

T20 Totalitarian: Mastering Score Predictions

1. INTRODUCTION

1.1 Project Overview

The "T20 Totalitarian: Mastering Score Predictions" project is a comprehensive attempt focused on the development of a predictive model for estimating the total score of a cricket team batting first in a T20 match. This project combines the realms of sports analytics and machine learning to provide a practical solution for cricket enthusiasts, analysts, and team strategists. The core objective of the project is to create a robust and accurate prediction system that considers various match-related parameters, including the identity of the batting and bowling teams, the city where the match is being played, the current score, the number of overs completed, the wickets fallen, and the runs scored in the last 5 overs. By leveraging historical match data and employing machine learning techniques, the model aims to make precise predictions regarding the batting team's final score. This project not only showcases the power of data science in the realm of sports but also highlights the significance of accurate predictions in enhancing team strategies and fostering a deeper understanding of the dynamics of T20 cricket matches. The resulting model is intended to be a valuable tool for cricket enthusiasts, team coaches, and analysts seeking to gain insights and make informed decisions in the fast-paced and highly competitive world of T20 cricket.

1.2 Purpose

The purpose of this project is to harness the capabilities of data science and machine learning to address a significant challenge in the realm of cricket, particularly in the fast-paced T20 format. The primary aim is to develop a reliable and data-driven predictive model that can accurately forecast the total score of a team batting first in a T20 match. This project serves multiple essential purposes. First and foremost, it facilitates a deeper understanding of the game by leveraging historical data and match-related factors, providing a comprehensive overview of how various elements impact the final score. Second, it offers practical utility to cricket teams, coaches, and strategists by enabling them to make more informed decisions

during matches, such as optimizing their batting strategies, setting achievable targets, and adapting to different match conditions. Third, it showcases the transformative potential of data analytics in the world of sports, underlining how technology can be harnessed to enhance performance and gain a competitive edge. Overall, the project's purpose is to bridge the gap between data science and sports, making the sport more data-informed and allowing stakeholders to embrace a data-driven approach in their decision-making processes.

2. LITERATURE SURVEY

2.1 Existing problem

The existing problem that this project seeks to address is the inherent unpredictability of T20 cricket matches, especially regarding the total score a team can achieve while batting first. In the T20 format, where matches are shorter and the pace of the game is frenetic, accurately predicting a team's final score is a challenging task. Traditional methods of score estimation often rely on heuristics and intuition rather than data-driven insights, leading to inconsistent and unreliable predictions. As a result, cricket teams and strategists face a significant challenge in setting realistic targets and adapting to various match conditions, which can vary widely depending on factors like the batting and bowling teams, the location of the match, and the state of the game. The lack of a precise predictive model has been a persistent issue in T20 cricket, and this project aims to rectify that by leveraging historical data and machine learning techniques to create a model that can provide more accurate score predictions, ultimately enhancing decision-making and strategies in the sport.

2.2 References

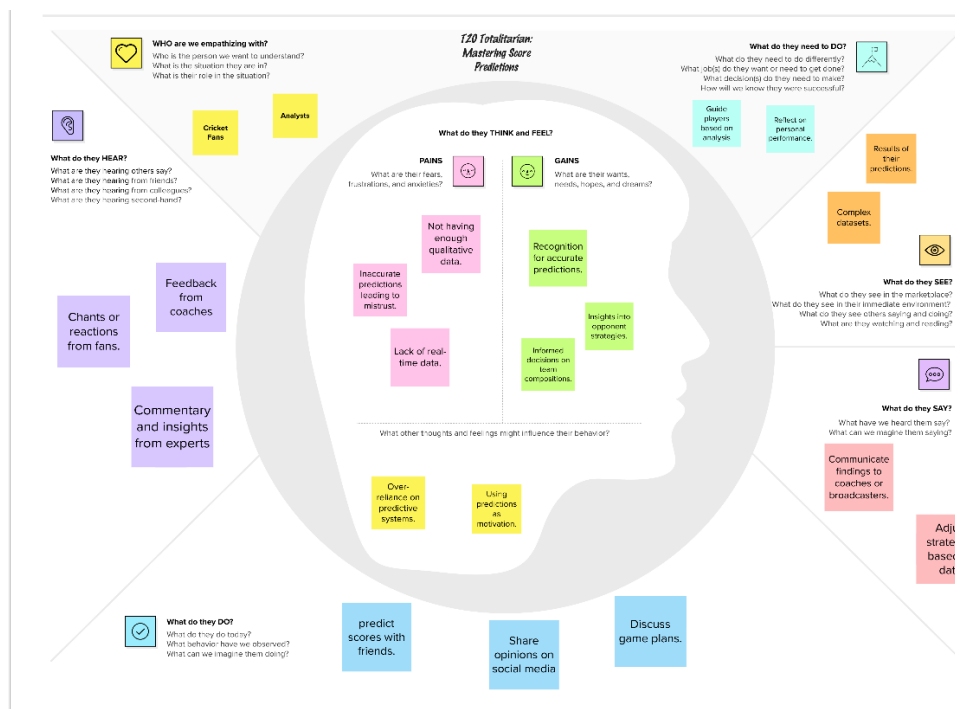
- <https://www.kaggle.com/datasets/veeralakrishna/cricsheet-a-retrosheet-for-cricket?select=t20s>
- <https://drive.google.com/file/d/1m-mGE2nDbxwmbVtEfSErD7NgUX2hkS7j/view>
- <https://medium.com/mllearning-ai/t20-cricket-score-predictor-end-to-end-ml-project-fbb3ec67dbb1>

