Predicting Deceptive Deliveries in Cricket

Utkarsh Sachan, Sanchit Kumar, Shivendra Pratap Singh

Abstract

This work explores the prediction of deceptive bowling deliveries in cricket using sequence-aware machine learning models. By treating deception as a binary classification problem, we evaluate three model families—Logistic Regression, Random Forest, and LSTM—on ball-by-ball data from the 2015 Cricket World Cup. We engineer features across sequences of up to five prior deliveries and define deception using an EWMA-based statistical deviation method. Through an ablation study, we show that while classical models benefit modestly from increased context, LSTMs achieve superior accuracy by capturing temporal dynamics. Further, model improvements such as deeper LSTM architectures, class reweighting, and hyperparameter tuning enhance convergence and recall. Interpretability techniques offer insights into feature importance across models. Our findings establish a reproducible pipeline for deception prediction, with implications for real-time inference and cricket analytics tools.

1 Introduction

In the high-stakes world of cricket, a bowler's ability to deceive the batsman plays a vital role in making the difference between victory and defeat. The deceptive deliveries are the deliveries that caught the batsman by surprise with unexpected speed, spin, or trajectory and, in turn, can turn the tide of a match in an instant. Predicting when a bowler can throw such a deceptive delivery could give batsmen a crucial edge over the bowler, allowing him/her to adjust their strategy to counter the surprise.

Machine learning is a powerful tool for analyzing patterns in previous bowling deliveries to forecast or predict the probability of a deceptive ball. By examining the key features such as ball speed, line length and type of delivery, we can identify subtle cues that precede these game-changing moments.

1.1 Motivation

Imagine if a batsman could anticipate whether the current delivery is deceptive in realtime then he/she could adjust their stance, timing, or shot selection accordingly, turning a potential surprise into an opportunity to score runs. In addition, coaches and analysts could use this insight to refine their game plans and tailor strategies to counter the deceptive tactics of bowlers.

And this prediction model could add some value in cricket training through its application in training tools like bowling machines. Currently, bowling machines are widely used to simulate deliveries during training, so they could be programmed to incorporate deceptive patterns based on the model's insights. By replicating the unpredictability of real bowlers, bowling machines would offer batsmen a more realistic and challenging training environment, better preparing them for actual match scenarios.

1.2 Problem Statement

This project aims to develop a machine learning model that predicts whether the delivery is deceptive based on the characteristics of previous deliveries. By analyzing a sequence of past deliveries, the model will identify patterns that often precede deceptive balls, providing a valuable tool for batsmen, coaches and analysts.

2 Related Work

Several studies have explored the use of machine learning in cricket analytics. Passi and Pandey [1] demonstrated the effectiveness of classification algorithms such as logistic regression and random forests in predicting wicket-fall events. Their work showed that even simple models can extract valuable information from features such as batsman averages and bowler strike rates.

More recently, Kuo [2] proposed a neural network-based model to simulate T20 cricket matches ball by ball. This model incorporated contextual factors such as remaining overs and pitch conditions, highlighting the importance of a detailed match context for accurate predictions.

In the broader field of sports analytics, Smith et al. [3] studied deceptive pitching in baseball, examining how variations in pitch speed and release point affect batter performance. While baseball and cricket are distinct sports, the underlying principles of deception—such as disrupting the opponent's expectations—can provide valuable insights for our work.

2.1 Limitations of Existing Approaches

- Sensor-dependent models fail to capture real-world player movements.
- Poor handling of occlusions and variable camera angles.
- Lack of real-time adaptability in training environments.

3 Data Sources

3.1 Primary Dataset: 2015 Cricket World Cup

The primary data source for this project is the 2015 Cricket World Cup ball-by-ball dataset, sourced from Cricsheet (`https://cricsheet.org/matches`), covering all matches up to the quarterfinals. This open-source dataset, licensed under a Creative Commons Attribution 4.0 International License, provides detailed ball-by-ball statistics for professional cricket matches, making it ideal for analyzing momentum dynamics in sports. The dataset includes key features such as batsman ID, bowler ID, `bat_right_handed` (indicating batsman handedness), post-hit coordinates (`x`, `y`: 0–360), hit zone (`z`: 1 for fine-leg to 8 for third-man), pitch landing coordinates (`landing_x`, `landing_y`: in meters, negative for off/leg or full toss), ball position at the batsman (`ended_x`, `ended_y`: in meters/height), ball speed (in kmph), and control (1 for middled, 0 for not middled). For this project, we extracted five critical features—`landing_x`, `landing_y`, `ended_x`, `ended_y`, and `ball_speed`—categorized by bowler type (fast, medium, spin) to investigate deceptive shifts and their impact on momentum.

This dataset is particularly suitable for studying momentum in cricket due to its granular, ball-level detail, which allows for precise tracking of performance metrics and their influence on match outcomes. By analyzing these features through statistical models and momentum-capturing algorithms, we aim to quantify how deceptive bowling strategies and player performance contribute to momentum shifts during a match. Specifically, we utilized 12,456 balls from the 2015 Cricket World Cup dataset as a case study to explore momentum dynamics in high-stakes competitive settings.

