CHRISTIANO RONALDO

Loading The Dataset

In [1]:
```
import pandas as pd
import numpy as np                          # For mathematical calculati
ons
import seaborn as sns                       # For data visualization
import matplotlib.pyplot as plt             # For plotting graphs
%matplotlib inline
import warnings                             # To ignore any warnings war
nings.filterwarnings("ignore")


from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
TrainDataPath = 'data.csv'
sample_submissionPath = 'sample_submission.csv'
# Loading the Training and Test Dataset
Train = pd.read_csv(TrainDataPath)
sample_submission = pd.read_csv(sample_submissionPath)
```

In [3]:
```
Train_oroginal=Train.copy()
```

In [4]:
```
print("Training Dataset Shape:")
print(Train.shape)
print("\n")
print("Training Dataset Columns/Features:")
print(Train.dtypes)
Train.head()
```

```
Training Dataset Shape:
(30697, 28)


Training Dataset Columns/Features:
Unnamed: 0                  int64
match_event_id            float64
location_x                float64
location_y                float64
remaining_min             float64
power_of_shot             float64
knockout_match            float64
game_season                object
remaining_sec             float64
distance_of_shot          float64
is_goal                   float64
area_of_shot               object
shot_basics                object
range_of_shot              object
team_name                  object
date_of_game               object
home/away                  object
shot_id_number            float64
lat/lng                    object
type_of_shot               object
type_of_combined_shot      object
match_id                    int64
team_id                     int64
remaining_min.1           float64
power_of_shot.1           float64
knockout_match.1          float64
remaining_sec.1           float64
distance_of_shot.1        float64
dtype: object
```

Out[4]:

	Unnamed: 0	match_event_id	location_x	location_y	remaining_min	power_of_shot	knock
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

In [5]:
```
print("Submission Dataset Shape:")
print(sample_submission.shape)
print("\n")
print("Submission Dataset Columns/Features:")
print(sample_submission.dtypes)
sample_submission.head()
```

```
Submission Dataset Shape:
(5000, 2)


Submission Dataset Columns/Features:
shot_id_number      int64
is_goal           float64
dtype: object
```

Out[5]:

	shot_id_number	is_goal
0	1	0.1
1	8	0.1
2	17	0.1
3	20	0.1
4	33	0.1

In [6]:
```
# checking missing data percentage in train data
total = Train.isnull().sum().sort_values(ascending = False)
percent = (Train.isnull().sum()/Train.isnull().count()*100).sort_va
lues(ascending = False)
missing_TrainData  = pd.concat([total, percent], axis=1, keys=['Tot
al', 'Percent'])
missing_TrainData.head(30)
```

Out[6]:

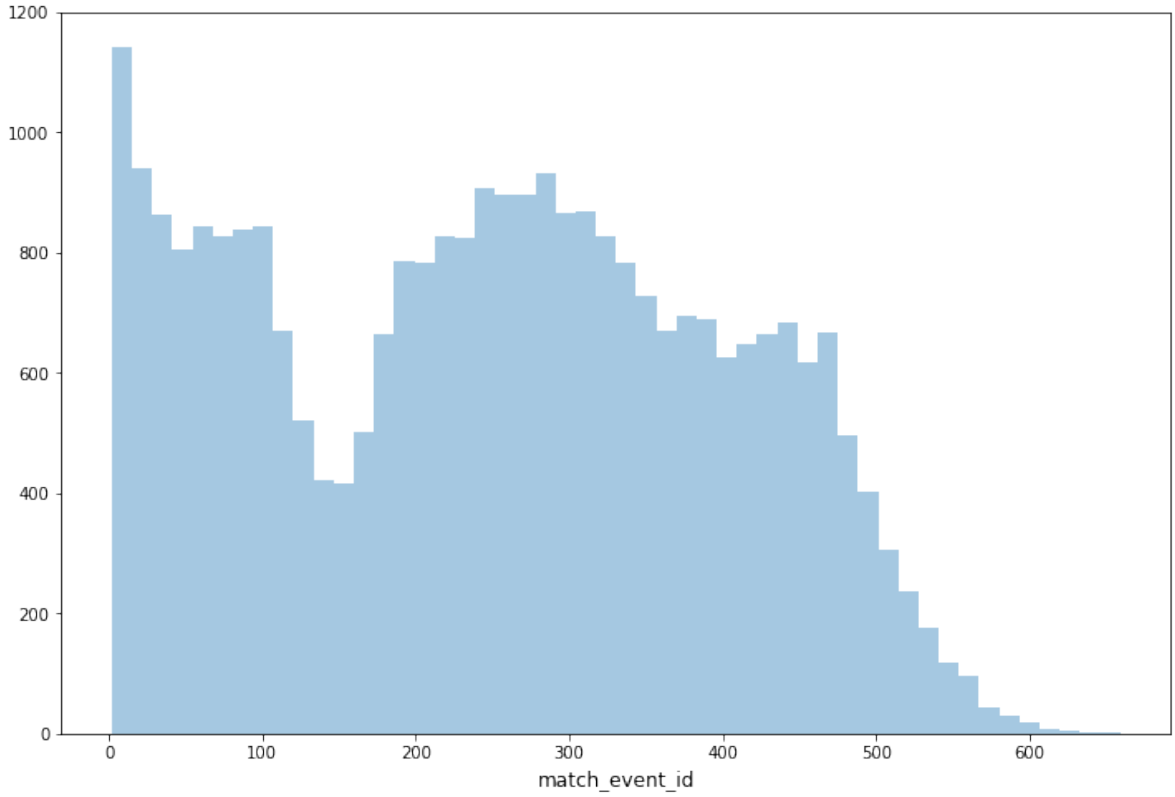
	Total	Percent
type_of_combined_shot	15417	50.223149
type_of_shot	15280	49.776851
is_goal	6268	20.418933
game_season	5862	19.096329
remaining_sec	1594	5.192690
shot_basics	1575	5.130795
distance_of_shot.1	1568	5.107991
distance_of_shot	1567	5.104733
lat/lng	1565	5.098218
range_of_shot	1564	5.094960
shot_id_number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

EDA(Exploratory Data Analysis)

