Prédiction des blessures majeures des footballeurs.

TAFZA Hanae

Mathématiques (Mathématiques appliquées)
Informatique (Informatique pratique)

2023 -2024

Plan

Introduction

Figure 1: L'apprentissage automatique et le football

L'intelligence artificielle ne cesse guerre à évoluer le domaine sportif, notamment l'entrainement des footballeurs et l'adoption d'une stratégie de jeu.

Les types du Machine Learning

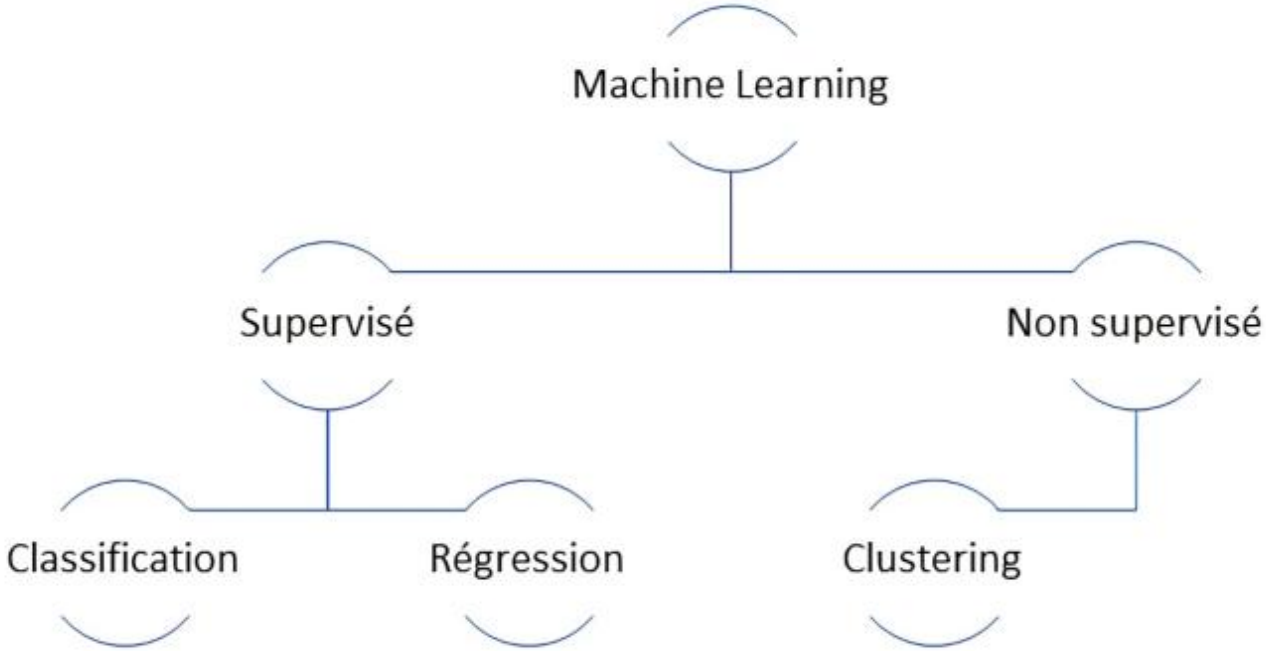

Figure 2: Les différents types de l'apprentissage automatique

Problématique

Comment aider à informer les entraineurs sur les risques potentiels de faire jouer certains joueurs à l'aide de l'apprentissage automatique?

