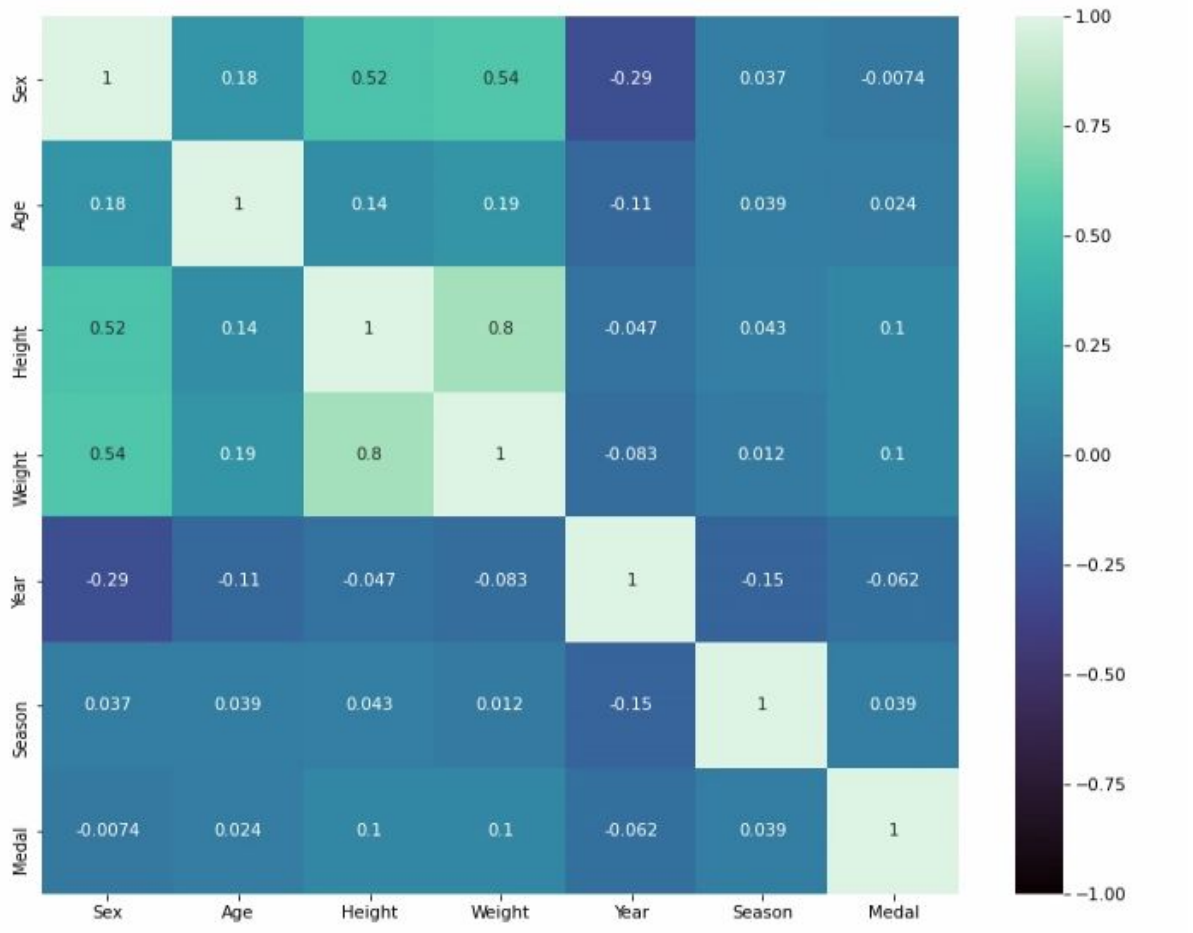
Pour les données catégoriques restantes on effectue un one hot encoding pour transformer ça en données quantitatives. Cela vectorise les données. On a cependant une grande augmentation de la dimensionnalité. On passe de 11 features et 2 labels à 2293 features et 2 labels.

Cela a posé quelques problèmes dû aux limitations hardware du matériel à notre disposition :

```
MemoryError: Unable to allocate 4.63 GiB for an array with shape (2293, 271116) and data type int64
```

Pour surmonter ce problème, nous avons emprunté un PC plus puissant.

Une fois les données catégoriques transformé en vecteurs utilisable par un modèle, nous faisons une rapide heatmap des corrélations des différentes features pour évaluer quels features seront pertinente pour l'apprentissage :



La corrélation la plus forte est entre le poids et la taille. Cependant, pour notre classe 'Medal' (i.e si un athlète à gagné une médaille ou non) les corrélations sont assez faibles.

