

**A PROJECT REPORT ON**  
**“IPL CRICKET SCORE PREDICTION”**  
**BY**  
**Ms. Akshata Shirwale**  
**[34303]**

**IN PARTIAL FULFILLMENT OF**  
**M.Sc. (Computer Science) Part II**

**Semester III**

**SIR PARASHURAMBHAU COLLEGE, PUNE**

**(2022-2023)**



Shikshana Prasarak Mandali's

**SIR PARASHURAMBHAU COLLEGE**  
**AUTONOMOUS**

TILAK ROAD, PUNE – 411 030.

Department of Computer Science

*Certificate*

*This is to certify that Ms. Akshata Dattatray Shirwale has  
presented project titled IPL CRICKET SCORE PREDICATION  
for M.Sc. (Computer Science) Part II Semester III.*

*Date: 10 / 12 / 2023*

*Teacher In-Charge*

*Head,  
Department of Computer Science*

*Internal Examiner*

*External Examiner*

## ACKNOWLEDGEMENT

We would like to express our profound thanks to all those who helped in making this project a reality. Much needed moral support and encouragement was provided on numerous occasions by all teachers and fellow students.

We all are mostly thankful to our Head of the Department **Mr. Deepak Londe** sir, we are also especially grateful and to our teacher and project guide **Mr. Ajay raut** sir for his time to time, much needed, valuable guidance. he has been available to help us whenever required. he has been our guiding star, guiding us through the right direction to make this project a success. Without his full support and cheerful encouragement, we would have not been able to complete the project in time. We are sincerely thankful to him.

## ABSTRACT

Sports data analytics are used not only in cricket but also in other sports for improving the overall team performance and maximizing winning chances. Real-time data analytics can help in gaining insights even during the game for changing tactics by the team and by associated businesses for economic benefits and growth. Besides historical analysis, predictive models are harnessed to determine the possible match outcomes that require significant number crunching and data science, visualization tools and capability to include newer observations in the analysis

The main objective of score prediction/sports analytics is to improve chances of winning by providing a real-time predicted score based on current features i.e runs, wickets, overs, opponents, thereby making the required changes in the batting/bowling lineup& hence improving chances of winning the game.

From the results, we are able to conclude that the Support Vector Machine algorithm has the highest accuracy of the prediction. So, we are using the Support Vector Machine model for the prediction purpose. Teams can use Current Run Rate (CRR) to predict the final score even before the 20 overs have been bowled. So, the teams can know when to accelerate and when to play aggressively to increase the run rate while putting a target. CRR can be used to help the team management to select a team that can improve on the records. It can be used by cricket lovers for predicting the final score of a live IPL match.

# TABLE OF CONTENTS

Sr. No.	Title of Chapter	Page No.
<b>1</b>	<b>Introduction</b>	8
1.1	Motivation	9
1.2	Problem Definition	9
<b>2</b>	<b>Literature Survey</b>	11
2.1	Multivariate data mining approach	15
2.1.1	International Cricket (2016)	15
2.1.2	Predicting ODI Cricket Result (2017)	16
2.2	Inference from the Literature Survey	17
<b>3</b>	<b>Model Architecture</b>	19
3.1	Data Collection	20
3.2	Data Cleaning	20
3.2.1	Validity	21
3.2.2	Accuracy	22
3.2.3	Completeness	22
3.2.4	Consistency	23
3.2.5	Uniformity	23
3.3	Process	23
3.3.1	Data Auditing	23
3.3.2	Workflow Specification	23
3.3.3	Workflow Execution	24
3.3.4	Post-processing and controlling	24
3.4	Others Include	24
3.4.1	Parsing	24
3.4.2	Data Transformation	24
3.4.3	Duplicate Elimination	25
3.4.4	Statistical Methods	25
3.5	Data Processing	25
3.6	Data Splitting	26
3.6.1	Train Set	26
3.6.2	Dev Set	26
3.6.3	Test Set	26
3.6.4	Cross-Validation Set	27
3.6.5	How to split data?	27
3.7	Types of Machine Learning Models	28
3.7.1	Supervised Learning	28
3.7.2	Unsupervised Learning	29
3.7.3	Cluster Analysis	29
3.7.4	Semi - Supervised Learning	29
3.7.5	Reinforcement Learning	30
3.7.6	Dimensionality Reduction	30

3.8	Types of Models used in the Project	30
3.8.1	Linear Regression	30
3.8.2	Ordinary Least Squares	32
3.8.3	Gradient Descent	33
3.8.4	Regularization	33
3.8.5	Making Predictions with Linear Regression	34
3.8.6	Random Forest Regressor	36
3.8.7	Lasso Regression	37
3.8.8	Support Vector Regressor	39
3.9	Model Generation	40
3.10	Final Prediction	41
<b>4</b>	<b>Model Diagrams</b>	42
4.1	System Design	42
4.2	Use - Case UML Diagram	43
4.3	UML Diagram	44
4.4	Deployment Diagram	45
4.5	Activity Diagram	46
4.6	DFD Diagram	47
<b>5</b>	<b>UI Flow</b>	48
5.1	Flask Framework	48
5.1.1	Configuration and Conventions	49
5.1.2	Growing with Flask	49
5.1.3	Minimal Applications	50
5.2	What did the code do?	50
<b>6</b>	<b>Results / Output</b>	52
<b>7</b>	<b>Conclusion and Future Scope</b>	58
7.1	Conclusion	58
7.2	Future Scope	58
<b>8</b>	<b>Publications</b>	59
<b>9</b>	<b>References</b>	60

## List of Figures

ITEM	Page No.
Fig 3.1 : Model Architecture Diagram	19
Fig 4.1 : System Design	42
Fig 4.2 :Use – Case UML Diagram	43
Fig 4.3 : UML Diagram	44
Fig 4.4 : Deployment Diagram	45
Fig 4.5 : Activity Diagram	46
Fig 4.6 : DFD Diagram	47
Fig 5.1 : UI Flow Diagram	48
Fig 6.1 : Final UI of the deployed model.	52
Fig 6.2 : Input values in the model deployed.	52
Fig 6.3 : Final Predicted Score.	53
Fig 6.4 : Comparison chart of different Algorithms used	53
Fig 6.5 :Decision Tree Regression	54
Fig 6.6 :Lasso Regression	54
Fig 6.7 :Linear Regression (Base Model)	55
Fig 6.8 :LTB Regression	55
Fig 6.9 :Ridge Regression	56
Fig 6.10 : SVR Regression	56
Fig 6.11 : Output Table	57

# Cricket Score Prediction”

## 1. INTRODUCTION

Cricket was started in the 16th century in England. Cricket is a sport with multiple formats, different playing standards and varying duration. Twenty20 is one of the three current forms of Cricket which is recognized by the International Cricket Council (ICC). In that format, two teams have a one innings each with a maximum of 20 overs. Because of the short time duration and the excitement, it generates, Twenty20 cricket has become such a huge success. There are many annual tournaments conducted at both domestic and international level. There is huge commercial interest in player performance prediction. This has motivated many analyses of individual and team performance, as well as prediction of future games, across all formats of the game. Currently, strategists rely on a combination of player experience, team constitution and “cricketing sense” for making instantaneous strategic decisions. We choose to focus our testing and evaluation on the most popular format, namely ODI.

Cricket score prediction is an area where the first innings score of a cricket match is predicted using some techniques. There are various systems and prediction methods used to predict the cricket score of the ODI and the T20 cricket matches. CRR method is widely used to predict the score of the cricket match. In the CRR method, the number of runs scored in an over is multiplied by the total number of overs in an innings. This method only focuses on the runs made in an over but it does not focus on the different parameters. This method can only predict the score based on the current score and not based on the various parameters. We are working to improve on the predictions by considering different parameters and to improve the accuracies of the existing systems. We are considering the T20 matches for the score prediction and we will be focusing on the live cricket score prediction. The prediction is based upon factors like the ease of scoring on the day according to the pitch, weather and boundary size. For the team batting first, it gives the prediction of the final total. For the team batting second, it gives the probability of the chasing team winning, although it does not just take the match situation into equation. Predictions are based on the average team playing against the average team in those conditions.<sup>[1]</sup> Principles of Machine Learning are used for developing the prediction system. There are two types of machine learning namely supervised machine learning and unsupervised machine learning. In supervised machine



learning we must train the machine by providing huge data sets and the outcomes. In this project we are going to build a flask app & models using various regression techniques to predict cricket score based on the current user input features.

## 1.1 Motivation

Currently, there is a system which can calculate the current run rate and from it calculates the final score of the team. It doesn't consider the fact about the no of wickets and also where the game is being played. The problem with the current system is that it is unable to predict the score based on the other features like pervious match history score, current run rate etc.

## 1.2 Problem Definition

Cricket is one of the most popular sports in the world, viewed by majority of world's population. It is a game played between two teams of eleven players each. With the advent of statistical modelling in sports, predicting the outcome of a game has been established as a fundamental problem. Cricket is one of the most popular team games in the world. The game of cricket is played in three formats - Test Matches, ODIs and T20s. We focus our research on T20s, the most popular format of the game. With this article, we embark on predicting the outcome of an Indian Premier League (IPL) cricket match. In an IPL season there may be a minimum of 8 to 10 teams playing and each team plays with remaining all teams for a minimum of two times. Matches are held at different venues. Initially toss plays as a crucial factor in deciding the winner of the match. Toss winning team can wish to either field or bat. The team batting first will try to pose as many runs as possible in their 20 overs in order to set a target. The team batting second need to chase the target in order to win the game with wickets in hand. For years while watching limited overs cricket, we have seen projected scores at different intervals being displayed on our television screens. Projected scores are completely based on runs scored and looking at different totals at the end of an innings, using various run rates. For example, if a team's score is 100 at the end of 10 overs. There could be four variations of projected scores:

- Current run-rate: 200
- 6 per over: 160
- 8 per over: 180

- 12 per over: 220

Considering only run rate may not yield good results since various factors might affect the score of the innings. Hence there is a need for ML Model/ML App that will predict the score based on various features present during the game.

## 2. LITERATURE SURVEY

[1] S.Muthuswamy and S.S.Lam (2018) proposed "Bowler Performance for One-Day International Cricket Using Neural Networks," in this paper predicting Indian bowler performance in 7 international teams. The neural network method that uses the Back Propagation Network [BPN] and the Radial Basis Function Network (RBFN) used to predict the performance of the Indian bowler team bowler. The recent performance of the BPN and RBPN model was compared with prediction and classification.

In this paper, the author uses machine learning, backpropagation (backprop,[1] BP) is a widely used algorithm for training feedforward neural networks. Generalizations of backpropagation exist for other artificial neural networks (ANNs), and for functions generally. These classes of algorithms are all referred to generically as "backpropagation".[2] In fitting a neural network, backpropagation computes the gradient of the loss function with respect to the weights of the network for a single input-output example, and does so efficiently, unlike a naive direct computation of the gradient with respect to each weight individually. This efficiency makes it feasible to use gradient methods for training multilayer networks, updating weights to minimize loss; gradient descent, or variants such as stochastic gradient descent, are commonly used. The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.

The term backpropagation strictly refers only to the algorithm for computing the gradient, not how the gradient is used; however, the term is often used loosely to refer to the entire learning algorithm, including how the gradient is used, such as by stochastic gradient descent.[4] Backpropagation generalizes the gradient computation in the delta rule, which is the single-layer version of backpropagation, and is in turn generalized by automatic differentiation, where backpropagation is a special case of reverse accumulation (or "reverse mode").[5] The term backpropagation and its general use in neural networks was announced in Rumelhart, Hinton & Williams, then elaborated and popularized in Rumelhart, Hinton & Williams .

In the field of mathematical modelling, a radial basis function network is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters.

Radial basis function networks have many uses, including function approximation, time series prediction, classification, and system control.

Radial basis function (RBF) networks typically have three layers: an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer. The input can be modelled as a vector of real numbers.

[2] G. D. I. Barr no-B. S. Kantor states that "How to Compare and Select Cricketers in Overs Overs Cricket,". In this paper the author outlines the main criteria for comparing the selection of strikers in limited colleges. This paper shows a clear 2D representation of the strike rate on one axis and the probability of exit, i.e.,  $P(\text{exit})$  on another axis. We are building a strike selection strategy based on this 2D framework that combines scale and strike rate As an example of this application we use this principle in beating the 2003 World Cup performance to show the strong and consistent performance of batsmen playing in Indian and Australia team.

Assessing batting performance in the one-day game, therefore, requires the application of at least a two-dimensional measurement approach because of the time dimension imposed on limited overs cricket. In this paper we use a new graphical representation with Strike rate on one axis and the Probability of getting out on the other, akin to the risk-return framework used in Portfolio Analysis, to obtain useful, direct and comparative insights into batting performance, particularly in the context of the one-day game. Within this two-dimensional framework we develop a selection criterion for batsmen which combines the average and the strike rate. As an example of the application we apply this criterion to the batting performances of the 2003 World Cup. We demonstrate the strong and consistent performances of the Australian and Indian batsmen as well as providing a ranking of batting prowess for the top 20 run scorers in the tournament.

The calculation of batting averages has received some attention in the statistical literature, most particularly from the perspective of the conditions under which the average represents a optimal estimator. Using a reliability and survival analysis approach Kimber et al.<sup>3</sup> note that if the underlying lifetimes (or scores) follow a geometric distribution then the maximum likelihood estimate of the population mean lifetime (or population mean score) is the average. Kimber et al. then go on to propose an alternative non-parametric estimator of the population batting mean which is robust to deviations from the geometric distribution.

