

Overview

- In this project we are gonna work with Cricket T20 world cup data. This data isto be trained for prediction for the fnal result of the match. The trained model is to deployed on Streamlit application which provides a interactive application to input the data and provide prediction of the match.
- Firstly we are gonna transform the data to create our features which can be used to train the data.
- Model will trained for 5 overs performance.
 - Last five over performance of a team will be recorded and provided to the model for it to predict the outcome
- Features we will be focusing on are:
 1. batting team name
 2. bowling team name
 3. current score - (runs scored)
 4. ball - (balls played)
 5. over - (current overs)
 6. ball_left - (balls left out of total ie. 120)
 7. wicket_left - (wickets left out of total ie. 10)
 8. crr - (current run rate)
 9. 5 ovr score - (last 5 overs run scored)
 10. final score - (total runs scored at the end of match)
 11. venue - (stadium where match is being played)

In [1]:

1

pip install dash

In [2]:

1

import pandas as pd

2

import numpy as np

3

import pickle

4

import gradio as gr

5

import warnings

6

warnings.filterwarnings('ignore')

In [3]:

1

df = pd.read_csv('t20_wc.csv')

2

df.head()

Out[3]:

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

In [4]:

1

df.info(), print(), print(f'Data has {df.shape} rows and columns respectively')

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63888 entries, 0 to 63887
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   match_id         63888 non-null  int64
1   batting_team     63888 non-null  object
2   bowling_team     63888 non-null  object
3   ball             63888 non-null  float64
4   runs             63888 non-null  int64
5   player_dismissed 63888 non-null  object
6   city             55340 non-null  object
7   venue           63888 non-null  object
dtypes: float64(1), int64(2), object(5)
memory usage: 3.9+ MB

Data has (63888, 8) rows and columns respectively
```

Out[4]: (None, None, None)

In [5]:

1

df.nunique()

Out[5]: match_id 527
batting_team 10
bowling_team 10
ball 179
runs 8
player_dismissed 519
city 81
venue 94
dtype: int64

In [6]:

1

df.isnull().sum()

Out[6]: match_id 0
batting_team 0
bowling_team 0
ball 0
runs 0
player_dismissed 0
city 8548
venue 0
dtype: int64

In [7]:

1

df[df.duplicated()]

Out[7]:

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	venue
38615	618	Pakistan	South Africa	0.1	1	0	Abu Dhabi	Sheikh Zayed Stadium

till now, we have checked data for any erros.

- all the columns have correct data type
- Only city column has null values. we will check it later
- there are no repeated rows.
- **Conclusion:** data seems to be clean

In [8]:

1

df.drop(38615, inplace=True)

```
In [9]: 1 df[df['city'].isnull()][ 'venue'].value_counts()
```

Out[9]: Dubai International Cricket Stadium 2969
Pallekele International Cricket Stadium 2066
Melbourne Cricket Ground 1453
Sydney Cricket Ground 749
Adelaide Oval 498
Harare Sports Club 372
Sharjah Cricket Stadium 249
Sylhet International Cricket Stadium 128
Carrara Oval 64
Name: venue, dtype: int64

Type *Markdown* and LaTeX: α^2

- we can identify the missing city names has venue names. And mostly stadium is named after city.
- Thus, we can verify all city names with null values have venue names starting with the name of a city.
- We can extract the first word and add them in city column

```
In [10]: 1 df['city'] = df['city'].mask(df['city'].isnull(), df['venue'].str.split().str.get(0))
2 df.head()
```

Out[10]:

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

```
In [11]: 1 df.isnull().sum()
```

Out[11]: match_id 0
batting_team 0
bowling_team 0
ball 0
runs 0
player_dismissed 0
city 0
venue 0
dtype: int64

- now, no column has null values
- for our model to work best, we require atleast 5 matches in a city to have accurate data.
- single inning contains 120 balls. 5 inning contains 600 balls

