**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. System Architecture
3. Data Processing Pipeline  
   3.1 Data Acquisition  
   3.2 Database Design and Integration  
   3.3 Feature Engineering and Aggregation  
   3.4 Sentiment Analysis Pipeline
4. Modeling and Machine Learning  
   4.1 Sequence Construction and Input Preparation  
   4.2 LSTM Model Architecture and Training  
   4.3 Encoder-Decoder LSTM for Multi-step Forecasting  
   4.4 Ensemble Modeling with XGBoost  
   4.5 Hyperparameter Tuning and Evaluation
5. Streamlit Dashboard Application
6. Results, Visual Analytics, and Findings
7. Challenges, Limitations, and Next Steps
8. Conclusion
9. Appendix: Notebooks, Code, and Demonstration Links

**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 supported.
* **API/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 (transfermarkt\_loop.py, scrape\_trfr\_record\_new.py): Multi-competition, multi-player web scraping with robust pagination and data normalization.
* **Injury & Team Performance:**
  + Supplemented with curated Kaggle datasets for English Premier League injury events.
* **Sentiment:**
  + Social media (Twitter, Reddit, Medium) 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 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 “playerfeatures” 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.
* **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 & Reddit:**
  + Access via Tweepy and 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 (makemultivariatemultisteparray 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.

**9. Appendix: 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