National College of Ireland

A comparative study of cricket par score using Machine Learning and DL method

MSc Research Project
Data Analytics

Omkar Tawade
Student ID: 19232136

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Omkar Tawade
Student ID:	19232136
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	23/09/2021
Project Title:	A comparative study of cricket par score using Machine Learning and DL method
Word Count:	6741
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Omkar Tawade
Date:	23rd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A comparative study of cricket par score using Machine Learning and DL method

Omkar Tawade
19232136

Abstract

Cricket has become popular with the advent of the T20 format, the shortest format of the game. Like other games, the outcome of a cricket match is also essential, but sometimes rain plays a spoilsport in this game. The current rain rule method, the Duckworth Lewis method, is used to calculate the revised score. Duckworth Lewis method invented for one-day international cricket is now used in T20 cricket by just scaling down the resource table could be contentious. This research aims to develop models based on a machine-learning algorithm to predict the score of rain-interrupted matches. The Indian premier league (2008-2020) dataset from Kaggle is used for the purpose of the study. This research introduced novel features, including runs, wickets, dot balls, and the number of boundaries scored in the last five overs. Extreme gradient boosting (XGBoost), Adaptive Boosting (AdaBoost) and Random forest algorithms are used in the research to produce models. Results showed that the proposed method successfully predicted scores with less margin of error when compared to the Duckworth Lewis method. Experiments showed that models based on second innings data provided better results in terms of RMSE and MAE.

1 Introduction

Cricket is a game played between two teams in three formats (Test Cricket, One Day Cricket and T20 Cricket). In T20 cricket, both teams play an innings of 20 overs and have 10 wickets in hand. The team batting first (Team 1) tries to maximise its score using over and wickets. The team bowling first (Team 2) tries to restrict this score because it will be its target. After the first innings, the bowling team (Team 2) gets to bat and tries to reach the target within 20 overs. It is the shortest version in cricket with a win/loss outcome. Bad weather often leads to quite exciting twists and turns in test cricket which is the most extended format in cricket but is not tolerated in the result oriented T20 games.

Cricket has long searched for a fair way to calculate the winning side when inclement weather affects the limited over game. As a result of an infamous situation from the 1992 world cup semi-final, two English statisticians Duckworth and Lewis (1998), introduced the Duckworth Lewis Method. International Cricket Council formally adopted it in 1999, and it is evolved now to become known as the Duckworth Lewis Stern method introduced by Steven (2016). Duckworth Lewis recognises that each team has two resources available with which to score runs, batters who are yet to be dismissed and the number of overs remaining. According to this, a team has 100 per cent resources (50 overs and ten wickets)

at the beginning of an inning, so whenever a match is interrupted, then Duckworth Lewis method takes the number of wickets in hand and number of overs that are lost into an account and calculates the percentages of resource remaining. Duckworth Lewis table has resource percentage remaining table 1 obtained by considering all available data of historic matches. The Duckworth Lewis takes care of all these things and eliminates possible unfair advantages.

Several matches have been impacted due to the use of the Duckworth Lewis method because of its robust nature. An incident happened in a T20 match between England vs West Indies, where West Indies was chasing a total, and they scored 30 runs off the first two overs, which suggested that they were 11 runs ahead of the par score at this moment. Rain interrupted the match, and the match was reduced to 22 balls only. It was challenging and unfair for England to claw back those 11 runs in just 22 balls.

The Duckworth-Lewis Resource table was initially intended for one-day cricket matches, but it has now been implemented to Twenty20 matches as well, by just scaling down the table. The reduction of overs from 50 to 20 in Twenty20 matches is the most significant variation, and this suggests that scoring patterns in Twenty20 matches may differ from those in One-Day contests. T20 cricket is also limited over format like One-Day cricket, but there are few changes in the rules like power-play restriction, bowling combination, etc. Twenty20 is a more animated version of the game, emphasising scoring fours and sixes. The Duckworth-Lewis technique and its associated resource table are based on one-day cricket score trends; hence their application in Twenty20 may be questioned.

1233 number of T20 matches were played in 2019, suggesting that around 49,320 over by over data is generated in 2019. This significant amount of the data generated every year can be used to improve the existing Duckworth Lewis method. Duckworth Lewis tables is a static method in which score is derived from a table, but in the proposed technique machine learning method will be able to analyse the trend of scoring pattern based on runs scored and the rate at which wickets are lost with respect to power-play restrictions.

A considerable amount of research has been done to predict the match's score and outcome using machine learning techniques. However, little research has been carried out to improve the Duckworth Lewis method. The rationale behind this research is to explore possible solutions for improving the existing Duckworth Lewis method by feature engineering and comparing the results with other states of the art methods.

The research aims at investigating the extent to which machine learning can help to predict logical scores, unlike the traditional Duckworth Lewis method. To address this research question following objectives have been derived.

- To study literature review and identify existing methods that can be implemented and used as criteria for the proposed research and evaluation metrics.
- To design an alternative of the Duckworth Lewis system using the Indian Premier League Dataset.
- Evaluate the results of the literature review and the proposed method and examine if the proposed method provides better results than the previous work.

Literature review is described in the Section 2 is based on other researchers work on the proposed approach. A comprehensive review of methods used earlier for improving the Duckworth Lewis method and discussed the advantage and shortcoming of the methods used in the past. Data prepossessing and how the data was obtained is explained

Table 1: Duckworth Lewis Resource Table (T20 Cricket Edition)

