Next-Gen Team Builder With Predictive AI

Report

By Team 97

December 4, 2024

Contents

1 Problem Understanding

Dream11 requires users to create teams by analyzing extensive player statistics and match conditions. While this can be an exciting and strategic process, it presents significant challenges, particularly for users who lack experience or familiarity with in-depth analytics of cricket.

1.1 Challenges

- **Data Complexity:** Users must evaluate large volumes of player and match data, making team selection time-consuming and complex. Less data-savvy users often struggle to understand or leverage these statistics effectively.
- **Usability and Engagement:** Existing features provide historical data and expert-curated suggestions but do not offer predictive or explainable insights that simplify the decision-making process.
- **Competitive Landscape:** With the fantasy cricket market becoming increasingly competitive, Dream11 must innovate to retain its leadership by providing differentiated and user-friendly tools.

1.2 Proposed Solution

- **Predictive Analytics:** Using an ML model (XGBoost Regressor) to forecast player performance based on historical data, match conditions, and contextual trends.
- **Explainability:** Employing LIME to justify recommendations and build user trust in the predictions.
- **Interactive Product UI:** A user-friendly interface that simplifies team selection with intuitive visualizations, actionable insights, and optional audio/video guides.

1.3 Goals

- **Enhance User Engagement:** Empower users to make data-driven decisions with clear, actionable insights.
- **Improve Accessibility:** Provide a seamless and intuitive experience catering to novice and experienced users.
- **Drive Innovation:** Establish Dream11 as a leader in fantasy sports by offering cutting-edge tools that differentiate it from competitors.

2 Data Acquisition and Preprocessing

For the data acquisition process, we utilized JSON files from Cricsheet, which contain detailed ball-by-ball data for cricket matches. The data from these files provides extensive information on player performances and match events, which is crucial for predicting fantasy points.

2.1 Data Collection Process

We sourced ball-by-ball data from Cricsheet.org, a publicly available repository of detailed cricket match statistics. Each match file in JSON format provided extensive information about the match, including the following key fields:

- **Match Information:** Metadata about the match, such as date, venue, and participating teams.
- **Ball-by-Ball Events:** Detailed information for every ball bowled, including:
  - Bowler and batsman involved
  - Runs scored on the delivery
  - Wickets and extras (e.g., no-balls, wides)
  - Non-striker's involvement

2.2 Data Transformation

To ensure the relevance and quality of the data, the following transformations were applied:

- **Filtering by Timeline:**
  - We removed the data prior to **2000s** because most players from that era are retired, and recent form is more critical for predicting current player performance.
  - Only matches played between **1st Jan 2014** and **30th June 2024** were retained for analysis.
- **Player-Level Data Aggregation:** The ball-by-ball data was aggregated to calculate key player statistics such as total runs, wickets, strike rates, and economy rates.
- **Additional Data Integration:**
  - The file, names.csv, was used to map Cricsheet player codes to the player's variant names, if any.
  - We integrated Dream11's fantasy points calculation system to compute ground-truth fantasy points for players given the match's data. This was essential for training and validating the predictive model.

2.3 Challenges in Data Preprocessing

- **Data Inconsistencies:** Certain fields, like those indicating wickets, were only present for specific types of dismissals, requiring special attention with conditional handling.
- **Encoding:** The data required thorough encoding and normalization to create consistent representations for downstream analysis. Nested JSON structures added complexity to extracting and transforming relevant information.

2.4 Final Data Representation

- After preprocessing, the data was organized into a player-centric data frame where each row represents an individual player's performance in a specific match.
- This structure ensures that each player's performance is isolated, aligning with the goal of predicting fantasy points at the player level.

3 Model Design

Our prediction model is built using **XGBoost** (Extreme Gradient Boosting), a machine learning algorithm known for its speed and accuracy in structured data problems. XGBoost is an ensemble method that combines multiple decision trees to optimize a predictive objective while minimizing errors through gradient boosting.

