PROJECT REPORT
ON
CRICKET WIN PREDICTION
(NOTES – 1 & 2)

By SHAJIL FERNANDEZ

05-05-2024

Table of Contents:

Project Notes - 1

List of Figures:

List of Tables:

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:

To extract actionable insights from the historical match data and create Machine Learning models which correctly predicts a win for the Indian Cricket Team.

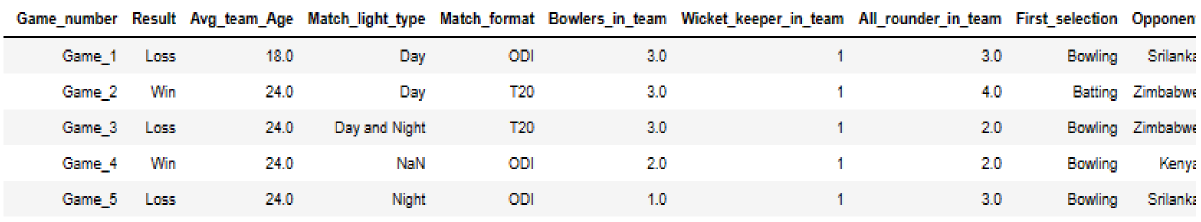
Business understanding:

Cricket is the most popular and most played sport in India. The BCCI is the governing body of Indian cricket, responsible for strategizing to win matches and selecting players to compete across different formats of the game. There are various factors such as weather conditions, match format, match light type, first selection, season etc. that needs to be considered while selecting players to ensure a win. So, based on historical data, machine learning models can be implemented to accurately predict a win for the Indian Cricket Team by considering all the given variables.

<u>Data report:</u>

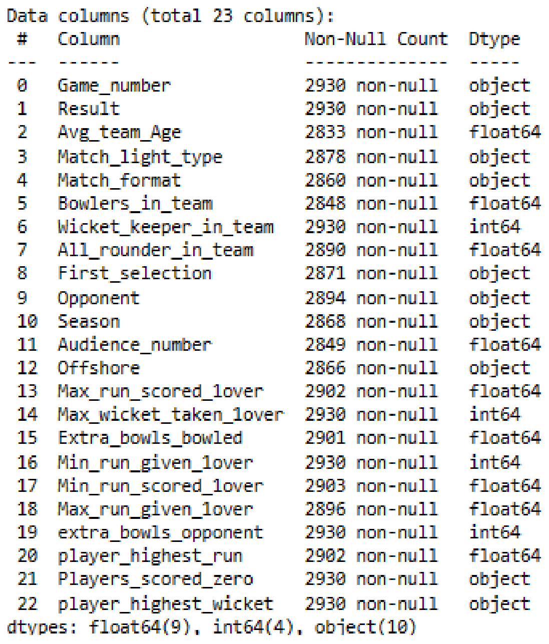
- ### <u>Basic info:</u>

Table 1: Top 5 rows of the dataset



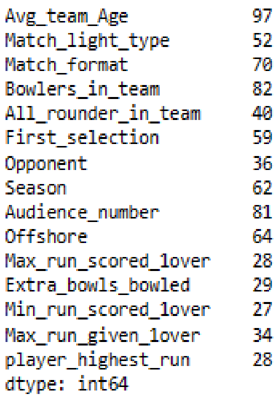
Game_number	Result	Avg_team_Age	Match_light_type	Match_format	Bowlers_in_team	Wicket_keeper_in_team	All_rounder_in_team	First_selection	Opponen
Game_1	Loss	18.0	Day	ODI	3.0	1	3.0	Bowling	Srilank
Game_2	Win	24.0	Day	T20	3.0	1	4.0	Batting	Zimbabw
Game_3	Loss	24.0	Day and Night	T20	3.0	1	2.0	Bowling	Zimbabw
Game_4	Win	24.0	NaN	ODI	2.0	1	2.0	Bowling	Keny
Game_5	Loss	24.0	Night	ODI	1.0	1	3.0	Bowling	Srilank

Table 2: Basic info of the dataset



```
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)
```

Table 3: Null values



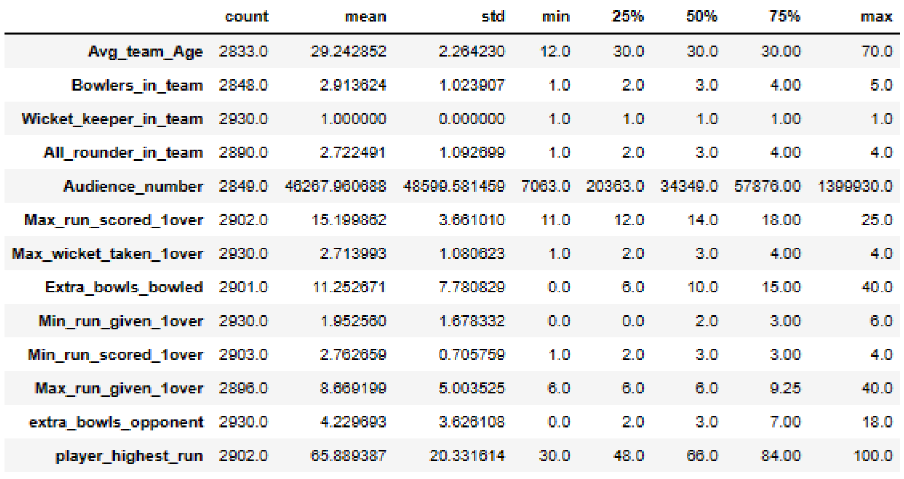
```
Avg_team_Age            97
Match_light_type        52
Match_format            70
Bowlers_in_team         82
All_rounder_in_team     40
First_selection         59
Opponent                36
Season                  62
Audience_number         81
Offshore                64
Max_run_scored_1over    28
Extra_bowls_bowled      29
Min_run_scored_1over    27
Max_run_given_1over     34
player_highest_run      28
dtype: int64
```

Insights:

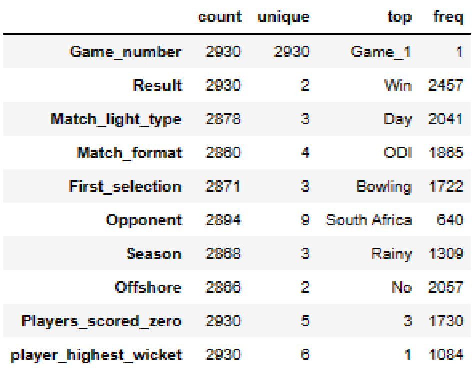
- There are 2930 rows and 23 columns in the given data.
- There are 9 float64, 4 int64 and 10 object datatypes.
- There are no duplicates values.
- There are 15 variables with missing values in the given data.

- **<u>Statistical summary of numerical and categorical data:</u>**

Table 4: Statistical summary of numerical and categorical data



	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



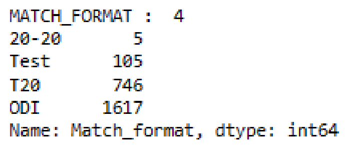
	count	unique	top	freq
Game_number	2930	2930	Game_1	1
Result	2930	2	Win	2457
Match_light_type	2878	3	Day	2041
Match_format	2860	4	ODI	1865
First_selection	2871	3	Bowling	1722
Opponent	2894	9	South Africa	640
Season	2868	3	Rainy	1309
Offshore	2866	2	No	2057
Players_scored_zero	2930	5	3	1730
player_highest_wicket	2930	6	1	1084

Insights:

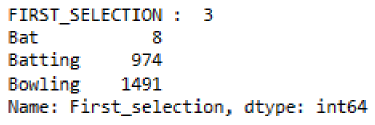
- We can observe that the maximum number of all-rounders played in a team is 4.
- The maximum number of wickets taken in an over is 4.
- About 2457 out of 2930 matches were won.
- Maximum number of matches were played against South Africa.

- **<u>Replacing incorrect inputs:</u>**

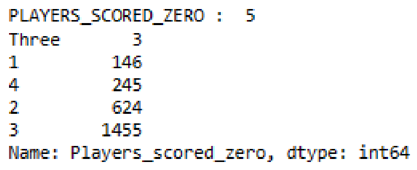
Table 5: Incorrect inputs



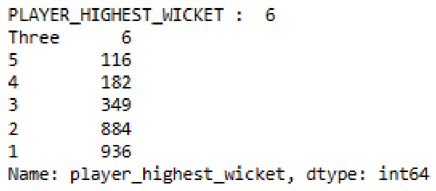
```
MATCH_FORMAT :  4
20-20       5
Test      105
T20       746
ODI      1617
Name: Match_format, dtype: int64
```



```
FIRST_SELECTION :  3
Bat            8
Batting      974
Bowling     1491
Name: First_selection, dtype: int64
```



```
PLAYERS_SCORED_ZERO :  5
Three        3
1          146
4          245
2          624
3         1455
Name: Players_scored_zero, dtype: int64
```



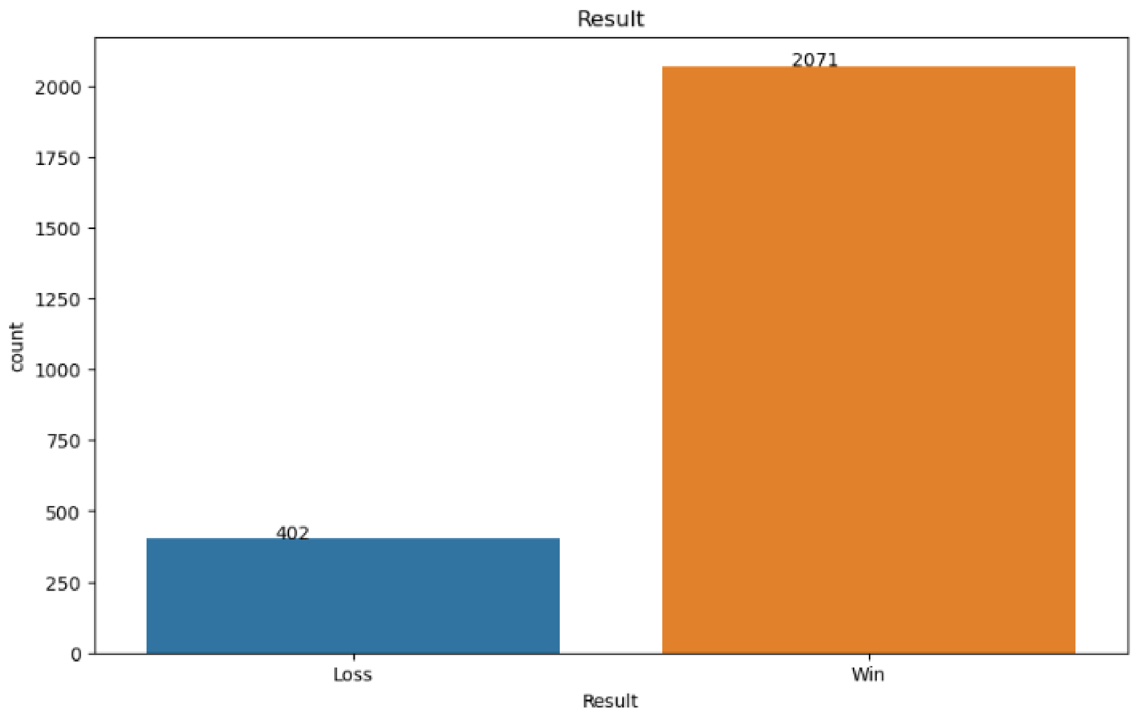
```
PLAYER_HIGHEST_WICKET :  6
Three      6
5        116
4        182
3        349
2        884
1        936
Name: player_highest_wicket, dtype: int64
```

- **We are replacing incorrect inputs with valid ones to avoid faulty analysis.**

<u>Exploratory Data Analysis:</u>

i) **<u>Result</u>**:

Figure 1: Result

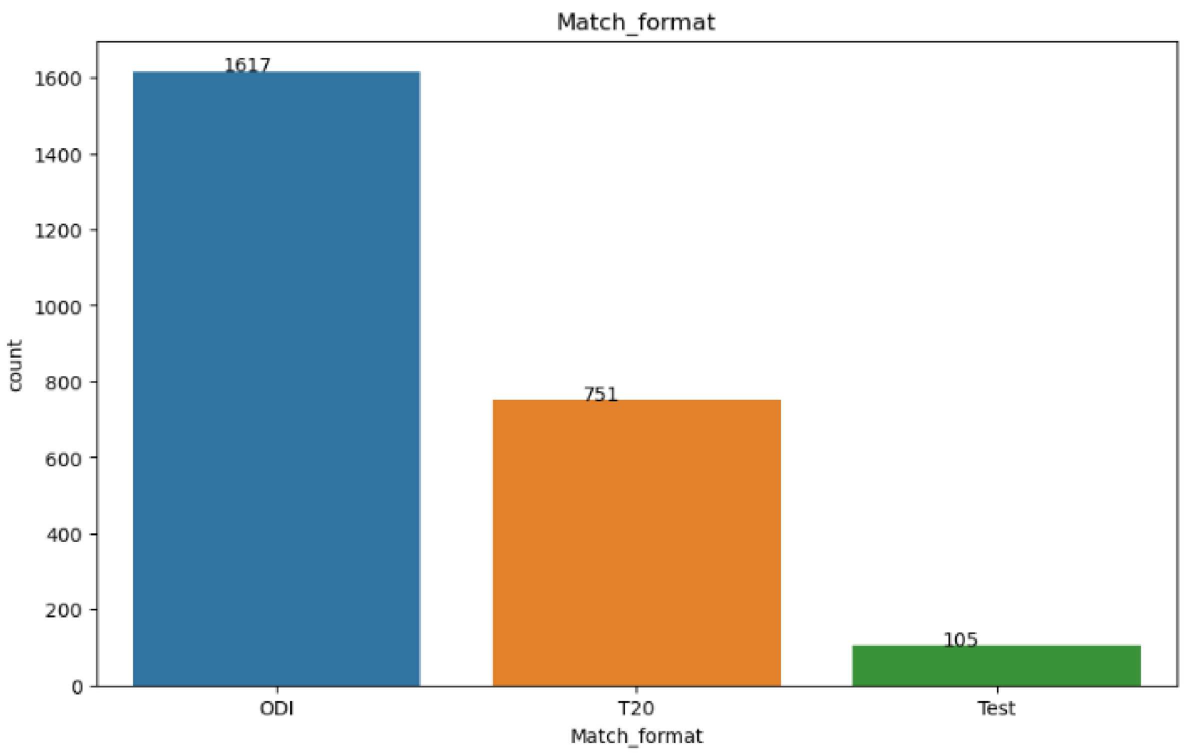


Insight:

- Of 2473 matches, 2071 resulted in wins, while 402 ended in losses.

ii) **<u>Match format</u>**:

Figure 2: Match_format

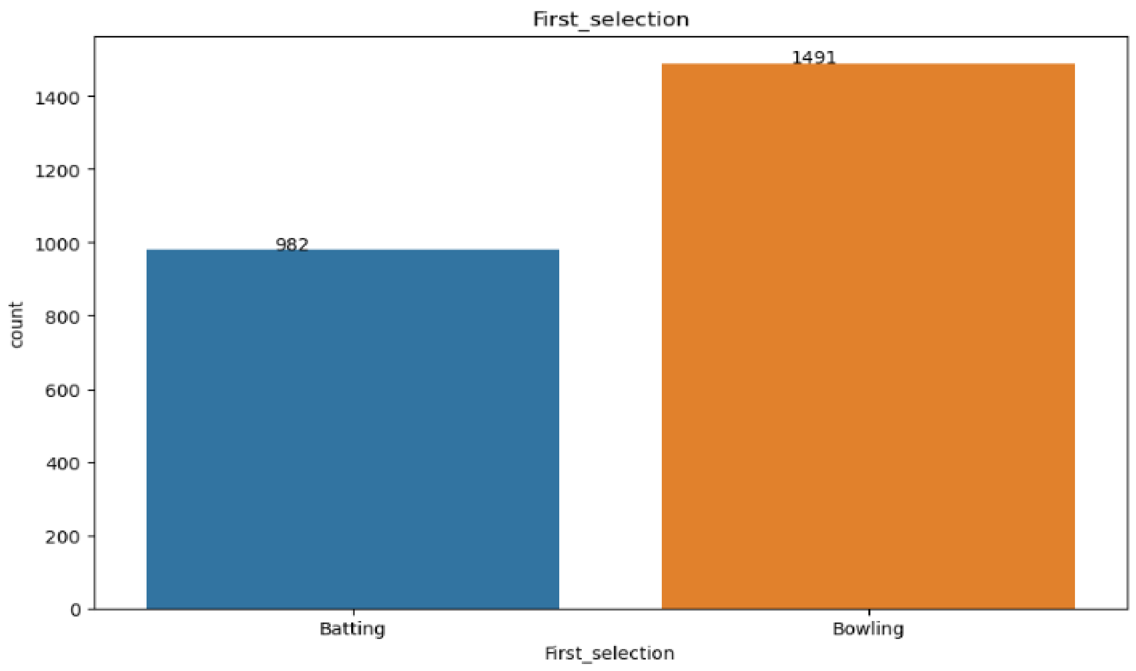


Insight:

- Out of 2473 matches, 1617 are ODI's, 751 are of T20 format and 105 are test matches.

iii) **<u>First selection</u>**:

Figure 3: First_selection

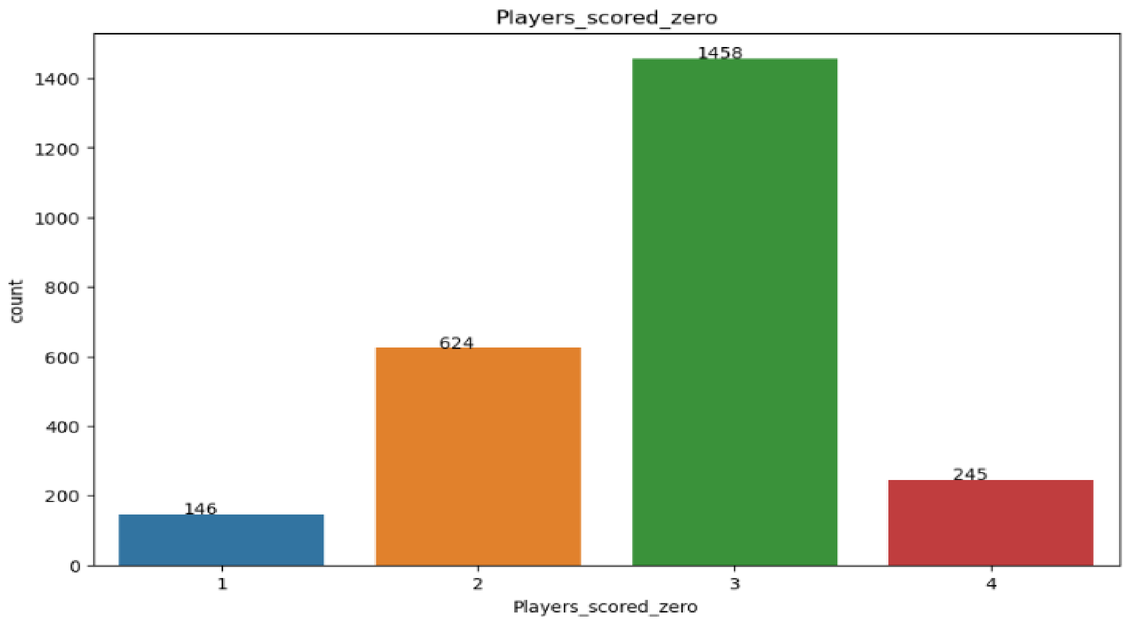


Insight:

- The Indian team has opted to bowl first 1491 times and bat first 982 times.

iv) **<u>Players scored zero</u>**:

Figure 4: Players_scored_zero



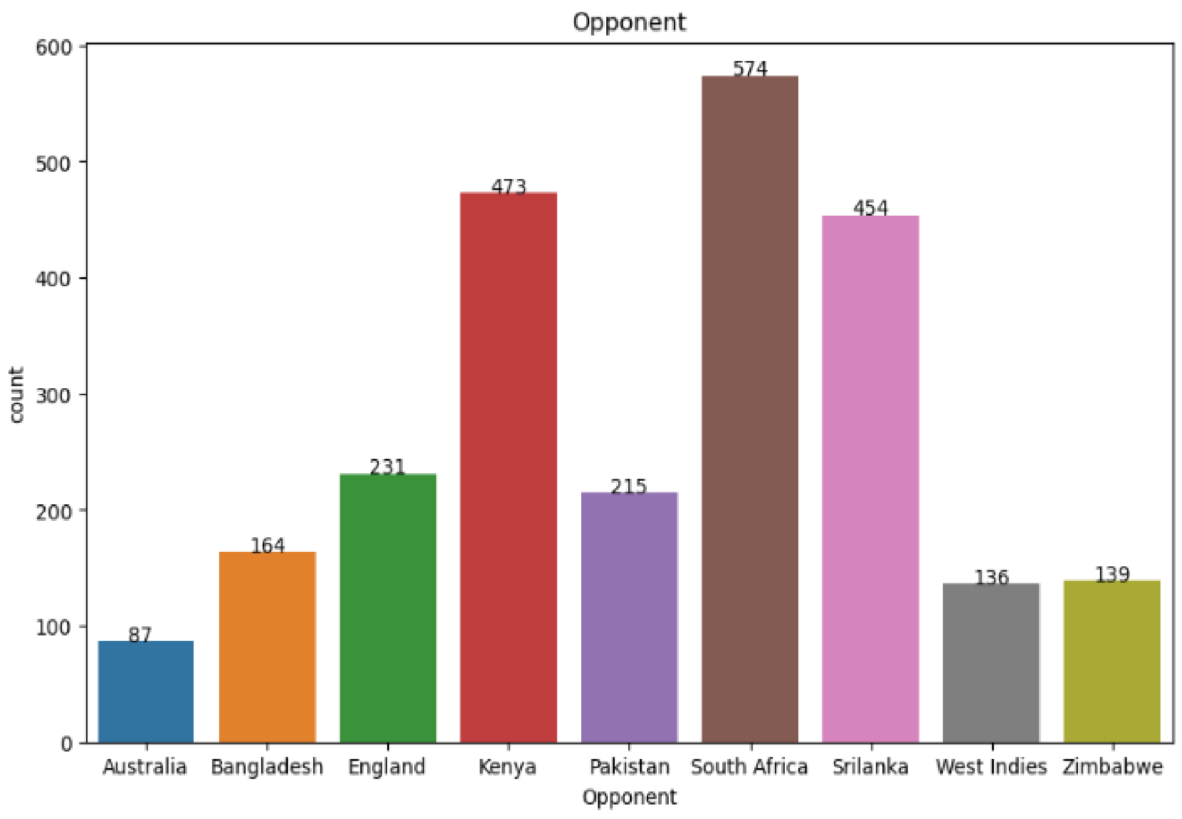
Insights:

- In 1458 matches, 3 players per match were dismissed for a duck.

- Only in 146 matches, 1 player per match got out for a duck.

v) **Opponent**:

Figure 5: Opponent



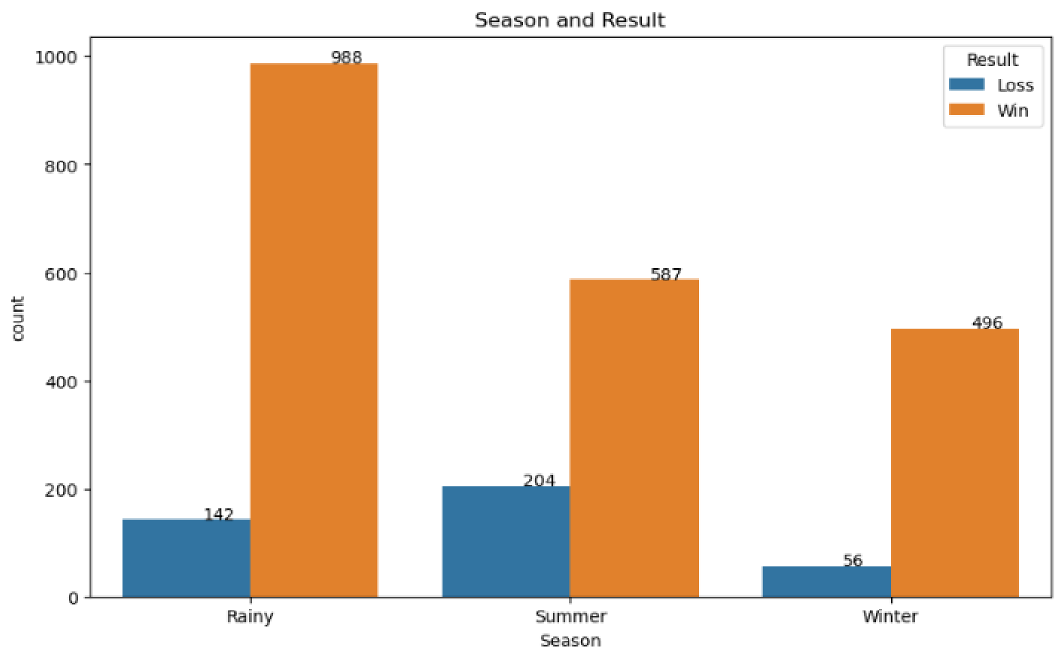
Insights:

- The Indian team played most number of matches against South Africa (574), followed by Kenya (473).

- Least number of matches were played against Australia (87).

vi) **<u>Season and Result</u>**:

Figure 6: Season and Result



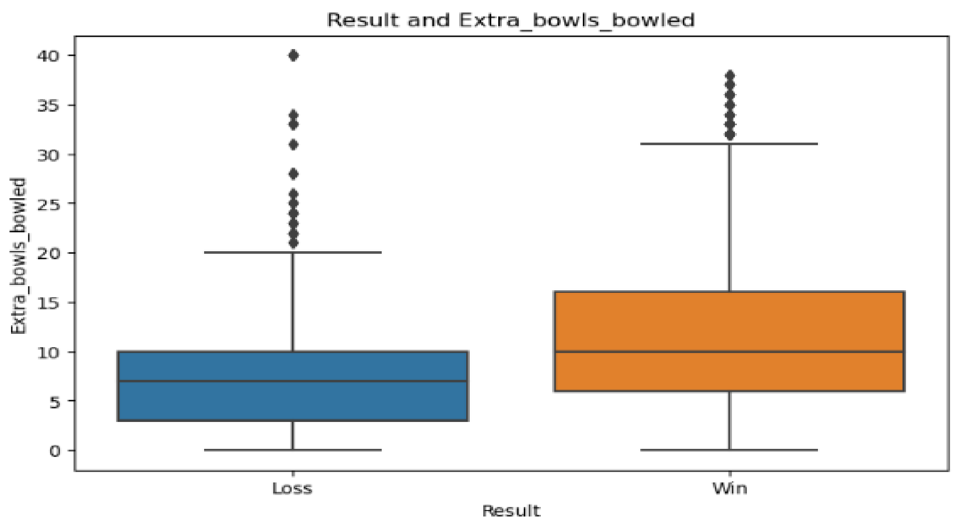
Insights:

- The most matches (1130) were played during the rainy season, with 988 resulting in wins and 142 in losses.

- Only 552 matches were played during winter season.

vii) <u>Result and extra bowls bowled:</u>

Figure 7: Result and Extra_bowls_bowled:

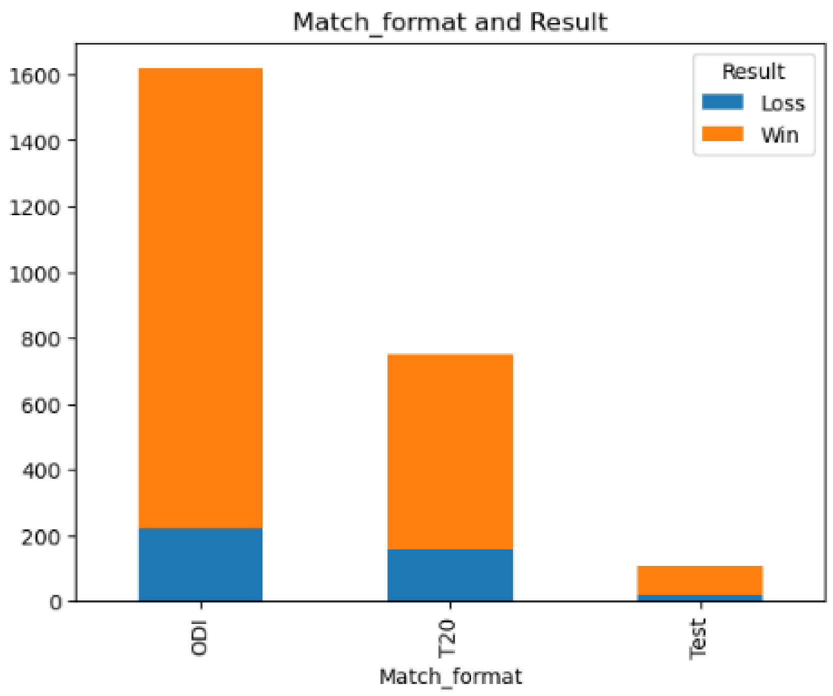


Insights:

- Mean extra bowls bowled per match in win matches is approximately 12.
- Maximum extra bowls bowled per match is 40.

viii) **Match format and result**:

Figure 8: Match_format and Result:

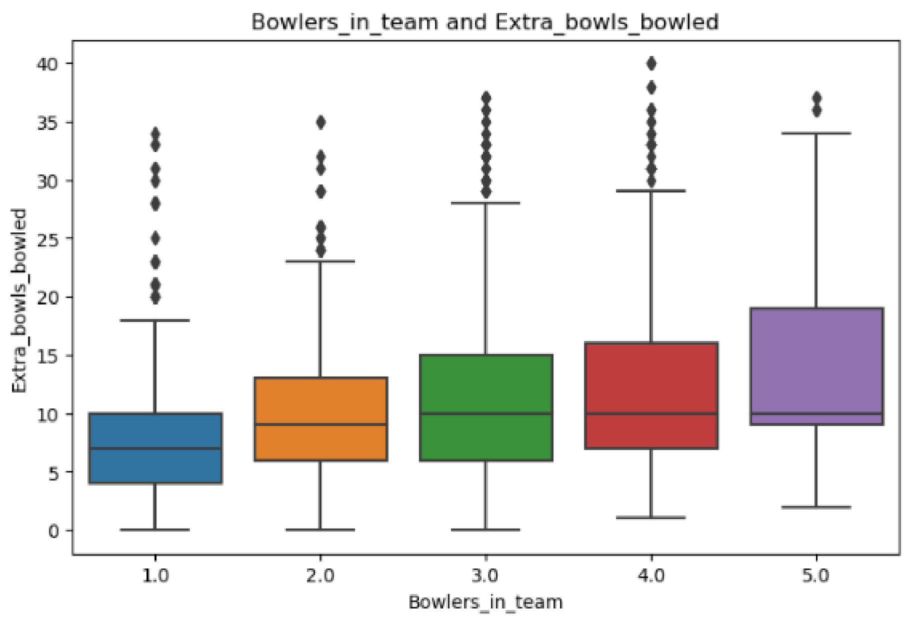


Insights:

- In ODI’s, they won 1394 matches and lost 223 matches.
- In T20’s, they won 593 matches and lost 158 matches.
- They played only 105 test matches of which they won 84 matches.

ix) **Bowlers in team and extra bowls bowled**:

Figure 9: Bowlers_in_team and Extra_bowls_bowled:

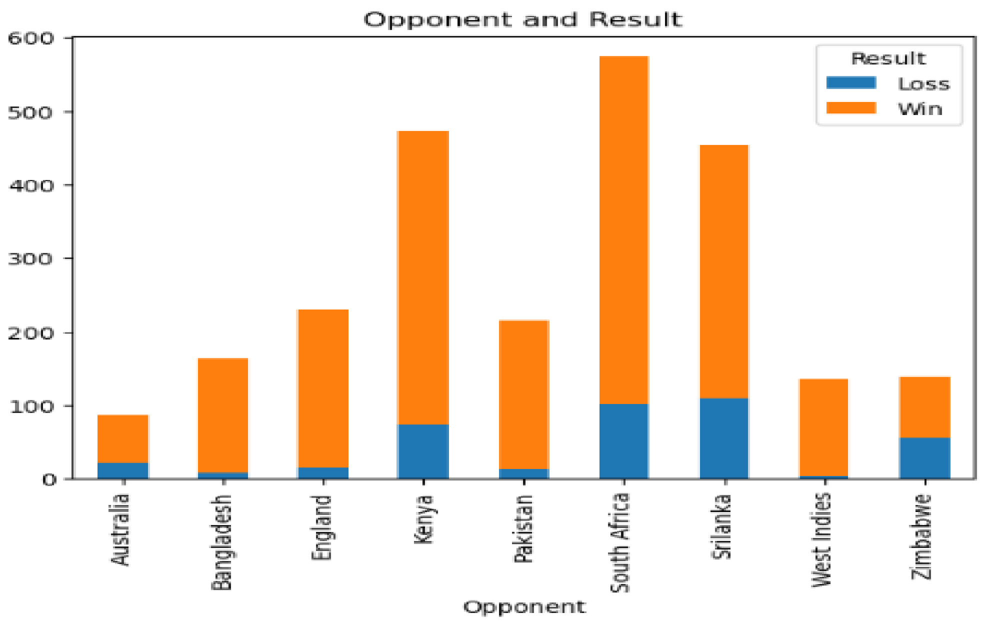


Insight:

- Mean extra bowls bowled when there was only 1 bowler in a team is 9 and mean extra bowls bowled when there were 5 bowlers in a team is 14.

x) **Opponent and result**:

Figure 10: Opponent and Result:



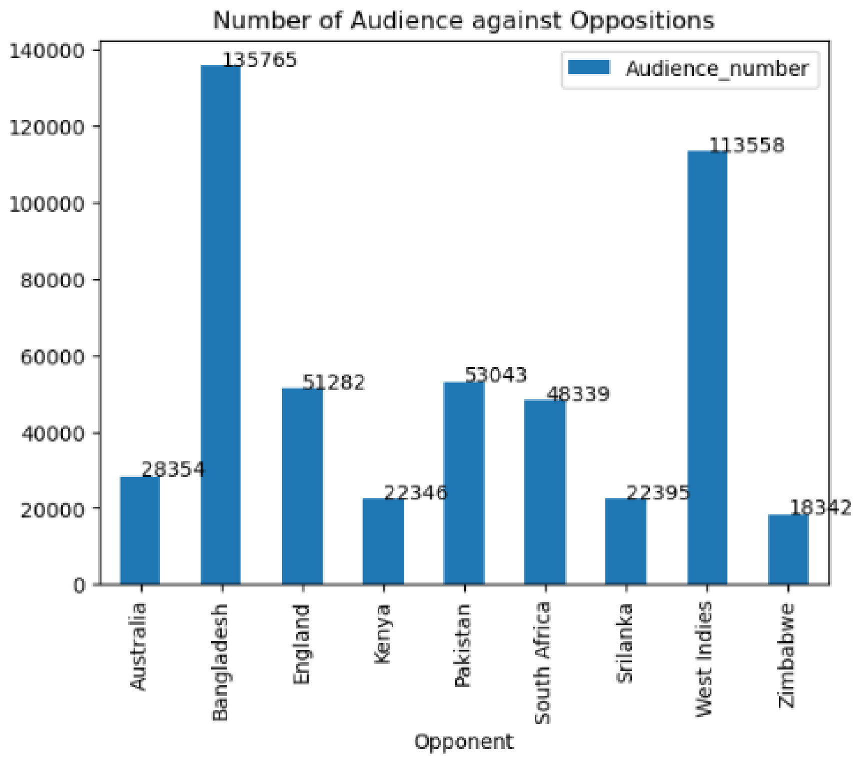
Insights:

- Against Pakistan they won 202 matches and lost just 13 matches.