Le modèle XGBoost

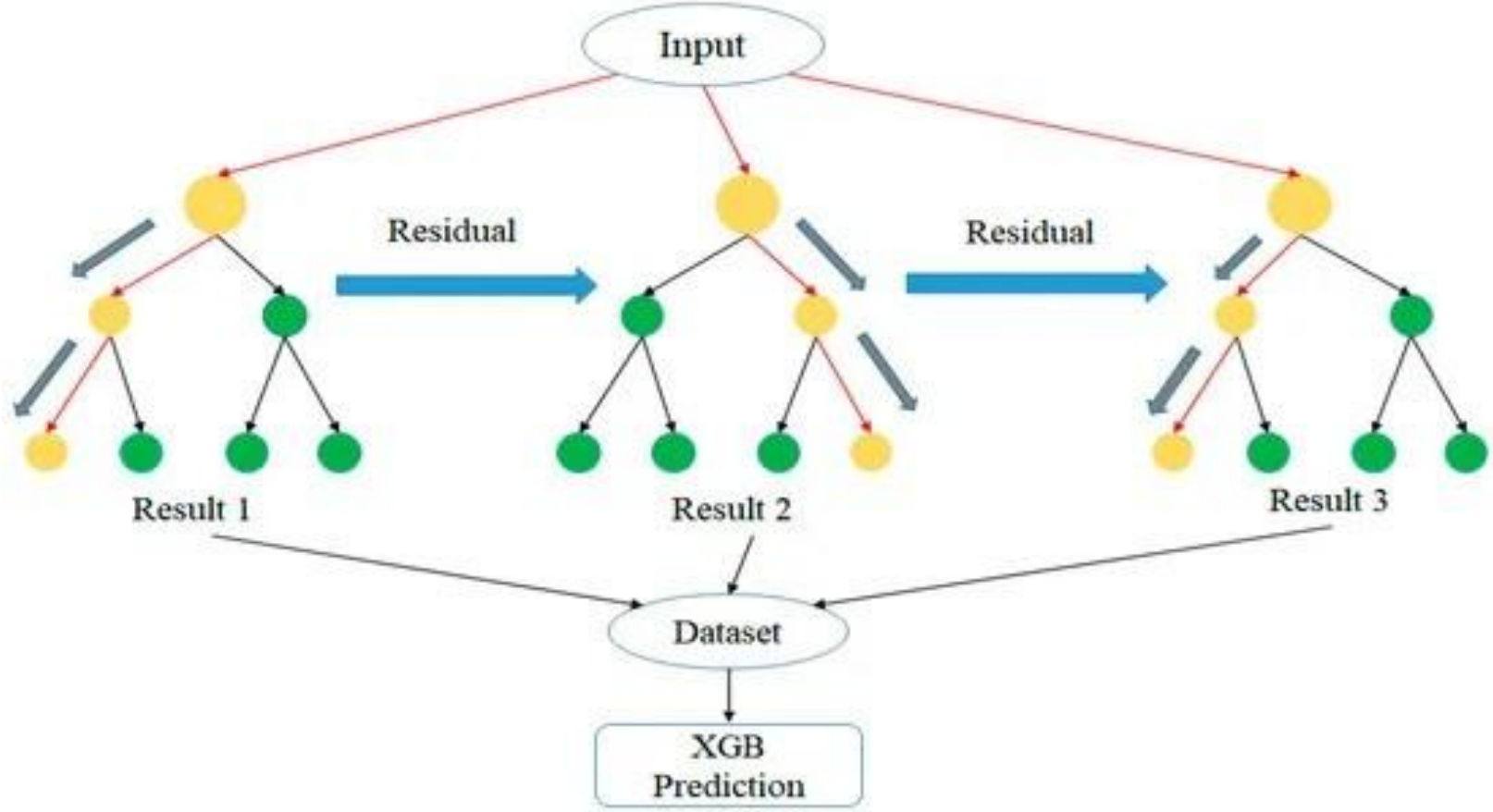

Figure 3: Le fonctionnement du modèle XGBoost

Algorithme de Gradient Boosting

Entrée : Ensemble d'entraînement $\{(x_i, y_i)\}_{i=1}^{n}$, une fonction de perte différentiable $L(y, F(x))$, nombre d'itérations M.

Algorithme :

1. Initialiser le modèle avec une valeur constante :

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. Pour $m = 1$ à M :
   1. Calculer ce que l'on appelle les pseudo-résidus :

$$r_{im} = -\left[\frac{\partial L(y_i, F(x))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{pour } i = 1, \ldots, n.$$

Algorithme de Gradient Boosting

2. Pour $m = 1$ à M (suite) :
   2. Ajuster un apprenant de base (ou un apprenant faible, par exemple un arbre) fermé sous la mise à l'échelle $h_m(x)$ à des pseudo-résidus, c'est-à-dire à l'entraîner à l'aide de l'ensemble d'apprentissage $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Calculer le multiplicateur γ_m en résolvant le problème d'optimisation unidimensionnelle suivant :

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Mettre à jour le modèle :

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Sortie $F_M(x)$.

Fonction de Perte pour Gradient Boosting (Classification)

Pour la classification binaire, la fonction de perte log-loss est définie comme suit :

$$L(y, F(x)) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log(F(x_i)) + (1 - y_i) \log(1 - F(x_i))]$$

où :

- n est le nombre d'exemples,
- y_i est la vraie étiquette de la i-ème observation,
- $F(x_i)$ est la prédiction de la probabilité que l'étiquette soit 1 pour la i-ème observation.

Changement de la Fonction de Perte

La fonction de perte nécessite des changements. Nous allons l'écrire en fonction de log(odds) plutôt que p. log(odds) est défini comme suit :

$$\log\left(\frac{p}{1-p}\right) = \log(\text{odds})$$

$$L(y_i, \log(\text{odds})) = -\frac{1}{n}\sum_{i=1}^{N}\left[y_i \log(\text{odds}) + \log(1 + e^{\log(\text{odds})})\right]$$

$$\frac{\partial L(y_i, \log(\text{odds}))}{\partial \log(\text{odds})} = -y_i + \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

ou

$$\frac{\partial L(y_i, F(x))}{\partial F(x)} = -y_i + p$$

Un équivalent de la fonction de perte

Pour chaque itération m dans l'algorithme de Gradient Boosting, la mise à jour du modèle est approximée comme suit :

$$L(y_i, F_{m-1}(x_i)+\gamma h_m(x_i)) \approx L(y_i, F_{m-1}(x_i))+\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\gamma h_m(x_i)$$

$$+\frac{1}{2}\frac{\partial^2 L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)^2}(\gamma h_m(x_i))^2 \text{ Où :}$$

$$h_m(x_i) = y_i - F_{m-1}(x_i) \quad \text{pour } i = 1, \ldots, n$$

Une partie cruciale de la méthode de Gradient Boosting est la régularisation par rétrécissement, qui modifie la règle de mise à jour comme suit :

$$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x), \quad 0 < \nu \leq 1,$$

où ν est le taux d'apprentissage.

