**Infosys Springboard 6.0 Internship**

**Comprehensive Technical Report**

****

**Dynamic Player Transfer Value Prediction Using AI and Multi-source Data**

Himanshu Saxena

Infosys Springboard 6.0 Internship

Mentor: Ms. Pranaya

Submission Date: 10th October 2025

**Table of Contents**

1. Project Overview ………………………………………………………………………………………………..…… 2
2. System Architecture ……………………………………………………………………………………………..…. 2
3. Data Processing Pipeline ………………………………………………………………………………………… 2  
   3.1 Data Acquisition ……………………………………………………………………………………………….. 2  
   3.2 Database Design and Integration ……………………………………………………………………… 3  
   3.3 Feature Engineering and Aggregation ………………………………………………………………. 3  
   3.4 Sentiment Analysis Pipeline ………………………………………………………………………………. 3
4. Modeling and Machine Learning ……………………………………………………………………………. 4  
   4.1 Sequence Construction and Input Preparation …………………………………………………. 4  
   4.2 LSTM Model Architecture and Training ……………………………………………………………… 4  
   4.3 Encoder-Decoder LSTM for Multi-step Forecasting ……………………………………………. 4  
   4.4 Ensemble Modeling with XGBoost ……………………………………………………………………. 4  
   4.5 Hyperparameter Tuning and Evaluation .……………………………………………………………. 4
5. Streamlit Dashboard Application …………………………………………………………………………..… 5
6. Results, Visual Analytics, and Findings …………………………………………………………………..… 5
7. Challenges, Limitations, and Next Steps ………………………………………………………………..… 6
8. Conclusion …………………………………………………………………………………………………………….… 6
9. Appendix - A: Database design and integration ………………….………………………………….…7
10. Appendix - B: List of MySQL tables and usage ………………….…………………………………..…11
11. Appendix - C: Notebooks, Code, and Demonstration Links ………………………………………15

**1. Project Overview**

TransferIQ is an integrated data science platform developed to dynamically predict professional football player transfer values. The prediction task uses a combinatorial, multi-source approach, leveraging performance, market, injury, and sentiment data with advanced machine learning—including sequence deep learning and ensemble methodologies—delivered in a user-friendly dashboard.

**2. System Architecture**

The pipeline is modular, encompassing ETL scripts, feature engineering modules, model definitions, and a cloud-ready Streamlit application (ensemble\_predictions\_app.py, dashboard\_integrated\_app.py), supported by a MySQL data warehouse.

* **Raw Data:** Collected from open data repositories and web scraping.
* **Database:** Structured schema for players, clubs, market values, injuries, and features.
* **Feature Engineering:** Modular scripts (merge-features.py, merge-sentiments.py) create high-value, rich feature spaces.
* **Modeling:** Both classical and deep learning models, in isolation and ensemble, are used.
* **UI:** Streamlit dashboard exposes the full prediction and evaluation experience.

**3. Data Processing Pipeline**

**3.1 Data Acquisition**

* **Performance Data:**
  + *StatsBomb* event and lineup files (import\_statsbomb\_mysql.py, import\_statsbomb\_lineup\_batch.py): Over 12M records across 3,464 files parsed and loaded.
* **Market Values & Transfers:**
  + *Transfermarkt* scrapes over 1500 players from 10 competitions (transfermarkt\_loop.py, scrape\_trfr\_record\_new.py): Multi-competition, multi-player web scraping with robust pagination and data normalization.
* **Injury & Team Performance:**
  + Multi-page scraped from Transfermarkt, based on Transfermarkt player ids, stored in MySQL database
* **Sentiment:**
  + Social media (Reddit) NLP-based analysis via multithreaded pipeline (social\_sentiment\_multithread\_with\_log.py), followed by mapped integration.

**3.2 Database Design and Integration**

* **Normalization:**
  + Data mapped between sources (Transfermarkt and Statsbomb) using both direct key and fuzzy logic (e.g., **rapidfuzz** for player name mapping via map\_players.py and auto\_player\_mapping.py).
* **Data Engineering Scripts:**
  + merge-features.py and merge-sentiments.py aggregate, validate, and create the centralized “player\_features” table in MySQL, summarizing market values, injuries, transfer stats, sentiment, and performance metrics for each player.

