In [1]:
```
# importing the necessary libraries
import pandas as pd
import numpy as np

import random
import warnings
warnings.filterwarnings('ignore')


from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

In [2]:
```
data = pd.read_csv("yds_data.csv")
```

In [3]:
```
data.shape # checking how many rows and columns are in the data
```

Out[3]:
```
(30697, 28)
```

In [4]:
```
data.head() # seeing how the data looks like
```

Out[4]:

| | Unnamed: 0 | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_ma |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 10.0 | 167.0 | 72.0 | 10.0 | 1.0 | |
| **1** | 1 | 12.0 | -157.0 | 0.0 | 10.0 | 1.0 | |
| **2** | 2 | 35.0 | -101.0 | 135.0 | 7.0 | 1.0 | |
| **3** | 3 | 43.0 | 138.0 | 175.0 | 6.0 | 1.0 | |
| **4** | 4 | 155.0 | 0.0 | 0.0 | NaN | 2.0 | |

5 rows × 28 columns

# A. Data Preprocessing

## 1. Exploring the Columns of Dataset

In [5]:
```
data.describe()
```

Out[5]:

| | Unnamed: 0 | match_event_id | location_x | location_y | remaining_min | power_of_shot |
|---|---|---|---|---|---|---|
| **count** | 30697.000000 | 29134.000000 | 29236.000000 | 29157.000000 | 29135.000000 | 29211.000000 |
| **mean** | 15348.000000 | 249.576028 | 7.383876 | 91.126933 | 4.883233 | 2.519359 |
| **std** | 8861.604943 | 150.186019 | 110.263049 | 87.676395 | 3.452533 | 1.153976 |
| **min** | 0.000000 | 2.000000 | -250.000000 | -44.000000 | 0.000000 | 1.000000 |
| **25%** | 7674.000000 | 111.000000 | -68.000000 | 4.000000 | 2.000000 | 1.000000 |
| **50%** | 15348.000000 | 254.000000 | 0.000000 | 74.000000 | 5.000000 | 3.000000 |
| **75%** | 23022.000000 | 369.000000 | 95.000000 | 160.000000 | 8.000000 | 3.000000 |
| **max** | 30696.000000 | 659.000000 | 248.000000 | 791.000000 | 11.000000 | 7.000000 |

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30697 entries, 0 to 30696
Data columns (total 28 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            30697 non-null  int64  
 1   match_event_id        29134 non-null  float64
 2   location_x            29236 non-null  float64
 3   location_y            29157 non-null  float64
 4   remaining_min         29135 non-null  float64
 5   power_of_shot         29211 non-null  float64
 6   knockout_match        29180 non-null  float64
 7   game_season           24835 non-null  object 
 8   remaining_sec         29103 non-null  float64
 9   distance_of_shot      29130 non-null  float64
 10  is_goal               24429 non-null  float64
 11  area_of_shot          29195 non-null  object 
 12  shot_basics           29122 non-null  object 
 13  range_of_shot         29133 non-null  object 
 14  team_name             29162 non-null  object 
 15  date_of_game          29147 non-null  object 
 16  home/away             29200 non-null  object 
 17  shot_id_number        29134 non-null  float64
 18  lat/lng               29132 non-null  object 
 19  type_of_shot          15417 non-null  object 
 20  type_of_combined_shot 15280 non-null  object 
 21  match_id              30697 non-null  int64  
 22  team_id               30697 non-null  int64  
 23  remaining_min.1       29162 non-null  float64
 24  power_of_shot.1       29158 non-null  float64
 25  knockout_match.1      29204 non-null  float64
 26  remaining_sec.1       29158 non-null  float64
 27  distance_of_shot.1    29129 non-null  float64
dtypes: float64(15), int64(3), object(10)
memory usage: 6.6+ MB
```

## 2. Checking for Missing Values

In [7]:
```
missing_data = pd.DataFrame({'total_missing': data.isnull().sum(), 'perc_missing':
missing_data
```

