Prédictions des valeurs marchandes des joueurs en 2024 !

Découvrez notre projet IA qui utilise les données de matchs pour anticiper les valeurs des joueurs en 2024.

Lancez la prédiction

Binôme :

- **Taibi Younes**
- **Agharmiou Massine**

Dataset : **Transfermarkt**

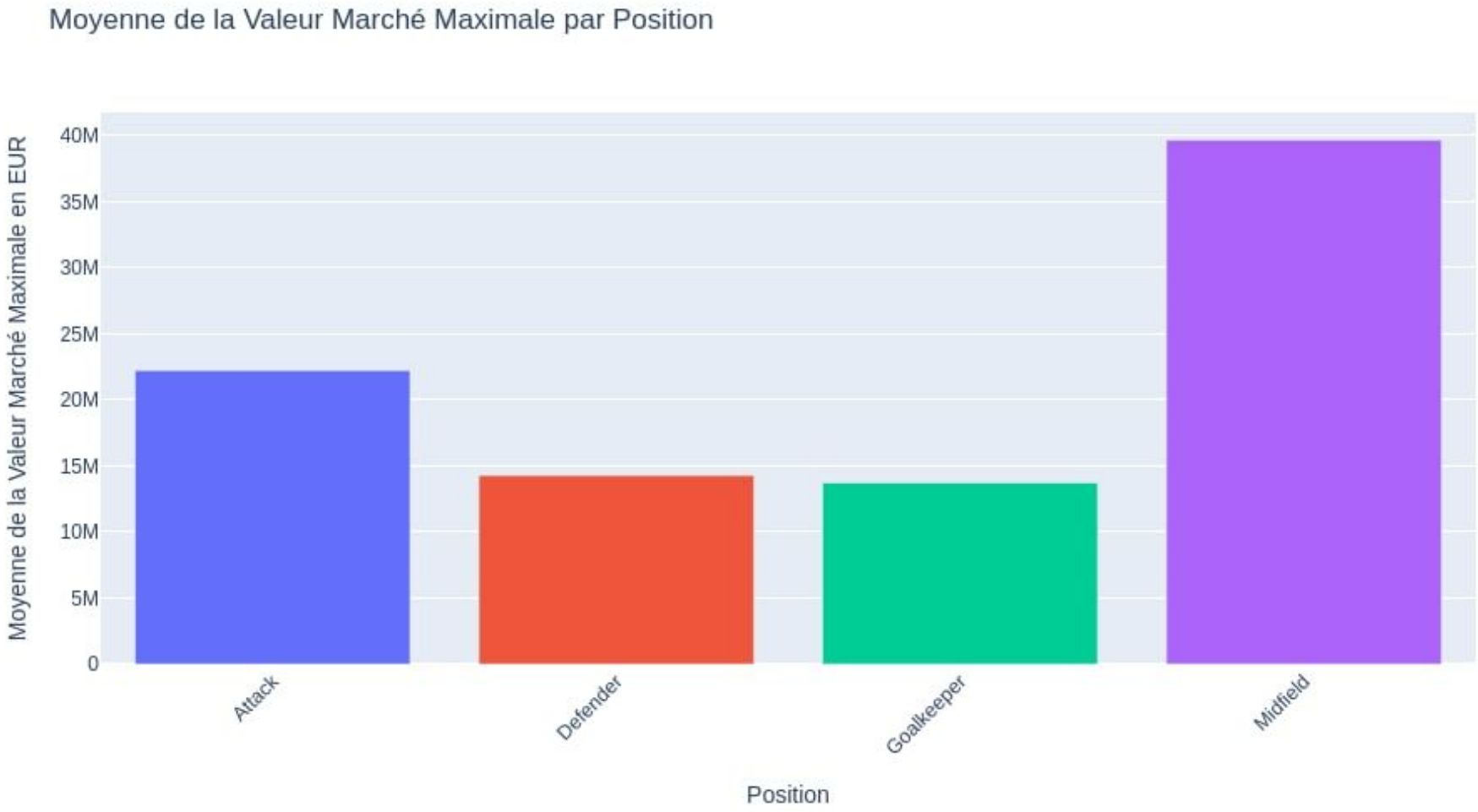
Source : kaggle.com

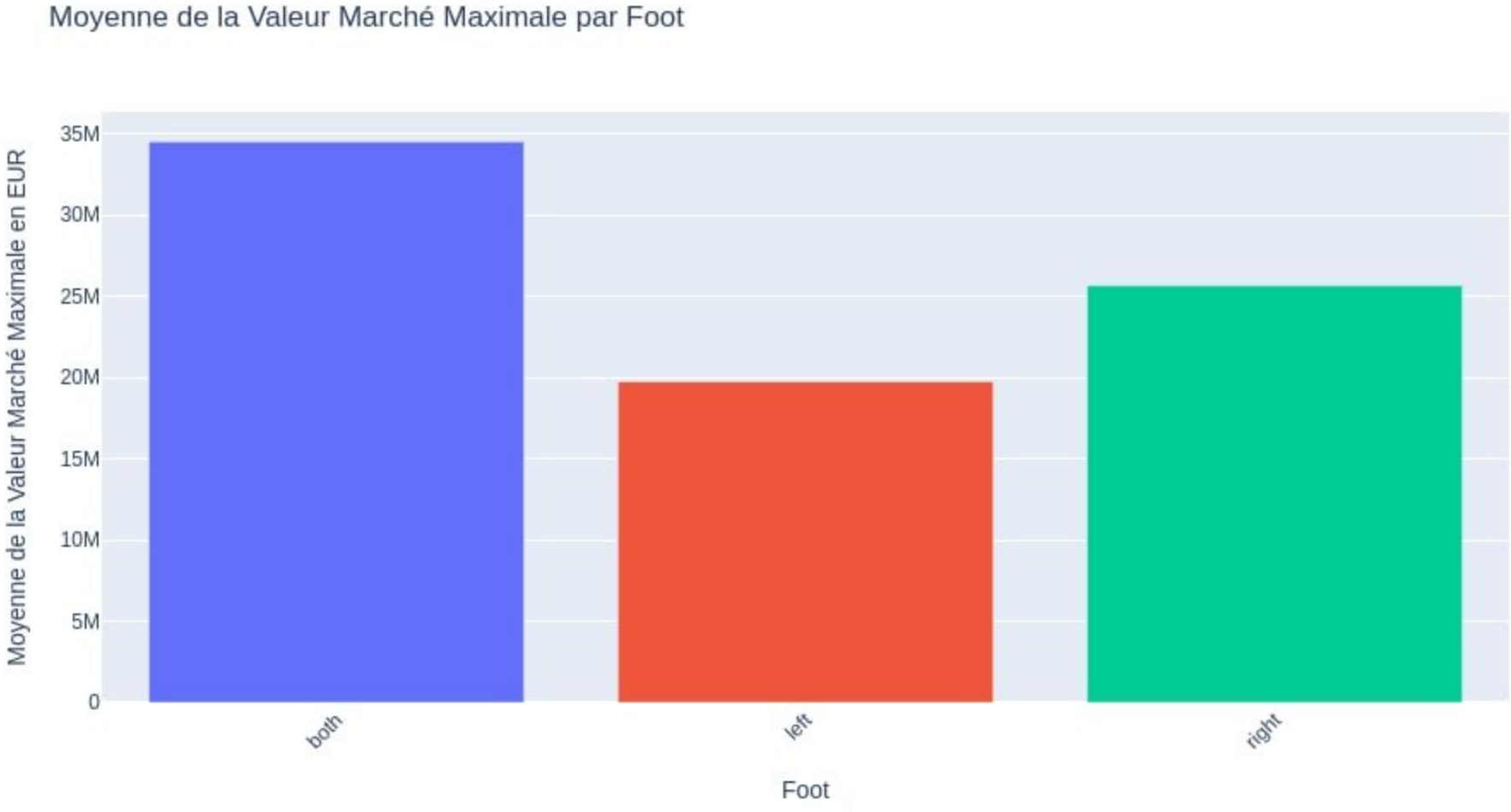
- `400+` clubs
- `30,000+` joueurs
- `60,000+` matchs de diverses saisons dans toutes les grandes compétitions