3.1 Exploration of Models

- **XGBoost Regressor:** Handles non-linear relationships effectively. Regularization mechanisms (L1 and L2) prevent overfitting. After extensive tuning, XGBoost consistently delivered the most accurate predictions, making it the final model of choice.
- **Neural Networks:** A neural network with 5 layers and 50 epochs was implemented and tested. While the model performed reasonably well, it could not outperform XGBoost. It also required extensive tuning and longer training times.
- **Time Series Models:** Each player's performance history was split to train individual models. Splitting the data player-wise resulted in insufficient data for many players. This method also required multiple model instances, and did not account for features like match condition

3.2 Fundamentals of XGBoost

XGBoost minimizes a regularized objective function, which balances model complexity and prediction accuracy. The objective function can be expressed as:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k),$$

where:

- $\ell(y_i, \hat{y}_i)$: Loss function measuring the difference between the true value y_i and the predicted value $\hat{y}_i$. Common choices include mean squared error (MSE) for regression and log-loss for classification.
- $\Omega(f_k) = \gamma T_k + \frac{1}{2}\lambda \|w_k\|^2$: Regularization term for tree k, controlling model complexity.
  - T_k: Number of leaves in the tree.
  - w_k: Leaf weights.
  - γ, λ: Regularization parameters.

At each iteration, XGBoost adds a new tree f_t to minimize the objective function. Using a second-order Taylor expansion, the objective for the t-th iteration becomes:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t),$$

where:

- $g_i = \frac{\partial \ell(y_i, \hat{y}_i)}{\partial \hat{y}_i}$: First derivative of the loss function (gradient).
- $h_i = \frac{\partial^2 \ell(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$: Second derivative of the loss function (Hessian).

The algorithm builds the tree f_t by optimizing this equation, ensuring the new tree improves predictions while controlling overfitting through $\Omega(f_t)$.

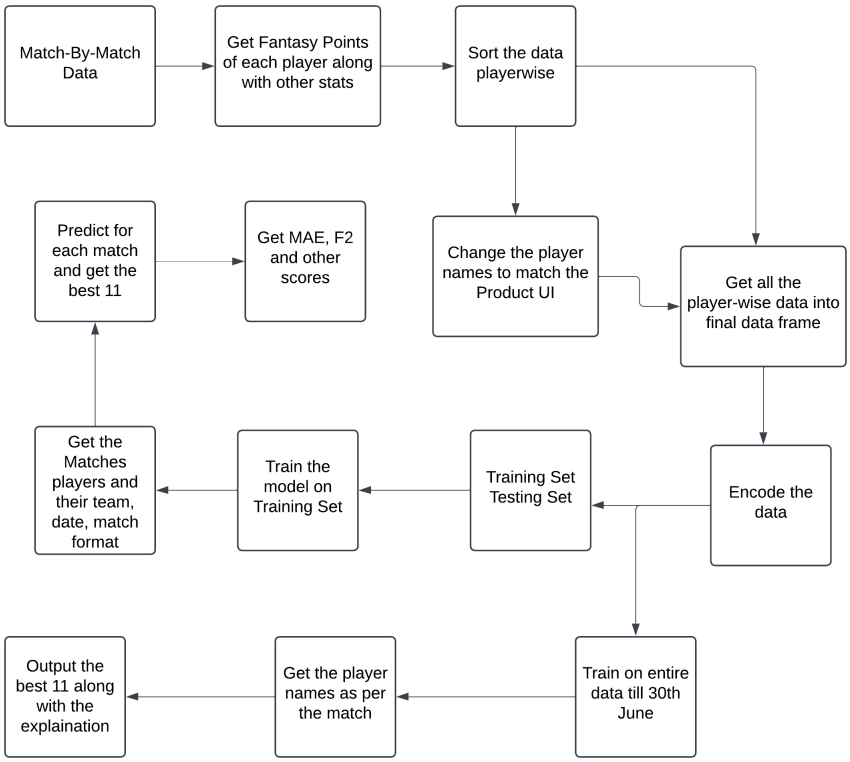

Figure 1: Solution Pipeline

3.3 Application to Fantasy Points Prediction

The XGBoost model is trained to predict a player's fantasy points (P_i) based on features (X_i) like historical performance, match conditions, and contextual factors:

$$P_i = f(X_i; \Theta),$$

where Θ represents the model parameters learned during training.