Out[7]:

| | total_missing | perc_missing |
|---|---|---|
| **Unnamed: 0** | 0 | 0.000000 |
| **match_event_id** | 1563 | 5.091703 |
| **location_x** | 1461 | 4.759423 |
| **location_y** | 1540 | 5.016777 |
| **remaining_min** | 1562 | 5.088445 |
| **power_of_shot** | 1486 | 4.840864 |
| **knockout_match** | 1517 | 4.941851 |
| **game_season** | 5862 | 19.096329 |
| **remaining_sec** | 1594 | 5.192690 |
| **distance_of_shot** | 1567 | 5.104733 |
| **is_goal** | 6268 | 20.418933 |
| **area_of_shot** | 1502 | 4.892986 |
| **shot_basics** | 1575 | 5.130795 |
| **range_of_shot** | 1564 | 5.094960 |
| **team_name** | 1535 | 5.000489 |
| **date_of_game** | 1550 | 5.049353 |
| **home/away** | 1497 | 4.876698 |
| **shot_id_number** | 1563 | 5.091703 |
| **lat/lng** | 1565 | 5.098218 |
| **type_of_shot** | 15280 | 49.776851 |
| **type_of_combined_shot** | 15417 | 50.223149 |
| **match_id** | 0 | 0.000000 |
| **team_id** | 0 | 0.000000 |
| **remaining_min.1** | 1535 | 5.000489 |
| **power_of_shot.1** | 1539 | 5.013519 |
| **knockout_match.1** | 1493 | 4.863667 |
| **remaining_sec.1** | 1539 | 5.013519 |
| **distance_of_shot.1** | 1568 | 5.107991 |

In [8]:
```
# Exploring The Target Variable 'is_goal'
data.is_goal.value_counts()
```

Out[8]:
```
0.0    13550
1.0    10879
Name: is_goal, dtype: int64
```

" It's a binary classification problem as there are only two values for the target ''is_goal" column

# B. Exploratory Data Analysis

## 1. Dropping unnessary Columns

In [9]:
```
#1. Droping Unnecessary Columns
data.drop(["Unnamed: 0",  'remaining_min.1', 'power_of_shot.1','knockout_match.1',
```

In [10]:
```
data.head() # looking at the dataset after transformation
```

Out[10]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 10.0 | 167.0 | 72.0 | 10.0 | 1.0 | 0.0 | 2( |
| **1** | 12.0 | -157.0 | 0.0 | 10.0 | 1.0 | 0.0 | 2( |
| **2** | 35.0 | -101.0 | 135.0 | 7.0 | 1.0 | 0.0 | 2( |
| **3** | 43.0 | 138.0 | 175.0 | 6.0 | 1.0 | 0.0 | 2( |
| **4** | 155.0 | 0.0 | 0.0 | NaN | 2.0 | 0.0 | 2( |

5 rows × 22 columns

In [11]:
```
data.columns # to see if the columns are dropped succesfully
```

Out[11]:
```
Index(['match_event_id', 'location_x', 'location_y', 'remaining_min',
       'power_of_shot', 'knockout_match', 'game_season', 'remaining_sec',
       'distance_of_shot', 'is_goal', 'area_of_shot', 'shot_basics',
       'range_of_shot', 'team_name', 'date_of_game', 'home/away',
       'shot_id_number', 'lat/lng', 'type_of_shot', 'type_of_combined_shot',
       'match_id', 'team_id'],
      dtype='object')
```

In [12]:
```
#2. Changing dtypes to datetime
data.date_of_game = pd.to_datetime(data.date_of_game, errors='coerce')
data['game_season'] = data['game_season'].astype('object')
data['game_season']
```

Out[12]:
```
0        2000-01
1        2000-01
2        2000-01
3        2000-01
4        2000-01
          ...
30692    1999-00
30693    1999-00
30694    1999-00
30695    1999-00
30696    1999-00
Name: game_season, Length: 30697, dtype: object
```

In [13]:
```
# Labelencoding the 'game_season'
```

In [14]:
```
l_unique = data['game_season'].unique() # fteching out the unique values from game_
l_unique
```

Out[14]:
```
array(['2000-01', nan, '2001-02', '2002-03', '2003-04', '2004-05',
       '2005-06', '2006-07', '2007-08', '2008-09', '2009-10', '2010-11',
       '2011-12', '2012-13', '2013-14', '2014-15', '2015-16', '1996-97',
       '1997-98', '1998-99', '1999-00'], dtype=object)
```

In [15]:
```
v_unique = np.arange(len(l_unique)) # obtaining values in the range of the length c
v_unique
```

