CAPSTONE PROJECT - WP_SP_01

Aniket Ganguly

PGPDSBA – JULY 2023

SOLUTION – NOTES 1

1. **Problem Understanding**
   a) Defining problem statement b) Need of the study/project c) Understanding business/social opportunity

a) Problem Statement

The aim of this project is to predict the performance of the Indian cricket team based on historical match data. Specifically, we seek to predict the outcome of matches (win/loss) using a variety of features related to the match conditions, player statistics, and other relevant variables.

b) Need of the study/project

Performance Prediction: Accurately predicting the outcomes of cricket matches can be highly valuable for various stakeholders:

- **Team Management:** Helps in strategizing and making informed decisions regarding team selection, game plans, and training focus.
- **Betting Agencies:** Assists in setting odds and understanding the likely outcomes of games.
- **Fans and Analysts:** Provides deeper insights and engagement with the sport by understanding the factors that influence match outcomes.

Improving Team Performance: By identifying key factors that contribute to winning or losing, the team can focus on improving specific areas, leading to better overall performance.

Resource Allocation: Helps in better allocation of resources and efforts towards aspects that have a significant impact on the match outcomes.

c) Understanding business/social opportunity

Commercial Opportunities:

- **Sports Analytics Services:** Developing a predictive analytics service that can be sold to cricket boards, franchises, and other organizations.
- **Betting Markets:** Enhancing the accuracy of betting odds and markets.

Enhancing Fan Engagement:

- **Fantasy Leagues:** Providing more accurate predictions can make fantasy leagues more competitive and engaging.

- **Broadcasting Enhancements:** Use predictions to enhance live commentary and analysis during the matches.

Data-Driven Decision Making:

- **Cricket Boards and Teams:** Use data to make strategic decisions regarding player selections, training regimes, and match tactics.

2. **Data Report**
a) Understanding how data was collected in terms of time, frequency and methodology b) Visual inspection of data (rows, columns, descriptive details) c) Understanding of attributes (variable info, renaming if required)

a) Understanding how data was collected in terms of time, frequency and methodology.

To effectively analyse and predict the performance of the Indian cricket team, understanding the context of the data collection is crucial. Here are key points to consider:

- **Time Period:** The data has been collected for cricket matches for all the seasons – Summer, Winter, Rainy.
- **Frequency:** The data has been collected for every match, every series and for specific tournaments as the matches has been played on different locations against different countries.

b) Visual inspection of data (rows, columns, descriptive details)

- **Total Rows and Columns:**

```
Dataset contains 2930 rows and 23 columns.
```

Fig 1

- **Top 5 rows:**

```
  Game_number Result  Avg_team_Age Match_light_type Match_format  \
0      Game_1   Loss          18.0              Day          ODI
1      Game_2    Win          24.0              Day          T20
2      Game_3   Loss          24.0    Day and Night          T20
3      Game_4    Win          24.0              NaN          ODI
4      Game_5   Loss          24.0            Night          ODI

   Bowlers_in_team  Wicket_keeper_in_team  All_rounder_in_team  \
0              3.0                      1                  3.0
1              3.0                      1                  4.0
2              3.0                      1                  2.0
3              2.0                      1                  2.0
4              1.0                      1                  3.0

  First_selection  Opponent  ... Max_run_scored_1over Max_wicket_taken_1over  \
0         Bowling  Srilanka  ...                 13.0                      3
1         Batting  Zimbabwe  ...                 12.0                      1
2         Bowling  Zimbabwe  ...                 14.0                      4
3         Bowling     Kenya  ...                 15.0                      4
4         Bowling  Srilanka  ...                 12.0                      4

  Extra_bowls_bowled  Min_run_given_1over  Min_run_scored_1over  \
0                0.0                    2                   3.0
1                0.0                    0                   3.0
2                0.0                    0                   3.0
3                0.0                    2                   3.0
4                0.0                    0                   3.0

   Max_run_given_1over  extra_bowls_opponent  player_highest_run  \
0                  6.0                     0                54.0
1                  6.0                     0                69.0
2                  6.0                     0                69.0
3                  6.0                     0                73.0
4                  6.0                     0                80.0

   Players_scored_zero  player_highest_wicket
0                    3                      1
1                    2                      1
2                    3                      1
3                    3                      1
4                    3                      1
```

Fig 2

- **Statistical Summary of the dataset**

	Avg_team_Age	Bowlers_in_team	Wicket_keeper_in_team
count	2833.000000	2848.000000	2930.0
mean	29.242852	2.913624	1.0
std	2.264230	1.023907	0.0
min	12.000000	1.000000	1.0
25%	30.000000	2.000000	1.0
50%	30.000000	3.000000	1.0
75%	30.000000	4.000000	1.0
max	70.000000	5.000000	1.0

	All_rounder_in_team	Audience_number	Max_run_scored_1over
count	2890.000000	2.849000e+03	2902.000000
mean	2.722491	4.626796e+04	15.199862
std	1.092699	4.859958e+04	3.661010
min	1.000000	7.063000e+03	11.000000
25%	2.000000	2.036300e+04	12.000000
50%	3.000000	3.434900e+04	14.000000
75%	4.000000	5.787600e+04	18.000000
max	4.000000	1.399930e+06	25.000000

	Max_wicket_taken_1over	Extra_bowls_bowled	Min_run_given_1over
count	2930.000000	2901.000000	2930.000000
mean	2.713993	11.252671	1.952560
std	1.080623	7.780829	1.678332
min	1.000000	0.000000	0.000000
25%	2.000000	6.000000	0.000000
50%	3.000000	10.000000	2.000000
75%	4.000000	15.000000	3.000000
max	4.000000	40.000000	6.000000

	Min_run_scored_1over	Max_run_given_1over	extra_bowls_opponent
count	2903.000000	2896.000000	2930.000000
mean	2.762659	8.669199	4.229693
std	0.705759	5.003525	3.626108
min	1.000000	6.000000	0.000000
25%	2.000000	6.000000	2.000000
50%	3.000000	6.000000	3.000000
75%	3.000000	9.250000	7.000000
max	4.000000	40.000000	18.000000

	player_highest_run
count	2902.000000
mean	65.889387
std	20.331614
min	30.000000
25%	48.000000
50%	66.000000
75%	84.000000
max	100.000000

Fig 3

- **Datatypes and Missing values in the columns**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Game_number            2930 non-null   object 
 1   Result                 2930 non-null   object 
 2   Avg_team_Age           2833 non-null   float64
 3   Match_light_type       2878 non-null   object 
 4   Match_format           2860 non-null   object 
 5   Bowlers_in_team        2848 non-null   float64
 6   Wicket_keeper_in_team  2930 non-null   int64  
 7   All_rounder_in_team    2890 non-null   float64
 8   First_selection        2871 non-null   object 
 9   Opponent               2894 non-null   object 
 10  Season                 2868 non-null   object 
 11  Audience_number        2849 non-null   float64
 12  Offshore               2866 non-null   object 
 13  Max_run_scored_1over   2902 non-null   float64
 14  Max_wicket_taken_1over 2930 non-null   int64  
 15  Extra_bowls_bowled     2901 non-null   float64
 16  Min_run_given_1over    2930 non-null   int64  
 17  Min_run_scored_1over   2903 non-null   float64
 18  Max_run_given_1over    2896 non-null   float64
 19  extra_bowls_opponent   2930 non-null   int64  
 20  player_highest_run     2902 non-null   float64
 21  Players_scored_zero    2930 non-null   object 
 22  player_highest_wicket  2930 non-null   object 
dtypes: float64(9), int64(4), object(10)
memory usage: 526.6+ KB
None
```

Fig 4

c) Understanding of attributes (variable info, renaming if required)

- **Renaming Variables:** Converted all the column names to lower case alphabets.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   game_number            2930 non-null   object
 1   result                 2930 non-null   object
 2   avg_team_age           2833 non-null   float64
 3   match_light_type       2878 non-null   object
 4   match_format           2860 non-null   object
 5   bowlers_in_team        2848 non-null   float64
 6   wicket_keeper_in_team  2930 non-null   int64
 7   all_rounder_in_team    2890 non-null   float64
 8   first_selection        2871 non-null   object
 9   opponent               2894 non-null   object
 10  season                 2868 non-null   object
 11  audience_number        2849 non-null   float64
 12  offshore               2866 non-null   object
 13  max_run_scored_1over   2902 non-null   float64
 14  max_wicket_taken_1over 2930 non-null   int64
 15  extra_bowls_bowled     2901 non-null   float64
 16  min_run_given_1over    2930 non-null   int64
 17  min_run_scored_1over   2903 non-null   float64
 18  max_run_given_1over    2896 non-null   float64
 19  extra_bowls_opponent   2930 non-null   int64
 20  player_highest_run     2902 non-null   float64
 21  players_scored_zero    2930 non-null   object
 22  player_highest_wicket  2930 non-null   object
dtypes: float64(9), int64(4), object(10)
memory usage: 526.6+ KB
None
```

Fig 5

- **Missing Values:** Missing values has been treated, the missing values in numerical columns has been filled with mean and the missing values in categorical columns have been filled with mode.