This predictive model underpins the Dream11 team recommendation system, providing accurate and explainable insights for optimal team selection.

4 Product UI

4.1 User Interface

The Product UI provides the users with an interface to interact with the fantasy Team Builder. It is built using ReactJS and TailwindCSS, offering a seamless user experience.

- **Components**
  - **Home:** Provides an overview of the product.
  - **Matches:** Categorized into 3 sections: Upcming Matches, Ongoing Matches and Past Matches.
  - A dedicated panel on the right side in the Matches page shows up-to-date cricket news fetched via RapidAPI (free-tier).
  - **About Us:** Contains detailed information about the product and its purpose.

  - **Support:** Offers a FAQ section, includes a form for users to submit specific queries.

- **Authentication:** We use JWT (JSON Web Token) for secure login sessions.

4.2 Key Features

- **Dream Team Recommendation:** The system identifies the 11 players predicted to score the highest combined fantasy points. Player selection is optimized based on Dream11 rules.

$$\text{Total Fantasy Points} = \sum_{i=1}^{11} P_i$$

  where P_i is the predicted fantasy points of player i, computed using the pretrained model.

- **Fantasy Points Computation:** Fantasy points for each player are calculated using a predefined point system, which evaluates their expected contributions across various match events (e.g., runs, wickets, catches).

- **Constraint Enforcement:**
  - The model ensures that the predicted fantasy cricket team adheres to the player composition rules. There will be at least one of each: Batsman, Bowler, All-Rounder, Wicket-Keeper in the 11-player team.
  - There is at least one player of each team in the predicted team.

- **Explainability with Insights:**
  - **LIME Integration:** The contribution of each player to the team's fantasy points is explained using Local Interpretable Model-agnostic Explanations (LIME).
  - **Mistral AI:** The output given by Lime is converted to an user understandable format by this model.

4.3 Output

- **Team Display:** The recommended team is presented as a visually intuitive list of 11 players. Each player entry includes:
  - Predicted fantasy points P_i
  - Contribution explanation via LIME

4.4 Pretrained Model Usage

The tool uses a pretrained XGBoost-based model named **ProductUI_Model**, stored in the folder **model_artifacts**.

- The model is trained on data up to **2024-06-30**.

- It uses inputs like team names, match date, and to predict fantasy points.

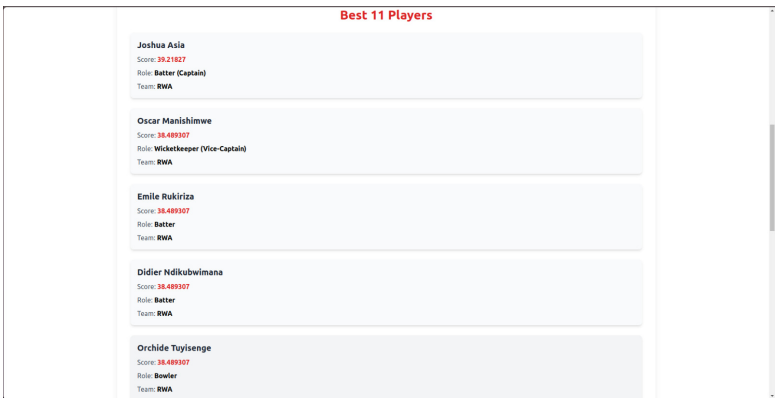

Figure 2: Product UI - Sample

5 Model UI

5.1 Goal

The Model UI is designed to allow evaluators and data scientists to comprehensively assess the performance of the predictive model across specific training and testing periods. It provides a streamlined interface for validating predictions, reviewing performance metrics, and managing updated model versions.

