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]:
```
import pandas as pd
import numpy as np
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('t20_wc.csv')
df.head()
```

Out[2]:

	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 [3]:
```
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[3]: (None, None, None)

In [4]:
```
df.nunique()
```

Out[4]:
```
match_id            527
batting_team         10
bowling_team         10
ball                179
runs                  8
player_dismissed    519
city                 81
venue                94
dtype: int64
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
match_id              0
batting_team          0
bowling_team          0
ball                  0
runs                  0
player_dismissed      0
city               8548
venue                 0
dtype: int64
```

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

Out[6]:

	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 [7]:
```
df.drop(38615, inplace=True)
```

In [8]:
```
df[df['city'].isnull()]['venue'].value_counts()
```

Out[8]:
```
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 [9]:
```
df['city'] = df['city'].mask(df['city'].isnull(), df['venue'].str.split().str.get(0))
df.head()
```

Out[9]:

	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 [10]:
```
df.isnull().sum()
```

Out[10]:
```
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 [11]:
```
city_count = df['city'].value_counts()
eli_city = city_count[city_count > 599].index.tolist()

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 [12]:
```
df_model = df[df['city'].isin(eli_city)]
df_model
```

Out[12]:

	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 boundries 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 [13]:
```
df_model['score'] = df_model.groupby('match_id')['runs'].cumsum()
df_model['over'] = df_model['ball'].astype(int)
df_model['ball'] = df_model['ball'].astype(str).str.extract(r"\d.(\d)").astype(int)
df_model['total balls'] = (df_model['over']*6) + df_model['ball']
```

In [14]:
```
df_model['extras'] = 0
df_model["4s"] = 0
df_model["6s"] = 0

for i in df_model['match_id'].unique():
    #extras
    mask_extras = (df_model['match_id'] == i) & (df_model['ball'] > 6)
    df_model.loc[mask_extras, 'extras'] = df_model.loc[mask_extras, 'ball'].gt(6).cumsum()

    #4's
    mask_4s = (df_model['match_id'] == i) & (df_model['runs'] == 4)
    df_model.loc[mask_4s, '4s'] = (df_model.loc[mask_4s, 'runs'] == 4 ).cumsum()

    #6's
    mask_6s = (df_model['match_id'] == i) & (df_model['runs'] == 6)
    df_model.loc[mask_6s, '6s'] = (df_model.loc[mask_6s, 'runs'] == 6).cumsum()
```

In [15]:
```
df_model['ball_left'] = 120 - df_model['total balls']
df_model['ball_left'].mask(df_model['ball_left']<0,0, inplace=True)

df_model['player_dismissed'] = df_model['player_dismissed'].mask(df_model['player_dismissed'] != '0', 1).astype(int)
df_model['player_dismissed'] = df_model.groupby('match_id')['player_dismissed'].cumsum()

df_model['wicket_left'] = 10 - df_model['player_dismissed']
```

In [16]:
```
df_model['crr'] = round(df_model.apply(lambda row: 0 if row['over'] == 0 else row['score'] / row['over'], axis=1),2)

df_model['5 ovr score'] = df_model.groupby('match_id')['runs'].rolling(window=30).sum().values.tolist()
```

In [17]:
```
df_model
```

Out[17]:

	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 [18]:
```
total_score = df_model.groupby('match_id')['runs'].sum().reset_index()
total_score
```

Out[18]:

	match_id	runs
0	2	168
1	4	187
2	10	195
3	11	194
4	12	185
...	...	...
411	958	129
412	960	150
413	961	120
414	963	263
415	964	128

In [19]:
```
df_model = df_model.merge(total_score, on='match_id')
df_model
```

Out[19]:

	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 [20]:
```
# df_model.columns
```

In [21]:
```
order = ['batting_team', 'bowling_team', 'runs_x', 'over', 'ball_left', 'wicket_left', 'crr', '5 ovr score', 'city',
         'runs_y']

final_df = df_model[order].copy()
```

In [22]:
```
final_df.rename(columns={'runs_x': 'current score', 'runs_y': 'final score', '5 ovr score':'5ovr_score'}, inplace=True)
```

In [23]:
```
final_df.isnull().sum()
```

Out[23]:
```
batting_team         0
bowling_team         0
current score        0
over                 0
ball_left            0
wicket_left          0
crr                  0
5ovr_score       12024
city                 0
final score          0
dtype: int64
```

