# CAPSTONE PROJECT - WP_SP_01

**Aniket Ganguly**

**PGPDSBA – JULY 2023**

## Contents:

# SOLUTION

1. **INTRODUCTION**

**a) AIM/OBJECTIVE**

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) 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. **EDA AND BUSINESS IMPLICATIONS**
   a) **DATA DESCRIPTION**

- **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 of 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

```
<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

```
<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

## b) Univariate analysis

**Purpose:** To understand the distribution and central tendency of each individual variable.

**Analysis:**

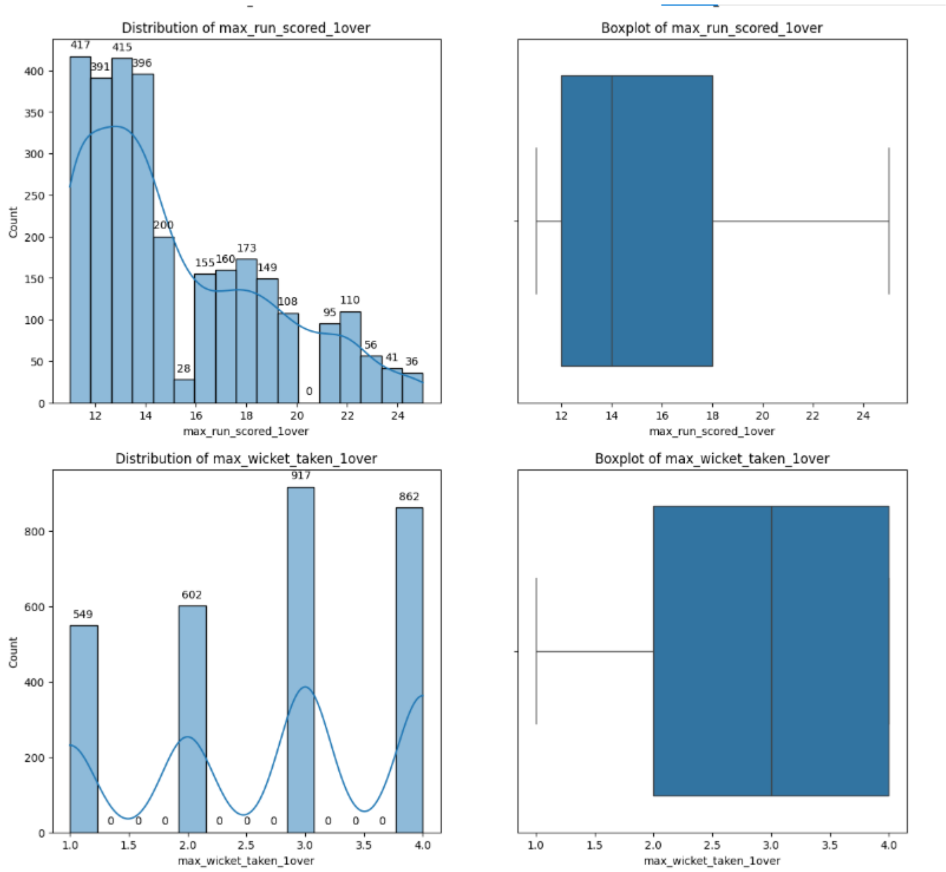
- **Distribution of Continuous Variables:** Histograms and box plots were used to visualize the distribution of continuous variables like Avg_team_Age, Max_run_scored_1over, etc.
- **Distribution of Categorical Variables:** Bar Graphs were used to visualize the frequency distribution of categorical variables like Match_light_type, Match_format, Result, etc.

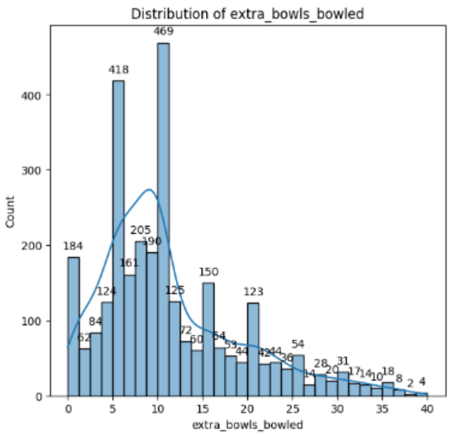
**Business Impact:**

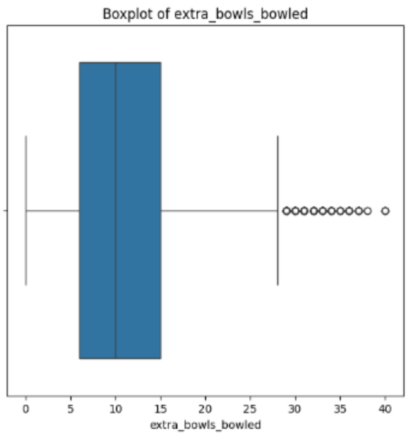
- **Team Composition:** Understanding the distribution of variables such as Avg_team_Age can help in selecting an optimal mix of experience and youth in the team.

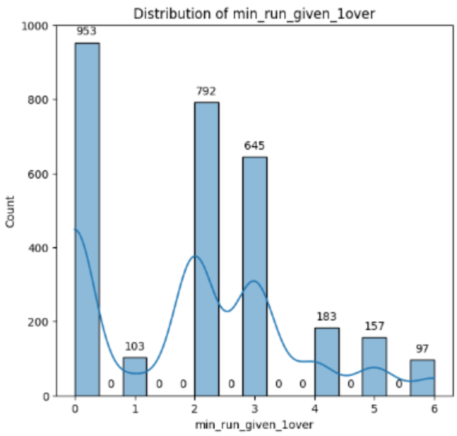
- **Match Preparation:** Frequency distribution of Match_format and Match_light_type helps in preparing for specific match conditions and formats.

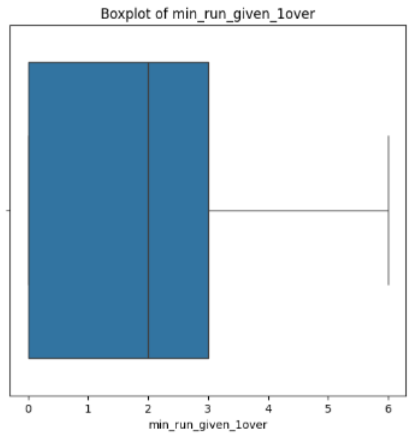
- **Continuous Attributes**

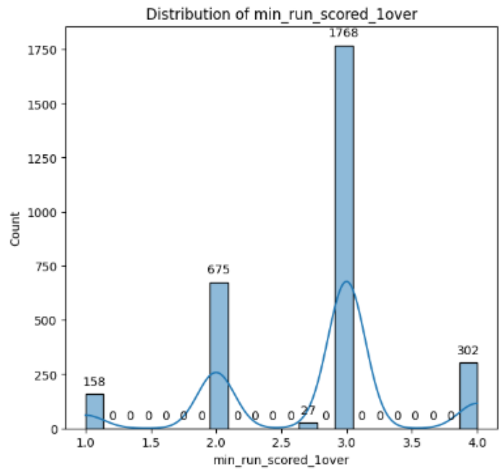


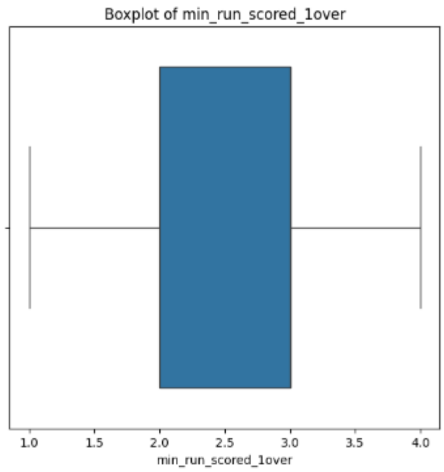

Distribution of extra_bowls_bowled
Count
extra_bowls_bowled
469
418
184
205
190
161
150
124
125
123
84
62
72
60
54
54

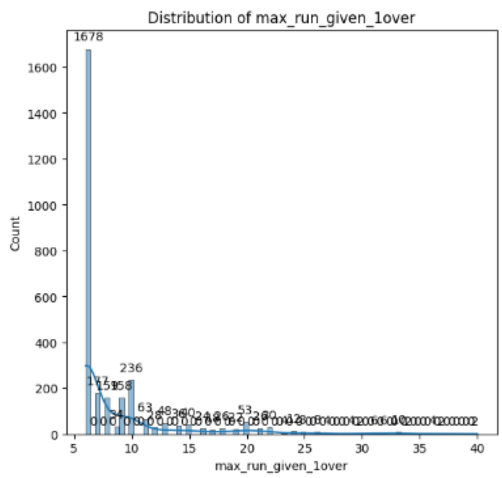

Boxplot of extra_bowls_bowled
extra_bowls_bowled

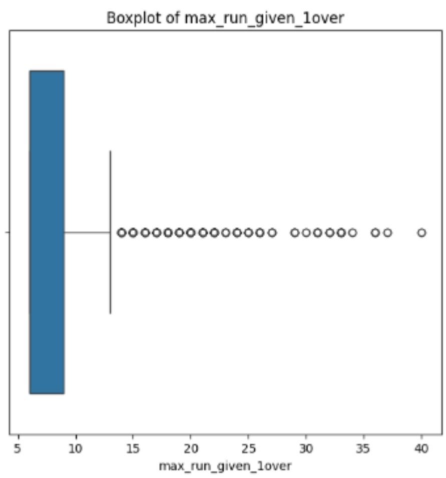

Distribution of min_run_given_1over
Count
min_run_given_1over
953
103
792
645
183
157
97

