Case Study : IPL Scores Prediction Using Neural Networks

Maitreya Sameer Ganu , IISER Thiruvananthapuram

17 th December 2024

Abstract

This project focuses on predicting IPL match scores using machine learning, leveraging advanced neural networks and a user-friendly interface built on Google Colab. By combining historical match data with dynamic feature inputs, users can interactively explore predictions of match scores based on different game conditions, such as the venue, teams, and players.

1 Introduction:

The Indian Premier League (IPL) is one of the most popular and competitive cricket leagues in the world, attracting millions of fans globally. Predicting the outcome of an IPL match, especially the score, is a complex problem that requires an in-depth understanding of various factors such as the teams playing, their individual players' performance, the match venue, and dynamic game conditions. In the world of sports analytics, predictive models can be valuable tools for providing insights into match outcomes. This project aims to predict the score of an IPL match using machine learning techniques. The model is built using historical match data, which includes features like the venue, batting and bowling teams, individual player performance, and other relevant match details. The model uses a neural network approach to predict scores dynamically, leveraging the flexibility and power of advanced machine learning techniques. One of the key features of this project is the interactive interface built using Google Colab, allowing users to input match details such as the venue, batting team, bowling team, striker, and bowler, and receive real-time score predictions. This provides an engaging experience where users can experiment with different match conditions and observe how predictions change accordingly. By providing these insights, this project aims to offer a practical example of how machine learning can be applied in sports analytics, particularly in cricket, and how it can contribute to making data-driven predictions in real-time.

2 Overview of the data set:

The following table describes the general overview of the IPL data set . There

	mid	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
count	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000	76014.000000
mean	308.627740	74.889349	2.415844	9.783068	33.216434	1.120307	24.962283	8.869287	160.901452
std	178.156878	48.823327	2.015207	5.772587	14.914174	1.053343	20.079752	10.795742	29.246231
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	67.000000
25%	154.000000	34.000000	1.000000	4.600000	24.000000	0.000000	10.000000	1.000000	142.000000
50%	308.000000	70.000000	2.000000	9.600000	34.000000	1.000000	20.000000	5.000000	162.000000
75%	463.000000	111.000000	4.000000	14.600000	43.000000	2.000000	35.000000	13.000000	181.000000
max	617.000000	263.000000	10.000000	19.600000	113.000000	7.000000	175.000000	109.000000	263.000000

Figure 1: Data description.

were no missing and no null values in this data set . Columns like 'date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5','mid', 'striker and 'non-striker' were ignored and not used for data analysis . Only the 'total' column was used for training.

3 Data Prepocessing:

3.1 Dataset Overview

The dataset contains historical IPL match data, including features such as:

- Venue of the match
- Batting and Bowling teams
- Individual players (batsman and bowler)
- Current match progress

3.2 Preprocessing

Data preprocessing involved the following steps:

- Encoding categorical variables (e.g., team names, player names) into numerical formats using `LabelEncoder`.
- Normalizing numerical features using a `StandardScaler` to improve model performance.
- Splitting the dataset into training and testing sets for model evaluation. 30% of the data was used for testing and 70% was used for training.

4 Machine Learning Model:

4.1 Model Architecture

The neural network model was designed to learn complex relationships in the data and predict the total score of the batting team. It ran for 100 epochs with a batch size of 64 . The architecture of the model consisted of the following components:

- **Input Layer:** The input layer accepts the features after they have been encoded and scaled. These features include encoded values for the venue, batting team, bowling team, batsman, and bowler. Scaling ensures that all numerical inputs have similar ranges, which is essential for efficient model training and convergence.
- **Two Hidden Layers:**
  - Each hidden layer consists of dense (fully connected) units.
  - The first hidden layer contains 512 neurons, and the second contains 216 neurons, with each neuron applying a Rectified Linear Unit (ReLU) activation function. The ReLU activation introduces non-linearity into the model, enabling it to capture complex relationships in the data.
  - These dense layers process the input features and extract meaningful patterns that are critical for predicting the final score.
- **Output Layer:** The output layer consists of a single neuron with a linear activation function. This setup is appropriate for regression tasks because the output can take on any continuous value, representing the predicted score of the batting team.

