# CricPredict: Resource-Aware Prediction of T20 Cricket Match

Ashish Kumar
School of Electronic Engineering and Computer Science
Queen Mary University
London, United Kingdom

Bilal Hassan
School of Computing and Digital Media
London Metropolitan University
London, United Kingdom

Muhammad Farooq Wasiq
Department of Creative Technologies
Air University
Islamabad, Pakistan

***Abstract*— One of the key problems in cricket is the increasing number of abandoned matches due to unusual circumstances. There is a total of three different formats in cricket e.g., Test, ODI and T20 international. Usually, the Duckworth–Lewis (D/L) method is used to decide the outcome of the match in Test and ODI cricket, resulting in favour of one team like completed matches. In contrast to the traditional D/L method, we tried to incorporate players' performance indicators into our proposed architecture despite the traditional D/L method which only includes the current state of the match and determines the outcome. To accomplish this task, we tried multiple different machine learning techniques e.g., Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naïve-Bayes, Linear Regression and Polynomial Regression and a deep learning model to predict the outcome of the match. To train and validate our developed architecture, we crawled data from the Indian Premier League (IPL) for the completed matches. Our proposed architecture takes complete matches as input and for the second batter, it predicts outcome at intermediate stages of matches. Later, the performance of our proposed architecture is computed using different performance indicators e.g., accuracy, Mean squared error etc. In our opinion, our proposed resource-aware prediction architecture is a unique contribution of its kind in the field.**

***Keywords*— *Cricket, Predictor, SVM, Naïve Bayes, K-NN, CNN, regression***

## I. Introduction

Cricket is a game where two teams each having eleven players compete against each other. In modern cricket, the ODI, T20, and Test formats are the most common. One of the most famous and shortest playing format of cricket is T20 which involves 20 overs each side. Although there are other T20 premier leagues in the globe, this study focuses on the biggest and most well-known one, the Indian Premier League. The question "What happens if the match stops unexpectedly due to bad playing conditions?" served as the basis for the concept of the project. How can we acquire a result under these circumstances, and how should it affect the scores? The D/L (Duckworth-Lewis) technique is a conventional approach that was first presented in 1997 and granted formal certification in 1999 It states that a team's ability to score runs at any time during the game may be determined by the combination of two resources that are available to them: the number of wickets remaining and the number of overs available to play. However, many other aspects can influence the game that are not considered in the usual fashion, such as pitch, past performance, and so on, which is why this method is also contentious. Sports analytics has become a popular research subject in data science since the development of deep learning and artificial intelligence. In particular, for the Indian Premier League (IPL), the goal of this research project is to provide a result and score prediction technique for T20 cricket matches as a backup to the D/L approach if play is suspended due to unfavourable playing circumstances. The model created for this project will take into account several input variables, including the pitch, toss, number of wickets lost, and past performance of both teams. The approach was developed expressly to address the unpredictable nature of modern cricket. i.e., Twenty-Twenty (T20) focusing exclusively on the Indian Premier League.

In their paper "Score and Winning Prediction in Cricket using Data Mining", Tejender, Vishal, and Prateek (2015) [1] introduced the linear regression method and the Nave Bayes classifier, although just for ODI matches [2] without considering the team's past performance or the toss. In the paper titled, "Outcome Prediction of ODI Cricket Matches Using Decision Trees and MLP Networks," the authors Jalaz, Rajeev, and Pushpender presented their work using multi-layer perceptron networks and decision trees. However, this algorithm turned out to be very sensitive to feature scaling and also became unstable even with small changes. M. Bailey and S.R. Clark [4] used multiple linear regressions to determine the probability of victory of opposing sides. Using a trial of 100 finished matches played in 2005, they built a regression model that correctly predicts the winning side 71% of the time. However, they didn't use any deep learning techniques, which may have increased the accuracy.