Out[15]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20])
```

In [16]:
```
data['game_season'].replace(to_replace=l_unique, value=v_unique, inplace=True) # re
data['game_season'].head()
```

Out[16]:
```
0    0
1    0
2    0
3    0
4    0
Name: game_season, dtype: int32
```

In [17]:
```
data['game_season'] = data['game_season'].astype('int') # converting the datatype c
data['game_season'].head()
```

Out[17]:
```
0    0
1    0
2    0
3    0
4    0
Name: game_season, dtype: int32
```

## 3. Handeling Missing Values

In [18]:
```
# Filling NaN values in Column "remaining_sec" with MEAN
data['power_of_shot'].fillna(value=data['power_of_shot'].mean(), inplace=True)
data.isnull().sum() # number of missing values for power_of_shot column should be z
```

Out[18]:
```
match_event_id           1563
location_x               1461
location_y               1540
remaining_min            1562
power_of_shot               0
knockout_match           1517
game_season                 0
remaining_sec            1594
distance_of_shot         1567
is_goal                  6268
area_of_shot             1502
shot_basics              1575
range_of_shot            1564
team_name                1535
date_of_game             1550
home/away                1497
shot_id_number           1563
lat/lng                  1565
type_of_shot            15280
type_of_combined_shot   15417
match_id                    0
team_id                     0
dtype: int64
```

In [19]:
```
# Filling NaN values in Column "type_of_combined_shot" with MODE
mode_com  = data.type_of_combined_shot.value_counts().keys()[0]
print('moded is: ',mode_com)
```

```
data.type_of_combined_shot.fillna(value=mode_com, inplace=True)
data.isnull().sum() # number of missing values for type_of_combined_shot column sho
```

Out[19]:

```
moded is:  shot - 3
match_event_id            1563
location_x                1461
location_y                1540
remaining_min             1562
power_of_shot                0
knockout_match            1517
game_season                  0
remaining_sec             1594
distance_of_shot          1567
is_goal                   6268
area_of_shot              1502
shot_basics               1575
range_of_shot             1564
team_name                 1535
date_of_game              1550
home/away                 1497
shot_id_number            1563
lat/lng                   1565
type_of_shot             15280
type_of_combined_shot        0
match_id                     0
team_id                      0
dtype: int64
```

In [20]:

```
# Filling NaN values in Column "remaining_sec" with MEDIAN
data.remaining_sec.fillna(value=data.remaining_sec.median(), inplace=True)
data.isnull().sum() # number of missing values for remaining_sec column should be z
```

Out[20]:

```
match_event_id            1563
location_x                1461
location_y                1540
remaining_min             1562
power_of_shot                0
knockout_match            1517
game_season                  0
remaining_sec                0
distance_of_shot          1567
is_goal                   6268
area_of_shot              1502
shot_basics               1575
range_of_shot             1564
team_name                 1535
date_of_game              1550
home/away                 1497
shot_id_number            1563
lat/lng                   1565
type_of_shot             15280
type_of_combined_shot        0
match_id                     0
team_id                      0
dtype: int64
```

In [21]:

```
# Shot_id_no.
data.shot_id_number = pd.Series(np.arange(1,data.shot_id_number.shape[0]+1))
data.isnull().sum() # number of missing values for shot_id_number column should be
```

Out[21]:
```
match_event_id            1563
location_x                1461
location_y                1540
remaining_min             1562
power_of_shot                0
knockout_match            1517
game_season                  0
remaining_sec                0
distance_of_shot          1567
is_goal                   6268
area_of_shot              1502
shot_basics               1575
range_of_shot             1564
team_name                 1535
date_of_game              1550
home/away                 1497
shot_id_number               0
lat/lng                   1565
type_of_shot             15280
type_of_combined_shot        0
match_id                     0
team_id                      0
dtype: int64
```

In [22]:
```
data['location_x'].fillna(value=0, inplace=True)
data['location_y'].fillna(value=0, inplace=True)
data.isnull().sum()
```

Out[22]:
```
match_event_id            1563
location_x                   0
location_y                   0
remaining_min             1562
power_of_shot                0
knockout_match            1517
game_season                  0
remaining_sec                0
distance_of_shot          1567
is_goal                   6268
area_of_shot              1502
shot_basics               1575
range_of_shot             1564
team_name                 1535
date_of_game              1550
home/away                 1497
shot_id_number               0
lat/lng                   1565
type_of_shot             15280
type_of_combined_shot        0
match_id                     0
team_id                      0
dtype: int64
```

In [23]:
```
print('Null values in column home/away before forward fill =',data['home/away'].isr
col = ['home/away','lat/lng', 'team_name','match_id','match_event_id', 'team_id', '
data.loc[:,col] = data.loc[:,col].ffill()
print('Null values in column home/away after the forward fill =',data['home/away'].
```

