Context-Aware Performance Modeling in Cricket: Quantifying Situational Impact and Opponent-Specific Predictions

Om Namdeo, Charu Jain, Debarpan Debnath

Abstract
This study develops a context-aware performance modeling framework for T20 international cricket, focusing on quantifying situational impacts and opponent-specific outcomes. Using ball-by-ball data from Cricsheet, we preprocess and filter matches to ensure quality, normalize runs to account for match variability, and apply temporal weighting to prioritize recent player form. Performance metrics for batsmen, bowlers, and their matchups are derived, capturing experience, averages, and strike rates. Two predictive models—an XGBoost regressor and a PyTorch neural network—are trained to forecast match-specific strike rates, leveraging weighted statistics. Despite robust feature engineering, both models exhibit overfitting, with limited generalization to test data due to time constraints. Visualizations of performance trends offer valuable insights. The code repository for this project is available on GitHub. [1]

1 Introduction

Cricket, a sport characterized by its intricate blend of strategy, skill, and situational dynamics, presents a rich domain for data-driven analysis. The performance of players in this game is not merely a function of raw statistics but is deeply contextual, influenced by factors such as game state, opponent strategies, and situational pressure. Traditional metrics, while valuable, often fail to capture the nuanced contributions of players across diverse scenarios or their adaptability against specific adversaries. This limitation motivates the need for advanced analytical models that can disentangle these complexities and provide a more granular understanding of player performance.

To address this, we propose a machine learning framework to model cricket player performance through the lens of contextual game dynamics and opponent-specific interactions. We intend to quantify individual contributions under varying pressure conditions and predict player outcomes against particular opponents, offering a novel perspective on performance evaluation. By integrating domain-inspired features with state-of-the-art predictive techniques, we seek to address two key challenges: (1) assessing the situational value of player actions, and (2) anticipating performance in player-opponent matchups. This proposed study aspires to contribute to the field of sports analytics, and more widely to context-aware modeling in sequential decision-making systems.

Through this framework, we hope to equip a range of stakeholders: coaches could optimize strategies by understanding player reliability in high-pressure moments; analysts and team selectors could enhance scouting and lineup decisions with predictive matchup insights; and fans could gain a deeper appreciation of the game through contextual, data-driven narratives that illuminate the nuances of cricket performance.

2 Related Works

The application of machine learning and statistical methods to cricket analytics has gained significant attention in recent years, driven by the availability of rich datasets and the sport's inherent complexity. Several works have explored player performance evaluation and match outcome prediction, laying the groundwork for our research.

Jhawar and Pudi [2] propose a quantitative framework for assessing player performance and predicting winners in One Day International (ODI) cricket. Their approach leverages statistical aggregates to model team strength, achieving promising results in outcome prediction. However, it overlooks situational context, such as pressure dynamics, limiting its ability to capture nuanced player contributions.

Similarly, Passi and Pandey [3] enhance prediction accuracy for cricket matches using machine learning techniques. Their work employs a range of models to forecast match outcomes, demonstrating improved performance over traditional statistics. Yet, the lack of opponent-specific analysis and game-state awareness restricts its applicability to fine-grained performance modeling.

Bharadwaj et al. [4] apply machine learning to predict player performance, integrating diverse algorithms to analyze historical statistics. Their multi-model approach offers a broad perspective on player evaluation but does not explicitly address situational factors or opponent interactions, which are critical in cricket's dynamic environment.

Quazi et al. [5] present a Bayesian approach to predict cricket tournament outcomes, with a focus on opponent-specific performance. By incorporating player history against specific adversaries and using Bayesian priors with stratified sampling, they simulate matches to estimate win probabilities, achieving notable results for the ICC 2023 Cricket World Cup. However, their model treats events as independent, an oversimplification of real-world dependencies, and neglects game-state context (e.g., pressure in death overs) and situational action value (e.g., a powerplay six versus a death-over six).

Sumathi et al. [6] explore machine learning algorithms, including Linear Regression, K-Means clustering, and Random Forests, to predict and evaluate player performance. Their framework effectively clusters players by similarity and forecasts future contributions, supporting team selection and strategy. Nevertheless, its reliance on data quality, the reduced interpretability of complex models like Random Forests, and the challenge of modeling cricket's unpredictable factors (e.g., weather, psychological elements) limit its practical utility.

Collectively, these works highlight the potential of data-driven methods in cricket analytics but fall short in capturing the interplay of situational pressure and opponent-specific dynamics. Our research will attempt to address these gaps by integrating context-aware features and opponent matchup modeling, offering a more comprehensive evaluation of player performance in cricket.

3 Data Source

To support our proposed framework for context-aware performance modeling in cricket, we leverage a comprehensive dataset of ball-by-ball records obtained from Cricsheet [7]. This data-source provides detailed match data across various cricket formats, including One Day Internationals (ODIs), Twenty20 Internationals (T20I), Test matches, multiple leagues and international tournaments. Each record encapsulates granular event-level information, such as the outcome of every delivery, player actions, and game context, making it an ideal resource for analyzing situational dynamics and opponent-specific interactions.