	Wickets Lost									
Overs Available	0	1	2	3	4	5	6	7	8	9
20	100	96.8	92.6	86.7	78.8	68.2	54.4	37.5	21.3	8.3
19	96.1	93.3	89.2	83.9	76.7	66.6	53.5	37.3	21	8.3
18	92.2	89.6	85.9	81.1	74.2	65	52.7	36.9	21	8.3
17	88.2	85.7	82.5	77.9	71.7	63.3	51.6	36.6	21	8.3
16	84.1	81.8	79	74.7	69.1	61.3	50.4	36.2	20.8	8.3
15	79.9	77.9	75.3	71.6	66.4	59.2	49.1	35.7	20.8	8.3
14	75.4	73.7	71.4	68	63.4	56.9	47.7	35.2	20.8	8.3
13	71	69.4	67.3	64.5	60.4	54.4	46.1	34.5	20.7	8.3
12	66.4	65	63.3	60.6	57.1	51.9	44.3	33.6	20.5	8.3
11	61.7	60.4	59	56.7	53.7	49.1	42.4	32.7	20.3	8.3
10	56.7	55.8	54.4	52.7	50	46.1	40.3	31.6	20.1	8.3
9	51.8	51.1	49.8	48.4	46.1	42.8	37.8	30.2	19.8	8.3
8	46.6	45.9	45.1	43.8	42	39.4	35.2	28.6	19.3	8.3
7	41.3	40.8	40.1	39.2	37.8	35.5	32.2	26.9	18.6	8.3
6	35.9	35.5	35	34.3	33.2	31.4	29	24.6	17.8	8.1
5	30.4	30	29.7	29.2	28.4	27.2	25.3	22.1	16.6	8.1
4	24.6	24.4	24.2	23.9	23.3	22.4	21.2	18.9	14.8	8
3	18.7	18.6	18.4	18.2	18	17.5	16.8	15.4	12.7	7.4
2	12.7	12.5	12.5	12.4	12.4	12	11.7	11	9.7	6.5
1	6.4	6.4	6.4	6.4	6.4	6.2	6.2	6	5.7	4.4

in the Section 3. Section 4 outlines the design specification and architecture to be implemented. Section 5 covers the details of implementation and proposed architecture. Section 6 provides a through description of experiments for evaluating the performance of the model.

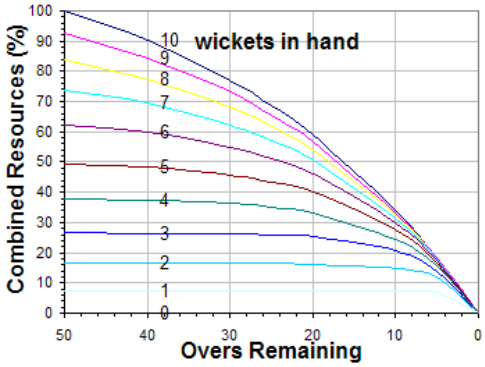


Figure 1: Duckworth Lewis Resource Graph, from Wikipedia, the free encyclopedia (2007)

2 Related Work

2.1 Literature review of Duckworth Lewis Method

Duckworth and Lewis (1998) proposed a unique and fair method of calculating the target when the rain stops the game. They considered the number of wickets in hands, which was a unique attribute compared to old methods like the average run rate method, most productive overs method, PARAB method, and Clarke's method. This attribute helped them to calculate the revised target more accurately. Figure 1 suggests the exponential relation between the number of overs remaining and wickets in hand, which they first identified. Based on this exponential relationship, they formulated a run production function to calculate the runs that can be scored in a specific number of overs and wickets in hands. They established few criteria that must be followed before applying the method on any match. The relative position of both teams must be exact before the interruption, sensible result in all situations, independent of team1's scoring pattern.

As cricket evolved, the average run rate of the one-day international matches also increased slightly, which suggested an increase in the number of high scoring matches. So Duckworth and Lewis (2004) updated their original model. In the previous model, the resource was declining exponentially for the number of wickets. However, in the updated model, they linearly declined the resource for the given number of wickets because each over is valued equally, the distribution of runs scored per over tends to be uniform, as long as the number of wickets lost remains constant.

Jayadevan (2002) found that the pattern of runs scoring in One-Day cricket is not exponential, as mentioned in the Duckworth Lewis method. After analysing the data of One-Day cricket matches, he found that the scoring rate of the first ten overs is always high due to field restrictions and then in the middle over scoring rate drops and again in death overs scoring rate increases. His method is based on two curves a normal curve which is a scoring pattern that the team follows before an interruption, and a target curve, which is the opposite of a normal curve because, after the interruption, it is evident that the number of overs is going to be reduced and the team will accelerate before the usual death overs. His analysis led him to found a cubical polynomial equation that was most suitable to represent the scoring pattern in One-Day cricket since the rate of progress in the score is not uniform in the normal curve. He found that due to the use of exponential curves in the Duckworth Lewis method, the revised target whenever a team scores above 300 runs is relatively low, but in his method, the revised for the team who score above 300 runs does not get affected that much due to the use of target curves which is designed such a way that it does not decrease drastically.

The increasing trend of high scoring matches led to the need for the Duckworth Lewis Stern method. The research performed by Steven (2016) updated the Duckworth Lewis method. He analysed the scoring pattern of high scoring matches (50 overs - 300 plus runs) and found that the exponential curve of such matches is straight compared to low or average scoring matches. To bring this nature of scoring pattern in the Duckworth Lewis method researcher introduced the damping factor in the equation. The data indicated that damping in accelerations takes place rapidly in an early stage of 50 over innings and more slowly at the end of innings. While scoring on the last ball of the innings, wickets in hands must not be the factor to influence the score. He analysed this criterion and updated this under his method. These were the two significant changes in the Duckworth Lewis method, which helped calculate the more realistic target compared to the modern-

day scoring pattern.

Jewson and French (2018) performed an analysis on the Duckworth Lewis method. They basically target the matches which were high scoring because in the traditional Duckworth Lewis method was increasing proportionally, but in a high scoring, this should not be the case. The run rate is always high, usually throughout the game. They had used English country cricket ball by ball data in their analysis. In order to observe the difference between a moderate scoring game and high scoring game, they observed the median runs coming for any resource combination. Since Duckworth Lewis is relative to the number of resources available, during this analysis, they found that in the case of high scoring games or especially in T20 cricket, it is not relative as the team tries to score at a high run rate even though they lose their few resources. They questioned the fairness of the Duckworth Lewis method, which only considers two parameters (overs remaining and wickets in hand). Based on ESPN-Cricinfo data, they described how toss could be influential in the rain-affected matches.

2.2 Literature review of Duckworth Lewis Method using machine learning

Bhattacharya et al. (2011) identified that the Duckworth Lewis method is parametric; however, various non-parametric curves might be used to fit the data and suggest using a possible advantage while using the non-parametric method. They estimated resources based on the data, but it suffered from the untidiness of observed data. Therefore isotonic regression was used to estimate missing values from the data and convert resources such that they decrease monotonically with remaining overs and wickets lost. A Bayesian strategy was used to produce a strictly monotonically decreasing resource table, and Gibbs sampling was used to obtain estimates of the posterior means of the resources used to create the T20 resource table. Since this model is non-parametric, it relied completely on observed data, so there was a need for data in which all possible combinations of overs and wickets were covered. They used only 85 matches data to create this resource table. Using isotonic regression, they estimated the data where no data was available and corrected the value for those with a low sample size. While concluding their research, they pointed out that their method is not intended to replace the Duckworth Lewis resource table; rather, it was intended to illustrate some problems with the Duckworth Lewis table.

