In [92]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [93]:
```
ball_by_ball = pd.read_csv('./Data/IPL_Ball_by_Ball_2008_2022.csv')
matches_result = pd.read_csv('./Data/IPL_Matches_Result_2008_2022.csv')
ipl_2023_teams = pd.read_csv('./Data/Ipl_2023 _cricketers - Team name.csv').rename(
    'Teams': 'team'
})
ipl_2023_venues = pd.read_csv('./Data/Ipl_2023 _cricketers - Venue.csv').rename(col
    'Venue': 'venue'
})
```

In [94]:
```
def log(*args):
    print('👉', *args)
```

In [95]:
```
def to_kebab_case(string):
    return '-'.join(
        string.replace(",", "").replace(".", "").split()
    ).lower()
```

Preprocessing

- Change column names, drop unnecessary columns [in ball_by_ball, matches_result]

In [96]:
```
ball_by_ball_orig = ball_by_ball

ball_by_ball = ball_by_ball.rename(columns={
    'ID': 'match_id',
    'ballnumber': 'ball_number',
    'non-striker': 'non_striker',
    'BattingTeam': 'batting_team',
}).loc[:, [
    'match_id',
    'innings',
    'batting_team',
    'overs',
    'ball_number',
    'batter',
    'bowler',
    'total_run',
]]
```

In [97]:
```
matches_result_orig = matches_result

matches_result = matches_result.rename(columns={
    'ID': 'match_id',
    'Team1': 'team_1',
    'Team2': 'team_2',
    'Venue': 'venue',
}).loc[:, [
    'match_id',
    'team_1',
    'team_2',
    'venue',
]]
```

In [98]:
```
print(ball_by_ball_orig.shape)
ball_by_ball_orig.head()
```

```
(225954, 17)
```

Out[98]:

	ID	innings	overs	ballnumber	batter	bowler	non-striker	extra_type	batsman
0	1312200	1	0	1	YBK Jaiswal	Mohammed Shami	JC Buttler	NaN	
1	1312200	1	0	2	YBK Jaiswal	Mohammed Shami	JC Buttler	legbyes	
2	1312200	1	0	3	JC Buttler	Mohammed Shami	YBK Jaiswal	NaN	
3	1312200	1	0	4	YBK Jaiswal	Mohammed Shami	JC Buttler	NaN	
4	1312200	1	0	5	YBK Jaiswal	Mohammed Shami	JC Buttler	NaN	

In [99]:
```
print(matches_result_orig.shape)
matches_result_orig.head()
```

```
(950, 20)
```

Out[99]:

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue
0	1312200	Ahmedabad	2022-05-29	2022	Final	Rajasthan Royals	Gujarat Titans	Narendra Modi Stadium, Ahmedabad
1	1312199	Ahmedabad	2022-05-27	2022	Qualifier 2	Royal Challengers Bangalore	Rajasthan Royals	Narendra Modi Stadium, Ahmedabad
2	1312198	Kolkata	2022-05-25	2022	Eliminator	Royal Challengers Bangalore	Lucknow Super Giants	Eden Gardens, Kolkata
3	1312197	Kolkata	2022-05-24	2022	Qualifier 1	Rajasthan Royals	Gujarat Titans	Eden Gardens, Kolkata
4	1304116	Mumbai	2022-05-22	2022	70	Sunrisers Hyderabad	Punjab Kings	Wankhede Stadium, Mumbai

In [100…

```
print(ball_by_ball.shape)
ball_by_ball.head()
```

```
(225954, 8)
```

Out[100]:

	match_id	innings	batting_team	overs	ball_number	batter	bowler	total_run
0	1312200	1	Rajasthan Royals	0	1	YBK Jaiswal	Mohammed Shami	0
1	1312200	1	Rajasthan Royals	0	2	YBK Jaiswal	Mohammed Shami	1
2	1312200	1	Rajasthan Royals	0	3	JC Buttler	Mohammed Shami	1
3	1312200	1	Rajasthan Royals	0	4	YBK Jaiswal	Mohammed Shami	0
4	1312200	1	Rajasthan Royals	0	5	YBK Jaiswal	Mohammed Shami	0

In [101…

```
print(matches_result.shape)
matches_result.head()
```

```
(950, 4)
```

Out[101]:

	match_id	team_1	team_2	venue
0	1312200	Rajasthan Royals	Gujarat Titans	Narendra Modi Stadium, Ahmedabad
1	1312199	Royal Challengers Bangalore	Rajasthan Royals	Narendra Modi Stadium, Ahmedabad
2	1312198	Royal Challengers Bangalore	Lucknow Super Giants	Eden Gardens, Kolkata
3	1312197	Rajasthan Royals	Gujarat Titans	Eden Gardens, Kolkata
4	1304116	Sunrisers Hyderabad	Punjab Kings	Wankhede Stadium, Mumbai