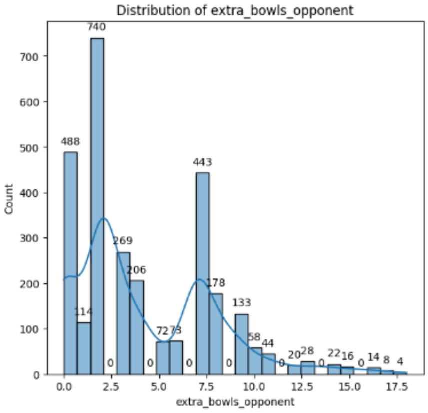

Boxplot of min_run_given_1over
min_run_given_1over

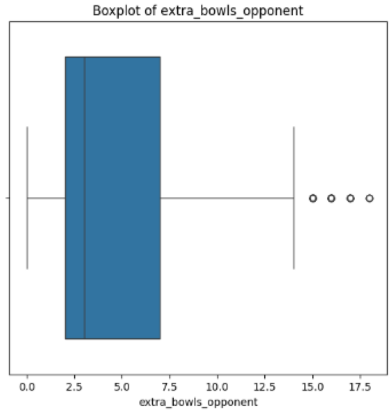

Distribution of min_run_scored_1over
Count
min_run_scored_1over
1768
675
302
158
27

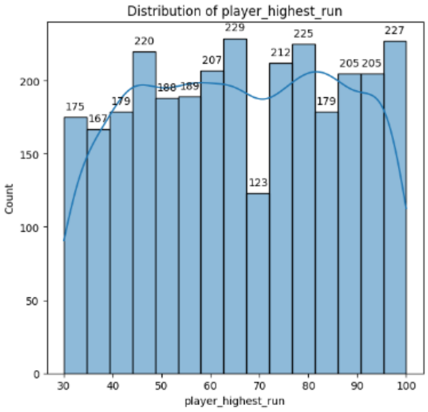

Boxplot of min_run_scored_1over
min_run_scored_1over

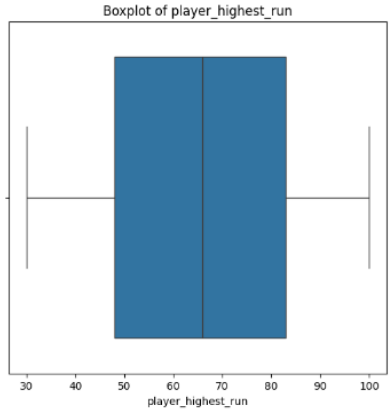

Distribution of max_run_given_1over
Count
max_run_given_1over
1678
236

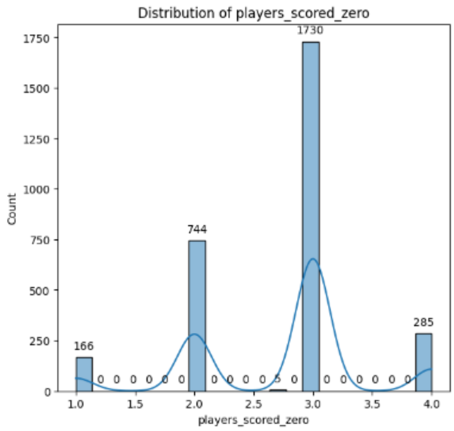

Boxplot of max_run_given_1over
max_run_given_1over

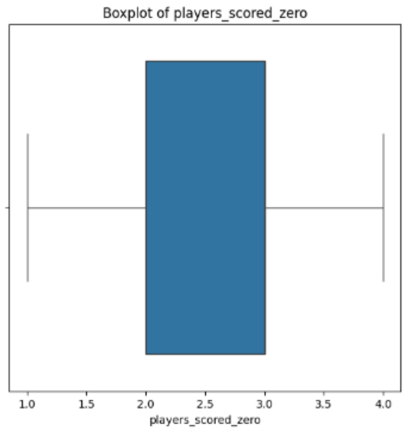

Distribution of extra_bowls_opponent
Count
extra_bowls_opponent

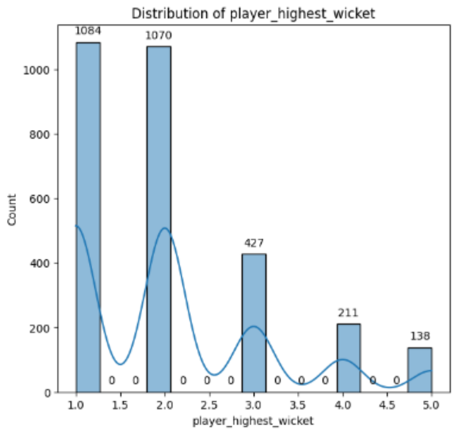

Boxplot of extra_bowls_opponent
extra_bowls_opponent

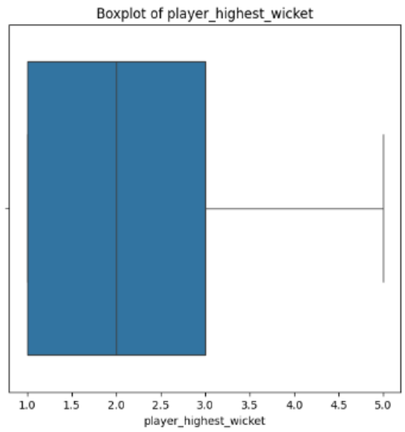

Distribution of player_highest_run
Count
player_highest_run


Boxplot of player_highest_run
player_highest_run


Distribution of players_scored_zero
Count
players_scored_zero


Boxplot of players_scored_zero
players_scored_zero


Distribution of player_highest_wicket
Count
player_highest_wicket


Boxplot of player_highest_wicket
player_highest_wicket

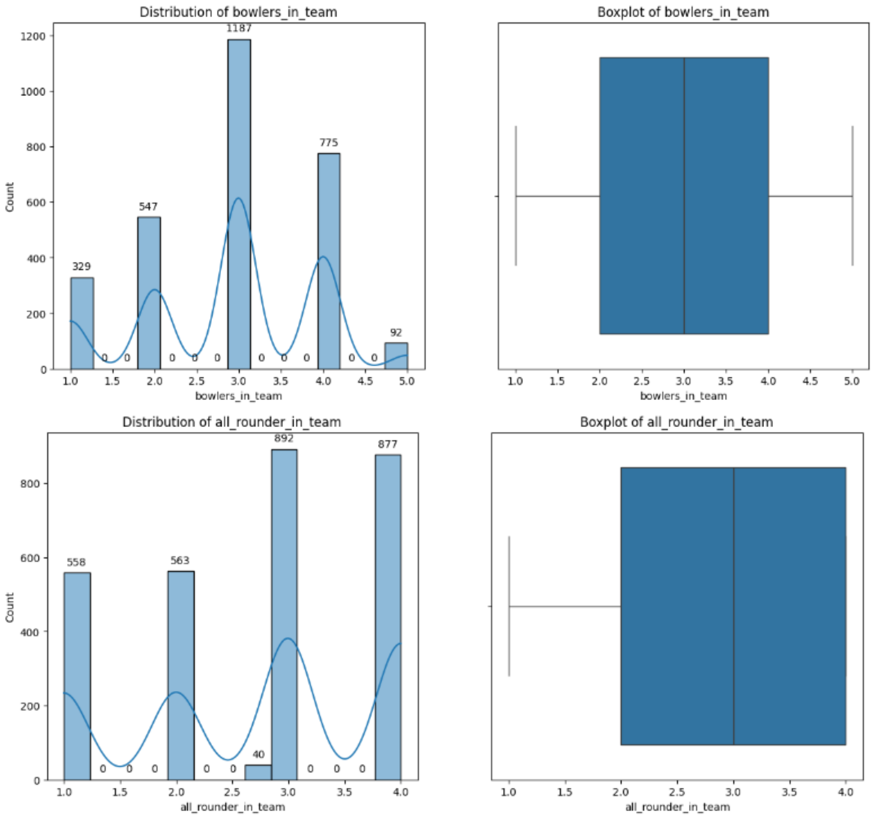
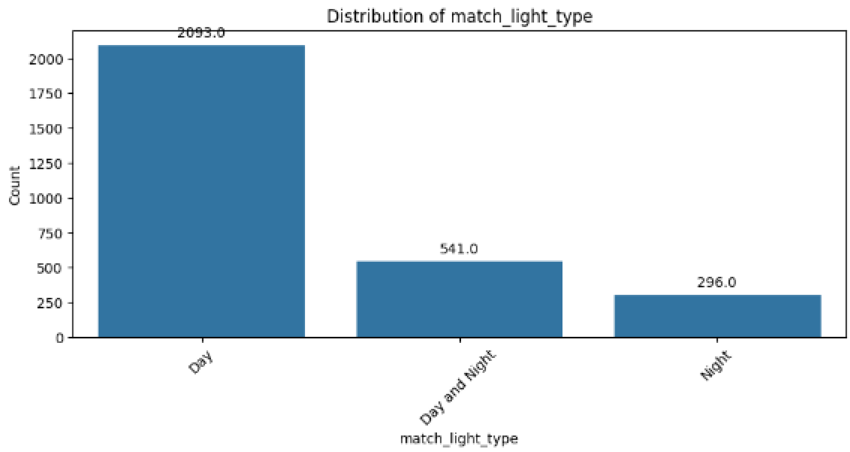
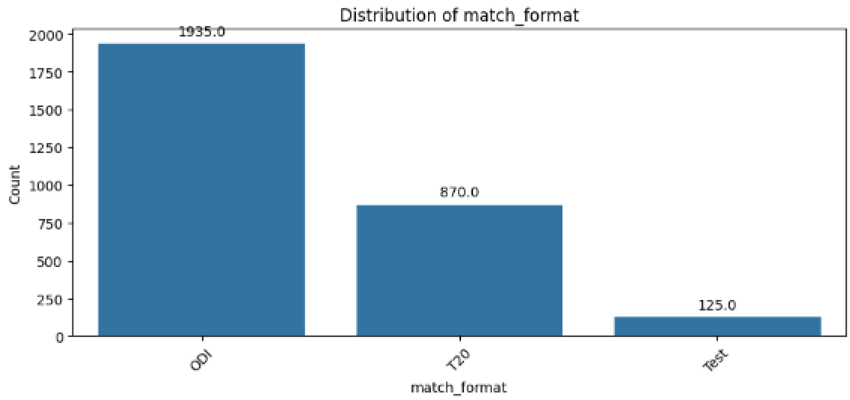
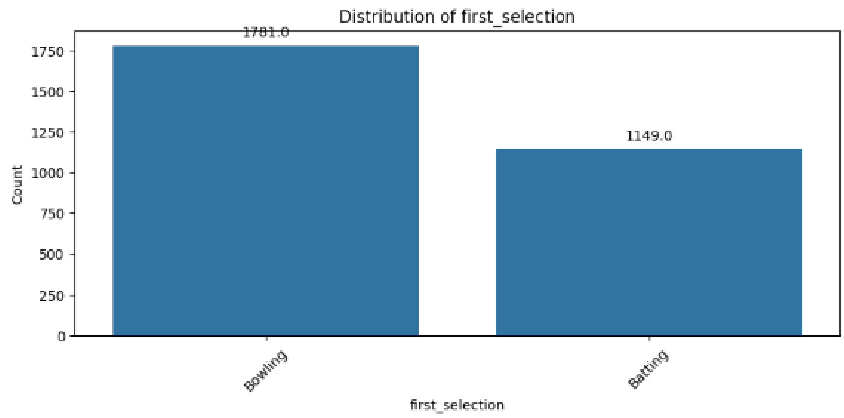