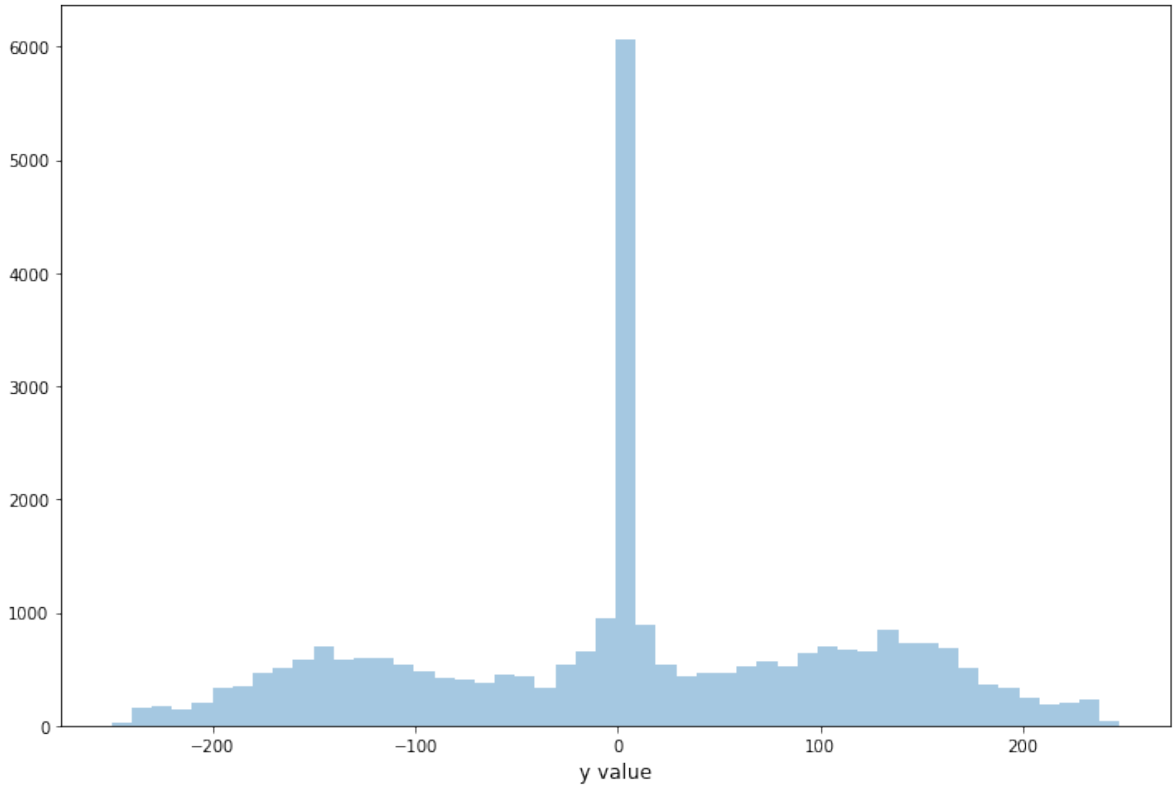
Univariate Analysis

In [7]:
```
plt.figure(figsize=(12,8))
sns.distplot(Train.match_event_id.values, bins=50, kde=False)
plt.xlabel('match_event_id', fontsize=12)
plt.show()
```

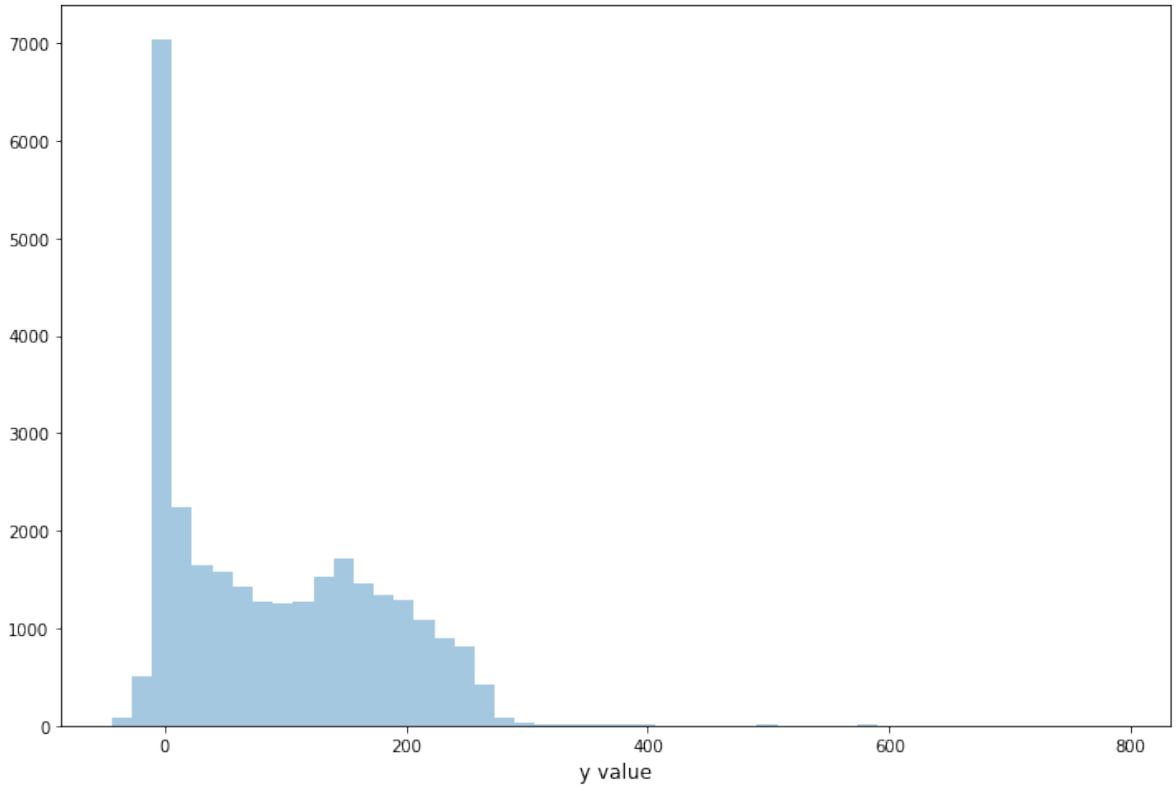
```
/Users/shubham/anaconda3/lib/python3.7/site-packages/numpy/lib/his
tograms.py:754: RuntimeWarning: invalid value encountered in great
er_equal
  keep = (tmp_a >= first_edge)
/Users/shubham/anaconda3/lib/python3.7/site-packages/numpy/lib/his
tograms.py:755: RuntimeWarning: invalid value encountered in less_
equal
  keep &= (tmp_a <= last_edge)
```



