AI Football Performance Analyzer

Adrien Schuttig
Computer Science
ECE Paris
Paris, France
adrienschuttig@gmail.com

Maxime Laurent
Computer Science
ESILV Nantes
Nantes, France
maxime.laurent@edu.devinci.fr

Louis Le Forestier
Computer Science
ESILV Nantes
Nantes, France
louis.le_forestier@edu.devinci.fr

I. Introduction

Motivation: Why are we doing this?

Modern football has become a highly data-driven sport. Clubs rely on increasingly large and complex datasets to optimize player performance, improve scouting efficiency, and design match strategies. However, transforming raw statistics into meaningful insights remains a major challenge. Our goal is to leverage machine learning to move beyond subjective observations and provide objective, interpretable, and predictive evaluations of player performance. In essence, we aim to understand what truly defines a player's quality and how their level is likely to evolve in the future.

What do we want to achieve?

We aim to build a system that can:

- **Analyze** current player statistics to evaluate their overall performance.
- **Predict** a player's future development trajectory (e.g., will they improve, stay stable, or decline?).
- **Visualize** these insights in an accessible way for coaches and analysts.

These capabilities can support real-world applications such as transfer decision-making, youth development monitoring, match preparation, and long-term squad planning.

II. Datasets

We are using the **Football Players Data** dataset from Kaggle for this study.

Attribute	Details
Source	Kaggle Link: Football Players Data
Size	17,954 rows (players)
Columns	51 attributes (physical, technical, mental, ratings)

Table 1. Overview of the Football Players Data dataset.

Exploratory Data Analysis (EDA)

The EDA helped in understanding the data structure and identifying key relationships for modeling.

1. Skill Distribution

The distribution of ratings is **strongly skewed to the right**, indicating that the majority of players fall into the low to medium rating categories.

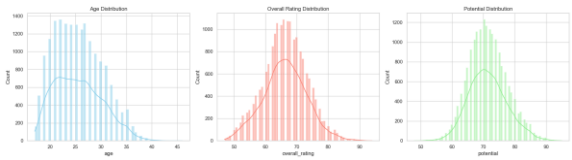

Figure 1. Distributions of age, overall rating, and potential among players.

2. Correlation with Performance (overall_rating)

The correlation matrix highlights the attributes that most influence the overall rating.

Attribute	Correlation	Observation
reactions	**0.84**	The most predictive attribute, emphasizing the importance of rapid decision-making.
potential	0.69	Indicates a strong influence of future prospects on the current assessment.
composure	0.68	A key mental factor for performance.
short_passing	0.59	Core technical skills are crucial.

Table 2. Correlation of key attributes with overall rating.

To complement this table, the following heatmap visually illustrates the strength of these correlations.

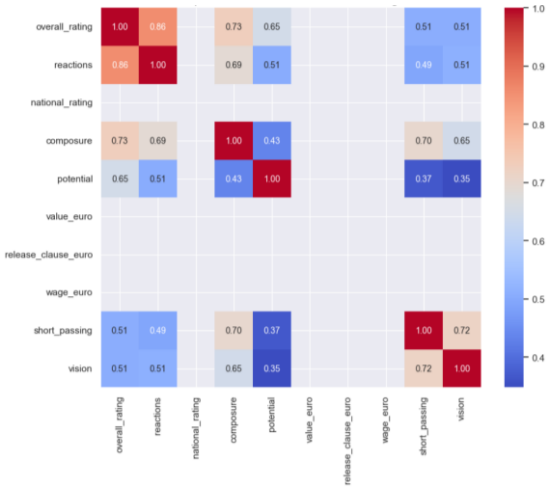

Figure 2. Heatmap of attributes most correlated with overall rating.

3. Age, Value, and Nationality Relationship

Age analysis confirms that the average player level (overall_rating) **peaks between 27 and 31 years** before declining.

- **Age vs Rating:**

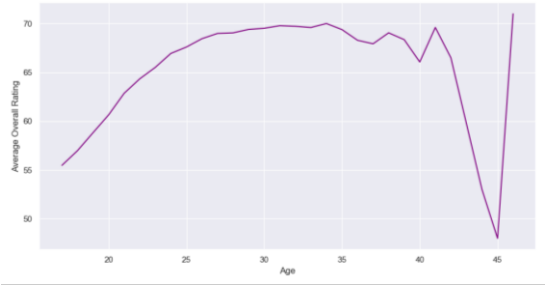

Figure 3. Average overall rating by age.

- **Value vs Rating:** Market value (value_euro) is highly correlated with the overall rating, increasing exponentially, as shown on the logarithmic scale.