The model's structure was chosen to balance complexity and computational efficiency, ensuring that it could capture the relationships in the data without overfitting.

4.2 Loss Function and Optimization

To train the model effectively, the Huber loss function and the Adam optimizer were employed:

Huber Loss Function:

- The Huber Loss is a robust loss function that combines the strengths of Mean Squared Error (MSE) and Mean Absolute Error (MAE). It is particularly suited for regression tasks with outliers in the dataset.
- For small differences between the predicted and actual values, Huber Loss behaves like MSE, ensuring sensitivity to minor deviations.

- For larger differences, it transitions to behave like MAE, reducing the impact of outliers by penalizing them linearly instead of quadratically.
- This hybrid nature makes Huber Loss a balanced approach for handling both small errors and significant outliers effectively.
- In this project, a threshold parameter (δ) was used to control the transition between the MSE and MAE behaviors. A value of 1.0 was selected, which provided a good balance for the given data.

Adam Optimizer:

- The Adam optimizer was used for training due to its efficiency in handling sparse gradients and noisy data.
- Adam combines the benefits of two popular optimization algorithms—AdaGrad and RMSProp—by adaptively adjusting the learning rate for each parameter during training.
- This ensures faster convergence while maintaining stability, making it ideal for this regression task.

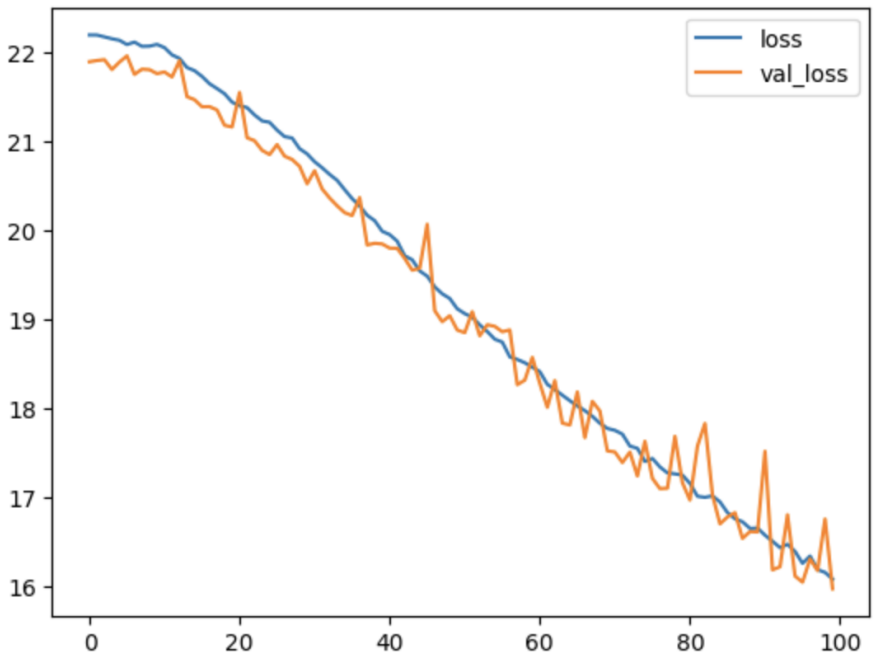

Figure 2: Variation of actual loss and variation loss (Y axis) with number of epochs (X axis) . The mean absolute error was approximately 16.47 .

5 Interactive Interface:

The project integrates `ipywidgets` to allow users to interact with the model:

- Dropdown menus for selecting the venue, batting team, bowling team, striker, and bowler.
- A "Predict Score" button to trigger the prediction.
- Real-time score predictions displayed in the Jupyter Notebook output cell.

6 Conclusion:

This project successfully integrates machine learning with an interactive user interface to predict IPL match scores. It demonstrates the potential of AI in sports analytics, providing a foundation for future enhancements.

7 Data and Code Availability:

- Access the data set by visiting geeksforgeeks :
  IPL Score Prediction.
- Access the code here :
  Code for IPL Score Prediction along with an interactive interface.