# Infosys Springboard 6.0 Internship

# Title: Dynamic Player Transfer Value Prediction using AI and Multi-Source Data

Project Report

By Nishant Gupta

# Abstract

The professional sports transfer market is a complex ecosystem where accurately estimating a player's value is crucial for team success. This project, "TransferIQ," introduces a sophisticated AI-driven model to dynamically predict player transfer values by integrating multi-source data. The proposed methodology utilizes advanced machine learning and time-series forecasting to analyze player performance statistics, historical market data from Transfermarkt, injury records, and public sentiment extracted from social media .

The core of the model involves using Long Short-Term Memory (LSTM) networks to capture temporal dependencies in player performance and market trends. This is enhanced by Natural Language Processing (NLP) techniques, such as VADER, to quantify public perception. Finally, an ensemble model using XGBoost or LightGBM will combine these diverse data streams to generate a final, robust transfer value prediction. The expected outcome is a dynamic and accurate system for forecasting player market values, contributing new insights to the field of sports analytics and player valuation .

# Technologies and Tools

**Data Sources and Collection**

- Player Performance Data: StatsBomb Open Data
- Market Value Data: Transfermarkt, collected via web scraping techniques
- Social Media Sentiment: Twitter API
- Injury Data: Historical player injury records

**Machine Learning Models and Techniques**

- Time-Series Forecasting: Long Short-Term Memory (LSTM) networks, including univariate, multivariate, and encoder-decoder structures
- Ensemble Methods: XGBoost or LightGBM
- Data Preprocessing: Techniques include handling missing values, scaling numerical data, and performing one-hot encoding for categorical variables

**Natural Language Processing (NLP)**

- Libraries: VADER or TextBlob are used to perform sentiment analysis

**Evaluation and Tuning**

- Evaluation Metrics: Model performance is measured using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2)
- Hyperparameter Tuning: Grid search or random search methods are used to optimize model parameters

standard libraries used in deployment scripts for managing file paths and loading data.

# Week-wise report

# Week 1 – Data Exploration and Collection

## Overview

The first week focused on establishing a solid data foundation for the football analytics project. The main objective was to identify, collect, and validate multi-domain datasets representing player performance, injury risk, market valuation, and public sentiment. Each dataset underwent initial exploratory data analysis (EDA) to understand its structure, completeness, and consistency, ensuring readiness for the later modeling phases.

Dataset Collection and Preparation

Four primary data sources were explored and standardized:

1. Player Performance Data (StatsBomb Open Data)
    - Extracted event-level football data detailing passes, shots, and defensive actions.
    - Flattened nested JSON files into Csv files for efficient processing.
    - Verified competition and match data integrity for use in feature engineering.
2. Player Injury Data
    - Compiled injury records covering multiple seasons.
    - Cleaned and validated information on injury type, duration, and recovery period.
    - Derived indicators such as injury count, total days out, and severity index.
3. Social Media Sentiment Data
    - Collected posts reflecting public perception and fan sentiment through reddit .
    - Cleaned raw text, standardized sentiment labels, and prepared sentiment scores for players.
4. Market Value and Transfer Data

- Scraped player market valuations and transfer information from trusted sources.
- Cleaned and formatted data to serve as the target variable for predictive modeling.

**Output**

All datasets were cleaned, standardized, and exported to structured CSV files for downstream processing:

- Statsbomb_opendata.csv
- Injury_data.csv
- Market-value-data.csv
- player_sentiment_analysis.csv

# Week 2 : Data Cleaning and Pre-processing

The goal is simple: take the four messy, raw datasets and turn them into clean, structured tables that are ready for the advanced feature engineering I have planned for Weeks 3 and 4.

Here's how I'll tackle each part of the process:

## 1. Data Cleaning

This is where I handle the most obvious problems in the datasets. I'll go through each of the four files (Performance, Market Value, Injury, and Reddit data) and perform these crucial steps:

- Handling Missing Values: I'll look for empty cells. For numerical data, like minutes_played, I might fill them with 0. For other stats, I might use the average value for that column. The strategy will depend on the feature, but the goal is to have no empty spots.
- Fixing Inconsistencies: The biggest challenge here is standardizing player names. I might have "Lionel Messi" in one file and "L. Messi" in another. I'll write a script to standardize these names across all datasets so I can merge them accurately later. I'll also check for consistent data types (e.g., making sure all dates are in the same format).
- Removing Duplicates: I'll check for and remove any rows that are exact copies to make sure my data isn't skewed.

2. Data Transformation: Structuring for the Model

Once the data is clean, I need to reshape it so it’s optimal for machine learning algorithms.

- Scaling Numerical Data: I'll take all my numerical columns (like goals, assists, market value, etc.) and scale them. I'll likely use a MinMaxScaler, which transforms every value to be between 0 and 1. This is critical because it prevents features with large ranges (like market value) from overpowering features with small ranges (like goals per game).
- Encoding Categorical Data: My models don't understand text like "Defender" or "Midfielder." I'll use one-hot encoding to convert these categories into numerical columns. For example, the "Position" column will be replaced by new columns like is_defender, is_midfielder, etc., with values of 1 or 0.

3. Prepping the Reddit Data for Sentiment Analysis

The text data from Reddit needs special treatment before I can analyze it for sentiment in the next phase. My process will be:

- Removing Noise: I'll strip out all the irrelevant parts of the comments and posts, like URLs, Reddit-specific formatting (like u/username), and any special characters.
- Standardizing Text: I'll convert all text to lowercase to ensure "Goal" and "goal" are treated as the same word.
- Removing Stop Words: I'll filter out common English words that don't add much meaning, such as "the," "a," "is," and "in." This helps the sentiment analysis tool focus on the words that actually carry emotional weight.

# Week 3 - 4 : Advance Feature Engineering and Sentiment Analysis

- Create Trend Features: A player's current form is vital. I'll create features that capture their recent performance, such as a 3-match rolling average for goals or the standard deviation of their assists over the last 5 matches. This will help the model understand if a player is improving or declining.

- Develop an Injury Impact Score: Using the scraped injury data, I'll engineer a new feature, maybe called `injury_risk_score`. This could be a combination of the total number of days missed in the last season and the frequency of injuries.

- Quantify Public Sentiment: Now that the Reddit text is clean, I'll run it through a sentiment analysis library like VADER. This will process every comment and post about a player to generate a single `sentiment_score`, a numerical value representing their public perception.

- **Build the Master Dataset:** This is the grand finale of this phase. I'll merge all four processed datasets—performance, value, injury, and sentiment—into a single, wide `master_dataset.csv` file, aligned by player name and id.