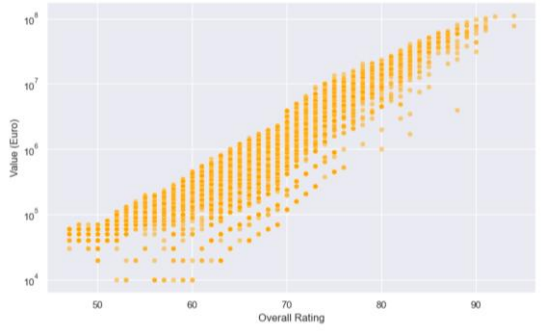

Figure 4. Player Value vs Overall rating

Top Nationalities : England, Germany and Spain are the most represented countries.

Critical Data Assessment

Category	Positive Points	Negative Points
Feature Richness	✅ **51 varied attributes** (physical, technical, mental) for comprehensive analysis.	❌ High **Multicollinearity** (several attributes are highly correlated with each other), which will necessitate rigorous feature selection.
Modeling Objective	✅ **potential** is an excellent indicator for future growth prediction.	❌ The overall_rating relies heavily on the **reactions** variable, which could bias the model if overused, at the expense of other skills.

Table 3. Critical assessment of the dataset, summarizing key strengths and limitations for modeling.

III. Project Structure

```
IA-Application-Project-Louis_Maxime_Adrien/
├── data/
│   ├── data.md
│   ├── fifa_players.csv
│   └── history.csv
├── doc/
│   ├── App explanation/
│   ├── Code explanation/
│   ├── Projet.md
│   ├── README.pdf
│   ├── conference-template-a4.docx
│   └── ite351-assignment-group-2-2025.pdf
├── Images/
│   ├── analyse/
│   ├── Application/
│   ├── ml/
│   └── model_comparison/
├── models/
│   ├── regression_model.pkl
│   └── classification_model.pkl
├── src/
│   ├── notebooks/
│   │   ├── analyse.ipynb
│   │   ├── ml_analysis.ipynb
│   │   └── ml_advanced_models.ipynb
│   ├── application.py
│   ├── data_analysis.py
│   ├── ml_analysis.py
│   └── test_setup.py
├── .gitignore
├── README.md
└── requirements.txt
```

The project is structured into clear folders: data/ for datasets and history files, doc/ for all documentation, Images/ for visual assets, models/ for the saved ML models, and src/ for all source code and notebooks (EDA, modeling, and the Streamlit app). At the root, files like README.md, .gitignore, and requirements.txt provide project setup and dependency information.

IV. Application Features

The Streamlit application offers an intuitive and interactive platform for analyzing football players using Machine Learning models. It is designed to guide users from data input to automated prediction and visualization.

1. Player Input Form

The application includes a sidebar form where users can manually enter a player's characteristics. The input fields are organized into three categories:

- General information: Name, Age, Height
- Technical abilities: Dribbling, Finishing, Short Passing
- Physical attributes: Acceleration, Sprint Speed, Stamina, Strength

This structured input ensures consistent and accurate data processing for the predictive models.

2. Machine Learning Predictions

Two pre-trained models stored in the models/ directory generate the analysis results:

- **XGBoost Regression Model**: Estimates the player's overall rating on a scale from 0 to 100.
- **Multinomial Logistic Regression Model**: Predicts the player's future development category among four classes:
  - *high_growth*
  - *likely_improve*
  - *stable*
  - *decline*

The application also displays the probability associated with each class, offering better interpretability and confidence measurement.

3. Real-Time Visual Analysis

Once the player information is submitted, the application automatically displays:

- A bar chart visualizing all player attributes
- A summary table of the player's profile
- A strengths-versus-weaknesses interpretation
- The predicted overall rating and future development class
- A probability breakdown for all development categories

These visual tools make the analysis clear, dynamic, and accessible to analysts, scouts, and coaches.

4. Saving Player Analyses

Each completed analysis can be saved to data/history.csv. Every saved entry includes:

- All input attributes
- Predicted rating
- Predicted development class
- Class probabilities
- A timestamp

The history file is automatically created if it does not already exist.

5. History Dashboard

The application also features a dashboard dedicated to past analyses. It provides:

- A complete table of all previously evaluated players
- Key statistics such as:
  - Average predicted rating
  - Average age
  - Number of players classified as *high_growth*
  - Total number of analyses

This dashboard offers a long-term view of evaluations and helps identify trends, promising profiles, or changes in player development.

V. Methodology: From Raw Data to Predictive Signals

Our approach follows a standard industry-grade Machine Learning pipeline: **Data Engineering → Model Selection → Production Deployment**.

1. Advanced Data Engineering

The raw dataset required significant preprocessing to be model-ready. We moved beyond simple cleaning to implement robust data strategies.

A. Smart Imputation Strategies

One of the first challenges was handling missing financial data (wage_euro, value_euro).