The calculation of batting averages has received some attention in the statistical literature, most particularly from the perspective of the conditions under which the average represents an optimal estimator. Using a reliability and survival analysis approach Kimber noted that if the underlying lifetimes (or scores) follow a geometric distribution then the maximum likelihood estimate of the population mean lifetime (or population mean score) is the average. Kimber et al. then go on to propose an alternative non-parametric estimator of the population batting mean which is robust to deviations from the geometric distribution.

The issue of a batsman's scores following a geometric distribution was first raised by Wood[4]. He found there to be considerable empirical support for this contention with the important implication that a batsman's chance of getting out was independent of the number of runs he had scored because of the memoryless property of the geometric distribution. Discussants of this paper at the time, indicated that cricketing lore would not support this position.

The advent and growing importance of the one-day limited overs game has brought a very different emphasis in the analysis of a batsman's contribution to the team's success or failure.

Rather than runs scored, runs scored or conceded per ball faced or delivered has become the essential measure of achievement in the one-day game. Therefore average runs per innings are a much less important estimate of a batsman's capabilities while the strike rate, the average runs scored per 100 balls faced has become the focus of attention.

[3] S. R. Iyer and R. Sharda (2010) reported on "Predicting Performance of Athletes Using Neural Networks: An Application to Cricket Team Selection," used later to predict future performance of players based on their previous performance in which they place the batsmen and throwers in three different categories "maker", "balance" and "failure" with the services of cricket professionals. They show how these heuristic scales will be used in selecting batsmen for the World Cup. By choice the callers should get a 1 or 2 "character" rating or a rated rating but not a "failure" rating. The conditions for selecting throwers are exactly the same as hitting, the thrower has "moderate" and "good" values and does not get a failure rating selected from the team.

The obtained data from [www.Cricketarchive.com](http://www.Cricketarchive.com) (2007) as well as from <http://www.Cricinfo.com> (2007) for our purposes. The data was segregated into bowling and batting i.e., a cricketer is assumed to represent just one category of cricket – bowling or

batting and the cricketers were rated subjectively. Although wicket-keeper is a vital part of the team there is not enough data to evaluate wicketkeepers, therefore we did not develop a model to predict performance of wicket keepers. Batsmen and bowlers were assigned subjective ratings based on heuristic rules.

Two experiments were performed using different neural networks. Neural networks in each experiment were trained and tested using primary ratings. After training and testing, each neural network generated its own ratings for all players.

According to the heuristic rules explained in Section a bowler is rated a “Performer” if he has bowled more than 7000 balls across both versions of the game; captured more than or equal to 150 wickets across both versions of the game, and captured at least 30 wickets in each version of the game. The approximate strike rate (S rate) by a bowler then turns out to be 47. For a bowler, lower the strike rate the better. To analyse the performance of a bowler in the World Cup, we use a benchmark strike rate of 47. If a bowler has a strike rate of lower than 47 we propose that the bowler has performed well. A bowler could capture just one wicket in 30 balls, which would give him a strike rate of 30. However, this performance alone does not lead to conclude that the bowler has performed well. Hence, we also place another filter that the bowler should have captured at least 5 wickets. In summary, the heuristic rules used to evaluate bowler performance

[4] I.P. Wickramasinghe explained "Guessing the performance of batsmen in test cricket," in this paper explained how the performance of batsmen in a series of tests can be predicted using long-distance and long-range methods. In this paper they collect sample data from test cricket batsmen who played during the 2006-2010 season in nine overseas teams and show that these sample data show long and high-level formations of three levels.

A comprehensive review of the literature regarding the performance of both the player and the game reveals following findings. Stretch researched about cricketers' injuries that could affect their performances in the game. In this study, the author applied a hierarchical linear model (HLM) to model the nature of injuries to South African cricketers, including doctors and physiotherapists working with the South African team. Kimber and Hansford proposed a method, which was based on nonparametric approach to assess the batting performance of cricket batsmen. Reaction time is regarded as one of the incalculable talents of cricketers in all the departments of the game. Balasaheb, Maman, and Sandh attempted to find the impact of visual skills training that could affect the performance of batsman. In their research, they showed how the visual skills improve the reaction time, depth perception, and

eye of the cricketers, which eventually improves the batting performance of the player. While discussing about necessary adjusted measures to analyse the player performance in the game of cricket, Lemmerinvestigated the performances of players' when players participate in small number of cricket matches. Christie discussed the physical demands needed for the batting performance of cricketers. In that work, the author pointed out that often cricketers do not pay adequate attention to the physical demand of the game, and suggested the necessity of good training sessions to improve the performance. In addition to these, Christie addressed the psychological and the musculoskeletal demands in cricket, which have an impact on the player performance.

Therefore, in this paper, a three-level HLM is used to model the performance of batsmen in the game of cricket. In this analysis, some anthropometric characteristics such as the height and the handedness of the player are considered. Team-level and cricket-match-level variables such as the rank of the team, the location of the match was played, and the match-series number that the player participated is included into this analysis. One of the main goals of this work is to find answers to the questions of what player-level and team-level characteristics influence the player performance. After identifying the variables that have an impact on the performance, a HLM is formed to predict the performance of batsmen.

## **2.1 Multivariate Data Mining Approach to Predict Match Outcome in One-Day**

### **2.1.1 International Cricket(2016):**

In this paper they have used different mathematical methods of data construction and tried different dividing methods to predict who will win the 50 over match. The winner prediction accuracy was 80%.Prediction model is chosen in order to guess the probability of an outcome on the basis of given input dataset. Statistical analysis is usually performed using univariate and multivariate analysis. Univariate analysis is the first phase in any statistical analysis and is used to determine the direct relationship between individual variables and an outcome. Although univariate tests give a decent indication to the strength and nature of the relationships of interest, they are by no means conclusive. Consequently, analyst switch towards multivariate model to further strengthen and validate results.

In machine learning, classification is the process of identifying the class of a new observation based on a training set containing observations with known classes. Classifier's performance highly depends on the characteristics of the data to be classified. Although there

is no single classifier that works best in every application, determining an appropriate classifier for a given scenario is however still more an art than a science. In spite of the fact that vast verity of classifiers exists in literature, six classifiers were short-listed for performance comparison in this dissertation. Selected classifiers are not only among the most influential data mining classifiers in the research community but also highly diverse in nature.

### **2.1.2 Predicting ODI Cricket Result (2017) :**

Here they predict the results of One Day International matches using the details of ICC match ratings, ICC scorers' scorers, home factor, ICC ratings scores and match results. Logistic Regression was applied to the data and found 74.9% accuracy of the predicted result of the games and in 81% of the games predicted the team that would succeed.

Statistical modelling has been used in sports since decades and has contributed significantly to the success on field. Cricket is one of the most popular sports in the world, second only to soccer. Various natural factors affecting the game, enormous media coverage, and a huge betting market have given strong incentives to model the game from various perspectives. However, the complex rules governing the game, the ability of players and their performances on a given day, and various other natural parameters play an integral role in affecting the final outcome of a cricket match. This presents significant challenges in predicting the accurate results of a game.

To retrieve all the required statistics, the entire dataset has been scraped from the cricinfo website. The dataset includes all the matches played between 2010 and 2014. The dataset contains the basic match details including the two competing teams, the outcome of the toss, the date when it was held, the venue and the winner of the match for all the matches. Along with these, the career statistics of the participating players and their performances in every match is also included. We have restricted our study to only top 9 ODI-playing teams, namely, Australia, South Africa, India, England, Sri Lanka, Pakistan, New Zealand, Bangladesh and West Indies. Since the impact of the nature on the game cannot be foreseen, a total of 109 matches which were either interrupted by rain or ended up in a draw/tie, have been removed from the dataset. Finally, we divided the dataset into two parts, namely, the test data and the training data. The training dataset contains all the matches played during the years 2010 to 2013, and the test dataset contains all the matches played in the year 2014. There is a total of 299 matches in training dataset and 67 matches in test dataset. Learning



**Weights:** To assign the weights to various features in the Algorithms 1 and 2, we have used the 5-match ODI series played between India and Sri Lanka in July, 2012. A series of consecutive matches was deliberately chosen to study the impact of the recent scores of a batsman on his upcoming performances. The estimated scores of the players are compared against their actual performances. After exhaustive experimentation, the final weights are chosen such that the top 6 performing batsmen and bowlers (in terms of runs scored and wickets taken, respectively) from both the teams match with the top 6 batsmen and bowlers estimated by our algorithms. **Binary Classifiers:** Using various binary and numeric features and the outcome of the match as the label, we evaluated a large number of binary classifiers using their scikit-learn implementations to generate supervised classification models, including SVM, Random Forests, Logistic Regression, Decision Trees and kNN. We used the sweep feature to experiment with all the possible values and combinations of the parameters for all the algorithms. The efficacy of the kNN algorithm, with  $k=4$ , was statistically superior to those obtained by the best models of other classifiers, as shown in Figure 2. The idea of using the data of future matches to predict the outcome of past matches is absurd. Consequently, we could not carry out any sort of cross-validation procedure as it would interfere with the chronological order of the data.

## 2.2 Inference from The Literature Survey

The inference derived from related works is that, in the field of predictive analysis, there have been many methods to get the task of prediction done. Also, from very early times, the task of prediction is processed using machine learning algorithms. Different authors propose different machine learning techniques to get this task done until the very concept of neural network was explored in this domain. As proposed by S.Muthuswamy and S.S.Lam, S. R. Iyer and R. Sharda, the authors use various machine learning algorithms for prediction of match outcome, but as proposed by Sasank Viswanadha, Kaustubh Sivalenka, Madan Gopal Jhawar and Vikram Pudi highlights the better accuracy of using random forest classifier for the task.

Predicting outcome of the game has recognized some fundamental problem. In the existing method, by extensive literature survey many research papers have researched on this topic but most of them have used primitive machine learning algorithms like Naïve Bayes and logistic regression. They have taken into consideration factors like teams, toss winners, winners by runs and winners by wickets. We aim to develop an effective machine learning

tool which is needed to predict the outcome of a game, taking humidity and wind speed into consideration. In the current situation, many franchise owners have lost money on the negative prediction results of players.

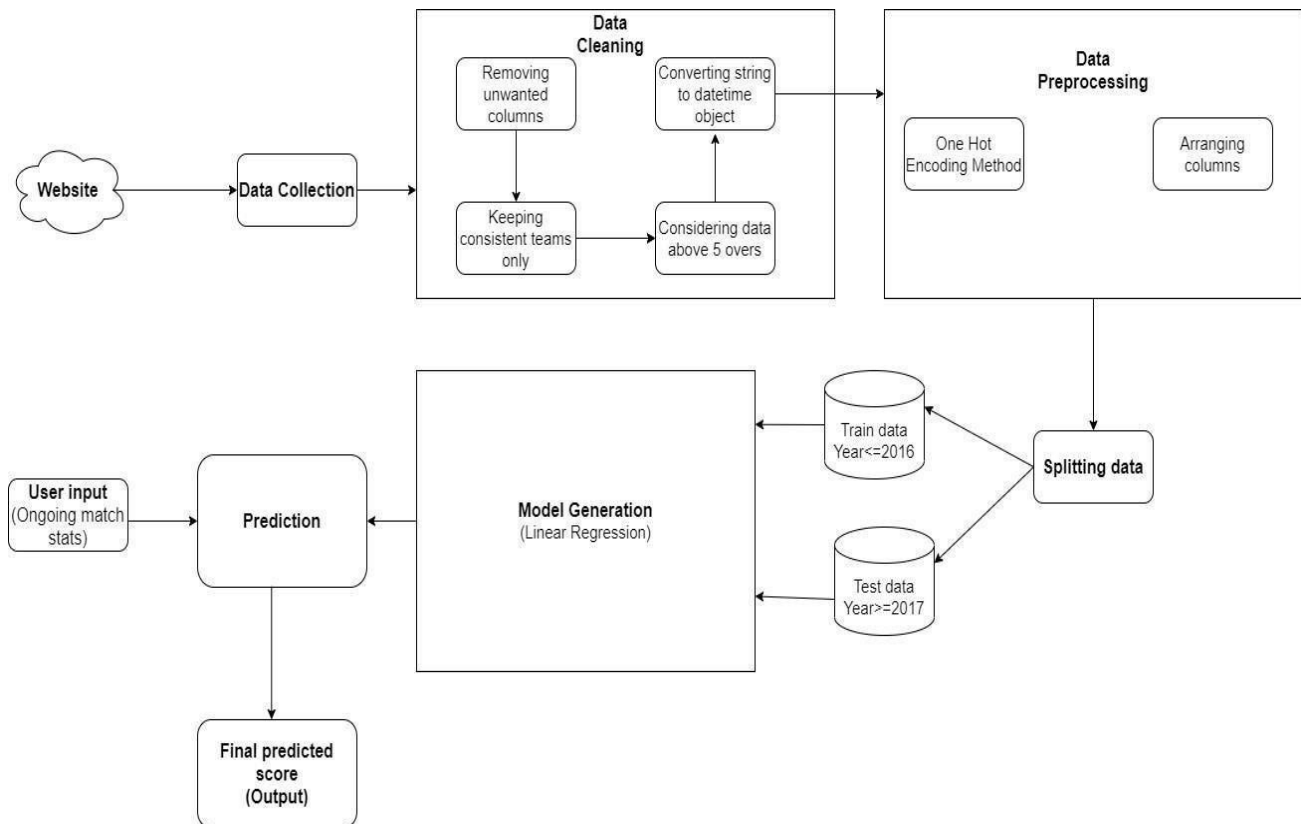
Also, having gone through the works of different authors and researchers, a keen interest in the wide applications of Machine Learning algorithms has been seen. The significant reason for the application of random forest classifier and adaboost, is because these algorithms can take a large amount of input features, train the decision trees and stumps and can give a very accurate prediction.