Préparation des données

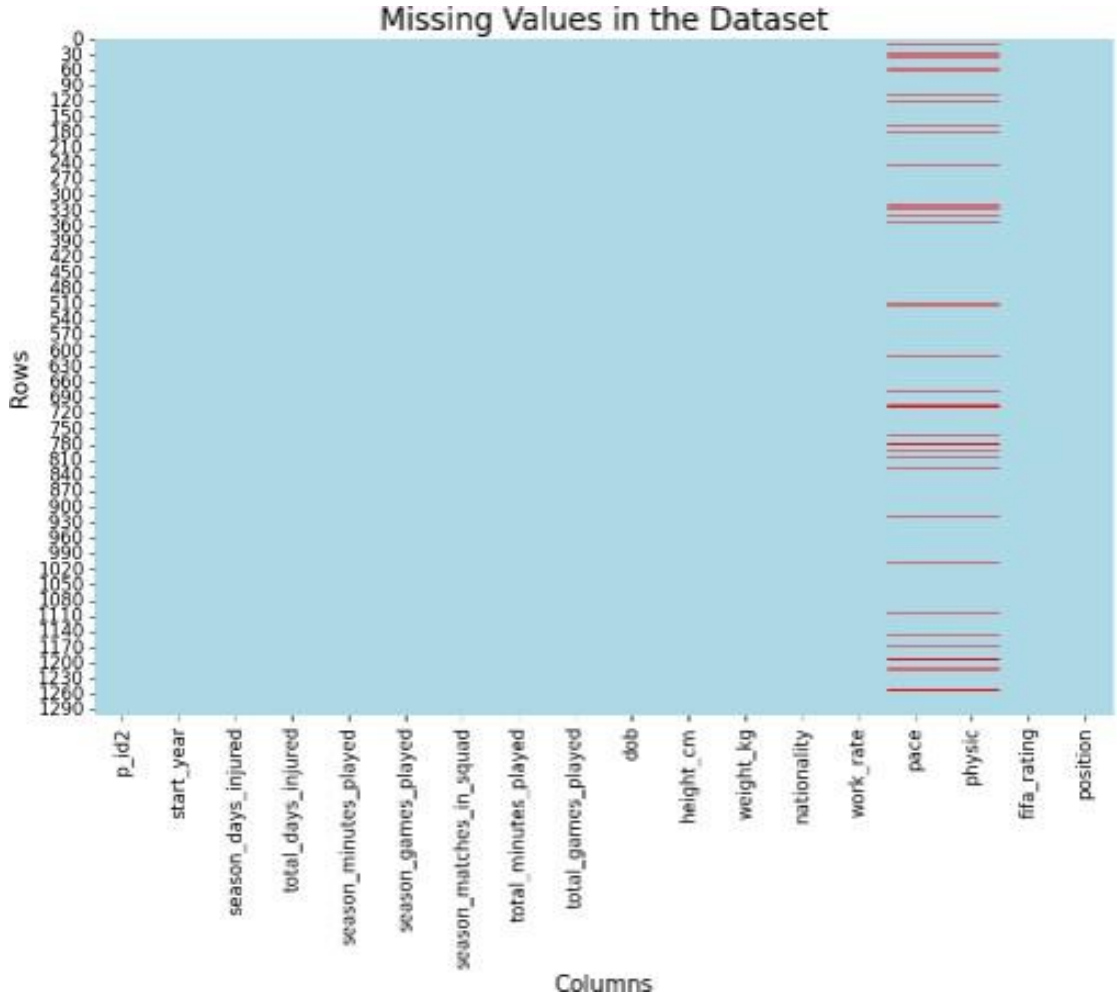

Figure 4: Les valeurs manquantes du Dataset

	p_id2	start_year	season_days_injured	total_days_injured	season_minutes_played	season_games_played	season_matches_in_squad	total_minutes_played	total_games_played	dob
0	aaronconnolly	2019	13	161	1312.0	24	28	2148.0	41	2000-01-28
1	aaronconnolly	2020	71	161	836.0	17	28	2148.0	41	2000-01-28
2	aaroncresswell	2016	95	226	2247.0	26	27	13368.0	149	1989-12-15
3	aaroncresswell	2018	87	226	1680.0	20	27	13368.0	149	1989-12-15
4	aaroncresswell	2019	35	226	2870.0	31	31	13368.0	149	1989-12-15

height_cm	weight_kg	nationality	work_rate	pace	physic	fifa_rating	position
175.333333	75.666667	Republic of Ireland	Medium/Low	72.333333	58.0	63.000000	Forward
175.333333	75.666667	Republic of Ireland	Medium/Low	72.333333	58.0	63.000000	Forward
171.666667	66.000000	England	High/Medium	74.333333	67.0	75.333333	Defender
171.666667	66.000000	England	High/Medium	74.333333	67.0	75.333333	Defender
171.666667	66.000000	England	High/Medium	74.333333	67.0	75.333333	Defender