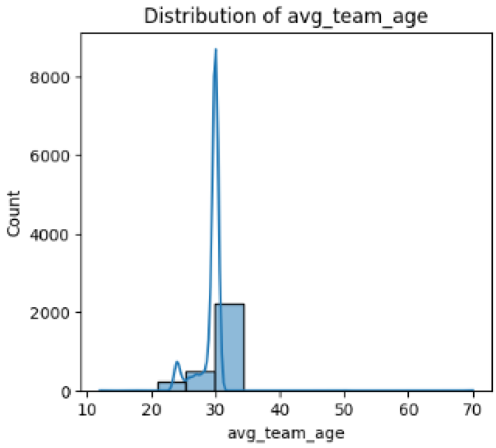
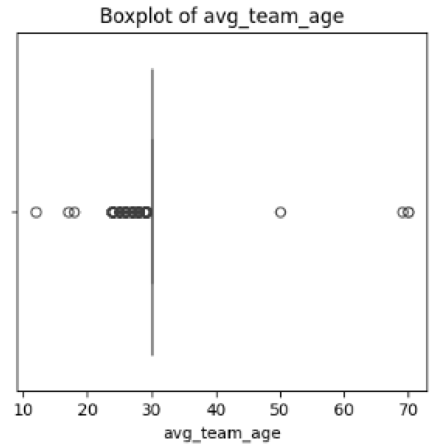
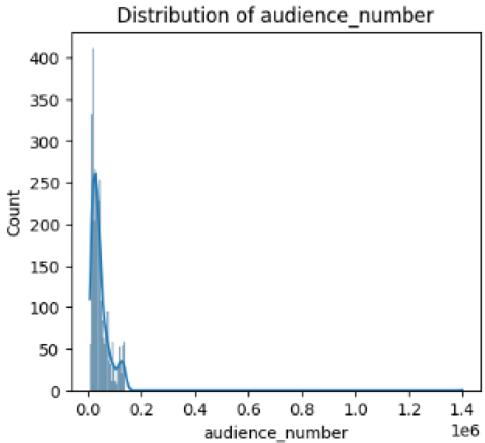
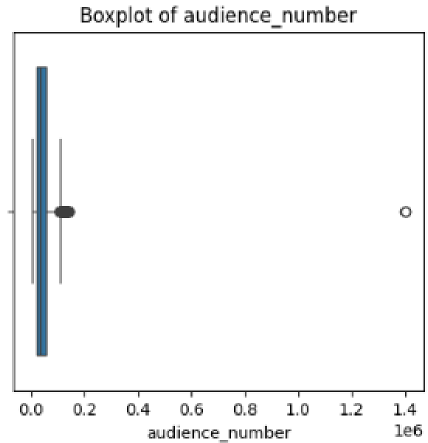
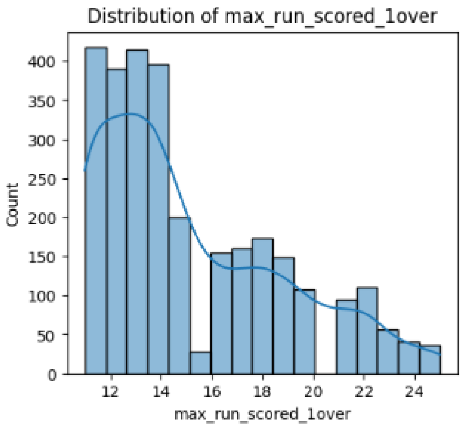
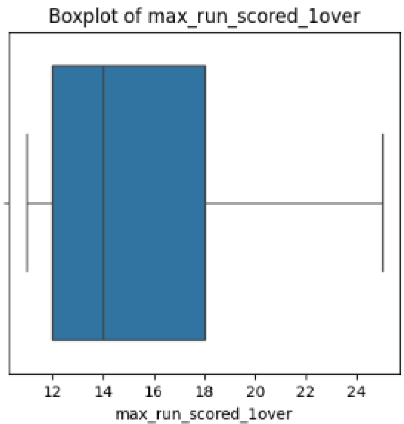
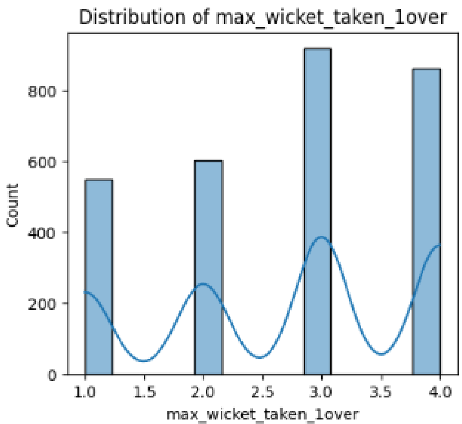
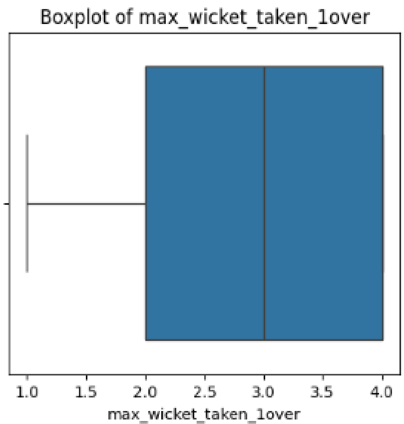
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   game_number            2930 non-null   object
 1   result                 2930 non-null   object
 2   avg_team_age           2930 non-null   float64
 3   match_light_type       2930 non-null   object
 4   match_format           2930 non-null   object
 5   bowlers_in_team        2930 non-null   float64
 6   wicket_keeper_in_team  2930 non-null   int64
 7   all_rounder_in_team    2930 non-null   float64
 8   first_selection        2930 non-null   object
 9   opponent               2930 non-null   object
 10  season                 2930 non-null   object
 11  audience_number        2930 non-null   float64
 12  offshore               2930 non-null   object
 13  max_run_scored_1over   2930 non-null   float64
 14  max_wicket_taken_1over 2930 non-null   int64
 15  extra_bowls_bowled     2930 non-null   float64
 16  min_run_given_1over    2930 non-null   int64
 17  min_run_scored_1over   2930 non-null   float64
 18  max_run_given_1over    2930 non-null   float64
 19  extra_bowls_opponent   2930 non-null   int64
 20  player_highest_run     2930 non-null   float64
 21  players_scored_zero    2930 non-null   float64
 22  player_highest_wicket  2930 non-null   float64
dtypes: float64(11), int64(4), object(8)
memory usage: 526.6+ KB
None
```

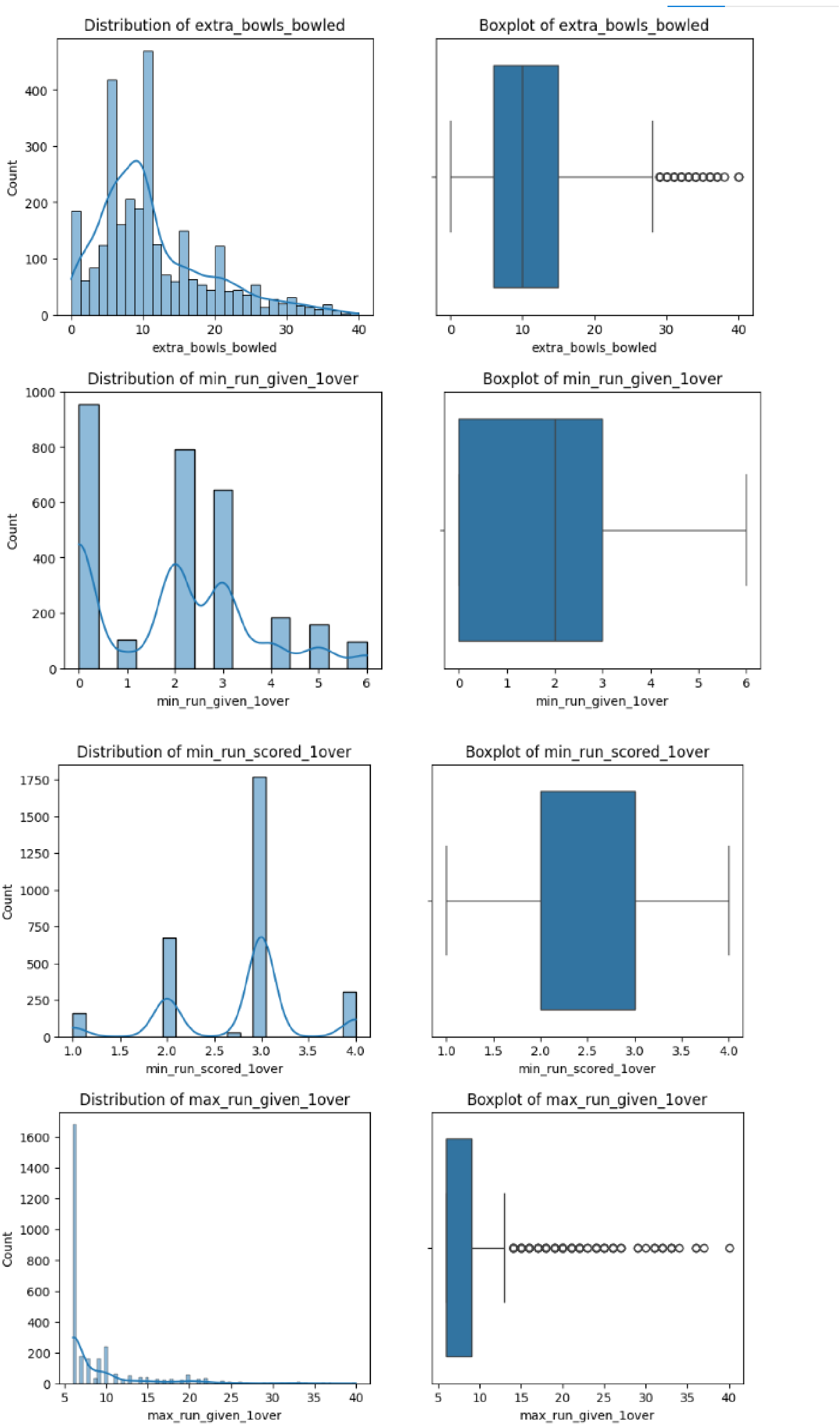
Fig 6

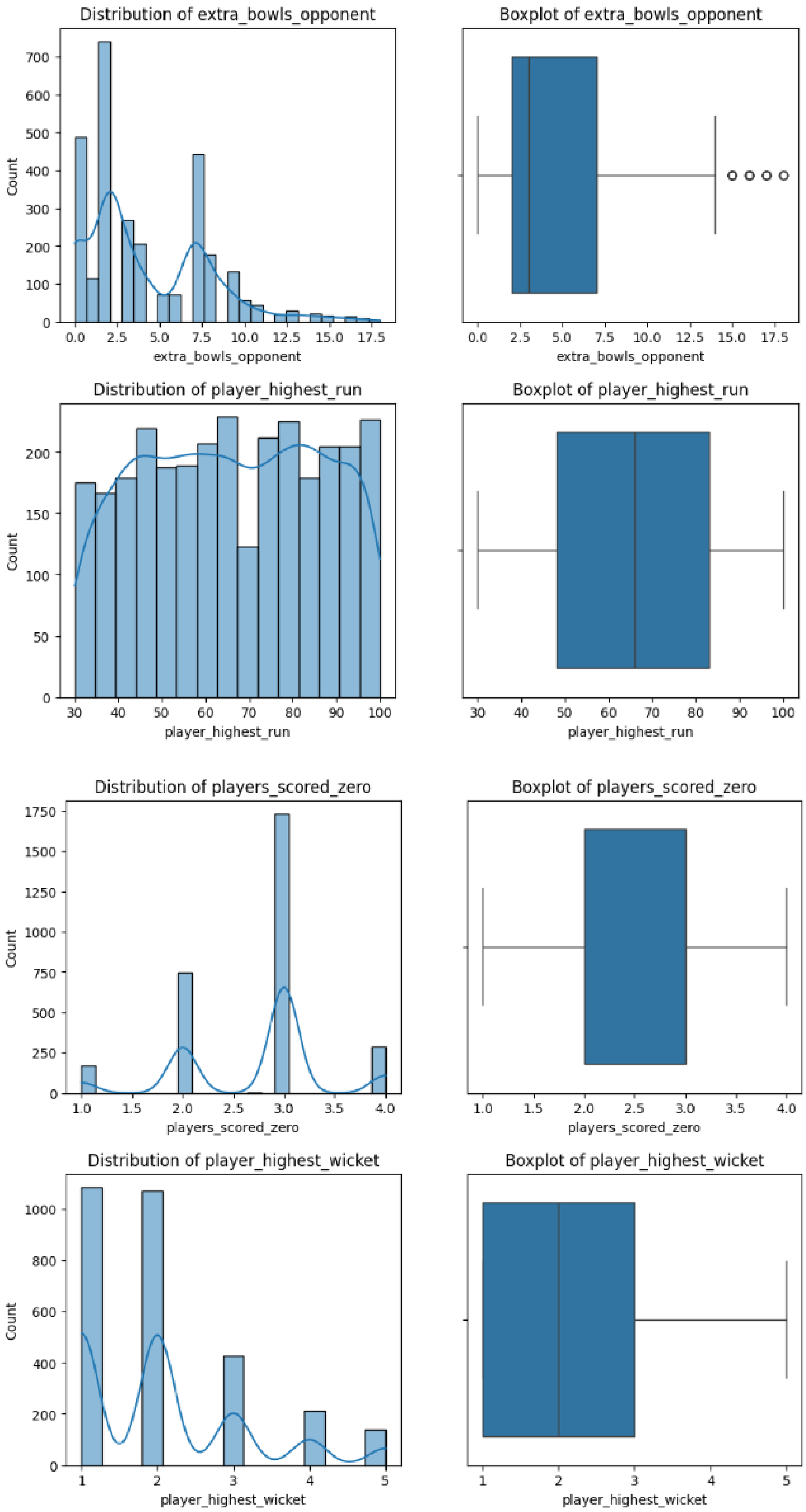
3. **Exploratory Data Analysis**
   a) **Univariate analysis (distribution and spread for every continuous attribute, distribution of data in categories for categorical ones) b) Bivariate analysis (relationship between different variables , correlations) c) Removal of unwanted variables (if applicable) d) Missing Value treatment (if applicable) e) Outlier treatment (if required) f) Variable transformation (if applicable) g) Addition of new**

a) **Univariate analysis (distribution and spread for every continuous attribute, distribution of data in categories for categorical ones)**

- **Continuous Attributes**


Distribution of avg_team_age
Count
avg_team_age

Boxplot of avg_team_age
avg_team_age

Distribution of audience_number
Count
audience_number
1e6

Boxplot of audience_number
audience_number
1e6

Distribution of max_run_scored_1over
Count
max_run_scored_1over

Boxplot of max_run_scored_1over
max_run_scored_1over

Distribution of max_wicket_taken_1over
Count
max_wicket_taken_1over

Boxplot of max_wicket_taken_1over