The old machine learning techniques like Naïve Bayes and Logistic Regression have run out of date and a little change in the application needs rebuilding the whole prediction system, according to the stakeholder's needs. With the advancements of parallel processing and increase in computational capacities the Machine Learning algorithms, tend to outperform the older systems and hence are significant motivation for our work.

### 3. MODEL ARCHITECTURE

We aim to build a model that can predict the first innings score of a live IPL match efficiently. We are looking to build a model that can consider various parameters that contribute to the score prediction.

Below is the model architecture diagram –



**Fig 3.1 : Model Architecture**

The Model Architecture in the fig. 3.1 describes the Machine-Learning phases which comprises of data collection, data cleaning, data preprocessing, model generation & prediction on user inputs.

### 3.1 Data Collection

**Data collection** is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. Data collection is a research component in all study fields, including physical and social sciences, humanities, and business. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same. The goal for all data collection is to capture quality evidence that allows analysis to lead to the formulation of convincing and credible answers to the questions that have been posed. Data collection and validation consists of four steps when it involves taking a census and seven steps when it involves sampling.

Regardless of the field of study or preference for defining data (quantitative or qualitative), accurate data collection is essential to maintain research integrity. The selection of appropriate data collection instruments (existing, modified, or newly developed) and delineated instructions for their correct use reduce the likelihood of errors.

A formal data collection process is necessary as it ensures that the data gathered are both defined and accurate. This way, subsequent decisions based on arguments embodied in the findings are made using valid data. The process provides both a baseline from which to measure and in certain cases an indication of what to improve.

There are 5 common data collection methods:

1. closed-ended surveys and quizzes,
2. open-ended surveys and questionnaires,
3. 1-on-1 interviews,
4. focus groups, and
5. direct observation.

We will be taking the dataset from the datasets available on Kaggle. The dataset will be taken in the CSV format. The data collected from the website will be cleaned in the next step.

### 3.2 Data Cleaning

**Data cleansing** or **data cleaning** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing,

modifying, or deleting the dirty or coarse data. Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

The actual process of data cleansing may involve removing typographical errors or validating and correcting values against a known list of entities. The validation may be strict (such as rejecting any address that does not have a valid postal code), or with fuzzy or approximate string matching (such as correcting records that partially match existing, known records). Some data cleansing solutions will clean data by cross-checking with a validated data set. A common data cleansing practice is data enhancement, where data is made more complete by adding related information. For example, appending addresses with any phone numbers related to that address. Data cleansing may also involve harmonization (or normalization) of data, which is the process of bringing together data of "varying file formats, naming conventions, and columns", and transforming it into one cohesive data set; a simple example is the expansion of abbreviations ("st, rd, etc." to "street, road, etcetera").

High-quality data needs to pass a set of quality criteria. Those include:

**3.2.1 Validity:** The degree to which the measures conform to defined business rules or constraints.

When modern database technology is used to design data-capture systems, validity is fairly easy to ensure: invalid data arises mainly in legacy contexts (where constraints were not implemented in software) or where inappropriate data-capture technology was used (e.g., spreadsheets, where it is very hard to limit what a user chooses to enter into a cell, if cell validation is not used). Data constraints fall into the following categories:

- **Data-Type Constraints**— e.g., values in a particular column must be of a particular data type, e.g., Boolean, numeric (integer or real), date, etc.
- **Range Constraints**: typically, numbers or dates should fall within a certain range. That is, they have minimum and/or maximum permissible values.
- **Mandatory Constraints**: Certain columns cannot be empty.

- **Unique Constraints:** A field, or a combination of fields, must be unique across a dataset. For example, no two persons can have the same social security number.
- **Set-Membership constraints:** The values for a column come from a set of discrete values or codes. For example, a person's sex may be Female, Male or Non-Binary.
- **Foreign-key constraints:** This is the more general case of set membership. The set of values in a column is defined in a column of another table that contains unique values. For example, in a US taxpayer database, the "state" column is required to belong to one of the US's defined states or territories: the set of permissible states/territories is recorded in a separate State table. The term foreign key is borrowed from relational database terminology.
- **Regular expression patterns:** Occasionally, text fields will have to be validated this way. For example, phone numbers may be required to have the pattern (999) 999-9999.
- **Cross-field validation:** Certain conditions that utilize multiple fields must hold. For example, in laboratory medicine, the sum of the components of the differential white blood cell count must be equal to 100 (since they are all percentages). In a hospital database, a patient's date of discharge from the hospital cannot be earlier than the date of admission.

**3.2.2 Accuracy:** The degree of conformity of a measure to a standard or a true value - see also Accuracy and precision. Accuracy is very hard to achieve through data cleansing in the general case because it requires accessing an external source of data that contains the true value: such "gold standard" data is often unavailable. Accuracy has been achieved in some cleansing contexts, notably customer contact data, by using external databases that match up zip codes to geographical locations (city and state) and also help verify that street addresses within these zip codes actually exist.

**3.2.3 Completeness:** The degree to which all required measures are known. Incompleteness is almost impossible to fix with data cleansing methodology: one cannot infer facts that were not captured when the data in question was initially recorded. (In some contexts, e.g., interview data, it may be possible to fix incompleteness by going back to the original source of data, i.e. re-interviewing the subject, but even this does not guarantee success because of problems of recall - e.g., in an interview to gather data on food consumption, no one is likely to remember exactly what one ate six months ago. In the case of systems that insist certain columns should not be empty, one may work around the problem by designating a value that

indicates "unknown" or "missing", but the supplying of default values does not imply that the data has been made complete.)

**3.2.4 Consistency:** The degree to which a set of measures are equivalent in across systems (see also Consistency). Inconsistency occurs when two data items in the data set contradict each other: e.g., a customer is recorded in two different systems as having two different current addresses, and only one of them can be correct. Fixing inconsistency is not always possible: it requires a variety of strategies - e.g., deciding which data were recorded more recently, which data source is likely to be most reliable (the latter knowledge may be specific to a given organization), or simply trying to find the truth by testing both data items (e.g., calling up the customer).

**3.2.5 Uniformity:** The degree to which a set data measures are specified using the same units of measure in all systems (see also Unit of measure). In datasets pooled from different locales, weight may be recorded either in pounds or kilos and must be converted to a single measure using an arithmetic transformation. The term **integrity** encompasses accuracy, consistency and some aspects of validation (see also data integrity) but is rarely used by itself in data-cleansing contexts because it is insufficiently specific. (For example, "referential integrity" is a term used to refer to the enforcement of foreign-key constraints above.)

### **3.3 Process**

**3.3.1 Data auditing:** The data is audited with the use of statistical and database methods to detect anomalies and contradictions: this eventually indicates the characteristics of the anomalies and their locations. Several commercial software packages will let you specify constraints of various kinds (using a grammar that conforms to that of a standard programming language, e.g., JavaScript or Visual Basic) and then generate code that checks the data for violation of these constraints. This process is referred to below in the bullets "workflow specification" and "workflow execution." For users who lack access to high-end cleansing software, Microcomputer database packages such as Microsoft Access or File Maker Pro will also let you perform such checks, on a constraint by constraint basis, interactively with little or no programming required in many cases.

**3.3.2 Workflow specification:** The detection and removal of anomalies are performed by a sequence of operations on the data known as the workflow. It is specified after the process of auditing the data and is crucial in achieving the end product of high-quality data. In order to

achieve a proper workflow, the causes of the anomalies and errors in the data have to be closely considered.

**3.3.3 Workflow execution:** In this stage, the workflow is executed after its specification is complete and its correctness is verified. The implementation of the workflow should be efficient, even on large sets of data, which inevitably poses a trade-off because the execution of a data-cleansing operation can be computationally expensive.

**3.3.4 Post-processing and controlling:** After executing the cleansing workflow, the results are inspected to verify correctness. Data that could not be corrected during the execution of the workflow is manually corrected, if possible. The result is a new cycle in the data-cleansing process where the data is audited again to allow the specification of an additional workflow to further cleanse the data by automatic processing.

Good quality source data has to do with “Data Quality Culture” and must be initiated at the top of the organization. It is not just a matter of implementing strong validation checks on input screens, because almost no matter how strong these checks are, they can often still be circumvented by the users. There is a nine-step guide for organizations that wish to improve data quality:

- Declare a high-level commitment to a data quality culture
- Drive process reengineering at the executive level
- Spend money to improve the data entry environment
- Spend money to improve application integration
- Spend money to change how processes work
- Promote end-to-end team awareness
- Promote interdepartmental cooperation
- Publicly celebrate data quality excellence
- Continuously measure and improve data quality

### **3.4 Others include**

**3.4.1 Parsing:** for the detection of syntax errors. A parser decides whether a string of data is acceptable within the allowed data specification. This is similar to the way a parser works with grammars and languages.

**3.4.2 Data transformation:** Data transformation allows the mapping of the data from its given format into the format expected by the appropriate application. This includes value



conversions or translation functions, as well as normalizing numeric values to conform to minimum and maximum values.

**3.4.3 Duplicate elimination:** Duplicate detection requires an algorithm for determining whether data contains duplicate representations of the same entity. Usually, data is sorted by a key that would bring duplicate entries closer together for faster identification.

**3.4.4 Statistical methods:** By analysing the data using the values of mean, standard deviation, range, or clustering algorithms, it is possible for an expert to find values that are unexpected and thus erroneous. Although the correction of such data is difficult since the true value is not known, it can be resolved by setting the values to an average or other statistical value. Statistical methods can also be used to handle missing values which can be replaced by one or more plausible values, which are usually obtained by extensive data augmentation algorithms. In the data cleaning step, we want to remove unwanted columns like match id, venue, batsman name, bowler name, a score of the striker, and score of the non-striker. These columns will not be required during prediction hence we will be dropping those columns. In the IPL dataset, some teams are not playing in the IPL anymore. Teams like Deccan Chargers, Kochi Tuskers Kerala, Pune Warriors India, Gujarat Lions, Rising Pune Supergiant, etc. are not part of IPL. So, we need to eliminate those teams from the dataset and we only need to consider the consistent teams. We will be considering the data after 5 overs. The date column in the dataset is present in the string format but we want to apply some operations on the date column for that we will need to convert the string to a date-time object.

### 3.5 Data Pre-processing

**Data pre-processing** refer to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the datamining process. The phrase "garbagein,garbageout" is particularly applicable to datamining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out of range values (e.g., Income: -100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), and missing values, etc. Analysing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis. Often, data pre-processing is the most important phase of a machine learning project, especially in computational biology.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Examples of data pre-processing include cleaning, instance selection, normalization, one hot encoding, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set.

Data pre-processing may affect the way in which outcomes of the final data processing can be interpreted. This aspect should be carefully considered when interpretation of the results is a key point, such in the multivariate processing of chemical data (chemometrics). After cleaning the data, we will need our data to be pre-processed. In the data pre-processing step, we will be performing one-hot encoding. One hot encoding is explained in detail in the implementation section. We will need to rearrange the columns of our dataset in the data pre-processing step. The purpose of rearranging columns is that we need our columns to be properly arranged in some sequence.

### **3.6 Data Splitting**

The data should ideally be divided into 3 sets – namely, train, test, and holdout cross validation or development (dev) set.

#### **3.6.1 Train Set**

The train set would contain the data which will be fed into the model. In simple terms, our model would learn from this data. For instance, a Regression model would use the examples in this data to find gradients in order to reduce the cost function. Then these gradients will be used to reduce the cost and predict data effectively.

#### **3.6.2 Dev Set**

The development set is used to validate the trained model. This is the most important setting as it will form the basis of our model evaluation. If the difference between error on the training set and error on the dev set is huge, it means the model has high variance and hence, a case of over-fitting.

#### **3.6.3 Test Set**

The test set contains the data on which we test the trained and validated model. It tells us how efficient our overall model is and how likely is it going to predict something which does not make sense. There is a plethora of evaluation metrics (like precision, recall,

accuracy, etc.) which can be used to measure the performance of our model. The test set can be sometimes omitted too. It is meant to get an unbiased estimate of algorithms performance in the real world. People who divide their dataset into just two parts usually call their Dev set the Test set. We try to build a model upon training set then try to optimize hyperparameters on the dev set as much as possible then after our model is ready, we try and evaluate the testing set.

### 3.6.4 Cross-Validation Set

We select the appropriate model or the degree of the polynomial (if using regression model only) by minimizing the error on the cross-validation set.

### 3.6.5 How to decide the ratio of splitting the dataset?

The answer generally lies in the dataset itself. The proportions are decided according to the size and type (for time series data, splitting techniques are a bit different) of data available with us. If the size of our dataset is between 100 to 10,00,000, then we split it in the ratio 60:20:20. That is 60% data will go to the Training Set, 20% to the Dev Set and remaining to the Test Set. If the size of the data set is greater than 1 million then we can split it in something like this 98:1:1 or 99:0.5:0.5

The main aim of deciding the splitting ratio is that all three sets should have the general trend of our original dataset. If our dev set has very little data, then it is possible that we'll end up selecting some model which is biased towards the trends only present in the dev set. Same is the case with training sets — too little data will bias the model towards some trends found only in that subset of the dataset. The models that we deploy are nothing but estimators learning the statistical trends in the data. Therefore, it is important that the data that is being used to learn and that being used to validate or test the model follow as similar statistical distribution as possible. One of the ways to achieve this as perfectly as possible is to select the subsets — here the training set, the dev set and/or the test set — randomly. For example, suppose that you are working on a face detection project and face training pictures are taken from the web and the dev/test pictures are from user's cell phone, then there will be a mismatch between the properties of train set and dev/test set.

