PrOBML: A machine learning approach to Predict, Optimise & Build fantasy Cricket teams using evolutionary algorithm

Akarsh Singh, Aparna Sharma, Utkarsh Singh

Abstract:
The idea of the selection of the correct players plays a crucial role in the team formation of any sport like it does for Cricket. Enforcing a point system, player points were predicted using historic match data. Different machine learning approaches were integrated and compared to form a model ensemble. Subsequently, using Genetic Algorithm (GA), these points were used to embody the model fantasy team. Our findings show that various approaches yield different results and hence, an amalgamation of such approaches can be utilised. The top players from each team could be recognised with as many as three being accurately identified out of five. Additionally, accuracy above 61.2% was secured in the formation of the fantasy team. The paper thus entails such decisive information to all the key stakeholders. Using this approach, one can thus predict, optimise and in turn build their fantasy team with the least losses.

Keywords- *Cricket, Sports Analytics, Betting, Fantasy Sports, Optimise, Machine learning, RNN, LSTM, Genetic Algorithm (GA).*

Introduction:

Fantasy sports popularity has been erupting in the sports and gambling industry for the past two decades with exponential growth potential. Enhanced user engagement in gambling and sports viewership is the by-product of surging fantasy sports popularity. Online fantasy sports gaming (OFSG) is a form of ability-based online sports gaming where users can create their sports teams by combining players from upcoming matches. Multiple constraints and objectives must comply as a pre-specified budget for team selection should be matched at best. Players get points based on match statistics and performances (when played in real-time), and the team with the highest points wins. Prizes might also be available for users finishing above a certain threshold of points. There are variations to team selection with every sport. Users can play for entertainment or with real money, which is monumental for the rapid development of such platforms. Its booming popularity has made fantasy sports a multi-billion dollar business globally.

Cricket continues to flourish in countries like India Australia, England, South Africa as well as West Indies [1]. An open-sourced report titled 'The evolving landscape of sports gaming in India' on KPMG's official website (one of the Big Four accounting organizations) claims that 90% of 'Indian fantasy sports users' market share vests with Dream11 (a fantasy sports company). It also proclaims that 85% of users play fantasy cricket on the dream11 application and that 71% of respondents in general play fantasy cricket in India. Dream11's user base has grown exponentially with approximately 50 million users in 2018 (around 300% growth from 2017) [2].

The process of extracting intelligence from data has made its mark in the sports industry as well. IoT devices, optical sensors, and computer vision applications collect volumes of data from sporting performances and training sessions. A multitude of structured and unstructured data in conjunction with analytics, sport sciences, and game theory can be valuable to sporting teams, coaches, and decision-makers [3]. Predicting and improving player performance metrics or player injury threat prediction are a few of many powerful applications in sports analytics [4].

Since cricket is the most popular sport in India by a mile, it is no surprise that fantasy cricket would crunch the numbers here too. Keeping this offset in mind and the tremendous amount of resources involved, we aim to forecast cricket players' performances using analytics, statistics, and machine learning algorithms. Further, we optimize predicted results as a multi-variate optimization problem with multiple pre-defined conditions to provide the best fantasy team possible. Each aspect of our proposed predictive model will be explored further in detail. Preliminary analysis, selection methods, and visualization have also been provided. We can take inspiration from this model and apply it to different sports. We have also considered all three cricket formats for expansive predictive capabilities that is, Test, Twenty20 (T20), and ODI (One Day International).

Related Work:
We have taken into account research work already done in the field of sports involving an element of forecasting. We perused through papers that were analytical and data-centric. The primary goal was to find research related to cricket and try to understand approaches that have been used for predicting outcomes. The focus was also on variable selection, derivation of performance metrics, algorithms used, and general flow. Secondarily, we documented research from various analytical studies from different domains and alternative team sports to find similarities and other unique approaches. This helped us to establish an overview of data-centric sporting research and clarity of flow for our research by taking inspiration from these academic studies.

Statistical mapping was used to forecast player performance amongst Bangladeshi international cricketers [5]. Basic bowling, batting, and performance parameters were considered. It narrowed down player performance to provide a list of top 30 players and further optimize results to deliver a 14 player squad by using Genetic Algorithm. Study [3] gave a concise overview of big data analytics and provided a flowchart for the data mining framework. Cricket data was specifically considered. Specific queries were written in the hive to create sub tables in a framework. In hindsight, this paper provided us with a systematic approach for big data projects. Shubham Agarwal et al. [6] statistically modelled player data to assess fit for selection in a game/series. They made use of metrics like bowling/batting average, recent form, and location with weights assigned to control the impact of each factor. Authors Iyer Subramanian & Sharda Ramesh [7] showcased promising results for best performing bowlers/batsmen. With defined heuristics for selection and performance evaluation multilayer perceptron, neural networks were used to make predictions The study also assets how these networks are capable of classifying different roles in cricket. Subsequently, [8] implemented new performance indicator variables and standard variables were efficiently explained as well as revised. Great insights were provided and the effect of various factors was efficiently quantified, for future studies. Variables like pitch impact and comparative performance explore various avenues and subdued indicators in international cricket. Moreover, authors in [9] established an efficient analysis of a defined decision-making unit for cricket team selection made use of bowling/batting statistics to create a team selection model for upcoming series competitions. M. M. Hatharasinghe and G. Poravi [10] systematically enlisted different areas of study in cricket that involved forecastings, such as player performance, match simulation, and team selection. Analysis of existing studies and data collection methods were done to finally provide with missing links and limitations. Lastly, various machine learning methods were used with independent feature selection approaches to predict IPL match results in different phases of the game. Models and visualizations were created on python [11].

Similarly, there were abundant research studies we retrieved that were related to other sports as well. The paper [12] evaluated the winner of a football match in the English Premier League implemented by using Gaussian Naïve Bayes, a machine learning algorithm. Literature also consisted of detailed a study [4] of comparison between amateur and professional writers and elements they tend to focus on while writing about fantasy sport. Clear methods were established for text analysis, determining correlations, and hypothesis testing. Differences in writing style were also evaluated thoroughly. A crisp observation corroborates that these writers focus on presenting facts, opinions and tend to make predictions. For basketball, [13] deeply studied analytical parameters to individual and team performance. These analytical parameters were classified into various tables for comparison and analysis. It also aimed to map the best players and their achievements into significant parameters for optimization. The complex performance index was calculated using various statistical points to forecast team performance and stand-out players for upcoming seasons. Additionally, [14] delves deep into finding solutions for classic fantasy team problem statements in the NFL. It involved the prediction of individual fantasy points to match-winning predictions against a spread value. In line with other machine learning predictions, algorithms like boosted decision trees averaged perceptron, and best league participant has been used to solve both cases. Permutations of biased and unbiased selection have also been used for fantasy team optimization. The demonstration of a pre-processed filtering mechanism for player selection in this research proved to be vital for monetary aspects of fantasy sports betting.

Problem Statement:
This research aims to create machine learning models to accurately predict cricket fantasy points for players. We have divided our study into two analogous segments to efficiently analyse the various technicalities of our predictions.

1. Firstly, to predict scores for players over a span of a certain number of games. We must specify a metric to measure the goodness of our results. The string of games can be from an international series or a tournament like IPL.
2. Secondly, to assimilate a dream team for a matchup adhering to Dream11 rules. This includes predicting scores for all players on a specific match day and later, using an optimizer to optimize the best team. Accuracy would be then, measured by tallying with the actual dream team for that matchup.

Data Collection:
PostgreSQL relational database management system monitored by sportsanalytics.in from their trusted predictions platform was the source of data collection. With over a hundred interconnected tables present in the database, an entity-relationship diagram was created for ease of use while accessing these tables. Further, respective problem statements were put forward before retrieving relevant data using query requests. Data related to every match was ephemeral to our model. We focused on incorporating relevant match data from the database. Primary cricket statistical metrics used in predicting fantasy points were included first. These data points will help us calculate actual fantasy points from every match. This data will also help us formulate a time series problem for prediction. Secondary data such as venue, date, and toss winner was also encompassed. In summary, data was organized and allocated into four different data frames for use in the model. A description of specific columns from these tables is systematically presented below

Methodology:

To predict fantasy points and estimate performance it was vital to thoroughly understand how fantasy points were calculated in the first place. We closely followed the Dream11 method for determining fantasy points after a competition of an inning. All constraints related to the assignment of points were incorporated into a custom function programmed on Python. Fielding points were not considered due to a lack of relevant data. Points calculation scheme for T20 cricket was used in this study as standard for all formats to avoid model complexity. (Specific conditions to the number of points awarded).

Table 1: Batting Points

Runs	+1
Boundary Bonus	+1
Six Bonus	+2
30 Run Bonus	+4
Half-Century Bonus	+8
Century Bonus	+16
Dismissal for a Duck	-2

- Any player scoring a century will only get points for the century. No points will be awarded as their 30 run Bonus or Half-century Bonus.
- If any runs are scored on an overthrow, points for those runs will be credited to the batsman on strike for that ball.

Table 2: Bowling Points

Wicket (excluding run out)	+25
LBW/Bowled	+8
3 Wicket Haul	+4
4 Wicket Haul	+8
5 Wicket Haul	+16
Maiden Over	+12

Table 3: Economy Rate Points

Below 5 runs per over	+6
Between 5 – 5.99 runs per over	+4
Between 6 – 7 runs per over	+2
Between 10 – 11 runs per over	-2
Between 11.01 – 12 runs per over	-4
Above 12 runs per over	-6

Note: (Minimum 2 overs to be bowled)

Table 4: Strike Rate Points

Above 170 runs per 100 balls	+6
Between 150.01 – 170 runs per 100 balls	+4
Between 130 – 150 runs per 100 balls	+2
Between 60 – 70 runs per 100 balls	-2
Between 50 – 59.99 runs per 100 balls	-4

Below 50 runs per 100 balls	-6

Note: (Minimum 10 balls to be played)

After establishing the system to calculate fantasy points, machine learning models were selected for accurate predictions. Our study is into two segments for ease of understanding. Innings played at that time for a player is denoted by *N*.

1. Predicting fantasy points of the starting squad before a match.
    - LSTM used when innings (*N*) played at the time was greater than equal to 65 ($N>=65$)
    - AR Model (1 or 2 degrees) used when innings played at the time was between 25 and 65 ($25<=N<65$)
    - Averages approach was used when innings played at the time was less than 25 ($N < 25$)
    - If no data was found then the value 1 was simply returned (i.e., $N = NaN$)

2. Optimization of the optimal fantasy team.
    - For match day prediction, we have also used an aggregate model for players where *N* was greater than equal to 65 ($N>=65$).

The reasoning for choosing these models for different values of N is provided later in this research.

The optimization problem's successful implementation is highly correlated with the accuracy of our predictive models. Model selection ought to be in sync with our problem statement and data resources, that is, univariate time-series prediction of discrete numeric data. We also had to be wary of the limited data points we had for training, testing, and cross-validating our models. Deep learning models are suited to train on datasets with over a million data points, However, It is rare for an athlete to cross the 500 match milestone in his senior career, this critically limits model training capabilities (with at most 500 data points). Also, the enormous uncertainty surrounding sporting performance inviting the room a lot of noise; makes this problem a tough nut to crack. Our data frame, for example, has 287 T20 match records for Rohit Sharma (batsman) that is the highest amongst 1664 players present. The importance of visualizations is vital for engaging user experience and understanding, multiple visualizations have been included to communicate our findings [15]. The process is portrayed in figure 1.

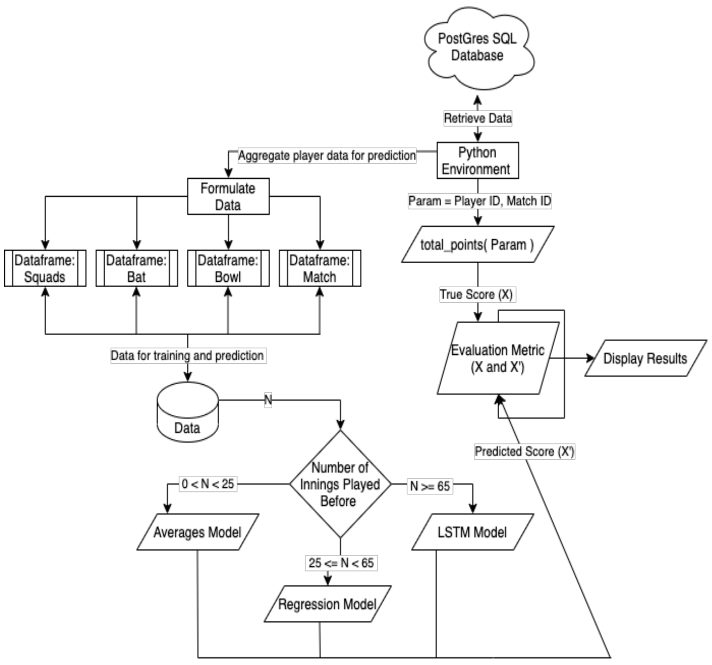


Figure 1: Process flowchart

LSTM

Recurrent neural networks (RNN's) are the most widely used deep learning algorithm for sequential data techniques. For our data, it's imperative to capture long-term, short-term dependencies and find complex patterns for good results [16]. The lack of data for training (as discussed above) makes it critically pivotal to capture these temporally significant traits. In theory, RNN's are robust to memory-keeping issues but practically experience vanishing/exploding gradient occurrence under respective circumstances. An ample amount of research has highlights the impact of these issues on model performance and proposing methods to overcome them. Figure 2 highlights the preliminary structure of RNN.

