Prediction of IPL matches using Machine Learning while tackling ambiguity in results

## Abstract

**Background/Objectives**: The IPL (Indian Premier League) is one of the most viewed cricket matches in the world. With a perpetual increase in the popularity and advertising associated with it, forecasting the IPL matches is becoming a need for the advertisers and the sponsors. This paper is centered on the implementation of machine learning to foretell the winner of an IPL match.

# Methods/Statistical analysis: The cricket in the T-20 format is highly unpredictable - many features contribute to the result of a cricket match, and each attribute feature has a weighted impact on the outcome of a game. In this paper, first, we define a meaningful dataset through data mining; next, derive essential features using various methods like feature engineering and Analytic Hierarchy Process. Alongside, we adopted different machine learning classification algorithms like Naïve Bayes, Random Forest, Logistic Regression, Extratreesclassifier, XGBoost to train the models for predicting the winner.

# Findings: As per our investigation, tree-based classifiers provided better results with our derived model. We observed the highest accuracy of 60.043% with Random Forest with a standard deviation of 6.3%.

# Novelty/Applications: This model can be leveraged by brands, sponsors and advertisers to keep up their marketing strategies.

**Keywords:** The Indian Premier League, Machine Learning, Analytic Hierarchy Process, Winner Prediction, IPL

## Introduction

The IPL (Indian Premier League) is a 20-20 cricket league in India where eight teams (representing eight cities in India) play against each other. This game is India's biggest cricket festival - the most celebrated and the most viewed, where the action is just not limited to the cricket field. The clatters, promotional events, cheerleaders, advertisements, fan clubs, interactions, and betting are celebrated along with the players and the matches.

The entire revenue cycle of the IPL revolves around advertising. IPL also utilizes television timeouts, and there are other humongous opportunities associated with advertising. Apart from national and global broadcasts, the matches are transmitted to regional channels in eight different languages. The brand value of the IPL was ₹475 billion (US$6.7 billion) in 2019[1]. The IPL cricket league has proved to be a 'game-changer' for both cricket and the entire Indian advertising industry[2].

“Due to the saturated market, it is especially important for sport organizations to function with maximum efficiency and to make smart business decisions”[3]. One of the most common area where Sports organizations use analytic is assessing the value of an athlete to their brand and strategizing their marketing activities.

In this paper, we explore and develop models using machine learning to predict IPL matches' outcome. Figure 1. illustrates the entire process followed while conducting the research.

During the research, a multi-step approach was taken to gather and pre-process the historical data. Feature engineering[4,5] techniques were applied to derive more insights about the current dataset. Further, analysis of the essential features was done using selection techniques, and simultaneously best players were marked based on their performances. Optimized features from players' performance were then added to the team data. The models were trained using multiple machine learning classification algorithms to develop a predictive model. The highest accuracy was observed with Random Forest, i.e., 60.043%, with a standard deviation of 6.3%.

## Related Works

Many researchers have contributed towards predicting the results of cricket matches. Authors[6] proposed a paper on predicting the outcome of an IPL match, where they acquired the dataset available for all the 11 seasons from the archives of the IPL website[7] and applied the concept of Multivariate Linear Regression to calculate the strengths of a team using the data from the Player Points section of the official IPL website. Later for prediction, the scholars utilized various classifiers, namely Naive Bayes, Extreme Gradient Boosting, Support Vector Machine, Logistic Regression, Random Forests, and Multilayer perceptron. In another study, authors[8] adopted the Team Composition method to predict the outcome of an ODI match. They utilized the players' data career statistics (both recent and overall performance) to calculate the player's strength and aggregate to finalize the team strength. They also included other features like venue and toss. The model derived from their research gives the best result with KNN algorithm.

Although not related to cricket match prediction, a study was conducted by the authors[9], for predicting the performance of bowlers. They used Multilayer Perceptron and created a new feature using the data called CBR (Combined Bowling Rate) and calculating the harmonic mean of the Bowling Average, Bowling Economy, and Bowling Strike Rate. Authors[10] used the pressure Index of the team batting in the second innings to predict the match at different points of the chase; they devised a formula to calculate the pressure index at each point and used various methods to calculate the probability of a win based on the pressure index.

