Capstone Project Cricket win prediction Report - Yogessh Balamurugan

Table of Contents

Introduction of the business problem 4
Problem Statement 4
Need of the study/project 4
Visual inspection of data (rows, columns, descriptive details) 4
Understanding of attributes (variable info, renaming if required) 5
Exploratory data analysis 5
Missing Value treatment (if applicable) 5
Univariate analysis 7
Bivariate analysis 11
Multivariate analysis 12
Outlier Treatment 12
Variable transformation 13
Label Encoding & Feature selection 13
Train – Test split: 14
Machine Learning Models: 14
Training Model 20
Testing Model 21
Interpretation of the model 22
Recommendation 22
Conclusion 25

Table of Figures

1. Top 5 Rows of the Dataset 4
2. Last 5 Rows of the Dataset 4
3. Descriptive statistics 5
4. Histplot of Avg team age, Distribution of bowlers in team, all rounder in team, Max run scored in 1 over, Max wicket taken in 1 over 7
5. Histplot of Min run given 1 over, Boxplot of extra bowls opponent, Count plot of Result, Match light type, first selection and opponent 8
6. Count plot of Season, offshore. Descriptive Analysis – Results, Match light type, Match format, First selection 9
7. Descriptive Analysis - Season, offshore, opponent 10
8. Bivariate analysis count plot - Result vs opponent, Result vs season, Results vs offshore, Result vs match format 11
9. Multivariate Analysis 12
10. Logistic Regression ROC curve 15
11. Linear Discriminant Analysis ROC curve 16
12. KNeighborsClassifier ROC curve 16
13. Decision Tree classifier ROC curve 17
14. Support vector classifier ROC curve 17
15. Bagging classifier ROC curve 18
16. Random Forest classifier ROC curve 19
17. Adaboost classifier ROC curve 19
18. Gradient Booting classifier ROC curve 20
19. Machine learning models - ROC curve 21
20. Interpretation of the models 22

Introduction of the business problem

Problem Statement

BCCI has hired an external analytics consulting firm for data analytics. The major objective of this tie up is to extract actionable insights from the historical match data and make strategic changes to make India win. Primary objective is to create Machine Learning models which correctly predicts a win for the Indian Cricket Team. Once a model is developed then you have to extract actionable insights and recommendation. Also, below are the details of the next 10 matches, India is going to play. You have to predict the result of the matches and if you are getting prediction as a Loss then suggest some changes and re-run your model again until you are getting Win as a prediction. You cannot use the same strategy in the entire series, because opponent will get to know your strategy and they can come with counter strategy. Hence for all the below 5 matches you have to suggest unique strategies to make India win. The suggestions should be in-line with the variables that have been mentioned in the given data set.

Do consider the feasibility of the suggestions very carefully as well.

1. 1 Test match with England in England. All the match are day matches. In England, it will be rainy season at the time to match.
2. 2 T20 match with Australia in India. All the match are Day and Night matches. In India, it will be winter season at the time to match.
3. 2 ODI match with Sri Lanka in India. All the match are Day and Night matches. In India, it will be winter season at the time to match.

Need of the study/project

The project needs to develop strategies for the 5 matches mentioned above to help India win by analyzing the past data of the Indian team matches.

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

Top 5 Rows of the dataset

Out[39]:

	Game_number	Result	Avg_team_Age	Match_light_type	Match_format	Bowlers_in_team	Wicket_keeper_in_team	All_rounder_in_team	First_selection	Oppor
0	Game_1	Loss	18.0	Day	ODI	3.0	1	3.0	Bowling	Srila
1	Game_2	Win	24.0	Day	T20	3.0	1	4.0	Batting	Zimba
2	Game_3	Loss	24.0	Day and Night	T20	3.0	1	2.0	Bowling	Zimba
3	Game_4	Win	24.0	NaN	ODI	2.0	1	2.0	Bowling	Ke
4	Game_5	Loss	24.0	Night	ODI	1.0	1	3.0	Bowling	Srila

1. Top 5 Rows of the Dataset

Last 5 Rows of the dataset

Out[42]:

	Game_number	Result	Avg_team_Age	Match_light_type	Match_format	Bowlers_in_team	Wicket_keeper_in_team	All_rounder_in_team	First_selection	Op
2925	Game_2926	Win	30.0	Day	T20	3.0	1	4.0	Batting	
2926	Game_2927	Win	30.0	Day	ODI	4.0	1	3.0	Bowling	
2927	Game_2928	Win	30.0	Day and Night	ODI	4.0	1	3.0	Bowling	F
2928	Game_2929	Win	30.0	Day	ODI	4.0	1	3.0	Batting	
2929	Game_2930	Win	30.0	Day	ODI	4.0	1	3.0	Batting	

2. Last 5 Rows of the Dataset

Descriptive statistics about the dataset

Out[45]:

	count	mean	std	min	25%	50%	75%	max
Avg_team_Age	2833.0	29.242852	2.264230	12.0	30.0	30.0	30.00	70.0
Bowlers_in_team	2848.0	2.913624	1.023907	1.0	2.0	3.0	4.00	5.0
Wicket_keeper_in_team	2930.0	1.000000	0.000000	1.0	1.0	1.0	1.00	1.0
All_rounder_in_team	2890.0	2.722491	1.092699	1.0	2.0	3.0	4.00	4.0
Audience_number	2849.0	46267.960688	48599.581459	7063.0	20363.0	34349.0	57876.00	1399930.0
Max_run_scored_1over	2902.0	15.199862	3.661010	11.0	12.0	14.0	18.00	25.0
Max_wicket_taken_1over	2930.0	2.713993	1.080623	1.0	2.0	3.0	4.00	4.0
Extra_bowls_bowled	2901.0	11.252671	7.780829	0.0	6.0	10.0	15.00	40.0
Min_run_given_1over	2930.0	1.952560	1.678332	0.0	0.0	2.0	3.00	6.0
Min_run_scored_1over	2903.0	2.762659	0.705759	1.0	2.0	3.0	3.00	4.0
Max_run_given_1over	2896.0	8.669199	5.003525	6.0	6.0	6.0	9.25	40.0
extra_bowls_opponent	2930.0	4.229693	3.626108	0.0	2.0	3.0	7.00	18.0
player_highest_run	2902.0	65.889387	20.331614	30.0	48.0	66.0	84.00	100.0

3. Descriptive statistics

Understanding of attributes (variable info, renaming if required)

Information about the dataset

```
<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
```

Exploratory data analysis

Missing Value treatment (if applicable)

```
Out[46]: Game_number               0
         Result                    0
         Avg_team_Age             97
         Match_light_type         52
         Match_format             70
         Bowlers_in_team          82
         Wicket_keeper_in_team     0
         All_rounder_in_team      40
         First_selection          59
         Opponent                 36
         Season                   62
         Audience_number          81
         Offshore                 64
         Max_run_scored_1over     28
         Max_wicket_taken_1over    0
         Extra_bowls_bowled       29
         Min_run_given_1over       0
         Min_run_scored_1over     27
         Max_run_given_1over      34
         extra_bowls_opponent      0
         player_highest_run       28
         Players_scored_zero       0
         player_highest_wicket     0
         dtype: int64
```

There are null values found in the columns of Avg_team_Age, Match_light_type, Match_format, Bowlers_in_team, All_rounder_in_team, First_selection, Opponent, Season, Audience_number, Offshore, Max_run_scored_1over, Extra_bowls_bowled, Min_run_scored_1over, Max_run_given_1over, player_highest_run.

Unique values in each column & imputed the null values of match format columns using KNN imputer.

```
Unique values in column 'Game_number': ['Game_1' 'Game_2' 'Game_3' ... 'Game_2928' 'Game_2929' 'Game_2930']
Unique values in column 'Result': ['Loss' 'Win']
Unique values in column 'Avg_team_Age': [18. 24. nan 17. 12. 25. 26. 27. 28. 29. 30. 70. 69. 50.]
Unique values in column 'Match_light_type': ['Day' 'Day and Night' nan 'Night']
Unique values in column 'Match_format': ['ODI' 'T20' 'Test' '20-20' nan]
Unique values in column 'Bowlers_in_team': [ 3.  2.  1. nan  4.  5.]
Unique values in column 'Wicket_keeper_in_team': [1]
Unique values in column 'All_rounder_in_team': [ 3.  4.  2.  1. nan]
Unique values in column 'First_selection': ['Bowling' 'Batting' 'Bat' nan]
Unique values in column 'Opponent': ['Srilanka' 'Zimbabwe' 'Kenya' 'Australia' 'England' 'South Africa'
 'Pakistan' 'West Indies' 'Bangladesh' nan]
Unique values in column 'Season': ['Summer' nan 'Winter' 'Rainy']
Unique values in column 'Audience_number': [ 9940.  8400. 13146. ... 20937. 28756. 14007.]
Unique values in column 'Offshore': ['No' 'Yes' nan]
Unique values in column 'Max_run_scored_1over': [13. 12. 14. 15. 16. 19. 21. 22. 25. 18. 11. 17. 24. 20. nan 23.]
Unique values in column 'Max_wicket_taken_1over': [3 1 4 2]
Unique values in column 'Extra_bowls_bowled': [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 17. 31. 13. 26. 24.
 22. 19. 23. 14. 15. 29. 28. 21. 25. 20. 16. 37. 38. 30. 40. 18. 36. 34.
 32. 33. nan 35. 27.]
Unique values in column 'Min_run_given_1over': [2 0 5 4 3 6 1]
Unique values in column 'Min_run_scored_1over': [ 3.  4.  1.  2. nan]
Unique values in column 'Max_run_given_1over': [ 6.  7.  8.  9. 10. 11. 25. 12. 14. 22. 15. 27. 21. 17. 13. 37. 16. 20.
 40. 24. 33. 19. 36. 18. 29. 31. 32. 34. 26. 30. nan 23.]
Unique values in column 'extra_bowls_opponent': [ 0  1  2  3  4  5  7  8  6  9 10 13 15 14 16 11 12 18 17]
Unique values in column 'player_highest_run': [ 54.  69.  73.  80.  97.  70.  33.  50.  79. 100.  47.  52.  37.  96.
  57.  66.  83.  32.  49.  84.  98.  87.  43.  31.  99.  39.  90.  65.
  36.  45.  58.  59.  60.  34.  94.  48.  85.  75.  63.  62.  93.  51.
  78.  61.  77.  38.  89.  55.  46.  41.  42.  91.  64.  40.  67.  81.
  68.  82.  71.  88.  53.  76.  95.  30.  74.  56.  86.  44.  92.  72.
  35.  nan]
Unique values in column 'Players_scored_zero': [3 2 1 4 'Three']
Unique values in column 'player_highest_wicket': [1 2 3 4 'Three' 5]
```

```
Out[58]: array(['T20', 'ODI', 'ODI', 'ODI', 'ODI', 'T20', 'T20', 'T20', 'ODI',
       'ODI', 'ODI', 'T20', 'ODI', 'ODI', 'ODI', 'ODI', 'T20', 'T20',
       'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'T20', 'ODI',
       'ODI', 'ODI', 'ODI', 'T20', 'T20', 'T20', 'ODI', 'ODI', 'ODI',
       'ODI', 'T20', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI',
       'ODI', 'T20', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'ODI',
       'ODI', 'ODI', 'ODI', 'ODI', 'ODI', 'T20', 'ODI', 'T20', 'ODI',
       'ODI', 'ODI', 'Test', 'ODI', 'ODI', 'ODI', 'ODI'], dtype=object)
```