In [102…

```
log('match_id.nunique:', ball_by_ball.match_id.nunique())
log('batting_team.nunique:', ball_by_ball.batting_team.nunique())
log('union1d(batter, bowler).shape:', np.union1d(
    ball_by_ball.batter.unique(), ball_by_ball.bowler.unique()
).shape)
log('innings.unique:', ball_by_ball.innings.unique())
log('overs.unique:', ball_by_ball.overs.unique())
```

```
👉 match_id.nunique: 950
👉 batting_team.nunique: 18
👉 union1d(batter, bowler).shape: (652,)
👉 innings.unique: [1 2 3 4 5 6]
👉 overs.unique: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [103…

```
log('match_id.nunique:', matches_result.match_id.nunique())
log('venue.nunique:', matches_result.venue.nunique())
log('union1d(team_1, team_2).shape:', np.union1d(
    matches_result.team_1.unique(), matches_result.team_2.unique()
).shape)
```

```
👉 match_id.nunique: 950
👉 venue.nunique: 49
👉 union1d(team_1, team_2).shape: (18,)
```

- ## Get Venues Mapping

In [104…

```
matches_result_orig.groupby(['City', 'Venue'], dropna=False)['Venue'].describe()
```

Out[104]:

City	Venue	count	unique	top	freq
Abu Dhabi	**Sheikh Zayed Stadium**	29	1	Sheikh Zayed Stadium	29
	Zayed Cricket Stadium, Abu Dhabi	8	1	Zayed Cricket Stadium, Abu Dhabi	8
Ahmedabad	**Narendra Modi Stadium, Ahmedabad**	7	1	Narendra Modi Stadium, Ahmedabad	7
	Sardar Patel Stadium, Motera	12	1	Sardar Patel Stadium, Motera	12
Bangalore	**M Chinnaswamy Stadium**	65	1	M Chinnaswamy Stadium	65
Bengaluru	**M.Chinnaswamy Stadium**	15	1	M.Chinnaswamy Stadium	15
Bloemfontein	**OUTsurance Oval**	2	1	OUTsurance Oval	2
Cape Town	**Newlands**	7	1	Newlands	7
Centurion	**SuperSport Park**	12	1	SuperSport Park	12
Chandigarh	**Punjab Cricket Association IS Bindra Stadium**	10	1	Punjab Cricket Association IS Bindra Stadium	10
	Punjab Cricket Association IS Bindra Stadium, Mohali	11	1	Punjab Cricket Association IS Bindra Stadium, ...	11
	Punjab Cricket Association Stadium, Mohali	35	1	Punjab Cricket Association Stadium, Mohali	35
Chennai	**MA Chidambaram Stadium**	9	1	MA Chidambaram Stadium	9
	MA Chidambaram Stadium, Chepauk	48	1	MA Chidambaram Stadium, Chepauk	48
	MA Chidambaram Stadium, Chepauk, Chennai	10	1	MA Chidambaram Stadium, Chepauk, Chennai	10
Cuttack	**Barabati Stadium**	7	1	Barabati Stadium	7
Delhi	**Arun Jaitley Stadium**	14	1	Arun Jaitley Stadium	14
	Arun Jaitley Stadium, Delhi	4	1	Arun Jaitley Stadium, Delhi	4
	Feroz Shah Kotla	60	1	Feroz Shah Kotla	60

City	Venue	count	unique	top	freq
Dharamsala	**Himachal Pradesh Cricket Association Stadium**	9	1	Himachal Pradesh Cricket Association Stadium	9
Dubai	**Dubai International Cricket Stadium**	13	1	Dubai International Cricket Stadium	13
Durban	**Kingsmead**	15	1	Kingsmead	15
East London	**Buffalo Park**	3	1	Buffalo Park	3
Hyderabad	**Rajiv Gandhi International Stadium**	15	1	Rajiv Gandhi International Stadium	15
	Rajiv Gandhi International Stadium, Uppal	49	1	Rajiv Gandhi International Stadium, Uppal	49
Indore	**Holkar Cricket Stadium**	9	1	Holkar Cricket Stadium	9
Jaipur	**Sawai Mansingh Stadium**	47	1	Sawai Mansingh Stadium	47
Johannesburg	**New Wanderers Stadium**	8	1	New Wanderers Stadium	8
Kanpur	**Green Park**	4	1	Green Park	4
Kimberley	**De Beers Diamond Oval**	3	1	De Beers Diamond Oval	3
Kochi	**Nehru Stadium**	5	1	Nehru Stadium	5
Kolkata	**Eden Gardens**	77	1	Eden Gardens	77
	Eden Gardens, Kolkata	2	1	Eden Gardens, Kolkata	2
Mumbai	**Brabourne Stadium**	10	1	Brabourne Stadium	10
	Brabourne Stadium, Mumbai	17	1	Brabourne Stadium, Mumbai	17
	Dr DY Patil Sports Academy	17	1	Dr DY Patil Sports Academy	17