- When played against West Indies they won 133 matches and lost only 3 matches.

xi) **<u>Opponent and Audience number</u>**:

Figure 11: Opponent and Audience_number:



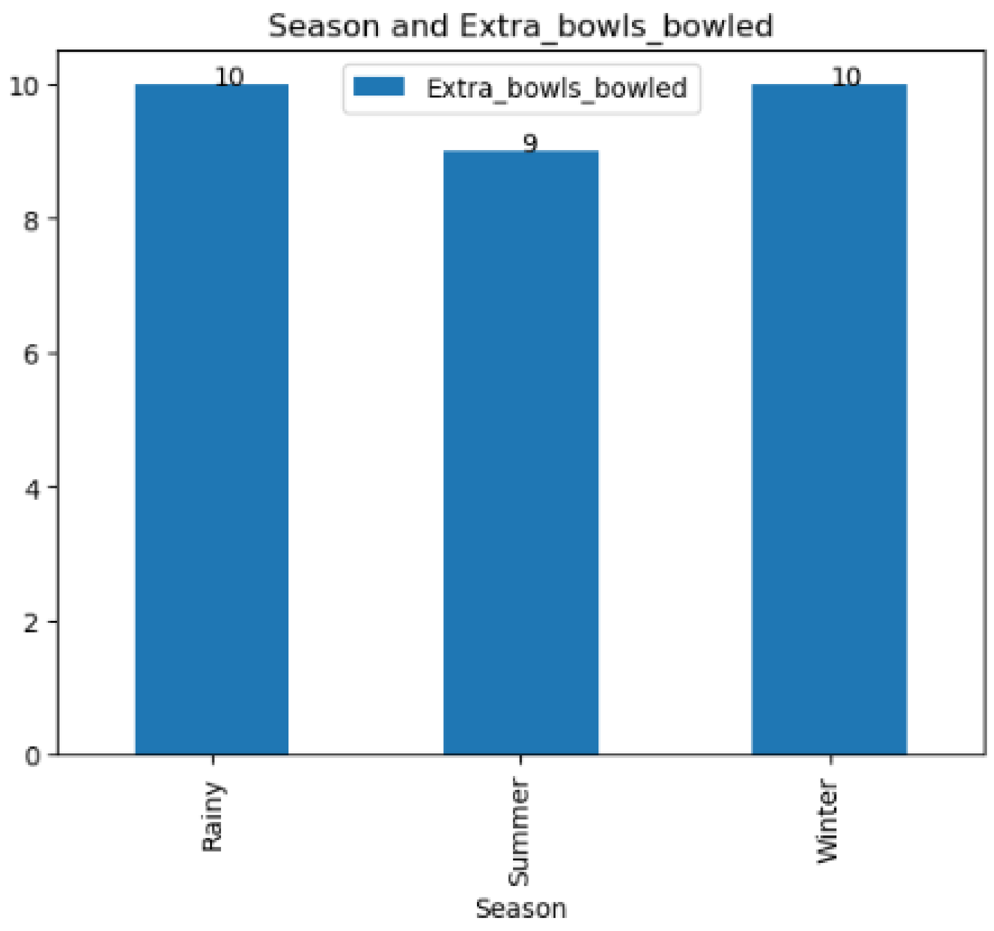
Insights:

- Highest number of audience (135765) were present when played against Bangladesh.

- The number of audiences when playing against Zimbabwe was18342 which is the lowest.

xii) **<u>Season and extra bowls bowled</u>**:

Figure 12: Season and Extra_bowls_bowled:

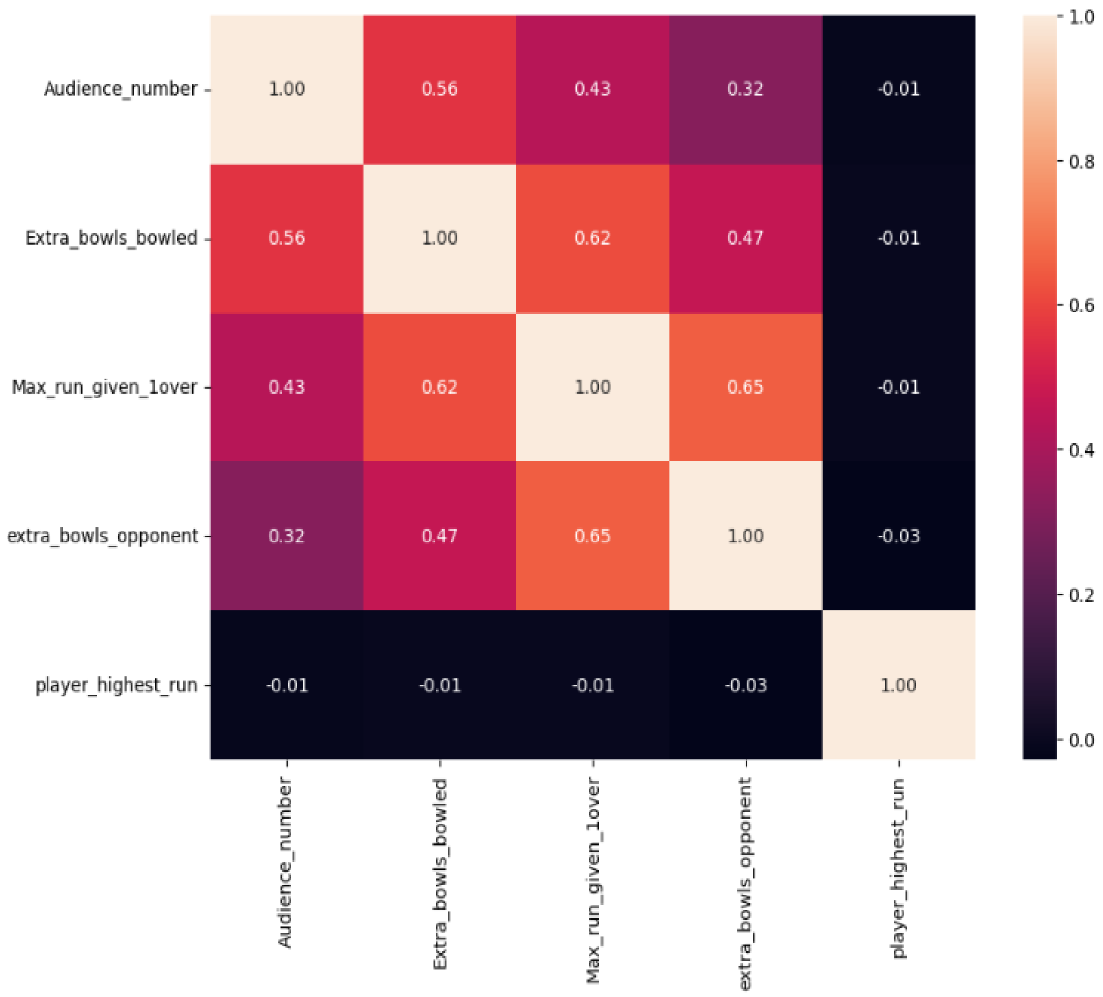


Insight:

- Mean extra bowls bowled during rainy and winter season is 10 and during summer season is 9.

xiii) **<u>Correlation</u>**:

Figure 13: Heatmap to check correlation between variables



Insight:

- The highest correlation of 0.65 is between the variables ‘extra_bowls_opponent’ and ‘Max_run_given_1over’.

- The lowest correlation of -0.03 is between the variables ‘player_highest_run’ and ‘extra_bowls_opponent’.

- **<u>Dropping unrequired columns:</u>**

  We are dropping unrequired columns such as ‘Game number’, ‘Avg team age’ and ‘Wicket_keeper_in_team’ due to the presence of extreme outliers and constant values that do not contribute to our analysis.

- **<u>Missing value treatment:</u>**

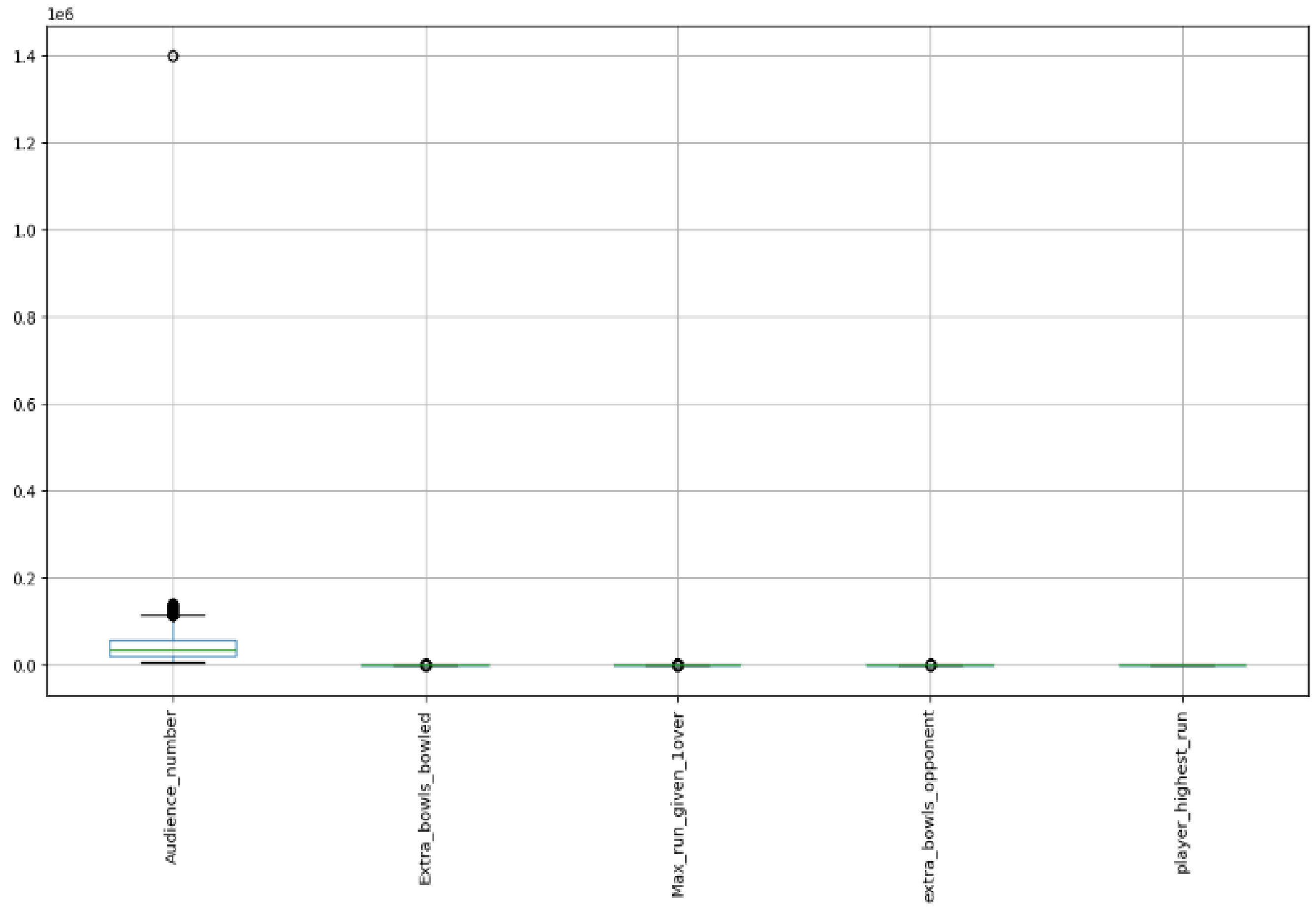
  For numerical data, we are replacing missing values with median values.

  For categorical data, we are replacing missing values with mode values.

  We deleted approximately 457 rows with null values and imputed the remaining null values using the median and mode. After treating null values, the dataset now contains 2473 rows.

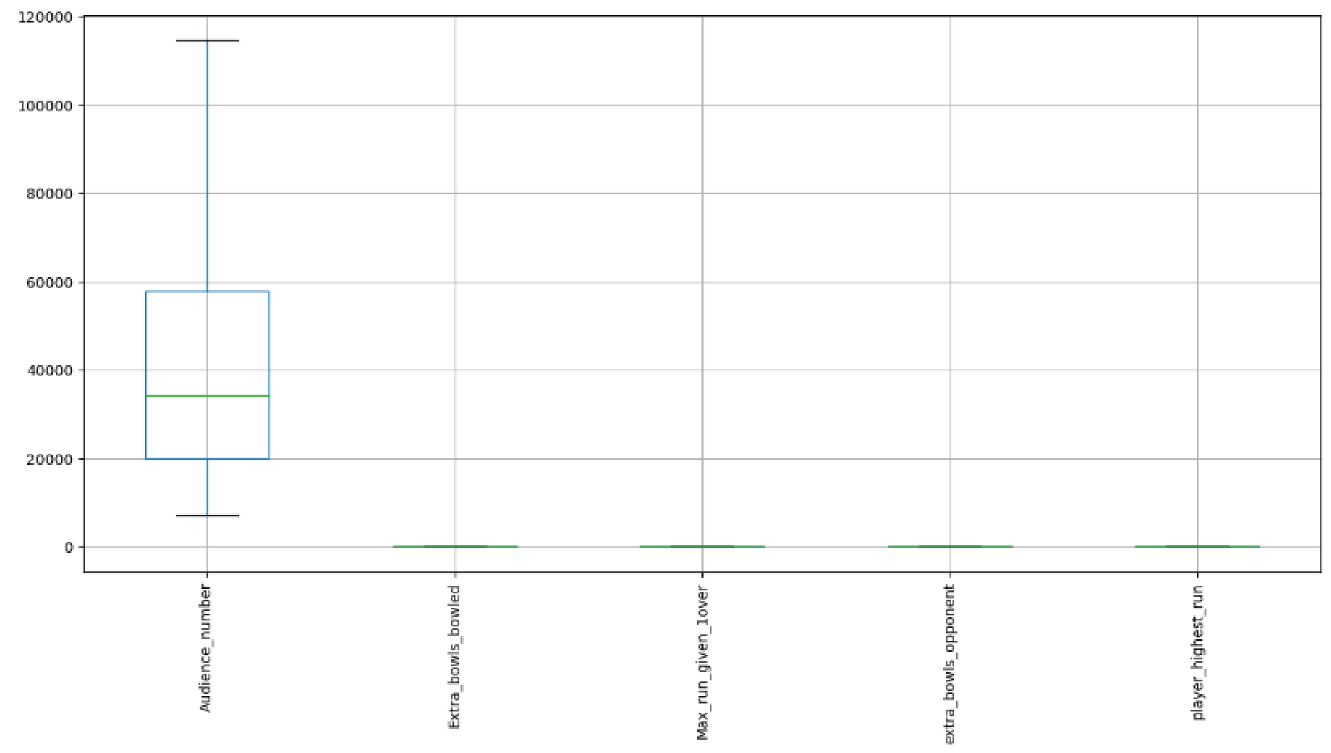
- **<u>Outlier treatment:</u>**

  Figure 14: Box plots before treating outliers



Since there are outliers present, we are treating the same using the percentile method in order to improve the accuracy and to avoid unnecessary fluctuations in the mean value of the given variables.

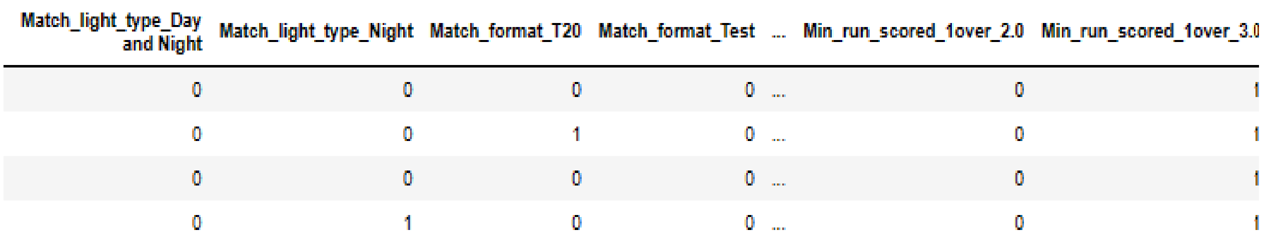
Figure 15: Box plots after treating outliers



- **<u>Variable transformation:</u>**

  We have used one hot encoding to convert categorical variables into numeric ones and created dummy variables to enhance system readability for better prediction.