For categorical columns such as Offshore, Season, First_selection, Match_light_type imputed null values with mode. The mode value for each categorical columns are listed below, Mode of Offshore: No,
Mode of Season: Rainy, Mode of First_selection: Bowling, Mode of Match_light_type: Day.
The columns Match_format and First_selection had duplicate values and replaced with correct values.

After imputing null values present in the dataset. The descriptive statistics and information about the dataset are enclosed below,

```
Game_number               0
Result                    0
Avg_team_Age              0
Match_light_type          0
Match_format              0
Bowlers_in_team           0
Wicket_keeper_in_team     0
All_rounder_in_team       0
First_selection           0
Opponent                  0
Season                    0
Audience_number           0
Offshore                  0
Max_run_scored_1over      0
Max_wicket_taken_1over    0
Extra_bowls_bowled        0
Min_run_given_1over       0
Min_run_scored_1over      0
Max_run_given_1over       0
extra_bowls_opponent      0
player_highest_run        0
Players_scored_zero       0
player_highest_wicket     0
dtype: int64
```

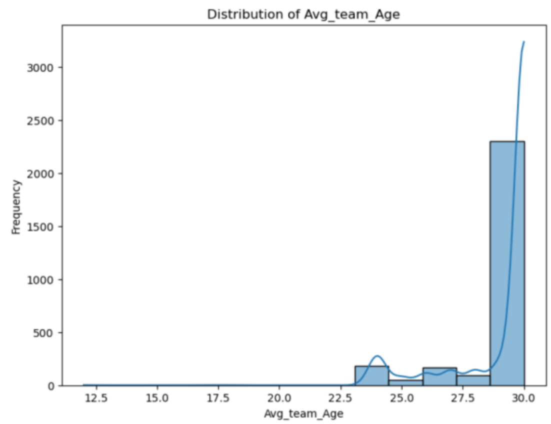
```
<class 'pandas.core.frame.DataFrame'>
Index: 2793 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Game_number             2793 non-null   object 
 1   Result                  2793 non-null   object 
 2   Avg_team_Age            2793 non-null   float64
 3   Match_light_type        2793 non-null   object 
 4   Match_format            2793 non-null   object 
 5   Bowlers_in_team         2793 non-null   int32  
 6   Wicket_keeper_in_team   2793 non-null   int32  
 7   All_rounder_in_team     2793 non-null   int32  
 8   First_selection         2793 non-null   object 
 9   Opponent                2793 non-null   object 
 10  Season                  2793 non-null   object 
 11  Audience_number         2793 non-null   float64
 12  Offshore                2793 non-null   object 
 13  Max_run_scored_1over    2793 non-null   int32  
 14  Max_wicket_taken_1over  2793 non-null   int32  
 15  Extra_bowls_bowled      2793 non-null   int32  
 16  Min_run_given_1over     2793 non-null   float64
 17  Min_run_scored_1over    2793 non-null   int32  
 18  Max_run_given_1over     2793 non-null   int32  
 19  extra_bowls_opponent    2793 non-null   int32  
 20  player_highest_run      2793 non-null   float64
 21  Players_scored_zero     2793 non-null   float64
 22  player_highest_wicket   2793 non-null   float64
dtypes: float64(6), int32(9), object(8)
memory usage: 425.5+ KB
```

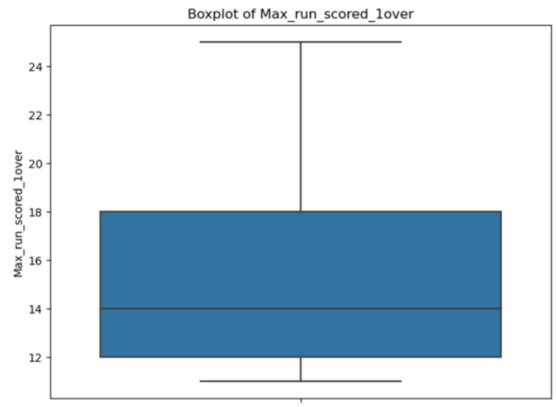
Out[62]:

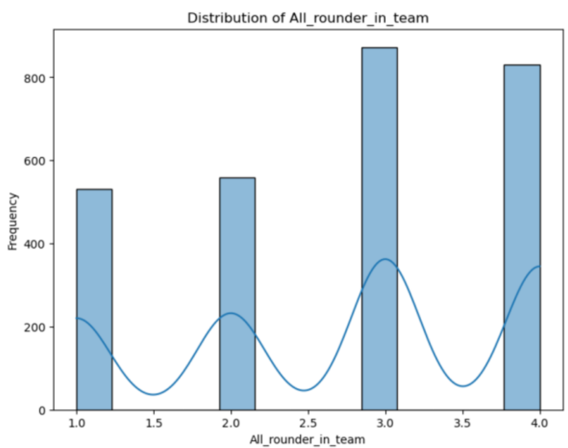
	count	mean	std	min	25%	50%	75%	max
Avg_team_Age	2793.0	29.182241	1.810729	12.0	30.0	30.0	30.0	30.0
Bowlers_in_team	2793.0	2.903187	1.022607	1.0	2.0	3.0	4.0	5.0
Wicket_keeper_in_team	2793.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0
All_rounder_in_team	2793.0	2.717150	1.084679	1.0	2.0	3.0	4.0	4.0
Audience_number	2793.0	46204.802005	48763.930610	7063.0	20328.0	34552.0	57568.0	1399930.0
Max_run_scored_1over	2793.0	15.215610	3.656942	11.0	12.0	14.0	18.0	25.0
Max_wicket_taken_1over	2793.0	2.721805	1.081699	1.0	2.0	3.0	4.0	4.0
Extra_bowls_bowled	2793.0	11.257429	7.775540	0.0	6.0	10.0	15.0	40.0
Min_run_given_1over	2793.0	1.963122	1.685434	0.0	0.0	2.0	3.0	6.0
Min_run_scored_1over	2793.0	2.765127	0.702901	1.0	2.0	3.0	3.0	4.0
Max_run_given_1over	2793.0	8.663086	4.978249	6.0	6.0	6.0	9.0	40.0
extra_bowls_opponent	2793.0	4.221984	3.621528	0.0	2.0	3.0	7.0	18.0
player_highest_run	2793.0	65.806731	20.204815	30.0	48.0	66.0	83.0	100.0
Players_scored_zero	2793.0	2.728965	0.707569	1.0	2.0	3.0	3.0	4.0
player_highest_wicket	2793.0	2.057644	1.107690	1.0	1.0	2.0	3.0	5.0

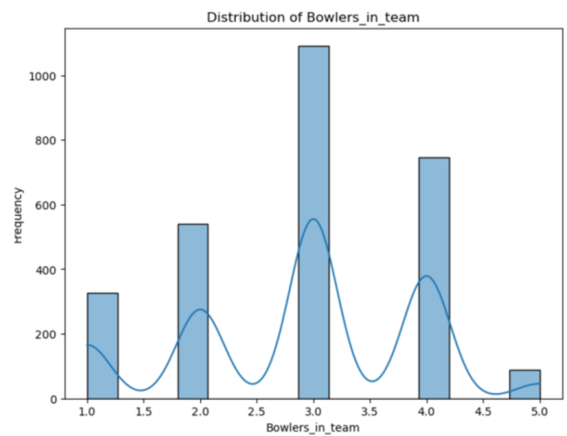
Univariate analysis

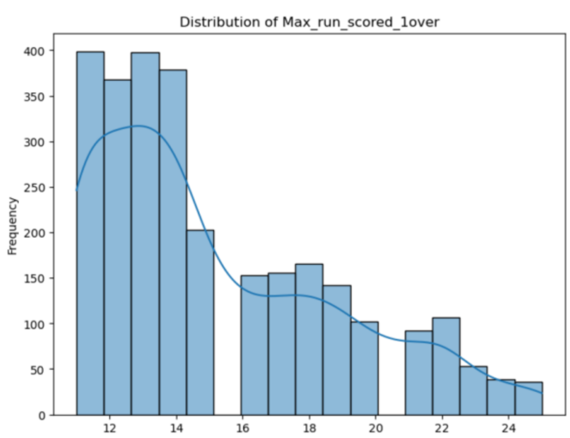
4. Histplot of Avg team age, Distribution of bowlers in team, all rounder in team, Max run scored in 1 over, Max wicket taken in 1 over