In their work "Duckworth-Lewis-Stern Method Comparison with Machine Learning Approach," Kumail and Sajjad [5] presented a machine learning method utilizing data from 3,470 ODI matches sourced from the CricInfo website. While this approach proved to be more accurate due to its use of an unpredictability index, it failed to account for performance differences between the teams. In recent years, academics have tried to address the issue and enhance score prediction [6] capabilities using data mining and machine learning technologies. To address the problem of ties in competition standings and quantify team strength, Basil, Greg, and Tim [7] proposed an extension of the D/L approach in 2001 to assess the extent of victory in one-day cricket. They introduced various covariates, transformed variables, and applied different match weightings, enabling a more equitable assessment of each team's strength. In 2005, Bailey M. [8] pioneered an empirical approach to accurately predict winning and losing outcomes in sports by incorporating team and player-specific data. This methodology accounted for a significant portion of variance—often exceeding 50%—in these outcomes. Over time, as computational capabilities advanced, larger datasets not only became invaluable but also opened up a vast field for exploration. This necessitated the application of systematic models after analyzing 2,200 One Day Internationals (ODIs) played to date.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

A. Tripathi et al. [9] presented the clustering method in the context of cricket score prediction in 2016. It was based solely on the teams' historical results and the locations of previous matches, which did not give a complete picture of the match because it ignored the circumstances of the match at hand. In 2017, Pranavan et al. [10] used the Support Vector Machine (SVM) approach to study the optimal set of qualities that have a strong influence on the match's outcome. However, their study lacked data to demonstrate a link between a team's winnability and the performance of individual players. In 2018, S. Agrawal et al. explored winner prediction by leveraging historical data and applied a diverse set of algorithms including Support Vector Machine, CTree, and Naïve Bayes [11]. It was mentioned in their paper that although adding the features (other available data fields) like weather conditions, toss outcomes and current run rates would further advance their model toward perfect learning, but it will just deepen us into more deep waters as they are hidden. Their main contribution within the research paper entitled "ICC T20 Cricket World Cup 2020 Winner Prediction Using Machine Learning Techniques" [1] was to present four kinds of key machine learning algorithms: Random Forest, Extra Trees, ID3, and C4.5. These are based on a dataset taken from CricInfo, but they did miss some very important numbers which could affect their predictions. In 2020, Nikhil et al. performed score prediction using linear regression, lasso model and ridge models [13]. The dataset was passed through different classifiers such as SVC, decision tree and random forest to predict the result of games based on only team information. However, this approach left out crucial features such as pitch and weather conditions that could potentially increase the predictive accuracy. Although, previous research has been done using machine learning on T20 matches related to traditional features but currently deep NNs have gained attention as they can accept large feature vectors [14-16].

## II. METHODOLOGY

The dataset was then assembled using a self-designed web crawler to scrape the data from espncricinfo.com, with an existing dataset taken from Kaggle. While previous studies have focused solely on predicting the outcome of a match, this dataset and system allow for the anticipation of not only the result of a partial match but also the score after each ball. As we aim to create an alternative to the D/L method, predictions are made once the second inning has started, with an Excel pivot table being used to calculate and store the first innings' score in the main dataset. In this project, the following kind of architecture is utilized.

### *A. Data pre-processing*

The match winner and runs scored were used as the outcome variables. On the other hand, the raw data collected using web scraping was combined with Kaggle dataset using the approach of match IDs. This data was then pre-processed/cleaned to generate a feature set – the purpose of which was to serves as input for the analysis. Later, the ordinal encoder was used to normalize and encode the new dataset. This dataset includes 754 matches from Indian Premier League season (2008-2019) and it contains approximately 86,000 data points. It is divided into training (75%) and testing (25%) subsets for further analysis, as illustrated in Fig. 2. The complete preprocessing pipeline is available in Fig.3.