```
In [12]: 1 city_count = df['city'].value_counts()
2 eli_city = city_count[city_count > 599].index.tolist()
3
4 print(eli_city, len(eli_city))
```

['Colombo', 'Mirpur', 'Johannesburg', 'Dubai', 'Auckland', 'Cape Town', 'London', 'Pallekele', 'Barbados', 'Sydney', 'Melbourne', 'Durban', 'St Lucia', 'Wellington', 'Lauderhi
ll', 'Hamilton', 'Centurion', 'Manchester', 'Abu Dhabi', 'Mumbai', 'Nottingham', 'Southampton', 'Mount Maunganui', 'Chittagong', 'Kolkata', 'Lahore', 'Delhi', 'Nagpur', 'Chand
igarh', 'Adelaide', 'Bangalore', 'St Kitts', 'Cardiff', 'Christchurch', 'Trinidad'] 35

```
In [13]: 1 df_model = df[df['city'].isin(eli_city)]
2 df_model
```

Out[13]:

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	venue
0	2	Australia	Sri Lanka	0.1	0	0	Melbourne	Melbourne Cricket Ground
1	2	Australia	Sri Lanka	0.2	0	0	Melbourne	Melbourne Cricket Ground
2	2	Australia	Sri Lanka	0.3	1	0	Melbourne	Melbourne Cricket Ground
3	2	Australia	Sri Lanka	0.4	2	0	Melbourne	Melbourne Cricket Ground
4	2	Australia	Sri Lanka	0.5	0	0	Melbourne	Melbourne Cricket Ground
...
63883	964	Sri Lanka	Australia	19.3	1	0	Colombo	R Premadasa Stadium
63884	964	Sri Lanka	Australia	19.4	0	0	Colombo	R Premadasa Stadium
63885	964	Sri Lanka	Australia	19.5	0	DM de Silva	Colombo	R Premadasa Stadium
63886	964	Sri Lanka	Australia	19.6	2	0	Colombo	R Premadasa Stadium
63887	964	Sri Lanka	Australia	19.7	1	0	Colombo	R Premadasa Stadium

50500 rows × 8 columns

Type *Markdown* and LaTeX: α^2

- Now lets work on transforming the data and creating our features

1. Below we have calculated the total score after each ball
2. Number of ongoing over
3. number of balls delivered
4. total balls delivered
5. total number of boundaries hit (4s & 6s)
6. number of balls left
7. number of players dismissed
8. total wickets left
9. Current run rate
10. last 5 over socre

```
In [14]: 1 df_model['score'] = df_model.groupby('match_id')['runs'].cumsum()
2 df_model['over'] = df_model['ball'].astype(int)
3 df_model['ball'] = df_model['ball'].astype(str).str.extract(r"\d.(\d)").astype(int)
4 df_model['total balls'] = (df_model['over']*6) + df_model['ball']
```

```
In [15]: 1 df_model['extras'] = 0
2 df_model["4s"] = 0
3 df_model["6s"] = 0
4
5 for i in df_model['match_id'].unique():
6     #extras
7     mask_extras = (df_model['match_id'] == i) & (df_model['ball'] > 6)
8     df_model.loc[mask_extras, 'extras'] = df_model.loc[mask_extras, 'ball'].gt(6).cumsum()
9
10    #4's
11    mask_4s = (df_model['match_id'] == i) & (df_model['runs'] == 4)
12    df_model.loc[mask_4s, '4s'] = (df_model.loc[mask_4s, 'runs'] == 4 ).cumsum()
13
14    #6's
15    mask_6s = (df_model['match_id'] == i) & (df_model['runs'] == 6)
16    df_model.loc[mask_6s, '6s'] = (df_model.loc[mask_6s, 'runs'] == 6).cumsum()
```

```
In [16]: 1 df_model['ball_left'] = 120 - df_model['total balls']
2 df_model['ball_left'].mask(df_model['ball_left']<0,0, inplace=True)
3
4 df_model['player_dismissed'] = df_model['player_dismissed'].mask(df_model['player_dismissed'] != '0', 1).astype(int)
5 df_model['player_dismissed'] = df_model.groupby('match_id')['player_dismissed'].cumsum()
6
7 df_model['wicket_left'] = 10 - df_model['player_dismissed']
```

```
In [17]: 1 df_model['crr'] = round(df_model.apply(lambda row: 0 if row['over'] == 0 else row['score'] / row['over'], axis=1),2)
2
3 df_model['5 ovr score'] = df_model.groupby('match_id')['runs'].rolling(window=30).sum().values.tolist()
```