The Cricsheet dataset is structured to include essential metadata—teams, players, dates, and venues — alongside sequential ball outcomes, such as runs scored, wickets taken, and extras. This richness enables the extraction of domain-specific features, such as pressure scenarios (e.g., death overs or chase

situations) and player-opponent matchups, which are central to our study. Collected and maintained by a community-driven effort, the data undergoes regular updates and validation, ensuring reliability for analytical purposes. By grounding our work in this resource, we aim to capture the intricacies of cricket performance at a level of detail unattainable through aggregate statistics alone.

4 Data Exploration

To establish a foundation for our proposed context-aware performance modeling, we undertake a preliminary exploration of the raw ball-by-ball data sourced from Cricsheet, delivered in JSON format. These files feature nested structures detailing match events, including ball outcomes, player actions, and contextual metadata. We begin by parsing these JSON files and converting the nested data into a flat pandas DataFrame, where each row represents the outcome of a single delivery. This tabular format enables efficient analysis of game sequences. A key observation from this process is the inherent time-series nature of the data—not only within the progression of a single match, where ball outcomes unfold sequentially, but also across the dataset as a whole, spanning multiple matches over years. This temporal structure has significant implications for our analysis, as we intend to ensure that, for instance, when modeling a match from 2015, only data available up to that point in time is utilized, preserving chronological integrity.

Using this flattened dataset, we reconstruct traditional cricket scorecards, computing aggregate statistics such as total runs scored by each batsman, team run rates, and strike rates (runs per 100 balls faced). For bowlers, we derive metrics like wickets taken, economy rates (runs conceded per over), and bowling strike rates (balls per wicket), providing a conventional performance baseline. Leveraging the ball-by-ball granularity, we further extract advanced metrics unattainable from game-level summaries, such as runs scored and strike rates of each batsman against individual bowlers. These matchup-specific measures are pivotal for our opponent-focused analysis, capturing player interactions that vary across encounters. The time-series aspect enhances this exploration, as we can track how these statistics evolve both within a match (e.g., early overs versus death overs) and historically across a player's career up to a given point.

This preliminary analysis highlights the dataset's suitability for nuanced performance evaluation and emphasizes the importance of adopting machine learning techniques tailored to time-series data. To model the sequential patterns within matches and enforce the constraint of using only prior data for historical predictions, we are considering a range of autoregressive approaches, such as Autoregressive Integrated Moving Average (ARIMA) models, which excel at capturing linear trends and seasonality in time-ordered data, and Vector Autoregression (VAR), which extends this capability to multivariate time series by modeling interdependencies among multiple variables (e.g., batsman and bowler performance metrics simultaneously). Additionally, correlation analysis offers a lightweight method to identify temporal dependencies between player performances and game states, potentially informing feature selection. For more complex, non-linear dynamics inherent in cricket—such as abrupt shifts in momentum or pressure-induced performance changes—we plan to explore deep learning techniques, including Recurrent Neural Networks (RNNs) and their advanced variants like Long Short-Term Memory (LSTM) units, which are well-suited to modeling long-term dependencies in sequential data.

Beyond these, we are also evaluating the applicability of Transformer-based architectures, which leverage attention mechanisms to weigh the relevance of past events when predicting future outcomes, offering flexibility for both within-match and cross-match analyses. Additionally, Hidden Markov Models (HMMs) could be employed to model latent game states (e.g., high-pressure versus stable phases) that influence observable ball outcomes, aligning with our focus on situational context. While this exploration is based on a subset of matches, it validates the feasibility of deriving both standard and

matchup-specific metrics, motivating further development of our framework with a comprehensive suite of time-series methods in subsequent stages.

5 Data Preprocessing

The `filter_data` function ensures data quality through:

- **Full Member Restriction**: Retains only matches between ICC full members for competitive consistency.
- **Exclusion of Invalid Matches**: Removes incomplete matches (no result, abandoned due to bad weather or other reasons) or those using the Duckworth-Lewis method to avoid biased metrics.
- **Super Over Removal**: Excludes super overs to focus on standard 20-over gameplay.

6 Feature Engineering

Feature engineering transforms raw data into predictive features, with two key techniques: normalization of runs and performance weighting by a discount factor. These address variability in match conditions and the temporal relevance of performances.