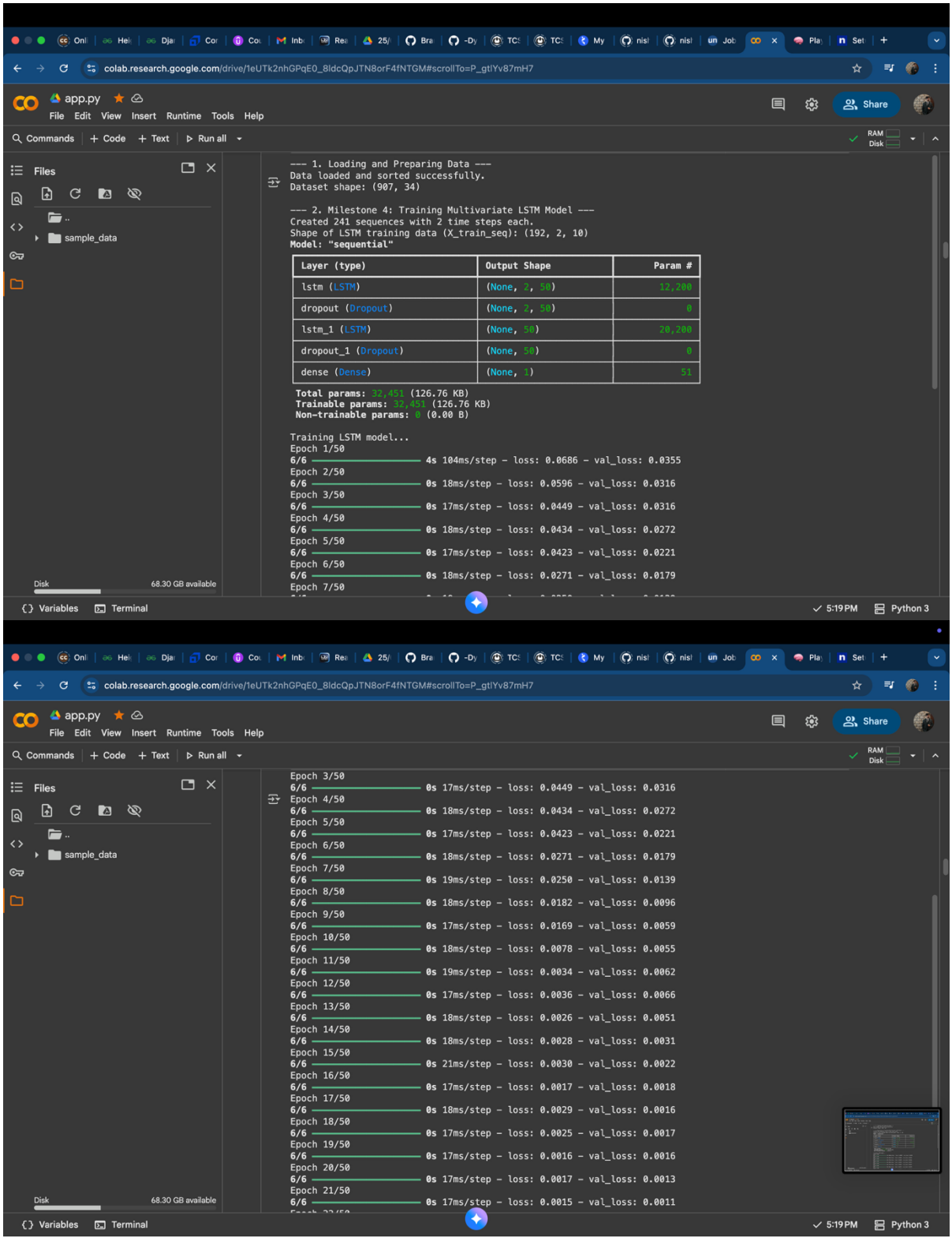
## Week 5 : LSTM Model Development for Time-series

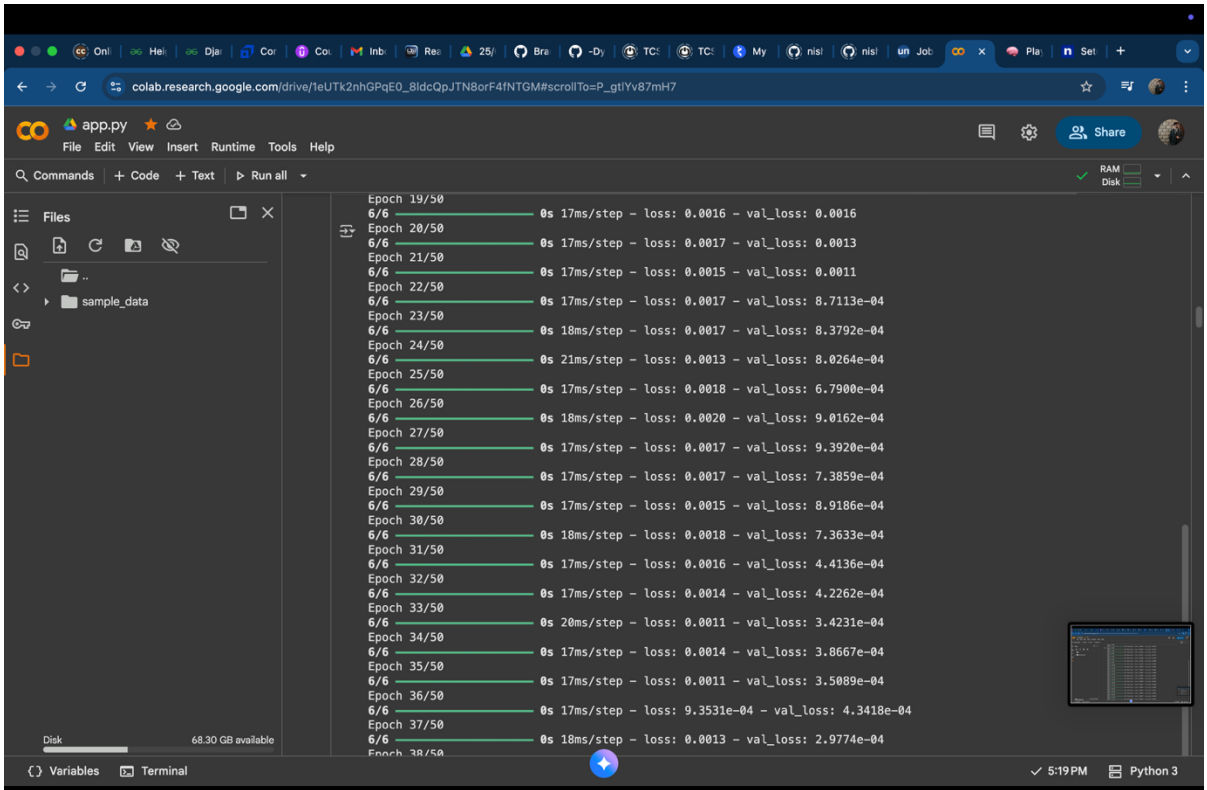
- Structure the Data for Time-Series: I'll format my data into sequences, where each sample consists of a player's stats over a series of, say, 10 matches, with the target being their market value after the 11th match.

