ODI_prediction_without_code

December 7, 2021

1 Problem Statement

To predict future ODI cricket match winner based on previous year's match result.

2 Introduction

- Cricket is one of the most popular sports in world, especially in India. The game is highly uncertain.
- It is the sport which generate high revenue.
- What if the winner team of the match can be predicted before the match, even have begin?
- Because we are predicting a output which is categorical value, that is the probelm is a classification problem.

2.1 Dataset Desciption

The dataset folder contains the following file:

ODI-data-1971-2017.csv = 3932 rows x 7 columns

Columns Provided in the Dataset

1. Scorecard
2. Team 1
3. Team 2
4. Winner
5. Margin
6. Ground
7. Match Date

ODI-data-2017-2021.csv = 495 x 8 columns

1. Scorecard
2. Team 1
3. Team 2
4. Winner
5. Margin
6. Ground
7. Match Date
8. Unnamed: 0

3 For this problem we will be using samples from 2010 to 2021.

[24]:
```
#importing necessary libraries
```

[25]:
```
# load the datasets ODI_data_1971_2017 and ODI_data_2017_2021
```

[26]:
```
# ODI_data_1971_2017 data
```

[26]:

	Scorecard	Team 1	…	Ground	Match Date
0	ODI # 1	Australia	…	Melbourne	Jan 5, 1971
1	ODI # 2	England	…	Manchester	Aug 24, 1972
2	ODI # 3	England	…	Lord's	Aug 26, 1972
3	ODI # 4	England	…	Birmingham	Aug 28, 1972
4	ODI # 5	New Zealand	…	Christchurch	Feb 11, 1973
…	…	…	…	…	…
3927	ODI # 3928	India	…	Mumbai	Oct 22, 2017
3928	ODI # 3929	South Africa	…	East London	Oct 22, 2017
3929	ODI # 3930	Pakistan	…	Sharjah	Oct 23, 2017
3930	ODI # 3931	India	…	Pune	Oct 25, 2017
3931	ODI # 3932	India	…	Kanpur	Oct 29, 2017

[3932 rows x 7 columns]

[27]:
```
# ODI_data_2017_2021 data
```

[27]:

	Unnamed: 0	Scorecard	…	Ground	Match Date
0	0	ODI # 3817	…	Brisbane	Jan 13, 2017
1	1	ODI # 3818	…	Melbourne	Jan 15, 2017
2	2	ODI # 3819	…	Pune	Jan 15, 2017
3	3	ODI # 3820	…	Perth	Jan 19, 2017
4	4	ODI # 3821	…	Cuttack	Jan 19, 2017
..	…	…	…	…	…
490	490	ODI # 4309	…	Colombo (RPS)	Jul 20, 2021
491	491	ODI # 4310	…	Bridgetown	Jul 20, 2021
492	492	ODI # 4311	…	Bridgetown	Jul 22-24, 2021
493	493	ODI # 4312	…	Colombo (RPS)	Jul 23, 2021
494	494	ODI # 4313	…	Bridgetown	Jul 26, 2021

[495 rows x 8 columns]

Droping rows of year 2017 pesent in ODI 2017 to 2021 dataset wich are alredy present in ODI 1971 to 2017 data

[28]:
```
# Droping rows with index range 0 to 116

# Reset index of the dataframe

# Drop the extra 'index' column from dataframe
```

[29]:
```
# Load ODI_data_1971_2021 the dataset
```

[30]:
```
# ODI_data_1971_2021 data
```

[30]:
```
      Scorecard       Team 1  …       Match Date Unnamed: 0
0       ODI # 1    Australia  …      Jan 5, 1971        NaN
1       ODI # 2      England  …     Aug 24, 1972        NaN
2       ODI # 3      England  …     Aug 26, 1972        NaN
3       ODI # 4      England  …     Aug 28, 1972        NaN
4       ODI # 5  New Zealand  …     Feb 11, 1973        NaN
…          …          …  …              …        …
4306  ODI # 4309    Sri Lanka  …     Jul 20, 2021      490.0
4307  ODI # 4310  West Indies  …     Jul 20, 2021      491.0
4308  ODI # 4311  West Indies  …  Jul 22-24, 2021      492.0
4309  ODI # 4312    Sri Lanka  …     Jul 23, 2021      493.0
4310  ODI # 4313  West Indies  …     Jul 26, 2021      494.0

[4311 rows x 8 columns]
```

[31]:
```
# Drop "Unnamed: 0" column
```

[32]:
```
# Drop values from 0 to 2936 as it all conten samples of before year 2010
# Store all remaining smaple in new dataframe
```

[33]:
```
# Reset index of the new dataframe
```

[34]:
```
# Remove extra 'index' Column from ew dataframe
```

[35]:
```
# ODI_data_2010_2021 data
```

[35]:
```
      Scorecard       Team 1  …         Ground       Match Date
0     ODI # 2937   Bangladesh  …          Dhaka      Jan 4, 2010
1     ODI # 2938        India  …          Dhaka      Jan 5, 2010
2     ODI # 2939   Bangladesh  …          Dhaka      Jan 7, 2010
3     ODI # 2940   Bangladesh  …          Dhaka      Jan 8, 2010
4     ODI # 2941        India  …          Dhaka     Jan 10, 2010
…          …          …  …              …              …
1370  ODI # 4309    Sri Lanka  …  Colombo (RPS)     Jul 20, 2021
1371  ODI # 4310  West Indies  …     Bridgetown     Jul 20, 2021
1372  ODI # 4311  West Indies  …     Bridgetown  Jul 22-24, 2021
1373  ODI # 4312    Sri Lanka  …  Colombo (RPS)     Jul 23, 2021
1374  ODI # 4313  West Indies  …     Bridgetown     Jul 26, 2021

[1375 rows x 7 columns]
```

4 Basic EDA

4.1 Identifying the number of features or columns

4.2 Know all the names of the columns¶

[36]:
```
# Check all column names
```

[36]:
```
Index(['Scorecard', 'Team 1', 'Team 2', 'Winner', 'Margin', 'Ground',
       'Match Date'],
      dtype='object')
```

4.3 Knows more about the data in the columns like data type it contains and total samples of each

[37]:
```
# Check info of complete dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1375 entries, 0 to 1374
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Scorecard   1375 non-null   object
 1   Team 1      1375 non-null   object
 2   Team 2      1375 non-null   object
 3   Winner      1375 non-null   object
 4   Margin      1326 non-null   object
 5   Ground      1375 non-null   object
 6   Match Date  1375 non-null   object
dtypes: object(7)
memory usage: 75.3+ KB
```

After checking the Dtypes of all the columns 1. object - String values 3. All the columns are of string datatype

4.4 Know more mathematical relations of the dataset like count, min, max values, standarad deviation values, mean and different percentile values

[38]:
```
# For more information on the dataset like the total count in all the columns
```

[38]:

	Scorecard	Team 1	Team 2	Winner	Margin	Ground	Match Date
count	1375	1375	1375	1375	1326	1375	1375
unique	1375	23	23	25	213	124	1150
top	ODI # 3422	England	Pakistan	India	6 wickets	Dhaka	Jul 10, 2010
freq	1	141	175	164	126	87	4

4.5 Get the total number of samples in the dataset using the len() function

[39]:
```
# print len of the datset
```

```
ODI dataset length:  1375
```

4.6 Get unique values

[40]:
```
# loop through datatset to find count of unique values of each column
```

```
Scorecard : 1375
Team 1 : 23
Team 2 : 23
Winner : 25
Margin : 214
Ground : 124
Match Date : 1150
```

4.7 Counting the total number of missing values

[41]:
```
# Check for missing values in all the columnns of the dataset
```

[41]:
```
Scorecard      0
Team 1         0
Team 2         0
Winner         0
Margin        49
Ground         0
Match Date     0
dtype: int64
```

By the observation gather from the ODI_data_2010_2021.info() , we can know there are missing values in the "Margin" column of dataset

4.8 Chi-square Test

1. The Chi Square statistic is commonly used for testing relationships between categorical variables.
2. The null hypothesis of the Chi-Square test is that no relationship exists on the categorical variables in the population; they are independent.
3. Example: Is there any significant relationship between gender and education qualification?
4. The Chi-Square statistic is most commonly used to evaluate Tests of Independence when using a crosstabulation.
5. Crosstabulation presents the distributions of two categorical variables simultaneously, with the intersections of the categories of the variables appearing in the cells of the table. that is values of one variable represents the row and other's value represents the column.

6. Formula: x^2 = Summation of((observed value - Expected value)^2/Expected value)
7. The Chi-Square statistic is based on the difference between what is actually observed in the data and what would be expected if there was truly no relationship between the variables.
8. This statistic can be evaluated by comparing the actual value against a critical value found in a Chi-Square distribution (where degrees of freedom is calculated as of rows – 1 x columns – 1), but it is easier to simply examine the p-value.
9. To make a conclusion about the hypothesis with 95% confidence. Significance(p value of the Chi-square statistic) should be less than 0.05.
    1. Alpha level = 0.05(i.e 5%) 95% confidence about conclusion and 5% risk of not making a correct conclusion.
    2. Interpret the key results for Chi-Square Test for Association

       Determine whether the association between the variables is statistically significant.

       Examine the differences between expected counts and observed counts to determine which variable levels may have the most impact on association.

[42]:
```
# Import necessary libraries needed for performing Chi-square test


# Helper function for performing chi-square test


    #Contingency Table


    #Observed Values


    #Expected Values

    #Degree of Freedom

    #Significance Level 5%


    #chi-square statistic


    #critical_value


    #p-value
```

```
    # check condition based on chi_square_statistic and critical value for␣
↪hypothesis rejection



    # check condition based on p value and alpha for hypothesis rejection
```

[43]:
```
# looping on whole dataset for performing chi-square test

    # Inner loop on whol edataset columns name

        # checking condition that both column names are should not be equal

            # passing both column names to chi-sqaure performance function
```

```
chi-square test on:  Scorecard   Team 1

Degree of Freedom:  30228
Significance level:  0.05
chi-square statistic:  2549.9999999999623
critical_value: 30633.567183419527
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Scorecard   Team 2

Degree of Freedom:  30228
Significance level:  0.05
chi-square statistic:  2592.0000000000355
critical_value: 30633.567183419527
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Scorecard   Winner

Degree of Freedom:  32976
Significance level:  0.05
chi-square statistic:  2557.000000000013
critical_value: 33399.55082702824
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
```

```
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Scorecard   Margin

Degree of Freedom:  280900
Significance level:  0.05
chi-square statistic:  2616.999999999969
critical_value: 282134.0083642959
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Scorecard   Ground

Degree of Freedom:  169002
Significance level:  0.05
chi-square statistic:  2704.0000000000127
critical_value: 169959.42251265424
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Scorecard   Match Date

Degree of Freedom:  1578726
Significance level:  0.05
chi-square statistic:  2748.000000000029
critical_value: 1581649.91327901
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Team 1   Scorecard

Degree of Freedom:  30228
Significance level:  0.05
chi-square statistic:  19.91373640273211
critical_value: 30633.567183419527
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Team 1   Team 2
```

```
Degree of Freedom:  484
Significance level:  0.05
chi-square statistic:  213.8780025679467
critical_value: 536.2873901981108
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 1   Winner

Degree of Freedom:  528
Significance level:  0.05
chi-square statistic:  876.277770083678
critical_value: 582.5640658777064
p-value: 0.0
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 1   Margin

Degree of Freedom:  4664
Significance level:  0.05
chi-square statistic:  40.64382263280688
critical_value: 4823.991425012426
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 1   Ground

Degree of Freedom:  2706
Significance level:  0.05
chi-square statistic:  484.67018600844995
critical_value: 2828.132105142811
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 1   Match Date

Degree of Freedom:  25278
Significance level:  0.05
chi-square statistic:  25.726223916532902
```

```
critical_value: 25648.973302649498
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 2   Scorecard

Degree of Freedom:  30228
Significance level:  0.05
chi-square statistic:  15.29559748427673
critical_value: 30633.567183419527
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 2   Team 1

Degree of Freedom:  484
Significance level:  0.05
chi-square statistic:  216.29691003320755
critical_value: 536.2873901981108
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 2   Winner

Degree of Freedom:  528
Significance level:  0.05
chi-square statistic:  481.78769617380294
critical_value: 582.5640658777064
p-value: 0.9257934239602428
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Team 2   Margin

Degree of Freedom:  4664
Significance level:  0.05
chi-square statistic:  66.4296828518047
critical_value: 4823.991425012426
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
```

```
------------------------------------------------------------------

chi-square test on:  Team 2   Ground

Degree of Freedom:  2706
Significance level:  0.05
chi-square statistic:  303.54958808046615
critical_value: 2828.132105142811
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
------------------------------------------------------------------

chi-square test on:  Team 2   Match Date

Degree of Freedom:  25278
Significance level:  0.05
chi-square statistic:  83.80508474576271
critical_value: 25648.973302649498
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
------------------------------------------------------------------

chi-square test on:  Winner   Scorecard

Degree of Freedom:  32976
Significance level:  0.05
chi-square statistic:  21.305084745762716
critical_value: 33399.55082702824
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
------------------------------------------------------------------

chi-square test on:  Winner   Team 1

Degree of Freedom:  528
Significance level:  0.05
chi-square statistic:  880.3020763165255
critical_value: 582.5640658777064
p-value: 0.0
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
------------------------------------------------------------------

chi-square test on:  Winner   Team 2
```

```
Degree of Freedom:  528
Significance level:  0.05
chi-square statistic:  477.7504573642663
critical_value: 582.5640658777064
p-value: 0.9426723687563274
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Winner   Margin

Degree of Freedom:  5088
Significance level:  0.05
chi-square statistic:  45.38295161044218
critical_value: 5255.055745020391
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Winner   Ground

Degree of Freedom:  2952
Significance level:  0.05
chi-square statistic:  368.09143741433684
critical_value: 3079.5131987381296
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Winner   Match Date

Degree of Freedom:  27576
Significance level:  0.05
chi-square statistic:  40.90254237288136
critical_value: 27963.418723067858
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------

chi-square test on:  Margin   Scorecard

Degree of Freedom:  280900
Significance level:  0.05
chi-square statistic:  26.56325507180305
critical_value: 282134.0083642959
```

```
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Margin   Team 1

Degree of Freedom:  4664
Significance level:  0.05
chi-square statistic:  342.8192553673018
critical_value: 4823.991425012426
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Margin   Team 2

Degree of Freedom:  4664
Significance level:  0.05
chi-square statistic:  362.45840742355006
critical_value: 4823.991425012426
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Margin   Winner

Degree of Freedom:  5088
Significance level:  0.05
chi-square statistic:  347.3744205396754
critical_value: 5255.055745020391
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Margin   Ground

Degree of Freedom:  26076
Significance level:  0.05
chi-square statistic:  344.32333119106755
critical_value: 26452.765724275003
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------
```

```
chi-square test on:  Margin   Match Date

Degree of Freedom:  235320
Significance level:  0.05
chi-square statistic:  275.59252336448606
critical_value: 236449.55822165185
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Ground   Scorecard

Degree of Freedom:  169002
Significance level:  0.05
chi-square statistic:  29.60919540229886
critical_value: 169959.42251265424
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Ground   Team 1

Degree of Freedom:  2706
Significance level:  0.05
chi-square statistic:  1224.7705065418688
critical_value: 2828.132105142811
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Ground   Team 2

Degree of Freedom:  2706
Significance level:  0.05
chi-square statistic:  508.9954875913096
critical_value: 2828.132105142811
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Ground   Winner

Degree of Freedom:  2952
```

```
Significance level:  0.05
chi-square statistic:  621.3048963391725
critical_value: 3079.5131987381296
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Ground   Margin

Degree of Freedom:  26076
Significance level:  0.05
chi-square statistic:  270.15301474881716
critical_value: 26452.765724275003
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Ground   Match Date

Degree of Freedom:  141327
Significance level:  0.05
chi-square statistic:  292.09722222222194
critical_value: 142202.62550573327
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Match Date   Scorecard

Degree of Freedom:  1578726
Significance level:  0.05
chi-square statistic:  2748.0000000000728
critical_value: 1581649.91327901
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
----------------------------------------------------------------------

chi-square test on:  Match Date   Team 1

Degree of Freedom:  25278
Significance level:  0.05
chi-square statistic:  2045.2222222222158
critical_value: 25648.973302649498
p-value: 1.0
```

```
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Match Date   Team 2

Degree of Freedom:  25278
Significance level:  0.05
chi-square statistic:  2080.5849978621004
critical_value: 25648.973302649498
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Match Date   Winner

Degree of Freedom:  27576
Significance level:  0.05
chi-square statistic:  2112.034722222227
critical_value: 27963.418723067858
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Match Date   Margin

Degree of Freedom:  235320
Significance level:  0.05
chi-square statistic:  2082.180000000031
critical_value: 236449.55822165185
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------

chi-square test on:  Match Date   Ground

Degree of Freedom:  141327
Significance level:  0.05
chi-square statistic:  2352.8095238095384
critical_value: 142202.62550573327
p-value: 1.0
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
-----------------------------------------------------------------
```