Mchale and Asif (2013) proposed a modified Duckworth Lewis model. While analysing the Duckworth Lewis method, they found that the method shows irregular patterns for the value of successive wickets. The alternate wicket partnership was valued more, so there were peaks identified in the graph of wicket partnership against runs. Duckworth and Lewis themselves had confirmed this problem and stated a need for smoothing for this problem. They achieved smoothing of those lines and formulated the new equation for function $F(w)$, which is translated as the percentage of runs scored when w wickets are lost versus when no wickets are lost. After this update, the model they had to change the function $Z(w)$ used for calculating the run remaining to be scored for a given number of wickets lost. Since they used cumulative function for $F(w)$, this provided them with a wide range of curves on which they can model $Z(w)$. They used the truncated Cauchy distribution as it provided the heavier tail instead of exponential, which was required since the contribution from lower-order batters was increased considerably.

Shah et al. (2015) reviewed the Duckworth Lewis method and identified the short-

comings of the method. In this research, they used the data of One-day international matches affected by the weather. Their first analysis was that the team winning the toss wins the match in 66% of cases when Duckworth Lewis is applied. They analysed that the team batting first wins in 64% of cases when Duckworth Lewis is used for target resetting. Using a statistical test, they found that the average run rate difference between the winning and losing teams is not significant. Based on this analysis, they concluded that the Duckworth Lewis method is influenced more by wickets than run rate or runs scored.

Jaipal (2017) proposed a method for improving the Duckworth Lewis method using machine learning methods such as support vector machine, logistic regression, binomial logistic regression, decision tree, random forest, and neural network. He used match summary data consisting of 32 variables. In his research, he encountered the class imbalance problem that he later solved using a library in R called ROSE (Randomly Oversampling Examples). He used the sampling method for his dataset because it helped him predict the resource and score correctly as the sensitivity increased. He created the Duckworth Lewis method using machine learning methods based on wickets remaining and overs remaining data. He also created an improved version of this method by adding net run rate, winning toss wins or loses the match. He concluded his research by comparing the accuracy of each model with its improved version and found that the improved version surpassed the traditional method except the logistic regression model, in which accuracy was decreased in the respective improved version of the model.

Abbas and Haider (2019) proposed a method to develop the Duckworth Lewis method using machine learning and also compared the same with the original method. They downloaded all the one-day international matches HTML pages to scrape attributes for each match and calculated the Duckworth Lewis par score for each over of every match. They applied neural network and bagging with Naïves algorithm and computed Duckworth Lewis winning prediction. Comparing this predictor variable with the actual result of the match, they got to know that Duckworth Lewis is performing less accurate for the first four wickets. When reviewing the Duckworth Lewis method, they observed that the table decreases from top to bottom, which is not ideal for how one-day cricket is played today. Whenever wickets fall in quick succession batting team tries to play conservative cricket, which is not reflected in the Duckworth Lewis graph. Researchers applied Particle Swarm Optimization (PSO) by developing a custom software application in .NET to optimise the values of the Duckworth Lewis table.

2.3 Literature review of outcome of a match using machine learning

Considerable research has been carried out in predicting the outcome of a match. This research will help identify the essential variables in cricket-related dataset and the implementation of features in our study.

Passi and Pandey (2018) created different features to calculate the players attributes to predict the performance of players. In their analysis, they categorised players as batsmen and bowlers. For batters, they had used a number of innings, batting average (average number of runs scored in each inning), strike rate (percentage of runs scored on each ball), number of centuries, number of ducks in the career, and highest score. For Bowlers, they had used the number of innings, the number of overs, bowling average, bowling strike rate, and five-wicket hauls. These parameters are used to define the abilities

of the players. Researchers used the weighting technique and weighted each performance indicator based on its relative value to other indicators. They had used the analytic hierarchy process to calculate these weights. Based on these weights, they formulated an equation to evaluate the player's consistency, form, and performance against a particular opponent. After generating this data, they applied Naïves Bayes, decision tree, random forest and support vector machine to predict the number of runs or predict the number of wickets. Random forest was one of the accurate classifiers in their research with respect to precision, F1 score, recall, AUROC, and RMSE.

Jayalath (2018) proposed an approach for analysing one-day international cricket match predictors. External factors sometimes influence the outcome of a match or players performance. Factors such as home advantage, toss result, batting first or second, and day vs day-night game format are discussed in this research. They formulated a logistic regression that predicts a team winning against a particular opponent based on previous results. They included predictors such as a home game, day game or day-night game, toss, batting first, opponent belongs to which continent in their model. This model suggested that home advantage is a significant factor for the majority of the teams. They categorised data based on the day and day-night games for further analysis and found that toss is a significant factor. Based on this result, they decided to use the result of toss to build a classification tree model to predict the outcome of a match. Further, they changed the predictor, which was an outcome of the match, to the margin of victory and performed a regression tree approach as this method helped interpret the results more clearly.

Nimmagadda et al. (2018) used statistical techniques to predict the outcome of a T20 match while it was still in progress. A multiple regression model is evaluated in order to create a prediction model. The key outcome was determined by the impact of the toss winner and the resulting match-winner. The predictive model used the innings score at regular intervals and the final scores to anticipate the match result. The model predicted score and run rate projected score was relatively close to the final score, with the model's score being more accurate when compared to the actual score.

Sudhamathy and Meenakshi (2020) used the IPL dataset consisting of matches starting from 2008 to 2017. They used the match summary dataset. They used the Boruta and Importance function for feature selection and found that umpire and venue are insignificant variables in the dataset. They used decision tree, random forest, Naïves Bayes, and k-nearest neighbour to predict the winner of IPL. Based on the IPL data, their model suggested that the Kolkata Knight Riders has more probability of winning.

Sinha et al. (2020) implemented a system to predict wins to improve the team performance using a support vector machine. They scraped the data from the official site of the Indian Premier League. They quantified player performance using multiple statistical indicators, and a rating index was established. The model was then run on the players chosen for both sides, and a prediction of win or loss probability was provided. Similar players were identified using KNN and K-mean using the data of their performance. This model was able to identify the replacement of injured players. They obtained 96.3% accuracy for the SVM model.