RNN's are trained by defining a loss function (*Ls*) that measures the error between the labelled output and the predicted output, aiming to minimize it. For a single time step, first, the input arrives and is processed through a hidden state, and the estimated label is calculated. In this phase, the loss function is determined to evaluate the model error here. The total loss function, **L**, is calculated (*Ls* calculated for every observation), and by that, the forward pass process culminates. The second part is the backward pass (backpropagation), where derivatives are calculated.

The vanishing gradient problem occurs when the backpropagation algorithm moves back through all of the neurons of the neural net to update their weights during a training epoch. The cost function computed at a deep layer of the neural net will be used to change the weights of neurons at shallower layers [17]. The gradient calculated in a node deep in the neural network will be multiplied back through the weights earlier in the chain (multiplicative property). This dilutes the gradient as it moves further back that can cause it to vanish and result in a loss of vital learnings.

The actual multiplied factor through a recurrent neural network in the backpropagation algorithm is referred to by the statistical variable W_{rec}.

- When W_{rec} is small, the gradient term approaches zero exponentially, creating barriers in learning long-term dependencies.

- When W_{rec} is large, the gradient term approaches infinity exponentially, making the process itself unstable.

Both of these issues call for unwanted learning bias displaying poor results. Unfortunately, this gap capturing long-term dependencies is subject to an increase in traditional RNN structures. Research has proved that despite this issue not arising in theory, practically can't be avoided when applied to real-world problems. Since the margin for error is thin for sports data, alternatives have to be used to tackle this problem.

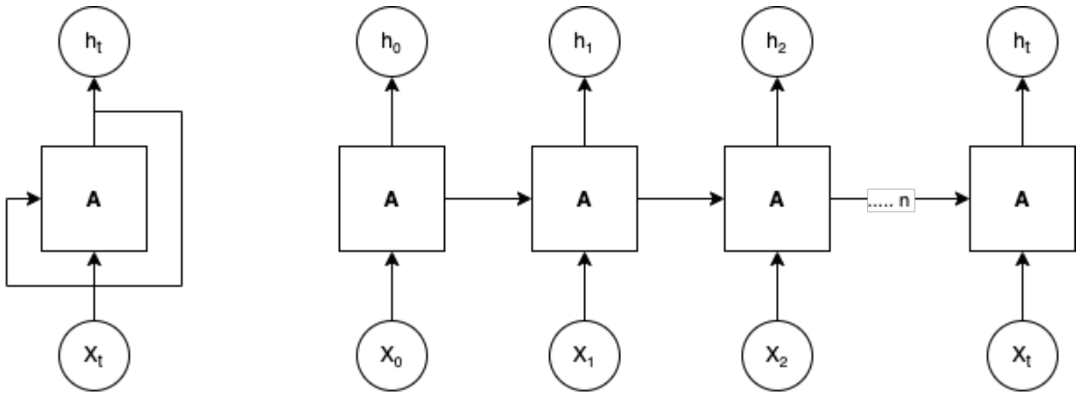
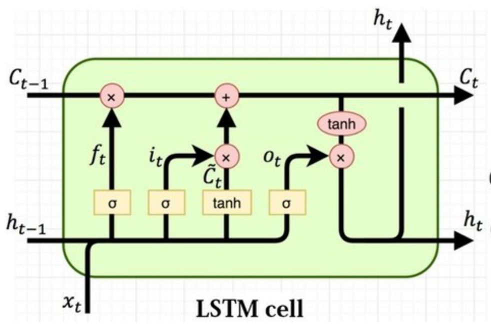


Figure 2: Basic RNN structure

Long Short Term Memory networks (LSTM networks) were designed to avoid the long-term dependency problem. They incorporated novel design paradigms within their structure to emphasize these issues. LSTM networks are still subject to continuous improvement by researchers tackling focused problem statements [18]. LSTM networks are gated RNN models where the LSTM cell replaces the traditional RNN cell. A 'cell state' is also an added connection to mitigate the vanishing/exploding gradient problem. The gated structure in LSTM networks has three gates that perform respective operations stated below:

- Input gate: Checks if the memory cell is updated and what information will be stored in this iteration.
- Forget gate: Checks if the memory cell is reset to zero and determines what information should be forgotten for the current iteration.
- Output gate: Checks if the input of the current state is to be made visible [19].

All of these gates use a sigmoid activation function to maintain the differentiability of the model and constitute smooth curves in the range of zero and one. Another vector $\bar{C}$ is used to modify the cell state that incorporates the *tan(h)* activation function. Primarily because *tan(h)* is a zero-centred, long-sum operation that distributes gradients well and allows cell state information to flow longer without vanishing/exploding.



$$i_t = \sigma\big(x_t U^i + h_{t-1} W^i\big)$$
$$f_t = \sigma\big(x_t U^f + h_{t-1} W^f\big)$$
$$o_t = \sigma\big(x_t U^o + h_{t-1} W^o\big)$$
$$\tilde{C}_t = \tanh\big(x_t U^g + h_{t-1} W^g\big)$$
$$C_t = \sigma\big(f_t * C_{t-1} + i_t * \tilde{C}_t\big)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 3: LSTM Model Summary [35]

Regression Model:
Regression models are one of the most fundamental yet extremely effective algorithms used in machine learning. The primary assumption for linear regression models is the relationship between the independent and the dependent variable. These models have two variables (one dependent and one independent) with an assumed linear relationship. As a result of this pre-define architecture, it is also known as a parametric learning algorithm [20]. If multiple independent variables are involved to determine the relationship with a dependent variable, the model is known as a multiple linear regression model. Usually used for predictive analysis and determining the relative impact among variables, it can provide decent results for our case. The presence of non-linear relationships will provide subpar results when trained on a strictly linear assumption. Polynomial regression is used to model non-linear relationships by introducing higher degree polynomials in the standard equation.

$$Y_i = f(X_i, \beta) + \varepsilon_i \tag{1.1}$$

Where,
Y_i = dependent variable
f = function
X_i = independent variable
β = unknown parameter
ε_i = error terms

For multiple linear regression, there is more than one X term present in the standard equation [21]. Hence, X_i is to be replaced by Xn_i in (1.1). Similarly, for polynomial regression, X_i can be replaced by X_i^2, $\log(X_i)$, etc depending on which polynomial feature fits best. Ordinary least squares (OLS) and gradient descent are a few of many methods to estimate the value of coefficients. Regularization modifications like Ridge and Lasso regression, various normalization techniques are used to fit the best models. However, elaborate discussions on these methods are out of scope for this study [22].

Since we are forecasting/predicting fantasy points by using past performances our model is autoregressive. Autoregressive models (AR models) predict future behaviour based on past behaviour. That is, to predict output variable (y) at a time interval t using previous values of itself (lagged versions of output). Although sporting performances may/may not have the strongest autocorrelations, it is safe to assume this for a generalized model. AR models also stress a certain degree of randomness and uncertainty (stochastic process) into them which perfectly aligns with our data and problem statement.

Standard Averages:
This method for prediction deals with measures of central tendencies. An average performance index is by far the most used statistical metric to rate sporting performances. For example, In football, minutes per goal is a massive parameter to determine how clinical a striker is (Total Minutes Played / Goals Scored) [23]. In Cricket, average runs/run rate has an impact on a player's run-scoring ability. Similar metrics are used in basketball and rugby to determine player ratings and quality. It is fair to say that some statistical metrics are more important than others in fantasy sports. However, this doesn't mean that they are any less of a consideration for actual team selection. For example, in an actual cricket match, one might give a heads up to Yuzvendra Chahal, a lethal wicket-taking spinner. However, would probably choose

Ravichandran Ashwin in his fantasy team because of his decent wicket-taking abilities as well as for his ability to hit boundaries as a bowler.

In football, one might choose Marcus Rashford in his fantasy team's midfield for his high goal contribution statistics but at the same time would vouch for someone like Ngolo Kante in the actual game for extraordinary defensive abilities. The sole fact that such statistics can be a handful in a fantasy sport, this method is included. Here we are dealing with median, mean, and weighted mean.

$$Standard\ Average = \frac{\sum_{n=1}^{k} x_n}{k} \quad , i = 1, 2 \ldots k \tag{1.2}$$

CHECK FORMULA ABOVE

Genetic Algorithm:

Genetic Algorithm, also abbreviated as GA, conforms to Darwin's theory of survival of the fittest [24]. Evolutionary algorithm is also a moniker given to it owing to its origin. Darwin theorised the postulation of natural selection. Natural selection referred to "the features most beneficial to organisms in relation to conditions of life" [25]. The idea was that the progeny inheriting the best features or characteristics would eventually sustain. Upon multiple iterations, the fittest of the group would survive [26]. GA was inspired by this biological phenomenon. The process of GA often incubated with the population. The population, in essence, was made up of the multiple solutions possible to a particular problem. Herein, they accounted for various team players. An individual solution was represented as a gene and was cumulated together to form a string, categorised as a chromosome. The [26] chromosomes formed the candidate solution and often took up the solution space of the algorithm. Subsequently, these solutions or chromosomes are assigned a fitness score which was tantamount to how 'fit' that solution is. This fitness function essentially entrusts each solution with an empirical value. It provides bases for if a particular offspring should be promoted to the next generation.

Cross-Over, Mutation, as well as Selection, made up the biological operators in this algorithm. The selection phase, as the name suggests would selectively pick 'parents' that would further undergo cross-over to produce off-springs. Perhaps the nub of the algorithm, the cross-over operator gave rise to the off-springs or the progeny by taking features from the ancestral lineage [27]. The cross-overs are often random and the newly formed offspring then become a part of the new population. Cross-overs are requisite, otherwise; they would render the replica of the parent. The mutation parameter ensures diversity and happened in an unordered fashion [26]. Certain parts of the chromosome were flipped and changed by value.

With the unwinding of the algorithm, the resulting population is an assortment of the finest individuals with the best probabilistic features.

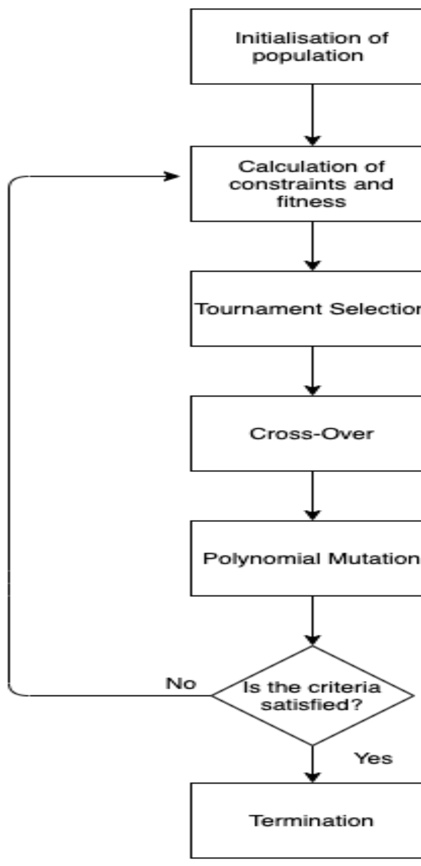

Initialisation of population
Calculation of constraints and fitness
Tournament Selection
Cross-Over
Polynomial Mutation
No
Is the criteria satisfied?
Yes
Termination

Figure 4: Genetic Algorithm Process Flowchart

Multi-Objective Genetic Algorithm:

Multi-objectives Genetic Algorithm (MOGA) is a multi-objective optimization technique based on the Darwinian evolutionary theory that emphasizes survival of the fittest. They are efficient optimization algorithms not requiring knowledge of calculus/derivatives and are easy to understand. The fact that they are easily paralleled stochastic models makes them a fit for our purpose. Diverse regions of a solution space can be explored with MOGA with quicker convergence and relaxed computation cost when compared to most other multi-objective optimization algorithms. These algorithms are based on various principles like decomposition (*MOEA/D*) or indicators (IBEA) [28, 29]. Here it was thoughtful to go with genetic algorithms based on Pareto dominance. This solution set of non-dominated Pareto frontiers help in choosing optimal solutions based on perceived tradeoffs (decided according to a specific situation). In a problem where more than one agent is present, if no change could lead to improved performance for some agent without some other agent losing, It is called a Pareto optimal [30]. Any room for improvement where one agent can gain and any other will at least not lose is called a Pareto improvement. A set of Pareto efficient allocations is known as a frontier. This set of results will help choose the best team based on a definite pattern we are looking to highlight. Solutions are presented in a population of chromosomes, each chromosome has a fixed number of genes, each gene discretely represents a characteristic that helps to quantify the goodness/rank of each chromosome in the population.

