Men T20 Score Predictor

Vikas Kumar

Department of Computer Science, Ashoka University

CS-IS-3023: Data Science: Sports

Professor Debayan Gupta

May 16th, 2024

Abstract

This report details the development of the " Men T20 Score Predictor" project which aims to do score prediction for the first inning of the match. Using a Kaggle dataset containing 1764 International T20 matches, we processed and transformed cricket match data to build a predictive model using XGBoost. The report outlines steps including data extraction, feature engineering, model training, and evaluation, culminating in the creation of a web-based tool using streamlit for predicting cricket scores.

Introduction

The advent of data science has revolutionized various fields, including sports analytics. Cricket, with its rich and detailed match data, offers numerous opportunities for predictive modeling. This project aims to create a model that predicts the scores of ICC Men's T20 World Cup matches. By leveraging historical match data, the model provides insights valuable for teams, analysts, and fans.

Keywords

T20 Cricket, Score Prediction, Data Science, XGBoost, Feature Engineering, Streamlit

Methods:

The methodology employed in this study involved the utilization of two Jupyter Notebooks (.ipynb files). The first Jupyter Notebook, titled "T20-dataExtraction.ipynb," was dedicated to extracting the first inning data from a YAML format and converting it into a pandas DataFrame. This process facilitated the subsequent data manipulation and analysis steps.

The second Jupyter Notebook was tasked with extracting the essential eight features required for prediction from the DataFrame generated in the previous step. Additionally, this notebook encompassed the creation of a data processing pipeline and the application of a machine learning model. The specific details of each file are outlined below:

Procedure

1. Data Extraction: Convert YAML files to JSON and load into a DataFrame.
2. Data Cleaning: Filter relevant columns and handle missing data.
3. Feature Engineering: Extract features and transform categorical variables.
4. Model Training: Split the data, train the XGBoost model, and save the model.
5. Deployment: Develop a web interface using Streamlit to make the model accessible.

Each steps explained in detail with their given code.

Data Collection and Preparation

The data for this project was obtained from Kaggle's "Cricsheet - A Retrosheet for Cricket" dataset. The original data, provided in YAML format, containing 1764 matches data with 28 columns, required conversion to a more manipulable format for analysis.

Step-by-Step Process in “T20-dataExtraction.ipynb”:

1. Loading Data:
   YAML files containing match data were read into Python using the yaml library. This YAML file contains the 1764 International T20 matches data. The data was converted into JSON format for easier manipulation.

```
#import necessary library need for data extraction
import numpy as np
import pandas as pd
from yaml import safe_load
import os
from tqdm import tqdm
```

[82] ✓ 0.0s

```
# Extract all yaml files and saved into filenames variable
filenames = []
for file in os.listdir('data'):
    filenames.append(os.path.join('data',file))

filenames[0:5]
```

[83] ✓ 0.0s

```
['data\\1001349.yaml',
 'data\\1001351.yaml',
 'data\\1001353.yaml',
 'data\\1004729.yaml',
 'data\\1007655.yaml']
```

```python
# Initialize an empty list to store DataFrames
final_dfs = []
counter = 1

# function to convert yaml files into json files
for file in tqdm(filenames):
    try:
        with open(file, 'r') as f:
            df = pd.json_normalize(safe_load(f))
            df['match_id'] = counter
            final_dfs.append(df)  # Append each DataFrame to the list
            counter += 1
    except Exception as e:
        print(f"Error processing file {file}: {e}")

# Concatenate all DataFrames in the list into a single DataFrame
final_df = pd.concat(final_dfs, ignore_index=True)
```

✓ 6m 2.9s

```
100%|██████████| 1764/1764 [05:59<00:00,  4.90it/s]
Error processing file data\README.txt:
```