Tripathi et al. (2020) implemented IPL match prediction using machine learning while tackling ambiguity in results. They collected historical data from various sites and created their player and team databases. They implemented feature engineering in which they included city, toss winner and toss decision. Also, for player's data, they added features such as the batting score of a player, bowling score of a player, a total score

of a player, and team strength. An analytic hierarchy process was used to obtain the weights used to calculate batting and bowling features. They implemented models using Naïves Bayes, AdaBoost, logistic regression, support vector machine, KNN, XGBoost, extra tree classifier and random forest classifier. These models were compared based on accuracy, cohen kappa and ambiguity to identify the issue of data symmetry. Models performance was tuned using hyperparameter tuning, and the random forest model was the best model with a standard deviation of 6.3%.

Kapadia et al. (2019) discussed whether machine learning could help predict accurate match results of IPL matches. They designed models to predict the outcome of the match based on the home ground factor and toss decision. Feature selection was applied to eliminate irrelevant features from the dataset. Home ground and toss winner features were created after the data processing. Their results indicated that toss results features performed slightly better to predict the match's outcome using the KNN algorithm based on accuracy, precision and recall. Naïves Bayes results were low for toss related features, but for home ground related features Naïves Bayes performance was reasonable compared to other algorithms.

Banasode et al. (2020) implemented a system to predict and analyse the results of IPL matches. Their study used the IPL dataset to analyse the runs scored by batsmen of each team over the years, batsmen performance across all venues to find their favourite venue. They calculated the bowler's economy rate and analysed bowlers for the economy rate over the years. Further, they analysed the performance of batters against all bowlers and bowlers distribution of wickets over the years. In their analysis, they found toss decision influences the outcome of the match.

Abdul et al. (2020) implemented a model to predict a winner in the T20 world cup. T20 International results, ranking of teams, fixtures of all teams, qualifiers and past appearances in the T20 world cup. Random forest, extra trees, ID3 and C4.5 algorithms were used in the research, and among them, the random forest was the best programming algorithm based on accuracy and residual score. The model predicted Australia and England would be the two teams in the final.

3 Methodology

Knowledge Discovery Database (KDD) has been adopted for this research. It starts with data source which contains the raw data. In our case, the raw data is the data file which we downloaded from the Kaggle. Further, we have convert this data in to an appropriate type, selecting particular subset of data which is required for the study, cleaning the data by removing unwanted noise and filling up the missing values and in the last step of data processing we have performed transformations on the dataset. So after selection prepossessing and transformation our data is ready for applying a data mining. In data mining code process, we discover a pattern by producing the models. In the last step of this methodology, we convert this statistical model in to visualisation and knowledge is then inferred with help of this visualisations.

This research involves implementing the solution for calculating the score of rain-affected matches or providing an alternative method for Duckworth Lewis, which is currently used for calculating the score of the rain-affected matches. As we discussed in the section 2.3, about features that were important to predict an outcome of the match. We have decided to implement few features in our research as they were significant features.

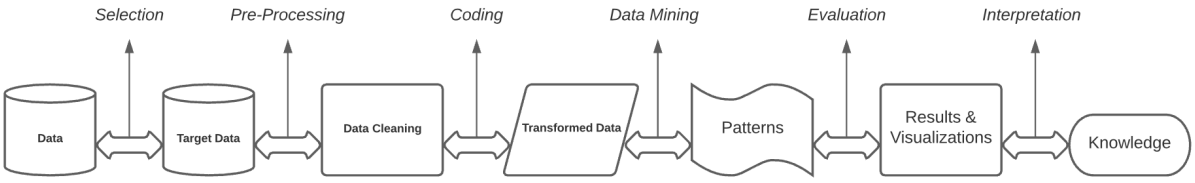


Figure 2: Knowledge Discovery Database Methodology

Figure 3 depicts the framework for the proposed research.

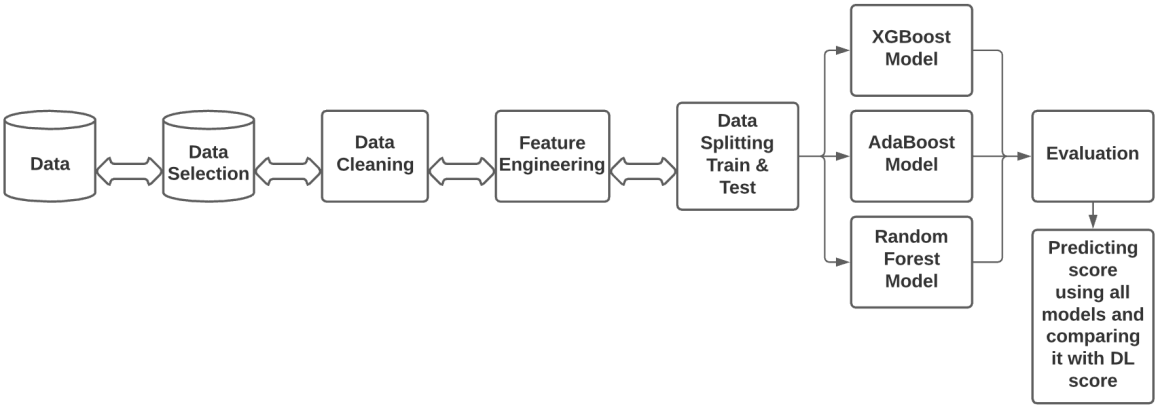


Figure 3: Research Framework

In this research, we have used the Indian Premier League (IPL) Dataset. IPL is one of the popular and challenging T20 leagues in the world. We have downloaded the dataset from the Kaggle. The dataset contains two files, a match summary that consists of team details, outcome, venue, toss decision and ball by ball detail file consists of team details, runs scored on every ball, wicket lost, extras and player details. We have used the only ball by ball dataset that consists of 18 columns and 34,82,424 records. This dataset contains records of 816 matches that were played from the 2008 to 2020 season. [1]

3.1 Feature Engineering

The dataset consisted of only a few variables like the number of over, balls, runs per ball, extras, boundaries, and wicket. It was impossible to use these variables to build a dataset that will be able to predict rain-interrupted match scores. So the idea behind feature engineering was to use this raw data to implement features that will help the model predict accurately. We added a column 'total runs scored till current ball' with the help of run per ball column. Further, we added a column for 'total wickets' in an innings because the raw dataset only suggested if a wicket is fallen or not, which could not add any relation with the score. Our main features in the project are the number of runs scored, the number of wickets fallen, dot balls scored, and the number of boundaries scored in the last five overs at any match stage. Cricket is a game of momentum, and the team with good momentum wins the match most of the time. Based on this idea, we have tried to add these features to the data, which will help the model understand the match's condition.