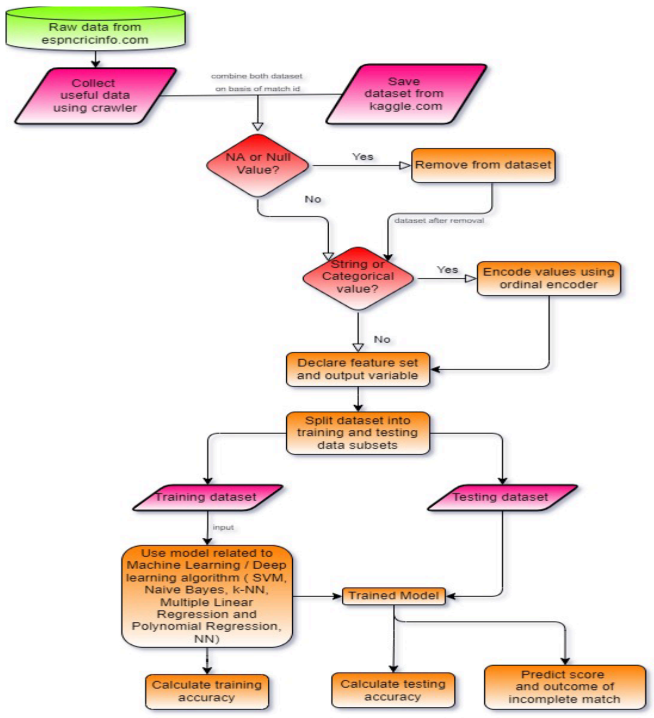

Raw data from espncricinfo.com
Collect useful data using crawler
combine both dataset on basis of match id
Save dataset from kaggle.com
NA or Null Value?
Yes
Remove from dataset
No
dataset after removal
String or Categorical value?
Yes
Encode values using ordinal encoder
No
Declare feature set and output variable
Split dataset into training and testing data subsets
Training dataset
Testing dataset
input
Use model related to Machine Learning / Deep learning algorithm ( SVM, Naive Bayes, k-NN, Multiple Linear Regression and Polynomial Regression, NN)
Trained Model
Calculate training accuracy
Calculate testing accuracy
Predict score and outcome of incomplete match

Fig. 1. Project Architecture

For score forecasting,

| **Feature Set** |
|---|
| season, venue_code, match_id, innings, ball, batting_team_code, bowling_team_code, striker_code, non_striker_code, bowler_code, run_1st_inning, out_1st_inning, out, Winner_Match_code, Winner_toss_code |
| **Outcome Variable** |
| runs |

For winner prediction,

| **Feature Set** |
|---|
| season, venue_code, match_id, innings, ball, batting_team_code, bowling_team_code, striker_code, non_striker_code, bowler_code, runs, out, run_1st_inning, out_1st_inning, Winner_toss_code |
| **Outcome Variable** |
| Winner_Match_code |

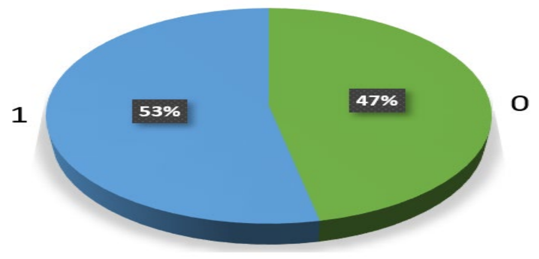


Fig. 2. Match Winner Summary

0: Win percentage for the team batting first (47%)

1: Win percentage for the team batting second (53%)

## *B. Models for Predicting Scores*

The estimation of scores is a regression-type problem in which the strength of the relationship between the independent and dependent variables must be ascertained. The specifics of the batter, bowler, pitch, over, first innings score, toss winners, etc., all have a significant impact on the number of runs scored in this scenario. Thus, the models used to address this problem are,

**Multiple Linear Regression**: It examines the relationship between an independent and a dependent variable, directly leading to predictions using historical data.

**Polynomial Regression**: Polynomial regression uses nth degree polynomials to represent the relationship between the independent variable (x) and the dependent variable (y), which creates a non-linear relationship between the output and the predictors. In order to make this happen, I had used the Polynomial Features module of scikit-learn to create polynomial features and they were then leveraged using these parameters for estimating a high-degree polynomial regression that was after all performed as linear regression.