**3.3 Feature Engineering and Aggregation**

* **Performance Trends:**
  + Slope and window-based aggregation from time-series event features.
* **Market Value Growth:**
  + Computed as the difference between a player's maximum and minimum recorded value and past transfers market values.
* **Injury Signals:**
  + Total injuries, average days out, recency, days since last injury, recent injury indicators.
* **Sentiment Features:**
  + Aggregates from social sentiment (mean polarity, positive/negative ratio, sentiment trend via regression).

**3.4 Sentiment Analysis Pipeline**

* **Twitter:**
  + Access via PRAW, using multi-threading for scale.
  + NLP: TextBlob for polarity extraction.
* **Bulk Insert & Logging:**
  + ETL logs each run, upserts batch results, creates specific “sentiments” tables in MySQL.
* **Trend Calculation:**
  + Regression on sentiment trajectory to produce a “sentiment trend” feature per player (merge-sentiments.py).

**4. Modeling and Machine Learning**

**4.1 Sequence Construction and Input Preparation**

* Sequences built per player using windowed time series (make\_multi\_variate\_multi\_step\_array in modeling scripts).
* Scaled feature vectors using MinMaxScaler for network compatibility.

**4.2 LSTM Model Architecture and Training**

* Implemented with Keras Sequential API (see definition in ensemble\_predictions\_app.py, encoder\_decoder\_multi\_app.py).
* Network:
  + One or more LSTM layers, dropout regularization, and dense output for sequence prediction.
* Model tuning:
  + Early stopping, batch size, latent dimensions, and learning rate sweep.

**4.3 Encoder-Decoder LSTM for Multi-step Forecasting**

* Full sequence-to-sequence (seq2seq) model structure in encoder\_decoder\_multi\_app.py.
* Composed of two LSTM blocks: Encoder compresses the history, Decoder outputs multi-step forecasts.
* Useful for recursive, windowed prediction (predicting multiple transfer windows).

**4.4 Ensemble Modeling with XGBoost**

* Ensemble combines LSTM outputs as features for an XGBoost regressor.
* Meta-feature vector: Flattened sequence past + LSTM predictions stacked for tabular boosting.
* XGBoost trained per time step ahead; ensemble averaging for final forecast.

**4.5 Hyperparameter Tuning and Evaluation**

* Dashboard-integrated controls for LSTM (latent\_dim, batch\_size, learning\_rate) and XGBoost (n\_estimators, max\_depth, learning\_rate, subsample).
* Real-time charting of RMSE and validation loss.
* Grid/random search implemented with results stored for reproducibility.

**5. Streamlit Dashboard Application**

* **Interactive UI:**
  + Player selection, sequence/window/epoch adjustment, live retraining, and forecast visualization.
* **Feature and Model Comparison:**
  + True vs predicted value charts, direct error metrics, and model (LSTM/XGBoost/ensemble) overlays.
* **Feature Influence:**
  + Visual comparisons of selected player features against cohort mean/max.
* **Multi-step Forecasts:**
  + Per-player, per-step graphical forecast for scenario planning.
* **Download:**
  + Model artifacts, parameters, and results available for export.

**6. Results, Visual Analytics, and Findings**

* **Modeling Success:**
  + Multivariate LSTM outperforms univariate (see “more\_features” images).
  + Ensemble (LSTM + XGBoost) delivers lowest RMSE and highest reliability.
* **Explainability:**
  + Feature trends (injury, sentiment, minutes played, etc.) have direct, interpretable impact—shown in feature tables and plots.
* **Generalization:**
  + New players (not in training set) predictions enabled; dashboard compares cohort metrics.
* **Reports and Screenshots:**
  + Images illustrate all major findings:
    - Model tuning tables and progress curves
    - Forecast comparisons (true vs predicted individual, multi-step, ensemble)
    - UI screenshots from the live dashboard