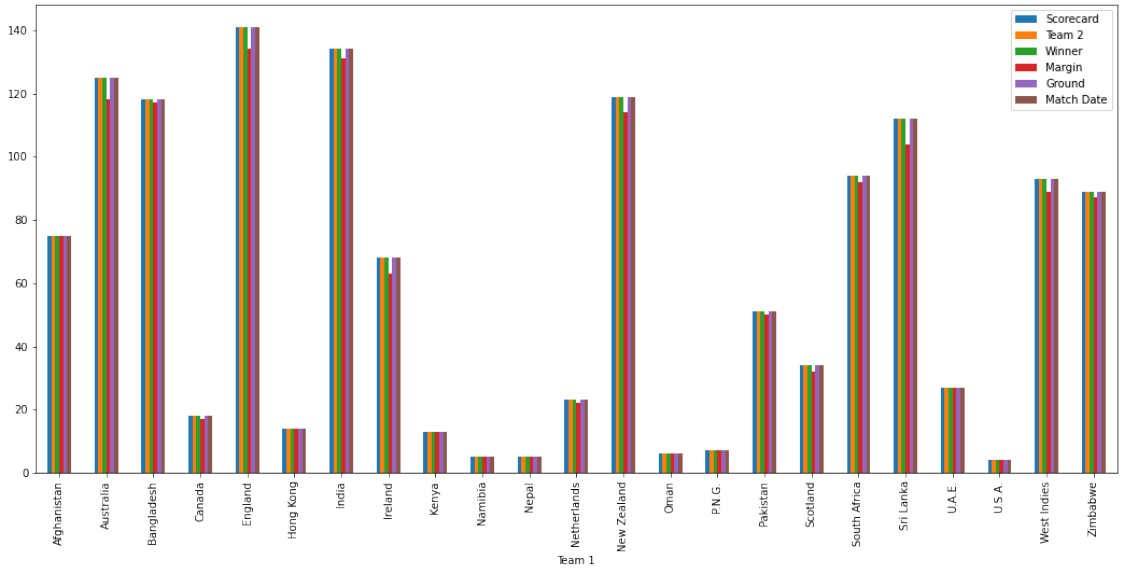
From above chi-square test:

There is correlation between Team 1 and Winner data.

4.9 groupby

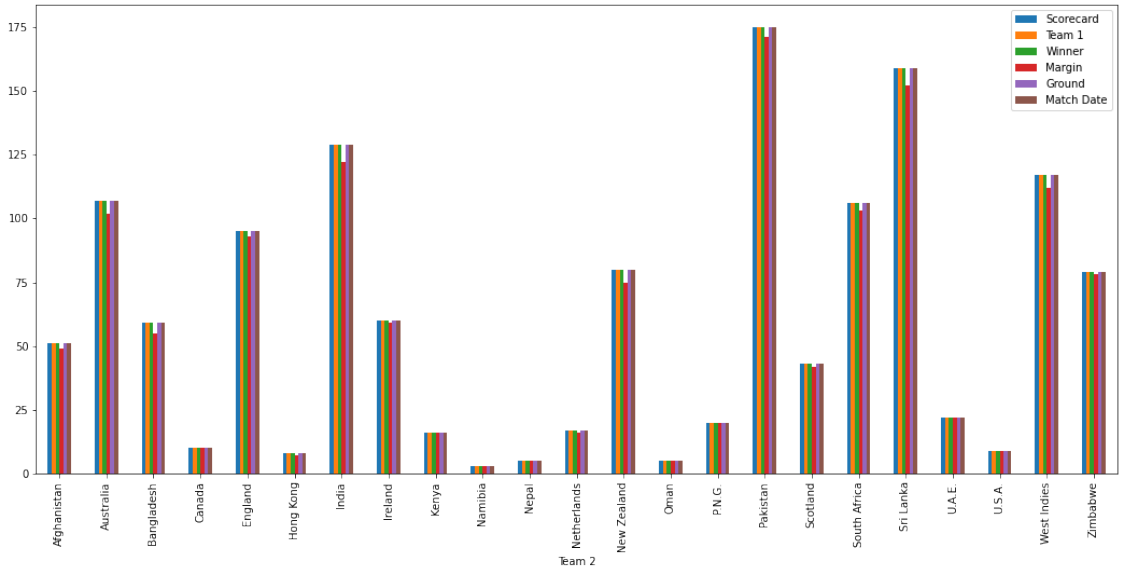
You can use groupby to chunk up your data into subsets for further analysis.

[44]:
```
# group data by Team 1 and plot count plot
```



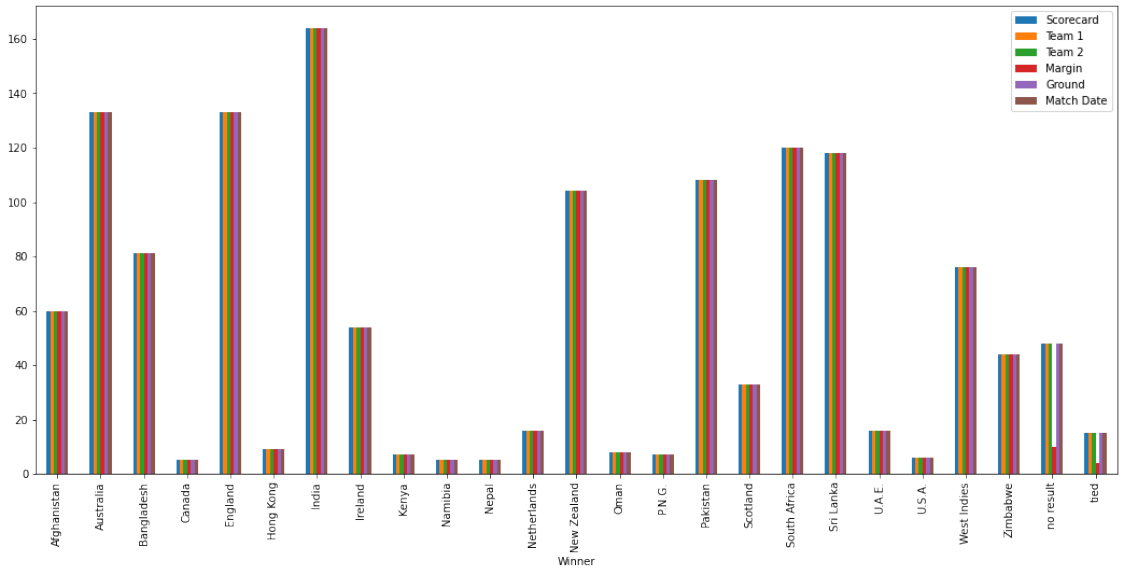
from above graph: 1. There are more samples of team 1 as Australia, Bangladesh, England, India, New Zealand, West Indies, Zimbabwe as compared to other teams 2. Samples of team 1 as Namibia, Oman, Nepal, P.N.G, U.S.A are very less

[45]:
```
# group data by Team 2 and plot count plot
```



from above graph: 1. There are more samples of team 2 as Afghanistan, Australia, Bangladesh, England, India, Pakistan, Ireland, New Zealand, West Indies, Sri Lanks, South Africa Zimbabwe as compared to other teams 2. Samples of team 2 as Namibia, Oman, Nepal, P.N.G, U.S.A, kenya, Hong Kong, Canada are very less

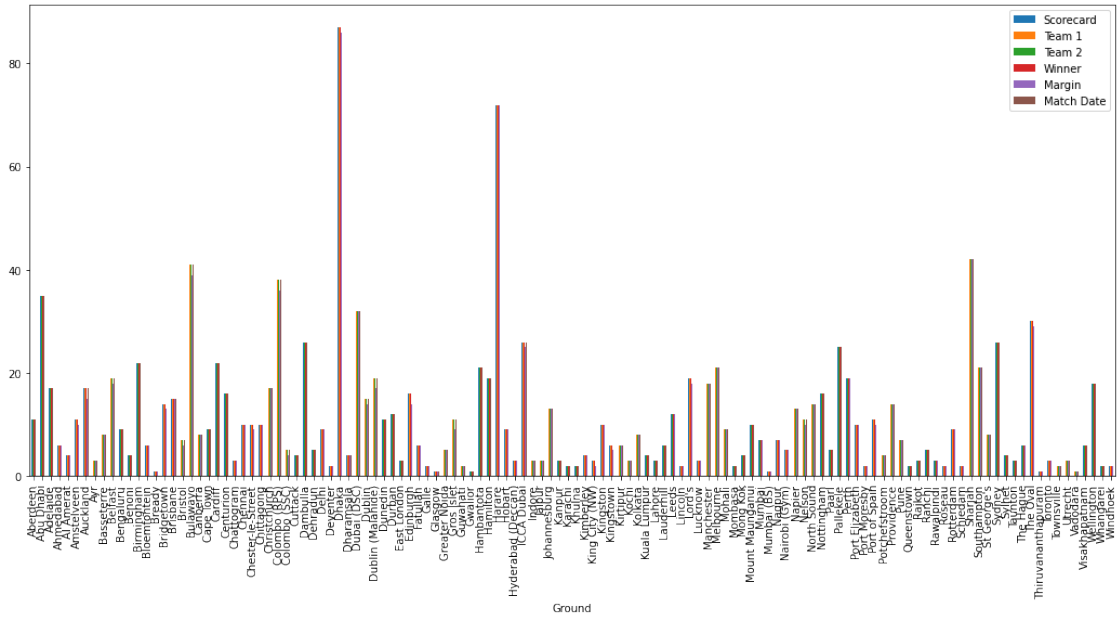
[46]:
```
# group data by Winner and plot count plot
```



from above graph: 1. There are more samples of Winner as Australia, Bangladesh, England, India, Pakistan, New Zealand, West Indies, Zimbabwe, South Africa, Sri Lanka as compared to other teams 2. Samples of Winner as Canada, Hong Kong, Kenya, Namibia, Oman, Nepal, P.N.G, U.S.A

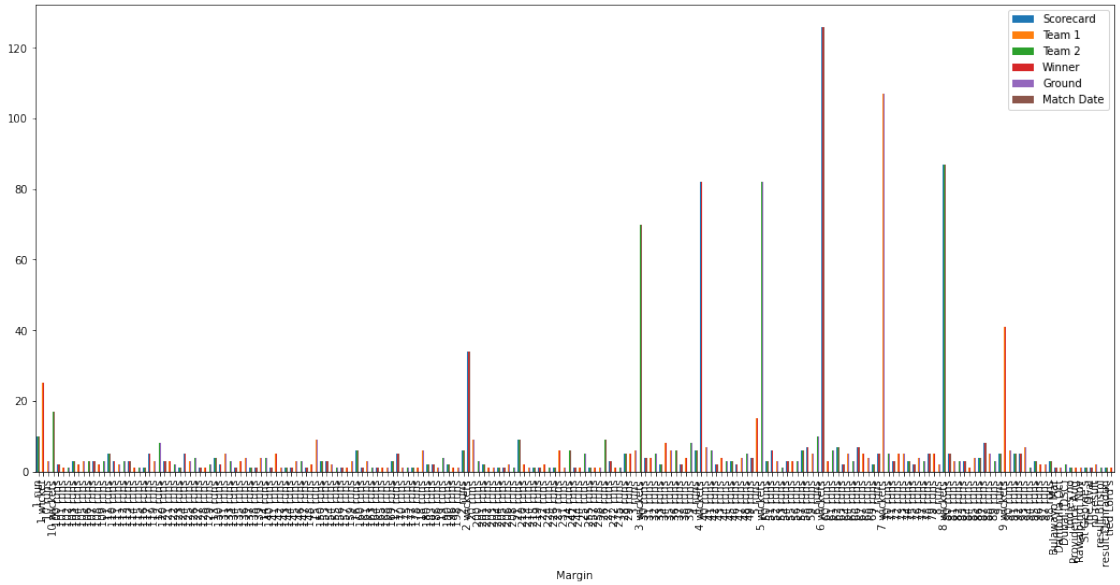
are very less 3. Also there are many matches with no result nad few which were tied.

[47]:
```
# group data by Ground and plot count plot
```



from above graph: 1. There are more samples of ground Dhaka, Harare, Abu dhabi, Bulawayo, Colombo (RPS), Sharjah

[48]:
```
# group data by Margin and plot count plot
```



from above graph: 1. Few Matches margin are very high then other matches, we need two sclae these numbers.

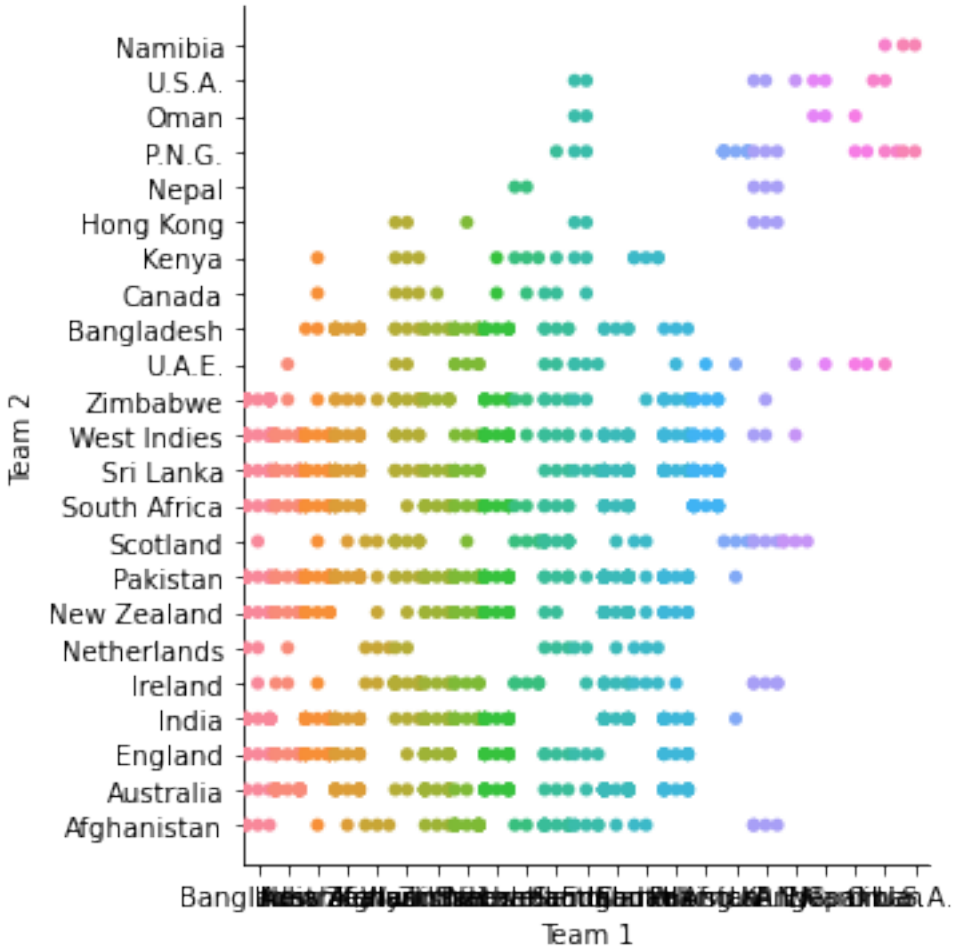
4.10 Catplot

Catplot shows frequencies of the of more than one categorical values at a time.