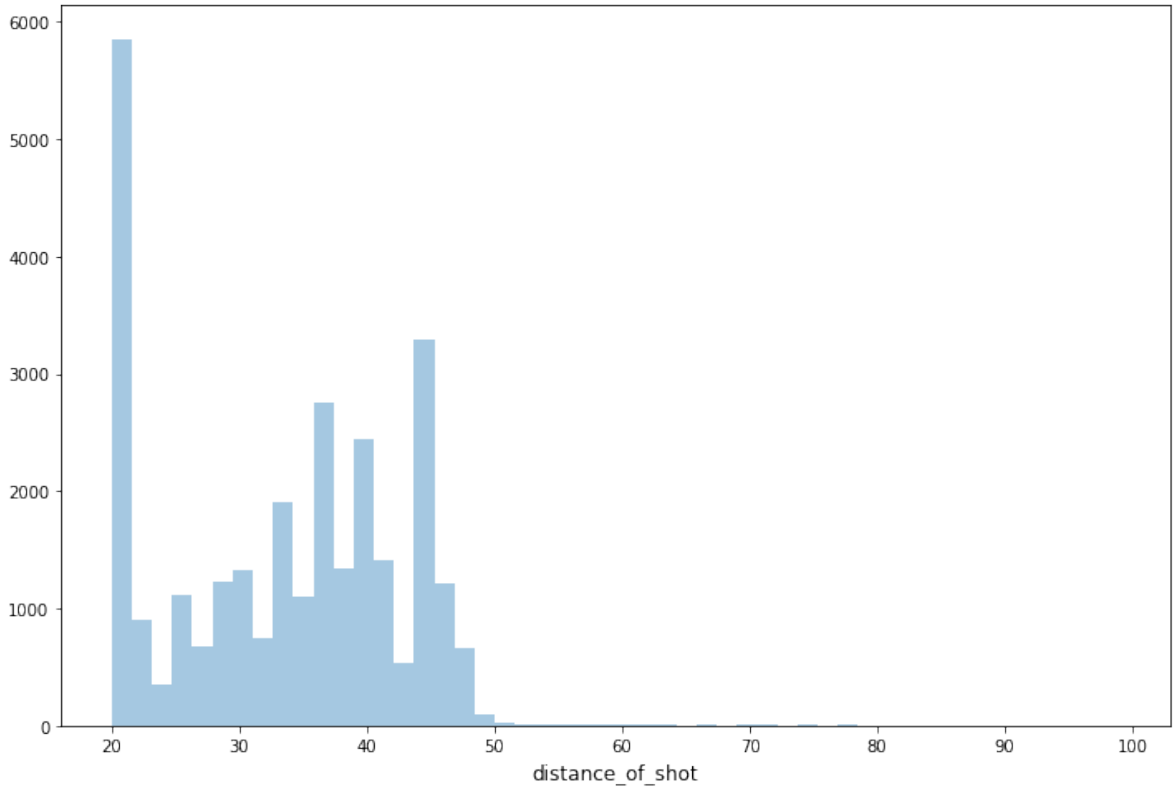
In [8]:
```
plt.figure(figsize=(12,8))
sns.distplot(Train.location_x.values, bins=50, kde=False)
plt.xlabel('y value', fontsize=12)
plt.show()
```



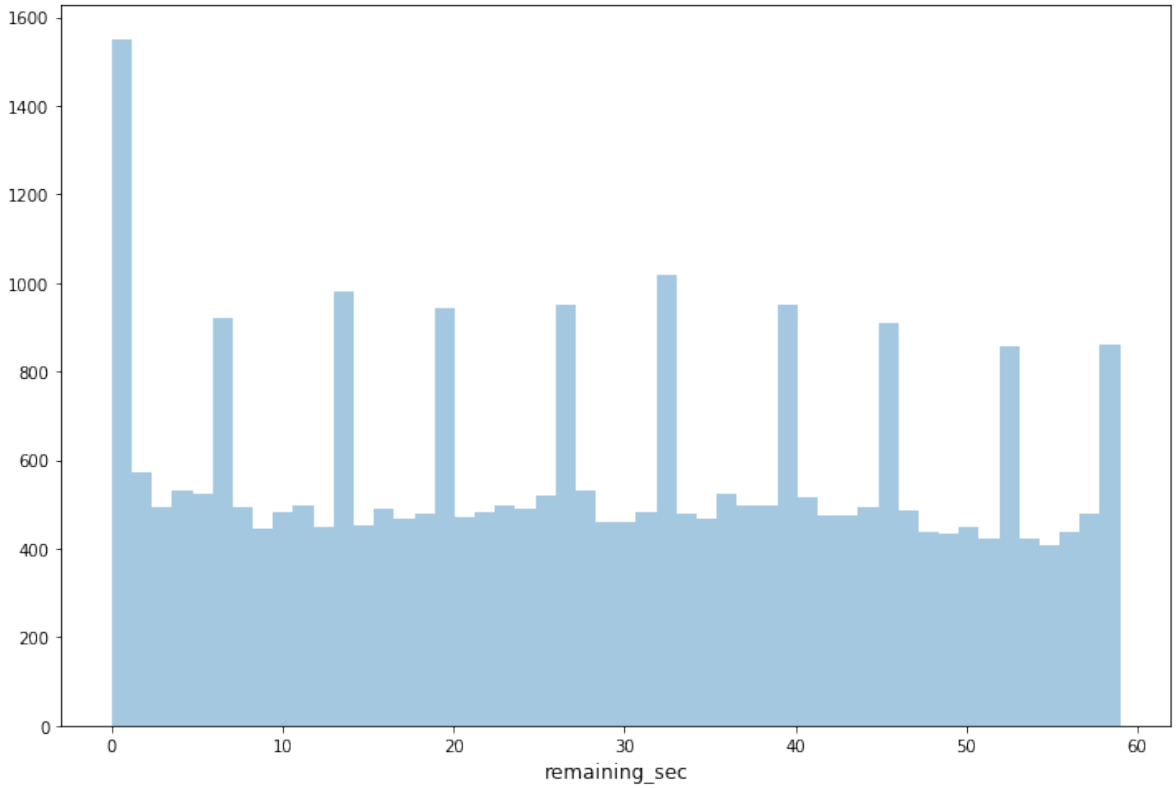
In [9]:
```
plt.figure(figsize=(12,8))
sns.distplot(Train.location_y.values, bins=50, kde=False)
plt.xlabel('y value', fontsize=12)
plt.show()
```



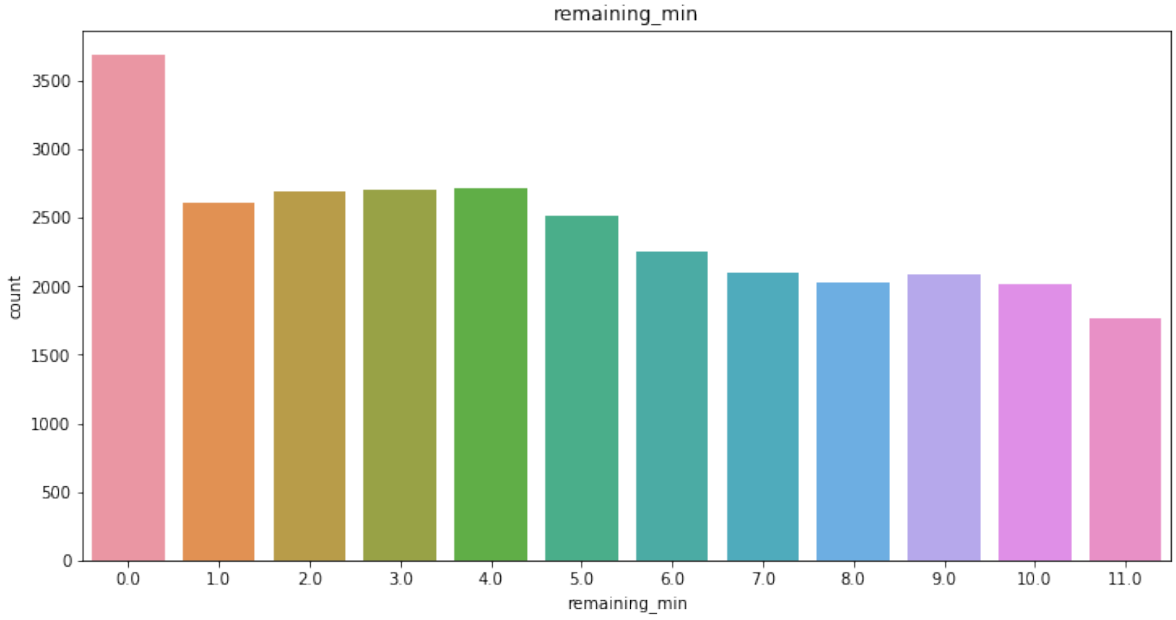
In [10]:
```
plt.figure(figsize=(12,8))
sns.distplot(Train.distance_of_shot.values, bins=50, kde=False)
plt.xlabel('distance_of_shot', fontsize=12)
plt.show()
```