2.3 Problem Statement Definition

The problem statement for this project is to develop a robust and accurate predictive model that can forecast the total score of a cricket team batting first in a T20 match. This problem statement entails addressing the inherent challenge of unpredictability in T20 cricket matches, where the rapid pace of the game and a multitude of variables make it difficult to estimate a team's final score. The goal is to create a data-driven solution that takes into account key match-related parameters, including the identity of the batting and bowling teams, the city where the match is being played, the current score, the number of overs completed, the wickets fallen, and the runs scored in the last 5 overs. The model's predictions should be accurate, reliable, and capable of providing insights into the dynamics of T20 matches. By doing so, this project seeks to enable cricket enthusiasts, teams, and strategists to make more informed decisions, such as setting realistic targets, adapting to different match conditions, and optimizing their gameplay. The problem statement aims to bridge the gap between traditional cricket strategies and data-driven insights, ultimately enhancing the competitiveness and strategic acumen in the fast-paced world of T20 cricket.

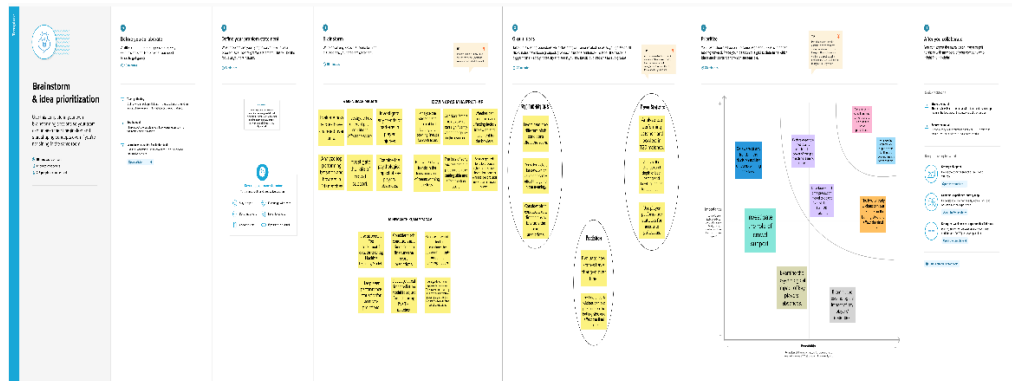
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



<https://app.mural.co/t/smdnumanmadani9809/m/smdnumanmadani9809/1697527404985/4906b910518400c777eb806e0baf091b13b11c83?sender=u00242924bf2d12a8c71d2444>

3.2 Ideation & Brainstorming



<https://app.mural.co/t/smdnumanmadani9809/m/smdnumanmadani9809/1697544675684/baa8e1fe055776f326dda2c53285dd86840c6dbd?sender=u00242924bf2d12a8c71d2444>

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

The functional requirements of this project encompass a range of functionalities and capabilities that the predictive model and associated system must possess to meet its objectives. Firstly, the system needs to acquire and preprocess historical T20 cricket match data, including information about the batting and bowling teams, the match location, the current score, overs completed, wickets fallen, and runs scored in the last 5 overs. It should efficiently handle missing or inconsistent data. Next, the core functionality involves the development of a machine learning model capable of predicting the total score of the batting team based on the input parameters. This model should encompass data preprocessing steps, such as one-hot encoding for categorical variables like teams and cities, and it should be trained on a comprehensive and representative dataset. The system should also facilitate user interaction through a user-friendly interface, such as a web application. Users should be able to input relevant match details, including team names, location, current score, overs played, wickets lost, and runs scored in the last 5 overs. The system should then provide a predicted total score in real time. Additionally, the project must employ proper data validation and error handling to ensure that user inputs are within

valid ranges and conform to expected data types. It should be able to gracefully handle unexpected or erroneous inputs. Furthermore, the model should be continually updated and refined as new match data becomes available, ensuring that it remains accurate and adaptable to changing cricket dynamics. Regular model evaluation and retraining are essential to maintain its predictive power. Finally, the system should provide clear and easily interpretable predictions, possibly including additional information or insights, to assist users in their decision-making processes. It should also be able to handle concurrent user requests and provide timely responses.

