Infosys Internship

Title: Football Player Market Value Prediction

Final Report

By – Abhinav Jha

Week 1: Data Exploration and Collection

The primary objective for Week 1 was to execute a comprehensive data discovery and collection phase. This involved identifying reliable data sources for player performance, physical risk (injuries), market valuation (transfers), and public perception (sentiment). The raw data was then subjected to initial Exploratory Data Analysis (EDA) to understand its size, structure, and inherent quality challenges.

1. StatsBomb Open Data Exploration

The project leveraged the StatsBomb Open Data repository, which provides rich, granular event data for professional football matches. Understanding the nested JSON structure of this data was paramount for later feature engineering, as most detailed performance metrics would be derived from these events.

2. Player Injury Analysis

The acquisition and analysis of injury data introduced the critical dimension of player risk and durability to the prediction model. Market value is known to penalize players with a history of frequent or severe injuries.

Feature Strategy:
The core feature engineering strategy centered on deriving injury_count, total_days_out, and the normalized days_per_injury (severity score).

3. Social Media & Sentiment Analysis

To capture the subjective, "soft factors" influencing market value—namely popularity, brand value, and fan perception—Twitter data was sourced.

- **Analytical Objective:**
  The goal was to pivot from raw tweet volume to an average sentiment score reflects the prevailing public toward a player.
- **Initial Findings (Example):**
  An initial scan for prominent athletes confirmed that the tweets captured relevant content, often related to performance anticipation, contract speculation, or fan opinion following matches.
- **Feature Engineering Challenge:**
  The primary challenge identified in this phase was the technical hurdle of accurately attributing a tweet to the correct player entity (especially given common names or non-explicit mentions) using the available player name lists. A mapping process would be required to assign Positive (1), Neutral (0), or Negative (-1) scores to each player mention before aggregation.

4. Market Value & Transfers

This dataset provided the core dependent variable (Y) for the entire predictive model, alongside essential features (e.g., transfer club, nationality).

- **Data Source:**
  The data was acquired from Transfermarkt, a primary source for football market valuation.

Conclusion:
The comprehensive data exploration in Week 1 successfully validated the integrity of the data sources, quantified initial risk factors, and established a clear path forward for the complex, multi-domain feature engineering executed in subsequent weeks.

Week 2: Data Cleaning and Preprocessing for Modeling

The second week marked the crucial transition from disparate, heterogeneous raw data into a unified, clean, and structured format suitable for predictive modeling. The core objectives centered on implementing robust data cleaning pipelines to ensure integrity and devising a feature engineering strategy to synthesize high-signal metrics from complex data structures—primarily the StatsBomb event data.

1. Preprocessing and Data Standardization

Data from Week 1's collection phase—covering market values, player injuries, social media sentiment, and event data—required thorough validation and standardization before they could be merged.

A. Data Integrity and Missing Value Handling

A standardized cleaning function was applied across all foundational datasets to ensure data integrity:

- **Deduplication:**
  All redundant entries across the datasets were systematically identified and removed. For instance, ensuring that multiple identical injury entries or market records for the same player were not retained.
- **Missing Value Imputation Strategy:**
  A differential imputation approach was adopted based on variable type and domain context:
  - **Numerical Imputation (Median):**
    For continuous numerical columns like Number of Likes (in Twitter data) or certain event metrics in the flattened StatsBomb logs, missing values were imputed using the median. The median was selected over the mean to mitigate the distorting effect of outliers, preserving the integrity of the underlying distribution while maintaining a complete dataset for modeling.
  - **Categorical/Text Imputation ('Unknown'):**
    Missing values in categorical columns, such as Position (injury data) or various event metadata fields (e.g., pass.outcome.name), were replaced with the string "Unknown". This retained the records while enabling downstream processes like one-hot encoding without data loss, treating "missingness" as a distinct, actionable category.