Avant de commencer l'entraînement de notre modèle on effectue une normalisation des données via StandardScaler de scikit-learn.

```
array([[ 0.61568294, -0.24955675,  0.45006609, ..., -0.00814841,
        -0.01572221, -0.01583913],
       [ 0.61568294, -0.40873789, -0.60545848, ..., -0.00814841,
        -0.01572221, -0.01583913],
       [ 0.61568294, -0.24955675,  0.23896118, ..., -0.00814841,
        -0.01572221, -0.01583913],
       ...,
       [ 0.61568294,  0.22798666,  0.02785626, ..., -0.00814841,
        -0.01572221, -0.01583913],
       [ 0.61568294,  0.70553007,  0.97782838, ..., -0.00814841,
        -0.01572221, -0.01583913],
       [ 0.61568294,  1.34225462,  0.97782838, ..., -0.00814841,
        -0.01572221, -0.01583913]])
```

Avant de commencer le training, on regarde la représentation de chaque classe (1 => à gagné une médaille, 0=> n'as pas gagné de médaille) y_train étant un array de 0 et de 1, on peut simplement calculer la représentation de ces classes via une moyenne :

```
14.7% / 85.3%
```

On voit qu'il y a une grosse disparité avec presque 15% de médaille gagné (label 1) et 85% de médaille perdue (label 0).

Cela crée un déséquilibre dans le dataset qu'il faudra prendre en compte sur le résultat du modèle.

Pour le modèle, on utilise 4 layers :

* Un layer d'input de dimension de la shape de notre dataset (X dans le code)
* un layer Dense de dimension 64 avec une activation Relu (Diminution de la dimensionnalité)
* un layer Dense de dimension 64 avec une activation Relu
* Un layer Dense d'ouput, de dimension 1 avec une activation sigmoid (>0.5 => output 1, <0.5 => output 0)

On utilisera l'accuracy et l'air under the curve (+ ROC) (https://towardsdatascience.com/understanding-the-roc-and-auc-curves-a05b68550b69) comme metrics puisqu'il y a un désiquilibre des classes dans le dataset.

On arrête l'entraînement du modèle si au bout de 3 epochs il n'y a pas d'améliorations.

On observe une accuracy à 91% et une AUC à 89%.

```
, 0.9143542051315308, 0.8923681378364563
```

Ce qui est plutôt bon comme résultat au vu du déséquilibre présent dans le dataset ainsi que de la faible corrélation vu dans la heatmap entre le fait d'avoir gagné une médaille et les features présent dans le dataset.

Si on prend le temps de regarder le rapport de classification :

```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95     69458
           1       0.81      0.54      0.65     11877

    accuracy                           0.91     81335
   macro avg       0.87      0.76      0.80     81335
weighted avg       0.91      0.91      0.91     81335
```

On constate une grande précision sur la classe 0 (médaille non gagnée), ce qui est attendu puisqu'on a un grand nombre de représentations de cette classe dans le dataset.
La précision pour la classe 1 (médaille gagnée) est plutôt bonne avec 81%.

Cependant, si on regarde le score de recall, on constate un bon score sur la classe 0 (médaille non gagnée) mais seulement 54% de recall sur la classe 1 (médaille gagnée). Ce n'est pas loin d'un pile-ou-face.

Si on regarde la matrice de confusion, on constate effectivement qu'il y a quasiment autant de vrai positif que de faux négatifs.

```
Confusion Matrix:
 [[68000  1458]
 [ 5508  6369]]
```

Le modèle est donc sûrement overfit par le déséquilibre du dataset et le manque de corrélations dans les données sur la victoire d'une médaille.

Librairies :

pandas : utilisation de la fonction read_csv pour extraire seulement les colonnes du dataset nécessaires à l'exécution du programme. Mise en forme des données, one hot encoding (get_dummies) et manipulation du dataset en général.

plotly : utilisation des fonctions Figure et Parcoords pour la création et mise en forme du graph ParallelCoords

myplotlib : utilisation des fonctions plot, title, xlabel, ylabel, savefig et close pour la création, mise en forme et enregistrement des graphes des scripts Scatter.

Seaborn : Librairie python basé sur matplotlib pour la visualisation de données (utilisé pour faire une heatmap)

Tensorflow : Librairie python permettant de créer des réseaux de neurones profonds.

Scikit-learn : Outil prédictif d'analyse de données, utilisé pour la normalisation des données (StandardScaler) ainsi que pour les métriques d'évaluation (matrice de confusion et rapport de classification).