2. Data Cleaning:
   Extracted relevant data fields from the JSON objects and organized them into a pandas DataFrame. Initially when converted into pandas Dataframe, there are 28 columns, removed 14 non-essential columns.

```python
# drop unnecessary columns
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)
```

```python
final_df.sample(2)
```

	innings	info.dates	info.gender	info.match_type	info.outcome.winner	info.overs	info.player_of_match	info.teams	info.toss.decision	info.toss.winner	info.umpires	info.venue	match_id	info.city
459	[{'1st innings': {'team': 'Mozambique', 'deliv...	[2018-10-30]	male	IT20	Mozambique	20	[FA Cossa]	[Mozambique, Lesotho]	bat	Mozambique	[Justine Muzungu, C Thorburn]	Botswana Cricket Association Oval 1	460	Gaborone
68	[{'1st innings': {'team': 'Ireland', 'deliveri...	[2017-01-20]	male	T20	Ireland	20	[GC Wilson]	[Ireland, Scotland]	field	Scotland	[Ahmed Shah Pakteen, BB Pradhan]	Dubai International Cricket Stadium	69	NaN

3. Filtering Dataset:

The dataset underwent a filtering process based on two distinct parameters. Firstly, matches involving female players were removed, resulting in the exclusion of **546 matches** from the dataset. Secondly, the dataset was filtered to include only international T20 matches, thereby eliminating any matches played in the 50-over format, approximately 5 in number. The inning column within the dataset was stored in a dictionary format. To align with the purpose of this model, only the first innings match datasets were extracted and retained for further processing.

The initial DataFrame was refined to include only the relevant columns: "match_id, batting_team, bowling_team, ball, runs, player_dismissed, city, and venue". This refinement ensured that the dataset contained only the essential features required for the subsequent analysis. The DataFrame underwent a comprehensive data cleaning process, involving the handling of missing values and ensuring data consistency. This step was crucial in preparing a clean and clear dataset for the feature extraction process.

Upon completion of the data cleaning process, the refined DataFrame was saved, enabling its utilization in the subsequent feature extraction stage. The filtering and data cleaning processes were essential in ensuring the quality and relevance of the dataset, thereby enhancing the reliability and accuracy of the downstream analysis and modeling tasks. Use the pickle to save our data extraction process in part1_dataset.pkl file for future reference.

```python
# checking T20 datas based on gender
final_df['info.gender'].value_counts()
```

```
info.gender
male      1217
female     546
Name: count, dtype: int64
```

```python
# dropping off the women's datasets.
final_df = final_df[final_df['info.gender'] == 'male']
final_df.drop(columns=['info.gender'],inplace=True)
final_df.sample(2)
```

	innings	info.dates	info.match_type	info.outcome.winner	info.overs	info.player_of_match	info.teams	info.toss.decision	info.toss.winner	info.umpires	info.venue	match_id	info.city
370	[{'1st innings': {'team': 'Isle of Man', 'deli...	[2018-08-29]	IT20	Belgium	20	[Ashiqullah Said]	[Isle of Man, Belgium]	bat	Isle of Man	[Rafi Ahmed, J Jensen]	VRA Ground	371	Amstelveen
58	[{'1st innings': {'team': 'Namibia', 'deliveri...	[2017-01-15]	IT20	United Arab Emirates	20	[Mohammad Naveed]	[United Arab Emirates, Namibia]	bat	Namibia	[Ahmed Shah Durrani, BB Pradhan]	Sheikh Zayed Stadium	59	Abu Dhabi

```python
# checking match type, how many are T20 datasets are there.
final_df['info.match_type'].value_counts()
```

```
info.match_type
T20     966
IT20    251
Name: count, dtype: int64
```

```python
# checking Match type based on overs
final_df['info.overs'].value_counts()
```

✓ 0.0s Python

```
info.overs
20    1213
50       4
Name: count, dtype: int64
```

```python
# dropping the 50 overs(ODI) data
final_df = final_df[final_df['info.overs'] == 20]
final_df.drop(columns=['info.overs','info.match_type'],inplace=True)
final_df.head()
```