6.1 Normalization of Runs

Normalization, implemented in `normalize_runs`, standardizes runs per delivery to ensure comparability across matches. Variations in venue, pitch conditions, and historical rule changes (e.g., powerplay rules) can significantly affect scoring. By computing the match-specific mean (μ) and standard deviation (σ) of runs, normalization mitigates these effects:

$$\text{runs}_{\text{norm}} = \frac{\text{runs} - \mu}{\sigma}$$

For teams dismissed before completing their allotted overs, the `update_mean_std` function adjusts statistics to account for zero runs on remaining balls, reflecting the mathematical cessation of scoring. This is crucial for accurate mean and standard deviation calculations. However, when a team successfully chases a target with overs remaining, these adjustments are not applied, as the innings concludes without hypothetical zeros. The updated mean and standard deviation are:

$$\text{new\_mean} = \frac{\mu}{1 + \frac{K}{N}}, \quad \text{new\_std} = \sqrt{\frac{\sigma^2}{1 + \frac{K}{N}} + \frac{K}{N} \cdot \text{new\_mean}^2}$$

where N is the number of deliveries, and K is the number of remaining balls. This ensures correct normalization, enhancing feature consistency.

6.2 Weighting by Discount Factor

The `calculate_weights` function applies a discount factor of 0.99962 to prioritize recent performances, recognizing that a player's current form is more predictive than older data. For a match k days ago, the weight is:

$$w = \gamma^k, \quad \gamma = 0.99962$$

The half-life, where weight reduces to 0.5, is:

$$0.5 = 0.99962^{k} \implies k = \frac{\ln(0.5)}{\ln(0.99962)} \approx 1822 \text{ days} \approx 5 \text{ years}$$

Thus, performances from approximately 5 years ago have half the influence of recent ones, aligning metrics with players' current abilities and form.

7 Player Performance Metrics

All player performance metrics are calculated cumulatively, weighted by a discount factor, for every date a match is recorded in our dataset. This approach ensures that more recent performances have a greater influence on the current metrics, reflecting the player's current form and ability. These statistics are computed for all players in the dataset, although the plots presented in the following subsections illustrate these weighted metrics over time for selected top players, providing insights into their performance trends.

Cumulative Weighted Performance Metrics Calculation

For each match date, we compute the weighted sum of a player's performances in all matches up to and including that date, with weights decreasing exponentially based on the time elapsed since those matches.

Consider an illustrative example with four matches played on days 0, 3, 10, and 14 (with time gaps of 3, 7, and 4 days, respectively). For three batsmen A, B, and C, their performance (let's say runs) in these matches are:

$$P = \begin{bmatrix} 10 & 25 & 31 & 6 \\ 41 & 22 & 90 & 13 \\ 21 & 37 & 9 & 17 \end{bmatrix}$$

where rows correspond to batsmen A, B, and C, and columns correspond to matches on days 0, 3, 10, and 14. The weight matrix W, is constructed as:

$$W = \begin{bmatrix} \gamma^{0} & \gamma^{0} & \gamma^{0} & \gamma^{0} \\ 0 & \gamma^{-3} & \gamma^{-3} & \gamma^{-3} \\ 0 & 0 & \gamma^{-10} & \gamma^{-10} \\ 0 & 0 & 0 & \gamma^{-14} \end{bmatrix}$$

The weighted runs up to each match date are computed via matrix multiplication $P \times W$, resulting in a matrix of shape (3 players, 4 dates).

$$P \times W = \begin{bmatrix} 10\gamma^{0} & 10\gamma^{0} + 25\gamma^{-3} & 10\gamma^{0} + 25\gamma^{-3} + 31\gamma^{-10} & 10\gamma^{0} + 25\gamma^{-3} + 31\gamma^{-10} + 6\gamma^{-14} \\ 41\gamma^{0} & 41\gamma^{0} + 22\gamma^{-3} & 41\gamma^{0} + 22\gamma^{-3} + 90\gamma^{-10} & 41\gamma^{0} + 22\gamma^{-3} + 90\gamma^{-10} + 13\gamma^{-14} \\ 21\gamma^{0} & 21\gamma^{0} + 37\gamma^{-3} & 21\gamma^{0} + 37\gamma^{-3} + 9\gamma^{-10} & 21\gamma^{0} + 37\gamma^{-3} + 9\gamma^{-10} + 17\gamma^{-14} \end{bmatrix}$$

This matrix operation efficiently computes the weighted performance for all players across all dates, applying the discount factor appropriately. Rows correspond to batsmen A, B, and C, and columns correspond to cumulative weighted performance till matches on days 0, 3, 10, and 14. Observe that the provided weight matrix uses γ^{-k}, which increases weights over time, which is equivalent of decreasing weights as we go back in time.

7.1 Batsman Statistics

The `analyze_batsmen` function calculates key metrics for batsmen based on normalized runs:

- **Experience**: Weighted balls faced.
- **Average**: Weighted runs (normalized) per dismissal.
- **Strike Rate**: Weighted runs (normalized) per 100 balls.