	Dr DY Patil Sports Academy, Mumbai	11	1	Dr DY Patil Sports Academy, Mumbai	11
	Wankhede Stadium	73	1	Wankhede Stadium	73
	Wankhede Stadium, Mumbai	31	1	Wankhede Stadium, Mumbai	31
Nagpur	**Vidarbha Cricket Association Stadium, Jamtha**	3	1	Vidarbha Cricket Association Stadium, Jamtha	3

City	Venue	count	unique	top	freq
Navi Mumbai	**Dr DY Patil Sports Academy, Mumbai**	9	1	Dr DY Patil Sports Academy, Mumbai	9
Port Elizabeth	**St George's Park**	7	1	St George's Park	7
Pune	**Maharashtra Cricket Association Stadium**	22	1	Maharashtra Cricket Association Stadium	22
	Maharashtra Cricket Association Stadium, Pune	13	1	Maharashtra Cricket Association Stadium, Pune	13
	Subrata Roy Sahara Stadium	16	1	Subrata Roy Sahara Stadium	16
Raipur	**Shaheed Veer Narayan Singh International Stadium**	6	1	Shaheed Veer Narayan Singh International Stadium	6
Rajkot	**Saurashtra Cricket Association Stadium**	10	1	Saurashtra Cricket Association Stadium	10
Ranchi	**JSCA International Stadium Complex**	7	1	JSCA International Stadium Complex	7
Sharjah	**Sharjah Cricket Stadium**	10	1	Sharjah Cricket Stadium	10
Visakhapatnam	**Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium**	13	1	Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket St...	13
NaN	**Dubai International Cricket Stadium**	33	1	Dubai International Cricket Stadium	33
	Sharjah Cricket Stadium	18	1	Sharjah Cricket Stadium	18

👇: https://www.iplt20.com/matches/schedule/men

In [105…
```
venue_mapping_normal = {
  "Arun Jaitley Stadium": "Arun Jaitley Stadium",
  "Arun Jaitley Stadium, Delhi": "Arun Jaitley Stadium",
  "Feroz Shah Kotla": "Arun Jaitley Stadium",
  "Barsapara Cricket Stadium": "Barsapara Cricket Stadium",
  "Barsapara Cricket Stadium, Guwahati": "Barsapara Cricket Stadium",
  "Bharat Ratna Shri Atal Bihari Vajpayee Ekana Cricket Stadium": "Bharat Ratna Shr
  "Bharat Ratna Shri Atal Bihari Vajpayee Ekana Cricket Stadium, Lucknow": "Bharat 
  "Eden Gardens": "Eden Gardens",
  "Eden Gardens, Kolkata": "Eden Gardens",
  "Himachal Pradesh Cricket Association Stadium": "Himachal Pradesh Cricket Associa
  "Himachal Pradesh Cricket Association Stadium, Dharamsala": "Himachal Pradesh Cri
  "M Chinnaswamy Stadium": "M Chinnaswamy Stadium",
  "M Chinnaswamy Stadium, Bengaluru": "M Chinnaswamy Stadium",
  "M Chinnaswamy Stadium, Bangalore": "M Chinnaswamy Stadium",
  "M.Chinnaswamy Stadium": "M Chinnaswamy Stadium",
  "M.Chinnaswamy Stadium, Bengaluru": "M Chinnaswamy Stadium",
  "M.Chinnaswamy Stadium, Bangalore": "M Chinnaswamy Stadium",
  "MA Chidambaram Stadium": "MA Chidambaram Stadium",
  "MA Chidambaram Stadium, Chennai": "MA Chidambaram Stadium",
  "MA Chidambaram Stadium, Chepauk": "MA Chidambaram Stadium",
  "MA Chidambaram Stadium, Chepauk, Chennai": "MA Chidambaram Stadium",
  "Narendra Modi Stadium": "Narendra Modi Stadium",
  "Narendra Modi Stadium, Ahmedabad": "Narendra Modi Stadium",
  "Punjab Cricket Association IS Bindra Stadium": "Punjab Cricket Association IS Bi
  "Punjab Cricket Association IS Bindra Stadium, Mohali": "Punjab Cricket Associati
  "Punjab Cricket Association Stadium, Mohali": "Punjab Cricket Association IS Bind
  "Rajiv Gandhi International Stadium": "Rajiv Gandhi International Stadium",
  "Rajiv Gandhi International Stadium, Hyderabad": "Rajiv Gandhi International Stad
  "Rajiv Gandhi International Stadium, Uppal": "Rajiv Gandhi International Stadium"
  "Sawai Mansingh Stadium": "Sawai Mansingh Stadium",
  "Sawai Mansingh Stadium, Jaipur": "Sawai Mansingh Stadium",
  "Wankhede Stadium": "Wankhede Stadium",
  "Wankhede Stadium, Mumbai": "Wankhede Stadium"
}
```

In [106…