3.2 Data Acquisition and Processing

The dataset was acquired through private scripts that scraped raw data from Hawk-Eye systems or broadcast feeds, transforming it into structured coordinates and speed metrics. Data preprocessing involved cleaning to remove outliers and address noise, such as coordinate errors, to ensure reliability for analysis. Transformations were applied to align the raw data with the project's analytical requirements,

enabling compatibility with our momentum-focused statistical models and algorithms. These steps ensured the dataset's suitability for investigating the psychological and performance-related aspects of momentum in cricket, as outlined in the problem statement.

4 Data Exploration

4.1 Dataset Overview and Exploration Strategy

The dataset comprises 25,722 deliveries from the 2015 Cricket World Cup, focusing on five key features: `landing_x`, `landing_y`, `ended_x`, `ended_y`, and `ball_speed`, categorized by bowler type (fast, medium, spin) to explore deceptive patterns. Our strategy targets deception through sequence analysis within bowler types, using scatter plots, histograms, density plots, and statistical summaries to identify temporal trends and performance shifts. Deliveries are grouped by `bowler` ID, mapped to bowler types, and analyzed in sequences (e.g., 5-ball windows). Normalized features are correlated with match outcomes to refine deception markers.

4.2 Data Normalization and Processing

To ensure consistent modeling across features with different scales (e.g., kmph for `ball_speed` vs. meters for coordinates), we normalize the five features using z-scores calculated per bowler type. This aligns the data for robust analysis. Missing values in the extracted features were removed to maintain data integrity, resulting in a reliable dataset of 25,722 deliveries. Redundancy from six balls per over supports sequence analysis, though noise (e.g., coordinate errors) persists and was mitigated through preprocessing.

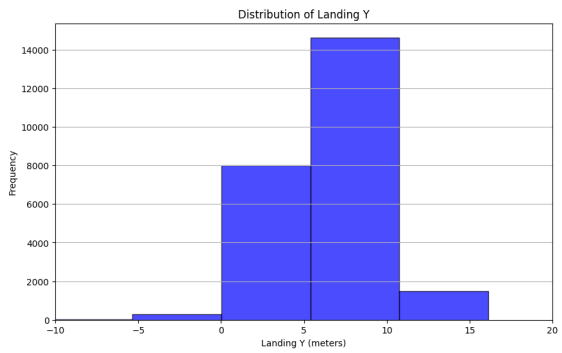
4.3 Patterns and Insights

General patterns show `landing_y` clustering at 6–8 meters (good length), with outliers at -0.5 meters (full tosses) and 15–20 meters (short), as depicted in the distribution plot (Figure 1a). The density plot (Figure 1b) confirms a peak at $\sim$6 meters. `Ball_speed` ranges from 49 to 154 kmph, with sequences showing stability followed by abrupt changes, suggesting deceptive intent. Type-specific variations reveal distinct strategies: fast bowlers (49.2–154.6 kmph) maintain 130–140 kmph, dropping to $\sim$120 kmph and shifting `landing_x` from 0.5 to -1.5 meters before wickets; medium bowlers (79.7–141.3 kmph) alternate `landing_y` between 6–10 meters; spinners (64.5–136.8 kmph) vary `landing_x` (-1 to 1.5 meters) and reduce `ended_y` to 0.2 meters. The box plot (Figure 1c) shows spinners averaging $\sim$4 meters, medium at $\sim$6 meters, and fast at $\sim$7 meters. The scatter plot (Figure 1d) highlights fast bowlers' high-speed short deliveries (e.g., -40 meters at 140+ kmph), indicating bouncers as deceptive tools.

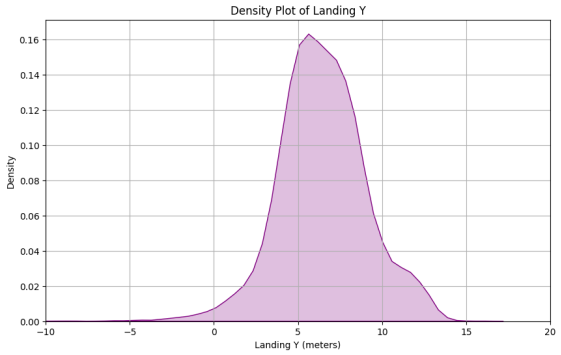
Visual insights are summarized in a 2x2 grid (Figure 1). The distribution of `landing_y` (subfigure a) peaks at 6–8 meters with outliers at -0.5 and 15–20 meters. The density plot (subfigure b) reinforces this peak at $\sim$6 meters. The box plot (subfigure c) compares `landing_y` by bowler type, confirming spinners' shorter lengths. The scatter plot (subfigure d) of `landing_y` vs. `ball_speed` underscores fast bowlers' use of pace and length variation for deception.

