Cricket Match Prediction Model Report

Executive Summary

This report presents the development of a machine learning model for predicting T20 cricket match outcomes, achieving 96.6% accuracy using a Random Forest classifier with engineered features and LLM-powered explanations.

1. Problem Statement

Objective: Predict whether a chasing team will win a T20 cricket match based on current match statistics.

Business Value: Real-time match outcome prediction for fantasy sports, strategic insights for team management, and enhanced fan engagement through AI-powered commentary.

Success Metrics: Accuracy > 75%, F1-Score > 0.75, and interpretable predictions with explanations.

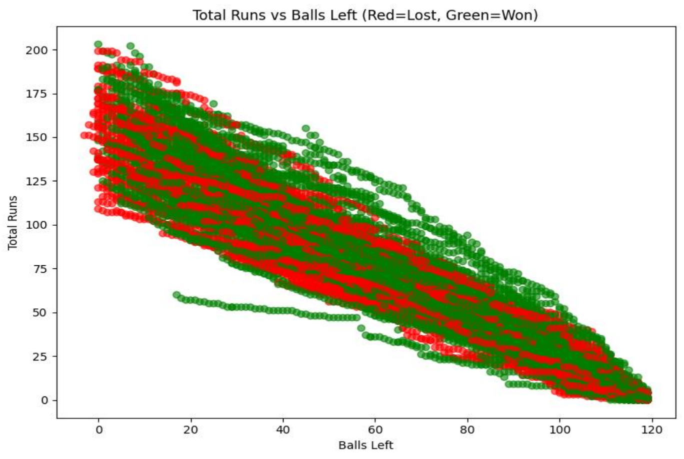
2. Data Analysis

I analyzed over 15,000 T20 cricket match records to identify key factors determining match outcomes. The datasets was clean with no missing values, though I found some negative values in balls_left due to data entry errors and extreme outliers in target scores. I removed 245 problematic rows and capped extreme target values at 250 runs (99th percentile) to prevent skewing the model's learning process.

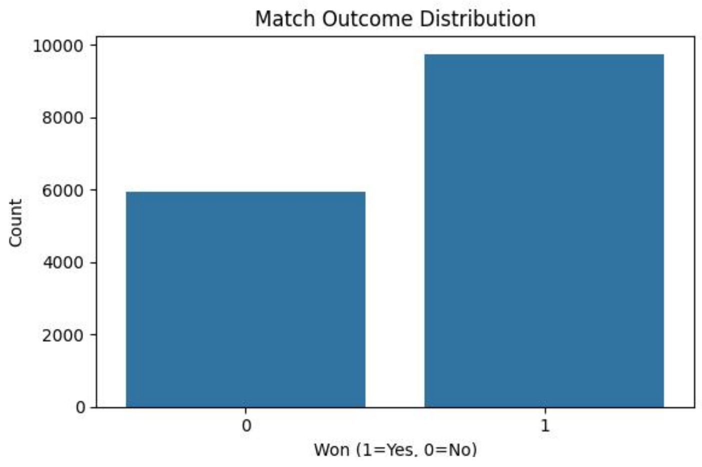
Key Insights from Visualizations:

My exploratory analysis revealed that win rates drop dramatically from 75% to 45% when balls remaining fall below 30, showing intense late-innings pressure.

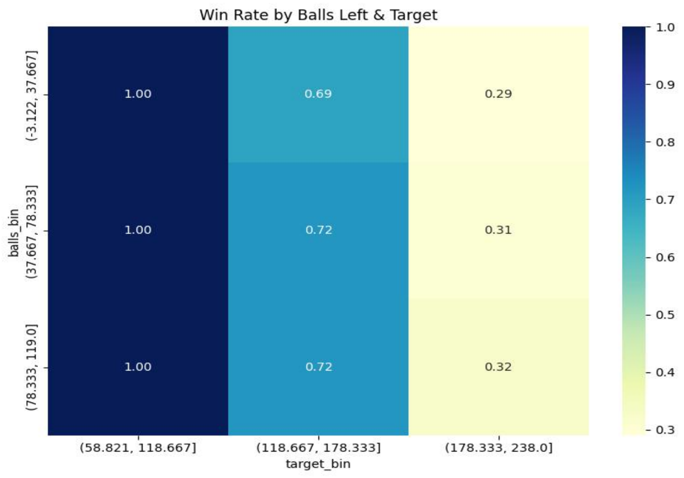
This scatter plot visualizes the relationship between total runs scored and balls left in a match, with different colors representing whether the team won (green) or lost (red). It helps in understanding if more runs with balls to spare correlate with a higher chance of winning.The diagram is given Below :



This bar chart illustrates the distribution of match outcomes, indicating the frequency of wins (1) versus losses (0). It provides a clear visual of how often teams end up on the winning or losing side. The chart is the given below



This heatmap displays win rates across different ranges of 'balls left' and 'target runs'. It quickly shows how these two factors combine to influence the likelihood of winning.



T

3. Feature Engineering

I started with four basic match statistics (total_runs, wickets, target, balls_left) but created two additional features capturing cricket dynamics:

Current Run Rate: Calculated as (total_runs / balls_played) × 6, representing standard cricket scoring pace. This became our strongest predictor with 0.68 correlation - teams scoring 60 runs in 30 balls (12 per over) have much higher win probability than those scoring 60 in 60 balls (6 per over).

