

Projet de Prédiction de la Valeur Marchande des Joueurs

Sommaire:

1. Découpage des NoteBooks.....	2
2. Les Urls.....	3
3. Le scraping.....	4
4. Le prétraitement des données.....	6
5. Visualisations.....	7
6. Feature Engineering.....	8
7. Prédiction du modèle.....	10
8. Conclusion et Sources:.....	11

Introduction:

Le but de ce projet est de développer un estimateur d'apprentissage supervisé pour prédire la valeur marchande d'un joueur à partir des données de Transfermarkt. Pour ce faire nous allons entraîner notre modèle avec les valeurs des précédents mercato. Le but étant de prédire des valeurs au plus proches de la réalité.

1. Découpage des NoteBooks

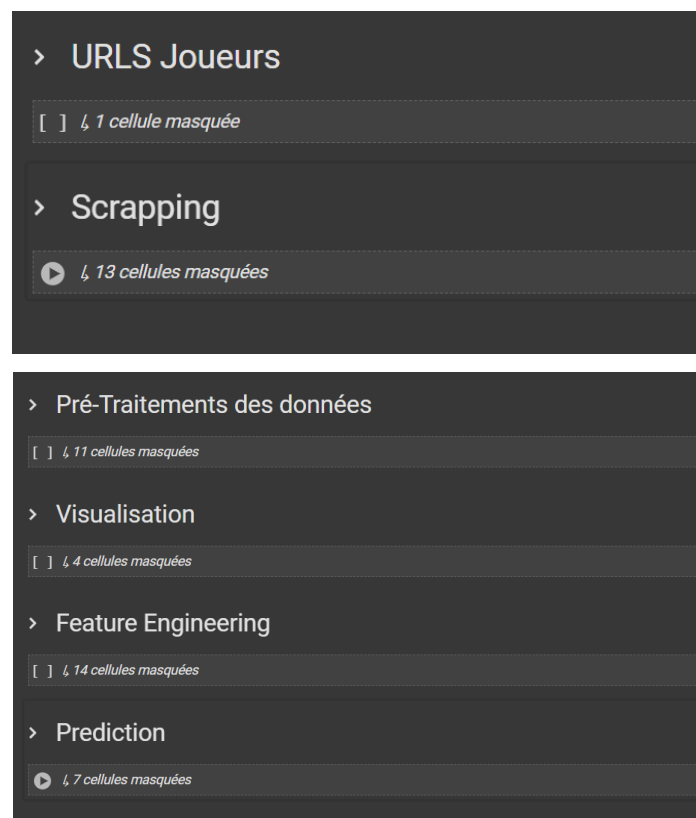
Afin de réaliser le projet, j'ai créé 2 Notebooks, l'un regroupant les sections:

1. Les URLs
2. Le scraping

Son but est de récupérer les données de Transfermarkt et de les convertir en fichier csv.

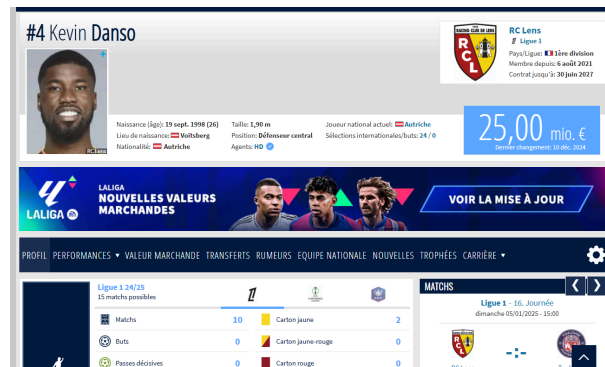
Le second Notebook a pour but de transformer la donnée brute en donnée utilisable pour mon modèle de prédiction :

3. Le prétraitement des données
4. Une visualisation
5. Le feature engineering
6. La prédiction du modèle.



2. Les Urls

Afin de réaliser le scraping, il était nécessaire, dans un premier temps, de savoir où les données pouvaient être récupérées. J'ai personnellement opté pour l'idée de scraper les données sur la page de chaque joueur de Transfermarkt.



Le point positif était de pouvoir trouver une grande quantité d'informations sur le joueur directement sur la même page.

Le point négatif était de devoir remplir manuellement la liste, ma liste finale se compose de plus de 300 joueurs provenant des 5 plus grands championnats.



Afin de déterminer quel joueur a été récemment transféré, je me suis informé sur la page transfert de chaque championnat.