4.2 Non-Functional requirements

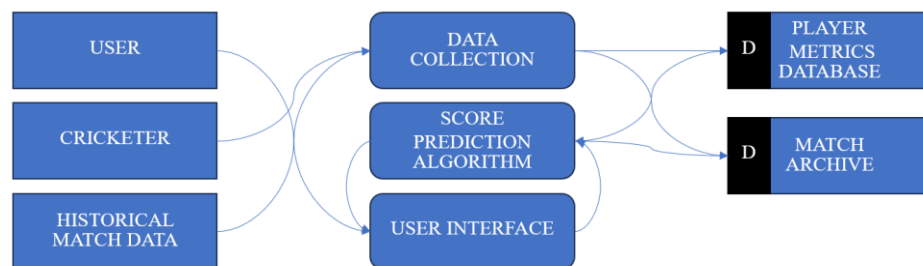
This project has several non-functional requirements that are crucial for its successful implementation and operation. Firstly, in terms of performance, the system must exhibit efficient and responsive behavior. It should provide rapid score predictions, ensuring minimal latency in response, even when handling multiple user requests simultaneously. Additionally, the system should be scalable, capable of handling an increasing volume of historical data and user inputs without a significant degradation in performance. Reliability and accuracy are paramount non-functional requirements. The model's predictions should be highly reliable and accurate, as they will influence critical decision-making in cricket matches. The system should be robust, able to handle unexpected data patterns and outliers without compromising its performance. Regular model evaluation and updating are necessary to maintain and enhance prediction accuracy over time. Security is another critical non-functional aspect. The system should ensure the privacy and security of sensitive data, both in terms of historical match data and user inputs. Adequate data encryption and access controls should be in place to protect against unauthorized access or data breaches. User-friendliness and ease of use are non-functional requirements, ensuring that the system interface is intuitive and accessible to a wide range of users, including cricket teams, coaches, and analysts. It should offer clear and informative feedback, enhancing user understanding of the predictions and insights provided. Scalability is also vital, as the system must be capable of handling growing data volumes as more T20 cricket matches occur. This requirement necessitates a well-designed and efficient data storage and retrieval system. Additionally, the system should be

adaptable and maintainable. It should be easy to update the model with new data and incorporate improvements in the prediction algorithm. This adaptability is essential in keeping the system relevant and reliable over the long term.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

Data Flow Diagram



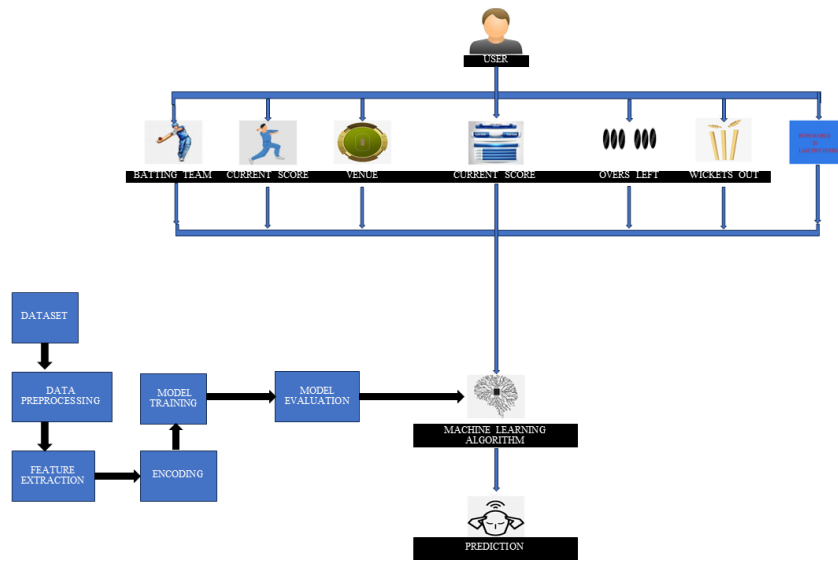
User Stories

- Gather past T20 match data.
- Clean and preprocess data.
- Define data structure and storage (e.g., database).
- Choose appropriate machine learning algorithm.
- Train initial prediction model.
- Evaluate model performance.
- Design a user-friendly interface.
- Implement interface with model integration.
- Conduct end-to-end system tests.
- Deploy to production environment.