We will be doing categorical scatterplot using “swarm” kind

[49]:
```
# cat plot between Team 1 and Team 2 column using kind="swarm"
```

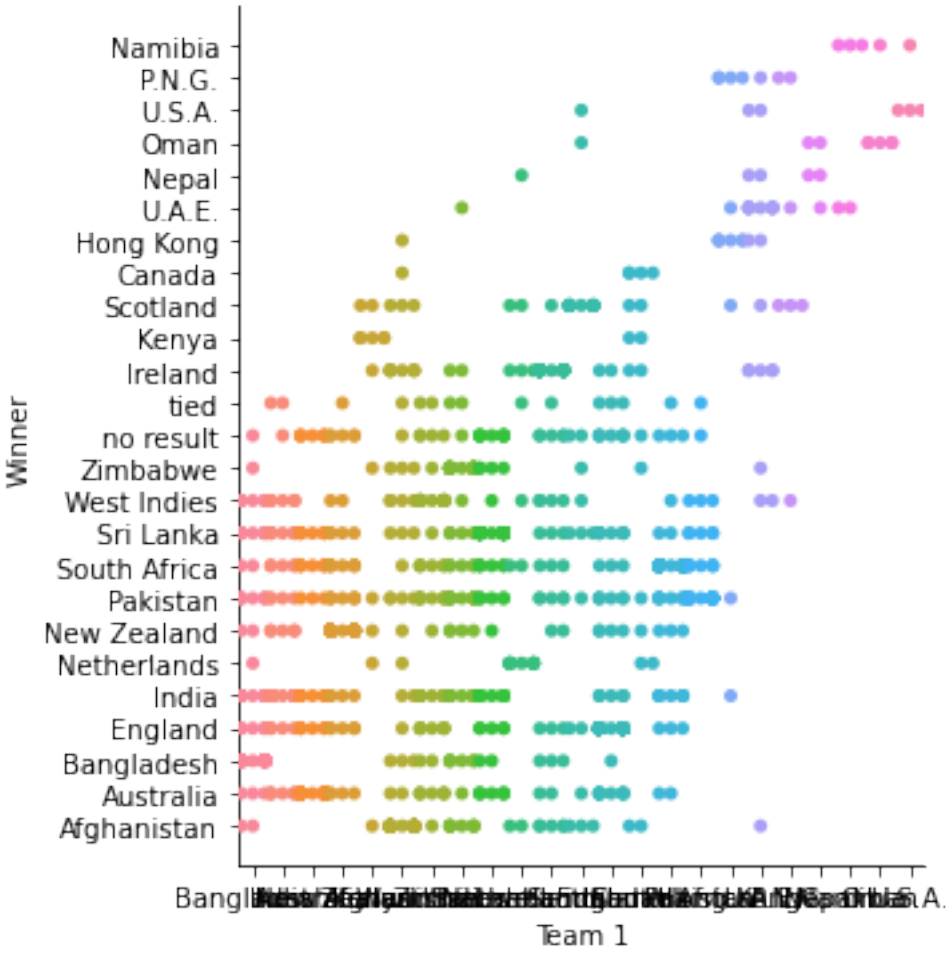
```
<Figure size 1296x576 with 0 Axes>
```



From the above graph: 1. Almost all team 1 had matches with almost all team 2 2. But there are few teams which have few matches with few team not with al teams.

[50]:
```
# cat plot between Team 1 and Winner column using kind="swarm"
```

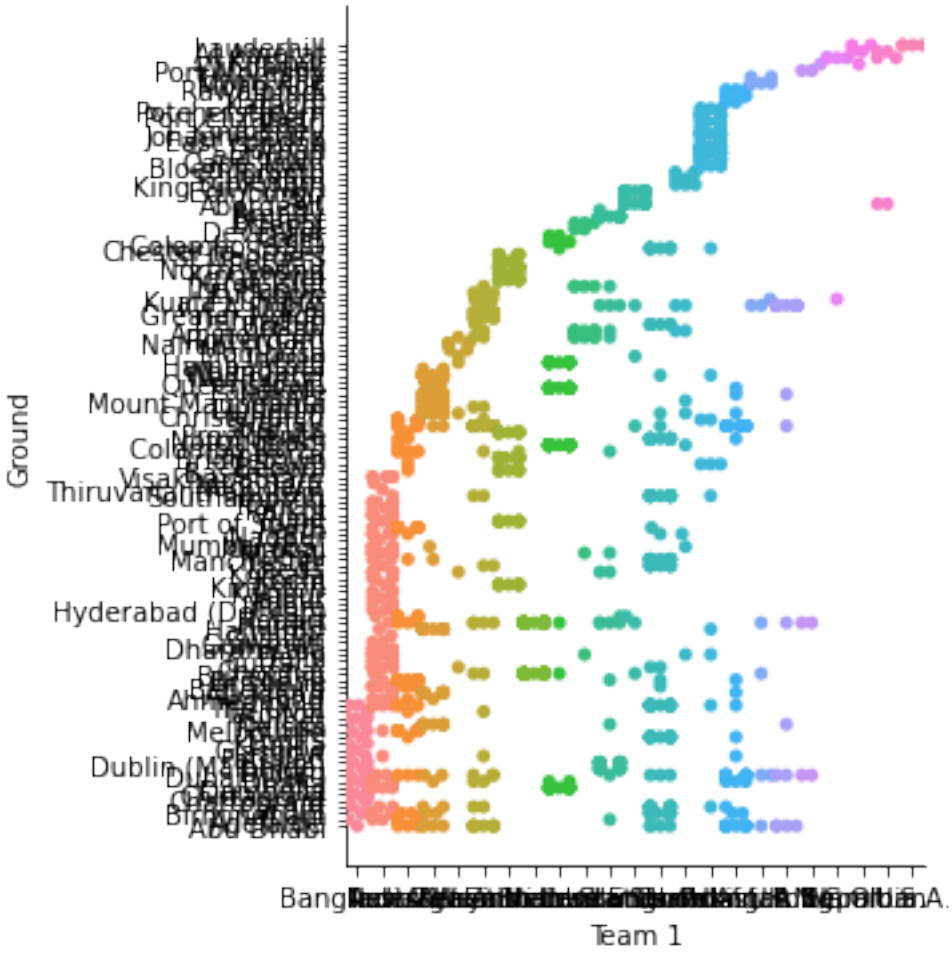
```
<Figure size 1296x576 with 0 Axes>
```



From the above graph: 1. Many teams in team 1 have distribution of winner team almost same range of countries. 2. But few team such as Namibia, P.N.G, U.S.A, Oman, Nepal , Hong Kong have very few distribution of winner team and the team range is different from other teams.

[51]:
```
# cat plot between Team 1 and Ground column using kind="swarm"
```

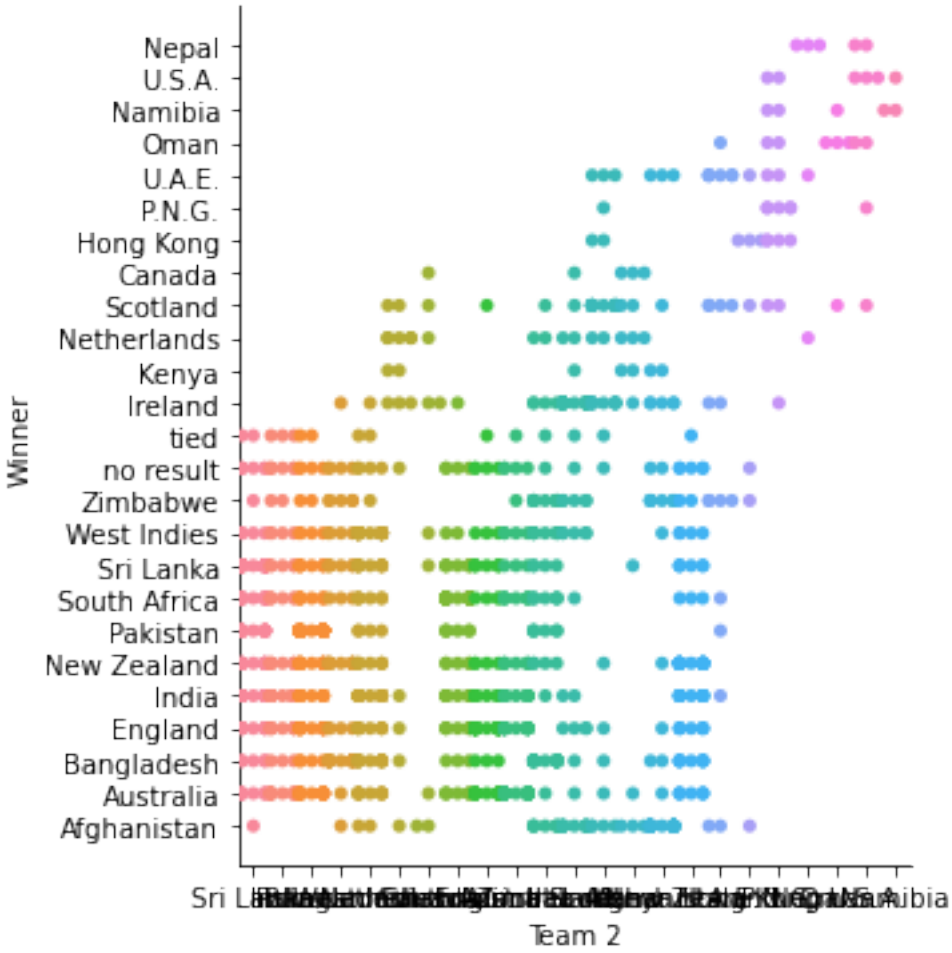
```
<Figure size 1296x576 with 0 Axes>
```



From the above graph: 1. Few teams are having matches repeatedly at few range of ground.

[52]:
```
# cat plot between Team 2 and Winner column using kind="swarm"
```

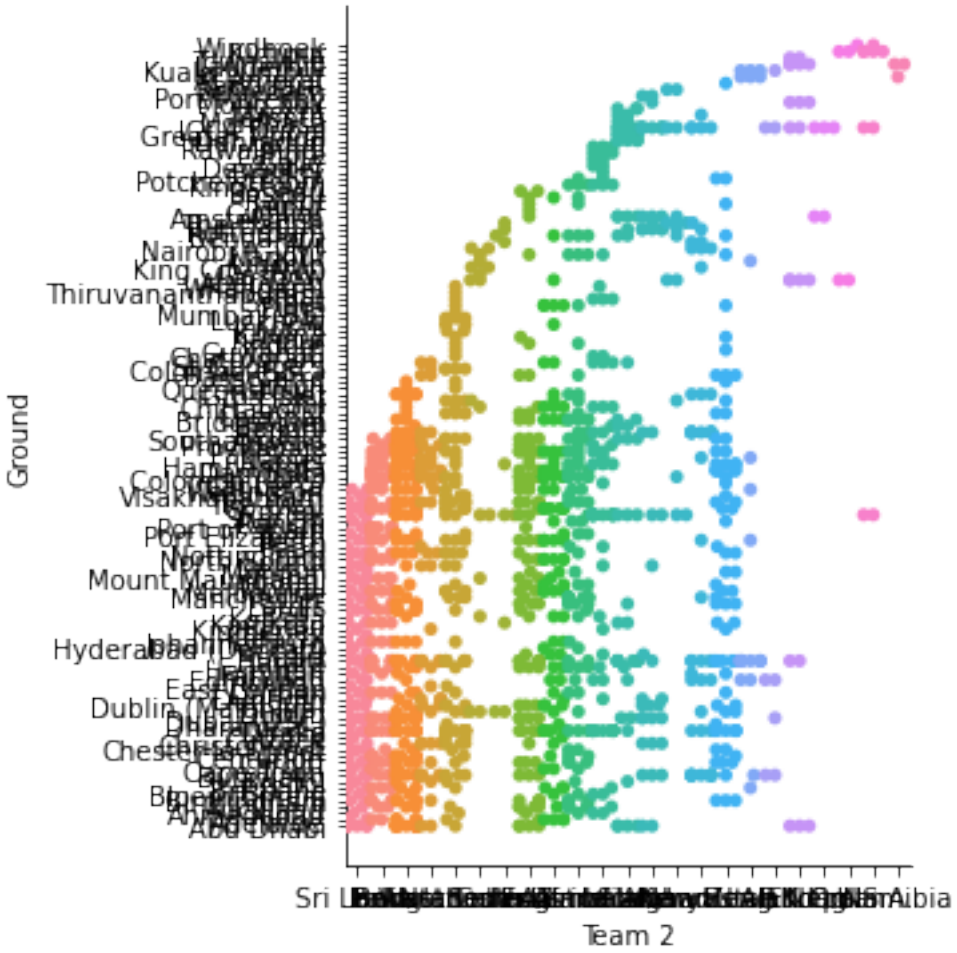
```
<Figure size 1296x576 with 0 Axes>
```



From the above graph: 1. Many teams in team 2 have distribution of winner team almost same range of countries. 2. But few team such as Namibia, P.N.G, U.S.A, Oman, Nepal , Hong Kong have very few distribution of winner team and the team range is different from other teams.