3. Le scraping



















Pour le scraping, j'ai eu l'idée de créer une fonction itérative qui récupère la liste des URLs et scrape les données pour les stocker dans une liste.

J'ai donc créé deux fonctions : la première s'occupe, comme le montre l'image de la partie 1, de récupérer les informations des joueurs. Voici les informations récoltées : (toutes ces valeurs ne seront pas utilisées pour le modèle).

```
record = {  
    "player_id": player_id,  
    "player_name": player_name,  
    "player_number": player_number,  
    "player_value": player_value,  
    "player_club": player_club,  
    "player_league": player_league,  
    "position": position,  
    "age": age,  
    "contract_end": date_fin_contrat,  
  
    # Données performance  
    "competition": i.get("competitionDescription"),  
    "season": i.get("nameSeason"),  
    "gamesPlayed": i.get("gamesPlayed"),  
    "goalsScored": i.get("goalsScored"),  
    "assists": i.get("assists"),  
    "yellowCards": i.get("yellowCards"),  
    "redCards": i.get("redCards"),  
    "minutesPlayed": i.get("minutesPlayed"),  
}  
all_records.append(record)
```

Suite à cela, j'ai créé une deuxième fonction de scraping dont le but est de récupérer les prix des anciens transferts, une donnée qui me semble importante pour le futur modèle.

Pourquoi créer une nouvelle fonction ? Parce qu'il m'était difficile de scraper cette information sur la page principale, et j'ai donc dû aller sur la page des transferts du joueur.

PROFIL PERFORMANCES ▼ VALEUR MARCHANDE TRANSFERTS RUMEURS EQUIPE NATIONALE NOUVELLES					
HISTORIQUE DES TRANSFERTS					
Saison	Date	Venant de	Allant à	VM	Montant de transfert
21/22	6 août 2021	 FC Augsburg	 RC Lens	5,00 mio. €	5,50 mio. € >
20/21	30 juin 2021	 F. Düsseldorf	 FC Augsburg	5,00 mio. €	Fin du prêt >
20/21	18 août 2020	 FC Augsburg	 F. Düsseldorf	6,30 mio. €	Montant du prêt: 500 K € >
19/20	30 juin 2020	 Southampton	 FC Augsburg	6,30 mio. €	Fin du prêt >
19/20	8 août 2019	 FC Augsburg	 Southampton	8,00 mio. €	Montant du prêt: 4,00 mio. € >
16/17	1 mars 2017	 FC Augsburg U19	 FC Augsburg	-	- >
15/16	1 juil. 2015	 FC Augsburg U17	 FC Augsburg U19	-	- >
13/14	1 janv. 2014	 Milton Keynes Do	 FC Augsburg U17	-	Tr. libre >
07/08	1 janv. 2008	 Reading Form.	 Milton Keynes Do	-	Tr. libre >
Montants totaux de transferts:					10,00 mio. €

Pour cette fonction, j'ai également utilisé la liste des URLs, que j'ai ensuite modifiée de manière itérative pour accéder à la page transfert du joueur.

```
for url in urls:
    # 1) Corriger l'URL si elle contient "/profil/"
    if "/profil/" in url:
        url = url.replace("/profil/", "/transfers/")
```

Une fois les deux scrapings terminés, je les regroupe dans un même df_final, que j'exporte au format CSV pour faciliter l'utilisation ultérieure des données.

4. Le prétraitement des données

Dans cette partie, mon objectif est de transformer les données brutes en données exploitables pour la suite.

Cela inclut, par exemple, de convertir une valeur string comme "20 mio. euro" en int, 20 000 000. Il est nécessaire de s'assurer que les types de features sont corrects. J'ai décidé de convertir toutes les données numériques du DataFrame en type int.

Mon fichier csv retourne plusieurs lignes différentes du même joueur en fonction des championnats : Ligue 1, coupe de France, LDC. J'ai donc dû les regrouper par une agrégation.