max_wicket_taken_1over


Distribution of extra_bowls_bowled
Count
extra_bowls_bowled
Boxplot of extra_bowls_bowled
extra_bowls_bowled
Distribution of min_run_given_1over
Count
min_run_given_1over
Boxplot of min_run_given_1over
min_run_given_1over
Distribution of min_run_scored_1over
Count
min_run_scored_1over
Boxplot of min_run_scored_1over
min_run_scored_1over
Distribution of max_run_given_1over
Count
max_run_given_1over
Boxplot of max_run_given_1over
max_run_given_1over


Distribution of extra_bowls_opponent
Boxplot of extra_bowls_opponent
Count
extra_bowls_opponent
Distribution of player_highest_run
Boxplot of player_highest_run
player_highest_run
Distribution of players_scored_zero
Boxplot of players_scored_zero
players_scored_zero
Distribution of player_highest_wicket
Boxplot of player_highest_wicket
player_highest_wicket

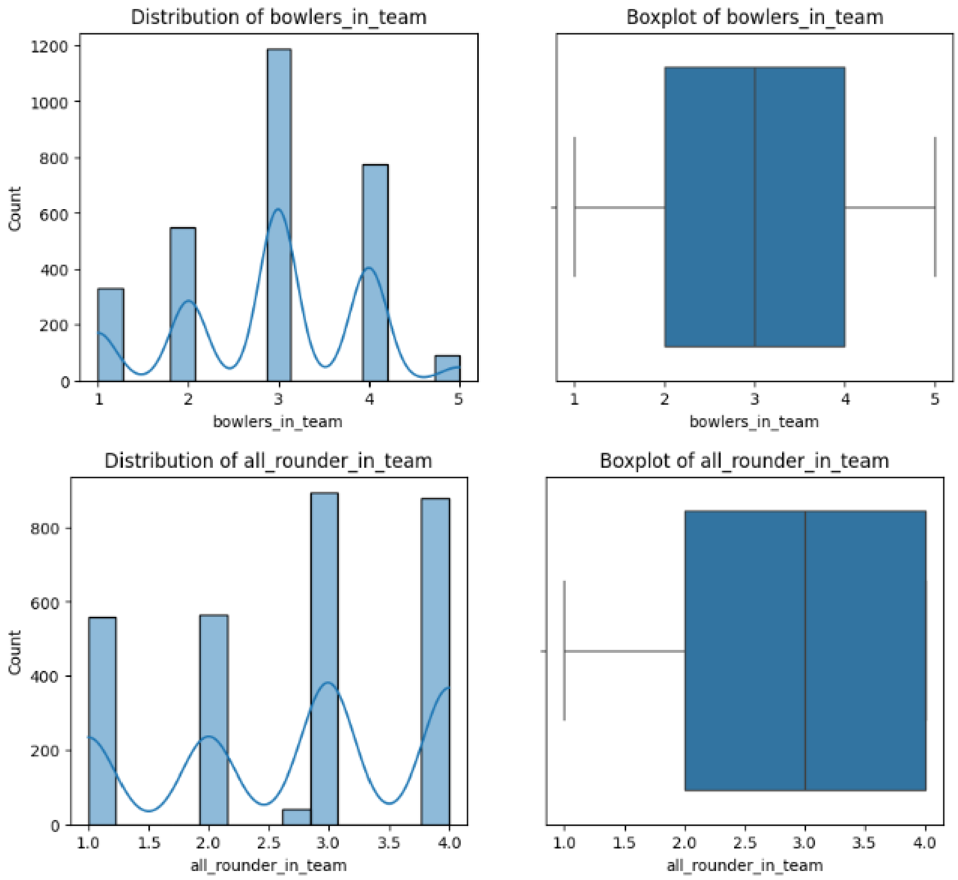
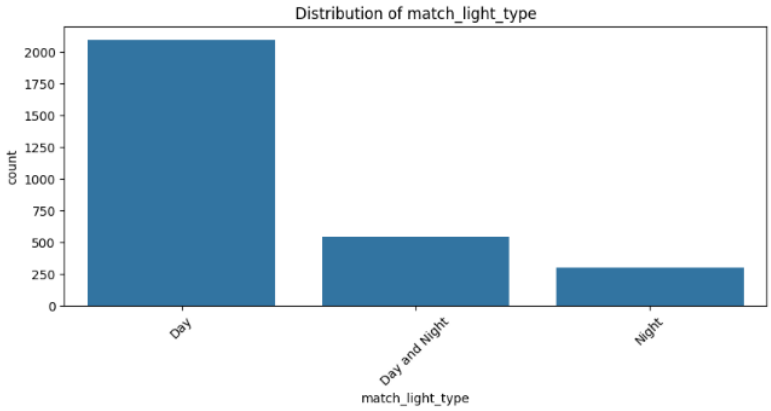
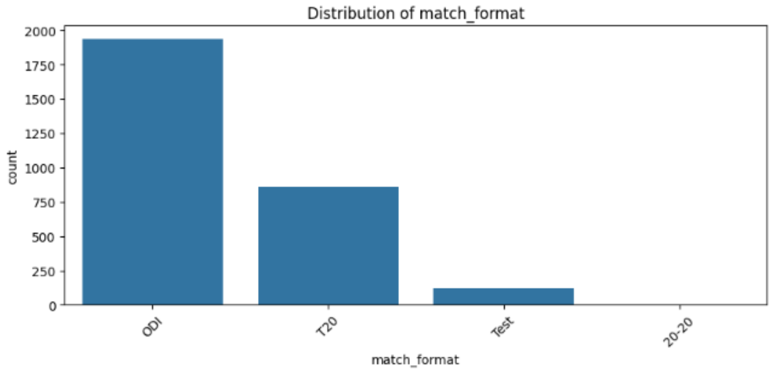
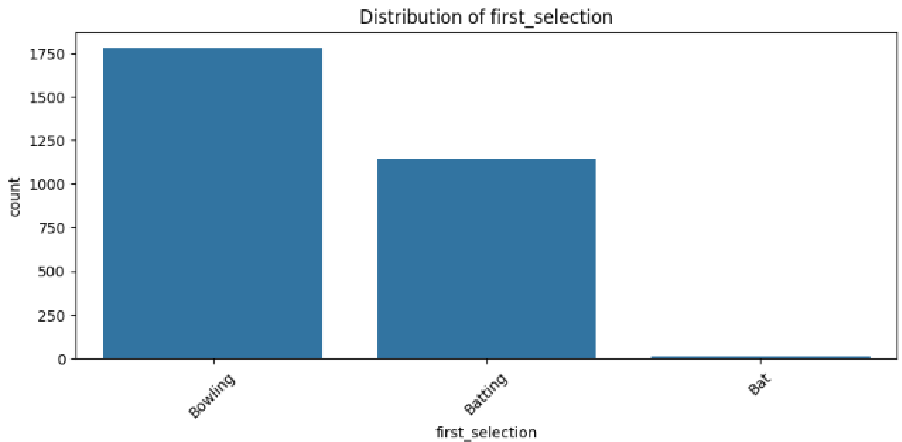

Distribution of bowlers_in_team
Boxplot of bowlers_in_team
Count
1200
1000
800
600
400
200
0
1
2
3
4
5
bowlers_in_team
Distribution of all_rounder_in_team
Boxplot of all_rounder_in_team
1.0
1.5
2.0
2.5
3.0
3.5
4.0
all_rounder_in_team

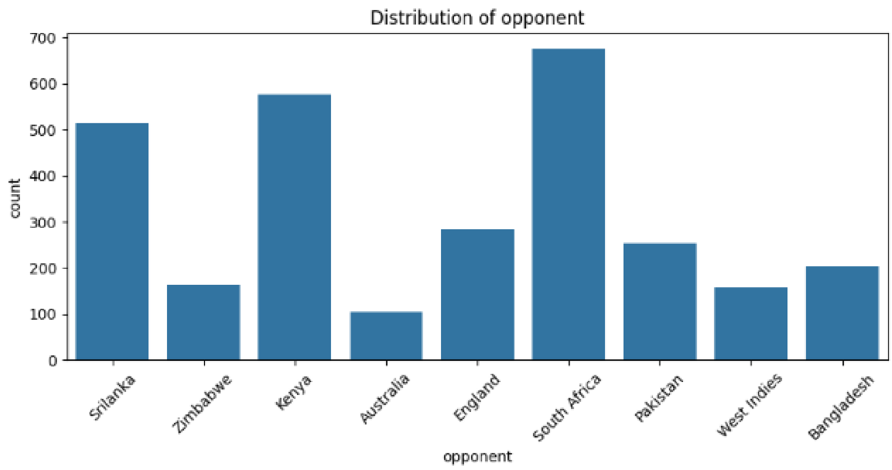
Fig 7

- **Categorical Attributes**









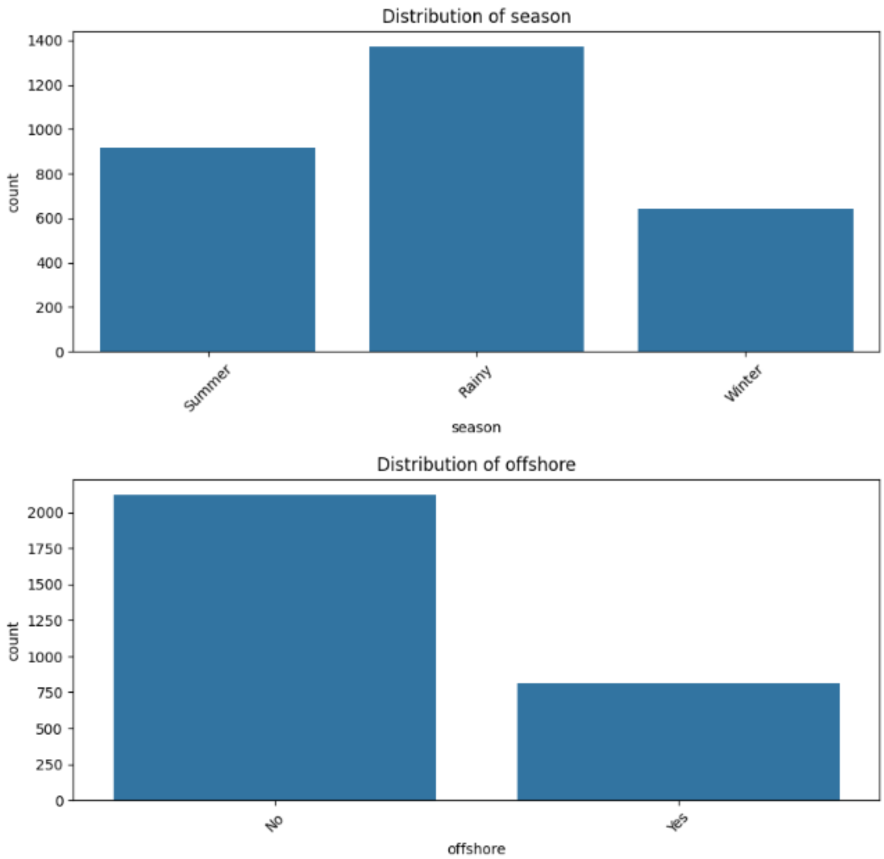

Distribution of season
1400
1200
1000
800
600
400
200
0
count
Summer
Rainy
Winter
season
Distribution of offshore
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
offshore

Fig 8

b) Bivariate analysis (relationship between different variables, correlations)

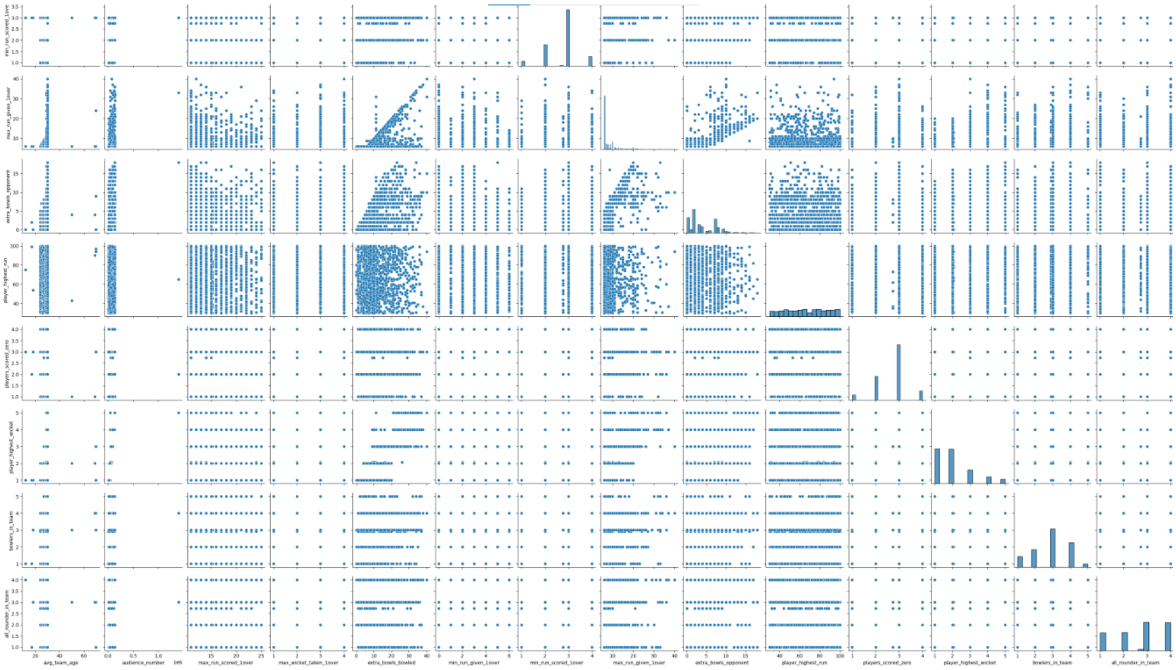
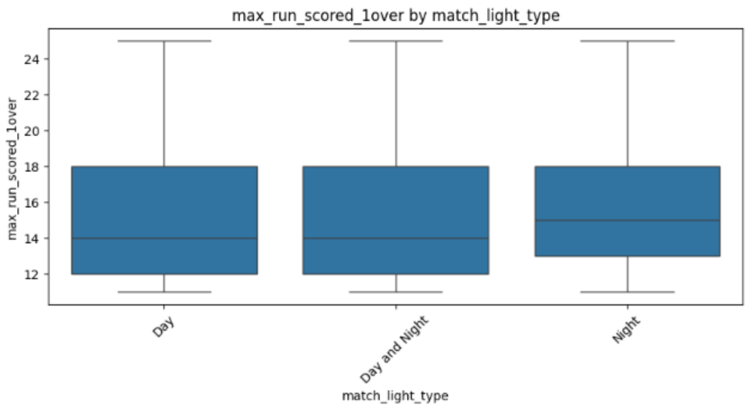
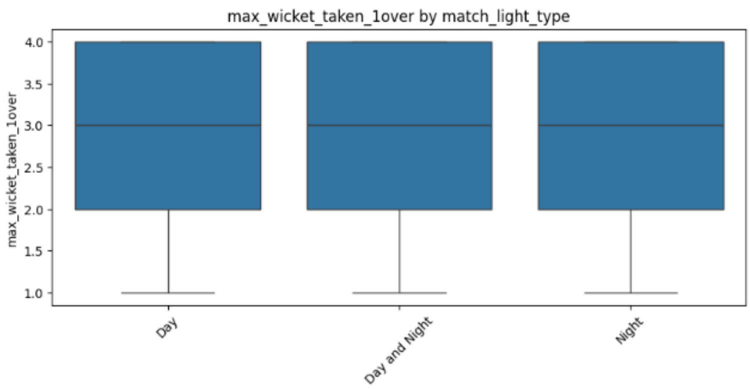
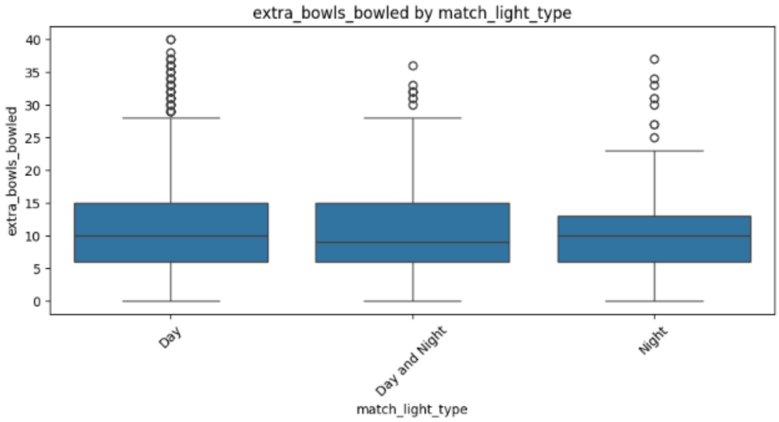
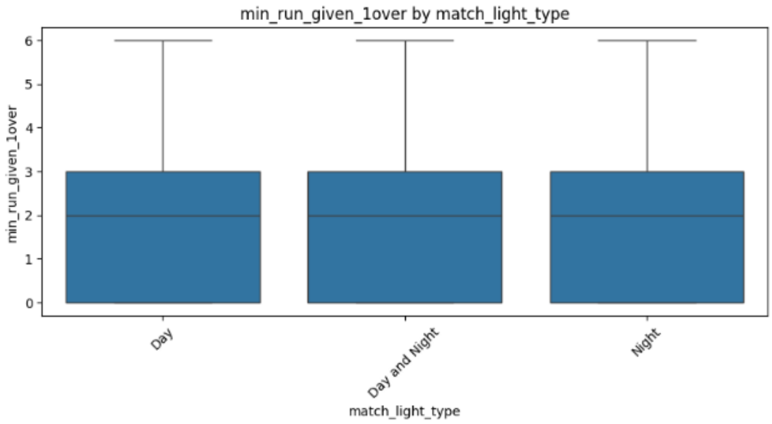


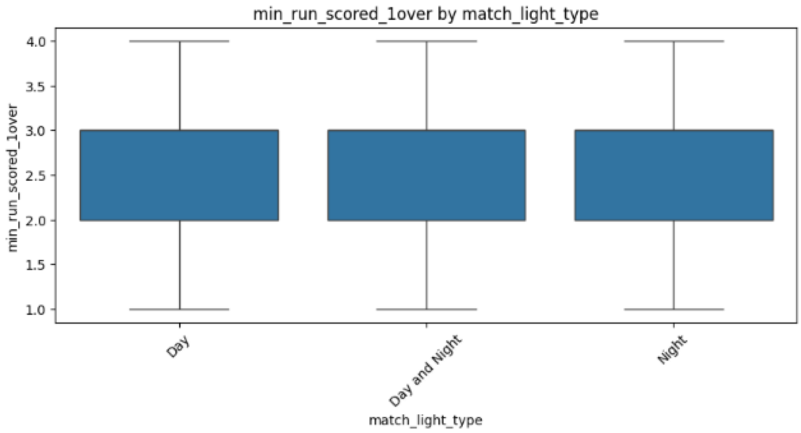
Fig 9

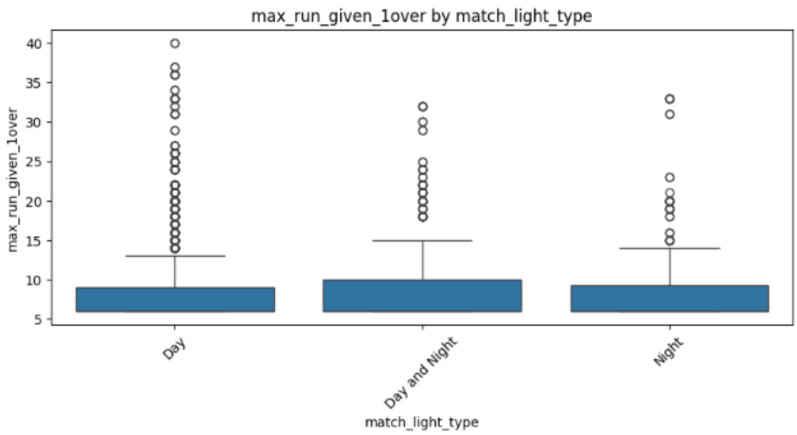


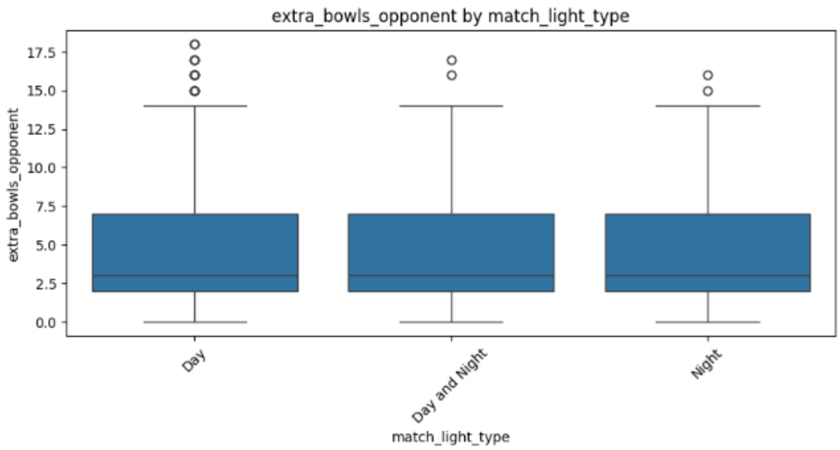


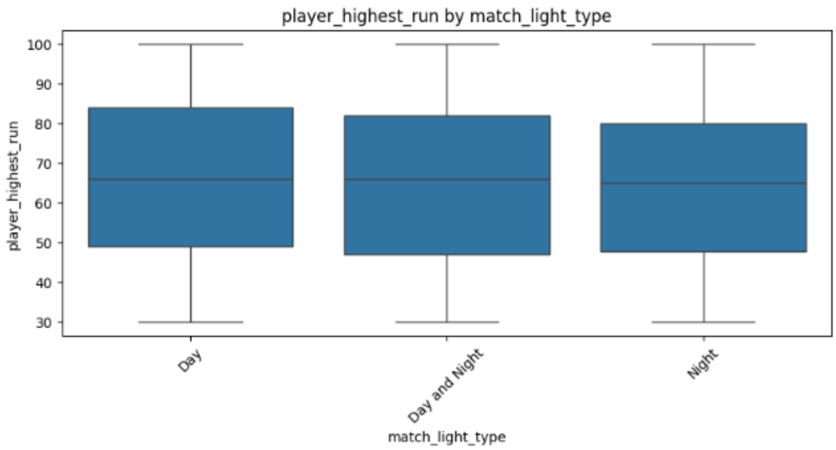

extra_bowls_bowled by match_light_type
extra_bowls_bowled
0
5
10
15
20
25
30
35
40
Day
Day and Night
Night
match_light_type

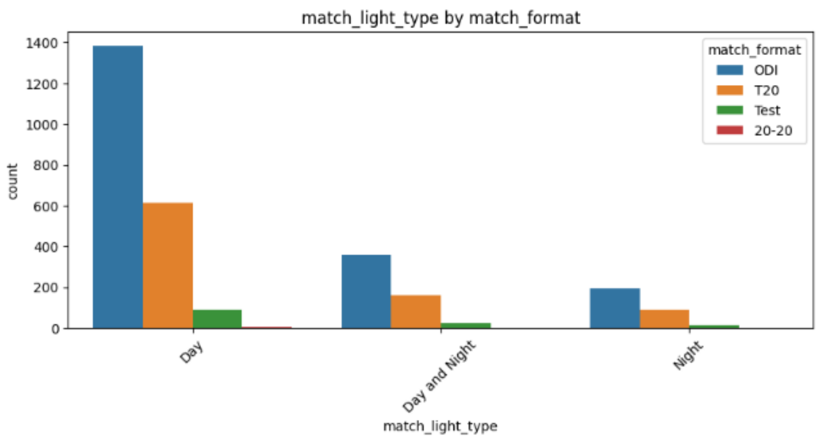

min_run_given_1over by match_light_type
min_run_given_1over
0
1
2
3
4
5
6
Day
Day and Night
Night
match_light_type

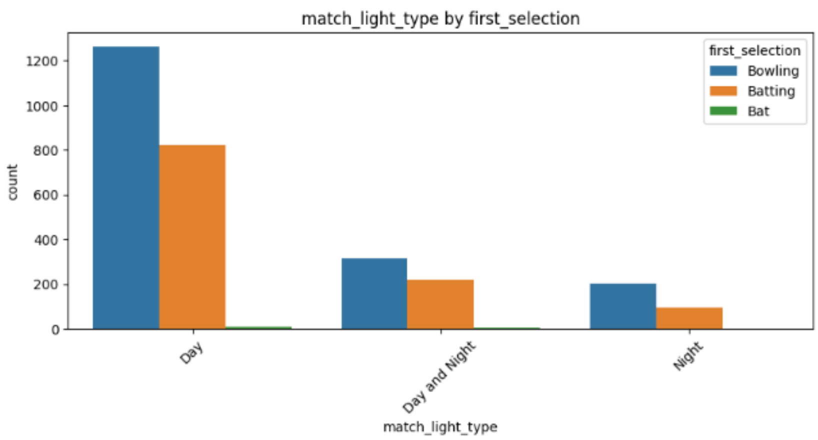

min_run_scored_1over by match_light_type
min_run_scored_1over
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Day
Day and Night
Night
match_light_type

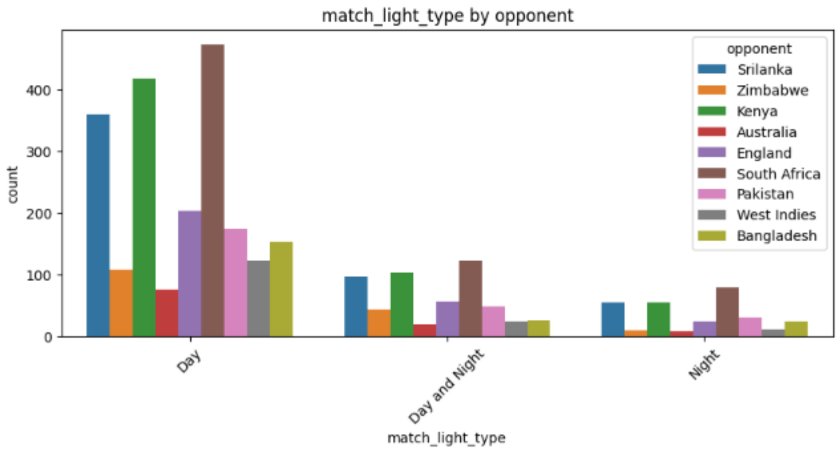

max_run_given_1over by match_light_type
max_run_given_1over
5
10
15
20
25
30
35
40
Day
Day and Night
Night
match_light_type

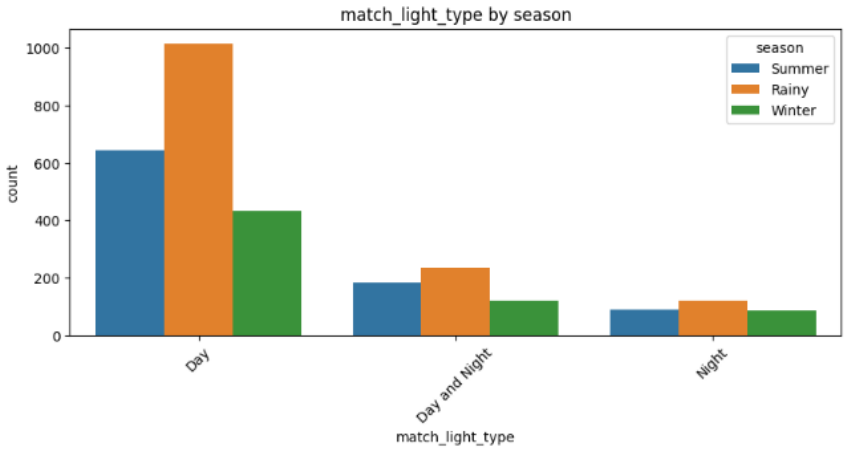

extra_bowls_opponent by match_light_type