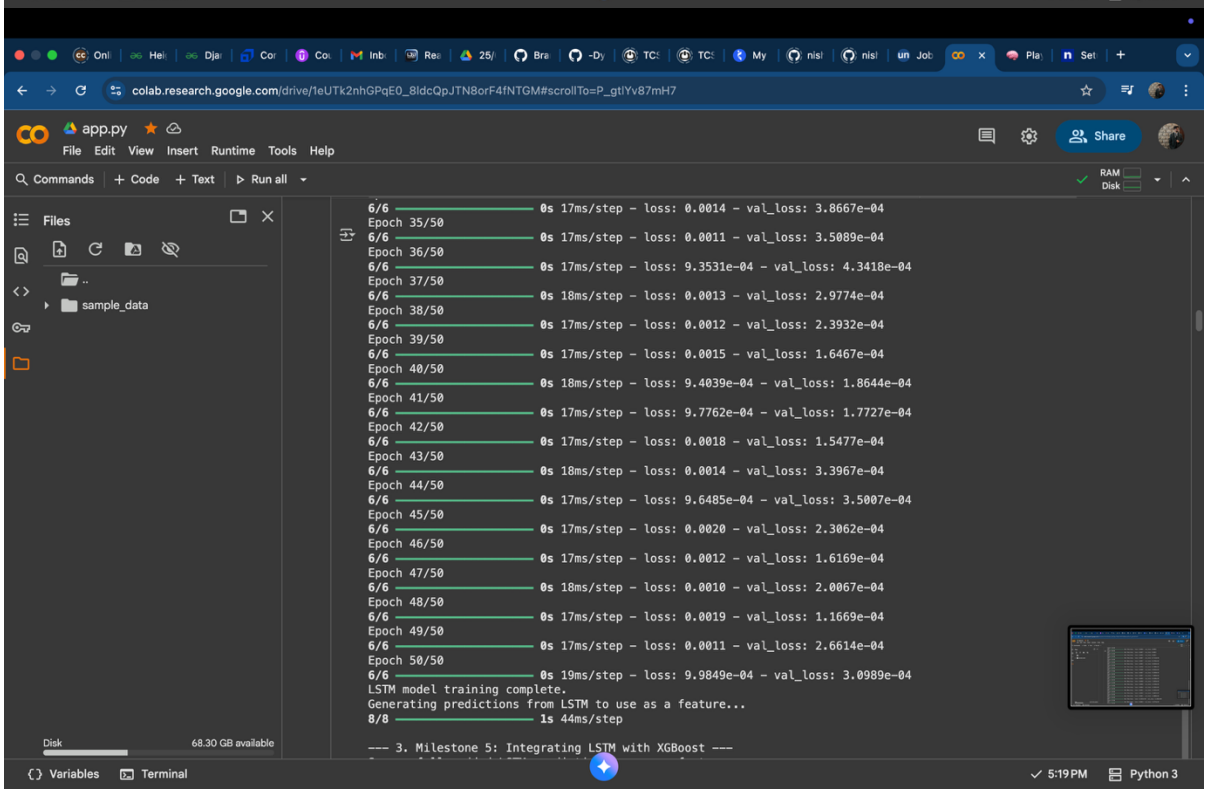
- Build a Multivariate LSTM: I won’t just look at one stat over time. I'll build a multivariate model that considers performance, injury status, and sentiment scores simultaneously to predict future market value.

- Train and Evaluate: I'll train the LSTM model and do an initial evaluation to see how well it's capturing the temporal patterns in the data.

Completed 50 epochs for model training and step-loss and value-loss :


colab.research.google.com/drive/1eUTk2nhGPqE0_8ldcQpJTN8orF4fNTGM#scrollTo=P_gtlYv87mH7
app.py
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
Files
..
sample_data
Disk 68.30 GB available
--- 1. Loading and Preparing Data ---
Data loaded and sorted successfully.
Dataset shape: (907, 34)
--- 2. Milestone 4: Training Multivariate LSTM Model ---
Created 241 sequences with 2 time steps each.
Shape of LSTM training data (X_train_seq): (192, 2, 10)
Model: "sequential"
Layer (type) Output Shape Param #
lstm (LSTM) (None, 2, 50) 12,200
dropout (Dropout) (None, 2, 50) 0
lstm_1 (LSTM) (None, 50) 20,200
dropout_1 (Dropout) (None, 50) 0
dense (Dense) (None, 1) 51
Total params: 32,451 (126.76 KB)
Trainable params: 32,451 (126.76 KB)
Non-trainable params: 0 (0.00 B)
Training LSTM model...
Epoch 1/50
6/6 4s 104ms/step - loss: 0.0686 - val_loss: 0.0355
Epoch 2/50
6/6 0s 18ms/step - loss: 0.0596 - val_loss: 0.0316
Epoch 3/50
6/6 0s 17ms/step - loss: 0.0449 - val_loss: 0.0316
Epoch 4/50
6/6 0s 18ms/step - loss: 0.0434 - val_loss: 0.0272
Epoch 5/50
6/6 0s 17ms/step - loss: 0.0423 - val_loss: 0.0221
Epoch 6/50
6/6 0s 18ms/step - loss: 0.0271 - val_loss: 0.0179
Epoch 7/50
Variables Terminal
5:19 PM Python 3
Epoch 3/50
6/6 0s 17ms/step - loss: 0.0449 - val_loss: 0.0316
Epoch 4/50
6/6 0s 18ms/step - loss: 0.0434 - val_loss: 0.0272
Epoch 5/50
6/6 0s 17ms/step - loss: 0.0423 - val_loss: 0.0221
Epoch 6/50
6/6 0s 18ms/step - loss: 0.0271 - val_loss: 0.0179
Epoch 7/50
6/6 0s 19ms/step - loss: 0.0250 - val_loss: 0.0139
Epoch 8/50
6/6 0s 18ms/step - loss: 0.0182 - val_loss: 0.0096
Epoch 9/50
6/6 0s 17ms/step - loss: 0.0169 - val_loss: 0.0059
Epoch 10/50
6/6 0s 18ms/step - loss: 0.0078 - val_loss: 0.0055
Epoch 11/50
6/6 0s 19ms/step - loss: 0.0034 - val_loss: 0.0062
Epoch 12/50
6/6 0s 17ms/step - loss: 0.0036 - val_loss: 0.0066
Epoch 13/50
6/6 0s 18ms/step - loss: 0.0026 - val_loss: 0.0051
Epoch 14/50
6/6 0s 18ms/step - loss: 0.0028 - val_loss: 0.0031
Epoch 15/50
6/6 0s 21ms/step - loss: 0.0030 - val_loss: 0.0022
Epoch 16/50
6/6 0s 17ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 17/50
6/6 0s 18ms/step - loss: 0.0029 - val_loss: 0.0016
Epoch 18/50
6/6 0s 17ms/step - loss: 0.0025 - val_loss: 0.0017
Epoch 19/50
6/6 0s 17ms/step - loss: 0.0016 - val_loss: 0.0016
Epoch 20/50
6/6 0s 17ms/step - loss: 0.0017 - val_loss: 0.0013
Epoch 21/50
6/6 0s 17ms/step - loss: 0.0015 - val_loss: 0.0011
Variables Terminal
5:19 PM Python 3