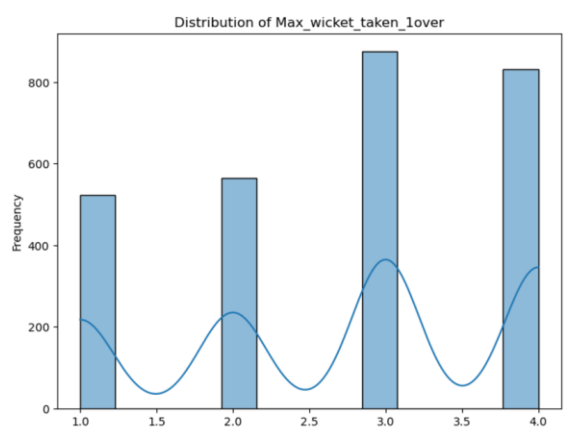




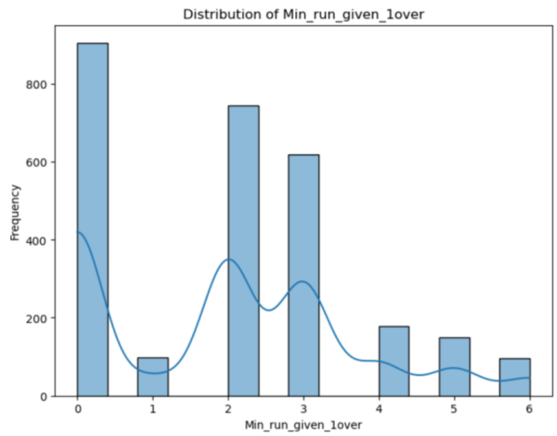


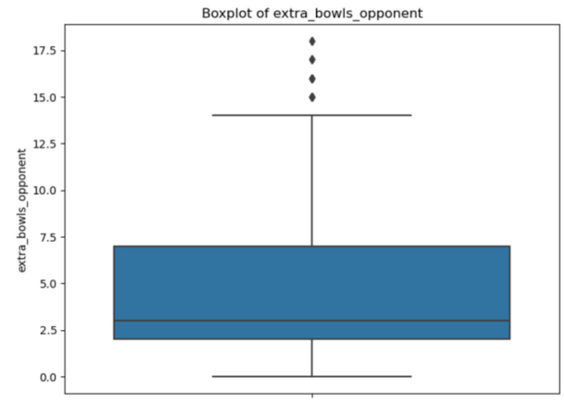


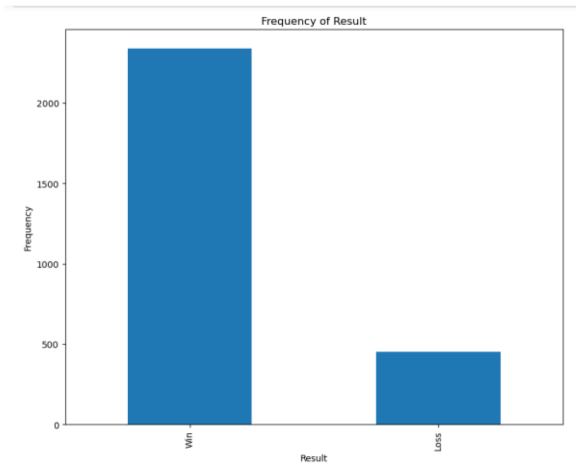


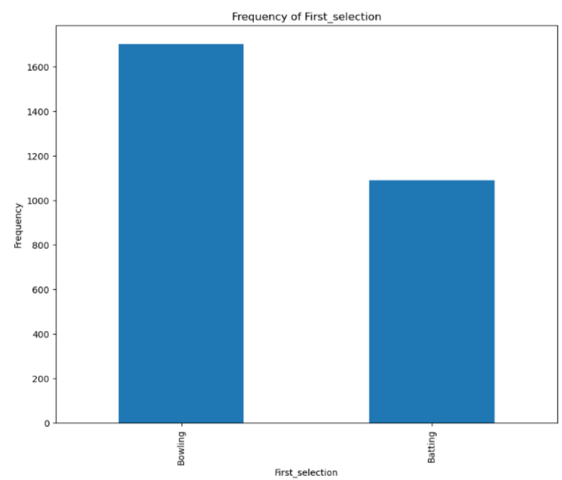


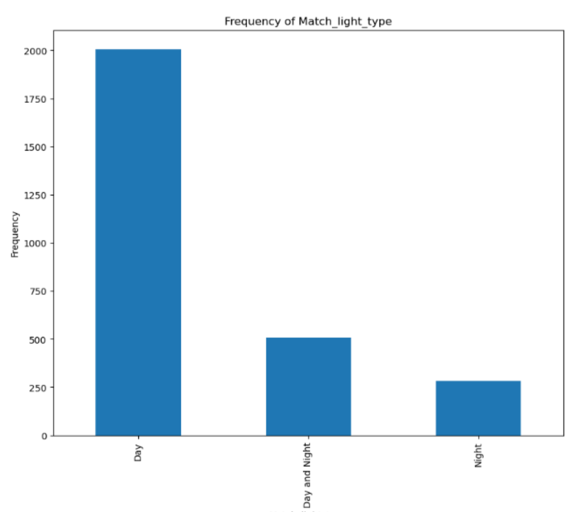
5. Histplot of Min run given 1 over, Boxplot of extra bowls opponent, Count plot of Result, Match light type, first selection and opponent

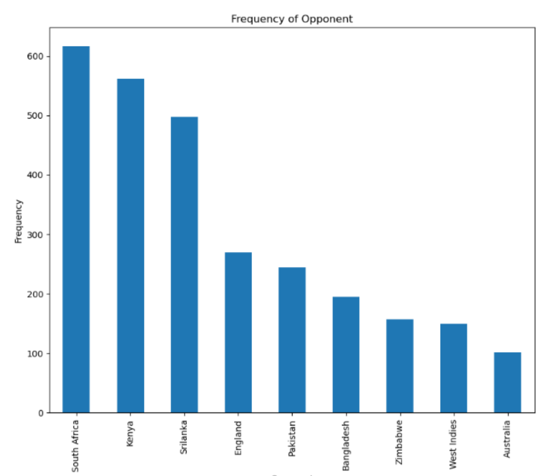




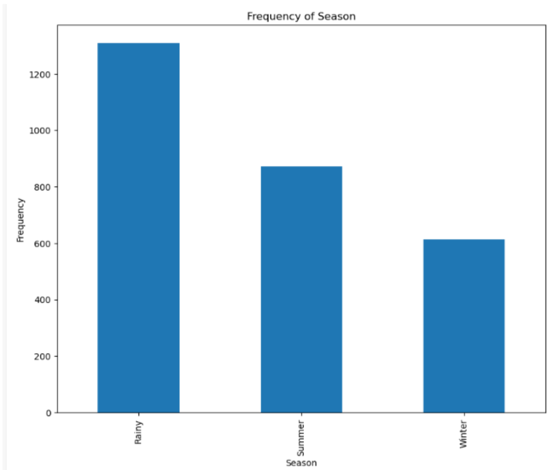


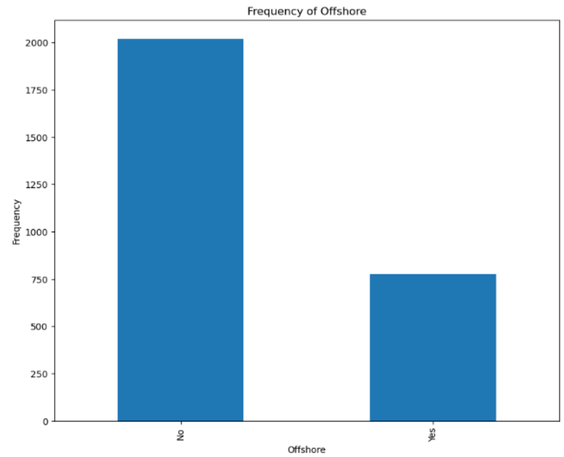






6. Count plot of Season, offshore. Descriptive Analysis – Results, Match light type, Match format, First selection






```
Descriptive Analysis for 'Result':
Unique values: ['Loss' 'Win']
Value counts:
Result
Win      2340
Loss      453
Name: count, dtype: int64
Value counts (%):
Result
Win      83.780881
Loss     16.219119
Name: proportion, dtype: float64
```

```
Descriptive Analysis for 'Match_format':
Unique values: ['ODI' 'T20' 'Test']
Value counts:
Match_format
ODI     1827
T20      844
Test     122
Name: count, dtype: int64
Value counts (%):
Match_format
ODI     65.413534
T20     30.218403
Test     4.368063
Name: proportion, dtype: float64
```

```
Descriptive Analysis for 'Match_light_type':
Unique values: ['Day' 'Day and Night' 'Night']
Value counts:
Match_light_type
Day              2005
Day and Night     507
Night             281
Name: count, dtype: int64
Value counts (%):
Match_light_type
Day              71.786609
Day and Night    18.152524
Night            10.060866
Name: proportion, dtype: float64
--------------------
```

```
Descriptive Analysis for 'First_selection':
Unique values: ['Bowling' 'Batting']
Value counts:
First_selection
Bowling    1702
Batting    1091
Name: count, dtype: int64
Value counts (%):
First_selection
Bowling    60.938059
Batting    39.061941
Name: proportion, dtype: float64
--------------------
```