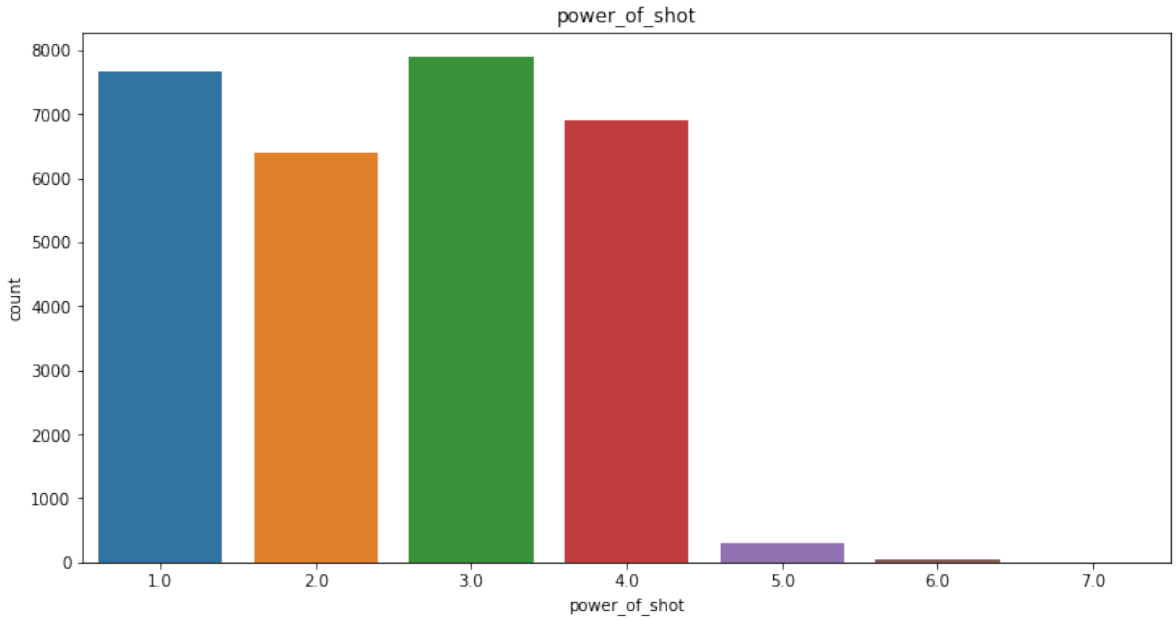
In [11]:
```
plt.figure(figsize=(12,8))
sns.distplot(Train.remaining_sec.values, bins=50, kde=False)
plt.xlabel('remaining_sec', fontsize=12)
plt.show()
```



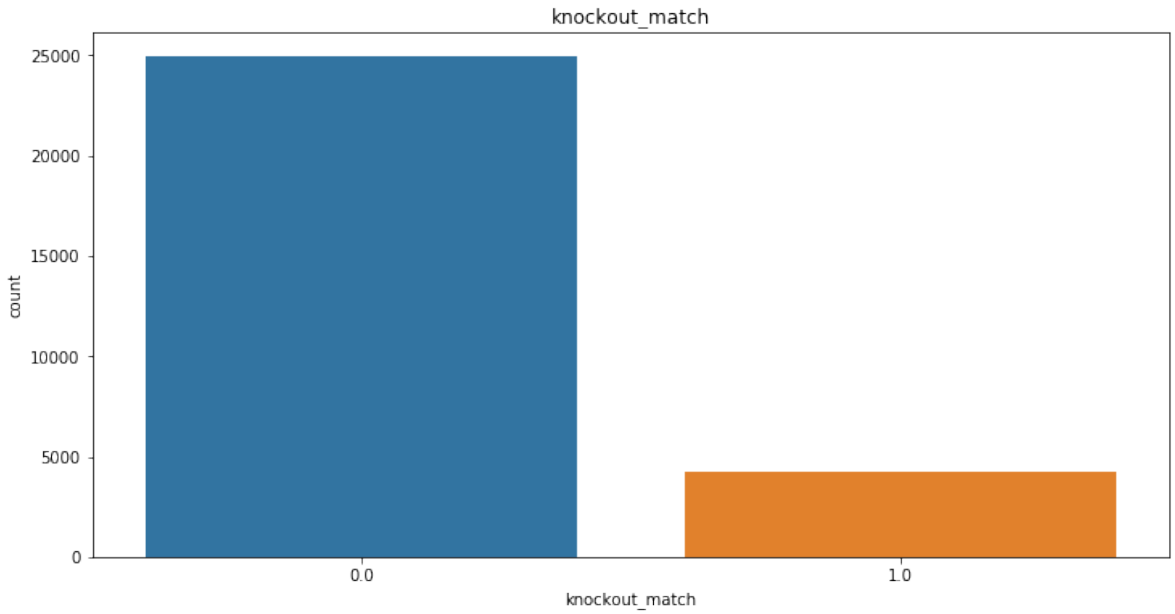
In [12]:
```
plt.figure(figsize=(12, 6))
sns.countplot(Train["remaining_min"])
plt.title('remaining_min')
plt.show()
```



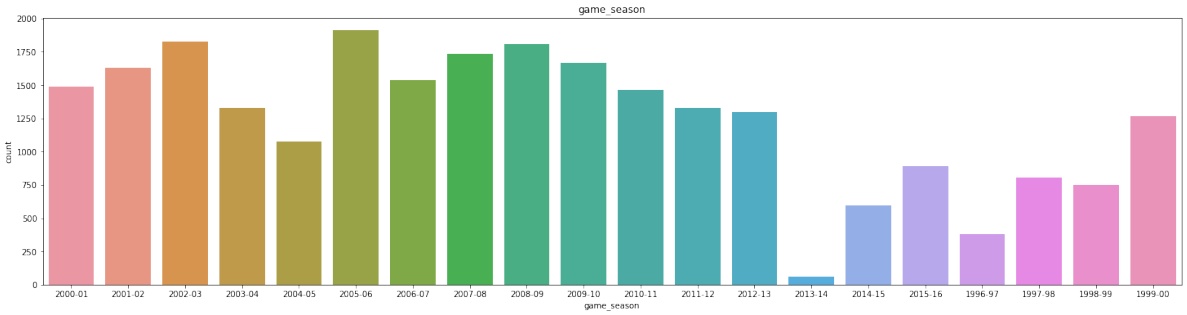
In [13]:
```
plt.figure(figsize=(12, 6))
sns.countplot(Train["power_of_shot"])
plt.title('power_of_shot')
plt.show()
```