✓ 0.0s Python

	innings	info.dates	info.outcome.winner	info.player_of_match	info.teams	info.toss.decision	info.toss.winner	info.umpires	info.venue	match_id	info.city
0	[{'1st innings': {'team': 'Australia', 'delive...	[2017-02-17]	Sri Lanka	[DAS Gunaratne]	[Australia, Sri Lanka]	field	Sri Lanka	[MD Martell, P Wilson]	Melbourne Cricket Ground	1	NaN
1	[{'1st innings': {'team': 'Australia', 'delive...	[2017-02-19]	Sri Lanka	[DAS Gunaratne]	[Australia, Sri Lanka]	field	Sri Lanka	[SD Fry, SJ Nogajski]	Simonds Stadium, South Geelong	2	Victoria
2	[{'1st innings': {'team': 'Australia', 'delive...	[2017-02-22]	Australia	[A Zampa]	[Australia, Sri Lanka]	field	Sri Lanka	[MD Martell, P Wilson]	Adelaide Oval	3	NaN
3	[{'1st innings': {'team': 'Hong Kong', 'delive...	[2016-09-05]	Hong Kong	NaN	[Ireland, Hong Kong]	bat	Hong Kong	[R Black, AJ Neill]	Bready Cricket Club, Magheramason	4	Londonderry
4	[{'1st innings': {'team': 'Zimbabwe', 'deliver...	[2016-06-18]	Zimbabwe	[E Chigumbura]	[Zimbabwe, India]	field	India	[TJ Matibiri, RB Tiffin]	Harare Sports Club	5	NaN

Feature Engineering

In T20-featureExtraction.ipynb, the cleaned dataset was used to extract and engineer features essential for predictive modeling. The focus mainly to extract 8 essential features needed to the model are: batting_team, bowling_team, city, current_score, ball_left, wicket_left, current_run_rate, last_five_over. To extract each feature, the process involved explained below with its code:

Step-by-Step Process in “T20-featureExtraction.ipynb”:

1. Feature Extraction:
    - Derived key features from the cleaned dataset:
        - Batting Team: The team currently batting.
        - Bowling Team: The team currently bowling.

To derive these two features, use the team column in the delivery_df datasets extract earlier.

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

[104] ✓ 0.1s

```
delivery_df
```

[105] ✓ 0.0s

	match_id	batting_team	ball	batsman	bowler	runs	player_dismissed	city	venue	bowling_team
0	2	Australia	0.1	AJ Finch	SL Malinga	0	0	NaN	Melbourne Cricket Ground	Sri Lanka
1	2	Australia	0.2	AJ Finch	SL Malinga	0	0	NaN	Melbourne Cricket Ground	Sri Lanka
2	2	Australia	0.3	AJ Finch	SL Malinga	1	0	NaN	Melbourne Cricket Ground	Sri Lanka
3	2	Australia	0.4	M Klinger	SL Malinga	2	0	NaN	Melbourne Cricket Ground	Sri Lanka
4	2	Australia	0.5	M Klinger	SL Malinga	0	0	NaN	Melbourne Cricket Ground	Sri Lanka
...	...	...	...	...	...	...	...	...	...	...
144991	1214	Sri Lanka	19.3	SMSM Senanayake	MA Starc	1	0	Colombo	R Premadasa Stadium	Australia
144992	1214	Sri Lanka	19.4	DM de Silva	MA Starc	0	0	Colombo	R Premadasa Stadium	Australia
144993	1214	Sri Lanka	19.5	DM de Silva	MA Starc	0	DM de Silva	Colombo	R Premadasa Stadium	Australia
144994	1214	Sri Lanka	19.6	SMSM Senanayake	MA Starc	2	0	Colombo	R Premadasa Stadium	Australia
144995	1214	Sri Lanka	19.7	SMSM Senanayake	MA Starc	1	0	Colombo	R Premadasa Stadium	Australia

83989 rows × 10 columns

- City: Extracting the city feature, which represents the city or venue where the match is being played, presented a challenge due to the presence of 11,396 missing city names in the dataset. To overcome this obstacle, the following approach was adopted. The venue column, which contained the names of the stadiums or grounds where the matches were held, was utilized as an alternative source of location information. A data preprocessing step was implemented to extract the first word or name from the venue column entries. This extracted name was then used to populate the corresponding city column, effectively replacing the missing city values with the venue-derived location information.

```python
# checking how many cities name are missing
# using their venue column to replace with it
df[df['city'].isnull()]['venue'].value_counts()
```

[5] ✓ 0.0s

```
venue
Dubai International Cricket Stadium        5286
Pallekele International Cricket Stadium    2066
Melbourne Cricket Ground                   1453
Sharjah Cricket Stadium                     908
Sydney Cricket Ground                       749
Adelaide Oval                               498
Harare Sports Club                          372
Carrara Oval                                 64
Name: count, dtype: int64
```

```python
# filling missing cities names with their first venue name
cities = np.where(df['city'].isnull(),df['venue'].str.split().apply(lambda x:x[0]),df['city'])
```

[6] ✓ 0.0s

```python
# adding city column in dataframe
df['city'] = cities
```

[7] ✓ 0.0s

```python
# checking again any missing values in df
df.isnull().sum()
```

✓ 0.0s

```
match_id            0
batting_team        0
bowling_team        0
ball                0
runs                0
player_dismissed    0
city                0
venue               0
dtype: int64
```

```python
# no need of venue column now, so dropping off it.
df.drop(columns=['venue'],inplace=True)
df
```

More...