One way we can divide the dataset into the train, test, cv with 0.6, 0.2, 0.2 ratios would be to use the `train_test_split` method twice: `from sklearn.model_selection import train_test_split, x_test, y, y_test = train_test_split (x_train, labels, test_size=0.2,`

```
train_size=0.8 ) x_train, x_cv, y_train, y_cv = train_test_split(x,y,test_size = 0.25, train_size
=0.75)
```

After data pre-processing, we will be splitting our data in such a way that IPL matches played before 2016 will be considered for the training of the model and IPL matches played after 2016 will be considered for test data.

### 3.7 Types of Machine-Learning Models

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

#### 3.7.1 Supervised learning

A support-vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white.

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Types of supervised learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

### **3.7.2 Unsupervised learning**

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labelled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

### **3.7.3 Cluster Analysis:**

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

### **3.7.4 Semi-supervised learning**

Semi-supervised learning falls between unsupervised learning (without any labelled training data) and supervised learning (with completely labelled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabelled data, when used in conjunction with a small amount of labelled data, can produce a considerable improvement in learning accuracy. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

### **3.7.5 Reinforcement learning**

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques.[38] Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

### **3.7.6 Dimensionality reduction**

Dimensionality reduction is a process of reducing the number of random variables under consideration by obtaining a set of principal variables.[39] In other words, it is a process of reducing the dimension of the feature set, also called "number of features". Most of the dimensionality reduction techniques can be considered as either feature elimination or extraction. One of the popular methods of dimensionality reduction is principal component analysis (PCA). PCA involves changing higher-dimensional data (e.g., 3D) to a smaller space (e.g., 2D). This results in a smaller dimension of data (2D instead of 3D), while keeping all original variables in the model without changing the data.[40] The manifold hypothesis proposes that high-dimensional data sets lie along low-dimensional manifolds, and many dimensionality reduction techniques make this assumption, leading to the area of manifold learning and manifold regularization.

## **3.8 Types of Models that will be used in Project**

### **3.8.1 Linear Regression for Machine Learning**

Linear regression is perhaps one of the most well-known and well understood algorithms in statistics and machine learning.

- We will understand that -
- Why linear regression belongs to both statistics and machine learning.

- The many names by which linear regression is known.
- The representation and learning algorithms used to create a linear regression model.
- How to best prepare your data when modelling using linear regression.

You do not need to know any statistics or linear algebra to understand linear regression. This is a gentle high-level introduction to the technique to give you enough background to be able to use it effectively on your own problems.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables ( $x$ ) and the single output variable ( $y$ ). More specifically, that  $y$  can be calculated from a linear combination of the input variables ( $x$ ).

When there is a single input variable ( $x$ ), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression. The representation is a linear equation that combines a specific set of input values ( $x$ ) the solution to which is the predicted output for that set of input values ( $y$ ). As such, both the input values ( $x$ ) and the output value are numeric.

The linear equation assigns one scale factor to each input value or column, called a coefficient and represented by the capital Greek letter Beta ( $B$ ). One additional coefficient is also added, giving the line an additional degree of freedom (e.g. moving up and down on a two-dimensional plot) and is often called the intercept or the bias coefficient.

For example, in a simple regression problem (a single  $x$  and a single  $y$ ), the form of the model would be:

$$y = B_0 + B_1 * x$$

In higher dimensions when we have more than one input ( $x$ ), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g.  $B_0$  and  $B_1$  in the above example). It is common to talk about the complexity of a regression model like linear regression. This refers to the number of coefficients used in the model.

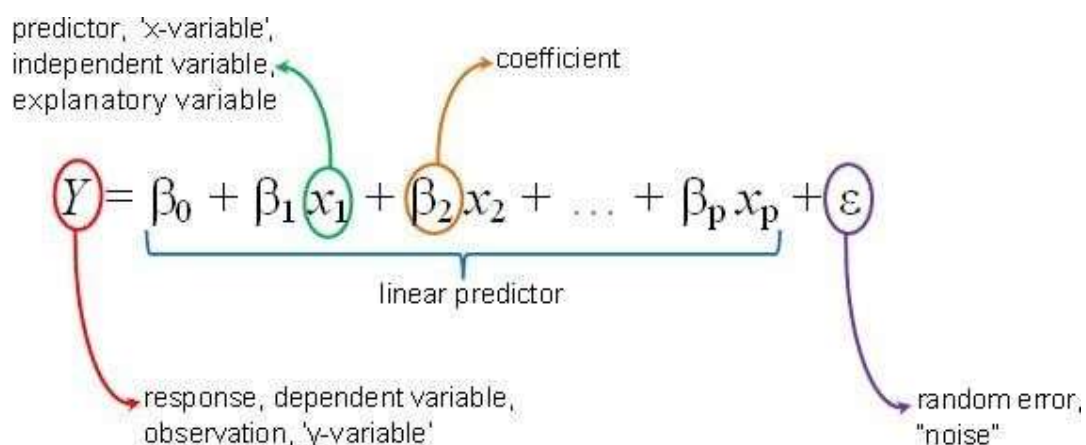
When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model ( $0 * x = 0$ ). This becomes relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available.

In this section we will take a brief look at four techniques to prepare a linear regression model. This is not enough information to implement them from scratch, but enough to get a flavour of the computation and trade-offs involved.

There are many more techniques because the model is so well studied. Take note of Ordinary Least Squares because it is the most common method used in general. Also take note of Gradient Descent as it is the most common technique taught in machine learning classes. With simple linear regression when we have a single input, we can use statistics to estimate the coefficients. This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics.

### Linear Regression Mathematical Model –



From the above figure we can conclude that  $y$  is the response variable,  $x$  is the predictor variable,  $\beta$  is the coefficient.

### 3.8.2 Ordinary Least Squares

When we have more than one input, we can use Ordinary Least Squares to estimate the values of the coefficients.



The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seek to minimize.

This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations.

It is unusual to implement the Ordinary Least Squares procedure yourself unless as an exercise in linear algebra. It is more likely that you will call a procedure in a linear algebra library. This procedure is very fast to calculate.

### **3.8.3 Gradient Descent**

When there are one or more inputs you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data.

This operation is called Gradient Descent and works by starting with random values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. A learning rate is used as a scale factor and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

When using this method, you must select a learning rate (alpha) parameter that determines the size of the improvement step to take on each iteration of the procedure.

Gradient descent is often taught using a linear regression model because it is relatively straightforward to understand. In practice, it is useful when you have a very large dataset either in the number of rows or the number of columns that may not fit into memory.

### **3.8.4 Regularization**

There are extensions of the training of the linear model called regularization methods. These seek to both minimize the sum of the squared error of the model on the training data (using ordinary least squares) but also to reduce the complexity of the model (like the number or absolute size of the sum of all coefficients in the model).

**Two popular examples of regularization procedures for linear regression are:**

*Lasso Regression:* where Ordinary Least Squares is modified to also minimize the absolute sum of the coefficients (called L1 regularization).

*Ridge Regression:* where Ordinary Least Squares is modified to also minimize the squared absolute sum of the coefficients (called L2 regularization).

These methods are effective to use when there is collinearity in your input values and ordinary least squares would overfit the training data.

Now that you know some techniques to learn the coefficients in a linear regression model, let's look at how we can use a model to make predictions on new data.

### 3.8.5 Making Predictions with Linear Regression

Equations for Linear Regression & Cost Function

$$\hat{y} = w[0] \times x[0] + w[1] \times x[1] + \dots + w[n] \times x[n] + b$$

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left( y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2$$

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs.

Let's make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

$$y = B_0 + B_1 \times x_1 \quad \text{or}$$

$$\text{weight} = B_0 + B_1 \times \text{height}$$

Where  $B_0$  is the bias coefficient and  $B_1$  is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

For example, let's use  $B_0 = 0.1$  and  $B_1 = 0.5$ . Let's plug them in and calculate the weight (in kilograms) for a person with the height of 182 centimetres.

$$\text{weight} = 0.1 + 0.5 * 182$$

$$\text{weight} = 91.1$$

You can see that the above equation could be plotted as a line in two-dimensions. The  $B_0$  is our starting point regardless of what height we have. We can run through a bunch of heights from 100 to 250 centimetres and plug them to the equation and get weight values, creating our line.

### **Sample Height vs Weight Linear Regression Sample Heights Weight Linear**

**Regression :** Now that we know how to make predictions given a learned linear regression model, let's look at some rules of thumb for preparing our data to make the most of this type of model. As such, there is a lot of sophistication when talking about these requirements and expectations which can be intimidating. In practice, you can use these rules more as rules of thumb when using Ordinary Least Squares Regression, the most common implementation of linear regression. Try different preparations of your data using these heuristics and see what works best for your problem.

**Linear Assumption.** Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).

**Remove Noise.** Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.

**Remove Collinearity.** Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.

**Gaussian Distributions.** Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on your variables to make their distribution more Gaussian looking.

Rescale Inputs: Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

### 3.8.6 Random Forest Regressor:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning. You can see how a random forest would look like with two trees:

**Two tree random forest** :Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the

best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

### **3.8.7 Lasso Regression:**

In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. It was originally introduced in geophysics.

Lasso was originally formulated for linear regression models. This simple case reveals a substantial amount about the estimator. These include its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates do not need to be unique if covariates are collinear.

Though originally defined for linear regression, lasso regularization is easily extended to other statistical models including generalized linear models, generalized estimating equations, proportional hazards models, and M-estimators. Lasso's ability to perform subset selection relies on the form of the constraint and has a variety of interpretations including in terms of geometry, Bayesian statistics and convex analysis.

Lasso Regression is a popular type of regularized linear regression that includes an L1 penalty. This has the effect of shrinking the coefficients for those input variables that do not contribute much to the prediction task. This penalty allows some coefficient values to go to the value of zero, allowing input variables to be effectively removed from the model, providing a type of automatic feature selection.

**Lasso Regression refers to a model that assumes a linear relationship between input variables and the target variable.**

Ridge Regression Cost Function-

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left( y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p w_j^2$$

With a single input variable, this relationship is a line, and with higher dimensions, this relationship can be thought of as a hyperplane that connects the input variables to the target variable. The coefficients of the model are found via an optimization process that seeks to minimize the sum squared error between the predictions (yhat) and the expected target values (y).

$$\text{loss} = \sum_{i=0}^n (y_i - \text{yhat}_i)^2$$

A problem with linear regression is that estimated coefficients of the model can become large, making the model sensitive to inputs and possibly unstable. This is particularly true for problems with few observations (samples) or less samples (n) than input predictors (p) or variables (so-called  $p \gg n$  problems).

One approach to address the stability of regression models is to change the loss function to include additional costs for a model that has large coefficients. Linear regression models that use these modified loss functions during training are referred to collectively as penalized linear regression.

A popular penalty is to penalize a model based on the sum of the absolute coefficient values. This is called the L1 penalty. An L1 penalty minimizes the size of all coefficients and allows some coefficients to be minimized to the value zero, which removes the predictor from the model.

$$\text{l1\_penalty} = \sum_{j=0}^p \text{abs}(\text{beta}_j)$$

An L1 penalty minimizes the size of all coefficients and allows any coefficient to go to the value of zero, effectively removing input features from the model.

This acts as a type of automatic feature selection.

... a consequence of penalizing the absolute values is that some parameters are actually set to 0 for some value of lambda. Thus the lasso yields models that simultaneously use regularization to improve the model and to conduct feature selection.

This penalty can be added to the cost function for linear regression and is referred to as Least Absolute Shrinkage And Selection Operator regularization (LASSO), or more commonly, “Lasso” (with title case) for short.

A popular alternative to ridge regression is the least absolute shrinkage and selection operator model, frequently called the lasso.

A hyperparameter is used called “lambda” that controls the weighting of the penalty to the loss function. A default value of 1.0 will give full weightings to the penalty; a value of 0 excludes the penalty. Very small values of lambda, such as 1e-3 or smaller, are common.

$\text{lasso\_loss} = \text{loss} + (\text{lambda} * \text{ll\_penalty})$

### 3.8.8 Support Vector Regressor:

In machine learning, Support Vector Machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. In Support Vector Regression, the straight line that is required to fit the data is referred to as **hyperplane**.

The objective of a support vector machine algorithm is to find a hyperplane in an n-dimensional space that distinctly classifies the data points. The data points on either side of the hyperplane that are closest to the hyperplane are called **Support Vectors**. These influence the position and orientation of the hyperplane and thus help build the SVM.

**Hyperparameters in SVR** - Now that we have an intuition of what a support vector machine is, we will take look into the various hyperparameters that are used in Support Vector Regression. Some of the key parameters used are as mentioned below:

- **Hyperplane:** Hyperplanes are decision boundaries that is used to predict the continuous output. The data points on either side of the hyperplane that are closest to the hyperplane are called Support Vectors. These are used to plot the required line that shows the predicted output of the algorithm.

- **Kernel:** A kernel is a set of mathematical functions that takes data as input and transform it into the required form. These are generally used for finding a hyperplane in the higher dimensional space.

### 3.9 Model Generation

The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learn from. The term ML model refers to the model artifact that is created by the training process.

The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

You can use the ML model to get predictions on new data for which you do not know the target. For example, let's say that you want to train an ML model to predict if an email is spam or not spam. You would provide Amazon ML with training data that contains emails for which you know the target (that is, a label that tells whether an email is spam or not spam). Amazon ML would train an ML model by using this data, resulting in a model that attempts to predict whether new email will be spam or not spam.