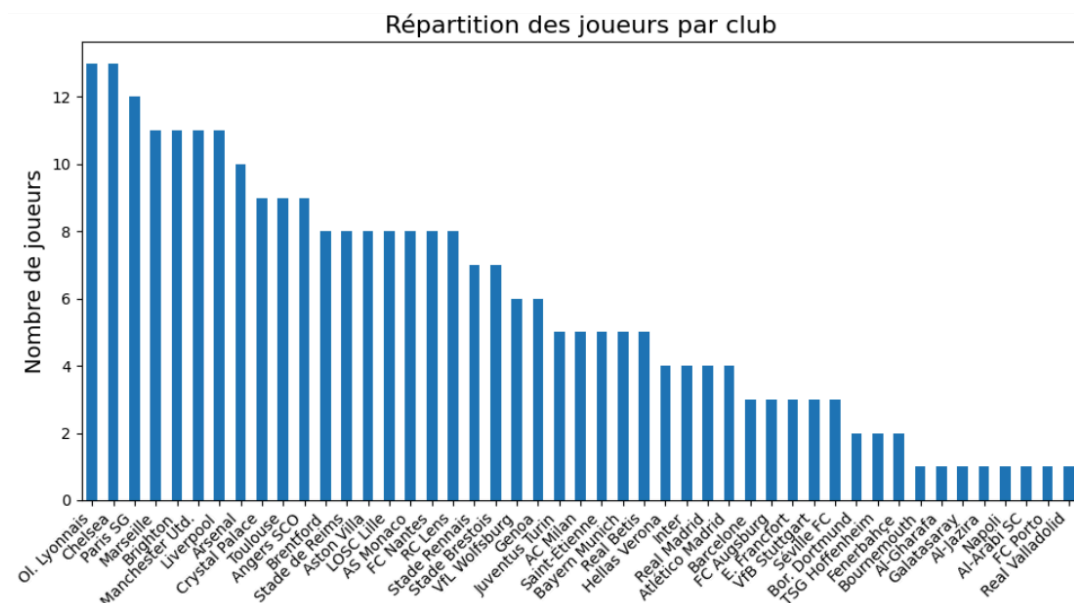
```
df = df.groupby("player_name").agg({
    "player_number": "first",
    "player_value": "first",
    "player_club": "first",
    "gamesPlayed": "sum",
    "goalsScored": "sum",
    "assists": "sum",
    "yellowCards": "sum",
    "redCards": "sum",
    "minutesPlayed": "sum",
    "minutesPlayed_UEFA": "sum",
    "goalsScored_UEFA": "sum",
    "position": "first",
    "age": "first",
    "player_league": "first",
    "transfer_fee": "first",
    "contract_end": "first",
}).reset_index()
```

Dans ce pré-traitement, j'ai aussi transformé la "date d'expiration du contrat" en "durée restante par année". Cela me permet de récupérer une valeur "int" afin de l'utiliser dans le future modèle.

5. Visualisations

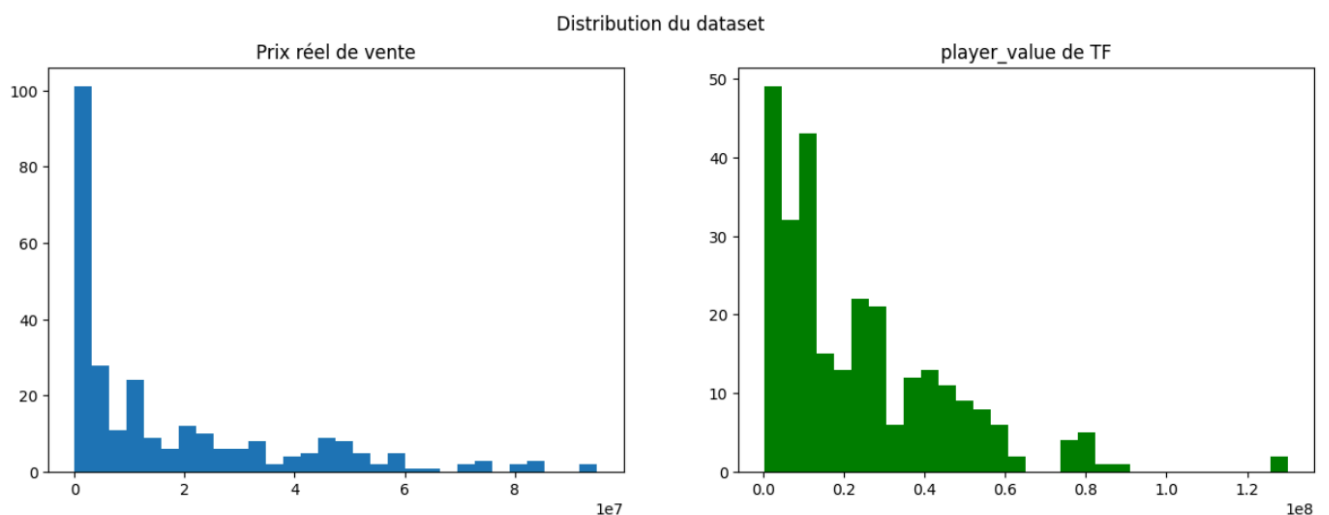
Le but de cette section est d'évaluer et de comprendre la répartition du dataset. C'est une étape importante pour s'assurer que nos données correspondent bien aux valeurs réelles.