```
Null values in column home/away before forward fill = 1497
Null values in column home/away after the forward fill = 0
```

In [24]:
```
# Filling Missing Values In "shot_basics" based on "range_of_short" column!
# if the range of the shot is 16-24 ft it's a mid range shot
data.loc[(data.range_of_shot == '16-24 ft.'), 'shot_basics'] = data[data.range_of_s

# if the range of the shot is less than 8 ft then randomly assign goal line or goal
```

```
data.loc[(data.range_of_shot == 'Less Than 8 ft.')&(data.shot_basics.isnull()), 'sh
# if the range of the shot is  8-16 ft then randomly assign goal line or mid range
data.loc[(data.range_of_shot == '8-16 ft.')&(data.shot_basics.isnull()), 'shot_basi
# if the range of the shot is more than 24 ft then randomly assign one of the value
data.loc[(data.range_of_shot == '24+ ft.')&(data.shot_basics.isnull()), 'shot_basic
# if the shot is a back court shot then randomly assign one of the values from''Mid
data.loc[(data.range_of_shot == 'Back Court Shot')&(data.shot_basics.isnull()), 'sh
data.isna().sum()
```

Out[24]:

```
match_event_id              0
location_x                  0
location_y                  0
remaining_min               0
power_of_shot               0
knockout_match              0
game_season                 0
remaining_sec               0
distance_of_shot         1567
is_goal                  6268
area_of_shot             1502
shot_basics                66
range_of_shot            1564
team_name                   0
date_of_game             1550
home/away                   0
shot_id_number              0
lat/lng                     0
type_of_shot            15280
type_of_combined_shot       0
match_id                    0
team_id                     0
dtype: int64
```

In [25]:

```
data['shot_basics'].unique() # now we have populated the shot types and reduced the
```

Out[25]:

```
array(['Mid Range', 'Goal Area', 'Goal Line', 'Penalty Spot', nan,
       'Right Corner', 'Mid Ground Line', 'Left Corner'], dtype=object)
```

In [26]:

```
# Filling Missing Values In "range_of_short" based on "short_basics" column!

# if shot_basics is Goal Area, then range of shot is Less Than 8 ft
data.loc[(data.shot_basics == 'Goal Area'), 'range_of_shot']       = data[data.shot
# if shot_basics is Penalty Spot, then range of shot is  24+ ft.
data.loc[(data.shot_basics == 'Penalty Spot'), 'range_of_shot']    = data[data.shot
# if shot_basics is Right Corner, then range of shot is  24+ ft.
data.loc[(data.shot_basics == 'Right Corner'), 'range_of_shot']    = data[data.shot
# if shot_basics is Left Corner, then range of shot is  24+ ft.
data.loc[(data.shot_basics == 'Left Corner'), 'range_of_shot']     = data[data.shot
# if shot_basics is Mid Ground Line , then range of shot is  Back Court Shot
data.loc[(data.shot_basics == 'Mid Ground Line'), 'range_of_shot'] = data[data.shot
# if shot_basics is Mid Range then randomly assign '16-24 ft.' or  '8-16 ft.' to ra
data.loc[(data.shot_basics == 'Mid Range')&(data.range_of_shot.isnull()), 'range_of
# if shot_basics is Goal Line then randomly assign ''8-16 ft.' or  'Less Than 8 ft.
data.loc[(data.shot_basics == 'Goal Line')&(data.range_of_shot.isnull()), 'range_of