5.2 Inputs

- Training Period: The Start and End dates for the training set. The model will train on all the matches in this training period.
- Testing Period: The Start and End dates for the test set. The model will generate predictions for matches specified in the test period.

5.3 Outputs

The output will be a CSV with the following columns:

- Match Date, Name of Team 1, Name of Team 2,
- Predicted Dream Team with Fantasy Points of each player
- Dream Team (Best) with fantasy points of each player
- Mean Absolute Error (Dream Team Total Fantasy Points - Total Fantasy Points of the predicted team)

6 Analysis and Experiments

7 Technical Challenges

7.1 Data Preprocessing

- **Inconsistent Formats:** Handling mixed data structures (arrays, dictionaries) and missing fields, such as specific dismissal details, posed challenges during preprocessing.
- **Data Volume:** Filtering relevant matches while maintaining consistency across large datasets spanning multiple years required careful transformations.

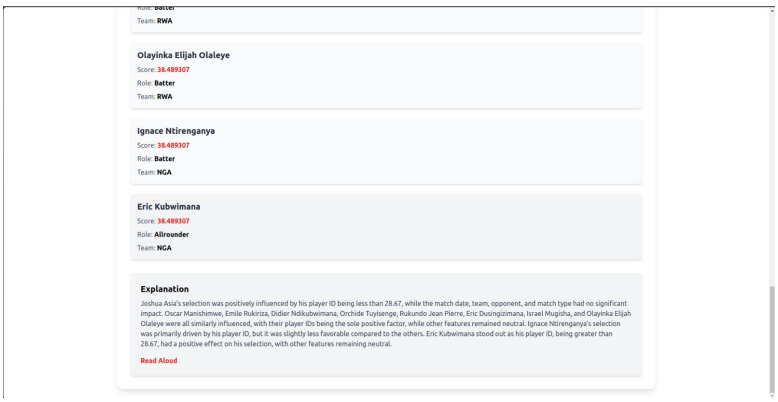

Figure 3: Product UI - Sample

7.2 Feature Engineering

- Designing impactful features, such as strike rates and economy rates, to capture non-linear relationships between match conditions and fantasy points was complex and required domain knowledge.

7.3 Model Selection

- Evaluating multiple models, including XGBoost, Neural Networks, and Time Series, required significant computational resources.
- Addressing limitations, such as insufficient player-specific data for Time Series models, highlighted challenges in balancing generalization and personalization.

7.4 Real-Time Integration

- Managing API rate limits and delays for live updates using free-tier APIs, such as RapidAPI.
- Dynamically handling squad updates and match details for upcoming games required robust workflows.

7.5 UI Challenges

- Integrating LIME for explainability while ensuring user accessibility posed a balance between complexity and usability.
- Optimizing response times to meet strict constraints, such as generating teams in under 10 seconds, required efficient implementation.

These challenges were addressed through careful design, iterative testing, and leveraging modern tools and frameworks.

8 Next Steps

8.1 Model Refinement

- Optimize XGBoost parameters to improve prediction accuracy. Incorporate additional features like weather conditions and pitch reports for more robust predictions.

8.2 Real-Time Data Integration

- Add APIs for live player stats and team updates. Enable dynamic squad updates for upcoming matches.

8.3 UI Enhancements

- Expand explainability with advanced visualizations, such as SHAP. Add personalized features like saved preferences and team performance analysis.

8.4 Model Validation

- Test the model on matches post-2024-07-01 to ensure real-world accuracy. Perform error analysis to identify improvement areas.

8.5 Expansion to Other Sports

- Adapt the system for other popular fantasy sports such as football, basketball, and kabaddi. Modify feature engineering and model training processes to account for sport-specific metrics and scoring rules.

8.6 Scalability and Deployment

- Deploy the system on the cloud for real-time access. Optimize performance for faster team generation.

These steps will strengthen the solution, ensuring scalability, versatility, and an enhanced user experience across multiple sports.