5.2 Solution Architecture

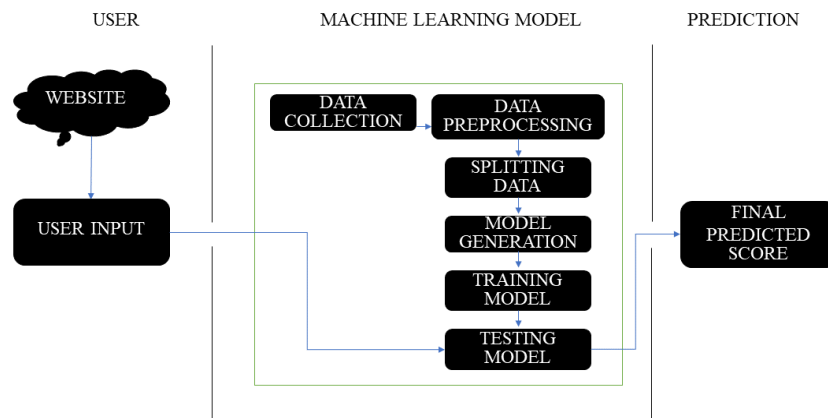
The solution architecture of this project is a structured pipeline that encompasses data collection, preprocessing, feature engineering, model selection, and evaluation. It begins with gathering the T20 cricket data and cleaning it meticulously. The relevant features are encoded to ensure the compatibility with Machine Learning models. Then different models are compared based on their efficiency and accuracy. Evaluation metrics like R-squared and Mean Absolute Error are employed to assess the model's performance. This architecture is modular, adaptable, and designed to efficiently

predict T20 cricket team scores while maintaining data quality and scalability.



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (planned)	Sprint Goal
1	20	8 days	Oct 15, 2023	Oct 23, 2023	Data Collection
2	20	8 days	Oct 24, 2023	Oct 31, 2023	Model Development
3	20	8 days	Nov 1, 2023	Nov 8, 2023	Final Testing and Deployment

6.3 Sprint Delivery Schedule

Sprint	Sprint Release Date
1	Oct 24, 2023
2	Nov 1, 2023
3	Nov 9, 2023

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Data Acquisition and Preprocessing

One of the foundational features of the project is acquiring historical T20 cricket match data and preprocessing it for use in the predictive model. This involves loading, cleaning, and structuring the data.

Handling missing values in the 'cities' column.

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

Removing irrelevant columns and filtering data by eligible teams

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


7.2 Calculation of Current Score and Other Metrics

The project calculates metrics like the current score, overs, balls bowled, wickets left, and the current run rate (crr) based on the match data.

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

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

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

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

```
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']
```

```
df['crr'] = (df['current_score']*6) / df['balls_bowled']
```

8. PERFORMANCE TESTING

8.1 Performance Metrics

In this project, the performance of the predictive model is a critical aspect, and several performance metrics are employed to assess the accuracy and reliability of the score predictions. These metrics help evaluate how well the model is performing and whether it meets the project's objectives. The key performance metrics used in the project are Mean Absolute Error (MAE), R-squared (R2) score, Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

Mean Absolute Error (MAE): MAE is used to measure the average absolute difference between the predicted scores and the actual scores in T20 cricket matches. It provides a straightforward way to assess the accuracy of the model's predictions. In the project code, MAE is calculated using the `mean_absolute_error` function from the Scikit-Learn library after the model makes predictions and is an essential metric for understanding the average prediction error.

R-squared (R²) Score: R² score, also known as the coefficient of determination, is used to evaluate the goodness of fit of the model. It quantifies the proportion of the variance in the actual scores that is explained by the model. In the project, the R² score is computed using the `r2_score` function from Scikit-Learn. A higher R² score indicates a better fit of the model to the data and stronger predictive power.

Mean Squared Error (MSE): Mean Squared Error is an essential performance metric that calculates the average of the squared differences between the predicted scores and the actual scores in T20 cricket matches. MSE provides a more comprehensive view of the prediction errors, giving higher weight to larger errors. It is a valuable metric for understanding the precision and reliability of the model. In the project code, MSE is calculated by using `'mean_squared_error'` from `'sklearn.metrics'`.