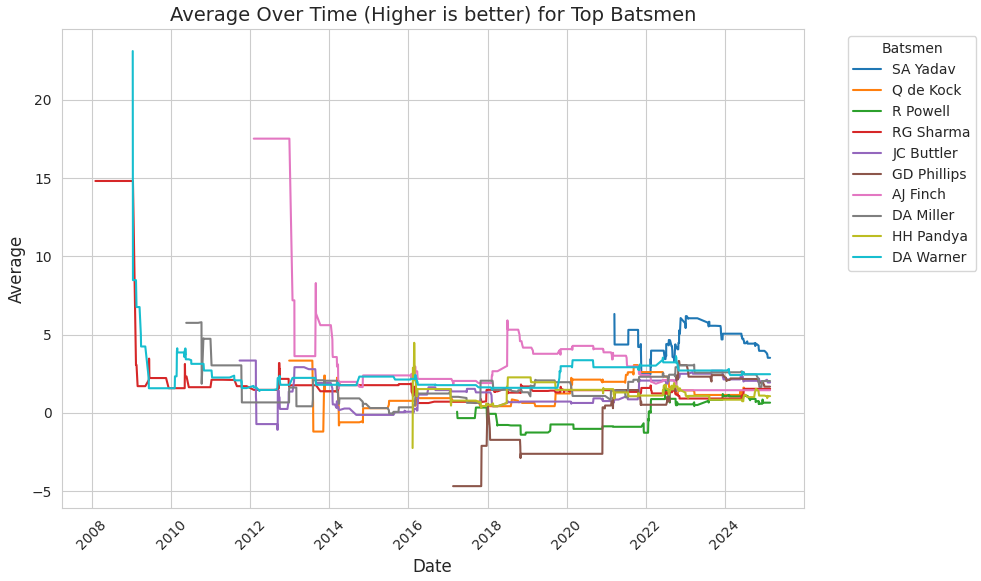

Figure 1: Weighted average over time (higher is better) for top batsmen.

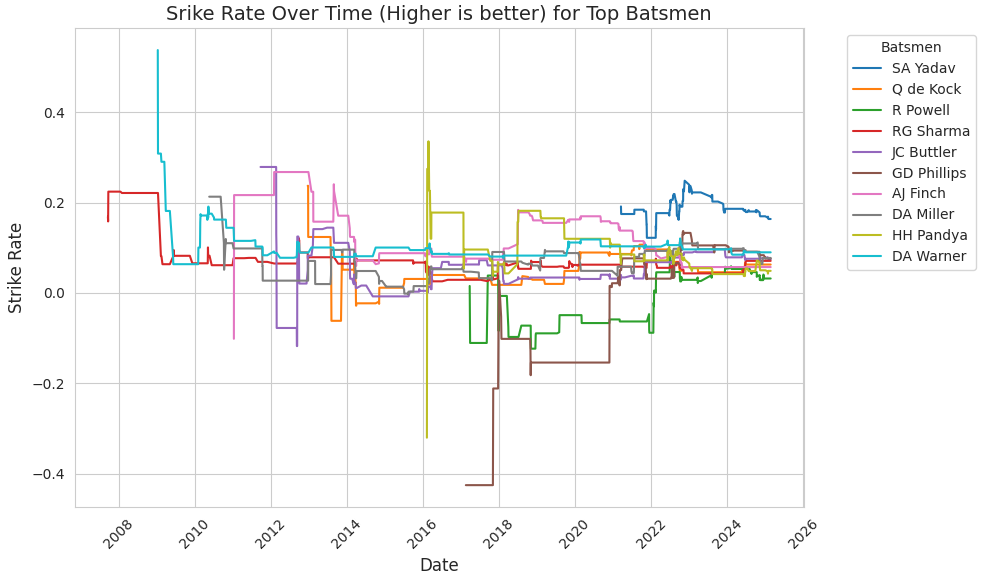

Figure 2: Weighted strike rate over time (higher is better) for top batsmen.

7.2 *Bowler Statistics*

The `analyze_bowlers` function computes metrics for bowlers based on normalized runs:

- **Experience**: Weighted legal balls bowled.
- **Average**: Weighted runs (normalized) conceded per wicket.
- **Economy**: Weighted runs (normalized) per over.
- **Strike Rate**: Weighted balls per wicket.