- As we caluculated `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 [24]:
```
final_df.dropna(inplace=True)
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       0
city             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 [25]:
```
model = final_df.sample(final_df.shape[0])
model
```

Out[25]:

	batting_team	bowling_team	current score	over	ball_left	wicket_left	crr	5ovr_score	city	final score
38784	New Zealand	Bangladesh	2	12	46	8	9.75	43.0	Mirpur	204
33762	West Indies	Australia	0	8	67	8	8.00	40.0	Colombo	205
36287	Australia	Pakistan	1	8	66	10	11.12	58.0	Dubai	168
38317	Sri Lanka	Pakistan	1	15	25	8	10.27	54.0	Dubai	211
39245	Sri Lanka	South Africa	1	9	63	8	9.11	39.0	Chittagong	165
...	...	...	...	...	...	...	...	...	...	...
40420	Pakistan	Bangladesh	0	16	21	7	9.25	62.0	Mirpur	190
43054	South Africa	Australia	1	15	29	6	4.80	21.0	Melbourne	101
27485	England	Pakistan	1	10	57	9	6.80	33.0	Dubai	148
14284	South Africa	New Zealand	1	17	15	3	7.12	19.0	Johannesburg	133
24261	South Africa	Afghanistan	6	8	67	8	8.62	30.0	Barbados	139

In [26]:
```
# model.columns
```

- We have successfully transformed the data and finalised our features.
- Next step is work training the model and creating the streamlit app
- below we have stored all features in `X` and output column in `y`

In [27]:
```
X = final_df.drop(columns = ['final score'])
y = final_df['final score']
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=1)
```

In [30]:
```
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
```

- `OneHotEncoder` helps encoding categorical data into binary format.
- As

In [31]:
```
trf = ColumnTransformer([
    ('trf',OneHotEncoder(sparse=False),['batting_team','bowling_team', 'city'])
]
,remainder='passthrough')
```

- Below we have created a pipline and defined 3 steps.
  1. OneHotEncode the categorical columns.
  2. StandardScaler is used to bring all the data under one scale range.
  3. XGRegressor helps in dealing with model containging many features and they provide a numberical outcome.

In [32]:
```
pipe = Pipeline(steps=[
            ('step1', trf),
            ('step2', StandardScaler()),
            ('step3', XGBRegressor(n_estimators=520, learning_rate=0.1, max_depth=12, random_state=1))
])
```

In [33]:
```
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))
```

```
0.9820417538009018
2.1399867936132355
```

- model trained and saved in pickle file format

In [34]:
```
pickle.dump(pipe, open("t20_model.pkl","wb"))
```

In [35]:
```
# df_model['city'].unique()
```

In [36]:
```
model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38476 entries, 38784 to 24261
Data columns (total 10 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   5ovr_score     38476 non-null  float64
 8   city           38476 non-null  object 
 9   final score    38476 non-null  int64  
dtypes: float64(2), int32(3), int64(2), object(3)
memory usage: 2.8+ MB
```

- below we created a interactive GUI for our model using `ipywidgets library` .
- we can input the parameters asked and it will display the predicted outcome.

In [37]:

```
from ipywidgets import widgets, Layout
from IPython.display import display
import joblib

# your pickle file
model = joblib.load('t20_model.pkl')


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

cities = ['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']

# input widgets and placeholders
batting_team = widgets.Dropdown(options=teams, description='Batting Team:', layout=Layout(width='200px'),
                                placeholder='Select Batting Team')

bowling_team = widgets.Dropdown(options=teams, description='Bowling Team:', layout=Layout(width='200px'),
                                placeholder='Select Bowling Team')

current_score = widgets.FloatText(description='Current Score:', layout=Layout(width='200px'),
                                  placeholder='Enter Current Score')

overs = widgets.FloatText(description='No. of Overs Done (>5):', layout=Layout(width='200px'),
                          placeholder='Enter Overs')

wickets = widgets.FloatText(description='Wickets Out:', layout=Layout(width='200px'), placeholder='Enter Wickets Out')
last_five_overs = widgets.FloatText(description='Runs Scored in Last Five Overs:',
                                    layout=Layout(width='200px'), placeholder='Enter Runs Scored')


city = widgets.Dropdown(options=cities, description='City:', layout=Layout(width='200px'), placeholder='Select City')

# function to predict final score
def predict_score(batting_team, bowling_team, current_score, overs, wickets, last_five_overs, city):
    balls_left = 120 - (overs * 6)
    wickets_left = 10 - wickets


    if overs > 5:
        crr = current_score / overs
    else:
        crr = 0

    input_data = {
        'batting_team': [batting_team],
        'bowling_team': [bowling_team],
        'current score': [current_score],
        'over': [overs],
        'ball_left': [balls_left],
        'wicket_left': [wickets_left],
        'crr': [crr],
        '5ovr_score': [last_five_overs],
        'city': [city]
    }

    # prediction
    predicted_score = model.predict(pd.DataFrame(input_data))
    result_label.value = "Predicted Score: " + str(int(predicted_score[0]))

# button for prediction
predict_button = widgets.Button(description='Predict Score')
predict_button.on_click(lambda b: predict_score(batting_team.value, bowling_team.value, current_score.value, overs.value,
                                                wickets.value, last_five_overs.value, city.value))


result_label = widgets.Label(value='Predicted Score:')

# widgets Display
display(batting_team)
display(bowling_team)
display(current_score)
display(overs)
display(wickets)
display(last_five_overs)
display(city)
display(predict_button)
display(result_label)

```

Batting Team: South Africa

Bowling Te… England

Current Sc… 152

No. of Over… 14

Wickets Out: 3

Runs Scor… 27

City: Melbourne

Predict Score

Predicted Score: 199