## Materials and Methods

### 1. Dataset

#### 1.1. Dataset Gathering

The historical dataset was obtained from various sources – Kaggle[11], ESPN Cricinfo[12] and iplt20[7]. The performance data of individual players was scraped from the ESPN Cricinfo website, by using Python Library Beautiful Soup[13], to calculate each player's strength and the team. This scrapped data demonstrates 26 features - including batting and bowling performances of the players. Additionally, match results data was obtained from Kaggle. This data displayed 18 features. The IPL winning point table yearly data was accumulated from the IPL website. This data demonstrated the point feature.

#### 1.2. Pre-processing of Data

##### 1.2.1 Conversion of Data Format (Label Encoding)

Most of the Machine Learning algorithms work better with numerical values than the string values. Hence, we converted all the string formats in the dataset to the numerical formats utilizing the Label Encoding. The features that were converted are: Team Names, Venue Names, Winning Team Name, Toss Winner Team Name.

##### 1.2.2 Data Cleaning and dealing with null values

To produce accurate results, we eliminated all the unnecessary features from the dataset, for example – Umpire Name, Stadium Name, Date, Dl applied, Player of the match. We also excluded the features that could result in data leakage, such as Win by Runs and Win by Wickets. Further, we eradicated all the match rows that were dismissed, drawn, or null.

#### 1.3. Class Imbalance

Class Imbalance is a problem in machine learning where the class distribution is highly imbalanced[14,15]. We noticed that predicting the results using the team's name is not feasible as it can cause a massive Class Imbalance between the groups. For example - MI winning more than 100 matches whereas KTK wining less than 10 matches is a Class Imbalance problem. Refer to Figure 2.

To rule out the class imbalance, we decided to design our model to predict the winner based on the essential features instead of the Team names, declaring either Team 1 or Team 2 as a winner. Moreover, we also discovered that the number of times Team 1 won is more than Team 2. To resolve this issue and balance the Team 1 winning and Team 2 winning in the label column, we interchanged a few values of column Team 1 with column Team 2.

#### 1.4. Assumptions

We followed a few assumptions to make our model accurate and robust. The names of few teams were changed by the owners due to legal actions or due to the change in owner, however, the players and team dynamics didn’t change. The name Delhi Capitals was changed to Delhi Daredevils, Deccan Chargers to Sunrisers Hyderabad, and Pune Warriors to Rising Pune Supergiant. In these cases, we considered them as the same team irrespective of the change in the name. Moreover, we considered the data of only 11 players for a team based on the highest number of matches they have played during the IPL.

### 2. Feature Engineering

#### 2.1. Base Features

##### 2.1.1. Features from the processed data

From the gathered and processed data, we extracted the following three meaningful features.

1. ***City 2. Toss Winner 3. Toss Decision***

##### 2.1.2. Dream11 Strength Calculation

In the first approach, we referred to the Dream11 points table to derive our formula. Refer to Appendix A (a, b, c, d) for the definitions of point system and their notations.

###### Batting Score of a player:

***Input:*** *Players p ∈ P, Career Statistics of player p: φ(p)*

***Output:*** *Batsmen Score of all the players: φBatsman Score*

*1: for all players p ∈ P do*

*2: φ ← φ(p)*

*3: u ← (1\* ΦRuns\_Scored) +(1\*Φnum\_4s) + (2\*Φnum\_6s) + (8\*Φfifties) + (16\*Φhundreds) - (2\*Φfduck)*

