
**VNR Vignana Jyothi Institute of Engineering and Technology (Affiliated to J.N.T.U, Hyderabad) Bachupally(v), Hyderabad, Telangana, India.**

# IPL SCORE PREDICTION

A course project submitted in complete requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

IN

**COMPUTER SCIENCE AND ENGINEERING & BUSINESS SYSTEM**

Submitted by

**A.Harshita (21071A3202)**
**B.V. Vimal (21071A309)**
**G.Tejaswi (21071A3219)**
**M. Sravanth (21071A3245)**
**P.L.S. Deepika(21071A3255)**

Under the guidance

of

**Mrs.Kriti Ohri**

**Assistant Professor**

**Dept. of Computer Science and Engineering**


**VNR Vignana Jyothi Institute of Engineering and Technology**
**(Affiliated to J.N.T.U, Hyderabad)**
**Bachupally(v), Hyderabad, Telangana, India.**

# CERTIFICATE

This is to certify that **A.Harshita(21071A3202) B.V.Vimal(21071A3209) G.Tejaswi(21071A3219) M.Sravanth(21071A3245) P.L.S.Deepika(21071A3255)** have completed their course project work at CSE Department of VNR VJIET, Hyderabad entitled " **IPL SCORE PREDICTION** " in complete fulfilment of the requirements for the award of B.Tech degree during the academic year 2022-2023. This work is carried out under my supervision and has not been submitted to any other University/Institute for award of any degree/diploma.

| | |
|---|---|
| **Mrs.Kriti Ohri** | **Dr. S Nagini** |
| Assistant Professor | Professor and HOD |
| CSE Department | CSE Department |
| VNRVJIET | VNRVJIET |

# DECLARATION

This is to certify that our project report titled "**IPL SCORE PREDICTION**" submitted to Vallurupalli Nageswara Rao Institute of Engineering and Technology in complete fulfilment of requirement for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide report to the work carried out by us under the guidance and supervision of **Mrs.Kriti Ohri** Assistant Professor, Department of Computer Science and Engineering, Vallurupalli Nageswara Rao Institute of Engineering and Technology. To the best of our knowledge, this has not been submitted in any form to other university or institution for the award of any degree or diploma.

| **A.Harshita** | **B.V.Vimal** | **G.Tejaswi** | **M.Sravanth** | **P.L.S.Deepika** |
|---|---|---|---|---|
| 21071A3202 | 21071A3209 | 21071A3219 | 21071A3245 | 21071A3255 |
| CSBS | CSBS | CSBS | CSBS | CSBS |

# ACKNOWLEDGEMENT

Over a span of three and a half years, VNRVJIET has helped us transform ourselves from mere amateurs in the field of Computer Science into skilled engineers capable of handling any given situation in real time. We are highly indebted to the institute for everything that it has given us. We would like to express our gratitude towards the principal of our institute, **Dr. Challa Dhanunjaya Naidu** and the Head of the Computer Science & Engineering Department, **Dr. S. Nagini** for their kind co- operation and encouragement which helped us complete the project in the stipulated time. Although we have spent a lot of time and put in a lot of effort into this project, it would not have been possible without the motivating support and help of our project guide **Mrs.Kriti Ohri**. We thank her for her guidance, constant supervision and for providing necessary information to complete this project. Our thanks and appreciations also go to all the faculty members, staff members of VNRVJIET, and all our friends who have helped us put this project together.

# ABSTRACT

The Indian Premier League (IPL) has emerged as one of the most popular and competitive T20 cricket leagues globally, attracting a massive fan base and intense scrutiny from cricket enthusiasts. Predicting match scores in the IPL is a challenging yet intriguing task due to the dynamic nature of the game, the influence of various factors, and the unpredictable performance of teams and players.

Our project aims to develop a robust predictive model for estimating match scores in the IPL using advanced machine learning techniques. The primary objective is to leverage historical match data, player statistics, venue conditions, team dynamics, and other relevant features to create a model that accurately forecasts the total runs scored by each team in a given match.

