Project Charter

Group Number 11

Tegveer Ghura

Tuesday, October 11, 2023

1 Overview

Cricket, the world's second most popular sport in the world, with over 2.5 billion fans, has captivated audiences globally for centuries[1]*. In fact, the world's first international cricket match was played between the US and Canada in 1844 and an estimated $100,000 to $120,000 worth of bets were placed on the match*[2]*! In recent decades, the advent of Twenty20 cricket, the shortest format of the game recognized by the ICC (International Cricket Council), has revolutionized the game, leading to the creation of high-energy tournaments like the Indian Premier League (IPL). The IPL, with its thrilling matches and star-studded lineups, has become not only a phenomenon that fans eagerly await every Indian summer, but also a massive commercial success, garnering millions of viewers and generating revenue in billions of US dollars*[3]*.*

As cricket fever rises, so does the interest in predicting match outcomes, a fascination that extends to the realm of sports betting. Betting on cricket is a popular pastime in India, and the IPL is no exception. However, illegal betting is a major problem in the country, and has been linked to corruption and match-fixing scandals[4]*. The IPL, being one of the most-watched cricket leagues globally, is a focal point of such betting activities, both legal and unauthorized, prompting questions about the accuracy and reliability of predictions.*

In this context, our project delves into the exciting intersection of cricket, data science, and predictive modeling. We aim to tackle the challenge of predicting IPL match outcomes ball by ball by utilizing machine learning classification algorithms. The central questions guiding our endeavor are:

1. *Can machine learning classification techniques, specifically ensemble methods, accurately predict the outcome of every ball bowled in an IPL match?*

[1] *A history of cricket: The World's second-most popular sport*. TheCollector. (2022, April 8). https://www.thecollector.com/history-of-cricket-worlds-second-most-popular-sport/

[2] Harris, J. (2001, May 16). *Cricket in Canada: A historical review*. ESPNcricinfo. https://www.espncricinfo.com/story/cricket-in-canada-a-historical-review-106282

[3] Aripaka, R. (2023, April 30). *IPL: How India became home to the biggest, baddest cricketing league in the world*. The Economic Times. https://economictimes.indiatimes.com/news/sports/ipl-how-india-became-home-to-the-biggest-baddest-cricketing-league-in-the-world/articleshow/99885321.cms?from=mdr

[4] ESPN Digital Media Private Limited. (1899, November 30). *Full coverage of the IPL spot-fixing allegations*. ESPNcricinfo. https://www.espncricinfo.com/story/full-coverage-of-the-ipl-spot-fixing-allegations-636375

2. *Which ensemble methods perform best in both generalizing to unseen data and catering to the user's request, such as highest probability of predicting a ball leading to a wicket, specified in the deployed application?*

By exploring these questions, we aim to shed light on the potential of tree-based ensemble methods, some of which are easily explainable, in enhancing our understanding of the game and making informed predictions in the dynamic world of modern-day cricket.

1.1 Project Background and Description

Cricket, one of the world's most beloved sports, has experienced a seismic shift with the advent of Twenty20 cricket, since 2003, and marquee tournaments like the Indian Premier League (IPL). Amidst the excitement, a pressing challenge emerges: the proliferation of illegal betting activities, especially through online mediums in the modern world, which threatens the game's integrity and fairness[5]. Not only does it affect the game, but society as a whole, given the cruel consequences of unfulfilled payments and the avariciousness that accompanies unregulated gambling. Addressing this issue is not just an imperative for the sporting community, but also a moral obligation to protect the well-being and ethics of society.

Given the aim of this project, one could argue that predicting every ball in a cricket match could provide an unfair advantage to bettors, if they gain access to such a model, potentially encouraging increased betting and manipulation of odds, especially if the predictions are highly accurate. This scenario could undermine the integrity of the sport and the fairness of betting activities. However, predicting every ball of a cricket match can lead to a deterrent to illegal betting activities for copious reasons. Comparing the model's predictions with that of sportsbooks or odds can help to identify suspicious betting patterns. For example, if a large number of bets are placed on a particular outcome that is then predicted to be unlikely, this could be a sign of match-fixing or other illegal activity.

Predictive models, hence, could be utilized by the Board of Control for Cricket in India (BCCI), the national governing body of cricket in India, to track illegal activity across Indian regions. Additionally, by monitoring betting patterns over time, it can be possible to identify individuals or groups who are consistently engaged in suspicious betting activity, enabling Indian national investigative agencies, like the Central Bureau Of Investigation (CBI), to catch the big players and rackets involved in corruption[6]. Therefore, machine learning classification models also have the potential to not only detect but also prevent humongous illegal bets from occurring.

[5] Ghosh , D. (n.d.). *Man arrested for online IPL betting: Kolkata News - Times of India*. The Times of India. https://timesofindia.indiatimes.com/city/kolkata/man-arrested-for-online-ipl-betting/articleshow/100433735.cms?from=mdr

[6] *CBI books Delhi-based company for connections with International Online Betting Platform Dafabet*. The Economic Times. (n.d.). https://economictimes.indiatimes.com/news/india/cbi-books-delhi-based-company-for-connections-with-international-online-betting-platform-dafabet/articleshow/94618084.cms?from=mdr

Somewhat ironically, it can help to educate bettors about the risks of illegal betting[7]. By understanding the factors that influence the outcome of a cricket match, bettors can make more informed decisions about placing bets. This can help to reduce the number of people who fall victim to illegal betting scams or who become addicted to gambling. Finally, it can help to promote fair play and the spirit of competition. When all bettors have access to the same information, it creates a level playing field and reduces the temptation to engage in illegal activities such as match-fixing. This phenomenon is known as information asymmetry, a term commonly used in economics and extensively studied in relation to betting in sports, like the National Football League (NFL) in the United States[8].

While it is true that predicting every ball of a cricket match could also be used to increase betting, it is important to note that illegal betting is already a major problem in the sport globally[9]. By being one step ahead of the bettors, it would make operating efficiently difficult for illegal bettors. It is also important to note that predicting every ball of a cricket match is not an exact science. There are multifarious factors that can influence the outcome of a ball, which we highlight in the sections below, and even the most complex models, such as deep neural networks, cannot predict every ball with 100% accuracy, given either the highly random nature of Twenty20 cricket or insufficient data for each match. This means that even if bettors know the predicted outcome of a ball, they cannot be certain of the actual outcome. In conclusion, the benefits of predicting every ball of a cricket match outweigh the risks. Machine learning classification methods can help to create a more transparent and fair betting environment in the sport of cricket ball-by-ball.