- `400,000+` enregistrements d'évaluations de joueurs sur le marché
- `1,200,000+` enregistrements des apparitions des joueurs dans tous les matchs.

Analyse du dataset

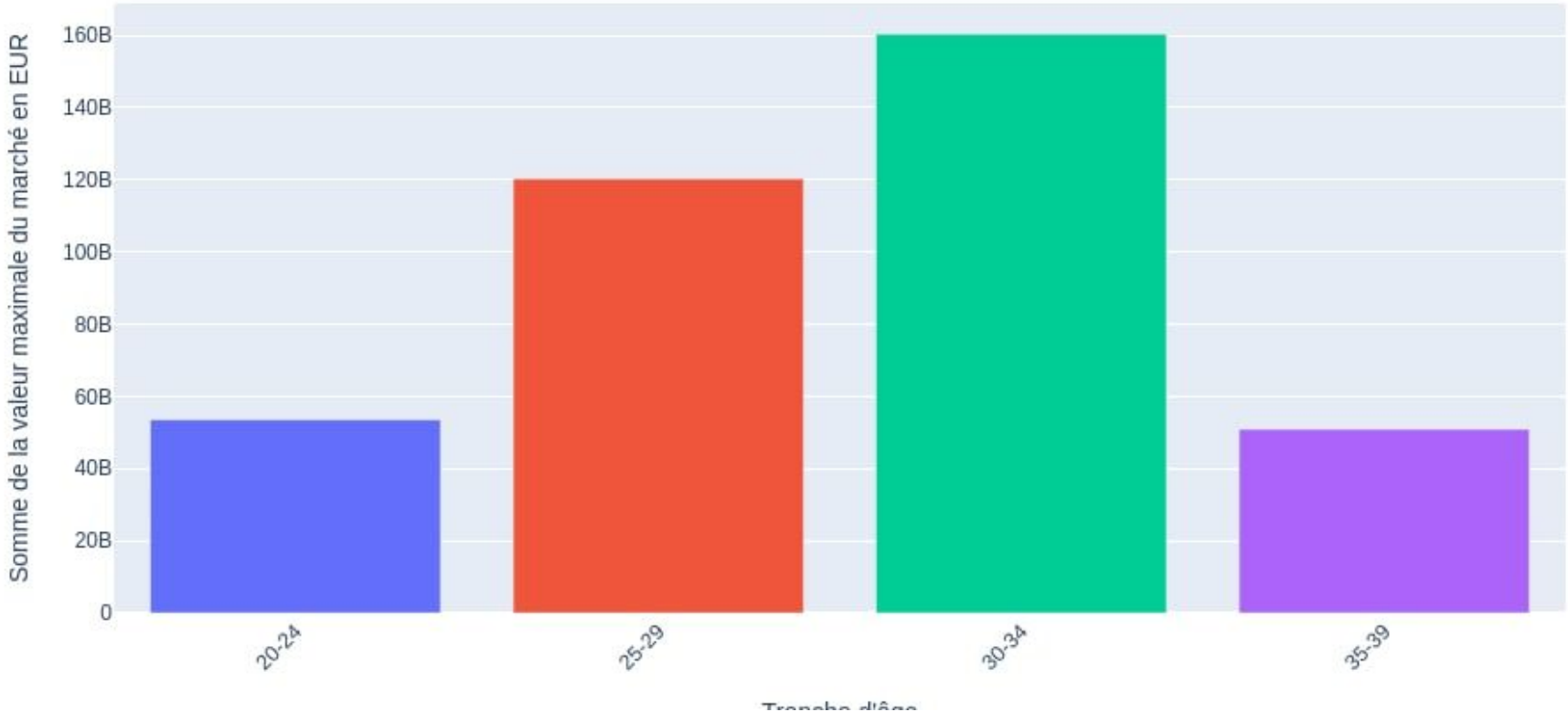
Le dataset est composé de multiples fichiers, avec diverses informations concernant les compétitions, les matchs, les clubs, les joueurs et leurs apparitions qui sont automatiquement mises à jour chaque semaine. Chaque fichier contient les attributs de l'entité et les ID qui peuvent être utilisés pour les jointures.