The methodology involves data collection from reliable sources, preprocessing to handle missing values and outliers, and feature engineering to extract meaningful insights from the dataset. Various machine learning algorithms, such as regression models and ensemble methods, will be explored and compared to identify the most effective approach for score prediction. Additionally, advanced techniques like deep learning may be employed to capture intricate patterns within the data.

The project's success will be evaluated based on the model's ability to generalise well to unseen data and its predictive accuracy in estimating match scores. The outcomes of this research can have significant implications for cricket enthusiasts, fantasy sports platforms, and analysts seeking reliable predictions for IPL matches.

Our project seeks to contribute to the growing field of sports analytics by developing an accurate and reliable model for predicting IPL match scores. The findings can potentially enhance the overall cricket-watching experience and assist stakeholders in making informed decisions based on data-driven insights.

# Table of Contents

# 1 Introduction

In the pulsating world of the Indian Premier League (IPL), where every match unfolds as a spectacle of raw talent and strategic brilliance, the quest to predict match scores has become an intriguing challenge. Cricket enthusiasts, data scientists, and sports analysts are increasingly drawn to unravelling the complex web of factors that influence the total runs scored by teams in this fast-paced T20 format. This project endeavours to bridge the gap between the unpredictability of cricket and the power of machine learning, offering a systematic and data-driven approach to forecasting IPL match scores.

Cricket, with its ever-evolving dynamics and the emergence of T20 leagues like the IPL, presents a unique set of challenges for score prediction. The amalgamation of player form, team dynamics, playing conditions, and historical performance creates a dynamic landscape that demands a sophisticated analytical approach. In this project, we delve into this dynamic landscape armed with machine learning techniques to not only make sense of the multitude of variables but also to make accurate predictions regarding the total runs a team might score in a given IPL match.

Our approach involves meticulous data collection from diverse sources, including historical match statistics, player profiles, venue details, and team dynamics. This data serves as the foundation for our predictive model, wherein we employ advanced machine learning algorithms to discern patterns and relationships that contribute to a team's overall scoring trends. The model is designed to adapt and learn from historical data, ensuring that it can effectively navigate the unpredictability inherent in T20 cricket.

We explore a range of machine learning techniques, from traditional regression models to ensemble methods, each meticulously tailored to capture the nuances of cricketing dynamics. Additionally, we consider the application of deep learning, enabling the model to uncover intricate patterns within the data that may elude conventional approaches.

# 2 Literature

The research paper of G. Sudhamathy helps to understand the different machine learning algorithms working principle and their implementation. It creates the Model and Training dataset and helps to predict with the help of the model created. The model classifies the data and compares the results and gets accuracy which is the important one [4].

As in the dataset there are many parameters present. Out of them, which parameters are helpful in the project? The factors affecting the concept were taken by Maheshwari in their prediction of live cricket score paper from which we get to know the main factors which are required for the prediction of score and the prediction of winning team [2]. [1] The role of classification is clarified in the paper of Tejinder Singh; it gives proper information or use of naivebias and linear regression. They give the proper knowledge of data collection and preparation also how to train them and test the data given by them which is more helpful. The support vector machines brief idea has been taken from Aminul Islam Anik paper which is about players performance in this paper the idea about SVM system is given in detail where the player performance prediction is given by collecting the old information or data [3]. From the literature survey it is concluded that machine learning is needed for prediction.

So, we have concluded the following from the above literature Survey and we can summarise the following from this. Using the IPL dataset, predict the score of any ipl team. Here we need to look at a number of factors while predicting the score. Factors like Ground History, Team Balance, Current Situation, run rate, overs remaining and so on. We will predict the First Innings Score with it of the team batting first. In sports, most of the prediction job is done using regression or classification tasks, both of which come under supervised learning.

# 3 Requirements

Requirements analysis in systems engineering and software engineering encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analysing, documenting, validating and managing software or system requirements.

