Infosys Internship

Project Title: Football Player Market Value Prediction

By – punyamurthy

Week 1 – Data Collection and Exploration

The first week focused on collecting and understanding all the data sources needed for building the player market value prediction model. I explored different datasets related to player performance, injuries, transfer market data, and public sentiment to ensure a comprehensive dataset.

---

1. StatsBomb Data Exploration

I used the StatsBomb Open Data as the main performance dataset since it provides detailed match event-level data like passes, shots, and goals. I studied the JSON structure to understand how to extract and use different event metrics for each player.

2. Player Injury Dataset

To capture how injuries affect a player's market value, I collected data about players' injury history.
Features like injury_count, total_days_out, and average_days_per_injury were planned to represent a player's durability and risk factors.

3. Social Media Sentiment

Since popularity also influences a player's market value, I collected tweets mentioning players.
Using a simple sentiment scoring system — Positive (1), Neutral (0), and Negative (-1) — I calculated an average sentiment score for each player.

4. Transfer Market Data

Transfer values from Transfermarkt were used as the main dependent variable for prediction. It also provided features like nationality, age, and previous transfers.

Conclusion:
By the end of Week 1, I had verified all data sources, identified gaps, and prepared the base for feature engineering in the following week.

Week 2: Data Cleaning and Preprocessing for Modeling

Week 2 focused on transforming the raw, unstructured datasets collected earlier into a unified, consistent, and model-ready format. This phase acted as the bridge between raw data acquisition and analytical modeling. The main goal was to establish a robust data preprocessing and cleaning pipeline to enhance data reliability, followed by structured feature engineering that could extract meaningful insights from complex event data, especially from the StatsBomb repository.

---

1. Data Preprocessing and Standardization

The datasets compiled during Week 1 — encompassing player transfer values, injuries, match events, and public sentiment — required systematic cleaning and normalization to ensure seamless integration in the modeling phase.

A. Data Consistency and Missing Value Treatment

To maintain high data quality, a standardized cleaning function was created and applied across all data sources. The process included two key operations:

- **Duplicate Removal:**
  Any repetitive entries within the datasets were eliminated to avoid redundant or misleading information. For instance, identical injury logs or repeated market entries for a single player were detected and removed to preserve data integrity.
- **Missing Data Imputation:**
  Missing values were handled using context-driven imputation strategies tailored to the nature of the data:
    - **Numerical Fields (Median Imputation):**
      For quantitative features such as likes on tweets or numerical match metrics in StatsBomb data, missing values were replaced with the median instead of the mean. This approach ensured that extreme values (outliers) did not distort the overall distribution, keeping the dataset balanced and statistically consistent.
    - **Categorical Fields ('Unknown' Label):**
      Categorical data gaps — such as missing player positions or event outcome labels — were filled with the string "Unknown". This preserved data completeness while allowing these categories to be treated as valid inputs during encoding and modeling.

Impact:
Through these systematic cleaning steps, data from multiple domains — performance, risk, and sentiment — were successfully aligned to individual players. This prevented data

fragmentation and ensured that every record mapped correctly to its corresponding player profile in later feature engineering steps.

---

2. Feature Engineering Approach: Deriving Structured Insights

After achieving a clean dataset, the next step involved generating powerful predictive features. Feature engineering transformed the cleaned raw data into structured quantitative metrics, capturing multiple perspectives of player evaluation — namely **Performance**, **Injury Risk**, and **Market Context**.

A. Performance Feature Engineering (StatsBomb Data)

To translate detailed match events into usable model inputs, an aggregation-first approach was implemented. Instead of using thousands of individual events per player, the data was summarized into meaningful, player-level statistics.

- **Event Filtering:**
  The StatsBomb events were first categorized based on their type (e.g., “Pass”, “Shot”, “Carry”). This segmentation helped isolate relevant subsets for calculating key performance indicators.
- **Passing Metrics:**
  A pass was treated as successful when pass.outcome.name was null (or “Unknown” after imputation), signifying a completed pass. Using this rule, **Passes Completed** and **Total Passes Attempted** were derived to compute ratios such as pass accuracy and overall ball distribution capability.