```
In [18]: 1 df_model
```

Out[18]:

	match_id	batting_team	bowling_team	ball	runs	player_dismissed	city	venue	score	over	total balls	extras	4s	6s	ball_left	wicket_left	crr	5 ovr score	
	0	2	Australia	Sri Lanka	1	0	0	Melbourne	Melbourne Cricket Ground	0	0	1	0	0	0	119	10	0.00	NaN
	1	2	Australia	Sri Lanka	2	0	0	Melbourne	Melbourne Cricket Ground	0	0	2	0	0	0	118	10	0.00	NaN
	2	2	Australia	Sri Lanka	3	1	0	Melbourne	Melbourne Cricket Ground	1	0	3	0	0	0	117	10	0.00	NaN
	3	2	Australia	Sri Lanka	4	2	0	Melbourne	Melbourne Cricket Ground	3	0	4	0	0	0	116	10	0.00	NaN
	4	2	Australia	Sri Lanka	5	0	0	Melbourne	Melbourne Cricket Ground	3	0	5	0	0	0	115	10	0.00	NaN

	63883	964	Sri Lanka	Australia	3	1	8	Colombo	R Premadasa Stadium	125	19	117	0	0	0	3	2	6.58	32.0
	63884	964	Sri Lanka	Australia	4	0	8	Colombo	R Premadasa Stadium	125	19	118	0	0	0	2	2	6.58	32.0
	63885	964	Sri Lanka	Australia	5	0	9	Colombo	R Premadasa Stadium	125	19	119	0	0	0	1	1	6.58	32.0
	63886	964	Sri Lanka	Australia	6	2	9	Colombo	R Premadasa Stadium	127	19	120	0	0	0	0	1	6.68	33.0
	63887	964	Sri Lanka	Australia	7	1	9	Colombo	R Premadasa Stadium	128	19	121	6	0	0	0	1	6.74	32.0

50500 rows × 18 columns

- Now we need to know the total runs scored at the end of match. And add it to the original Data Frame

```
In [19]: 1 total_score = df_model.groupby('match_id')['runs'].sum().reset_index()
2 total_score
```

Out[19]:

	match_id	runs
	0	168
	1	187
	2	195
	3	194
	4	185

	411	129
	412	150
	413	120
	414	263
	415	128

```
In [20]: 1 df_model = df_model.merge(total_score, on='match_id')
2 df_model
```

Out[20]:

	match_id	batting_team	bowling_team	ball	runs_x	player_dismissed	city	venue	score	over	total balls	extras	4s	6s	ball_left	wicket_left	crr	5 ovr score	runs_y	
	0	2	Australia	Sri Lanka	1	0	0	Melbourne	Melbourne Cricket Ground	0	0	1	0	0	0	119	10	0.00	NaN	168
	1	2	Australia	Sri Lanka	2	0	0	Melbourne	Melbourne Cricket Ground	0	0	2	0	0	0	118	10	0.00	NaN	168
	2	2	Australia	Sri Lanka	3	1	0	Melbourne	Melbourne Cricket Ground	1	0	3	0	0	0	117	10	0.00	NaN	168
	3	2	Australia	Sri Lanka	4	2	0	Melbourne	Melbourne Cricket Ground	3	0	4	0	0	0	116	10	0.00	NaN	168
	4	2	Australia	Sri Lanka	5	0	0	Melbourne	Melbourne Cricket Ground	3	0	5	0	0	0	115	10	0.00	NaN	168

	50495	964	Sri Lanka	Australia	3	1	8	Colombo	R Premadasa Stadium	125	19	117	0	0	0	3	2	6.58	32.0	128
	50496	964	Sri Lanka	Australia	4	0	8	Colombo	R Premadasa Stadium	125	19	118	0	0	0	2	2	6.58	32.0	128
	50497	964	Sri Lanka	Australia	5	0	9	Colombo	R Premadasa Stadium	125	19	119	0	0	0	1	1	6.58	32.0	128
	50498	964	Sri Lanka	Australia	6	2	9	Colombo	R Premadasa Stadium	127	19	120	0	0	0	0	1	6.68	33.0	128
	50499	964	Sri Lanka	Australia	7	1	9	Colombo	R Premadasa Stadium	128	19	121	6	0	0	0	1	6.74	32.0	128

50500 rows × 19 columns

- Now filter the columns and select features