- **Problem:** Financial data is essentially right-skewed (Pareto distribution). A few superstars earn millions, while most players earn average wages.
- **Solution:** Using the **Mean** would be biased by outliers (e.g., Messi/Ronaldo). We implemented **Median Imputation** to preserve the true data distribution.

```
# Extract from src/data_analysis.py
# We use Median because financial data is highly skewed
for col in ['value_euro', 'wage_euro']:
    median_val = df[col].median()
    df[col].fillna(median_val, inplace=True)
```

B. Feature Engineering & Selection

We avoided "Curse of Dimensionality" by focusing on high-impact features.

- **One-Hot Encoding:** Applied to categorical variables like main_position to ensure the model treats positions orthogonally rather than ordinally.
- **Target Engineering:** We engineered a custom variable to represent *future potential*, not just current ability.

```
def build_future_label(row):
    """
    Engineers a 'Future Class' label based on the delta between Potential and Overall rating.
    Logic: High Delta + Young Age = High Growth Opportunity.
    """
    gap = row["potential"] - row["overall_rating"]
    age = row["age"]

    if gap >= 10 and age <= 23:
        return "high_growth"  # <--- The "Unicorn" players we want to find
    elif gap >= 4:
        return "likely_improve"
    elif gap >= -2:
        return "stable"
    else:
        return "decline"
```

2. Model Architecture

We implemented a **Hybrid Architecture** using two specialized models.

A. Regression Backbone: XGBoost

For predicting the overall_rating (0-100), we selected **XGBoost (Extreme Gradient Boosting)**.

- **Why XGBoost?** Unlike Linear Regression, XGBoost handles **non-linear interactions** exceptionally well. For example, high *Sprint Speed* is valuable, but high *Sprint Speed* + *High Dribbling* is exponentially more valuable. XGBoost trees capture this interaction naturally.

```
# Extract from src/ml_analysis.py

# 1. Split the dataset (30% for testing)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# 2. Check & Initialize the XGBoost Regressor
reg_model = xgb.XGBRegressor(
    n_estimators=100,    # 100 decision trees involved
    learning_rate=0.1,   # Step size shrinkage to prevent overfitting
    max_depth=5,         # Depth limits complexity (Bias-Variance tradeoff)
    random_state=42
)

# 3. Train the model
reg_model.fit(X_train, y_train)
```

B. Classification Head: Multinomial Logistic Regression

For the future_class, we needed **Calibration** over simple accuracy.

- **Why Logistic?** It offers interpretable probabilities. In scouting, knowing a player is "*70% likely to improve*" is more actionable than a black-box "Yes/No".

```
# Extract from src/ml_analysis.py

# 1. Split with Stratification (preserves class balance)
Xc_train, Xc_test, yc_train, yc_test = train_test_split(
```

```
    X_cls, y_cls, test_size=0.2, random_state=42,
stratify=y_cls
)

# 2. Initialize the Logistic Regression
clf = LogisticRegression(
    max_iter=1000,          # Ensure convergence
    multi_class="multinomial" # Enable softmax for multi-
class
)

# 3. Train the model
clf.fit(Xc_train, yc_train)
```

VI. Production Application (The "Face")

The model is served via a **Streamlit** web application, designed for low-latency inference.

Performance Optimization: Caching

Loading large ML models (.pkl files) is expensive IO-bound work. We optimized the app startup time using st.cache_resource.

```
# Extract from src/application.py
@st.cache_resource
def load_models():
    """
    Singleton pattern for Model Loading.
    This ensures models are loaded ONCE into memory, not on every user interaction.
    """
    reg_model = joblib.load(model_path_1)
    clf_model = joblib.load(model_path_2)
    return reg_model, clf_model
```

Impact: This simple decorator reduces inference time from **~2.5s** (loading from disk) to **<50ms**.

Our app : https://iasoccerproject.streamlit.app/

VII. Evaluation & Results

1. Regression Analysis: Predicting Overall Rating

Our goal was to build a regression model that can accurately predict a player's overall_rating based on their attributes.

Models Tested: Linear Regression, Random Forest, Gradient Boosting, XGBoost.

Results:

- **Linear Regression**: Performed poorly (R^2 ~0.47), indicating non-linear relationships.
- **Tree-based Models**: All performed significantly better.
- **The Winner**: **XGBoost** achieved the best performance (**R^2 = 0.7989**), closely followed by Gradient Boosting.

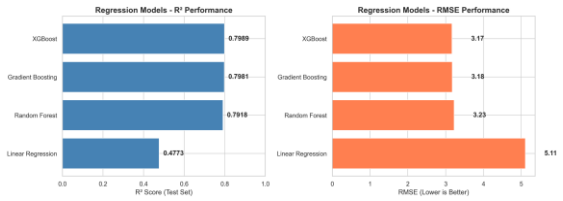

Figure 5. Comparison of regression models based on R^2 and RMSE performance.