[7] NewIndianXpress. (2023, April 16). *TN Man Ends Life After Losing Money in IPL betting, online rummy*. The New Indian Express. https://www.newindianexpress.com/states/tamil-nadu/2023/apr/16/tn-man-ends-life-after-losing-money-in-ipl-bettingonline-rummy-2566320.html

[8] Shank, C. A. (2022, September 15). *Information asymmetry in the NFL gambling market: Inside information versus informed bettors*. Journal of Behavioral and Experimental Finance. https://www.sciencedirect.com/science/article/abs/pii/S2214635022000806

[9] TNN / Updated: Feb 5, 2023. (n.d.). *Ahmedabad Crime Branch Bust Cricket Betting Racket Worth Rs 1,414 crore: Ahmedabad News - Times of India*. The Times of India. https://timesofindia.indiatimes.com/city/ahmedabad/ahmedabad-police-crime-branch-bust-cricket-betting-racket-worth-rs-1414-crore/articleshow/97614070.cms?from=mdr

1.2 Project Scope

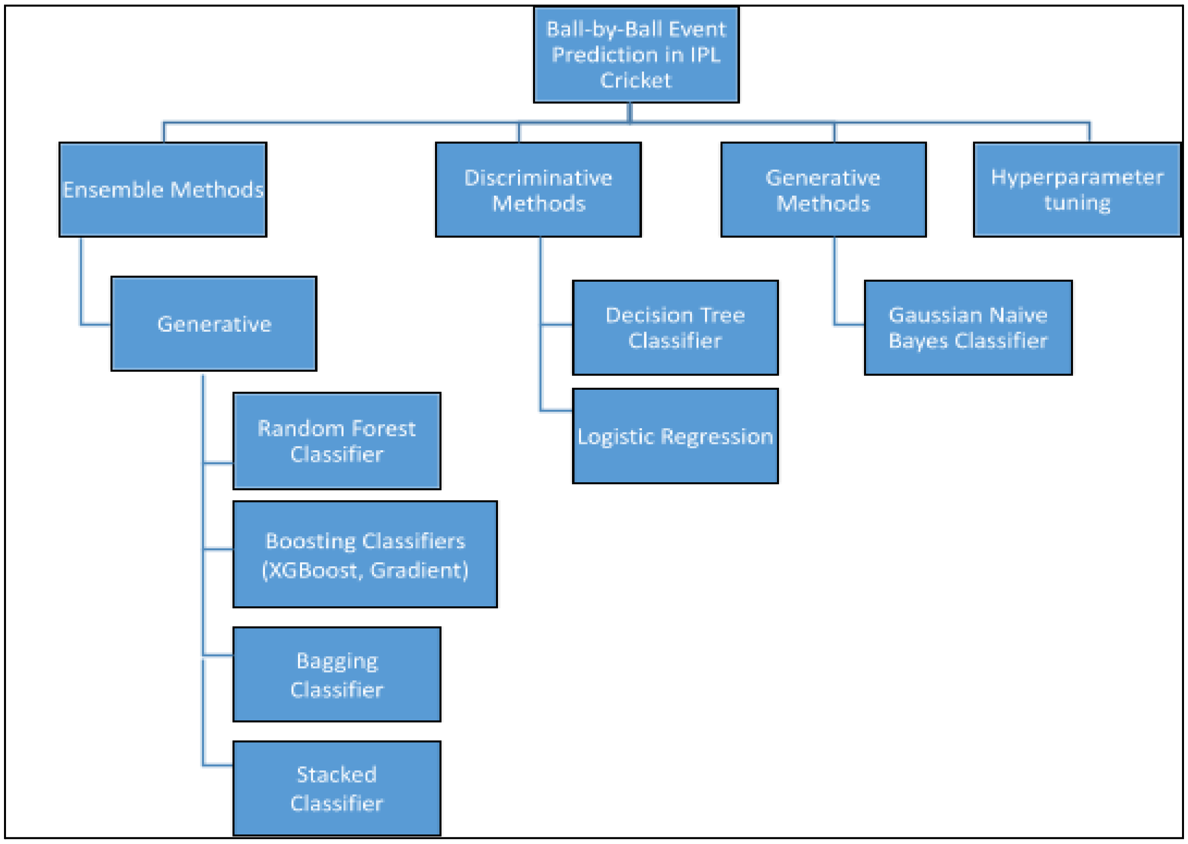


Figure 1 Project Stage Map

1. Generative Methods Based Analysis of the dataset
    a. Traditional exploration and analysis to predict the dependent variable, Ball Outcome (aliased `Event`).
        i. Preliminary visual exploration and analysis of the data set (exploratory data analysis).
        ii. Proposing a possible outcome from the preliminary exploration of the dataset. We will analyze which features are highly correlated with each other
    b. Use boosting classifiers coupled with cross-validation to evaluate the dataset. Perform hyperparameter-tuning for the boosting procedure (for `n_estimators` use [10, 25, 50, 75, 100, 125, 150]; for `learning_rate` use [0.1, …, 1.0]; for `booster` use [‘gbtree’, ‘gblinear’, ‘dart’]).
        i. Compare the results of the `n_estimators`, `learning_rate`, and `booster` over the IPL dataset and pick the best parameter set, evaluated over accuracy and weighted f1-score. Dump the weights of the best model as a pickle file for deployment.

   c. Use bagging, with and without bootstrap replacement , to evaluate the dataset. Alter the parameters for the boosting procedure (for `n_estimators` use [10, 25, 50, 75, 100, 125, 150]; for `learning_rate` use [0.1, …, 1.0]; for `max_samples` use [5%, 10%, 20% of the train data rows]; for `max_features` use [20%, 40%, 60%, 80%, 100% of the train data features]; for `bootstrap` use [True, False]; for `bootstrap_features` use [True, False]).

2. Repeat 1.1-1.3 for the Discriminative methods, Gaussian Naive Bayes (generative, non-ensemble method), and Random Forest.
3. Hyperparameter tuning and analysis will be performed based on the results of 1 and 2

Deliverables

1. General Deliverables
   a. Common approaches, tools, and sharable implementations in support of ML/AI-based solutions for the project, including:

   Data definition and normalization practices: The data was collected from a Kaggle user, who already scraped ball-by-ball IPL data from the ESPNCricinfo website. The data comprises seasons of the IPL from its inception in 2008 till the 2019 season, leading to 132,000 observations. In order to understand the dataset, we must also understand the general rules of the game. The variables employed in model training are as follows:

- ***`match_type`***: Match ID serves as a unique identifier for each game in the dataset. It has been categorized into different types of matches, including round-robin games and various playoff stages such as semi-final, eliminator, quarter-final, and final. This categorization allows for a detailed analysis of different match types and their outcomes in the IPL.
- ***`venue`***: Venue represents the location where the match is played. The dataset comprises matches held in 17 different venues, with the initial matches of the 2014 IPL played in UAE and the entire 2009 season played in South Africa being filtered out. This ensures consistency in the training data, focusing on Indian venues for analysis.
- ***`inning`***: Inning denotes which half of the match it is. In the IPL, there are two innings per match. Instances where matches were reduced to only one inning due to rain or unexpected circumstances have been filtered out for data accuracy.
- ***`batting_team`***: Batting team indicates the team currently at bat, attempting to score runs.
- ***`bowling_team`***: Bowling team signifies the team currently fielding and trying to dismiss the batsmen and restrict the batting team's score.