extra_bowls_opponent
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Day
Day and Night
Night
match_light_type

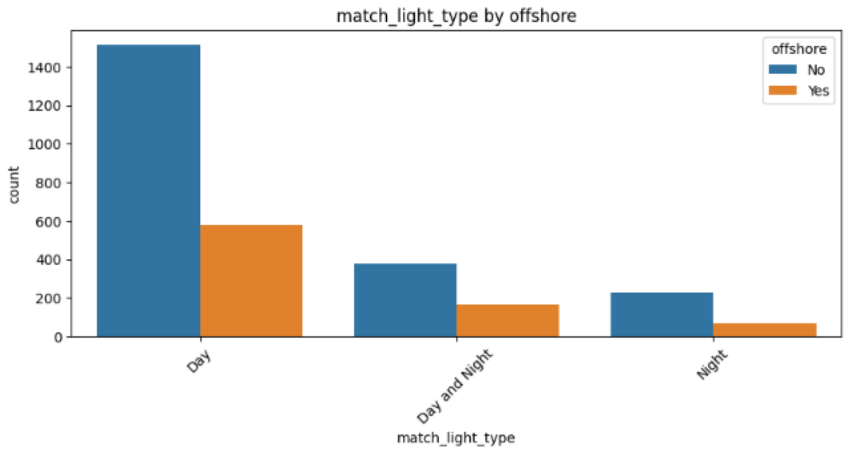

player_highest_run by match_light_type
player_highest_run
100
90
80
70
60
50
40
30
Day
Day and Night
Night
match_light_type

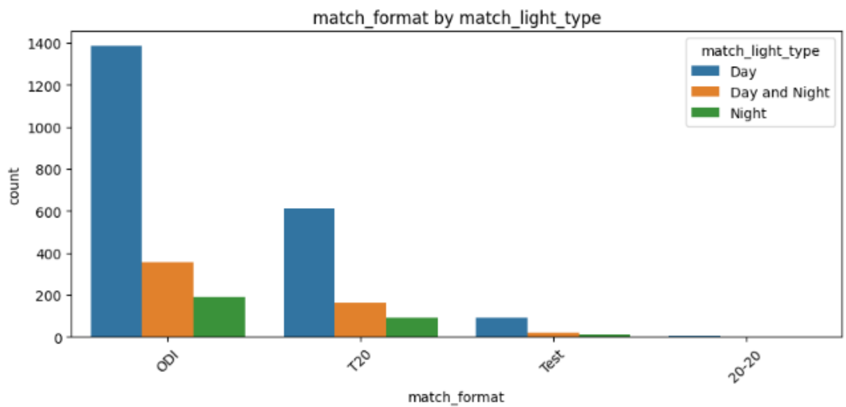

match_light_type by match_format
count
1400
1200
1000
800
600
400
200
0
match_format
ODI
T20
Test
20-20
Day
Day and Night
Night
match_light_type


match_light_type by first_selection
count
1200
1000
800
600
400
200
0
first_selection
Bowling
Batting
Bat
Day
Day and Night
Night
match_light_type


match_light_type by opponent
opponent
Srilanka
Zimbabwe
Kenya
Australia
England
South Africa
Pakistan
West Indies
Bangladesh
count
0
100
200
300
400
Day
Day and Night
Night
match_light_type


match_light_type by season
season
Summer
Rainy
Winter
count
0
200
400
600
800
1000
Day
Day and Night
Night
match_light_type


match_light_type by offshore
offshore
No
Yes
count
0
200
400
600
800
1000
1200
1400
Day
Day and Night
Night
match_light_type


match_format by match_light_type
match_light_type
Day
Day and Night
Night
count
0
200
400
600
800
1000
1200
1400
ODI
T20
Test
20-20
match_format

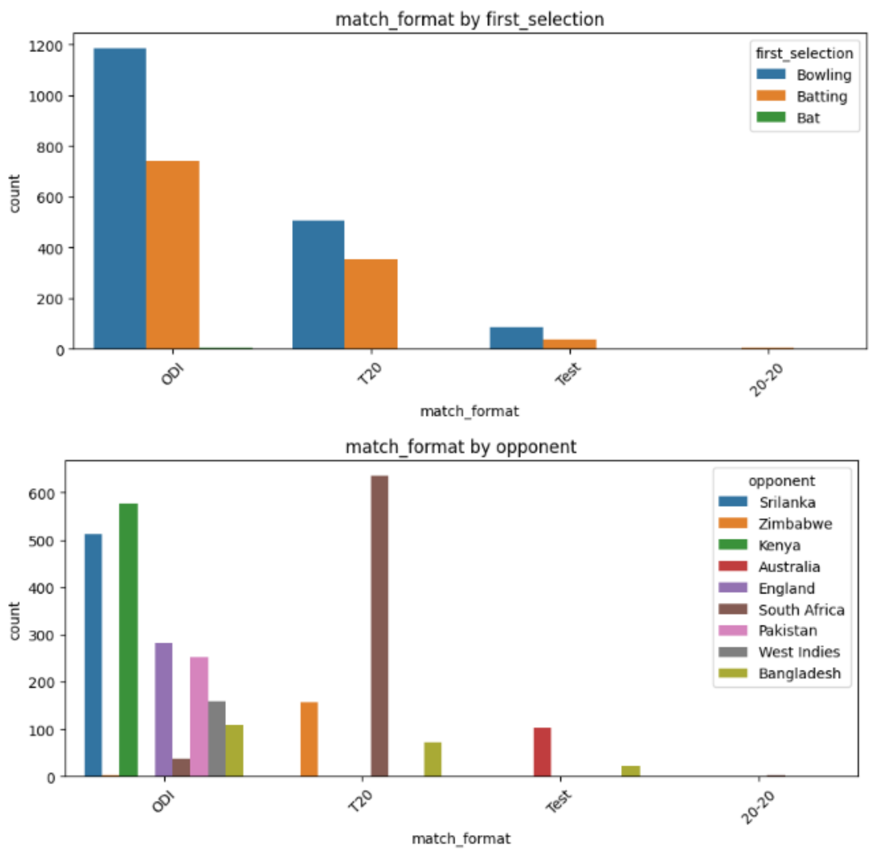


Fig 10

c) Removal of unwanted variables (if applicable)

Not Applicable

d) Missing Value treatment (if applicable)

Missing values treated.

BEFORE:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   game_number            2930 non-null   object 
 1   result                 2930 non-null   object 
 2   avg_team_age           2833 non-null   float64
 3   match_light_type       2878 non-null   object 
 4   match_format           2860 non-null   object 
 5   bowlers_in_team        2848 non-null   float64
 6   wicket_keeper_in_team  2930 non-null   int64  
 7   all_rounder_in_team    2890 non-null   float64
 8   first_selection        2871 non-null   object 
 9   opponent               2894 non-null   object 
 10  season                 2868 non-null   object 
 11  audience_number        2849 non-null   float64
 12  offshore               2866 non-null   object 
 13  max_run_scored_1over   2902 non-null   float64
 14  max_wicket_taken_1over 2930 non-null   int64  
 15  extra_bowls_bowled     2901 non-null   float64
 16  min_run_given_1over    2930 non-null   int64  
 17  min_run_scored_1over   2903 non-null   float64
 18  max_run_given_1over    2896 non-null   float64
 19  extra_bowls_opponent   2930 non-null   int64  
 20  player_highest_run     2902 non-null   float64
 21  players_scored_zero    2930 non-null   object 
 22  player_highest_wicket  2930 non-null   object 
dtypes: float64(9), int64(4), object(10)
memory usage: 526.6+ KB
None
```