[53]:
```
# cat plot between Team 2 and Ground column using kind="swarm"
```

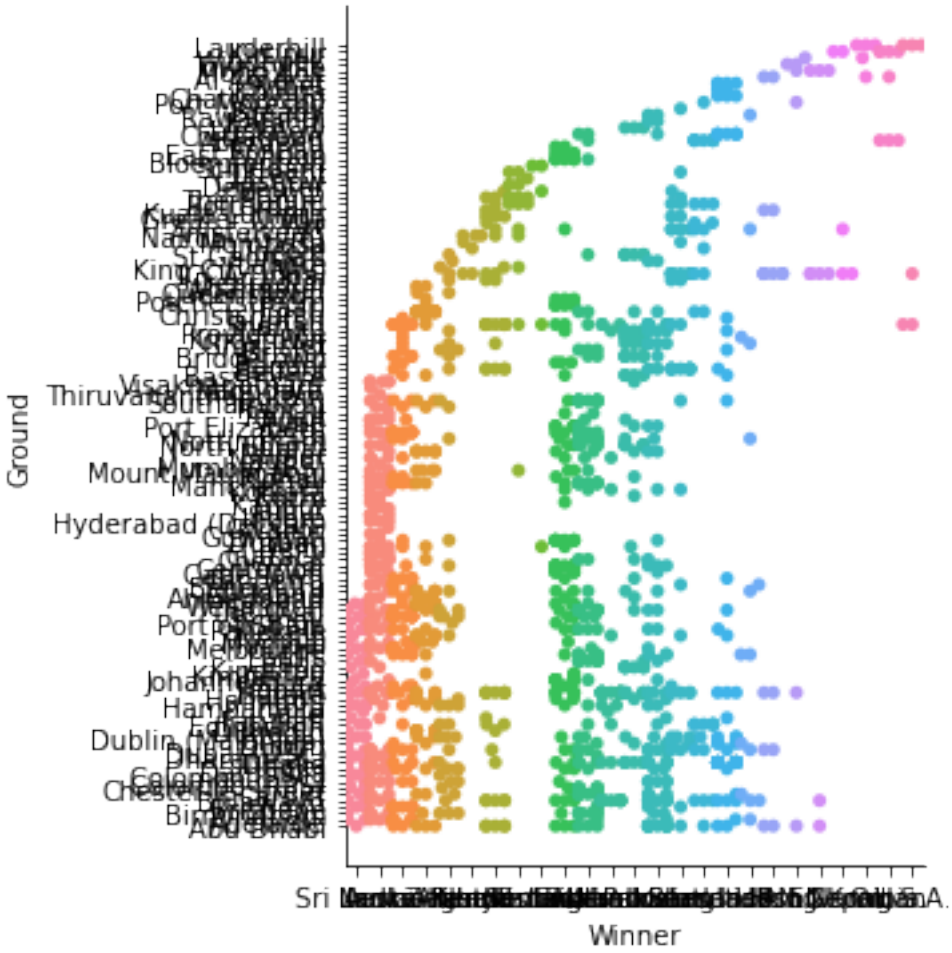
```
<Figure size 1296x576 with 0 Axes>
```



From the above graph: 1. Many teams are having matches repeatedly at a particular range of ground.

[54]:
```
# cat plot between Winner and Ground column using kind="swarm"
```

```
<Figure size 1296x576 with 0 Axes>
```

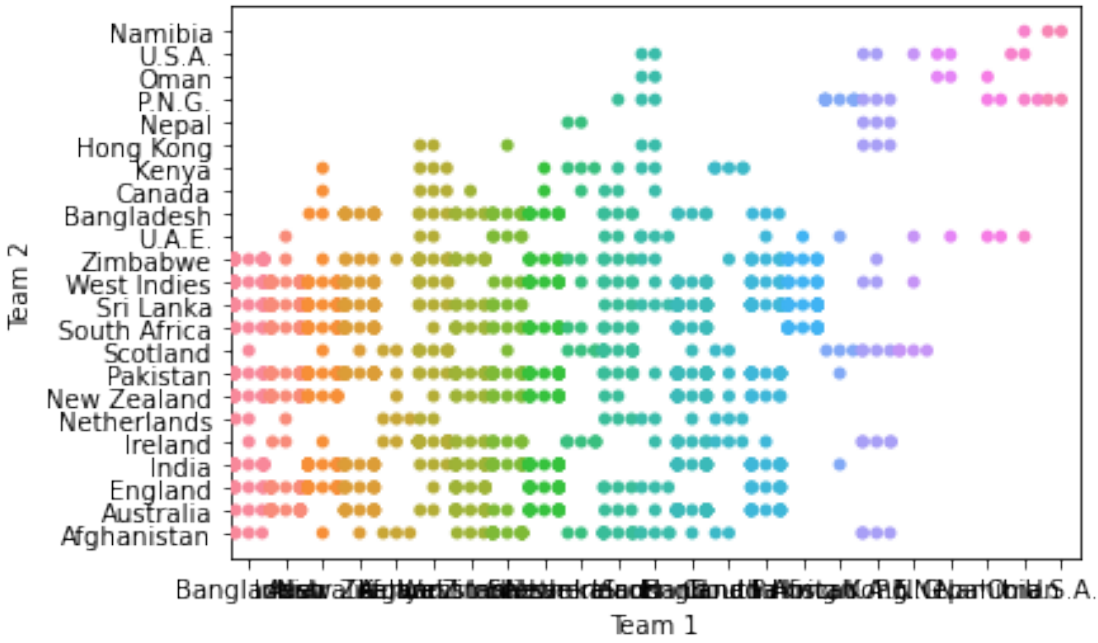


From the above graph: 1. Many winner teams are having matches repeatedly at a particular range of ground.

4.11 Swarm Plot

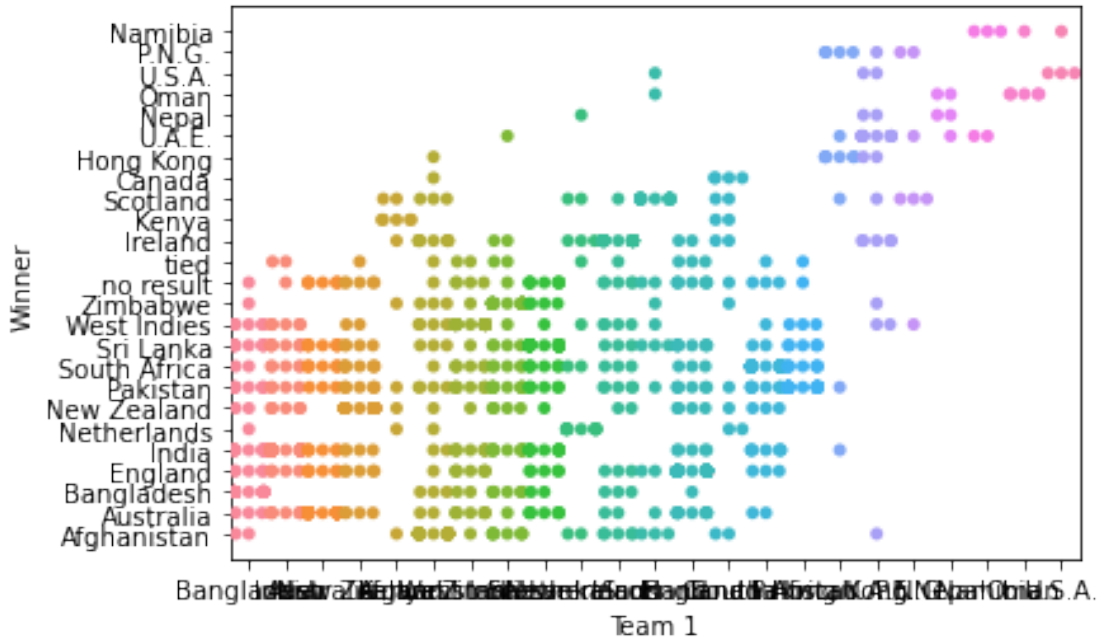
1. The swarm plot is a type of scatter plot, but helps in visualizing different categorical variables.
2. Scatter plots generally plots based on numeric values, but most of the data analyses happens on categorical variables. So, swarm plots seem very useful in those cases.

[55]:
```
# swarm plot between Team 1 and Team 2 column
```



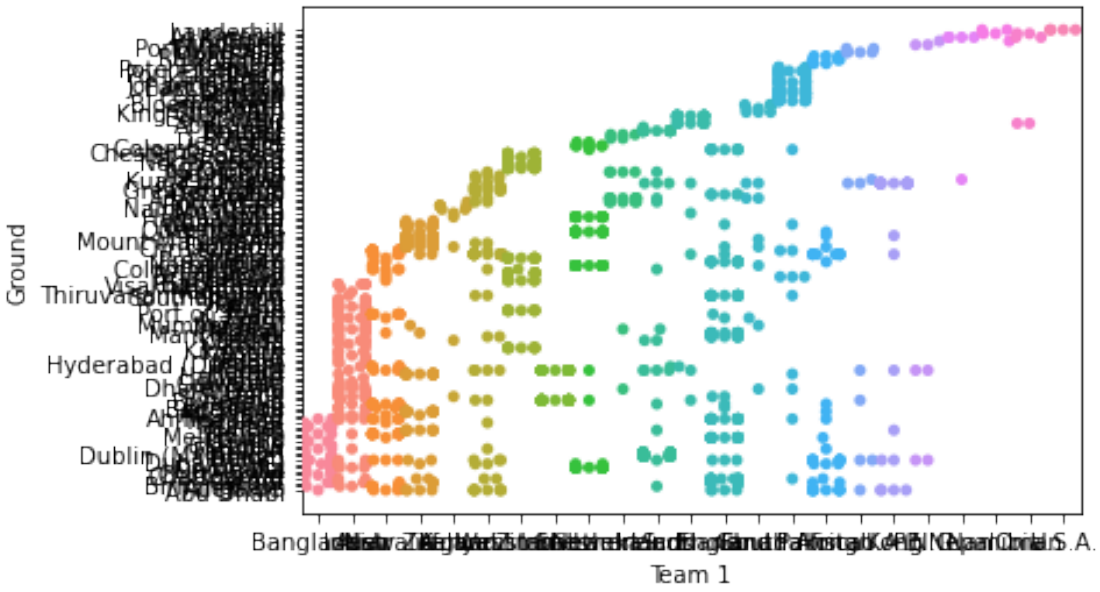
From the above graph: 1. Almost all team 1 had matches with almost all team 2 2. But there are few teams which have few matches with few team not with al teams.

[56]:
```
# swarm plot between Team 1 and Winner column
```



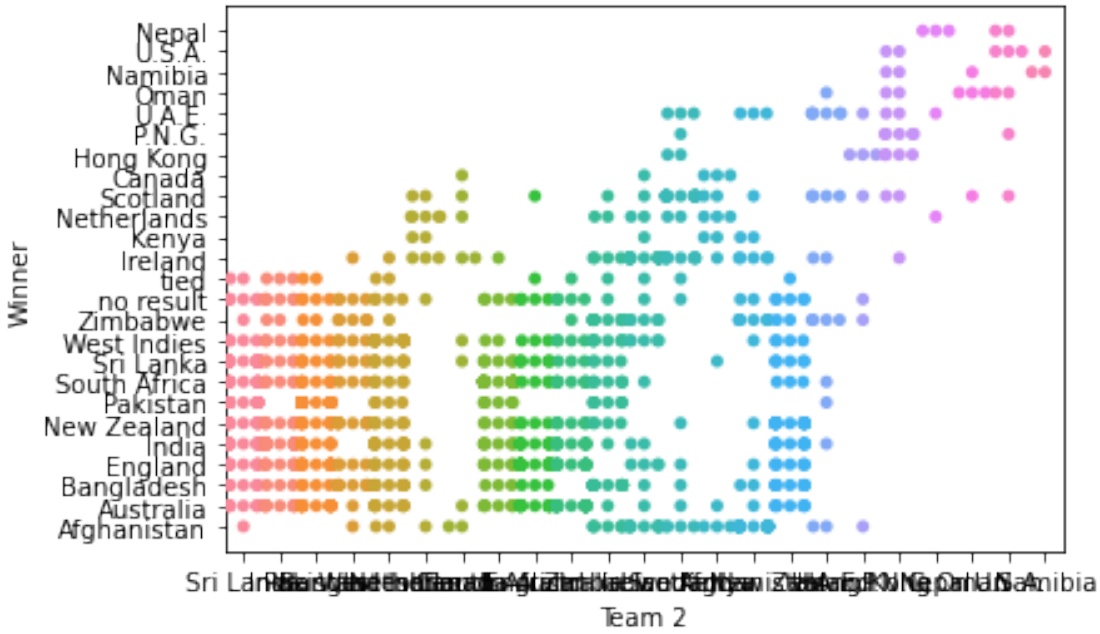
From the above graph: 1. Many teams in team 1 have distribution of winner team almost same range of countries. 2. But few team such as Namibia, P.N.G, U.S.A, Oman, Nepal , Hong Kong have very few distribution of winner team and the team range is different from other teams.

[57]:
```
# swarm plot between Team 1 and Ground column
```



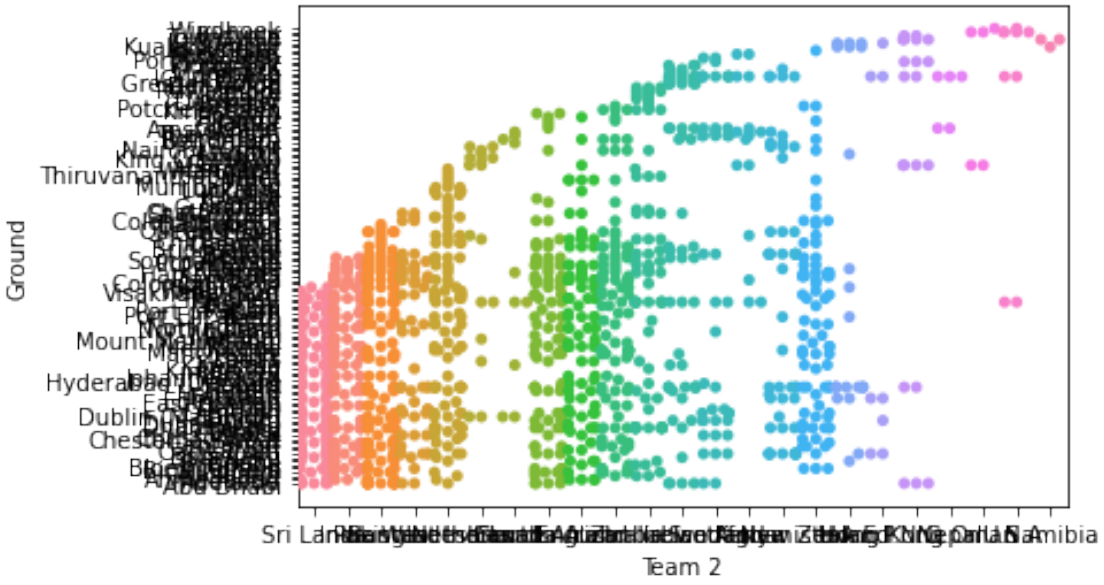
From the above graph: 1. Few teams are having matches repeatedly at few range of ground.

[58]:
```
# swarm plot between Team 2 and Winner column
```



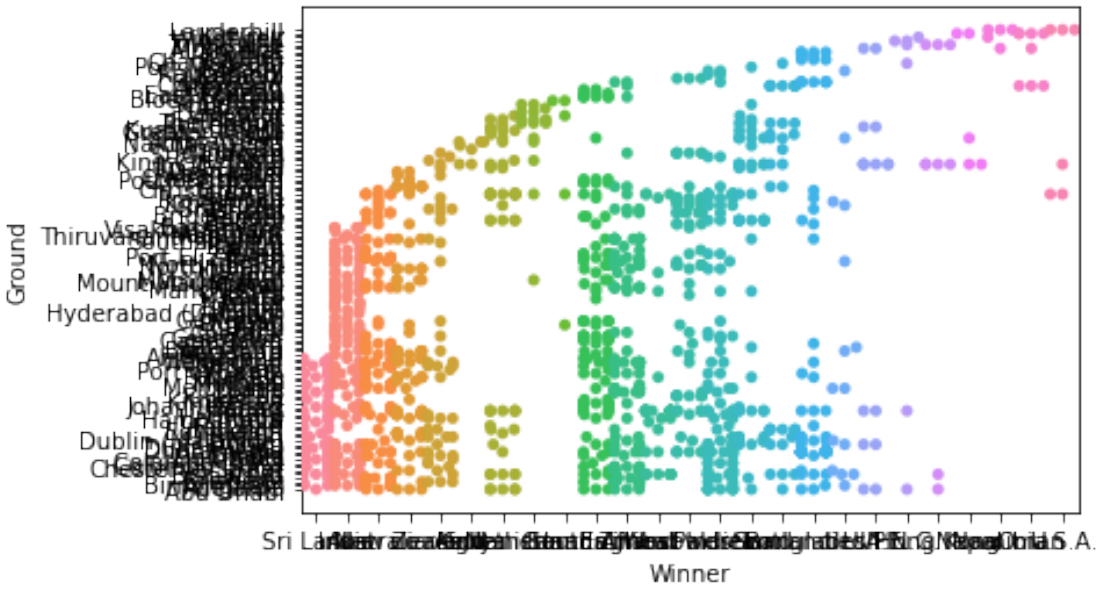
From the above graph: 1. Many teams in team 2 have distribution of winner team almost same range of countries. 2. But few team such as Namibia, P.N.G, U.S.A, Oman, Nepal , Hong Kong have very few distribution of winner team and the team range is different from other teams.