- **Attacking Metrics (xG & Assists):**
    - **Expected Goals (xG):**
      This variable measured the probability of scoring based on shot quality. It was computed by summing the shot.statsbomb_xg values for each player, providing a realistic indicator of scoring potential independent of chance.
    - **Assists:**
      Assists were computed by joining the Pass and Shot events. When a pass's id matched a shot.key_pass_id, it signified a successful assist. This allowed direct identification of players contributing to goal creation.

B. Injury Feature Engineering (Risk Evaluation)

Player injury data was crucial for quantifying physical risk and availability. The raw chronological records were summarized into compact, player-level indicators.

- **Injury Frequency and Duration:**
  Metrics such as **injury_count** (number of injuries) and **total_days_out** (cumulative recovery time) were calculated for each player.
- **Severity Ratio (days_per_injury):**
  This derived variable measured the average recovery time per injury (total_days_out / injury_count). It served as a reliability index, identifying players with frequent minor issues versus those with fewer but more severe injuries. A high value in this feature typically signals reduced market confidence in the player's durability.

C. Market Feature Engineering (Transfer and Financial Context)

To strengthen the predictive scope of the target variable (market value), contextual attributes around each player's transfer and club details were incorporated. This included variables like transfer club, nationality, contract duration, and movement frequency. These enrichments helped interpret valuation changes in relation to transaction-specific factors.

D. Sentiment Feature Engineering (Public Perception Metrics)

Integrating sentiment data added a behavioral and psychological layer to player valuation — representing how fan perception, media hype, or social reactions could influence market worth.

- **Entity Recognition:**
  A mapping logic was designed to associate tweets with the correct player names from a master list, overcoming the challenge of ambiguous or incomplete mentions.
- **Scoring Mechanism:**
  Each matched tweet was assigned a numerical sentiment score — Positive (1), Neutral (0), or Negative (–1). The **avg_sentiment** score was then computed per player, capturing overall fan sentiment polarity.

This process transformed unstructured text into measurable, model-friendly attributes reflecting external perceptions beyond on-field performance.

---

3. Deliverables and Transition to Week 3

By the end of Week 2, all essential feature subsets were finalized and exported as individual files:

- performance_features.csv – capturing derived gameplay statistics
- injury_features.csv – summarizing risk-related features
- market_features.csv – detailing financial and contextual elements

These structured datasets laid the foundation for the **Week 3 integration phase**, where merging, scaling, and normalization operations would produce the unified feature matrix for modelling

Week 3: Data Integration, Scaling, and Model Preparation

Week 3 marked a significant turning point in the project — transitioning from individual feature creation to constructing a unified, analytics-ready dataset for modeling. This stage focused on **merging**, **normalizing**, and **preparing features** from multiple domains to ensure that the upcoming predictive models would receive a balanced, coherent input.
The process also included **feature correlation checks**, **scaling transformations**, and **data partitioning** to create a stable training environment.

---

1. Data Integration and Merging Framework

After completing Week 2's feature generation (performance, injury, market, and sentiment metrics), the first major task was to

consolidate all outputs into a single master feature matrix.
This unified dataset served as the central foundation for model training.

A. Key-Based Merging Logic

Each dataset — such as performance_features.csv, injury_features.csv, and market_features.csv — was merged using a **common player identifier** (player_id or player_name).
The join operation followed a **left merge strategy**, ensuring that all players with valid performance metrics were retained, even if corresponding market or sentiment data were missing. This approach helped preserve model inclusivity while maintaining data completeness.

B. Handling Overlapping Columns and Null Values

During integration, overlapping or partially missing features were resolved through:

- **Column Deduplication:** Preventing redundant attributes such as multiple “club_name” or “nationality” columns from causing confusion.
- **Null Value Reassessment:** Post-merge missing values were rechecked and imputed again using the same strategies defined in Week 2 — median for numeric fields, and “Unknown” for categorical fields.