Fig 11.1

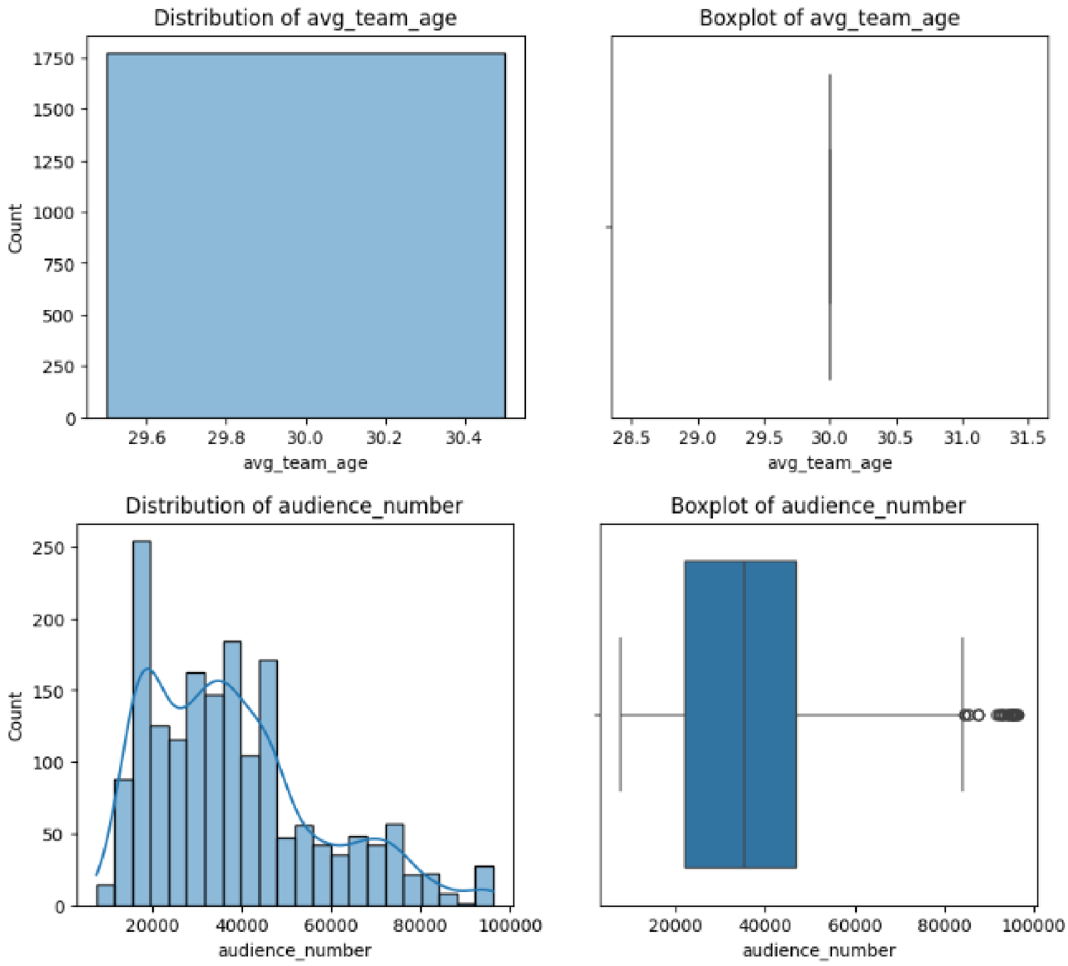
AFTER:

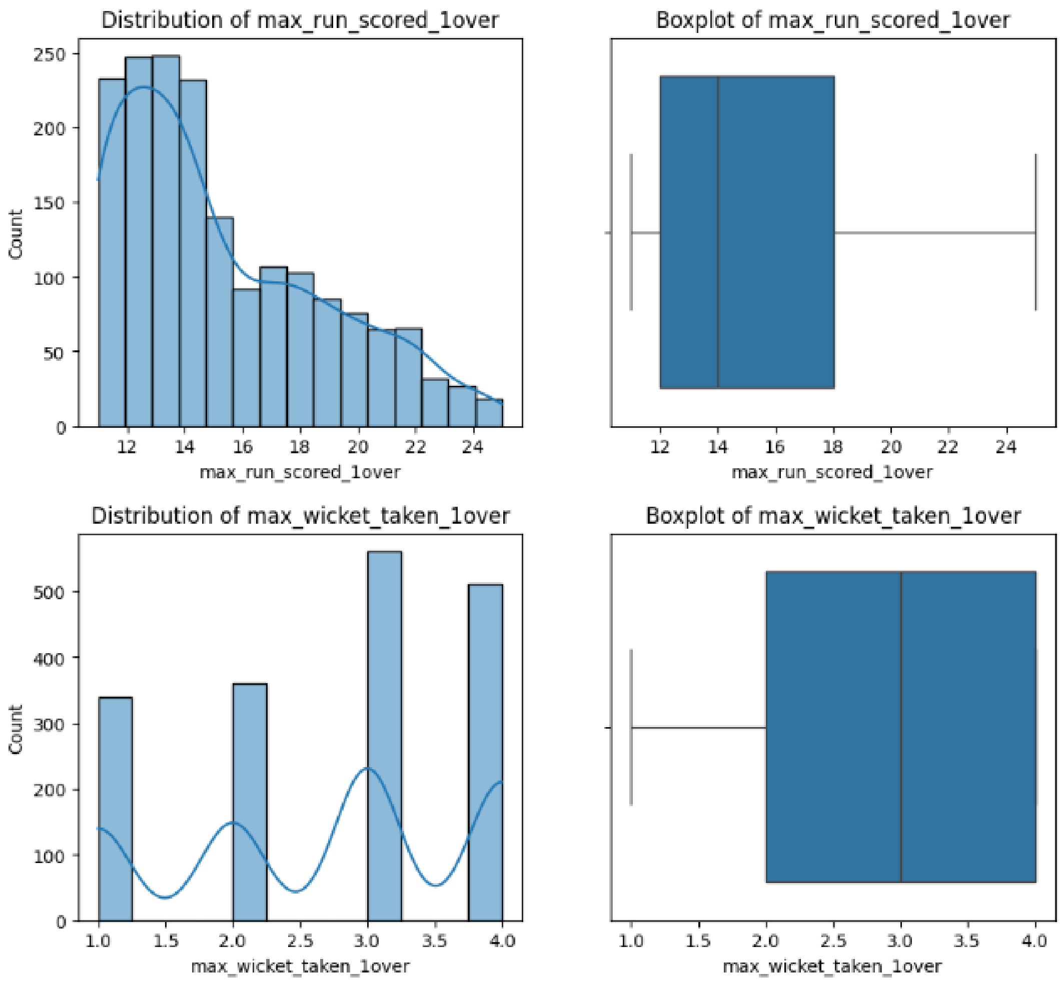
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   game_number            2930 non-null   object 
 1   result                 2930 non-null   object 
 2   avg_team_age           2930 non-null   float64
 3   match_light_type       2930 non-null   object 
 4   match_format           2930 non-null   object 
 5   bowlers_in_team        2930 non-null   float64
 6   wicket_keeper_in_team  2930 non-null   int64  
 7   all_rounder_in_team    2930 non-null   float64
 8   first_selection        2930 non-null   object 
 9   opponent               2930 non-null   object 
 10  season                 2930 non-null   object 
 11  audience_number        2930 non-null   float64
 12  offshore               2930 non-null   object 
 13  max_run_scored_1over   2930 non-null   float64
 14  max_wicket_taken_1over 2930 non-null   int64  
 15  extra_bowls_bowled     2930 non-null   float64
 16  min_run_given_1over    2930 non-null   int64  
 17  min_run_scored_1over   2930 non-null   float64
 18  max_run_given_1over    2930 non-null   float64
 19  extra_bowls_opponent   2930 non-null   int64  
 20  player_highest_run     2930 non-null   float64
 21  players_scored_zero    2930 non-null   float64
 22  player_highest_wicket  2930 non-null   float64
dtypes: float64(11), int64(4), object(8)
memory usage: 526.6+ KB
None
```

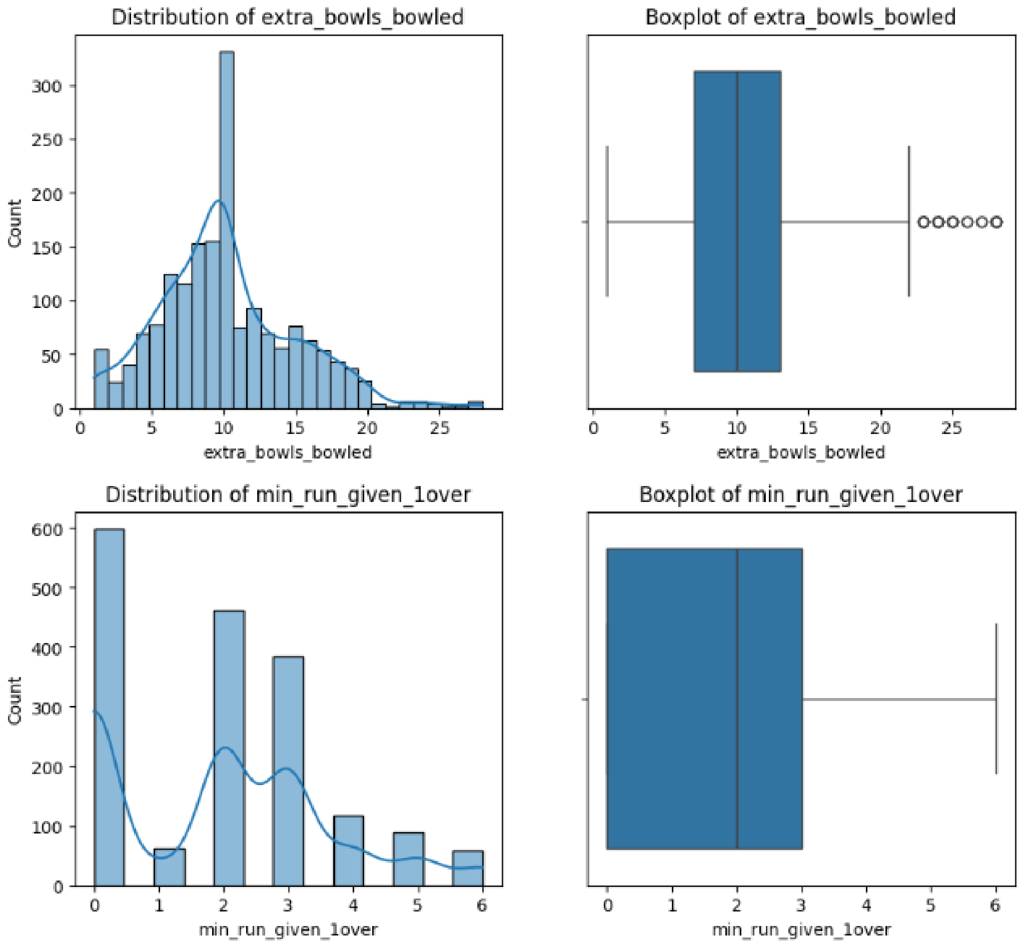
Fig 11.2

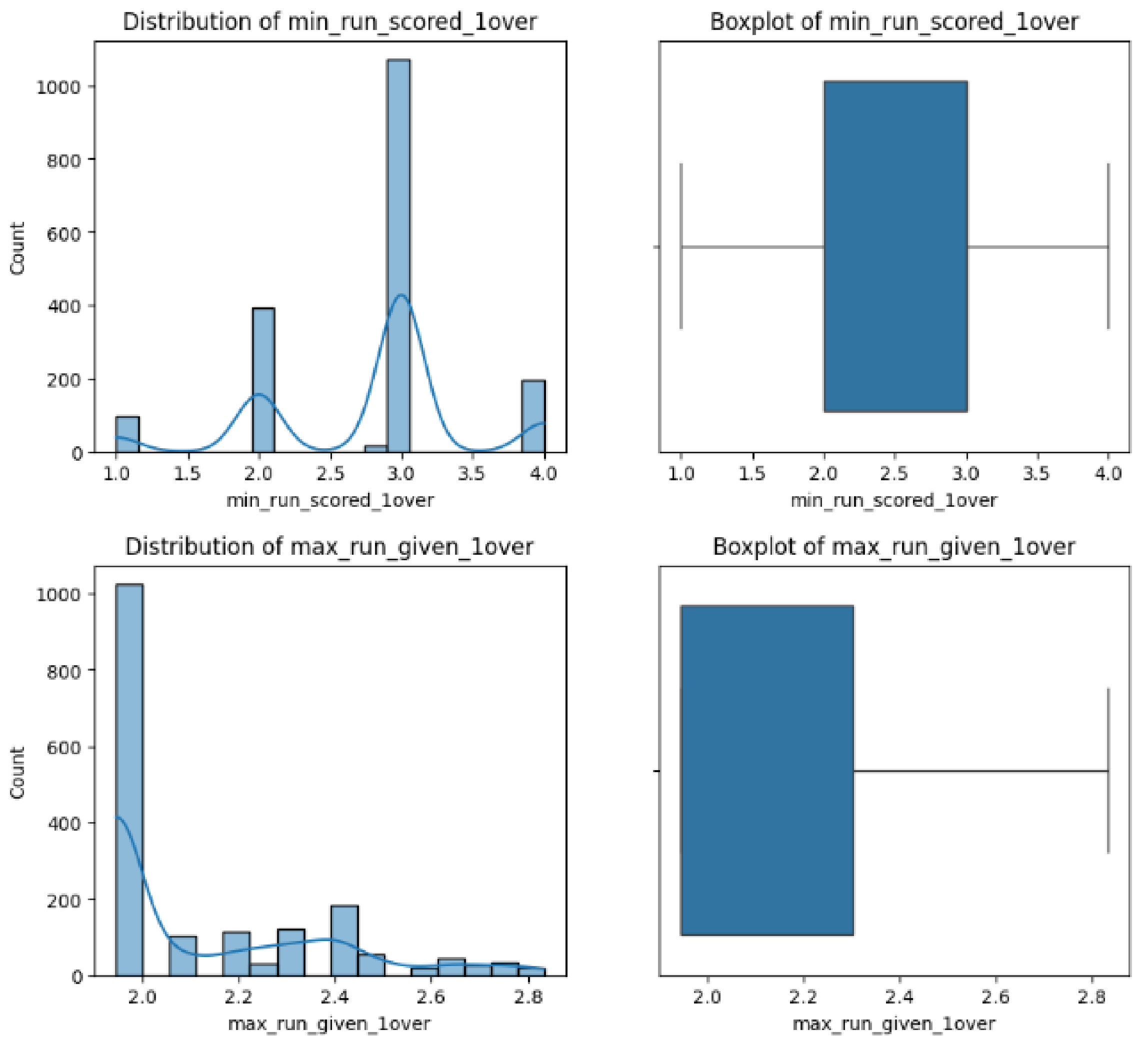
e) Outlier treatment (if required)

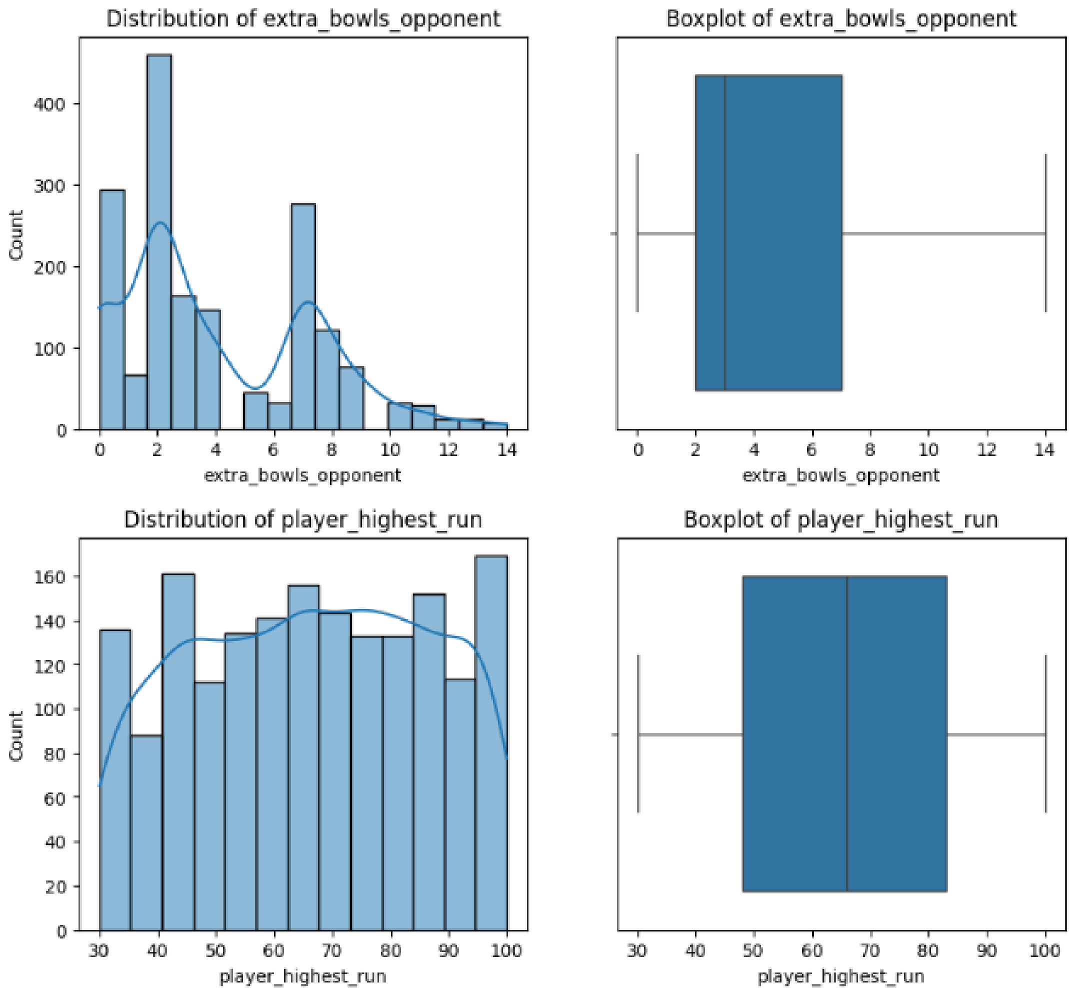
Outlier treated for Continuous attributes.

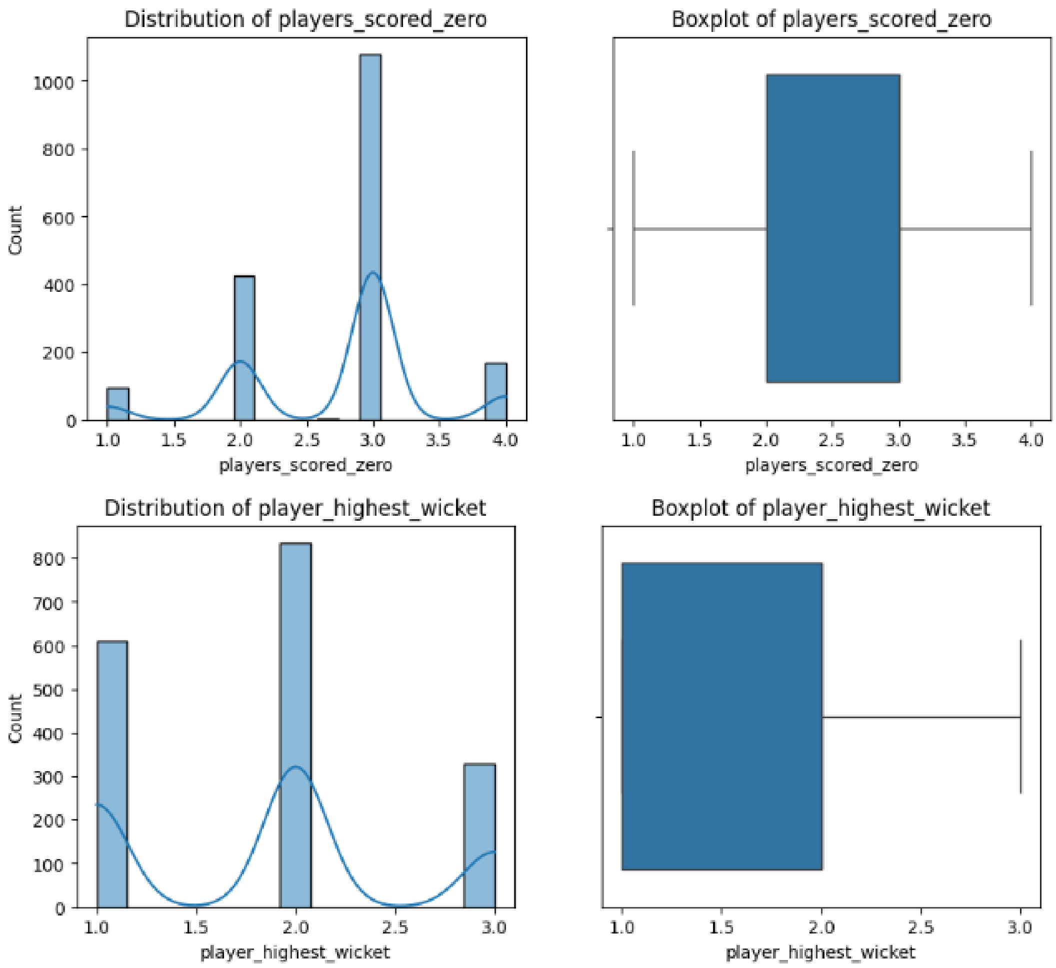



Distribution of max_run_scored_1over
Count
250
200
150
100
50
0
12
14
16
18
20
22
24
max_run_scored_1over
Boxplot of max_run_scored_1over
12
14
16
18
20
22
24
max_run_scored_1over
Distribution of max_wicket_taken_1over
Count
500
400
300
200
100
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
max_wicket_taken_1over
Boxplot of max_wicket_taken_1over
1.0
1.5
2.0
2.5
3.0
3.5
4.0
max_wicket_taken_1over


Distribution of extra_bowls_bowled
Boxplot of extra_bowls_bowled
Count
extra_bowls_bowled
extra_bowls_bowled
Distribution of min_run_given_1over
Boxplot of min_run_given_1over
Count
min_run_given_1over
min_run_given_1over


Distribution of min_run_scored_1over
Count
min_run_scored_1over
Boxplot of min_run_scored_1over
min_run_scored_1over
Distribution of max_run_given_1over
Count
max_run_given_1over
Boxplot of max_run_given_1over
max_run_given_1over


Distribution of extra_bowls_opponent
Count
extra_bowls_opponent
Boxplot of extra_bowls_opponent
extra_bowls_opponent
Distribution of player_highest_run
Count
player_highest_run
Boxplot of player_highest_run
player_highest_run


Distribution of players_scored_zero
Boxplot of players_scored_zero
Count
players_scored_zero
players_scored_zero
Distribution of player_highest_wicket
Boxplot of player_highest_wicket
Count
player_highest_wicket
player_highest_wicket

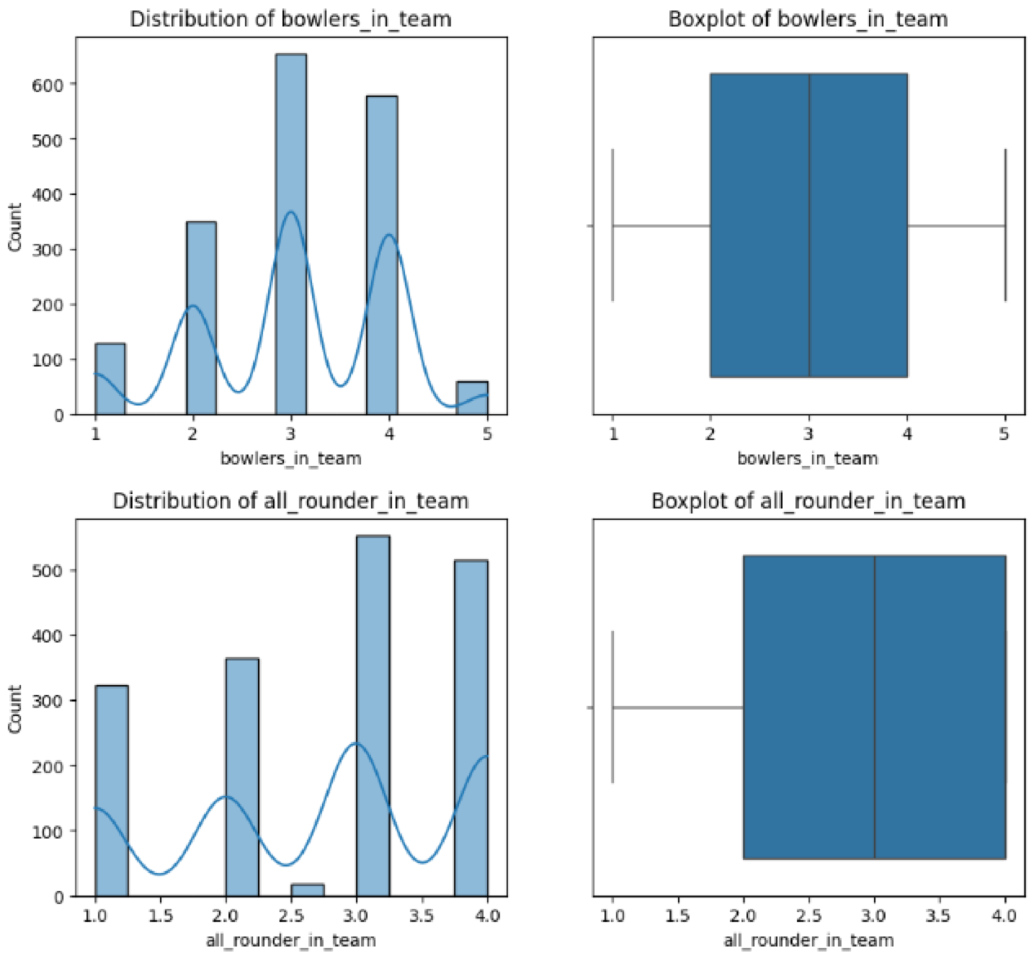


Fig 12

Outlier treated for Continuous vars, but we can still see that there are outliers present for 2 variables. Audience numbers and numbers of extra bowls bowled in an over. This is because some variables inherently have high variability, and what appears to be an outlier might be a natural part of the distribution. For instance, in sports data, exceptional performances (like an unusually high score or an extraordinarily low economy rate) are naturally occurring outliers.

f) Variable transformation (if applicable)

Transformed variables to meet assumptions of modelling techniques and to improve interpretability.

g) Addition of new

Not Required.

4. **Business insights from EDA**
   a) Is the data unbalanced? If so, what can be done? Please explain in the context of the business b) Any business insights using clustering (if applicable) c) Any other business insights

a) **Is the data unbalanced? If so, what can be done? Please explain in the context of the business**
   Yes.