*4: if Φbat\_strike\_rate < =50:*

*v ← -6*

*else if Φbat\_strike\_rate > 50 and Φbat\_strike\_rate < =60:*

*v ← -4*

*else if Φbat\_strike\_rate > 60 and Φbat\_strike\_rate < = 70:*

*v ← -2*

*endif*

*5: w ← v \* Φbat\_strike\_rate\* Φbat\_innings*

*6: y ← u + w*

*7: φBatsman Score ← y*

*8: end for*

###### Bowling Score of a player:

***Input:*** *Players p ∈ P, Career Statistics of player p: φ(p)*

***Output:*** *Bowling Score of all the players: φBowling\_Score*

*1: for all players p ∈ P do*

*2: φ ← φ(p)*

*3: u ← (25\*Φwickets) + ( 8\*Φctchs) + (12\*Φstmp) + (8\*Φ4\_wicket\_haul ) + (16\*Φ5\_wicket\_haul) + (8\*Φmaidens)*

*4: if Φbowl\_economy < = 6 and Φbowl\_economy > 5:*

*v ← 2*

*else if Φbowl\_economy > 4 and Φbowl\_economy < = 5:*

*v ← 4*

*else if Φbowl\_economy < = 4:*

*v ← 6*

*else if Φbowl\_economy > = 9 and Φbowl\_economy < 10:*

*v ← -2*

*else if Φbowl\_economy >= 10 and Φbowl\_economy < 11 :*

*v ← -4*

*else if Φbowl\_economy >= 11:*

*v ← -6*

*endif*

*5: w ← v \* Φbowl\_economy\* Φbowl\_innings*

*6: y ← u + w*

*7: φBowling\_Score ← y*

*8: end for*

###### Total Score of a player:

***Input:*** *Players p ∈ P, φBowling\_Score, φBatsman\_Score*

***Output:*** *Total Strength← φ Total \_Strength, φBatsman\_Strength, φBowling\_Strength*

*1: for all players p ∈ P, φBowling\_Score, φBatting\_Score do*

*2: φ Total\_Strength ← (φBowling\_Score + φBatting\_Score) / φtot\_matches*

*3: φBatsman\_Strength ← φBatting\_Score / φBat\_innings*

*4: φBowling\_Strength ←φBowling\_Score/ φBowl\_innings*

*5: endfor*

###### Team Strength

***Input:*** *Top 11 Players p ∈ P, φ Total \_Strength, φBatsman\_Strength, φBowling\_Strength*

***Output:*** *Team Strength: φTeam \_Strength*

*1: for all players p ∈ P , φTotal\_Strength do*

*2: φTeam \_Strength ← ( / φmax\_matches)*

*3: φTeam\_Batting\_Strength ← ( / φmax\_matches)*

*4: φTeam \_Bowling\_Strength ← ( / φmax\_matches)*

*5: endfor*

###### Cumulative Team Strength

For a particular year, Team Strength represents the previous year's performance, whereas the Cumulative Team Strength signifies the mean of the Team Strength of all the last years. For example – for the Mumbai Indians in 2016, the Strength will be the 2015 strength, and Cumulative Strength will be the mean of the Strength from 2008 to 2015. From this section, we collected the eight significant features mentioned below:

|  |  |
| --- | --- |
| 1. ***Team\_1\_BattingStrength*** | 1. ***Team\_2\_BattingStrength*** |
| 1. ***Team\_1\_BowlingStrength*** | 1. ***Team\_2\_BowlingStrength*** |
| 1. ***Team\_1\_Strength*** | 1. ***Team\_2\_Strength*** |
| 1. ***Team\_1\_CumulativeStrength*** | 1. ***Team\_2\_CumulativeStrength*** |

##### 2.1.3. Analytic Hierarchy Process for Strength Calculation

Different measures highlight different aspects of a player's ability, which makes some features essential compared to others. For example, the strike rate is a necessary feature for a game - especially T20. In T20, the number of overs is less, which makes this feature more crucial as it adds to the team's ability to score maximum runs. We weighted the features according to their relative importance over other measures (features) in our research. We adopted the Analytic Hierarchy Process (AHP) to determine these weights for each player to calculate their bowling and batting features. Besides, we calculated the weights for each team based on their past performance.