**7. Challenges, Limitations, and Next Steps**

* **Data Quality:**
  + Scraping/merging from multiple sources required extensive validation, fuzzing, and re-scraping for consistency.
* **Model Drift:**
  + Market trends can shift abruptly, which may require scheduled retraining.
* **Future Upgrades:**
  + Expand features (e.g., advanced contract modeling, richer social sentiment sources)
  + Explore transformer-based time sequence models
  + Systematize MLOps (integrate CI/CD for new models, dashboards, and alerts)

**8. Conclusion**

TransferIQ is a deeply engineered, end-to-end AI system for player market value forecasting, blending established statistical, ML, and deep learning practices in an open, auditable, and cloud-ready manner. The solution is visually and interactively presented to maximize transparency, usability, and impact for decision-makers in football analytics and business operations.

**Appendix – A  
Database design and integration**

**StatsBomb Data Ingestion & Event Tables**

**Scripts:**

* import\_statsbomb\_mysql.py
* import\_statsbomb\_lineup\_batch.py
* import\_lineups\_folder.py
* import\_matches\_statsbomb\_mysql.py

**MySQL Tables:**

* **eventsNew**1:  
  Store parsed event data from StatsBomb event JSONs for each match and batch, such as player actions, timestamps, event type, locations, related events.
* **matches**:  
  Contains match details—match ID, teams, scores, stadium, referee, date—imported from matches JSON.
* **competitions**:  
  Stores competition metadata such as league/cup name, country, gender, season, and match availability.
* **teams**:  
  Stores team information from lineup files—teamid, teamname.
* **players**:  
  Stores player details—playerid, playername/nickname, jersey number, country.
* **lineuppositions**:  
  Holds player position and substitutions/movements for each match.
* **playercards**:  
  Records yellow/red cards per player per match, with type, period, reason.

**Transfermarkt Data & Mappings**

**Scripts:**

* import\_transfermrkt\_normalize.py
* merge-features.py
* auto\_player\_mapping.py
* map\_players.py

**MySQL Tables:**

* **playerstrfrmrkt**:  
  Master list of players scraped and normalized from Transfermarkt.
* **clubstrfrmrkt**:  
  List of clubs referenced in player/career/value records.
* **competitionstrfrmrkt**:  
  List of competitions/leagues for reference integrity and mapping.
* **marketvaluestrfrmrkt**:  
  History of player market value snapshots (player, club, competition, market value, date).
* **playerinjuriestrfrmrkt**:  
  Injury records for scraped players—dates, description, time out, links to player IDs.
* **playertransferhistory**:  
  Transfer records—fees, types, clubs, date, reasons for moves.
* **playermapping**:  
  Fuzzy/Direct mapping between StatsBomb and Transfermarkt (IDs, canonical name, match confidence/type).

**Social Sentiment Data**

**Scripts:**

* social\_sentiment\_multithread\_with\_log.py
* merge-sentiments.py
* reddit\_sentiment.py

**MySQL Tables:**

* **twittersentiments**:  
  Tweets collected for players—playername, tweetid, tweet timestamp, content, sentiment class, polarity score.
* **redditsentiments**:  
  Reddit posts/comments for player—playername, postid, subreddit, title, selftext, sentiment label, polarity score.
* **mediumsentiments**:  
  Medium articles, if scraped for sentiment pipeline.
* **sentimentrunlog**:  
  Log of sentiment collection runs (for batch audits).

**Aggregated Features & Final Modeling Tables**

**Scripts:**

* merge-features.py
* merge-sentiments.py
* ensemble\_predictions\_app.py
* encoder\_decoder\_multi\_app.py
* hyper\_parameter\_tuning\_app.py

**MySQL Tables:**

* **playerfeatures**:  
  Aggregates all engineered features for each player, including market value stats, injury metrics, transfer history, sentiment summary, cards, performance trends, positions played.
* **hyperparameterresults**:  
  Stores best results (parameters, RMSE, validation loss) for LSTM and XGBoost/ensemble random/grid searches and tuning sessions.

