

# AI Football Performance Analyzer

---

## Members:

- **Adrien Schuttig**, ECE Paris, [adrienschuttig@gmail.com](mailto:adrienschuttig@gmail.com)
  - **Maxime Laurent**, ESILV Nantes, [maxime.laurent@edu.devinci.fr](mailto:maxime.laurent@edu.devinci.fr)
  - **Louis Le Forestier**, ESILV Nantes, [louis.le\\_forestier@edu.devinci.fr](mailto:louis.le_forestier@edu.devinci.fr)
- 

## Installation

```
git clone https://github.com/louislefo/IA-Application-Project-Louis_Maxime.  
cd IA-Application-Project-Louis_Maxime_Adrien
```

Create your Python environment and install the required packages:

```
pip install -r requirements.txt
```

You can if you want create the models (already in the repository) :

```
python scr/ml_analysis.py
```

And run the application with this command :

```
streamlit run scr/app.py
```

## Table of Contents

1. [Introduction](#)
  2. [Datasets](#)
  3. [Project Structure](#)
  4. [Application Features](#)
  5. [Methodology: From Raw Data to Predictive Signals](#)
  6. [Production Application \(The "Face"\)](#)
  7. [Evaluation & Results](#)
  8. [Related Work](#)
  9. [Conclusion](#)
  10. [Possible Improvements](#)
-

# 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:

- 1. **Analyze** current player statistics to evaluate their overall performance.
- 2. **Predict** a player's future development trajectory (e.g., will they improve, stay stable, or decline?).
- 3. **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.

Source and Description

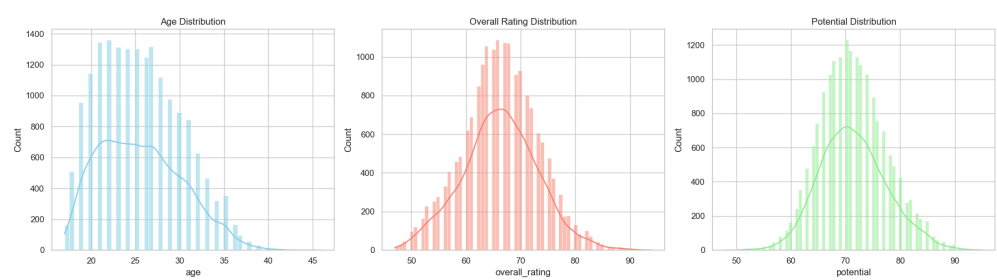
Attribute	Details
Source	<a href="#">Kaggle Link: Football Players Data</a>
Size	17,954 rows (players)
Columns	51 attributes (physical, technical, mental, ratings)

Exploratory Data Analysis (EDA)