The variant of MOGA chosen for this study is Non-dominated Sorting Genetic Algorithm II (NSGA-II). The NSGA-II optimizes each objective without being dominated by any other solution [31]. NSGA-II has the following three features that make it superior to most other optimization algorithms:

1. It uses an elitist principle where the elites of a population are deliberately passed to the next generation. This principle helps in swifter convergence for optimization and generalizes the sample space well to provide Pareto frontier solutions.
2. It uses an explicit diversity preserving mechanism that is crowding distance sorting. The crowding distance used in the NSGA-II depends on the significance of the solution sets and their distance to the respective solution boundaries. In layman terms, ranking and distance. Of course, better-ranked solutions are preferred, but when ranks between two solutions are the same, the distance metric helps in the final selection [32]. This helps in preserving diversity for future generations and prevents the algorithm to get stuck in the local maximum/minimum solution.
3. It emphasizes non-dominated solutions that hail the importance of Pareto dominance as explained above.

The addition of these features brings down the total complexity of this algorithm to O(NM^2). A drastic decrease in complexity when compared to the foregoing versions where it was O(NM^3) [33]. A step-by-step walkthrough of an entire optimization generation cycle is explained further in this study for an adequate understanding.

Models Parameter and Hyperparameter Tuning:

Well-documented packages for machine learning and statistics can be used without in-depth knowledge of their complex back-end architecture. Hence, vital skills like feature engineering, parameter/hyperparameter tuning, regularization, etc, are refinement techniques that can place models ahead of the curve, cementing the fact that, how important this is for machine learning models. In layman terms, we can think of parameter tuning in the context of designing a vase. Here, deciding the material, machines used to create the vase, and decorative elements are equivalent to parameters. The rough dimensions, placement of decorative material, the shade of picked color for painting, and other such aesthetic modifications can resemble hyperparameters.

In predictive modeling: feature selection, feature engineering, parametric weights assignment, etc are examples of parameter tuning. Whereas determining the alpha value for learning rate in gradient descent, deciding lambda value for *L1* regularization, depth in decision trees, etc are the model's hyperparameters. They do not explicitly reflect in the actual model are salient contributors to success.

There are many factors because of which generalization is very hard for sports data. Each player is very different and performs differently under unique circumstances. Some players are much more consistent than others, and for some players, adequate data is not present for training, largely affecting performance and increasing the margin for error. Adequate noise reduction is difficult to remove due to the natural uncertainty in sports. While predicting parameters and hyperparameters in each case, variant individual player data was considered and average values were selected. If a certain metric performed well on a random cluster of players then it was tested on other random clusters of the same size. If accuracy dropped by more than 10%, it was rejected. This process was standard for all algorithms for parameter selection. The number of data points required to use an algorithm was also specified beforehand. For example, regression was not feasible if data points were less than 30 because it had terrible accuracy. Therefore, incorporating this standard process while selecting model specifications improved efficiency.

Auto-Regressive and Polynomial Regression Models
For AR models:

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \dots \alpha_n T_{t-n} \qquad (2.1)$$

For an auto-regressive model, we calculated its order (*p*) as standard. This term signifies how many lags in time *t* does the current value depends. We negated that time series models usually have data in fixed intervals of time for our model (because sporting matches do not have fixed time intervals). We calculated this using autocorrelation (ACF) and partial-autocorrelation functions (PACF). The correlation of an entity (at time t_2) with itself at a previous timestamp ($t_{1)}$ is known as autocorrelation. Partial autocorrelation is defined as autocorrelation between two timestamps after confidently controlling timestamps further precedent in time. For example, the autocorrelation of Y_{t-3} with Y_{t-2} after controlling autocorrelation with Y_{t-1} and so on. Hence, PACF and ACF have the same value at the first timestamp after which they slightly differ. These functions also inform us about the stationarity of the data. Stationary is crucial for good results in time series models because of their stochastic nature. If data is not stationary it has to be converted into the respective form.

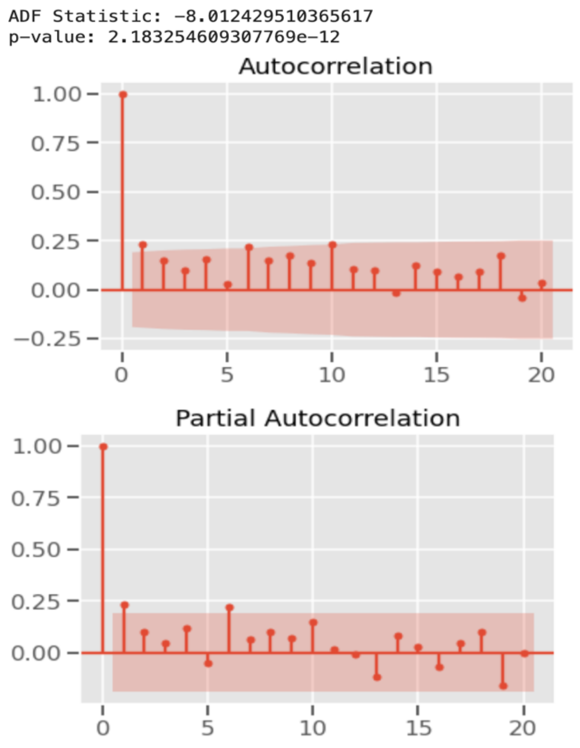
Firstly, for regression, we only considered cases where the data points were greater than 30. This assumption is solely by looking at the availability of player data we possess and the minimum data points required to give us fair results. With players with datapoints lesser than this at a given timestamp, we will use the averages approach. For standard reference, we have considered the start of the IPL 2020 season for tuning parameters simply for the following reasons.

1. The maximum amount of players we have in our dataset have played in the IPL 2020
2. It is a good generalization for our parameters with 12-16 matches available for final testing and hyperparameter tuning.
3. It has also had a fair mix of well-performing rookies (players who were playing their first season), established star players, and underperforming/out-of-form players.

Stationarity was confirmed as the average *p*-value was less than 5% (in the range of 10^{-12} to 10^{-10}, also because of fewer datapoints), confidently reject the null hypothesis of non-stationarity. The augmented dickey fuller test statistic (ADF statistic) was also largely negative in all cases (between -9.23 and -5.65) that corroborates for the same. Since we were using for prediction was calculated fantasy points from previous games, order (*p*) was then calculated.

We looked at ACF and PACF plots from various player performances and figured that in most cases autocorrelation was not very strong. Solely because of the nature of sporting data and the lack of data points. After analysis, we found that decent Autocorrelation was seen in the ACF plot with 4 and 5 lags in data. In some cases, lags 3 and 7 also showed equal autocorrelation but were outnumbered by lags 4 and 5. PACF plot also supported this observation with slight differences in underlying values. To our surprise, plenty of data points also showed a high negative correlation with previous timestamp values. This meant that statistically, some players perform well after a bad performance and vice-versa.

PACF and ACF plots showed an autocorrelation of 0.2 to 0.3 on average with lags 4 and 5. This trend was observed more with *p*=4 lags and since in very few cases value at lag 5 (order *p*= 5) were more than lag 4 (order *p* = 4) if not equal, we choose (order *p* = 4) for our generalized AR model.



```
ADF Statistic: -8.012429510365617
p-value: 2.183254609307769e-12
```

Figure 5: ACF/PACF plot for a random player

After figuring our lag value, we also tried implementing polynomial coefficients to see how well they fit. Value of quadratic order to be 1 gave good linear results with generalized, yet safe results for performance. order 2 polynomials performed well when datapoints were above 75/100 but gave terrible results otherwise. Due to the lack of data points for training and testing for each player, any value above 2 was out of the question. Hence, we stuck with orders 1 or 2 depending on the case for our model. Pearson's coefficient and heat maps were also plotted lastly to check for correlation. Results yet again matched with ACF/PACF plots with a positive correlation peaking to 0.3 and a few negative correlations depending from player to player. SK-learn library in python was used to fit this model on our data for training and testing.

LSTM-RNN:

We have used the Keras library supported by TensorFlow in our Python notebook form modeling our network. In our LSTM model, we choose five timestamp lags as input. As we saw autocorrelation of data points from iterative, random player clusters was the strongest when order *p*= 4/5 by heatmaps (Pearson's coefficient), ACF plot, and PACF plot. Here, we chose five lags as input for the following two reasons:

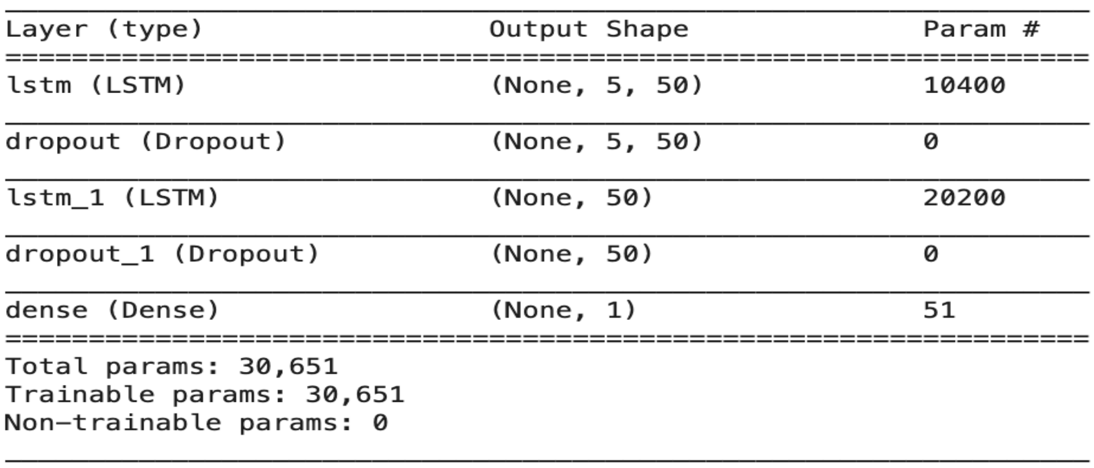
1. It provided more flexibility to the model as more parameters and patterns were captured by neuron layers in the LSTM model.
2. The tradeoff for slightly less autocorrelation (with order *p*= 5) among data points and enhanced feature learning capabilities was worth this cost.

Form research work on RNN model architecture and usage we found out that even 10,000 rows of data points can fail to yield results with absolute certainty. At least 500 rows should be available for effectively training data and capturing the true, generalized relationship.

However, in our case, we have considered 100 rows of data points to be the bare minimum for analysis.

After data frames were ready, we converted input parameters into NumPy arrays. This was important because reshaping NumPy arrays are easy and work well with the Keras-TensorFlow architecture. NumPy arrays are also much faster than python lists (four times faster) and occupy less space as well. Our data was a 2D array with five columns and *n* rows (*n* is the number of matches the respective player has played before). Each vector had a shape of (5,1). It had to be reshaped into a 3-D tensor with shape (1,5,1) because of TensorFlow architectural requirements. Hence, forming the input we had to give to our LSTM models input layer.

We have used two recurrent LSTM layers in our model. This structure is also known as stacked LSTM. The LSTM layer above provides a sequence output rather than a sole output to the LSTM layer below, this will help maintain long-term dependencies effectively. After an in-depth study of best LSTM practices in practical applications and research work, we have also included dropout layers after every LSTM layer. Dropout layers help to reduce the variance in training fit. They also reduce the dependency/sensitivity of the model concerning individual nodes. This helps in faster learning and avoids over-fitting. Layer nodes have also been chosen to keep in mind the high amount of uncertainty present in the data with vast variance among individual players. Complex relationships between performances add to the complexity too. Hence, we used 50 nodes in each LSTM layer. After each LSTM layer, a dropout layer was added for regularization. The dropout rate was set to 0.2 as a unanimously accepted standard practice for LSTM layers. Lastly, a fully connected dense layer was used for outputting the predictions. Rectified Linear Activation Function (*ReLU*) was used to appropriately approximate linear functions and learn non-linear dependencies. Since we expected plentiful non-linear relationships and noise in our input data, Adam optimizer was used to efficiently tackle sparse gradients in the standard deep learning optimization method. Adaptive learning rate with decay was added for quicker convergence.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 5, 50)             10400
_________________________________________________________________
dropout (Dropout)            (None, 5, 50)             0
_________________________________________________________________
lstm_1 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 30,651
Trainable params: 30,651
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Descriptive statistics for LSTM-RNN

Averages:
This approach was linear for simplifying our model in scenarios with fewer data available. Here, we modelled the linear relationship of past performance solely based on weighted mean and median values. This approach was used when data points were less than 30. If data points were less than ten, then the standard average of past performances (fantasy points) was used as the prediction for the next match. In cases where data was more, we introduced a weighted average. One term here signified standard average as calculated before, and the other accounted

for recent form. We tried various combinations for weights manually and ended up giving both terms equal weightage for stability purposes.

Results:
Considering our first problem statement and entire methodology for the model designed, we predicted performances for the 2020 IPL season. It should be noted that none of the models considers any data from IPL 2020 itself. That is, we don't add to our model any events from matchday 1 of IPL 2020 to predict matchday 2 performances.

Checking for some hyperparameters was a process of continuous improvement and hence a few results have been included here as well. Firstly, we mapped around 77 players in our dataset eligible for regression analysis (that is, more than 30 matches played for effective training). Our results showed that our data had an average intercept value of 35.53. This means that the mean predicted value, where all other terms in regression having weight=0, will be 35.53. The result was for order *(p)*=4, and highest polynomial order=2. This was true for IPL 2020 and these values will significantly vary for different formats of the game and different intervals of time. The standard deviation of the model came to be around 4 (SD_{Model}= 4). We can also say that fantasy point performance is loosely, normally distributed. The graph shows intercept values for all different players in our dataset. The deepness of the shade is a measure of how close those intercept values are to the mean.