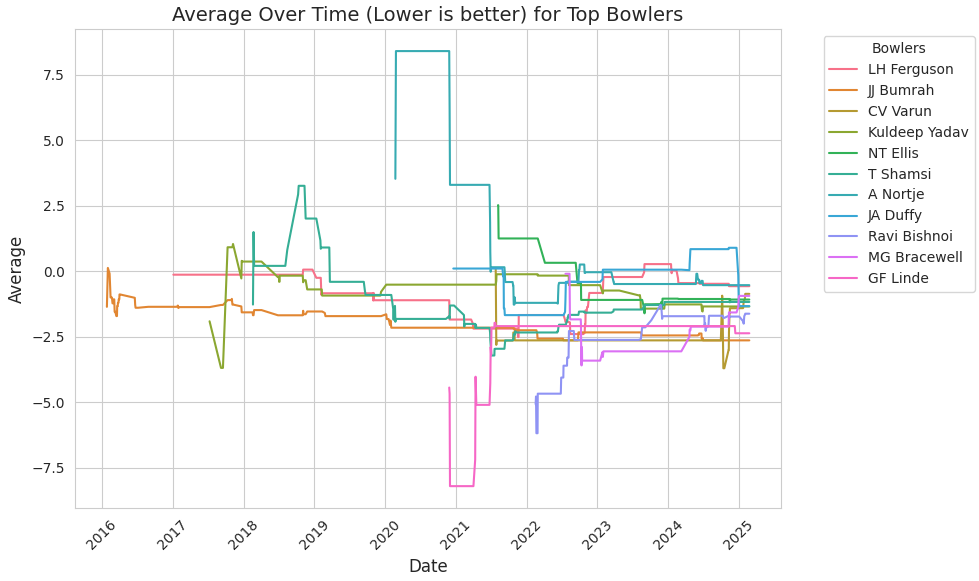

Figure 3: Weighted average over time (lower is better)for top bowlers.

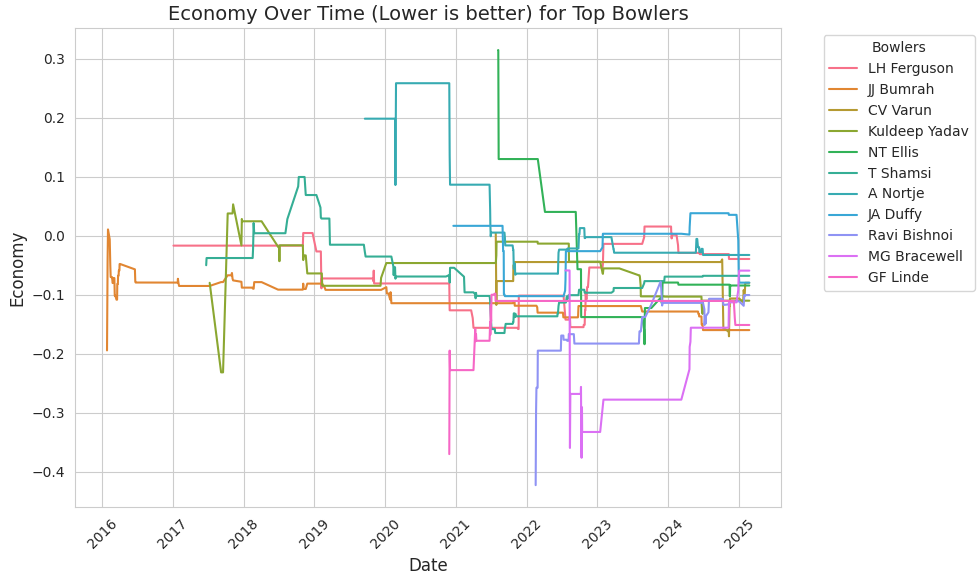

Figure 4: Weighted economy over time (lower is better)for top bowlers.

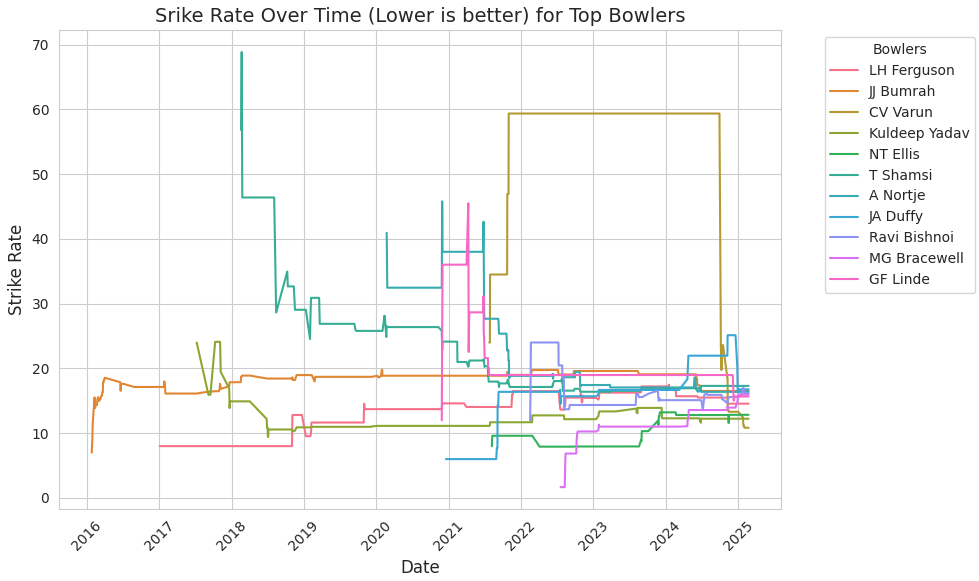

Figure 5: Weighted strike rate over time (lower is better) for top bowlers.