The Analytic Hierarchy Process is a method for decision-making in complex conditions in which many variables or criteria are considered in prioritizing and selecting options[16]. AHP generates a weight for each evaluation criterion. The higher the weight for a corresponding criterion, the more important is the corresponding criterion (Refer to Appendix B). Finally, the AHP combines the criteria weights and the options amounts, thus determining a global score for each option and a consequent ranking. The global score for a given option is a weighted sum of the scores it obtained with respect to all the criteria[17].

###### Batting AHP

Priority Order: We arranged the attributes in their decreasing order of importance based on our knowledge and experience of the T20 cricket matches as below:

***Batting Average > Innings > Strike Rate > 50’s > 100’s > 0’s***

Subsequently, we create a matrix to compare the importance of each attribute. Refer to Table 1.

Finally, we got weights for each attributes: Batting Average: 0. 3887, Innings: 0. 2601, Strike Rate: 0. 1754, Fifties: 0. 0834, Centuries: 0. 0550, Zeros: 0. 0373. Using these values we calculated the Batting strength through AHP.

*AHP bat = 0.3887 \* Average + 0.2600 \* Innings + 0.1754 \* Strike Rate + 0.0834 \* 50’s + 0.0550 \* 100’s - 0.0373 \* 0’s*

###### Bowling AHP

Priority Order: We arranged the attributes in their decreasing order of importance based on our knowledge and experience of the T20 cricket matches as below:

***Overs > Economy > Wickets > Bowling Average > Bowling Strike Rate > 4W Haul***

Subsequently, we create a matrix to compare the importance of each attribute. Refer to Table 2.

Finally, we got weights for each attributes: Overs: 0.4174, Economy: 0.2634, Wickets: 0.1602, Bowling Average: 0.0975, Bowling Strike Rate: 0.067862, 4-Wickets Haul: 0.0615. Using these values we calculated the Bowling strength through AHP.

*AHP bowl = 0.387509 \* Overs + 0.281308 \* Economy + 0.158765 \* Wickets + 0.073609 \* Bowling Average + 0.067862 \* Bowling Strike Rate + 0.030947 \* 4W Haul*

From this section, we formed the four essential features mentioned below:

|  |  |
| --- | --- |
| 1. ***Team\_1\_AHP\_BAT*** | 1. ***Team\_1\_AHP\_Ball*** |
| 1. ***Team\_2\_AHP\_BAT*** | 1. ***Team\_2\_AHP\_Ball*** |

##### 2.1.3. Rank Calculation using AHP

Using the AHP, we derived the coefficient for the win rate of each team against the other. Assumption: We dropped the KTK and GL Teams while calculating the weights as they never played against each other.

Priority Order: We calculated the priority order through AHP with the dataset of the matches played for win/loss for each team against each team. For example, the Team CSK (Chennai Super Kings) and MI (Mumbai Indians) played 27 matches against each other, and according to the dataset, MI won 16, and CSK won the rest 11 games. In this instance, in the MI row, the input will be 16/11 = 1.454545, and in the CSK row, it will be reciprocal, which is 11/16 or 1/1.4545454 = 0.6875. Refer to Table 3.

We found the yearly ranks of each team based on the win ratios. The ranks were derived using AHP. Refer to Table 4.

For the KTK and GL, we took the mean value which is 1 as the coefficients. We formed two features from this section as below:

1. ***Team\_1\_Rank 2. Team\_2\_Rank***

##### 2.1.4. Win Rate

For a cricket match, the win rate almost determines the overall performance of a team. If a team is continuously winning the matches against other teams, it is a sign that the team's form is good, and the probability of the team winning the subsequent matches is higher. On the other hand, a losing team reflects that it is not in good form and may even lose games further.