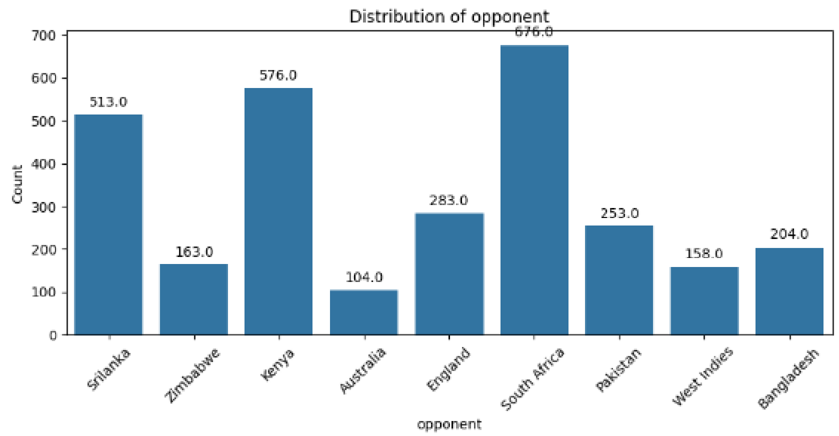

Fig 7

- **Categorical Attributes**


Distribution of match_light_type
2093.0
541.0
296.0
Count
match_light_type
Day
Day and Night
Night


Distribution of match_format
1935.0
870.0
125.0
Count
match_format
ODI
T20
Test


Distribution of first_selection
1781.0
1149.0
Count
first_selection
Bowling
Batting


Distribution of opponent
513.0
163.0
576.0
104.0
283.0
676.0
253.0
158.0
204.0
Count
opponent
Srilanka
Zimbabwe
Kenya
Australia
England
South Africa
Pakistan
West Indies
Bangladesh

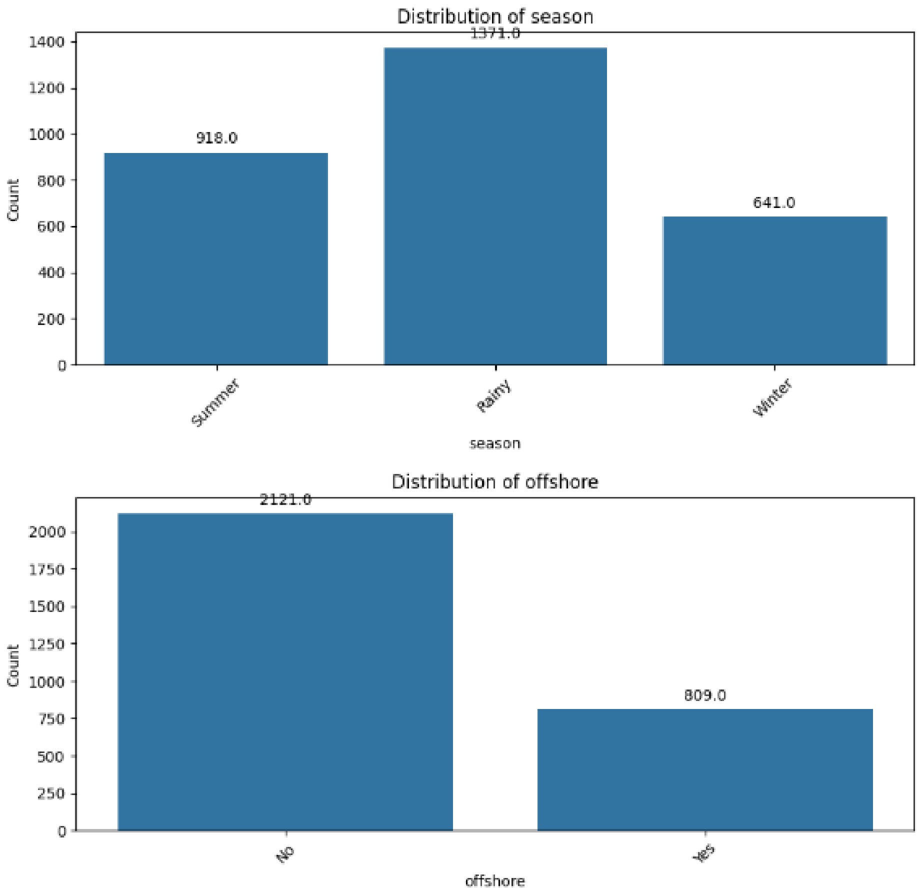



Fig 8

### c) Multivariate analysis

**Purpose:** To understand the relationship among multiple variables simultaneously.

**Analysis:**

- **Correlation Heatmap:** Shows the correlation between continuous variables and their relationship with Result.
- **Box Plots for Continuous Variables against Result:** Provides a detailed view of how each continuous variable varies with match outcomes.

**Business Impact:**

- **Predictive Modeling:** Correlation analysis helps identify key predictors of match outcomes, aiding in the development of predictive models for match results.
- **Strategy Optimization:** Multi-variate analysis reveals complex interactions between variables, helping to optimize team composition and match strategies.