Table 6: One Hot Encoding and dummy variables



Match_light_type_Day and Night	Match_light_type_Night	Match_format_T20	Match_format_Test	...	Min_run_scored_1over_2.0	Min_run_scored_1over_3.0
0	0	0	0	...	0	1
0	0	1	0	...	0	1
0	0	0	0	...	0	1
0	1	0	0	...	0	1

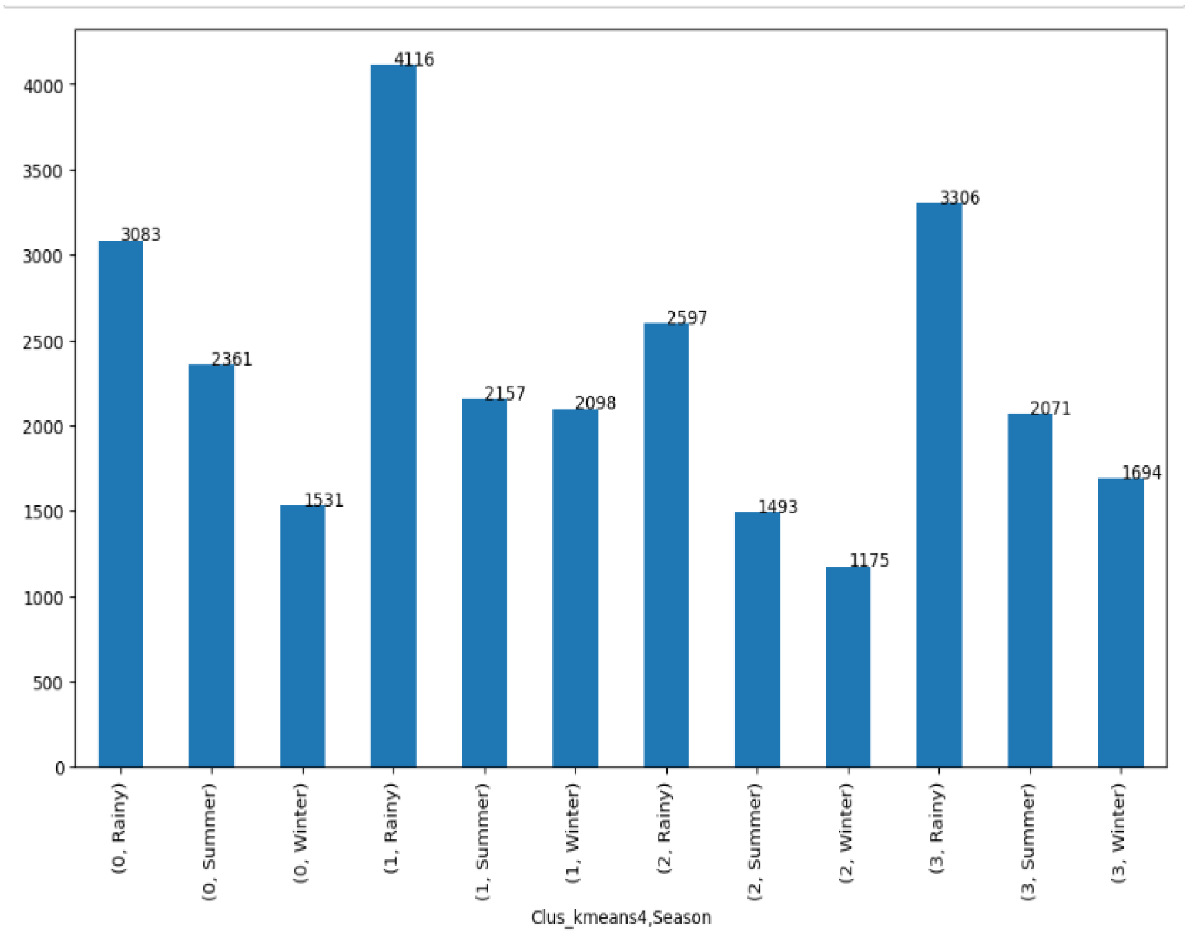
<u>Advanced EDA using clustering:</u>

We have scaled the data and are using Recursive Feature Elimination technique for feature ranking.

K-means method is used to identify clusters of data objects in a dataset, and the Elbow method is used to determine the number of centroids. We are selecting the optimum number of clusters, k= 4, based on the Within-Cluster Sum of Squares (WSS).

xiv) <u>**Season and extra bowls bowled based on cluster profiles**</u>:

Figure 16: Season and Extra_bowls_bowled based on cluster profiles:



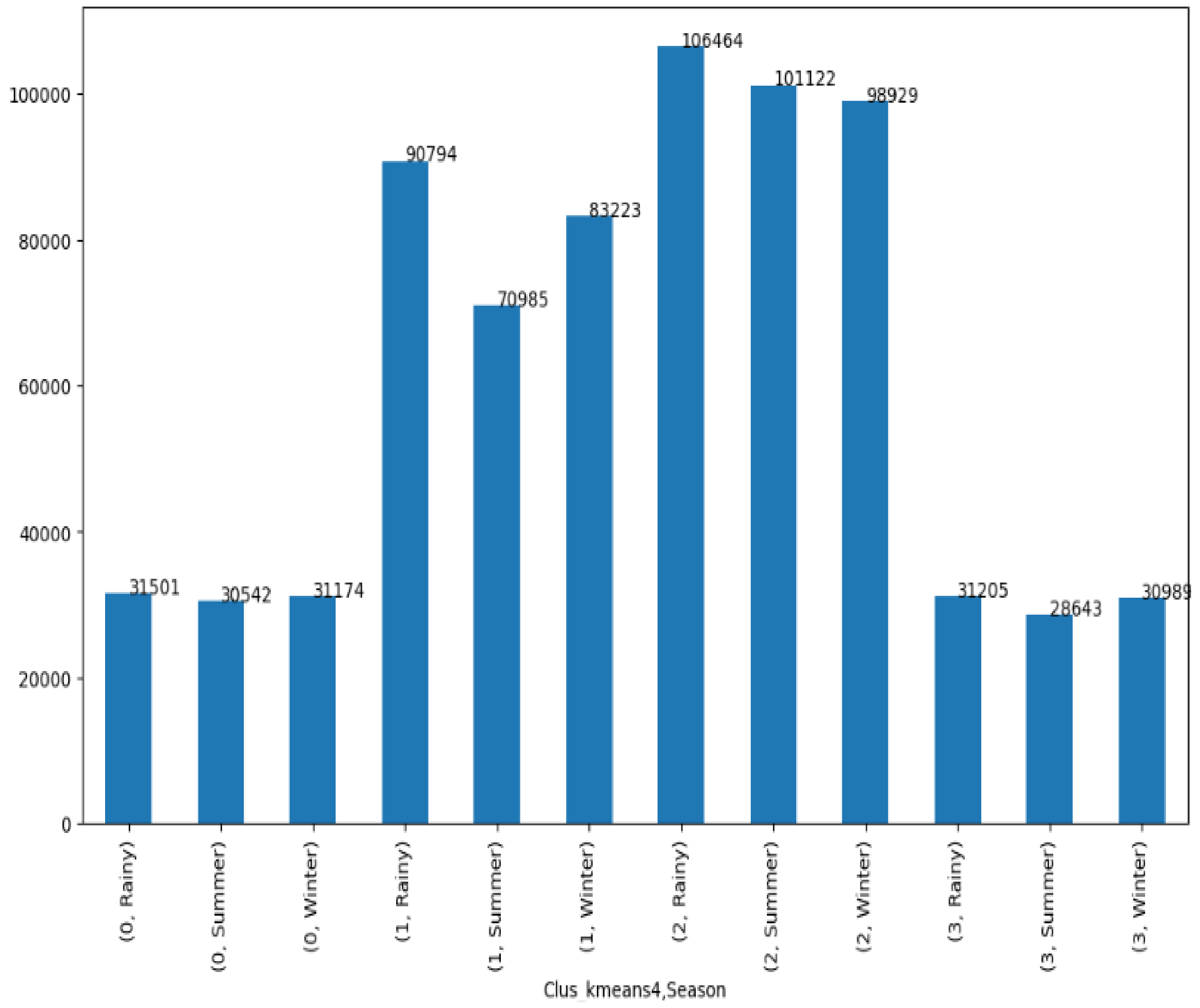
Insights:

- We can observe that the maximum number of extra bowls bowled during the rainy season belongs to the cluster profile 1.

- The matches in Cluster 2 have the lowest number of extra bowls bowled.

xv) **<u>Season and Audience number based on cluster profiles</u>**:

Figure 17: Season and Audience_number based on cluster profiles:



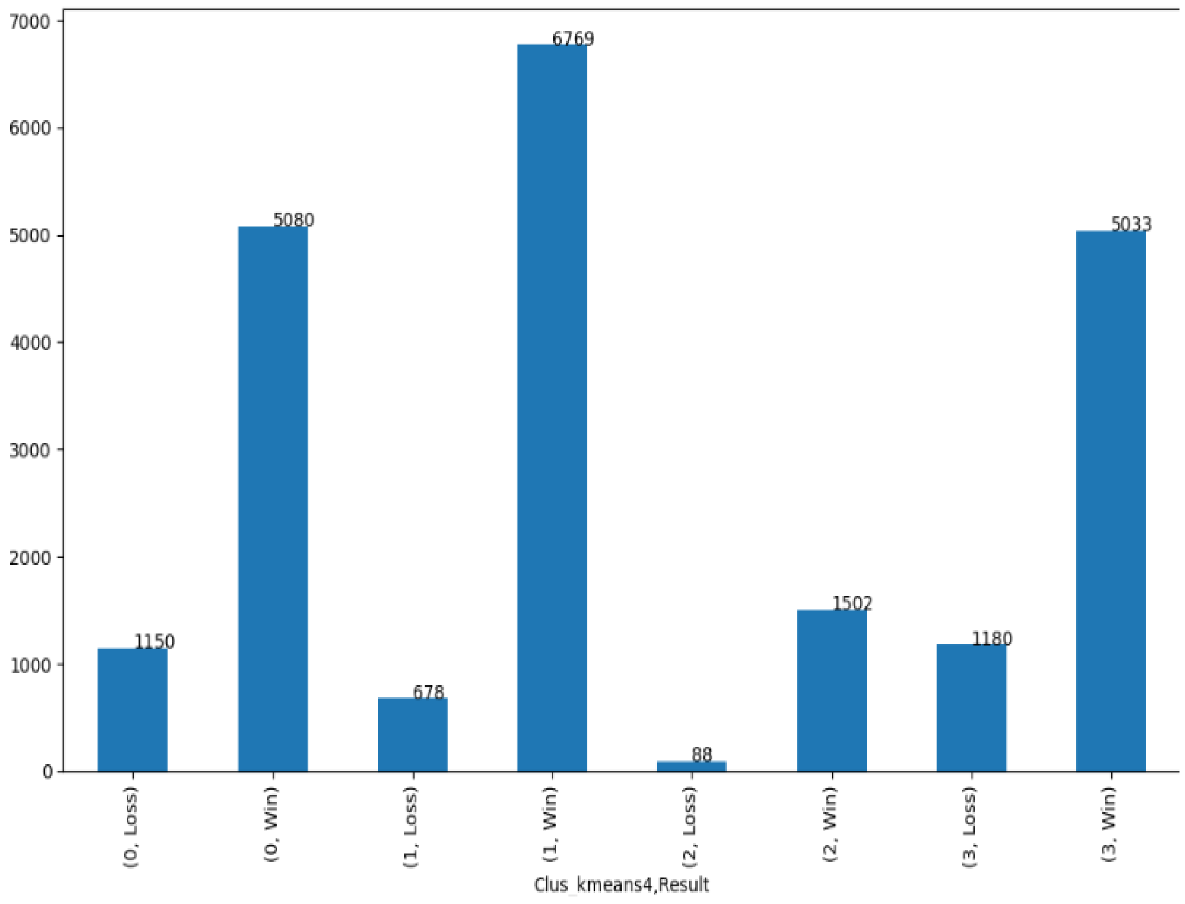
Insights:

- The matches in Cluster 3 have the lowest number of audiences when compared with other cluster profiles.

- The maximum number of audiences were present in matches belonging to Cluster 2.

xvi) **<u>Result and Max run given 1over based on cluster profiles</u>**:

Figure 18: Result and Max_run_given_1over based on cluster profiles:



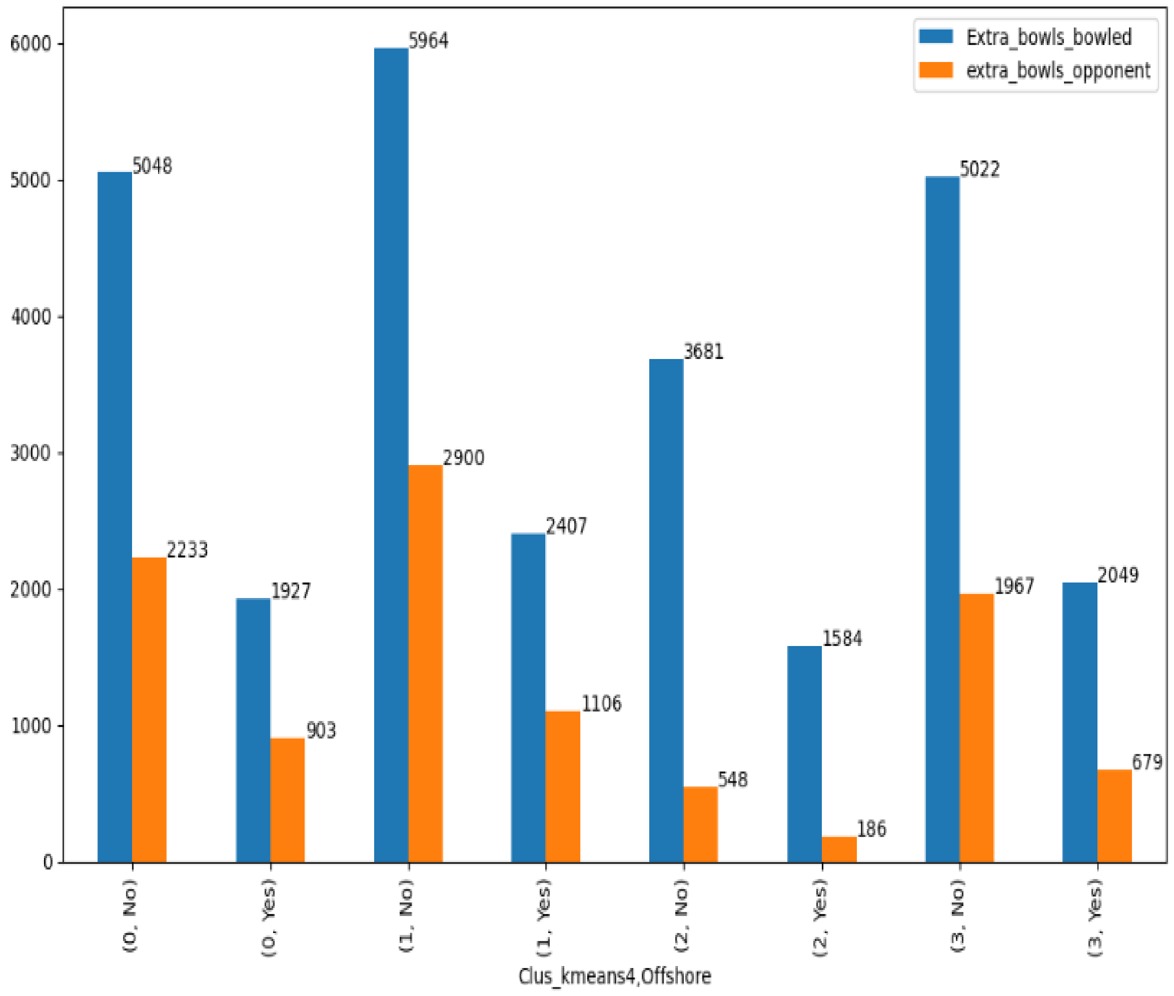
Insights:

- Among the matches lost, the least runs given in an over belongs to Cluster 2.