[9] ✓ 0.0s

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city
0	2	Australia	Sri Lanka	0.1	0	0	Melbourne
1	2	Australia	Sri Lanka	0.2	0	0	Melbourne
2	2	Australia	Sri Lanka	0.3	1	0	Melbourne
3	2	Australia	Sri Lanka	0.4	2	0	Melbourne
4	2	Australia	Sri Lanka	0.5	0	0	Melbourne
...	...	...	...	...	...	...	...
144991	1214	Sri Lanka	Australia	19.3	1	0	Colombo
144992	1214	Sri Lanka	Australia	19.4	0	0	Colombo
144993	1214	Sri Lanka	Australia	19.5	0	DM de Silva	Colombo
144994	1214	Sri Lanka	Australia	19.6	2	0	Colombo
144995	1214	Sri Lanka	Australia	19.7	1	0	Colombo

83617 rows × 7 columns

- Current Score: The current score of the batting team. To find the current score of the batting team, firstly we apply the groupby() function on match_id and then cumulative score, hence easily get the current score at each ball.

```python
# creating current_score column using runs columns
df['runs'] = pd.to_numeric(df['runs'], errors='coerce')
df['current_score'] = df.groupby('match_id')['runs'].cumsum()
df.head()
```

✓ 0.0s

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	current_score
0	2	Australia	Sri Lanka	0.1	0	0	Melbourne	0
1	2	Australia	Sri Lanka	0.2	0	0	Melbourne	0
2	2	Australia	Sri Lanka	0.3	1	0	Melbourne	1
3	2	Australia	Sri Lanka	0.4	2	0	Melbourne	3
4	2	Australia	Sri Lanka	0.5	0	0	Melbourne	3

- Balls Left: Number of balls remaining in the innings.To calculate the "balls left" feature in a T20 cricket match, we begin by identifying the total number of balls that can be bowled in an innings, which is typically 120 (20 overs with 6 balls each). From this, we subtract the number of balls already bowled, determined by multiplying the completed overs by six and adding the additional balls bowled in the current over. This subtraction gives us the number of deliveries remaining in the innings. Adjustments are made for any extra deliveries, such as wides or no-balls, which do not count towards the over's six legitimate deliveries. This final count of balls left provides a real-time view of how many opportunities the batting team still has to score runs, an essential factor in predicting the inning's final score.

```python
# finding over and ball no colums from the ball column
df['over'] = df['ball'].apply(lambda x:str(x).split(".")[0])
df['ball_no'] = df['ball'].apply(lambda x:str(x).split(".")[1])
```

[47] ✓ 0.0s

```python
# created balls_bowled using over and ball_no columns
df['balls_bowled'] = (df['over'].astype('int')*6) + df['ball_no'].astype('int')
```

[48] ✓ 0.0s

```python
# creating balls_left colums and handle edge case when in a match ball goes >120 due to wides or noball etc
df['balls_left'] = 120 - df['balls_bowled']
df['balls_left'] = df['balls_left'].apply(lambda x:0 if x<0 else x)
df.head()
```

[49] ✓ 0.0s

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	current_score	over	ball_no	balls_bowled	balls_left
0	2	Australia	Sri Lanka	0.1	0	0	Melbourne	0	0	1	1	119
1	2	Australia	Sri Lanka	0.2	0	0	Melbourne	0	0	2	2	118
2	2	Australia	Sri Lanka	0.3	1	0	Melbourne	1	0	3	3	117
3	2	Australia	Sri Lanka	0.4	2	0	Melbourne	3	0	4	4	116
4	2	Australia	Sri Lanka	0.5	0	0	Melbourne	3	0	5	5	115