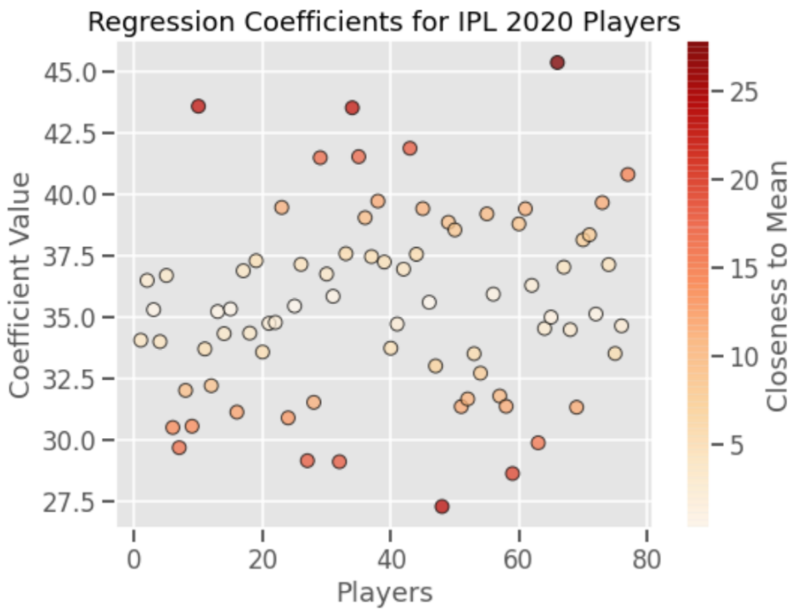


Figure 7: Regression Coefficients Values for Players

Furthermore, it was important to fixate the epoch value for our LSTM model. This was done by seeing the performance of models over randomly chosen clusters of players and aggregating them after which the majority was taken for final deployment. One iteration is visualized, calculated, and shown in table number 5. We were to select the best model from 4 different epoch values of 250, 500, 750, and 1000. Each iteration consisted of 9 players and 20 iterations were performing in this case. For season performance prediction the metric of accuracy for each model was determined by the following formula.

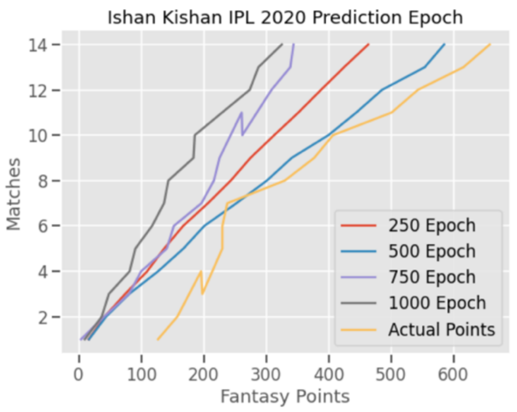
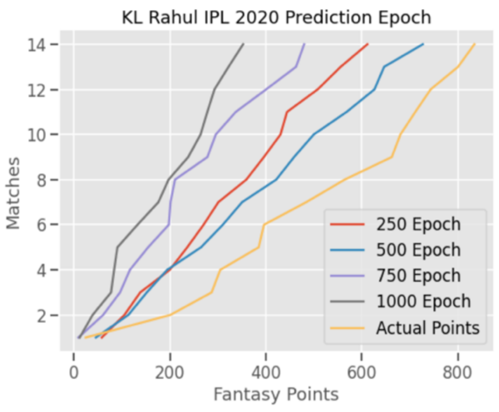
$$A_c = \frac{| E_n - A_{perf} |}{A_{perf}} * 100. \quad (2.2)$$

Where E_n (for example, E_{500}) is the predicted value on a specific epoch and A_{perf} reflects the actual value of that player's performance. The average of this value of A_c was calculated for all epochs and players. The minimum A_c value after averaging was considered the best epoch for that cluster. The mode from all 20 clusters was chosen as the final parameter. A table carrying out the calculations is shown (Table 5). Insights gained for the effect of different epoch values in our LSTM model are also discussed further.

For the table and plotted graphs we can conclude that for this cluster batch the best performing model was the one with 500 epochs. We could observe A_c value = 9.08 as the lowest in this case. This also means that the average error from the actual value was 9.08%. The A_c value at 750 epochs being 16.38 was the second-best performer. For all other clusters we tested, the majority of the times, the model with 500 epochs performed the best (16 out of 20 times) and was chosen as the final value in our deployed model. In the remaining iterations (4 out of 20 times) the 750 epoch model outperformed the 500 epoch model slightly. The other two models (250 epoch and 1000 epoch) seemed to be slightly under-fitting (250 epoch) and over-fitting (1000 epoch) the training data in most cases. This high bias in training (250 epoch) had poor test results in most cases. Similarly, high variance (1000 epoch) in training has its cons and fails to pick up the true non-linear relationships. Hence the range of 500 to 750 for epoch values truly generalize performance data and give the best results. On further scrutiny and trials, we might find a better model in the range (500, 750) but are using 500 epochs as it provides stability and consistency. Hyperparameters can get better if we decide to dig in deeper for any ML model but our goal is to find the hyperparameters that work just fine.

Table 5: Comparison of Epoch Values in Random Cluster of Players

Sr no	**Player Name**	**Actual Score**	**E_{250}**	**A_c250**	**E_{500}**	**A_c500**	**E_{750}**	**A_c750**	**E_{1000}**	**A_c1000**
1	Virat Kohli	**579**	681	17.61	612	5.69	784	35.4	961	65.97
2	Rohit Sharma	**427**	363	14.98	472	10.53	448	4.91	392	8.19
3	KL Rahul	**834**	613	26.49	729	12.58	490	41.24	373	55.27
4	Rashid Khan	**612**	564	7.84	566	7.51	614	0.32	663	8.33
5	Kagiso Rabada	**826**	735	11.01	791	4.23	684	17.19	1086	31.47
6	MS Dhoni	**232**	454	95.68	268	15.51	326	40.51	384	65.51
7	Rishab Pant	**420**	577	37.38	489	16.42	424	0.95	468	11.42
8	Yuzvendra Chahal	**534**	618	15.73	534	0	497	6.92	472	11.61
9	Rashid Khan	**612**	552	9.8	555	9.31	612	0	667	8.98
	Average A_c Values			**26.28**		**9.08**		**16.38**		**29.63**


Ishan Kishan IPL 2020 Prediction Epoch
Matches
Fantasy Points
250 Epoch
500 Epoch
750 Epoch
1000 Epoch
Actual Points

KL Rahul IPL 2020 Prediction Epoch
Matches
Fantasy Points
250 Epoch
500 Epoch
750 Epoch
1000 Epoch
Actual Points

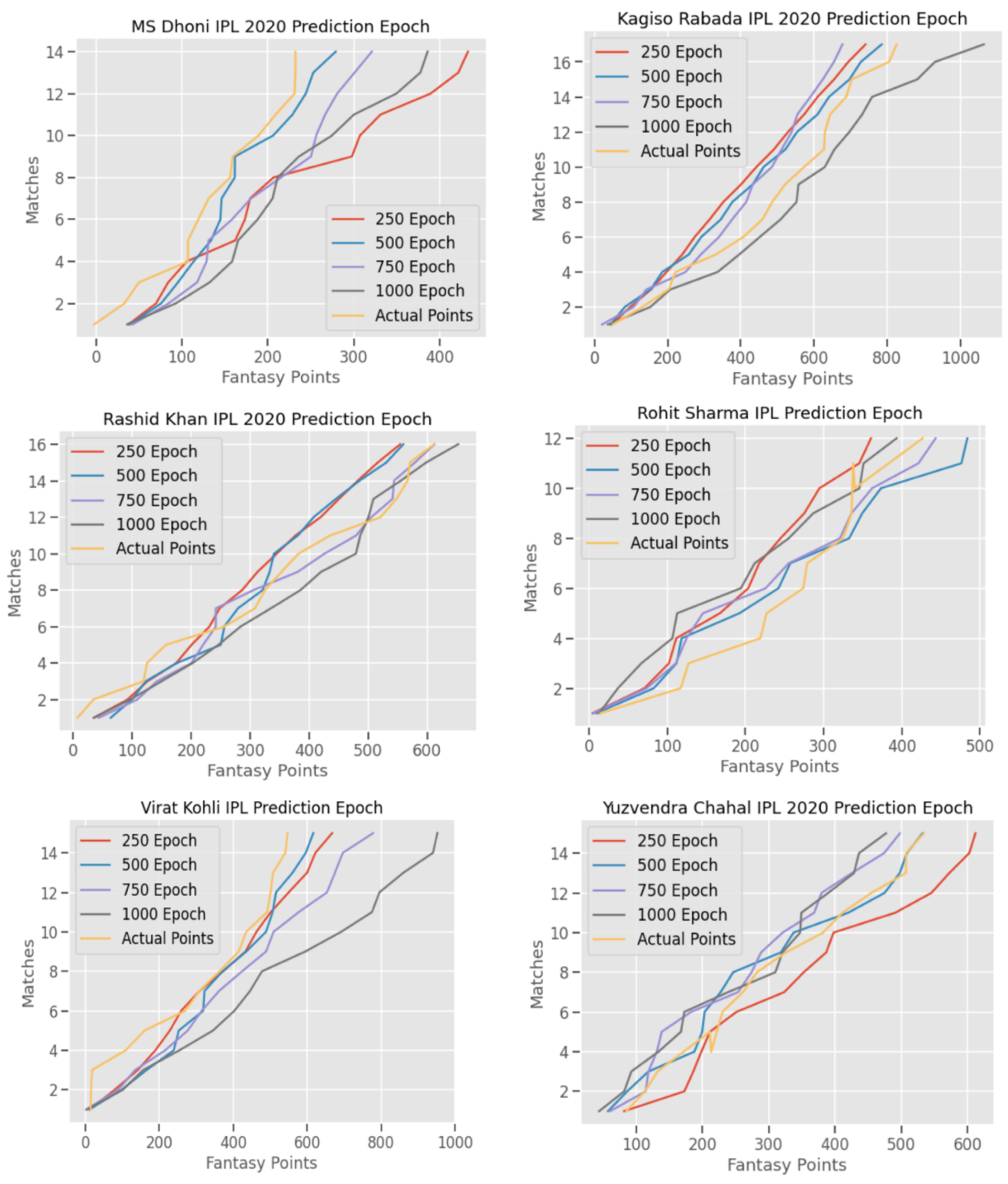


Figure 8: Prediction points for various Cricket players

Finally getting to results, we have used 500 epochs as standard (for players with more than 65 matches played). AR model varied with the degree of the polynomial (1 or 2) depending on the amount of data (25 to 40 matches played for 1 degree and 41 to 64 matches played for 2 degrees). The averages approach was used for player data below 25. If no data was found, 1 was simply returned. Players were also segregated based on performance to test how well our models generalized for all possible cases. The segregation was based on the following conditions stated below. This knowledge comes from comments and analysis from expert pundits, performance statistic reports from reliable websites (like IPL, Espn, etc), and various sports enthusiasts like us.

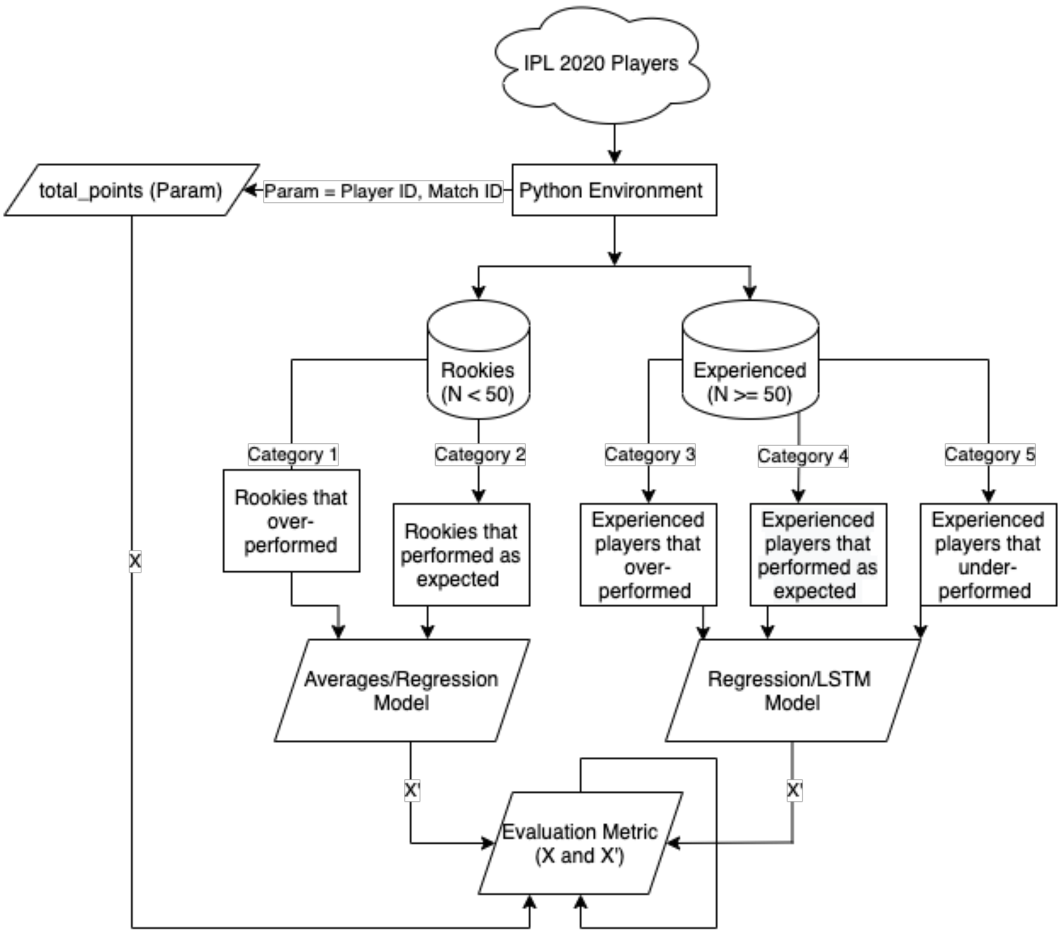

IPL 2020 Players
total_points (Param)
Param = Player ID, Match ID
Python Environment
Rookies (N < 50)
Experienced (N >= 50)
Category 1
Category 2
Category 3
Category 4
Category 5
Rookies that over-performed
Rookies that performed as expected
Experienced players that over-performed
Experienced players that performed as expected
Experienced players that under-performed
X
Averages/Regression Model
Regression/LSTM Model
X'
X'
Evaluation Metric (X and X')