7. Descriptive Analysis - Season, offshore, opponent

```
--------------------
Descriptive Analysis for 'Season':
Unique values: ['Summer' 'Rainy' 'Winter']
Value counts:
Season
Rainy     1308
Summer     871
Winter     614
Name: count, dtype: int64
Value counts (%):
Season
Rainy     46.831364
Summer    31.185106
Winter    21.983530
Name: proportion, dtype: float64
```

```
Descriptive Analysis for 'Offshore':
Unique values: ['No' 'Yes']
Value counts:
Offshore
No     2018
Yes     775
Name: count, dtype: int64
Value counts (%):
Offshore
No     72.252059
Yes    27.747941
Name: proportion, dtype: float64
--------------------
```

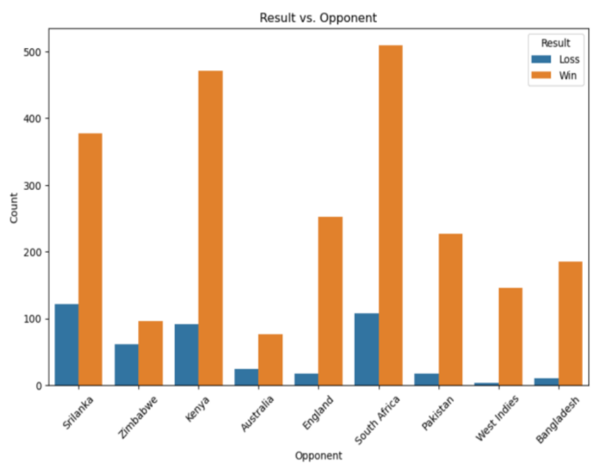
```
Descriptive Analysis for 'Opponent':
Unique values: ['Srilanka' 'Zimbabwe' 'Kenya' 'Australia' 'England' 'South Africa'
 'Pakistan' 'West Indies' 'Bangladesh']
Value counts:
Opponent
South Africa    617
Kenya           562
Srilanka        498
England         269
Pakistan        244
Bangladesh      195
Zimbabwe        157
West Indies     150
Australia       101
Name: count, dtype: int64
Value counts (%):
Opponent
South Africa    22.090942
Kenya           20.121733
Srilanka        17.830290
England          9.631221
Pakistan         8.736126
Bangladesh       6.981740
Zimbabwe         5.621196
West Indies      5.370569
Australia        3.616183
Name: proportion, dtype: float64
```

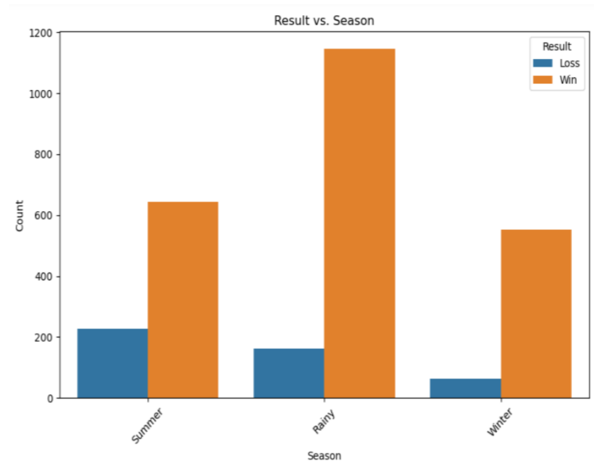
- The average team age found to be 30 years across all formats. The wicket keeper in the team is found to be 1. The maximum number of bowlers found to be 3 acorss all formats. The maximum number of scored in 1 over is 25. The maximum wicket taken per over is 4. The minimum run given per over is 6 runs. Player highest run scores is 100 runs. The mean total number of audience is 46204.
- Most of the matches was conducted in Day format (71%) , followed by Day and night format (18%) and the least was night format (10%). Indian team has played ODI format (65%) the most followed by T20 (30%) and test formats(4%). Indian team has significantly more number of wins (84%) than lose (16%) in the matches and they have opted for first bowling (61%) has their choice when they won the toss than batting (39%).
- Indian team has frequently faced with the South Africa (22%) as opponenet, followed by Kenya (20%), Srilanka (18%), England(10%), Pakistan (9%), Bangladesh(7%) , Zimbabwe (6%), West Indies(5%) and Australaia(4%). Most number of matches are played in rainy seasons (47%). Indian players played minimum matches in the off shore (28%). When the player highest wicket is high then the probablity of win in the match is also high.Player highest wicket is found to be high in Day format compared to Day & night and night format.
- Player highest run mean value by match format is found to be 66 runs in ODI and T20 format, 60 runs in the test format. The mean value of highest run is high when the opt to choose for batting first.

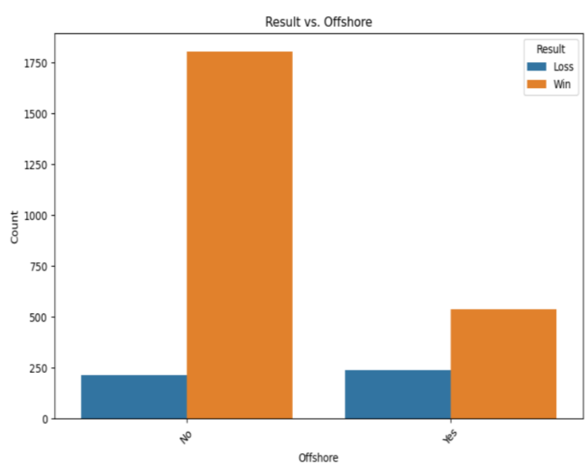
- The mean value of the maximum run given per over is close to 14 runs per over when the Indian team played against Bangladesh as an opponent and 10 runs per over when they played against West Indies as an opponent.
- The maximum wicket taken per over is found to be the least when they played against Srilanka as an opponent.

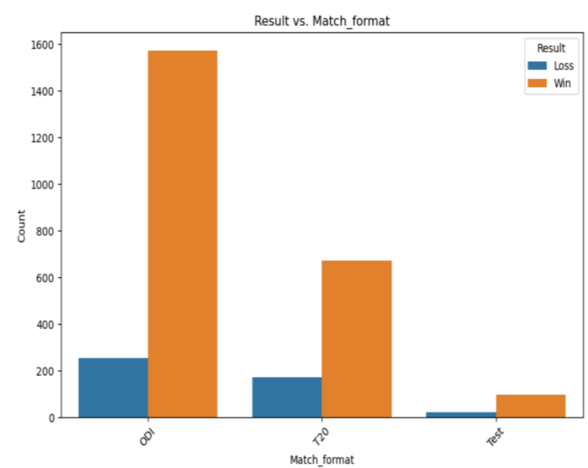
Bivariate analysis

8. Bivariate analysis count plot - Result vs opponent, Result vs season, Results vs offshore, Result vs match format










```
Descriptive Statistics for Result vs. Season:
Result       Loss        Win
Season
Rainy   12.461774  87.538226
Summer  26.176808  73.823192
Winter  10.097720  89.902280
--------------------
Descriptive Statistics for Result vs. Offshore:
Result         Loss        Win
Offshore
No        10.654113  89.345887
Yes       30.709677  69.290323
--------------------
```

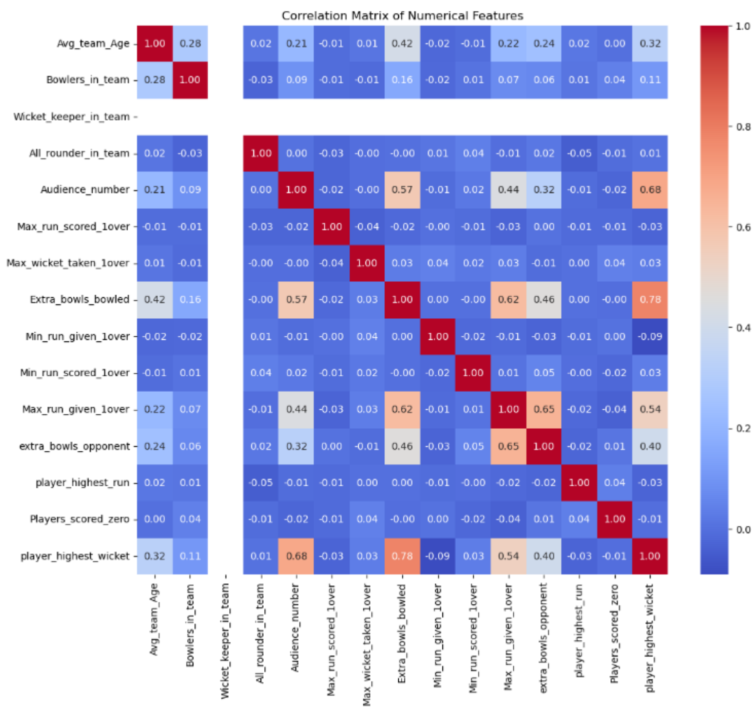
```
Descriptive Statistics for Result vs. Opponent:
Result             Loss        Win
Opponent
Australia     23.762376  76.237624
Bangladesh     5.128205  94.871795
England        6.319703  93.680297
Kenya         16.192171  83.807829
Pakistan       6.967213  93.032787
South Africa  17.504052  82.495948
Srilanka      24.297189  75.702811
West Indies    2.666667  97.333333
Zimbabwe      38.853503  61.146497
--------------------
```

- The Indian team has won 89% of matches played in India and 69% offshore.