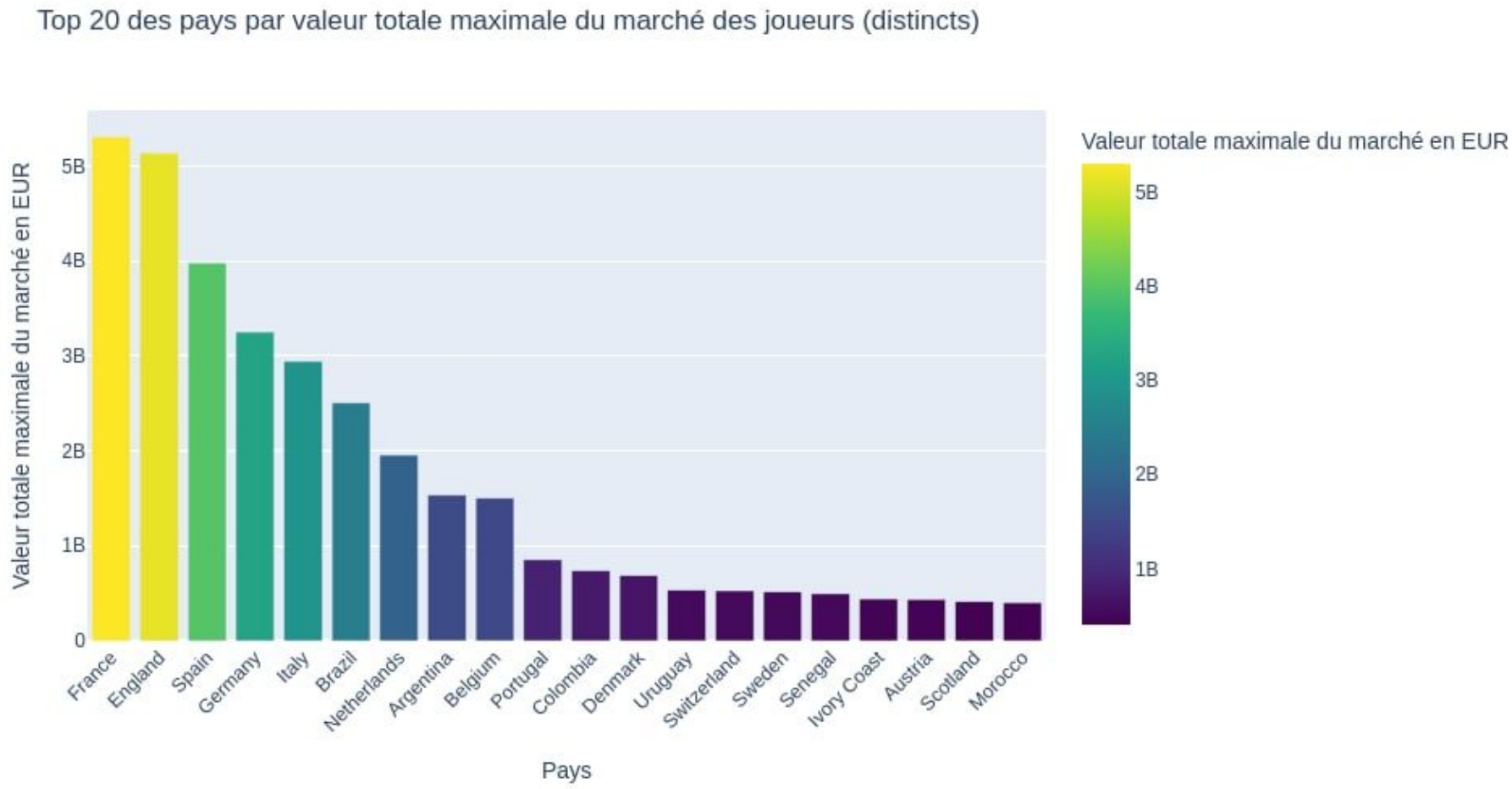
Après observations, on gardé que trois fichiers.

