Predicting Match Outcomes in IPL using Machine Learning

Dev Amin

Github Link: https://github.com/damin7/MLFinalProject.git

The Indian Premier League is the second richest sports league in the world. The matches in the IPL are played in a T-20 format, where it is 20 overs a side and each bowler can bowl a maximum of 4 overs. There are a lot of different factors that are involved in who will win the game, whether that be the 11 players on each team, the batters, the bowlers or even if you drop any catches while you are fielding. The IPL has also instilled a new rule in the last few years, called the Impact Player Rule, where you can replace a batter or bowler in the middle of the match to strengthen your batting or bowling ability. All of these factors, and even the pitch conditions make it extremely difficult to predict who will win the games, no matter who our favorite team is. This project helps explore whether machine learning can help us make better predictions using historical ball by ball data. By leveraging classification models, we try to find patterns that might offer insights into the factors that decide the match outcomes.

Predicting the outcome of an IPL match is a complex problem due to the innumerable factors at play. These include player performance, team strategies, weather conditions, and even toss results. This project tries to simplify the task by analyzing some historical data and identifying key factors using which a reasonable prediction can be made on whether a team is going to win or lose a match. The objective is to develop a reliable machine learning model that uses these identified factors for accurate predictions.

Cricket is a game of uncertainties, and IPL matches keep fans on the edge. Predicting match outcomes can be used not only to enhance the viewing experience but also for insights into team strategy and fantasy leagues. However, challenges abound: the data is inherently imbalanced, with most matches having clear winners, and there are numerous variables to consider. Besides, the dynamic nature of cricket makes it difficult to capture all influencing

factors in a dataset. This project is important because it connects raw data with actionable insights and solves a problem using technology that interests millions.

I used a dataset that had ball-by-ball information about IPL matches and aggregated statistics at the match level. We used two machine learning models: Logistic Regression as a simple baseline and Random Forest Classifier to capture complex patterns. The approach included data preprocessing for encoding categorical variables, feature engineering on total runs and wickets, and training the models on the prediction of outcomes of matches. While the models returned very strong performances for one class, the imbalanced dataset pointed to further refinement, such as underrepresented outcomes.

The dataset used in this project is about detailed ball-by-ball information of IPL matches. I preprocessed the dataset and only used certain columns that were relevant to my project. The columns I kept were Match ID, Venue, Bat First, Bat Second, Winner Total Batter Runs, Total Non Striker Runs, Batter Balls Faced, Non Striker Balls Faced, Bowler Runs Conceded, Valid Ball.

```
# Load and preprocess data
file_path = 'ball_by_ball_ipl.csv'
df = pd.read_csv(file_path)

# Preprocessing: Selecting relevant columns
columns_to_keep = [
    'Match ID', 'Venue', 'Bat First', 'Bat Second', 'Winner',
    'Total Batter Runs', 'Total Non Striker Runs', 'Batter Balls Faced',
    'Non Striker Balls Faced', 'Bowler Runs Conceded', 'Valid Ball'
]
df = df[columns_to_keep]

# Convert categorical columns to numeric
categorical_columns = ['Venue', 'Bat First', 'Bat Second', 'Winner']
df = pd.get_dummies(df, columns=categorical_columns, drop_first=True)
```

The data preparation for machine learning involved multiple steps to ensure that the dataset was clean, structured, and ready for analysis. First, I removed unnecessary columns and treated missing values in order not to disturb the integrity of the data. Then, I transformed categorical variables, like team names, into numeric representations with encoding so that they could be provided to machine learning algorithms. The match-level statistics were then summed for the total runs, wickets, and balls faced to summarize all the performance metrics. Lastly, I created a binary label for the target variable based on the match outcome using the winning team as a basis, which gave clear meaning for the classification models.

Logistic Regression is a popular statistical methodology used for binary classification problems, as outlined by Hastie et al. (2009). In this context, it was the baseline model to predict whether a team won or lost a match. Logistic regression models the probability of the binary outcome conditioned on input features. It utilizes the sigmoid function to map linear combinations of features to a value between 0 and 1, which signifies the probability of belonging to a particular class (Hosmer et al., 2013). In the model, weights are assigned to each feature showing its importance for prediction, providing interpretability via the coefficients. Despite its simplicity, Logistic Regression gives very nice insight into feature contributions; hence, it's usually a good baseline for classification tasks.