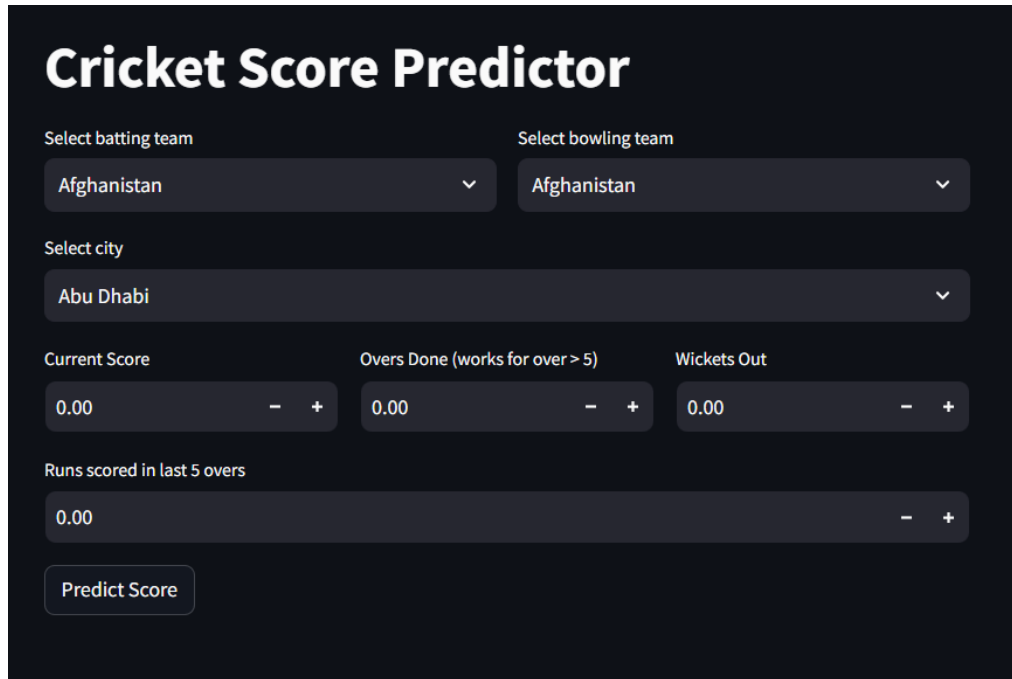
Root Mean Squared Error (RMSE): RMSE is another commonly used metric that calculates the square root of the mean of the squared differences between predicted and actual scores. It penalizes larger prediction errors more heavily and is a useful measure for understanding the model's precision. In the project code, RMSE is calculated as the root of MSE by using the `sqrt()` of Numpy library on the MSE and provides additional insights into prediction accuracy.

These performance metrics collectively allow the project team and stakeholders to assess the effectiveness of the predictive model in providing accurate T20 cricket score predictions. By regularly monitoring and evaluating these metrics, the project can identify areas for improvement, refine the model, and ensure that it continues to meet its objectives of providing reliable and valuable score predictions for cricket enthusiasts, teams, and analysts.

9. RESULTS

9.1 Output Screenshots

Website before entering the data:



The screenshot shows the 'Cricket Score Predictor' interface with default values. The batting team is 'Afghanistan', the bowling team is 'Afghanistan', and the city is 'Abu Dhabi'. All numerical inputs (Current Score, Overs Done, Wickets Out, and Runs scored in last 5 overs) are set to 0.00. A 'Predict Score' button is visible at the bottom.

Cricket Score Predictor

Select batting team: Afghanistan

Select bowling team: Afghanistan

Select city: Abu Dhabi

Current Score: 0.00

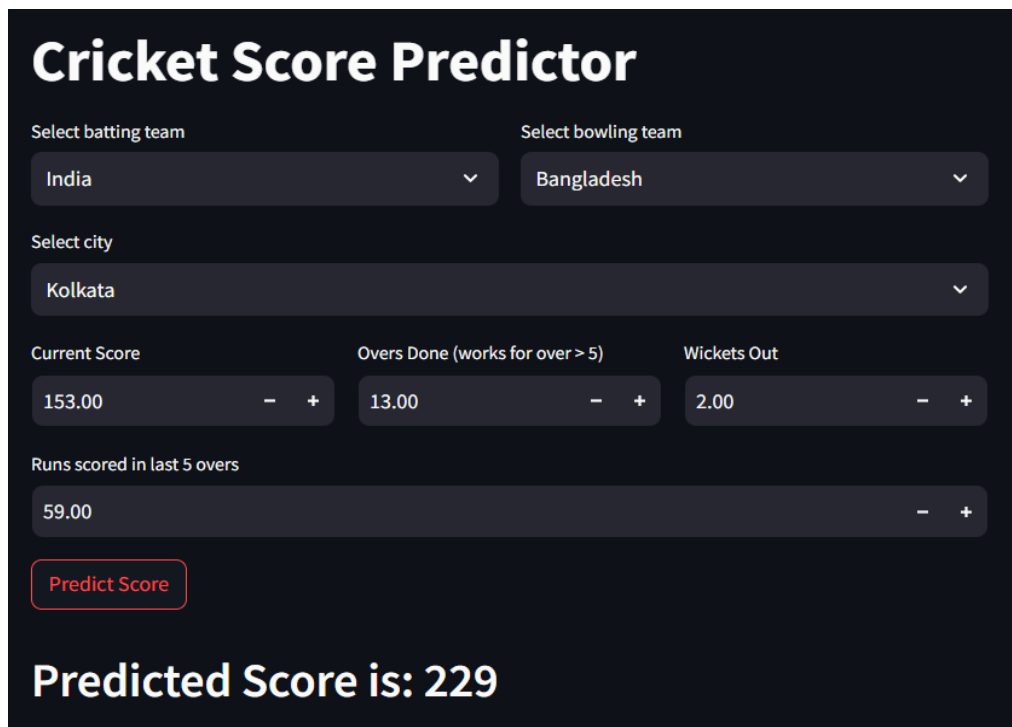
Overs Done (works for over > 5): 0.00

Wickets Out: 0.00

Runs scored in last 5 overs: 0.00

Predict Score

Website after entering the data:



The screenshot shows the same 'Cricket Score Predictor' interface but with updated data. The batting team is 'India', the bowling team is 'Bangladesh', and the city is 'Kolkata'. The numerical inputs are: Current Score 153.00, Overs Done 13.00, Wickets Out 2.00, and Runs scored in last 5 overs 59.00. The 'Predict Score' button is highlighted with a red border. Below the form, the predicted score is displayed as 'Predicted Score is: 229'.

Cricket Score Predictor

Select batting team: India

Select bowling team: Bangladesh

Select city: Kolkata

Current Score: 153.00

Overs Done (works for over > 5): 13.00

Wickets Out: 2.00

Runs scored in last 5 overs: 59.00

Predict Score

Predicted Score is: 229

10. ADVANTAGES & DISADVANTAGES

Advantages:

The project offers several notable advantages:

Enhanced Decision-Making: By providing accurate T20 cricket score predictions, the project empowers teams and analysts to make informed and strategic decisions during matches. This can be particularly valuable for captains, coaches, and analysts in planning their team's approach in real-time.

Data-Driven Insights: The project utilizes historical match data and advanced modeling techniques to provide data-driven insights. These insights can aid cricket enthusiasts, teams, and broadcasters in gaining a deeper understanding of match dynamics and player performance.

Improved Fan Engagement: Accurate score predictions can enhance fan engagement and viewership. Fans can use predictions to fuel their excitement and participate in fantasy cricket games or engage in discussions about expected match outcomes.

Support for In-Game Strategy: The project's accurate score predictions can provide real-time support for in-game strategy development. Teams can use the predictions to adjust their tactics, including field placements, bowling changes, and target setting, based on the expected score. This dynamic strategy adaptation can lead to a more competitive and exciting cricket match, benefiting both players and fans by creating an enhanced viewing experience.

Disadvantages:

While the project offers numerous advantages, it also comes with some potential disadvantages:

Prediction Uncertainty: Cricket is a sport with inherent unpredictability, and even the most sophisticated models may struggle to account for unexpected events. This can result in prediction errors and the occasional inability to foresee match twists accurately.

Model Updates: Cricket is an evolving sport with rule changes and player performances that can impact match outcomes. Regular model updates are necessary to adapt to these changes, which can be resource intensive.

Privacy and Ethical Considerations: The project may need to consider ethical concerns related to data privacy, particularly if it includes player-

specific data. Striking a balance between data-driven analysis and privacy rights is crucial.

11. CONCLUSION

This project represents a significant stride in the realm of sports analytics, specifically in the context of T20 cricket. The project's primary objective of predicting the total score of a team batting first in a T20 match has been addressed through meticulous data acquisition, preprocessing, and the development of predictive models. The incorporation of performance metrics, such as Mean Absolute Error (MAE), R-squared (R²) score, and Mean Squared Error (MSE), has enabled an accurate assessment of the model's performance. This project's significance extends to multiple stakeholders. Cricket teams, captains, and coaches can harness the score predictions to make strategic decisions, optimize their gameplay, and maximize their chances of victory. Cricket enthusiasts benefit from an enhanced viewing experience, with data-driven insights contributing to a deeper understanding of the game's dynamics. Moreover, broadcasters and fantasy cricket platforms can use predictions to engage and excite their audiences. Nonetheless, the project is not without its challenges. The inherent unpredictability of cricket and the need for high-quality, up-to-date historical data pose ongoing hurdles. Furthermore, the project's success requires continuous model updates and a balance between data-driven analysis and cricket's intrinsic spontaneity. In the ever-evolving landscape of sports analytics, the "T20 Totalitarian: Mastering Score Predictions" project serves as an educational and practical resource. It underscores the value of data-driven decision-making, the potential of machine learning in sports, and the role of analytics in enhancing fan engagement and strategic acumen. As the project moves forward, it is essential to acknowledge its potential ethical considerations, particularly regarding data privacy and the responsible use of predictive modeling in sports. Overall, this project reflects a commitment to innovation and data-driven excellence in the world of cricket, contributing to a richer and more engaging cricketing experience for players, fans, and analysts alike.