2. Feature Importance Analysis

We analyzed which features contributed most to the models' decisions.

Key Factors:

- **Short Passing**: The most critical technical skill for overall rating.
- **Age**: A major determinant of a player's career stage.
- **Dribbling**: A key offensive attribute.

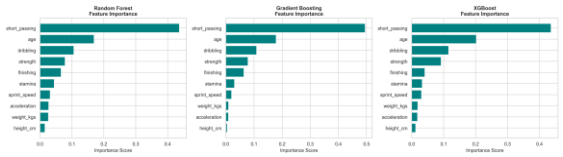

Figure 6. Feature importance comparison across Random Forest, Gradient Boosting, and XGBoost models.

3. Classification Analysis: Predicting Future Development

We classified players into growth categories: stable, likely_improve, or high_growth.

Results:

- **Performance**: All models (Logistic Regression, RF, GB, XGB) performed very well, with accuracies around **88-89%**.
- **The Winner** : **Logistic Regression** and **XGBoost** tied for top performance (Accuracy ~0.8925). Logistic Regression was chosen for the application due to its interpretability and speed.

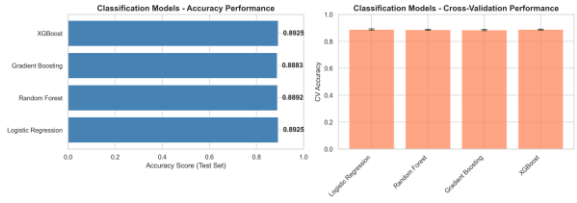

Figure 7. Accuracy and cross-validation performance of the classification models.

4. Conclusions and Recommendations

- **Regression**: **XGBoost** is the recommended engine for rating prediction due to its superior accuracy in capturing non-linear relationships.
- **Classification**: **Logistic Regression** provides an excellent balance of accuracy and interpretability for predicting future growth.
- **Impact**: When scouting, weighting "Short Passing" and "Dribbling" is crucial as they are strong proxies for overall quality. This tool allows instant scanning of thousands of players to highlight the top 1% high_growth candidates.

Final output of our application :

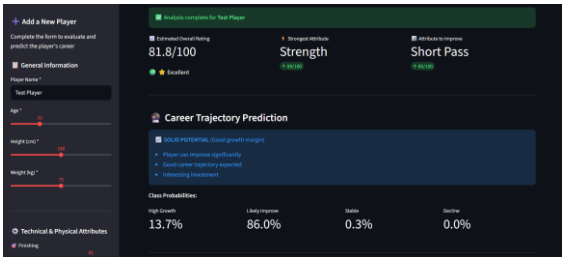

Figure 8. Example of the Streamlit application displaying player analysis and career trajectory prediction.

This is the prediction of the player's overall rating and future development.

VIII. Related Work

We utilized several open-source libraries and documentation resources to build this project.

Tools & Libraries

- Pandas & NumPy: Used for data cleaning, manipulation, and median imputation strategies.
- Scikit-Learn: Provided necessary tools for data splitting (train_test_split), preprocessing (StandardScaling), and the Logistic Regression model.
- XGBoost: The core gradient boosting library used for our high-performance regression model.
- Streamlit: The framework chosen to turn our analysis scripts into an interactive user-facing web application.
- Matplotlib & Seaborn: Used for Exploratory Data Analysis (EDA) visualizations.

References

- **Dataset:** Kaggle - Football Players Data
- **Documentation:** Official documentation for XGBoost and Scikit-Learn were consulted for hyperparameter tuning and pipeline optimization.

IX. Conclusion

This project demonstrates the power of Data Science in modernizing sports analytics. By integrating robust data engineering with machine learning, we successfully built the AI Football Performance Analyzer, a tool that moves beyond subjective observation to provide objective, predictive player evaluations.Our results validate the hybrid modeling approach: XGBoost effectively captured complex skill interactions for rating prediction ($R^2 \approx 0.80$), while Logistic Regression accurately classified future potential with ~89% accuracy. The Streamlit deployment proves that these advanced insights can be made instantly accessible and actionable for decision-makers.Ultimately, this project bridges the gap between raw statistics and concrete scouting decisions. While future improvements could include real-time data integration, the current solution successfully empowers analysts to identify "High Growth" talent using rigorous statistical evidence.

X. Possible Improvements

To further enhance the capabilities of the AI Football Performance Analyzer, several features could be implemented in future iterations:

- **Salary Estimation :** Integrating a mechanism to suggest a **fair market value** or recommended salary range for players based on their predicted performance and potential.
- **Position Recommendation :** Developing a classifier to analyze a player's attribute profile and suggest **optimal alternative**

positions (e.g., suggesting a fast winger could be retrained as a wing-back).

- **Real-Time Data:** Connecting to live match data feeds to provide dynamic, week-to-week performance tracking.