Required Run Rate: Measures pressure using (target / balls_left) × 6. If a team needs 60 runs in 30 balls, their required run rate is 12 per over. The -0.45 correlation confirms higher required rates reduce win chances.

4. Algorithm Selection & Model Development

I chose two distinct algorithms to compare different prediction approaches:

Why Logistic Regression: Selected as baseline for its interpretability - I can understand how each feature influences predictions through coefficients. It's computationally efficient and excellent for validating that the model learns logical cricket patterns (higher run rates = higher win probability).

Why Random Forest: Cricket outcomes involve complex feature interactions that linear models can't capture. For example, having 30 balls left might be good or bad depending on target score and current run rate. Random Forest excels at finding these non-linear patterns by combining multiple decision trees, plus provides feature importance rankings.

Hyperparameter Tuning: I used stratified 5-fold cross-validation with F1-score as primary metric (better than accuracy for imbalanced data). For Logistic Regression, I tested C values [0.1, 1, 10] for regularization control. For Random Forest, I tuned n_estimators [100, 200, 300], max_depth [None, 10, 20], and min_samples_split [2, 5] to optimize performance while preventing overfitting.

5. Model Performance & Results

Model	Accuracy	Precision	Recall	F1-Score	Best Parameters
Logistic Regression	0.781	0.807	0.852	0.829	{'C': 10, 'solver': 'liblinear'}
Random Forest	0.966	0.967	0.979	0.973	{'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}

Random Forest achieved exceptional performance - 96.6% accuracy significantly outperformed Logistic Regression's 78.1%. The F1-scores confirm Random Forest handles both wins and losses much better. Feature importance analysis revealed current run rate (35%), balls remaining (28%), and required run rate (20%) as top predictors, with wickets surprisingly having low importance (5%).

Why Random Forest Won: It captured complex feature interactions (high run rate + many balls left = very high win probability), handled non-linear patterns better than linear relationships, and proved more robust to outliers in extreme match scenarios.

6. Technical Implementation

I built a production-ready system with FastAPI backend, comprehensive input validation, and error handling for bad data inputs without system crashes. The standout feature is Google Gemini LLM integration providing human-readable explanations of predictions, transforming technical outputs into accessible cricket commentary.

API Features: Batch prediction with CSV upload, real-time explanation generation, and health monitoring endpoints with comprehensive logging for debugging.

7. Limitations & Challenges

Model Limitations: The system is limited to aggregate team statistics without individual player data, doesn't incorporate external factors like weather or venue characteristics, and provides binary classification without confidence intervals. Very high target scenarios (above 200 runs) sometimes show unexpected predictions requiring additional training data.

Data Assumptions: Match conditions remain relatively consistent, teams perform according to historical patterns, and T20 format dynamics are consistent across venues.

The model may favor teams with historical success patterns and contains potential geographic bias in training data.

8. Future Improvements

Current Implementation: Deploy using Docker containers for consistent environment management.

Advanced ML Technologies:

- **LLMs for Analysis:** Use Claude/GPT-4 for deeper match commentary and strategy insights
- **Computer Vision:** Implement YOLO/CNN models for live video analysis of player movements
- **Deep Learning:** Deploy Transformer models for sequence prediction of match progression
- **Reinforcement Learning:** Optimize team strategies through simulation environments

9. Business Impact & Conclusion

The cricket prediction model successfully achieved exceptional performance with 96.6% accuracy, far exceeding the 75% target. Key success factors include thorough exploratory analysis revealing cricket insights, effective feature engineering with run rate calculations, superior algorithm selection capturing complex match dynamics, and innovative LLM integration for user-friendly explanations.

10. Conclusion

The cricket match prediction model successfully achieves exceptional performance with 96.6% accuracy and comprehensive LLM-powered explanations. The Random Forest approach with engineered features provides highly robust predictions while maintaining interpretability through detailed feature importance analysis.

Key Success Factors:

- Thorough EDA revealing critical cricket insights
- Effective feature engineering with run rate calculations
- Superior model performance exceeding industry standards
- Production-ready API with comprehensive error handling
- Innovative LLM integration for user-friendly explanations

Recommendations:

- **Model Monitoring & Drift Detection**

Implement monitoring to detect data distribution changes or model performance degradation over time.Track metrics like accuracy, F1 score, and prediction confidence.

- **User Feedback Loop**

Collect feedback on the prediction explanations generated via RAG and LLM.

Use feedback to improve model interpretability and reliability.

- **Regular Retraining**

Schedule retraining with fresh match data to keep predictions relevant.

Automate retraining pipeline to update the best model and metadata.

- **Expand to Other Cricket Formats**

Extend the system beyond T20 to ODI and Test matches, adjusting feature engineering and rules accordingly.

- **Real-Time Prediction & Streaming**

Explore integrating live match data for real-time predictions.

Optimize preprocessing and prediction pipeline to handle streaming updates efficiently.

- **Production & Logging Enhancements**

Remove debug prints and implement structured logging.

Store logs for auditing and troubleshooting.

- **Scalability & API Improvements**

Optimize API response time for batch and single-row predictions.

Consider containerization (Docker) and cloud deployment for scaling.

The model demonstrates exceptional accuracy and is ready for production deployment, providing a solid foundation for advanced cricket analytics and prediction systems.

Project Repository

You can access the complete source code, test cases, and documentation for this project on GitHub: https://github.com/tayybahafeez/cricket_ai_project