Figure 9: Flowchart to showcase prediction approach used

1. **Rookies that over-performed (Category 1):** Any player with less than one season of experience or total matches played less than 50 is considered a rookie (rookie: a young player with considerably less experience at the highest level). MI's Ishan Kishan, an outstanding young performing batsman in IPL 2020, was chosen here. The criteria for selection were players with more than 15% of total fantasy points from the average expected score to better off the latter. (Fantasy Points> 1.15 x Matches Played in IPL 2020 x Player career average points)

2. **Rookies that performed as expected (Category 2):** This category consists of players that performed up to the mark based on numbers and fan expectations. The criteria for selection were players with not more than 15% deviation of total fantasy points from the average expected score. (0.85 x Matches Played in IPL 2020 x Player career average points <= Fantasy Points <= 1.15 x Matches Played in IPL 2020 x Player career average points)

3. **Experienced players that over-performed (Category 3):** Players that performed better than expected like opening batsman: KL Rahul, spin-bowler: Yuzuvendra Chahal, and fast bowler: Kagiso Rabada who many people bank on as key players in the team. The criteria for selection were players with more than 15% of total fantasy points from the average expected score to better off the latter. (Fantasy Points >= 1.15 x Matches Played in IPL 2020 x Player career average points)

4. **Experienced Players that performed as expected (Category 4):** Players like Rohit Sharma, Virat Kohli, and Rashid khan are included here. These players with high experience and are expected to perform well in every match. The criteria for selection were players with not more than 15% deviation of total fantasy points from the average expected score. (0.85 x Matches Played in IPL 2020 x Player career average points <= Fantasy Points <= 1.15 x Matches Played in IPL 2020 x Player career average points)

5. **Players who underperformed in IPL 2020 (Category 5):** Someone like MS Dhoni, a star of the game for many years, to everyone's surprise, has his worst IPL season ever. He has been a winning captain in 3 IPL seasons for Chennai Super Kings (CSK), only second to Rohit Sharma's 4 IPL trophies with MI. Hence, we have included him as a part of our visualization in the results segment of this paper. The criteria for selection were players with more than 15% deviation of total fantasy points from the average expected score to worse off the latter. (Fantasy Points <= 0.85 x Matches Played in IPL 2020 x Player career average points)

From the histogram mapped in Figure 10, we can see that around 75 players fall into the Rookie category (1st and 2nd bin). 45% of the players in the first bin have either 0 or less than 5 games of experience. Such players have been directly excluded from our analysis simply due to a lack of data. The remaining 55% have been included for the sole purpose of team making. A simple average approach is used for prediction. In the second bin, we have used regression techniques for prediction, and for players, more than 65 matches played LSTM models were introduced as specified. Players falling under the 3rd, 4th, and 5th bin can are labelled as moderately experienced, experienced, and highly experienced respectively. It is to note that the definition of a rookie, experienced player, good and bad performance is critically subject to a difference in opinion. We have heavily relied on statistics to make such decisions and structure our analysis in a certain manner.

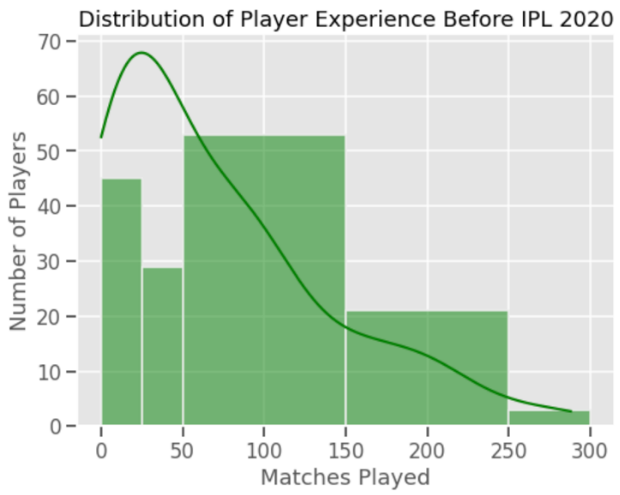


Figure 10: Histogram Plot for Matches Played

Category 1: It is clear that in the case of an unexpectedly good performance or bad performance, the actual points graph significantly differs from the averages graph that we have used. This was observed in almost all rookies who fall in category 1. AR models with 2-degree polynomials could capture this trend of rising form significantly, with players that have 25-50 matches of experience. LSTM model also performed well in capturing this trend with good performance in some cases (like Ishan Kishan) and not so well in others. In this case, a regression model with a specific degree outperformed all other models and hence, with fewer data points, can prove to be the best approach to capture non-linearity.

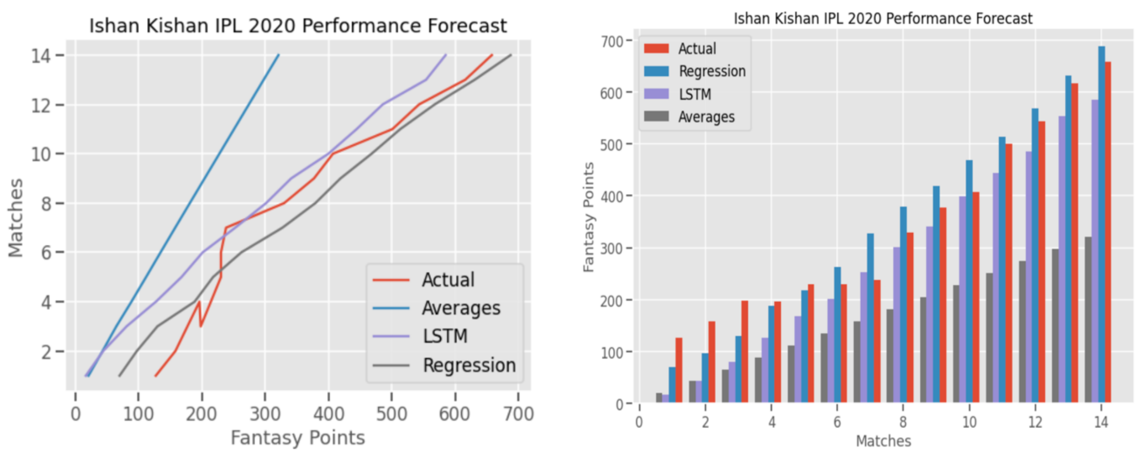


Figure 11: Performance forecast for Category 1 players

Category 2: The averages approach will always have the best total if a player is significantly consistent. In this case also where players who ended up maintaining consistent performance have the closest total to averages. We also observed that the LSTM model performed significantly well with nearly 75% of players in this category to capture trends in performance. However, In over 50% of such cases, it either over-performed/under-performed towards the end or in the start. AR models slightly overperformed in 90% of the cases and were second to LSTM in capturing mid-season trends. Hence, for safe predictions, a mixture of averages and LSTM can prove to be the best solution for category 2.

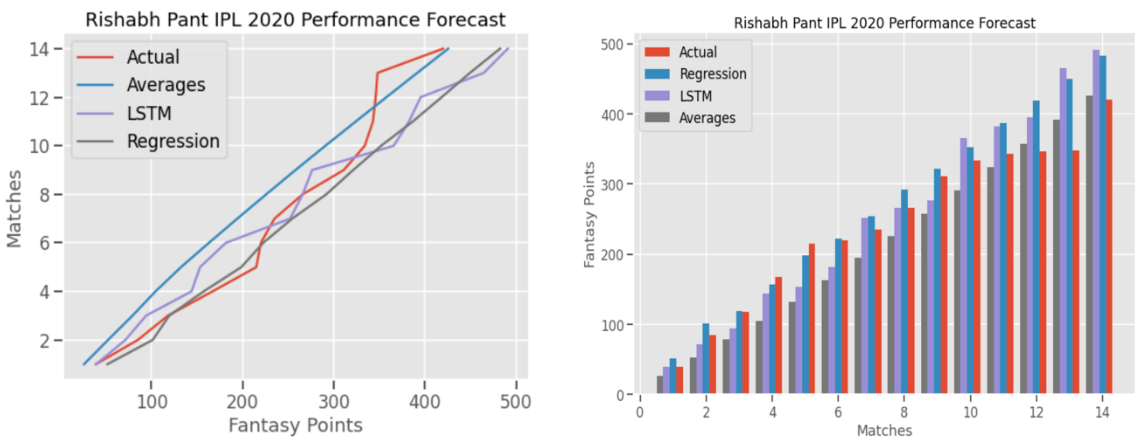


Figure 12: Performance forecast for Category 2 players

Category 3: We can again see how significantly the plot of the averages differs from the actual performance. Hence using averages for predictions is the most basic approach for sports prediction as it never captures non-linear relationships. This category has around twice the number of data points that were available in categories 1 and 2. With more data points, we could observe how LSTM outperforms all other approaches by far in all departments. Trends we captured very well in all cases with errors ranging from 0.1% to 12.58% end-to-end.

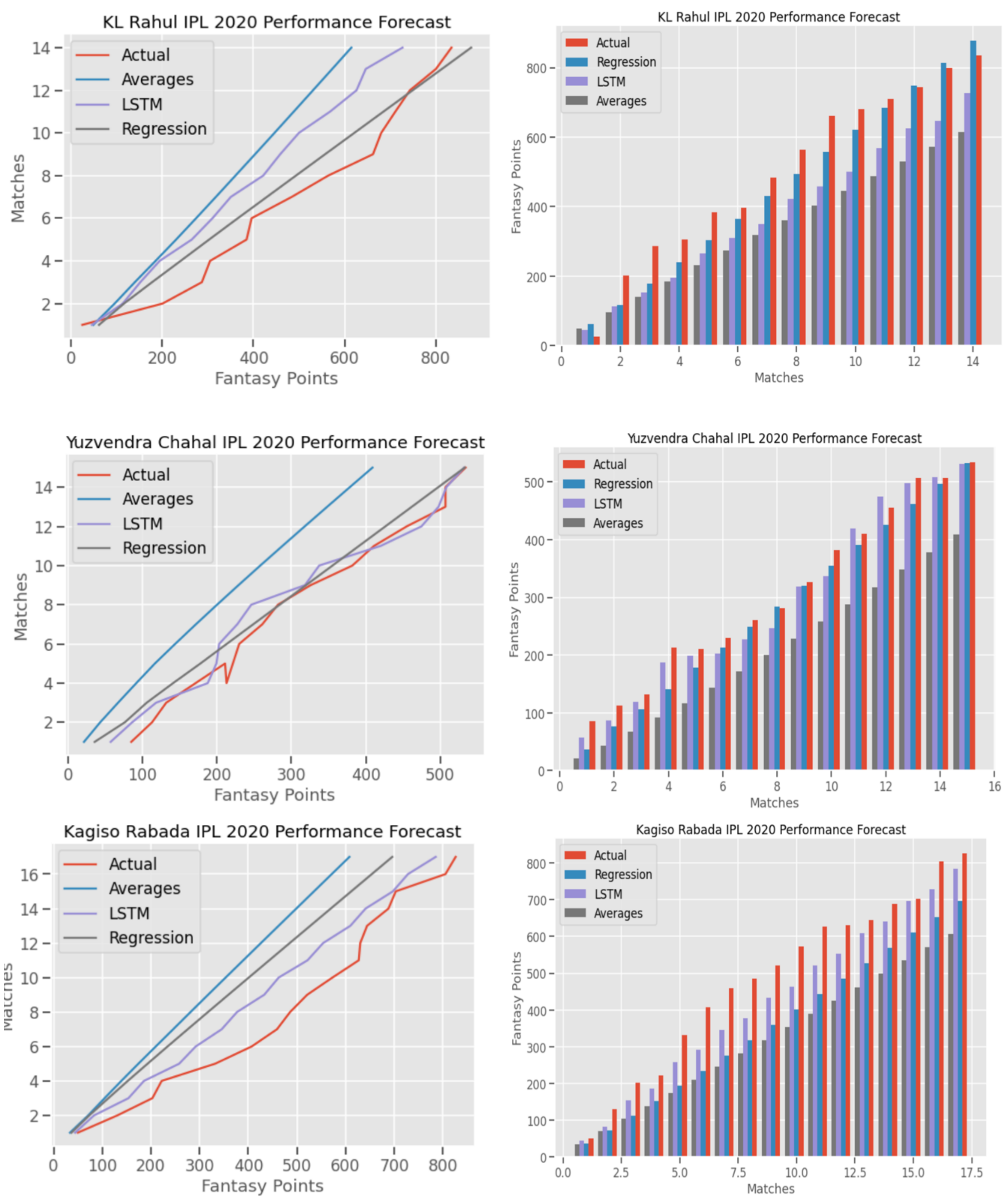


Figure 13: Performance forecast for Category 3 players

Category 4: Players in this category have paramount experience and are considered pivotal to the teams' success. From 76 (Figure 10: bin 3, 4, and 5) players having experience of more than 50 games, around 40 players were segregated in category 4, with the remaining 24 players in category 3 and 12 players in category 5. In summary of our projections, we noticed that roughly all models could capture the total points secured at the end of the season with little error margin and standard error deviation. Consistency and experience are highly correlated to good predictions in this case. Trends showed that the error margin for regression and LSTM models was on average 11.54 and 8.45 per cent respectively for all 40 players showing that throughout a season, these players were the safest to keep in one's fantasy team as they provided stability and consistency.