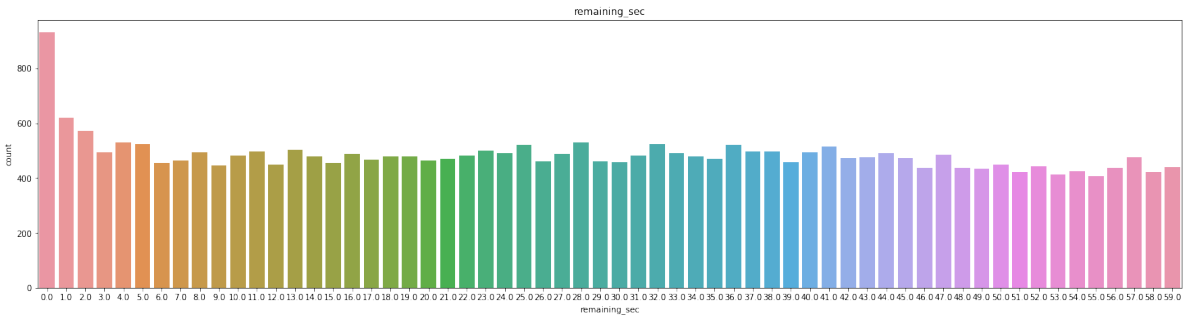
In [14]:
```
plt.figure(figsize=(12, 6))
sns.countplot(Train["knockout_match"])
plt.title('knockout_match')
plt.show()
```



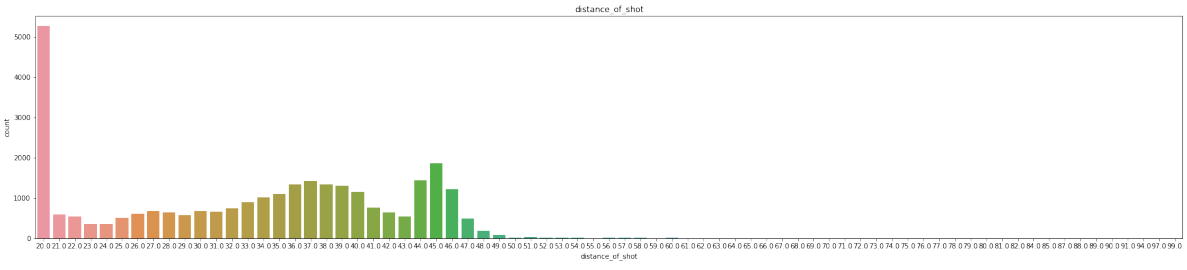
In [15]:
```
plt.figure(figsize=(25, 6))
sns.countplot(Train["game_season"])
plt.title('game_season')
plt.show()
```



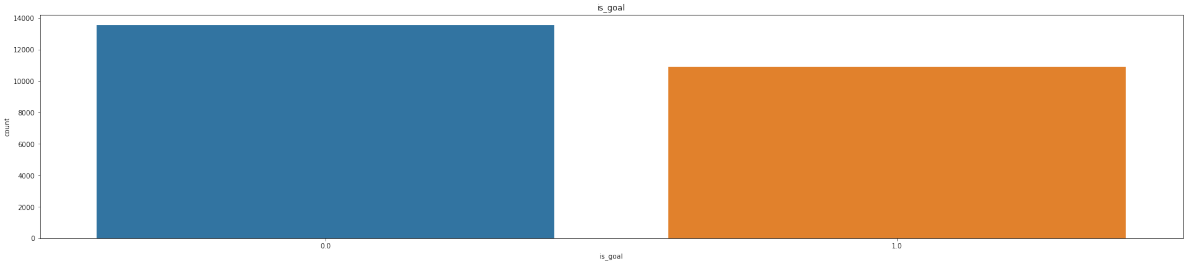
In [16]:
```
plt.figure(figsize=(25, 6))
sns.countplot(Train["remaining_sec"])
plt.title('remaining_sec')
plt.show()
```



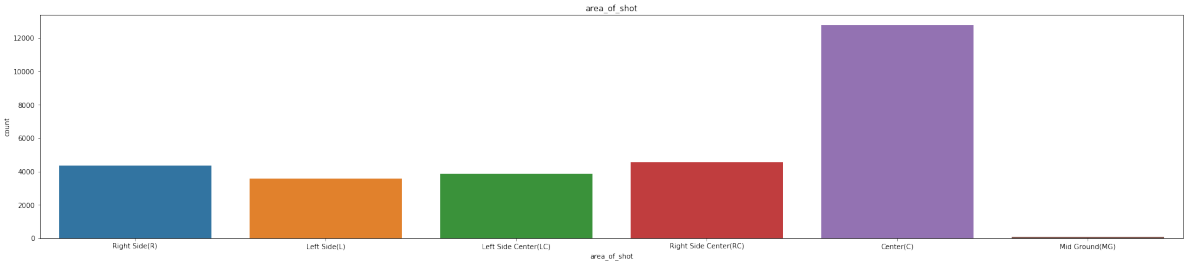
In [17]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["distance_of_shot"])
plt.title('distance_of_shot')
plt.show()
```



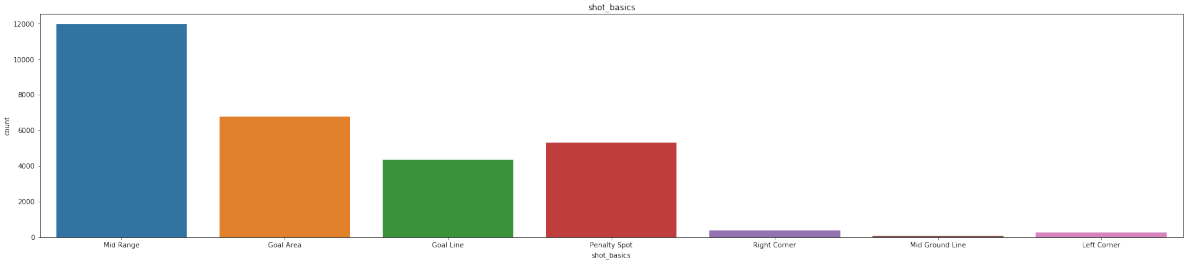
In [18]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["is_goal"])
plt.title('is_goal')
plt.show()
```