Dans notre cas, nous observons:



nb: Le club Francfort et Manchester City sont des outliers provenant du scrapping qui seront par la suite supprimés.

Tous ces joueurs ne seront pas traités car tout dépend si leur transfert est récent.



Nous pouvons aussi comparer la valeur de vente à celle afficher sur transferMarkt.
On observe une tendance de surévaluation des prix par

Sur le notebook se trouve plus d'analyse de la répartition du dataset.

6. Feature Engineering

Cette étape est cruciale pour préparer les données avant de les intégrer dans notre modèle. Elle consiste notamment à identifier les caractéristiques (features) pertinentes et à déterminer sous quelle forme les traiter.

Ma target value est: "Le montant du transfert du joueur" sur la saison 24/25 (Le mercato n'étant pas assez fourni, je me suis aussi basé sur le mercato précédent de 23/24.). Mon but étant d'avoir une approximation du prix réel du joueur sur une période récente. Les autres features permettent d'approcher cette valeur réelle du joueur tout en ajoutant une variation en fonction de ses performances. Si celles-ci sont meilleures, sa valeur réelle va légèrement augmenter et inversement.

En reprenant étape par étape:

1. Je me suis rendu compte qu'une features pour chaque club rend le programme beaucoup trop complexe avec très peu d'informations. Pour le rendre plus simple, j'ai aussi décidé de classer les clubs par catégories en fonction de la somme de leurs valeurs marchandes.

Tier A	Tier B	Tier C
Arsenal	Roma	Al-Arabi SC
Atlético Madrid	AS Monaco	Al-Gharafa
Barcelone	Brentford	Al-Jazira
Chelsea	Crystal Palace	Angers SCO
Liverpool	FC Porto	Bournemouth
Manchester Utd.	Fenerbahçe	FC Nantes
Napoli	LOSC Lille	Real Valladolid
Paris SG	Marseille	Saint-Étienne
Real Madrid	Ol. Lyonnais	Stade Brestois
Tottenham	Real Betis	Stade de Reims
Manchester City	Stade Rennais	Toulouse
AC Milan	Séville FC	Genoa
Juventus Turin	E. Francfort	Hellas Verona
Aston Villa	VfB Stuttgart	FC Augsburg
Brighton	Borussia Dortmund	TSG Hoffenheim
	VfL Wolfsburg	
	Galatasaray	
	RC Lens	

Nous avons au total 3 catégories qui permettent au modèle de beaucoup plus facilement analyser les données.

2. J'ai normalisé (scaled) les variables numériques pour garantir une échelle uniforme. Par exemple, pour la durée restante du contrat, j'ai appliqué une transformation logarithmique afin de mieux gérer les écarts importants entre les valeurs.

3. La variable cible a été isolée dans un DataFrame distinct nommé Y.

4. j'ai regroupé les postes des joueurs en grandes catégories (Gardien, défenseurs, milieux de terrain, attaquants) afin de réduire la complexité du modèle tout en conservant des informations pertinentes.

```
x['Gardien'] = x['Gardien de but']

x['Défenseur'] = (
    x['Arrière droit'] +
    x['Arrière gauche'] +
    x['Défenseur central']
)

x['Milieu'] = (
    x['Milieu central'] +
    x['Milieu défensif'] +
    x['Milieu offensif']
)

x['Attaquant'] = (
    x['Ailier droit'] +
    x['Ailier gauche'] +
    x['Avant-centre'] +
    x['Deuxième attaquant']
)
```

5. Afin de voir la corrélation entre les features et la target value:

player_value	0.798873
gamesPlayed	0.091537
goalsScored	0.250224
assists	0.220355
yellowCards	0.054195
minutesPlayed	0.060619
minutesPlayed_Uefa	0.329064
goalsScored_Uefa	0.425456
age	0.001309
remaining_years	0.226584
club_category_A	0.647903
club_category_B	-0.329823
club_category_C	-0.318343
Bundesliga	0.029512
LaLiga	0.091443
Liga Portugal	-0.026280
Ligue 1	-0.357729
Premier League	0.347243
Serie A	-0.020413
Stars League	-0.083227
Süper Lig	-0.046843
UAE Pro League	-0.057915
Gardien	-0.095934
Défenseur	-0.068500
Milieu	-0.062943
Attaquant	0.184222
dtype: float64	

7. Prédiction du modèle

À l'aide du document *International Journal of Financial Studies*, nous avons observé que le meilleur modèle à utiliser est une régression linéaire couplée à une validation croisée K-fold. Cependant, mon dataset n'étant pas suffisamment fourni, j'ai utilisé une pénalité Ridge pour éviter le surapprentissage (overfitting).