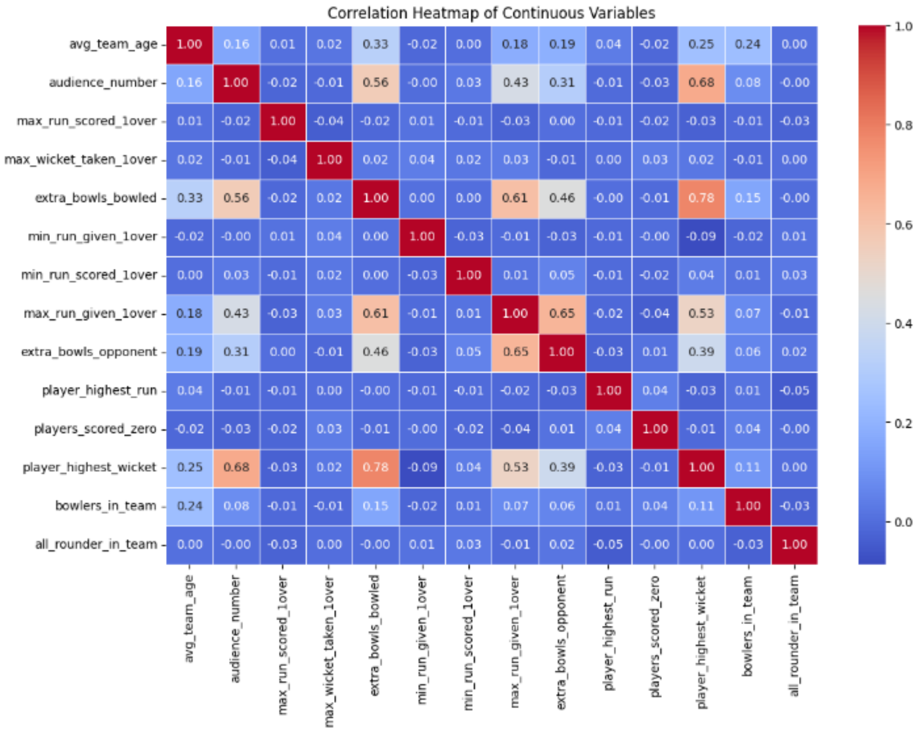
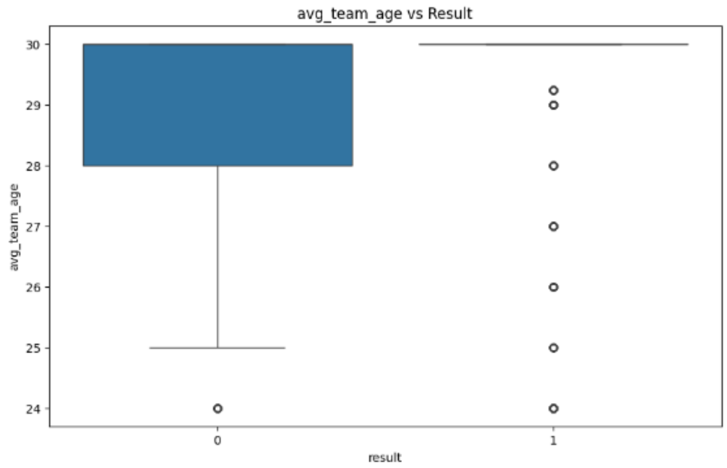
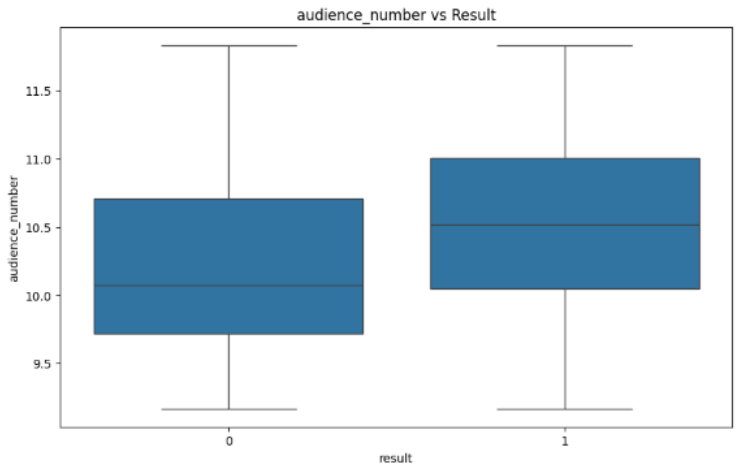
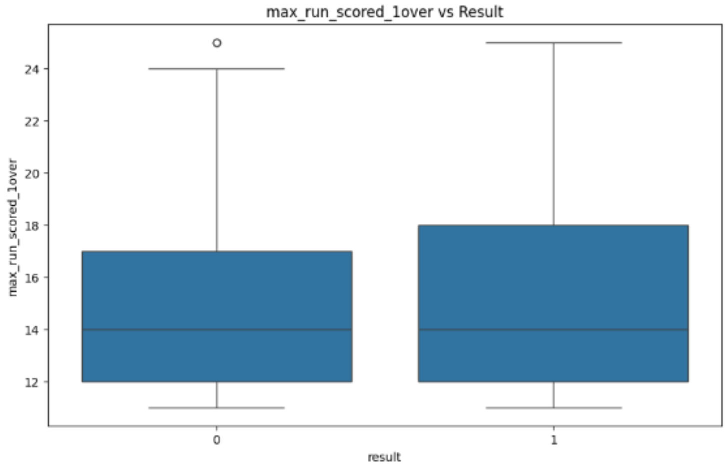
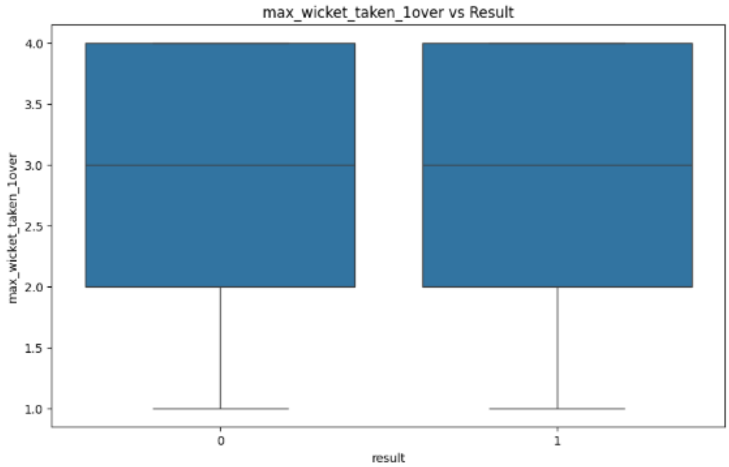

Correlation Heatmap of Continuous Variables
avg_team_age
audience_number
max_run_scored_1over
max_wicket_taken_1over
extra_bowls_bowled
min_run_given_1over
min_run_scored_1over
max_run_given_1over
extra_bowls_opponent
player_highest_run
players_scored_zero
player_highest_wicket
bowlers_in_team
all_rounder_in_team
1.0
0.8
0.6
0.4
0.2
0.0

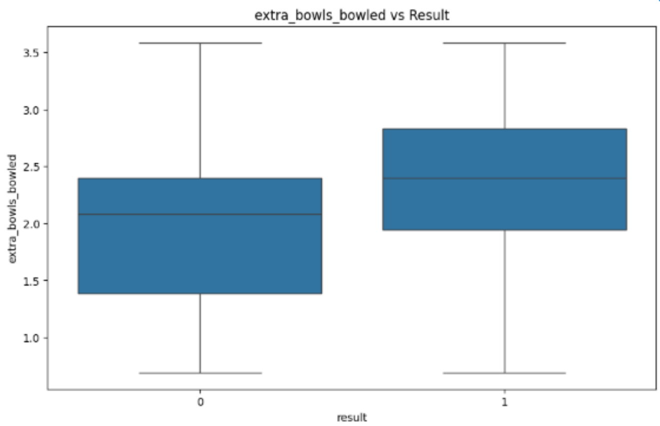
Fig 9.1

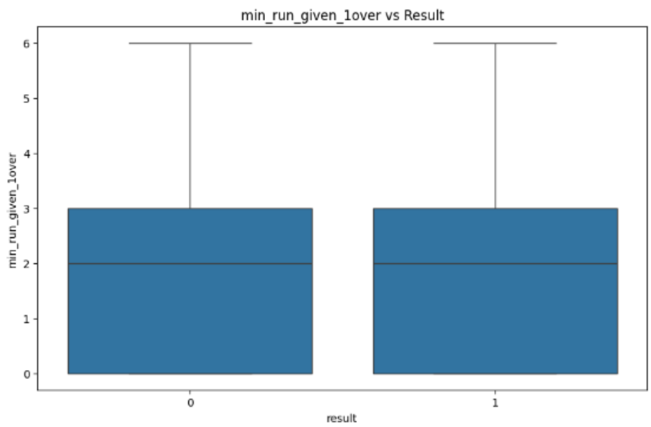

avg_team_age vs Result
30
29
28
27
26
25
24
avg_team_age
0
1
result

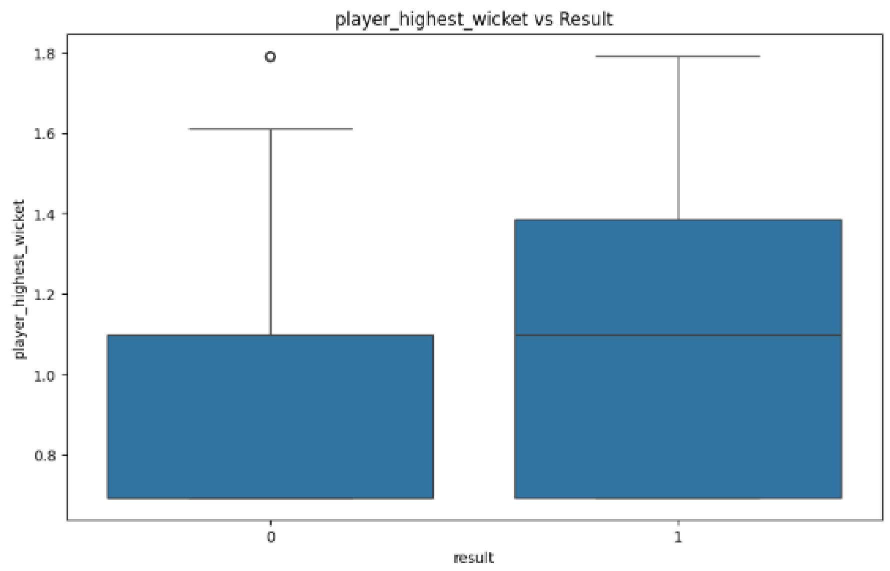

audience_number vs Result
11.5
11.0
10.5
10.0
9.5
audience_number
0
1
result


max_run_scored_1over vs Result
max_run_scored_1over
result
24
22
20
18
16
14
12
0
1


max_wicket_taken_1over vs Result
max_wicket_taken_1over
result
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1


extra_bowls_bowled vs Result
extra_bowls_bowled
result
3.5
3.0
2.5
2.0
1.5
1.0
0
1