```
result
Win      1516
Loss      255
Name: count, dtype: int64
```

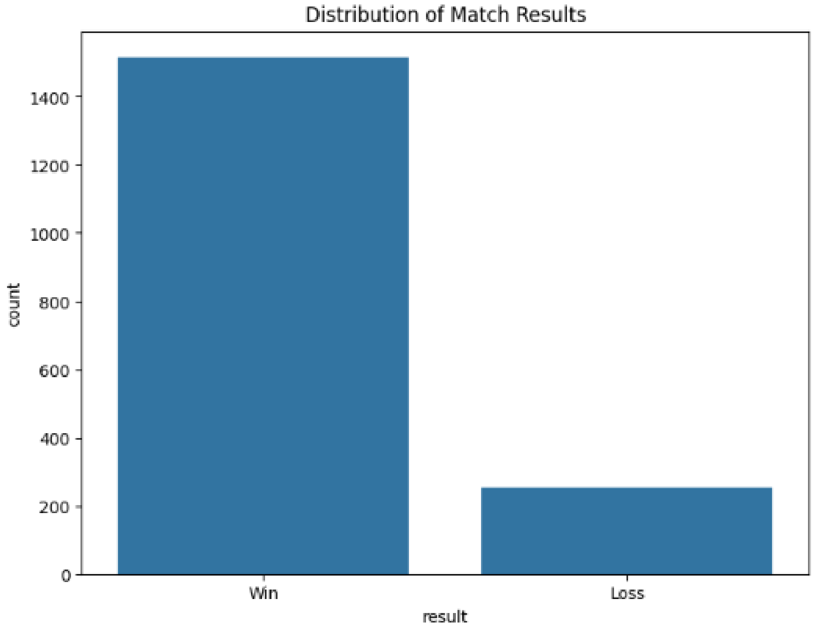


Fig 13

Resampling:

- **Oversampling:** Increase the number of instances in the minority class.
- **Undersampling:** Decrease the number of instances in the majority class.

- **SMOTE (Synthetic Minority Over-sampling Technique):** Generate synthetic examples for the minority class.

Algorithmic Approaches:

- **Adjusting Class Weights:** Modify the algorithm to give more importance to the minority class.
- **Anomaly Detection:** Treat the minority class as anomalies and use anomaly detection techniques.

Ensemble Methods:

- Use ensemble techniques like bagging and boosting that can handle unbalanced datasets better.

b) Any business insights using clustering (if applicable)

Performance Clusters: Identifing clusters where the team performs exceptionally well or poorly and analysing the conditions (e.g., match format, opponent, season) contributing to these performances.

Strategy Development: Tailor strategies for matches based on the identified clusters. For instance, if a cluster shows poor performance in T20s against a particular opponent, specific strategies can be developed for those scenarios.

c) Any other business insights

Player Performance Analysis:

Age and Experience: Analysed the impact of team age and experience on match outcomes. As the team age went high the winning percentage went more and more. Age seems to bring experience to the team for better and accurate performance of the team.

Match Conditions:

Home vs. Away Performance: Win rate is high on home turf whereas the losing rate is same on home turf and away turf.

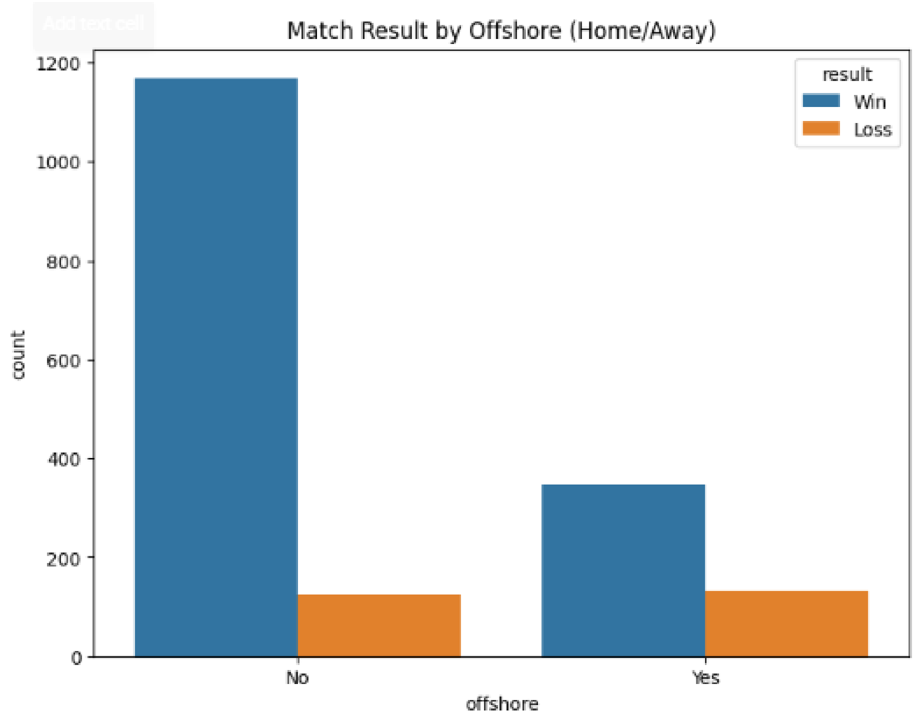


Fig 14

Seasonal Impact: Performance of batsmen are better in Day and Day and Night conditions as compared to Night matches.

Audience Influence:

Audience Size: Analysed that larger audience numbers correlate with better performance, indicating a possible boost from crowd support.

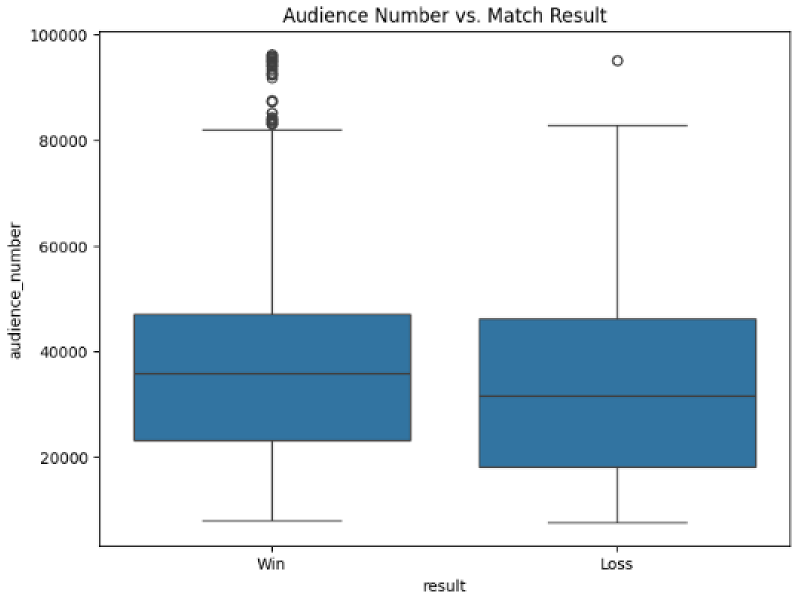


Fig 15

SOLUTION – NOTES 2

1). Model building and interpretation.

a. Build various models (You can choose to build models for either or all descriptive, predictive, or prescriptive purposes)

b. Test your predictive model against the test set using various appropriate performance metrics.