[59]:
```
# swarm plot between Team 2 and Ground column
```



From the above graph: 1. Many teams are having matches repeatedly at a particular range of ground.

[60]:
```
# swarm plot between Winner and Ground column
```



From the above graph: 1. Many winner teams are having matches repeatedly at a particular range of ground.

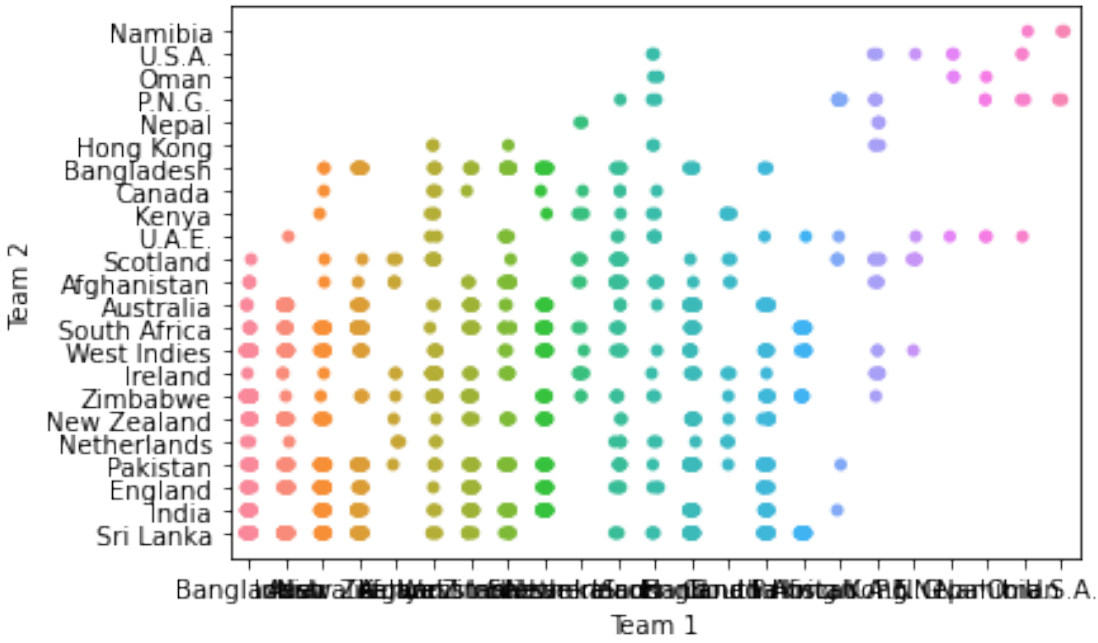
5 Strip Plot

A strip plot is a graphical data anlysis technique for summarizing a univariate data set. The strip plot consists of:

```
1. Horizontal axis = the value of the response variable;
2. Verticalal axis = all values are set to 1.
```

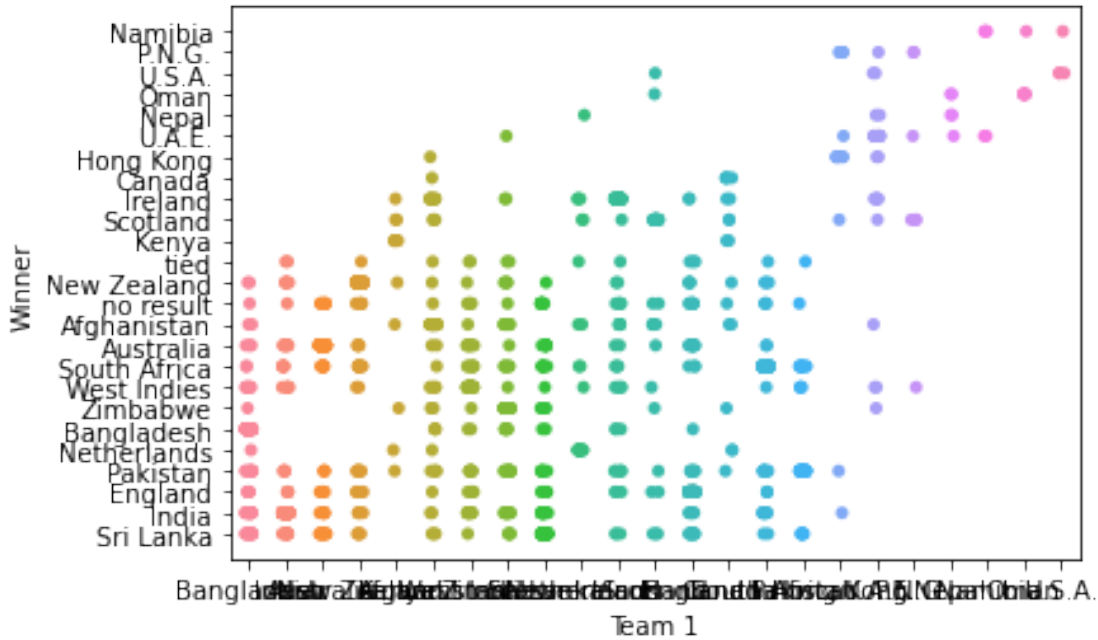
That is, a strip plot is simply a plot of the sorted response values along one axis. The strip plot is an alternative to a histogram or a density plot. It is typically used for small data sets (histograms and density plots are typically preferred for larger data sets).

[61]:
```
# strip plot between Team 1 and Team 2 column
```



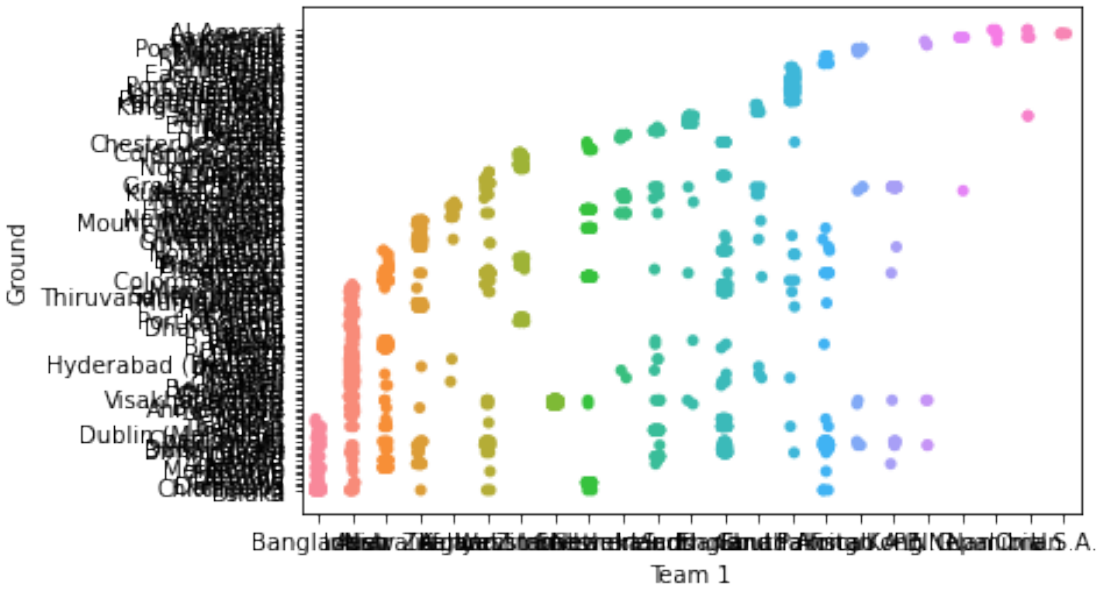
From the above graph: 1. Almost all team 1 had matches with almost all team 2 2. But there are few teams which have few matches with few team not with al teams.

[62]:
```
# strip plot between Team 1 and Winner column
```



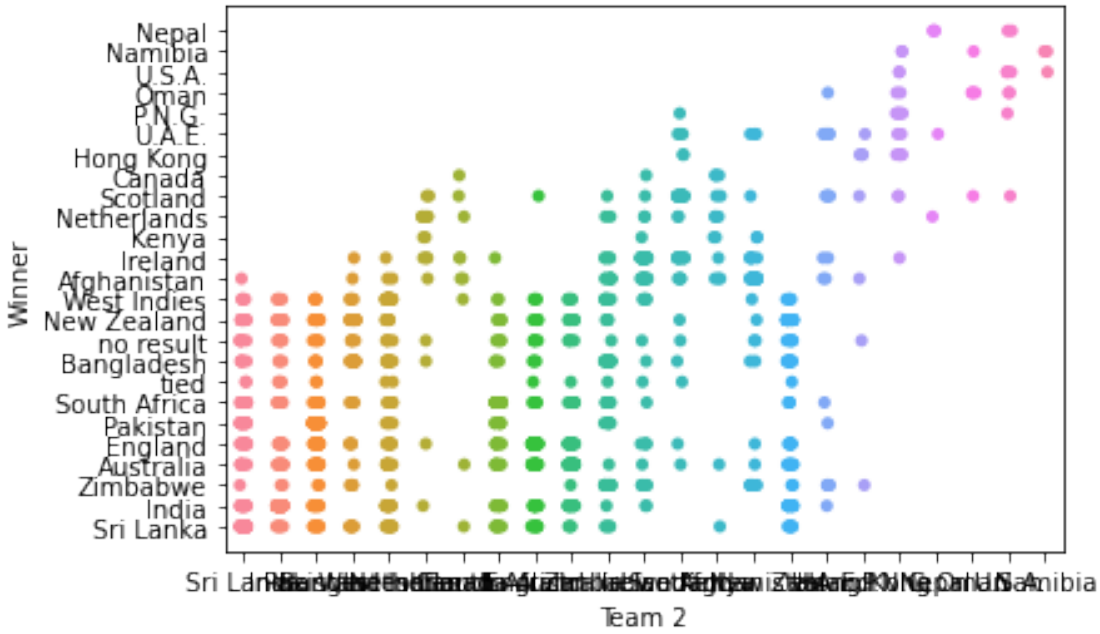
From the above graph: 1. Many teams in team 1 have distribution of winner team almost same range of countries. 2. But few team such as Namibia, P.N.G, U.S.A, Oman, Nepal , Hong Kong have very few distribution of winner team and the team range is different from other teams.

[63]:
```
# strip plot between Team 1 and Ground column
```



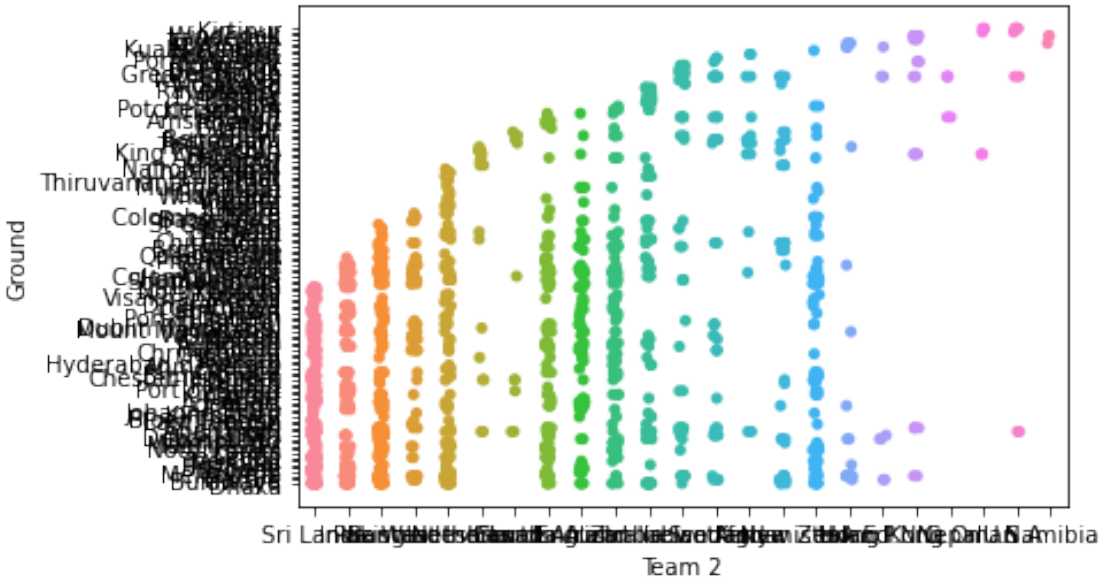
From the above graph: 1. Few teams are having matches repeatedly at few range of ground.

[64]:
```
# strip plot between Team 2 and Winner column
```



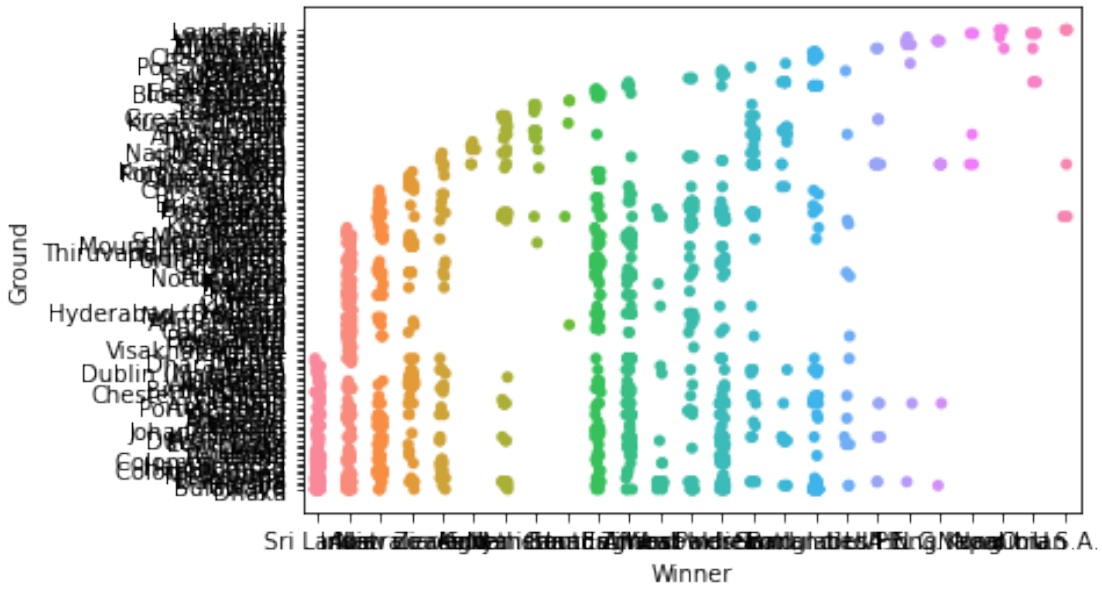
From the above graph: 1. Many teams in team 2 have distribution of winner team almost same range of countries. 2. But few team such as Namibia, P.N.G, U.S.A, Oman, Nepal , Hong Kong have very few distribution of winner team and the team range is different from other teams.