min_run_given_1over vs Result
min_run_given_1over
result
6
5
4
3
2
1
0
0
1



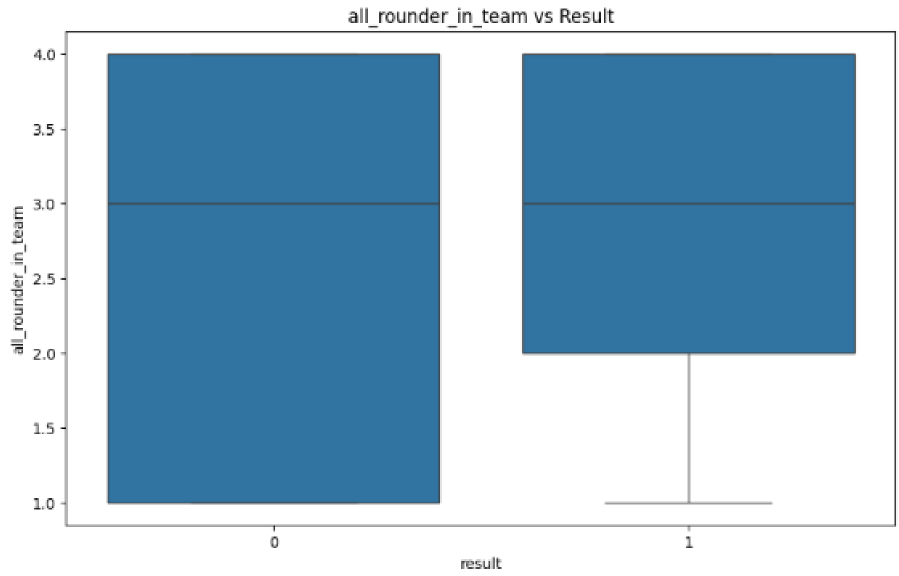




Fig 9.2

**d) Bi-variate Analysis**

**Purpose**: To understand the relationship between two variables.

**Analysis**:

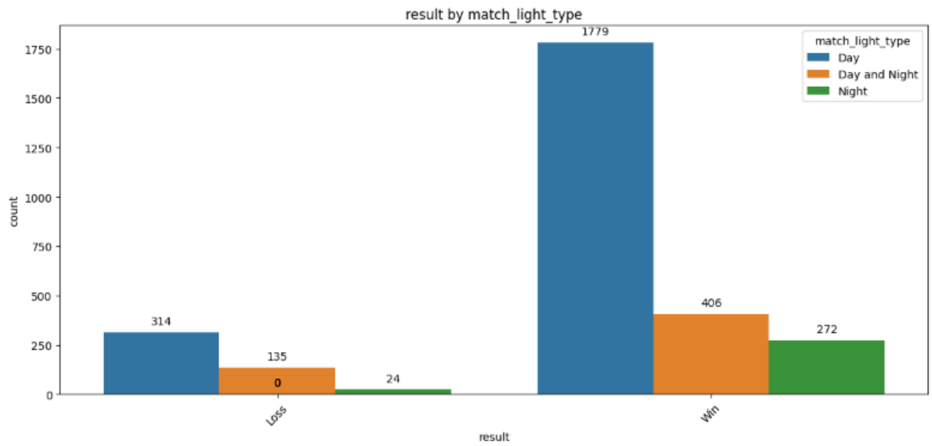
- **Continuous vs Categorical**: Box plots were used to compare continuous variables against the target variable Result.
- **Categorical vs Categorical**: Count plots with hue were used to compare two categorical variables and their relationship with Result.

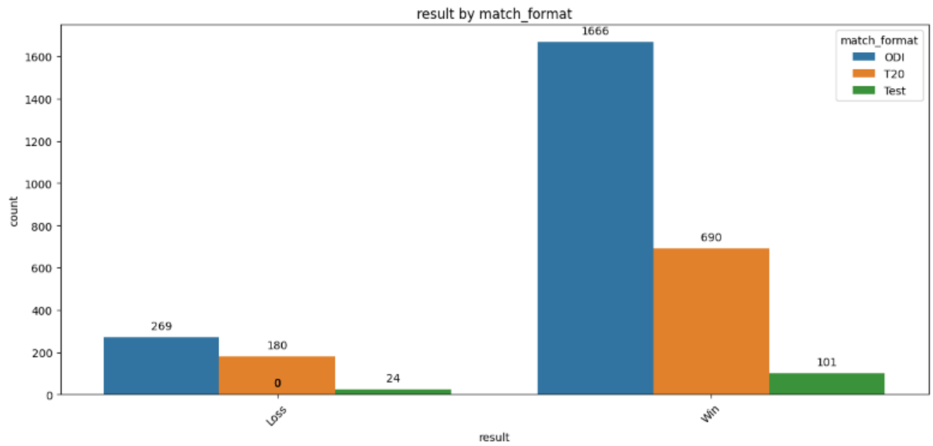
**Examples**:

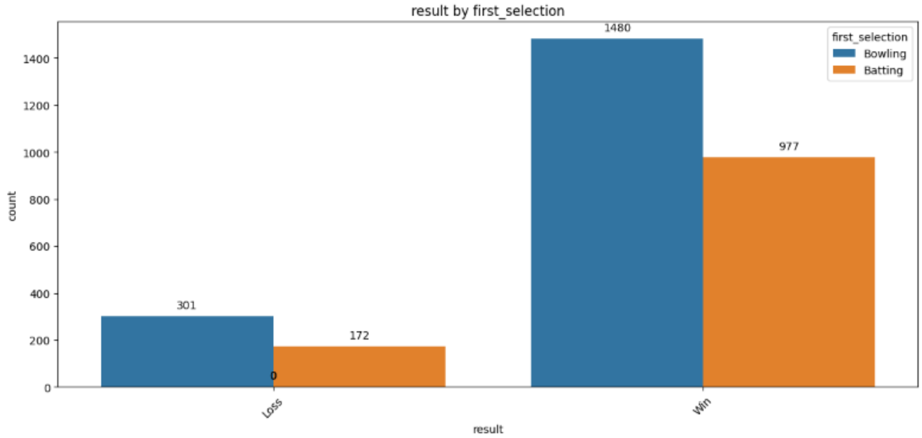
- **Box Plot for Max_run_scored_1over vs Result**:

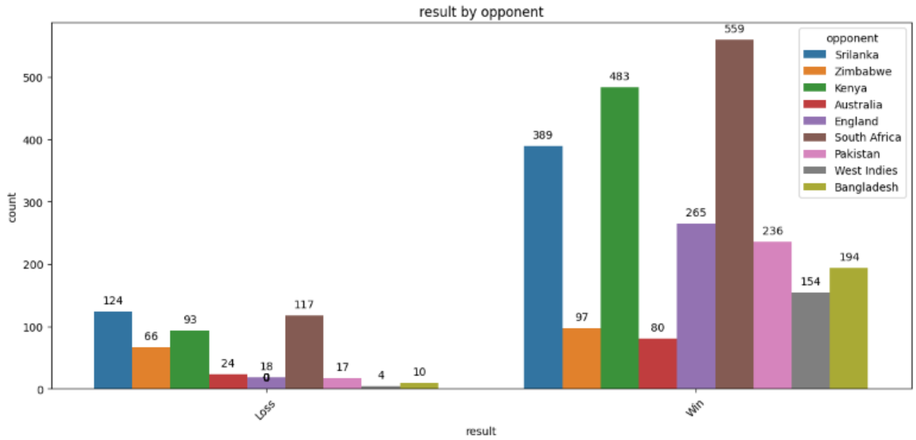
**Business Impact**:

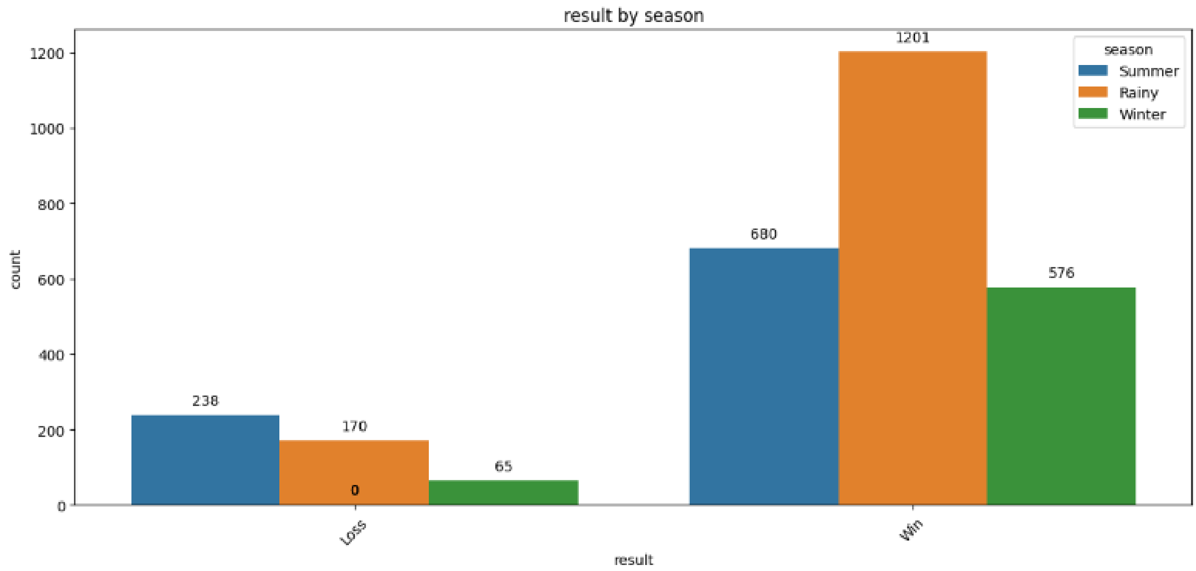
- **Performance Insights**: Comparing Max_run_scored_1over with Result can show how aggressive batting impacts match outcomes, helping in strategy formulation for different match formats.
- **Resource Allocation**: Understanding how Match_light_type impacts performance can guide scheduling and resource allocation for practice sessions and match preparations.




result by first_selection
first_selection
Bowling
Batting
1480
977
301
172
0
count
result
Loss
Win
0
200
400
600
800
1000
1200
1400


result by opponent
opponent
Srilanka
Zimbabwe
Kenya
Australia
England
South Africa
Pakistan
West Indies
Bangladesh
124
66
93
24
18
0
117
17
4
10
389
97
483
80
265
559
236
154
194
count
result
Loss
Win
0
100
200
300
400
500



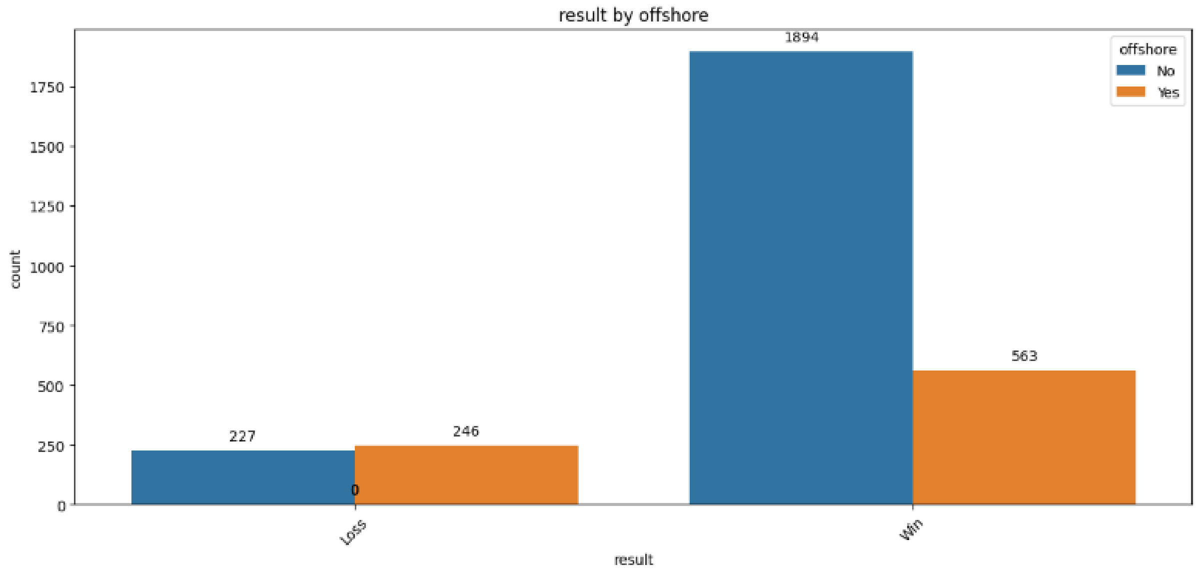




Fig 10

## 3. Data Cleaning and Preprocessing

## a) Missing Values

- Missing values are identified using the info(), built-in-method in python. So, there were total 2930 rows in the dataset, any column having less than that value are having missing values.
- **Approach used:** Missing values were treated using various imputation techniques, such as mean imputation for continuous variables and mode imputation for categorical variables.