This ensured that the final feature matrix was **consistent, clean, and free from mismatched records** across data sources.

---

2. Feature Scaling and Normalization

Once integration was complete, feature distributions were analyzed to bring all variables to a uniform scale. Since the datasets combined different units (e.g., goals, injury days, sentiment scores, and market prices), normalization was essential to prevent model bias.

A. Scaling Strategy

Two primary transformations were applied based on the nature of features:

- **Standardization (Z-Score):**
  For continuous variables like xG, assists, injury_duration, or transfer_fee, standardization was used to center the mean at 0 and scale the variance to 1.
  This transformation helps models such as Linear Regression and Ridge Regression perform efficiently without being influenced by feature magnitude.
- **Min–Max Normalization:**
  For bounded metrics such as sentiment scores or ratios (e.g., pass_accuracy), values were normalized between 0 and 1 to retain interpretability and simplify cross-feature comparisons.

B. Outlier Detection and Treatment

Outlier analysis was conducted to identify extreme values that could distort model behavior — particularly in variables like total_days_out or market_value.
Using the **Interquartile Range (IQR) method**, outliers beyond 1.5×IQR from the quartiles were flagged and either capped or smoothed, depending on their contextual significance.

---

3. Feature Correlation and Redundancy Analysis

To enhance model efficiency and interpretability, correlation checks were performed to understand interdependencies between variables.

- **Correlation Matrix (Heatmap):**
  A visual heatmap was generated to identify features with high pairwise correlation (above 0.9). Redundant variables — for example, total_passes and pass_accuracy — were reviewed for removal or dimensional reduction.
- **Multicollinearity Check (VIF Score):**
  Variance Inflation Factor (VIF) analysis was used to detect multicollinearity among predictors. Features with extremely high VIF values were either excluded or transformed to prevent instability during model training.

Through this step, the dataset was optimized for both **computational efficiency** and **predictive clarity**, ensuring that only truly informative variables were carried forward.

---

4. Data Splitting and Sampling for Modeling

Once the dataset was fully standardized and refined, it was partitioned into **training** and **testing** sets.

- **Training Set (80%)** was used to fit and calibrate the model.
- **Testing Set (20%)** was reserved for independent evaluation to prevent overfitting.

Additionally, **K-Fold Cross-Validation** (typically with k=5) was planned for Week 4 to further ensure model robustness across different subsets of the data.

---

5. Output Generation and Storage

The final integrated dataset was exported as:

- final_feature_matrix.csv — containing all processed player-level features ready for modeling.
- Metadata files documenting feature definitions and data source lineage were also created for transparency and reproducibility.

This clean, balanced, and documented dataset formed the **core modeling input** for Week 4's predictive experiments.

---

6. Summary and Transition

By the end of Week 3, the project achieved:

- Successful merging of all engineered datasets into a single structured matrix
- Standardized and normalized feature distributions
- Elimination of redundancy and mitigation of outliers
- Clear readiness for model implementation

Week 4: Model Development, Training, and Evaluation

Week 4 marked the beginning of the **predictive modeling phase**, where all the cleaned and standardized data from Week 3 were transformed into actionable predictive insights.

The focus was on selecting appropriate regression algorithms, optimizing their performance through fine-tuning, and validating their effectiveness using cross-validation and evaluation metrics. This stage established the foundation for identifying the **most reliable model** for predicting football players' market values.

---

1. Model Selection and Setup

After finalizing the final_feature_matrix.csv, various regression models were shortlisted based on their suitability for continuous target prediction and robustness to high-dimensional data.

A. Candidate Algorithms

The chosen algorithms represented a balanced mix of **linear** and **ensemble-based** learning techniques:

- **Linear Regression (Baseline):**
  Served as the benchmark model to assess initial data fit and general trends.
- **Ridge and Lasso Regression:**
  Regularized versions of linear regression used to handle multicollinearity and reduce overfitting by penalizing large coefficients.
- **Random Forest Regressor:**
  A tree-based ensemble model capable of capturing non-linear interactions among features.
- **XGBoost Regressor:**
  The primary advanced model, known for its strong predictive performance, gradient boosting mechanism, and ability to handle noisy or missing data efficiently.