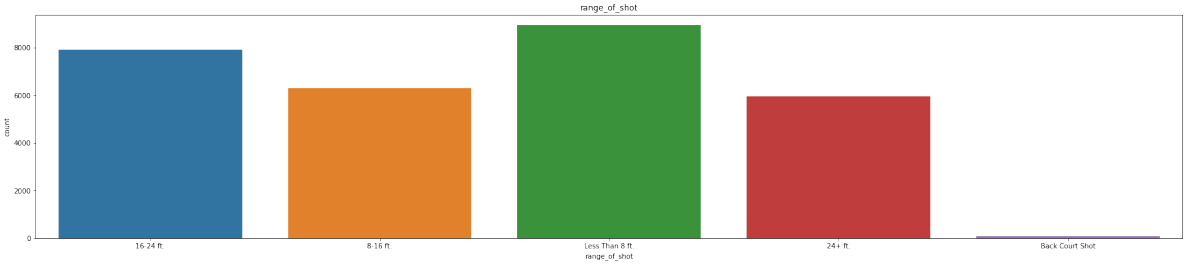
In [19]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["area_of_shot"])
plt.title('area_of_shot')
plt.show()
```



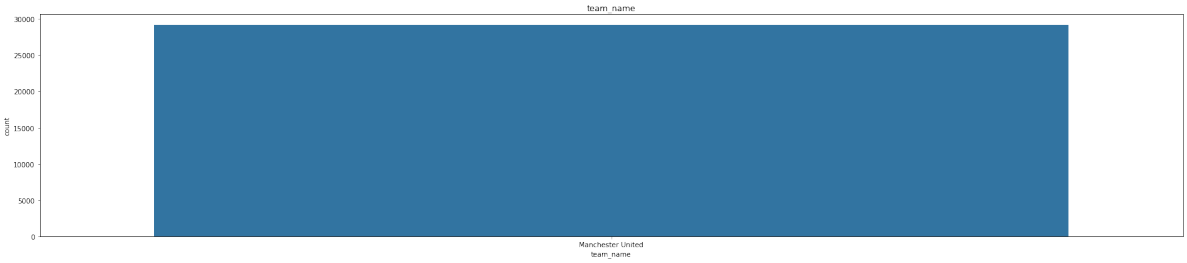
In [20]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["shot_basics"])
plt.title('shot_basics')
plt.show()
```



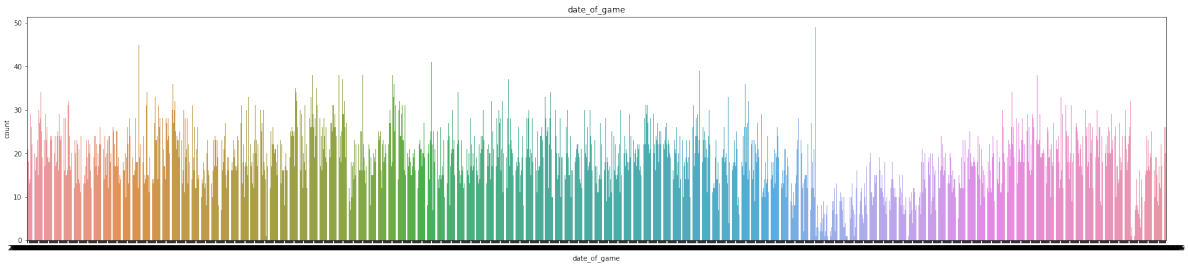
In [21]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["range_of_shot"])
plt.title('range_of_shot')
plt.show()
```



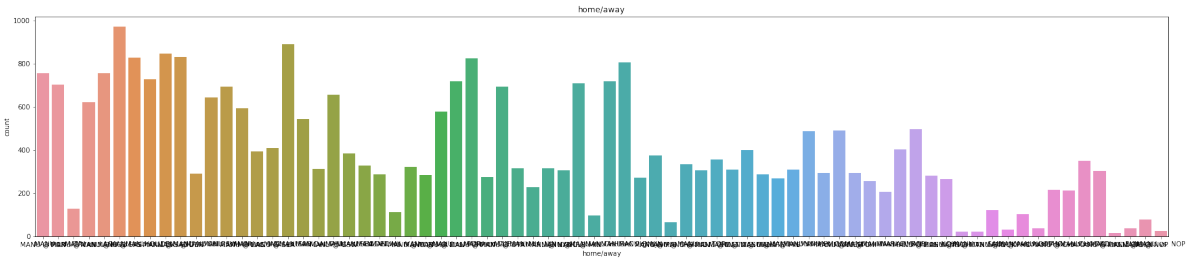
In [22]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["team_name"])
plt.title('team_name')
plt.show()
```



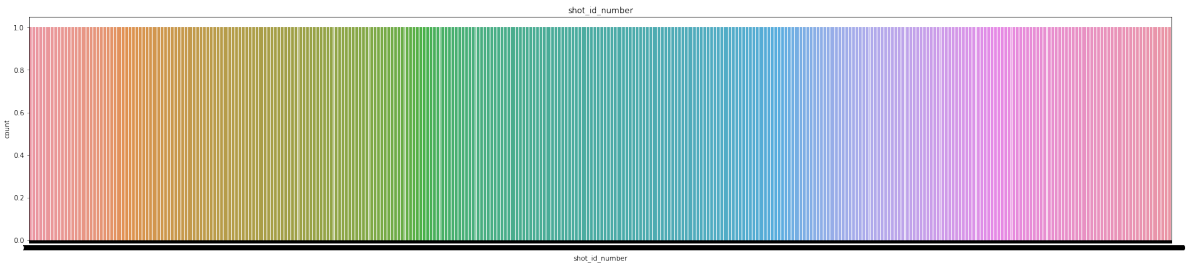
In [23]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["date_of_game"])
plt.title('date_of_game')
plt.show()
```



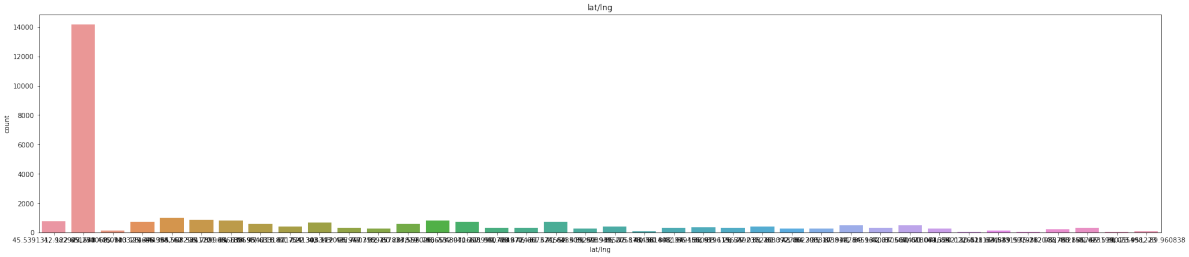
In [24]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["home/away"])
plt.title('home/away')
plt.show()
```