- Among the matches won, the maximum runs given in an over belongs to Cluster 1.

xvii) **<u>Offshore, extra bowls bowled and extra bowls opponent based on cluster profiles</u>**:

Figure 19: Offshore, Extra_bowls_bowled and extra_bowls_opponent based on cluster profiles:



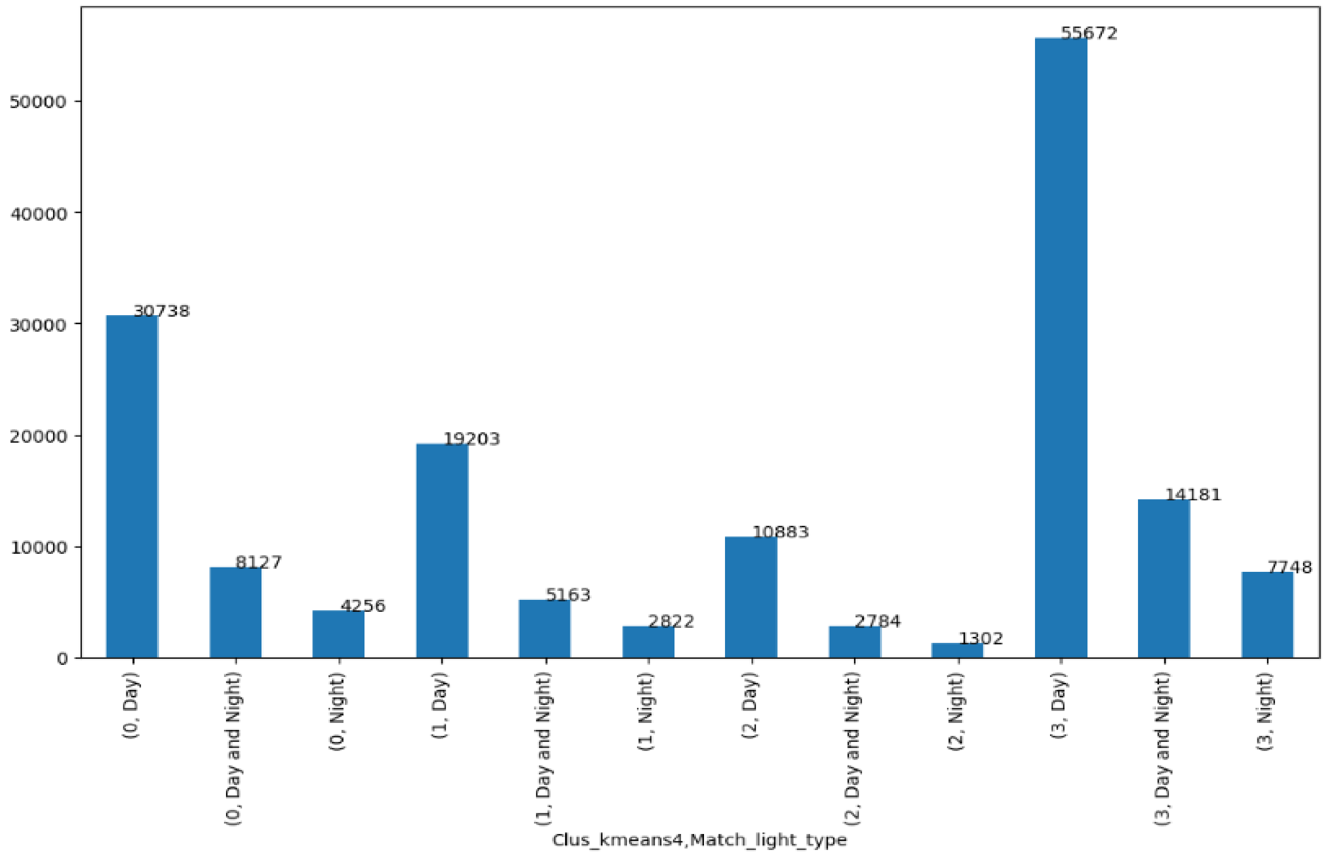
Insights:

- Within home matches, the maximum number of extra bowls bowled by both the Indian team and the opponents belongs to Cluster 1.

- Among offshore matches, Cluster 2 has the lowest number of extra bowls bowled.

xviii) Match light type and player highest run based on cluster profiles:

Figure 20: Match_light_type and player_highest_run based on cluster profiles



Insights:

- Among night matches, the lowest score by a single player belongs to Cluster 2.

- Cluster 3 includes matches where one player scored the highest, and these were played during the Day.

- **Imbalanced data:**

  After treating the null values, we have 2473 rows, of which they won 2071 matches and lost only 402 matches. This indicates that almost 84% of the data comprises wins. So, we can say that the data is highly imbalanced, which negatively affects the accuracy and prediction of the machine learning models. We can use **SMOTE** technique to treat the imbalanced data i.e., by generating synthetic samples for the minority class.

--------------------------------**PROJECT NOTES – 1-----THE END**-----------------------------

PROJECT NOTES – 2

- **Train and Test Split:**

Assigning the independent variables to 'X' and dependent variables to 'y' and splitting the data into Train and Test at 80:20 ratio.

- **Scaling data:**

We have scaled data for certain models as we have continuous and ordinal variables with different measures.

Model Building, Tuning and Interpretation:

We are building 12 models using balanced and imbalanced data.

We have balanced the data by using SMOTE upsampling method.

In this project, we are building only non-parametric models.

1) Random Forest model:

While building the Random Forest model, we noted overfitting issues with the training data. So, we are tuning the model in order to address this issue.

Model Tuning:

We have applied **GridsearchCV** for Random Forest model:

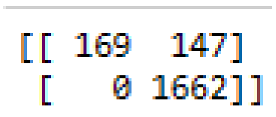
- **Accuracy:**

  - Accuracy score of **Train** dataset is 0.93

  - Accuracy score of **Test** dataset is 0.89

- Train data:

Table 7: Random Forest – Confusion Matrix and Classification report of train data

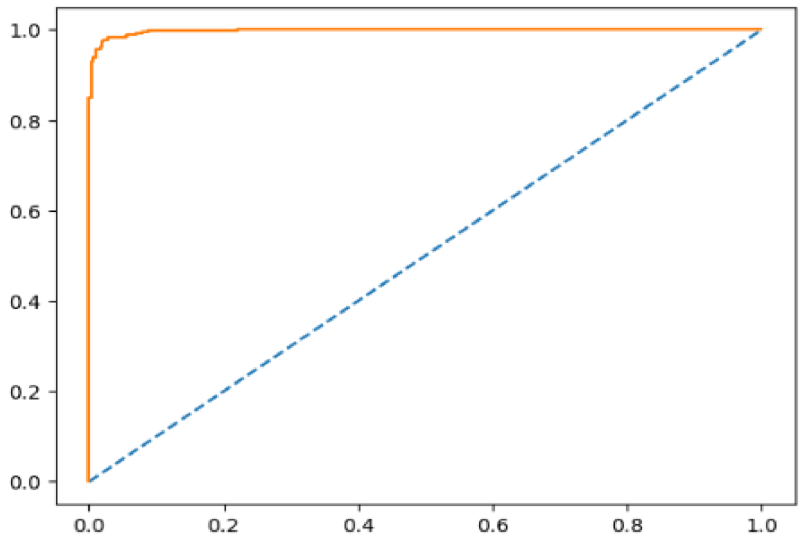


```
[[ 169  147]
 [   0 1662]]
```

```
              precision    recall  f1-score   support

           0       1.00      0.53      0.70       316
           1       0.92      1.00      0.96      1662

    accuracy                           0.93      1978
   macro avg       0.96      0.77      0.83      1978
weighted avg       0.93      0.93      0.92      1978
```

Figure 21: Random Forest - ROC Curve of train data

```
AUC: 0.998
```



- **ROC_AUC score of Train data is 0.998**

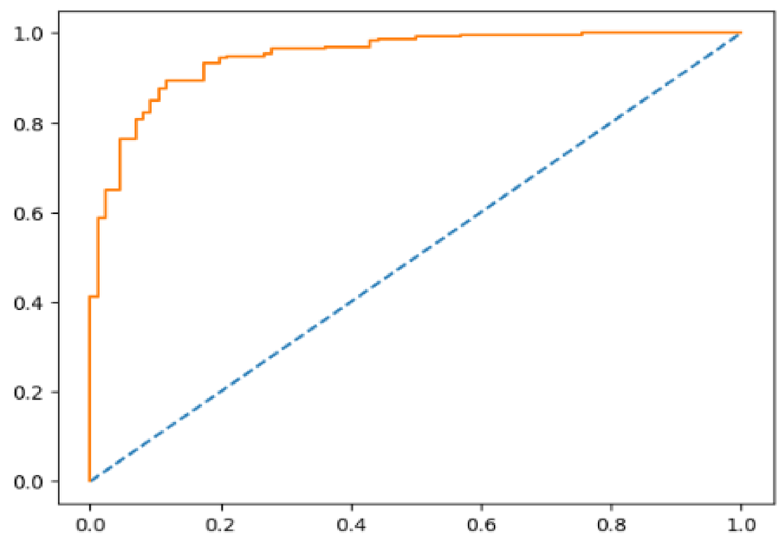
- <u>Test</u> <u>data</u>:

Table 8: Random Forest – Confusion Matrix and Classification report of test data

```
[[ 32  54]
 [  2 407]]
              precision    recall  f1-score   support

           0       0.94      0.37      0.53        86
           1       0.88      1.00      0.94       409

    accuracy                           0.89       495
   macro avg       0.91      0.68      0.73       495
weighted avg       0.89      0.89      0.87       495
```

Figure 22: Random Forest - ROC Curve of test data

```
AUC: 0.998
```



- **ROC_AUC score for Test data is 0.998**

Observations:

- Accuracy score is good for both train (0.93) and test data (0.89).

- F1-score for predicting wins remains strong, with only 2% decrease from the training data, i.e., from 96% it got reduced 94% in the test data.

- The Precision for predicting losses is good for both the train and test data.

2) <u>Random Forest with SMOTE:</u>

We noted overfitting issues with the training data for Random Forest model with SMOTE as well.

We have applied **GridsearchCV** for Random Forest model with SMOTE to address this issue

- **<u>Accuracy:</u>**

  - Accuracy score of **Train** dataset is 0.99

  - Accuracy score of **Test** dataset is 0.92

- **Train** data:

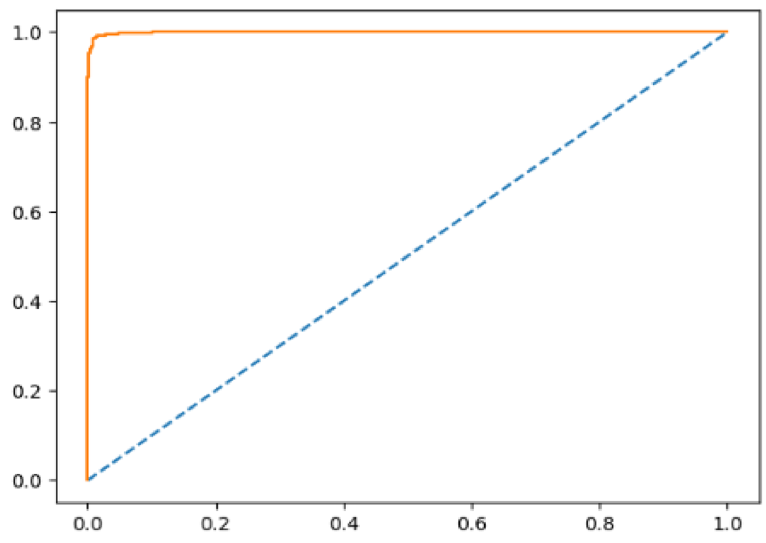
Table 9: Random Forest with SMOTE – Confusion Matrix and Classification report of train data

```
[[1637   25]
 [  14 1648]]
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1662
           1       0.99      0.99      0.99      1662

    accuracy                           0.99      3324
   macro avg       0.99      0.99      0.99      3324
weighted avg       0.99      0.99      0.99      3324
```

Figure 23: Random Forest with SMOTE - ROC Curve of train data

```
AUC: 0.999
```



- **ROC_AUC score of Train data is 0.999**

- **Test** data:

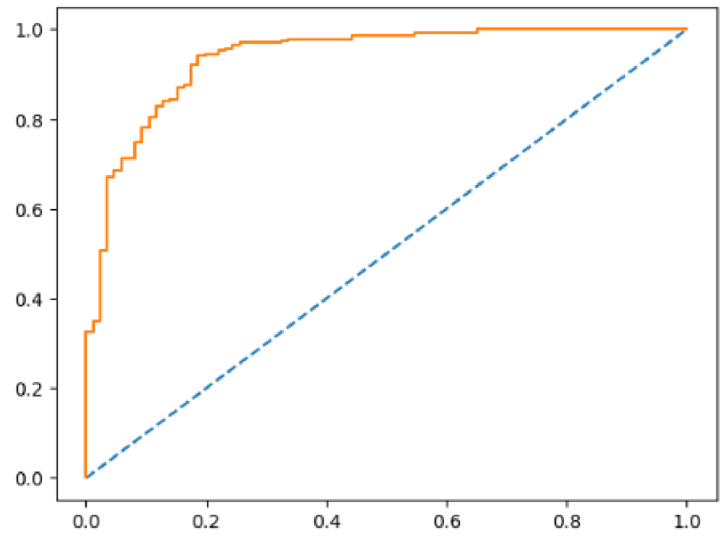
Table 10: Random Forest with SMOTE – Confusion Matrix and Classification report of test data

```
[[ 57  29]
 [  9 400]]
```

```
              precision    recall  f1-score   support

           0       0.86      0.66      0.75        86
           1       0.93      0.98      0.95       409

    accuracy                           0.92       495
   macro avg       0.90      0.82      0.85       495
weighted avg       0.92      0.92      0.92       495
```

Figure 24: Random Forest with SMOTE - ROC Curve of test data

AUC: 0.999



- **ROC_AUC score for Test data is 0.999**

Observations:

- The accuracy score has decreased from 99% to 92% on both the train and test data.

- F1-score has reduced drastically from 99% to 75% while predicting losses.

- The model appears to be overfitting to the training data.

- The precision and recall for predicting wins are above 0.93% for both the train and test data.

3) <u>Naive Bayes model:</u>

Naïve Bayes is a supervised learning algorithm for classification problems which does not require the data to be scaled, hence we are using train and test data from the original data frame.

- **Accuracy:**

  - Accuracy score of **Train** dataset is 0.75

  - Accuracy score of **Test** dataset is 0.747

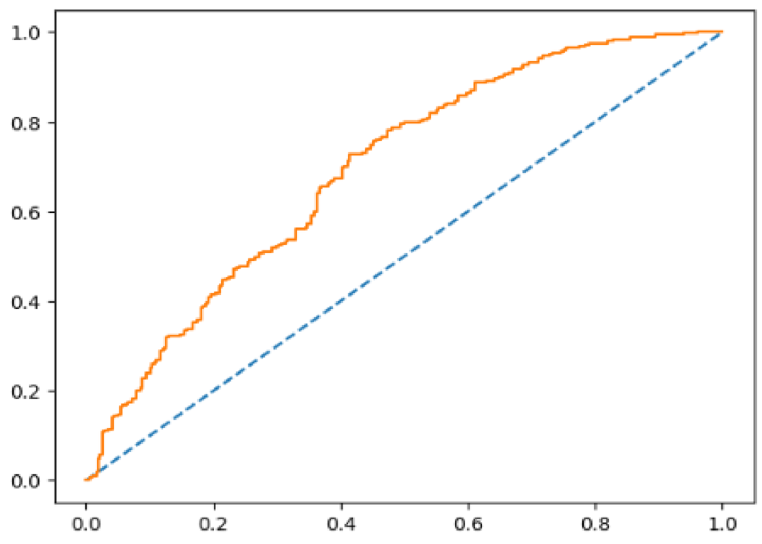
- Train data:

Table 11: Naive Bayes – Confusion Matrix and Classification report of train data

```
0.7507583417593529
[[ 158  158]
 [ 335 1327]]
              precision    recall  f1-score   support

           0       0.32      0.50      0.39       316
           1       0.89      0.80      0.84      1662

    accuracy                           0.75      1978
   macro avg       0.61      0.65      0.62      1978
weighted avg       0.80      0.75      0.77      1978
```

Figure 25: Naive Bayes - ROC Curve of train data

AUC: 0.695



- **ROC_AUC score of Train data is 0.695**

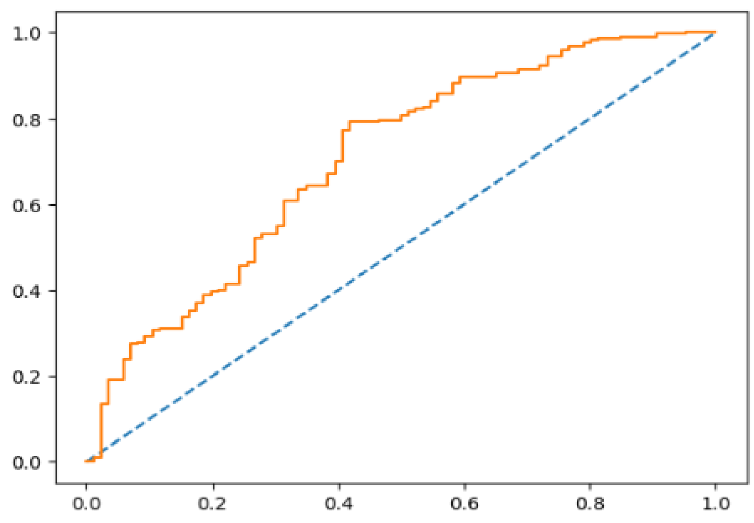
- <u>Test</u> <u>data</u>:

Table 12: Naive Bayes – Confusion Matrix and Classification report of test data

```
0.7474747474747475
[[ 46  40]
 [ 85 324]]
              precision    recall  f1-score   support

           0       0.35      0.53      0.42        86
           1       0.89      0.79      0.84       409

    accuracy                           0.75       495
   macro avg       0.62      0.66      0.63       495
weighted avg       0.80      0.75      0.77       495
```

Figure 26: Naive Bayes - ROC Curve of test data

```
AUC: 0.695
```



- **ROC_AUC score for Test data is 0.695**

Observations:

- ROC_AUC score is less (0.695) when compared with other models.