**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:**

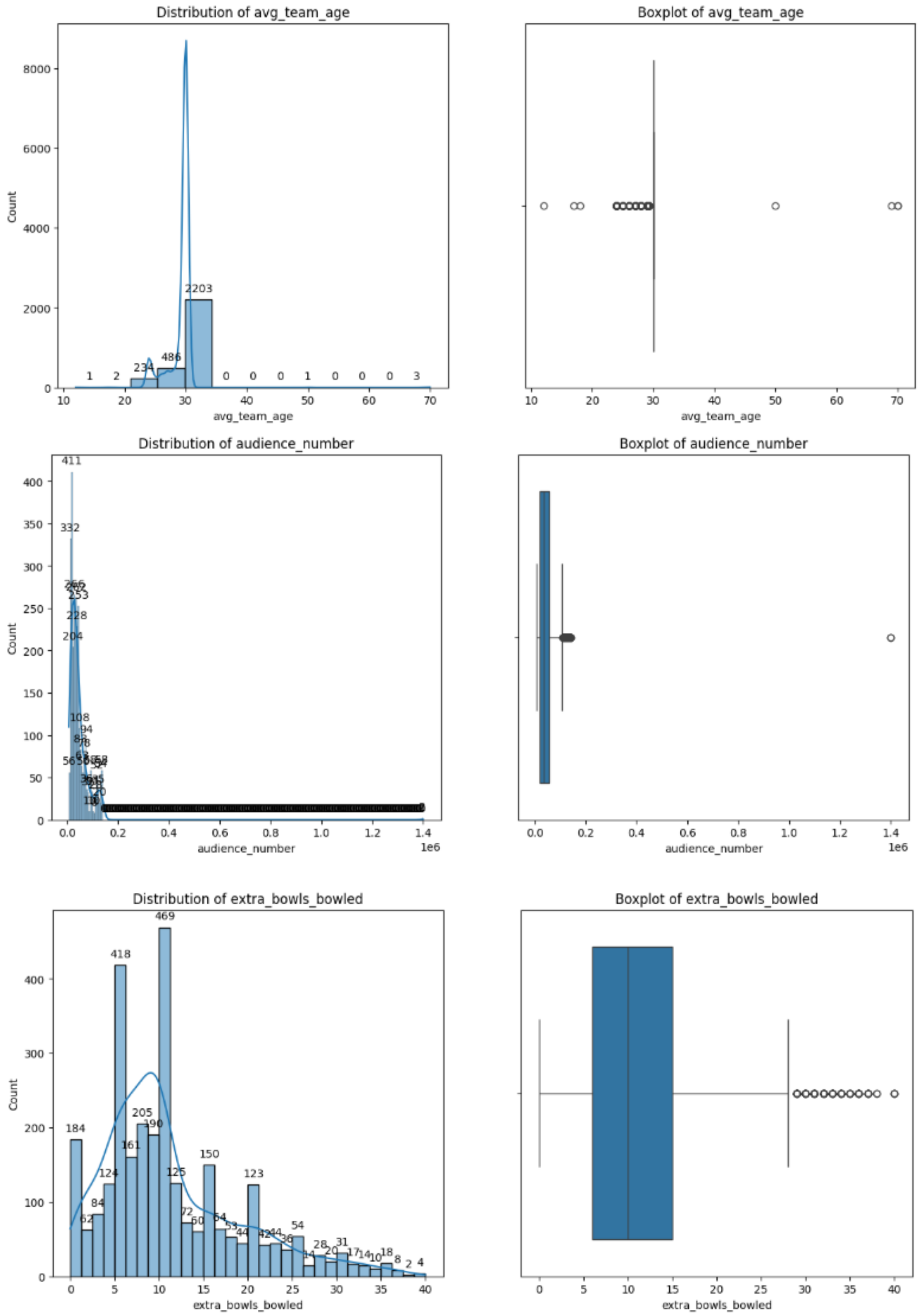
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   game_number              2930 non-null   object 
 1   result                   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

**b) Outlier treatment (if required)**

- Outliers were found using Visualizations, majorly Box Plots were used.
- Outliers were identified using statistical methods like the IQR (Interquartile Range) method.
- **Approach Used for Treatment:**
- Capping/Flooring: Extreme values were capped at the 1st and 99th percentiles to reduce their influence.
- Transformation: Log transformation was applied to variables with significant skewness to reduce the impact of outliers.
- Outlier treated for Continuous attributes.

**BEFORE:**


Distribution of avg_team_age
Count
avg_team_age
1 2 234 486 2203 0 0 0 1 0 0 0 3
Boxplot of avg_team_age
avg_team_age
Distribution of audience_number
Count
audience_number
1e6
411
332
228
204
108
Boxplot of audience_number
audience_number
1e6
Distribution of extra_bowls_bowled
Count
extra_bowls_bowled
469 418 205 190 184 161 150 124 125 123 84 62 72 60 64 53 44 42 44 36 54 28 31 20 17 14 14 10 18 8 2 4
Boxplot of extra_bowls_bowled
extra_bowls_bowled

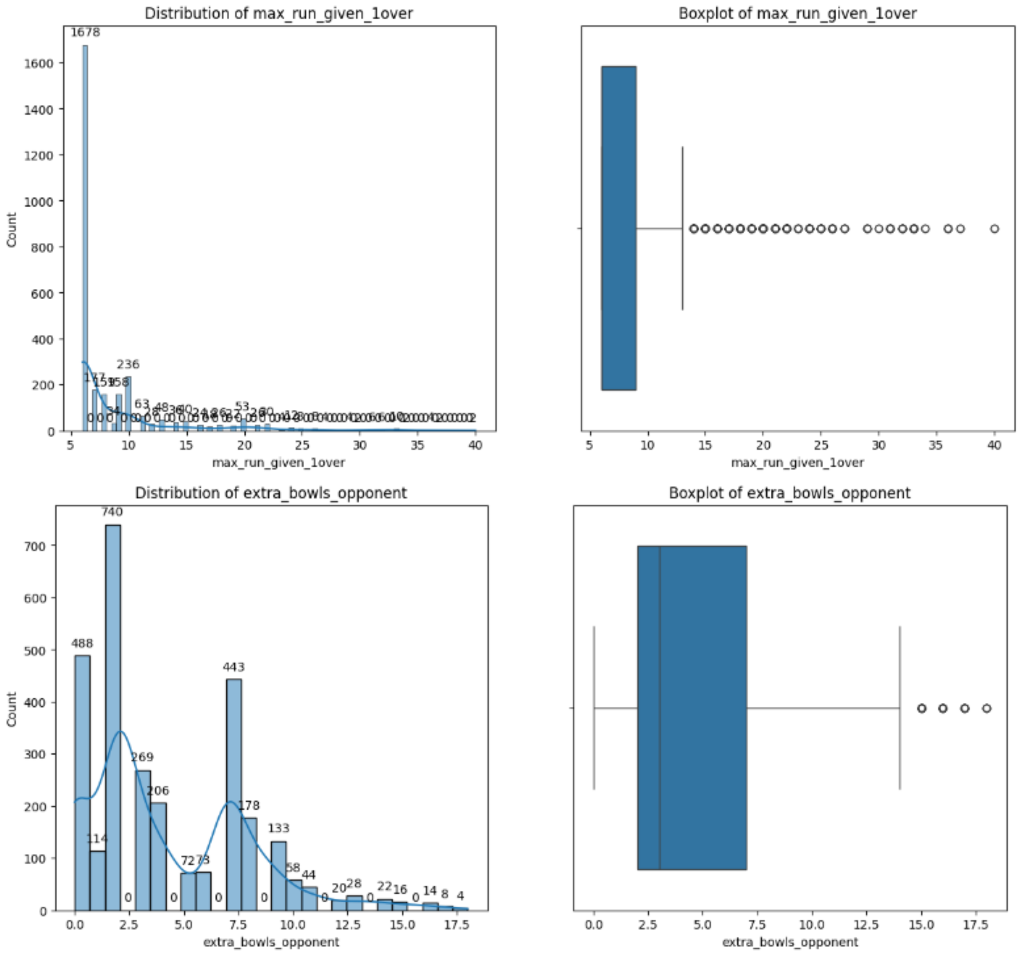
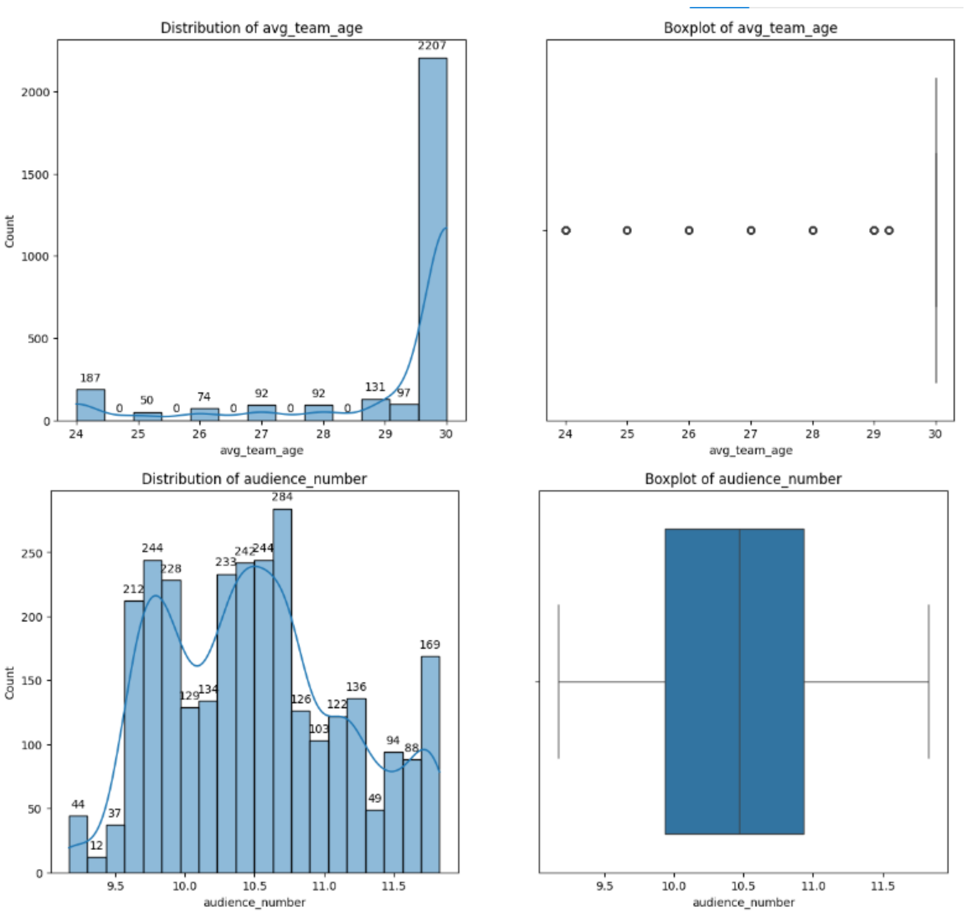
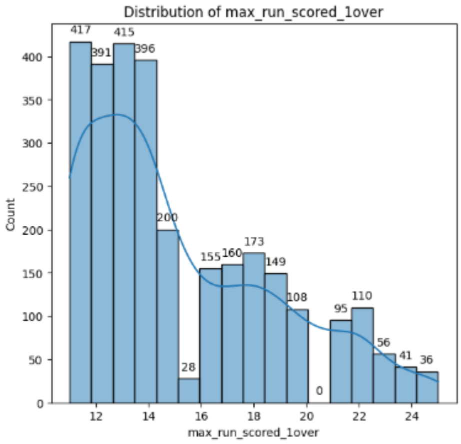

