

T20 Totalitarian: Mastering Score Predictions

Team ID	PNT2022TMID593080
Project Name	T20 Totalitarian: Mastering Score Predictions

Team Size : 4

Team Leader : V Vijayaalayan

Team member : A.S. Jayashri

Team member : Ragul V

Team member : R. Nachiyappan

INTRODUCTION

Project Overview:

T20 Totalitarian: Mastering Score Predictions

Objective: Develop an end-to-end machine learning solution for predicting the T20 score of the batting team. The primary goal is to leverage machine learning algorithms to extract relevant features from input data and accurately predict T20 cricket scores.

Key Components:

1. Data Collection:
 - Gather cricket-related data, including historical match data, player statistics, and other relevant information.
 - Utilize data sources such as Cricket Data APIs, Real-Time Data Streams, and Social Media APIs for comprehensive data collection.
2. Data Preprocessing:
 - Clean and preprocess the collected data to ensure consistency and remove any inconsistencies or outliers.
 - Handle missing values, standardize features, and prepare the dataset for model training.
3. Feature Extraction:
 - Identify and extract key features from the dataset that contribute to the prediction of T20 scores.
 - Utilize machine learning techniques to enhance feature selection and extraction.
4. Model Training:

- Implement machine learning algorithms, with a focus on Convolutional Neural Networks (CNN) for effective score prediction.
- Train the model using historical data to learn patterns and correlations.
- 5. Real-Time Adaptation:
 - Implement mechanisms for real-time adaptation, allowing the model to adjust and improve predictions based on live match data.
- 6. Web Application:
 - Develop a user-friendly web application that allows cricket enthusiasts and analysts to input relevant match details and receive score predictions.
 - Integrate the trained machine learning model into the web app for seamless predictions.
- 7. Solution Evaluation:
 - Assess the accuracy and performance of the machine learning model using metrics such as Mean Squared Error (MSE), R-squared, and others.
 - Continuously evaluate and improve the model based on real-world predictions and user feedback.

Benefits:

- Provide accurate T20 score predictions for cricket enthusiasts and analysts.
- Showcase the capabilities of machine learning in sports analytics.
- Demonstrate the potential of Convolutional Neural Networks in predicting cricket scores.

Skills Required:

- Machine Learning (CNN)
- Web App Development

Project Purpose:

The purpose of this project is to develop an end-to-end machine learning solution for predicting the t20 score of the batting team. The proposed solution involves the use of Machine learning algorithms to extract relevant features from the input and predict the accurate score. Creating an end-to-end machine learning project to predict T20 cricket scores can offer a range of benefits, both from a sports analytics perspective and as a showcase of machine learning capabilities.

LITERATURE SURVEY

- **Existing problem:**

Model Overfitting:

To Identify challenges related to model overfitting, especially in the context of predicting T20 scores where match dynamics can change rapidly.

Data Quality and Consistency:

To Explore literature discussing challenges related to the quality and consistency of cricket data, which can impact model training and predictions.

User Interface Challenges:

Look for studies that address challenges in designing user-friendly interfaces for sports prediction applications, considering the diverse user base.

Real-Time Adaptation Issues:

Identify issues related to adapting machine learning models in real-time based on live match data, considering the dynamic nature of T20 matches.

References:

<https://www.geeksforgeeks.org/artificial-intelligence-and-machine-learning-technologies-in-sports/>

<https://www.geeksforgeeks.org/ipl-score-prediction-using-deep-learning/>

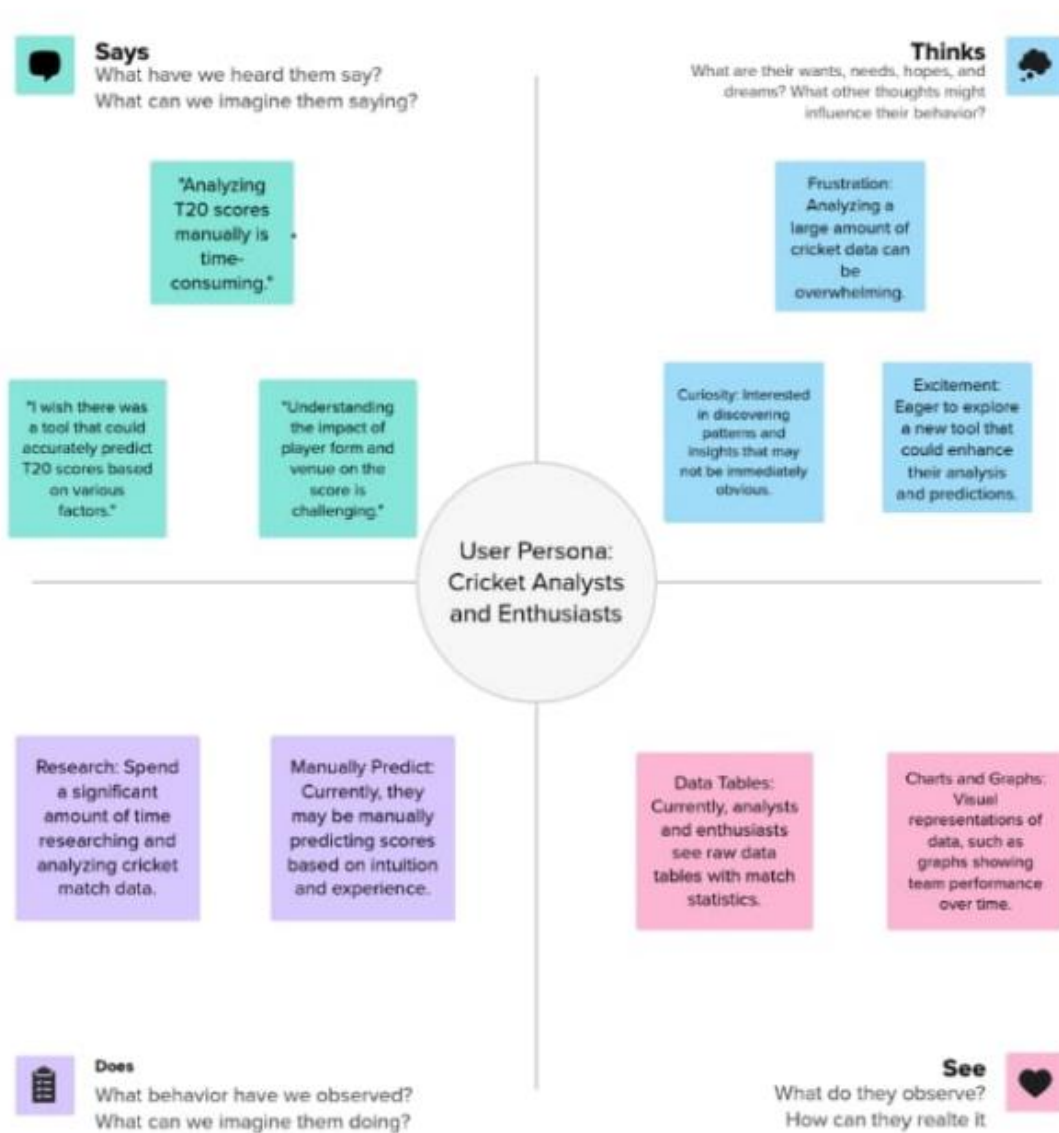
<https://www.analyticsvidhya.com/blog/2022/05/ipl-team-win-prediction-project-using-machine-learning/>

Problem Statement Definition

In the context of T20 cricket, accurately predicting the batting team's score is a challenging task that involves various dynamic factors such as player performance, match conditions, and opposition strength. The traditional methods of score prediction often lack precision and fail to harness the full potential of machine learning algorithms. To address this gap, the T20 Totalitarian project aims to develop an end-to-end machine learning solution that leverages advanced algorithms to extract relevant features from historical and real-time data, ultimately providing a more accurate prediction of the T20 cricket team's score.

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas:



Ideation & Brainstorming:

Brainstorm

Vijayalayan

Historical
Performance
Analysis

Player Form
and Conditions

Raghul

Team
Composition
and Strategy

Player Head-
to-Head
Statistics

Jayashri

Inning
Progression
Model

Machine
Learning
Algorithms

Nachiyappan

Live Data
Integration

Social Media
Integration

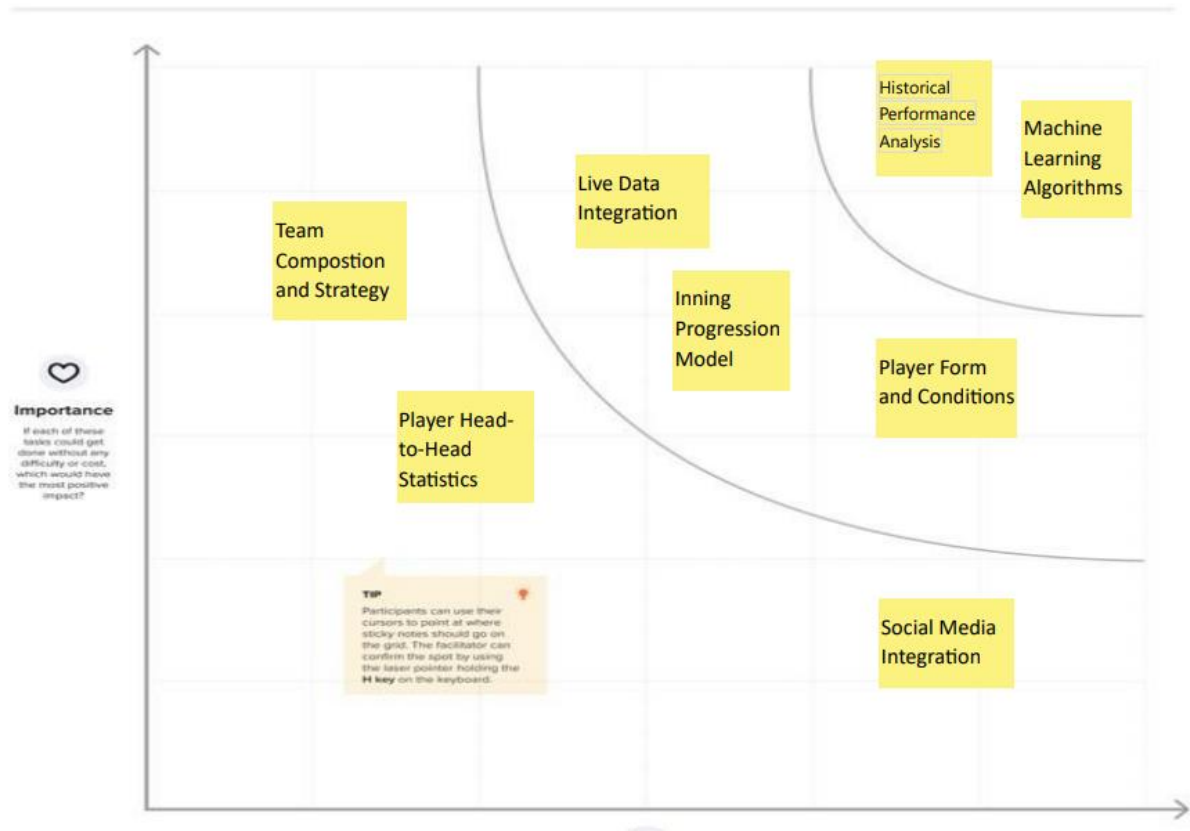
Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⌚ 20 minutes



REQUIREMENT ANALYSIS

Functional requirement:

1. Data Ingestion:
 - The system should be able to gather historical T20 match data, player statistics, and relevant contextual information from various sources.
 - Real-time data streams, including live match statistics and social media sentiments, should be ingested seamlessly.
2. Feature Extraction:
 - Implement feature engineering techniques to extract meaningful features from the collected data for model training.
 - Ensure that features encompass player performance, match conditions, historical trends, and other relevant factors.
3. Model Training:

- Utilize machine learning algorithms, potentially including Convolutional Neural Networks (CNN), to train a predictive model.
- The model should continuously learn and adapt to new data, improving its predictive accuracy over time.
- 4. Real-Time Adaptation:
 - Integrate mechanisms to adapt the model in real-time based on incoming live match statistics and other dynamic factors.
 - Ensure that the adaptation process is seamless and does not disrupt the overall prediction capabilities.
- 5. Score Prediction:
 - Provide a robust mechanism for predicting the T20 cricket team's score based on the trained model and incoming data.
 - Ensure that the predictions are accurate, timely, and accessible through the user interface.
- 6. Web Application:
 - Develop a user-friendly web application interface for users to interact with the system.
 - The interface should allow users to input relevant parameters, visualize predictions, and receive real-time updates.

Non-Functional requirements:

1. Scalability:
 - The system should be scalable to handle an increasing volume of historical data and real-time inputs without compromising performance.
2. Reliability:
 - Ensure high reliability in predicting T20 scores, minimizing errors, and providing consistent results.
3. User Interface Design:
 - The web application should have an intuitive and visually appealing design, promoting user engagement.
4. Real-Time Responsiveness:
 - Achieve low-latency real-time adaptation of the prediction model to ensure responsiveness during live matches.
5. Security:
 - Implement security measures to protect the integrity of the data, ensuring that it is not tampered with or compromised.
6. Compatibility:
 - Ensure compatibility with a variety of devices and browsers for a seamless user experience.
7. Performance:
 - The system should deliver high-performance predictions, meeting or exceeding user expectations for accuracy and speed.
8. Maintainability:
 - Design the system with modular components and clear documentation to facilitate ease of maintenance and future updates.

PROJECT DESIGN:

Data Flow Diagrams & User Stories

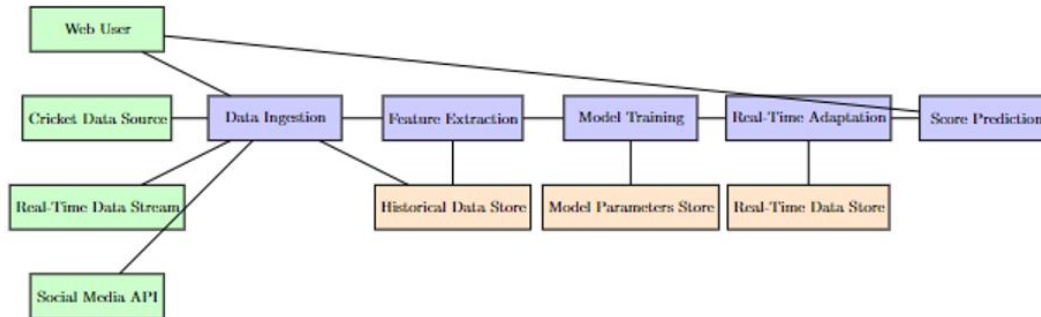


Figure 1: T20 Totalitarian Data Flow Diagram

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Cricket Enthusiast	Score Prediction	US001	As a cricket enthusiast, I want to access the T20 Totalitarian platform to get accurate predictions of the batting team's score in T20 matches.	1. The platform should provide a user-friendly interface to input match-related data. 2. The prediction should be displayed in a clear and understandable format. 3. The predicted score should be updated in real-time during live matches.	High	1.0

Data Analyst	Feature Extraction	US002	As a data analyst, I want to have access to advanced feature extraction tools to analyze and select relevant features for the machine learning model.	1. Feature extraction should be customizable based on specific match or player requirements. 2. The extracted features should be available for further analysis and model training.	High	1.0
Team Manager	Real-Time Adaptation	US003	As a team manager, I want to receive real-time insights and adaptations based on the T20 Totalitarian predictions during live matches.	1. The platform should provide real-time notifications and updates on predicted scores. 2. Adaptations should include suggestions for strategic changes based on the evolving match dynamics.	Low	1.1
System Administrator	System Monitoring	US004	As a system administrator, I want to monitor the health and performance of the T20 Totalitarian system to ensure its reliability.	1. The system should generate alerts for any anomalies or downtimes. 2. Performance logs should be available for analysis and optimization.	Medium	1.2

Solution Architecture

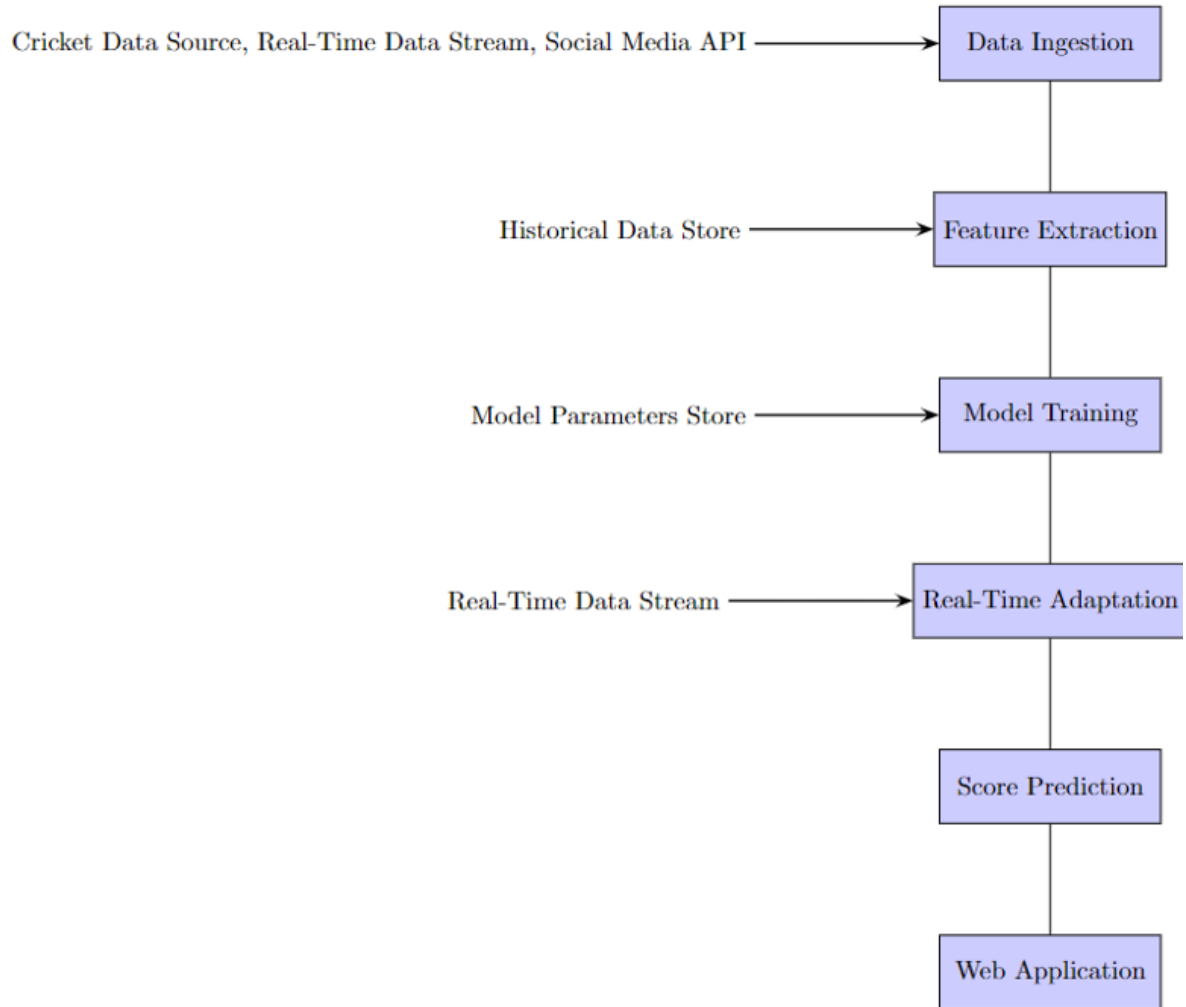


Figure 1: T20 Totalitarian Solution Architecture

PROJECT PLANNING & SCHEDULING

Technical Architecture:

The technical architecture of a T20 score predictor refers to the high-level design and structure of the system, outlining how different components interact to achieve the desired functionality. For a T20 score predictor, the technical architecture might include:

Data Ingestion	Retrieve historical match data, real-time match updates, and player statistics.
Data Processing	Analyze and preprocess the data to extract relevant features for prediction models.
Machine Learning Models	Implement algorithms for predicting T20 scores based on historical performance, player form, and other factors.
User Interface	Develop a user-friendly interface for users to interact with the predictor, input preferences, and view predictions.
Live Data Integration	Incorporate mechanisms to continuously update predictions based on real-time match data.
Notification System	Implement a system to alert users about significant events and score updates during matches.

Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Score Prediction Model	USN-1	As a cricket fan, I want a model to predict T20 scores	3	High	Ragul
Sprint-2	Feature Extraction	USN-2	Extract relevant features from match data	2	medium	Jayashri
Sprint-3	Real-time Data Integration	USN-3	Integrate real-time data during matches	3	High	Nachiyappan
Sprint-4	Web App Development	USN-4	Develop a web app for users to interact with the predictions	3	High	Vijayaalayan Vinod

Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	15 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	40	21 Nov 2022

CODING & SOLUTIONING:

Importing the libraries

```
In [4]: import pandas as pd
        from joblib import dump
```

Reading the dataset

```
In [5]: hf = pd.read_csv("ipl.csv")
```

```
In [6]: df.head()
```

```
Out[6]:
```

	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

Filtering the columns

```
In [7]: columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']
```

```
In [8]: df.drop(labels=columns_to_remove, axis=1, inplace=True)
```

```
In [9]: df.head()
```

```
Out[9]:
```

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222

The code `df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isin(consistent_teams))]` filters a DataFrame (df) to include only rows where both the 'bat_team' and 'bowl_team' columns have values that are present in the list `consistent_teams`.

```
In [10]: consistent_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',  
                             'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',  
                             'Delhi Daredevils', 'Sunrisers Hyderabad']
```

```
In [11]: df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isin(consistent_teams))]
```

```
In [12]: df.head()
```

```
Out[12]:
```

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222

The code `df = df[df['overs'] >= 5.0]` filters a DataFrame (df) to include only rows where the value in the 'overs' column is greater than or equal to 5.0.

```
In [11]: df = df[df['overs']>=5.0]
```

```
In [12]: df.head()
```

```
Out[12]:
```

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
32	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	0	5.1	59	0	222
33	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.2	59	1	222
34	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.3	59	1	222
35	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.4	59	1	222
36	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.5	58	1	222

The code `df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))` is used to convert the 'date' column in a DataFrame (df) from a string format to a datetime format.

```
In [13]: from datetime import datetime
```

```
In [14]: df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
```

```
In [15]: df.head()
```

```
Out[15]:
```

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222

The code `encoded_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team'])` is used to perform one-hot encoding on the categorical columns 'bat_team' and 'bowl_team' in the DataFrame df. One-hot encoding is a technique used to convert categorical variables into a binary matrix (0s and 1s) to make them suitable for machine learning algorithms.

```
In [16]: encoded_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team'])
```

```
In [17]: encoded_df.head()
```

```
Out[17]:
```

	date	runs	wickets	overs	runs_last_5	wickets_last_5	total	bat_team_Chennai Super Kings	bat_team_Delhi Daredevils	bat_team_Kings XI Punjab	...	bat_team_Royal Challengers Bangalore	bat_team_Sunrisers Hyderabad
32	2008-04-18	61	0	5.1	59	0	222	0	0	0	...	0	0
33	2008-04-18	61	1	5.2	59	1	222	0	0	0	...	0	0
34	2008-04-18	61	1	5.3	59	1	222	0	0	0	...	0	0
35	2008-04-18	61	1	5.4	59	1	222	0	0	0	...	0	0
36	2008-04-18	61	1	5.5	58	1	222	0	0	0	...	0	0

5 rows × 23 columns

`encoded_df[['date', ... 'total']]`: This part of the code is selecting a subset of columns from the DataFrame `encoded_df`. The specified column names include both the original columns ('date', 'overs', 'runs', 'wickets', 'runs_last_5', 'wickets_last_5', 'total') and the one-hot encoded columns for the batting and bowling teams.

```
In [17]: encoded_df = encoded_df[['date', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils', 'bat_team_Kings XI Punjab',
    'bat_team_Kolkata Knight Riders', 'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
    'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
    'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils', 'bowl_team_Kings XI Punjab',
    'bowl_team_Kolkata Knight Riders', 'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
    'bowl_team_Royal Challengers Bangalore', 'bowl_team_Sunrisers Hyderabad',
    'overs', 'runs', 'wickets', 'runs_last_5', 'wickets_last_5', 'total']]
```

Splitting data into train and test sets

This code is splitting the DataFrame `encoded_df` into training and testing sets based on the year in the 'date' column. The goal is to create two subsets of data: one for training a machine learning model (data from years up to 2016) and another for testing the model's performance (data from 2017 onwards).

These lines of code are creating the target variable arrays (`y_train` and `y_test`) for the training and testing sets. The target variable in this case is 'total', representing the total runs scored in a T20 cricket match.

```
In [18]: X_train = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year <= 2016]
    X_test = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year >= 2017]

In [19]: y_train = encoded_df[encoded_df['date'].dt.year <= 2016]['total'].values
    y_test = encoded_df[encoded_df['date'].dt.year >= 2017]['total'].values
```

`X_train.drop(labels='date', axis=True, inplace=True)`: This line removes the 'date' column from the training set (`X_train`). The `drop` method is used to eliminate columns or rows from a DataFrame. The argument `labels='date'` specifies the column to be dropped. The `axis=True` indicates that the operation is performed along columns. Finally, `inplace=True` means the changes are made directly to the existing DataFrame (`X_train`) without the need to create a new DataFrame.

```
In [20]: X_train.drop(labels='date', axis=True, inplace=True)
    X_test.drop(labels='date', axis=True, inplace=True)
```

Linear Regression

This line trains the linear regression model using the training data. The `fit` method takes two arguments:

`X_train`: The feature matrix of the training set, which includes all the input features used for training.

`y_train`: The target variable of the training set, which contains the corresponding output values.

After this step, the regressor object is ready to make predictions on new, unseen data using the learned model.

```
In [21]: from sklearn.linear_model import LinearRegression

    regressor = LinearRegression()
    regressor.fit(X_train, y_train)
```

```
Out[21]: LinearRegression
    LinearRegression()
```

Model Deployment

This line imports the `pickle` module, which is a standard Python module for serializing and deserializing objects.

After running this code, the trained linear regression model will be serialized and saved to a binary file named 'T20-lr-model.pkl'.

```
filename = 'innings-score-lr-model.joblib'
dump(regressor, filename)
```

```
import pickle
pickle.dump(regressor, open('T20-lr-model.pkl', 'wb'))
```

```
import pickle

filename = 'T20-lr-model.pkl'
regressor = pickle.load(open(filename, 'rb'))
```

To calculate Mean squared error and r squared

```
In [22]: y_pred = regressor.predict(X_test)
```

```
In [24]: from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

Mean Squared Error: 403.22328348760095
```

```
In [25]: from sklearn.metrics import r2_score

# Calculate R-squared
r2 = r2_score(y_test, y_pred)
print(f'R-squared: {r2}')

R-squared: 0.607038043216662
```

```
In [26]: X_train
```

Out[26]:

id	...	bowl_team_Kolkata Knight Riders	bowl_team_Mumbai Indians	bowl_team_Rajasthan Royals	bowl_team_Royal Challengers Bangalore	bowl_team_Sunrisers Hyderabad	overs	runs	wickets	runs_last_5	wickets_last_5
0	...	0	0	0	1	0	0.1	1	0	1	0
0	...	0	0	0	1	0	0.2	1	0	1	0
0	...	0	0	0	1	0	0.2	2	0	2	0
0	...	0	0	0	1	0	0.3	2	0	2	0
0	...	0	0	0	1	0	0.4	2	0	2	0
...
0	...	0	0	0	1	0	19.2	194	7	54	4
0	...	0	0	0	1	0	19.3	200	7	56	4
0	...	0	0	0	1	0	19.4	201	7	56	4
0	...	0	0	0	1	0	19.5	202	7	57	4

```
In [28]: from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, y_pred)
```

```
In [30]: print(f'Mean Absolute Error: {mae}')

Mean Absolute Error: 14.861585174238238
```

```
In [36]: import numpy as np
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')

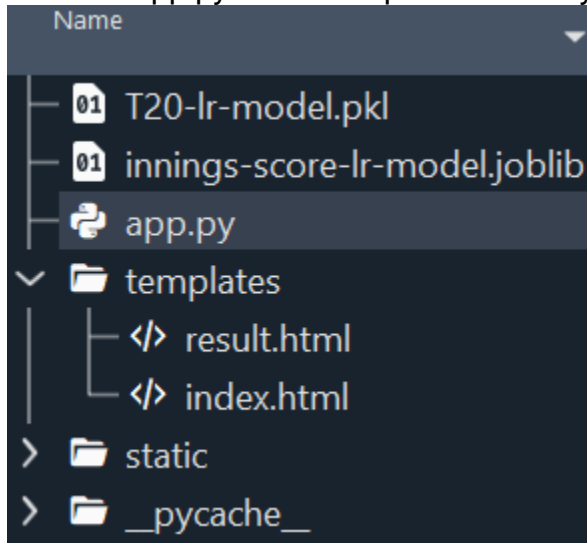
Root Mean Squared Error: 20.08042040116693
```

Thus we have done the following:

- 1- Data Preprocessing and some Exploratory Data Analysis to understand the data
- 2- Data preparation: Feature Engineering and Scaling
- 3- Feature Selection using RFE and Model Building
- 4- Regression Assumptions Validation and Outlier Removal
- 5- Rebuilding the Model Post Outlier Removal: Feature Selection & RFE
- 6- Removing Multicollinearity, Model Re-evaluation and Assumptions Validation

WEB APP DEVELOPMENT

Create a app.py file and import necessary packages:



Importing the flask to deploy it for machine learning:


```
C:\Users\Vijayaalayan\Desktop\Flask\app.py
index.html X result.html X app.py X
1  from flask import Flask, render_template, request
2
3  import numpy as np
4  import pickle
5
6  filename = 'T20-lr-model.pkl'
7  regressor = pickle.load(open(filename, 'rb'))
8
9  app = Flask(__name__)
10
11  @app.route('/')
12  def home():
13      return render_template('index.html')
14
15  @app.route('/predict', methods=['POST'])
16  def predict():
17      temp_array = list()
18      if request.method == 'POST':
19
```

Accepting the input from user and prediction:

```

15 @app.route('/predict', methods=['POST'])
16 def predict():
17     temp_array = list()
18     if request.method == 'POST':
19
20         batting_team = request.form['batting-team']
21         if batting_team == 'Chennai Super Kings':
22             temp_array = temp_array + [1,0,0,0,0,0,0,0]
23         elif batting_team == 'Delhi Daredevils':
24             temp_array = temp_array + [0,1,0,0,0,0,0,0]
25         elif batting_team == 'Kings XI Punjab':
26             temp_array = temp_array + [0,0,1,0,0,0,0,0]
27         elif batting_team == 'Kolkata Knight Riders':
28             temp_array = temp_array + [0,0,0,1,0,0,0,0]
29         elif batting_team == 'Mumbai Indians':
30             temp_array = temp_array + [0,0,0,0,1,0,0,0]
31         elif batting_team == 'Rajasthan Royals':
32             temp_array = temp_array + [0,0,0,0,0,1,0,0]
33         elif batting_team == 'Royal Challengers Bangalore':
34             temp_array = temp_array + [0,0,0,0,0,0,1,0]
35         elif batting_team == 'Sunrisers Hyderabad':
36             temp_array = temp_array + [0,0,0,0,0,0,0,1]
37
38
39         bowling_team = request.form['bowling-team']

```

```

        bowling_team = request.form['bowling-team']
        if bowling_team == 'Chennai Super Kings':
            temp_array = temp_array + [1,0,0,0,0,0,0,0]
        elif bowling_team == 'Delhi Daredevils':
            temp_array = temp_array + [0,1,0,0,0,0,0,0]
        elif bowling_team == 'Kings XI Punjab':
            temp_array = temp_array + [0,0,1,0,0,0,0,0]
        elif bowling_team == 'Kolkata Knight Riders':
            temp_array = temp_array + [0,0,0,1,0,0,0,0]
        elif bowling_team == 'Mumbai Indians':
            temp_array = temp_array + [0,0,0,0,1,0,0,0]
        elif bowling_team == 'Rajasthan Royals':
            temp_array = temp_array + [0,0,0,0,0,1,0,0]
        elif bowling_team == 'Royal Challengers Bangalore':
            temp_array = temp_array + [0,0,0,0,0,0,1,0]
        elif bowling_team == 'Sunrisers Hyderabad':
            temp_array = temp_array + [0,0,0,0,0,0,0,1]

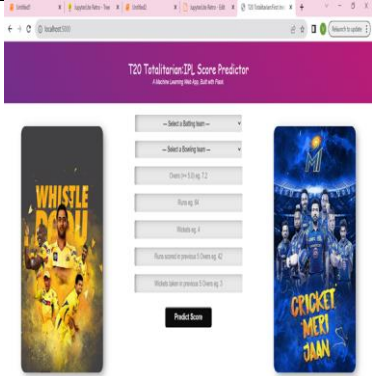
        overs = float(request.form['overs'])
        runs = int(request.form['runs'])
        wickets = int(request.form['wickets'])
        runs_in_prev_5 = int(request.form['runs_in_prev_5'])
        wickets_in_prev_5 = int(request.form['wickets_in_prev_5'])

        temp_array = temp_array + [overs, runs, wickets, runs_in_prev_5, wickets_in_prev_5]

        data = np.array([temp_array])
        my_prediction = int(regressor.predict(data)[0])

        return render_template('result.html', lower_limit = my_prediction-10, upper_limit = my_prediction+5)
    
```

PERFORMANCE TESTING

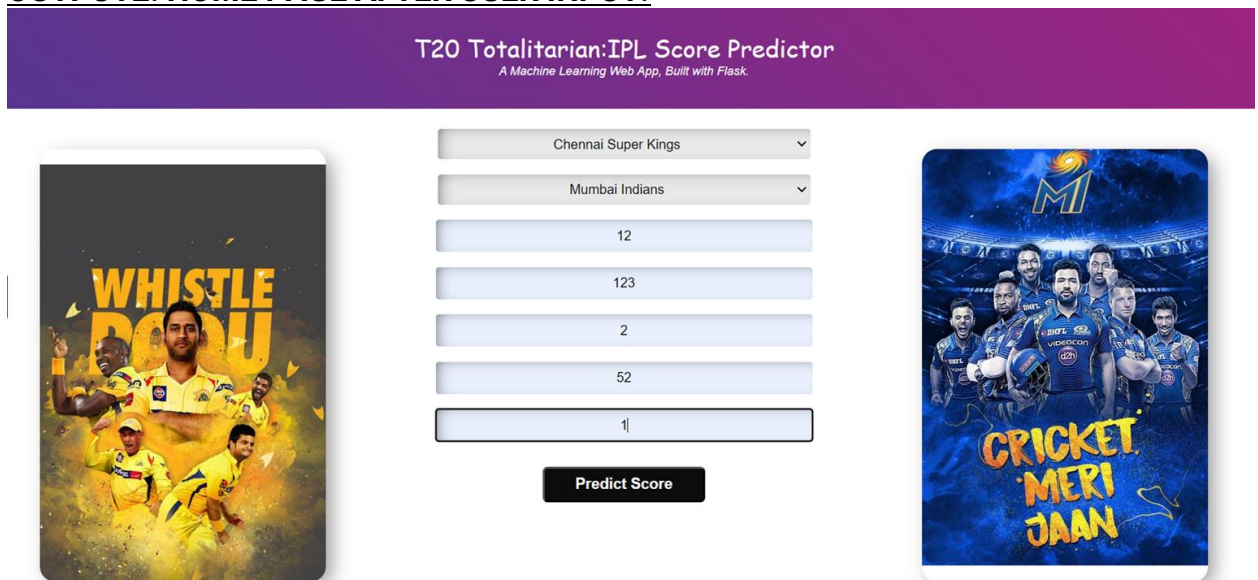
S.No.	Parameter	Values	Screenshot
1.	Model Summary	<p>This Project is designed to predict the Score of Indian Premier League using Regression Analysis with PYTHON, FLASK AND HTML.</p>	
2.	Accuracy	<p>Training Accuracy -</p> <p>R-squared: 0.607038043216662 Mean Squared Error: 403.22328348760095</p> <p>Validation Accuracy - R-squared: 0.607038043216662</p>	<p>Mean Squared Error: 403.22328348760095</p> <p>R-squared: 0.607038043216662</p>

RESULTS

OUTPUT 1:HOME PAGE



OUTPUT2: HOME PAGE AFTER USER INPUT:



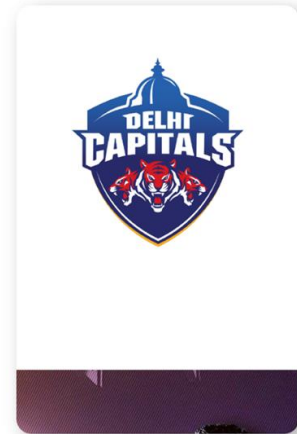
OUTPUT 3:RESULT PAGE

T20 Totalitarian:IPL Score Predictor

A Machine Learning Web App, Built with Flask.



The final predicted score (range): 187 to 202



← → ↻ localhost:5000/predict

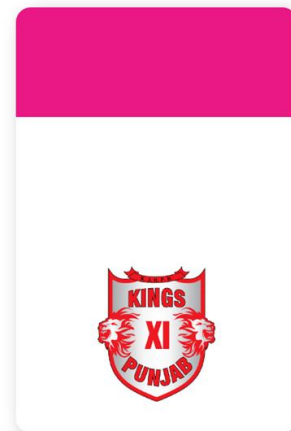
🔗 ☆ 📱 🔄 Relaunch to update

T20 Totalitarian:IPL Score Predictor

A Machine Learning Web App, Built with Flask.



The final predicted score (range): 195 to 210



ADVANTAGES & DISADVANTAGES

Advantages:

1. Improved Decision-Making in Cricket: Accurate score predictions can provide valuable insights for teams, coaches, and players, helping them make informed decisions during a T20 match.
2. Strategic Planning: Teams can use predicted scores to formulate game strategies, such as adjusting field placements, choosing bowlers, or setting batting orders.

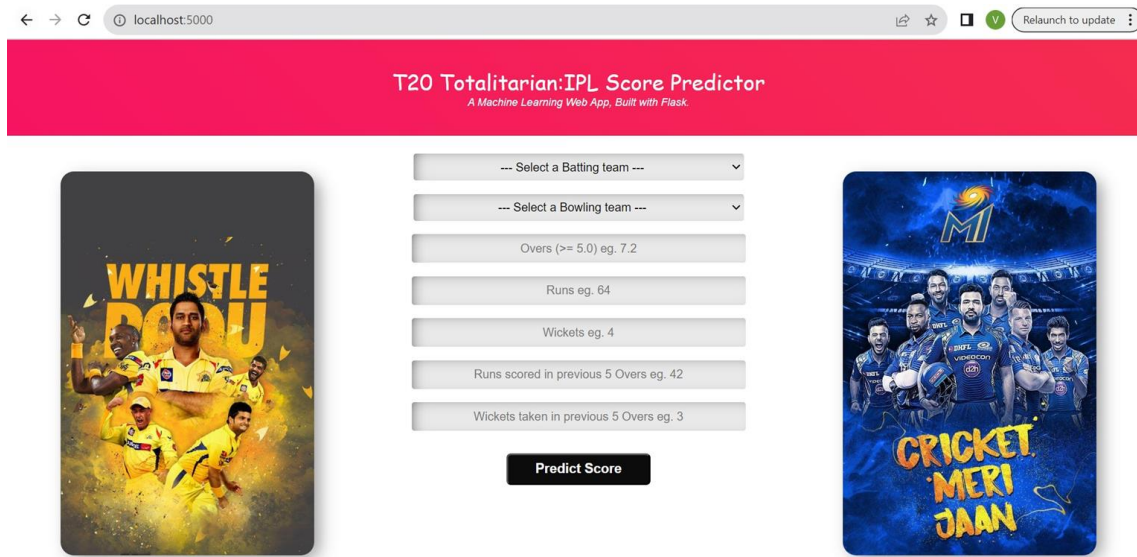
3. **Data-Driven Insights:** The project can showcase the power of data-driven decision-making in the sports domain, demonstrating how machine learning can enhance understanding and strategy.
4. **Fan Engagement:** Accurate score predictions can increase fan engagement by providing them with additional insights and predictions for upcoming matches.
5. **Scalability:** Once the model is trained and validated, it can be applied to various T20 matches, making it a scalable solution for predicting scores across different scenarios.
6. **Learning Opportunity:** The project offers an opportunity for individuals involved to learn and apply machine learning techniques in a real-world scenario, which can contribute to skill development in the field.

Disadvantages:

1. **Data Limitations:** The accuracy of predictions heavily relies on the quality and quantity of historical data available. Limited or biased data can lead to inaccurate predictions.
2. **Dynamic Nature of Cricket:** Cricket is a dynamic sport with multiple factors influencing the game, such as player form, weather conditions, and pitch variations. Incorporating all these variables accurately into a predictive model can be challenging.
3. **Model Complexity:** Developing an accurate predictive model may require a complex algorithm that is difficult to interpret or explain, making it challenging for stakeholders to understand the decision-making process.
4. **Overfitting:** There's a risk of overfitting the model to historical data, resulting in poor generalization to new, unseen data. Regular updates and retraining may be necessary to maintain accuracy.
5. **Ethical Considerations:** Predicting scores with high accuracy may raise ethical concerns, especially if the information is misused for betting purposes.
6. **User Adoption:** Coaches and teams may be hesitant to fully adopt machine learning predictions, relying instead on traditional methods and experience.

CONCLUSION

By leveraging machine learning algorithms, the T20 Totalitarian project contributes to the evolution of sports analytics. It demonstrates how data-driven insights can enhance decision-making processes in the dynamic and strategic realm of T20 cricket. The use of Flask and Python ensures scalability, allowing the model to be applied to various T20 matches. Regular updates and continuous monitoring can enhance the model's adaptability to changing conditions, ensuring relevance and accuracy over time. In conclusion, the T20 Totalitarian project, implemented using Flask and Python to create an end-to-end machine learning solution for predicting T20 cricket scores, represents a significant advancement in the field of sports analytics and showcases the capabilities of machine learning in a practical scenario.



FUTURE SCOPE

The future scope for the T20 Totalitarian project using Artificial Intelligence and Machine Learning for predicting T20 cricket scores is promising and can be extended in several directions:

1. Enhanced Predictive Models:
 - Continuously refine and enhance the machine learning models to improve prediction accuracy. Experiment with advanced algorithms and techniques to capture the dynamic and evolving nature of T20 cricket.
2. Incorporation of Real-time Data:
 - Integrate real-time data feeds during matches to make predictions based on the latest information, including live player statistics, weather conditions, and pitch behavior. This can enhance the model's adaptability to the current state of the game.
3. Player Performance Analytics:
 - Extend the project to include individual player performance predictions. Develop models that can forecast how specific players are likely to perform in a given match, considering their recent form, historical data, and situational factors.
4. Interactive User Interfaces:
 - Develop interactive and user-friendly interfaces for cricket enthusiasts, coaches, and teams. Provide detailed insights, visualizations, and explanations for the predictions, making the information accessible to a broader audience.
5. Integration with Fantasy Cricket Platforms:
 - Collaborate with fantasy cricket platforms to integrate score predictions into their systems. This can enhance the user experience for fantasy cricket players and contribute to the growing fantasy sports industry.
6. Collaboration with Cricket Teams and Analysts:
 - Work closely with cricket teams, coaches, and analysts to customize the model for specific teams and playing styles. Incorporate feedback from cricket experts to make the predictions more relevant and actionable for professional teams.
7. Development of Mobile Applications:

- Create mobile applications that provide on-the-go access to score predictions, match analyses, and real-time updates. This can cater to a broader audience, including fans, analysts, and team management.
- 8. Integration with Broadcasts and Commentary:
 - Explore collaborations with broadcasters to integrate score predictions into live broadcasts and commentary. This can add an additional layer of analysis and engagement for television and online viewers.
- 9. Global Expansion:
 - Expand the scope of the project to cover T20 leagues and tournaments worldwide. This can include international matches, franchise leagues, and other T20 competitions, providing a more comprehensive and global perspective.
- 10. Research and Publications:
 - Contribute to the academic and research community by publishing findings, methodologies, and advancements made in the field of sports analytics. This can foster collaboration, knowledge sharing, and further innovations in the domain.
- 11. Ethical Considerations and Fair Play:
 - Address ethical considerations related to the use of predictive models in sports. Collaborate with sports organizations to ensure fair play, responsible use of data, and adherence to ethical standards.

In summary, the future scope for the T20 Totalitarian project involves continuous improvement, expansion into new areas, and collaboration with various stakeholders to make the project more impactful and widely applicable in the world of T20 cricket and sports analytics.

APPENDIX

<https://github.com/smartinternz02/SI-GuidedProject-603836-1699951343>

<https://drive.google.com/file/d/1ecdTrVhOfIkubnPyEymwFhrV1qZaNz8c/view?usp=sharing>