- Precision, recall and f1-score is comparatively less while predicting losses.

- F1-score for predicting wins remains the same for both the train and test data.

4) **Naive Bayes with SMOTE**:

Building Naïve Bayes model with the balanced data.

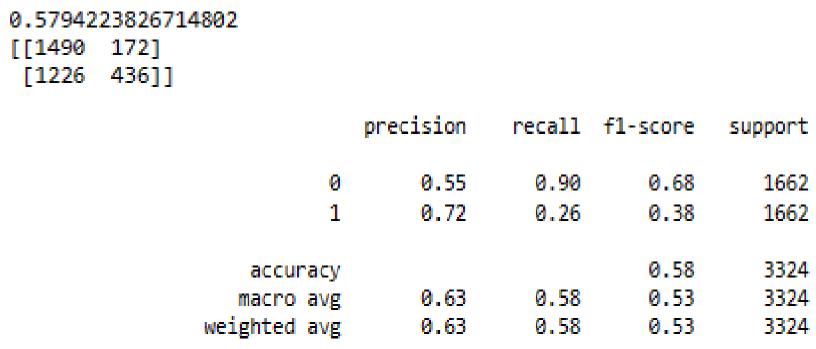
- **Accuracy:**

- Accuracy score of **Train** dataset is 0.58

- Accuracy score of **Test** dataset is 0.39

- Train data:

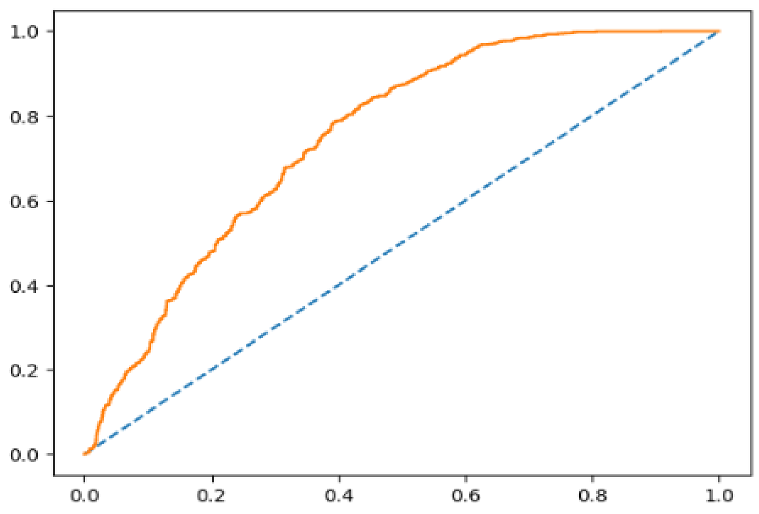
Table 13: Naive Bayes with SMOTE – Confusion Matrix and Classification report of train data



```
0.5794223826714802
[[1490  172]
 [1226  436]]
              precision    recall  f1-score   support

           0       0.55      0.90      0.68      1662
           1       0.72      0.26      0.38      1662

    accuracy                           0.58      3324
   macro avg       0.63      0.58      0.53      3324
weighted avg       0.63      0.58      0.53      3324
```

Figure 27: Naive Bayes with SMOTE - ROC Curve of train data

AUC: 0.750



- **ROC_AUC score of Train data is 0.750**

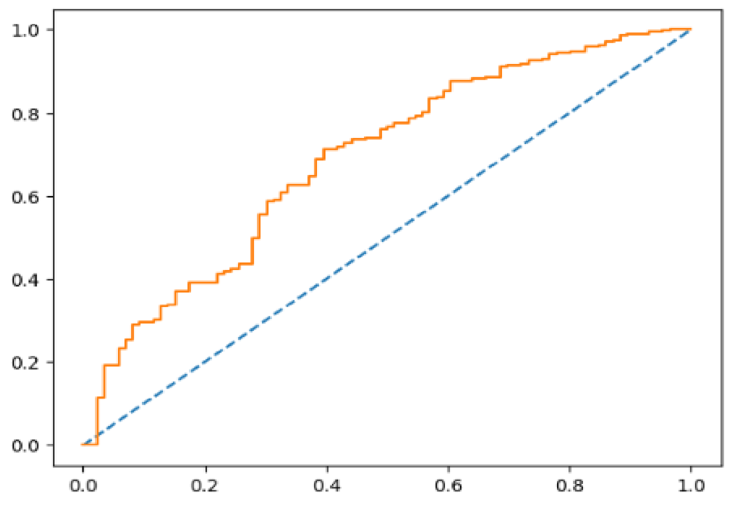
- **Test** data:

Table 14: Naive Bayes with SMOTE – Confusion Matrix and Classification report of test data

```
0.3939393939393939
[[ 79   7]
 [293 116]]
              precision    recall  f1-score   support

           0       0.21      0.92      0.34        86
           1       0.94      0.28      0.44       409

    accuracy                           0.39       495
   macro avg       0.58      0.60      0.39       495
weighted avg       0.82      0.39      0.42       495
```

Figure 28: Naive Bayes with SMOTE - ROC Curve of test data

```
AUC: 0.750
```



- **ROC_AUC score for Test data is 0.750**

Observations:

- The accuracy score for train and test data is only 58% and 39%.

- Even the f1-scores are less when compared with other machine learning models. Tuning will be required in order to improve the performance.

5) KNN model:

KNN model is a supervised learning algorithm for classification and regression problems.

KNN model needs the data to be scaled, hence we are using the scaled data.

- **Accuracy:**

  - Accuracy score of **Train** dataset is 0.87

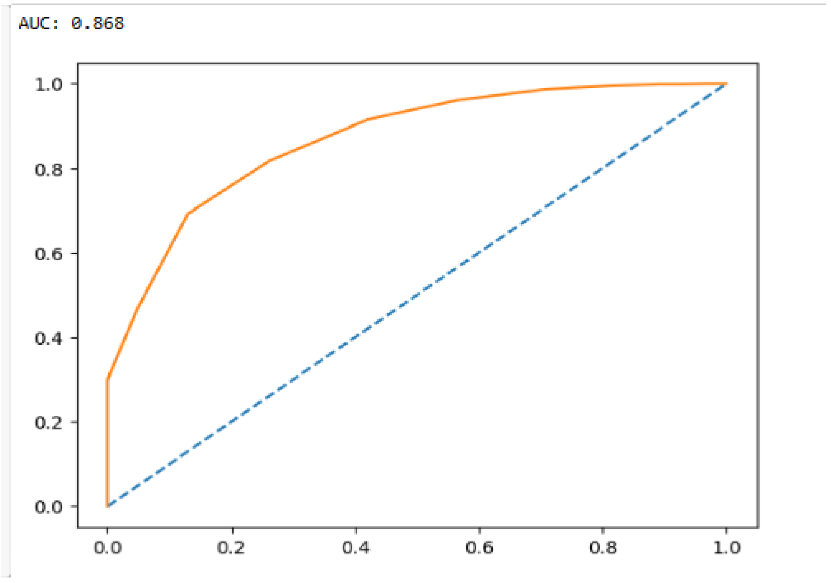
  - Accuracy score of **Test** dataset is 0.85

- Train data:

Table 15: KNN – Confusion Matrix and Classification report of train data

```
0.8660262891809909
[[  59  257]
 [   8 1654]]
              precision    recall  f1-score   support

           0       0.88      0.19      0.31       316
           1       0.87      1.00      0.93      1662

    accuracy                           0.87      1978
   macro avg       0.87      0.59      0.62      1978
weighted avg       0.87      0.87      0.83      1978
```

Figure 29: KNN - ROC Curve of train data



- **ROC_AUC score of Train data is 0.868**

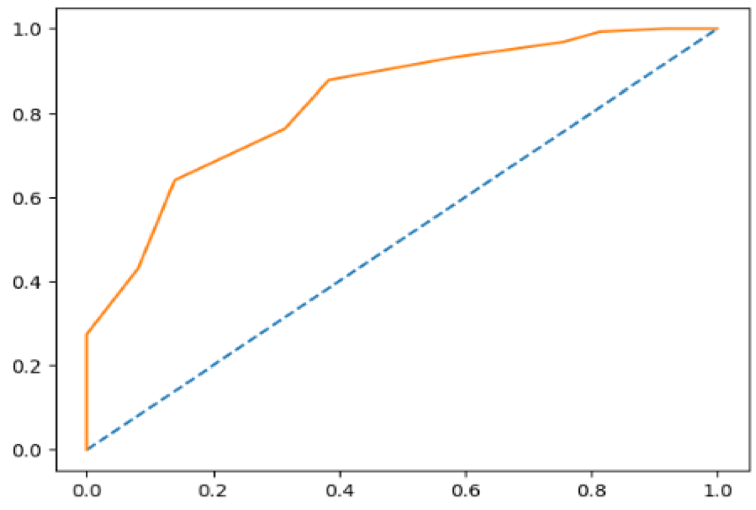
- Test data:

Table 16: KNN – Confusion Matrix and Classification report of test data

```
0.8525252525252526
[[ 16  70]
 [  3 406]]
              precision    recall  f1-score   support

           0       0.84      0.19      0.30        86
           1       0.85      0.99      0.92       409

    accuracy                           0.85       495
   macro avg       0.85      0.59      0.61       495
weighted avg       0.85      0.85      0.81       495
```

Figure 30: KNN - ROC Curve of test data

```
AUC: 0.868
```



- **ROC_AUC score for Test data is 0.868**

Observations:

- The accuracy score is 87% for both the train and test data.

- F1-score and recall for predicting losses are very low, whereas they are high for predicting wins.

6) KNN model with SMOTE:

Building a KNN model using the scaled and balanced data.

- **Accuracy:**

  - Accuracy score of **Train** dataset is 0.75

  - Accuracy score of **Test** dataset is 0.56

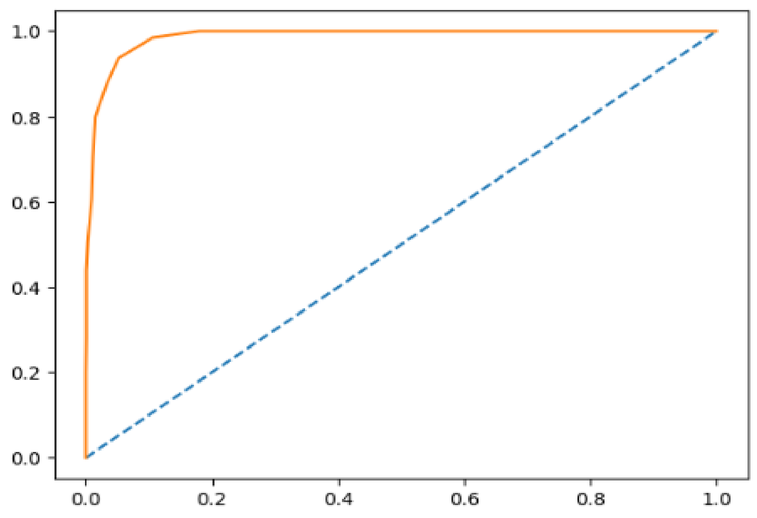
- Train data:

Table 17: KNN with SMOTE – Confusion Matrix and Classification report of train data

```
0.7512033694344163
[[1657    5]
 [ 822  840]]

              precision    recall  f1-score   support

           0       0.67      1.00      0.80      1662
           1       0.99      0.51      0.67      1662

    accuracy                           0.75      3324
   macro avg       0.83      0.75      0.74      3324
weighted avg       0.83      0.75      0.74      3324
```

Figure 31: KNN with SMOTE - ROC Curve of train data

```
AUC: 0.987
```



- **ROC_AUC score of Train data is 0.987**

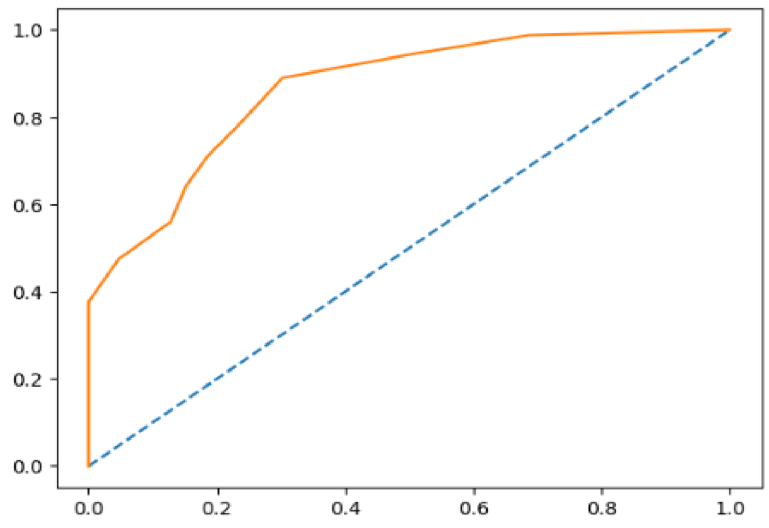
- **Test** data:

Table 18: KNN with SMOTE – Confusion Matrix and Classification report of test data

```
0.5575757575757576
[[ 82   4]
 [215 194]]

              precision    recall  f1-score   support

           0       0.28      0.95      0.43        86
           1       0.98      0.47      0.64       409

    accuracy                           0.56       495
   macro avg       0.63      0.71      0.53       495
weighted avg       0.86      0.56      0.60       495
```

Figure 32: KNN with SMOTE - ROC Curve of test data

```
AUC: 0.987
```



- **ROC_AUC score for Test data is 0.987**

Observations:

- The accuracy score of train and test data has reduced from 75% to 56%.

- ROC_AUC score for both the train and test for balanced data is 98%.

- The precision score for predicting losses is only 28% on the test data.

7) **Decision Tree model**:

We are **tuning** the model by Regularising the Decision Tree to address overfitting issues with the training data.

- **Accuracy:**

- Accuracy score of **Train** dataset is 0.89

- Accuracy score of **Test** dataset is 0.85

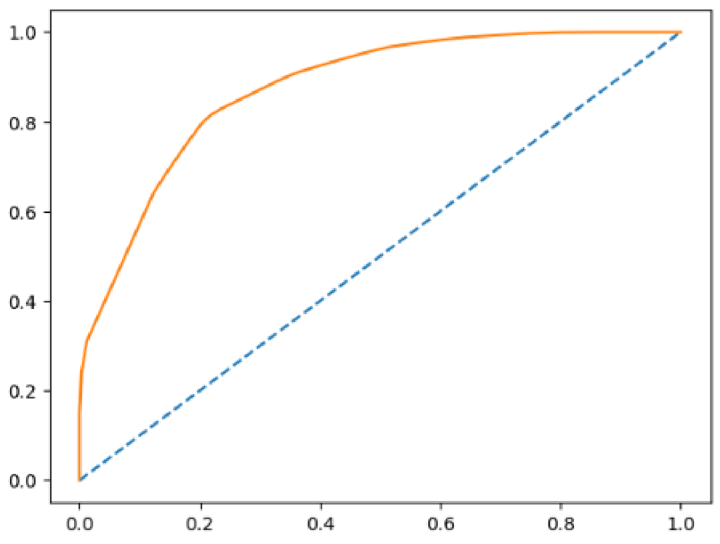
- Train data:

Table 19: Decision Tree – Confusion Matrix and Classification report of train data

```
0.8897876643073812
[[ 153  163]
 [  55 1607]]

              precision    recall  f1-score   support

           0       0.74      0.48      0.58       316
           1       0.91      0.97      0.94      1662

    accuracy                           0.89      1978
   macro avg       0.82      0.73      0.76      1978
weighted avg       0.88      0.89      0.88      1978
```

Figure 33: Decision Tree - ROC Curve of train data

AUC: 0.875



- **ROC_AUC score of Train data is 0.875**

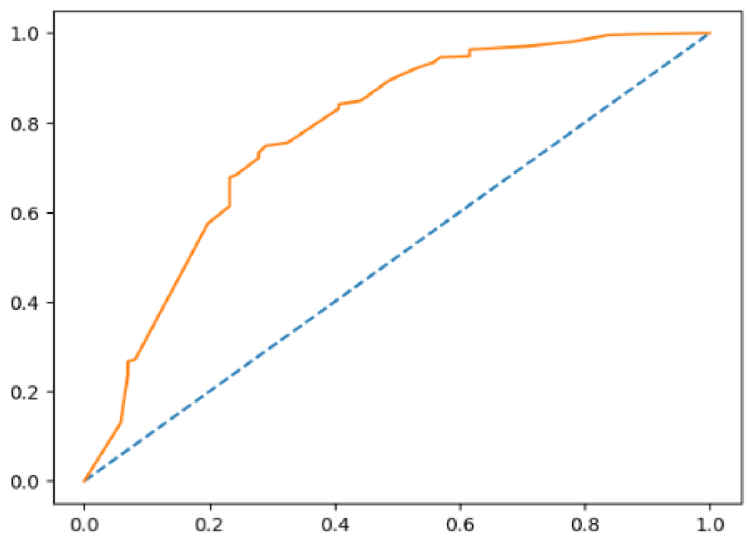
- Test data:

Table 20: Decision Tree – Confusion Matrix and Classification report of test data

```
0.8484848484848485
[[ 38  48]
 [ 27 382]]
              precision    recall  f1-score   support

           0       0.58      0.44      0.50        86
           1       0.89      0.93      0.91       409

    accuracy                           0.85       495
   macro avg       0.74      0.69      0.71       495
weighted avg       0.84      0.85      0.84       495
```

Figure 34: Decision Tree - ROC Curve of test data

```
AUC: 0.875
```



- **ROC_AUC score for Test data is 0.875**

Observations:

- ROC_AUC score for both the train and test for balanced data is 88%.