colab.research.google.com/drive/1eUTk2nhGPqE0_8ldcQpJTN8orF4fNTGM#scrollTo=P_gtlYv87mH7
app.py
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
Files
..
sample_data
Epoch 19/50
6/6 0s 17ms/step - loss: 0.0016 - val_loss: 0.0016
Epoch 20/50
6/6 0s 17ms/step - loss: 0.0017 - val_loss: 0.0013
Epoch 21/50
6/6 0s 17ms/step - loss: 0.0015 - val_loss: 0.0011
Epoch 22/50
6/6 0s 17ms/step - loss: 0.0017 - val_loss: 8.7113e-04
Epoch 23/50
6/6 0s 18ms/step - loss: 0.0017 - val_loss: 8.3792e-04
Epoch 24/50
6/6 0s 21ms/step - loss: 0.0013 - val_loss: 8.0264e-04
Epoch 25/50
6/6 0s 17ms/step - loss: 0.0018 - val_loss: 6.7900e-04
Epoch 26/50
6/6 0s 18ms/step - loss: 0.0020 - val_loss: 9.0162e-04
Epoch 27/50
6/6 0s 17ms/step - loss: 0.0017 - val_loss: 9.3920e-04
Epoch 28/50
6/6 0s 17ms/step - loss: 0.0017 - val_loss: 7.3859e-04
Epoch 29/50
6/6 0s 17ms/step - loss: 0.0015 - val_loss: 8.9186e-04
Epoch 30/50
6/6 0s 18ms/step - loss: 0.0018 - val_loss: 7.3633e-04
Epoch 31/50
6/6 0s 17ms/step - loss: 0.0016 - val_loss: 4.4136e-04
Epoch 32/50
6/6 0s 17ms/step - loss: 0.0014 - val_loss: 4.2262e-04
Epoch 33/50
6/6 0s 20ms/step - loss: 0.0011 - val_loss: 3.4231e-04
Epoch 34/50
6/6 0s 17ms/step - loss: 0.0014 - val_loss: 3.8667e-04
Epoch 35/50
6/6 0s 17ms/step - loss: 0.0011 - val_loss: 3.5089e-04
Epoch 36/50
6/6 0s 17ms/step - loss: 9.3531e-04 - val_loss: 4.3418e-04
Epoch 37/50
6/6 0s 18ms/step - loss: 0.0013 - val_loss: 2.9774e-04
Disk 68.30 GB available
Variables Terminal
5:19 PM Python 3


colab.research.google.com/drive/1eUTk2nhGPqE0_8ldcQpJTN8orF4fNTGM#scrollTo=P_gtlYv87mH7
app.py
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
Files
..
sample_data
6/6 0s 17ms/step - loss: 0.0014 - val_loss: 3.8667e-04
Epoch 35/50
6/6 0s 17ms/step - loss: 0.0011 - val_loss: 3.5089e-04
Epoch 36/50
6/6 0s 17ms/step - loss: 9.3531e-04 - val_loss: 4.3418e-04
Epoch 37/50
6/6 0s 18ms/step - loss: 0.0013 - val_loss: 2.9774e-04
Epoch 38/50
6/6 0s 17ms/step - loss: 0.0012 - val_loss: 2.3932e-04
Epoch 39/50
6/6 0s 17ms/step - loss: 0.0015 - val_loss: 1.6467e-04
Epoch 40/50
6/6 0s 18ms/step - loss: 9.4039e-04 - val_loss: 1.8644e-04
Epoch 41/50
6/6 0s 17ms/step - loss: 9.7762e-04 - val_loss: 1.7727e-04
Epoch 42/50
6/6 0s 17ms/step - loss: 0.0018 - val_loss: 1.5477e-04
Epoch 43/50
6/6 0s 18ms/step - loss: 0.0014 - val_loss: 3.3967e-04
Epoch 44/50
6/6 0s 17ms/step - loss: 9.6485e-04 - val_loss: 3.5007e-04
Epoch 45/50
6/6 0s 17ms/step - loss: 0.0020 - val_loss: 2.3062e-04
Epoch 46/50
6/6 0s 17ms/step - loss: 0.0012 - val_loss: 1.6169e-04
Epoch 47/50
6/6 0s 18ms/step - loss: 0.0010 - val_loss: 2.0067e-04
Epoch 48/50
6/6 0s 17ms/step - loss: 0.0019 - val_loss: 1.1669e-04
Epoch 49/50
6/6 0s 17ms/step - loss: 0.0011 - val_loss: 2.6614e-04
Epoch 50/50
6/6 0s 19ms/step - loss: 9.9849e-04 - val_loss: 3.0989e-04
LSTM model training complete.
Generating predictions from LSTM to use as a feature...
8/8 1s 44ms/step
--- 3. Milestone 5: Integrating LSTM with XGBoost ---
Disk 68.30 GB available
Variables Terminal
5:19 PM Python 3