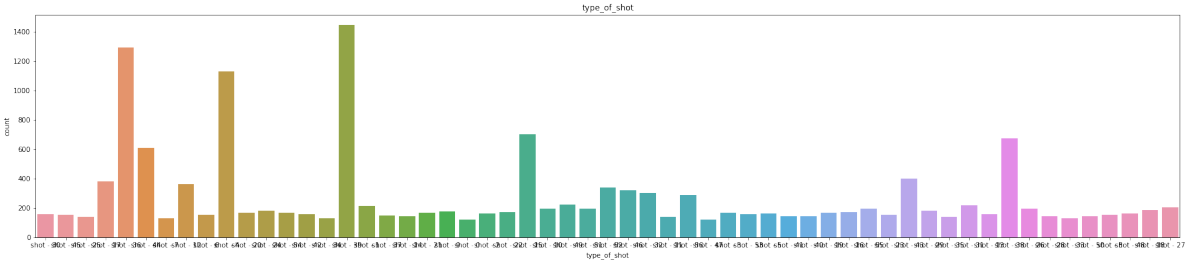
In [25]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["shot_id_number"])
plt.title('shot_id_number')
plt.show()
```



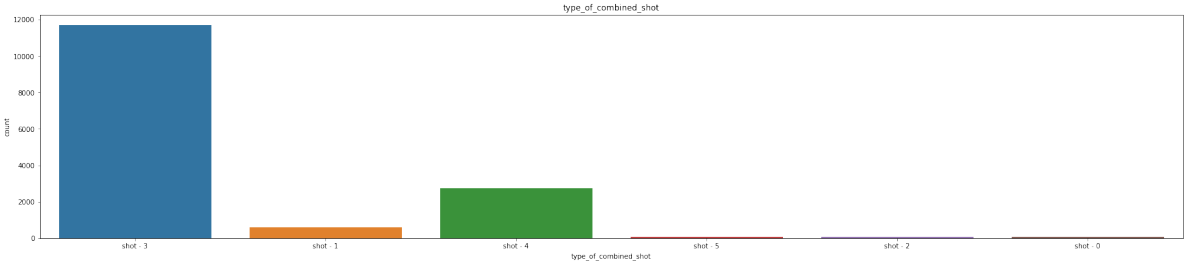
In [26]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["lat/lng"])
plt.title('lat/lng')
plt.show()
```



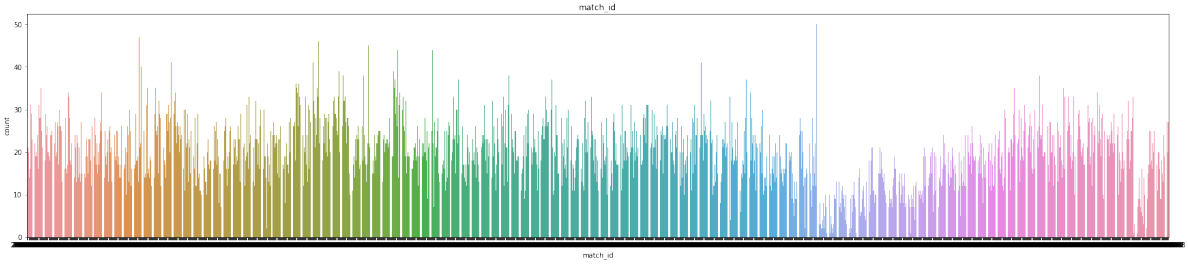
In [27]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["type_of_shot"])
plt.title('type_of_shot')
plt.show()
```



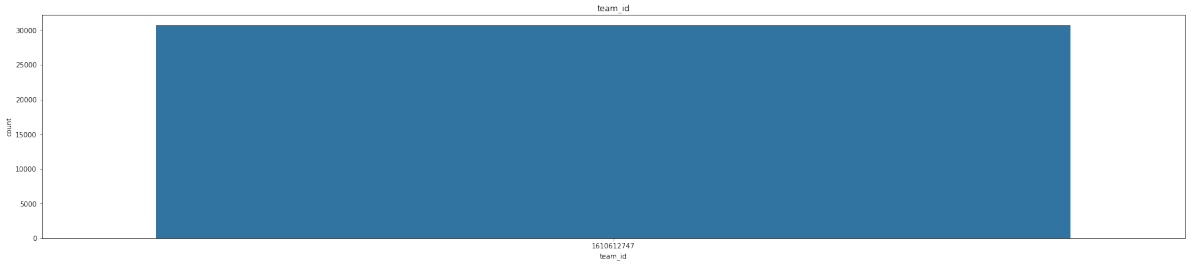
In [28]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["type_of_combined_shot"])
plt.title('type_of_combined_shot')
plt.show()
```