Impact:
This painstaking process ensured that statistical data from a player's performance events could be accurately mapped to their corresponding injury history and market value, eliminating potential data fragmentation errors in the final feature matrix.

2. Feature Engineering Strategy: Aggregation from Raw Data

Feature engineering transformed the raw, disparate information into a set of structured, predictive variables. The strategy focused on deriving quantitative metrics that capture three primary predictive dimensions: Performance, Risk (Injury), and Market/Transaction context.

A. Performance Feature Strategy (StatsBomb)

An aggregation-first approach was used to convert thousands of granular events per match into single, descriptive career metrics per player.

- **Event Filtering:**
  The core logic involved filtering the massive Events DataFrame by type.name (e.g., "Pass", "Shot") and then applying domain-specific aggregation functions.
- **Pass Metrics:**
  To derive metrics like Passes Completed, the methodology exploited the StatsBomb structure: a pass is considered completed if the pass.outcome.name column is null/NaN (or imputed as "Unknown" after initial cleaning), signifying no interruption or failed reception. Total passes were derived from the count of all pass events.
- **Goal/Attacking Metrics (xG & Assists):**
  - **Expected Goals (xG):**
    This continuous variable represents the quality of a player's scoring chances, irrespective of whether they scored. It was calculated by summing the pre-calculated shot.statsbomb_xg variable for all shots taken by a player.
  - **Assists:**
    This required a slightly more complex join operation. An assist was defined as a Pass event whose id matched the shot.key_pass_id associated with a Goal event. This relational logic was implemented to link the goal scorer back to the last player who set up the shot.

B. Injury Feature Strategy (Risk Quantification)

The strategy translated chronological injury records into summary features representing player fragility and chronic risk:

- **Count and Duration:**
  The straightforward metrics injury_count and total_days_out (summing the difference between Date of Injury and Date of Return across all records) were calculated first.
- **Severity Metric:**
  The key derived metric was days_per_injury, calculated as Total Days Out / Injury Count. This ratio serves as a severity index, differentiating a player who missed 10 days due to ten different minor knocks from a player who missed 100 days due to one major injury. This feature is hypothesized to have a strong negative correlation with market value.

C. Market Value Feature Strategy (Transaction Context)

The goal was to enrich the core market value figure with features providing context about the transaction itself:

D. Sentiment Feature Strategy (Public Perception)

The primary challenge was overcoming the lack of explicit player tags in the generalized Twitter dataset. The initial plan formulated the steps:

- **Entity Mapping:**
  Iterating through the large list of unique player keys and attempting to match those strings within the text of the 22,524 tweets.
- **Score Calculation:**
  Once a tweet was mapped to a player, the inherent Positive (1), Neutral (0), or Negative (–1) sentiment was used to calculate an avg_sentiment score for that player.

3. Output for Next Phase

The outputs from Week 2 feature engineering—including performance_features.csv, injury_features.csv, and market_features.csv—were essential for the Week 3 merging and scaling activities, which culminate in the final, comprehensive feature matrix.

Week 3: Advanced Feature Engineering and Data Finalization

The third week built directly upon the foundation of data cleaning established in Week 2, concentrating on completing the complex feature engineering tasks required to build the final predictive dataset. This week involved synthesizing the clean data assets into a comprehensive feature matrix and performing the necessary technical preparations for the upcoming modeling phases.

1. Feature Engineering Execution: Granular Metrics

Building on the methodological strategies defined in Week 2, the team executed the final calculations for the high-signal features that capture player value from multiple perspectives.

A. Performance Feature Finalization (StatsBomb)

The raw event streams, once standardized and filtered, were aggregated to quantify player contribution on the pitch.