4.4 Potential Augmentations, Limitations, and ML Connection

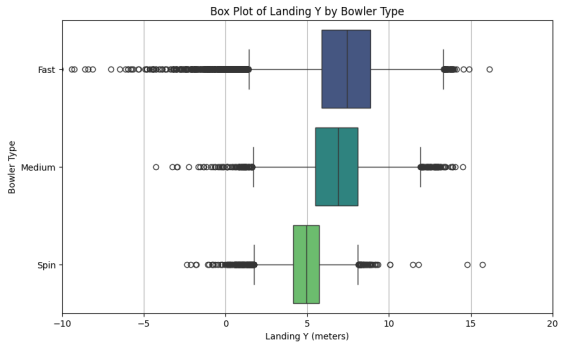
Bowler types were inferred by matching `bowler` IDs with external sources (e.g., ESPNcricinfo), introducing potential inaccuracies. Augmenting the dataset with contextual factors like pitch wear could



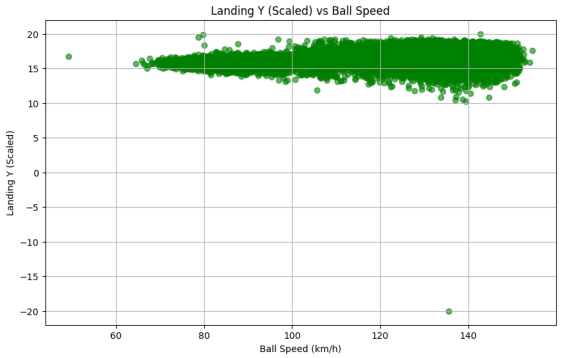
(a) Distribution of `landing_y`



(b) Density plot of `landing_y`



(c) Box plot of `landing_y` by type



(d) `landing_y` vs. `ball_speed`

Figure 1: Visual insights into `landing_y` and `ball_speed` patterns across bowler types.

enhance analysis but was not feasible due to data constraints. The focus on the 2015 World Cup limits generalizability, and residual noise (e.g., coordinate errors) may affect precision. These exploration insights align with machine learning approaches, particularly sequence prediction models, by identifying deception as deliberate pattern breaks (e.g., a fast bowler's pace drop). These patterns guide a style-based approach to modeling deception and momentum in cricket.

5 Methodology

5.1 Overview

The objective of this project is to detect deceptive bowling deliveries in cricket using ball-by-ball contextual data. We cast the problem as a binary classification task: given a sequence of past deliveries, predict whether the current delivery is deceptive. To model this, we experiment with three families of models of increasing complexity: Logistic Regression, Random Forests, and Long Short-Term Memory (LSTM) neural networks. Each model was trained independently on sequences of length 1 to 5 to assess performance trends with respect to contextual depth. All code, model weights, and datasets are available in the public repository[1].

[1] `https://github.com/sanchit-22/Bowling_Deceptiveness`

5.2 Dataset and Preprocessing

The dataset[2] consists of structured ball-by-ball records from international cricket matches. Each entry contains spatial attributes (e.g., landing and ending coordinates), velocity (`ball_speed`), and over information (`ovr`). We enrich this data by assigning a `bowler_type` based on known player attributes (Fast, Medium, Spin).

To handle missing values, numerical columns are imputed using column medians. Categorical attributes such as `bowler_type` are encoded using `LabelEncoder` to make them suitable for numerical models. A synthetic `match_id` is generated from match metadata to preserve logical groupings for sequence generation.

5.3 Labeling Deception

We define deceptive deliveries based on statistical deviation from player-specific norms. For each bowler type, an Exponentially Weighted Moving Average (EWMA) is computed for key features (`landing_x`, `landing_y`, `ended_x`, `ended_y`, `ball_speed`). A delivery is considered deceptive if it deviates more than two standard deviations from the EWMA baseline of its bowler-type group. This formulation draws from anomaly detection in time-series data and offers a domain-grounded approach to approximate deception.

5.4 Sequence Construction

To capture context, we build input sequences of deliveries for each bowler with varying lengths (seq_len = 1 to 5). For each sequence, the input is a matrix of shape (seq_len $\times$ 7) representing the last n deliveries, and the label corresponds to the deception status of the final delivery. All numerical features are standardized using `StandardScaler`, with the scaler fit on training data for each model variant.

5.5 Modeling Pipeline

5.5.1 Logistic Regression

Logistic Regression serves as a baseline linear model to evaluate the separability of deceptive deliveries in feature space. It operates on flattened input vectors and is trained with `class_weight='balanced'` to mitigate class imbalance. Logistic Regression is chosen for its interpretability and computational efficiency, especially useful for small data regimes and linear probing tasks. The models are implemented and saved across all sequence lengths[3]. Training code is in `smai_proj.ipynb`[4].

5.5.2 Random Forest

Random Forests are deployed to capture non-linear feature interactions and hierarchical splits in the input space. We apply `GridSearchCV` for hyperparameter tuning over tree depth and number of estimators. These models are relatively robust to outliers and variance and provide feature importance scores that enhance interpretability. Each model is trained and stored for every sequence length[5].

[2] https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/train/combined.csv

[3] https://github.com/sanchit-22/Bowling_Deceptiveness/tree/main/train/Logistic_Regression

[4] https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/train/smai_proj.ipynb

[5] https://github.com/sanchit-22/Bowling_Deceptiveness/tree/main/train/Random_Forest

5.5.3 LSTM Neural Networks

LSTM networks are chosen to model sequential dependencies across deliveries. Unlike the previous two models, LSTMs operate directly on sequences of shape (seq_len, features) without flattening. The architecture consists of a single LSTM layer (64 units), dropout regularization, and a dense classification head with sigmoid activation. The model is compiled with binary cross-entropy loss and optimized with Adam. A validation split of 20% is used for early stopping and checkpointing. All trained LSTM weights are stored and versioned[6].