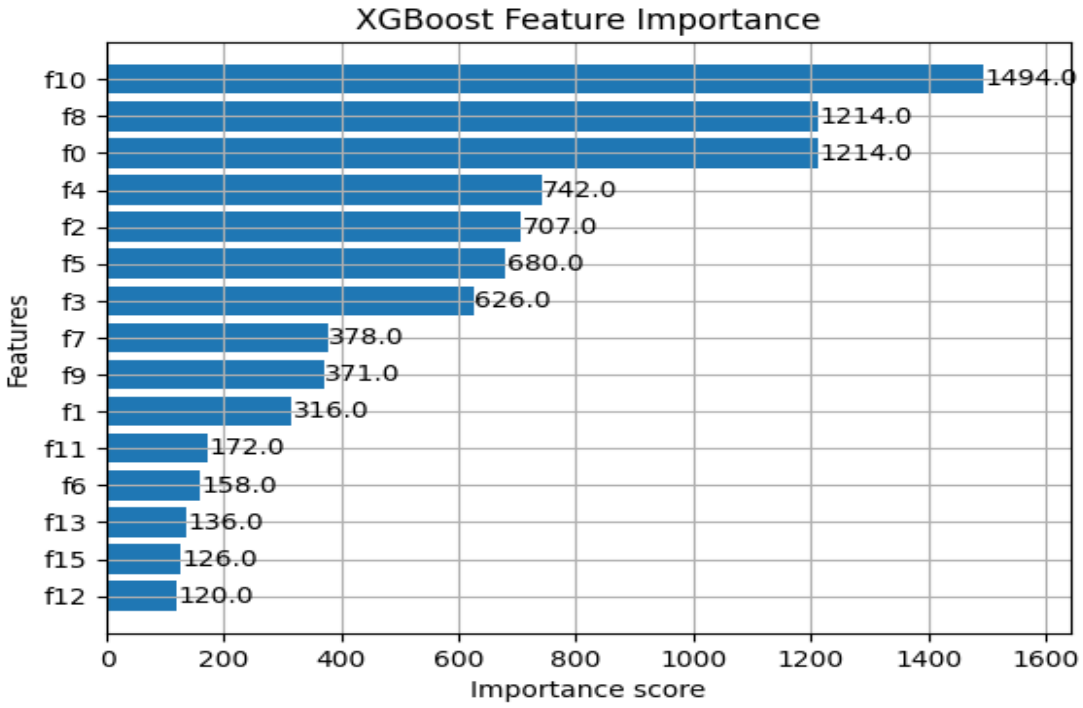
# Week 6 : Ensemble Model Development & Integration

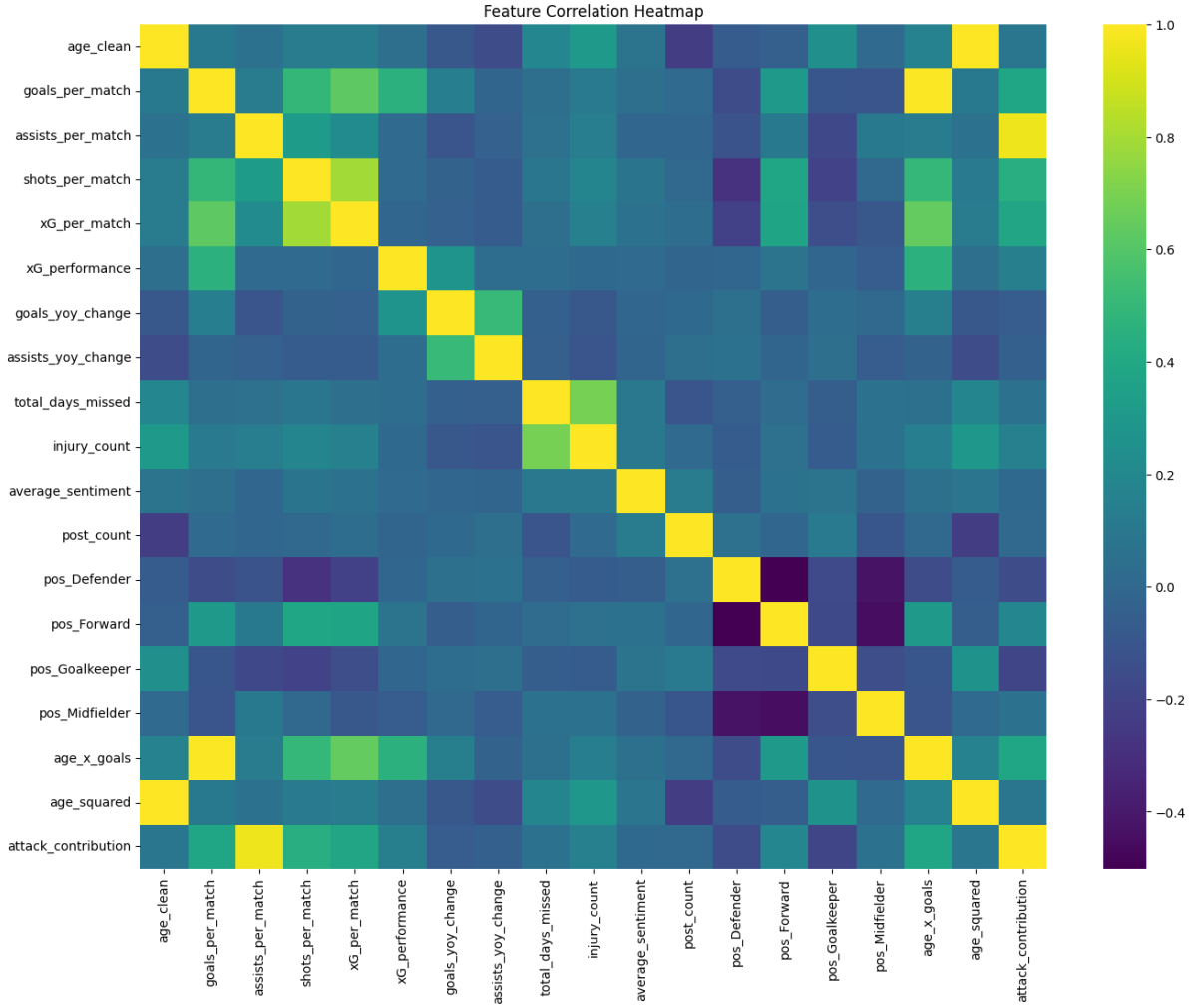
- Train an XGBoost Model: I'll use my master dataset to train an XGBoost model. XGBoost is incredibly powerful and is known for its high accuracy on structured data like mine.

- Integrate the Models: This is the key step. I will take the predictions generated by my LSTM model from Week 5 and feed them as a brand-new feature into my XGBoost model. This gives the XGBoost model the best of both worlds: all the rich static features, plus the time-series intelligence from the LSTM.