12. FUTURE SCOPE

This project holds significant potential for future expansion and refinement, paving the way for a multitude of exciting possibilities in the domain of sports analytics. The future scope of this project includes:

Advanced Data Sources: This project can benefit from incorporating more comprehensive and granular data sources. This may include player-specific data, such as batting and bowling averages, recent form, and even physical and mental fitness indicators. Additionally, the inclusion of real-time match data could further enhance the model's predictive accuracy.

Player Performance Predictions: Beyond team score predictions, the project can evolve to predict individual player performances, including the number of runs scored, wickets taken, or player-of-the-match contenders. This level of granularity would offer a deeper understanding of player contributions and enhance fantasy cricket and team selection.

Live Updates and In-Game Predictions: Integrating the project into live match coverage, either through mobile apps or broadcast graphics, can provide viewers with real-time score predictions, adding an interactive and engaging dimension to the viewing experience. These in-game predictions can be valuable for in-match commentary and analysis.

Enhanced Machine Learning Models: Future iterations can explore more sophisticated machine learning algorithms and techniques, such as deep learning and neural networks, to further improve predictive accuracy. These models can adapt to changing cricket dynamics and account for a broader range of variables.

User-Driven Customization: Providing users with the ability to customize prediction models based on their preferences and factors they consider most relevant can be a valuable addition. This level of personalization can cater to the diverse needs and expectations of cricket fans and analysts.

Educational and Training Tools: The project can extend its reach as an educational tool by offering resources and tutorials on sports analytics and data science. It can help aspiring data scientists and sports enthusiasts learn about data analysis and machine learning in a sports context.

Privacy and Ethical Considerations: As the project collects and uses player data, it should stay abreast of evolving privacy regulations and ethical guidelines. Ensuring that data is handled responsibly and with due consideration for individual privacy rights is critical.

The future of the "T20 Totalitarian: Mastering Score Predictions" project is bright, with ample room for innovation and expansion. As the world of sports analytics continues to evolve, this project has the potential to shape the way cricket is analyzed, understood, and enjoyed, both by the avid cricket fan and the professional analyst.

13. APPENDIX

Source Code

Training.ipynb:

```
import numpy as np
import pandas as pd
from yaml import safe_load
import yaml
import os
from tqdm import tqdm

filenames = []

for file in os.listdir(r"C:\Users\maddu\OneDrive\Desktop\T20
Score Predictor\Dataset\t20s"):

    filenames.append(os.path.join(r"C:\Users\maddu\OneDrive\Deskt
op\T20 Score Predictor\Dataset\t20s", file))
filenames [0:5]

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 this file.")
        counter = counter + 1

final_df.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)

final_df['info.gender'].value_counts()

final_df = final_df[final_df['info.gender'] == 'male']
final_df.drop(columns=['info.gender'], inplace=True)
final_df

final_df['info.match_type'].value_counts()

final_df['info.overs'].value_counts()

final_df = final_df[final_df['info.overs'] == 20]
final_df.drop(columns=['info.overs', 'info.match_type'], inplace=True)
final_df

import pickle
pickle.dump(final_df, open('dataset_level1.pkl', 'wb'))

matches = pickle.load(open('dataset_level1.pkl', 'rb'))
matches.iloc[0]['innings'][0]['1st innings']['deliveries']

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']['p
layer_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)

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

delivery_df['bowling_team'] = delivery_df.apply(bowl, axis=1)
delivery_df

delivery_df['batting_team'].value_counts()

teams = [
    'Australia',
    'India',
    'Bangladesh',
    'New Zealand',
    'South Africa',
    'England',
    'West Indies',

```

```

    'Afghanistan',
    'Pakistan',
    'Sri Lanka'
]

delivery_df=delivery_df[delivery_df['batting_team'].isin(teams)]
delivery_df=delivery_df[delivery_df['bowling_team'].isin(teams)]

delivery_df.drop(columns = ['teams'], inplace=True)

output = delivery_df[['match_id', 'batting_team', 'bowling_team',
'ball', 'runs', 'player_dismissed', 'city', 'venue']]

pickle.dump(output,open('dataset_level2.pkl', 'wb'))

df = pickle.load(open('dataset_level2.pkl', 'rb'))

df.isnull().sum()

df[df['city'].isnull()]['venue'].value_counts()

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

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

df['current_score'] = df.groupby('match_id').cumsum()['runs']
df

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

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

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

```

```

df['player_dismissed'].apply(lambda x: 1 if x != '0' else '0')

df['player_dismissed'] = df['player_dismissed'].apply(lambda x: 1
if x != '0' else 0)
df

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']
df

df['crr'] = (df['current_score']*6) / df['balls_bowled']
df

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())

df['last_five'] = last_five
df

final_df =
df.groupby('match_id').sum()['runs'].reset_index().merge(df,on='m
atch_id')
final_df

final_df=final_df[['batting_team','bowling_team','city','current_
score','balls_left','wickets_left','crr','last_five','runs_x']]

final_df.dropna(inplace=True)

final_df.isnull().sum()

final_df = final_df.sample(final_df.shape[0])
final_df

X = final_df.drop(columns=['runs_x'])
y = final_df['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)
X_train