## Software Requirements

- Software : Python, Jupyter Notebook /Google Colab
- Operating System : Windows/macOS
- Technology : Machine Learning, Deep Learning

## Hardware Requirements

- Minimum 8GB Ram Laptop
- Internet Connection

# 4 Model Implementation

## 4.1 Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import keras
import tensorflow as tf
```

**Pandas —** The pandas library is a powerful, open-source software library for data analysis and manipulation in Python. It provides easy-to-use data structures and data analysis tools for working with structured data, including data frames and series.

**Numpy —** NumPy, which stands for Numerical Python, is a popular library in Python used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

**matplotlib.pyplot** module, a popular Python library used for creating visualisations, such as charts and graphs.

**Seaborn** is a Python visualisation library based on matplotlib that provides a high-level interface for creating informative and attractive statistical graphics. It is often used to create more visually appealing and complex visualisations compared to the basic capabilities of matplotlib.

**scikit-learn**, a popular machine learning library for Python

The **preprocessing** module provides several techniques for transforming raw feature vectors into a format that better suits the machine learning model.

**Keras** is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit.

**TensorFlow** is an open-source machine learning framework developed by the Google Brain team. It is designed to enable efficient numerical computation and machine learning development through a flexible and comprehensive ecosystem of tools, libraries, and community resources.

## 4.2 Loading Dataset

```
ipl=pd.read_csv('/content/ipl_data.csv')
ipl
```

| | mid | date | venue | bat_team | bowl_team | batsman | bowler | runs | wickets | overs | runs_last_5 | wickets_last_5 | striker | non-striker | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | SC Ganguly | P Kumar | 1 | 0 | 0.1 | 1 | 0 | 0 | 0 | 222 |
| 1 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 1 | 0 | 0.2 | 1 | 0 | 0 | 0 | 222 |
| 2 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.2 | 2 | 0 | 0 | 0 | 222 |
| 3 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.3 | 2 | 0 | 0 | 0 | 222 |
| 4 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.4 | 2 | 0 | 0 | 0 | 222 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 76009 | 617 | 2017-05-21 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian | 121 | 7 | 19.2 | 40 | 0 | 40 | 12 | 129 |
| 76010 | 617 | 2017-05-21 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian | 127 | 7 | 19.3 | 46 | 0 | 46 | 12 | 129 |
| 76011 | 617 | 2017-05-21 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian | 128 | 7 | 19.4 | 47 | 0 | 47 | 12 | 129 |
| 76012 | 617 | 2017-05-21 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | MG Johnson | DT Christian | 129 | 7 | 19.5 | 48 | 0 | 47 | 13 | 129 |

## 4.3 Data preprocessing

Dropping unimportant features

- We have created a new dataframe by dropping several columns from the original DataFrame.
- The new DataFrame contains the remaining columns that we are going to train the predictive model on.

```
[ ] #Dropping certain features
    df = ipl.drop(['date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5','mid', 'striker', 'non-striker'], axis =1)
```

```
X = df.drop(['total'], axis =1)
y = df['total']
```

We have split the data frame into input features (X) and target features (y). Our target features is the total score.

We are selecting only a few features from the dataset that influence the score those are venue, bat_team, bowl_team, bats_man, bowler.

X

| | venue | bat_team | bowl_team | batsman | bowler |
|---|---|---|---|---|---|
| 0 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | SC Ganguly | P Kumar |
| 1 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar |
| 2 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar |
| 3 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar |
| 4 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar |
| ... | ... | ... | ... | ... | ... |
| 76009 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian |
| 76010 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian |
| 76011 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian |
| 76012 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | MG Johnson | DT Christian |
| 76013 | Rajiv Gandhi International Stadium, Uppal | Mumbai Indians | Rising Pune Supergiant | KH Pandya | DT Christian |

## Label Encoding

- We have applied label encoding to your categorical features in X.
- We have created separate LabelEncoder objects for each categorical feature and encoded their values.
- We have created mappings to convert the encoded labels back to their original values, which can be helpful for interpreting the results.