As our next steps, we crawled the entire IPL match list played every year by each team from 2008 to 2019. If the two teams played against each other for the first time, we reset the win rate to 0 for both the teams. Subsequently, we checked for all the played matches and noted the winners for those occurrences. This helped in defining a ratio for each team. For a match, we considered the past win rate ratio of the team as below:

*Φwin\_rate(Match R) = Total Number of wins till match R-1/ Total Number of matches played till R-1*

We derived two important features from this section as below:

1. ***Team\_1\_Win\_Rate 2. Team\_2\_Win\_Rate***

##### 2.1.5. Team Points

  The IPL is a league tournament based on point system. Every year, two teams play against each other twice before they enter the semi-final match, if not eliminated. The point table comprises teams, match won/lost/tied, and net run rate. The ranking of the teams was done according to the points. We fed this past performance features of the teams to our model to predict the results. We formed four significant features from this section as below:

|  |  |
| --- | --- |
| 1. ***Team\_1\_Point*** | 1. ***Team\_1\_Cumulative Point*** |
| 1. ***Team\_2\_Point*** | 1. ***Team\_2\_Cumulative Point*** |

For a particular year, Team Point represents the previous year’s performance, whereas the Cumulative Team Point represents the mean of the strengths of all the previous years.

#### 2.2. Intersection Features

###### Consistency

The consistency of a team adds more weightage to its current performance than the overall performance. Therefore, we allotted 80 percent weightage to the current performance of a team and 20 percent weightage to their overall performance.

*Team 1 Consistency = (Team 1 Strength \* 0.8 + Team 1 Cumulative Team Strength\*0.2)/2*

*Team 2 Consistency = (Team 2 Strength \* 0.8 + Team 2 Cumulative Team Strength \*0.2)/2*

We formed two features from this section mentioned below:

1. ***Team\_1\_Consistency 2. Team\_2\_Consistency***

###### Win Rate Strength

The individual strength of a team represents how strong a team is by considering the stats. However, there are various other factors that impact the winning of a team - for example - playing sequence of a team, performance as a team, sentiments of the audience. We captured this information by multiplying the strength with the previous win rate of the team.

*Team 1 Win Strength =Team 1 Win Rate \* Team 1 Strength*

*Team 2 Win Strength =Team 2 Win Rate \* Team 2 Strength*

*Team 1 Win Cumulative Strength =Team 1 Win Rate \* Team 1 Cumulative Strength*

*Team 2 Win Cumulative Strength =Team 2 Win Rate \* Team 2 Cumulative Strength*

We derived four features from this section mentioned below:

|  |  |
| --- | --- |
| 1. ***Team\_1\_WinStrength*** | 1. ***Team\_1\_WinStrength*** |
| 1. ***Team\_2\_Win\_Cumulative Strength*** | 1. ***Team\_2\_Win\_Cumulative Strength*** |

#### 2. 3. Transformation Features

With all the formulated Base and Intersection features we developed Transformed features. These features are created by subtracting two base features or intersection features from the same category. For example: Team1\_Team\_Strength is subtracted from the Team2\_Team\_Strength to create a new feature.

 Since we created a lot of new features based on base and intersection features for our model, multicollinearity[18] could occur. Multicollinearity occurs when multiple features in a model are highly linearly related, which means one variable can be predicted quite accurately using the other variable. The problem with multicollinearity is that it causes the model to overfit. To deal with multicollinearity in our model we dropped all the base and intersection features which were used to create the new features.

#### 2.4. Addressing the Symmetry in Data

As per our primary assumption, every team's performance is independent of the opposition team, toss decision, home-field advantage, and progress into the series. This allowed us to make independent team features that will be present in both TEAM1 and TEAM2. The features generated can be broadly bucketed into Match and Team Features. Now, we observed that there are similar features for both TEAM1 and TEAM2, which leads to symmetry in the dataset(Refer to Table 5).

It is apparent to a human that while switching TEAM1 with TEAM2, the results will be the same. However, a machine learning model is asymmetric in nature and is neither capable of identifying the symmetry of features nor has a way to input the information about the symmetry of features. Hence, we entered this information to the model by generating a symmetric duplicate for every row in the training set (Refer to Table 6).