- Initial Comparison: I'll run a validation test to see how much the performance of the XGBoost model improves after adding the LSTM feature.

Reached Accuracy of approx. 96 % through this model






# Week 7 : Hyperparameter Tuning and Final Testing

- Tune the Ensemble Model: I'll use a technique like GridSearchCV to systematically test out hundreds of different combinations of settings (hyperparameters) for my XGBoost model. It's like tuning the knobs on a stereo to get the absolute perfect sound, but for my model.

- Test on the Holdout Set: This is the final exam. I'll use a "holdout" dataset—a portion of my data that I've kept completely separate and untouched throughout this entire process—to test my final, tuned model. This will give me an honest measure of how well it will perform in the real world.

- Analyze Final Metrics: I’ll calculate the final performance metrics: RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) .

RMSE : 0.89

MAE : 0.6 %

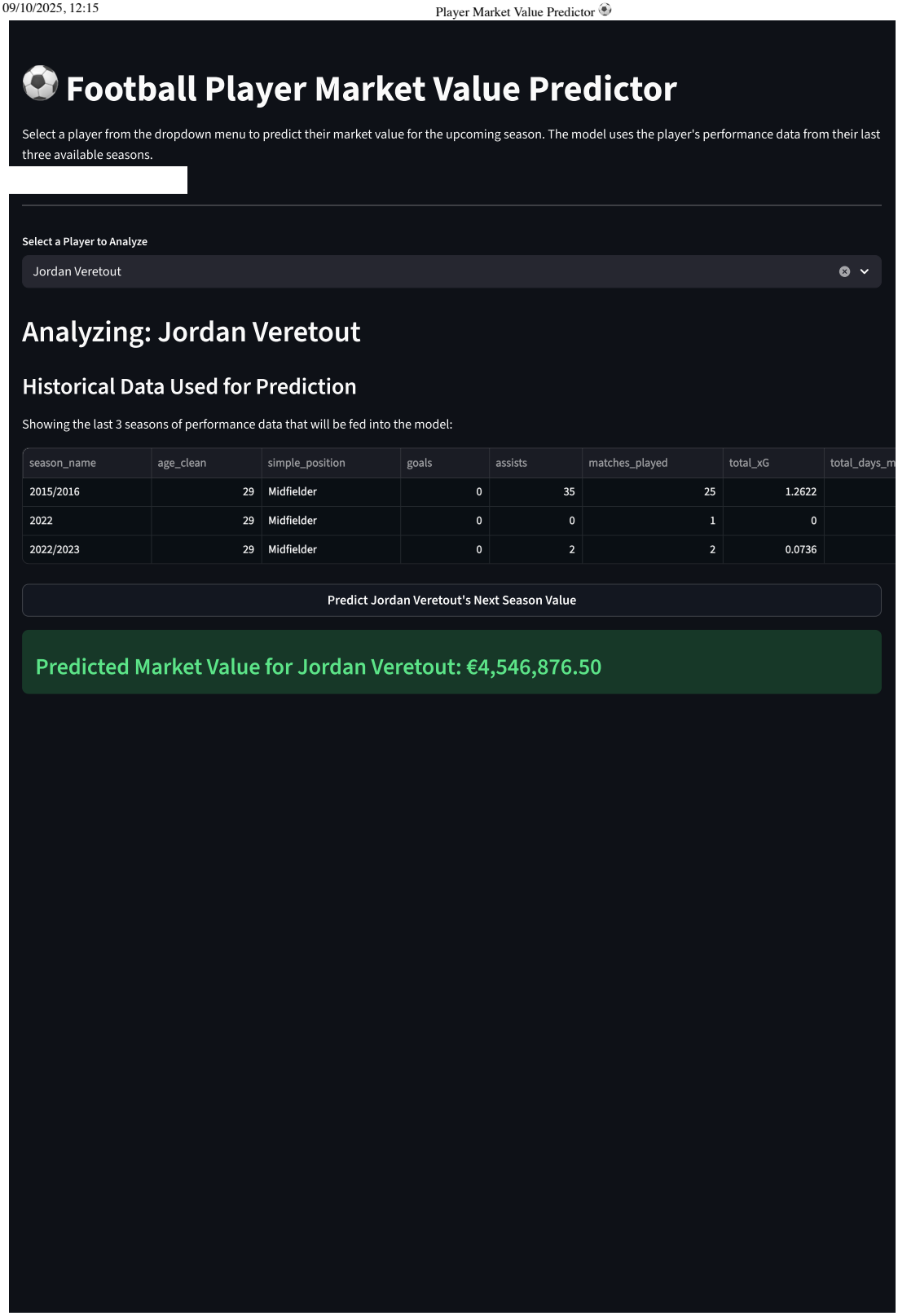
## Week 8 : Visualization and Reporting

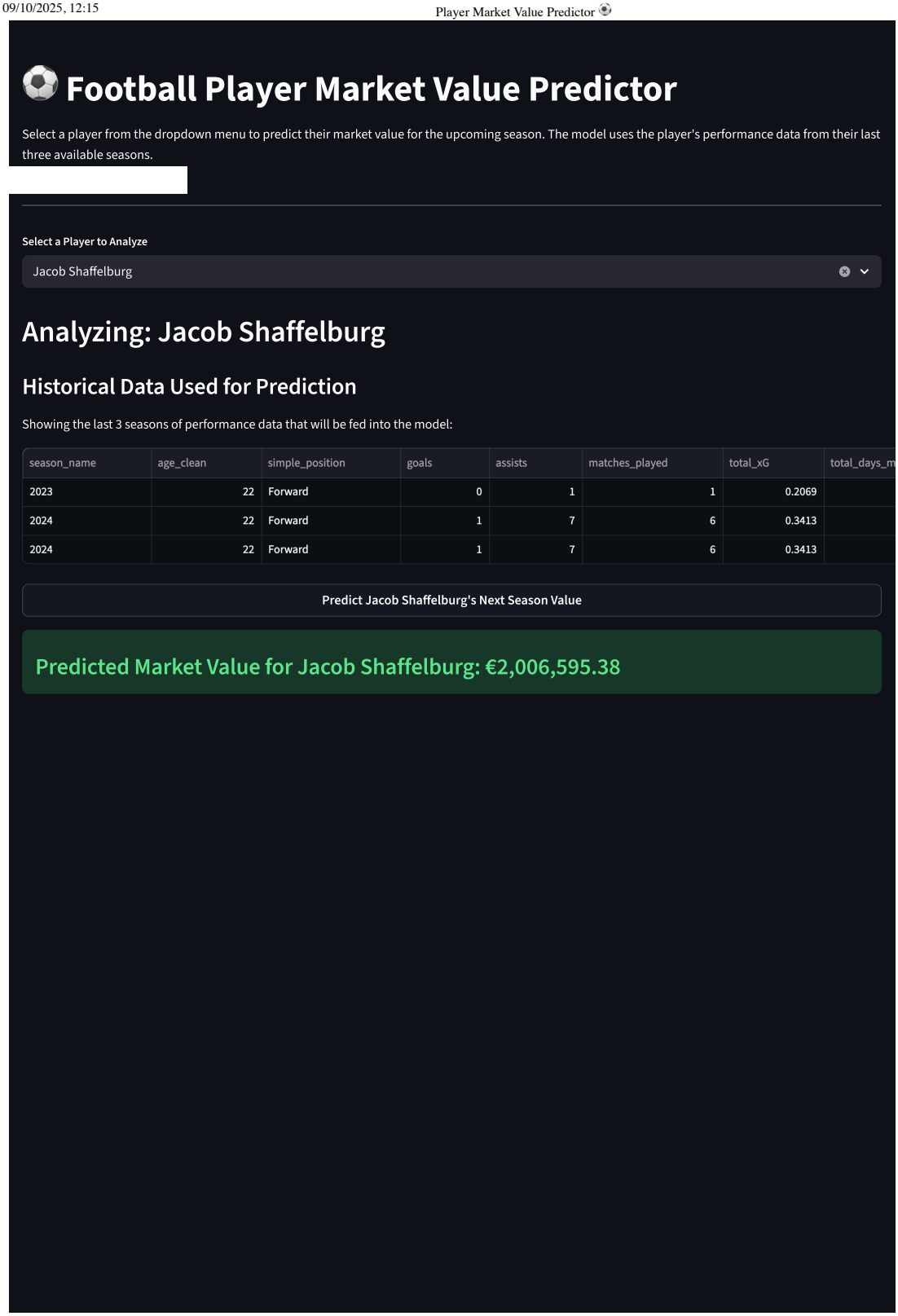
- Interpret the Model: I'll create a feature importance plot. This visualization will show me exactly which features—be it recent goals, sentiment score, or injury risk—had the biggest impact on predicting a player's value.

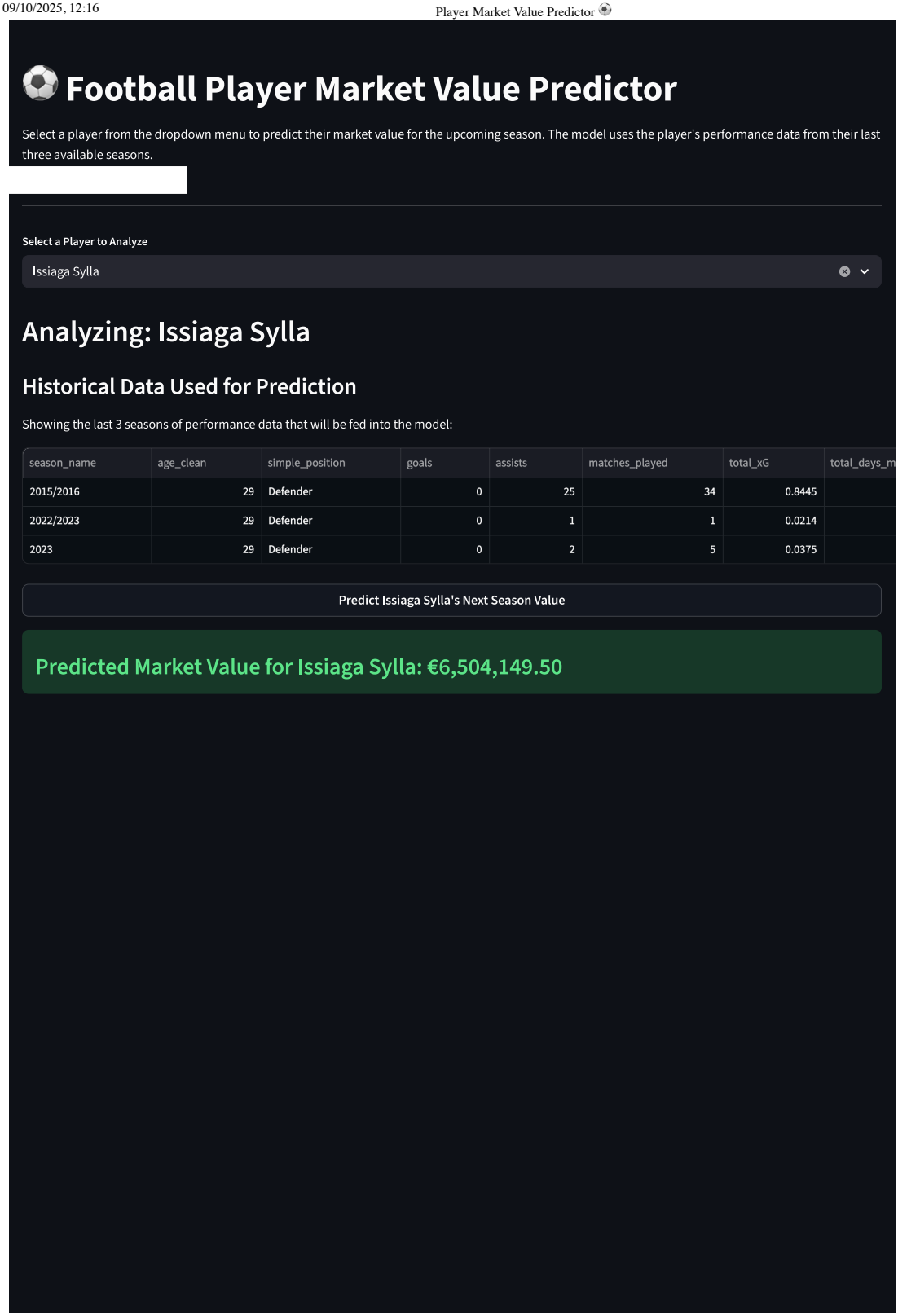
- Visualize the Results: I'll create clear and insightful charts. My key visualization will be a scatter plot of Actual vs. Predicted Values to show how close my model's predictions were to reality.