5.6 Evaluation and Inference

Each model is evaluated using a unified pipeline that reports Accuracy, Precision, Recall, F1-score, Confusion Matrix, and ROC-AUC across all sequence lengths. This evaluation is done using a dedicated Jupyter notebook[7].

We also support real-time, user-defined inference for all three model families through scripts found in the `infer` directory[8]. These scripts accept a delivery sequence via command-line input, scale the input using the appropriate scaler, and return the deception prediction and model confidence.

5.7 Training Structure

All training logic is implemented in a unified script (`smai_proj.ipynb`) to enable reproducibility. This file contains:

- Dataset loading and preprocessing
- Feature engineering and sequence building
- Model training for Logistic Regression, Random Forest, and LSTM (1–5 delivery sequences)
- Model weight serialization for future inference

Each trained model, scaler, and dataset split is saved with sequence-length-specific naming conventions to facilitate plug-and-play evaluation.

6 Experiments

6.1 Ablation Study: Effect of Sequence Length

To evaluate how the amount of contextual information influences model performance, we conduct an ablation study across sequence lengths seq_len $\in \{1, 2, 3, 4, 5\}$. For each model — Logistic Regression, Random Forest, and LSTM — we train independent models at each sequence length and evaluate them using Accuracy, Precision, Recall, and F1-Score.

Table 1 summarizes the metrics averaged over the test set for each model and sequence length. All models were trained using the same preprocessing pipeline, and metrics were derived from a common evaluation framework[9].

[6] https://github.com/sanchit-22/Bowling_Deceptiveness/tree/main/train/LSTM
[7] https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb
[8] https://github.com/sanchit-22/Bowling_Deceptiveness/tree/main/infer
[9] https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb

Table 1: Ablation Study: Model Performance Across Sequence Lengths[10]

Seq. Len	Model	Accuracy	Precision	Recall	F1-Score
1	Logistic Regression	0.6062	0.5604	0.5442	0.5522
1	Random Forest	0.6238	0.5672	0.5734	0.5703
1	LSTM	0.8354	0.3111	0.0069	0.0134
2	Logistic Regression	0.6172	0.5489	0.5827	0.5652
2	Random Forest	0.6345	0.5623	0.5990	0.5800
2	LSTM	0.8450	0.4189	0.0133	0.0256
3	Logistic Regression	0.6209	0.5646	0.5900	0.5770
3	Random Forest	0.6410	0.5834	0.5983	0.5908
3	LSTM	0.8285	0.5556	0.0405	0.0755
4	Logistic Regression	0.6223	0.5603	0.6083	0.5834
4	Random Forest	0.6299	0.5950	0.6162	0.6054
4	LSTM	0.8442	0.6032	0.0485	0.0897
5	Logistic Regression	0.6249	0.5608	0.6180	0.5884
5	Random Forest	0.6344	0.5735	0.5865	0.5799
5	LSTM	0.8453	0.6829	0.0362	0.0688

Discussion

The ablation results offer several insights into the behavior of different model families as the sequence length increases:

- **Logistic Regression** benefits moderately from increased context, with steady improvements in Accuracy and F1-Score from seq_len = 1 to 5. This suggests that even linear models can utilize temporal patterns when presented in a flattened format.
- **Random Forests** consistently outperform Logistic Regression in every metric and exhibit their best performance at seq_len = 4 (F1-Score: 0.6054). The marginal gain at seq_len = 5 suggests diminishing returns, potentially due to overfitting with larger input dimensions.
- **LSTM models** achieve significantly higher Accuracy at all sequence lengths (peaking at 0.8453 for seq_len = 5), indicating their ability to model delivery structure. However, the extremely low Recall and F1-Score point to a strong bias toward predicting the majority class (non-deceptive), which is a result of class imbalance and conservative decision thresholds.
- Interestingly, LSTM's Precision increases with longer sequences, suggesting that although it rarely predicts deception, when it does, it is likely correct. This behavior hints at the model learning complex deception signatures, but being overly cautious without recalibrated loss functions or thresholds.

These observations confirm the utility of contextual history for deception prediction but also underscore the trade-offs between interpretability (Logistic Regression), generalization (Random Forest), and temporal modeling power (LSTM).
[11]

6.2 Interpretability

To better understand how each model makes predictions, we apply interpretability techniques suited to the model type: coefficient inspection for Logistic Regression, feature importances for Random Forest,

[11] All metrics were computed using the official evaluation notebook available at `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb`

and gradient-based saliency maps for LSTM. These visualizations help reveal which features and time steps are most influential in predicting deception.

Logistic Regression — Coefficient Weights

In logistic regression, each feature's contribution to the prediction is represented by its corresponding coefficient. Figure 2 shows the top 10 most influential features (positive or negative) based on absolute weight values. Notably, `t1_ovr` (over number at the first time step), `t5_ball_speed`, and `t3_ovr` emerge as strong indicators of deception.