- When the Indian team decided to bat first, they won 85% of matches and lost 15% of matches.
- The team has a win record in different formats: 85% of matches were in ODI format, 79.5% of matches were in T20 format, and 80.3% of matches were in Test format.
- They have won 73.8% of matches by playing in the summer season, 87.5% of matches by the rainy season, and 89.9% of matches by playing in the winter season.
- The Indian team has the highest success rate when playing against the West Indies, with 97.3%. They have played the most matches against South Africa, with a success rate of 75.7%, and they have lost 24.3% of matches against the same opponent.

Multivariate analysis

9. Multivariate Analysis



- We can see multicollinearity among the variables in the heatmap.
- There is a high correlation between player's highest wicket and Extra bowls bowled.
- There are also a few negative correlations found between the player's highest run and the all-rounder in the team.

Outlier Treatment

- I have Dropped rows where Avg_team_Age is greater than 50, the reason is the Indian players' average team age greater than 50 seems unrealistic and the players will not be fit at this age to play the match, and the age is too old. Thus, I have dropped the rows.
- Drop rows with null values in 'Opponent' columns, since we are unaware of the opponent data, and imputing the values doesn't get the right data while strategizing against the opponent.
- For other variables outliers seem to be a real value.

Variable transformation

- The columns such as Players_scored_zero, and player_highest_wicket have object datatype due to string values present in them, has been replaced with the value and converted into integer datatype. Converted columns into integer format such as All_rounder_in_team, Max_run_scored_1over, Extra_bowls_bowled, Min_run_scored_1over, Max_run_given_1over.

Label Encoding & Feature selection

- Converted categorical variables to numerical variables using a label encoder.
- Feature selection was done using SelectKBest with ANOVA F-value and also variance for each variable is calculated. Dropped the variables Wicket_keeper_in_team and Audience_number for further model building.
- To predict the categorical output to predict the result and having numerical input features, I have used SelectKBest with ANOVA F-value for feature selection and also calculated the variance for each numerical feature.
- Dropped the 'Game number' column, it represents only the game number of matches.
- Dropped the column 'Wicket keeper in team' from the inference of selectKBest feature. After dropping the 'Audience_number' column the accuracy in the logistic regression has increased. Thus, both variables have been eliminated for the train and test model.

Label Encoding for categorical features

```
<class 'pandas.core.frame.DataFrame'>
Index: 2793 entries, 0 to 2929
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Game_number            2793 non-null   int32
 1   Result                 2793 non-null   int32
 2   Avg_team_Age           2793 non-null   float64
 3   Match_light_type       2793 non-null   int32
 4   Match_format           2793 non-null   int32
 5   Bowlers_in_team        2793 non-null   int32
 6   Wicket_keeper_in_team  2793 non-null   int32
 7   All_rounder_in_team    2793 non-null   int32
 8   First_selection        2793 non-null   int32
 9   Opponent               2793 non-null   int32
 10  Season                 2793 non-null   int32
 11  Audience_number        2793 non-null   float64
 12  Offshore               2793 non-null   int32
 13  Max_run_scored_1over   2793 non-null   int32
 14  Max_wicket_taken_1over 2793 non-null   int32
 15  Extra_bowls_bowled     2793 non-null   int32
 16  Min_run_given_1over    2793 non-null   float64
 17  Min_run_scored_1over   2793 non-null   int32
 18  Max_run_given_1over    2793 non-null   int32
 19  extra_bowls_opponent   2793 non-null   int32
 20  player_highest_run     2793 non-null   float64
 21  Players_scored_zero    2793 non-null   float64
 22  player_highest_wicket  2793 non-null   float64
dtypes: float64(6), int32(17)
memory usage: 338.2 KB
None
```

The output of SelectKBest with ANOVA F-value for feature selection & variance of each numerical feature:

```
Out[41]: Wicket_keeper_in_team     0.000000e+00
         Result                    1.359339e-01
         Offshore                  2.005564e-01
         First_selection           2.381211e-01
         Match_format              3.252775e-01
         Match_light_type          4.376247e-01
         Min_run_scored_1over      4.945779e-01
         Players_scored_zero       5.006543e-01
         Season                    6.266318e-01
         Bowlers_in_team           1.045809e+00
         Max_wicket_taken_1over    1.170072e+00
         All_rounder_in_team       1.177535e+00
         player_highest_wicket     1.226977e+00
         Min_run_given_1over       2.840688e+00
         Avg_team_Age              3.278739e+00
         Opponent                  4.065274e+00
         extra_bowls_opponent      1.311546e+01
         Max_run_scored_1over      1.337258e+01
         Max_run_given_1over       2.478592e+01
         Extra_bowls_bowled        6.046074e+01
         player_highest_run        4.082346e+02
         Audience_number           2.377921e+09
         dtype: float64
```

Out[40]:

	Feature	Score	P-value
10	Offshore	176.079056	5.157853e-39
0	Avg_team_Age	99.935111	3.839595e-23
13	Extra_bowls_bowled	88.962239	8.183609e-21
20	player_highest_wicket	82.418758	2.028842e-19
17	extra_bowls_opponent	73.853930	1.377311e-17
19	Players_scored_zero	58.339190	3.010627e-14
7	Opponent	43.583898	4.843638e-11
9	Audience_number	36.467336	1.758309e-09
14	Min_run_given_1over	27.473737	1.711304e-07
5	All_rounder_in_team	24.496410	7.887364e-07
16	Max_run_given_1over	17.821340	2.503989e-05
2	Match_format	16.153839	5.994500e-05
15	Min_run_scored_1over	12.286666	4.633906e-04
12	Max_wicket_taken_1over	5.862136	1.553339e-02
3	Bowlers_in_team	3.305835	6.914142e-02
6	First_selection	1.856399	1.731509e-01
11	Max_run_scored_1over	0.983172	3.215030e-01
8	Season	0.561880	4.535669e-01
18	player_highest_run	0.119778	7.293012e-01
1	Match_light_type	0.000880	9.763360e-01
4	Wicket_keeper_in_team	NaN	NaN

Variables
Game_number
Result
Avg_team_Age
Match_light_type
Match_format
Bowlers_in_team
Wicket_keeper_in_team
All_rounder_in_team
First_selection
Opponent
Season
Audience_number
Offshore
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

Eliminated features

After post-processing and cleaning of data, the data is balanced with 2793 rows and 23 columns.

Train – Test split:

The train–test split was done with a ratio of 70 percent of the train dataset and 30 percent of the test dataset.

```
(1955, 21) (838, 21) (1955,) (838,)
```

Machine Learning Models:

The target variables is 'Result'.

The various classification model was used to predict the categorical variables 'Result' are:

- Logistic Regression
- Linear Discriminant Analysis (LDA)
- KNN
- Decision Tree
- Bagging
- Random Forest
- Support Vector Machine (SVM)
- AdaBoost
- Gradient Boost

Logistic Regression:

```
Accuracy: 0.8400954653937948
              precision    recall  f1-score   support

           0       0.54      0.10      0.17       136
           1       0.85      0.98      0.91       702

    accuracy                           0.84       838
   macro avg       0.69      0.54      0.54       838
weighted avg       0.80      0.84      0.79       838

Confusion Matrix:
```

```
Out[43]: array([[ 14, 122],
                [ 12, 690]], dtype=int64)
```

Dropped Audience number column due to high variance, to check the model's performance.

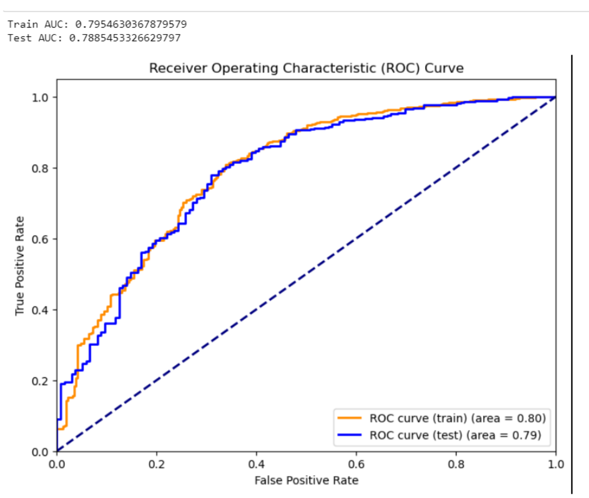
```
Accuracy: 0.8544152744630071
              precision    recall  f1-score   support

           0       0.69      0.18      0.29       136
           1       0.86      0.98      0.92       702

    accuracy                           0.85       838
   macro avg       0.78      0.58      0.60       838
weighted avg       0.83      0.85      0.82       838
```

Confusion matrix

```
Out[44]: array([[ 25, 111],
                [ 11, 691]], dtype=int64)
```

The accuracy of the model slightly increased from 0.84 to 0.854.