```

```

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 sklearn.linear_model import LinearRegression
import xgboost
from xgboost import XGBRegressor
from sklearn.metrics import
r2_score,mean_absolute_error,mean_squared_error

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

pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',LinearRegression())
])

pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
mse = (mean_squared_error(y_test,y_pred))
print(mse)
print(np.sqrt(mse))

pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',RandomForestRegressor())
])

pipe.fit(X_train,y_train)
y_pred = pipe.predict(X_test)
print(r2_score(y_test,y_pred))
print(mean_absolute_error(y_test,y_pred))
mse = (mean_squared_error(y_test,y_pred))
print(mse)
print(np.sqrt(mse))

pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),

```

```

        ('step3',XGBRegressor(n_estimators=500,learning_rate=0.5,max_
depth=12,random_state=1))
    ])

    pipe.fit(X_train,y_train)
    y_pred = pipe.predict(X_test)
    print(r2_score(y_test,y_pred))
    print(mean_absolute_error(y_test,y_pred))
    mse = (mean_squared_error(y_test,y_pred))
    print(mse)
    print(np.sqrt(mse))

    pipe = Pipeline(steps=[
        ('step1',trf),
        ('step2',StandardScaler()),
        ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.2,max_
depth=12,random_state=1))
    ])

    pipe.fit(X_train,y_train)
    y_pred = pipe.predict(X_test)
    print(r2_score(y_test,y_pred))
    print(mean_absolute_error(y_test,y_pred))
    mse = (mean_squared_error(y_test,y_pred))
    print(mse)
    print(np.sqrt(mse))

    pipe = Pipeline(steps=[
        ('step1',trf),
        ('step2',StandardScaler()),
        ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.1,max_
depth=12,random_state=1))
    ])

    pipe.fit(X_train,y_train)
    y_pred = pipe.predict(X_test)
    print(r2_score(y_test,y_pred))
    print(mean_absolute_error(y_test,y_pred))
    mse = (mean_squared_error(y_test,y_pred))
    print(mse)
    print(np.sqrt(mse))

    import pickle
    pickle.dump(pipe,open(r"C:\Users\maddu\OneDrive\Desktop\T20 Score
Predictor\Model\pipe.pkl", 'wb'))

```

app.py:

```
import streamlit as st

import pickle
import pandas as pd
import numpy as np

pipe = pickle.load(open(r"C:\Users\maddu\OneDrive\Desktop\T20
Score Predictor\Model\pipe.pkl", 'rb'))

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

cities = ['Colombo',
    'Mirpur',
    'Johannesburg',
    'Dubai',
    'Auckland',
    'Cape Town',
    'London',
    'Pallekele',
    'Barbados',
    'Sydney',
    'Melbourne',
    'Durban',
    'St Lucia',
    'Wellington',
    'Lauderhill',
    'Hamilton',
    'Centurion',
    'Manchester',
    'Abu Dhabi',
    'Mumbai',
    'Nottingham',
    'Southampton',
    'Mount Maunganui',
    'Chittagong',
    'Kolkata',
```

```

        'Lahore',
        'Delhi',
        'Nagpur',
        'Chandigarh',
        'Adelaide',
        'Bangalore',
        'St Kitts',
        'Cardiff',
        'Christchurch',
        'Trinidad']

st.title('Cricket Score Predictor')

col1, col2 = st.columns(2)

with col1:
    batting_team = st.selectbox('Select batting team',
sorted(teams))

with col2:
    bowling_team = st.selectbox('Select bowling team',
sorted(teams))

city = st.selectbox('Select city', sorted(cities))

col3,col4,col5 = st.columns(3)

with col3:
    current_score = st.number_input('Current Score')

with col4:
    overs = st.number_input('Overs Done (works for over > 5)')

with col5:
    wickets = st.number_input('Wickets Out')

last_five = st.number_input("Runs scored in last 5 overs")

if st.button('Predict Score'):
    balls_left = 120 - (overs * 6)
    wickets_left = 10 - wickets
    crr = current_score/overs

    input_df = pd.DataFrame(
        {'batting_team': [batting_team], 'bowling_team':
[bowling_team], 'city': city, 'current_score': [current_score],
        'balls_left': [balls_left], 'wickets_left': [wickets],
        'crr': [crr], 'last_five': [last_five]})

```

```
result = pipe.predict(input_df)
st.header("Predicted Score is: " + str(int(result[0])))
```

GitHub & Project Demo Link

GitHub:

<https://github.com/smartinternz02/SI-GuidedProject-603458-1697611252>

Project Demo Video Link:

[Project Demo T20 Score Prediction.mp4](#)