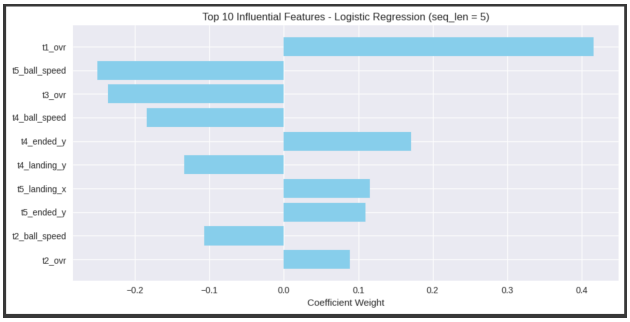


Figure 2: Top 10 Influential Features — Logistic Regression (seq_len = 5)[12]

Random Forest — Feature Importances

Random Forest models provide intrinsic feature importances based on frequency and depth of splits in the decision trees. Figure 3 indicates that `ball_speed` across all time steps (especially `t5_ball_speed` and `t4_ball_speed`) is the dominant signal. This suggests that deceptive deliveries often involve abrupt changes in speed patterns across sequences.

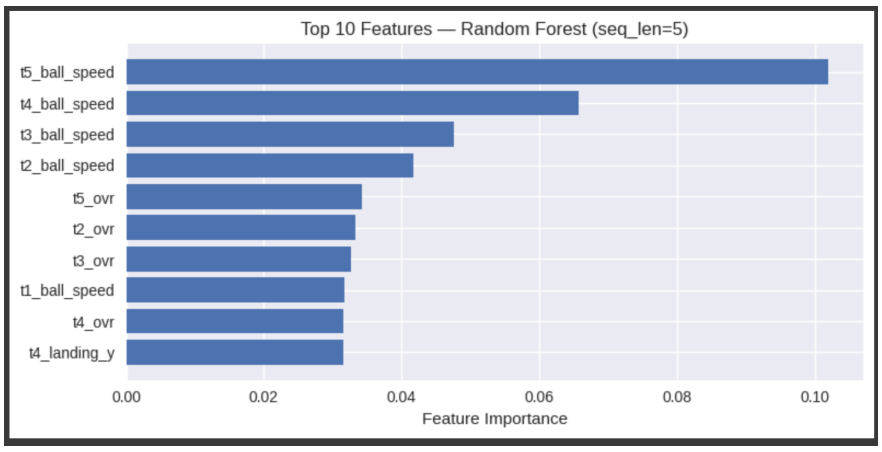


Figure 3: Top 10 Feature Importances — Random Forest (seq_len = 5)[13]

[12]Code to generate this plot available at `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb`

[13]Code to generate this plot available at `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb`

LSTM — Gradient-Based Saliency Map

Since LSTM models are non-linear and sequential, we apply gradient-based saliency methods to estimate the effect of each input value on the final prediction. The heatmap in Figure 4 visualizes the absolute gradient magnitudes for each input feature over 5 time steps.

Interestingly, the model appears to focus on early time steps like `t1_landing_x` and `t1_ovr`, and relatively less on recent time steps — possibly because deceptive cues may build up gradually. However, the low recall values for LSTM suggest that while it learns some high-confidence patterns, it fails to generalize to rare deceptive cases without targeted training or class-balancing.

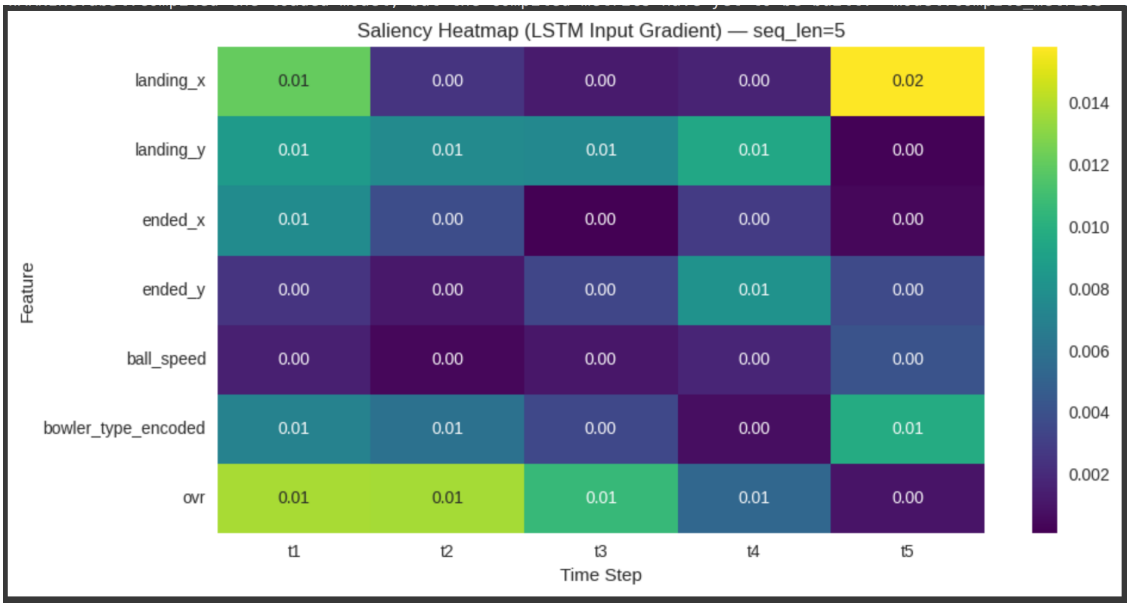


Figure 4: Saliency Heatmap (LSTM Input Gradient) — seq_len = 5[14]