- ***`bowler_type`***: In cricket, bowlers are of two types: spinners or pacers. This categorization has been performed manually using a dictionary of spinners, providing insights into the performance of different types of bowlers in the IPL matches.
- ***`batsman`***: Similar to ***`bowler_type`***, we also engineered a feature for the position a batsman generally bats in. In cricket, the batting order is the sequence in which batters play through their team's innings, there always being two batters taking part at any one time. All eleven players in a team are required to bat if the innings is completed (i.e., if the innings does not close early due to a declaration or other factor). Therefore, we categorized all players into the following: top order, middle order, or tail enders.
- ***`non_striker`***: Because at any given time two batsmen, the batter on strike/facing the ball and the partner not on strike, from a team need to be playing on the field, the same method employed to engineer **`batsman`** was employed for **`non_striker`.**
- ***`ball`***: Ball corresponds to an over, with each over consisting of 6 legal deliveries. The format is represented as X.Y, where X is the over number and Y is the ball number within that over.
- ***`wickets`***: Wickets represent the number of batsmen dismissed by the bowling team. Each inning has a total of 10 wickets, and if all wickets fall, the inning concludes.
- ***`ball_length`***: Ball length describes the length of the delivery bowled. This information has been obtained from commentary data and categorized as short, full, yorker, etc., providing valuable insights into the bowling strategies employed during matches.
- ***`cumulative_runs`***: The raw dataset has a variable **`total_runs`** scored per ball. Using this feature, we can calculate the total runs scored cumulatively across every inning. This feature provides the user to specify the number of runs.
- `***runs_left_to_score***`: By reverse engineering ***`cumulative_runs`*** for only the balls bowled in the second innings, we could create a column that tracks how many runs are left to win for the batting team from the specified ball until the end of the 20 overs. Given a match scenario for example, if a lot of runs are required to win at any point in the game, the batsman will take added risks to maximize runs for the team, which would increase the likelihood of a wicket falling.
- ***`event`***: Event is the feature-engineered response variable derived from original columns in the dataset. It encompasses various aspects of the game, including runs scored by the batsmen (`batsman_runs`), extra runs (`wide_runs`, `bye_runs`, `legbye_runs`, `noball_runs`, `penalty_runs`), `total_runs` of the inning, and dismissal types (`dismissal_kind`). This comprehensive variable captures key events during each delivery, aiding in detailed analysis and prediction of match outcomes.

The main drawback of the data gathered is that it lacks any information about positions of fielders per ball. This is a very crucial component of the game that affects how a batsman thinks or anticipates a bowler’s next delivery to prepare for a shot. A batsman would also, generally, avoid hitting the ball to the better fielders in the bowling team, possibly leading to more predictability for our model.

In terms of normalization, we only utilize one-hot-encoding on the feature categorical variables.

Processing pipelines and optimization methods: We will utilize Docker and Flask for the deployment of the project. We will also begin prototyping on an .ipynb file and then refactor the code into .py files for our pipeline.

2. *Analysis of the dataset and Trained Model*

2.1 Exploratory Data Analysis

Since we are using a secondary dataset, we have decided to perform exploratory analysis before moving forward. The following graphs show each IPL team's general batting performance, in terms of total runs scored, and bowling performance, in terms of total wickets taken over the whole dataset. The dataset does not have any missing values.

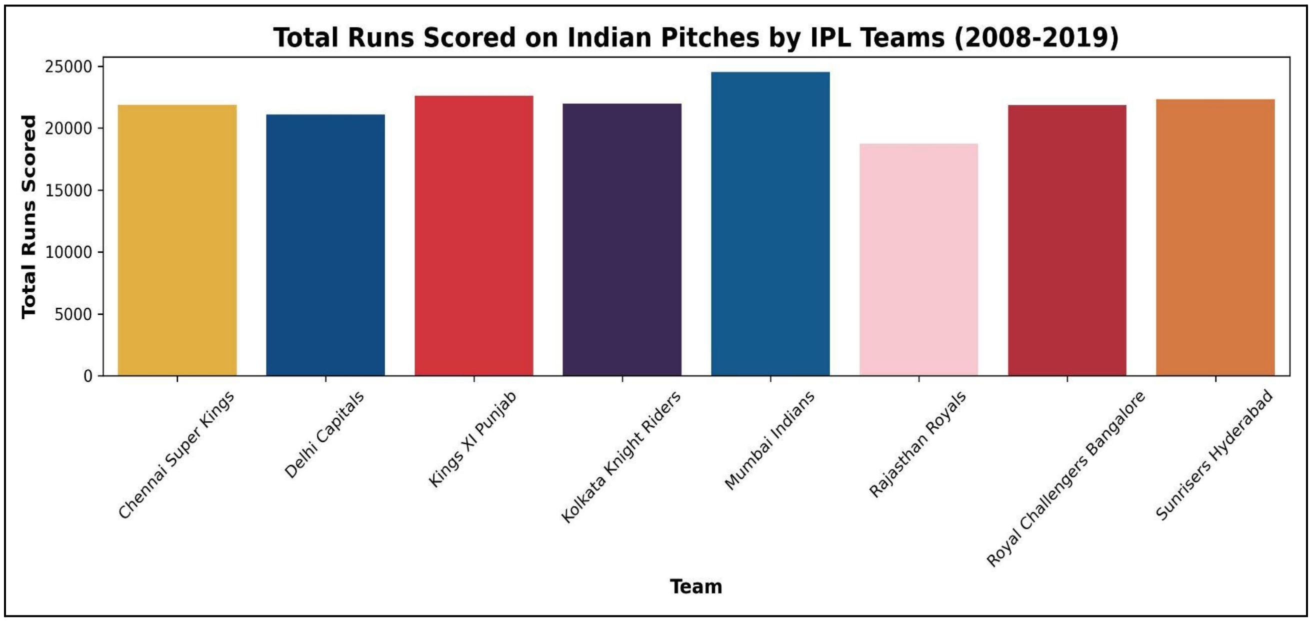


Figure 1. Total Runs Scored on Indian Venues by Team

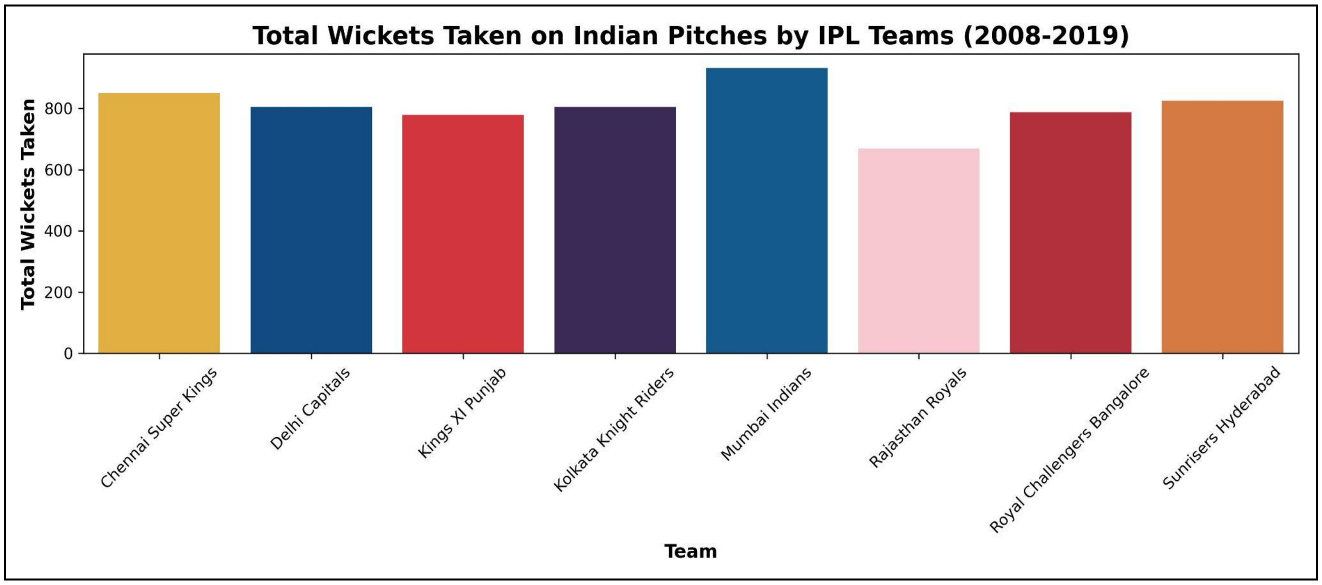


Figure 2. Total Wickets Taken on Indian Venues by Team

Both figures above (Figure 1 and Figure 2) provide a general overview of team performance across IPL matches held in India. The dataset does not include the entirety of IPL 2009, which was held in South Africa, and some matches of IPL 2014 that were held in Dubai, UAE. The dataset also does not account for the games played by teams that were dissolved at any given time, including Kochi Tuskers Kerala, Rising Pune Supergiants, and Gujarat Lions. Teams have also changed names over the course of the IPL, including Deccan Chargers (Sunrisers Hyderabad) and Delhi Daredevils (to Delhi Capitals), for whom data was present individually but combined to maintain current team names. Lastly, Chennai Super Kings and Rajasthan Royals, although never dissolved, missed out on two IPL seasons due to a betting scandal in 2013.