Figure 5: Les cinqs premiers lignes de l'ensmeble de données

Source du Dataset:https://www.kaggle.com/

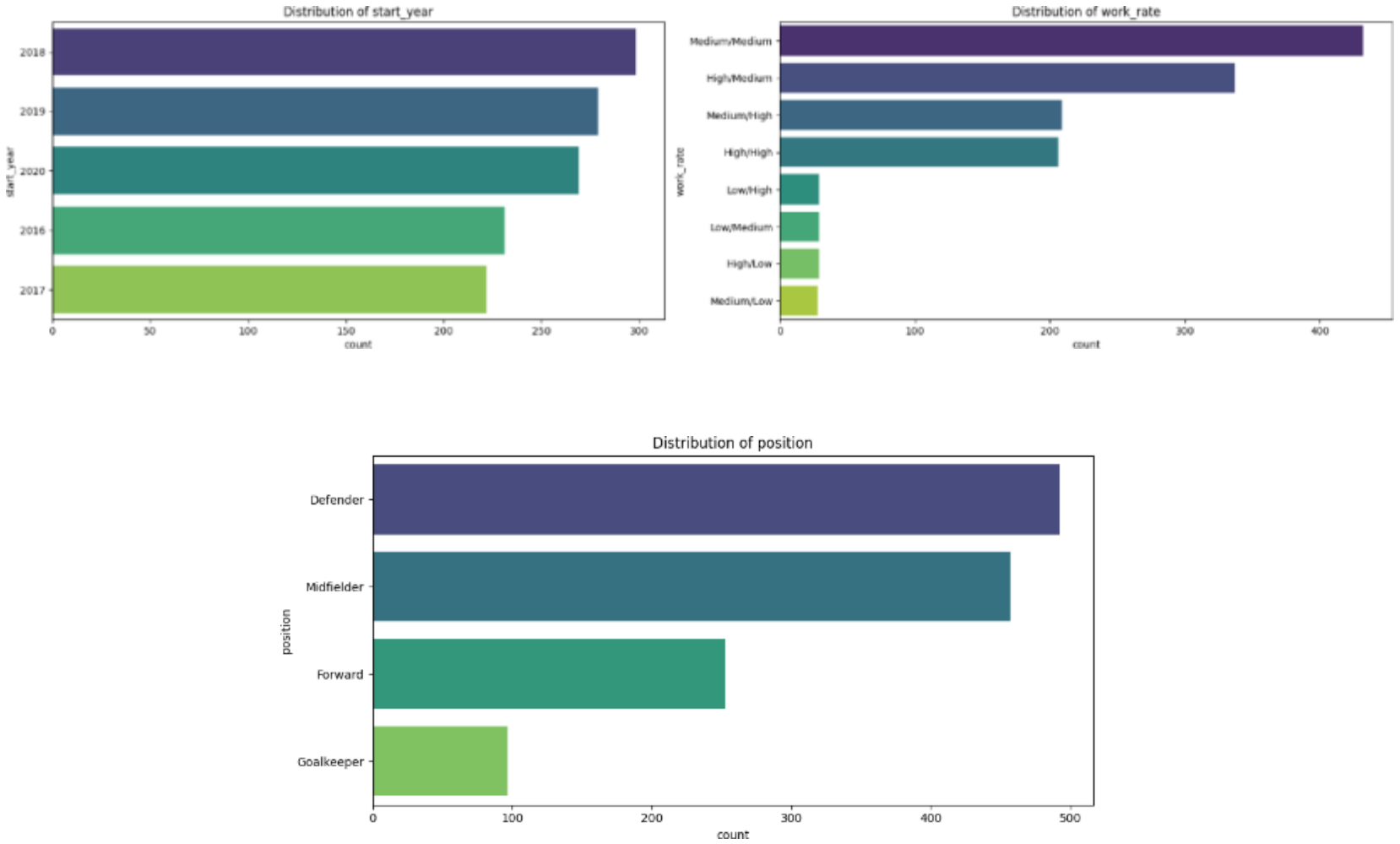

Figure 6: Les distributions des caractéristques catégorielles

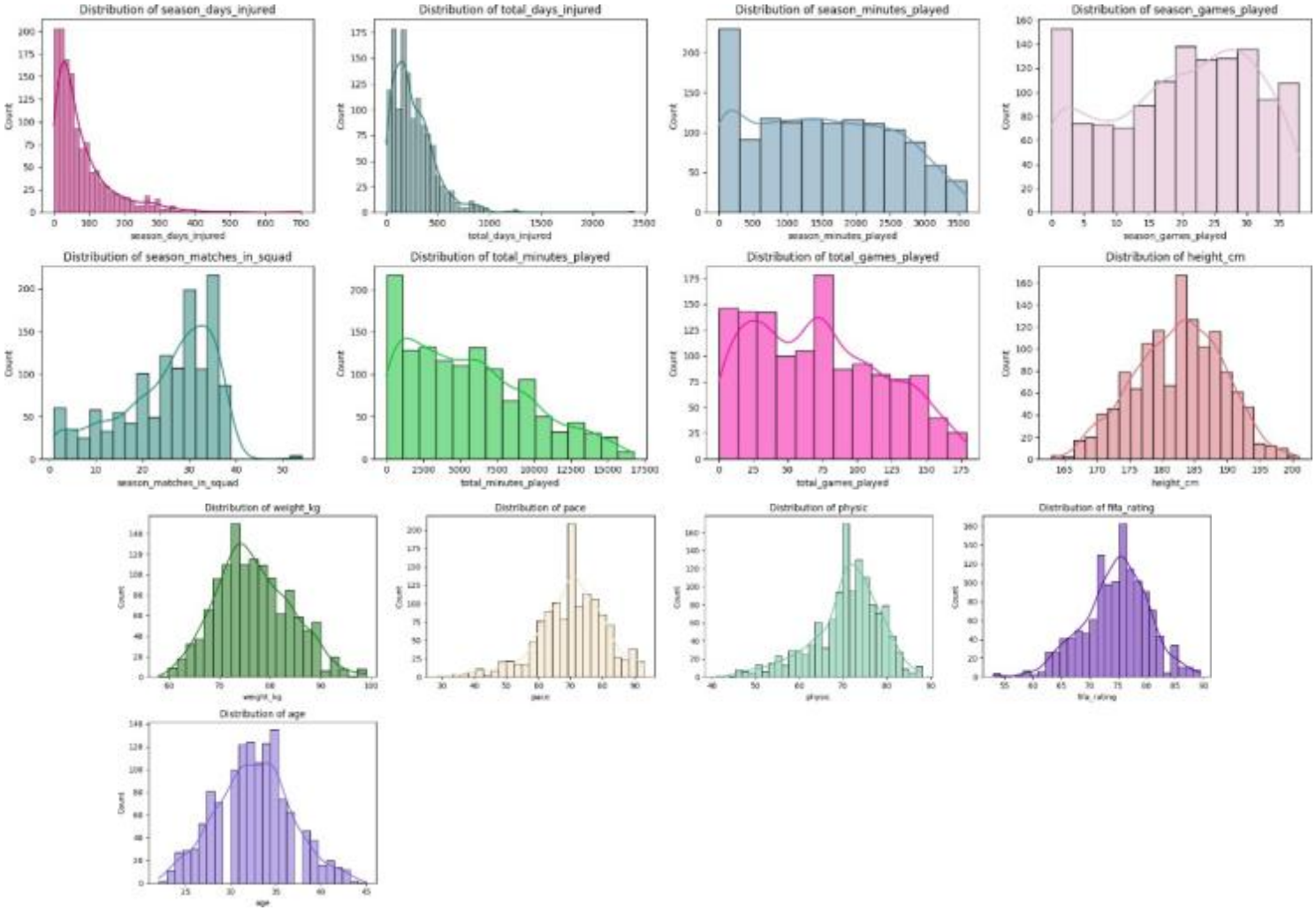

Figure 7: Les distributions des caractéristques numériques

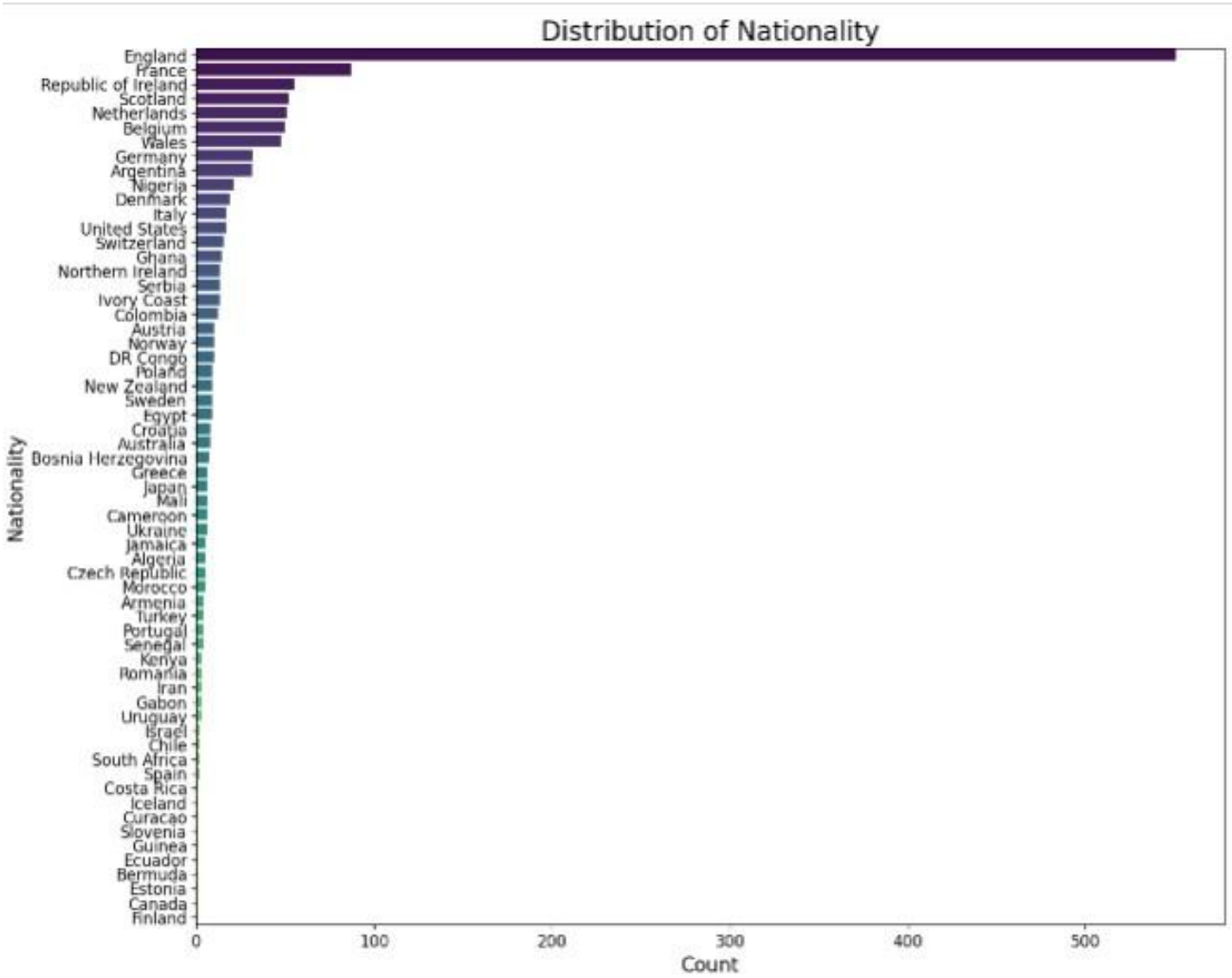

Figure 8: Les distributions des Nationalités