B. Environment and Training Pipeline

All models were trained using **scikit-learn** and **XGBoost** libraries within a controlled Python environment.
A unified pipeline was designed to:

- Load the processed feature matrix
- Split data into training and testing subsets
- Apply scaling and encoding where necessary
- Fit each model systematically and log results for performance comparison

This consistent workflow ensured transparency, reproducibility, and efficient experimentation.

---

2. Hyperparameter Optimization

To enhance accuracy and prevent underfitting or overfitting, **hyperparameter tuning** was conducted using **GridSearchCV** and **RandomizedSearchCV** techniques.

A. Optimization for Ensemble Models

- **Random Forest Parameters:**
  Tuned parameters included n_estimators, max_depth, and min_samples_split. The best combination was chosen based on the highest **R^2 score** and lowest **RMSE** on validation folds.
- **XGBoost Parameters:**
  Fine-tuning focused on learning_rate, n_estimators, max_depth, and subsample ratio.
  Early stopping was implemented to halt training when

validation performance plateaued, reducing computational cost and overfitting.

B. Regularization in Linear Models

- Ridge and Lasso regressions were optimized by adjusting the **alpha** parameter, balancing model flexibility and generalization strength.

Each optimization cycle was followed by model retraining and evaluation on the validation set to confirm the stability of performance improvements.

---

3. Model Evaluation and Validation

After optimization, each model's effectiveness was tested using multiple evaluation metrics to capture both accuracy and error spread.

A. Evaluation Metrics

- **R^2 Score (Coefficient of Determination):**
  Indicates how well the model explains variance in the target variable.
- **Mean Absolute Error (MAE):**
  Measures the average magnitude of prediction errors.
- **Root Mean Squared Error (RMSE):**
  Penalizes larger deviations, making it ideal for detecting high-impact prediction errors.

These metrics provided a balanced understanding of both predictive precision and reliability.

B. Cross-Validation (K-Fold Validation)

To ensure robustness and generalizability, **K-Fold Cross-Validation (k=5)** was applied.
The dataset was divided into five subsets, with the model trained on four and validated on the remaining one in rotation. The final score represented the average performance across all folds — minimizing bias from random data splits.

C. Performance Insights

- **Linear and Ridge Models:**
  Provided stable but limited accuracy, serving as strong baselines.
- **Random Forest:**
  Improved performance significantly by capturing non-linear patterns but required more computation.
- **XGBoost:**
  Achieved the **highest R^2 score** and the **lowest RMSE**, demonstrating superior handling of mixed-type features and interaction effects.
  It was selected as the **final production model** for deployment.

---

4. Model Interpretation and Feature Importance

Beyond predictive accuracy, interpretability was a key consideration for practical insights.

A. Feature Importance Extraction

The **XGBoost feature importance module** was utilized to quantify the contribution of each predictor.
Top contributing variables included:

- Expected Goals (xG)
- Total Assists
- Injury Severity Ratio (days_per_injury)
- Avg Sentiment Score
- Market Transaction Context

These insights validated domain assumptions — confirming that performance, fitness, and public perception jointly influence player valuation.

B. Visualization

Two main graphs were generated for presentation and reporting:

1. **Feature Importance Chart:** Highlighting the top 10 predictors influencing the model.
2. **Predicted vs. Actual Market Value Plot:** Showing how closely the predictions aligned with real market values.

Both graphs were crucial in visually communicating the model's reliability during evaluation.

---

5. Model Export and Deployment Readiness

After final validation, the selected **XGBoost model** and its corresponding **preprocessor pipeline** were serialized using joblib.

- Model file: xgb_model_final.joblib
- Preprocessor: preprocessor_final.joblib

These files were later integrated into the **Streamlit web app** (developed in Week 5) to enable real-time predictions.