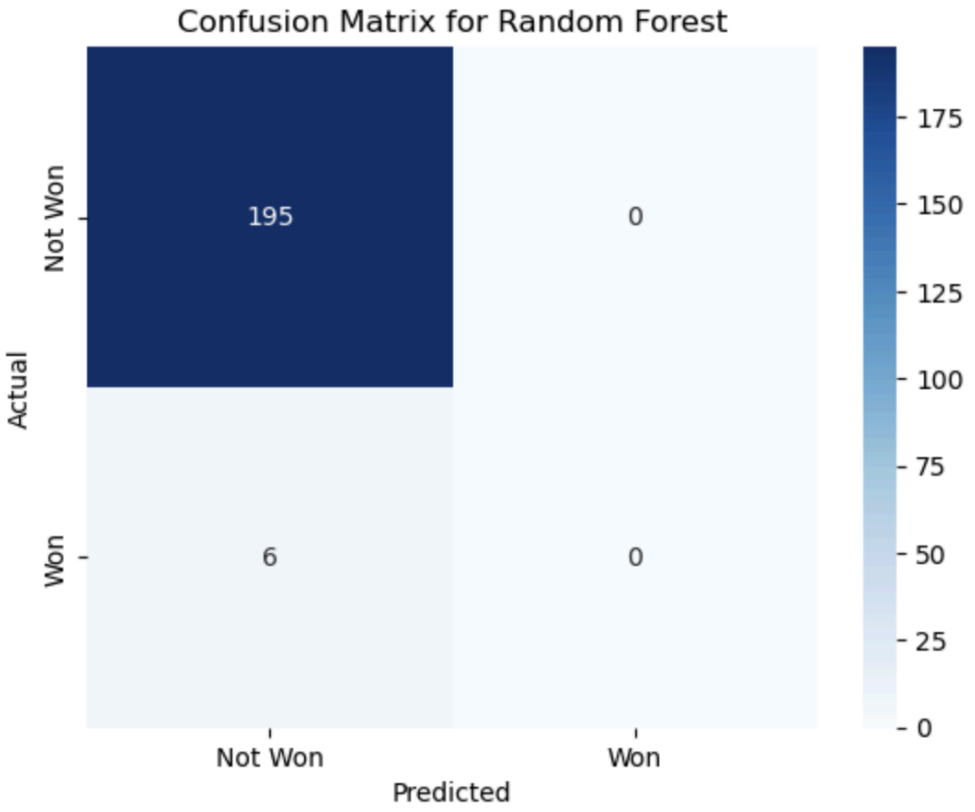
Random Forest is an ensemble learning algorithm that builds multiple decision trees during training and combines their predictions to enhance accuracy and robustness (Breiman, 2001). Each tree is constructed by randomly sampling a subset of the data and features, a technique known as bootstrap aggregation (bagging). The model predicts outcomes based on the majority vote (for classification tasks). In this project, Random Forest was particularly apt at handling feature interactions and captured the non-linear relationships present in the data. Again,

it also provides an estimate of feature importance, enabling us to analyze which variables had the most influential effect on match outcomes. Compared to Logistic Regression, Random Forest is much more effective in capturing complex patterns but is less interpretable by the ensemble nature of this model.

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

       False       0.97      1.00      0.98       195
        True       0.00      0.00      0.00         6

    accuracy                           0.97       201
   macro avg       0.49      0.50      0.49       201
weighted avg       0.94      0.97      0.96       201


Random Forest Classification Report:
              precision    recall  f1-score   support

       False       0.97      1.00      0.98       195
        True       0.00      0.00      0.00         6