data.isnull().sum() # number of missing values for range_of_shot column should have
```

Out[26]:
```
match_event_id              0
location_x                  0
location_y                  0
remaining_min               0
power_of_shot               0
knockout_match              0
game_season                 0
remaining_sec               0
distance_of_shot         1567
is_goal                  6268
area_of_shot             1502
shot_basics                66
range_of_shot              66
team_name                   0
date_of_game             1550
home/away                   0
shot_id_number              0
lat/lng                     0
type_of_shot            15280
type_of_combined_shot       0
match_id                    0
team_id                     0
dtype: int64
```

In [27]:
```
data['range_of_shot'].unique() # the number of missing values has fallen from 1564
```

Out[27]:
```
array(['16-24 ft.', '8-16 ft.', 'Less Than 8 ft.', '24+ ft.', nan,
       'Back Court Shot'], dtype=object)
```

In [28]:
```
# Filling the remaining missing values incase they both have NaN values using the f
data.shot_basics.fillna(method='ffill', inplace=True)
data.range_of_shot.fillna(method='ffill', inplace=True)
data.isnull().sum() # number of missing values for shot_basics and range_of_shot co
```

Out[28]:
```
match_event_id              0
location_x                  0
location_y                  0
remaining_min               0
power_of_shot               0
knockout_match              0
game_season                 0
remaining_sec               0
distance_of_shot         1567
is_goal                  6268
area_of_shot             1502
shot_basics                 0
range_of_shot               0
team_name                   0
date_of_game             1550
home/away                   0
shot_id_number              0
lat/lng                     0
type_of_shot            15280
type_of_combined_shot       0
match_id                    0
team_id                     0
dtype: int64
```

In [29]:
```
# Filling the missing value in "ärea_of_short" Column
data.area_of_shot.fillna(value='Center(C)', inplace=True) # all the missing values
data.isnull().sum() # number of missing values for area_of_shot column should be ze
```

Out[29]:
```
match_event_id              0
location_x                  0
location_y                  0
remaining_min               0
power_of_shot               0
knockout_match              0
game_season                 0
remaining_sec               0
distance_of_shot         1567
is_goal                  6268
area_of_shot                0
shot_basics                 0
range_of_shot               0
team_name                   0
date_of_game             1550
home/away                   0
shot_id_number              0
lat/lng                     0
type_of_shot            15280
type_of_combined_shot       0
match_id                    0
team_id                     0
dtype: int64
```

In [30]:
```
data['distance_of_shot'].unique()
```

Out[30]:
```
array([38., 35., 36., 42., 20., 34., 22., 32., 45., 37., nan, 29., 25.,
       40., 31., 27., 46., 39., 28., 33., 21., 47., 48., 44., 43., 24.,
       41., 67., 30., 49., 62., 23., 68., 50., 65., 26., 53., 56., 82.,
       51., 90., 63., 58., 57., 60., 52., 76., 55., 75., 71., 88., 59.,
       61., 84., 70., 69., 79., 80., 74., 94., 64., 81., 85., 72., 54.,
       66., 78., 89., 77., 73., 87., 91., 97., 99.])
```

In [31]:
```
#Filling the Missing values in "distance_of_shot"
# if distance_of_shot isnull randomly assign a value from 20,45,44,37
data.loc[data['distance_of_shot'].isnull(), 'distance_of_shot'] = pd.Series(data.lo
data.isnull().sum() # number of missing values for distance_of_shot column should b
```

Out[31]:
```
match_event_id              0
location_x                  0
location_y                  0
remaining_min               0
power_of_shot               0
knockout_match              0
game_season                 0
remaining_sec               0
distance_of_shot            0
is_goal                  6268
area_of_shot                0
shot_basics                 0
range_of_shot               0
team_name                   0
date_of_game             1550
home/away                   0
shot_id_number              0
lat/lng                     0
type_of_shot            15280
type_of_combined_shot       0
match_id                    0
team_id                     0
dtype: int64
```

# Making the Train and Test Dataset

# train and test data are divided based on the vaue of is goal column

In [32]:
```
# Making the train Dataset
train = data[data.is_goal.notnull()]
print('the Shape of Train Dataset',train.shape)
train.set_index(np.arange(train.shape[0]),inplace=True)
train.head()
```

```
the Shape of Train Dataset (24429, 22)
```

Out[32]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 12.0 | -157.0 | 0.0 | 10.0 | 1.0 | 0.0 | |
| **1** | 35.0 | -101.0 | 135.0 | 7.0 | 1.0 | 0.0 | |
| **2** | 43.0 | 138.0 | 175.0 | 6.0 | 1.0 | 0.0 | |
| **3** | 155.0 | 0.0 | 0.0 | 6.0 | 2.0 | 0.0 | |
| **4** | 244.0 | -145.0 | -11.0 | 9.0 | 3.0 | 0.0 | |

5 rows × 22 columns

In [33]:
```
# Making the Test Dataset
test = data[data.is_goal.isnull()]
print('The Shape of Test Dataset',test.shape)
test.set_index(np.arange(test.shape[0]), inplace=True)
test.head()
```