To train an ML model, you need to specify the following:

- Input training data source
- Name of the data attribute that contains the target to be predicted
- Required data transformation instructions
- Training parameters to control the learning algorithm

Typically, machine learning algorithms accept parameters that can be used to control certain properties of the training process and of the resulting ML model. In Amazon Machine Learning, these are called training parameters. You can set these parameters using the Amazon ML console, API, or command line interface (CLI). If you do not set any parameters, Amazon ML will use default values that are known to work well for a large range of machine learning tasks. You can specify values for the following training parameters:

- Maximum model size



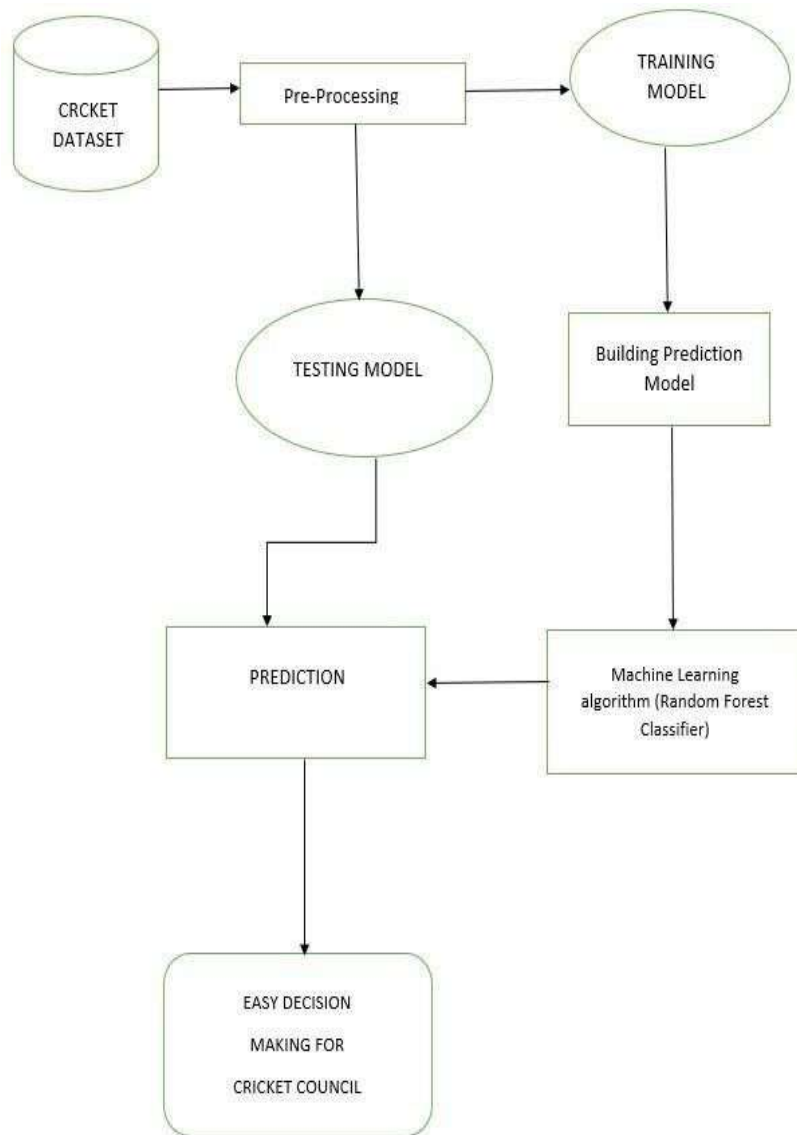
- Maximum number of passes over training data
- Shuffle type
- Regularization type
- Regularization amount

### **3.10 Final Prediction**

Finally, the data will be passed through the model and then the user inputs will be taken. After getting the user inputs and matching them with the historical data we will be predicting a range of the score i.e. from lower bound to the upper bound.

## 4. MODEL DIAGRAMS

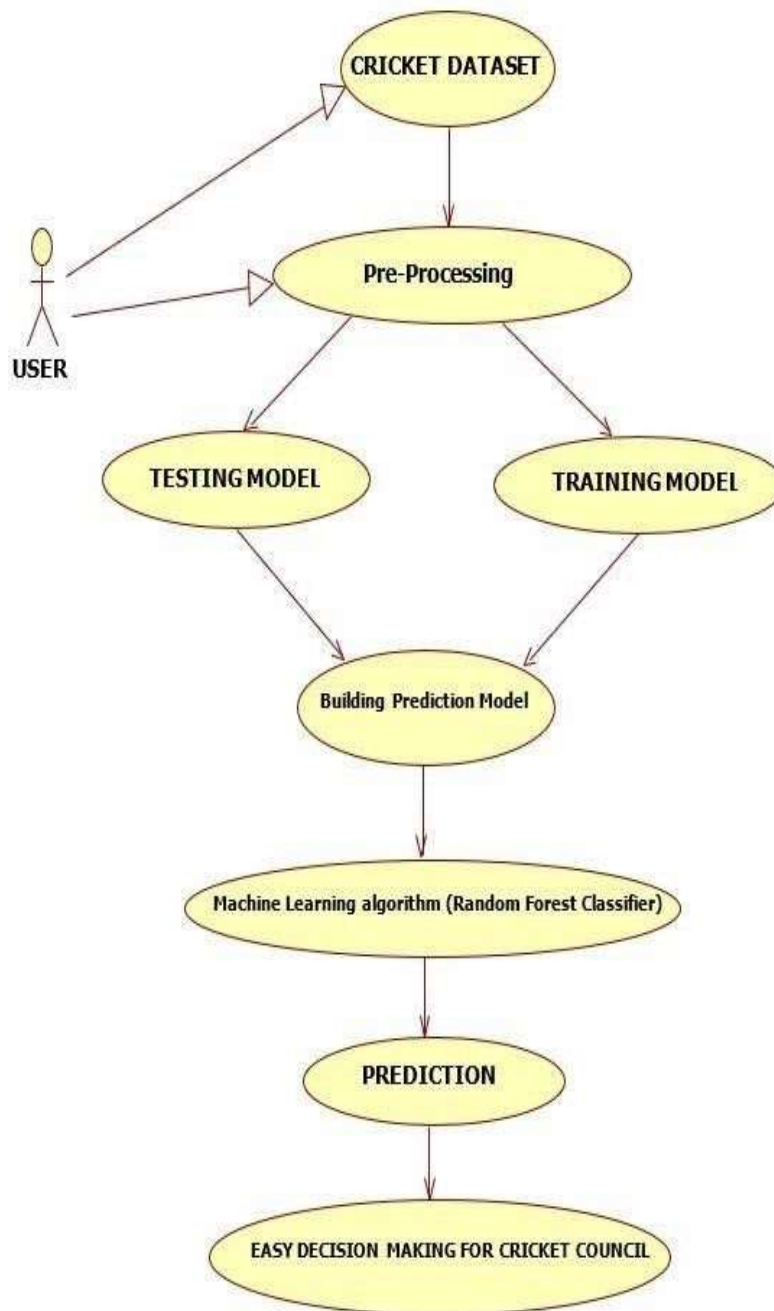
### 4.1 System Design



**Fig 4.1 : System Design**

Above System Design Diagram shows us how streamline of the data & how it can be used for easy decision making for the cricket council. It shows flow of the system & components in the system.

## 4.2 Use - Case UML Diagram



**Fig4.2 : Use Case UML Diagram**

Above UML Diagrams shows us how ML Model is chosen & it is saved. Later it is used by the flask app where the end user inputs his parameters & App predicts based on his parameters.

### 4.3 UML Diagram

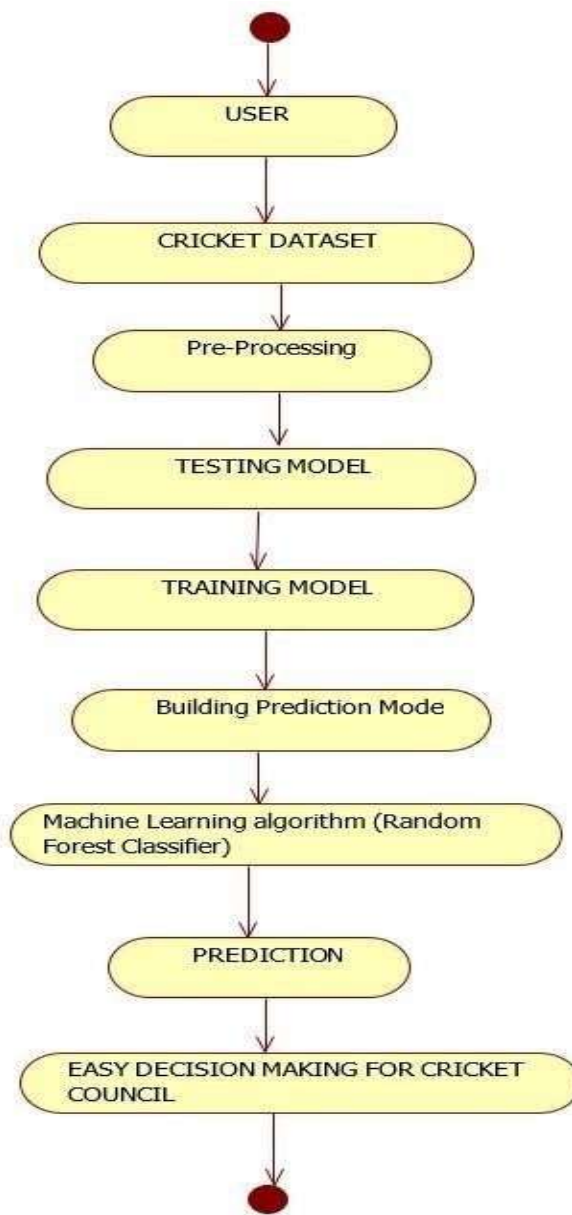


Fig 1.3 : UML Diagram

Above Diagram represents us the following:

- **User :** User will input the features
- **Pre-Processing :** Pre-Processing part of the dataset.
- **Training & Testing :** It will depict training & testing
- **ML Model:** Building an ML Model.
- **Flask App:** End Product of Cricket Score Prediction.

## 4.4 Deployment Diagram

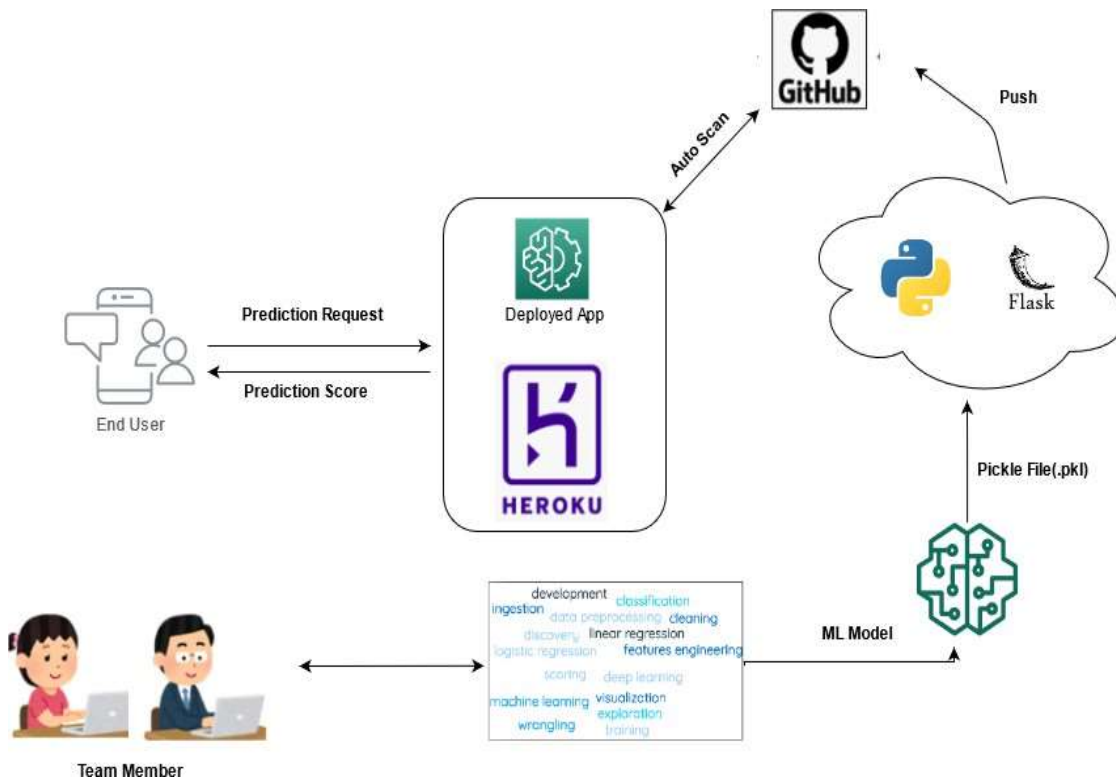
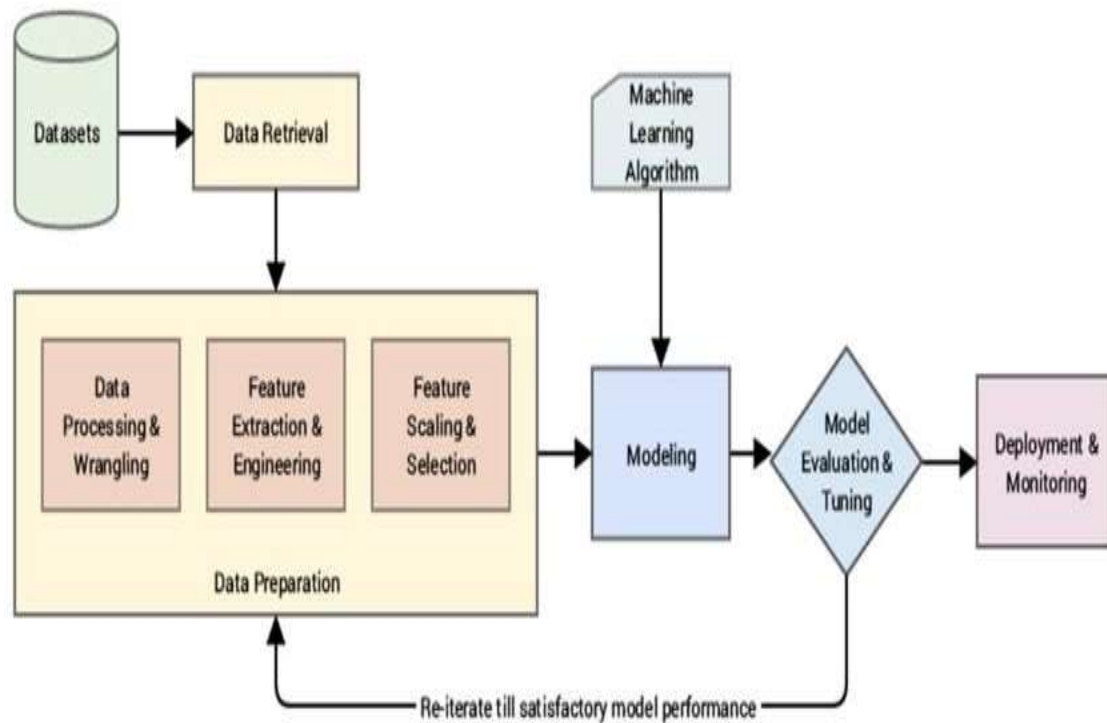


Fig 4.4 :Deployment Diagram

Above Diagram represents us that how a client will interact with our cricket dataset.