Expected Goals (xG) Calculation:

2023/2024 season, successfully loading:

- **Matches:** All **34 matches** for the selected season.
- **Events:** A total of **137,765 events** (including Passes, Shots, Carries, etc.) associated with those 34 matches.
- computed by summing the shot.statsbomb_xg values. This metric, which serves as a quality-adjusted measure of offensive threat, was critical for decoupling a player's true scoring potential from variance in finishing luck. The final xG metric was normalized per player to represent their overall offensive value.
- **Pass and Possession Metrics:**
  Total passes attempted and completed were finalized. The key derived metric, Pass Accuracy, calculated as Passes Completed / Passes Attempted, served as a foundational measure of efficiency in possession play.
- **Assists (Chance Creation):**
  The precise linking of successful passes to goal events allowed for the quantification of Assists. This metric isolated a player's value as a creator and final-third facilitator.

B. Injury and Risk Feature Finalization

The risk metrics were finalized using the cleaned injury data:

- injury_count: Total number of distinct injury instances recorded per player.
- total_days_out: The aggregate number of days a player was unavailable due to injury.

- days_per_injury: The calculated ratio of total days missed to injury count, serving as a primary indicator of chronic injury severity and risk.

2. Data Integration and Technical Preparation

The core achievement of Week 3 was the integration of all engineered features— Performance, Risk, and Market—into two final, model-ready datasets.

A. Feature Merging

Using the standardized player_key derived in Week 2, the individual feature tables (Performance, Injury, Market) were merged using outer joins to preserve all unique players identified across the sources.

- **Imputation of Missing Feature Blocks:**
  The use of outer joins introduced blocks of NaN values where a player existed in one source (e.g., Market data) but not in another (e.g., Performance data). A crucial imputation strategy was employed:
  - Performance metrics (xG, Assists, Passes) were imputed with zero (0) where missing, based on the assumption that a player not registered in the event logs for the Bundesliga season had a zero contribution to those metrics.
  - Injury metrics were imputed with zero (0) where missing, based on the assumption that a player not listed in the injury logs had zero historical injuries and days out.

B. Final Dataset Generation and Standardization

The merged and imputed data was prepared in its final numerical format for direct consumption by machine learning models.

- **Z-Score Normalization (StandardScaler):**
  All remaining continuous numeric features (e.g., fees, pass counts, days out) were transformed using Z-score normalization. This step is vital to ensure that all model inputs operate on the same scale (mean 0, standard deviation 1), preventing any single feature from unfairly dominating the learning process due to its magnitude.
- **Dataset Deliverables:**
  This intensive phase resulted in two core outputs for subsequent modeling:
  - player_features_model_all_imputed.csv (90,000 rows):
    The comprehensive dataset containing all players and all imputed, scaled features. This dataset is suitable for models attempting to predict value across the entire player population.

3. Conclusion of Week 3

Week 3 successfully completed the quantitative data preparation phase. By executing a meticulous merging strategy and applying necessary scaling and imputation, the project transitioned from raw, fragmented inputs to a single, robust, and clean feature matrix ready.

Week 4–5: Modeling (LSTM) & Sentiment Analysis

Weeks 4 and 5 marked the pivotal shift from data preparation to predictive modeling and the initial integration of soft features via sentiment analysis. The core effort centered on leveraging the complex, combined feature matrix (created in Weeks 2–3) to train a deep learning model capable of capturing temporal trends in player market valuation.

1. Predictive Modeling: Multivariate LSTM Baseline

The primary task was building the baseline predictive model using a Multivariate Long Short-Term Memory (LSTM) network, which is particularly suited for processing sequential data like a player's career statistics over time.

A. Data Preparation for LSTM Input

The final feature matrix, player_features_model_all_imputed.csv, served as the source, having been imputed and scaled previously.

- **Feature Selection:**
  The model utilized key features as input, capturing essential components of player value across all domains:
  passes_attempted, expected_goals, goals, assists, injury_count, total_days_out, and the target variable, etc.
- **Sequential Transformation:**
  A crucial step for the LSTM was transforming the tabular data into sequences using a lookback window of 3 time steps.This structured the data into a 3D array where the model learns to predict the next market value based on the previous three snapshots of the player's performance/risk profile.
- **Scaling:**
  All inputs were normalized using a MinMaxScaler (0–1), ensuring that all features contributed equally to the learning process regardless of their initial magnitude.