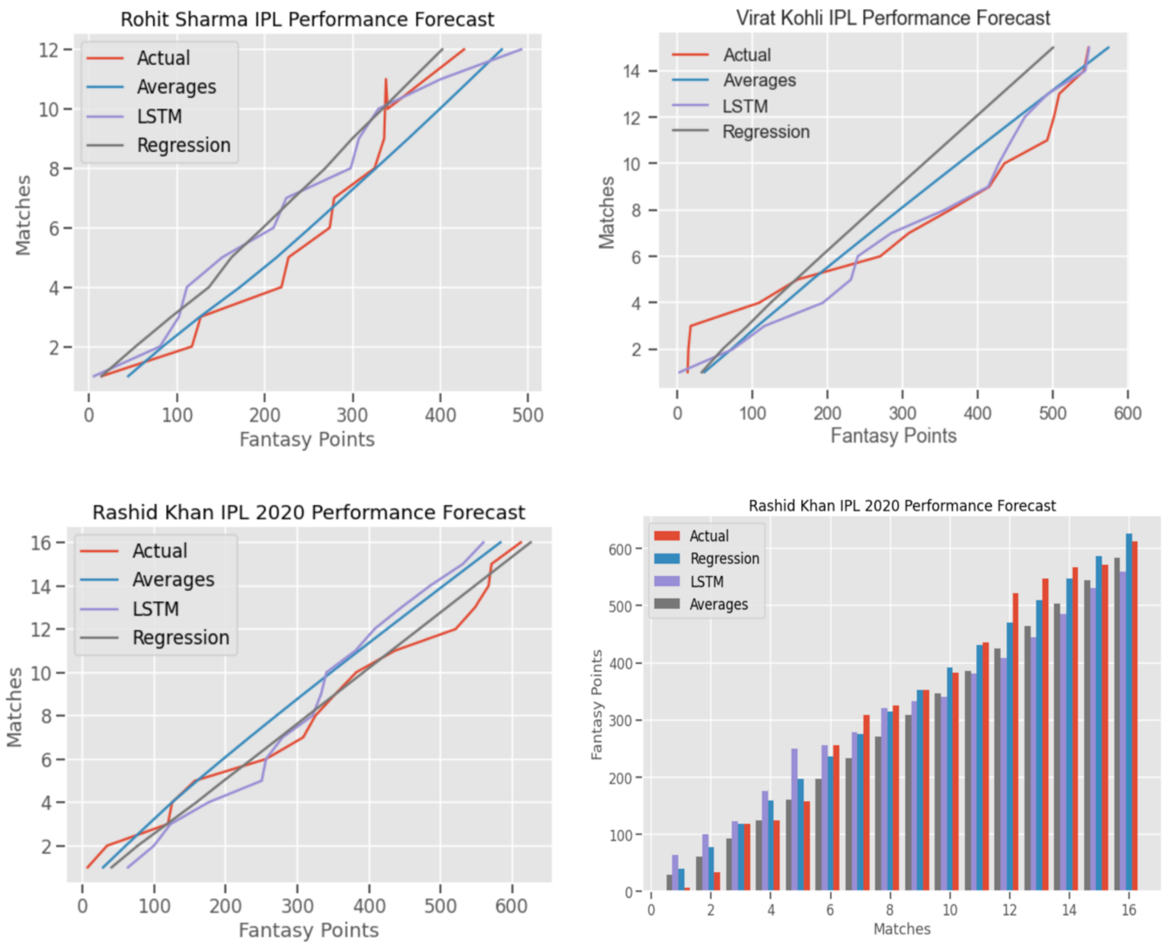
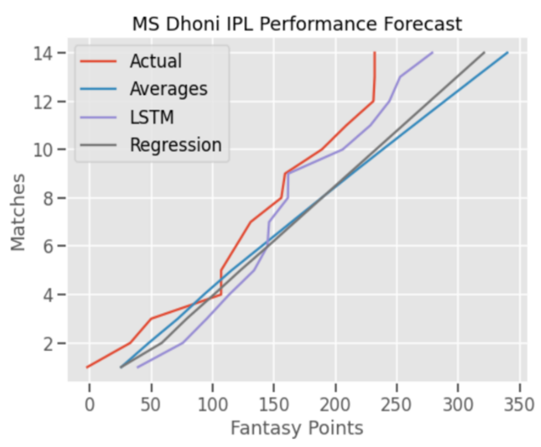


Figure 14: Performance forecast for Category 4 players

Category 5: All performances under this category had subpar results for the regression and averages approach used. The total percentage error grossed over 50% in 7 out of 10 players, and around 17.5-22.5% in the remaining 3 players considered in this category. For LSTM models percentage error ranged between 15.51 - 28.5% with faintly trends captured in between matches. Our best performing LSTM model was with the case of MS Dhoni is included for visualization (15.51% error). It also showed promising trends with the slightest bias towards a higher total (As all players in this category have rich historic data for high scores). We could confirm that poor recent form resulted in the best results for this category. Undoubtedly more data would be required to efficiently analyze such a sudden dip in performance as we faced high bias issues in our training fit that lead to inferior results.



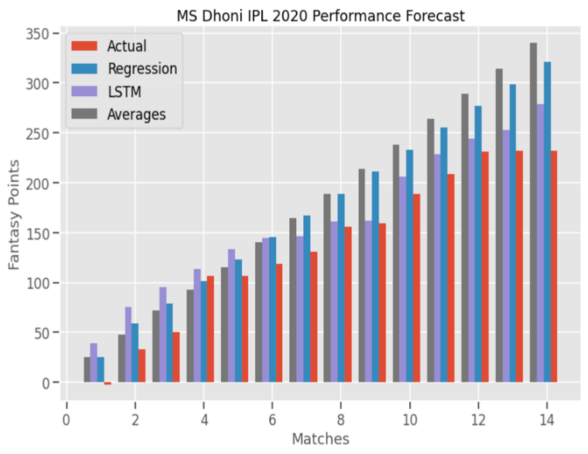


Figure 15: Performance forecast for Category 5

Since we systematically documented our predictions for all the players in IPL 2020, we created a table for the top 5 predicted performers in all 8 participating teams (40 players in total). This information will be vital for making sustainable long-term profits from a gambling perspective or above average results in other fantasy competitions [34]. Tabular results are presented below. We correctly estimated 28/40 players giving us a high accuracy of 70%.

Table 6: Chennai Super Kings Top 5 Actual vs Predicted Players

Chennai Super Kings (CSK)				
	Actual	**Points**	**Predicted**	**Points**
1	Faf du Plessis	557	Sam Curran	603.54
2	Sam Curran	555	Faf du Plessis	582.11
3	Ambati Rayudu	449	Shane Watson	423.43
4	Ravindra Jadeja	446	Deepak Chahar	388.19
5	Shane Watson	382	MS Dhoni	268

Table 7: Mumbai Indians Top 5 Actual vs Predicted Players

Mumbai Indians (MI)				
	Actual	**Points**	**Predicted**	**Points**
1	Jasprit Bumrah	768	Hardik Pandya	780.63
2	Trent Boult	671	Quinton de Kock	614.95
3	Ishan Kishan	658	Krunal Pandya	561.11
4	Quinton de Kock	635	Trent Boult	534.74
5	Suryakumar Yadav	603	Jasprit Bumrah	439.98

Table 8: Royal Challengers Banglore Top 5 Actual vs Predicted Players

Royal Challengers Banglore (RCB)				
	Actual	**Points**	**Predicted**	**Points**
1	AB de Villiers	579	Virat Kohli	739.06
2	Virat Kohli	547	AB de Villiers	524.57
3	Yuzvendra Chahal	534	Yuzvendra Chahal	514.34
4	Washington Sundar	381	Aaron Finch	405.69
5	Chris Morris	355	Washington Sundar	356.79

Table 9: Sunrisers Hyderabad Top 5 Actual vs Predicted Players

Sunrisers Hyderabad (SRH)				
	Actual	**Points**	**Predicted**	**Points**
1	David Warner	680	Jonny Bairstow	747.81
2	Rashid Khan	612	David Warner	684.87
3	Manish Pandey	532	Kane Williamson	621.44
4	Jonny Bairstow	428	Rashid Khan	606.62
5	Jason Holder	427	T Natarajan	399

Table 10: Rajasthan Royals Top 5 Actual vs Predicted Players

Rajasthan Royals (RR)				
	Actual	**Points**	**Predicted**	**Points**
1	Jofra Archer	668	Jos Buttler	765.52
2	Rahul Tewatia	588	Sanju Samson	455.55
3	Sanju Samson	476	Steven Smith	429.61
4	Jos Buttler	407	Rahul Tewatia	429.42
5	Ben Stokes	407	Jofra Archer	426.5

Table 11: Kolkata Knight Riders Top 5 Actual vs Predicted Players

Kolkata Knight Riders (RR)				
	Actual	**Points**	**Predicted**	**Points**
1	Shubman Gill	542	Nitish Rana	594.5
2	Eoin Morgan	534	Varun Chakravarthy	461
3	Pat Cummins	505	Sunil Narine	360.33
4	Varun Chakravarthy	461	Pat Cummins	349.01
5	Nitish Rana	437	Eoin Morgan	328.46

Table 12: Delhi Capitals Top 5 Actual vs Predicted Players

Delhi Capitals (DC)				
	Actual	**Points**	**Predicted**	**Points**
1	Kagiso Rabada	826	Shreyas Iyer	704.34
2	Shikhar Dhawan	777	Kagiso Rabada	695.84
3	Marcus Stoinis	742	Shikhar Dhawan	564.69
4	Shreyas Iyer	627	Rishabh Pant	530.06
5	Anrich Nortje	533	Marcus Stoinis	526.67

Table 13: Kings XI Punjab Top 5 Actual vs Predicted Players

Kings XI Punjab (KXIP)				
	Actual	**Points**	**Predicted**	**Points**
1	KL Rahul	834	KL Rahul	681.06
2	Mayank Agarwal	534	Nicholas Pooran	445.88
3	Mohammed Shami	478	Mohammed Shami	319.37

4	Nicholas Pooran	452	Glenn Maxwell	276.27
5	Chris Gayle	373	Chris Jordan	263.66

For our second segment, where we were predicting the best team on match day was two-folds more complicated. Our defined optimization objective comes in handy to make these predictions possible. Even though we can predict performance throughout the season with considerably high accuracy, the dynamics change when it comes to match-day score predictions. Visualizations for Virat Kohli and Rohit Sharma are included to describe underlying setbacks faced. We can see that for Rohit Sharma, on matchdays 2 and 12 high scores were accurately predicted. This is a desirable outcome but at the same time results seen on matchdays, 4 and 5 are terrible predictions (where our models predicted a low score when the player had a high score and vice versa). Similar flaws can be observed for Virat Kohli on matchdays, 2 and 3. In fact, A string of poor predictions was observed for all players even though these effects mitigate, when we consider total predicted points as discussed earlier in the paper.