- **templates** :- This folder contains the html files (index.html, predict.html) that would be used by our main file (*app.py*) to generate the front end of our application
- **app.py** :- This is the main application file, where all our code resides and it binds everything together.
- **requirements.txt** :- This file contains all the dependencies/libraries that would be used in the project (whenever a virtual environment is created it can use this requirements file directly to download all the dependencies you need not to install all the libraries manually, you just need to put all of them in this file)
- **model.pkl** :- This is our classification model, that we would be using, in this case it is a Logistic Regression Model, which I had trained already.
- **vectorizer.pkl** :- This is vectorizer file that is used to convert text into a vector for the model to process, in this case we have used a tf-idf vectorizer
- **Procfile** :- This is a special file that would be required when we would be deploying the application in a public server (heroku)

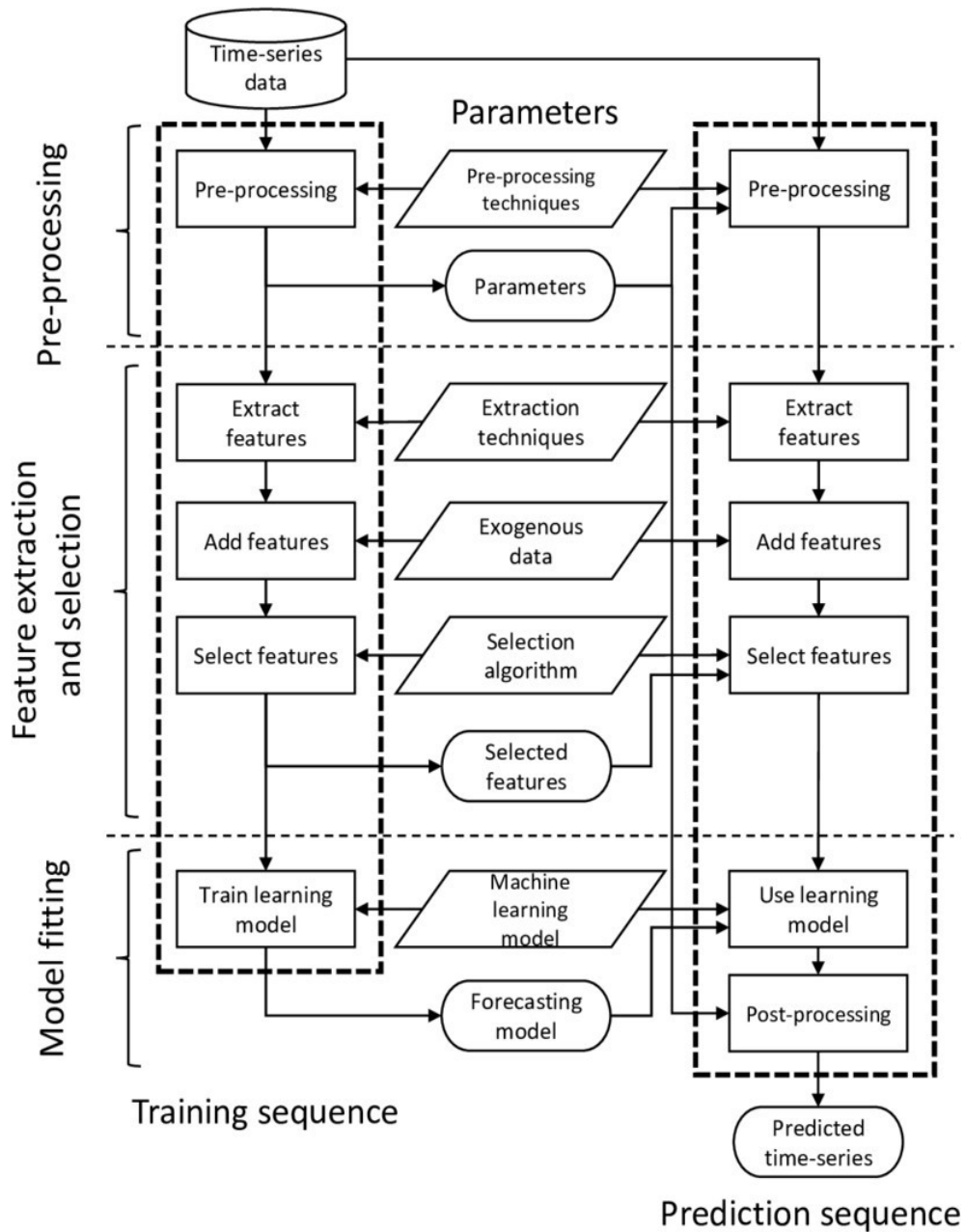
## 4.5 Activity Diagram



**Figure 4.5 : Activity Diagram**

Above Activity Diagram (Fig 4.5) depicts which phases are implemented during a machine learning project. The typical phases include data collection, data pre-processing, building datasets, model training and refinement, evaluation, and deployment to production. You can automate some aspects of the machine learning operations workflow, such as model and feature selection phases, but not all.

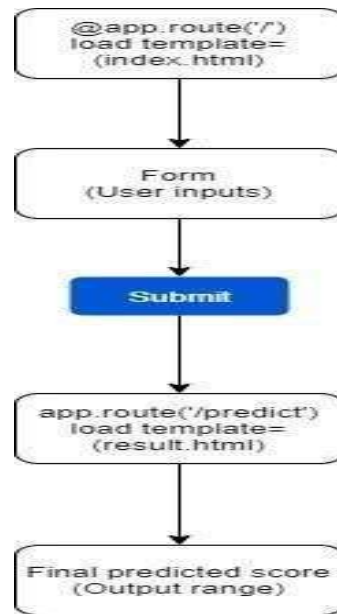
#### 4.6 DFD Diagram



**Figure 4.6 : DFD Diagram**

Above data flow diagram (DFD) maps out the flow of ML Model for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

## 5. UI FLOW



**Figure 5.1 : UI Flow Diagram**

Above UI Flow Diagram depicts how App.route method routes to the index.html, Form inputs user features, Submit button redirects to the predict web page, Final Score is predicted on the predicted page.

### 5.1 Flask Framework

A microframework is a term used to refer to minimalistic web application frameworks. It is contrasted with full-stack frameworks.

It lacks most of the functionality which is common to expect in a full-fledged web application framework, such as:

Accounts, authentication, authorization, roles .

Database abstraction via an object-relational mapping .

Input validation and input sanitation .

Web template engine



Typically, a microframework facilitates receiving an HTTP request, routing the HTTP request to the appropriate controller, dispatching the controller, and returning an HTTP response. Microframeworks are often specifically designed for building the APIs for another service or application. For example, Lumen microframework is designed for Microservices development and API development.

What does “micro” mean? “Micro” does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The “micro” in microframework means Flask aims to keep the core simple but extensible. Flask won’t make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don’t.

### **5.1.1 Configuration and Conventions**

Flask has many configuration values, with sensible defaults, and a few conventions when getting started. By convention, templates and static files are stored in subdirectories within the application’s Python source tree, with the names templates and static respectively. While this can be changed, you usually don’t have to, especially when getting started.

### **5.1.2 Growing with Flask**

Once you have Flask up and running, you’ll find a variety of extensions available in the community to integrate your project for production. As your codebase grows, you are free to make the design decisions appropriate for your project. Flask will continue to provide a very simple glue layer to the best that Python has to offer. You can implement advanced patterns in SQLAlchemy or another database tool, introduce non-relational data persistence as appropriate, and take advantage of frameworkagnostic tools built for WSGI, the Python web interface.

Flask includes many hooks to customize its behavior. Should you need more customization, the Flask class is built for subclassing. If you are interested in that, check out the Becoming Big chapter. If you are curious about the Flask design principles, head over to the section about Design Decisions in Flask.

By default, Flask does not include a database abstraction layer, form validation or anything else where different libraries already exist that can handle that. Instead, Flask

supports extensions to add such functionality to your application as if it was implemented in Flask itself. Numerous extensions provide database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be “micro”, but it’s ready for production use on a variety of needs.

### 5.1.3 A Minimal Application

A minimal Flask application looks something like this:

```
from flask import Flask

app = Flask(__name__)

@app.route("/") def
hello_world():
    return "<p>Hello, World!</p>" So
```

## 5.2 What did that code do?

First we imported the Flask class. An instance of this class will be our WSGI application.

Next we create an instance of this class. The first argument is the name of the application’s module or package. `__name__` is a convenient shortcut for this that is appropriate for most cases. This is needed so that Flask knows where to look for resources such as templates and static files.

We then use the `route()` decorator to tell Flask what URL should trigger our function.

The function returns the message we want to display in the user’s browser. The default content type is HTML, so HTML in the string will be rendered by the browser.

We will be using the Flask framework for the development of this project. In the Flask framework, the routing of the pages is done based on the URLs. If our browser finds the `'/'` in the URL then we will be routing the user to the home page. On the home page, we will be having our main form. In that form, we will be taking the user inputs. The user inputs include, Name of the batting team, name of the bowling team, number of runs scored, number of overs bowled, number of wickets taken, number of runs scored in the previous 5 overs, and number of wickets taken in the previous 5 overs. Once the user hits predict score button then the form is submitted. After submission of the form, our model comes into the

picture in the backend. The inputs are compared with the historical data and then a score is predicted. After the submission of the form, the user is redirected to the '/result' URL i.e. the user is redirected to the result page where the user can see the actual predicted score. On the prediction page, the user will be getting the output in the form of a range i.e. from the lower bound to the upper bound.

## 6. RESULTS / OUTPUT



The screenshot shows the 'Indian Premier League (IPL)' score prediction interface. It features a central form with several input fields and two team logos on the sides. The background is a cricket stadium at night.

**Indian Premier League (IPL)**

Left side logo: CHENNAI SUPER KINGS

Right side logo: MUMBAI INDIANS

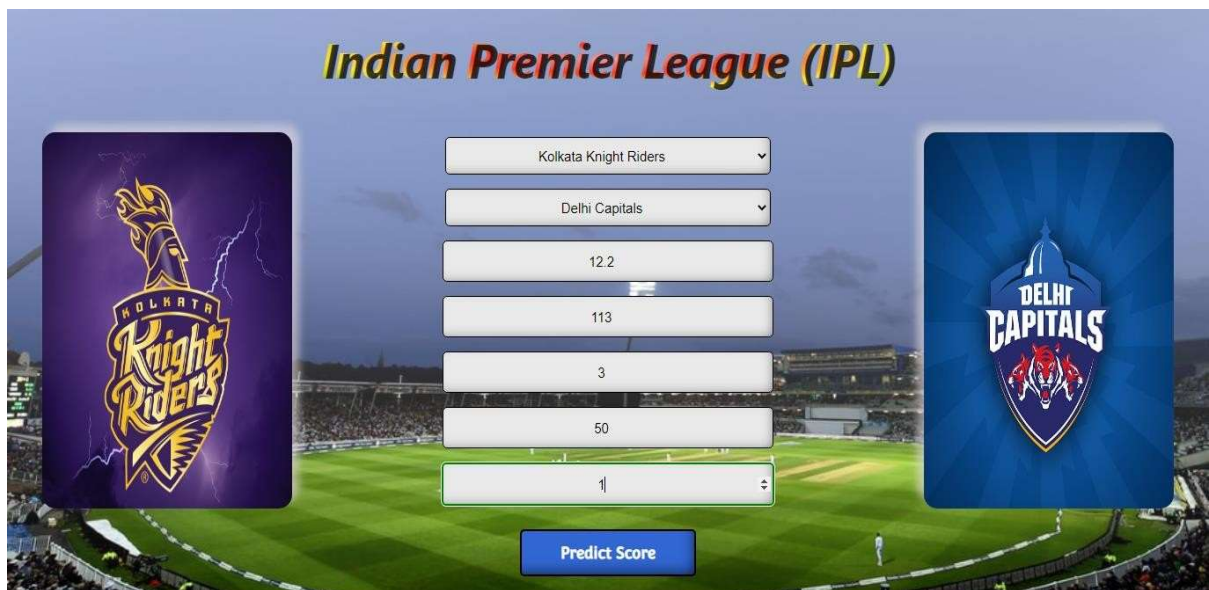
Form fields (from top to bottom):

- Select a Batting team --- (dropdown)
- Select a Bowling team --- (dropdown)
- Overs ( $\geq 5.0$ ) ( $< 19.4$ ) (eg: 7.2)
- Runs (eg: 64)
- Wickets (eg: 4)
- Runs scored in previous 5 Overs (eg: 42)
- Wickets taken in previous 5 Overs (eg: 3)

Predict Score button

**Fig 6.1 : Final UI after deployment of our model.**

Above image depicts the final UI for our cricket score prediction model. The model would run on the local host in our system and according to the inputs provided it would process the algorithm in the Backend.