Given the above information, the teams that have performed best with bat and ball are Mumbai Indians, Chennai Super Kings, and Sunrisers Hyderabad. Therefore, it is also not surprising that Mumbai Indians and Chennai Super Kings have won multiple seasons of the IPL, more than all other teams combined.

3. *Model Selection and Evaluation*

Obtaining a baseline Multinomial Logistic Regression model provided us necessary insight for model selection. Following the preliminary regression analysis, we have decided to evaluate Gaussian Naive Bayes, Decision Tree Classifier, Random Forest Classifier, Bagging

Classifier, Adaptive Boosting Classifier, Gradient Boosting Classifier, XGB Classifier, and CatBoost Classifier as candidate models.

Figure 3 and Figure 4 below show the model accuracy and weighted f1-score results respectively. We can see that all boosting models, Gradient Boosting, Extreme Gradient Boosting, and Adaptive Boosting perform the best (have relatively highest mean accuracy and weighted f1-score). However, accuracy, while an essential metric, might not be the sole measure of a model's effectiveness, especially in this imbalanced dataset, where two classes, Extras and Runs, significantly outnumber the others, Dots and Wickets. As can be seen from the confusion matrices in Figure 5 below, majority classes dominate the accuracy metric, leading to a misrepresentative view of holistic model performance. In cricket, especially in the scenario of betting, correctly predicting the minority class, Wickets, is of paramount importance.

Precision and recall are crucial metrics in such cases. Precision measures the accuracy of positive predictions, while recall gauges the model's ability to find all the relevant cases in a dataset. From the confusion matrices, we observed that Naive Bayes and Decision Tree models exhibited better precision and recall for minority classes. This means that when these models predict a wicket or a boundary ball, they are more likely to be correct. In a business or real-world setting, this could translate to better decision-making and assist organizations that are trying to track illegal betting activity.

Machine learning models often involve trade-offs. Boosting algorithms, while achieving high overall accuracy and weighted F1 scores, might sometimes sacrifice precision and recall for minority classes in favor of generalizing patterns across the majority class. On the other hand, Naive Bayes and Decision Tree models, being less complex, can capture intricate details in minority classes but might not generalize as well on the majority class.

Therefore, choosing the right model depends on the specific objectives of our prediction task and, eventually, the user who will be utilizing our deployed application. If the focus is on correctly identifying wickets and boundary balls, we might prioritize models like Naive Bayes or Decision Trees. However, if a more balanced performance across all classes is desired, as showcased by Weighted F1-Score, the boosting algorithms could be a better choice.

Because of this, we will select them as the candidate models for level 0 in the stacking regression. We will use Light Gradient Boosting Regression as the level 1 combiner or metamodel to aggregate the results of the level 0 models.

In addition to the selected models, we have also opted for cross-validation of the data set. For the current iteration of the model, we will use five-fold validation. Figures 6 and 7 show the performance of the standalone and stacked models. The performance of the stacked model seems to be worse than the candidate model. Stacked models performance can be improved by tuning the hyperparameters.