10. Logistic Regression ROC curve

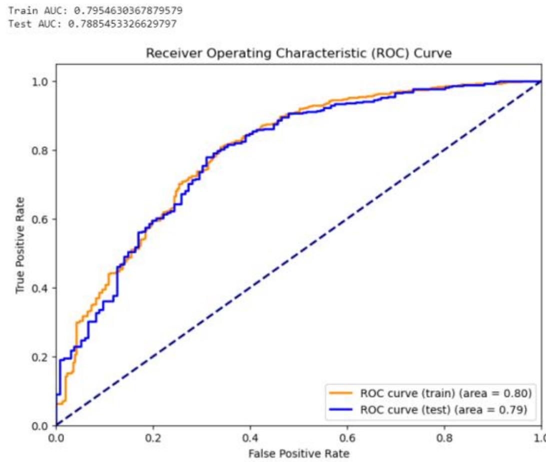


Dropped wicketkeeper in team from the model, to check the model prediction.

```
Accuracy: 0.8544152744630071
              precision    recall  f1-score   support

           0       0.69      0.18      0.29       136
           1       0.86      0.98      0.92       702

    accuracy                           0.85       838
   macro avg       0.78      0.58      0.60       838
weighted avg       0.83      0.85      0.82       838
```

Confusion matrix

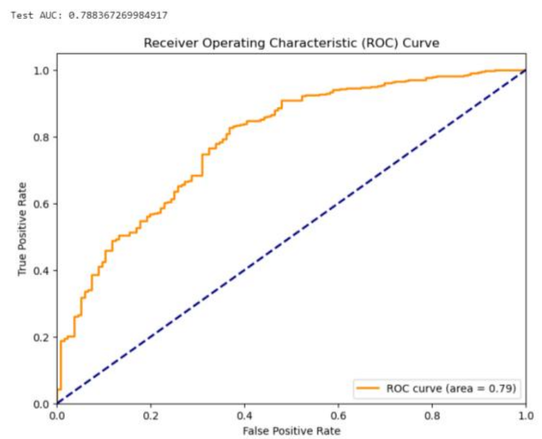
```
Out[48]: array([[ 25, 111],
                [ 11, 691]], dtype=int64)
```



Linear Discriminant Analysis

The output of LDA test result – classification report, confusion matrix, accuracy and roc curve

```
Accuracy: 0.850835322195704
              precision    recall  f1-score   support

           0       0.60      0.24      0.34       136
           1       0.87      0.97      0.92       702

    accuracy                           0.85       838
   macro avg       0.74      0.60      0.63       838
weighted avg       0.82      0.85      0.82       838

Confusion Matrix:
Out[51]: array([[ 32, 104],
                [ 21, 681]], dtype=int64)
```

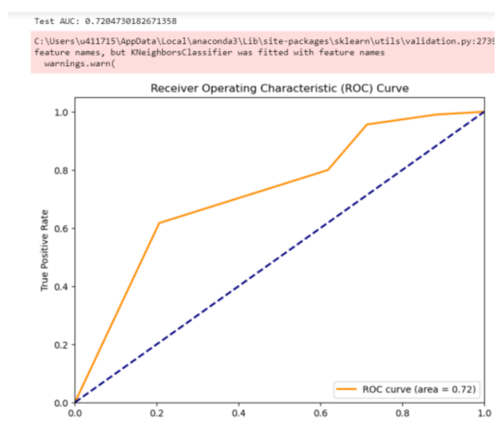


11. Linear Discriminant Analysis ROC curve

KNeighborsClassifier

The output of Kneighbors classifier test result – classification report, confusion matrix, accuracy, and roc curve.

```
Accuracy: 0.847255369928401
              precision    recall  f1-score   support

           0       0.56      0.29      0.38       136
           1       0.87      0.96      0.91       702

    accuracy                           0.85       838
   macro avg       0.72      0.62      0.65       838
weighted avg       0.82      0.85      0.83       838

Confusion Matrix:
Out[54]: array([[ 39,  97],
                [ 31, 671]], dtype=int64)
```

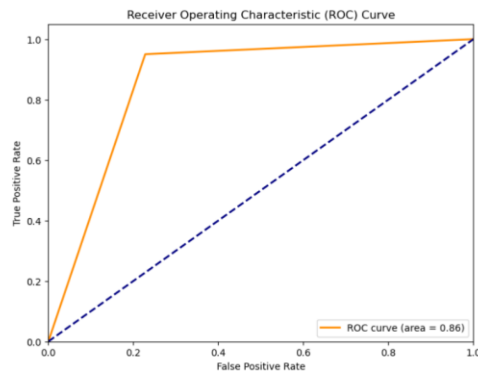


12. KNeighborsClassifier ROC curve

Decision Tree classifier

The output of Decision tree classifier test result – classification report, confusion matrix, accuracy, and roc curve

```
Accuracy: 0.9212410501193318
              precision    recall  f1-score   support

           0       0.75      0.77      0.76       136
           1       0.96      0.95      0.95       702

    accuracy                           0.92       838
   macro avg       0.85      0.86      0.86       838
weighted avg       0.92      0.92      0.92       838

Confusion Matrix:
[[105  31]
 [ 35 667]]
Test AUC: 0.861100636835931
```



13. Decision Tree classifier ROC curve

Gaussian Naïve Bayes Model

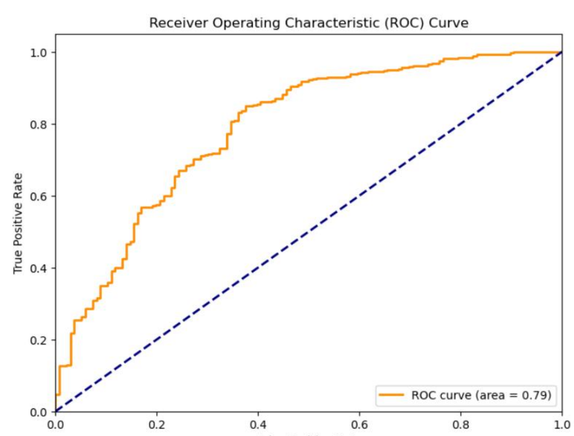
The output of Naïve bayes test result – classification report, confusion matrix, accuracy, and roc curve

```
Accuracy: 0.8233890214797136
              precision    recall  f1-score   support

           0       0.46      0.49      0.47       136
           1       0.90      0.89      0.89       702

    accuracy                           0.82       838
   macro avg       0.68      0.69      0.68       838
weighted avg       0.83      0.82      0.83       838

Confusion Matrix:

Out[60]: array([[ 66,  70],
       [ 78, 624]], dtype=int64)
```

Support Vector Classifier model

The output of support vector classifier test result – classification report, confusion matrix, accuracy, and roc curve

```
Accuracy: 0.837708830548926
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       136
           1       0.84      1.00      0.91       702

    accuracy                           0.84       838
   macro avg       0.42      0.50      0.46       838
weighted avg       0.70      0.84      0.76       838

Confusion Matrix:
[[  0 136]
 [  0 702]]
Test AUC: 0.7876550192726663
```



14. Support vector classifier ROC curve

Model Tuning and business implication

Used Grid search CV model tuning on logistic regression model

```
{'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Best parameters: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Best cross-validation score: 0.8526854219948848
Test accuracy of the best model: 0.8520286396181385
```

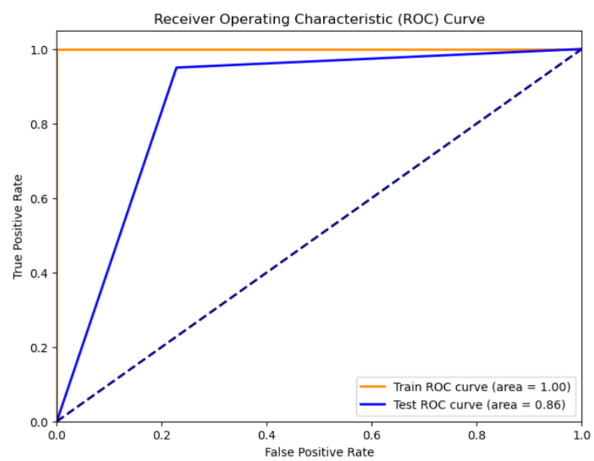
Using Grid search CV model tuning on random forest

```
Best Hyperparameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Best Random Forest Model Accuracy: 0.9498806682577565
```

Decision Tree Classifier model with Gini criterion

The output of Decision tree classifier with Gini criterion both train & test result – classification report, confusion matrix, accuracy and roc curve

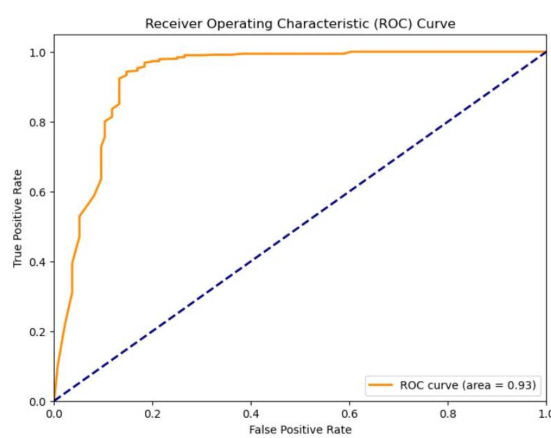
```
Training Accuracy: 1.0
Training Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       317
           1       1.00      1.00      1.00      1638

    accuracy                           1.00      1955
   macro avg       1.00      1.00      1.00      1955
weighted avg       1.00      1.00      1.00      1955

Training Confusion Matrix:
[[ 317    0]
 [   0 1638]]
Test Accuracy: 0.9212410501193318
Test Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.77      0.76       136
           1       0.96      0.95      0.95       702

    accuracy                           0.92       838
   macro avg       0.85      0.86      0.86       838
weighted avg       0.92      0.92      0.92       838

Test Confusion Matrix:
[[105  31]
 [ 35 667]]
```



Bagging Classifier

The output of Bagging classifier test result – classification report, confusion matrix, accuracy and roc curve

```
Bagging Model Accuracy: 0.9463007159904535
              precision    recall  f1-score   support

           0       0.90      0.75      0.82       136
           1       0.95      0.98      0.97       702

    accuracy                           0.95       838
   macro avg       0.93      0.87      0.89       838
weighted avg       0.94      0.95      0.94       838

Confusion Matrix:
[[102  34]
 [ 11 691]]
Test AUC: 0.9287173202614378
```