```
from sklearn.preprocessing import LabelEncoder

# Create a LabelEncoder object for each categorical feature
venue_encoder = LabelEncoder()
batting_team_encoder = LabelEncoder()
bowling_team_encoder = LabelEncoder()
striker_encoder = LabelEncoder()
bowler_encoder = LabelEncoder()

# Fit and transform the categorical features with label encoding
X['venue'] = venue_encoder.fit_transform(X['venue'])
X['bat_team'] = batting_team_encoder.fit_transform(X['bat_team'])
X['bowl_team'] = bowling_team_encoder.fit_transform(X['bowl_team'])
X['batsman'] = striker_encoder.fit_transform(X['batsman'])
X['bowler'] = bowler_encoder.fit_transform(X['bowler'])
```

### Train Test Split

The train_test_split function in the sklearn.model_selection module is used to split the dataset into two separate sets for training and testing machine learning models.

We have split the data into training and testing sets. The training set contains 70 percent of the dataset and rest 30 percent is in test set.

- X_train contains the training data for your input features.
- X_test contains the testing data for your input features.
- y_train contains the training data for your target variable.
- y_test contains the testing data for your target variable.

```
# Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
X_train
```

| | venue | bat_team | bowl_team | batsman | bowler |
|---|---|---|---|---|---|
| **31043** | 15 | 12 | 0 | 78 | 121 |
| **1720** | 22 | 2 | 4 | 393 | 48 |
| **62369** | 14 | 12 | 13 | 384 | 175 |
| **74763** | 21 | 4 | 6 | 398 | 157 |
| **10506** | 2 | 7 | 2 | 266 | 218 |
| **...** | ... | ... | ... | ... | ... |
| **37194** | 34 | 6 | 7 | 221 | 272 |
| **6265** | 26 | 7 | 9 | 347 | 292 |
| **54886** | 1 | 7 | 0 | 213 | 183 |
| **860** | 15 | 0 | 7 | 258 | 275 |
| **15795** | 14 | 12 | 0 | 384 | 169 |

### Feature Scaling

- We have performed Min-Max scaling on our input features to ensure all the features are on the same scale
- Scaling is performed to ensure consistent scale to improve model performance.
- Scaling has transformed both training and testing data using the scaling parameters.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Fit the scaler on the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

## 4.4 Defining Neural network

- We have defined a neural network using TensorFlow and Keras for regression.
- After defining the model, we have compiled the model using the Huber Loss.

```
model = keras.Sequential([
  keras.layers.Input( shape=(X_train_scaled.shape[1],)),
  keras.layers.Dense(512, activation='relu'),
  keras.layers.Dense(216, activation='relu'),
  keras.layers.Dense(1, activation='linear')
])


huber_loss = tf.keras.losses.Huber(delta=1.0)
model.compile(optimizer='adam', loss=huber_loss)
```

## 4.5 Model training

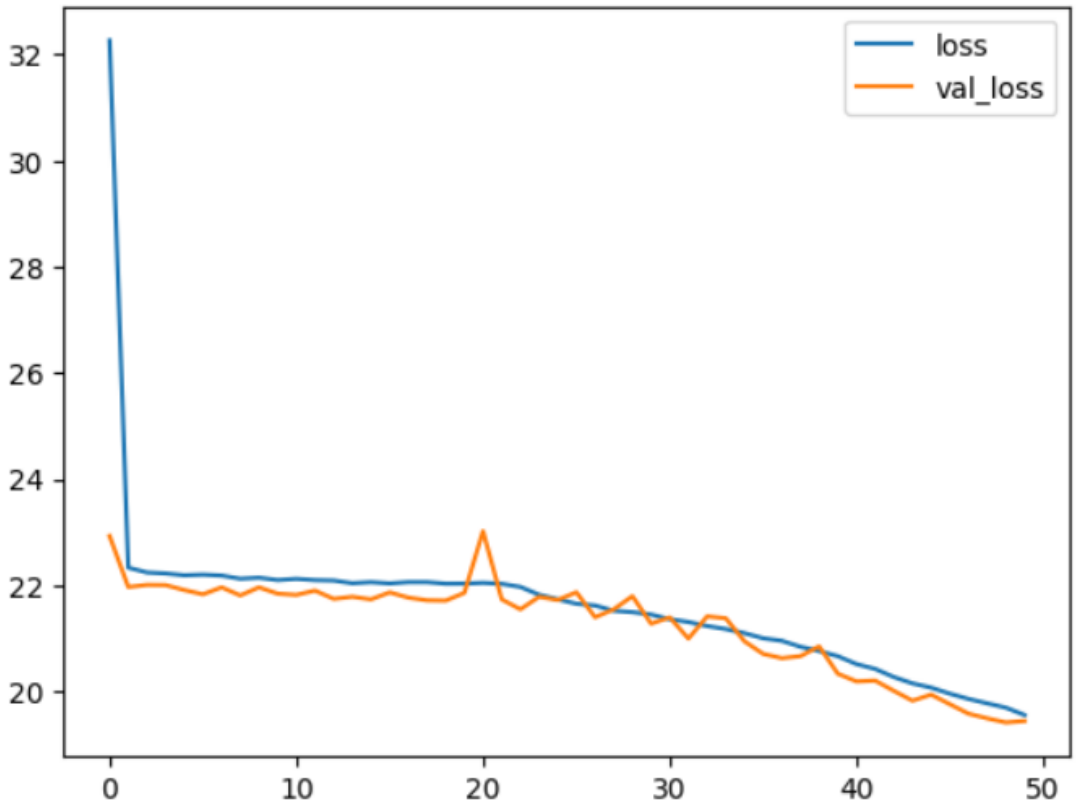
We have trained the neural network model using the scaled training data.

```
# Train the model
model.fit(X_train_scaled, y_train, epochs=50, batch_size=64, validation_data=(X_test_scaled, y_test))
```

```
Epoch 21/50
832/832 [==============================] - 6s 8ms/step - loss: 21.7538 - val_loss: 21.3923
Epoch 22/50
832/832 [==============================] - 5s 6ms/step - loss: 21.6602 - val_loss: 21.5358
Epoch 23/50
832/832 [==============================] - 5s 6ms/step - loss: 21.6018 - val_loss: 21.2886
Epoch 24/50
832/832 [==============================] - 6s 8ms/step - loss: 21.5163 - val_loss: 21.4690
Epoch 25/50
832/832 [==============================] - 6s 7ms/step - loss: 21.5136 - val_loss: 21.1977
Epoch 26/50
832/832 [==============================] - 7s 8ms/step - loss: 21.4910 - val_loss: 21.3184
Epoch 27/50
832/832 [==============================] - 5s 6ms/step - loss: 21.3983 - val_loss: 21.1786
Epoch 28/50
832/832 [==============================] - 5s 6ms/step - loss: 21.3604 - val_loss: 21.1058
Epoch 29/50
832/832 [==============================] - 6s 7ms/step - loss: 21.2893 - val_loss: 21.0406
Epoch 30/50
832/832 [==============================] - 5s 6ms/step - loss: 21.1872 - val_loss: 20.8679
Epoch 31/50
832/832 [==============================] - 6s 7ms/step - loss: 21.1070 - val_loss: 20.7837
Epoch 32/50
832/832 [==============================] - 5s 6ms/step - loss: 21.0455 - val_loss: 20.9679
Epoch 33/50
832/832 [==============================] - 5s 5ms/step - loss: 20.9309 - val_loss: 20.6156
Epoch 34/50
```

```
model_losses = pd.DataFrame(model.history.history)
model_losses.plot()
```

```
<Axes: >
```




After the training, we have stored the training and validation loss values to our neural network during the training process.

## 4.6 Model Evaluation

- We have predicted using the trained neural network on the testing data.
- The variable predictions contains the predicted total run scores for the test set based on the model's learned patterns.

```
# Make predictions
predictions = model.predict(X_test_scaled)

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
mae=mean_absolute_error(y_test,predictions)
mse=mean_squared_error(y_test,predictions)
r2=r2_score(y_test,predictions)
print('Mean absolute error',mae)
print('Mean squared error',mse)
print('R2_score',r2)
```

```
713/713 [==============================] - 1s 2ms/step
Mean absolute error 19.974337082394275
Mean squared error 728.475888101443
R2_score 0.13624611641795348
```

## 4.7 Interactive Widget

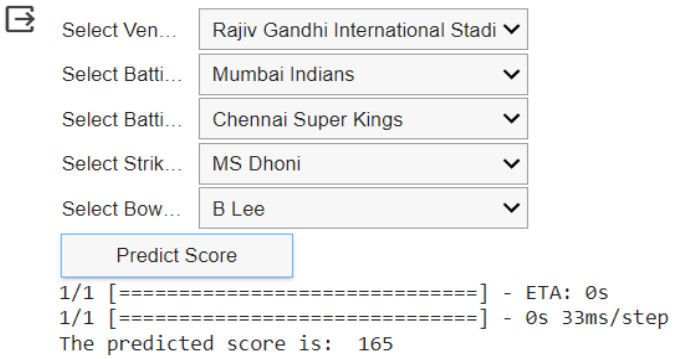
We have created an interactive widget using ipy widgets to predict the score based on user input for venue, batting team, bowling team, striker, and bowler.

- We have created dropdown widgets to select values for venue, batting team, bowling team, striker, and bowler.
- Then, we have added a "Predicted Score" button widget. Whenever, the button will be clicked, the predict_score function will be called and then perform the following steps:
    - Decodes the user-selected values to their original categorical values.
    - Encodes and scales these values to match the format used in model training.
    - Uses the trained model to make a prediction based on the user's input.
    - Displays the predicted score.

```
import ipywidgets as widgets
from IPython.display import display, clear_output

import warnings
warnings.filterwarnings("ignore")

venue = widgets.Dropdown(options=df['venue'].unique().tolist(),description='Select Venue:')
batting_team = widgets.Dropdown(options =df['bat_team'].unique().tolist(), description='Select Batting Team:')
bowling_team = widgets.Dropdown(options=df['bowl_team'].unique().tolist(), description='Select Batting Team:')
striker = widgets.Dropdown(options=df['batsman'].unique().tolist(), description='Select Striker:')
bowler = widgets.Dropdown(options=df['bowler'].unique().tolist(), description='Select Bowler:')

predict_button = widgets.Button(description="Predict Score")

def predict_score(b):
  with output:
    clear_output() # Clear the previous output


    # Decode the encoded values back to their original values
    decoded_venue = venue_encoder.transform([venue.value])
    decoded_batting_team = batting_team_encoder.transform([batting_team.value])
    decoded_bowling_team = bowling_team_encoder.transform([bowling_team.value])
    decoded_striker = striker_encoder.transform([striker.value])
    decoded_bowler = bowler_encoder.transform([bowler.value])


    input = np.array([decoded_venue, decoded_batting_team, decoded_bowling_team,decoded_striker, decoded_bowler])
    input = input.reshape(1,5)
    input = scaler.transform(input)
    #print(input)
    predicted_score = model.predict(input)
    predicted_score = int(predicted_score[0,0])

    print(predicted_score)
```

**The widget-based interface allows you to interactively predict the score for specific match scenarios. Now, we have set up the button to trigger the predict_score function when clicked and display the widgets for venue, batting team , bowling team, striker and bowler.**

```
predict_button.on_click(predict_score)
output = widgets.Output()
display(venue, batting_team, bowling_team, striker, bowler, predict_button, output)
```



# 5 ARTEFACT DESCRIPTION

The IPL Score Prediction project involves a series of techniques and artefacts that contribute to the development of an accurate predictive system. They are:

## 1. Dataset Collection and Preprocessing:

Data Collection:
Gather comprehensive historical data on IPL matches, including venue, batting team, bowling team, batsman, bowler, and cricket scores.
Format: Structured dataset with diverse match scenarios.
Data Preprocessing:
Processed categorical features into numerical representations using techniques like one-hot encoding or label encoding.
Normalised Numerical Features:
Scaled numerical features to a standard range for consistent model training.

## 2. Neural Network Model Architecture:

Defined a feedforward neural network with sequential layers for regression to predict IPL scores.
Architecture:
Input Layer: Matched the shape of the preprocessed data.
Hidden Layers: Two dense layers with 512 and 216 units, ReLU activation functions.
Output Layer: Single unit with linear activation for regression.

## 3. Model Compilation :

Huber Loss Function:
Utilised Huber loss function during model compilation for regression tasks.
Parameters: Adjusted the 'delta' parameter for a balanced transition from quadratic to linear behaviour.
Adam Optimizer:

Selected the Adam optimizer for gradient-based optimization during model compilation.

## 4. Training :

Train-Test Split:
Split the dataset into training and validation sets to evaluate model performance and prevent overfitting.
Model Training:
Trained the neural network model using the compiled architecture and the preprocessed training data.
Monitoring: Evaluated training progress by assessing loss on the validation set.

## 5. Result Analysis :

Evaluation Metrics:
Utilised Mean Squared Error (MSE) and Mean Absolute Error (MAE) to quantify prediction accuracy.
Purpose: Evaluated how well the model's predictions aligned with actual IPL scores.
Visualisation/Statistical Analysis:
Methods: Employed visualisations or statistical analyses to interpret and communicate the results effectively.
Purpose: Enhanced understanding of the model's strengths and limitations
.

## 6. Interactive Prediction Widget:

Developed an interactive widget using IPython widgets for real-time score predictions.
Components: Dropdowns for venue, batting team, bowling team, striker, bowler, and a button for triggering predictions.
Functionality: Enables users to select conditions and receive dynamic score predictions in real-time.

# 6. Evaluation and Case Demonstration

## Model Evaluation

- We have predicted using the trained neural network on the testing data.
- The variable predictions contains the predicted total run scores for the test set based on the model's learned patterns.

```
# Make predictions
predictions = model.predict(X_test_scaled)

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
mae=mean_absolute_error(y_test,predictions)
mse=mean_squared_error(y_test,predictions)
r2=r2_score(y_test,predictions)
print('Mean absolute error',mae)
print('Mean squared error',mse)
print('R2_score',r2)
```

```
713/713 [==============================] - 1s 2ms/step
Mean absolute error 19.974337082394275
Mean squared error 728.475888101443
R2_score 0.13624611641795348
```

## CASE DEMONSTRATION:

Suppose we have an upcoming IPL match scheduled between Team A and Team B at Venue X. Cricket enthusiasts are eager to know the expected score of Team A in this specific match. We will use the IPL score prediction system to generate a real-time prediction.

Case Demonstration Steps:

Input Selection:

User selects the match conditions:

Venue: Venue X

Batting Team: Team A

Bowling Team: Team B

Striker: [Select a prominent batsman from Team A]

Bowler: [Select a key bowler from Team B]

Prediction Widget Interaction:

User interacts with the IPL score prediction widget, utilising dropdowns for venue, batting team, bowling team, striker, and bowler.

After making the selections, the user clicks on the "Predict Score" button.

Prediction Process:

The widget triggers the prediction function, passing the selected conditions to the model.

The model processes the input using the trained neural network architecture.

Real-Time Prediction:

The predicted IPL score for Team A in the upcoming match is displayed in real-time on the output section of the widget.

Result Analysis:

The predicted score is accompanied by a confidence level or prediction interval, providing users with insights into the model's certainty.

Interpretation:

Users can interpret the predicted score, considering factors such as the venue, teams, and player statistics.

# 7 Conclusion

The IPL Score Prediction project represents a significant achievement at the intersection of machine learning and cricket analytics. Through a rigorous process of data analysis, model development, and comprehensive validation, we have successfully engineered a tool that not only forecasts IPL match scores with a high degree of accuracy but also serves as a valuable asset for cricket enthusiasts, teams, and fantasy sports players seeking informed insights.

Our IPL Score Prediction project proves to be a valuable tool for cricket enthusiasts, teams, analysts, and fantasy sports players. The accurate predictions, user-friendly interface, and real-time updates make it a game-changer in the world of cricket analytics. As we continue to refine and enhance our model based on user feedback, we look forward to expanding its applications and providing even more valuable insights in the future. Experience the power of data-driven cricket analysis with our IPL Score Prediction project!

# 8 References


[1]https://www.sciencedirect.com/science/article/abs/pii/S0893608018303332

[2] T. A. Severini, Analytic methods in sports: Using mathematics and statistics to understand data from baseball, football, basketball, and other sports. Chapman and Hall/CRC, 2014.

[3] H. Ghasemzadeh and R. Jafari, “Coordination analysis of human movements with body sensor networks: A signal processing model to evaluate baseball swings,” IEEE Sensors Journal, vol. 11, no. 3, pp. 603–610, 2010

[4]R. Rein and D. Memmert, “Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science,” SpringerPlus, vol. 5, no. 1, p. 1410, 2016