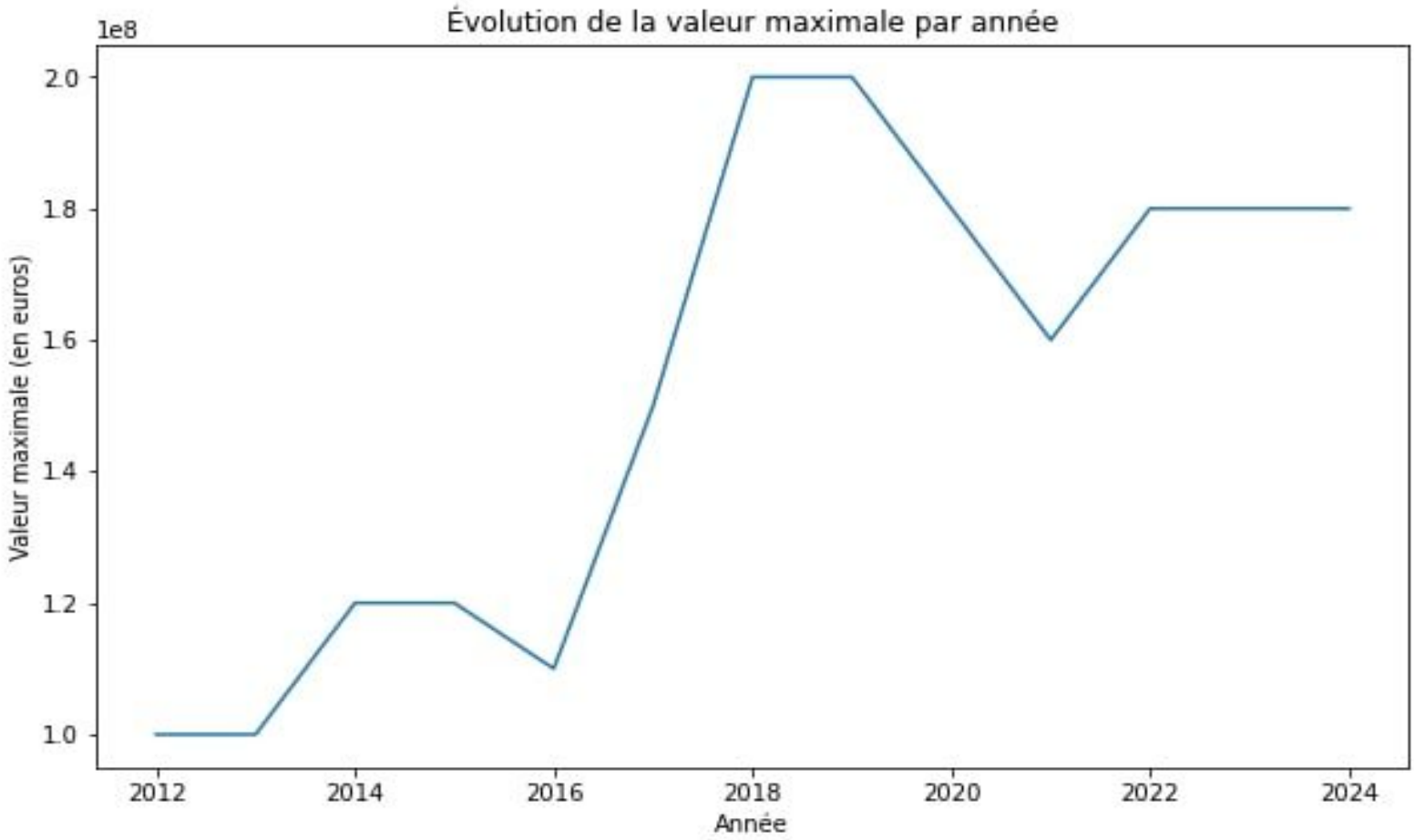

Moyenne de la Valeur Marché Maximale par Position
Moyenne de la Valeur Marché Maximale en EUR
40M
35M
30M
25M
20M
15M
10M
5M
0
Attack
Defender
Goalkeeper
Midfield
Position

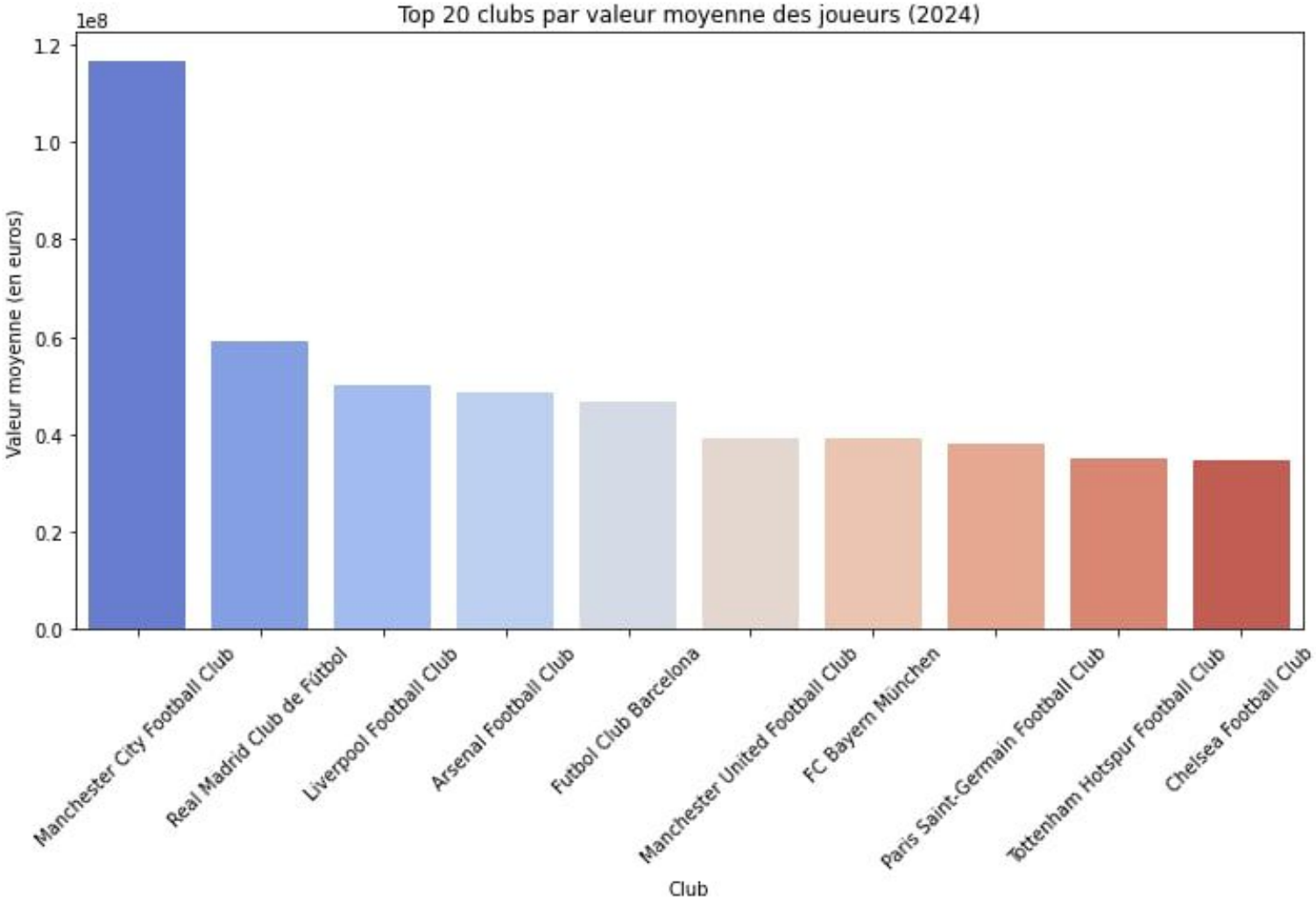

Moyenne de la Valeur Marché Maximale par Foot
Moyenne de la Valeur Marché Maximale en EUR
35M
30M
25M
20M
15M
10M
5M
0
both
left
right
Foot