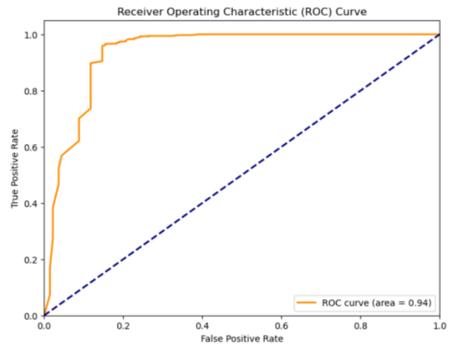


15. Bagging classifier ROC curve

Random Forest classifier

The output of Random Forest classifier test result – classification report, confusion matrix, accuracy and roc curve

```
Accuracy: 0.9474940334128878
              precision    recall  f1-score   support

           0       0.96      0.71      0.81       136
           1       0.95      0.99      0.97       702

    accuracy                           0.95       838
   macro avg       0.95      0.85      0.89       838
weighted avg       0.95      0.95      0.94       838

Confusion Matrix:
[[ 96  40]
 [  4 698]]
Test AUC: 0.9361278699513993
```

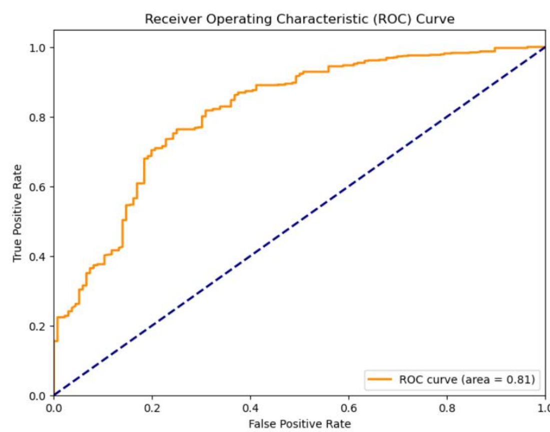


16. Random Forest classifier ROC curve

AdaBoost Classifier

Adaptive Boosting, is an ensemble learning technique that combines multiple weak classifiers to form a strong classifier. The main idea behind AdaBoost is to iteratively train weak classifiers on the training data, adjusting the weights of misclassified instances to focus more on difficult cases. The output of Adaboost classifier test result – classification report, confusion matrix, accuracy, and roc curve

```
Accuracy: 0.8615751789976134
              precision    recall  f1-score   support

           0       0.66      0.31      0.42       136
           1       0.88      0.97      0.92       702

    accuracy                           0.86       838
   macro avg       0.77      0.64      0.67       838
weighted avg       0.84      0.86      0.84       838

Confusion Matrix:
[[ 42  94]
 [ 22 680]]
Test AUC: 0.8136521702698174
```

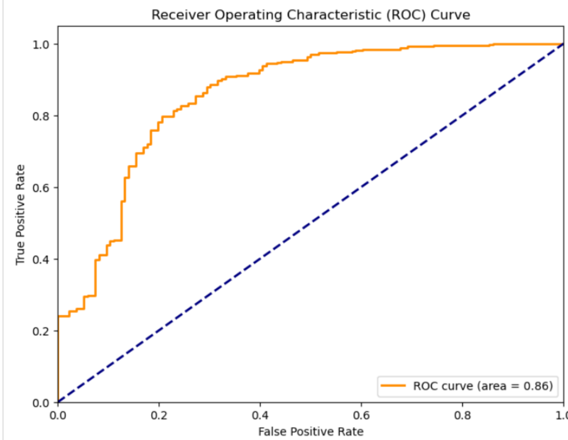


17. Adaboost classifier ROC curve

Gradient Boosting classifier

Gradient boosting is one of the most effective techniques for building machine learning models. It is based on the idea of improving the weak learners (learners with insufficient predictive power). The output of Gradient boosting classifier test result – classification report, confusion matrix, accuracy and roc curve

```
Accuracy: 0.8878281622911695
              precision    recall  f1-score   support

           0       0.81      0.40      0.54       136
           1       0.89      0.98      0.94       702

    accuracy                           0.89       838
   macro avg       0.85      0.69      0.74       838
weighted avg       0.88      0.89      0.87       838

Confusion Matrix:
[[ 55  81]
 [ 13 689]]
Test AUC: 0.8561986760516171
```



18. Gradient Booting classifier ROC curve

Training Model

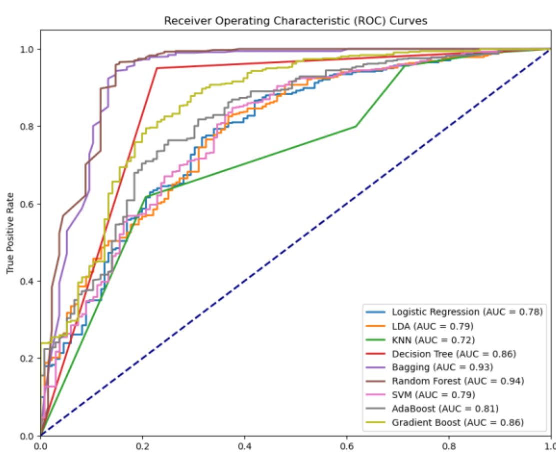
```
--- Logistic Regression ---
Confusion Matrix:
[[  58  259]
 [  22 1616]]
Precision: 0.8618666666666667
F1 Score: 0.9200113862795332
Recall: 0.9865689865689866
Accuracy: 0.8562659846547315
------------------------------
--- LDA ---
Confusion Matrix:
[[  81  236]
 [  45 1593]]
Precision: 0.8709677419354839
F1 Score: 0.9189501009518316
Recall: 0.9725274725274725
Accuracy: 0.8562659846547315
------------------------------
--- KNN ---
Confusion Matrix:
[[ 113  204]
 [  37 1601]]
Precision: 0.8869806094182825
F1 Score: 0.930002904443799
Recall: 0.9774114774114774
Accuracy: 0.8767263427109975
------------------------------
--- Decision Tree ---
Confusion Matrix:
[[ 317    0]
 [   0 1638]]
Precision: 1.0
F1 Score: 1.0
Recall: 1.0
Accuracy: 1.0
------------------------------
```

```
------------------------------
--- Bagging ---
Confusion Matrix:
[[ 317    0]
 [   0 1638]]
Precision: 1.0
F1 Score: 1.0
Recall: 1.0
Accuracy: 1.0
------------------------------
--- Random Forest ---
Confusion Matrix:
[[ 317    0]
 [   0 1638]]
Precision: 1.0
F1 Score: 1.0
Recall: 1.0
Accuracy: 1.0
------------------------------
--- SVM ---
Confusion Matrix:
[[   0  317]
 [   0 1638]]
Precision: 0.837851662404092
F1 Score: 0.9117728917339271
Recall: 1.0
Accuracy: 0.837851662404092
```

```
--- AdaBoost ---
Confusion Matrix:
[[  93  224]
 [  34 1604]]
Precision: 0.8774617067833698
F1 Score: 0.9255626081938835
Recall: 0.9792429792429792
Accuracy: 0.8680306905370844
------------------------------
--- Gradient Boost ---
Confusion Matrix:
[[ 179  138]
 [  12 1626]]
Precision: 0.9217687074829932
F1 Score: 0.9559082892416225
Recall: 0.9926739926739927
Accuracy: 0.9232736572890026
------------------------------
```

Testing Model

```
Evaluation for Logistic Regression:
Confusion Matrix:
 [[ 19 117]
 [  7 695]]
Precision: 0.8559113300492611
F1 Score: 0.9180977542932629
Recall: 0.99002849002849
Accuracy: 0.8520286396181385
------------------------------
Evaluation for LDA:
Confusion Matrix:
 [[ 32 104]
 [ 21 681]]
Precision: 0.867515923566879
F1 Score: 0.9159381304640215
Recall: 0.9700854700854701
Accuracy: 0.850835322195704
------------------------------
Evaluation for KNN:
Confusion Matrix:
 [[ 39  97]
 [ 31 671]]
Precision: 0.8736979166666666
F1 Score: 0.9129251700680272
Recall: 0.9558404558404558
Accuracy: 0.847255369928401
------------------------------
Evaluation for Decision Tree:
Confusion Matrix:
 [[105  31]
 [ 35 667]]
Precision: 0.9555873925501432
F1 Score: 0.9528571428571428
Recall: 0.9501424501424501
Accuracy: 0.9212410501193318
------------------------------
Evaluation for Bagging:
Confusion Matrix:
 [[102  34]
 [ 11 691]]
Precision: 0.953103448275862
F1 Score: 0.9684653118430273
Recall: 0.9843304843304843
Accuracy: 0.9463007159904535
------------------------------
Evaluation for Random Forest:
Confusion Matrix:
 [[ 96  40]
 [  4 698]]
Precision: 0.94579945799458
F1 Score: 0.9694444444444444
Recall: 0.9943019943019943
Accuracy: 0.9474940334128878
------------------------------
Evaluation for SVM:
Confusion Matrix:
 [[  0 136]
 [  0 702]]
Precision: 0.837708830548926
F1 Score: 0.9116883116883117
Recall: 1.0
Accuracy: 0.837708830548926
------------------------------
Evaluation for AdaBoost:
Confusion Matrix:
 [[ 42  94]
 [ 22 680]]
Precision: 0.8785529715762274
F1 Score: 0.9214092140921409
Recall: 0.9686609686609686
Accuracy: 0.8615751789976134
------------------------------
Evaluation for Gradient Boost:
Confusion Matrix:
 [[ 55  81]
 [ 13 689]]
Precision: 0.8948051948051948
F1 Score: 0.936141304347826
Recall: 0.9814814814814815
Accuracy: 0.8878281622911695
------------------------------
```