The screenshot shows the 'Indian Premier League (IPL)' score prediction interface with manual input. It features a central form with several input fields and two team logos on the sides. The background is a cricket stadium at night.

**Indian Premier League (IPL)**

Left side logo: KOLKATA Knight Riders

Right side logo: DELHI CAPITALS

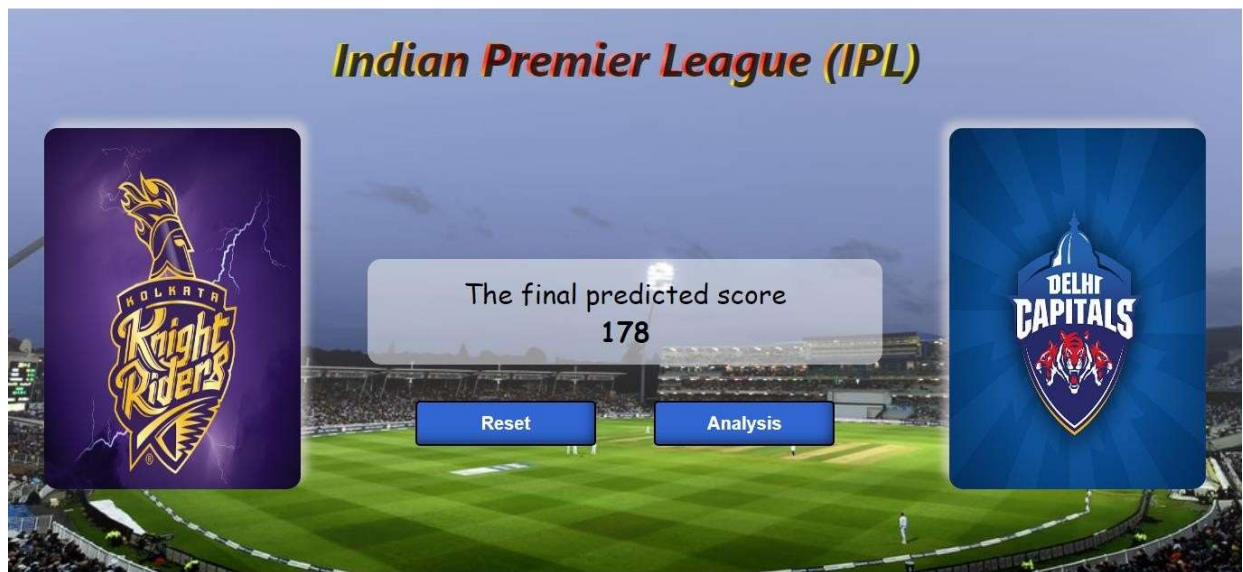
Form fields (from top to bottom):

- Kolkata Knight Riders (dropdown)
- Delhi Capitals (dropdown)
- 12.2
- 113
- 3
- 50
- 1 (spin button)

Predict Score button

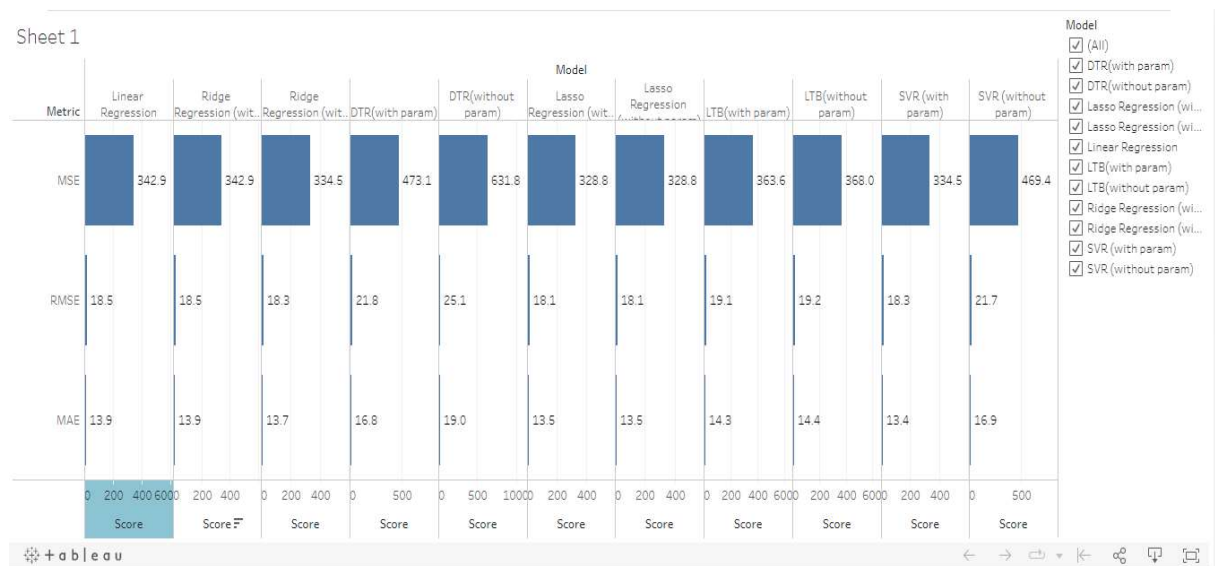
**Fig 6.2 : Manually putting the data in each of the fields mentioned for predicting the score.**

Above image depicts how we will give the input to our cricket score prediction model.



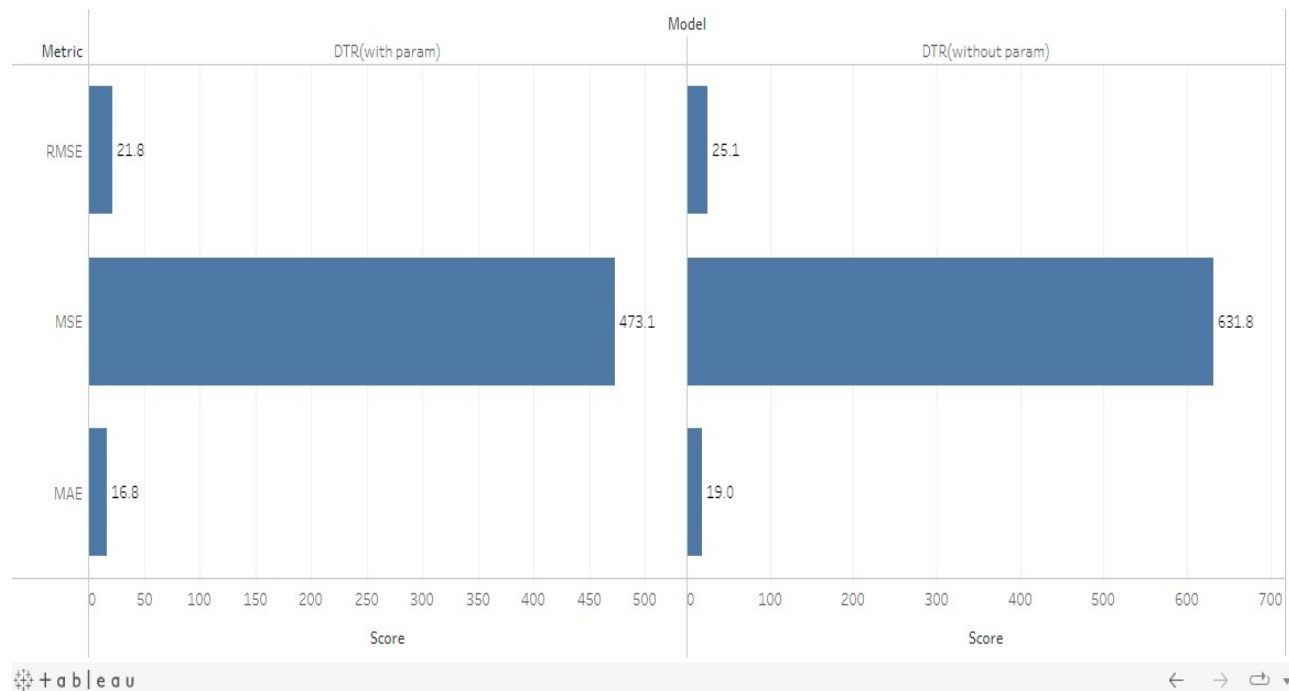
**Fig 6.3 : Final Predicted Score of the match of the input values.**

Above model depicts how our app will show the predicted score. This score has been calculated using the inputs shown/given in the previous image(fig 6.2). After the substitution, the model will process the algorithms in the backend and will provide with the most accurate score calculated by it.

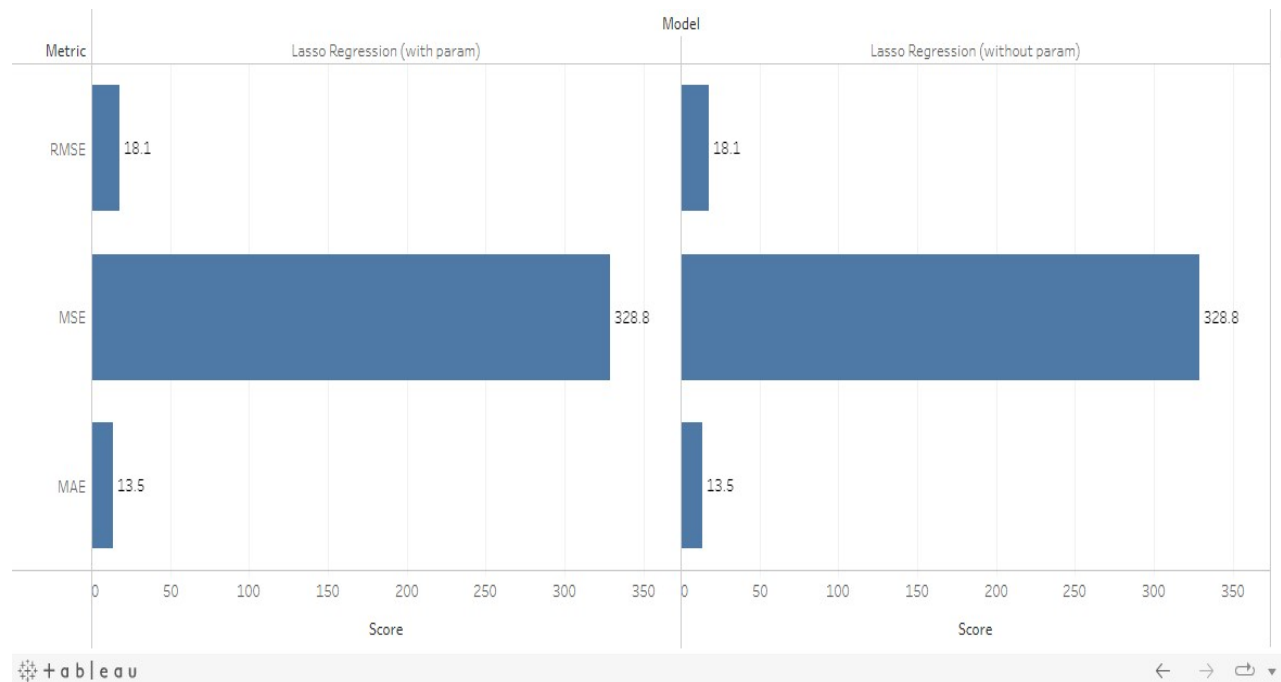


**Fig6.4 :Accuracy chart of our model by using different ML Algorithms – The Score comparisons before Hyperparameter Tuning and after Hyperparameter Tuning.**

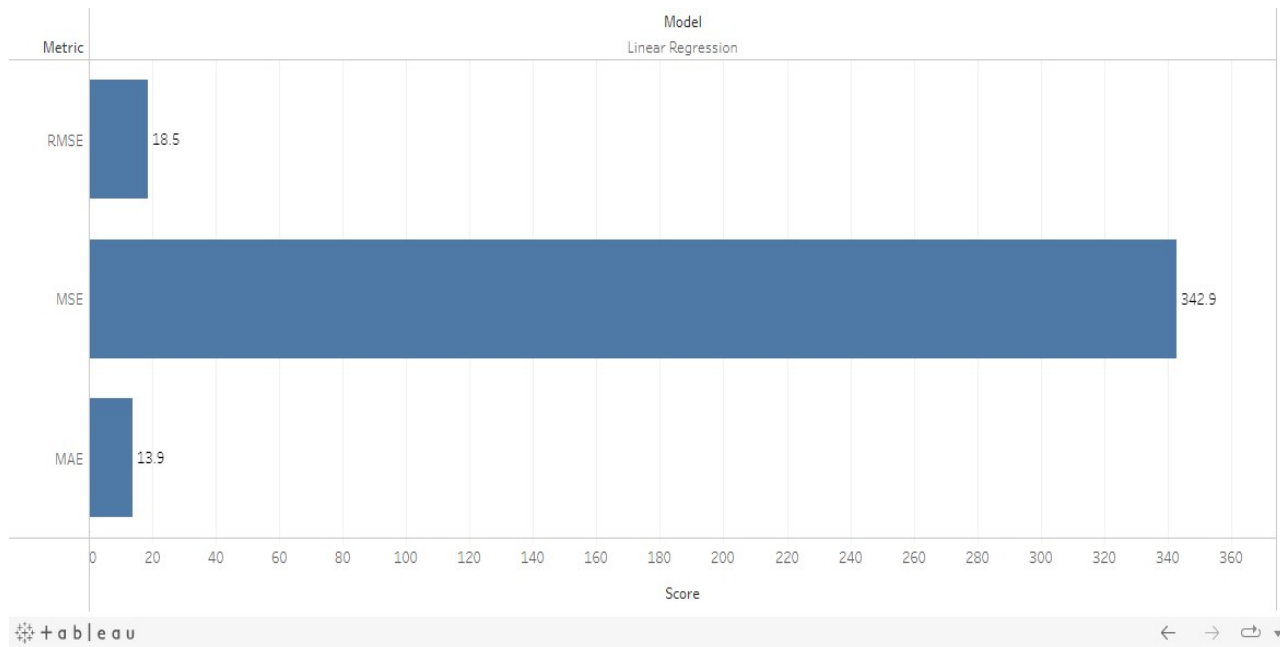
The above image depicts the overall analysis of the Algorithms used and it's score differences. Individual analysis of scores of the Algorithms used are shown below -