```
Encoded target classes: ['Loss' 'Win']
```

	Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
0	Decision Tree (Unbalanced)	0.929577	0.927199	0.929577	0.928050	0.836494
1	Random Forest (Unbalanced)	0.954930	0.957183	0.954930	0.951315	0.922278
2	AdaBoost (Unbalanced)	0.892958	0.885464	0.892958	0.873174	0.842944
3	XGBoost (Unbalanced)	0.952113	0.951166	0.952113	0.949644	0.916989
4	SVM (Unbalanced)	0.856338	0.733315	0.856338	0.790066	0.536765
5	KNN (Unbalanced)	0.836620	0.787017	0.836620	0.803360	0.694272
6	LDA (Unbalanced)	0.887324	0.878005	0.887324	0.864386	0.832495
7	Naive Bayes (Unbalanced)	0.861972	0.846978	0.861972	0.807948	0.797085
8	Decision Tree (Balanced)	0.921127	0.917186	0.921127	0.918283	0.807082
9	Random Forest (Balanced)	0.943662	0.941907	0.943662	0.940451	0.953722
10	AdaBoost (Balanced)	0.839437	0.838127	0.839437	0.838774	0.821336
11	XGBoost (Balanced)	0.949296	0.947804	0.949296	0.946950	0.917763
12	SVM (Balanced)	0.600000	0.808686	0.600000	0.659692	0.654380
13	KNN (Balanced)	0.704225	0.823070	0.704225	0.744083	0.705528
14	LDA (Balanced)	0.757746	0.856369	0.757746	0.788882	0.827012
15	Naive Bayes (Balanced)	0.591549	0.846693	0.591549	0.650887	0.767608

Fig 16

c. Interpretation of the model(s).

- **Best Models for Unbalanced Data:** The XGBoost model performs the best in terms of both F1-Score and AUC-ROC, closely followed by Random Forest.
- **Best Models for Balanced Data:** Again, XGBoost shows excellent performance with a high F1-Score and decent AUC-ROC, while Random Forest is also performing well.

- **AUC-ROC Comparison:** The AUC-ROC for XGBoost (Unbalanced) is the highest among all models, indicating it has the best ability to distinguish between the classes.

2). Model Tuning and business implication.

a. **Ensemble modelling, wherever applicable**
b. **Any other model tuning measures(if applicable).**

	Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
0	Decision Tree (Unbalanced)	0.929577	0.927199	0.929577	0.928050	0.836494
1	Random Forest (Unbalanced)	0.954930	0.957183	0.954930	0.951315	0.922278
2	AdaBoost (Unbalanced)	0.892958	0.885464	0.892958	0.873174	0.842944
3	XGBoost (Unbalanced)	0.952113	0.951166	0.952113	0.949644	0.916989
4	SVM (Unbalanced)	0.856338	0.733315	0.856338	0.790066	0.536765
5	KNN (Unbalanced)	0.836620	0.787017	0.836620	0.803360	0.694272
6	LDA (Unbalanced)	0.887324	0.878005	0.887324	0.864386	0.832495
7	Naive Bayes (Unbalanced)	0.861972	0.846978	0.861972	0.807948	0.797085
8	Decision Tree (Balanced)	0.921127	0.917186	0.921127	0.918283	0.807082
9	Random Forest (Balanced)	0.943662	0.941907	0.943662	0.940451	0.953722
10	AdaBoost (Balanced)	0.839437	0.838127	0.839437	0.838774	0.821336
11	XGBoost (Balanced)	0.949296	0.947804	0.949296	0.946950	0.917763
12	SVM (Balanced)	0.600000	0.808686	0.600000	0.659692	0.654380
13	KNN (Balanced)	0.704225	0.823070	0.704225	0.744083	0.705528
14	LDA (Balanced)	0.757746	0.856369	0.757746	0.788882	0.827012
15	Naive Bayes (Balanced)	0.591549	0.846693	0.591549	0.650887	0.767608
16	Tuned Random Forest	0.938028	0.935997	0.938028	0.933796	0.950690
17	Tuned XGBoost	0.952113	0.951166	0.952113	0.949644	0.923568
18	Ensemble (Tuned RF + XGB)	0.949296	0.948300	0.949296	0.946406	0.949239

Fig 17

c. **Interpretation of the most optimum model and its implication on the business**

Ensemble Model (Tuned RF + XGB):

- **High Accuracy and AUC-ROC:** The ensemble model combining Tuned Random Forest and XGBoost has the highest accuracy and AUC-ROC, indicating it is the most reliable model for distinguishing between match outcomes.

- **Balanced Performance:** With a high precision, recall, and F1-score, the ensemble model provides a balanced performance, reducing the chances of both false positives and false negatives.
- **Generalization:** The ensemble approach typically improves generalization, making it effective for a wider range of scenarios and reducing overfitting.

Tuned XGBoost (Balanced):

- **High Predictive Power:** Tuned XGBoost also shows excellent performance metrics, making it a robust model for predictive purposes.
- **Feature Importance:** XGBoost provides insights into feature importance, which can be critical for understanding the factors affecting match outcomes.

Business Implications

Strategic Planning:

- **Data-Driven Decisions:** By implementing these models, team management can make data-driven decisions on player selection, match strategies, and training focus.
- **Opponent Analysis:** The models can help analyse the strengths and weaknesses of opponents, allowing the team to prepare targeted strategies.
- **Match Conditions:** Insights into how different conditions (e.g., day/night matches, home/away games) affect performance can guide preparation and tactics.

Performance Monitoring:

- **Real-Time Predictions:** These models can be integrated into a real-time match prediction system, providing up-to-date insights, and allowing for adjustments during matches.
- **Continuous Improvement:** By continuously updating the models with new data, the team can ensure the predictions remain accurate and relevant.

Training and Development:

- **Focus Areas:** Identifying key features that influence match outcomes can help direct training efforts towards specific skills or areas.
- **Player Development:** Understanding individual player performance metrics can aid in personalized training and development plans.