19. Machine learning models - ROC curve

Interpretation of the model

Train/ Test Dataset	Models	Precision	F1 Score	Recall	Accuracy
Train Dataset	Logistic Regression	0.86	0.92	0.99	0.86
Test Dataset	Logistic Regression	0.86	0.92	0.99	0.85
Train Dataset	LDA	0.87	0.92	0.97	0.86
Test Dataset	LDA	0.87	0.92	0.97	0.85
Train Dataset	KNN	0.89	0.93	0.98	0.88
Test Dataset	KNN	0.87	0.91	0.96	0.85
Train Dataset	Decision Tree	1.00	1.00	1.00	1.00
Test Dataset	Decision Tree	0.96	0.95	0.95	0.92
Train Dataset	Bagging	1.00	1.00	1.00	1.00
Test Dataset	Bagging	0.95	0.97	0.98	0.95
Train Dataset	Random Forest	1.00	1.00	1.00	1.00
Test Dataset	Random Forest	0.95	0.97	0.99	0.95
Train Dataset	SVM	0.84	0.91	1.00	0.84
Test Dataset	SVM	0.84	0.91	1.00	0.84
Train Dataset	AdaBoost	0.88	0.93	0.93	0.87
Test Dataset	AdaBoost	0.88	0.92	0.97	0.86
Train Dataset	Gradient Boost	0.92	0.96	0.99	0.92
Test Dataset	Gradient Boost	0.89	0.94	0.98	0.89

20. Interpretation of the models

Test AUC Score:

	Logistic Regression	LDA	KNN	Decision Tree	Bagging	Random Forest	SVM	AdaBoost	Gradient Boost
AUC	0.78	0.79	0.72	0.86	0.93	0.94	0.79	0.81	0.86

- In this classification problem the most important measurement matrix we see is Recall, precision, accuracy, and F1-Score. Precision is the total predicted win and loss. Recall is total Actually win and loss.
- F1- score is the harmonic mean of precision and recall. In this case our most important matrix is Recall because we must predict winning for the Indian team and must reduce the false positive rate.
- Consolidating all models, I am going to use the 'Random Forest Model' for prediction.
- Random Forest is an ensemble technique. It has less False positive and False negative for both win and loss Classes. Compare to other model it has Higher Precision, Recall and Accuracy for both Train and Test.

Recommendation

1 Test match with England in England. All the match are day matches. In England, it will be rainy season at the time to match.

Strategy 1	
'Bowlers_in_team'	3
'All_rounder_in_team'	4
'Wicket_keeper_in_team'	1
'Avg_team_Age'	30
'First_selection'	Batting

'Max_run_scored_1over'	18
'Max_wicket_taken_1over'	2
'Extra_bowls_bowled'	20
'Min_run_given_1over'	6
'Min_run_scored_1over'	10
'Max_run_given_1over'	16
'extra_bowls_opponent'	4
'player_highest_run'	150
'Players_scored_zero'	2
'player_highest_wicket'	4
Probability of India Win in Test match Vs England in England	**85.00%**
Probability of India Loss in Test match Vs England in England	**15.00%**

Bowlers in Team: Select bowlers in team should have minimum of 3 bowlers with good economy rate.
All-rounder in Team: The all-rounder in team should have minimum of 4 players for rotation of players.
Players scored Zero: With the help of bowlers and all-rounders in team try to get 2 early wickets to pressurize the opponent team.
Player highest run: The opening batsman and partnership is crucial and getting 150 runs will increase the Indian total runs.

2 T20 match with Australia in India. All the matches are Day and Night matches. In India, it will be winter season at the time to match.

Strategy 1		Strategy 2	
'Bowlers_in_team'	3	'Bowlers_in_team'	3
'All_rounder_in_team'	4	'All_rounder_in_team'	4
'Wicket_keeper_in_team'	1	'Wicket_keeper_in_team'	1
'Avg_team_Age'	30	'Avg_team_Age'	31
'First_selection'	Bowling	'First_selection'	Batting
'Max_run_scored_1over'	18	'Max_run_scored_1over'	20
'Max_wicket_taken_1over'	2	'Max_wicket_taken_1over'	2
'Extra_bowls_bowled'	3	'Extra_bowls_bowled'	3
'Min_run_given_1over'	6	'Min_run_given_1over'	10
'Min_run_scored_1over'	12	'Min_run_scored_1over'	8
'Max_run_given_1over'	16	'Max_run_given_1over'	12
'extra_bowls_opponent'	4	'extra_bowls_opponent'	4
'player_highest_run'	50	'player_highest_run'	60
'Players_scored_zero'	2	'Players_scored_zero'	2
'player_highest_wicket'	3	'player_highest_wicket'	4
Probability of India Win in T20 match Vs Australia in India	**92.50%**	**Probability of India Win in T20 match Vs Australia in India**	**89.50%**

Probability of India Loss in T20 match Vs Australia in India	**7.50%**	**Probability of India Loss in T20 match Vs Australia in India**	**10.50%**

Strategy 1:

Average team age: Choose players who are fit and have good records. The players' average should not be greater than 30.

Choosing the first selection as a bowling option, trying to minimize the run given per over to 6 runs, and trying to get an early wicket with a player scored zero to 2 will pressure the opponent and help the Indian team to win with the probability of 92.5%.

Strategy 2:

Choosing batting as the first option, opening partnership plays a crucial role and the player's highest run is 60 runs. The maximum run scored per over is 20 runs and the minimum run rate scored per over is 8 runs. The playing 11 teams should consist of 4 all-rounders in a team, 3 bowlers in a team, and 1 wicket-keeper in the team will help the Indian team to win with the probability of 89.5%.

2 ODI match with Sri Lanka in India. All the matches are Day and Night matches. In India, it will be winter season at the time to match.

Strategy 1		**Strategy 2**	
'Bowlers_in_team'	3	'Bowlers_in_team'	3
'All_rounder_in_team'	4	'All_rounder_in_team'	5
'Wicket_keeper_in_team'	1	'Wicket_keeper_in_team'	1
'Avg_team_Age'	30	'Avg_team_Age'	30
'First_selection'	Batting	'First_selection'	Bowling
'Max_run_scored_1over'	18	'Max_run_scored_1over'	16
'Max_wicket_taken_1over'	1	'Max_wicket_taken_1over'	3
'Extra_bowls_bowled'	20	'Extra_bowls_bowled'	10
'Min_run_given_1over'	8	'Min_run_given_1over'	5
'Min_run_scored_1over'	10	'Min_run_scored_1over'	10
'Max_run_given_1over'	16	'Max_run_given_1over'	16
'extra_bowls_opponent'	4	'extra_bowls_opponent'	14
'player_highest_run'	100	'player_highest_run'	120
'Players_scored_zero'	2	'Players_scored_zero'	2
'player_highest_wicket'	5	'player_highest_wicket'	5
Probability of India Win in ODI match Vs Sri Lanka in India	**89.00%**	**Probability of India Win in ODI match Vs Sri Lanka in India**	**94.00%**
Probability of India Loss in ODI match Vs Sri Lanka in India	**11.00%**	**Probability of India Loss in ODI match Vs Sri Lanka in India**	**6.00%**

Strategy 1: Average team age: Choose players who are fit and have good records. The players' average should not be greater than 30.

Choosing the first selection as a batting option, the batsman should score 100 runs to increase the total, the minimum run scored in 1 over should be 10 runs, getting early wickets of the highest player wicket 5 will help the Indian team to win with a probability of 89%.

Strategy 2:

Choosing bowling as the first option, getting a player's highest wicket of 5 and the players scored zero of 2, the minimum run given per over with 5 runs and the minimum run rate scored per over is 10 runs. The playing 11 teams should consist of 5 all-rounders in a team, 3 bowlers in a team, and 1 wicket-keeper in the team will help the Indian team to win with a probability of 94%.

Conclusion

Strategies:

In the Indian team, selecting all-rounders of a minimum 3 members and bowlers of a minimum of 3 members will ensure the Indian team to win the matches across any format.

Test Match Strategy (With England in England, Rainy Season):

1. Prioritize a strong batting lineup: given the rainy conditions, batting first and setting up a competitive total is key.
2. Utilize all-rounders: Select all-rounders who can contribute with both bat and ball, providing balance in challenging conditions.
3. Strong opening partnership: The opening pair should aim to get a quick start to put pressure on the opponents.
4. Rotate bowlers carefully: Use the bowlers according to pitch and weather conditions.

T20 Match Strategy (With Australia in India, Winter):

1. Focus on explosive batting: given the shorter format and conducive weather, the team should focus on aggressive batting from the outset.
2. Quick wickets: Focus on taking early wickets to put pressure on opposition batting.
3. Utilize spinners: The winter season may aid spinners, so a well-rounded spin attack could be crucial.
4. Death bowling: focus on Yorkers and slower balls to restrict the run flow.
5. Agility and fielding: Maintain an extremely agile fielding strategy, crucial for run-saving and taking quick wickets.

ODI Match Strategy (With Sri Lanka in India, Winter, Day/Night):

1. Utilize Spin Attack: Favor spinners, exploiting the slower, turning pitch likely due to the winter conditions.
2. Dew Factor: Plan for the dew factor during night matches (if significant), affecting both batting and bowling.
3. Consistent batting: Ensure consistent batting throughout the innings, preventing collapses.
4. Strong death bowling: focus on accurate Yorkers to defend the total in the slog overs.
5. Analyze Sri Lanka's batting order: identify their strengths and weaknesses to tailor your bowling attack accordingly.