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

## 1. Skill Distribution

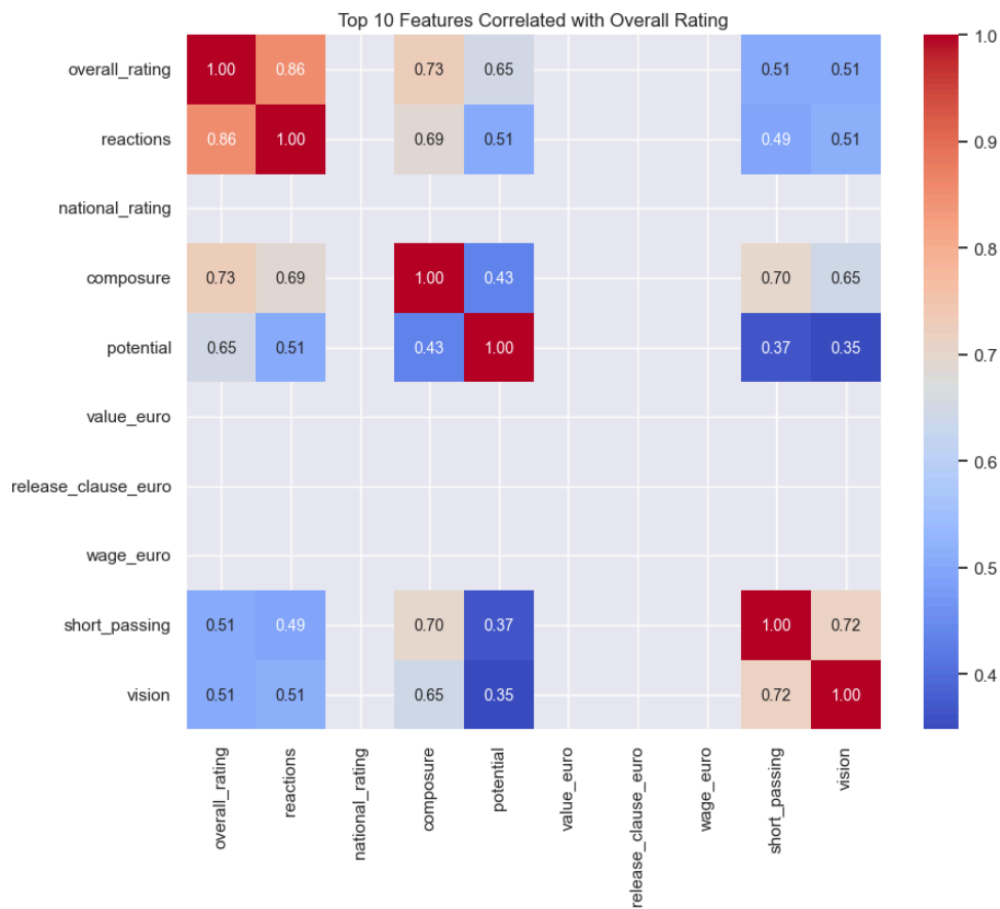
The distribution of ratings (overall\_rating and potential) is **strongly skewed to the right**, indicating that the majority of players fall into the low to medium rating categories.



2. Correlation with Performance (overall\_rating)

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

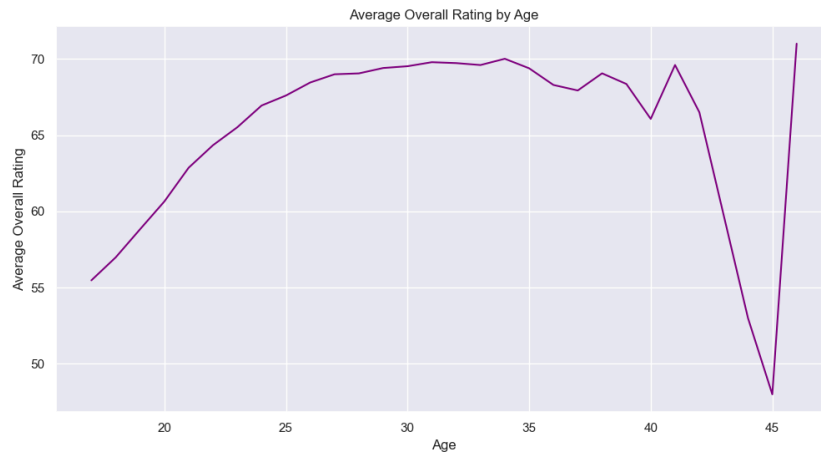
Attribute	Correlation	Observation
reactions	0.84	The most predictive attribute, emphasizing the importance of rapid decision-making.
potential	0.69	Indicates a strong influence of future prospects on the current assessment.
composure	0.68	A key mental factor for performance.
short_passing	0.59	Core technical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