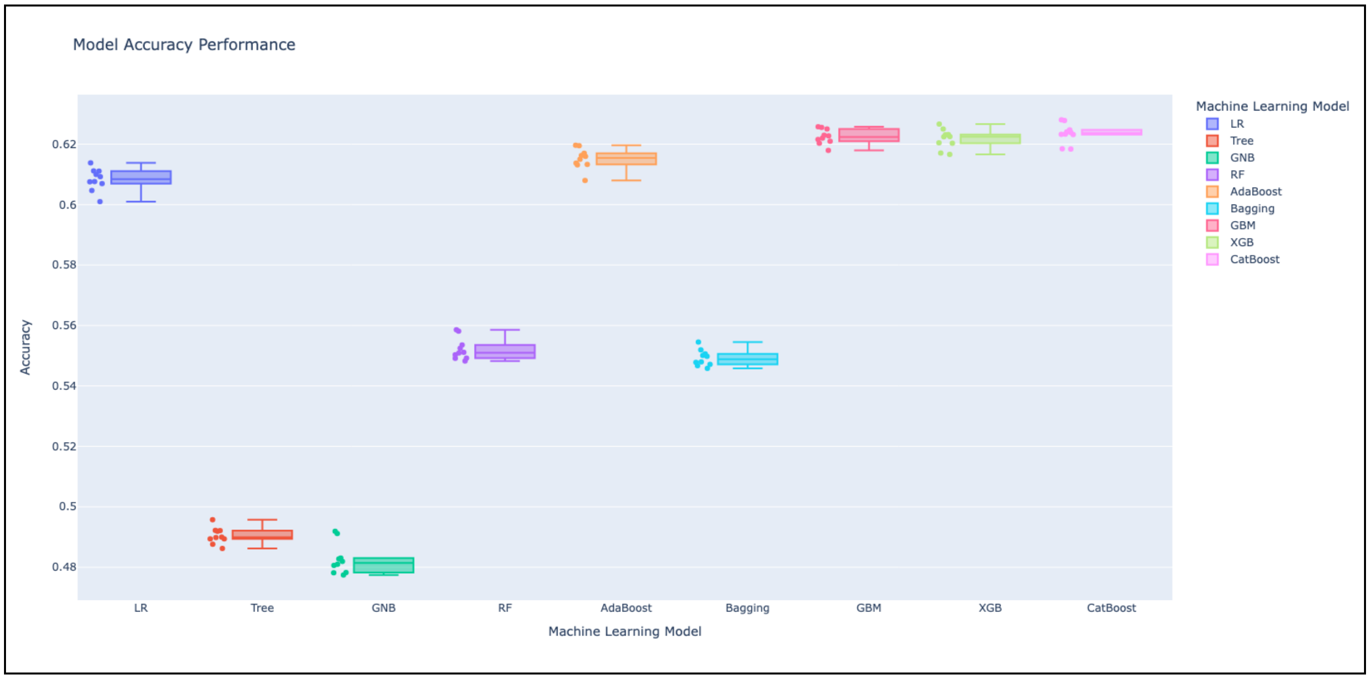


Figure 3. Model Accuracy Performance on Training Set

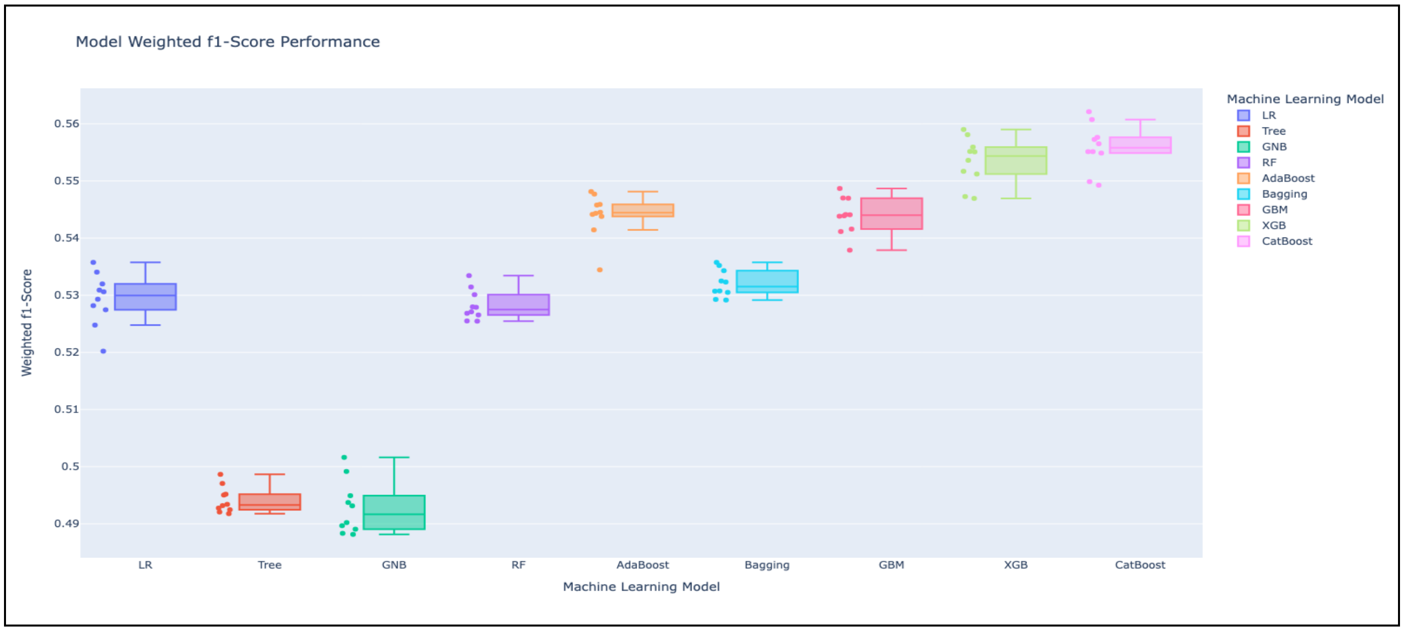


Figure 4. Model Weighted F-1 Score Performance on Training Set

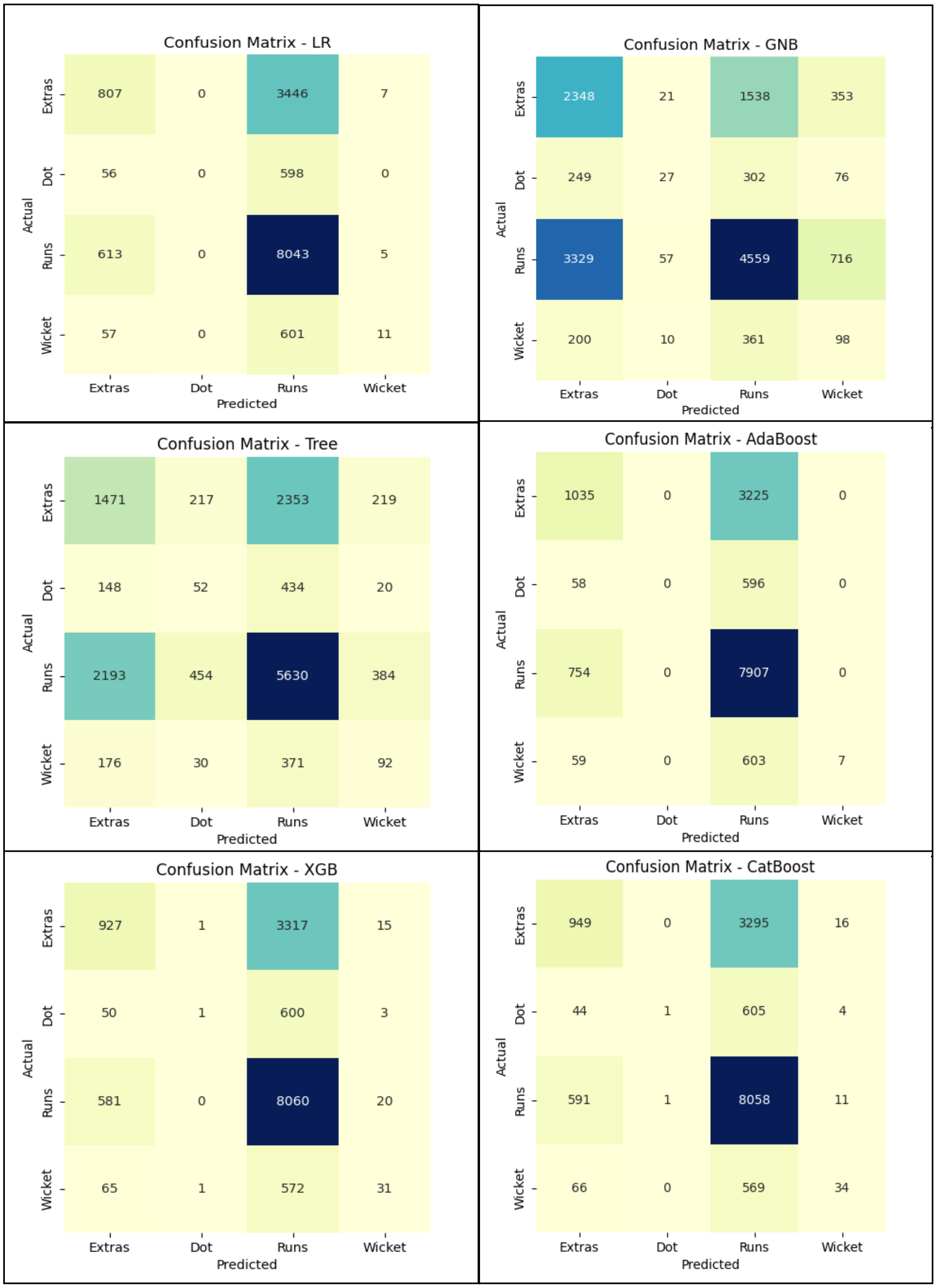


Figure 5. Model Confusion Matrices on Test Set

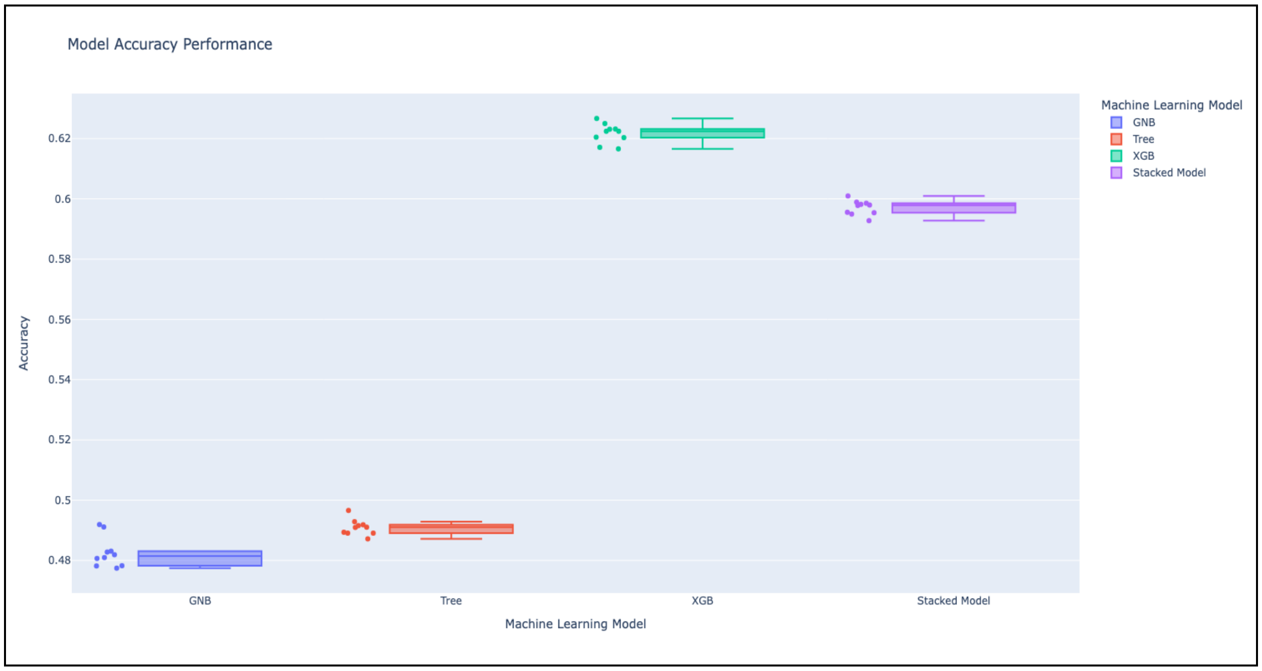


Figure 6. Stacked Model Accuracy Performance on Training Set

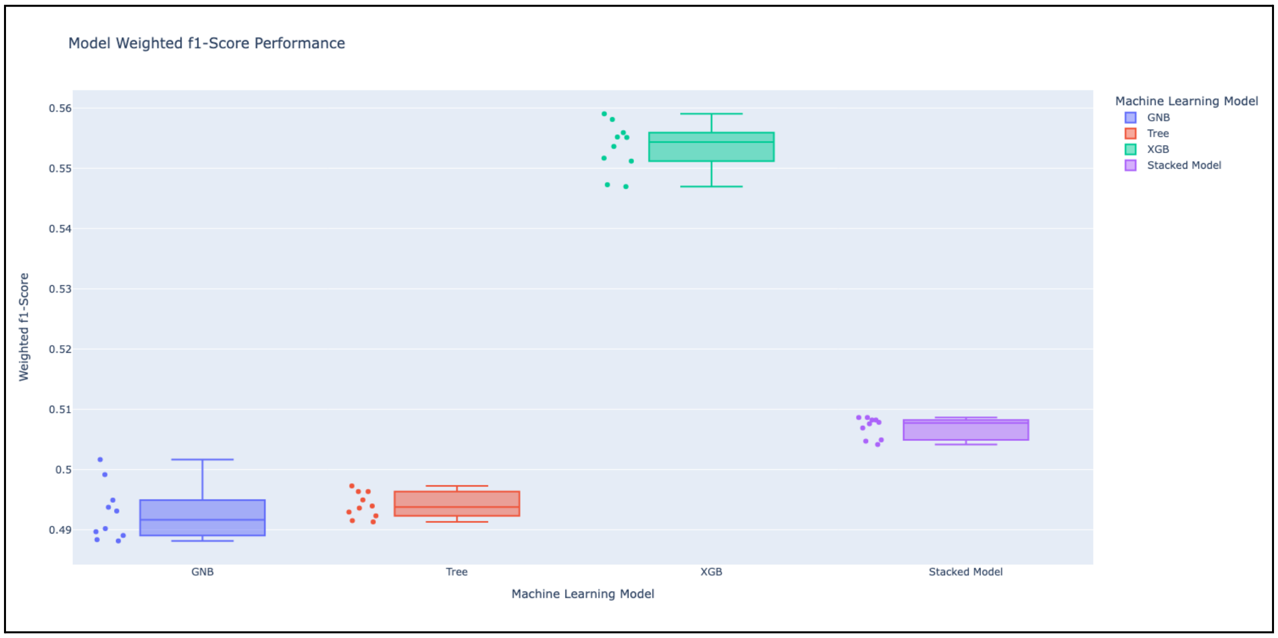


Figure 7. Stacked Model Weighted F-1 Score Performance on Training Set

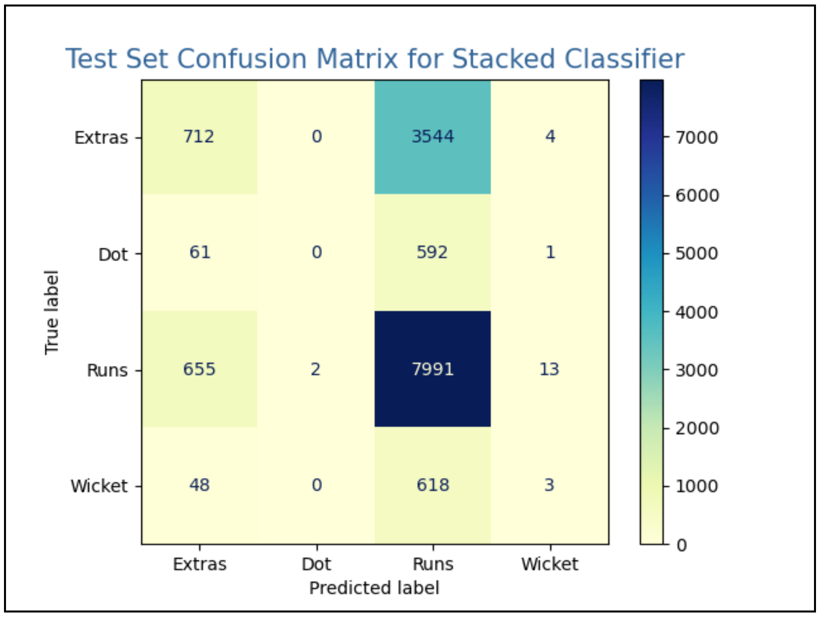


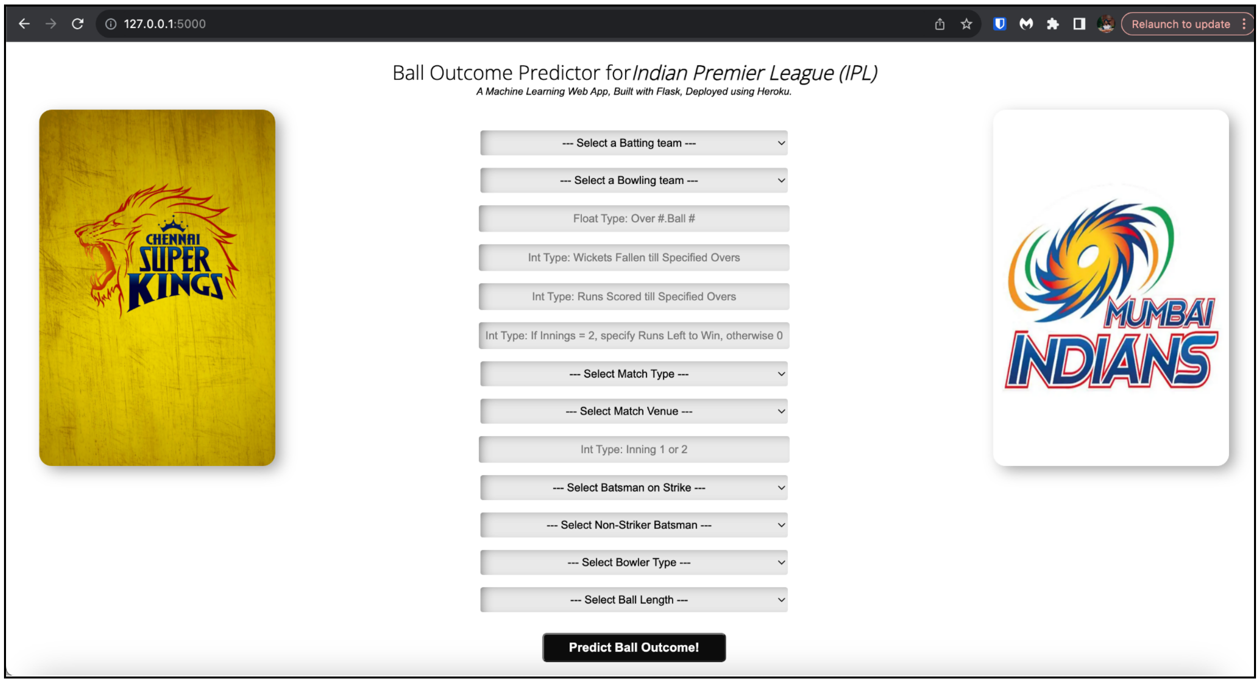
Figure 8. Stacked Model Confusion Matrix on Test Set

4. Initial Deployment

We have selected Heroku as the final deployment site with Flask as the python package to render the HTML pages and to build the interactive application for the user. The site will display two HTML pages, the landing page for the user to fill in the features and the page that displays the prediction from the model for the features inputted by the user.

4.1 Screen 1