[1] `https://www.kaggle.com/patrickb1912/ipl-complete-dataset-20082020`

Table 2: Feature Engineering

total_ score	total_ wickets	total	prev_ runs_ 5_overs	prev_ wickets_ 5_overs	prev_ 5_overs_ dot_balls	prev_ 5_overs_ boundaries
1	0	222	1.0	0.0	0.0	0.0
1	0	222	1.0	0.0	1.0	0.0
2	0	222	2.0	0.0	1.0	0.0
2	0	222	2.0	0.0	2.0	0.0
2	0	222	2.0	0.0	3.0	0.0

3.2 Hyperparameter Optimization

Model optimisation was one of the time consuming process in our research. In machine learning, hyperparameter refers to the parameters of a machine learning algorithm that produce the best results when tested on a validation set. They are set before the training. The number of trees in a random forest, XGBoost, and AdaBoost is a hyperparameter learned during the training. Table 3 shows the hyperparameters used in our research for respective methods. Hyperparameter optimisation finds a combination of hyperparameters that returns an optimal model, which reduces a predefined loss function and, in turn, increases the accuracy on given independent data. Hyperparameters can have a direct impact on the training of machine learning algorithms. This to achieve maximal performance, it is important to understand how to optimise them. Often some of the hyperparameters matter much more than others.

Random search finds good values with great precision for essential parameters. Random search sets up a grid of hyperparameter values and selects random combinations to train the model and score. This allows to explicitly control the number of parameter combinations that are attempted. The number of search iteration is set based on time or resources. Bergstra and Bengio (2012) showed that the random trials are more more efficient than grid trials for hyper parameter optimizations. One of the important advantage of random search is that it does not allocate too many trials to explore unimportant decisions.

Our research begins with data cleaning steps to deal with the null values in the dataset. The column that indicates whether a player is dismissed or not contains the null value. We are replacing this null values with zero. Although we will not use this variable in model building but we will use this variable to create wickets in last 5 overs variable. As we discussed in section 3.1, our next step involves the feature engineering process. After performing all data preparation and cleaning processes, our next step is to build a model using extreme gradient boosting (XGBoost), adaptive boosting (AdaBoost), and random forest algorithms. The purpose behind using these algorithms is that there is no mathematical relations between independent and dependent variables, so in such case decision tree methods can be useful. XGBoost and AdaBoost are the novel methods for this research. As we seen in the section 2.2, random forest was used for this problem but in those research different dataset was used, we will verify if our transformed data helps the random forest to yeild better results. The next step involves the evaluation of these models using evaluation metrics like root mean squared error (RMSE), mean absolute error (MAE), and variance explained. Our last step of the research involves comparing

Table 3: Hyperparameters used in the models

Hyperparameters	Description	Model
learning_rate	To correct the residual errors in the prediction, new trees are created based on previous sequence of trees. This effect can lead to overfitting. Learning rate allows to slow down the learning by applying weighting for the corrections.	XGBoost and AdaBoost
max_depth	It is used to avoid overfitting by restricting the growth of tree. Higher depth leads model to analyse pattern for particular feature	XGBoost and Random forest
min_child_weight	It is the minimum sum of weights of all observations required in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning.	XGBoost
gamma	It allows split only when there is positive reduction in the loss function. It specifies the minimal loss reduction to make a split.	XGBoost
colsample_bytree	It is the percentage of features considered for producing trees. This sub-sampling of features avoids model to train on one strong specific feature.	XGBoost
n_estimators	It specifies the number of stumps or decision tree	AdaBoost and Random forest

the machine learning models with the actual Duckworth Lewis score.

4 Design Specification

We have used three machine-learning algorithms in our research that are XGBoost, AdaBoost and Random Forest. All three algorithms are used for estimating relationships between a 'total score' variable and the rest of other variables, also known as independent variables, for regression analysis. It is used to determine the strength of a relationship between variables and to predict how they will interact in the future.

4.1 Extreme Gradient Boosting (XGBoost)

Tianqi and Carlos (2016) implemented the advanced version of the gradient boosting method known as the extreme gradient boosting (XGBoost) method. The main two features of this algorithm are that it is fast and performs well compared to the other machine learning methods. It uses the parallelisation concept, cache optimisation performed by keeping all its intermediate statistics in the memory, out of memory computation that optimises the memory so it can work on the data that is larger than the size of the memory. All these features help the XGBoost model to perform faster. It involves regularisation

that helps the model to prevent overfitting. It is derived from the decision tree model, but the XGBoost model can auto prune to maintain bias-variance. It can handle the missing value that makes the XGBoost model perform effectively.

4.2 Adaptive Boosting (AdaBoost)

Freund and Schapire (1997) implemented an ensemble learning method known as adaptive boosting algorithm (AdaBoost). It involves a sequential process in which trees are dependent on the output of the previous tree. The decision trees in this model do not have equal weightage. Also, the decision trees are not large, they just have one root node and two leaf nodes, and they are also known as stumps in the AdaBoost method. At the start, the algorithm provides the weights to each record such that the summation of weights is one that suggests all variable is equally important. Features in the data are used to create the stumps, and these stumps are further compared based on the entropy and Gini index, and the stump, which has a lesser value, is selected for the base learner model. The performance of the stump is calculated to update the initial weights. The weight of the wrong classified record is increased, and the weights of the rest of the other records are decreased. The process will continue depending on the number of estimators passed in the AdaBoost function. The mean value predicted by the stumps is considered during testing to combine the weak learners and make them strong learners.

4.3 Random Forest

Breiman (2001) implemented an extension of the random decision tree algorithm know as random forest. Random forests minimise the overfitting and high variance problem that arises when tree length increases. Bags are created in this method where the subset of records and columns are selected, and a decision tree is fitted on each bag. While testing, random forest calculates the mean of predicted values of all decision trees. Bagging provides a model to extract the pattern from all features instead of extracting pattern only from strong variables.