- Wickets Left: Number of wickets remaining. To extract the "wickets left" feature in a T20 cricket match, we monitor the "player_dismissed" column, which logs whether a wicket fell on each delivery. Initially, all entries that signify a dismissal (typically player names) are converted to a numerical format, generally marking each dismissal with a '1'. By calculating the cumulative sum of these values, we can determine the total number of wickets taken up to any point in the innings. Subtracting this total from the maximum of 10 wickets available per innings gives us the "wickets left," reflecting the number of wickets the batting team still has in hand, crucial for assessing their potential to build or sustain their innings score.

```python
# Convert 'player_dismissed' column to 0 if '0', else 1
df['player_dismissed'] = df['player_dismissed'].apply(lambda x: 0 if x == '0' else 1)

# Convert 'player_dismissed' column to integer dtype
df['player_dismissed'] = df['player_dismissed'].astype('int')

# Cumulatively sum 'player_dismissed' within each match
df['player_dismissed_cumsum'] = df.groupby('match_id')['player_dismissed'].cumsum()

# Calculate remaining wickets left in the match
df['wickets_left'] = 10 - df['player_dismissed_cumsum']

df.head()
```

[84] 0.0s Python

match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	current_score	over	ball_no	balls_bowled	balls_left	player_dismissed_cumsum	wickets_left
2	Australia	Sri Lanka	0.1	0	0	Melbourne	0	0	1	1	119	0	10
2	Australia	Sri Lanka	0.2	0	0	Melbourne	0	0	2	2	118	0	10
2	Australia	Sri Lanka	0.3	1	0	Melbourne	1	0	3	3	117	0	10
2	Australia	Sri Lanka	0.4	2	0	Melbourne	3	0	4	4	116	0	10
2	Australia	Sri Lanka	0.5	0	0	Melbourne	3	0	5	5	115	0	10

- Current Run Rate: The run rate at the current point in the match. To calculate the "current run rate" feature in a T20 cricket match, we first compute the cumulative total of runs scored up to each delivery using a cumulative sum function on the "runs" column. This gives the total runs scored by the batting team at any given point in the innings. To determine the current run rate, we divide this cumulative run total by the number of overs completed, converted to a fraction of the 20-over innings. Specifically, the number of balls bowled is divided by six to convert it into overs for this calculation. The resulting figure represents the average number of runs scored per over at that point in the match, providing a real-time assessment of the batting team's scoring efficiency.

```python
# creating current_run_column using  balls_bowled and current_score
df['crr'] = (df['current_score']*6)/df['balls_bowled']
```

49] 0.0s Python

```python
df.head()
```

50] 0.0s Python

tch_id	batting_team	bowling_team	ball	runs	player_dismissed	city	current_score	over	ball_no	balls_bowled	balls_left	player_dismissed_cumsum	wickets_left	crr
2	Australia	Sri Lanka	0.1	0	0	Melbourne	0	0	1	1	119	0	10	0.0
2	Australia	Sri Lanka	0.2	0	0	Melbourne	0	0	2	2	118	0	10	0.0
2	Australia	Sri Lanka	0.3	1	0	Melbourne	1	0	3	3	117	0	10	2.0
2	Australia	Sri Lanka	0.4	2	0	Melbourne	3	0	4	4	116	0	10	4.5
2	Australia	Sri Lanka	0.5	0	0	Melbourne	3	0	5	5	115	0	10	3.6

- Runs in Last Five Overs: Runs scored in the last five overs to capture recent performance. To calculate the "runs in the last five overs" feature in a T20 cricket match, we utilize a rolling window calculation over the "runs" column, specifically focusing on the last 30 balls (equivalent to five overs, given that each over consists of six deliveries). This method sums the runs scored over each set of 30 consecutive deliveries throughout the innings. The result captures the total runs scored in the immediate past five overs at each point in the match, offering insights into the team's recent scoring momentum and effectiveness, which is crucial for predicting their potential scoring trajectory in the remaining overs.

```python
# define functions to calcualate the score of last five overs
xecute Cell (Ctrl+Alt+Enter) roupby('match_id')

match_ids = df['match_id'].unique()
last_five = []
for id in match_ids:
    last_five.extend(groups.get_group(id)['runs'].rolling(window=30, min_periods=1).sum().values.tolist())
```

51] ✓ 0.3s Python

```python
# adding last_five column in dataframe
df['last_five'] = last_five
df.head()
```