Figure 8 below shows the application for our `XGBClassifier()` model deployed locally. There are 13 unique features, from high level descriptors like the Venue of the match to the length of the ball bowled, that encompass a scenario of a ball bowled in a match. The user has to fill in all provided features, given either the dropdown menu or an appropriate keyboard input, in order to get a prediction. This input is processed and then used to predict one of four outcomes, including Runs from the bat, WICKET!, Extras, and Dot Ball, for the given feature vector.


Ball Outcome Predictor for *Indian Premier League (IPL)*

A Machine Learning Web App, Built with Flask, Deployed using Heroku.

--- Select a Batting team ---

--- Select a Bowling team ---

Float Type: Over #.Ball #

Int Type: Wickets Fallen till Specified Overs

Int Type: Runs Scored till Specified Overs

Int Type: If Innings = 2, specify Runs Left to Win, otherwise 0

--- Select Match Type ---

--- Select Match Venue ---

Int Type: Inning 1 or 2

--- Select Batsman on Strike ---

--- Select Non-Striker Batsman ---

--- Select Bowler Type ---

--- Select Ball Length ---

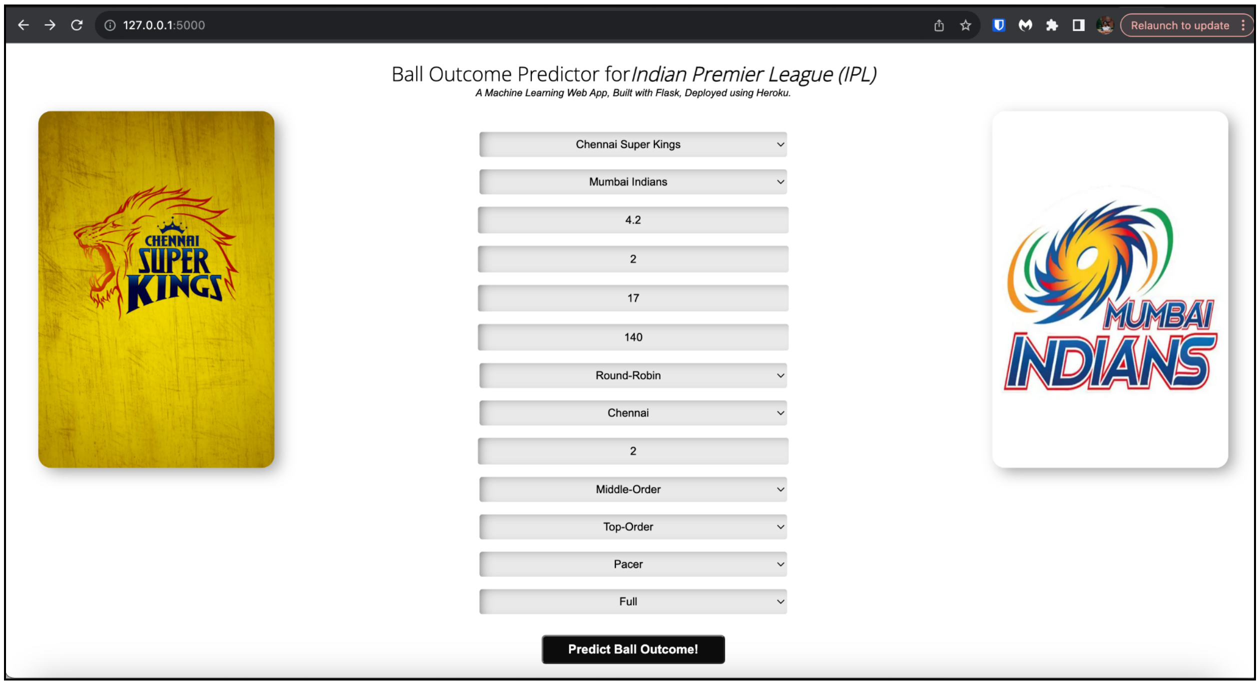
Predict Ball Outcome!

Figure 8. Screenshot of Landing Page for the Deployed Application

4.2 Screen 2

Figure 9 below shows an example of the values of the features a user could input to get a fair prediction for the ball bowled. The application will not accept keyboard inputs from the user that are not within the range of the specified feature. For example, there are 20 overs and 6 legal balls bowled per over in a match. Therefore, in the over number feature, the value must be a float type, ranging from 0.0 to 20.0, indicating number of overs, with increments of 0.1 in each over up till a maximum of 0.6 for each over, indicating the number of balls. Therefore, the input 12.8 or 20.1 would be deemed invalid and the user would have to conform to the feature's range of values set by us.

Moreover, the user's due diligence must also come into play when interacting with the application to gain the best prediction from the model. Even if the user inputs valid keyboard inputs for the respective features, choosing the correct values from the dropdown would determine the model's accuracy. For example, if the user inputs 8 wickets fallen but does not select the Tail-Ender value for either the Batsman on Strike or Nonstriker Batsman, then that would not reflect the realistic


127.0.0.1:5000

Ball Outcome Predictor for *Indian Premier League (IPL)*

A Machine Learning Web App, Built with Flask, Deployed using Heroku.