```
venue_mapping_kebab = {
  "arun-jaitley-stadium": "Arun Jaitley Stadium",
  "arun-jaitley-stadium-delhi": "Arun Jaitley Stadium",
  "feroz-shah-kotla": "Arun Jaitley Stadium",
  "barsapara-cricket-stadium": "Barsapara Cricket Stadium",
  "barsapara-cricket-stadium-guwahati": "Barsapara Cricket Stadium",
  "bharat-ratna-shri-atal-bihari-vajpayee-ekana-cricket-stadium": "Bharat Ratna Shr
  "bharat-ratna-shri-atal-bihari-vajpayee-ekana-cricket-stadium-lucknow": "Bharat R
  "eden-gardens": "Eden Gardens",
  "eden-gardens-kolkata": "Eden Gardens",
  "himachal-pradesh-cricket-association-stadium": "Himachal Pradesh Cricket Associa
  "himachal-pradesh-cricket-association-stadium-dharamsala": "Himachal Pradesh Cric
  "m-chinnaswamy-stadium": "M Chinnaswamy Stadium",
  "m-chinnaswamy-stadium-bengaluru": "M Chinnaswamy Stadium",
  "m-chinnaswamy-stadium-bangalore": "M Chinnaswamy Stadium",
  "mchinnaswamy-stadium": "M Chinnaswamy Stadium",
  "mchinnaswamy-stadium-bengaluru": "M Chinnaswamy Stadium",
  "mchinnaswamy-stadium-bangalore": "M Chinnaswamy Stadium",
  "ma-chidambaram-stadium": "MA Chidambaram Stadium",
  "ma-chidambaram-stadium-chennai": "MA Chidambaram Stadium",
  "ma-chidambaram-stadium-chepauk": "MA Chidambaram Stadium",
  "ma-chidambaram-stadium-chepauk-chennai": "MA Chidambaram Stadium",
  "narendra-modi-stadium": "Narendra Modi Stadium",
  "narendra-modi-stadium-ahmedabad": "Narendra Modi Stadium",
  "punjab-cricket-association-is-bindra-stadium": "Punjab Cricket Association IS Bi
  "punjab-cricket-association-is-bindra-stadium-mohali": "Punjab Cricket Associatio
  "punjab-cricket-association-stadium-mohali": "Punjab Cricket Association IS Bindr
  "rajiv-gandhi-international-stadium": "Rajiv Gandhi International Stadium",
  "rajiv-gandhi-international-stadium-hyderabad": "Rajiv Gandhi International Stadi
  "rajiv-gandhi-international-stadium-uppal": "Rajiv Gandhi International Stadium",
  "sawai-mansingh-stadium": "Sawai Mansingh Stadium",
  "sawai-mansingh-stadium-jaipur": "Sawai Mansingh Stadium",
  "wankhede-stadium": "Wankhede Stadium",
  "wankhede-stadium-mumbai": "Wankhede Stadium"
}
```

In [107…
```
np.setdiff1d(matches_result.venue.unique(), list(venue_mapping_normal.keys()))
```

Out[107]:
```
array(['Barabati Stadium', 'Brabourne Stadium',
       'Brabourne Stadium, Mumbai', 'Buffalo Park',
       'De Beers Diamond Oval', 'Dr DY Patil Sports Academy',
       'Dr DY Patil Sports Academy, Mumbai',
       'Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium',
       'Dubai International Cricket Stadium', 'Green Park',
       'Holkar Cricket Stadium', 'JSCA International Stadium Complex',
       'Kingsmead', 'Maharashtra Cricket Association Stadium',
       'Maharashtra Cricket Association Stadium, Pune', 'Nehru Stadium',
       'New Wanderers Stadium', 'Newlands', 'OUTsurance Oval',
       'Sardar Patel Stadium, Motera',
       'Saurashtra Cricket Association Stadium',
       'Shaheed Veer Narayan Singh International Stadium',
       'Sharjah Cricket Stadium', 'Sheikh Zayed Stadium',
       "St George's Park", 'Subrata Roy Sahara Stadium',
       'SuperSport Park', 'Vidarbha Cricket Association Stadium, Jamtha',
       'Zayed Cricket Stadium, Abu Dhabi'], dtype=object)
```

- ## Get Teams Mapping

In [108…
```
set(matches_result['team_1'].unique()) == set(matches_result['team_2'].unique()) ==
```

Out[108]:
```
True
```

In [109…
```
# Rajasthan Royals
# Gujarat Titans
# Royal Challengers Bangalore
# Lucknow Super Giants
# Sunrisers Hyderabad
# Punjab Kings [Kings XI Punjab]
# Delhi Capitals [Delhi Daredevils]
# Mumbai Indians
# Chennai Super Kings
# Kolkata Knight Riders