5 Implementation

Our research started by reading the data and creating a dataset sub-sample consisting of only necessary variables. The data contains matches from the 2008 season to the 2020 season. The dataset contains teams whose names were changed, and few teams like Pune, Kochi and Gujarat played the IPL for two or three seasons. To make data consistent, we decided to remove Pune, Kochi and Gujarat team data from the dataset and rename teams to their latest updated name. In order to make easily readable data, we decided to replace the team name with the initials of the teams. Further, feature engineering was implemented on the data. As Team 2, which bats second, knows the target, which can influence the scoring pattern, we decided to build separate models for the first and second innings. We produce two models based on first and second innings data for each algorithm. The batting team and bowling team variables were categorical variables, so we implemented one-hot encoding. Data was divided into a ratio of 75:25 for training and testing purposes.

Python version 3.7 is used for implementing the research, and the code is written using google colab. The miscellaneous operating system interface (OS) library reads the

data from the given directory. Data was stored on google drive, so the google colab library was used to link the drive. Numpy library was used to store the train and test data. Data were split using the sci-kit learn library. Sci-kit learn libraries are utilised for Random forest, Adaptive boosting and RandomSearchCV. Also, the matplot library was used to visualise the variance in the models. The configuration of the machine used for the research is as follows:

- Intel Core i3 10^{th} Gen (macOS)
- 1.1 GHz Dual-Core
- 8 Gb Ram
- 256 Gb SSD

6 Evaluation

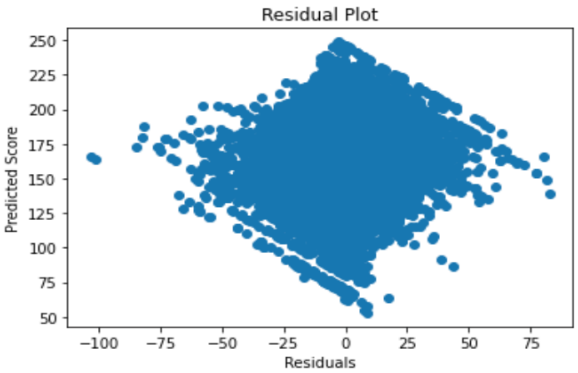
Extensive analysis has been performed on all three models. Hyperparameter tuning was performed to obtain the best performance to produce the best models. Data was first initially split according to the first and second innings. The first innings data consists of 86,121 records and the second innings data consists of 80,341 records. Further, each innings data was split into training and test data. All the results obtained from the models are evaluated using the Mean absolute error (MAE), Root Mean Squared Error (RMSE), and Variance explained.

6.1 Extreme Gradient Boosting Model (XGBoost)

Random search cross-validation was performed to find the best hyperparameters for the XGBoost model on both innings data. The model was initialised with the best parameters. Figure 4a and figure 5a shows the parameters used for each innings respectively. The results varied after hyperparameter tuning. 84.8 % accuracy was obtained on training data of first innings data, and 78% accuracy was obtained for the validation set of first innings. Compared to the first innings, the model obtained using second innings data performed excellent with the training accuracy of 99.9% and validation accuracy of 99.2%.

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.5, gamma=0.4,
             importance_type='gain', learning_rate=0.3, max_delta_step=0,
             max_depth=8, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

(a) XGBoost Model Parameters for First Innings



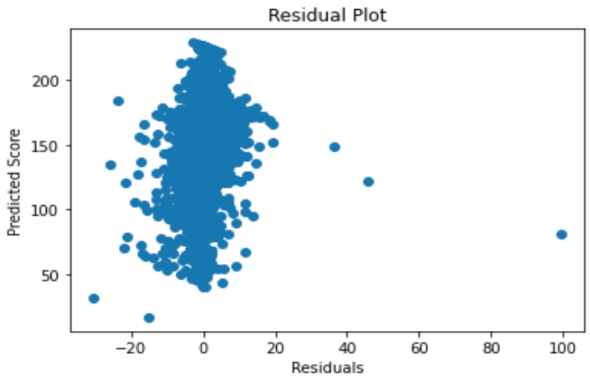
(b) XGBoost Model First Innings Residual Plot

Figure 4: XGBoost Model - First Innings

The first innings model's MAE is 9.92, which suggest that the error between the predicted score and actual is not high, and the predicted score is deviated by nine runs. RMSE suggested that the predicted score can be deviated from the actual score by 13 runs. Variance explained score is 78% suggest that model is not biased. Also, Figure 5b helps to understand that most of the predicted score deviated by actual score around 25 runs.

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.7, gamma=0.0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=15, min_child_weight=7, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

(a) XGBoost Model Parameters for Second Innings



(b) XGBoost Model Second Innings Residual Plot

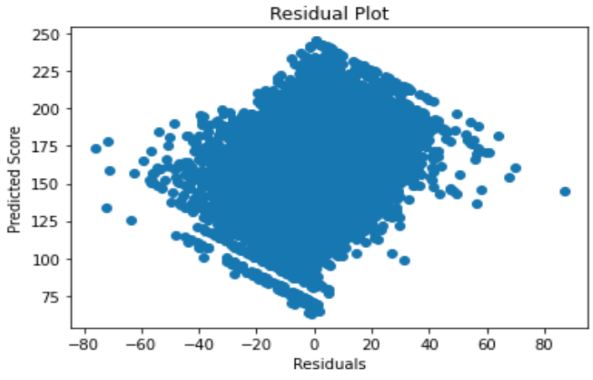
Figure 5: XGBoost Model - Second Innings

6.2 Adaptive Boosting Model (AdaBoost)

In the adaptive boosting algorithm, we had used XGBoost base learners. We passed the same hyperparameters for the base learners that we used in the XGBoost model. Figure 6a shows the parameters used in the model. Since we used XGBoost weak learner, it is evident that the accuracy of AdaBoost 1st inning model compared to XGBoost first innings model should increase. Our AdaBoost model performed as expected as the accuracy of training data is 89.4%, and validation accuracy is 82.2% which is better than the XGBoost model. Figure 6b suggest that error is more for predicting score around 150 runs.

```
AdaBoostRegressor(base_estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                              colsample_bylevel=1,
                                              colsample_bynode=1,
                                              colsample_bytree=0.5, gamma=0.4,
                                              importance_type='gain',
                                              learning_rate=0.3,
                                              max_delta_step=0, max_depth=8,
                                              min_child_weight=1, missing=None,
                                              n_estimators=100, n_jobs=1,
                                              nthread=None,
                                              objective='reg:squarederror',
                                              random_state=0, reg_alpha=0,
                                              reg_lambda=1, scale_pos_weight=1,
                                              seed=None, silent=None,
                                              subsample=1, verbosity=1),
                  learning_rate=0.05, loss='linear', n_estimators=90,
                  random_state=None)
```