Chennai Super Kings

Mumbai Indians

4.2

2

17

140

Round-Robin

Chennai

2

Middle-Order

Top-Order

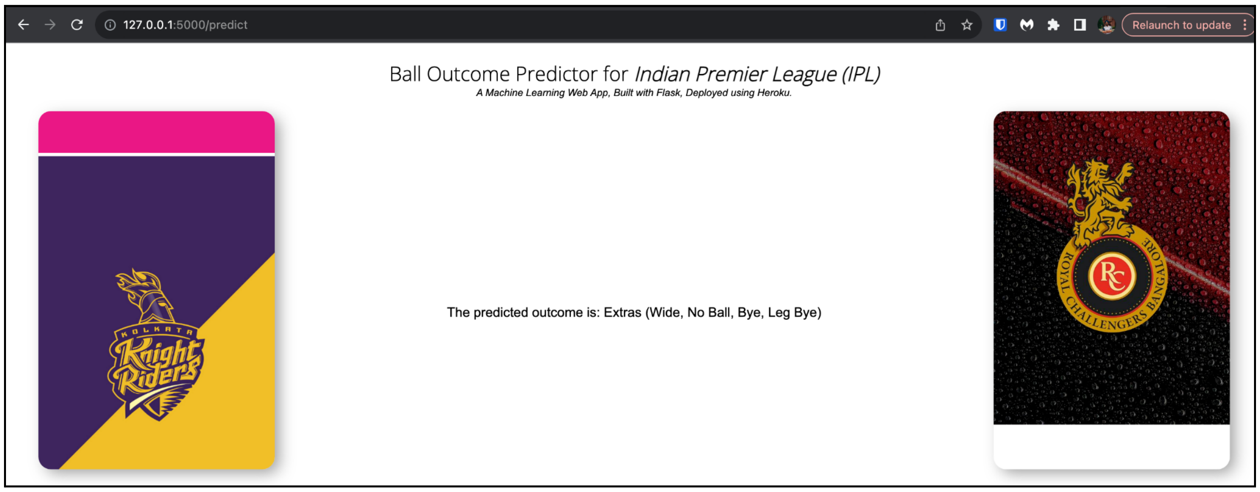
Pacer

Full

Figure 9. Screenshot of Valid Feature Inputs for the Deployed Application

4.3 Screen 3

Figure 10 below displays the prediction from the model (saved as a pickle file) for the feature values shown in Figure 9. To provide a better prediction to the user, we can make use of the `predict_proba()` function instead of the currently used `predict()` function to present the user with the probabilities of each of the four outcomes occurring for the given feature vector. Accordingly, the user can then gauge the confidence level of the model's predictions and assess the reliability of the suggested prediction.


127.0.0.1:5000/predict

Ball Outcome Predictor for *Indian Premier League (IPL)*

A Machine Learning Web App, Built with Flask, Deployed using Heroku.

The predicted outcome is: Extras (Wide, No Ball, Bye, Leg Bye)

Figure 10. Screenshot of Prediction page for the Deployed Application

5. Conclusion

In the dynamic and ever-evolving world of modern cricket, where every ball bowled presents a unique challenge to all players on the field and the ball's outcome is never certain, our end-to-end project sought to harness the power of data science and to predict the outcome of each delivery in the Indian Premier League (IPL). We set out to answer two research questions:

1. *Can machine learning classification techniques, specifically ensemble methods, accurately predict the outcome of every ball bowled in an IPL match?*
2. *Which ensemble methods perform best in both generalizing to unseen data and catering to the user's request, such as highest probability of predicting a ball leading to a wicket, specified in the deployed application?*

Given the data at hand, the answer to the first question would be yes, to some extent. We saw that the ensembles generally overfit to the majority response labels "Extras" and "Runs", yet performed better on accuracy and weighted f-1 score metrics compared to other classification algorithms, such as Logistic regression, Gaussian Naive Bayes, and Decision Trees. The non-ensemble methods, although exhibited moderate accuracy and weighted f-1 scores, did not overfit as much as the un-tuned ensemble models. These models were, therefore, better suited for predicting the minority labels "Wicket" and "Dot Ball", which would be of more interest to bettors and regulatory bodies alike. In an attempt to achieve higher accuracy and weighted f-1 score for predicting ball outcome, we also trained a regularized artificial neural network with 162,584 parameters; however, its performance was very similar to that of the untuned XGBClassifier(), indicating that ensembles not only are at par with state-of-the-art machine learning algorithms in terms of robustness, but are explainable, a key distinguishing feature from black box neural networks.

The second question builds upon the first question and narrows down the performance of the selected models to the user's objective for maximizing the model's prediction probability for a particular class of the response variable, `Event`. Although `XGBClassifier` showcased the best performance on the chosen metrics, `BaggingClassifier()` is the ensemble method that generalized better to unseen data, as seen in the confusion matrices in Figure 5. In fact, upon closer inspection of the model performances, we noticed that the boosting ensembles faced challenges in accurately predicting the minority class "Dot Ball". This observation was consistent across all boosting ensemble models, as demonstrated by the absence of any predictions for the "Dot Ball" class in the respective confusion matrices presented in Figure 5. Despite their overall robustness and competitive performance, the boosting ensembles struggled to generalize effectively to this specific class, even after hyperparameter tuning. Therefore, for the app to satisfy a user's aim of maximizing the probability of predicting a wicket, alternative models, including Decision Trees and Gaussian Naive Bayes could be utilized.

However, the performance of all these models that were analyzed can mainly be attributed to the data, which has multifarious limitations, fed into them. A crucial aspect of any ball bowled in cricket is the placement of fielders (9 outfielders excluding bowler and wicket-keeper), which our data lacked as either one or many features. We could have devised a way to engineer one or more columns for gathering field placement data like we did when creating the `ball_length` feature by matching regex patterns on the `commentary` feature. In fact, the paper “Assessing the Impact of Fielding in Twenty20 Cricket” from Simon Fraser University does just that[10]. They collected commentary data from both internal and domestic Twenty20 matches and filtered each individual commentary for a ball using a set of contextual words and phrases. Then, they carried out a textual analysis using random forest methodology to quantify the impact of fielding. Similarly, we could have also created a dictionary of contextual words for gathering data on the line of the ball, i.e where the ball was bowled on the pitch in relation to the wickets.

Another key aspect of these tournaments is the yearly auction that results in either players leaving the tournament (by their own discretion or going unsold in the auction), new, emerging players joining the tournament (the IPL has been a platform for young, talented cricketers to showcase their skills and go on to represent the country), or players switching teams due to the auction. Given this occasional disruption between each IPL season, we were still able to tackle this challenge effectively. By incorporating a player profiling system that identifies players as either pacers or spinners, and predicting their most likely batting order, we enhanced the user experience of our application. This player categorization allowed us to better understand the team compositions and strategize model predictions based on the unique strengths and weaknesses of each player type amidst the ever-changing player dynamics in the league. However, instead of grouping players, the stark opposite of building a classification model for a specific player, for example the batsman Virat Kohli, could also be suitable to predict only his performance in future matches.

Although these limitations are present in the data gathered, we were able to build a deployed model that can act as a starting point for a model to deploy for the regulatory Indian bodies. Access to fielding data from sports analytics organizations coupled with the addition of betting odds per ball, we can augment the current model with a more comprehensive and accurate prediction application. Continued refinement and expansion of the dataset will further bolster the model's effectiveness, paving the way for more sophisticated approaches to sports analytics and betting regulation.

[10] Perera, H., Davis, J., & Swartz, T. B. (n.d.). Assessing the impact of fielding in twenty20 cricket. https://www.sfu.ca/~tswartz/papers/fielding.pdf