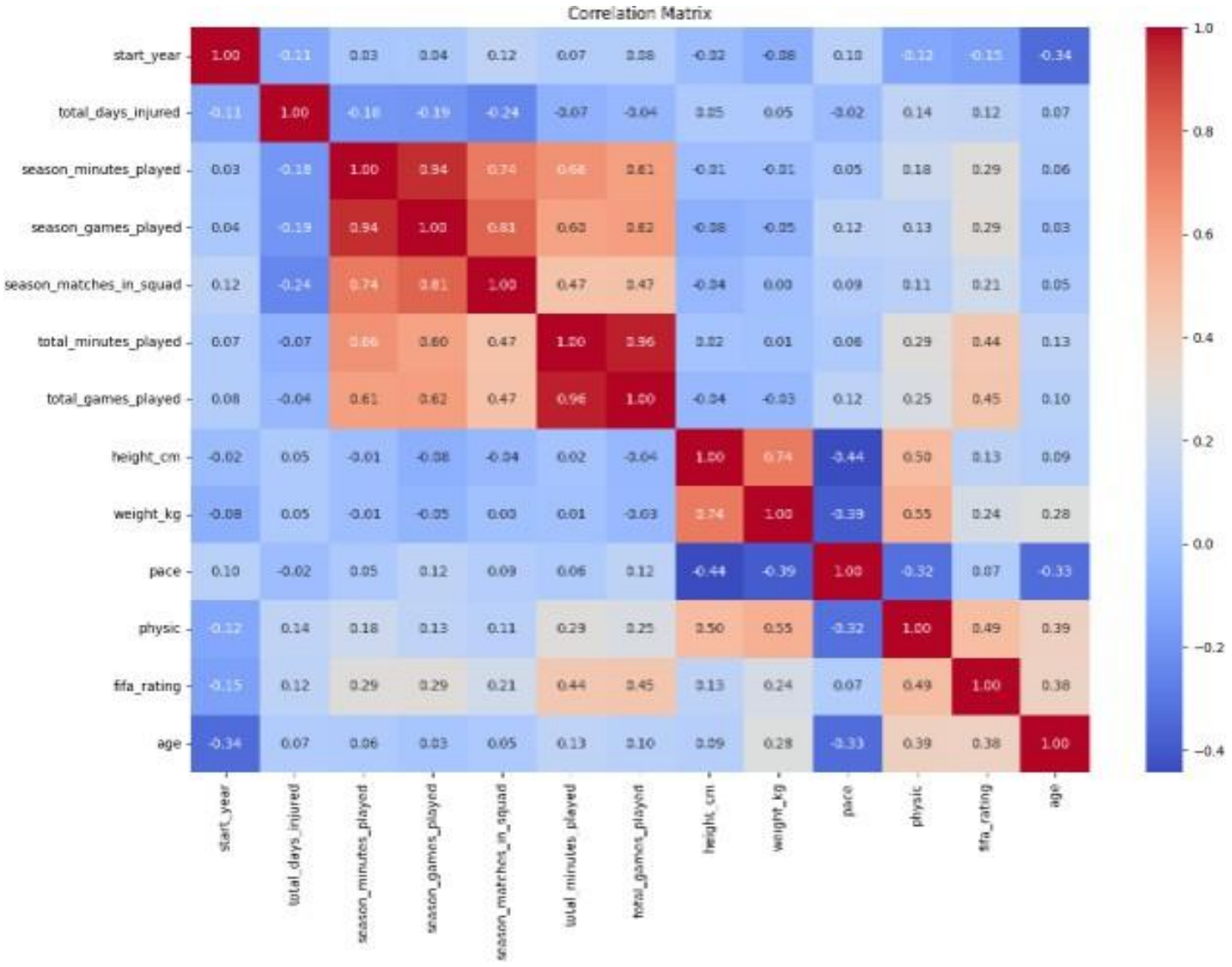

Figure 9: La matrice de corrélation

Modélisation et optimisation des hyperparamètres

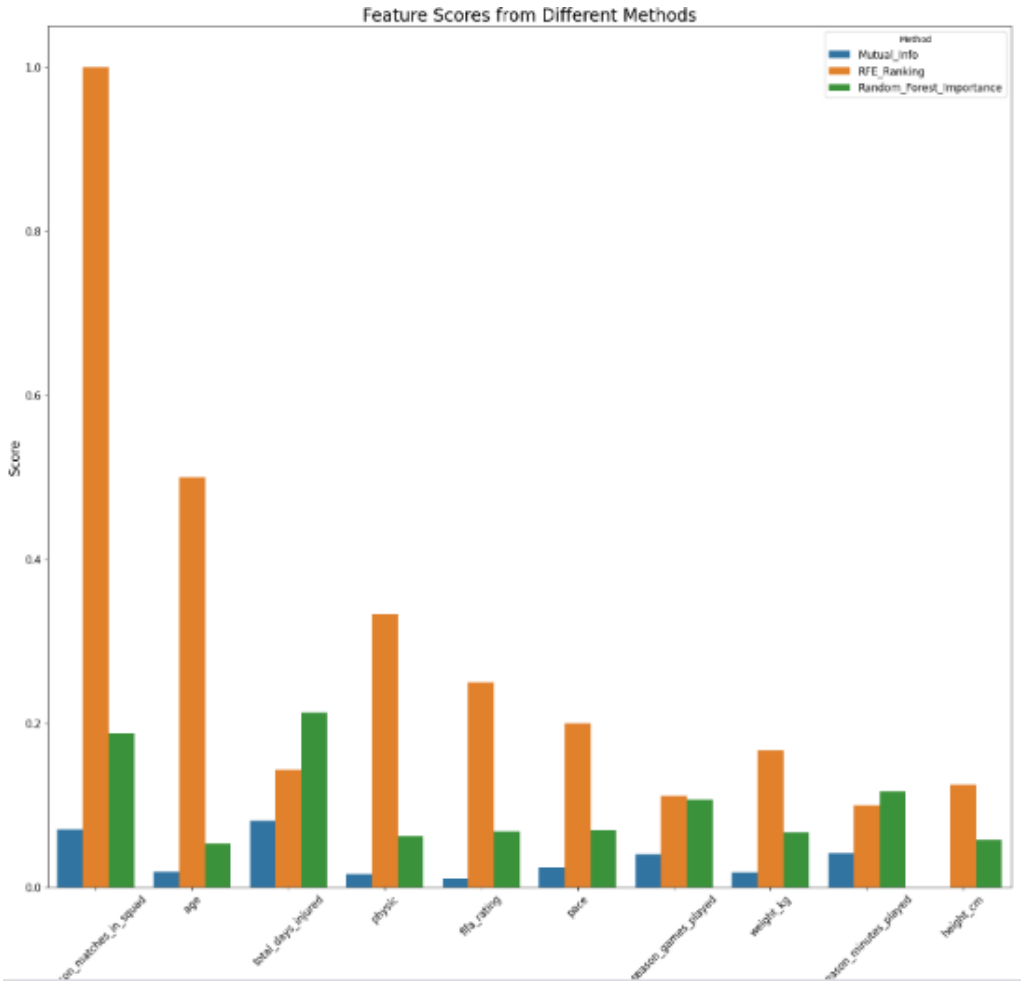

Figure 10: Scores des caractéristiques selon différentes méthodes

Les meilleurs paramétres obtenus: {'colsample_bytree': 0.5353970008227678, 'learning_rate': 0.06879485872574555, 'max_depth': 3, 'n_estimators': 89, 'subsample': 0.8378115394712806}

	précision	rappel	f1-score	support
0	0.67	0.54	0.60	57
1	0.88	0.93	0.90	203
Précision			0.84	260
La moyenne macro	0.78	0.73	0.75	260
La moyenne pondérée	0.83	0.84	0.84	260

Table 1: La classe 0 correspond aux joueurs non aptes à avoir des blessures majeures, tandis que la classe 1 correspond aux joueurs aptes à avoir des blessures majeures.