```
In [21]: 1 # df_model.columns
```

Out[21]: Index(['match_id', 'batting_team', 'bowling_team', 'ball', 'runs_x', 'player_dismissed', 'city', 'venue', 'score', 'over', 'total balls', 'extras', '4s', '6s', 'ball_left', 'wicket_left', 'crr', '5 ovr score', 'runs_y'], dtype='object')

```
In [22]: 1 order = ['batting_team', 'bowling_team', 'runs_x', 'over', 'ball_left', 'wicket_left', 'crr', '5 ovr score',
2           'runs_y']
3
4 final_df = df_model[order].copy()
```

```
In [23]: 1 final_df.rename(columns={'runs_x': 'current score', 'runs_y': 'final score', '5 ovr score':'5ovr_score'}, inplace=True)
```

```
In [24]: 1 final_df.isnull().sum()
```

Out[24]:

```
batting_team      0
bowling_team      0
current score     0
over              0
ball_left         0
wicket_left       0
crr               0
5ovr_score      12024
final score       0
dtype: int64
```

- As we calculated 5 ovr score , the first 4 over over data of this column will be null.
- our model requires data with minimum 5 overs
- The null rows will be droppped as keeping it will harm the model's accuracy.

```
In [25]: 1 final_df.dropna(inplace=True)
2 final_df.isnull().sum()
```

Out[25]:

```
batting_team      0
bowling_team      0
current score     0
over              0
ball_left         0
wicket_left       0
crr               0
5ovr_score        0
final score       0
dtype: int64
```

- We need complete data for the model.
- Thus, we will create a sample of complete data which will jumble up the rows to reduce the biasness.

```
In [26]: 1 model = final_df.sample(final_df.shape[0])
2 model
```

Out[26]:

	batting_team	bowling_team	current score	over	ball_left	wicket_left	crr	5ovr_score	final score
39024	Afghanistan	Bangladesh	0	11	53	5	4.91	18.0	72
24624	South Africa	New Zealand	1	6	79	9	8.33	32.0	170
4073	Pakistan	New Zealand	1	12	43	8	9.92	34.0	201
19568	Australia	New Zealand	1	7	77	7	7.57	34.0	150
25541	Pakistan	South Africa	1	14	33	6	7.50	57.0	148
...
27141	Sri Lanka	India	1	15	26	7	9.60	51.0	215
9010	Pakistan	New Zealand	1	9	61	9	7.33	39.0	166
19954	India	Bangladesh	1	9	61	9	8.33	35.0	180
39393	Pakistan	Australia	4	13	41	7	9.77	62.0	191
8629	Pakistan	Australia	0	7	74	10	9.14	49.0	150

```
In [27]: 1 # model.columns
```

Out[27]:

```
Index(['batting_team', 'bowling_team', 'current score', 'over', 'ball_left',
      'wicket_left', 'crr', '5ovr_score', 'final score'],
      dtype='object')
```

- We have successfully transformed the data and finalised our features.
- Next step is work training the model and creating the streamlit app

```
In [28]: 1 X = final_df.drop(columns = ['final score'])
2 y = final_df['final score']
```

```
In [29]: 1 from sklearn.model_selection import train_test_split
```

```
In [30]: 1 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=1)
```

```
In [31]: 1 from sklearn.compose import ColumnTransformer
2 from sklearn.preprocessing import OneHotEncoder
3 from sklearn.pipeline import Pipeline
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.ensemble import RandomForestRegressor
6 from xgboost import XGBRegressor
7 from sklearn.metrics import r2_score,mean_absolute_error
```

```
In [32]: 1 trf = ColumnTransformer([
2     ('trf',OneHotEncoder(sparse=False),['batting_team','bowling_team'])
3 ]
4 ,remainder='passthrough')
```

```
In [33]: 1 pipe = Pipeline(steps=[
2     ('step1',trf),
3     ('step2',StandardScaler()),
4     ('step3',XGBRegressor(n_estimators=1000,learning_rate=0.2,max_depth=12,random_state=1))
5 ])
```