The below steps were taken to the train and test sets:

1. The original dataset is split using *sklearn train\_test\_split* into training and test sets
2. The training set is then mirrored as shown above and append to the original training set which increases in training set size
3. The test set is also mirrored but we don't append them and create two test sets

#### 2.5. Model Ambiguity

The mirroring of the rows only tells the model about the existence of a symmetric scenario, but the model will still interpret the mirrored rows as new training set rows completely unrelated to the original rows. This asymmetric nature of the model leads to ambiguity in the results in certain rows ( Refer to Table 7).

We tested the model for a given match in two configurations. The model interprets both the cases as two different test cases. As a result, sometimes, the model returns different predictions for the same case. Such an occurrence is called Model Ambiguity. Note: This occurrence is not an incorrect prediction, as the prediction will be counted correct in either test set 1 accuracy or test set 2 accuracy.

To tackle this phenomenon of Model Ambiguity, we evaluated the model using five parameters apart from just training and test accuracy:

* Training Accuracy: % of correct predictions in mirrored and merged train set
* Test 1 Accuracy: % of correct prediction in the original test set
* Test 2 Accuracy: % of correct prediction in the mirrored test set
* Real Test Accuracy: % of correct prediction after discrediting the scores for ambiguous rows
* Ambiguity: % of rows in which ambiguity is observed

The objective of hyperparameter tuning was to maximize real test accuracy by driving down the ambiguity while evaluating the overfitting of the model using training accuracy and test 1 & 2 accuracies.

### 3. Data Set Split:

We noticed that by changing the random state in our dataset the accuracy differs a lot. This change is because the training and testing dataset is randomly split based on the state we put. To prevent such a scenario and to make our model robust we used RepeatedStratifiedKFold**.** We selected 10 folds[19] and 2 iterations to give a total of 20 folds. We prefered *RepeatedStratifiedKFold* over *StratifiedKFold* as our dataset is small, and *RepeatedStratifiedKFold* gives more fold with larger validation set.

**Constant:** We took Random State = 827 throughout the project

We evaluated our model using accuracy and Standard Deviation, Cohen Kappa[20], Skewness[21, 22] and Kurtosis[21, 22]. To check or visualize the performance of the multi - class classification problem, we plotted the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. These curves are one of the most important evaluation metrics for checking any classification model’s performance[23].

### 4. Results and Discussions:

#### Model Implementation using Naïve Bayes

We got the Real test accuracy of 58.233 % with a standard deviation of 5.5 % ( Refer to Table 8).

The Area under the Curve is 0.63. The ROC curve was plotted with the best result that we got using Naïve Bayes. The distribution of Real Test Accuracy was done to derive Skewness and Kurtosis of the Real Test Accuracy (Refer to Figure 3).

* *Kurtosis of the Real Test Accuracy is -0.7954.*
* *Skewness of the Real Test Accuracy: -0.2357*

#### Model Implementation using Logistic Regression

We tuned our model over 1232 combinations. Refer to Appendix C (a). The best results derived : Real Test Accuracy of 57.78% with a standard deviation of 5.8% ( Refer Table 9).

Further, the ROC curve with the best result was made and we got the AUC value of 0.57. The Real test accuracy distribution was plotted for deriving the Kurtosis and Skewness. Refer Figure 4.

* *Kurtosis of the Real Test Accuracy: 0.5892*
* *Skewness of the Real Test Accuracy: 1.4699*

#### Model Implementation using ADABOOST

We have tuned our model over 56 combinations. Refer to Appendix C (b). The best result with the corresponding hyper-parameters were derived - Real test accuracy is 60.035% with a standard deviation of 6.2% (Refer to Table 10).

* *Kurtosis of the Real Test Accuracy: -0.6021*
* *Skewness of the Real Test Accuracy: -0.4677*

#### Model Implementation using XGBOOST

We have tuned our model over 3600 combinations. Refer to Appendix C (c). The best result with the corresponding hyper-parameters were derived - Real test accuracy is 55.42 % with a standard deviation of 5.9% (Refer to Table 11).