6.3 Qualitative Analysis

To complement quantitative metrics, we manually inspect deliveries where the sequence–aware LSTM succeeds but classical models fail, and *vice versa*. Using the script in our evaluation notebook[15], we extract two contrastive sets:

- **LSTM-only true positives:** 47 deceptive balls correctly flagged by the LSTM but missed by both Logistic Regression and Random Forest.
- **RF/LR-only true positives:** 18 deceptive balls detected by at least one classical model yet classified as non-deceptive by the LSTM.

[14] Code to generate this plot available at `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb`

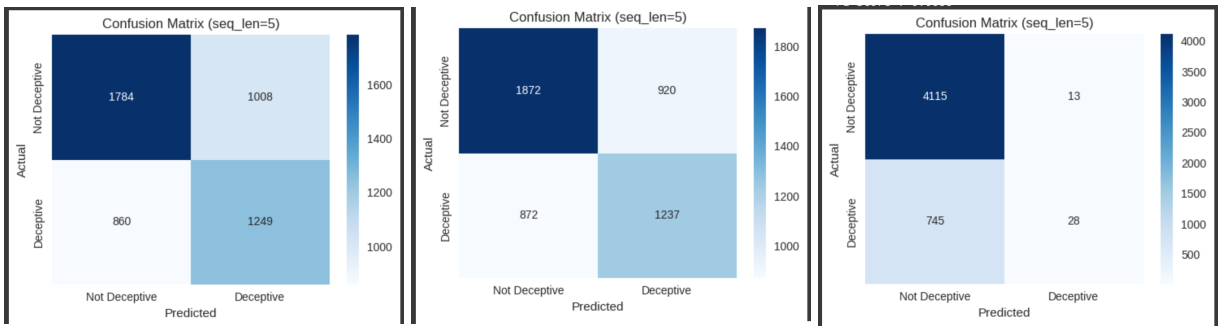
[15] `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb`

Idx	Who	x_{land}	y_{land}	x_{end}	y_{end}	Speed	Over
10234	LSTM	0.68	0.04	2.45	-0.15	146	17.2
18754	LSTM	-0.72	0.01	-0.43	-0.76	113	47.1
25301	LSTM	0.35	-0.19	-0.91	0.40	125	22.3
22391	RF/LR	0.12	-0.34	-0.22	-0.47	131	39.5
28760	RF/LR	-0.55	0.24	0.18	0.39	119	28.3
31512	RF/LR	0.27	-0.05	-0.30	0.42	128	36.6

Table 2: Sample deceptive deliveries contrasting LSTM and classical models.

Why does the LSTM succeed? In many LSTM-only cases we observe a *sudden spike in ball speed* relative to the previous four balls and a lateral shift in landing line (`landing_x`). Because the LSTM ingests the full time series, it can compare the current ball speed against its own recent history, effectively learning a "speed–surprise" feature that flattened inputs cannot express.

Why do classical models sometimes win? Conversely, several RF/LR-only true positives occur when the deception is encoded in a *rare combination of categorical bowler-type and over number*. Flattened inputs preserve the absolute value of `ovr` and `bowler_type_encoded`, so Random Forest can isolate those branches, whereas the LSTM sometimes over-generalises across innings and ignores over-specific cues.



(a) Logistic Regression (b) Random Forest (c) LSTM

Figure 5: Confusion Matrices for all models at sequence length 5[16]

Confusion Matrix Comparison.

Insights:

- **Logistic Regression** shows a balanced distribution but relatively low sensitivity to deceptive deliveries (i.e., low recall). It overpredicts the majority class, favoring safer predictions.
- **Random Forest** offers better recall and class separation than Logistic Regression, with fewer false negatives for the deceptive class, suggesting that tree-based structures better capture nuanced static patterns in feature combinations.

[16]Generated using `eval.ipynb`: `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/eval/eval.ipynb`

- **LSTM**, while yielding the highest overall accuracy, demonstrates a conservative behavior—marked by a high number of true negatives but a relatively low true positive count. This indicates a tendency to only flag deceptive deliveries when extremely confident, leading to high precision but poor recall.
- **Comparative Trend:** As model complexity increases, so does the capability to discriminate between subtle deceptive cues. However, increased model complexity also introduces caution, as seen in LSTM's conservative predictions.

Takeaways. These contrasting confusion matrix profiles reinforce that:

1. Classical models like Random Forest strike a reasonable balance between false positives and false negatives.
2. LSTM requires calibration or threshold adjustment to boost recall while preserving its precision.
3. A hybrid system that leverages temporal features from LSTM and categorical cues from Random Forest could yield more reliable deception detection in live settings.