Somme de la valeur maximale du marché des joueurs par tranche d'âge




Top 20 des pays par valeur totale maximale du marché des joueurs (distincts)
Valeur totale maximale du marché en EUR
5B
4B
3B
2B
1B
0
France
England
Spain
Germany
Italy
Brazil
Netherlands
Argentina
Belgium
Portugal
Colombia
Denmark
Uruguay
Switzerland
Sweden
Senegal
Ivory Coast
Austria
Scotland
Morocco
Pays
Valeur totale maximale du marché en EUR
5B
4B
3B
2B
1B


Évolution de la valeur maximale par année
1e8
2.0
1.8
1.6
1.4
1.2
1.0
Valeur maximale (en euros)
2012
2014
2016
2018
2020
2022
2024
Année


Top 20 clubs par valeur moyenne des joueurs (2024)
1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Valeur moyenne (en euros)
Manchester City Football Club
Real Madrid Club de Fútbol
Liverpool Football Club
Arsenal Football Club
Futbol Club Barcelona
Manchester United Football Club
FC Bayern München
Paris Saint-Germain Football Club
Tottenham Hotspur Football Club
Chelsea Football Club
Club

Nettoyage et enrichissement des données

La DF Players :

- Suppressions des lignes où il y avait des valeurs manquantes.
- Avant suppression : 24000 lignes - Après suppression 13600 lignes
- Les colonnes gardées : **player_id, date_of_birth, name, position, last_season, foot, current_club_name, country_of_citizenship, contract_expiration_date, highest_market_value_in_eur**

  On a ajouté une colonne **Age**.

Nettoyage et enrichissement des données

La DF Evaluation :

- Pas de valeurs manquantes.
- Nombre de lignes : 487953
- Colonnes gardées : **player_id, date, market_value_in_eur, current_club_id**

Nettoyage et enrichissement des données

La DF Appearance :

- Nombre de lignes : 1647829
- Les colonnes gardées : **player_id, date, competition_id, yellow_cards, red_cards, goals, assists, minutes_played**
- On a filtré cette DF pour ne sélectionner que les compétitions de ces pays **Italy, Netherlands, Portugal, Spain, France, Belgium, England, Germany**
- Enrichissement de la DF en ajoutant les colonnes : **minutes_per_goal, minutes_per_assist, minutes_per_yellow_card, minutes_per_red_card**

Jointure entre les DF

- Jointure entre **DF Players** et **DF Player_Evaluation**, sur la colonne player_ID. On a uniquement gardé les correspondances (**DF Merged**)
- Jointure entre **DF Appearance** et **DF Merged**, sur les colonnes **player_ID** et **date** (**DF_FINAL**)

DF Final

- Dans la **DF_FINAL** on a créé **market_value_club**, elle contient pour chaque année, la valeur d'un club qui est la somme de la valeur de ses joueur pour une année x.
- On a ajouté une autre colonne, **value_to_club_ratio** pour chaque joueur, sa valeur est la valeur du joueur divisé par la valeur du club.

```
df_selected['value_to_club_ratio'] = df_selected['market_value_in_eur'] / df_selected['market_value_club']
```

Encoding

On a utilisé **one-hot encoding** :

- Pour la colonne position, on l'a éclaté en 4 : **attack, middle, defend, goalkeeper**
- Pour la colonne foot : **foot_both, foot_left, foot_right**

Normalisation

Pour la normalisation, on a appliqué le logarithme sur **minutes_per_goal, minutes_per_assist, minutes_per_yellow_card, minutes_per_red_card**, **market_value_in_eur, market_value_club**

Pour **date_valuation = dates_courante - date_valuation**

On a exclut les colonnes **: date_of_birth, player_ID, last_season …**

Modèle

Création d’une fonction qui fait des test en utilisant trois modèles différents :

Random forests, Régression Linéaire, Plus proches voisins (KNN)

- Dès la première exécution, random forests a donné un score de 0.86
- Pour la validation croisée on a eu un score de 0.5

Fuite de données

Market_value_club et **value_to_club_ratio** sont des colonnes qui sont calculées à partir de la colonne cible, on a exclut ces deux colonnes, le résultat était très bas : score global 0.2 et pour la validation croisé le score était négatif.