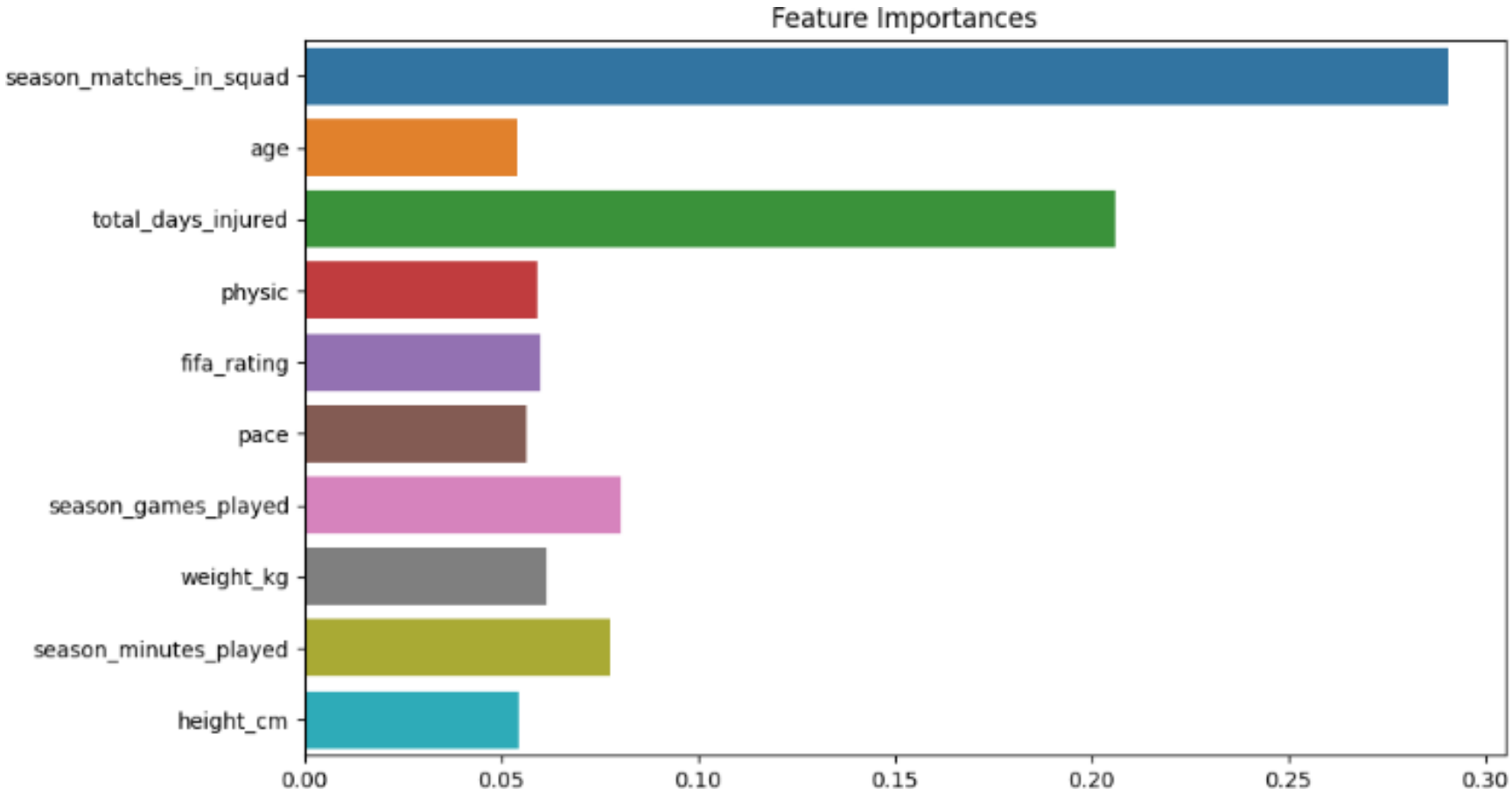

Figure 11: L'importance des caractéristiques

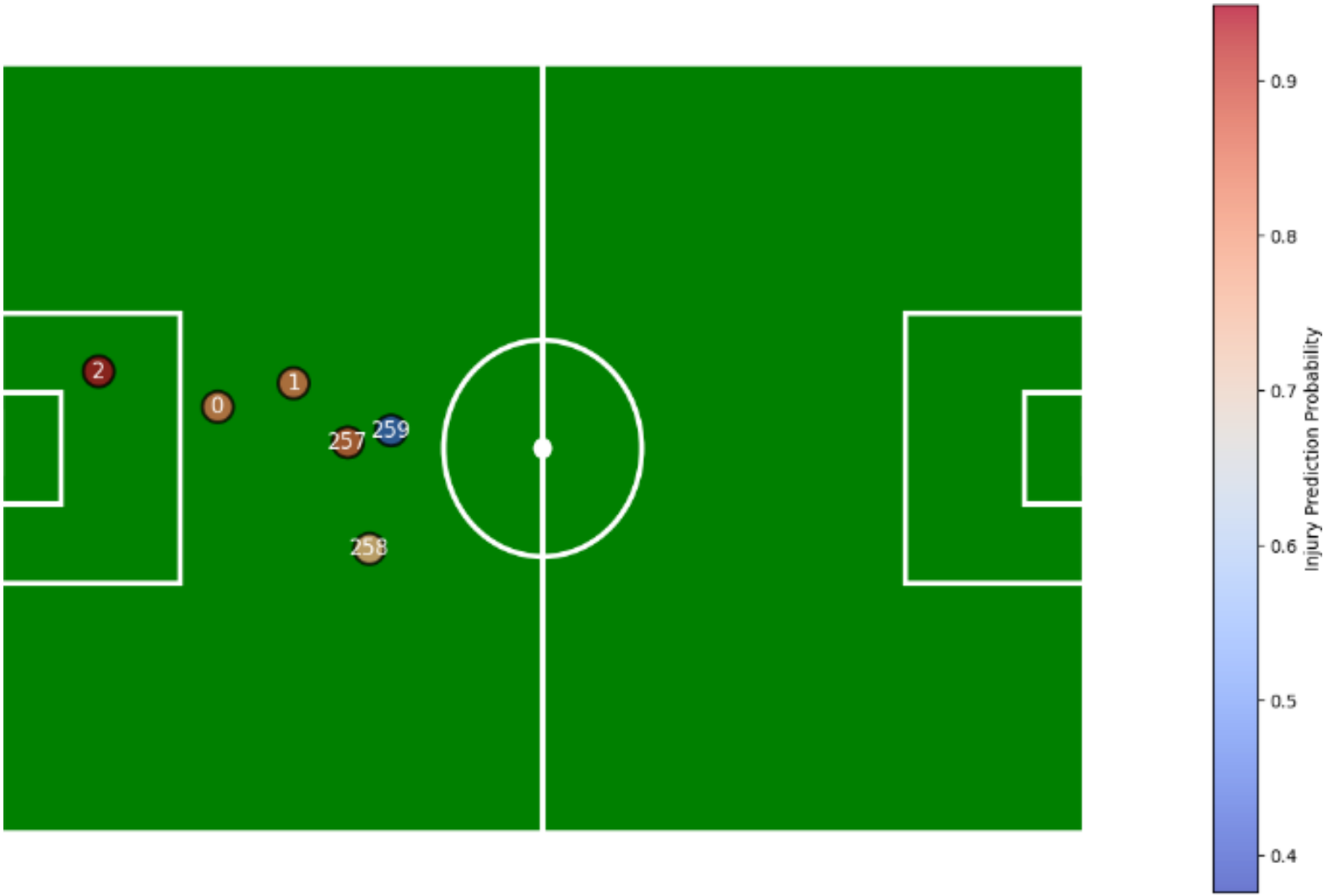

Figure 12: Les probabilités des blessures majeures visualisées sur le terrain

Conclusion

- Ce projet a utilisé le modèle XGBoost pour prédire si un joueur de football subira une blessure majeure ou non.
- Les caractéristiques utilisées dans le modèle incluent des données sur les matchs joués, l'âge, les jours blessés, la condition physique, la note FIFA, la vitesse, le poids, les minutes jouées et la taille.
- Le modèle a été entraîné et optimisé en utilisant RandomizedSearchCV pour trouver les meilleurs hyperparamètres.
- Les résultats montrent une bonne précision et une capacité de prédiction fiable.

Annexe

```
# Supprimer les avertissements
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)

# Tracer la distribution de la nationalité sous forme de diagramme à barres horizontales
plt.figure(figsize=(14, 7))
sns.countplot(y=df['nationality'], order=df['nationality'].value_counts().index, palette='viridis')
plt.title('Distribution of Nationality', fontsize=20)
plt.xlabel('Count', fontsize=15)
plt.ylabel('Nationality', fontsize=15)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()

# Creer la variable cible pour la classification binaire
injury_threshold = 28
df['target_injury'] = (df['season_days_injured'] > injury_threshold).astype(int)

# Définir les caractéristiques et la variable cible pour la modélisation
features = ['start_year', 'total_days_injured', 'season_minutes_played', 'season_games_played',
            'season_matches_in_squad', 'total_minutes_played', 'total_games_played', 'height_cm',
            'weight_kg', 'pace', 'physic', 'fifa_rating', 'age']

X = df[features]
y = df['target_injury']