Metrics used to evaluate different regression models,

- Mean squared error (MSE)
- Root mean square error (RMSE)
- Coefficient of determination (R2)

## *C. Models used for predicting the outcome of the match*

The outcome of an interrupted cricket game is complex to foresee like predicting the winner of a debate with multiple speakers yet to argue. Whether the side batting initially emerges victorious or the team batting later depends on the run total in the first innings along with particulars of the bowlers and batsmen deployed, as well as the team's past performances. Consequently, addressing this multifaceted issue necessitates models examining run tallies, player data, and records to deduce if the team batting first or second will carry the contest once play resumes. Thus, the models used to address this problem are:

**Support Vector Machine:** Support Vector Machine identifies optimal hyperplanes in multidimensional feature spaces to divide cases into target categories. This supervised learning technique is well-suited for problems with a small number of attributes yet frequently encounters issues as dimensionality increases. The current experiment incorporates fifteen features encompassing a wide array of metrics to delineate patterns within the data. SVM determines each feature's contribution by mapping instances as points spanning numerous axes across space. It then calculates the ideal boundary splitting classes as widely as possible, maximizing margins between parallel hyperplanes on either side for enhanced generalization. While hyperplanes offer an intuitive depiction, the actual separating structure may exist in a radically higher dimensional space mapped by kernel functions.

**Gaussian Naïve Bayes:** This algorithm makes strong assumptions that the presence or absence of a specific attribute of a class is independent of other attributes. These classifiers assume that each feature contributes independently to the probability of an item belonging to a class, regardless of the values of other features. Despite its simplicity, Naïve Bayes classifiers are highly effective, particularly in supervised scenarios with labeled training data. It is a common practice to assume that the values for each class follow a normal distribution when dealing with continuous numeric data,.

**K-Nearest Neighbour (KNN):** The KNN is a supervised learning method which identifies a predefined number of training examples closest in distance to a new data point and predicts the label based on these "k" nearest examples. It operates by computing the Euclidean distance between the new point and existing cases, selecting those with the shortest distances. The label is determined by the most frequent class among the k nearest neighbours. In this project, 75% of the dataset is used as reference points, while the remaining 25% is treated as unknown data, with predictions made by majority vote from its 75 nearest samples.

**Deep learning with three-layered neural networks:** The neural network used here consists of three layers of interconnected nodes in a feedforward architecture, with 15 input attributes and two possible outputs. It features 400 neurons in the first hidden layer, 200 in the second, and 100 in the final output layer, as shown in the Fig. 4. The data flow through the neural network is shown in the diagram above. The network begins with 15 input features, followed by the application of the ReLU activation function, batch normalization, and dropout to prevent overfitting. The output from the 400 neurons in the first hidden layer is fed into the second layer, which then produces 200 neurons' output. This output is subsequently used as input for the final layer, resulting in 100 neurons' output, which ultimately classifies the data into two classes (0 or 1). Optimizers are crucial for model performance, as they manage key aspects of neural networks, such as weight and learning rate, to minimize loss. By refining the training process and selecting the most suitable optimizers for the prediction task, we can further improve model performance. We have evaluated the following optimizers for this task:

- SGD Optimizer
- AdaGrad Optimizer
- AdaDelta Optimizer

Metrics used to evaluate different classification models,

- Accuracy
- Confusion matrix
- Loss function

## III. METHODOLOGY

When building the model to predict the score, we first evaluated a linear regression model with a single feature and response variable, however, the mean squared error was 1723.8. We used multiple linear regression model with fifteen predictors and one output variable that provided a significant degree of improvement. While the association between these variables and the outcome may seem straightforward, the reality is more complex. Both linear and nonlinear models were examined to elucidate their relationship. A quadratic approach showed promise in fitting the training information yet struggled markedly with unseen data, reflecting overfitting. A cubic form provided a preferable balance, depicting the design adequately without being overly specific. Processing demand and validation performance were also weighed to find the most prudent solution. In conclusion, a third-degree polynomial sufficiently captured the trend without the drawbacks of higher-order fits or simpler linear assumptions.