team_mapping = { # 10 teams
 'Rajasthan Royals': 'Rajasthan Royals',
 'Gujarat Titans': 'Gujarat Titans',
 'Royal Challengers Bangalore': 'Royal Challengers Bangalore',
 'Lucknow Super Giants': 'Lucknow Super Giants',
 'Sunrisers Hyderabad': 'Sunrisers Hyderabad',
 'Mumbai Indians': 'Mumbai Indians',
 'Chennai Super Kings': 'Chennai Super Kings',
 'Kolkata Knight Riders': 'Kolkata Knight Riders',

 'Kings XI Punjab': 'Punjab Kings',
 'Punjab Kings': 'Punjab Kings',

 'Delhi Daredevils': 'Delhi Capitals',
 'Delhi Capitals': 'Delhi Capitals',
}
```

In [110…

```
print(np.setdiff1d(
   list(team_mapping.keys()), matches_result['team_1'].unique()
))

print(np.setdiff1d(
    matches_result['team_1'].unique(), list(team_mapping.keys())
))
```

```
[]
['Deccan Chargers' 'Gujarat Lions' 'Kochi Tuskers Kerala' 'Pune Warriors'
 'Rising Pune Supergiant' 'Rising Pune Supergiants']
```

- ## Apply Venues/Teams Mapping [in matches_result, ball_by_ball]

In [111…

```
matches_result.venue = matches_result.venue.map(venue_mapping_normal)

matches_result.team_1 = matches_result.team_1.map(team_mapping)
matches_result.team_2 = matches_result.team_2.map(team_mapping)

ball_by_ball.batting_team = ball_by_ball.batting_team.map(team_mapping)
```

In [116…

```
print(matches_result.loc[matches_result.venue.isnull()].shape)
```

```
(359, 4)
```

In [117…

```
print(matches_result.loc[matches_result.team_1.isnull()].shape)
print(matches_result.loc[matches_result.team_2.isnull()].shape)
```

```
(99, 4)
(96, 4)
```

In [118…

```
print(matches_result.shape)
print(matches_result.dropna().shape)
```

```
(950, 4)
(499, 4)
```

In [120…

```
print(ball_by_ball.shape)
print(ball_by_ball.dropna().shape)
```

```
(225954, 8)
(202849, 8)
```

In [27]:

```
ball_by_ball.loc[ball_by_ball.batting_team.isnull()].shape
```

Out[27]:

```
(23105, 8)
```

- ## Remove unnecessary Teams [in ball_by_ball] and Venues [in matches_result]

```
In [28]: matches_result = matches_result.dropna(subset=['team_1', 'team_2', 'venue'])
         # matches_result = matches_result.dropna(subset=['venue'])

         print(matches_result_orig.shape)
         print(matches_result.shape)

(950, 20)
(279, 4)
```

```
In [29]: ball_by_ball = ball_by_ball.dropna(subset=['batting_team'])

         print(ball_by_ball_orig.shape)
         print(ball_by_ball.shape)

(225954, 17)
(202849, 8)
```

- ## Select first 6 overs, Select innings 1 & 2, Map innings (1,2) to (0,1) [in ball_by_ball]

```
In [30]: ball_by_ball.innings.unique()

Out[30]: array([1, 2, 3, 4, 5, 6], dtype=int64)
```

```
In [31]: ball_by_ball.overs.unique()

Out[31]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19], dtype=int64)
```

```
In [32]: ball_by_ball = ball_by_ball.loc[(ball_by_ball.overs <= 5) & (ball_by_ball.innings <
         ball_by_ball.innings = ball_by_ball.innings.replace({1: 0, 2: 1})
         ball_by_ball.shape

Out[32]: (63652, 8)
```

```
In [33]: ball_by_ball.innings.unique()

Out[33]: array([0, 1], dtype=int64)
```

```
In [34]: ball_by_ball.overs.unique()