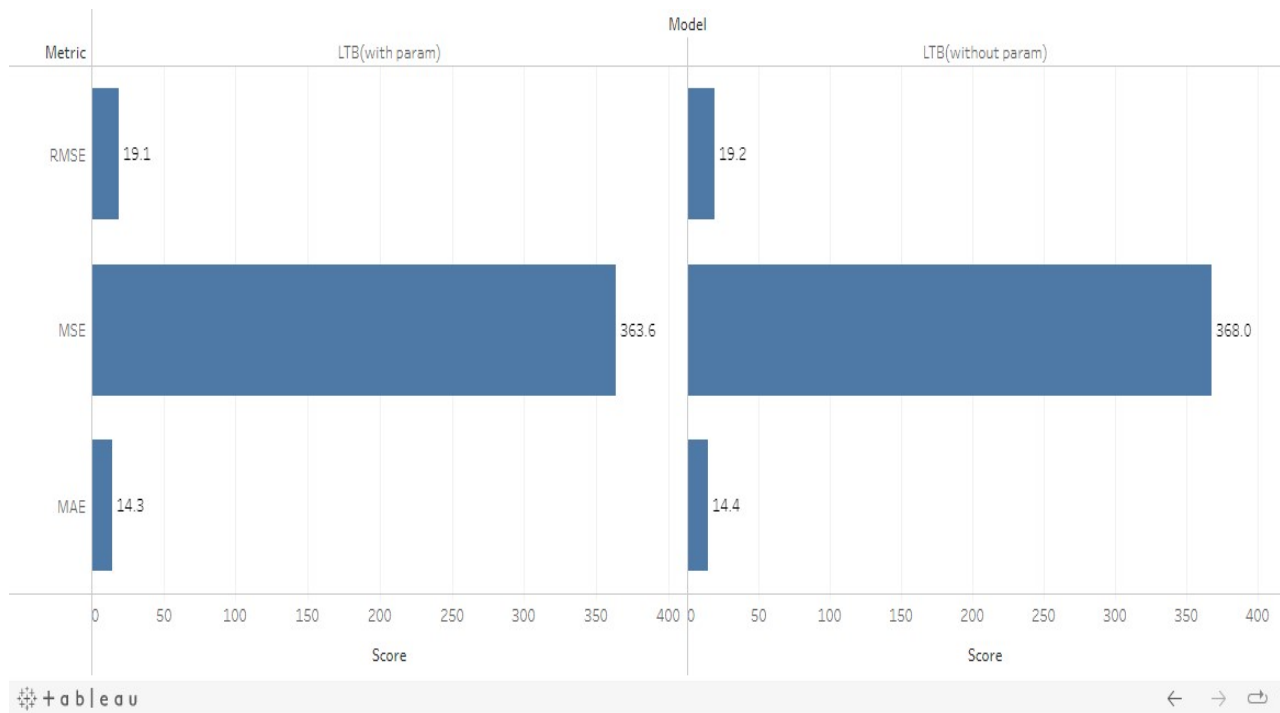
**Fig6.5:Decision Tree Regression**



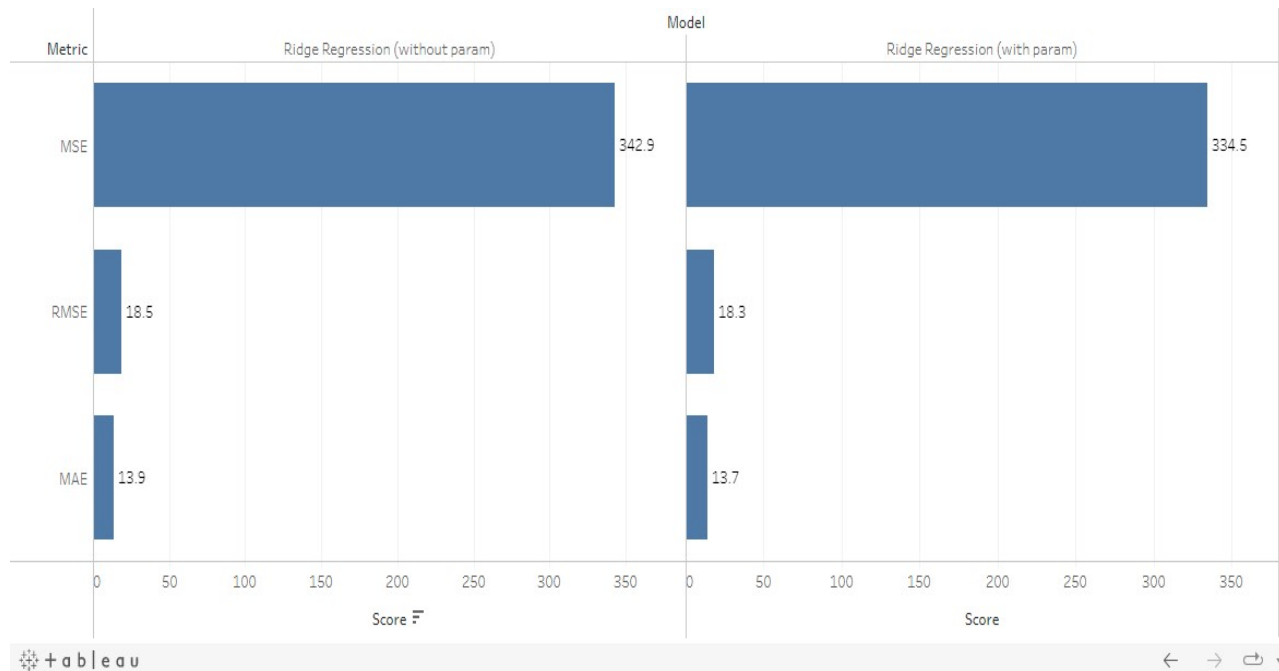
**Fig6.6: Lasso Regression**



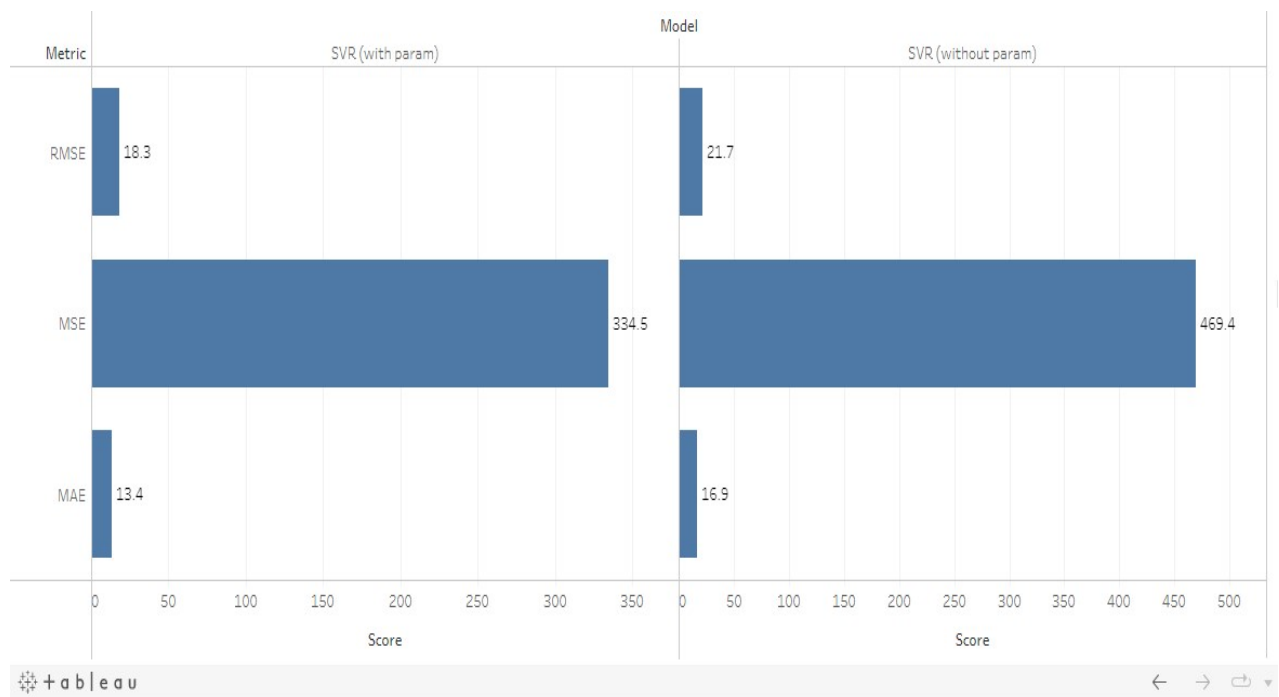
**Fig6.7: Linear Regression (Base Model)**



**Fig6.8: LTB Regression**



**Fig6.9: Ridge Regression**



**Fig6.10: SVR Regression**



Model	Metric	Score
Linear Regression	RMSE	18.51
Ridge Regression (without parameter)	RMSE	18.51
Ridge Regression (with parameter)	RMSE	18.28
Lasso Regression (without parameter)	RMSE	18.13
Lasso Regression (with parameter)	RMSE	18.13
SVR (without parameter)	RMSE	21.66
SVR (with parameter)	RMSE	18.29
DTR(without parameter)	RMSE	25.13
DTR(with parameter)	RMSE	21.75
LTB(without parameter)	RMSE	19.18
LTB(with parameter)	RMSE	19.06

**Fig6.11: Output Table**

## **7. CONCLUSION AND FUTURE SCOPE**

### **7.1 Conclusion**

We studied three novel approaches in this project to predict the first innings score of a live IPL match. From the results, we can conclude that the Support Vector Machine algorithm has the highest accuracy of the prediction. So, we are using the Support Vector Machine model for the prediction purpose. Teams can use Current Run Rate (CRR) to predict the final score even before the 20 overs have been bowled. So, the teams can know when to accelerate and when to play aggressively to increase the run rate while putting a target. This model can predict the score by putting the current ongoing live match stats and predict the results beforehand by using their previous records. Hence, this can be used by cricket lovers for predicting the final score of a live IPL match.

### **7.2 Future Scope**

In the future, we can implement a model for predicting the chasing probability. We can work on improving the accuracy of the model used in this project. Factors like venue, pitch, and the opponent team can be considered for the prediction.

## 8. PUBLICATIONS

**Journal Name:** INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR)

**Journal No:** 43602

**Paper ID:** 247866

**Volume:** 9

**Issue:** 2

**Page No.:** 250 - 254

**Title:** CRICKET SCORE PREDICTION

**Link:** [http://ijrar.org/viewfull.php?&p\\_id=IJRAR22B2361](http://ijrar.org/viewfull.php?&p_id=IJRAR22B2361)

## 9. REFERENCES

- [1] S.Muthuswamy and S.S.Lam (2018) : "Bowler Performance for One-Day International Cricket Using Neural Networks"
- [2] G. D. I. Barr no-B. S. Kantor (2012) : "How to Compare and Select Cricketers in Overs Overs Cricket".
- [3] S. R. Iyer and R. Sharda (2010) : "Predicting Performance of Athletes Using Neural Networks: An Application to Cricket Team Selection"
- [4] I.P. Wickramasinghe (2015) : "Guessing the performance of batsmen in test cricket"
- [5] "M. Ahmad, A. Doud, L. Wang, H. Hong (October 2016): "Prediction of rising stars in cricket".
- [6] "P. Somaskandhan, G. Wijesinghe, L. Bashitha,(2017) : Identifying optimal set of attributes that impose high impact on end results of cricket match using machine learning."
- [7] "M. Rehman, O. Shamim, S. Ismail, October (2018) : "An analysis of Bangladesh One day international cricket: Machine learning approach".
- [8] "A. Tripathi, J. Vanker, B. Vaje, V. Varekar : "Cricket Score Prediction system using clustering algorithm."
- [9] M. Bailey and S. Clarke : "Predicting the Match Outcome in One Day International Cricket Matches, while the Game is in Progress", Journal of sports science & medicine, vol. 05, no. 04, pp. 480-487, 2006.
- [10] N. Pathak and H. Wadhwa : "Applications of Modern Classification Techniques to Predict the Outcome of ODI Cricket", Procedia Computer Science, vol. 87, pp. 55-60, 2016.
- [11] P. Satao, A. Tripathi, J. Vankar, B. Vaje and V. Varekar : "Cricket Score Prediction System (CSPS) Using Clustering Algorithm", International Journal of Current Engineering and Scientific Research, vol. 03, no. 04, pp. 43-46, 2016.
- [12] [https://web.archive.org/web/20130120040151/http://www.icccricket.com/match\\_zone/historical\\_ranking.php](https://web.archive.org/web/20130120040151/http://www.icccricket.com/match_zone/historical_ranking.php).