**Table Uses Summary:**

| **Table Name** | **Main Use / Script(s)** |
| --- | --- |
| eventsNew1 | Store event-level player actions, parsed from StatsBomb events JSON |
| lineups | Store lineup/player positional data, from StatsBomb lineup batch import |
| matches | Store high-level match records, from matches JSON import |
| competitions | Store competition/league metadata |
| teams, players | Store team and player master data |
| lineuppositions | Tracks player movements, substitutions, tactical shifts |
| playercards | Store yellow/red card events per match |
| playerstrfrmrkt | Normalized player master list from Transfermarkt |
| clubstrfrmrkt | Master list of clubs from Transfermarkt scraping |
| competitionstrfrmrkt | Master list of competitions/leagues |
| marketvaluestrfrmrkt | Historical market values for players |
| playerinjuriestrfrmrkt | Injury records by player |
| playertransferhistory | Transfer summary (dates, clubs, fees, types) |
| playermapping | Crosswalk between StatsBomb and Transfermarkt player IDs |
| twittersentiments | NLP-processed Twitter posts/tweets scored for sentiment |
| redditsentiments | NLP-processed Reddit posts/comments scored for sentiment |
| mediumsentiments | NLP-processed Medium articles for sentiment |
| sentimentrunlog | Batch run logs for sentiment scraping |
| playerfeatures | Final aggregated engineered features table for each player |
| hyperparameterresults | Results/parameters of random/grid search for model tuning |

**Appendix - B:  
List of MySQL tables and usage**

**eventsNew1**:

* Main Use / Data Stored: Player-level event records for all matches, including actions, timestamps, types, and related metadata.
* Created/Used By (Scripts/Modules): import\_statsbomb\_mysql.py, import\_statsbomb\_lineup\_batch.py
* Important Columns/Keys: id (PK), filename, matchid, indexno, period, timestamp, type, player, team, locationx, locationy, relatedevents

**matches:**

* Main Use / Data Stored: Stores high-level match details such as teams, scores, stadium, referee, and date.
* Created/Used By (Scripts/Modules): import\_matches\_statsbomb\_mysql.py
* Important Columns/Keys: matchid (PK), competitionid (FK), seasonid, matchdate, hometeam, awayteam, homescore, awayscore, stadium, referee

**competitions:**

* Main Use / Data Stored: Holds league/cup metadata: competition name, country, season, gender, and match availability.
* Created/Used By (Scripts/Modules): import\_matches\_statsbomb\_mysql.py
* Important Columns/Keys: id (PK), filename, countryname, competitionname, competitiongender, seasonid

**teams:**

* Main Use / Data Stored: Master list of teams referenced in StatsBomb and transfer data.
* Created/Used By (Scripts/Modules): import\_lineups\_folder.py, import\_statsbomb\_lineup\_batch.py
* Important Columns/Keys: teamid (PK), teamname

**players:**

* Main Use / Data Stored: Master player list containing playerid, names, nicknames, jersey number, and country.
* Created/Used By (Scripts/Modules): import\_lineups\_folder.py, import\_statsbomb\_lineup\_batch.py
* Important Columns/Keys: playerid (PK), playername, playernickname, jerseynumber, countryid, countryname

**lineuppositions:**

* Main Use / Data Stored: Tracks substitutions, tactical shifts, positions played over time for each player in a match.
* Created/Used By (Scripts/Modules): import\_lineups\_folder.py
* Important Columns/Keys: id (PK), playerid (FK), teamid (FK), positionid, positionname, fromtime, totime, fromperiod, toperiod, startreason, endreason

**playercards:**

* Main Use / Data Stored: Stores yellow/red card events per player for each match.
* Created/Used By (Scripts/Modules): import\_statsbomb\_lineup\_batch.py
* Important Columns/Keys: playerid (FK), matchid (FK), cardtype, period, reason

**players\_trfrmrkt:**

* Main Use / Data Stored: Master list for Transfermarkt player entries.
* Created/Used By (Scripts/Modules): import\_transfermrkt\_normalize.py
* Important Columns/Keys: id (PK), name

**clubs\_trfrmrkt:**

* Main Use / Data Stored: Master list for clubs found in player/market value records.
* Created/Used By (Scripts/Modules): import\_transfermrkt\_normalize.py
* Important Columns/Keys: id (PK), name