Out[34]: array([0, 1, 2, 3, 4, 5], dtype=int64)
```

- ## Grouping

```
In [35]: ball_by_ball_gb = ball_by_ball.groupby(['match_id', 'innings', 'batting_team'])
         total_runs = ball_by_ball_gb['total_run'].sum()
         batsmen = ball_by_ball_gb['batter'].unique()
         bowlers = ball_by_ball_gb['bowler'].unique()
```

```
In [36]: total_runs = total_runs.to_frame(name = 'total_runs').reset_index()
```

```
batsmen = batsmen.to_frame(name = 'batsmen').reset_index()
bowlers = bowlers.to_frame(name = 'bowlers').reset_index()
```

```
In [37]: data = total_runs.merge(
             batsmen.merge(bowlers, how='right', on=['match_id','innings','batting_team']),
             how='right', on=['match_id','innings','batting_team']
         )
```

```
In [38]: data = data.merge(matches_result, on=['match_id'])
```

```
In [39]: mask = data['batting_team'] == data['team_1']
         data.loc[mask, 'bowling_team'] = data['team_2']
         data.loc[~mask, 'bowling_team'] = data['team_1']
```

```
In [40]: # match_id == 829763, data for one innings is missing
         # match_id == 829813, total_runs for one innings is 2 (probably a mistake in data e
         data = data.drop(data[(data['match_id'] == 829763) | (data['match_id'] == 829813)].
```

```
In [41]: data['count_batsmen'] = [len(x) for x in data['batsmen']]
         data['count_bowlers'] = [len(x) for x in data['bowlers']]
```

```
In [42]: data = data.drop(columns=['match_id', 'batsmen', 'bowlers', 'team_1', 'team_2'])
         data = data[['venue', 'innings', 'batting_team', 'bowling_team', 'count_batsmen', '
```

```
In [43]: data
```

Out[43]:

	venue	innings	batting_team	bowling_team	count_batsmen	count_bowlers
0	{'aliases': ['M Chinnaswamy Stadium, Bengaluru...	0	Kolkata Knight Riders	Royal Challengers Bangalore	3	3
1	{'aliases': ['M Chinnaswamy Stadium, Bengaluru...	1	Royal Challengers Bangalore	Kolkata Knight Riders	6	3
2	{'aliases': ['Wankhede Stadium, Mumbai'], 'tag...	0	Mumbai Indians	Royal Challengers Bangalore	5	3
3	{'aliases': ['Wankhede Stadium, Mumbai'], 'tag...	1	Royal Challengers Bangalore	Mumbai Indians	3	3
4	{'aliases': ['Sawai Mansingh Stadium, Jaipur']...	0	Punjab Kings	Rajasthan Royals	3	3
...	...	...	...	...	...	...
552	{'aliases': ['Wankhede Stadium, Mumbai'], 'tag...	1	Mumbai Indians	Kolkata Knight Riders	2	4
553	{'aliases': ['MA Chidambaram Stadium, Chennai'...	0	Chennai Super Kings	Mumbai Indians	4	5
554	{'aliases': ['MA Chidambaram Stadium, Chennai'...	1	Mumbai Indians	Chennai Super Kings	4	2
555	{'aliases': ['Rajiv Gandhi International Stadi...	0	Mumbai Indians	Chennai Super Kings	4	3
556	{'aliases': ['Rajiv Gandhi International Stadi...	1	Chennai Super Kings	Mumbai Indians	3	4

554 rows × 7 columns

- ## Encoding of categorical inputs and feature scaling

In [44]:
```
X = data.iloc[:, :-1]
y = data["total_runs"]
```

In [45]:
```
ct = ColumnTransformer(transformers = [
    ('ohe', OneHotEncoder(categories = "auto", drop='first', sparse_output=False),
], remainder = 'passthrough')

scaler = StandardScaler()

X_ohe = pd.DataFrame(ct.fit_transform(X))
X_std = scaler.fit_transform(X_ohe)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_enco
de.py:170, in _unique_python(values, return_inverse, return_counts)
    169 try:
--> 170     uniques_set = set(values)
    171     uniques_set, missing_values = _extract_missing(uniques_set)

TypeError: unhashable type: 'dict'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In[45], line 7
      1 ct = ColumnTransformer(transformers = [
      2     ('ohe', OneHotEncoder(categories = "auto", drop='first', sparse_output=F
alse), ['venue', 'batting_team', 'bowling_team'])
      3 ], remainder = 'passthrough')
      5 scaler = StandardScaler()
----> 7 X_ohe = pd.DataFrame(ct.fit_transform(X))
      8 X_std = scaler.fit_transform(X_ohe)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_set_
output.py:140, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\compose\_co
lumn_transformer.py:727, in ColumnTransformer.fit_transform(self, X, y)
    724 self._validate_column_callables(X)
    725 self._validate_remainder(X)
--> 727 result = self._fit_transform(X, y, _fit_transform_one)
    729 if not result:
    730     self._update_fitted_transformers([])