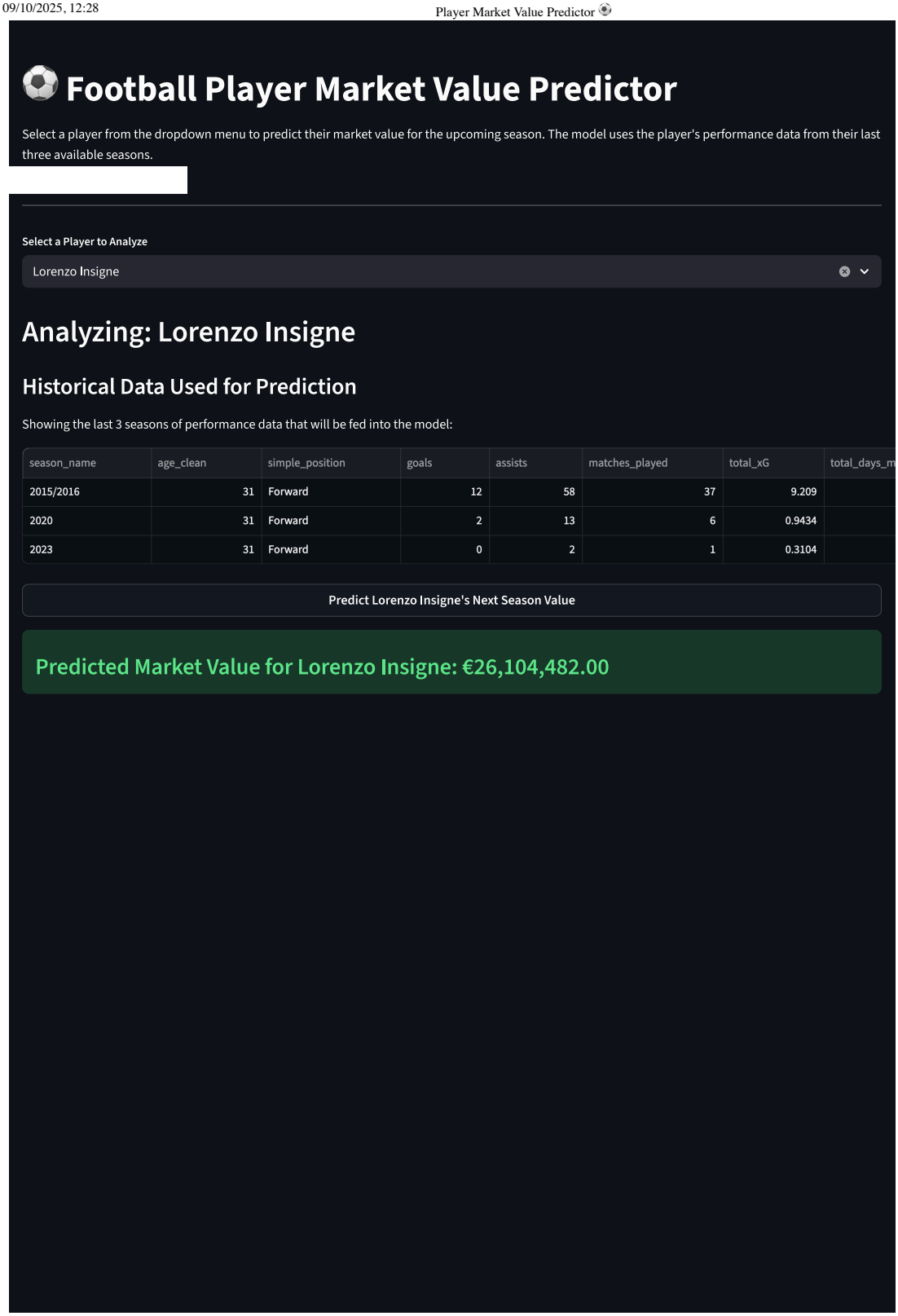
## Deployement :

Built the Frontend and hosted it on Streamlit and used NGrok for hosting .
Working fine with updating player names and showing the result in a pan of seconds.


⚽ Football Player Market Value Predictor
Select a player from the dropdown menu to predict their market value for the upcoming season. The model uses the player's performance data from their last three available seasons.
Select a Player to Analyze
Jordan Veretout
Analyzing: Jordan Veretout
Historical Data Used for Prediction
Showing the last 3 seasons of performance data that will be fed into the model:
season_name age_clean simple_position goals assists matches_played total_xG total_days_m
2015/2016 29 Midfielder 0 35 25 1.2622
2022 29 Midfielder 0 0 1 0
2022/2023 29 Midfielder 0 2 2 0.0736
Predict Jordan Veretout's Next Season Value
Predicted Market Value for Jordan Veretout: €4,546,876.50


⚽ Football Player Market Value Predictor
Select a player from the dropdown menu to predict their market value for the upcoming season. The model uses the player's performance data from their last three available seasons.
Select a Player to Analyze
Jacob Shaffelburg
Analyzing: Jacob Shaffelburg
Historical Data Used for Prediction
Showing the last 3 seasons of performance data that will be fed into the model:
season_name age_clean simple_position goals assists matches_played total_xG total_days_m
2023 22 Forward 0 1 1 0.2069
2024 22 Forward 1 7 6 0.3413
2024 22 Forward 1 7 6 0.3413
Predict Jacob Shaffelburg's Next Season Value
Predicted Market Value for Jacob Shaffelburg: €2,006,595.38


⚽ Football Player Market Value Predictor
Select a player from the dropdown menu to predict their market value for the upcoming season. The model uses the player's performance data from their last three available seasons.
Select a Player to Analyze
Issiaga Sylla
Analyzing: Issiaga Sylla
Historical Data Used for Prediction
Showing the last 3 seasons of performance data that will be fed into the model:
season_name age_clean simple_position goals assists matches_played total_xG total_days_m
2015/2016 29 Defender 0 25 34 0.8445
2022/2023 29 Defender 0 1 1 0.0214
2023 29 Defender 0 2 5 0.0375
Predict Issiaga Sylla's Next Season Value
Predicted Market Value for Issiaga Sylla: €6,504,149.50


⚽ Football Player Market Value Predictor
Select a player from the dropdown menu to predict their market value for the upcoming season. The model uses the player's performance data from their last three available seasons.
Select a Player to Analyze
Lorenzo Insigne
Analyzing: Lorenzo Insigne
Historical Data Used for Prediction
Showing the last 3 seasons of performance data that will be fed into the model:
season_name age_clean simple_position goals assists matches_played total_xG total_days_m
2015/2016 31 Forward 12 58 37 9.209
2020 31 Forward 2 13 6 0.9434
2023 31 Forward 0 2 1 0.3104
Predict Lorenzo Insigne's Next Season Value
Predicted Market Value for Lorenzo Insigne: €26,104,482.00

Thank you