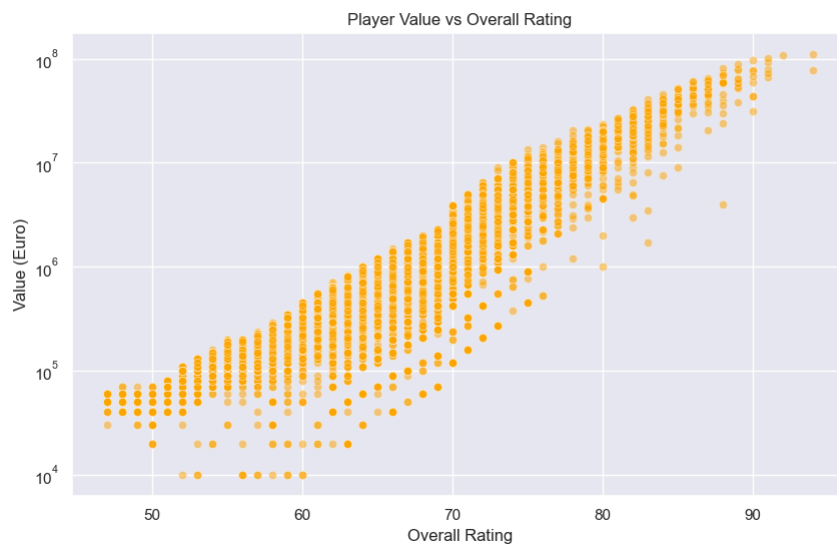
### 3. Age, Value, and Nationality Relationship

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

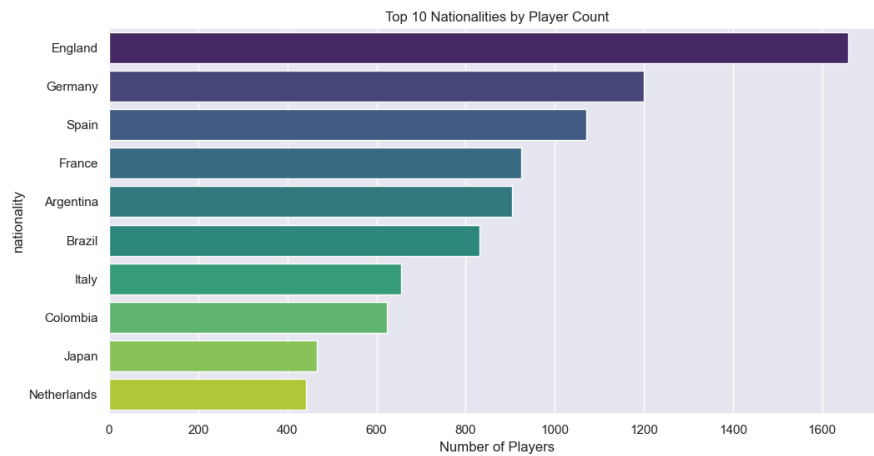
- **Age vs Rating:**



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



- **Top Nationalities:** **ENGLAND**, **GERMANY**, and **SPAIN** are the most represented countries.