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\compose\_co
lumn_transformer.py:658, in ColumnTransformer._fit_transform(self, X, y, func, fitte
d, column_as_strings)
    652 transformers = list(
    653     self._iter(
    654         fitted=fitted, replace_strings=True, column_as_strings=column_as_str
ings
    655     )
    656 )
    657 try:
--> 658     return Parallel(n_jobs=self.n_jobs)(
    659         delayed(func)(
    660             transformer=clone(trans) if not fitted else trans,
    661             X=_safe_indexing(X, column, axis=1),
    662                 y=y,
    663                 weight=weight,
    664                 message_clsname="ColumnTransformer",
    665                 message=self._log_message(name, idx, len(transformers)),
    666             )
    667             for idx, (name, trans, column, weight) in enumerate(transformers, 1)
    668         )
    669 except ValueError as e:
    670     if "Expected 2D array, got 1D array instead" in str(e):

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\paral
lel.py:63, in Parallel.__call__(self, iterable)
     58 config = get_config()
     59 iterable_with_config = (
     60     (_with_config(delayed_func, config), args, kwargs)
     61     for delayed_func, args, kwargs in iterable
     62 )
---> 63 return super().__call__(iterable_with_config)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:
1048, in Parallel.__call__(self, iterable)
   1039 try:
   1040     # Only set self._iterating to True if at least a batch
   1041     # was dispatched. In particular this covers the edge
   (...)
   1045     # was very quick and its callback already dispatched all the
   1046     # remaining jobs.
   1047     self._iterating = False
-> 1048     if self.dispatch_one_batch(iterator):
   1049         self._iterating = self._original_iterator is not None
   1051     while self.dispatch_one_batch(iterator):

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:
864, in Parallel.dispatch_one_batch(self, iterator)
    862     return False
    863 else:
--> 864     self._dispatch(tasks)
    865     return True

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:
782, in Parallel._dispatch(self, batch)
    780 with self._lock:
    781     job_idx = len(self._jobs)
--> 782     job = self._backend.apply_async(batch, callback=cb)
    783     # A job can complete so quickly than its callback is
    784     # called before we get here, causing self._jobs to
    785     # grow. To ensure correct results ordering, .insert is
    786     # used (rather than .append) in the following line
    787     self._jobs.insert(job_idx, job)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\_parallel_ba
ckends.py:208, in SequentialBackend.apply_async(self, func, callback)
    206 def apply_async(self, func, callback=None):
    207     """Schedule a func to be run"""
--> 208     result = ImmediateResult(func)
    209     if callback:
```

```
    210         callback(result)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\_parallel_ba
ckends.py:572, in ImmediateResult.__init__(self, batch)
    569 def __init__(self, batch):
    570     # Don't delay the application, to avoid keeping the input
    571     # arguments in memory
--> 572     self.results = batch()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:
263, in BatchedCalls.__call__(self)
    259 def __call__(self):
    260     # Set the default nested backend to self._backend but do not set the
    261     # change the default number of processes to -1
    262     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 263         return [func(*args, **kwargs)
    264                 for func, args, kwargs in self.items]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:
263, in <listcomp>(.0)
    259 def __call__(self):
    260     # Set the default nested backend to self._backend but do not set the
    261     # change the default number of processes to -1
    262     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 263         return [func(*args, **kwargs)
    264                 for func, args, kwargs in self.items]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\paral
lel.py:123, in _FuncWrapper.__call__(self, *args, **kwargs)
    121     config = {}
    122 with config_context(**config):
--> 123     return self.function(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\pipeline.p
y:893, in _fit_transform_one(transformer, X, y, weight, message_clsname, message, **
fit_params)
    891 with _print_elapsed_time(message_clsname, message):
    892     if hasattr(transformer, "fit_transform"):
--> 893         res = transformer.fit_transform(X, y, **fit_params)
    894     else:
    895         res = transformer.fit(X, y, **fit_params).transform(X)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_set_
output.py:140, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:878
, in TransformerMixin.fit_transform(self, X, y, **fit_params)
```