Further the ROC curve with the best result was made and we got the AUC value of 0.62. The Real test accuracy distribution with XGBOOST was plotted for deriving the Kurtosis and Skewness (Refer Figure 6).

* *Kurtosis of the Real Test Accuracy is -0.8633*
* *Skewness of the Real Test Accuracies: 0.0456*

#### Model Implementation using ExtraTreesClassifiers

We tuned our model over 320 combinations. Refer to Appendix C (d). The best results derived : Real Test Accuracy of 59.506 % with a standard deviation of 5.9% (Refer to Table 12).

Further the ROC curve with the best result was made and we got the AUC value of 0.64. The Real test accuracy distribution with ExtraTreesClassifiers was plotted for deriving the Kurtosis and Skewness (Refer to Figure 7).

* *Kurtosis of the Real Test Accuracy is -0.2121*
* *Skewness of the Real Test Accuracies: 0.5902*

#### Model Implementation using Random Forest Classifier

We have tuned our model over 1200 combinations[24]. Refer to Appendix C (e). The best result with the corresponding hyper-parameters were derived - Real test accuracy is 60.043 % with a standard deviation of 6.3% (Refer to Table 13).

Further, the ROC curve with the best result was made and we got the AUC value of 0.62. The Real test accuracy distribution with Random Forest Classifier was plotted for deriving the Kurtosis and Skewness (Refer to Figure 8).

* *Kurtosis of the Real Test Accuracy is -0.8606*
* *Skewness of the Real Test Accuracies: -0.2491*

# Conclusion and Future Works

The research's focus was to predict the winner for an IPL match using machine learning and utilizing the available historical data of IPL from season 2008-2019. In the process, various Data Science methods were adopted to conduct the study, including data mining, visualization, preparation of database, feature engineering, applying the Analytic hierarchical process, creating prediction models, and training classification techniques.

The IPL dataset was gathered and pre-processed. The missing values were removed, and variables were encoded into the numerical format to make the dataset uniform. The essential features were then derived from data using the domain knowledge to extract raw data features via data mining techniques and the final results were derived from our model. Several machine learning models were applied to the selected features to predict the IPL match results. The best results were concluded using the tree-based classifiers. We observed the highest accuracy of 60.043% with Random Forest with a standard deviation of 6.3% (Refer to Table 14).

For this research, we took the series wise performance of the player rather than their match wise performance while calculating the player’s strength. For a more exhaustive approach to further develop this research, match wise data can be considered. Also, the research cab be further enhanced by adding other factors like comparing players’ performances at a particular stadium.