```
The Shape of Test Dataset (6268, 22)
```

Out[33]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 10.0 | 167.0 | 72.0 | 10.0 | 1.0 | 0.0 | |
| **1** | 254.0 | 1.0 | 28.0 | 8.0 | 3.0 | 0.0 | |
| **2** | 100.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| **3** | 249.0 | 0.0 | 0.0 | 10.0 | 3.0 | 0.0 | |
| **4** | 265.0 | 134.0 | 127.0 | 9.0 | 3.0 | 0.0 | |

5 rows × 22 columns

Handeling Missing Values in train and Test Dataset

## Filling the Nan value with a random choice from given list with there appropriate probablities

In [34]:
```
l_goal   = train[train.is_goal == 1].type_of_shot.value_counts().head(6).keys()
l_goal
```

Out[34]:
```
Index(['shot - 4', 'shot - 39', 'shot - 44', 'shot - 36', 'shot - 15',
       'shot - 38'],
      dtype='object')
```

In [35]:
```
p_g_sum  = train[train.is_goal == 1].type_of_shot.value_counts().head(6).sum() # To
p_goal   = (train[train.is_goal == 1].type_of_shot.value_counts().head(6) / p_g_sum
p_goal
```

Out[35]:
```
[0.2682060390763766,
 0.19182948490230906,
 0.14653641207815277,
 0.1447602131438721,
 0.12966252220248667,
 0.11900532859680284]
```

In [36]:
```
# if is_goal is 1, if type of shot is a string value, fill with the same or else fi
g = pd.Series(train[train.is_goal == 1].type_of_shot.apply(lambda x: x if type(x)==
g
```

Out[36]:
```
1        shot - 25
3         shot - 4
5        shot - 44
6        shot - 36
9        shot - 44
           ...
24411    shot - 33
24413    shot - 39
24415    shot - 44
24421    shot - 36
24426     shot - 4
Name: type_of_shot, Length: 10879, dtype: object
```

In [37]:
```
# # if is_goal is 1, if type of shot is null then type of shot becomes equal to the
train.loc[(train.is_goal == 1)&(train.type_of_shot.isnull()), 'type_of_shot'] = g
```

In [38]:
```
train['type_of_shot'].isna().sum() # number of missing values got reduced from more
```

Out[38]:
```
6723
```

## and we have applied similar concept for the scenarios when there was no goal

In [39]:
```
l_no_goal   = train[train.is_goal == 0].type_of_shot.value_counts().head(5).keys()
p_no_sum  = train[train.is_goal == 0].type_of_shot.value_counts().head(5).sum()
p_no_goal   = (train[train.is_goal == 0].type_of_shot.value_counts().head(5) / p_nc
ng = pd.Series(train[train.is_goal == 0].type_of_shot.apply(lambda x: x if type(x)=
train.loc[(train.is_goal == 0)&(train.type_of_shot.isnull()), 'type_of_shot'] = ng
train['type_of_shot'].isna().sum() # number of missing values got reduced to zero
```

Out[39]:
```
0
```

In [40]:
```
#Handeling the remaing values in test dataset with a smilira approach
test.loc[test['type_of_shot'].isnull(), 'type_of_shot'] = pd.Series(test.loc[test['
```

In [41]:
```
test['type_of_shot'].isna().sum() # we have removed the missing values from test se
```

Out[41]:
```
0
```

## Label Encoding the Object type Columns

In [42]:
```
%%time
# Labeling the catagories with integers
for col in train.columns:
    if train[col].dtypes == object: # if the column has categorical values
        l_unique = train[col].unique() # find the unique values
        v_unique = np.arange(len(l_unique)) # create a list of number from zero to
        train[col].replace(to_replace=l_unique, value=v_unique, inplace=True) # rep
        train[col] = train[col].astype('int') # change the type from int64 to int32

        # same has been done for test data as well
        test[col].replace(to_replace=l_unique, value=v_unique, inplace=True)
        test[col] = test[col].astype('int')
```