(a) AdaBoost Model Parameters for First Innings



(b) AdaBoost Model First Innings Residual Plot

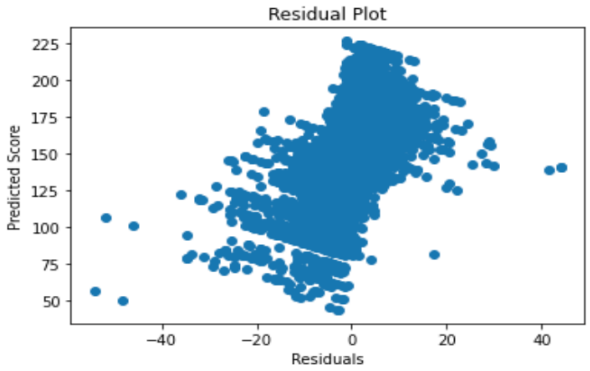
Figure 6: AdaBoost Model - First Innings

The second innings model using the AdaBoost algorithm is the best model, with a training accuracy of 99% and a validation accuracy of 97.6%. Also, the RMSE and MAE

for this model were dropped to 4.43 and 2.94 respectively. Figure 7b is a similar result that we obtained in the XGBoost method.

```
AdaBoostRegressor(base_estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                              colsample_bylevel=1,
                                              colsample_bynode=1,
                                              colsample_bytree=0.3, gamma=0.3,
                                              importance_type='gain',
                                              learning_rate=0.2,
                                              max_delta_step=0, max_depth=15,
                                              min_child_weight=1, missing=None,
                                              n_estimators=100, n_jobs=1,
                                              nthread=None,
                                              objective='reg:squarederror',
                                              random_state=0, reg_alpha=0,
                                              reg_lambda=1, scale_pos_weight=1,
                                              seed=None, silent=None,
                                              subsample=1, verbosity=1),
                  learning_rate=0.05, loss='linear', n_estimators=100,
                  random_state=None)
```

(a) AdaBoost Model Parameters for Second Innings



(b) AdaBoost Model Second Innings Residual Plot

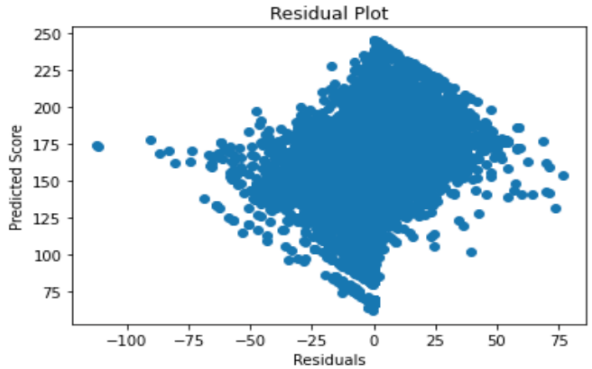
Figure 7: AdaBoost Model - Second Innings

6.3 Random Forest Model

Random search cross validation was performed to find best parameter to tune the random forest model. Figure 8a shows the best parameter obtained in hyper parameter tuning. Among all three models, random forest algorithm for first innings data performed better with a training accuracy of 94.3% and validation accuracy of 86.1%. RMSE and MAE value for random forest was 10.98 and 7.16 respectively which were lowest for first innings model when compared with other two models. The residuals plot shown in figure 8b is similar to other two models.

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=20, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=60, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

(a) Random forest Model Parameters for First Innings



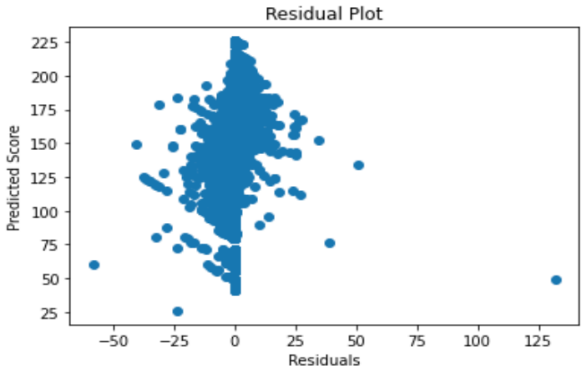
(b) Random Model First Innings Residual Plot

Figure 8: Random forest Model - First Innings

As we seen in other two models, similar trend was seen in the random forest model. The training and validation accuracy of the model observed was 99.7% and 99%, respectively. RMSE and MAE values are 2.76 and 1.31, which were closest to the XGBoost model. Figure 9b suggest that residuals are scattered less, and we can observe that margin of error is very less when the model is predicting based on validation data.

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=20, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

(a) Random forest Model Parameters for Second Innings



(b) Random Model Second Innings Residual Plot

Figure 9: Random forest Model - Second Innings

Table 4: Evaluation Metrics - First Innings Models

First Innings				
Model	Training Accuracy	Validation Accuracy	RMSE	MAE
XGBoost	84.8%	78.0%	13.82	9.92
AdaBoost	89.4%	82.2%	12.45	9.33
Random Forest	94.3%	86.1%	10.98	7.16

Table 5: Evaluation Metrics - Second Innings Models

Second Innings				
Model	Training Accuracy	Validation Accuracy	RMSE	MAE
XGBoost	99.9%	99.2%	2.58	1.55
AdaBoost	99.9%	97.5%	4.43	2.94
Random Forest	99.7%	99.0%	2.76	1.13

6.4 Discussion

Jaipal (2017) in his research provided the Duckworth Lewis based random forest model which obtained an accuracy of 95% . The research followed the basic idea of the Duckworth Lewis method that is using overs remaining and wickets remaining data of the match. Along with this data, the researcher added two features, such as the win or loss variable and the run rate, which were significant as they increased the model's performance. Based on the results obtained, it is evident that our model performed slightly better than the mentioned research for the random forest algorithm.