Distribution of max_run_given_1over
Count
max_run_given_1over
Boxplot of max_run_given_1over
max_run_given_1over
Distribution of extra_bowls_opponent
Count
extra_bowls_opponent
Boxplot of extra_bowls_opponent
extra_bowls_opponent

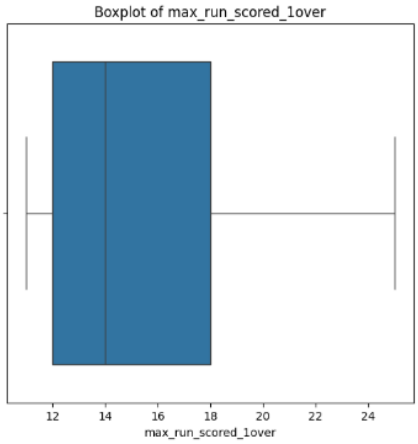
Fig 12.1

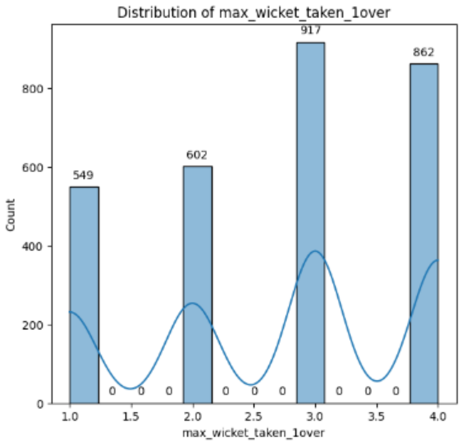
**AFTER:**

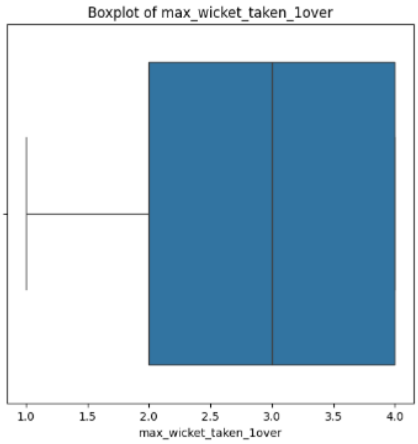


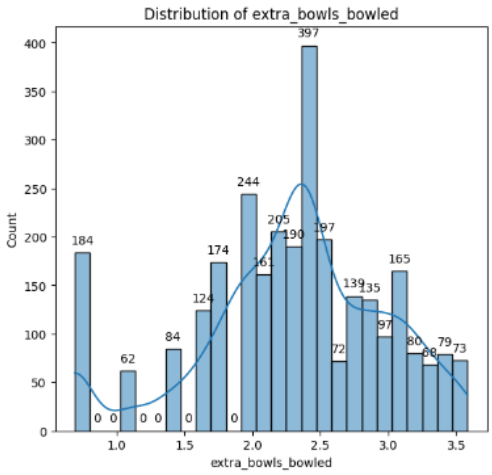
Distribution of avg_team_age
Boxplot of avg_team_age
Count
avg_team_age
2207
187
50
74
92
92
131
97
0
Distribution of audience_number
Boxplot of audience_number
audience_number
284
244
228
212
233
242
244
129
134
126
122
136
103
169
94
88
49
44
12
37

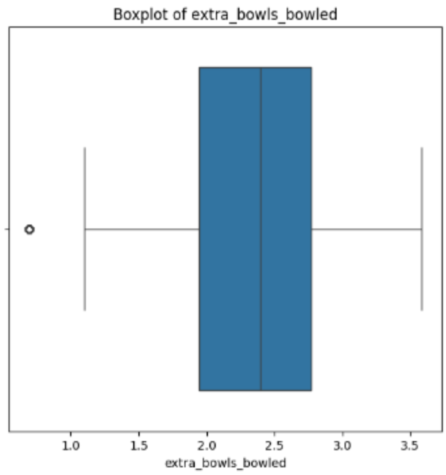

Distribution of max_run_scored_1over
417
391
415
396
200
155
160
173
149
108
28
0
95
110
56
41
36
Count
max_run_scored_1over


Boxplot of max_run_scored_1over
max_run_scored_1over


Distribution of max_wicket_taken_1over
549
602
917
862
Count
max_wicket_taken_1over


Boxplot of max_wicket_taken_1over
max_wicket_taken_1over


Distribution of extra_bowls_bowled
184
62
84
124
174
244
161
205
190
397
197
72
139
135
97
165
80
68
79
73
Count
extra_bowls_bowled


Boxplot of extra_bowls_bowled
extra_bowls_bowled

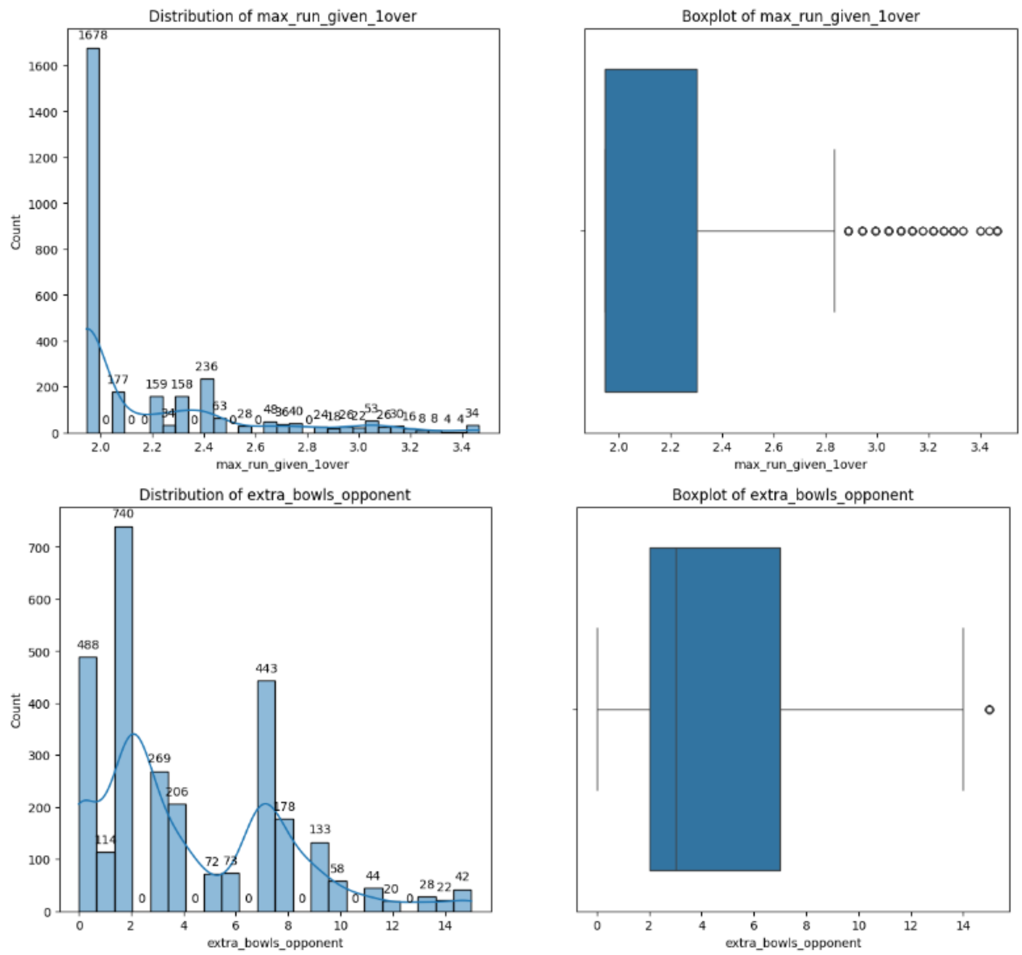



Fig 12.2

Outlier treated for Continuous vars, but we can still see that there are outliers present for few variables. 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.