52] ✓ 0.0s Python

tting_team	bowling_team	ball	runs	player_dismissed	city	current_score	over	ball_no	balls_bowled	balls_left	player_dismissed_cumsum	wickets_left	crr	last_five
Australia	Sri Lanka	0.1	0	0	Melbourne	0	0	1	1	119	0	10	0.0	0.0
Australia	Sri Lanka	0.2	0	0	Melbourne	0	0	2	2	118	0	10	0.0	0.0
Australia	Sri Lanka	0.3	1	0	Melbourne	1	0	3	3	117	0	10	2.0	1.0
Australia	Sri Lanka	0.4	2	0	Melbourne	3	0	4	4	116	0	10	4.5	3.0
Australia	Sri Lanka	0.5	0	0	Melbourne	3	0	5	5	115	0	10	3.6	3.0

2. Data Preparation:
   - Ensured the dataset was reshuffled to avoid any potential bias.
   - Split the dataset into training and testing sets to facilitate model training and evaluation.

```python
final_df.sample(2)
```
[221] ✓ 0.0s

	batting_team	bowling_team	city	current_score	balls_left	wickets_left	crr	last_five	runs_x
13606	Afghanistan	Bangladesh	Dehradun	20	110	10	12.000000	20.0	145
69031	West Indies	Pakistan	Mirpur	107	18	5	6.294118	33.0	166

```python
# split the datasets into train and test datasets
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=0.3,random_state=1)
```
[222] ✓ 0.1s

```python
X_train.head()
```
[241] ✓ 0.0s

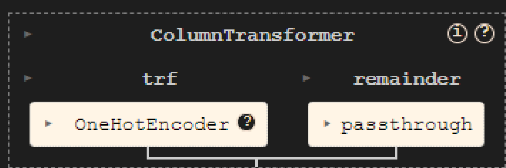
	batting_team	bowling_team	city	current_score	balls_left	wickets_left	crr	last_five
43643	South Africa	New Zealand	Barbados	58	69	8	6.823529	29.0
16637	England	West Indies	Basseterre	53	68	6	6.115385	25.0
14029	Bangladesh	West Indies	Lauderhill	98	59	7	9.639344	32.0
6415	New Zealand	West Indies	Mount Maunganui	157	41	9	11.924051	63.0
83357	Sri Lanka	India	Mirpur	68	46	6	5.513514	35.0

3. Handling Categorical Variables:
    - Applied a ColumnTransformer to convert categorical variables into a numerical format suitable for machine learning algorithms.

```python
# import all the necessary libraries and dependencies needs
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
```
[272] ✓ 0.0s

```python
# use ColumnTransformer to convert the categorical columns
trf = ColumnTransformer([
    ('trf', OneHotEncoder(drop='first', sparse_output=False), ['batting_team', 'bowling_team', 'city'])
], remainder='passthrough')

trf
```
[274] ✓ 0.0s



Model Training

The XGBoost model, or Extreme Gradient Boosting, is a highly efficient and powerful machine learning algorithm known for its speed and performance, particularly in structured data settings and regression tasks. In the specified model configuration, XGBRegressor is utilized with key parameters set to optimize prediction accuracy: n_estimators=1000 increases the number of gradient boosted trees used, enhancing the learning capability and potentially leading to better results. learning_rate=0.2 controls the rate at which the model adjusts errors from previous trees, balancing speed and accuracy. max_depth=12 allows the trees to grow deeper, enabling the model to capture more complex patterns at the risk of overfitting, mitigated by the large dataset. Finally, random_state=1 ensures reproducibility by providing a fixed seed for random number generation used in the algorithm's processes. These parameters are chosen to fine-tune the model's ability to analyze and predict scores accurately, leveraging its robust framework to handle diverse and complex datasets effectively.