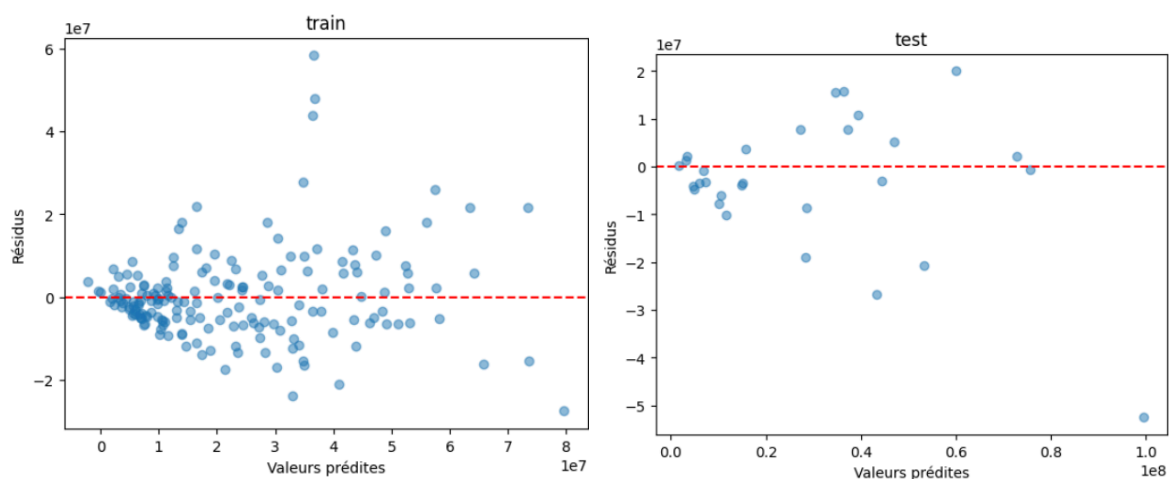
Voici mes résultats :

R2 score:

```
La performance du model pour le training set
-----
Mean R2 score est 0.7383 ± 0.0112

La performance du model pour le testing set
-----
Mean R2 score est 0.6905 ± 0.0643
```

Les performances du modèle montrent une meilleure précision sur l'ensemble d'entraînement que sur l'ensemble de test, ce qui indique un léger surapprentissage. Cela pourrait être lié à une faible quantité de données, à une complexité excessive du modèle, mais aussi au fait de la variation de la valeur du joueur au fil de la saison.



Les graphiques montrent la distribution des résidus (différence entre les valeurs réelles et prédites). On observe dans l'ensemble une bonne adaptation du modèle avec malgré quelques outliers.

8. Conclusion et Sources:

Afin de réaliser ce projet je me suis donc servi d'un modèle de supervised learning sur les prix des joueurs lors du mercato. Ceci me permet donc d'entraîner mon modèle à prédire au plus proche la vraie valeur d'un joueur. Le mercato 24/25, n'étant pas assez fourni, je me suis aussi basé sur le mercato précédent de 23/24. Certes cela entraîne une légère divergence avec la réalité, mais ça aide grandement le modèle à généraliser la donnée. Je me suis basé sur le fait que la valeur d'un joueur ne change pas drastiquement d'une saison à l'autre (sauf exception, ex: fin de contrat).

Une autre approche pourrait être d'implémenter un semi-supervised learning. A l'aide notamment d'un RandomForest. Cela consiste à entraîner le modèle avec des joueurs issue du mercato 24/25 et des joueurs qui n'ont pas d'étiquettes. Mais cette méthode soulève d'autres problèmes comme la complexité de l'algorithme, ou de la qualité des valeurs non-étiquetées.

Sources:

[How to Scrape TransferMarkt.com for Football Data in Python - YouTube](#)

[Web scraping with Python - Stack Overflow](#)

[GitHub - felipeall/transfermarkt-api: API service to get data from Transfermarkt](#)

Statistical Modeling of Football Players' Transfer Fees Worldwide

[Semi-Supervised Deep Learning on Large Scale Climate Models](#)

[Ridge vs Lasso Regression, Visualized!!!](#)