## Critical Data Assessment

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

## 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
```

## IV. Application Features

The Streamlit application provides an interactive environment to analyze football players using Machine Learning models.

### 1. Player Input Form

Users can manually enter player information through the sidebar interface.

#### **General attributes:**

- Name
- Age
- Height

#### **Technical abilities:**

- Dribbling
- Finishing
- Short Passing

#### **Physical attributes:**

- Acceleration
- Sprint speed
- Stamina

- Strength

## 2. Machine Learning Predictions

Two models located in the /models folder power the predictions:

- **XGBoost Regression:** predicts the player's estimated *overall rating* (0–100).
- **Multinomial Logistic Regression:** predicts the player's *future development class*:
  - high\_growth
  - likely\_improve
  - stable
  - decline

The app also shows the **probabilities** of each class for better interpretability.

## 3. Real-Time Visual Analysis

After submitting a player, the app automatically generates:

- A bar chart of all attributes
- A table summarizing the player profile
- A “strengths vs weaknesses” summary
- The predicted rating and predicted development class
- The probability breakdown for all classes

This makes the analysis intuitive and visual.

## 4. Save Player to History

Each analysis can be saved into: data/history.csv

Each saved line includes:

- All input attributes
- Predicted rating
- Predicted future class
- Prediction probabilities
- Timestamp

The file is created automatically if it does not exist.

## 5. History Dashboard

A dedicated section in the app displays:

- A full table of all previously analyzed players
- Statistics such as:
  - Average rating
  - Average age

- Number of predicted high\_growth players
- Total number of analyses

This provides a complete overview of past evaluations and long-term insights.

## 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
    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** (Accurate Probabilities) 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 int
    """
    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** (reading from RAM).

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

## VII. Evaluation & Results

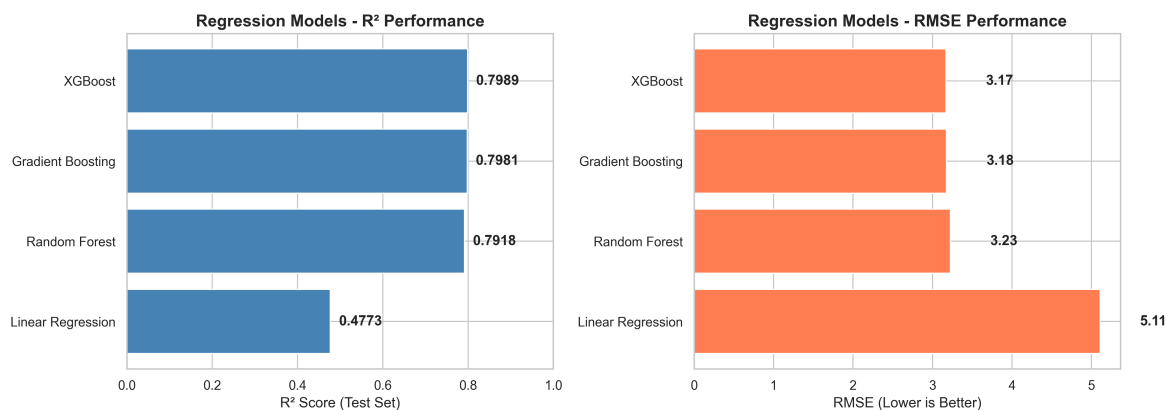
### 1. Regression Analysis: Predicting Overall Rating

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 ( $R^2 \sim 0.47$ ), 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.

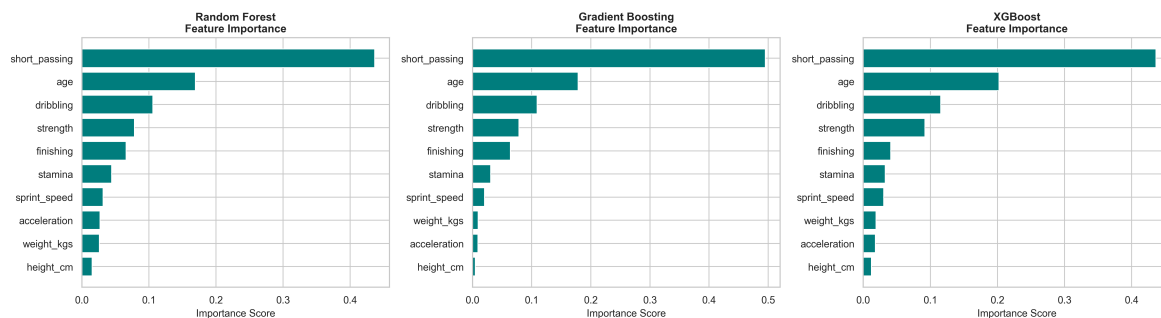


### 2. Feature Importance Analysis

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.

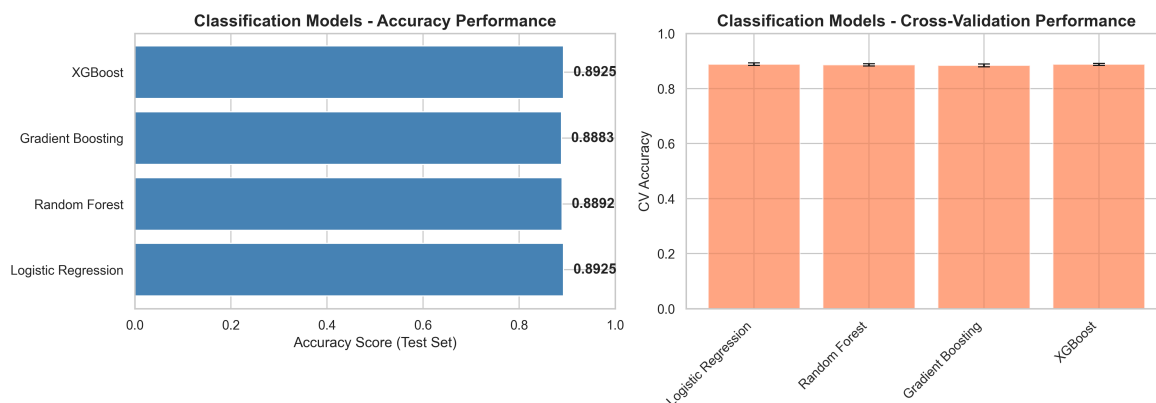


### 3. Classification Analysis: Predicting Future Development

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

#### Results:

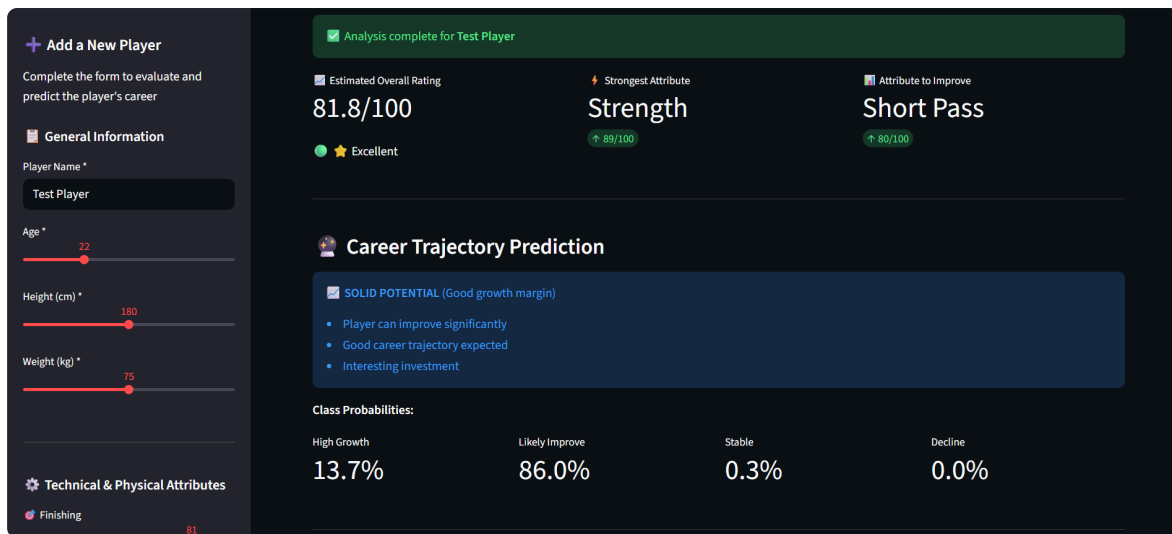
- **Performance:** All models (Logistic Regression, RF, GB, XGB) performed very well, with accuracies around **88-89%**.
- **The Winner: Logistic Regression** and **XGBoost** tied for top performance (Accuracy ~0.8925). Logistic Regression was chosen for the application due to its interpretability and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



### 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 :



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. The tool 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.