---

6. Summary and Transition

By the end of Week 4:

- All regression models were trained, tuned, and evaluated.
- XGBoost emerged as the top-performing model with strong generalization capability.
- Comprehensive performance metrics and interpretability plots validated its predictive trustworthiness.
- Model artifacts were stored for seamless deployment.

This completed the modeling and evaluation pipeline, leading directly into **Week 5**, which focused on building the **interactive AI-driven web application** for end-user access and visualization.

Week 5: Model Deployment and Streamlit Application

Week 5 focused on deploying the trained XGBoost model into an **interactive Streamlit web application**, enabling real-time prediction of player market values. The aim was to convert the analytical model into a usable interface that provides clear, data-driven insights to end users.

---

1. Objective and Overview

This phase transitioned the project from modeling to **practical deployment**.
The app allows users to:

- Input or select player attributes
- Generate instant market value predictions
- Compare predicted and actual values
- View analytics through visual dashboards

The emphasis was on **simplicity, clarity, and interactivity**.

---

2. Deployment Framework

The deployment stack included:

- **Streamlit** for the interface
- **Python (XGBoost, Scikit-learn)** for model inference
- **Joblib** for loading the model and preprocessor
- **Pandas, NumPy, Altair** for data handling and charts

Files were organized into a clean project structure with folders for models, data, and the Streamlit app.

---

3. Streamlit Interface Design

The application was divided into two main sections:

- **User Input Panel:**
  Users can select a player or enter features like goals, assists, injuries, sentiment score, and age.
  These inputs are automatically processed using the same preprocessor used during training.
- **Output and Visualization:**
  The predicted value is displayed with clear numeric cards

and a simple comparison chart showing actual vs. predicted value.
Additional visuals highlight performance and risk insights for interpretability.

---

4. Model Integration

The backend connected the trained model and preprocessor:

model = joblib.load("xgb_model_final.joblib")

preprocessor = joblib.load("preprocessor_final.joblib")

This ensured consistent transformations during both training and prediction phases.

---

5. Visualization and Insights

The app included key charts such as:

- **Feature Importance:** Showing which metrics influenced predictions most.
- **Comparison Graphs:** Comparing a player's metrics with peers.
- **Trend Lines:** Displaying changes in predicted value over time.

These visuals made outputs easier to understand and more engaging.

---

6. Testing and Validation

Before deployment, the app was tested for:

- **Response speed** (real-time prediction)
- **Error handling** (missing or invalid inputs)
- **Cross-browser performance** (Chrome, Edge)

All tests confirmed stable operation and accurate prediction results.

---

7. Deployment and Results

The application was deployed using **Streamlit Cloud** for easy access and version control.
Once live, it successfully demonstrated AI-based player valuation with interactive analytics and quick performance.

---

8. Outcomes

- The final app delivered **real-time, explainable predictions**.
- Integration of visuals improved usability for analysts and non-technical users.
- Deployment validated the end-to-end model pipeline from raw data to decision-ready output.

Week 6: Evaluation, Documentation, and Final Presentation

Week 6 marked the **completion phase** of the project, focusing on model evaluation, final documentation, visualization, and presentation readiness. The goal was to ensure the developed

system was both **technically sound** and **professionally deliverable**.

---

1. Objective

The week aimed to:

- Validate the model's performance
- Document the full workflow
- Prepare visuals, graphs, and explanations for the final presentation
- Ensure the Streamlit app and report align with project goals

---

2. Model Evaluation and Performance Analysis

Comprehensive testing was done to measure how well the XGBoost model performed on unseen data.
The evaluation included key metrics like:

- **R^2 Score:** To check overall prediction accuracy
- **MAE (Mean Absolute Error):** To assess average deviation
- **Cross-Validation:** To confirm model stability across multiple data folds

Results showed strong accuracy and minimal overfitting, proving the model's generalization ability.

---

3. Comparative Study

The XGBoost model's performance was compared with baseline models such as **Linear Regression** and **Random Forest**.