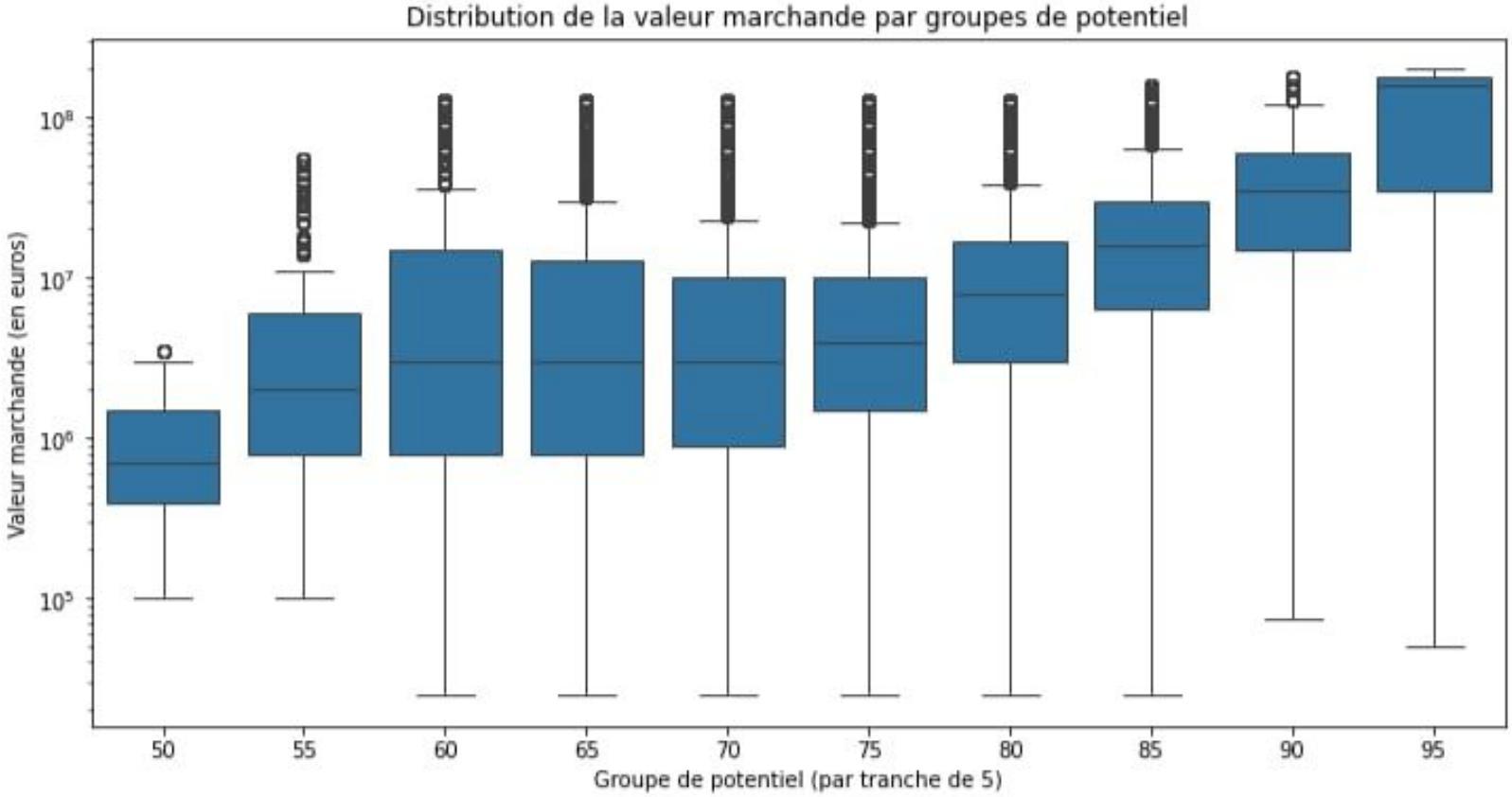
On a essayé d’autres manière de normalisation et on a exclu parfois des colonnes et parfois d’autres mais toujours sans succès.

Dataset FIFA

Nombre de lignes : 17954, nombre de colonnes : 51

On a supprimé les colonnes : **age, positions, value_euro, name, national_team, national_rating, preferred_foot, body_type, release_clause_euro, national_team_position**

On a créé une colonne **nationality_frquency**, où on comptait les occurrences des nationalités dans notre DF.


Distribution de la valeur marchande par groupes de potentiel
Valeur marchande (en euros)
10^8
10^7
10^6
10^5
50
55
60
65
70
75
80
85
90
95
Groupe de potentiel (par tranche de 5)

Jointure DF_FIFA et DF_Players

```python
def match_names(df_players, df_fifa):
    matches = []

    for _, player_row in df_players.iterrows():
        name_player = player_row['name']

        matched_row = df_fifa[df_fifa['name'].str.contains(name_player, case=False, na=False)]

        if not matched_row.empty:
            for _, row in matched_row.iterrows():
                combined_row = {**player_row.to_dict(), **row.to_dict()}
                matches.append(combined_row)

    return pd.DataFrame(matches)


df_merge = match_names(df_players, df_fifa)

print(df_merge.head())
```

Fuites de données

La colonne **highest_market_value_in_eur**, pour éviter qu'elle produise une fuite de données, on l'a actualisé jusqu'à l'année 2023.

```
df_upto_2023 = df_final[df_final['year'] <= 2023]

max_market_value_upto_2023 = (
    df_upto_2023.groupby('player_id')['market_value_in_eur'].transform('max')
)

df_final['highest_market_value_in_eur'] = (
    df_final.merge(
        df_upto_2023.groupby('player_id')['market_value_in_eur'].max(),
        on='player_id',
        how='left',
        suffixes=('', '_max_upto_2023')
    )['market_value_in_eur_max_upto_2023']
)
```

1ère évaluation

Pour KNN : K = 10

R-squared: 0.8411759185405852

Cross-Validation R2 Mean: 0.18617385454466734

Pour Random Forests :

R-squared: 0.6656606892969854

Cross-Validation R2 Mean: 0.6059202999554948

Pour régression linéaire :

R-squared: 0.5290378974437702

Cross-Validation R2 Mean:

0.32287902052156164

Commentaires

Après divers essais, on a pas pu éviter le sur-apprentissage sur le modèles **KNN**, on a décidé de nous focaliser sur **Random Forests,** on ensuite testé avec plusieurs paramètres, nombre de forêt (50 à 200), sur la limitation du nombre de noeuds.

En affichant l'importance des colonnes, on a manipulé ces dernières, parfois on exclut certaines, parfois d'autres, on a utilisé plusieurs manières de normalisation, mais le score ne dépassait pas 0.7

Standardisation des données FIFA avec PCA

```
columns_to_standardize = [
    'height_cm', 'weight_kgs', 'overall_rating', 'potential',
    'crossing', 'finishing', 'heading_accuracy', 'short_passing', 'volleys',
    'dribbling', 'curve', 'freekick_accuracy', 'long_passing', 'ball_control',
    'acceleration', 'sprint_speed', 'agility', 'reactions', 'balance', 'shot_power',
    'jumping', 'stamina', 'strength', 'long_shots', 'aggression', 'interceptions',
    'positioning', 'vision', 'penalties', 'composure', 'marking', 'standing_tackle',
    'sliding_tackle','total_minutes_played', 'minutes_per_goal', 'minutes_per_assist',
    'minutes_per_yellow_card', 'minutes_per_red_card',
    'wage_euro'
]

scaler = StandardScaler()

df_standardized = df_selected.copy()
df_standardized[columns_to_standardize] = scaler.fit_transform(df_standardized[columns_to_standardize])
```

```
n_components = np.argmax(explained_variance_ratio >= 0.95) + 1
print(f"Nombre optimal de composantes pour 95% de variance: {n_components}")

pca_optimal = PCA(n_components=n_components)
pca_transformed_data = pca_optimal.fit_transform(df_standardized[columns_to_standardize])

pca_columns = [f'PC{i+1}' for i in range(n_components)]
pca_df = pd.DataFrame(pca_transformed_data, columns=pca_columns, index=df_selected.index)

df_selected = pd.concat([df_standardized.drop(columns=columns_to_standardize), pca_df], axis=1)

df_selected.head(10)
```