**competitions\_trfrmrkt:**

* Main Use / Data Stored: Master list of competitions/leagues for reference and mapping.
* Created/Used By (Scripts/Modules): import\_transfermrkt\_normalize.py
* Important Columns/Keys: id (PK), name

**marketvalues\_trfrmrkt:**

* Main Use / Data Stored: Historical timeline of player market values (player, club, competition, value, date).
* Created/Used By (Scripts/Modules): import\_transfermrkt\_normalize.py, merge-features.py
* Important Columns/Keys: id (PK), playerid (FK), clubid (FK), competitionid (FK), marketvalue, snapshotdate

**player\_injuries\_trfrmrkt:**

* Main Use / Data Stored: Stores injury records for players, including type, date range, and time missed.
* Created/Used By (Scripts/Modules): import\_transfermrkt\_normalize.py, merge-features.py
* Important Columns/Keys: id (PK), playerid (FK), start\_date, end\_date, days\_out, injurydesc

**player\_transfer\_history:**

* Main Use / Data Stored: Player transfers: from/to clubs, transfer fee/type, date, and description/history.
* Created/Used By (Scripts/Modules): scrape\_trfr\_record\_new.py
* Important Columns/Keys: id (PK), playerid (FK), fromclub, toclub, fee, date, type, description

**player\_mapping:**

* Main Use / Data Stored: Fuzzy and explicit mapping between StatsBomb and Transfermarkt player IDs; confidence score.
* Created/Used By (Scripts/Modules): auto\_player\_mapping.py, map\_players.py
* Important Columns/Keys: statsbombplayerid, trfrmrktplayerid, canonicalname, matchtype, confidence

**reddit\_sentiments:**

* Main Use / Data Stored: Reddit posts/comments for each player, analyzed for sentiment/polarity.
* Created/Used By (Scripts/Modules): reddit\_sentiment.py, merge-sentiments.py
* Important Columns/Keys: id (PK), playername, postid, subreddit, createdat, title, selftext, sentiment, polarity

**sentiment\_run\_log:**

* Main Use / Data Stored: Logs each sentiment scraping run (batch audits, status, date, platform).
* Created/Used By (Scripts/Modules): social\_sentiment\_multithread\_with\_log.py
* Important Columns/Keys: runid (PK), date, platform, status

**player\_features:**

* Main Use / Data Stored: Final engineered feature vectors for players (for ML modeling and dashboard).
* Created/Used By (Scripts/Modules): merge-features.py
* Important Columns/Keys: playerid (PK), latestmarketvalue, marketvaluegrowth, totalinjuries, avgdaysout, lastinjurydate, totaltransfers, totaltransferfees, freetransfers, sentimentmean, sentimentpositiveratio, sentimenttrend, avgcardspermatch, positionsplayed, currentclubid

**hyperparameter\_results:**

* Main Use / Data Stored: Stores model tuning (LSTM/XGBoost/ensemble) best parameters, RMSE, validation loss, metadata.
* Created/Used By (Scripts/Modules): hyper\_parameter\_tuning\_app.py, dashboard apps
* Important Columns/Keys: id (PK), modeltype, parameters, rmse, val\_loss, timestamp

**Appendix C  
Notebooks, Code, and Demonstration Links**

* **Source Code:**
  + ETL and DB: import\_statsbomb\_mysql.py, import\_statsbomb\_lineup\_batch.py, merge-features.py, merge-sentiments.py
  + Mapping/Utility: auto\_player\_mapping.py, map\_players.py
  + Sentiment: social\_sentiment\_multithread\_with\_log.py, reddit\_sentiment.py
  + Modeling: ensemble\_predictions\_app.py, encoder\_decoder\_multi\_app.py, ensemble\_app.py
* **Notebooks:**
  + EDA & Modeling: univariate.ipynb, multivariate\_multi\_step.ipynb
* **Visuals:**
  + All charts and images referenced in this report
* **Dashboard Demo Video:**
  + [YouTube link](https://youtu.be/vkRj8UAwzxU)
* **Presentation Slides:**
  + TransferIQ-Presentation-Himanshu-Saxena-Infosys-Springboard-6.0.pptx