[65]:
```
# strip plot between Team 2 and Ground column
```



From the above graph: 1. Many teams are having matches repeatedly at a particular range of ground.

[66]:
```
# strip plot between Winner and Ground column
```

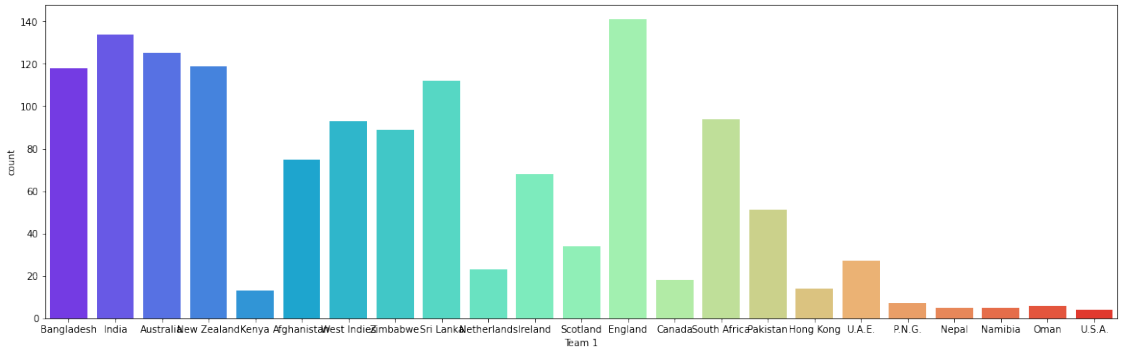


From the above graph: 1. Many Winner teams are having matches repeatedly at a particular range of ground.

5.1 Count Plot

1. A countplot is kind of like a histogram or a bar graph for some categorical area.
2. It simply shows the number of occurrences of an item based on a certain type of category.

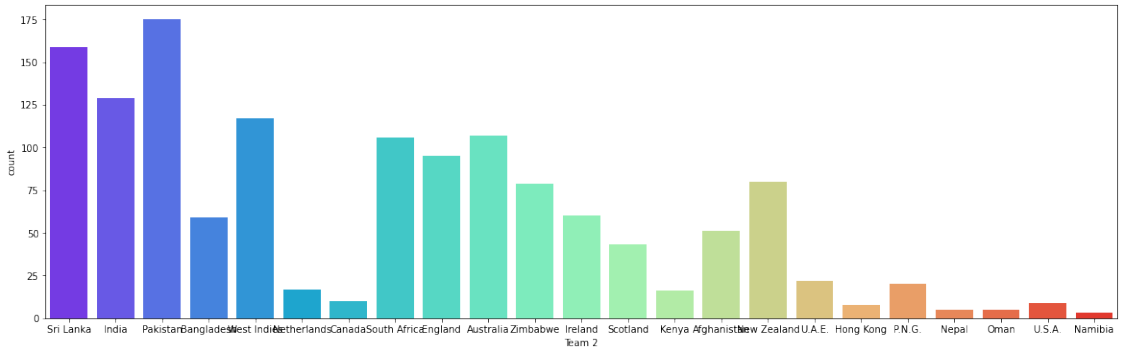
[67]:
```
# count plot for Team 1 column
```



From above count plot

distribution of values of Team 1 is not equal over complete dataset, skewed left multimodel.

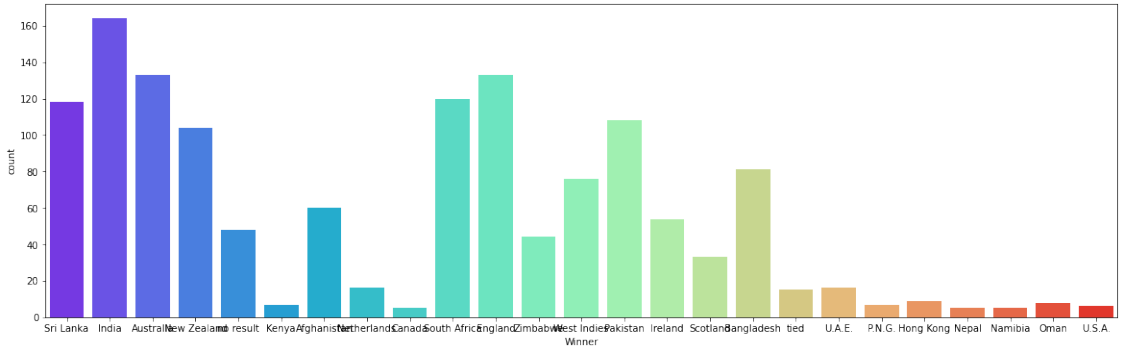
[68]:
```
# count plot for Team 2 column
```



From above count plot

distribution of values of team 2 is not equal over complete dataset, skewed left.

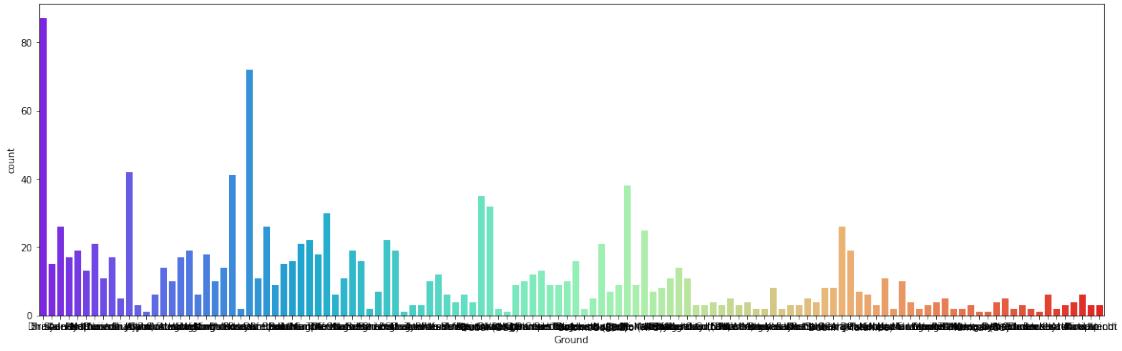
[69]:
```
# count plot for Winner column
```



From above count plot

distribution of values of winner teams is not equal over complete dataset, skewed left, multimodel.

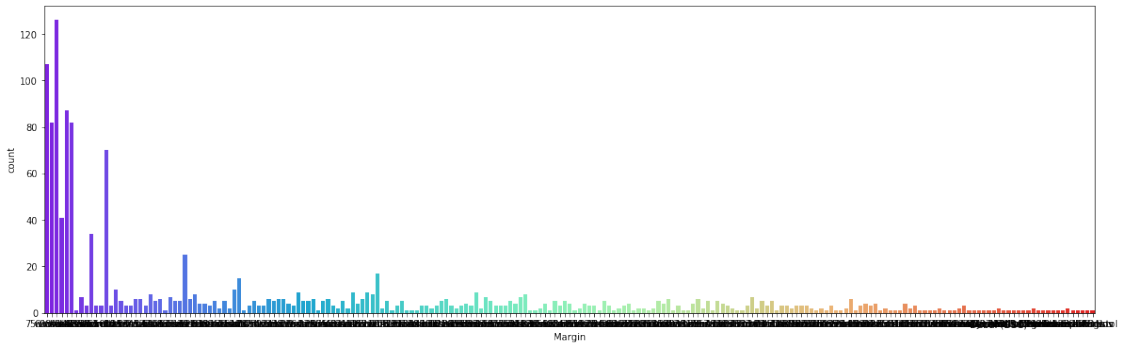
[70]:
```
# count plot for Ground column
```



From above count plot

distribution of values of ground is not equal over complete dataset, multimodel.

[71]:
```
# count plot for Margin column
```

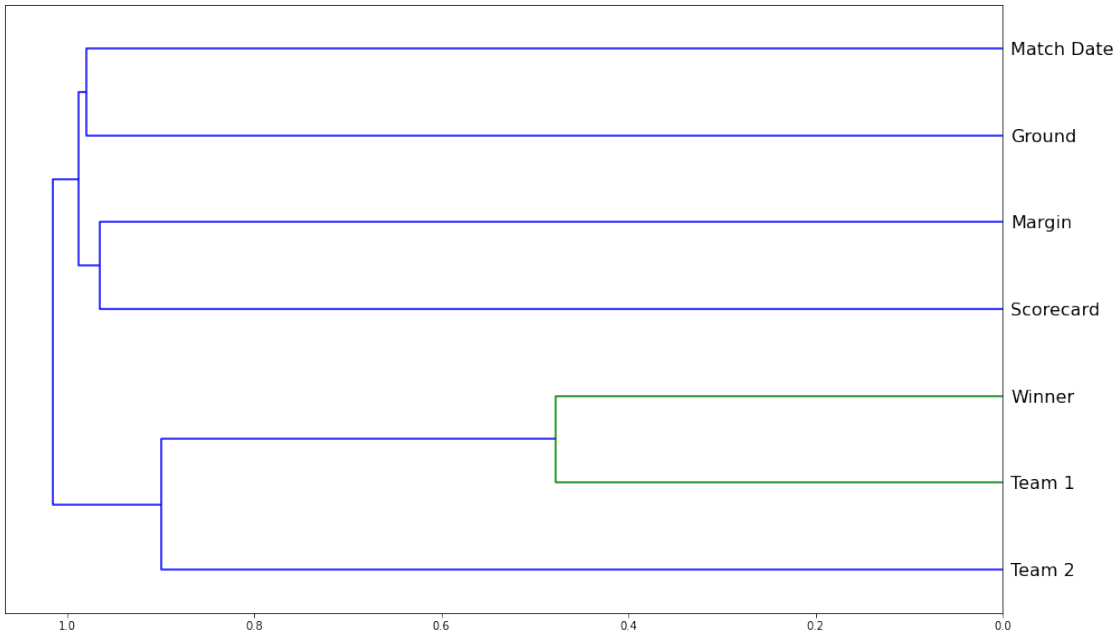


From above count plot

distribution of values of Margin is not equal over complete dataset, skewed left.

5.2 Dendrogram

[72]:
```
# Plot a Dendrogram on the columns of the dataset

# create a varibake to store ODI_data_2010_2021 after dropin nan values


# import scipy


# import hierarchy from skleanr.cluster


# create corr variable to store correlation results

# condense the corr variable


# pass the condensed corr variable to hierarchy linkage for getting average and␣
 ↪sore it in new varibale


# plot a dendogram with values of new varibale just created above.
```



observation from dendrogram

Strongly correlated variables: Team 1 and Winner

5.3 Since, there are missing values in Margin column of dataset

We need to drop those empty columns. Beacase all of the data is categorical and if we replace the missing values with random values it will affect the correctness of prediction.

[73]:
```
# Droping rows with missing values

# Reseting index of samples

# Droping extra "index" column from dataset
```

[74]:
```
# Filtering out the rows which contain winner column values as 'no result' and␣
↪'tied'

# Rest index of the datframe

# Drop the extra 'index' column from dataframe
```

5.4 Feature Engineering

It is one of the most important step in workflow of machine learning. Machine learning model work well if the data provided to the model is relevant and useful.

We can break down the Margin column which is Object data type into two columns named Won by runs and won by wickets both are numerical datatype int64

[75]:
```
# Create a list for storing runs

# Create a list for storing wickets

# looping through dataset column Margin

  # Splitting the string data by space and making a list


  # Exceptin handling

    # Search for index of "run" in string, after success the index of "run"␣
↪will be stored in index variable

    # Appending the the first element from splitted data after converting to␣
↪integer to won by run list created before looping

    # Appending the the 0 from splitted data to won by wicket list created␣
↪before looping


    # After fail --> print("-")


  # Exceptin handling

    # Search for index of "runs" in string, after success the index of "runs"␣
↪will be stored in index variable

    # Appending the the first element from splitted data after converting to␣
↪integer to won by run list created before looping

    # Appending the the 0 from splitted data to won by wicket list created␣
↪before looping


    # After fail --> print("-")


  # Exceptin handling

    # Search for index of "wicket" in string, after success the index of␣
↪"wicket" will be stored in index variable
```

```
    # Appending the the first element from splitted data after converting to␣
↪integer to won by wicket list created before looping

    # Appending the the 0 from splitted data to won by run list created before␣
↪looping

    # After fail --> print("-"):


  # Exceptin handling

    # Search for index of "wickets" in string, after success the index of␣
↪"wickets" will be stored in index variable

    # Appending the the first element from splitted data after converting to␣
↪integer to won by wicket list created before looping

    # Appending the the 0 from splitted data to won by run list created before␣
↪looping

    # After fail --> print("-")


# Add a new column named 'Won_By_Runs' to Datframe with the respectiev data list

# Add a new column named 'Won_By_Wickets' to Datframe with the respectiev data␣
↪list
```

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–

–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–

–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–

–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–

–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–

```
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
```

[75]:
```
        Scorecard       Team 1  … Won_By_Runs Won_By_Wickets
0     ODI # 2937   Bangladesh  …           0              7
1     ODI # 2938        India  …           0              5
2     ODI # 2939   Bangladesh  …           0              6
3     ODI # 2940   Bangladesh  …           0              9
4     ODI # 2941        India  …           0              8
…            …           …  …         …            …
1307  ODI # 4309    Sri Lanka  …           0              3
1308  ODI # 4310  West Indies  …         133              0
1309  ODI # 4311  West Indies  …           0              4
1310  ODI # 4312    Sri Lanka  …           0              3
1311  ODI # 4313  West Indies  …           0              6

[1312 rows x 9 columns]
```

6 Further feature engineering

We can create a two new columns named team_1_first_batting and team_2_first_batting from Won_by_runs and Won_by_wickets

Explanation: If a team wins by run it means that the team got first chance for batting else if a team wins by wickets that means the team got first chance for bowling.

[76]:
```
# Create list to store team 1 first batting

# Create list to store team 2 first batting

# looping through Team 1, Team 2, Won_by_wickets, Won_by_runs, Winner columns␣
↪simultaneously using zip method in python

  # Check condition if runs are greater than 0 and team 1 is the winner

    # Append 1 to team_1_first_batting list

    # Append 0 to team_2_first_batting list

  # Check condition if runs are greater than 0 and team 2 is the winner

    # Append 0 to team_1_first_batting list

    # Append 1 to team_2_first_batting list

  # Check condition if wickets are greater than 0 and team 1 is the winner

    # Append 0 to team_1_first_batting list

    # Append 1 to team_2_first_batting list

  # Check condition if wickets are greater than 0 and team 2 is the winner

    # Append 1 to team_1_first_batting list

    # Append 0 to team_2_first_batting list



# Add a column named 'Team_1_First_Batting' to Dataframe with respective data␣
↪list

# Add a column named 'Team_2_First_Batting' to Dataframe with respective data␣
↪list
```

[77]:
```
# ODI_data_2010_2021 data
```

[77]:
```
      Scorecard       Team 1  … Team_1_First_Batting Team_2_First_Batting
0     ODI # 2937   Bangladesh  …                    1                    0
1     ODI # 2938        India  …                    1                    0
2     ODI # 2939   Bangladesh  …                    1                    0
3     ODI # 2940   Bangladesh  …                    1                    0
4     ODI # 2941        India  …                    0                    1
…          …          …  …                    …                    …
1307  ODI # 4309    Sri Lanka  …                    1                    0
1308  ODI # 4310  West Indies  …                    0                    1
1309  ODI # 4311  West Indies  …                    0                    1
1310  ODI # 4312    Sri Lanka  …                    0                    1
1311  ODI # 4313  West Indies  …                    1                    0

[1312 rows x 11 columns]
```

6.0.1 Dealing with Multi Class Problem

Instead of using winner team name as target values we can use, 1 to represent team 1 as winner and 2 to represent team 2 as winner. It will reduce multiclass classification problem as using team name as label we the model have to decide correct label from more than 10 labels. Hence, using only two labels will improve the performance of the model.