```
In [34]: 1 pipe.fit(X_train,y_train)
2 y_pred = pipe.predict(X_test)
3 print(r2_score(y_test,y_pred))
4 print(mean_absolute_error(y_test,y_pred))
```

0.9193706684249296
5.049574699322548

```
In [35]: 1 pickle.dump(pipe, open("t20_model.pkl","wb"))
```

```
In [36]: 1 df_model['city'].unique()
```

Out[36]: array(['Melbourne', 'Adelaide', 'Mount Maunganui', 'Auckland',
'Southampton', 'Cardiff', 'Nagpur', 'Bangalore', 'Lauderhill',
'Dubai', 'Abu Dhabi', 'Sydney', 'Wellington', 'Hamilton',
'Barbados', 'Trinidad', 'Colombo', 'St Kitts', 'Manchester',
'Delhi', 'Lahore', 'Johannesburg', 'Centurion', 'Cape Town',
'Mumbai', 'Kolkata', 'Durban', 'Chandigarh', 'Christchurch',
'London', 'Nottingham', 'St Lucia', 'Pallekele', 'Mirpur',
'Chittagong'], dtype=object)

```
In [37]: 1 model.info()
```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 38476 entries, 39024 to 8629
Data columns (total 9 columns):
Column Non-Null Count Dtype
--- -
0 batting_team 38476 non-null object
1 bowling_team 38476 non-null object
2 current score 38476 non-null int64
3 over 38476 non-null int32
4 ball_left 38476 non-null int32
5 wicket_left 38476 non-null int32
6 crr 38476 non-null float64
7 Sovr_score 38476 non-null float64
8 final score 38476 non-null int64
dtypes: float64(2), int32(3), int64(2), object(2)
memory usage: 2.5+ MB


```
In [44]: 1 import ipywidgets as widgets
2 from IPython.display import display
3 import joblib
4 import pandas as pd
5
6 # Load your trained model
7 model = joblib.load('t20_model.pkl')
8
9 # Define widgets
10 batting_team = widgets.Dropdown(
11     options=['Australia', 'New Zealand', 'South Africa', 'England', 'India', 'West Indies', 'Pakistan', 'Bangladesh', 'Afghanistan', 'Sri Lanka'],
12     description='Batting Team:',
13 )
14
15 bowling_team = widgets.Dropdown(
16     options=['Australia', 'New Zealand', 'South Africa', 'England', 'India', 'West Indies', 'Pakistan', 'Bangladesh', 'Afghanistan', 'Sri Lanka'],
17     description='Bowling Team:',
18 )
19
20 current_score = widgets.IntText(
21     description='Current Score:',
22 )
23
24 overs = widgets.FloatText(
25     description='Overs Done (>5):',
26 )
27
28 wickets_out = widgets.IntText(
29     description='Wickets Out:',
30 )
31
32 runs_last_five_overs = widgets.IntText(
33     description='Runs Scored in Last Five Overs:',
34 )
35
36 output = widgets.Output()
37
38 def predict_score(b):
39     # Calculate additional features
40     ball_left = (120 - overs.value * 6)
41     wicket_left = 10 - wickets_out.value
42     crr = current_score.value / overs.value if overs.value > 5 else 0
43
44     input_data = pd.DataFrame({
45         'batting_team': [batting_team.value],
46         'bowling_team': [bowling_team.value],
47         'current_score': [current_score.value],
48         'over': [overs.value],
49         'ball_left': [ball_left],
50         'wicket_left': [wicket_left],
51         'crr': [crr],
52         '5ovr_score': [runs_last_five_overs.value]
53     })
54
55     # Use your trained model to predict the final score
56     predicted_score = model.predict(input_data)
57
58     with output:
59         output.clear_output()
60         print(f'Predicted Final Score: {int(predicted_score[0])}')
61
62 button = widgets.Button(description="Predict")
63 button.on_click(predict_score)
64
65 # Display the widgets
66 display(batting_team, bowling_team, current_score, overs, wickets_out, runs_last_five_overs, button, output)
67
```

Batting Team:

Bowling Te...

Current Sc...

Overs Don...

Wickets Out:

Runs Scor...

Predict