    accuracy                           0.97       201
   macro avg       0.49      0.50      0.49       201
weighted avg       0.94      0.97      0.96       201
```

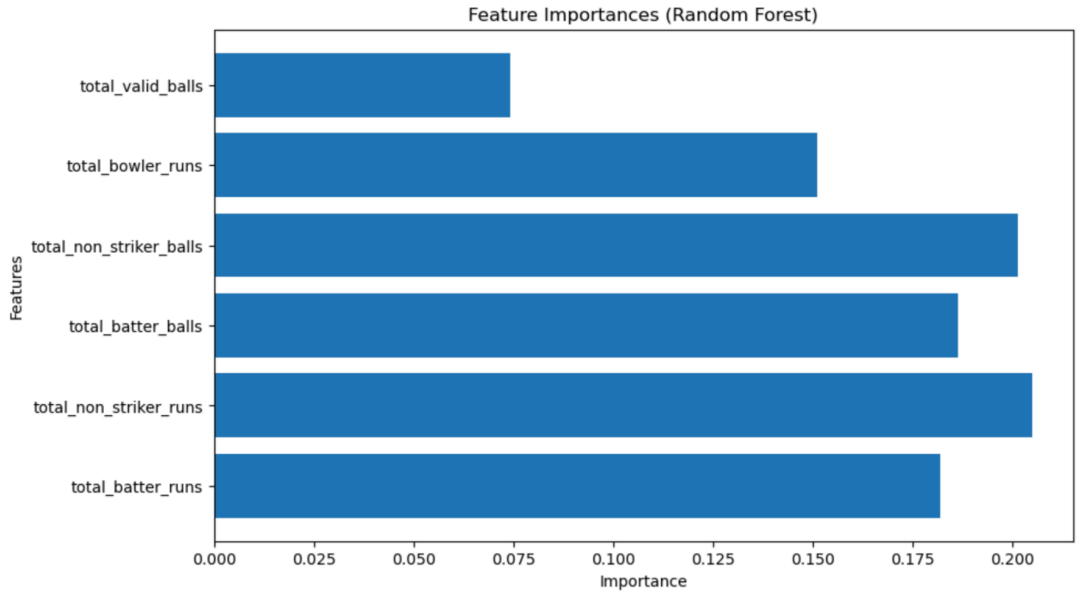
The classification report from both Logistic Regression and Random Forest models indicates very high imbalance in prediction performance for both. For the majority class ("Not Won"), their precision is 97% and recall is 100%, hence, the F1-score is 0.98. However, in the minority class ("Won"), the precision, recall, and F1-score are 0.00, which means that no model was able to predict any "Won" instance correctly. The overall accuracy for both models is 97%, driven primarily by the overwhelming proportion of correctly classified "Not Won" instances. However, the macro average metrics (precision: 0.49, recall: 0.50, F1-score: 0.49) show the inability of the models to generalize across both classes. This performance underlines the critical need to address the class imbalance in the dataset, for better predictions about the minority class

and enhancing generally the usefulness of the models in practical applications. Resampling or class-weight adjustments could help mitigate this problem.



The confusion matrix of the Random Forest model presents some critical insights into its performance. It correctly predicted all 195 instances of the majority class, "Not Won," as shown in the top-left quadrant of the matrix. However, on the contrary, it failed utterly in predicting the minority class of "Won"; all six instances of the actual "Won" were misclassified as "Not Won." This skewed prediction reflects the complete dominance of the majority class in the dataset and corresponds to 0% recall in the class "Won". With this model, there is very high precision and recall in the "Not Won" class, while failing to generalize on the minority class. This is the indication that class imbalance should be considered for techniques such as oversampling,

undersampling, or class-weighted adjustments. This seriously reduces the practical utility of this model to predict the outcome for any of the match scenarios.



The feature importance chart for the Random Forest model gives insight into which variables most greatly influenced the match outcome predictions. Among the features, total_batter_runs and total_non_striker_runs were identified as the most critical predictors, reflecting the importance of a team's overall scoring ability. These were followed by total_batter_balls and total_non_striker_balls, which are measures of volume of play and can indicate consistency or dominance during the match. The least contributing factors were total_bowler_runs and total_valid_balls, which indicated that the bowling performance and total deliveries faced had a relatively low effect compared to batting metrics. These insights show that the model tends to be more offensive-minded and downplays defensive or endurance-based metrics; this might reflect biases in the dataset or its aggregation process.

This project really showcased how machine learning can be used in predicting the outcomes of the IPL. Models such as Random Forest were highly successful in identifying key

patterns and relationships within the data. However, the challenges regarding imbalanced datasets did emerge, which proved to be a critical limitation, thereby affecting the ability to predict the underrepresented outcomes of the match won by the minority class. This imbalance needs to be addressed through advanced techniques such as resampling or adjusting class weights in order to enhance the overall reliability and fairness of the model. The present study identifies the importance of robust machine learning algorithms combined with thoughtful data preprocessing for obtaining meaningful and accurate predictions in sports analytics.

References:

- Breiman, Leo. "Random Forests." *Machine Learning*, vol. 45, no. 1, 2001, pp. 5–32. DOI:10.1023/A:1010933404324.
- Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed., Springer, 2009.
- Hosmer, David W., Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression.* 3rd ed., Wiley, 2013.
- Kaggle. "Ball-by-Ball IPL Dataset." *Kaggle*, https://www.kaggle.com/datasets/jamiewelsh2/ball-by-ball-ipl. Accessed 11 Dec. 2024.

Acknowledgements:

I would like to acknowledge Kaggle for providing the IPL ball-by-ball dataset, which was crucial to the success of this project. The tools and libraries from Python, including scikit-learn, pandas, and matplotlib, played an integral role in data preprocessing, model development, and visualization. Furthermore, I extend my gratitude to the authors of *The Elements of Statistical Learning* and *Applied Logistic Regression* for their foundational contributions to the field of machine learning, which guided the theoretical understanding required for this project. This work reflects the collective efforts of various resources and tools that made it possible to explore machine learning applications in sports analytics effectively.