Steps Involved:

1. Pipeline Creation:
    - Created a machine learning pipeline to streamline the process of data transformation and model fitting.
    - The pipeline included steps for handling missing values, transforming categorical features, and fitting the XGBoost model.

```
# creating the pipeline and use the XGboost model to train my datasets
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',StandardScaler()),
    ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.2,max_depth=12,random_state=1))
])
[275]  ✓ 0.4s
```

2. Model Training:
    - Trained the XGBoost model using the training set.
    - Evaluated the model's performance on the test set, ensuring the accuracy and reliability of the predictions.

```
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))
76]  ✓ 20.3s

0.9702514628863155
2.378308029256558
```

3. Model Saving:
    - Saved the trained model to a file (pipe.pkl) for deployment.

```
pickle.dump(pipe,open('pipe.pkl','wb'))
```

277] ✓ 0.3s

Deployment

The deployment of the ICC Men T20 World Cup Score Predictor using Streamlit involves creating an interactive web application that enables users to input specific match details for real-time score predictions. Here’s a concise overview of the deployment process:

Setup and User Interface

The Streamlit application initializes by importing necessary libraries and loading the pre-trained XGBoost model. The interface is set up with a user-friendly layout where users can input critical match details through dropdowns and numerical fields. These details include the batting and bowling teams, match city, current score, balls left, wickets remaining, current run rate, and runs scored in the last five overs.