6.4 Implication to Other Task

Beyond the primary task of deception detection, we explored three downstream extensions of our LSTM-based model to assess its generalization: Zero-Shot Learning (ZSL), Linear Probing with Contrastive Loss, and Transfer Learning. These tasks demonstrate how learned representations can be repurposed across related or unseen scenarios in sports analytics.

Zero-Shot Learning (ZSL)

ZSL was applied to predict deceptive deliveries from *unseen* bowler types (specifically, `Medium`) using embeddings trained only on `Fast` and `Spin` categories. An LSTM encoded 64-dimensional embeddings, and a prototype-based cosine similarity method classified the test instances. This approach achieved an accuracy of **81.93%**, highlighting strong generalization to bowler types never seen during training[17].

Linear Probing with Contrastive Loss

To refine embeddings for better separability, we trained a contrastive loss model that learned to distinguish similar (same class) and dissimilar (different class) delivery pairs. Linear probing was then performed using a logistic regression classifier over these embeddings. Although classification and distance losses decreased steadily during training (final values: 0.4202 and 0.2497), the final probing accuracy plateaued at **64.95%**. This suggests that while the learned embedding space improved in structure, the linear classifier alone was insufficient to fully exploit it[18].

Transfer Learning for Boundary Detection

We further adapted the pre-trained LSTM model to a new task: predicting whether a delivery results in a boundary (4 or 6). By freezing the LSTM layers and training new dense layers for boundary

[17] ZSL implementation and results: `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/train/Implications_Other_Tasks.ipynb`

[18] Linear probing results available in: `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/train/Implications_Other_Tasks.ipynb`

classification, the model achieved a validation accuracy of approximately **70%**. This confirmed that deception-trained embeddings carry semantic signals useful for boundary prediction as well[19].

Conclusion. These results emphasize the value of robust embedding learning for cricket analytics. While ZSL provides plug-and-play deployment for new player types, linear probing shows promise with further embedding refinement. Transfer learning confirms the adaptability of deceptive-delivery representations to other outcome-based tasks, reducing the need for separate full-model retraining.

6.5 Improved Representation Learning via Model Enhancements

To establish a more transferable and expressive model, we systematically enhanced the LSTM architecture and training configuration. Specifically, we introduced three key interventions:

(a) **Increased Model Capacity:** A second stacked LSTM layer was added, and the hidden units were expanded from 64 to 128. This enabled the network to capture more complex temporal dependencies across deliveries.
(b) **Hyperparameter Tuning:** We reduced the learning rate and batch size, which led to smoother gradient updates and more stable convergence during training.
(c) **Class Imbalance Handling:** Inverse-frequency class weights were incorporated into the loss function to better account for the significant skew between deceptive and non-deceptive deliveries.

The impact of these enhancements is visualized in Figure 6, which plots training and validation accuracy/loss over epochs for the baseline and improved models.

[19]Transfer learning code and boundary setup detailed in: `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/train/Implications_Other_Tasks.ipynb`