# S'assurer que toutes les colonnes de caractéristiques sont numériques avant la conversion
print("Checking for NaNs before converting to numeric:")
print(X.isnull().sum())

# Supprimer les colonnes avec trop de valeurs NaN (plus de 50% de NaN)
threshold = 0.5
X = X.loc[:, X.isnull().mean() < threshold]


X = X.dropna()
y = y.loc[X.index]

# Vérifier les formes après avoir supprimé les NaN.
print("Shape of X after dropping NaNs:", X.shape)
print("Shape of y after dropping NaNs:", y.shape)

# Convertir les colonnes restantes en types de données numériques
X = X.apply(pd.to_numeric, errors='coerce')

# Vérifier s'il reste des NaN après la conversion
print("\nMissing values in features after conversion to numeric:")
print(X.isnull().sum())

# Supprimer à nouveau toutes les valeurs NaN restantes après la conversion.
X = X.dropna()
y = y.loc[X.index]

print("Shape of X after handling NaNs:", X.shape)
print("Shape of y after handling NaNs:", y.shape)

# Afficher la forme et les premières lignes du jeu de données nettoyé
print("Shape of the dataset after handling missing values:", df.shape)
print("\nFirst few rows of the cleaned dataset:")
print(df.head())

# Visualiser la matrice de corrélation
plt.figure(figsize=(15, 10))
correlation_matrix = X.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# Afficher la corrélation des caractéristiques avec la variable cible
correlation_with_target = df[features + ['target_injury']].corr()['target_injury'].sort_values
print("\nCorrelation of features with target:")
print(correlation_with_target)

# Sélectionner les caractéristiques principales basées sur la corrélation avec la variable cib
top_features = correlation_with_target.head(10).index.tolist()
top_features.remove('target_injury')
print("\nTop features based on correlation with target:")
print(top_features)
```

Merci de votre attention