- XGBoost achieved higher accuracy and lower error rates.
- It handled complex, non-linear relationships better than the baseline models.
  This comparison confirmed that XGBoost was the best choice for final deployment.

---

4. Visualization and Interpretation

Graphs and dashboards were refined to clearly explain:

- Predicted vs Actual player market values
- Feature importance rankings
- Model consistency across folds

These visuals helped in **communicating the results effectively** during the final presentation.

---

5. Documentation and Report Finalization

A complete **technical report** was prepared summarizing all six weeks:

- Data collection, preprocessing, and feature engineering
- Model training, tuning, and deployment
- Final evaluation and insights

All scripts, datasets, and model files were properly organized for reproducibility.

---

6. Final Presentation Preparation

A PowerPoint presentation was designed to visually narrate the project journey.
Key slides included:

- Project Overview
- Data and Feature Pipeline
- Model Results and Graphs
- Streamlit App Demonstration
- Future Scope and Conclusion

Mock presentations were done to ensure time control and clarity during delivery.

---

7. Key Learnings and Takeaways

- Gained strong understanding of **data preprocessing**, **feature engineering**.
- Learned how to build end-to-end AI systems with real-time prediction
- Improved presentation, visualization, and technical documentation skills

Week 7: Model Tuning and Optimization

Objective:
Refine the Ensemble Model (LSTM + XGBoost) built in Week 6 to achieve optimal accuracy for predicting the **Overall Rating**.

---

1. Data Preparation & Feature Encoding

- Reconfirmed preprocessing pipelines from previous weeks.
- **Categorical Encoding:** Converted *attacking* and *defensive work rates* from categorical labels *(low, medium, high)* to numeric scale *(0, 1, 2)*.
- **Time-Series Formation:** Data structured into sequential time steps suitable for LSTM input.
- **Feature Scope:** Comprehensive feature set finalized as model input.

---

2. Hyperparameter Tuning Process
Optimization focused on fine-tuning both components of the ensemble.

A. LSTM Optimization

- **Goal:** Minimize Validation Loss.
- **Search Parameters:**
    - LSTM Units: *32 → 128*
    - Dropout Rates: *0.0, 0.2, 0.3*
    - Learning Rates: *1e-2, 1e-3, 1e-4*
- **Best Configuration:**
    - Units: **96**, Dropout: **0.0**, Learning Rate: **0.0001**

B. XGBoost Optimization (RandomizedSearchCV)

- **Goal:** Minimize RMSE (scoring = *neg_mean_squared_error*).
- **Meta-Learning Setup:** LSTM predictions added as an **extra input feature** to the XGBoost model, allowing it to act as a meta-learner and correct residual errors.
- **Best Parameters:** Identified combination favoring **lower learning rate** for smoother convergence and higher stability.

---

3. Outcome & Insights

- Final tuning phase completed under **Milestone 6**.
- Resulted in a **fully optimized ensemble** delivering the **lowest prediction error**.
- Comparative evaluation confirmed the **Tuned Ensemble (LSTM + XGBoost)** as the top-performing predictive model.

Week 8: Final Integration, Analysis & Conclusion

Purpose:
Serve as the capstone stage—integrating all model components, validating results, and preparing the project for deployment.

---

1. Final Validation & Performance Testing

- Conducted performance evaluation on a **held-out unseen dataset**.
- Metrics reaffirmed the **efficiency of the LSTM-based sequential learning approach** and the strength of the engineered features.

---

2. Deliverables & Model Assets

All processed data, trained models, and deployment scripts were finalized for future reuse:

- best_lstm_model.h5 → Final sequential prediction engine
- week6_xgboost_model.pkl → Benchmark model for comparison
- player_features_model_all_imputed.csv → Final cleaned dataset
- Final_Prediction_Streamlit_App.py → Deployment-ready prediction interface

---

3. **Project Closure Summary**
   Week 8 marked the **successful conclusion** of the full predictive modeling pipeline—
   from data preparation and feature engineering to **model tuning, validation, and deployment readiness.**

THANK YOU