B. LSTM Model Architecture and Training

A basic, sequential LSTM network was designed as the initial deep learning baseline:

- **Architecture:**
  The model consisted of a single LSTM layer with 50 units (using a 'relu' activation) followed by a Dense(1) output layer, designed to predict a single future market value. The model contained total trainable parameters.
- **Compilation:**
  The model was compiled using the standard configuration of the Adam optimizer and the Mean Squared Error (MSE) loss function.
- **Training & Evaluation:**
  The model was trained for 50 epochs using an 80/20 split for training and validation

data, focusing on minimizing the error between predicted and actual scaled market values.

- **Baseline Performance:**
  The final Validation RMSE achieved by this initial LSTM model on the scaled data. This metric established the initial deep learning benchmark, providing the target for subsequent tuning efforts.

2. Sentiment Analysis and Interpretation

This phase finalized the analytical integration of the unstructured social media data, providing crucial "soft factors" that often influence market value outside of raw on-pitch performance.

A. Sentiment Data Cleaning and Distribution

The raw Twitter data was processed to quantify public opinion:

- **Cleaning:**
  The raw data was cleaned by dropping an initial unnamed index column and converting the Date Created field to a datetime format.
- **Overall Distribution:**
  Analysis of the predefined sentiment labels showed a balanced, but slightly positive skew in public discourse.

B. Sentiment Feature Generation

The final step was generating the quantifiable sentiment score for the predictive model:

- **Scoring Logic:**
  A map was used to convert the categorical sentiment labels into numerical scores: Positive (1), Neutral (0), and Negative (–1).
- **Feature Calculation:**
  The avg_sentiment score for each player was calculated as the mean of all sentiment scores associated with their name across all analyzed tweets. This feature served as a critical non-performance metric in the final consolidated feature table.

C. Final Deliverables (Week 5 Outputs)

The following assets were finalized and saved for the model tuning and final comparison phases in Weeks 6–8.

Week 6: Baseline Model Comparison and Prediction

Generation

Week 6 was dedicated to establishing the initial performance benchmark for the player market value prediction system by training the two primary, distinct modeling architectures. This involved running the Multivariate Long Short-Term Memory (LSTM) model and an Ensemble model (likely an XGBoost hybrid) on the final processed feature set from Week 3, and then analyzing the results to set the optimization target for Week 7.

1. Model Objectives and Selection Rationale

The project's central hypothesis required testing if sequential deep learning (LSTM) could outperform traditional, non-temporal machine learning models (Ensemble/XGBoost) in predicting an inherently time-series variable like player market value. Week 6 was the first validation point for this comparative approach.

A. Multivariate LSTM Rationale

The LSTM model was employed specifically to exploit the sequential nature of the data. By processing the player's career history (represented as a 3-step sequence of performance, injury, and market metrics) as an ordered input, the LSTM was uniquely positioned to capture:

- **Temporal Dependencies:**

  How past performance trends and injury accumulation sequentially influence current and future value.

- **Non-Linear Patterns:**

  The complex, non-linear relationships between disparate features like market hype (Sentiment) and on-pitch quality (xG).

B. Ensemble Model Rationale

The Ensemble model, often implemented as an XGBoost Regressor or a hybrid combining XGBoost with the LSTM output, provided a critical benchmark. Gradient boosting machines are powerful but typically treat the input features (X) as independent, cross-sectional samples.

Its performance measurement validated the extent to which the problem could be solved without explicit time series modeling:

- **Feature Importance:**
  The Ensemble model readily reveals which non-sequential features (e.g., total days out, high-value transfer flags) drive the majority of the predictive decision.
- **Robust Baseline:**
  Its calculated RMSE provided the strict threshold that the Optimized LSTM model needed to surpass in Week 7 to justify the added computational complexity of deep learning.