```
Wall time: 760 ms
```

In [43]:
```
# Dropping the unnecessary Columns
train.drop(['date_of_game'], axis=1, inplace=True)
train.head()
```

Out[43]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 12.0 | -157.0 | 0.0 | 10.0 | 1.0 | 0.0 | |
| **1** | 35.0 | -101.0 | 135.0 | 7.0 | 1.0 | 0.0 | |
| **2** | 43.0 | 138.0 | 175.0 | 6.0 | 1.0 | 0.0 | |
| **3** | 155.0 | 0.0 | 0.0 | 6.0 | 2.0 | 0.0 | |
| **4** | 244.0 | -145.0 | -11.0 | 9.0 | 3.0 | 0.0 | |

5 rows × 21 columns

In [44]:
```
test.drop(['date_of_game'], axis=1, inplace=True)
test.head()
```

Out[44]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 10.0 | 167.0 | 72.0 | 10.0 | 1.0 | 0.0 | |
| **1** | 254.0 | 1.0 | 28.0 | 8.0 | 3.0 | 0.0 | |
| **2** | 100.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| **3** | 249.0 | 0.0 | 0.0 | 10.0 | 3.0 | 0.0 | |
| **4** | 265.0 | 134.0 | 127.0 | 9.0 | 3.0 | 0.0 | |

5 rows × 21 columns

In [45]:
```
# Splliting the Target Column from the Dataset
y = train.is_goal
y.head()
```

Out[45]:
```
0    0.0
1    1.0
2    0.0
3    1.0
4    0.0
Name: is_goal, dtype: float64
```

In [46]:
```
train.drop(['is_goal'], axis=1, inplace=True)
train.head()
```

Out[46]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 12.0 | -157.0 | 0.0 | 10.0 | 1.0 | 0.0 | |
| **1** | 35.0 | -101.0 | 135.0 | 7.0 | 1.0 | 0.0 | |
| **2** | 43.0 | 138.0 | 175.0 | 6.0 | 1.0 | 0.0 | |
| **3** | 155.0 | 0.0 | 0.0 | 6.0 | 2.0 | 0.0 | |
| **4** | 244.0 | -145.0 | -11.0 | 9.0 | 3.0 | 0.0 | |

In [47]:
```
test.drop(['is_goal'], axis=1, inplace=True)
test.head()
```

Out[47]:

| | match_event_id | location_x | location_y | remaining_min | power_of_shot | knockout_match | game_s |
|---|---|---|---|---|---|---|---|
| **0** | 10.0 | 167.0 | 72.0 | 10.0 | 1.0 | 0.0 | |
| **1** | 254.0 | 1.0 | 28.0 | 8.0 | 3.0 | 0.0 | |
| **2** | 100.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| **3** | 249.0 | 0.0 | 0.0 | 10.0 | 3.0 | 0.0 | |
| **4** | 265.0 | 134.0 | 127.0 | 9.0 | 3.0 | 0.0 | |

In [48]:
```
train.info() # we have converted all the categorical columns to numeric ones
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24429 entries, 0 to 24428
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   match_event_id        24429 non-null  float64
 1   location_x            24429 non-null  float64
 2   location_y            24429 non-null  float64
 3   remaining_min         24429 non-null  float64
 4   power_of_shot         24429 non-null  float64
 5   knockout_match        24429 non-null  float64
 6   game_season           24429 non-null  int32  
 7   remaining_sec         24429 non-null  float64
 8   distance_of_shot      24429 non-null  float64
 9   area_of_shot          24429 non-null  int32  
 10  shot_basics           24429 non-null  int32  
 11  range_of_shot         24429 non-null  int32  
 12  team_name             24429 non-null  int32  
 13  home/away             24429 non-null  int32  
 14  shot_id_number        24429 non-null  int32  
 15  lat/lng               24429 non-null  int32  
 16  type_of_shot          24429 non-null  int32  
 17  type_of_combined_shot  24429 non-null  int32  
 18  match_id              24429 non-null  int64  
 19  team_id               24429 non-null  int64  
dtypes: float64(8), int32(10), int64(2)
memory usage: 3.0 MB
```

In [49]:
```
train.isna().sum() # we have don't have any missing values as well. Our data is rea
```

Out[49]:
```
match_event_id          0
location_x              0
location_y              0
remaining_min           0
power_of_shot           0
knockout_match          0
game_season             0
remaining_sec           0
distance_of_shot        0
area_of_shot            0
shot_basics             0
range_of_shot           0
team_name               0
home/away               0
shot_id_number          0
lat/lng                 0
type_of_shot            0
type_of_combined_shot   0
match_id                0
team_id                 0
dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: