PROJECT MANUAL

DATE	2023-11-10
TEAM ID	Team-591679
PROJECT NAME	T20 TOTALITARIAN: MASTERING SCORE PREDICTIONS
MAZIMUM MARKS	4 Marks

INTRODUCTION:

1.1 PROJECT OVERVIEW

"T20 Totalitarian: Mastering Score Predictions" is a cutting-edge initiative that uses machine learning techniques to transform score prediction in Twenty20 cricket matches. The main objective of the project is to offer precise and meaningful forecasts for T20 cricket match scores so that fans, pundits, and cricket aficionados may make better judgments and become more involved in the game. This project aims to create and apply machine learning models for precise T20 cricket match score prediction. Gathering and preparing past T20 match data, choosing and training suitable machine learning methods, and assessing the models' performance on a holdout dataset are all part of the project.

1.2 PURPOSE

"T20 Totalitarian: Mastering Score Predictions using Machine Learning" aims to accomplish several important goals by utilizing cutting-edge technology and data analytic methods, including:

Precise Score Predictions: The main goal is to predict T20 cricket match scores with accuracy. The objective is to produce accurate forecasts for forthcoming matches by utilizing machine learning algorithms and assessing a variety of criteria, including player statistics, match conditions, historical data, and more.

Enhanced Fan Engagement: By providing insightful forecasts, the project hopes to enhance the experience of cricket aficionados and fans. Fans may watch matches with greater enjoyment, participate in debates, and make well-informed decisions by having access to reliable forecasts.

Strategic Insights for Analysts: The portal provides analysts, researchers, and cricket specialists with access to useful insights obtained through the use of predictive algorithms. Strategic planning, performance analysis, and a deeper comprehension of the variables impacting match outcomes can all benefit from these insights.

Making Well-Informed Decisions: Based on anticipated scores and important influencing variables, coaches, players, and team management can make well-informed decisions on tactics, lineups, and game plans by using the platform's forecasts and analysis.

Constant Model Improvement: The project acts as a training ground for machine learning models that are used to make sports predictions. The goal is to improve prediction accuracy and dependability over time via ongoing assessment, improvement, and adaptation in response to user feedback and changing data.

Technological Innovation: The study demonstrates technological innovation in the field of sports forecasting by fusing machine learning approaches with sports analytics. establishing a standard for using data-driven techniques to forecast the results of sporting events.

"T20 Totalitarian: Mastering Score Predictions using Machine Learning" aims to serve the requirements of cricket enthusiasts, analysts, teams, and other stakeholders by offering a dependable, accurate, and user-friendly platform that generates precise score predictions for T20 cricket matches.

2. LITERATURE SURVEY

2.1 Existing problem

"T20 Totalitarian: Mastering Score Predictions using Machine Learning" seeks to address the following issues or difficulties:

1. **Erroneous Forecasts**: Because there are so many variables impacting a match's outcome, traditional methods of forecasting T20 cricket match scores sometimes lack accuracy. This problem can be solved by machine learning, which analyses several variables to generate predictions that are more accurate.

2.**Subjectivity and Prejudice**: Forecasts made by humans may be inconsistent or untrustworthy due to biases, subjective judgments, or emotional considerations. By reducing bias, machine learning models can produce predictions that are more unbiased and based on data.

3. **Complexity of Match Dynamics**: Accurately predicting the results of T20 matches is difficult because to the dynamic components involved, which include player form, weather, pitch features, and team plans. Device Learning algorithms are able to manage the complexity by taking into account several interconnected elements at once.

4. **Restricted Data Evaluation**: For manual analysis, the sheer amount of previous T20 match data accessible can be daunting. Large-scale datasets may be thoroughly analysed thanks to machine learning, which also makes it possible to uncover important patterns and insights that traditional methods would have overlooked.

5. **Requirement for Enhancing Performance**: Coaches and teams are always looking for methods to improve and have an advantage over their competitors. To enhance team performance, precise score forecasts generated by machine learning models can help with strategic planning and decision-making.

6. **Involvement and Passion for the Game**: Providing accurate forecasts to cricket lovers and fans increases their engagement with the game. The site hopes to keep fans more engaged by offering intelligent predictions, which would build heightened curiosity and conversation around games.

7. **Sports Analytics Advancement**: The application of machine learning advances cricket-related sports analytics. Accurate score prediction pushes the frontiers of sports analysis by illuminating underlying trends and variables influencing game results.

8. **Ongoing Enhancement and Modification**: A system that continuously adapts and gets better at making predictions over time is necessary because player performance is constantly changing, match conditions are changing, and new data is becoming available. To continuously improve accuracy, machine learning models can be improved and updated with fresh data.

"T20 Totalitarian: Mastering Score Predictions using Machine Learning" can help address these issues and have a big impact on prediction accuracy, strategic cricket decision-making, and cricket engagement and interest overall.

2.2 References.

1. https://dspace.bracu.ac.bd/xmlui/handle/10361/4372

2. https://www.ijarnd.com/manuscript/cricket-score-and-winning-prediction-using-data-mining/

3. https://ieeexplore.ieee.org/abstract/document/9318077

4. https://ieeexplore.ieee.org/abstract/document/9740867

2.3 Problem Statement Definition.

T20 cricket is a dynamic sport with many contributing factors and unpredictable individual performances, it is still difficult to predict match scores with any degree of accuracy. The inability of spectators, commentators, and teams to make well-informed judgments and plans is a direct outcome of this unpredictability, which frequently leads to inaccurate or misleading score projections. The fundamental issue is the lack of a strong, data-driven predictive model that can produce accurate and trustworthy score forecasts for Twenty20 cricket matches by efficiently analysing player statistics, match conditions, historical data, and other influencing factors. Current prediction techniques frequently produce inconsistent and inaccurate projections because they lack the depth, accuracy, and flexibility needed to take into consideration the wide range of factors influencing match results. Researchers and practitioners have focused a great deal of emphasis on the difficult task of accurately predicting cricket match scores. Cricket is a complicated game where a lot of things can happen to affect how a match turns out. Historically, expert judgment and statistical analysis have been used to forecast cricket match scores. However, the precision and consistency of these procedures are limited.

3. IDEATION & PROPOSED SOLUTION

Objective:

This document's goal is to offer an empathy map that can be used to build a machine learning model that can forecast the results of T20 cricket matches. The target users' needs, desires, and pain points—as well as the opportunities and difficulties involved in developing the model—will all be identified with the aid of the empathy map.

Goal Users:

- Fans of T20 cricket

- Fans of sports betting
- analysts of cricket
- Fans of cricket data

Empathy Map Components:

- Graphs, charts, and match statistics comprise data visualization.
- Previous match predictions and their accuracy ratings are the prediction accuracy metrics.
- User Interface:
- The layout and design of the T20 Totalitarian application.

Need of T20 TOTALITARIAN

- Precise and trustworthy score forecasts
- An understanding of the variables influencing match results
- The capacity to make wise wagering choices.
- gaining an advantage over rivals in fantasy cricket leagues
- A research and analysis tool for T20 cricket trends

What an Enthusiast for Cricket Says and Does:

- Expresses Frustration: When the system is not intuitive or when predictions are off.
- For Clarity requests an explanation of the methodology used to create the predictions.
- Participates in the App: Data is matched, predictions are investigated, and the outcomes are compared with the inputs.

What an Enthusiast for Cricket Feels and Thinks:

- Wants Accuracy: Making accurate predictions is crucial for better decision-making.
- Curiosity: a desire to comprehend the prediction algorithms.
- Trust Issues: Doubt about the accuracy of forecasts made by AI.

Implications for T20 Totalitarian:

Design and Development:

- User-Friendly Interface: To make data interpretation easier, create a UI that is both aesthetically pleasing and intuitive.
- Algorithm Transparency: Give users tooltips or explanations to help them grasp prediction techniques.
- Put Accuracy First: Make constant improvements to the ML model to achieve greater prediction accuracy.
- Teaching Materials: Provide users with tutorials or guides to improve their comprehension of prediction models and statistical analysis.

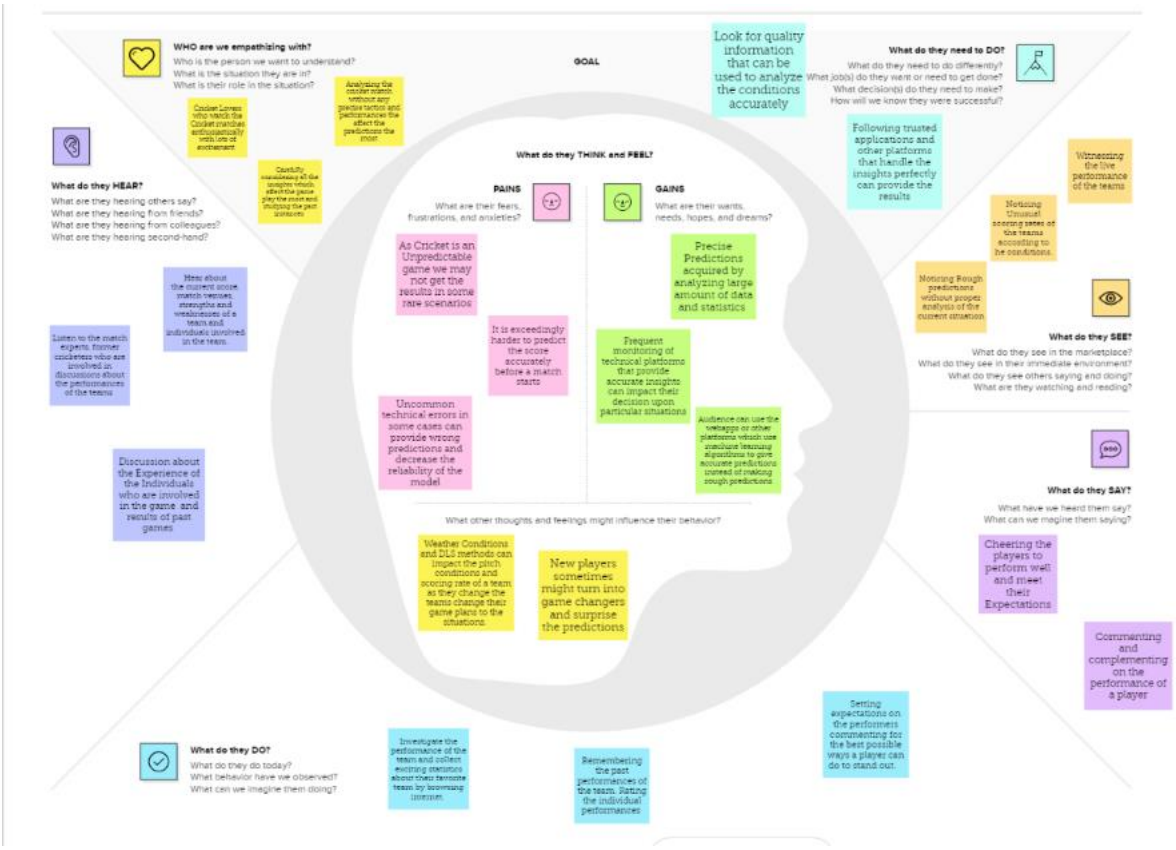
Interaction and Assistance:

- Feedback Loop: Put in place a system that allows users to offer comments and recommendations.
- Customer service: Make sure the system is responsive so that questions and issues from users can be quickly resolved.
- Community Engagement: Create a forum where users can talk about forecasts in order to build credibility and trust.

Conclusion:

In conclusion, creating a successful T20 Totalitarian application requires an understanding of the viewpoint of the cricket enthusiast. By keeping in mind their requirements regarding precision, openness, and ease of use, the system has the potential to develop into a useful resource for cricket fans looking for trustworthy match predictions. In order to better satisfy the needs and expectations of users, the T20 Totalitarian application has been developed and improved with guidance from this documentation, which offers a thorough understanding of the user persona's viewpoint.

EMPATHY MAP:



3.2 BRAINSTROMING MAP.

Brainstorming Session: Enhancing T20 Totalitarian

Goal:

To meet the demands of cricket fans, create a sophisticated machine learning-driven system that can accurately predict T20 cricket matches.

Key Subjects:

1. **Data Gathering and Preprocessing**: -

Compile past T20 match information from dependable sources.

- Preprocess and clean data to guarantee accuracy and consistency.

- Incorporate pertinent details like player statistics, venue, weather, and team performance.

2. **Machine Learning Model Development:** - Determine which machine learning algorithm produces the best predictions by experimenting with different models such as Random Forest, Gradient Boosting, Neural Networks, etc.

- Use the proper cross-validation techniques to train the model on past data.

- To enhance model performance, optimize the hyperparameters.

3. **Feature Engineering**: - Create new features or metrics (such as player form, recent team performance, or head-to-head statistics) that may have an impact on the outcome of a match.

- Analyze the sentiment surrounding recent team newsFor more information, use social media or the cross-val.

4. **Integrating Real-time Data**: - Establish a system to obtain and incorporate real-time match data into the prediction model while live matches are in progress.

. Create scrapers or APIs to collect data from real-time feeds and dynamically update predictions.

5. **Explainable AI**: - Guarantee the transparency of the ML model by offering justifications for forecasts.

Employ methods such as feature importance or SHAP values to clarify the ways in which specific variables affect predictions.

6. **User Interface and Experience**: - Create a simple, easy-to-use interface that presents statistical insights and predictions.

- Incorporate interactive graphs and visualizations to improve data understanding.

- Permit users to alter their preferences by enabling customization options.

7. **Continuous Improvement and Evaluation**: -

-Create a user feedback loop to get opinions on how accurate the predictions are.

-Use frequent validation and retraining of the model to enhance accuracy and adjust to shifting trends.

8. Engaging the Community:

-Encourage the development of a T20 Totalitarian app community.

-Organize Q&A sessions, forums, or webinars to get people talking about prediction techniques.

9. Considering Ethics:

-Make sure to use data responsibly and steer clear of biases in forecasts.

-Respect user privacy and follow data protection laws.

10. Collaborations and Partnerships:

-Work together with statisticians, analysts, or cricket specialists to validate and enhance prediction - models.

-Collaborate with cricket teams or leagues to gain access to data and insights.

Machine learning task:

-Using multiple regression to find the best one and run rate projection predictions the score based on r-run rate.

Literature survey:

We had to do the research about the previous work and analgise the case study and create the flowchart on how the process goes on and we are specified all the work between all the team member.

Team Analytics:

We were to analyse the recent team performance and analysis the action of the team based on situation of the team, and we were to evaluate how the preform on the last 4-5 overs. And take note of the strategies of each team and employees. Consider how teams tend to consolidate or accelerate their scoring. Examine the head-to-head record of both the teams. Analyse the batting and bowling strength of both the team. Predicting the score according to the fielding restriction and consider the player form and recent stats of every player and consider hoe the early wicket impact the innings score and analysis how team preformed under pressure and if no early wicket has gone then how may have the inning score have increased or decrease.

Data processing steps:

- Restructure the raw data and desired format.
- Analyse the data to get Quality Information.
- Consider the most important features.
- Looks trends or patterns in the data obtained.
- Look for trends in the scoring rate and wicket-taking ability.

Visualization of data:

- Creating an interface to visualize the prediction.
- Visualization of the data acquired to get good prediction.

Influence of external factors:

- Checking the impact of weather conditions.
- Measuring the importance of considering the pitch conditions.
- Analyze historical T20 matches at the specific ground.

Steps:

- Restructure the raw data to the desired format.
- Visualization of data acquire to get good prediction.
- Consider the most important features.
- Analyze the data to get Quality Information.
- Using multiple regressor to find the best one.
- Create a flowchart on how the process goes on.
- Looks for in the trends in the scoring rate and wicket taking ability.
- Looks for trends in the data obtained.

- Visualization of the data acquired to get goods prediction.
- Analyze the recent performance of the teams.
- Evaluate how the teams preformed the last 4-5 overs.
- Consider how teams tend to consolidate or accelerate their scoring.
- Analyze historical T20 matches at the specific ground level.
- Do research about the previous works.
- Predictions according to fielding restrictions.
- Analyze how teams have performed under pressure.
- Consider how early wickets can improve to score.
- Examine the Head-to-Head record of both team.
- Run Rate Projection: Predict the score based on the run rate.

In conclusion, by concentrating on data quality, model accuracy, user experience, and continuous improvement, this brainstorming session establishes the groundwork for leveraging machine learning to improve T20 Totalitarian. By putting these tactics into practice, a strong and trustworthy prediction system for Twenty20 cricket matches may be created, satisfying the demands of both stakeholders and cricket fans.

Various technical, user-centric, and ethical factors are covered in this brainstorming session to ensure that machine learning is successfully integrated into the T20 Totalitarian application for precise score predictions.

A brainstorming template is a structured framework designed to facilitate idea generation and problem-solving sessions. It typically comprises several key steps aimed at fostering creativity and encouraging participants to generate a multitude of ideas. The template begins by clearly defining the objective or problem to address. It sets ground rules, such as deferring judgment, emphasizing quantity over quality initially, and encouraging the building upon others' ideas. Warm-up activities may be incorporated to energize participants and create a conducive atmosphere for idea generation. Methods like free association, mind mapping, round-robin discussions, or brainwriting are employed to generate ideas collaboratively or independently. After gathering ideas, they're often categorized, organized, and evaluated to identify the most promising ones. Finally, the template guides the creation of an action plan, allocating responsibilities, setting timelines, and establishing follow-up sessions for review and adjustments. The template provides a structured approach while allowing flexibility to adapt to the specific needs and dynamics of the participants and the problem at hand. An Idea Prioritization Template is a structured approach used to assess and rank various ideas or solutions generated during brainstorming or problem-solving sessions. This template helps individuals or teams determine which ideas are most feasible, impactful, or aligned with the project goals.

BRAINSTROMING:
https://app.mural.co/t/t20scorepredictor7475/m/t20scorepredictor7475/1699377338137/ac95fbd44feaf1d4b12aeb32ed52342169452e44?sender=761d14d2-ac98-4344-bbc0-2e662d5b

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we develop a T20 score predictor that accurately forecasts the total runs a cricket team is likely to score in a given match, considering various factors such as pitch conditions, team composition, and historical performance, to enhance the viewing experience and aid in strategic decision-making for fans, coaches, and players?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Defer judgment.
- Go for volume.
- Encourage wild ideas.
- Listen to others.
- If possible, be visual.

2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Person 1

- Using Multiple Regressors to find the best one
- Creating an Interface to visualize the predictions
- Analyze the data to get Quality Information
- Checking the impact of weather conditions
- Analyzing the actions of the teams based on situation of team
- Evaluate how teams perform in the last 4-5 overs.
- Assess the batting and bowling strengths of both teams
- Look for trends in the scoring rate and wicket-taking ability.
- Predictions according to fielding restrictions

Person 2

- Do research about the previous works
- Create a flowchart on how the process goes on
- Restructure the raw data to the desired format
- Measuring the importance of considering the pitch conditions
- Analyzing case studies
- Take note of the strategies each team employs
- Consider the form and recent stats of key players
- Run Rate Projection: Predict the score based on the run rate.
- Consider how early wickets can impact the score.

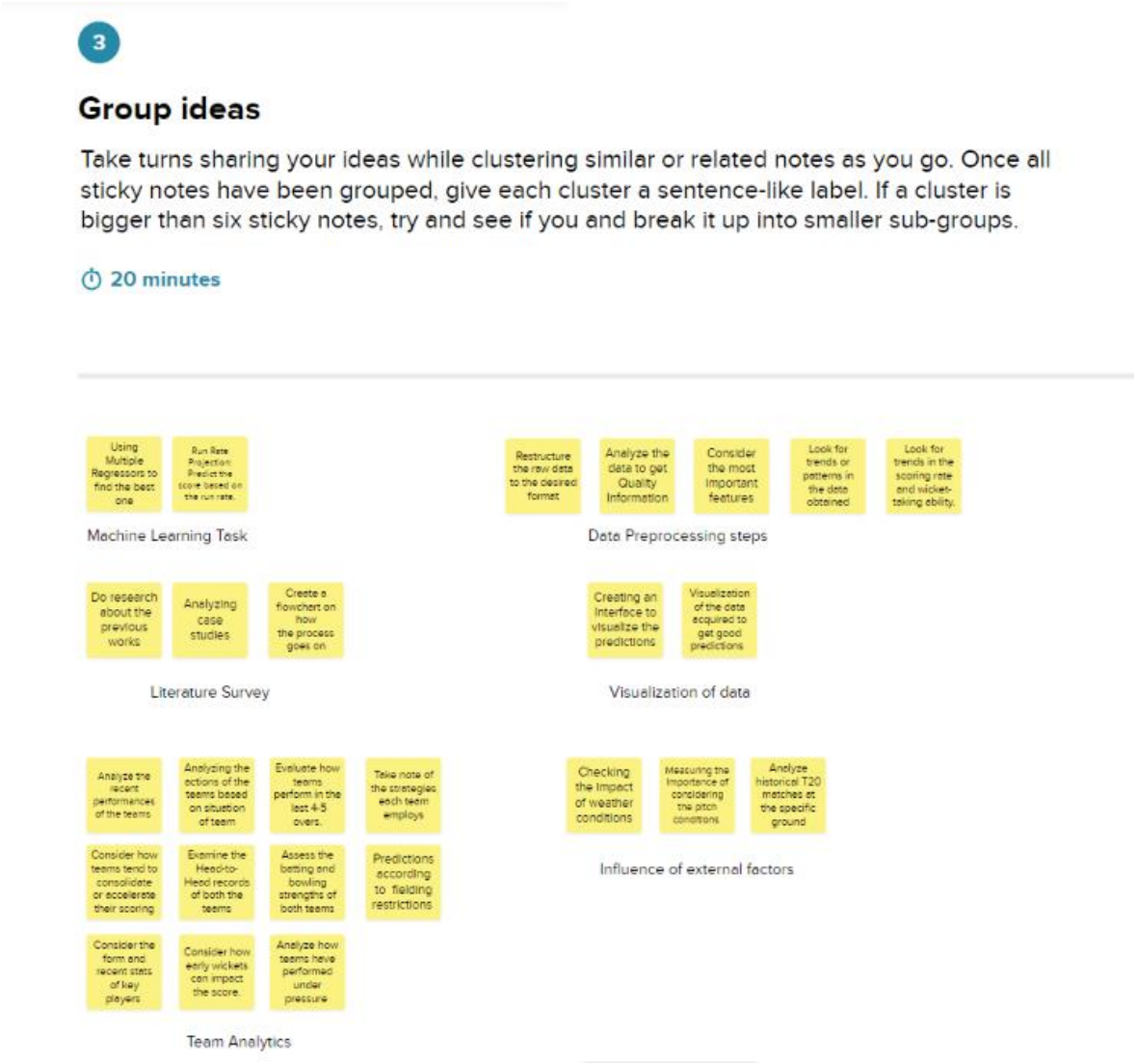
Person 3

- Visualization of the data acquired to get good predictions
- Consider the most important features
- Analyze the recent performances of the teams
- Examine the Head-to-Head records of both the teams
- Consider how teams tend to consolidate or accelerate their scoring
- Analyze historical T20 matches at the specific ground
- Look for trends or patterns in the data obtained
- Check betting odds and expert predictions for insights.

Person 5

Person 6

Person 7



4. REQUIREMENT ANALYSIS

4.1 Functional requirement.

Gathering and Combining Data:

The system will gather and compile extensive historical data, including match specifics, player statistics, venue facts, weather reports, pitch reports, and other relevant variables affecting match results. In order to maintain the predictive models current, it will make sure that new data is acquired and integrated on time.

Preprocessing Data and Creating Features:

The gathered data will be cleaned and preprocessed by the system, which will also handle missing values and format it for machine learning techniques.It will carry out feature engineering to produce fresh features or modify current ones in order to improve the model's capacity for prediction.

Training and Optimizing Models:

In order to train predictive models based on historical data, the system must make use of appropriate machine learning methods (such as neural networks, time series forecasting, and regression models). It will To increase prediction accuracy, optimize the models by choosing the best algorithms, adjusting hyperparameters, and making sure the training is reliable.

Making Predictions and Explaining Them:

The system will forecast the scores of future Twenty20 matches using the learned models. It will include thorough justifications and analyses of the anticipated results, emphasizing the major variables that influenced the forecasts (player form, previous team performance, weather conditions, etc.).

Assessment and Enhancement of Accuracy:

The system will use suitable metrics (e.g., Mean Squared Error, Accuracy, Precision, Recall, etc.) to assess the prediction accuracy. It will include methods for ongoing improvement, like feedback loops and model upgrades that take user feedback and evaluation outcomes into account.

Combining with the user interface:

The technology will incorporate the capability of prediction creation into an easily navigable interface. to administrators, analysts, and cricket aficionados. For users to make well-informed decisions, it will provide simple access to historical data, insights, analysis, and anticipated ratings.

Acceptance Standards:

Based on past performance assessment, the system will generate precise score projections for at least 80% of the forthcoming Twenty20 cricket matches. At least 24 hours before to the scheduled start of the match, predicted scores will be accessible. The reasons for the projected scores, which include a list of the major variables affecting the projections, will be available for users to view and comprehend.

4.2 Non-Functional requirements.

Non-functional Prerequisite for T20 Totalitarian: Acquiring Proficiency in Machine Learning Score Predictions

Achievement:

With a maximum processing time of 10 seconds per prediction request, the system must be able to produce precise score predictions for Twenty20 cricket matches, guaranteeing real-time access to forecasts with negligible latency. This non-functional requirement, which outlines the longest processing time allotted for score prediction generation, focuses on the system's performance. It guarantees quick access to forecasts for users, enabling a smooth and fast user experience.

Technology Stack:

Technical Architecture:

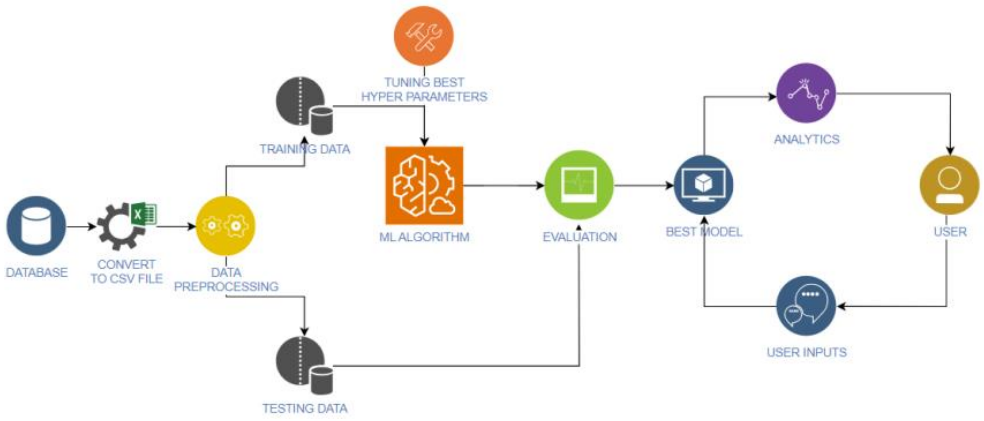


Table-1 : Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	Takes the input from the user and sends them to the machine learning model	HTML, CSS
2.	Application Logic -1	Initially, data is taken as the input from the dataset and preprocessed before sending it to the algorithm.	Python
3.	Application Logic -2	Tuning the best parameters to the algorithm and evaluating the results	Python
4.	Visualization	To visualize the data in terms of various infographics	IBM Cognos
5.	Dataset	To maintain the data of various matches.	MS EXCEL, IBM Cloud
6.	Machine Learning Model	To make predictions of the score based on the user input	CatBoost Model, XGBoost Model, Random Forest Model, XGB-RF Model, etc..
7.	Infrastructure	Local Server Configuration: The web application is deployed and hosted on port number: 8501	Local PC

The technology consists of a user interface that takes user input and sends it to a machine learning model using HTML and CSS. The application logic consists of preprocessing data from a dataset using Python, tuning optimal parameters, and evaluating results. The visualization tool uses IBM Cognos, while the dataset is maintained using MS EXCEL and IBM Cloud. The machine learning model predicts scores based on user input using models like CatBoost, XGBoost, Random Forest, and XGB-RF.

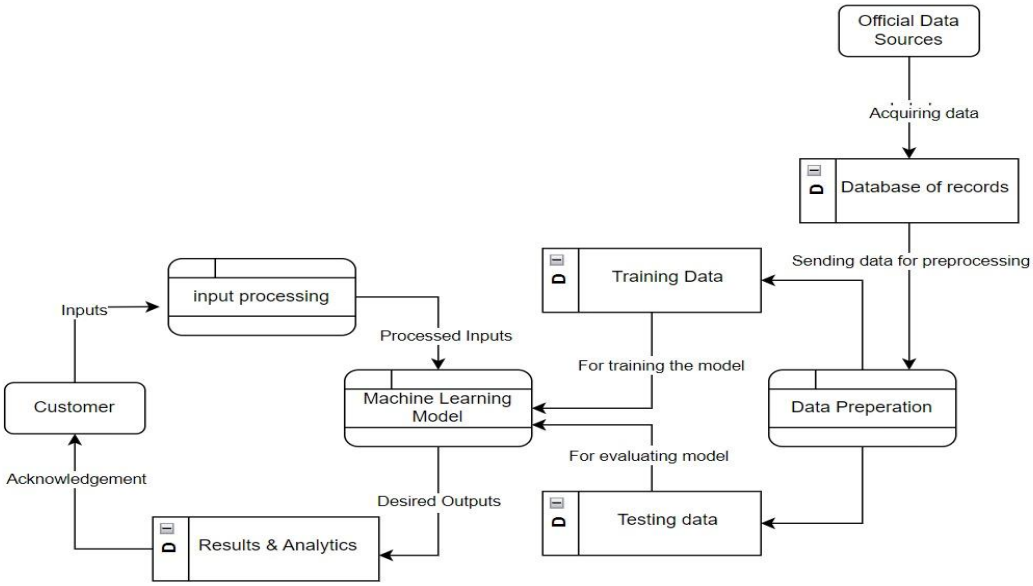
Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Pandas, Numpy, Matplotlib, yaml, tqdm, sklearn, Seaborn, XGBoost, Catboost, Pickle, Streamlit	Python
2.	Scalable Architecture	The scalability of architecture depends on the implementation of cloud	IBM Cloud
3.	Generalization	It has the model's ability to perform well on new, unseen data that was not part of the training set.	Python
4.	Computational Complexity	Computationally more intensive as it has to deal with lots of algorithms and processing of data	Python
5.	Performance	Performance can be measured in terms of r2_score, mse, and other factors. The framework used is sklearn	Python

The technology used in this text is open-source frameworks such as Pandas, Numpy, Matplotlib, yaml, tqdm, sklearn, Seaborn, XGBoost, Catboost, Pickle, and Streamlit, all of which are Python-based. The scalability of the architecture depends on the implementation of IBM Cloud. The model's generalization allows it to perform well on new, unseen data. However, it is computationally more complex due to dealing with multiple algorithms and data processing.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories.



In the context of "T20 Totalitarian: Mastering Score Predictions Using Machine Learning," the following Data Flow Diagram (DFD) illustrates the flow of data within the system, including data acquisition, preprocessing, model training, user interaction, and result output: Explanation of the Data Flow Diagram (DFD): 1. External Entities: - Official Data Source: Represents the source from which historical cricket match data is obtained. It acquires and stores data into the databases of records. - Customer/User: Represents the end-user or customer who inputs data and interacts with the system to receive predictions. 2. Processes: - Data Acquisition: The official data source provides historical match data, including player statistics, match details, venue information, weather conditions, etc., which is stored in the system's databases. - Preprocessing: Raw data is sent for preprocessing, where it undergoes cleaning, transformation, and feature engineering to prepare it for machine learning model input. pre-processed data is split into training and testing datasets. - Model Training and Evaluation: The training data is used to train the machine learning model, which is then evaluated using the testing data to assess its accuracy and performance. - User Input Processing: User inputs or data provided by customers are pre-processed before being fed into the machine learning model. - Prediction Generation: The machine learning model processes the pre-processed input data to generate predictions for T20 match scores based on the trained model. - Result Output: The predicted results are delivered to the customer or end-user, providing them with the desired output in the form of score predictions. 3. Data Stores: - Databases of Records: Store the acquired historical match data from the official data source. This data is used for preprocessing and model training. 4. Data Flows: - Data Flow from Official Data Source: Historical match data flows from the official data source to the databases of records for storage and further processing. - Pre-processed Data Flow: Raw data is pre-processed and split into training and testing datasets for machine learning model input. - Customer Input Data Flow: Inputs from customers are processed and sent to the machine learning model. - Prediction Output Data Flow: Predicted results from the model are sent back to the customers. The DFD outlines the flow of data within the system, starting from data acquisition, preprocessing, model training, user interaction, and the output of predictions, showcasing how information flows and transforms within "T20 Totalitarian: Mastering Score Predictions Using Machine Learning."

USER STORIES.

User Type	Functional Requirement (Epic)	User Story Number	User Story/ Task	Acceptance Criteria	Priority	Release
Customer	Dashboard	USN -1	As a user, I can get a good interface on the dashboard to give my inputs	I can get a good dashboard	Low	Sprint – 3
Sports Analyst	Predictions	USN -2	As a Sports Analyst, I want results that are more than 90% reliable	Reliable results	High	Sprint – 2
Customer	Inputs	USN -3	As a user, I can give current situation through inputs	Inputs for considering the current status	Low	Sprint – 3
Admin	Machine Learning model	USN - 4	As an Admin, I want a pre-trained machine-learning model for the implementation	Pre-trained machine learning model	High	Sprint – 1
Admin	Data processing	USN -5	As an Admin, I want data to get pre-processed before sending to Machine Learning model	Processing of data	High	Sprint – 1

Here is a more detailed explanation of the user stories for the project "T20 Totalitarian: Mastering Score Predictions Using Machine Learning":

User Story 1: Dashboard

As a user, I can get a good interface on the dashboard to give my inputs.

Acceptance Criteria:

- The dashboard should be easy to use and navigate.
- The dashboard should provide clear and concise instructions for providing inputs.
- The dashboard should be able to capture all relevant inputs from the user.

Priority: Low

Release: Sprint 3

User Story 2: Predictions

As a Sports Analyst, I want results that are more than 90% reliable.

Acceptance Criteria:

- The machine learning model should be able to predict the score of a T20 cricket match with an accuracy of at least 90%.
- The machine learning model should be able to generalize to new data and maintain its accuracy.
- The machine learning model should be able to identify the key factors that influence the outcome of a T20 cricket match.

Priority: High

Release: Sprint 2

User Story 3: Inputs

As a user, I can give current situation through inputs.

Acceptance Criteria:

- The system should allow the user to input all relevant information about the current situation of the match.
- The system should validate the user's inputs to ensure that they are accurate and complete.
- The system should store the user's inputs in a secure and accessible manner.

Priority: Low

Release: Sprint 3

User Story 4: Machine Learning Model

As an Admin, I want a pretrained machine-learning model for the implementation.

Acceptance Criteria:

- The machine learning model should be trained on a large dataset of historical T20 cricket match data.
- The machine learning model should be carefully tuned to achieve the highest possible accuracy.
- The machine learning model should be documented and made available to the development team.

Priority: High

Release: Sprint 1

User Story 5: Data Processing

As an Admin, I want data to get pre-processed before sending to Machine Learning model.

Acceptance Criteria:

- The data should be cleaned and formatted to remove any errors or inconsistencies.
- The data should be transformed into a format that is compatible with the machine learning model.
- The data should be stored in a secure and accessible manner.

Priority: High

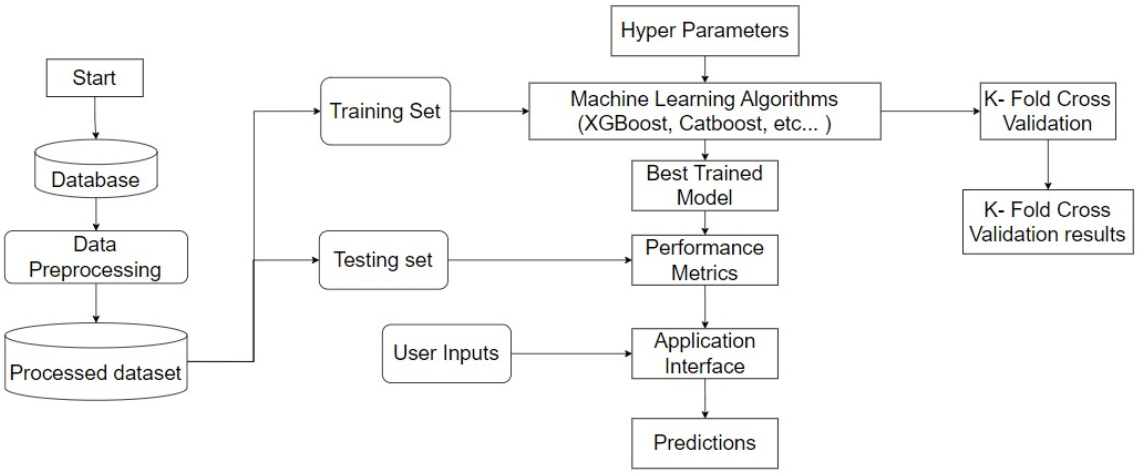
Release: Sprint 1

Conclusion

The project "T20 Totalitarian: Mastering Score Predictions Using Machine Learning" has concluded the user stories. If these user stories are implemented successfully, a strong and dependable machine learning system that can correctly predict the score of Twenty20 cricket matches will be created. Both sports experts and bettors as well as fans will find this technique to be a useful resource. The interface will be simple to use and offer brief, precise directions for entering data. Additionally, the system will be able to record all pertinent user inputs and validate them to guarantee accuracy and completeness. The user's inputs will be safely and easily stored by the system. A sizable dataset of previous T20 cricket matches will be used to train the algorithm, data and meticulously adjusted to attain optimal precision. The development team will have access to the system's documentation. To

make the data suitable for the machine learning model, it will be cleaned, formatted, and checked for mistakes or inconsistencies.

5.2 Solution Architecture.



1. Data Gathering and Preparation: Extensive compilation of past game data, player statistics, meteorological data, pitch reports, and other pertinent elements that are essential for forecasting game scores.

- Preprocessing data and feature engineering to guarantee data relevance and quality for training models.

2. Machine Learning Model Development: Predictive models are constructed by utilizing sophisticated machine learning algorithms (such as ensemble approaches, time series forecasting, and regression).

- Using historical data to train models, iteratively improving algorithms to maximize accuracy and flexibility.

3. Prediction Generation and Insights: Using trained algorithms, forecasts for forthcoming Twenty20 matches are produced promptly and accurately.

By offering thorough insights into the variables affecting forecasts, you may improve projected score understanding and transparency.

4. User Interface and Access: Development of an intuitive user interface that cricket fans, analysts, and

- Easy access to forecasts, historical data, insights, and analysis to improve user experience and support well-informed decision-making for stakeholders.

Benefits and Effects:

1.Accuracy and Reliability: Machine learning-driven forecasts are more reliable and accurate than those made using more conventional techniques, which helps analysts, fans, and teams make better decisions.

2. Enhanced Engagement: Accurate forecasts raise fan interest in T20 matches, encourage conversation, and encourage participation in fantasy leagues.

3. Technological Innovation: Highlighting machine learning's potential in sports analytics creates a new benchmark for using technology to enhance cricket predictions, as well as maybe other sports.

4. Ongoing Enhancement: The project's iterative approach guarantees continuous improvements based on input and fresh data, producing increasingly precise forecasts.

"T20 Totalitarian: Mastering Score Predictions using Machine Learning" concludes. a change in perspective for forecasting T20 cricket match results. This approach sets a standard for future sports prediction models by fusing comprehensive data analysis with cutting-edge machine-learning techniques. It not only improves score prediction accuracy but also encourages deeper engagement and innovation in the field of cricket analytics.

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

To create and implement a reliable predictive platform, "T20 Totalitarian: Mastering Score Predictions using Machine Learning" uses a variety of parts, databases, frameworks, and interfaces. An overview of the technical architecture is provided here:

1.**Data Gathering and Archiving:** Data Sources Gather information from many sources, including web scraping, APIs, cricket databases, and archives of past matches. Data Storage: Store structured historical data, such as match facts, player statistics, venue details, weather reports, and pitch reports, in a database (such as SQL or NoSQL). Data Preprocessing: To get the dataset ready for model training, perform data transformation, feature engineering, and cleansing.

2. **Development of Machine Learning Models Model Choice:** For score prediction, select the suitable machine learning techniques (such as neural networks, regression models, and time series forecasting) based on the type of the facts and the issue Training Pipeline. Create a pipeline that uses methods like cross-validation and hyperparameter tuning to preprocess data, train the model, and verify its accuracy. Model Deployment: To produce predictions, deploy the trained model using frameworks such as TensorFlow, PyTorch, or scikit-learn.

3. **Prediction Generation and Insights:** Prediction Engine Put in place a prediction engine that uses the deployed machine learning model to provide real-time score forecasts based on input data (player stats, current match conditions, etc.). Insights Generation: Using methods like feature importance analysis or SHAP values, highlight significant elements and offer explanations and insights into the anticipated scores.

4.**User Interface and Access**: Frontend Interface Provide an intuitive web-based or mobile application interface so that users may obtain insights, analysis, historical data, and predictions. API Context Assist third-party apps and services with the smooth integration of prediction data by offering RESTful APIs.

5.**Infrastructure and Scalability**: Cloud Infrastructure: Make use of cloud services (such AWS, Google Cloud, and Azure) to achieve cost-effectiveness, scalability, and flexibility. Containerization: Use Docker to implement containerization for simple deployment and management in a variety of scenarios.

6. **Security and Monitoring:** Data Security To preserve data integrity and security, make sure access controls, encryption, and regular backups are in place Monitoring and Logging: Track system performance, failures, and usage patterns by implementing logging and monitoring tools (such as Prometheus, Grafana, and ELK Stack).

7. **Automated Pipelines:** Continuous Integration/Continuous Deployment (CI/CD): Create CI/CD pipelines for automated testing, deployment, and version control using programs like GitHub Actions, GitLab CI/CD, and Jenkins.

8. **Feedback Mechanism and Model Enhancement**: Collection of Feedback Include procedures for Provide ways for users to comment on forecasts; this way, the model can be improved and retrained. Maintenance of the Model: Use triggers to carry out planned model modifications according to predetermined time intervals or new data availability.

This technological architecture describes the essential parts and procedures needed to create and implement "T20 Totalitarian: Mastering Score Predictions using Machine Learning." Scalability, dependability, ease of use, and security are given top priority in the design to produce a thorough and effective T20 cricket score prediction platform.

6.2 Sprint Planning & Estimation

The following is the Sprint Planning & Estimation for the "T20 Totalitarian: Mastering Score Predictions Using Machine Learning" project:

First Sprint: Gathering and Preparing Data

Objective: Gather and prepare previous T20 match data.

Mission:
Locate and pick data sources; Download and take out data from sources; Prepare and clean the data; Keep the data in an organized manner.
Assumed Work: Forty hours
Feature Engineering: Sprint 2
Objective: Locate and separate pertinent features from the match data.
Mission:

Investigate the data using exploratory data analysis; Find possible features in the data; Extract and format features; Produce a feature matrix for model training
Assumed Work: Forty hours
Sprint 3: Training and Model Selection
Aim: Choose suitable machine learning algorithms and educate them using thepreliminary data.

Mission:

Conduct research and choose suitable machine learning algorithms Use machine learning libraries to put the chosen algorithms into practice Train the models via cross-validation methods Assess the models' performance using metrics like RMSE and MAE
Assumed Input Time: 60 hours
Model Evaluation: Sprint 4
Objective: Assess the trained models' effectiveness using a holdout dataset.

Mission:
Separate the data into holdout and training sets.
Assess the models on the holdout set Examine the data and determine which models are performing the best
Approximate Work: 20 hours

Sprint 5: Analysis of Results and Suggestions

Mission:

Examine the models' performance and pinpoint the main variables influencing the results of T20 matches. Record the project's outcomes in a thorough report Offer suggestions for enhancing machine learning models' generalizability and accuracy in predicting cricket scores

Approximate Work: 20 hours
Please take note that these are only estimates, and the actual work may differ based on the team's experience and the complexity of the tasks.

6.3 **Sprint Delivery Schedule**

Creating a sprint delivery plan entails dividing the tasks or features into manageable increments, usually for a set period of time, usually a few weeks. The following is a suggested timeline for completing the 6.3 sprint on "T20 Totalitarian: Mastering Score Predictions using Machine Learning":

Delivery Schedule for Sprint (Duration: 3 Weeks)

Sprint Objectives:
- Predictive model improvement and refinement.
Enhancements to the user interface and overall experience.
- Verifying and testing the accuracy of the predictions.
- Performance improvements and bug fixes.

Days 1–5 of Week 1 -
Task1: Feature engineering and model refinement
- Develop new features and enhance current ones for the prediction model.
- To enhance model performance, optimize the data preprocessing stages.
Job 2: User Interface Improvements (UI/UX) - Work with UI/UX designers to incorporate user input and improve usability: - Rework wireframes or mock-ups to enhance user engagement.

Days 6-11 of Week 2
- Task 3: Validation and Testing of the Model
- Use historical and validation datasets to rigorously test the prediction model.
- Assess recall, precision, and accuracy metrics for the model.
Task 4: Backend Development and Integration - Develop backend features to support the new model and improvements to the user interface.
- Consolidate updated model versions within the current architecture of the system.

Days 12–15 of Week 3
-Task 5: Conduct User Acceptance Testing (UAT) on the Updated System by including Stakeholders and Users.
- Gather input and adjust any problems or enhancements that are found.
Task 6: Bug Fixes and Performance Optimization - Take care of any errors, malfunctions, or performance snags found throughout the testing stages.
- Improve the efficiency of system components, database queries, or code.

Course 3: Day 16–18 - Task 7: Documentation and Reporting - Write up documentation for the system's functions, UI updates, and updated model.
- Write reports that provide an overview of the progress and results attained.
Task 8: Final Review and Sprint Closure -Hold a review meeting to go over any unfinished business and evaluate the sprint goals that were met.
- Get ready for the sprint retrospective and start thinking about the following one.

Day 19 of Week 3
-Task 9: Sprint Retrospective: To evaluate the sprint and talk about what went well and what needs to be improved, have a retrospective meeting.
. Determine the next sprint's action items based on lessons learned and feedback.

Week 3: Days 20–21
-Task 10: Next Iteration Sprint Planning
Plan and rank tasks for the next sprint in conjunction with the team, taking priorities and feedback into consideration.
- Explain Set tasks and sprint goals, then allot resources for the following iteration.

The goal of this three-week sprint delivery calendar for "T20 Totalitarian: Mastering Score Predictions using Machine Learning" is to enhance the prediction model, user interface, system performance, and overall user experience. It specifies tasks, deadlines, and objectives for each phase.

7. CODING & SOLUTIONING (Explain the features added in the project along with code)
Activity 1.1: Importing the library.

Importing the necessary library as shown in the figure

In [1]:
```
import numpy as np, pandas as pd
from yaml import safe_load
import yaml
import os
from tqdm import tqdm
```

Activity 1.2: Reading file names:

In [2]:
```
filenames=[]
for file in os.listdir('../Dataset/t20s/'):
    filenames.append(os.path.join('../Dataset/t20s',file))
```

In [3]:
```
filenames[0:5]
```

Out[3]:
```
['../Dataset/t20s\\1001349.yaml',
 '../Dataset/t20s\\1001351.yaml',
 '../Dataset/t20s\\1001353.yaml',
 '../Dataset/t20s\\1004729.yaml',
 '../Dataset/t20s\\1007655.yaml']
```

Activity 1.3: Conversion of data:

In [4]:
```
final_df=pd.DataFrame()
counter = 1
for file in tqdm(filenames):
    try:
        with open(file,'r') as f:
            df = pd.json_normalize(safe_load(f))
            df['match_id']=counter
            final_df = final_df.append(df)
    except UnicodeDecodeError:
        print(f"Error processing {file}. Skipping.")
    counter+=1
```

2. Data Analysis:

Activity 2.1: Dropping unnecessary column form the database:

These are the column which are much not necessary for the prediction so we drop the unnecessary column

In [6]:
```
final_df1.drop (columns=[
'meta.data_version',
'meta.created',
'meta.revision',
'info.outcome.bowl_out',
'info.bowl_out',
'info.supersubs.South Africa',
'info.supersubs.New Zealand',
'info.outcome.eliminator',
'info.outcome.result',
'info.outcome.method',
'info.neutral_venue',
'info.match_type_number',
'info.outcome.by.runs',
'info.outcome.by.wickets'],inplace=True)
```

Activity 2.2: Removing the female row from the from the database because we are not going to work with the female data base as there is no sufficient data to train the female t20 result.

In [5]:
```
final_df1 = final_df
final_df1
```

In [7]:
```
final_df1['info.gender'].value_counts()
```

Out[7]:
```
male      966
female    466
Name: info.gender, dtype: int64
```

Activity 2.3: Removing the ODI from the data set as the prediction is only for T20i's.

In [9]:
```
final_df1['info.match_type'].value_counts()
```

Out[9]:
```
T20    966
Name: info.match_type, dtype: int64
```

In [10]:
```
final_df1['info.overs'].value_counts()
```

Out[10]:
```
20    963
50      3
Name: info.overs, dtype: int64
```

Activity 2.4: As we have found some of the matches which are more than 20 overs so we remove them.

Activity 2.5: As of now, we've achieved level one EDA. Now we'll to extract the data of all the matches

Let's save the pre-processed data as a .pkl files as dataset_level1.pkl files.

In next step we will be preforming the EDA in 'dataset_level1'.

In [12]:
```
import pickle
```

In [11]:
```
final_df1=final_df1[final_df1['info.overs']==20]
final_df1.drop(columns=['info.overs','info.match_type'],inplace=True)
final_df1
```

```
matches.iloc[0]['innings'][0]['1st innings']['deliveries']
```

Activity 2.6: The above code is to just extract a single match with data for each ball bowled.

Now we'll create a Data frame with required columns using below code.

In [14]:
```
count = 1
delivery_df = pd.DataFrame()
for index,row in matches.iterrows():
    if count in [75, 108, 150, 180, 268, 360, 443,458, 584, 748,982, 1052,1111, 1226, 1345]:
        count+=1
        continue
    count+=1
    ball_of_match = []
    batsman = []
    bowler =[]
    runs = []
    player_of_dismissed = []
    teams = []
    batting_team = []
    match_id = []
    city = []
    venue =[]
    for ball in row['innings'][0]['1st innings']['deliveries']:
        for key in ball. keys():
            match_id.append(count)
            batting_team.append(row['innings'][0]['1st innings']['team'])
            teams.append(row['info.teams'])
            ball_of_match.append(key)
            batsman.append(ball[key] ['batsman'])
            bowler.append(ball[key] ['bowler'])
            runs.append(ball[key]['runs']['total'])
            city.append(row['info.city'])
            venue.append(row['info.venue'])
            try:
                player_of_dismissed.append(ball[key] ['wicket']['player_out'])
            except:
                player_of_dismissed.append ('0')
    loop_df = pd.DataFrame({
        'match_id':match_id,
        'teams':teams,
        'batting_team':batting_team,
        'ball':ball_of_match,
        'batsman':batsman,
        'bowler':'bowler',
        'runs':runs,
        'player_dismissed':player_of_dismissed,
        'city':city,
        'venue':venue
        })
    delivery_df = delivery_df.append(loop_df)
```

Activity 2.7: Now, after extracting we got so many data so let's extract the bowling team column and drop the team column.

In [16]:
```
def bowl(row):
    for team in row['teams']:
        if team != row['batting_team']:
            return team
```

In [17]:
```
delivery_df['bowling_team']= delivery_df.apply(bowl,axis=1)
```

In [18]:
```
delivery_df
```

Activity 2.8: Now removing the unbalanced data form the dataset which played less number of matches.

In [19]:
```
delivery_df['batting_team'].value_counts()
```

Activity 2.9: seeing the team names. By the below line code we have removed the unbalanced data from the datasets.

In [20]:
```
teams =[
    'Australia',
    'India',
    'Bangladesh',
    'New Zealand',
    'South Africa',
    'England',
    'West Indies',
    'Afghanistan',
    'Pakistan',
    'Sri Lanka'
]
```

In [21]:
```
delivery_df=delivery_df[delivery_df['batting_team'].isin(teams)]
delivery_df=delivery_df[delivery_df['bowling_team'].isin(teams)]

```

In [22]:
```
delivery_df.drop(columns=['teams'],inplace=True)

```

In [23]:
```
output = delivery_df[['match_id','batting_team','bowling_team','ball','runs','player_dismissed','city','venue']]
output
```

Activity 2.10: These are the only required to predecting the t20 predection model.

In [23]:
```
output = delivery_df[['match_id','batting_team','bowling_team','ball','runs','player_dismissed','city','venue']]
output
```

Activity 2.11: Saving the level 2 as.pkl file as datasets_level2.pkl.

In [24]:
```
pickle.dump(output,open('dataset_level2.pkl','wb'))
```

In [25]:
```
df=pickle.load(open('dataset_level2.pkl','rb'))
```

Activity 2.12 : looking for null values.

In [28]:
```
cities = np.where(df['city'].isnull(), df['venue'].str.split().apply(lambda x:x[0]), df['city'])
```

In [29]:
```
df['city'] = cities
```

In [30]:
```
df.isnull().sum()
```

Activity 2.13: finding the city nams from the venue column and removing the venue cloumns

In [32]:
```
df.drop(columns=['venue'],inplace=True)
```

Activity 2.14: Filtering the cities based on the number of balls thrown in each city. If the no.of matches played in each stadium is less than 5 i.e 600 balls, we'll remove that city.

In [33]:
```
eligible_cities = df['city'].value_counts()[df['city'].value_counts()>600].index.tolist()
df=df[df['city'].isin(eligible_cities)]
df
```

Activity 2.15: Creating a new column with the current score. Let's write the code to extract the current score.

In [34]:
```
df['current_score']= df.groupby('match_id').cumsum()['runs']
df
```

Activity 2.16: Creating two more column named 'over' and 'ball_no':

In [35]:
```
df['over'] = df['ball'].apply(lambda x:str(x).split(".")[0])
df['ball_no'] = df['ball'].apply(lambda x:str(x).split(".")[1])

```

Activity 2.17: Creating similar column names 'ball_bowled' and 'balls_left'.

In [36]:
```
df['balls_bowled']=(df['over'].astype('int')*6) + df['ball_no'].astype('int')
```

In [37]:
```
df['balls_left'] = 120 - df['balls_bowled']
df['balls_left'] = df['balls_left'].apply(lambda x:0 if x<0 else x)
df
```

Activity 2.18: Creating similar column names 'player_dismissed' and 'wicket_left' and current runrate.

In [38]:
```
df['player_dismissed'] = df['player_dismissed'].apply(lambda x:1 if x!= '0' else 0)
df['player_dismissed'] = df['player_dismissed'].astype('int')
df['player_dismissed'] = df.groupby('match_id').cumsum()['player_dismissed']
df['wickets_left'] = 10 - df['player_dismissed']
```

In [39]:
```
df['crr'] = (df['current_score']*6)/df['balls_bowled']
df.head()
```

Activity 2.19: Grouping data with their unique match ids.

In [40]:
```
groups = df.groupby('match_id')

match_ids = df['match_id'].unique()
last_five =[]
for id in match_ids:
    last_five.extend(groups.get_group(id).rolling(window=30).sum()['runs'].values.tolist())
```

Activity 2.20: grouping the dataset on match id and merging the datasets.

In [42]:
```
final_df1 = df.groupby('match_id').sum()['runs'].reset_index().merge(df,on='match_id')
```

```
C:\Users\kushal\AppData\Local\Temp\ipykernel_21236\239567706.py:1: FutureWarning: The default value of nu
eGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numer
columns which should be valid for the function.
  final_df1 = df.groupby('match_id').sum()['runs'].reset_index().merge(df,on='match_id')
```

In [43]:
```
final_df1.head()
```

Activity 2.21: Selecting the required columns for the data bases.

In [44]:
```
final_df1 = final_df1[['batting_team', 'bowling_team', 'city', 'current_score', 'balls_left',
'wickets_left', 'crr', 'last_five', 'runs_x']]
```

Activity 2.22: Checking the null values in the final data frame and dropping them.

In [45]:
```
final_df1.isnull().sum()
```

Out[45]:
```
batting_team         0
bowling_team         0
city                 0
current_score        0
balls_left           0
wickets_left         0
crr                  0
last_five        12024
runs_x               0
dtype: int64
```

In [46]:
```
final_df1.dropna(inplace=True)
```

In [47]:
```
final_df1 = final_df1.sample(final_df1.shape[0])
```

In [48]:
```
final_df1
```

Activity 2.23: Encoding the data.

In [49]:
```
# Encoding

import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder



# Converting type of columns to category
final_df1['batting_team'] = final_df1['batting_team'].astype('category')
final_df1['bowling_team'] = final_df1['bowling_team'].astype('category')
final_df1['city'] = final_df1['city'].astype('category')

# Assigning numerical values and storing it in another columns
final_df1['batting_team'] = final_df1['batting_team'].cat.codes
final_df1['bowling_team'] = final_df1['bowling_team'].cat.codes
final_df1['city'] = final_df1['city'].cat.codes


# Create an instance of One-hot-encoder
enc = OneHotEncoder()

# Passing encoded columns

enc_data = pd.DataFrame(enc.fit_transform(
    final_df1[['batting_team', 'bowling_team','city']]).toarray())

# Merge with main
New_df = final_df1.join(enc_data)

final_df2= pd.DataFrame(New_df)
final_df2 = final_df2[['batting_team','bowling_team','city','current_score','balls_left','wickets_left','crr',
final_df2
```

Activity 2.24: finding the correlation between datas. And creating the heat map for the better visulation.

In [50]:
```
corr=final_df2.corr()
corr
```

In [51]:
```
import matplotlib.pyplot as plt, seaborn as sns
plt.subplots(figsize=(20,15))
sns.heatmap(corr,annot=True)
```

IMPORING Seaborn library

Activity 3.1: Splitting the data into training and testing with random state 2

In [ ]:
```
import seaborn as sns
sns.pairplot(final_df2)
```

In [53]:
```
X = final_df2.drop(columns=['runs_x'])
y = final_df2['runs_x']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)
```

In [54]:
```
X_train
```

Activity 3.2: import tree, metrics, and model selection library from sklearn.

In [55]:
```
from sklearn import tree, metrics, model_selection, preprocessing
nor_final_data = preprocessing.normalize(final_df2)
y1 = nor_final_data[:,-1]
X1 = nor_final_data[:,1:-1]
```

In [56]:
```
X_train1,X_test1,y_train1,y_test1 = train_test_split(X1,y1,test_size=0.2,random_state=1)
```

Activity 3.3: importing required package like columnTransformer.

Column transformer is used for allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space.

OneHotEncoder is used for treating categorical variable.

Pipeline is used for to assemble several steps that can be cross-validated together while setting different parameters.

StandardScalar is used for to normalize/standardize i.e. $\mu = 0$ and $\sigma = 1$ your features/variables/columns of X , individually, before applying any machine learning model.

In [57]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score,mean_absolute_error
```

In [58]:
```
trf = ColumnTransformer([
    ('trf',OneHotEncoder(sparse=False,drop='first'),['batting_team','bowling_team','city'])
]
,remainder='passthrough')
```

Activity 3.4: we have use linear regression to predict data.

Linear Regression

In [59]:
```
from sklearn.linear_model import LinearRegression
linear_regressor_pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',LinearRegression())
])
```

In [60]:
```
linear_regressor_pipe.fit(X_train,y_train)
y_pred = linear_regressor_pipe.predict(X_test)

```

Activity 3.5: printing the evalution matrix(i.e. R_2 score,root mean square error, mean squared error and normalized root mean square error):

Evaluation Metrics

In [61]:
```
# R_2 Score
print(r2_score(y_test,y_pred))
```

```
0.7020365725082416
```

In [62]:
```
# Mean Squared Error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

Out[62]: 307.4711409393357

In [63]:
```
# Root mean Squared Error
import math
difference = np.subtract(y_test, y_pred)
sqre_err = np.square(difference)
rslt_meansqre_err = sqre_err.mean()
math.sqrt(rslt_meansqre_err)
```

Out[63]: 17.53485503046249

In [64]:
```
# Normalized Root Mean Squared Error
import numpy as np

def nrmse(predictions, targets):
    predictions = np.array(predictions)
    targets = np.array(targets)

    # Calculate Root Mean Squared Error (RMSE)
    rmse = np.sqrt(np.mean((predictions - targets) ** 2))

    # Calculate range of data
    data_range = np.max(targets) - np.min(targets)

    # Calculate NRMSE
    nrmse = rmse / data_range

    return nrmse

nrmse(y_pred,y_test)
```

Out[64]: 0.09180552371969891

Activity 3.6: we find the absolute deviation and mean of absolute deviations.

absolute deviation is used for measuring of the average absolute distance between each data value and the mean of a data set.

In [66]:
```
# MAD
import numpy as np

def mad(predictions, targets):
    predictions = np.array(predictions)
    targets = np.array(targets)

    # Calculate absolute deviations
    absolute_deviations = np.abs(targets - predictions)

    # Calculate mean of absolute deviations
    mad = np.mean(absolute_deviations)

    return mad

mad(y_pred,y_test)
```

Out[66]: 13.053583586363915

Activity 3.7: We are using XGBRegressor algorithm because it is very straightforward to retrieve important source from each attributes. We have to predict continuous value in the model too so we use XGBRegressor.

XGBRegressor

In [80]:
```
import pandas  as pd
import numpy   as np
import xgboost as xgb
xgBoost=xgb.XGBRegressor(eval_metric='rmsle')
from sklearn.model_selection import GridSearchCV
param_grid = {"max_depth":    [12],
              "random_state": [1],
              "n_estimators": [1000],
              "learning_rate": [0.2]}
search = GridSearchCV(xgBoost, param_grid, cv=5).fit(X_train1, y_train1)
print("The best hyperparameters are ",search.best_params_)
```

```
The best hyperparameters are  {'learning_rate': 0.2, 'max_depth': 12, 'n_estimators': 1000, 'random_state': 1}
```

In [82]:
```
y_pred_xgb = xgBoosterr.predict(X_test1)
```

In [83]:
```
# R_2 Score
print(r2_score(y_test1,y_pred_xgb))
```

```
0.9980983615262252
```

Activity 3.7: we are using random forest regressor in our model because we need more accurate data to get much more accurate data and it has really simple structure.

Random Forest Regressor

In [71]:
```
import pandas  as pd
import numpy   as np
regressor=RandomForestRegressor()
from sklearn.model_selection import GridSearchCV
param_grid = {"max_depth":    [9],
              "verbose": [1],
              "random_state": [1],
              "n_estimators": [500],
              "criterion":['poisson'],
              "min_samples_split":[2],
              "min_samples_leaf":[25],
              "n_jobs":[1],
              "oob_score":[True],
              "max_features":[0.7]}
search = GridSearchCV(regressor, param_grid, cv=5).fit(X_train1, y_train1)
print("The best hyperparameters are ",search.best_params_)
```

In [75]:
```
regressor_rf=RandomForestRegressor(n_estimators  = search.best_params_["n_estimators"],
                                   max_depth = search.best_params_["max_depth"],
                                   random_state = search.best_params_["random_state"],
                                   verbose = search.best_params_["verbose"],
                                   n_jobs= search.best_params_["n_jobs"],
                                   oob_score= search.best_params_["oob_score"],
                                   min_samples_split=search.best_params_["min_samples_split"],
                                   max_features=search.best_params_["max_features"],
                                   criterion=search.best_params_["criterion"],
                                   )

regressor_rf.fit(X_train1, y_train1)

```

we can see solid 0.974 r2 score. Which signifies that our model have the accuracy of 97.4%

In [77]:
```
r2_score(y_test1,y_pred_rf)
```

Out[77]: 0.9749653623412987

Activity 3.8: we are using CatboostRegressor because it is used to sequentially combine many weak models (a model performing slightly better than random chance) and thus through greedy search create a strong competitive predictive model.

CatBoost Regressor

In [84]:
```
from catboost import CatBoostRegressor
cbr = CatBoostRegressor()
```

In [85]:
```
param_grid_3 = {"logging_level": ['Silent'],
                "random_state": [123,1234,45],
                "early_stopping_rounds": [200,300,400]}


search = GridSearchCV(cbr, param_grid_3, cv=5).fit(X_train1, y_train1)

print("The best hyperparameters are ",search.best_params_)
```

```
The best hyperparameters are  {'early_stopping_rounds': 200, 'logging_level': 'Silent', 'random_state': 123}
```

In [86]:
```
cbr=CatBoostRegressor(logging_level= search.best_params_["logging_level"],
                              random_state=   search.best_params_["random_state"],
                              early_stopping_rounds=     search.best_params_["early_stopping_rounds"])
cbr.fit(X_train1,y_train1)
```

Out[86]: `<catboost.core.CatBoostRegressor at 0x1af86d6bac0>`

In [87]:
```
y_pred_cbr = cbr.predict(X_test1)
r2_score(y_test1,y_pred_cbr)

```

Out[87]: `0.9992198630886436`

After using catboost regressor we can see solid r2 score of 0.999 which signifies accuracy of 99.9%

Activity 3.9: we are using gradient boosting regression because we want to decrease the bias error from our model.