Evaluation

```python
def evaluate_models(df_selected):

    X = df_selected.drop(columns=['market_value_in_eur'])
    y = df_selected['market_value_in_eur']  # Cible

    X.fillna(0, inplace=True)

    train_data = df_selected[df_selected['date_valuation'] > 0]
    test_data = df_selected[df_selected['date_valuation'] == 0]

    X_train = train_data.drop(columns=['market_value_in_eur'])
    y_train = train_data['market_value_in_eur']
    X_test = test_data.drop(columns=['market_value_in_eur'])
    y_test = test_data['market_value_in_eur']

    results = {}

    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)

    y_pred_rf = rf_model.predict(X_test)

    print(f"Nombre de prédictions effectuées : {len(y_pred_rf)}")
    mse_rf = mean_squared_error(y_test, y_pred_rf)
    r2_rf = r2_score(y_test, y_pred_rf)
    cv_scores_rf = cross_val_score(rf_model, X, y, cv=10, scoring='r2')
    results['Random Forest'] = {
        'MSE': mse_rf,
        'R2': r2_rf,
        'Cross-Validation R2 Mean': cv_scores_rf.mean(),
        'Cross-Validation R2 Scores': cv_scores_rf.tolist(),
        'Predictions Count': len(y_pred_rf)
    }
```

Evaluation

```
knn_model = KNeighborsRegressor(n_neighbors=50)
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
mse_knn = mean_squared_error(y_test, y_pred_knn)
r2_knn = r2_score(y_test, y_pred_knn)
cv_scores_knn = cross_val_score(knn_model, X, y, cv=5, scoring='r2')

results['KNN'] = {
    'MSE': mse_knn,
    'R2': r2_knn,
    'Cross-Validation R2 Mean': cv_scores_knn.mean(),
    'Cross-Validation R2 Scores': cv_scores_knn.tolist()
}
```

Résultat Random Forests

```
Nombre de prédictions effectuées : 4123

### Random Forest ###
Mean Squared Error (MSE): 73354764519342.34
R-squared: 0.8967558833495094
Cross-Validation R2 Mean: 0.7019305309714368
Cross-Validation R2 Scores: [0.14692746943539647, 0.857840282704299, 0.6776067293706001, 0.7006630691719276, 0.74752
28208273182, 0.6806512762844199, 0.8709239600741954, 0.5824158117830958, 0.8772099720374864, 0.8775439180256279]
```

Résultat KNN

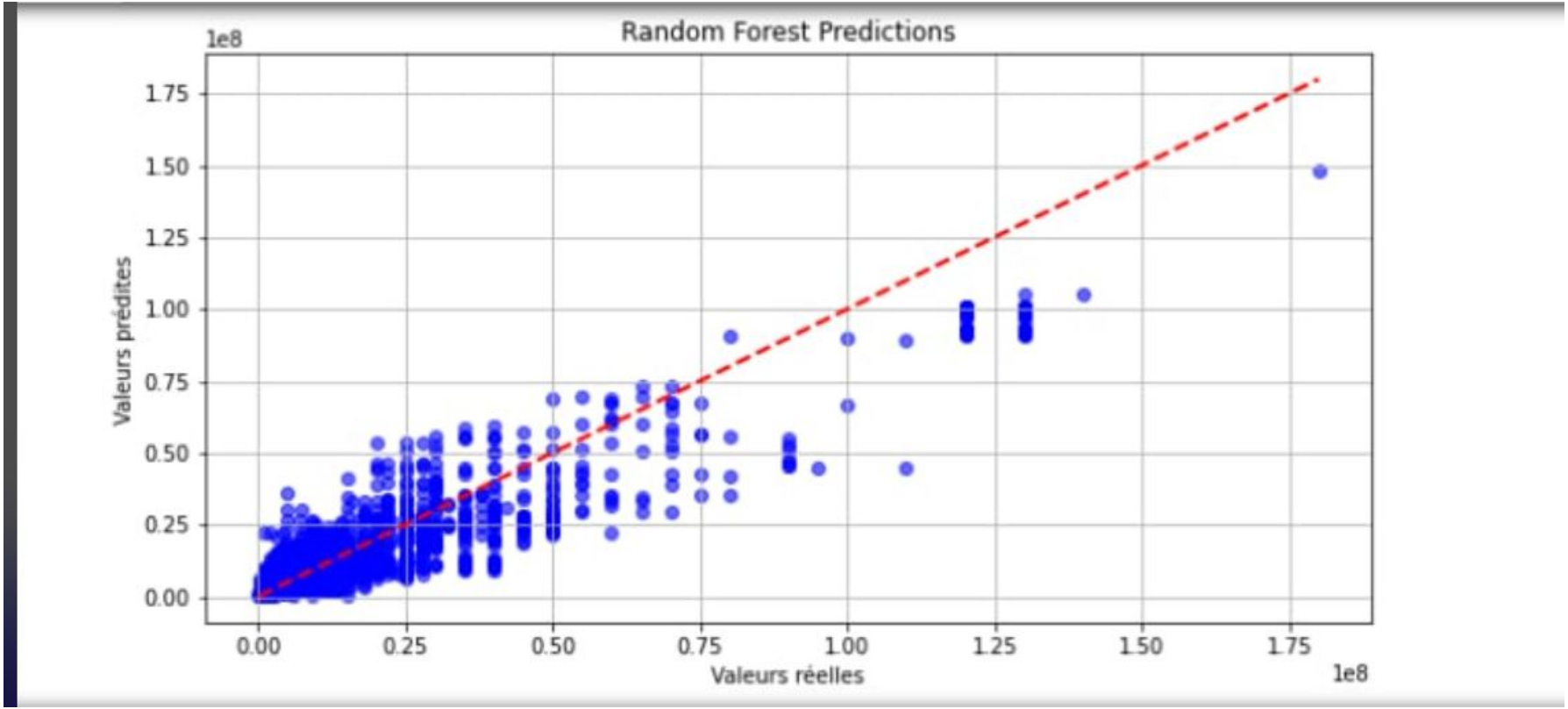
```
### KNN ###
Mean Squared Error (MSE): 255281570943851.53
R-squared: 0.6407006353036848
Cross-Validation R2 Mean: 0.5811775290078744
Cross-Validation R2 Scores: [0.42976899140288927, 0.5383430814378071, 0.630926127069035, 0.7424414188946004, 0.56440
80262350405]
```