### c) Variable transformation (if applicable)

**Log Transformation**:

- Applied log transformation to variables with highly skewed distributions to normalize the data.

**Standardization**:

- Standardized continuous variables to have a mean of 0 and a standard deviation of 1 to improve model performance.

**Why**:

- Log transformation reduces skewness and makes the data more normally distributed.
- Standardization ensures that variables are on a similar scale, which is important for algorithms sensitive to data scaling (e.g., SVM, KNN).

- **Identifying Missing Values**: Used EDA and summary statistics.
- **Treating Missing Values**: Imputed continuous variables with mean/median and categorical variables with mode.
- **Identifying Outliers**: Used box plots and the IQR method.
- **Treating Outliers**: Applied capping and flooring.
- **Variable Transformation**: Used log transformation for skewed variables and standardization for continuous variables.
- **Variable Removal**: Removed highly correlated variables to avoid multicollinearity.
- **Variable Addition**: Added interaction terms to capture combined effects.

These steps ensure that the data is clean, well-distributed, and suitable for modeling, which leads to more accurate and reliable predictive models.

4. **Model building.**

**Model Selection Criteria**

**1. Decision Tree:**

- **Interpretability:** Easy to understand and visualize.
- **Handling Non-linear Relationships:** Capable of capturing non-linear patterns.
- **Feature Importance:** Provides insight into the importance of different features.

**2. Random Forest:**

- **Ensemble Technique:** Reduces overfitting by averaging multiple decision trees.
- **Robustness:** Handles missing values and maintains performance with noisy data.
- **Feature Importance:** Offers better feature importance measures compared to a single decision tree.

**3. AdaBoost:**

- **Boosting Technique:** Combines weak learners to form a strong learner, improving accuracy.
- **Handling Bias:** Focuses on difficult-to-classify instances, reducing bias.
- **Versatility:** Can be combined with various base learners (e.g., decision trees).

**4. XGBoost:**

- **Efficiency:** Highly efficient and optimized implementation of gradient boosting.
- **Accuracy:** Often provides superior performance in predictive modelling competitions.
- **Regularization:** In-built regularization to prevent overfitting.

**5. Support Vector Machine (SVM):**

- **High-dimensional Spaces:** Effective in high-dimensional spaces and with non-linear decision boundaries.
- **Margin Maximization:** Focuses on maximizing the margin between classes, enhancing generalization.

**6. K-Nearest Neighbours (KNN):**

- **Simplicity:** Simple and easy to implement.
- **Non-parametric:** Makes no assumptions about data distribution.
- **Local Patterns:** Captures local patterns and relationships in data.

**7. Linear Discriminant Analysis (LDA):**

- **Linear Separation:** Suitable for problems with linearly separable classes.
- **Dimensionality Reduction:** Reduces dimensionality while maintaining class separability.

**8. Naive Bayes:**

- **Probabilistic Model:** Simple and efficient probabilistic model.

- **Independence Assumption:** Assumes feature independence, which simplifies computation.

**Effort to Improve Model Performance**

**Model Tuning and Performance Improvement**

**1. Hyperparameter Tuning:**

- **Grid Search:** Used GridSearchCV to find the optimal hyperparameters for each model.

**2. Ensemble Modelling:**

- **Stacking:** Combined multiple models to form an ensemble for better performance.

**3. Feature Engineering:**

- **Interaction Terms:** Added interaction terms to capture non-linear relationships.
- **Polynomial Features:** Generated polynomial features to improve model complexity.

**4. Cross-Validation:**

- **K-Fold Cross-Validation:** Used K-fold cross-validation to ensure model robustness and avoid overfitting.

**5. Feature Selection:**

- **Recursive Feature Elimination (RFE):** Used RFE to select the most important features.

**a. Build various models**

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

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

| | AUC-ROC |
|---|---|
| 0 | 0.836494 |
| 1 | 0.922278 |
| 2 | 0.842944 |
| 3 | 0.916989 |
| 4 | 0.536765 |
| 5 | 0.694272 |
| 6 | 0.832495 |
| 7 | 0.797085 |
| 8 | 0.807082 |
| 9 | 0.953722 |
| 10 | 0.821336 |
| 11 | 0.917763 |
| 12 | 0.654380 |
| 13 | 0.705528 |
| 14 | 0.827012 |
| 15 | 0.767608 |

Fig 13

5. **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.

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

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

| | AUC-ROC |
|---|---|
| 0 | 0.836494 |
| 1 | 0.922278 |
| 2 | 0.842944 |
| 3 | 0.916989 |
| 4 | 0.536765 |
| 5 | 0.694272 |
| 6 | 0.832495 |
| 7 | 0.797085 |
| 8 | 0.807082 |
| 9 | 0.953722 |
| 10 | 0.821336 |
| 11 | 0.917763 |
| 12 | 0.654380 |
| 13 | 0.705528 |
| 14 | 0.827012 |
| 15 | 0.767608 |
| 16 | 0.950690 |
| 17 | 0.923568 |
| 18 | 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**.**

6. **Recommendations**

**1. Team Composition Strategy:**

- **All-rounder in Team:**
  - **Univariate Analysis:** Matches with a higher number of all-rounders show a higher probability of winning.
  - **Recommendation:** Increase the number of all-rounders in the playing XI. All-rounders provide balance to the team by contributing both with bat and ball, increasing the chances of winning.

**2. Match Format Specific Strategies:**

- **T20 and ODI Matches:**
  - **Bivariate Analysis:** High correlation between having more aggressive batsmen (high max runs scored in an over) and winning T20 and ODI matches.

- **Recommendation:** In T20 and ODI formats, prioritize aggressive batsmen who can score quickly and maximize runs per over. This strategy aligns with the fast-paced nature of these formats.

**3. Bowling Strategy:**

- **Bowlers in Team:**
  - **Univariate Analysis:** Matches with more full-time bowlers tend to have better outcomes.
  - **Recommendation:** Ensure enough full-time bowlers in the team, especially for Test matches. Full-time bowlers are crucial for taking wickets and controlling the run rate.

**4. Match Conditions and Opponent Analysis:**

- **Day/Night Matches:**
  - **Bivariate Analysis:** Matches played under lights (night or day & night) show different performance patterns.
  - **Recommendation:** Prepare the team for the unique challenges of day/night matches. This includes practicing under lights and understanding how the conditions change (e.g., dew factor, visibility).
- **Opponent Specific Strategy:**
  - **Bivariate Analysis:** Different opponents exhibit varied weaknesses and strengths.
  - **Recommendation:** Conduct a detailed analysis of the opponent team's past performance and tailor the team's strategy accordingly. For instance, if the opponent is weak against spin bowling, include more spinners in the team.

**5. Handling Pressure Situations:**

- **Players Scored Zero:**
  - **Univariate Analysis:** Matches with more players scoring zero runs tend to have negative outcomes.
  - **Recommendation:** Provide mental conditioning and pressure-handling training to players to reduce the number of ducks. Encourage players to build their innings and avoid reckless shots.

**6. Performance Metrics:**

- **Highest Runs and Wickets:**
  - **Bivariate Analysis:** Matches where individual players score high runs or take multiple wickets tend to have better outcomes.

- **Recommendation:** Identify and nurture key players who consistently perform well. Ensure they receive adequate support and training to maintain their performance levels.

**7. Audience and Home Advantage:**

- **Audience Number and Offshore Matches:**
  - **Bivariate Analysis:** Matches played at home or with a higher audience turnout often have positive outcomes.
  - **Recommendation:** Leverage home advantage by understanding local pitch conditions and ensuring strong crowd support. When playing offshore, prepare thoroughly for the different conditions and crowd dynamics.

**Data-Driven Insights:**

- **Correlation Analysis:**
  - **Max Runs Scored in 1 Over:** Positive correlation with match wins.
  - **Max Wickets Taken in 1 Over:** Positive correlation with match wins.
  - **All-rounders in Team:** Positive impact on match results.
- **Heatmap Insights:**
  - **Audience Number:** Higher audience numbers correlate with better performance.
  - **Offshore Matches:** Matches played within the country show a higher win rate.

**Summary of Recommendations:**

1. Increase all-rounders in the team to enhance flexibility and balance.
2. Prioritize aggressive batsmen in T20 and ODI formats to maximize run rates.
3. Ensure a sufficient number of full-time bowlers, especially in Test matches.
4. Prepare for day/night matches with specific strategies to handle changing conditions.
5. Analyze opponent weaknesses and tailor the strategy accordingly.
6. Provide pressure-handling training to players to reduce the incidence of ducks.
7. Leverage home advantage and prepare thoroughly for offshore matches to mitigate unfamiliar conditions.
8. Nurture key players who consistently perform well in terms of runs and wickets.

These recommendations are based on detailed univariate and bivariate analyses, ensuring that they are data-driven and specific to the observed patterns in the dataset.