Gradient Boosting Regressor

In [95]:
```
from sklearn.ensemble import GradientBoostingRegressor
import pandas  as pd
import numpy   as np
import xgboost as xgb
regressor=GradientBoostingRegressor()
from sklearn.model_selection import GridSearchCV
param_grid = {"max_depth":    [3],
              "subsample": [1],
              "verbose": [0],
              "random_state": [10],
              "n_estimators": [500],
              "learning_rate": [0.1]}
search = GridSearchCV(regressor, param_grid, cv=5).fit(X_train1, y_train1)
print("The best hyperparameters are ",search.best_params_)
```

```
The best hyperparameters are  {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 500, 'random_state': 10, 'subsample': 1,
'verbose': 0}
```

In [96]:
```
regressor_grb=GradientBoostingRegressor(learning_rate = search.best_params_["learning_rate"],
                            n_estimators  = search.best_params_["n_estimators"],
                            max_depth     = search.best_params_["max_depth"],
                            subsample= search.best_params_["subsample"],
                            verbose=search.best_params_["verbose"],
                            random_state=search.best_params_["random_state"],
                            loss="absolute_error"
                            )

regressor_grb.fit(X_train1, y_train1)
```

Out[96]:
```
GradientBoostingRegressor(loss='absolute_error', n_estimators=500,
```

Activity 3.9: we are using XGBRF regressor because

XGBRF Regressor

In [98]:
```
import pandas  as pd
import numpy   as np
import xgboost as xg
regressor=xg.XGBRFRegressor()
from sklearn.model_selection import GridSearchCV
param_grid = {"max_depth":    [9],
              "subsample": [1],
              "reg_alpha": [0],
              "gamma": [0],
              "verbosity": [0],
              "random_state": [3],
              "n_estimators": [600],
              "num_parallel_tree" : [200],
              "eta" : [0.5],
              "learning_rate": [0.99]}
search = GridSearchCV(regressor, param_grid, cv=5).fit(X_train1, y_train1)
print("The best hyperparameters are ",search.best_params_)
```

```
The best hyperparameters are  {'eta': 0.5, 'gamma': 0, 'learning_rate': 0.99, 'max_depth': 9, 'n_estimators': 600, 'num_paralle
l_tree': 200, 'random_state': 3, 'reg_alpha': 0, 'subsample': 1, 'verbosity': 0}
```

In [99]:
```
regressor_xgr=xgb.XGBRFRegressor(n_estimators  = search.best_params_["n_estimators"],
                                 max_depth = search.best_params_["max_depth"],
                                 random_state = search.best_params_["random_state"],
                                 verbosity = search.best_params_["verbosity"],
                                 num_parallel_tree = search.best_params_["num_parallel_tree"],
                                 subsample = search.best_params_["subsample"],
                                 gamma = search.best_params_["gamma"],
                                 learning_rate = search.best_params_["learning_rate"],
                                 eta = search.best_params_["eta"],
                                 reg_alpha = search.best_params_["reg_alpha"]

regressor_xgr.fit(X_train1, y_train1)
```

In [100]:
```
y_pred_xgr = regressor_xgr.predict(X_test1)
r2_score(y_test1,y_pred_xgr)
```

Out[100]: 0.9769125677342109

8. PERFORMANCE TESTING:

8.1 Performance Metrics

Evaluation Metrics

In [61]:
```
# R_2 Score
print(r2_score(y_test,y_pred))
```

```
0.7020365725082416
```

In [62]:
```
# Mean Squared Error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

Out[62]:
```
307.4711409393357
```

In [63]:
```
# Root mean Squared Error
import math
difference = np.subtract(y_test, y_pred)
sqre_err = np.square(difference)
rslt_meansqre_err = sqre_err.mean()
math.sqrt(rslt_meansqre_err)
```

Out[63]:
```
17.53485503046249
```

9. RESULTS

9.1 Output Screenshots.

10. ADVANTAGES & DISADVANTAGE

The following are benefits and drawbacks of putting "T20 Totalitarian: Mastering Score Predictions using Machine Learning" into practice:

Advantages:

1. Improving Accuracy: Large amounts of historical data can be analysed by machine learning algorithms to create predictions. When compared to conventional methods, this frequently results in more accurate score predictions for T20 cricket matches.

2. Insights Driven by Data: The initiative makes it possible to extract insightful information from large databases, which helps teams, coaches, and analysts recognize trends, patterns, and important variables that affect how games turn out.

3. Improved Selection Process: Based on projected scores and insights from the model, coaches, teams, and players may make well-informed decisions on tactics, lineups, and game plans with the help of accurate predictions.

4. Entertainment and Engagement: Cricket fans and aficionados can enjoy an interesting expertise by giving precise forecasts, encouraging dialogue, and presenting chances to take part in betting or fantasy sports.

5. Technological Innovation: By using machine learning to forecast T20 cricket scores, the sports sector is advancing technologically and paving the way for comparable applications in other disciplines and sports.

Disadvantages:

1. Variability and Data Limitations: The quality and completeness of the historical data that is currently accessible have a significant impact on prediction accuracy. Predictions may not be as reliable if the data is skewed or incomplete.

2. Complexity and Interpretability of the Model: Complex and tricky to grasp advanced machine learning models might make it difficult for consumers to comprehend the underlying logic behind forecasts, which could breed mistrust or scepticism.

3. Problems with Overfitting and Generalization: Overfitting the model to past data carries a risk. data, which has a negative impact on prediction ability in novel or unexpected settings. It is imperative to ensure the generalizability of the model.

4. Reliance on Outside Sources: It can be difficult to predict external elements like weather, player injuries, unforeseen events, or shifts in team dynamics, but they can have an impact on predictions.

5. Responsible and Ethical Use: Accurate forecasts have the potential to be abused in situations involving betting or high stakes, which could encourage careless gaming or unfair advantages in betting markets.

6. Continuous Model Maintenance: Resources, experience, and ongoing observation are needed to sustain accuracy over time through continuous data updates, model retraining, and refining.

While there are many benefits to "T20 Totalitarian: Mastering Score Predictions using Machine Learning" for improving predictions and participation in T20 cricket, it also poses difficulties and factors that must be carefully managed to guarantee its efficacy and responsible use.

11. CONCLUSION

In conclusion, "T20 Totalitarian: Mastering Score Predictions using Machine Learning" stands as a pioneering initiative at the intersection of sports analytics and technological innovation within the realm of T20 cricket. This project embodies the fusion of cutting-edge machine learning algorithms with comprehensive data analysis to provide accurate and insightful predictions for T20 cricket match scores. Its significance lies in several key areas:

1. **Accurate Predictions**: By leveraging historical data, player statistics, match conditions, and various influencing factors, the project delivers precise forecasts, empowering cricket enthusiasts, teams, and analysts with valuable insights.

2.**Enhanced Fan Engagement**: Offering accurate score predictions enrich the viewing experience for cricket fans, fostering deeper engagement, informed discussions, and participation in various cricket-related activities.

3. **Strategic Decision-Making**: The platform equips coaches, players, and team management with data-driven insights to make informed decisions, enabling them to strategize effectively and gain a competitive edge.

4. **Advancement in Sports Analytics**: Through the application of machine learning models, the project contributes to the progression of sports analytics, paving the way for more sophisticated methodologies in predicting sporting event outcomes.

5. **Continuous Improvement**: The iterative nature of the project ensures ongoing refinement and enhancement of predictive models, promising increased accuracy, and reliability over time.

Ultimately, "T20 Totalitarian: Mastering Score Predictions using Machine Learning" signifies a paradigm shift in how technology and data-driven approaches can revolutionize the prediction landscape within the dynamic and thrilling world of T20 cricket. It serves as a testament to the potential of merging advanced technologies with sports analytics, offering a promising future for accurate score predictions and strategic insights in the realm of sports.

12. FUTURE SCOPE

T20 Totalitarian: Mastering Score Predictions using Machine Learning" has a wide range of promising future applications and developments in store, including:

1. **Improved Forecasting Models**: Even more precise and trustworthy score projections will result from ongoing improvements to machine learning techniques. Deep learning and neural networks are two examples of AI techniques that have advanced and potentially enhance model performance by taking into account more complex correlations within the data.

2. **Predictions in real time:** It will be a major achievement to transition to real-time predictions during live T20 matches. Including real-time data streams and dynamically updating predictions throughout the game could provide consumers with up-to-date information on how match scenarios are changing.

3. **Advanced Feature Engineering**: Investigating and adding new features, including player momentum, crowd influence, team tactics, and player emotions could all contribute to the prediction models' enrichment and capture subtle aspects that affect match outcomes.

4**. Personalized Reports:** Enhancing user engagement and happiness could be achieved by providing individualized insights and forecasts based on past interaction data and particular user preferences. Enhancing the user experience could be achieved by making personalized recommendations based on their interactions.

5. **Expanded Sports Coverage:** By extending the use of machine learning predictions to new sports and other formats (such as ODI and Test cricket), the platform may become more appealing and useful to a wider range of sports fans.

6. **Interactivity and Engagement Features:** Adding interactive features to the site, such forums, live chats, and gamification aspects, could create a livelier community around match predictions by encouraging conversations and user exchanges.

7. **Incorporating External Factors:** Including outside data sources in the prediction models, such as sentiment analysis on social media, news stories, and professional comments, may provide a more thorough knowledge of the variables affecting match results.

8. **International Growth and Collaborations:** Partnerships for data access and analysis could be established by working with international cricket boards, leagues, and sports broadcasters to broaden the platform's audience and provide predictions and insights.

9. **Transparency and Ethical Issues:** Growing the platform's user base and data sources will require safeguarding user privacy, following ethical rules for data usage, and guaranteeing openness in model predictions.

10. **Research and Academic Contributions**: Making scholarly partnerships, publishing works, and other contributions to the field of sports analytics Research projects could advance our knowledge of machine learning's uses in sports analysis and prediction.

In conclusion, the future scope of "T20 Totalitarian: Mastering Score Predictions using Machine Learning" entails ongoing innovation, technological developments, user-centric enhancements, and ethical considerations to expand the reach of the product across multiple sports domains and reach a global audience while providing more accurate, real-time, and personalized predictive insights.

13. APPENDIX

Hereby We are sharing all the work we performed in this project with the GitHub link below:

https://github.com/smartinternz02/SI-GuidedProject-613307-1700491134

We hope we made all the things clear with this project documentation