7.3 Batsman-Bowler Matchup Statistics

The `analyze_batsmen_bowlers` function derives matchup-specific metrics:

- **Experience**: Weighted balls in the matchup.
- **Match Strike Rate**: Per-match strike rate in the matchup. (This metric is used as the prediction target for our models).
- **Batting Strike Rate**: Weighted batsman strike rate vs. bowler (normalized runs).
- **Average**: Weighted runs (normalized) per dismissal.
- **Bowling Strike Rate**: Weighted balls per wicket.

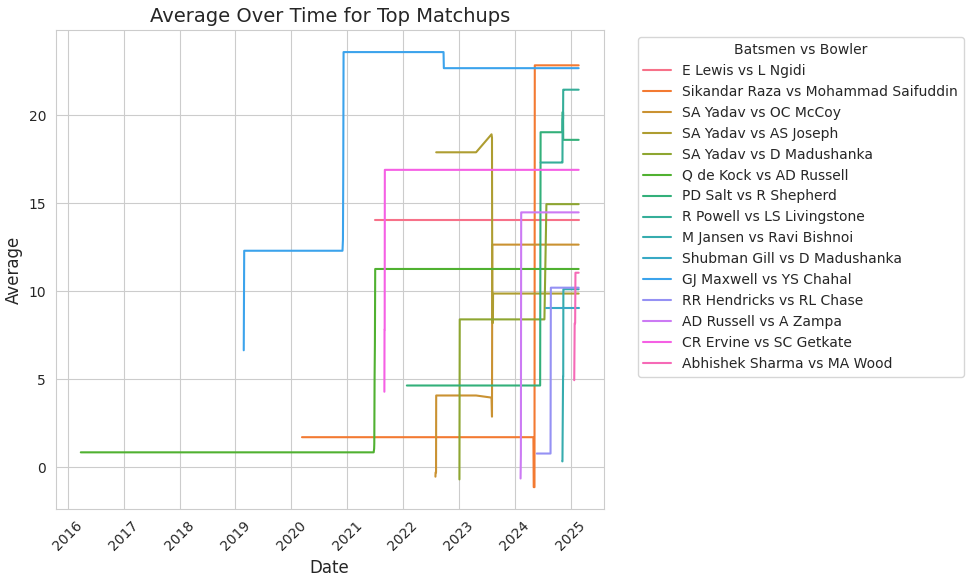

Figure 6: Weighted average over time for top matchups.

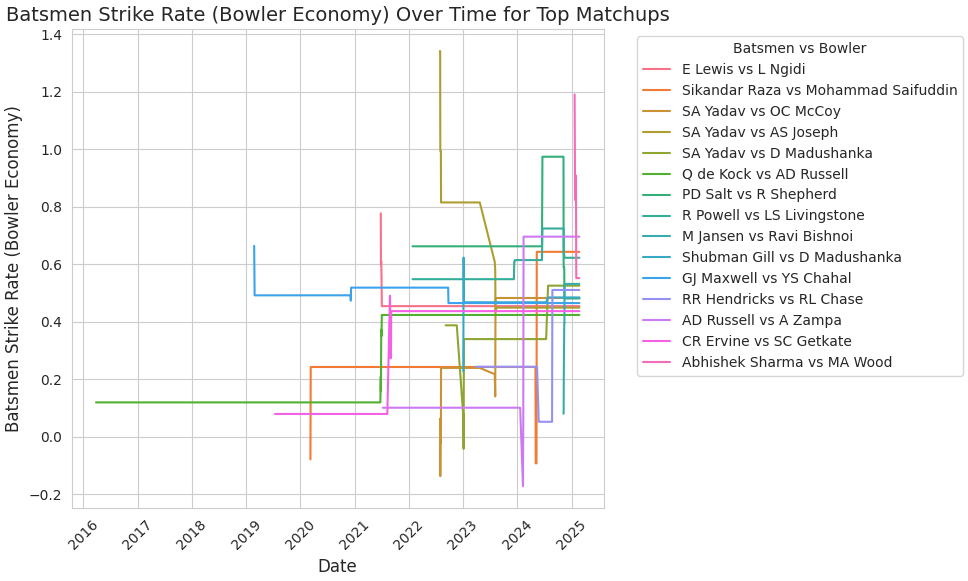

Figure 7: Weighted batsmen strike rate (bowler economy) for top matchups.