```
    874 # non-optimized default implementation; override when a better
    875 # method is possible for a given clustering algorithm
    876 if y is None:
    877     # fit method of arity 1 (unsupervised transformation)
--> 878     return self.fit(X, **fit_params).transform(X)
    879 else:
    880     # fit method of arity 2 (supervised transformation)
    881     return self.fit(X, y, **fit_params).transform(X)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessi
ng\_encoders.py:878, in OneHotEncoder.fit(self, X, y)
    874     self.sparse_output = self.sparse
    876 self._check_infrequent_enabled()
--> 878 fit_results = self._fit(
    879     X,
    880     handle_unknown=self.handle_unknown,
    881     force_all_finite="allow-nan",
    882     return_counts=self._infrequent_enabled,
    883 )
    884 if self._infrequent_enabled:
    885     self._fit_infrequent_category_mapping(
    886         fit_results["n_samples"], fit_results["category_counts"]
    887     )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessi
ng\_encoders.py:93, in _BaseEncoder._fit(self, X, handle_unknown, force_all_finite,
 return_counts)
     90 Xi = X_list[i]
     92 if self.categories == "auto":
---> 93     result = _unique(Xi, return_counts=return_counts)
     94     if return_counts:
     95         cats, counts = result

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_enco
de.py:41, in _unique(values, return_inverse, return_counts)
     10 """Helper function to find unique values with support for python objects.
     11 
     12 Uses pure python method for object dtype, and numpy method for
   (...)
     38     array. Only provided if `return_counts` is True.
     39 """
     40 if values.dtype == object:
---> 41     return _unique_python(
     42         values, return_inverse=return_inverse, return_counts=return_counts
     43     )
     44 # numerical
     45 return _unique_np(
     46     values, return_inverse=return_inverse, return_counts=return_counts
     47 )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_enco
de.py:178, in _unique_python(values, return_inverse, return_counts)
    176 except TypeError:
    177     types = sorted(t.__qualname__ for t in set(type(v) for v in values))
--> 178     raise TypeError(
    179         "Encoders require their input to be uniformly "
    180         f"strings or numbers. Got {types}"
    181     )
    182 ret = (uniques,)
    184 if return_inverse:

TypeError: Encoders require their input to be uniformly strings or numbers. Got ['di
ct']
```

- ## Train-test split

In [ ]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_std, y, test_size = 0.2)
```

In [ ]:
```python
def evaluate(regressor):
    regressor.fit(X_train, y_train)
    y_pred = np.round(regressor.predict(X_test), 2) # Round predictions to 2 decima
    rmse = np.sqrt(((y_test - y_pred) ** 2).mean()) # RMSE calculation
    mae = np.abs((y_test - y_pred)).mean() # MAE calculation
    print(f"RMSE: {rmse:.2f}") # Use f-string to format output
    print(f"MAE: {mae:.2f}") # Use f-string to format output
```

- ## Models

In [ ]:
```python
# from sklearn.metrics import r2_score
# AdaBoostRegressor(learning_rate=0.15, loss='exponential', n_estimators=20,
#                    random_state=2154)
```

In [ ]:
```python
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
evaluate(regressor)
```

In [ ]:
```python
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
evaluate(regressor)
```

In [ ]:
```python
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor()
evaluate(regressor)
```

In [ ]:
```python
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor()
evaluate(regressor)
```

In [ ]:
```python
from sklearn.svm import SVR
regressor = SVR()
evaluate(regressor)
```

In [ ]:
```python
import xgboost as xgb
regressor = xgb.XGBRegressor()
evaluate(regressor)
```

In [ ]:
```
# import tensorflow as tf
# from tensorflow.keras import layers, models

# # Define the model architecture
# model = models.Sequential([
#     layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
#     layers.Dense(128, activation='relu'),
#     layers.Dense(1)
# ])

# # Compile the model
# model.compile(optimizer='adam', loss='mean_absolute_error', metrics=['mae'])

# # Fit the model to the training data
# history = model.fit(X_train, y_train, epochs=200, batch_size=128, verbose=False)

# # Evaluate the model on the test set
# test_loss = model.evaluate(X_test, y_test)

# # Print the test loss
# print('Test loss:', test_loss)
```

In [ ]:
```
# import tensorflow as tf
# from tensorflow.keras import layers, models

# # Define a matrix of hyperparameters to test
# params = {
#     'batch_size': [16, 32],
#     'epochs': [50, 100],
#     'learning_rate': [0.001, 0.01]
# }

# # Define the model architecture
# def build_model(learning_rate=0.001):
#     model = models.Sequential([
#         layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
#         layers.Dense(32, activation='relu'),
#         layers.Dense(1)
#     ])
#     optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
#     model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
#     return model

# # Loop through the hyperparameter matrix and fit the model for each combination
# for batch_size in params['batch_size']:
#     for epochs in params['epochs']:
#         for learning_rate in params['learning_rate']:
#             print(f"Fitting model with batch_size={batch_size}, epochs={epochs},
#             model = build_model(learning_rate=learning_rate)
#             history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch
#             test_loss, test_mae = model.evaluate(X_test, y_test)
#             print(f"Test loss: {test_loss}, Test MAE: {test_mae}")
```