2. Execution and Prediction Generation

Both models were trained using the player features model all imputed.csv dataset and then used to generate predictions on the held-out test set.

The predictions from these models (lstm_preds and ensemble_preds) were compiled alongside the true scaled values (y_test) into a unified output file, week6_predictions.csv, enabling direct quantitative comparison.

3. Baseline Performance Metrics

The performance was evaluated using Root Mean Squared Error (RMSE) on the normalized validation/test set, where a lower score indicates better predictive accuracy.

The initial LSTM model from the development phase had a validation RMSE of **0.1168**. The Week 6 execution confirmed the following baselines:

This narrowly outperformed the standalone Tuned Multivariate LSTM's RMSE of 0.1168. The marginal improvement confirmed the value of the ensemble approach, validating that the optimal model structure integrates both the temporal prediction (LSTM) and the feature interaction analysis (XGBoost).

4. Conclusion for Week 6

The core takeaway from Week 6 was the confirmation of the modeling baseline and the specific challenge ahead:

- **Baseline Established:**
  The maximum predictive efficiency achieved with initial configurations stood at **0.1168 RMSE**.

Week 7: Model Tuning and Optimization

The core objective was to take the **Ensemble Model** developed in Week 6 and optimize its components (the LSTM and XGBoost Regressor) to achieve the best possible performance for predicting the target variable (**Overall Rating**).

1. Data Preparation and Feature Encoding

The notebook confirms the data preparation steps from previous weeks:

- **Categorical Encoding:** The attacking work rate and defensive work rate were encoded from categorical strings (low, medium, high) to numerical values (0, 1, 2).
- **Time-Series Input:** The data was transformed into time-step sequences () for the LSTM model.
- **Feature Count:** A comprehensive set of **features** were used as input for the models.

2. Hyperparameter Tuning Implementation

Structured tuning was successfully implemented for both components of the final ensemble:

A. LSTM Tuning

This tuning focused on optimizing the deep learning model's core architecture and learning process.

Objective: Minimize the Validation Loss

Search Space: Key hyperparameters tuned included the number of LSTM units (32 to 128), regularization via the Dropout rate (0.0, 0.2, 0.3), and the Learning Rate (10^{-2},10^{-3},10^{-4}).

Best Configuration Found: The best standalone LSTM achieved its optimal performance with Units:96 and a minimal Dropout:0.0, using a Learning Rate:0.0001.

B. XGBoost Tuning (Using RandomizedSearchCV)

This process optimized the tree-based model, particularly its role within the ensemble framework.

Objective: Maximize the score, achieved by minimizing RMSE (scoring="neg_mean_squared_error").

Ensemble Strategy: The output predictions from the Tuned LSTM were intentionally added as an extra input feature (X_train_ensemble) to the XGBoost model. This technique allows XGBoost to effectively function as a meta-learner, correcting errors and learning the residuals of the LSTM's predictions.

Result: A wide grid of parameters, including n_estimators and max_depth, was evaluated. The best XGBoost model parameters were successfully identified, favoring a lower learning rate for stability.

This phase, corresponding to Milestone 6, executed the final quality assurance and optimization of the predictive pipeline. The goal was to refine the models to achieve the lowest possible prediction error and formally compare all final model iterations.

Summary of Final Findings

The project successfully executed the entire planned methodology, culminating in the Tuned Ensemble (LSTM + XGBoost) as the quantitatively best model for predicting Overall Rating.

Week 8: Final Integration, Analysis, and Project

Conclusion

Week 8 served as the capstone of the 8-week project, moving beyond iterative development to a comprehensive final analysis, model validation, and the preparation of deployment-ready deliverables. The central objectives were to definitively compare the performance of the Optimized LSTM Model (Week 7) against the initial baselines (Week 6), interpret the final feature contributions, and synthesize the project's analytical success.