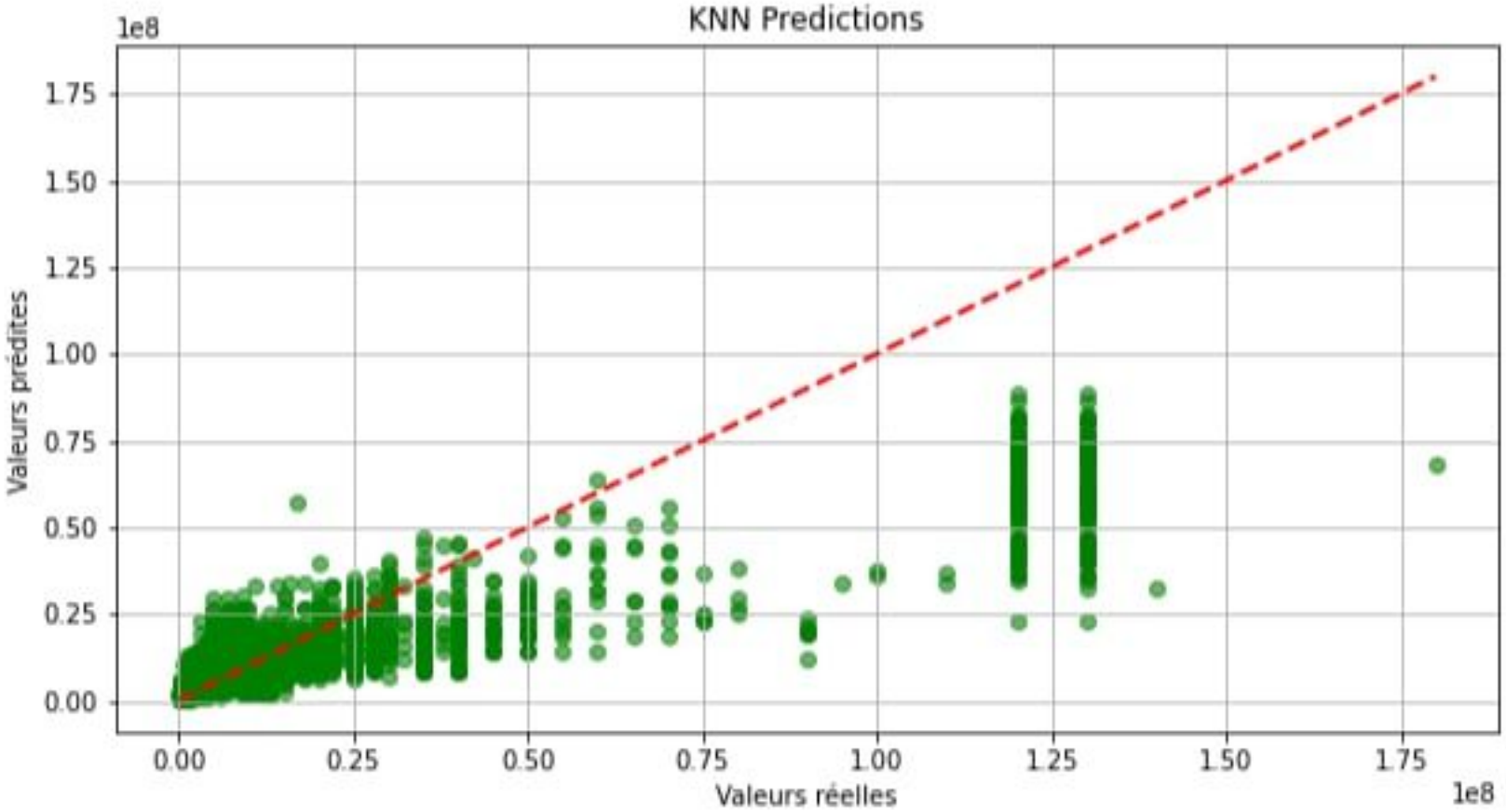
Importance des colonne (RF)

Feature Importance (Random Forest)

	Feature	Importance
0	highest_market_value_in_eur	0.629449
12	date_valuation	0.168535
11	age	0.057246
22	PC10	0.029737
28	PC16	0.020094
13	PC1	0.013427
15	PC3	0.007845
16	PC4	0.006110
1	international_reputation(1-5)	0.005145
17	PC5	0.005065
14	PC2	0.004565
30	PC18	0.004495
27	PC15	0.004031
29	PC17	0.003886
24	PC12	0.003578
21	PC9	0.003513
26	PC14	0.003391
23	PC11	0.003310
31	PC19	0.003269
25	PC13	0.003147
18	PC6	0.003052
20	PC8	0.003002
19	PC7	0.002960
32	PC20	0.002683
7	position_Midfield	0.002462
3	skill_moves(1-5)	0.001029
10	foot_right	0.000979
9	foot_left	0.000921
4	position_Attack	0.000765
5	position_Defender	0.000744
8	foot_both	0.000670
2	weak_foot(1-5)	0.000534
6	position_Goalkeeper	0.000362

Visualisation des valeurs prédites et réelles




KNN Predictions
1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Valeurs prédites
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Valeurs réelles
1e8

Contributions

- **Idée du projet**: Les deux (passionné par le foot)
- **Code**:
  - Partie “observation”: Younes
  - Partie “nettoyage et préparation”: Massine~40%, Younes~60%
  - Partie “normalisation et standardisation”: Massine~40%, Younes~60%
  - Partie “visualisation”: Massine~60%, Younes~40%
  - Partie “training”: Massine~40%, Younes~60%
- **Graphes:** Younes~40%, Massine~60%
- **Slides**: Younes~40%, Massine~60%