In the second innings model, we have included the first innings total column, which helped the model to perform well. All three models predicted using less margin of error using the second innings model. XGBoost was the best model for the second innings data. We also observed a trend in first innings models, as they were creating more errors

while predicting the score of matches which ranges around 150 runs. The average score of IPL matches ranges from 150 to 160 runs, so the data for this range is more compared to matches where the score of a match is above 200 runs or below 100 runs.

We compared our model with the actual match situation. We considered an IPL match played between Sunrisers Hyderabad (SRH) and Delhi Capitals (DC) in 2021. This match was finished without an interruption, but to test our model, we are considering that match has been interrupted at 14^{th} over in the second innings. Feeding all details to XGBoost model, our model predicted that the SRH lost the match by three runs, and according to the Duckworth Lewis method, SRH won the match by seven runs, but the actual result of the match was a tie which suggests that our model predicted score with less margin of error. Table 6 suggest that random forest model predicted score more accurately and overall all our three models performed better for predicting the runs.

Table 6: Comparison of method using an IPL match SRH vs DC 2021

Duckworth Lewis Method	SRH won by 7 runs
XGBoost Model	SRH lost by 3 runs
AdaBoost Model	SRH lost by 4 runs
Random forest Model	SRH lost by 1 run
Actual Result	Match was tie

7 Conclusion and Future Work

The research aimed whether machine learning model can used to predict score in rain interrupted matches. In order to address this study, we studied about the features which can be useful during our research in the section 2.3. Hyperparameter optimization and random search cross validation method were used to find the best parameter for all three models. As seen in the section 6, all models were trained and tested on the transformed data and result shows that margin of error for our model is less compared to the Duckworth Lewis method.

The proposed models produces good results with less RMSE and MAE values as well as good variance score. The first innings model performance was not however good when compared to second innings model. Second innings model contains first innings total similarly in future work could involve considering the average score of each team in IPL. Also features related to team strength based on previous performance or rankings. Batters and bowlers feature can also considered to check if model produces better results.

Acknowledgement

Firstly, I would like to thank my supervisor, Dr Noel Cosgrave, for his constant support, encouragement and guidance that I received from him throughout the course of this project. I would like to thanks my classmates, friends and roommates for providing suggestions and feedback for the research and last, but not least, I would like to thank my parents for their support to help me to achieve my goals.

References

Abbas, K. and Haider, S. (2019). Duckworth-lewis-stern method comparison with machine learning approach, *International Conference on Frontiers of Information Technology (FIT)* pp. 197–1975.

Abdul, B., Bux, A. M., Hassan, J. F., Majdah, A., Kashif, M. and Ali, S. R. (2020). Icc t20 cricket world cup 2020 winner prediction using machine learning techniques, *2020 IEEE 23rd International Multitopic Conference (INMIC)* pp. 1–8.

Banasode, P., Patil, M. and Verma, S. (2020). Analysis and predicting results of ipl t20 matches, *IOP Conf. Series: Materials Science and Engineering* **1065**(1): 1–8.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization, *Journal of Machine Learning Research* **13**: 281–305.

Bhattacharya, R., Gill, P. S. and Swartz, T. B. (2011). Duckworth–lewis and twenty20 cricket, *Journal of the Operational Research Society* **62**(11): 1951–1957.

Breiman, L. (2001). Random forests, *Machine Learning* **45**: 5–32.

Duckworth, F. C. and Lewis, A. J. (1998). A fair method for resetting the target in interrupted one-day cricket matches, *Journal of the Operational Research Society* **49**(3): 220–227.

Duckworth, F. C. and Lewis, A. J. (2004). A successful operational research intervention in one-day cricket,, *Journal of the Operational Research Society* **55**(7): 749–759.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**: 119–139.

Jaipal, M. S. (2017). Improving duckworth lewis method by using machine learning.

Jayadevan, V. (2002). A new method for the computation of target scores in interrupted, limited-over cricket matches, *Current science* **83**: 577–586.

Jayalath, K. (2018). A machine learning approach to analyze odi cricket predictors, *Journal of Sports Analytics* **04**: 73–84.

Jewson, J. and French, S. (2018). A comment on the duckworth–lewis–stern method, *Journal of the Operational Research Society* **69**(7): 1160–1163.

Kapadia, K., Abdel-Jaber, H., Thabtah, F. and Hadi, W. (2019). Sport analytics for cricket game results using machine learning: An experimental study, *Applied Computing and Informatics* .

Mchale, I. and Asif, M. (2013). A modified duckworth–lewis method for adjusting targets in interrupted limited overs cricket, *European Journal of Operational Research* **225**(2): 353–362.

Nimmagadda, A., Kalyan, N. V., Venkatesh, M., Teja, N. and Raju, C. G. (2018). Cricket score and winning prediction using data mining, *International Journal for Advance Research and Development* **03**: 299–302.

Passi, K. and Pandey, N. (2018). Increased prediction accuracy in the game of cricket using machine learning, *International Journal of Data Mining & Knowledge Management Process (IJDKP)* **08**(02): 19–36.

Shah, H., Sampat, J., Savla, R. and Bhowmick, K. (2015). Review of duckworth lewis method, *ISSN* **45**: 95–100.

Sinha, S., Tripathi, P., Vishwakarma, A. and Sankhe, A. (2020). Ipl win prediction system to improve team performance using svm, *International Journal of Future Generation Communication and Networking* **13**(01): 17–23.

Steven, S. (2016). The duckworth-lewis-stern method: extending the duckworth-lewis methodology to deal with modern scoring rates, *Journal of the Operational Research Society* **67**(12): 1469–1480.

Sudhamathy, G. and Meenakshi, G. R. (2020). Prediction on ipl data using machine learning techniques in r package, *CTACT Journal on Soft Computing* **11**(01): 2199–2204.

Tianqi, C. and Carlos, G. (2016). Xgboost: A scalable tree boosting system, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* .

Tripathi, A., Islam, R., Khandor, V. and Murugan, V. (2020). Prediction of ipl matches using machine learning while tackling ambiguity in results, *Indian Journal of Science and Technology* **13**(38): 17–23.

Wikipedia, the free encyclopedia (2007). Duckworth lewis resource graph, `https://en.wikipedia.org/wiki/User:Jono4174#/media/File:DuckworthLewisEng.png`. Online; accessed July, 2021.