- The f1-scores for predicting losses is below 0.6, while for predicting wins, it is above 0.8.

- The accuracy score of both the train and test data is good, i.e., 89% and 85%.

8) Decision Tree with SMOTE:

We have noted overfitting issues with the training data, so we are **tuning** the model by Regularising the Decision Tree for balanced data.

- **Accuracy:**

- Accuracy score of **Train** dataset is 0.86

- Accuracy score of **Test** dataset is 0.71

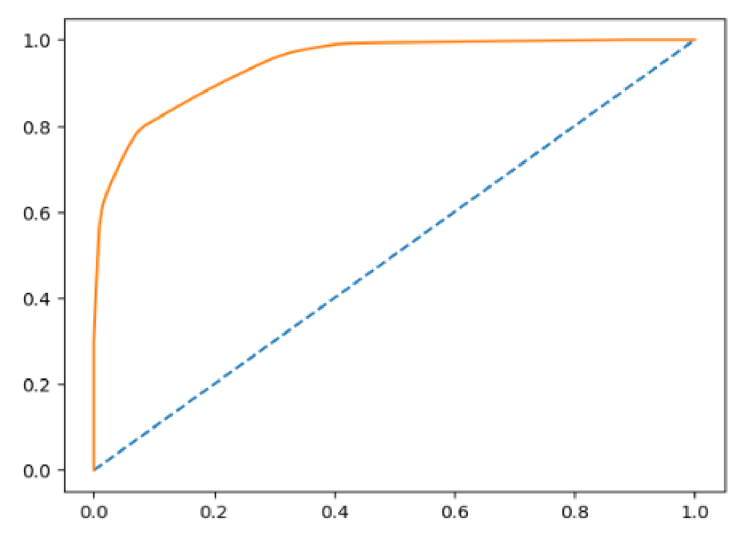
- Train data:

Table 21: Decision Tree with SMOTE – Confusion Matrix and Classification report of train data

```
0.8580024067388689
[[1520  142]
 [ 330 1332]]
              precision    recall  f1-score   support

           0       0.82      0.91      0.87      1662
           1       0.90      0.80      0.85      1662

    accuracy                           0.86      3324
   macro avg       0.86      0.86      0.86      3324
weighted avg       0.86      0.86      0.86      3324
```

Figure 35: Decision Tree with SMOTE - ROC Curve of train data

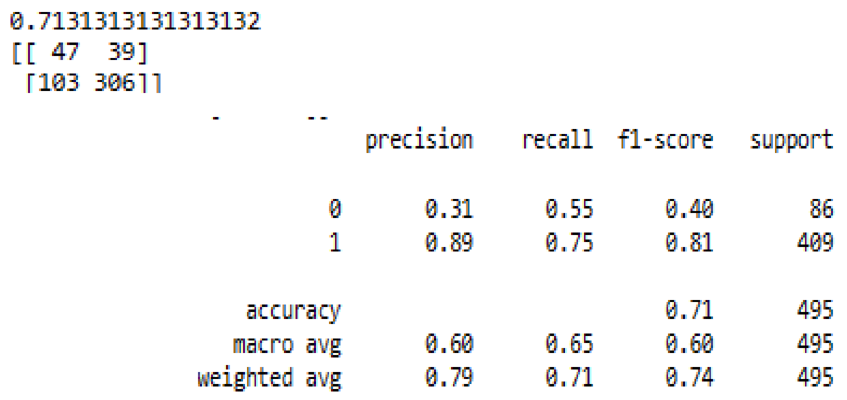
```
AUC: 0.944
```



- **ROC_AUC score of Train data is 0.944**

- **<u>Test</u>** <u>data</u>:

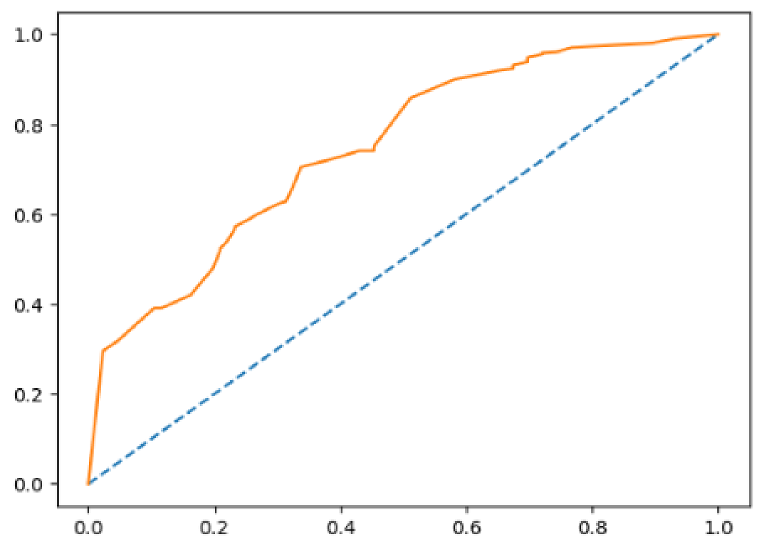
Table 22: Decision Tree with SMOTE – Confusion Matrix and Classification report of test data



```
0.7131313131313132
[[ 47  39]
 [103 306]]
              precision    recall  f1-score   support

           0       0.31      0.55      0.40        86
           1       0.89      0.75      0.81       409

    accuracy                           0.71       495
   macro avg       0.60      0.65      0.60       495
weighted avg       0.79      0.71      0.74       495
```

Figure 36: Decision Tree with SMOTE - ROC Curve of test data

```
AUC: 0.944
```



- **ROC_AUC score for Test data is 0.944**

Observations:

- The precision, recall and f1-scores for predicting losses are below 0.6.

- The accuracy score of both the train and test data is 86% and 71%.

9) **AdaBoosting**:

AdaBoosting is an ensemble learning used for classification problems.

We applied GridSearchCV in order to improve the performance, but the model performed well without tuning.

- **Accuracy:**

- Accuracy score of **Train** dataset is 0.89

- Accuracy score of **Test** dataset is 0.888

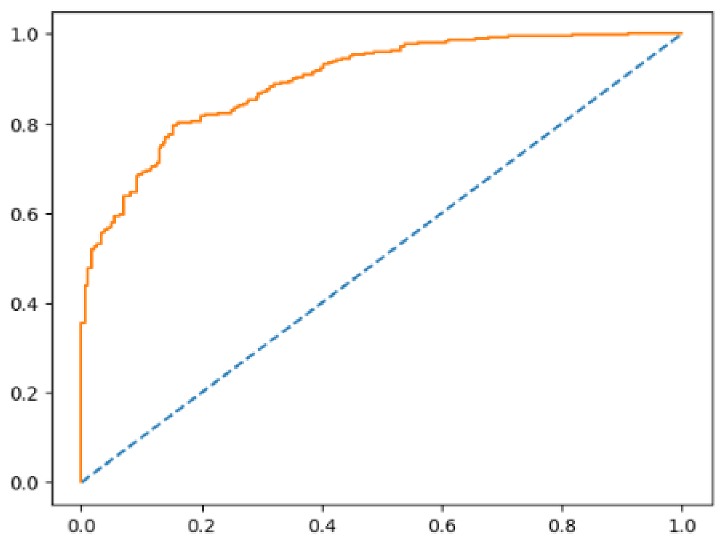
- Train data:

Table 23: AdaBoosting – Confusion Matrix and Classification report of train data

```
0.8933265925176946
[[ 144  172]
 [  39 1623]]
              precision    recall  f1-score   support

           0       0.79      0.46      0.58       316
           1       0.90      0.98      0.94      1662

    accuracy                           0.89      1978
   macro avg       0.85      0.72      0.76      1978
weighted avg       0.89      0.89      0.88      1978
```

Figure 37: AdaBoosting - ROC Curve of train data

```
AUC: 0.896
```



- **ROC_AUC score of Train data is 0.896**

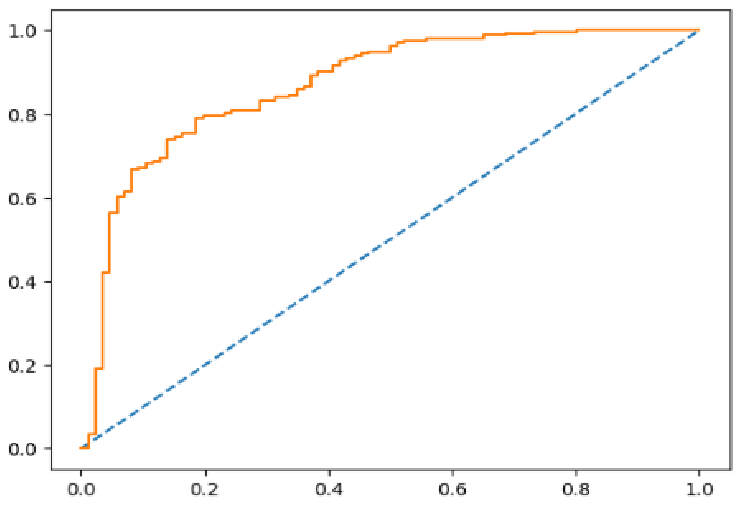
- Test data:

Table 24: AdaBoosting – Confusion Matrix and Classification report of test data

```
0.8888888888888888
[[ 41  45]
 [ 10 399]]
              precision    recall  f1-score   support

           0       0.80      0.48      0.60        86
           1       0.90      0.98      0.94       409

    accuracy                           0.89       495
   macro avg       0.85      0.73      0.77       495
weighted avg       0.88      0.89      0.88       495
```

Figure 38: AdaBoosting - ROC Curve of test data

```
AUC: 0.896
```



- **ROC_AUC score for Test data is 0.896**

Observations:

- ROC_AUC score for train and test data is 90%.

- The f1-score for predicting wins is 94%.

- The accuracy score is 89% for both the train and test data.

10) **AdaBoosting with SMOTE**:

Building a AdaBoosting model using the balanced data.

- **Accuracy:**

- Accuracy score of **Train** dataset is 0.89

- Accuracy score of **Test** dataset is 0.84

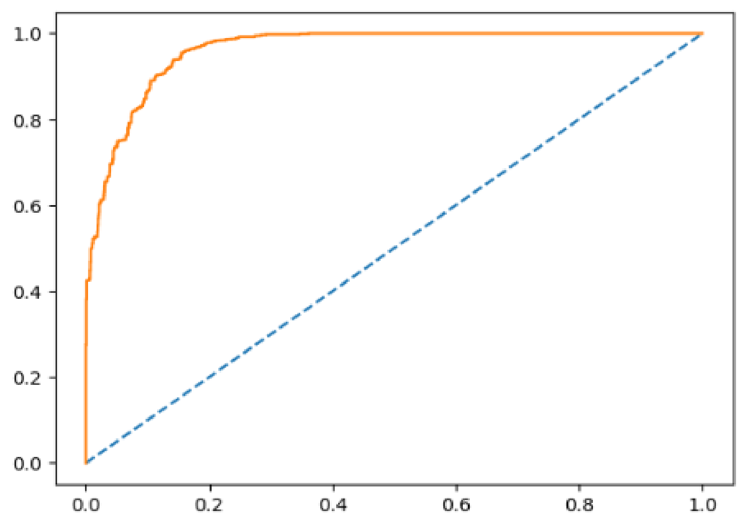
- **Train** data:

Table 25: AdaBoosting with SMOTE – Confusion Matrix and Classification report of train data

```
0.891395908543923
[[1442  220]
 [ 141 1521]]
              precision    recall  f1-score   support

           0       0.91      0.87      0.89      1662
           1       0.87      0.92      0.89      1662

    accuracy                           0.89      3324
   macro avg       0.89      0.89      0.89      3324
weighted avg       0.89      0.89      0.89      3324
```

Figure 39: AdaBoosting with SMOTE - ROC Curve of train data

```
AUC: 0.962
```



- **ROC_AUC score of Train data is 0.962**

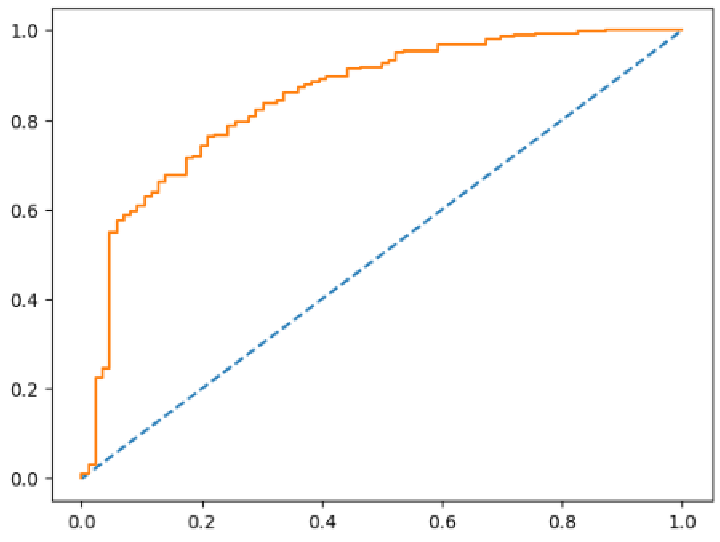
- **Test** data:

Table 26: AdaBoosting with SMOTE – Confusion Matrix and Classification report of test data

```
0.8424242424242424
[[ 52  34]
 [ 44 365]]

              precision    recall  f1-score   support

           0       0.54      0.60      0.57        86
           1       0.91      0.89      0.90       409

    accuracy                           0.84       495
   macro avg       0.73      0.75      0.74       495
weighted avg       0.85      0.84      0.85       495
```

Figure 40: AdaBoosting with SMOTE - ROC Curve of test data

```
AUC: 0.962
```



- **ROC_AUC score for Test data is 0.962**

Observations:

- The precision score for predicting losses has decreased from 91% to 54%, which denotes that the model performs poorly on predicting losses.

- ROC_AUC score for train and test data is 96%.

11) <u>SVM model</u>:

Support Vector Machine is a supervised learning algorithm for classification and regression problems.

SVM model needs the data to be scaled, hence we are using the scaled data.

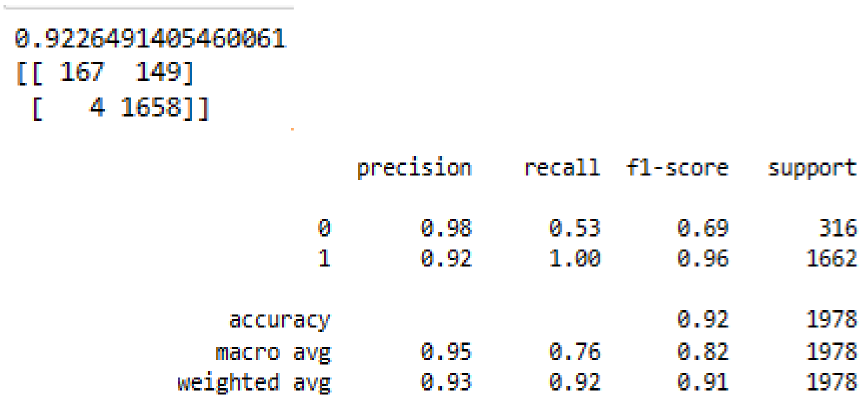
- **<u>Accuracy:</u>**

  - Accuracy score of **Train** dataset is 0.92

  - Accuracy score of **Test** dataset is 0.88

- <u>Train</u> <u>data:</u>

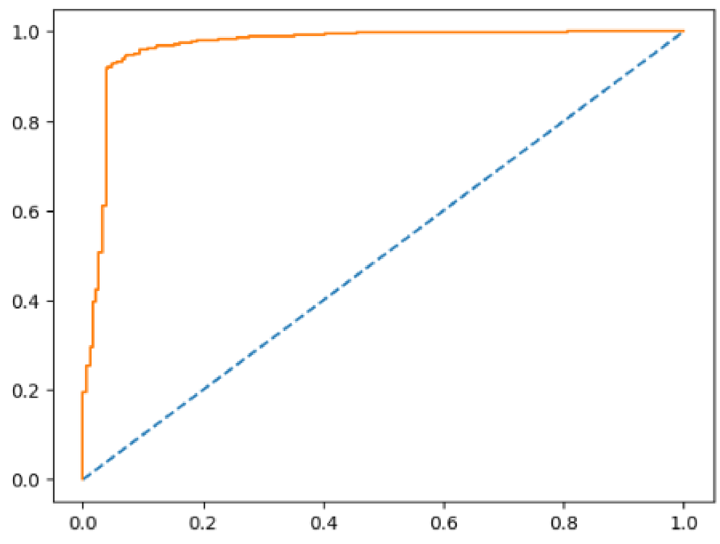
Table 27: SVM – Confusion Matrix and Classification report of train data



```
0.9226491405460061
[[ 167  149]
 [   4 1658]]

              precision    recall  f1-score   support

           0       0.98      0.53      0.69       316
           1       0.92      1.00      0.96      1662

    accuracy                           0.92      1978
   macro avg       0.95      0.76      0.82      1978
weighted avg       0.93      0.92      0.91      1978
```

Figure 41: SVM - ROC Curve of train data

```
AUC: 0.967
```



- **ROC_AUC score of Train data is 0.967**

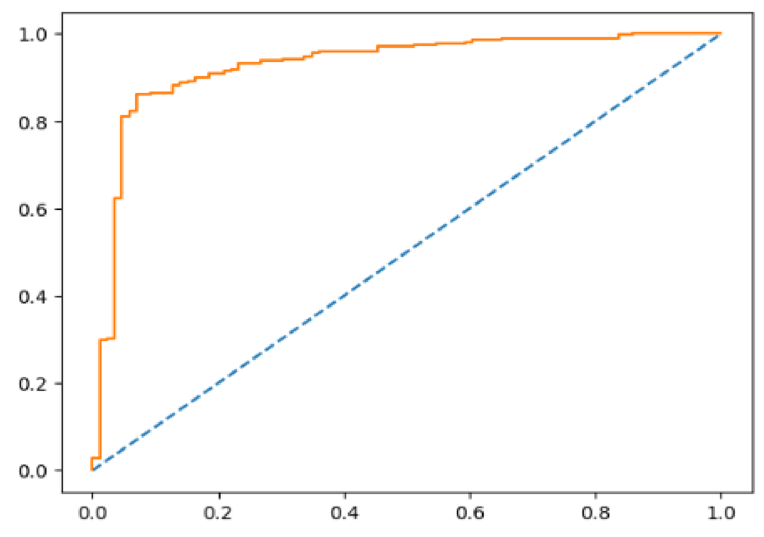
- **Test** data:

Table 28: SVM – Confusion Matrix and Classification report of test data

```
0.8828282828282829
[[ 34  52]
 [  6 403]]
              precision    recall  f1-score   support

           0       0.85      0.40      0.54        86
           1       0.89      0.99      0.93       409

    accuracy                           0.88       495
   macro avg       0.87      0.69      0.74       495
weighted avg       0.88      0.88      0.86       495
```

Figure 42: SVM - ROC Curve of test data

AUC: 0.967



- **ROC_AUC score for Test data is 0.967**

Observations:

- The recall score for predicting losses is 53% and 40% on train and test data.