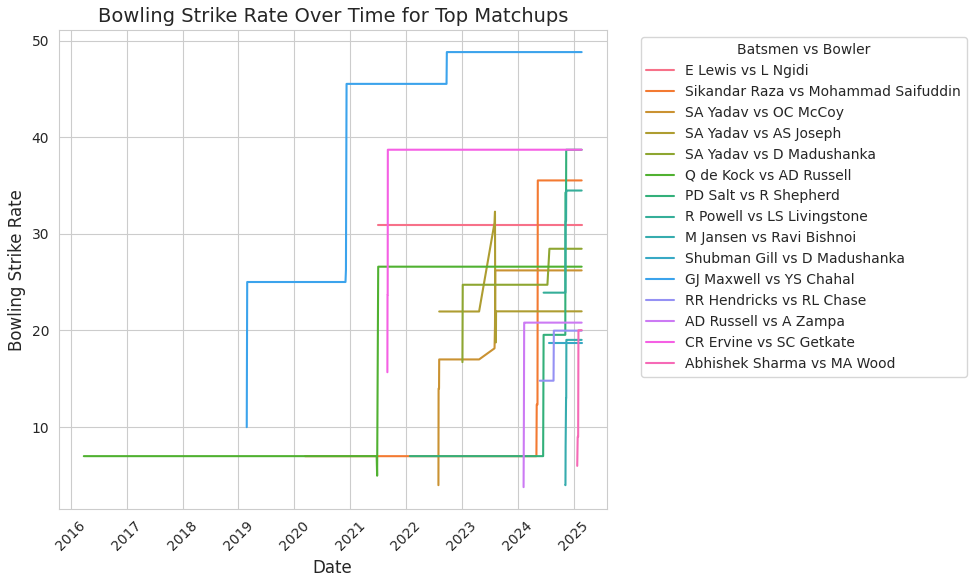

Figure 8: Weighted bowling strike rate for top matchups.

8 Model Development

The dataset for model training consists of features derived from the weighted performance metrics of batsmen and bowlers, as well as their matchup-specific statistics. The target variable is the match-specific strike rate (`match_sr`) based on normalized runs. Table 1 shows a few sample rows from the training data to illustrate the features and target.

Table 1: Sample rows from the training dataset.

batter_experience	batter_avg	batter_sr	bowler_experience	bowler_avg	bowler_eco	bowler_sr	match_sr
29.669	9.377	0.697	243.363	1.975	0.086	22.930	-0.154
19.857	3.775	0.419	57.942	-0.604	-0.050	12.000	-0.268
44.399	6.248	0.310	95.862	-1.359	-0.194	7.000	-0.913
29.669	9.377	0.697	568.904	1.760	0.084	21.058	-0.626
44.399	6.248	0.310	1937.904	-0.928	-0.045	20.512	-0.584

8.1 XGBoost Model

The XGBoost regression model was employed due to its effectiveness with structured data and ability to handle non-linear relationships. The model was configured with early stopping after 20 rounds to prevent overfitting. Features included the weighted performance metrics described earlier.

8.2 PyTorch Neural Network Model

In addition to XGBoost, a neural network model was implemented using PyTorch to explore the potential of deep learning for this task. The network architecture consists of two hidden layers with 64 and 32 neurons, respectively, each followed by a ReLU activation function, and a single output neuron for regression. The features were standardized using StandardScaler before training. The model was trained using the Adam optimizer with a learning rate of 0.001 and mean squared error (MSE) loss function for 100 epochs with a batch size of 32. The model with the lowest test loss during training was saved for evaluation.

9 Model Evaluation

To assess the performance of the models, we used the root mean squared error (RMSE) metric on both the training and test sets. Despite efforts to capture the dynamics of cricket performance through feature engineering, both models exhibited signs of overfitting, with limited generalization to unseen data.

9.1 XGBoost Model Evaluation

For the XGBoost model, the training RMSE was 0.5308, while the test RMSE was 0.5383. Although the test RMSE is only slightly higher than the training RMSE, indicating mild overfitting, the overall performance suggests that although the model captures some patterns in the data but it is unable to account for the complexities of cricket matchups. Figure 9 shows the training and validation loss curves, illustrating that the model doesn't learn much.

Figure 9: Training and validation loss vs. number of trees for the XGBoost model.

Additionally, the feature importance plot in Figure 10 highlights the most influential features, such as recent strike rates and matchup experience, which align with domain knowledge of cricket performance factors.