[78]:
```
# Create a list for storing winner team label 1 or 2

# looping through dataset columns Team 1, Team 2 and Winner simultaneously␣
 ↪using zip method in python

  # Checking condition if team 1 is winner

    # Appending 1 to winner list

  # Checking condition if team 2 is winner

    # Appending 2 to winner list


# Add a column named 'Winning_team' to Dataframe with respective data list
```

[79]:
```
# ODI_data_2010_2021 data
```

[79]:
```
      Scorecard       Team 1  … Team_2_First_Batting Winning_team
0     ODI # 2937   Bangladesh  …                    0            2
1     ODI # 2938        India  …                    0            2
2     ODI # 2939   Bangladesh  …                    0            2
3     ODI # 2940   Bangladesh  …                    0            2
4     ODI # 2941        India  …                    1            1
…          …          …  …                    …          …
```

```
1307  ODI # 4309    Sri Lanka  …                   0            2
1308  ODI # 4310  West Indies  …                   1            2
1309  ODI # 4311  West Indies  …                   1            1
1310  ODI # 4312    Sri Lanka  …                   1            1
1311  ODI # 4313  West Indies  …                   0            2

[1312 rows x 12 columns]
```

6.1 Scaling

Scaling is very crucial part of teh workflow. As the data we have for example: Won_by_run has some values greater than 100 and some values around zero the model will be dominated by this high values causing the model to under perform. Thus, we need to scale this data between particular numerical range. We use MinMaxScaler in this problem.

[80]:
```
# Helper function for scaling all the numerical data using MinMaxScalar
def scale_data(df,col):

    # Import MinMaxScaler

    # Instantiate MinMaxScaler

    # fit transform the data


    # return scaled dataframe
```

[81]:
```
# Making a list of the column names to be scaled


# passing data and name for scaling
```

[82]:
```
# Dumify the dataset columns Team 1 and Team 2 and store it in new variable
```

[83]:
```
# dumified_data
```

[83]:
```
       Scorecard       Winner  … Team 2_West Indies Team 2_Zimbabwe
0     ODI # 2937    Sri Lanka  …                   0               0
1     ODI # 2938    Sri Lanka  …                   0               0
2     ODI # 2939        India  …                   0               0
3     ODI # 2940    Sri Lanka  …                   0               0
4     ODI # 2941        India  …                   0               0
…          …          …  …                 …             …
1307  ODI # 4309        India  …                   0               0
1308  ODI # 4310    Australia  …                   0               0
1309  ODI # 4311  West Indies  …                   0               0
```

```
1310  ODI # 4312    Sri Lanka  …                 0               0
1311  ODI # 4313    Australia  …                 0               0

[1312 rows x 56 columns]
```

[84]:
```
# Seperate feature and target variables
# create variable to store datframe without␣
 ↪'Scorecard','Margin','Winner','Ground','Winning_team','Match Date' data

# Create varaiable to store 'Winning_team' data
```

[85]:
```
# Create test_data_x and test_data_y variable with samples all 2021 data
```

[86]:
```
# Drop range of 2021 data from feature variable
```

[87]:
```
# # Drop range of 2021 data from feature variable
```

[88]:
```
#importing Sklearn library for spliting train dataset into train and test␣
 ↪dataset


# split the data into train set of size 80% and valid set of size 20% with␣
 ↪random_state = 123
```

6.2 Modeling

[89]:
```
# importing necessary libraries for calculating metrics of model


# Function for calculating all the relevant metrics with parameter as model␣
 ↪instance

  # Calculate the classification report of model passed to the function
```

[90]:
```
# Visualize importance of all the features in the dataset for the prediction
# Helper function for Visualizing importance of all the features in the dataset␣
 ↪for the prediction


    # creating dataframe for feature name and feature importance



    # grouping all data and sorting in descending order
```

```
    # ploting feature importance data using boxenplot


    # return fig,ax
```

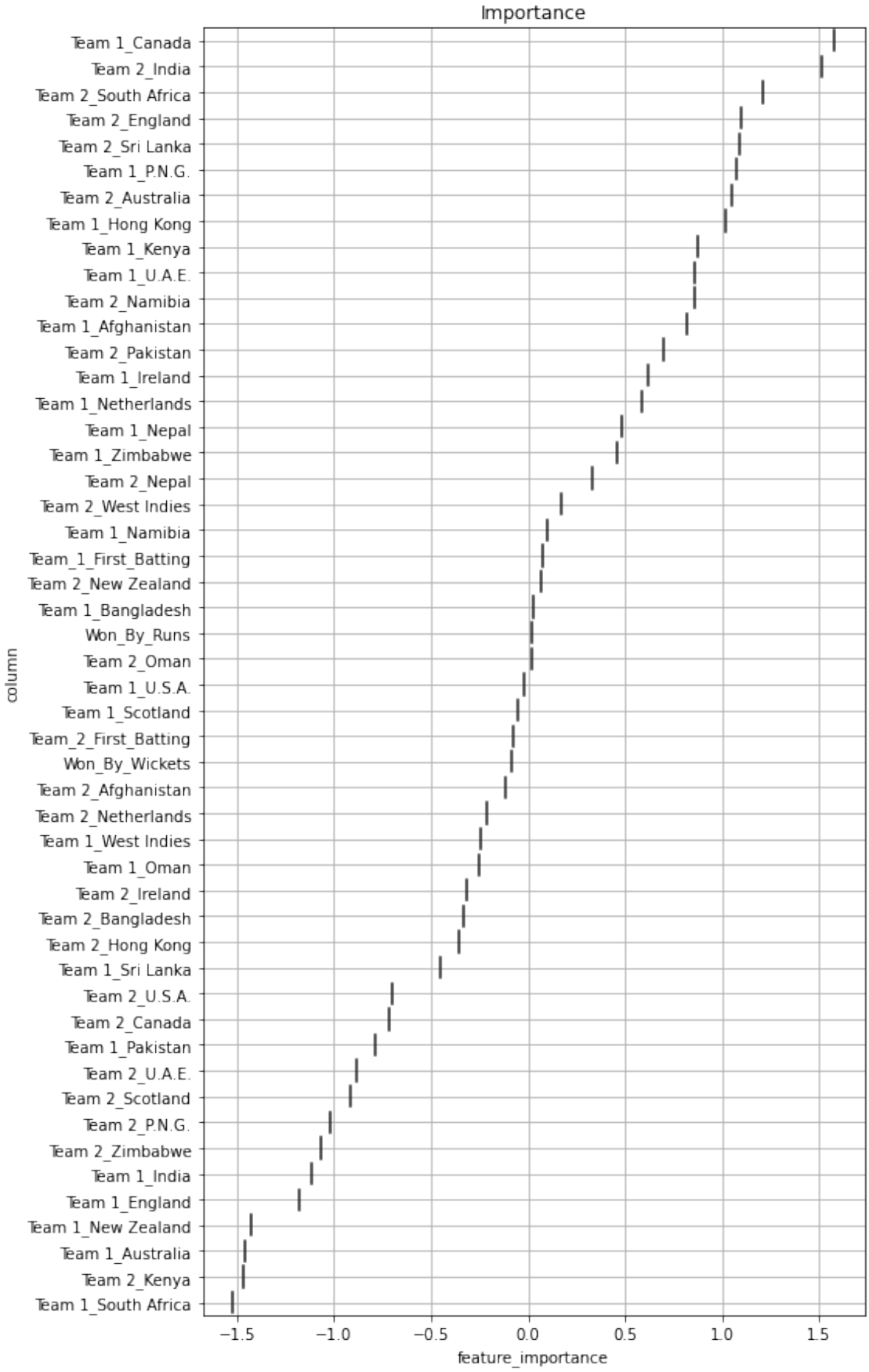
6.2.1 LogisticRegression

1. Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique.
2. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

[91]:
```
%%time
# Fit a LogisticRegression model to the train dataset

# Import LogisticRegression


# Instantiate the model


# Fit the model to the data

# print score on train and valid set


# print classification report of the model using function created before


# visualizing importance of features
```

```
Training set accuracy:  0.708
Test set accuracy:  0.677
Classification Report
              precision    recall  f1-score   support

           1       0.68      0.85      0.76       151
           2       0.66      0.43      0.52       103

    accuracy                           0.68       254
   macro avg       0.67      0.64      0.64       254
weighted avg       0.67      0.68      0.66       254

CPU times: user 854 ms, sys: 236 ms, total: 1.09 s
```

```
Wall time: 874 ms
```


Importance
Team 1_Canada
Team 2_India
Team 2_South Africa
Team 2_England
Team 2_Sri Lanka
Team 1_P.N.G.
Team 2_Australia
Team 1_Hong Kong
Team 1_Kenya
Team 1_U.A.E.
Team 2_Namibia
Team 1_Afghanistan
Team 2_Pakistan
Team 1_Ireland
Team 1_Netherlands
Team 1_Nepal
Team 1_Zimbabwe
Team 2_Nepal
Team 2_West Indies
Team 1_Namibia
Team_1_First_Batting
Team 2_New Zealand
Team 1_Bangladesh
Won_By_Runs
Team 2_Oman
Team 1_U.S.A.
Team 1_Scotland
Team_2_First_Batting
Won_By_Wickets
Team 2_Afghanistan
Team 2_Netherlands
Team 1_West Indies
Team 1_Oman
Team 2_Ireland
Team 2_Bangladesh
Team 2_Hong Kong
Team 1_Sri Lanka
Team 2_U.S.A.
Team 2_Canada
Team 1_Pakistan
Team 2_U.A.E.
Team 2_Scotland
Team 2_P.N.G.
Team 2_Zimbabwe
Team 1_India
Team 1_England
Team 1_New Zealand
Team 1_Australia
Team 2_Kenya
Team 1_South Africa
column
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
feature_importance

6.2.2 RandomForestClassifier

Random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

[92]:
```
%%time
# Fit a Random Forest Classifier model to the train dataset

# Import RandomForestClassifier


# Instantiate the model

# Fit the model to the data


# print score on train and valid set


# print classification report of the model using function created before


# visualizing the inportance of features.
```

```
Training set accuracy:  1.000
Test set accuracy:  0.988
Classification Report
              precision    recall  f1-score   support

           1       0.98      1.00      0.99       151
           2       1.00      0.97      0.99       103

    accuracy                           0.99       254
   macro avg       0.99      0.99      0.99       254
weighted avg       0.99      0.99      0.99       254

CPU times: user 788 ms, sys: 60.7 ms, total: 848 ms
Wall time: 847 ms
```

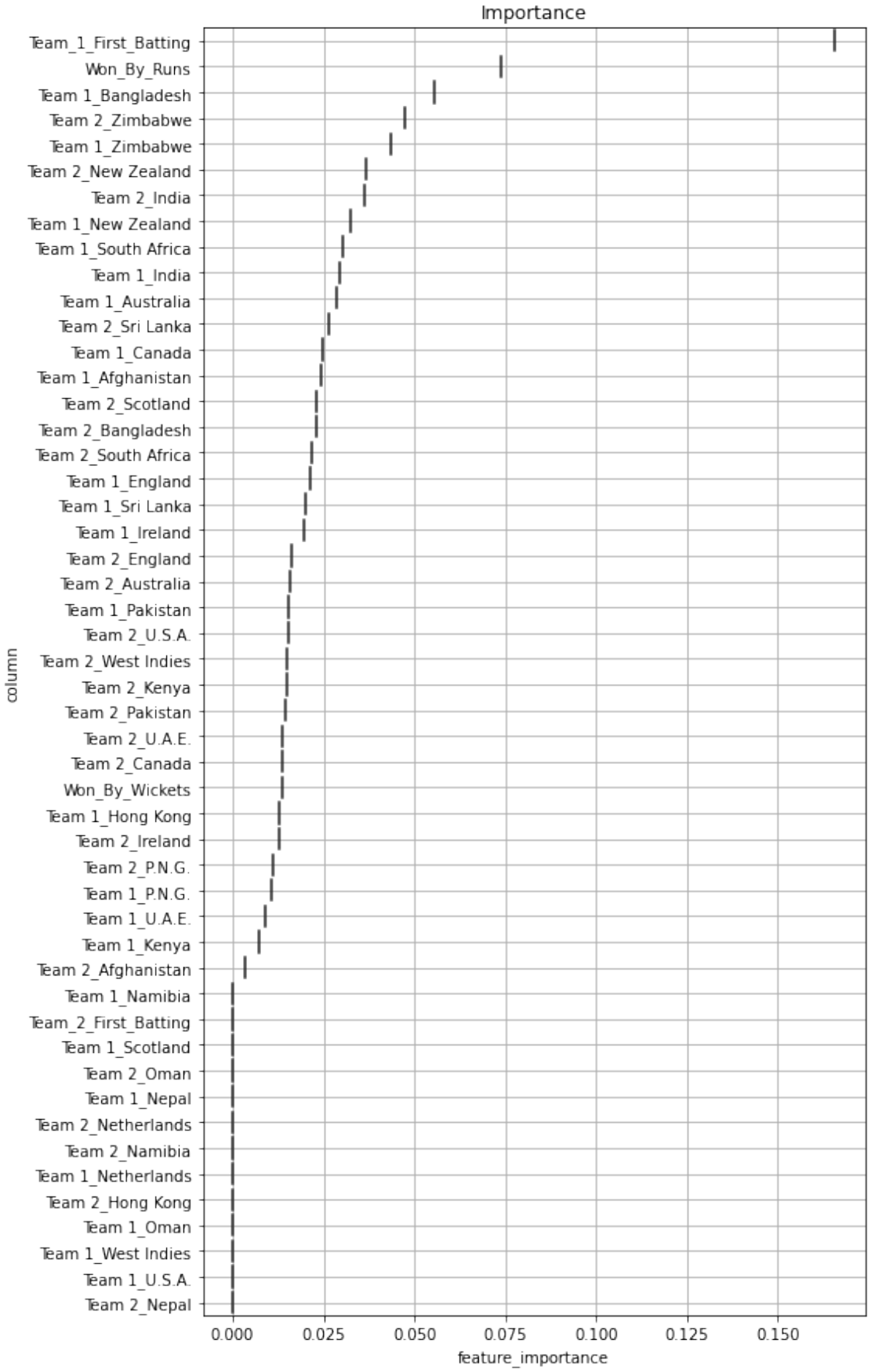

Importance
Won_By_Runs
Won_By_Wickets
Team_2_First_Batting
Team_1_First_Batting
Team 2_India
Team 2_Zimbabwe
Team 1_Australia
Team 1_New Zealand
Team 1_South Africa
Team 2_South Africa
Team 2_Sri Lanka
Team 1_Zimbabwe
Team 1_England
Team 1_Bangladesh
Team 2_Pakistan
Team 1_Sri Lanka
Team 1_Afghanistan
Team 2_England
Team 2_West Indies
Team 1_India
Team 1_Ireland
Team 2_Australia
Team 1_Canada
Team 2_New Zealand
Team 2_Bangladesh
Team 2_Scotland
Team 2_Afghanistan
Team 1_West Indies
Team 1_Scotland
Team 2_Ireland
Team 2_U.A.E.
Team 2_Kenya
Team 1_Pakistan
Team 1_U.A.E.
Team 2_P.N.G.
Team 1_Hong Kong
Team 1_Kenya
Team 1_Netherlands
Team 2_Netherlands
Team 1_P.N.G.
Team 2_Nepal
Team 1_Nepal
Team 2_Namibia
Team 2_U.S.A.
Team 2_Canada
Team 2_Hong Kong
Team 1_U.S.A.
Team 1_Oman
Team 1_Namibia
Team 2_Oman
column
0.00
0.05
0.10
0.15
0.20
0.25
feature_importance