- ROC_AUC score for train and test data is 97%.

- The accuracy score is 92% on the train and 88% on the test data.

12) **<u>SVM with SMOTE</u>**:

Building a SVM model using the scaled and balanced data.

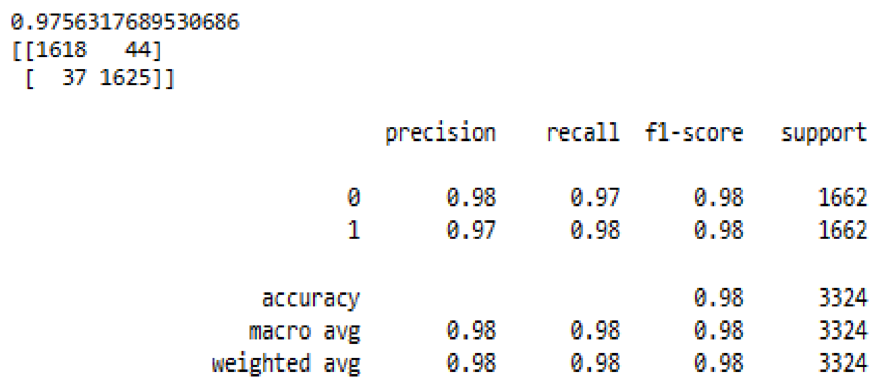
- **<u>Accuracy:</u>**

- Accuracy score of **Train** dataset is 0.98

- Accuracy score of **Test** dataset is 0.89

- <u>Train</u> <u>data:</u>

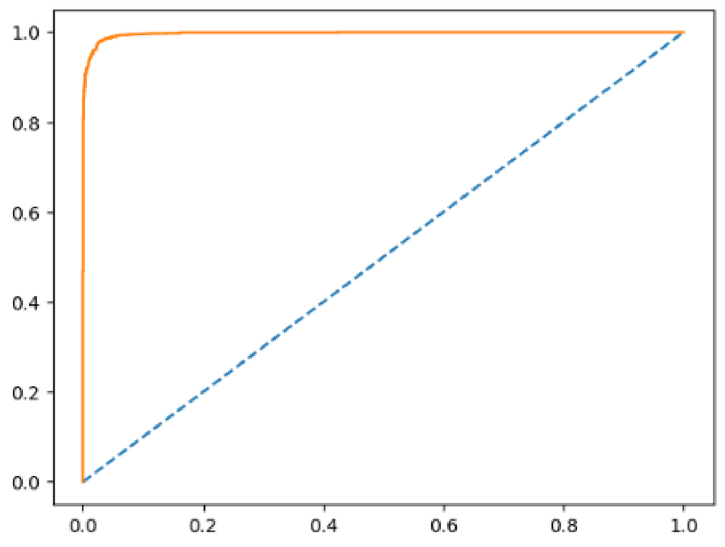
Table 29: SVM with SMOTE – Confusion Matrix and Classification report of train data



```
0.9756317689530686
[[1618   44]
 [  37 1625]]

              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1662
           1       0.97      0.98      0.98      1662

    accuracy                           0.98      3324
   macro avg       0.98      0.98      0.98      3324
weighted avg       0.98      0.98      0.98      3324
```

Figure 43: SVM with SMOTE - ROC Curve of train data

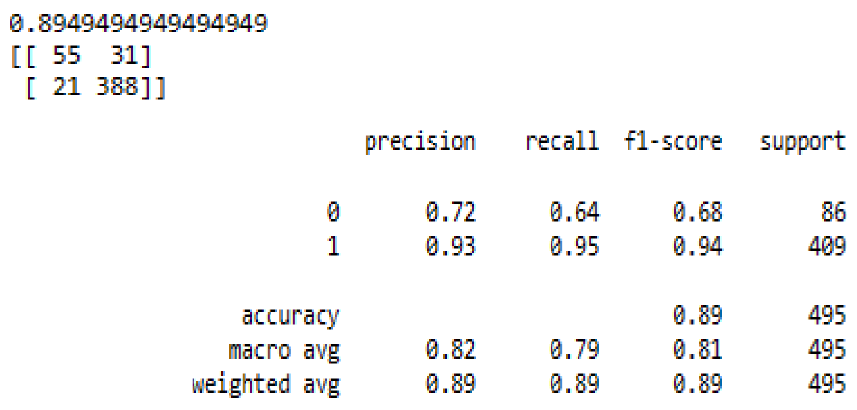
```
AUC: 0.997
```



- **ROC_AUC score of Train data is 0.997**

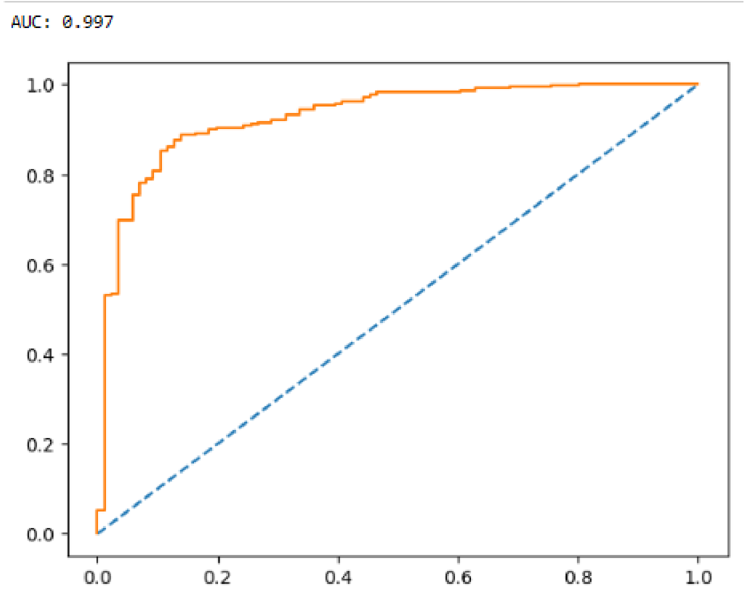
- **<u>Test</u>** <u>data</u>:

Table 30: SVM with SMOTE – Confusion Matrix and Classification report of test data



```
0.8949494949494949
[[ 55  31]
 [ 21 388]]
              precision    recall  f1-score   support

           0       0.72      0.64      0.68        86
           1       0.93      0.95      0.94       409

    accuracy                           0.89       495
   macro avg       0.82      0.79      0.81       495
weighted avg       0.89      0.89      0.89       495
```

Figure 44: SVM with SMOTE - ROC Curve of test data



```
AUC: 0.997
```

- **ROC_AUC score for Test data is 0.997**

Observations:

- The recall rate for predicting losses has reduced to 64% (test) from 97% (train) whereas recall is too good while predicting wins, i.e., 98% on train data and 95% on test data.

- The accuracy score is 98% on the train and 89% on the test data.

- The f1-scores on the test data are 68% for predicting losses and 94% for predicting wins.

Table 31: Models and model performances

Models	**Model Performances**			
			Train	**Test**
1. **Random Forest**	Accuracy		0.93	0.89
	AUC		0.998	0.998
	F1-score	**0**	0.70	0.53
		1	0.96	0.94
	Precision	**0**	1.00	0.94
		1	0.92	0.88
	Recall	**0**	0.53	0.37
		1	1.00	1.00
2. **Random Forest with SMOTE**	Accuracy		0.99	0.92
	AUC		0.999	0.999
	F1-score	**0**	0.99	0.75
		1	0.99	0.95
	Precision	**0**	0.99	0.86
		1	0.99	0.93
	Recall	**0**	0.98	0.66
		1	0.99	0.98

3. Naive Bayes model	Accuracy		0.75	0.75
	AUC		0.695	0.695
	F1-score	**0**	0.39	0.42
		1	0.84	0.84
	Precision	**0**	0.32	0.35
		1	0.89	0.89
	Recall	**0**	0.50	0.53
		1	0.80	0.79
4. Naive Bayes with SMOTE	Accuracy		0.58	0.39
	AUC		0.750	0.750
	F1-score	**0**	0.68	0.34
		1	0.38	0.44
	Precision	**0**	0.55	0.21
		1	0.72	0.94
	Recall	**0**	0.90	0.92
		1	0.26	0.28
5. KNN model	Accuracy		0.87	0.85
	AUC		0.868	0.868
	F1-score	**0**	0.31	0.30
		1	0.93	0.92

	Precision	**0**	0.88	0.84
		1	0.87	0.85
	Recall	**0**	0.19	0.19
		1	1.00	0.99
6. KNN model with SMOTE	Accuracy		0.75	0.56
	AUC		0.987	0.987
	F1-score	**0**	0.80	0.43
		1	0.67	0.64
	Precision	**0**	0.67	0.28
		1	0.99	0.98
	Recall	**0**	1.00	0.95
		1	0.51	0.47
7. Decision Tree model	Accuracy		0.89	0.85
	AUC		0.875	0.875
	F1-score	**0**	0.58	0.50
		1	0.94	0.91
	Precision	**0**	0.74	0.58
		1	0.91	0.89
	Recall	**0**	0.48	0.44
		1	0.97	0.93

8. Decision Tree with SMOTE	Accuracy		0.86	0.71
	AUC		0.944	0.944
	F1-score	**0**	0.87	0.40
		1	0.85	0.81
	Precision	**0**	0.82	0.31
		1	0.90	0.89
	Recall	**0**	0.91	0.55
		1	0.80	0.75
9. AdaBoosting	Accuracy		0.89	0.89
	AUC		0.896	0.896
	F1-score	**0**	0.58	0.60
		1	0.94	0.94
	Precision	**0**	0.79	0.80
		1	0.90	0.90
	Recall	**0**	0.46	0.48
		1	0.98	0.98
10. AdaBoosting with SMOTE	Accuracy		0.89	0.84
	AUC		0.962	0.962
	F1-score	**0**	0.89	0.57
		1	0.89	0.90

	Precision	**0**	0.91	0.54
		1	0.87	0.91
	Recall	**0**	0.87	0.60
		1	0.92	0.89
11. SVM model	Accuracy		0.92	0.88
	AUC		0.967	0.967
	F1-score	**0**	0.69	0.54
		1	0.96	0.93
	Precision	**0**	0.98	0.85
		1	0.92	0.89
	Recall	**0**	0.53	0.40
		1	1.00	0.99
12. SVM with SMOTE	Accuracy		0.98	0.89
	AUC		0.997	0.997
	F1-score	**0**	0.98	0.68
		1	0.98	0.94
	Precision	**0**	0.98	0.72
		1	0.97	0.93
	Recall	**0**	0.97	0.64
		1	0.98	0.95

Best Performing model:

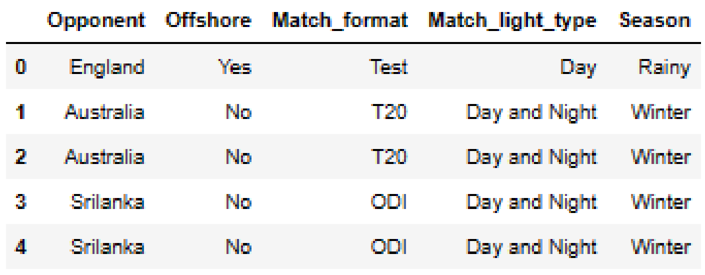
Random Forest model with SMOTE is the best performing model when compared with other models. The model may be slightly overfitted towards predicting losses; however, the primary focus is on predicting wins. Also, we used balanced data for predicting wins.

We opted best performing model, based on the **f1-score** (95% on test data), **precision** (93% on test data), and **recall** scores (98% on test data) for predicting wins.

Let us now, predict the wins for upcoming **five** matches (problem statement).

- Creating new data frame as per the problem statement.

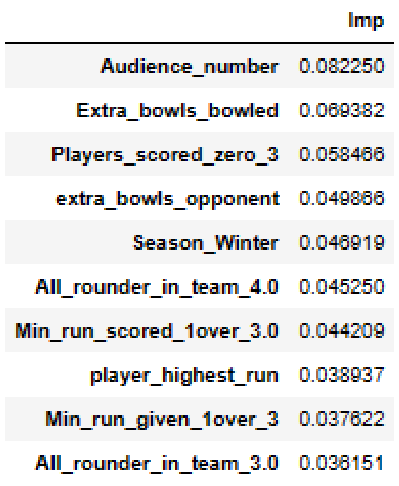
Table 32: New dataframe



	Opponent	Offshore	Match_format	Match_light_type	Season
0	England	Yes	Test	Day	Rainy
1	Australia	No	T20	Day and Night	Winter
2	Australia	No	T20	Day and Night	Winter
3	Srilanka	No	ODI	Day and Night	Winter
4	Srilanka	No	ODI	Day and Night	Winter

- Created dummy variables from categorical features.
- Top 10 Important features used in the best performing model.

Table 33: Top 10 Important features



	Imp
Audience_number	0.082250
Extra_bowls_bowled	0.069382
Players_scored_zero_3	0.058466
extra_bowls_opponent	0.049866
Season_Winter	0.046919
All_rounder_in_team_4.0	0.045250
Min_run_scored_1over_3.0	0.044209
player_highest_run	0.038937
Min_run_given_1over_3	0.037622
All_rounder_in_team_3.0	0.036151

- Concatenating new dataframe to the test data and replacing NaN values with 0.

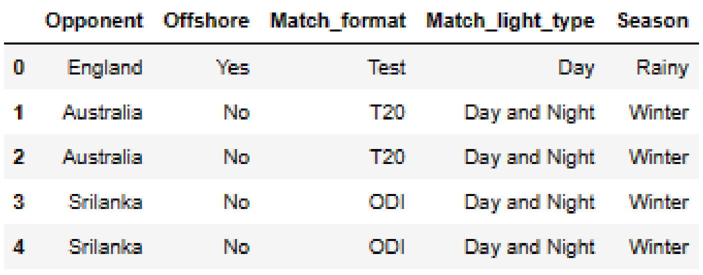
<u>Problem statement</u>

a) 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.

b) 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.

c) 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.

Table 34: Table showing upcoming matches



	Opponent	Offshore	Match_format	Match_light_type	Season
0	England	Yes	Test	Day	Rainy
1	Australia	No	T20	Day and Night	Winter
2	Australia	No	T20	Day and Night	Winter
3	Srilanka	No	ODI	Day and Night	Winter
4	Srilanka	No	ODI	Day and Night	Winter

As per the **best performing model** (Random Forest with SMOTE model)

1) 1st match – **Loss**
2) 2nd match – **Win**
3) 3rd match – **Win**
4) 4th match – **Win**
5) 5th match - **Win**

<u>Recommendation</u>:

- The first match against England can be won by adding at least 3 all-rounders.
- BCCI can select players based on their performance in offshore matches and their adaptability to the particular weather conditions.

- The initial selection (batting or bowling) also needs to be considered.
- To enhance the accuracy of the model, we may require additional information such as bowling statistics (pace or spin).

---------------------------------**PROJECT NOTES – 2-----THE END**-------------------------------