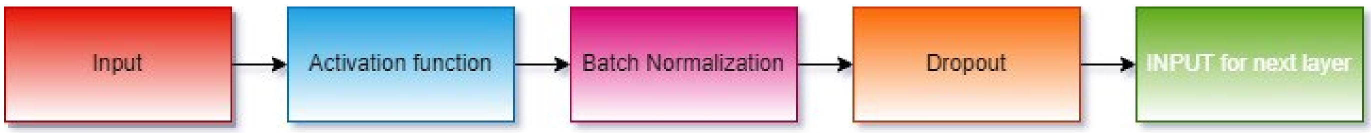


Fig 3. Processing Pipeline

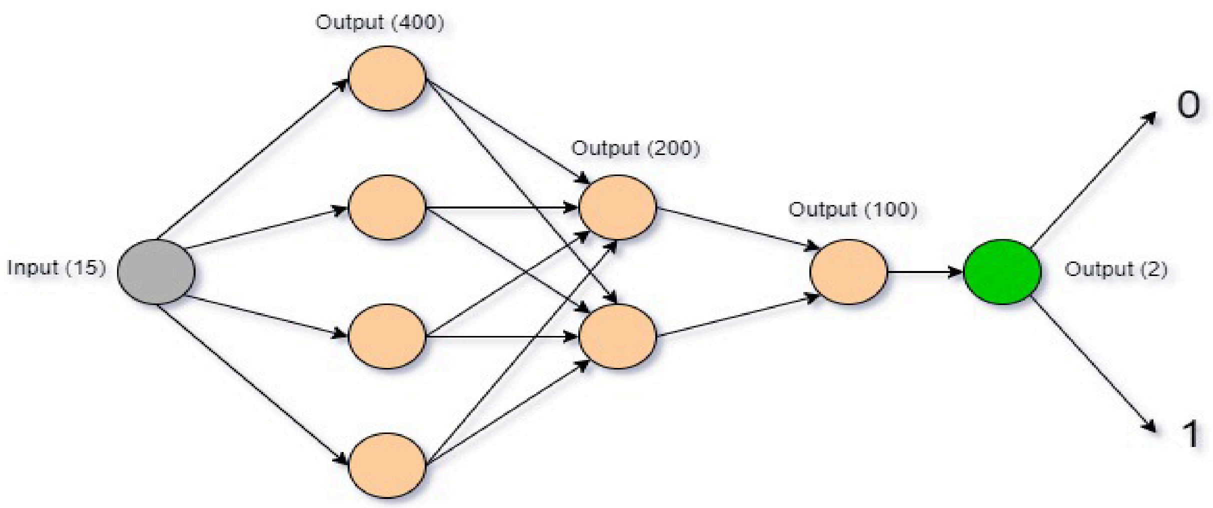


Fig 4. Flow chart of 3-layered NN

TABLE I. ERROR COMPARISON

| Model | RMSE | MSE | $R^2$ |
|---|---|---|---|
| Multiple Linear Regression | 12.579041 | 158.232279 | 0.926096 |
| Polynomial Regression (deg = 3) | 9.721125 | 94.500277 | 0.955862 |

We also attempted to predict the score using the Support Vector Regressor, but it returned a negative coefficient of determination. In the end, the results of all the experiments indicated that the polynomial regression model is the most appropriate for the score estimation problem statement. In Fig 5, the forecast score is shown using one match from test data through polynomial regression with a degree of 3.

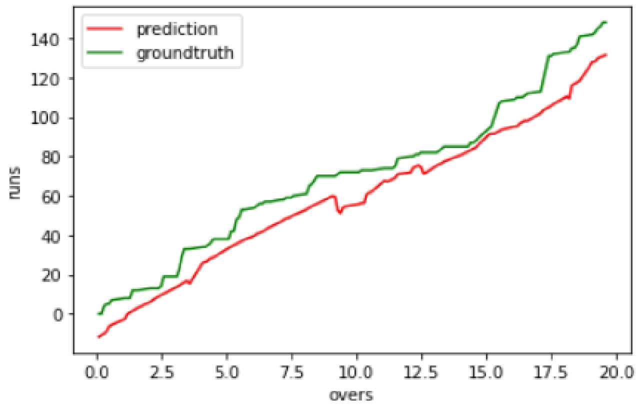


Fig 5. Score Forecast

The next goal was to predict the outcome of the unfinished match using a classification approach with two classes: (0) the team batting first won and (1) the team batting second won, applied across the entire dataset. Initially, we used one of the most common classifiers, SVM, with a normalization parameter of 1.0 and an 'rbf' kernel, which yielded an accuracy score of 0.53. We experimented with various kernels, and the polynomial (poly) kernel improved the accuracy by 4%. Next, we applied a Gaussian Naive Bayes model with no prior, allowing it to adjust the prior probability based on the data distribution, and with a variance smoothing parameter of 1e-09 for computational stability. This model achieved an accuracy score of 0.67 on the test dataset. We also tested the k-nearest neighbors (KNN) model with uniform weights and an auto algorithm to determine the best computation based on input data. Initially, with smaller k values, the model showed high accuracy on the training data but performed poorly on the test data, indicating overfitting. Increasing k to 11 led to a decrease in accuracy, and we found that odd k values provided better precision than even k values. Ultimately, we obtained an accuracy score of 0.86 with k=11, as shown in Table II.