6.2.3 XGBClassifier

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

[93]:
```
%%time
# Fit a XGBClassifier model to the train dataset

# Import XGBClassifier


# Instantiate the model


# fitting the model on train data


# print score on train and valid set


# print classification report of the model using function created before

# visualizing the inportance of features.
```

```
Training set accuracy:  0.916
Test set accuracy:  0.839
Classification Report
               precision    recall  f1-score   support

           1       0.80      0.97      0.88       151
           2       0.93      0.65      0.77       103

    accuracy                           0.84       254
   macro avg       0.87      0.81      0.82       254
weighted avg       0.85      0.84      0.83       254

CPU times: user 368 ms, sys: 26.3 ms, total: 394 ms
Wall time: 660 ms
```


Importance
Team_1_First_Batting
Won_By_Runs
Team 1_Bangladesh
Team 2_Zimbabwe
Team 1_Zimbabwe
Team 2_New Zealand
Team 2_India
Team 1_New Zealand
Team 1_South Africa
Team 1_India
Team 1_Australia
Team 2_Sri Lanka
Team 1_Canada
Team 1_Afghanistan
Team 2_Scotland
Team 2_Bangladesh
Team 2_South Africa
Team 1_England
Team 1_Sri Lanka
Team 1_Ireland
Team 2_England
Team 2_Australia
Team 1_Pakistan
Team 2_U.S.A.
Team 2_West Indies
Team 2_Kenya
Team 2_Pakistan
Team 2_U.A.E.
Team 2_Canada
Won_By_Wickets
Team 1_Hong Kong
Team 2_Ireland
Team 2_P.N.G.
Team 1_P.N.G.
Team 1_U.A.E.
Team 1_Kenya
Team 2_Afghanistan
Team 1_Namibia
Team_2_First_Batting
Team 1_Scotland
Team 2_Oman
Team 1_Nepal
Team 2_Netherlands
Team 2_Namibia
Team 1_Netherlands
Team 2_Hong Kong
Team 1_Oman
Team 1_West Indies
Team 1_U.S.A.
Team 2_Nepal
column
0.000
0.025
0.050
0.075
0.100
0.125
0.150
feature_importance

6.2.4 Support Vector Classifier

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

[94]:
```
%%time
# Fit a SVC model to the train dataset

# Import SVC


# Instantiate the model


# Fit the model to the data


# print score on train and valid set


# print classification report of the model using function created before
```

```
Training set accuracy:  0.976
Test set accuracy:  0.945
Classification Report
              precision    recall  f1-score   support

           1       0.93      0.98      0.95       151
           2       0.97      0.89      0.93       103

    accuracy                           0.94       254
   macro avg       0.95      0.94      0.94       254
weighted avg       0.95      0.94      0.94       254

CPU times: user 135 ms, sys: 1.89 ms, total: 137 ms
Wall time: 140 ms
```

6.2.5 GaussianNB

A Gaussian Naive Bayes algorithm is a special type of NB algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e, normal distribution.

[95]:
```
%%time
# Fit a GaussianNB model to the train dataset

# Import GaussianNB


# Instantiate the model


# Fit the model to the data


# print score on train and valid set


# print classification report of the model using function created before
```

```
Training set accuracy:  0.629
Test set accuracy:  0.638
Classification Report
              precision    recall  f1-score   support

           1       0.63      0.93      0.75       151
           2       0.67      0.21      0.32       103

    accuracy                           0.64       254
   macro avg       0.65      0.57      0.54       254
weighted avg       0.65      0.64      0.58       254

CPU times: user 14.2 ms, sys: 2.99 ms, total: 17.2 ms
Wall time: 19.2 ms
```

6.2.6 K Neighbors Classifier

K-Nearest Neighbor is a simple supervised classification algorithm. It can be used for regression as well as classification. It is non-parametric as it does not make assumption on the data distribution.

[96]:
```
%%time
# Fit a K-Neighbour classifier model to the train dataset

# Import KNeighborsClassifier


# Instantiate the model


# fitting the model on train data

```

```
# print score on train and valid set


# print classification report of the model using function created before
```

```
Training set accuracy:  0.967
Test set accuracy:  0.929
Classification Report
              precision    recall  f1-score   support

           1       0.91      0.98      0.94       151
           2       0.97      0.85      0.91       103

    accuracy                           0.93       254
   macro avg       0.94      0.92      0.92       254
weighted avg       0.93      0.93      0.93       254

CPU times: user 210 ms, sys: 382 µs, total: 211 ms
Wall time: 212 ms
```

6.2.7 Decision Tree Classifier

Decision Tree Classifier is a simple and widely used classification technique. It applies a straitforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the calss label of the record is reached.

[97]:
```
%%time
# Fit a DecisionTreeClassifier model to the train dataset

# Import DecisionTreeClassifier


# Instantiate the model


# fitting the model on train data


# print score on train and valid set
```

```
Training set accuracy:  1.000
Test set accuracy:  0.965
CPU times: user 14.3 ms, sys: 93 µs, total: 14.4 ms
Wall time: 13.1 ms
```

6.2.8 Gradient Boosting Classifier

Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

[98]:
```
%%time
# Fit a Gradient Boosting Classifier model to the train dataset

# Import GradientBoostingClassifier


# Instantiate the model


# fitting the model on train data


# print score on train and valid set


# print classification report of the model using function created before
```

```
Training set accuracy:  0.872
Test set accuracy:  0.783
Classification Report
              precision    recall  f1-score   support

           1       0.77      0.91      0.83       151
           2       0.82      0.59      0.69       103

    accuracy                           0.78       254
   macro avg       0.80      0.75      0.76       254
weighted avg       0.79      0.78      0.78       254

CPU times: user 193 ms, sys: 1.81 ms, total: 195 ms
Wall time: 196 ms
```

6.2.9 Bagging Classifier

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

[99]:
```
%%time
# Fit a Bagging Classifier model to the train dataset

# Import BaggingClassifier

```

```
# Instantiate the model


# fitting the model on train data


# print score on train and valid set


# print classification report of the model using function created before
```

```
Training set accuracy:  1.000
Test set accuracy:  0.984
Classification Report
              precision    recall  f1-score   support

           1       0.97      1.00      0.99       151
           2       1.00      0.96      0.98       103

    accuracy                           0.98       254
   macro avg       0.99      0.98      0.98       254
weighted avg       0.98      0.98      0.98       254

CPU times: user 56 ms, sys: 852 µs, total: 56.8 ms
Wall time: 55.8 ms
```

6.2.10 Easy Ensemble Classifier

This algorithm is known as EasyEnsemble. The classifier is an ensemble of AdaBoost learners trained on different balanced boostrap samples. The balancing is achieved by random under-sampling.

[100]:
```
%%time
# Fit a EasyEnsembleClassifier model to the train dataset

# Import EasyEnsembleClassifier


# Instantiate the model


# fitting the model on train data


# print score on train and valid set


```

```
# print classification report of the model using function created before
```

```
Training set accuracy:  0.686
Test set accuracy:  0.634
Classification Report
              precision    recall  f1-score   support

           1       0.71      0.65      0.68       151
           2       0.54      0.61      0.58       103

    accuracy                           0.63       254
   macro avg       0.63      0.63      0.63       254
weighted avg       0.64      0.63      0.64       254

CPU times: user 1.47 s, sys: 18.3 ms, total: 1.49 s
Wall time: 1.54 s
```

6.2.11 AdaBoost Classifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

[101]:
```
%%time
# Fit a AdaBoost classifier model to the train dataset

# Import AdaBoostClassifier


# Instantiate the model


# fitting the model on train data


# print score on train and valid set


# print classification report of the model using function created before
```

```
Training set accuracy:  0.697
Test set accuracy:  0.673
Classification Report
              precision    recall  f1-score   support

           1       0.69      0.83      0.75       151
           2       0.64      0.45      0.53       103
```

```
    accuracy                           0.67       254
   macro avg       0.66      0.64      0.64       254
weighted avg       0.67      0.67      0.66       254

CPU times: user 172 ms, sys: 634 µs, total: 173 ms
Wall time: 177 ms
```

6.2.12 Now working with test dataset (year 2021 matches)

[102]:
```
# test data labels
```

[102]:

	Winning_team
1269	2
1270	1
1271	1
1272	1
1273	1
1274	1
1275	1
1276	1
1277	1
1278	1
1279	1
1280	1
1281	1
1282	1
1283	2
1284	1
1285	2
1286	1
1287	2
1288	1
1289	2
1290	1
1291	1
1292	2
1293	1
1294	2
1295	1
1296	1
1297	1
1298	1
1299	1
1300	1
1301	1
1302	2

```
1303            2
1304            2
1305            2
1306            2
1307            2
1308            2
1309            1
1310            1
1311            2
```

[103]:
```
# Creating a dataframe for with data Team 1 , Team 2 and Winner with data of all year 2021 matches
```

[103]:

	Team 1	Team 2	Winner
1269	U.A.E.	Ireland	Ireland
1270	Bangladesh	West Indies	Bangladesh
1271	Afghanistan	Ireland	Afghanistan
1272	Bangladesh	West Indies	Bangladesh
1273	Afghanistan	Ireland	Afghanistan
1274	Bangladesh	West Indies	Bangladesh
1275	Afghanistan	Ireland	Afghanistan
1276	West Indies	Sri Lanka	West Indies
1277	West Indies	Sri Lanka	West Indies
1278	West Indies	Sri Lanka	West Indies
1279	New Zealand	Bangladesh	New Zealand
1280	New Zealand	Bangladesh	New Zealand
1281	India	England	India
1282	New Zealand	Bangladesh	New Zealand
1283	India	England	England
1284	India	England	India
1285	South Africa	Pakistan	Pakistan
1286	South Africa	Pakistan	South Africa
1287	South Africa	Pakistan	Pakistan
1288	Netherlands	Scotland	Netherlands
1289	Netherlands	Scotland	Scotland
1290	Bangladesh	Sri Lanka	Bangladesh
1291	Bangladesh	Sri Lanka	Bangladesh
1292	Bangladesh	Sri Lanka	Sri Lanka
1293	Netherlands	Ireland	Netherlands
1294	Netherlands	Ireland	Ireland
1295	Netherlands	Ireland	Netherlands
1296	England	Sri Lanka	England
1297	England	Sri Lanka	England
1298	England	Pakistan	England
1299	England	Pakistan	England
1300	Ireland	South Africa	Ireland
1301	England	Pakistan	England

```
1302      Zimbabwe    Bangladesh    Bangladesh
1303       Ireland  South Africa  South Africa
1304      Zimbabwe    Bangladesh    Bangladesh
1305     Sri Lanka         India         India
1306      Zimbabwe    Bangladesh    Bangladesh
1307     Sri Lanka         India         India
1308   West Indies     Australia     Australia
1309   West Indies     Australia   West Indies
1310     Sri Lanka         India     Sri Lanka
1311   West Indies     Australia     Australia
```

[108]:
```
# Predicting on test data


#  Creating a dataframe with columns "Actual" and "Predicted"


# 'Actual' column data is true test y label


# 'Predicted' column is predicted labels


# creat a list for storing coverted winner data from numerical label to object


# looping through res['Predicted'],teams['Team 1'],teams['Team 2']␣
↪simultaneously using zip function in python

    # Check condition if predicted label is equal to 1

        # Append winner list the object name of team 1

    # Check condition if predicted label is equal to 2

        # Append winner list the object name of team 2


# Add a column 'PPred_Winner' with predicted winner data list
```

[109]:
```
# match_result data
```

[109]:
```
            Team 1        Team 2        Winner   Pred_Winner
1269        U.A.E.       Ireland       Ireland       Ireland
1270    Bangladesh   West Indies    Bangladesh    Bangladesh
1271   Afghanistan       Ireland   Afghanistan   Afghanistan
1272    Bangladesh   West Indies    Bangladesh    Bangladesh
```

1273	Afghanistan	Ireland	Afghanistan	Afghanistan
1274	Bangladesh	West Indies	Bangladesh	Bangladesh
1275	Afghanistan	Ireland	Afghanistan	Afghanistan
1276	West Indies	Sri Lanka	West Indies	West Indies
1277	West Indies	Sri Lanka	West Indies	West Indies
1278	West Indies	Sri Lanka	West Indies	West Indies
1279	New Zealand	Bangladesh	New Zealand	New Zealand
1280	New Zealand	Bangladesh	New Zealand	New Zealand
1281	India	England	India	India
1282	New Zealand	Bangladesh	New Zealand	New Zealand
1283	India	England	England	England
1284	India	England	India	India
1285	South Africa	Pakistan	Pakistan	Pakistan
1286	South Africa	Pakistan	South Africa	South Africa
1287	South Africa	Pakistan	Pakistan	Pakistan
1288	Netherlands	Scotland	Netherlands	Netherlands
1289	Netherlands	Scotland	Scotland	Scotland
1290	Bangladesh	Sri Lanka	Bangladesh	Bangladesh
1291	Bangladesh	Sri Lanka	Bangladesh	Bangladesh
1292	Bangladesh	Sri Lanka	Sri Lanka	Sri Lanka
1293	Netherlands	Ireland	Netherlands	Ireland
1294	Netherlands	Ireland	Ireland	Ireland
1295	Netherlands	Ireland	Netherlands	Netherlands
1296	England	Sri Lanka	England	England
1297	England	Sri Lanka	England	England
1298	England	Pakistan	England	England
1299	England	Pakistan	England	England
1300	Ireland	South Africa	Ireland	Ireland
1301	England	Pakistan	England	England
1302	Zimbabwe	Bangladesh	Bangladesh	Bangladesh
1303	Ireland	South Africa	South Africa	South Africa
1304	Zimbabwe	Bangladesh	Bangladesh	Bangladesh
1305	Sri Lanka	India	India	India
1306	Zimbabwe	Bangladesh	Bangladesh	Bangladesh
1307	Sri Lanka	India	India	India
1308	West Indies	Australia	Australia	Australia
1309	West Indies	Australia	West Indies	West Indies
1310	Sri Lanka	India	Sri Lanka	Sri Lanka
1311	West Indies	Australia	Australia	Australia

6.3 Conclusion

As we used different models for predicting match winner. we have seen that Random forest outperformed all other model. Accuracy of 100% on train data set and around 98% on validation set with f1 score more than 95% that is really good.

Also other models such as SVC, KNN classifier, Decision tree classifier and Gradient Boost classifier performed really well with acuarcy more than 90% and f1 score above 90%.

We understood how important is it to do feature engineering, feature scaling before feeding the data to model also we handled multiclass classification problem by converting it two class.

The prediction of winner in matches played in year 2021 were almost all were predicted correctly.