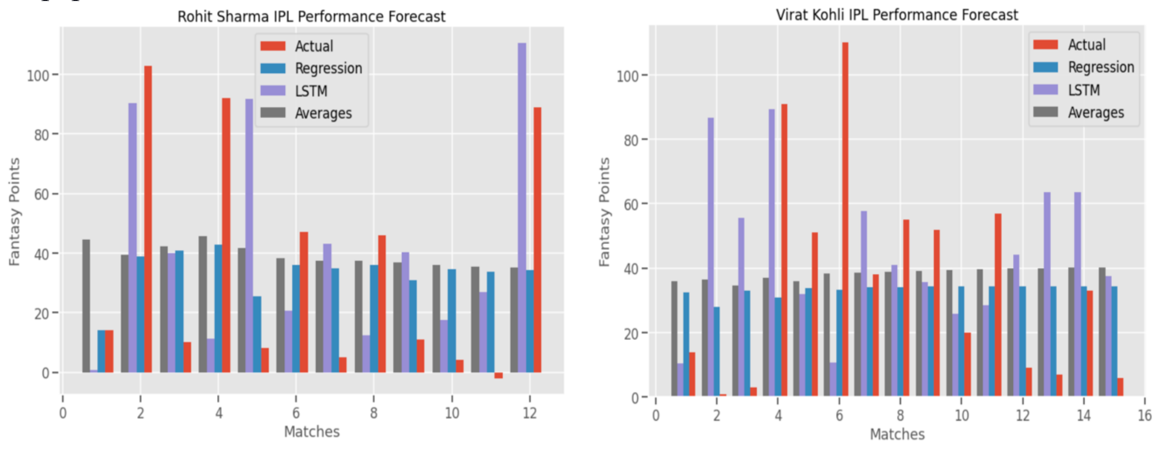
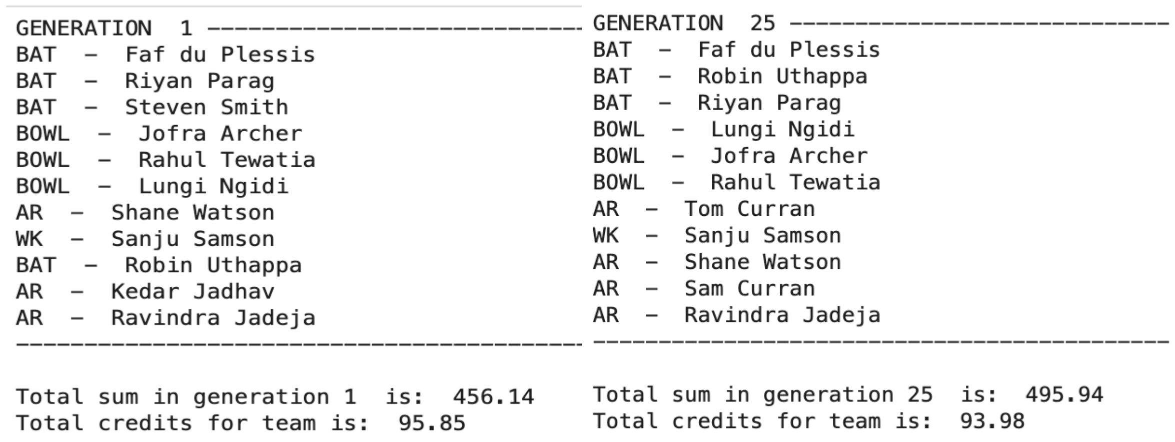


Figure 16: Trends in performance forecasts

Since our end goal after optimization is to predict as many players as possible in the actual dream team, variability in prediction doesn't matter much. For example, if a player scored 150 runs in the match, we predicted his score to be 65 but included him in the dream team then we have a positive outcome (irrespective of more than 50% error in the predicted score). Hence, for players where predictions were made using the LSTM model, we replaced it with a weighted average of all three models (LSTM, Autoregressive, and average) with LSTM having a weightage of 60% and the other two equally sharing 40% (i.e., 20% each). This technique helped us to appease overconfident predictions and raise value if super low scores were predicted. LSTM was given the highest weightage due to its desired high variance and superior capabilities to capture trends. By this method, we are compromising fantasy score accuracy for better final team selection, a justifiable trade-off. Better predictions lead to better optimization as well.

For our results, we predicted dream teams for every match played in IPL 2020. In our worst predictions, we could correctly identify 5 players in the actual dream team. This numeric was 9 in our best predictions. Our mode for correct predictions in each game was 7. In a total of 660 players in dream teams throughout the season (60 x 11) we could accurately predict 404 giving us an accuracy metric of 61.21%. This also means our predicted teams will perform above average 61.21% of the time. We could asset sustained profits ranging from 5% to 17%

could be made with this model throughout the season (variation is based on fielding points, 1.5x, and 2x points for the elected captain and vice-captain that we did not include).



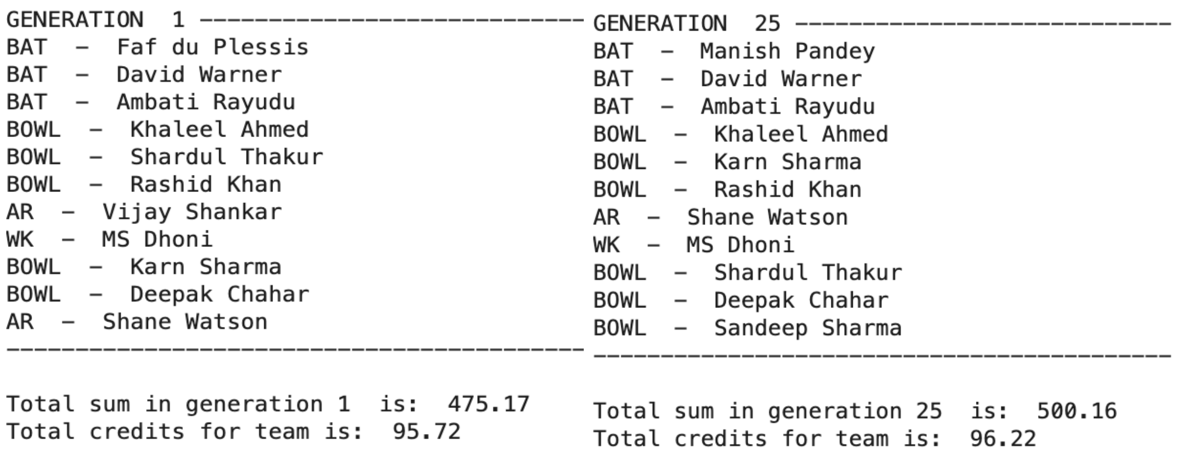
```
GENERATION  1 -----------------------------
BAT  -  Faf du Plessis
BAT  -  Riyan Parag
BAT  -  Steven Smith
BOWL  -  Jofra Archer
BOWL  -  Rahul Tewatia
BOWL  -  Lungi Ngidi
AR  -  Shane Watson
WK  -  Sanju Samson
BAT  -  Robin Uthappa
AR  -  Kedar Jadhav
AR  -  Ravindra Jadeja
------------------------------------------

Total sum in generation 1  is:  456.14
Total credits for team is:  95.85
```

```
GENERATION  25 ----------------------------
BAT  -  Faf du Plessis
BAT  -  Robin Uthappa
BAT  -  Riyan Parag
BOWL  -  Lungi Ngidi
BOWL  -  Jofra Archer
BOWL  -  Rahul Tewatia
AR  -  Tom Curran
WK  -  Sanju Samson
AR  -  Shane Watson
AR  -  Sam Curran
AR  -  Ravindra Jadeja
------------------------------------------

Total sum in generation 25  is:  495.94
Total credits for team is:  93.98
```

Figure 17: Predicted output after GA optimization I (first and last generation)



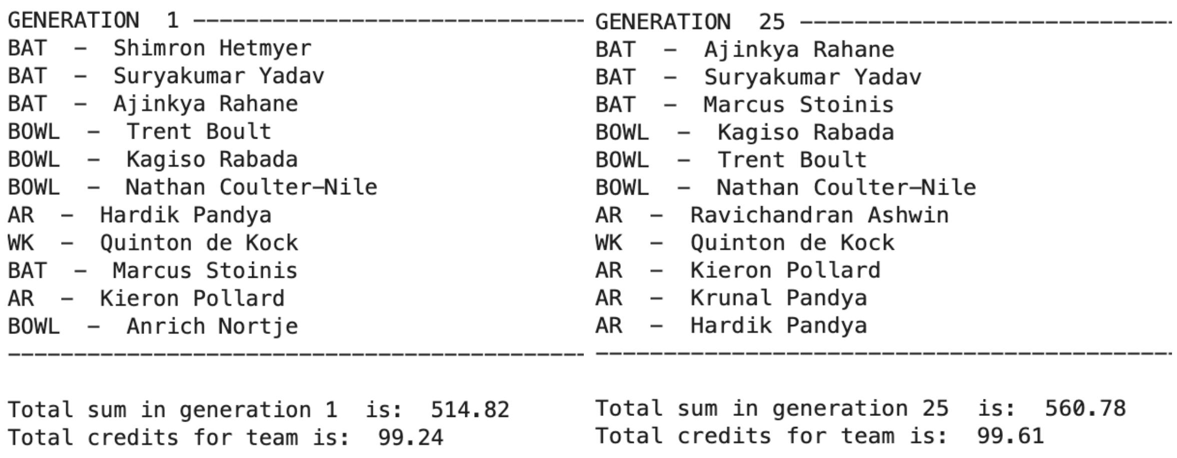
```
GENERATION  1 -----------------------------
BAT  -  Faf du Plessis
BAT  -  David Warner
BAT  -  Ambati Rayudu
BOWL  -  Khaleel Ahmed
BOWL  -  Shardul Thakur
BOWL  -  Rashid Khan
AR  -  Vijay Shankar
WK  -  MS Dhoni
BOWL  -  Karn Sharma
BOWL  -  Deepak Chahar
AR  -  Shane Watson
------------------------------------------

Total sum in generation 1  is:  475.17
Total credits for team is:  95.72
```

```
GENERATION  25 ----------------------------
BAT  -  Manish Pandey
BAT  -  David Warner
BAT  -  Ambati Rayudu
BOWL  -  Khaleel Ahmed
BOWL  -  Karn Sharma
BOWL  -  Rashid Khan
AR  -  Shane Watson
WK  -  MS Dhoni
BOWL  -  Shardul Thakur
BOWL  -  Deepak Chahar
BOWL  -  Sandeep Sharma
------------------------------------------

Total sum in generation 25  is:  500.16
Total credits for team is:  96.22
```

Figure 18: Predicted output after GA optimization II (first and last generation)



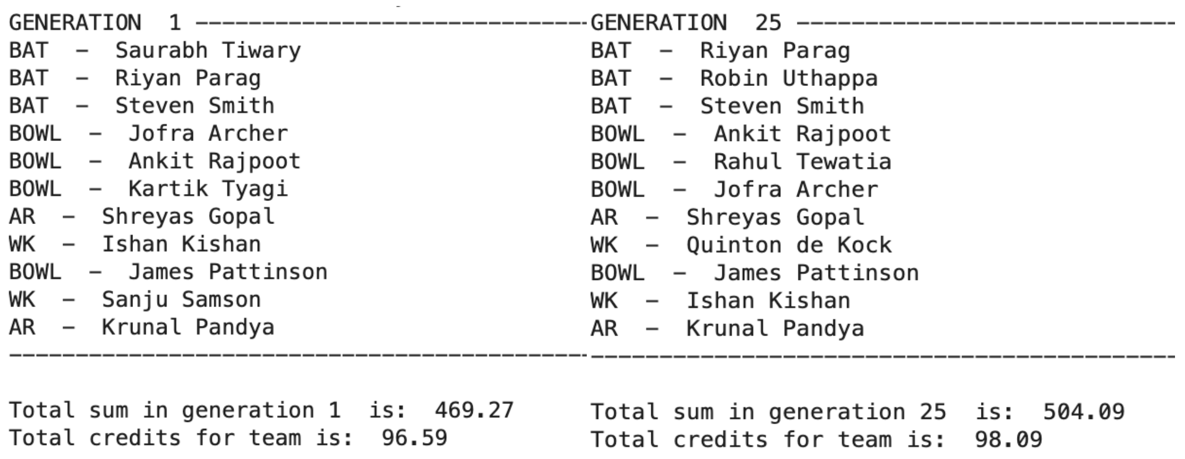
```
GENERATION  1 -----------------------------
BAT  -  Shimron Hetmyer
BAT  -  Suryakumar Yadav
BAT  -  Ajinkya Rahane
BOWL  -  Trent Boult
BOWL  -  Kagiso Rabada
BOWL  -  Nathan Coulter-Nile
AR  -  Hardik Pandya
WK  -  Quinton de Kock
BAT  -  Marcus Stoinis
AR  -  Kieron Pollard
BOWL  -  Anrich Nortje
------------------------------------------

Total sum in generation 1  is:  514.82
Total credits for team is:  99.24
```

```
GENERATION  25 ----------------------------
BAT  -  Ajinkya Rahane
BAT  -  Suryakumar Yadav
BAT  -  Marcus Stoinis
BOWL  -  Kagiso Rabada
BOWL  -  Trent Boult
BOWL  -  Nathan Coulter-Nile
AR  -  Ravichandran Ashwin
WK  -  Quinton de Kock
AR  -  Kieron Pollard
AR  -  Krunal Pandya
AR  -  Hardik Pandya
------------------------------------------

Total sum in generation 25  is:  560.78
Total credits for team is:  99.61
```

Figure 19: Predicted output after GA optimization III (first and last generation)



```
GENERATION  1 -----------------------------GENERATION  25 ---------------------------
BAT  -  Saurabh Tiwary                    BAT  -  Riyan Parag
BAT  -  Riyan Parag                       BAT  -  Robin Uthappa
BAT  -  Steven Smith                      BAT  -  Steven Smith
BOWL  -  Jofra Archer                     BOWL  -  Ankit Rajpoot
BOWL  -  Ankit Rajpoot                    BOWL  -  Rahul Tewatia
BOWL  -  Kartik Tyagi                     BOWL  -  Jofra Archer
AR  -  Shreyas Gopal                      AR  -  Shreyas Gopal
WK  -  Ishan Kishan                       WK  -  Quinton de Kock
BOWL  -  James Pattinson                  BOWL  -  James Pattinson
WK  -  Sanju Samson                       WK  -  Ishan Kishan
AR  -  Krunal Pandya                      AR  -  Krunal Pandya
-------------------------------------------------------------------------------------

Total sum in generation 1  is:  469.27    Total sum in generation 25  is:  504.09
Total credits for team is:  96.59         Total credits for team is:  98.09
```

Figure 20: Predicted output after GA optimization IV (first and last generation)

Selecting the right captain and vice-captain can make a monumental difference, apt selection can have a greater influence than getting 5-6 players right. Our data is restricted to bowling/batting points and its univariate nature limits us to make confident predictions in this regard. Since we are usually getting six players correct, the probability for selecting a captain among these is upwards of 0.54. The probability of selecting both captain and vice-captain among them drops down to 0.27. However, these numbers don't account for the fact that how good the performance was, i.e., we might choose the worst player in the actual dream team as captain/vice-captain, this will have a significant impact too. In retrospect, even after predicting the correct 22 players, the probability of selecting the top two players is 0.002 or 0.2%. Injection of sporting uncertainty makes this even more difficult. Even though not concrete analysis for the selection of captain/vice-captain is added in this research, selecting experienced players with high averages, selecting opening-batsmen and death over bowlers, or selecting one bowler and batsmen as captain and vice-captain can be yielding practices in the world of fantasy sport.