# References

1. Varun Gupta and Santosh N. Duff & Phelps Launches IPL Brand Valuation Report 2019. Duff & Phelps. 2019, September 20. <https://www.duffandphelps.com/about-us/news/ipl-brand-valuation-report-2019>
2. Akshay Badwe. IPL Advertising: All You Need to Know About The Game Of Revenue. Kreedon. 2019, April 17. <https://www.kreedon.com/ipl-advertising-all-you-need-to-know/>
3. G. Fried and C. Mumcu. Sport analytics: A data-driven approach to sport business and management. Taylor & Francis. 2016, November 10. <https://doi.org/10.4324/9781315619088>
4. J. Heaton. An empirical analysis of feature engineering for predictive modeling. SoutheastCon 2016, Norfolk, VA, 2016, pp. 1-6. [10.1109/SECON.2016.7506650](file:///C:\Users\Amit\Downloads\10.1109\SECON.2016.7506650).
5. Zheng, Alice and Amanda Casari. (2018). Feature engineering for machine learning: principles and techniques for data scientists. " O'Reilly Media, Inc.".
6. Lamsal, R. and Choudhary, A. Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning. 2018.  <https://arxiv.org/abs/1809.09813>.
7. IPL website: <https://www.iplt20.com>
8. Jhanwar, M.G. and Pudi, V. Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. 2016, September. <https://api.semanticscholar.org/CorpusID:35555185>
9. Hemanta Saikia, Dibyojyoti Bhattacharjee and H. Hermanus Lemmer.  Predicting the Performance of Bowlers in IPL: An Application of Artificial Neural Network, International Journal of Performance Analysis in Sport, 12:1. 75-89. 2012. DOI: [10.1080/24748668.2012.11868584](https://doi.org/10.1080/24748668.2012.11868584)
10. Dibyojyoti Bhattacharjee and Priyanka Talukdar. Predicting outcome of matches using pressure index: evidence from Twenty20 cricket, Communications in Statistics - Simulation and Computation. 2019. DOI: [10.1080/03610918.2018.1532003](file:///C:\Users\Amit\Downloads\10.1080\03610918.2018.1532003)
11. Kaggle: https://www.kaggle.com/manasgarg/ipl
12. ESPNCricInfo: <https://stats.espncricinfo.com/>
13. Beautiful Soup Python Library: <https://pypi.org/project/beautifulsoup4/>
14. Jason Brownlee. A Gentle Introduction to Imbalanced Classification. Machine Learning Mastery. 2019, December 29. <https://machinelearningmastery.com/what-is-imbalanced-classification/>
15. Johnson, J.M. and Khoshgoftaar. T.M. Survey on deep learning with class imbalance. *J Big Data* 6,27. 2019. <https://doi.org/10.1186/s40537-019-0192-5>
16. Emrouznejad, A. and Ho, W. Fuzzy Analytic Hierarchy Process. New York: Chapman and Hall/CRC. 2018. <https://doi.org/10.1201/9781315369884>
17. Kalpdrum Passi and Niravkumar Pandey. Increased Prediction Accuracy In The Game of Cricket Using Machine Learning. International Journal of Data Mining & Knowledge Management Process(IJDKP) Vol.8,No.2. 2018, April 9. <https://arxiv.org/abs/1804.04226>
18. Sahika Gokmen, Rukiye Dagalp and Serdar Kilickaplan. Multicollinearity in measurement error models, Communications in Statistics - Theory and Methods. Communications in Statistics - Theory and Methods. 2020, April 14. <https://doi.org/10.1080/03610926.2020.1750654>
19. R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. International Joint Conference on Artificial Intelligence, vol. 14, pp. 1137– 1145, Montreal, Canada. 1995. <http://ai.stanford.edu/~ronnyk/accEst.pdf>
20. Susana M. Vieira, Uzay Kaymak and João M. C. Sousa. Cohen’s kappa coefficient as a performance measure for feature selection. International Conference on Fuzzy Systems*.* 2010, September 2010. [10.1109/fuzzy.2010.5584447](file:///C:\Users\Amit\Downloads\10.1109\fuzzy.2010.5584447)
21. Blanca, M. J., Arnau, J., López-Montiel, D., Bono, R. and Bendayan, R. Skewness and kurtosis in real data samples. Methodology, 9, 78-84. 2013. DOI: [10.1027/1614-2241/a000057](https://ericssonmediasolutions-my.sharepoint.com/personal/nidhi_tripathi_mediakind_com/Documents/Desktop/DD-TOP/Ayush-Research/10.1027/1614-2241/a000057)
22. Cain, M.K., Zhang, Z. and Yuan, K. Univariate and multivariate skewness and kurtosis for measuring nonnormality: Prevalence, influence and estimation. Behav Res 49, 1716–1735. 2017, October 26. <https://doi.org/10.3758/s13428-016-0814-1>
23. Sarang Narkhede. Understanding AUC - ROC Curve. Towards Data Science. 2018, June 26. <https://www.48hours.ai/files/AUC.pdf>
24. Probst, Philipp, Marvin N. Wright and Anne‐Laure Boulesteix. Hyperparameters and Tuning Strategies for Random Forest.Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9.3. 2019. <https://arxiv.org/abs/1804.03515v2>.

­­