TABLE II. ACCURACY COMPARISON

| Model | Accuracy |
|---|---|
| SVM | 57.217 % |
| k-NN | 86.665 % |
| Gaussian Naïve Bayes | 67.154 % |

The introduction of deep learning techniques for outcome determination was the primary goal of this study. To accomplish this task, we constructed a neural network architecture using three interconnected models, each employing an activation mechanism, batch standardization, and randomly excluded connections. Initially, we transformed the feature set and corresponding output variable into a tensor format before dividing the data into 75% for training and 25% for validation, reserving the latter for testing model performance. The primary layer contained 200 neurons while the secondary and tertiary layers consisted of 100 and 50 neurons respectively. A regularization momentum of 0.1 and 40% probability of randomly dropping connections during training helped optimize model fitting, yielding an accuracy of 67% on validation data. Error was quantified using cross-entropy loss against a learning rate of 0.01 to minimize costs over iterations. In an effort to boost precision, we experimented with creating more complex networks comprising four interconnected parts with varying numbers of neurons from 200 down to 25 within hidden strata to analyze impacts on prediction prowess. Nevertheless, even though it produced an accuracy of 0.74 and performed well on the test dataset, it was computationally expensive. Next, in an attempt to achieve the intended outcome, we experimented with several optimizers, but the accuracy only went up by 3%. The corresponding accuracy values that were attained by testing with several optimizers, such as AdaGrad, AdaDelta, and SGD, are displayed in Table III.

TABLE III. ACCURACY COMPARISON (NN)

| **Model** | | **Accuracy** |
|---|---|---|
| **3-layered NN** | | 67.82 % |
| **Optimizer** | SGD | 74.32 % |
| | AdaGrad | 74.76 % |
| | AdaDelta | 77.44 % |

While AdaDelta generated the most accurate model, its four-layer configuration incurred excessive computational costs. Therefore, we concentrated tuning efforts on the cheaper three-layer network. Initially, incremental upgrades to layer sizes modestly boosted performance. However, expanding each layer to hold more neurons yielded far better results. Specifically, allocating the first layer 400 units, the second 200, and the final 100 neurons markedly raised accuracy. With these augmented dimensions, we trained the architecture across 200 epochs. The normalization momentum remained at 0.1 and dropout was set to 0.4, preserving stability while avoiding overfitting. Through iterative experimentation, this balanced configuration proved best able to learn from the data in a cost-effective manner. While on test data, the model achieved an accuracy score of 91.461% as shown in Table IV.

TABLE IV. BEST ACCURACY USING NEURAL NETWORK

| **Model** | **Accuracy** |
|---|---|
| 3-layered NN (AdaDelta optimal) | 91.46 % |

The loss curve and confusion matrix are displayed in the graphics below (see Fig 6, Fig.7 and Fig 8.) to provide a better understanding of how the deep model learns during training and how it predicts results during testing phase.