It took us approximately 3 minutes to predict scores and optimize a team of 11 players using the genetic algorithm on our system. Processing time for the same parametric settings could be reduced with enhanced GPU processing capabilities or by using cloud-based compilers. A few snapshots for optimizations are provided below where the first and last generation teams are shown (Figures 17-20). Accuracy metrics and predictions would vary slightly for different tournaments and formats of the game.

Limitations:
Even though we observed positive results in both segments (seasonal and per matchday), we did get subpar results for specific cases, types of players, and score variations. The major underlying issue for this was the lack of data. Inclusion of factors like metrics for form, venue, home/away, pitch type, the strength of the opponent team, and climate conditions could improve results. Lack of defined strategy for choosing captain/vice-captain, as explained above, comes with a fair share of limitations too. For optimization, genetic algorithms are comparatively flexible to implement and understand, unrequired knowledge of differential calculus (unlike other optimization methods like gradient-based optimization) gives it a broader reach. However, issues related to local minimum points have were observed throughout our study. Variability in cricket is also proportional to in-game situations, for example, batsmen tend to score at a higher run rate when chasing a high total, in similar situations, bowlers might concede more runs or take more wickets, it's all too difficult to assess

pre-match. We would also like to see this extended to but not limited to: ODIs and Test matches.

Conclusion:
Selection of the correct players is often the basal feature in the success of the team. It is almost pivotal to accurately assort the players, especially when one indulges in sports betting. This paper reviewed different methods to predict player points and effectively containerised players according to their level of experience.

In this paper, we effectively used historic data pertaining to bowling and batting statistics; modelled PrOBML to not only predict match points but also form the fantasy team. Taking inspiration from the big players in the field of sports betting, we institutionalised scoring points for various matchday events. Incorporating this into different machine learning approaches, points were anticipated. This would be of immense help to predict how much a batsman is going to score pre-match or how would the bowler flair on that particular day.

With the above extensive analysis, we conclude that there is no approach that fetches us a *six* in terms of accurately predicting these match points. Multiple iterations were done in order to realise the most proximal results. LSTM models glimmered in most categories, however, was too optimistic in certain use cases. AR models' aptitude was considerable in Category 1 while taking a simple weighted average was the better way around for players with lesser experience on the field. Players who have a significant history with their positive performance in the past yielded good predictions albeit the approach used. Certain models can be used in lieu of the others subject to parameters which we discussed earlier.

With these points, we can precisely discern at least three out of the top five players in each participating team for the IPL 2020. This is a central statistic in betting on the right horse and can prove to have an empirical advantage for stakeholders.

After the points prognosis, an evolutionary algorithm came into play in persuasively amassing the fantasy team. The best performing players were sifted and we were left the best selection of team players. With an accuracy metric of 61.2%, a user of PrOBML is sure to eliminate the chances of having monetary losses. PrOBML is thus, an effective way to mitigate risks while venturing into the realm of betting. With as many as five players (out of eleven) being accurately predicted in each iteration, there is a statistical advantage that one can recoup the money vested into the betting.

Furthermore, we would encourage incorporating PrOBML in various other sports. In the future, this work can be elaborated to include even further parameters and assigning heavier weightage to the selection of the captains and vice-captains. This also could be insightful in providing batting strategies or bowling orders. As a cessation of the study, sports betting is an enthralling field that is erupting rapidly and with the correct usage of such data-centric models, one can minimise financial risks and not go astray with blunders.

Declaration:
• **Conflict of Interest:** The authors of this research study declare that there is NO conflict of interest.
• **Funding:** Authors of this paper confirm that there is no funding received for this research work.

• **Acknowledgment:** The authors would like to thank the anonymous reviewers and respected editors for taking valuable time to go through the manuscript.

• **Data Availability Statement:** The generated data during and/or analysed during the current study are available from the corresponding author on reasonable request.

References:

[1] Das NS, Usman J, Choudhury D, Abu Osman NA (2014) Nature and Pattern of Cricket Injuries: The Asian Cricket Council Under-19, Elite Cup, 2013. PLoS ONE 9(6): e100028.

[2] KPMG, & Indian Federation of Sports Gaming. (March 20, 2019). Number of users of top fantasy sports companies across India in 2018 (in millions) [Graph]. In *Statista*. Retrieved June 30, 2021, from https://ezproxy.svkm.ac.in:2307/statistics/1065111/india-number-of-users-of-top-fantasy-sports-companies/

[3] G. Kaur and G. Jagdev, "Analyzing and Exploring the Impact of Big Data Analytics in Sports Science," *2020 Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN)*, 2020, pp. 218-224, doi: 10.1109/Indo-TaiwanICAN48429.2020.9181320.

[4] R. White and M. Cheung, "Communication of Fantasy Sports: A Comparative Study of User-Generated Content by Professional and Amateur Writers," in *IEEE Transactions on Professional Communication*, vol. 58, no. 2, pp. 192-207, June 2015, doi: 10.1109/TPC.2015.2430051.

[5] M. J. Hossain, M. A. Kashem, M. S. Islam and M. E-Jannat, "Bangladesh Cricket Squad Prediction Using Statistical Data and Genetic Algorithm," *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, 2018, pp. 178-181, doi: 10.1109/CEEICT.2018.8628076.

[6] Shubham Agarwal, Lavish Yadav, Shikha Mehta, Cricket Team Prediction with Hadoop: Statistical Modeling Approach, Procedia Computer Science, Volume 122, 2017, Pages 525-532, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.11.402.

[7] Iyer, Subramanian & Sharda, Ramesh. (2009). Prediction of athletes performance using neural networks: An application in cricket team selection. Expert Syst. Appl.. 36. 5510-5522. 10.1016/j.eswa.2008.06.088.

[8] Prashant Premkumar, Jimut Bahan Chakrabarty, Shovan Chowdhury, Key performance indicators for factor score based ranking in One Day International cricket, IIMB Management Review, Volume 32, Issue 1, 2020, Pages 85-95, ISSN 0970-3896, https://doi.org/10.1016/j.iimb.2019.07.008. (https://www.sciencedirect.com/science/article/pii/S0970389617300836)

[9] R. Chaudhary, S. Bhardwaj and S. Lakra, "A DEA Model for Selection of Indian Cricket Team Players," *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 224-227, doi: 10.1109/AICAI.2019.8701424.

[10] M. M. Hatharasinghe and G. Poravi, "Data Mining and Machine Learning in Cricket Match Outcome Prediction: Missing Links," *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 2019, pp. 1-4, doi: 10.1109/I2CT45611.2019.9033698.

[11] Thenmozhi, D. & Palaniappan, Mirualini & Sakthi, S.M.Jai & Vasudevan, Srivatsan & Kannan, V & Sadiq, S. (2019). MoneyBall - Data Mining on Cricket Dataset. 1-5. 10.1109/ICCIDS.2019.8862065

[12] Nazim Razali, Aida Mustapha, Faiz Ahmad Yatim and Ruhaya Ab Aziz, "Predicting Football Matches Results using Bayesian Networks for English Premier League

(EPL)”, International Conference on Material Science and Engineering, vol. 226, no. 1, pp. 012099:1–6, 2013.]

[13] Sarlis, Vangelis & Tjortjis, Christos. (2020). Sports analytics – Evaluation of basketball players and team performance. Information Systems. 93. 101562. 10.1016/j.is.2020.101562

[14] J. R. Landers and B. Duperrouzel, "Machine Learning Approaches to Competing in Fantasy Leagues for the NFL," in IEEE Transactions on Games, vol. 11, no. 2, pp. 159-172, June 2019, doi: 10.1109/TG.2018.2841057.

[15] Euman, Rob & Abdelnour Nocera, José. (2013). Data Visualisation, User Experience and Context: A Case Study from Fantasy Sport. 8006. 146-155. 10.1007/978-3-642-39265-8_16.

[16] Xiangxue, W., Lunhui, X. & Kaixun, C. Data-Driven Short-Term Forecasting for Urban Road Network Traffic Based on Data Processing and LSTM-RNN. *Arab J Sci Eng* 44**,** 3043–3060 (2019). https://doi.org/10.1007/s13369-018-3390-0

[17] M. A. Istiake Sunny, M. M. S. Maswood and A. G. Alharbi, "Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model," *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 87-92, doi: 10.1109/NILES50944.2020.9257950

[18] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735

[19] Thapa, K.N.K., Duraipandian, N. Malicious Traffic classification Using Long Short-Term Memory (LSTM) Model. *Wireless Pers Commun* (2021). https://doi.org/10.1007/s11277-021-08359-6

[20] Bisong E. (2019) Linear Regression. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA.

[21] A. Santra, A. Sinha, P. Saha and A. K. Das, "A Novel Regression based Technique for Batsman Evaluation in the Indian Premier League," *2020 IEEE 1st International Conference for Convergence in Engineering (ICCE)*, 2020, pp. 379-384, doi: 10.1109/ICCE50343.2020.9290569.

[22] M. S. Acharya, A. Armaan and A. S. Antony, "A Comparison of Regression Models for Prediction of Graduate Admissions," *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, 2019, pp. 1-5, doi: 10.1109/ICCIDS.2019.8862140.

[23] R. Stanojevic and L. Gyarmati, "Towards Data-Driven Football Player Assessment," *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, pp. 167-172, doi: 10.1109/ICDMW.2016.0031.

[24] Katoch, S., Chauhan, S.S. & Kumar, V. A review on genetic algorithm: past, present, and future. *Multimed Tools Appl* **80,** 8091–8126 (2021).

[25] Sloan, Phillip, "Darwin: From *Origin of Species* to *Descent of Man*", *The Stanford Encyclopedia of Philosophy* (Summer 2019 Edition), Edward N. Zalta (ed.)

[26] Lingaraj, Haldurai. (2016). A Study on Genetic Algorithm and its Applications. International Journal of Computer Sciences and Engineering. 4. 139-143.

[27] Kumar S G, Varun & Panneerselvam, Ramasamy. (2017). A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator. International Journal of Computational Intelligence Research. Volume 13. Page 1717-1733.

[28] Haiping Ma, Haoyu Wei, Ye Tian, Ran Cheng, Xingyi Zhang, A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints, Information Sciences, Volume 560, 2021, Pages 68-91, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2021.01.029.

[29] Deb K. (2011) Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. In: Wang L., Ng A., Deb K. (eds) Multi-objective Evolutionary Optimisation for Product Design and Manufacturing. Springer, London. https://doi.org/10.1007/978-0-85729-652-8_1

[30] Faez Ahmed, Kalyanmoy Deb, Abhilash Jindal, Multi-objective optimization and decision making approaches to cricket team selection, Applied Soft Computing, Volume 13, Issue 1, 2013, Pages 402-414, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2012.07.031.(https://www.sciencedirect.com/science/article/pii/S1568494612003584)

[31] Yusliza Yusoff, Mohd Salihin Ngadiman, Azlan Mohd Zain, Overview of NSGA-II for Optimizing Machining Process Parameters, Procedia Engineering, Volume 15, 2011, Pages 3978-3983, ISSN 1877-7058

[32] Xin-She Yang, Chapter 14 - Multi-Objective Optimization, Editor(s): Xin-She Yang, Nature-Inspired Optimization Algorithms, Elsevier, 2014, Pages 197-211, ISBN 9780124167438, https://doi.org/10.1016/B978-0-12-416743-8.00014-2

[33] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.

[34] Levitt, S.D. (2004), Why are gambling markets organised so differently from financial markets?*. The Economic Journal, 114: 223-246.

[35] Varsamopoulos, Savvas & Bertels, Koen & Almudever, Carmen. (2018). Designing neural network based decoders for surface codes.