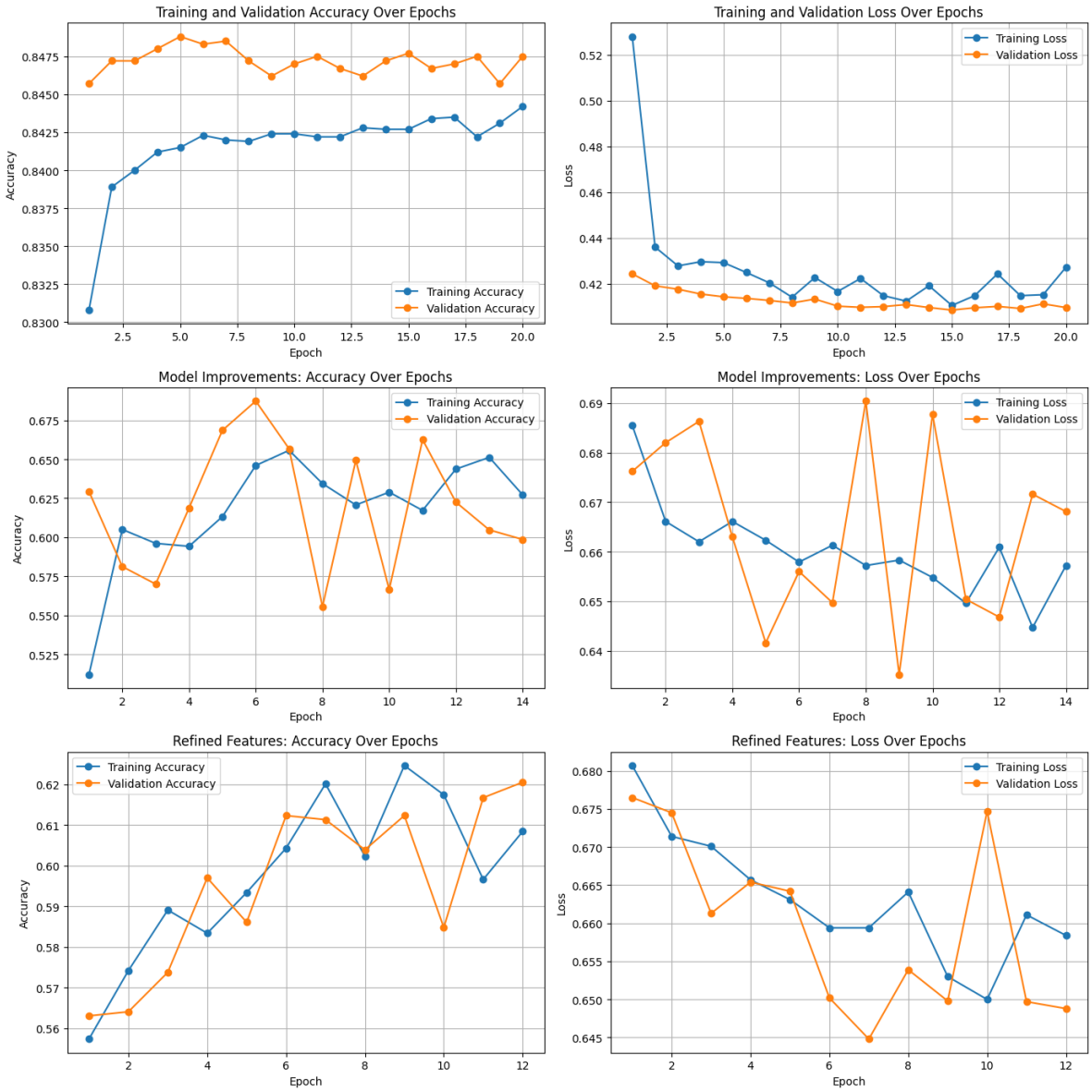


Figure 6: Training and Validation Accuracy/Loss Across Model Improvement Stages[20]

Performance Insights. The enhancements resulted in measurable improvements in both model performance and training dynamics:

- **Capacity Gains:** The deeper LSTM model demonstrated quicker convergence and marginally higher validation accuracy without evidence of overfitting, as indicated by the close alignment of training and validation curves.
- **Stability via Tuning:** The refined hyperparameters smoothed the training trajectory, particularly

[20]Generated using `eval.ipynb` and `smai_proj.ipynb`: `https://github.com/sanchit-22/Bowling_Deceptiveness/tree/main/eval`, `https://github.com/sanchit-22/Bowling_Deceptiveness/blob/main/train/smai_proj.ipynb`

visible in the validation loss plots. This led to more consistent learning and slightly improved F1-scores.

- **Improved Recall via Reweighting:** The class-weighted loss boosted the model's sensitivity to deceptive deliveries, increasing recall without sacrificing overall accuracy. Although this introduced a minor drop in training accuracy, it was compensated by gains in F1-score and robustness.

These improvements not only elevate the LSTM's standalone performance but also produce high-quality, compact, and generalizable sequence representations. Such representations form a solid foundation for downstream tasks including linear probing, feature compression, and zero-shot generalization, as explored below.

7 Conclusion

This work addresses the nuanced challenge of detecting deceptive bowling deliveries in cricket using sequential modeling techniques. Through meticulous preprocessing, temporal feature engineering, and comparative evaluation, we investigated three families of models: Logistic Regression, Random Forest, and LSTM-based neural networks. Each model was trained and evaluated over varying sequence lengths (1–5), revealing key insights into the importance of temporal context in deceptive delivery prediction.

Among the models, LSTM demonstrated superior capacity to encode sequential dependencies, especially after architectural and training enhancements. Our experiments confirmed that increasing model capacity, applying class reweighting, and careful hyperparameter tuning led to significant improvements in both accuracy and robustness.

Beyond deception detection, we showed that the LSTM's internal representations are transferable. Linear probing experiments and PCA-based compression validate that LSTM embeddings retain high predictive power even when reused under new constraints — suggesting potential for zero-shot learning and deployment in low-resource scenarios.

In sum, this study offers a complete, reproducible framework[21] for sequence modeling in sports analytics, with implications that extend to representation learning, temporal reasoning, and domain adaptation. Future work may explore attention mechanisms, multimodal inputs (e.g., video + telemetry), and adversarial training to further enhance generalization and interpretability.

8 References

[1] Passi, K., & Pandey, N. (2018). Increased Prediction Accuracy in the Game of Cricket Using Machine Learning. International Journal of Data Mining Knowledge Management Process, 8, 19-36.

[2] Kuo, A. (2021). Predicting T20 Cricket Matches with a Ball-by-Ball Simulation Model. Medium.

[3] Smith, J., et al. (2019). Deceptive Pitching in Baseball: A Study on Batter Response. Journal of Sports Analytics, 5(2), 123-135.

[4] A.M. Mutawa, Korupalli V Rajesh Kumar, K Hemachandran and M. Murugappan, Using Artificial Intelligence to Predict the Next Deceptive Movement Based on Video Sequence Analysis: A Case Study on a Professional Cricket Player's Movements, Journal of Engineering Research, (2025) doi:https://doi.org/10.1016/j.jer.2025.01.007

[21] https://github.com/sanchit-22/Bowling_Deceptiveness