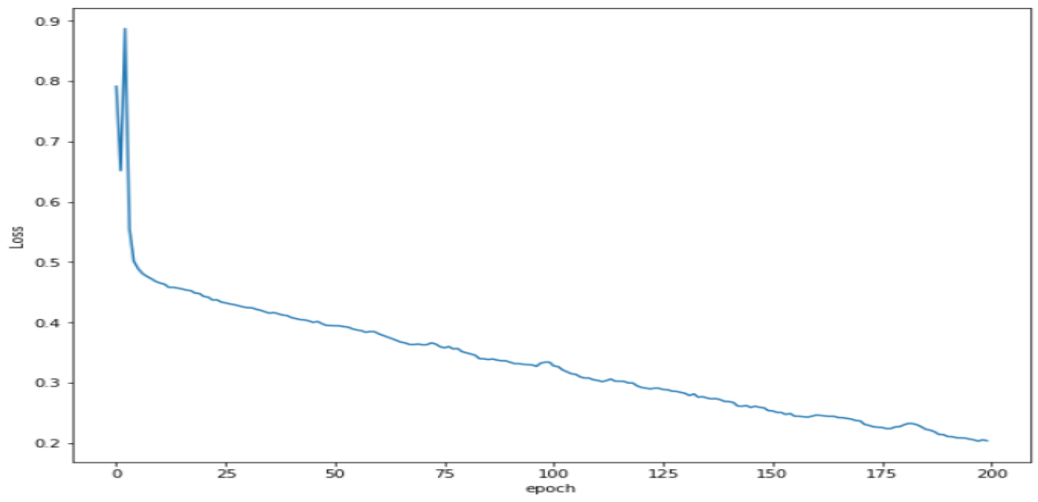
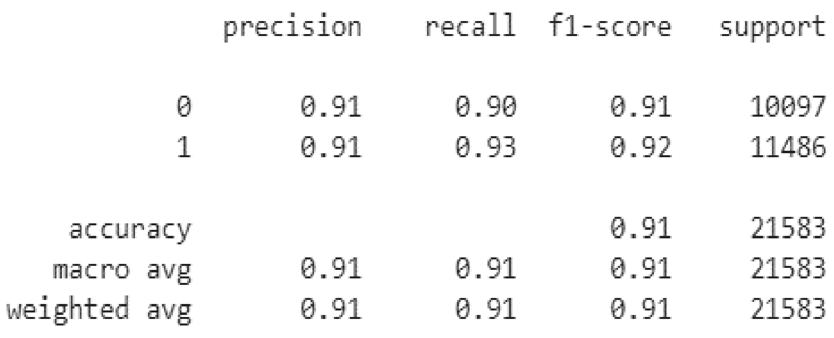


Fig 6. Epoch vs Loss plot during Model Training



```
              precision    recall  f1-score   support

           0       0.91      0.90      0.91     10097
           1       0.91      0.93      0.92     11486

    accuracy                           0.91     21583
   macro avg       0.91      0.91      0.91     21583
weighted avg       0.91      0.91      0.91     21583
```