1. Model Validation and Final Performance Metrics

The ultimate test of the predictive system was its performance on a held-out, unseen dataset. The final validation metrics confirmed the efficacy of the sequential deep learning approach and the comprehensive feature engineering strategy.

The final deliverables bundled all processed data and modeling assets for future use and deployment:

- best_lstm_model.h5 — The final sequential prediction engine
- week6_xgboost_model.pkl — Retained for competitive benchmarking and interpretability
- player_features_model_all_imputed.csv — The source of truth for the final analysis
- Final_Prediction_Streamlit_App.py — The front-end code for TransferIQ

Deployment Part

1. Final Data Consolidation for Deployment

The primary technical task was to synthesize the complex, analytical data—which was spread across multiple stages and file formats—into one simple, deployable CSV file. This merged file serves as the source of truth for the application interface.

A. Consolidation Process

The script **Player_Market_Value_Prediction_Trained_Dataset_Coding.py** orchestrated the merging of four distinct data assets using the normalized Player Name key:

1. **Imputed Features** (Player names from the 1,148-player set).
2. **Core Player Attributes** (Age, Position, Injury Status).
3. **Sentiment Data** (Sentiment Label).
4. **Prediction Results** (Normalized predictions and inverse-scaled market values from the Week 6 models).

B. Final Dataset Structure

The resulting dataset, **Player_Market_Value_Prediction_Dataset.csv**, contained all essential inputs and outputs in a single row per player for easy lookup in the application:

- **Profile Columns:** Player Name, Age, Injury Status, Sentiment Label.
- **Final Output Column:** Market Value (M) (The final predicted and averaged value).
- **Transparency Columns:** lstm_market_value and ensemble_market_value (The individual model predictions).

2. Model Integration and Prediction Logic

The script load player features.py defines the logic for how the trained models and necessary transformation pipelines would function in a live prediction environment, ensuring consistency with the training data flow.

A. Model and Scaler Loading

The prediction environment is instantiated by loading the production-ready artifacts:

- **Model:** The weights of the optimized LSTM model.

B. Live Prediction and Inverse Transformation

The prediction routine ensures that new player data adheres precisely to the preprocessing steps used during training:

- **Feature Collection:** For a selected player, all required features are looked up and combined. Missing features are implicitly imputed with 0.0, mirroring the Week 3 imputation strategy.
- **Scaling and Prediction:** The input is scaled (scaler.transform) and passed to the loaded model.predict() function.
- **Value Calculation:** The model's output (a value between 0 and 1) is converted back to a practical monetary figure using scaler.inverse_transform().
- **Final Averaged Value:** The final deployed value (Market Value (M)) is calculated as the **mean of the LSTM and Ensemble predictions**, combining the strengths of both models for a robust prediction.

3. Application Interface Design (TransferIQ)

The app.py script defines the structure of the **TransferIQ** dashboard, the final user-facing deliverable built using **Streamlit**.

A. User Interface and Navigation

- Title: **Transfer IQ Player Market Value**
- **Functionality:** Users interact with a sidebar menu, selecting a player to instantly load their predicted valuation and profile data.

B. Key Metrics Displayed

The dashboard prioritizes clarity and immediate financial context:

- **Predicted Value (Metric Cards):** The final predicted value is prominently displayed in two synchronized formats:
  - **USD Value (Millions):** $XXX.XX M
  - **INR Value (Lakhs/Crores):** The predicted USD value is converted to Indian Rupees (INR) using a fixed exchange rate of 88.73 for local relevance (USD Value×1,000,000×88.73).
- **Player Profile:** Provides immediate context for the prediction, displaying the qualitative feature inputs: **Age**, **Injury~Status**, and the derived **Sentiment~Label**.
- **Prediction Breakdown:** For transparency, the interface breaks down the final value into the raw output of the primary models, allowing users to compare the LSTM prediction against the Ensemble prediction.

Thank You