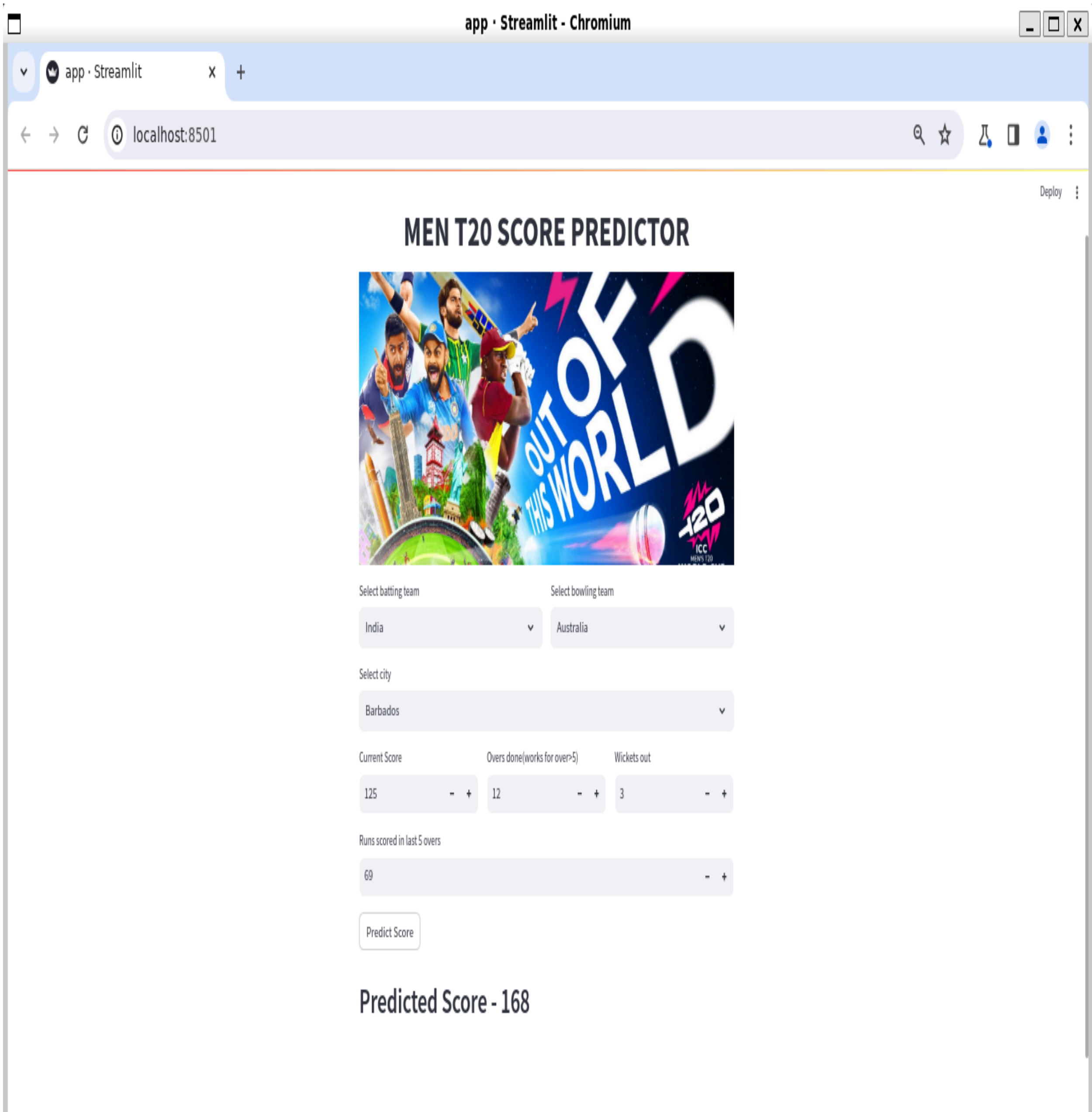
Data Collection And Prediction

Users submit the match details, which the application processes and formats appropriately for the XGBoost model. The model then uses this data to predict the final score based on the current match conditions and historical data patterns.

The predicted score is displayed directly on the web interface, offering users instant insight into the potential outcome of the match based on the inputs provided.

Streamlit's framework supports dynamic interactions, allowing the application to update predictions in real time as users modify the inputs. This makes the tool both accessible and valuable for cricket fans and analysts, providing advanced analytics without the need for technical expertise in data science.

The user interface was attached below:


app · Streamlit - Chromium
app · Streamlit
localhost:8501
Deploy
MEN T20 SCORE PREDICTOR
OUT OF THIS WORLD
ICC
Select batting team
India
Select bowling team
Australia
Select city
Barbados
Current Score
125
Overs done(works for over>5)
12
Wickets out
3
Runs scored in last 5 overs
69
Predict Score
Predicted Score - 168

Results

The XGBoost model excelled in predicting T20 cricket scores, achieving a mean absolute error (MAE) of 2.37 and an R2 score of 96.8%. These metrics indicate that the model's predictions are highly accurate, with minimal deviation from actual scores, reflecting its strong performance in capturing the variability of match outcomes. The high R2 score confirms that nearly all the variance in cricket scores is effectively captured by the model. The XGBoost model provides a reliable and precise tool for predicting scores in T20 matches, proving to be an invaluable asset for cricket analysts and enthusiasts.

Discussion

The success of the "Men T20 Score Predictor" highlights the potential of machine learning in sports analytics. The use of the XGBoost algorithm and thorough feature engineering played key roles in achieving precise predictions. To further enhance the model's performance, several improvements can be considered:

1. Diverse Data: Including data from a broader range of teams, especially lesser-known cricketing nations, can improve the model's generalizability and accuracy.
2. Environmental Factors: Integrating real-time weather conditions and pitch characteristics could refine predictions, as these factors significantly influence gameplay.
3. Advanced Techniques: Employing advanced strategies like ensemble learning could increase the robustness of predictions by combining multiple models to reduce errors.
4. Temporal and Player Dynamics: Incorporating changes in team strategies and player forms over time, as well as player-specific performance data, could provide a more nuanced and dynamic predictive model.

These enhancements will not only improve the accuracy of the model but also provide deeper insights for strategic decision-making in cricket

Conclusion

This project showcases the application of data science to cricket score prediction. By leveraging historical data and machine learning, we developed a robust model that can predict T20 match scores with high accuracy. The deployment of this model through a user-friendly web interface highlights its practical utility. Future work will focus on incorporating more data and exploring advanced modeling techniques to further improve predictive performance.

Works Cited

Wikipedia, https://www.kaggle.com/datasets/veeralakrishna/cricsheet-a-retrosheet-for-cricket/data. Accessed 17 May 2024.

Wikipedia, https://github.com/dhrupad17/T20-World-Cup-Score-Predictor?tab=readme-ov-file. Accessed 17 May 2024.

Wikipedia, the free encyclopedia, https://www.youtube.com/watch?v=tZd1okZiijo&ab_channel=CampusX. Accessed 17 May 2024.

Wikipedia, https://github.com/campusx-official/cricket-score-predictor. Accessed 17 May 2024.

Wikipedia, https://chatgpt.com/c/d084d173-d374-4226-b1dc-44c6a9b1a948. Accessed 17 May 2024.

Wikipedia, https://github.com/3241121/T20-Men-Score-Predictor. Accessed 17 May 2024.

"T20 Cricket Score Predictor — End to End ML project." *Harsh Mishra*, 20 November 2021, https://harshmishraandheri.medium.com/t20-cricket-score-predictor-end-to-end-ml-project-fbb3ec67dbb1. Accessed 17 May 2024.