Fig 7. Classification Outcome after Testing of the Model

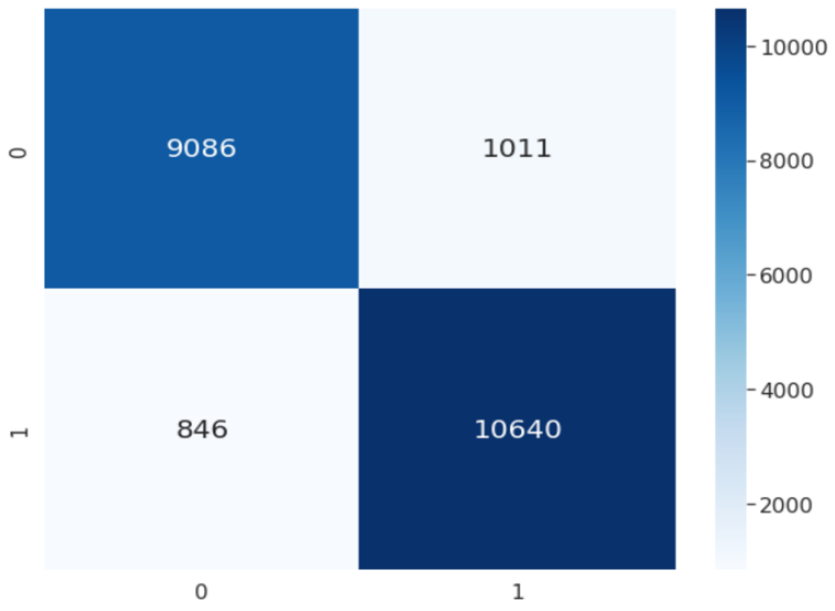


Fig 8. Confusion Matrix after Model was run on Test data

## IV. Conclusion & Future Work

Several approaches were explored in this research, including linear, and multiple linear regression, and support vector regression. However, polynomial regression proved to be the most effective model for predicting scores, achieving a mean squared error of 94.5. Support vector machines, Gaussian naive Bayes categorization, k-nearest neighbor categorization, and deep learning using a neural system containing three layers were also evaluated, with the most accurate model for forecasting match conclusions reported as the three-layer neural system attaining 91.46 percent precision. Interestingly, while the simpler techniques fell short, the polynomial regression and deep learning approaches leveraged their enhanced adaptability to glean further insights from the information, thereby winning out overall. This paper presents an AI alternative that takes into consideration all additional parameters, such as pitch, toss, and first-inning score and raises issues about the usage of the traditional D/L approach for incomplete matches. In contrast, my proposed method, which utilizes a broader feature set, achieved higher accuracy with reduced computational costs. This study can be of interest to cricket managers, sports analysts, and scholars interested in sports analysis. We can extend this research to different formats of cricket like the ODI, and Test, by utilizing additional attributes like the weather and pitch conditions. The T20 format of cricket often leads to great unpredictability since even minor adjustments to a bowler's delivery or placement of fielders can tremendously impact the outcome of a game, especially during the restricted overs at the beginning. Therefore, we might test introducing one random element to gauge how it could influence the match proceedings. It may also be possible to forecast which player will be named man of the match for their stellar performance, predict who will score the most runs, and determine which bowler claims the highest number of wickets in upcoming contests. Ultimately, this entire project has the potential to be adapted to forecast comparable results in numerous other sports as well, such as baseball, tennis, American football, and more.

## References

[1] Tejinder Singh, Vishal Singla, Parteek Bhatia; - Score and Winning Prediction in Cricket through Data Mining; Oct 8-10, 2015

[2] Animal Islam Anik, Sakif yeaser, A.G.M. Emam Hussain, Amitabha Chakraborty; Player's Performance Prediction in ODI Cricket Using Machine Learning Algorithms;2018

[3] Jalaz Kumar, Rajeev Kumar, Pushpender Kumar; Outcome Prediction of ODI Cricket Matches using Decision Trees and MLP Networks;2018

[4] M. Bailey and S.R. Clark; Predicting the Match Outcome in One Day International Cricket Matches, while the Game is in Progress;2006

[5] Kumail and Sajjad; Duckworth-Lewis-Stern Method Comparison with Machine Learning Approach; 2021

[6] Prasad, Vighnesh, Yash; Review Paper On Cricket Score Prediction; 2021

[7] Basil M. de Silva, Greg R. Pond and Tim B. Swartz; Estimation of the Magnitude of Victory in One-Day Cricket;2001

[8] Bailey M.; Predicting sporting outcomes: A statistical approach;2005

[9] A. Tripathi, J. Vanker, B. Vaje, V. Varekar; Cricket Score Prediction system using clustering algorithm;2016

[10] P. Somaskandhan, G. Wijesinghe, L. Bashitha; Identifying optimal set of attributes that impose high impact on end results of cricket match using machine learning;2017

[11] S. Agrawal, S. P. Singh and J. K. Sharma; Predicting Results of Indian Premier League T-20 Matches using Machine Learning;2018

[12] A. Basit, M. B. Alvi, F. H. Jaskani, M. Alvi, K. H. Memon and R. A. Shah; ICC T20 Cricket World Cup 2020 Winner Prediction Using Machine Learning Techniques; 2020

[13] Nikhil Dhonge, Shraddha Dhole, Nikita Wavre, Mandar Pardakhe, Amit Nagarale; IPL CRICKET SCORE AND WINNING PREDICTION USING MACHINE LEARNING TECHNIQUES; 2020

[14] H. Barot, A. Kothari, P. Bide, B. Ahir and R. Kankaria, "Analysis and Prediction for the Indian Premier League," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-7, doi: 10.1109/INCET49848.2020.9153972.

[15] A. Basit, M. B. Alvi, F. H. Jaskani, M. Alvi, K. H. Memon and R. A. Shah, "ICC T20 Cricket World Cup 2020 Winner Prediction Using Machine Learning Techniques," 2020 IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 2020, pp. 1-6, doi: 10.1109/INMIC50486.2020.9318077.

[16] A. Bandulasiri, "Predicting the Winner in One Day International Cricket", Journal of Mathematical Sciences & Mathematics Education, Vol. 3, No. 1.