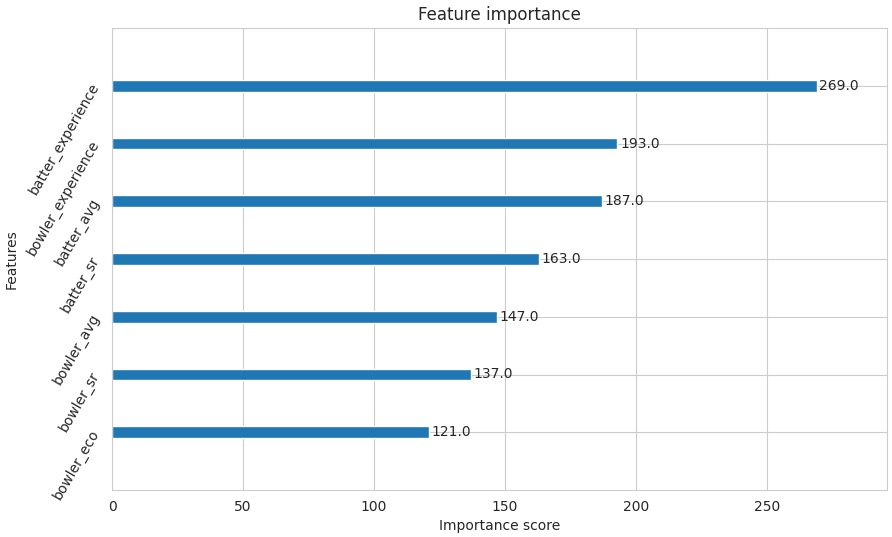

Figure 10: Feature importance for the XGBoost model.

9.2 PyTorch Neural Network Model Evaluation

The neural network model was trained with the goal of capturing more complex interactions between features. However, similar to the XGBoost model, it showed signs of overfitting. The training loss decreased steadily over the epochs, but the test loss barely showed any improvements, as shown in Figure 11.

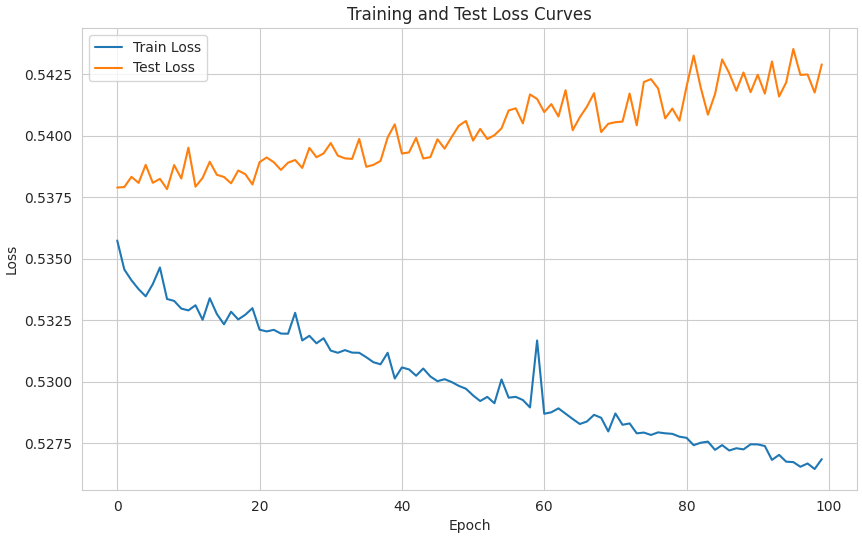

Figure 11: Training and test loss curves for the PyTorch neural network model.

The best test RMSE achieved by the neural network was comparable to that of the XGBoost model, indicating that while the neural network has the capacity to learn intricate patterns, the current feature

set and model architecture may not be sufficient to significantly improve predictive performance.

Due to time constraints, extensive hyperparameter tuning and further feature engineering were not feasible. Future efforts could focus on optimizing model parameters, exploring different neural network architectures, or incorporating additional contextual features to enhance model generalization.

10 Conclusion

This project aimed to model cricket performance by incorporating situational and opponent-specific factors through normalization of runs and temporal weighting of player statistics. While the feature engineering approach successfully captured aspects of player form and matchup dynamics, the predictive models—XGBoost and a neural network—demonstrated limited generalization to unseen data, as evidenced by the marginal improvement in test loss compared to training loss. This indicates that the complexity of cricket outcomes may require more sophisticated modeling techniques or additional features to achieve higher predictive accuracy.

Despite the challenges in prediction, the visualizations of player performance metrics over time provide valuable insights into trends and form, which can be useful for analysts and teams.

11 References

[1] "Github repo for the project." `https://github.com/debnath-d/contextual_cricket`.

[2] M. G. Jhawar and V. Pudi, "Honest mirror: Quantitative assessment of player performance in an odi cricket match," pp. 62–72.

[3] K. Passi and N. Pandey, "Increased prediction accuracy in the game of cricket using machine learning," *arXiv preprint arXiv:1804.04226*, 2018.

[4] F. Bharadwaj, A. Saxena, R. Kumar, R. Kumar, S. Kumar, and Ž. Stević, "Player performance predictive analysis in cricket using machine learning.," *Revue d'Intelligence Artificielle*, vol. 38, no. 2, pp. 231–238, 2024.

[5] M. Quazi, J. Clifford, and P. Datta, "Predicting cricket outcomes using bayesian priors," *arXiv preprint arXiv:2203.10706*, 2022.

[6] M. Sumathi, S. Prabu, and M. Rajkamal, "Cricket players performance prediction and evaluation using machine learning algorithms," in *2023 International Conference on Networking and Communications (ICNWC)*, pp. 1–6, 2023.

[7] "Cricsheet." `https://cricsheet.org/`.