In [29]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["match_id"])
plt.title('match_id')
plt.show()
```



In [30]:
```
plt.figure(figsize=(30, 6))
sns.countplot(Train["team_id"])
plt.title('team_id')
plt.show()
```

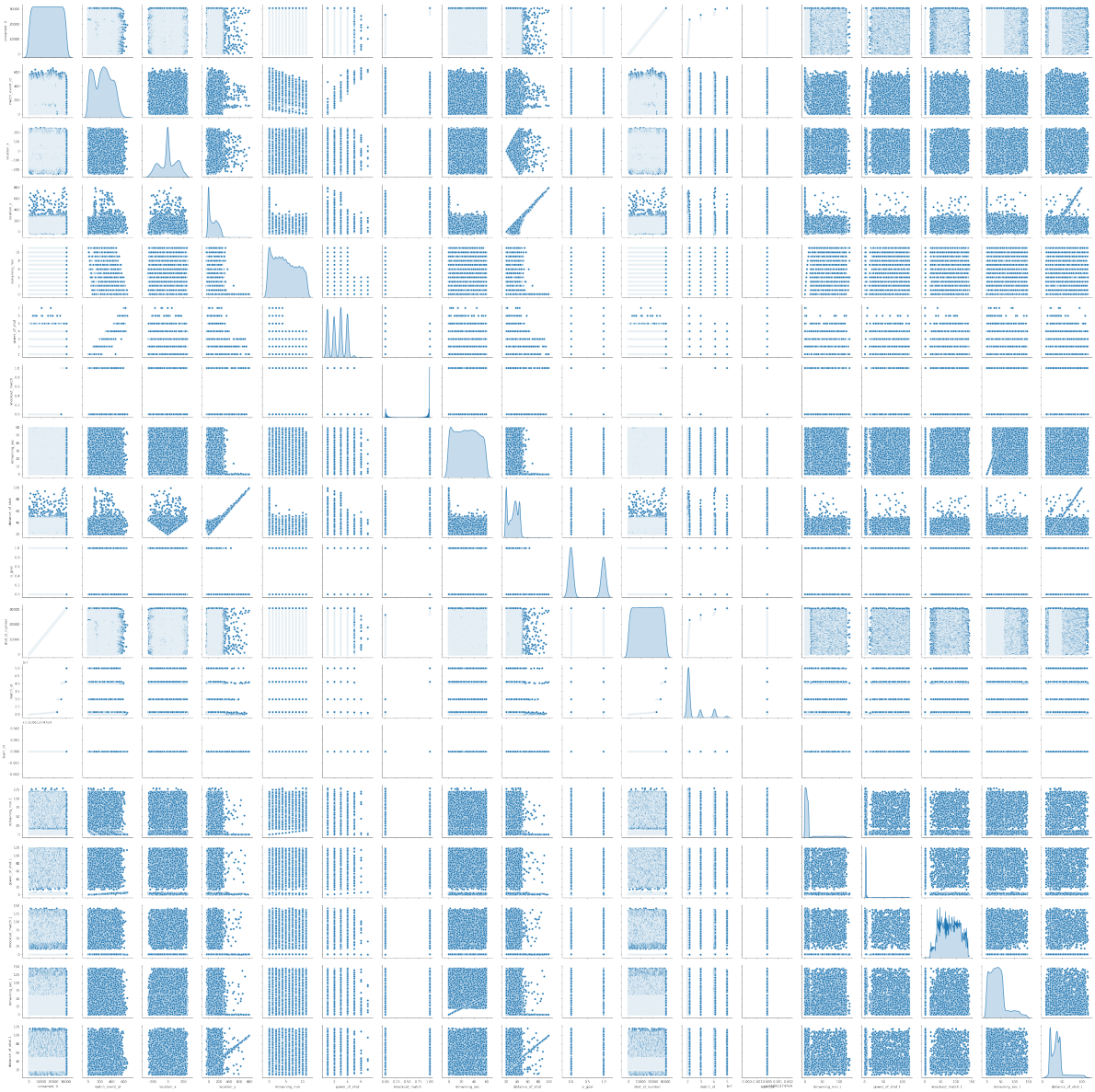


Bivariate Analysis

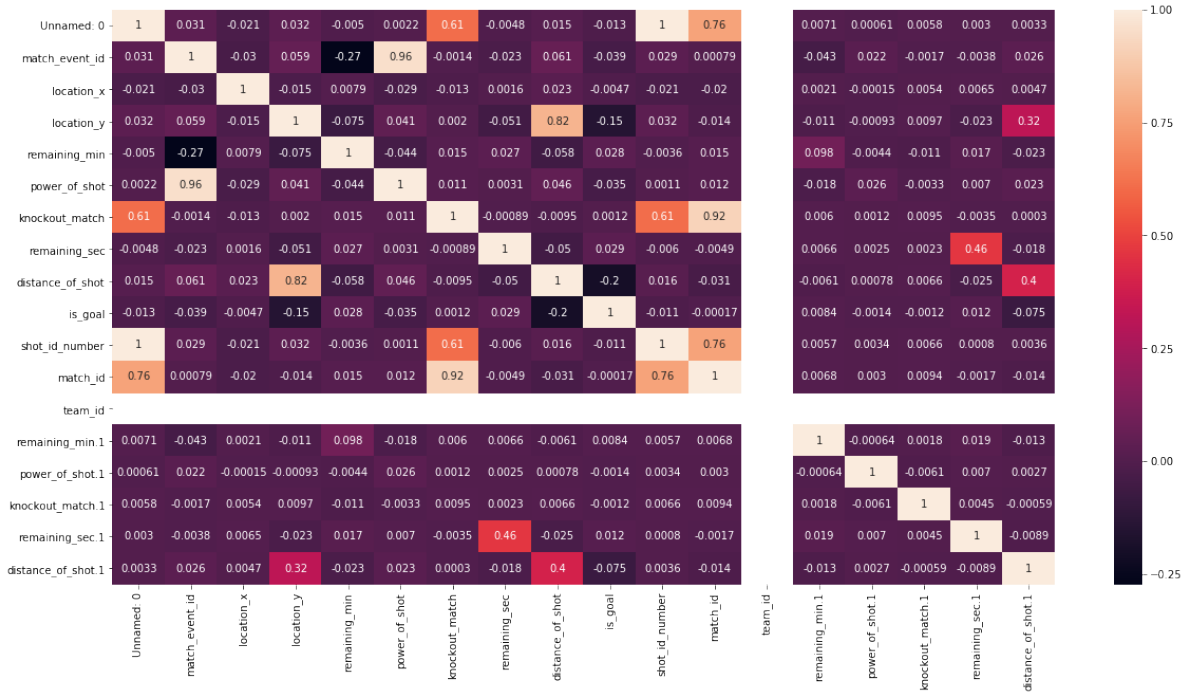
In [31]:
```
plt.figure(figsize=(30,30))

sns.pairplot(Train, diag_kind='kde');
```

```
/Users/shubham/anaconda3/lib/python3.7/site-packages/scipy/stats/s
tats.py:1713: FutureWarning: Using a non-tuple sequence for multid
imensional indexing is deprecated; use `arr[tuple(seq)]` instead o
f `arr[seq]`. In the future this will be interpreted as an array i
ndex, `arr[np.array(seq)]`, which will result either in an error o
r a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sum
val
/Users/shubham/anaconda3/lib/python3.7/site-packages/statsmodels/n
onparametric/kde.py:448: RuntimeWarning: invalid value encountered
in greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for
two columns.
/Users/shubham/anaconda3/lib/python3.7/site-packages/statsmodels/n
onparametric/kde.py:448: RuntimeWarning: invalid value encountered
in less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for
two columns.
/Users/shubham/anaconda3/lib/python3.7/site-packages/statsmodels/n
onparametric/kde.py:488: RuntimeWarning: invalid value encountered
in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/Users/shubham/anaconda3/lib/python3.7/site-packages/statsmodels/n
onparametric/kdetools.py:34: RuntimeWarning: invalid value encount
ered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
/Users/shubham/anaconda3/lib/python3.7/site-packages/numpy/core/fr
omnumeric.py:83: RuntimeWarning: invalid value encountered in redu
ce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

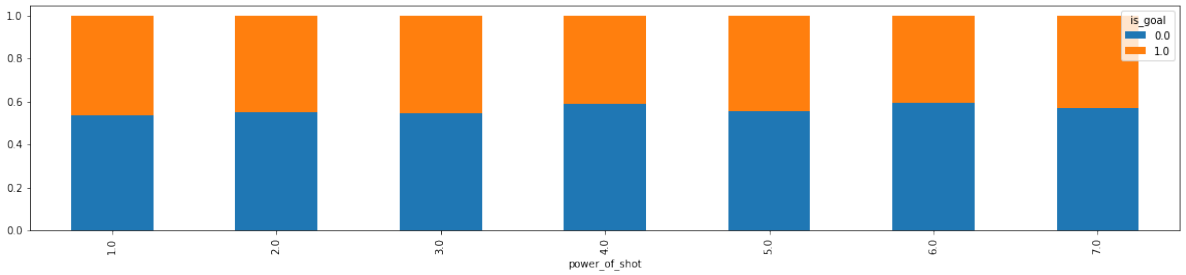
<Figure size 2160x2160 with 0 Axes>



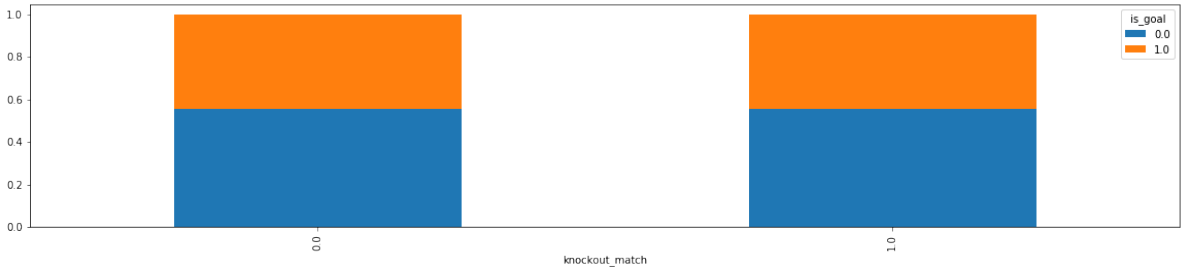
In [32]:
```
#correlation matrix
plt.figure(figsize=(20, 10))

vg_corr = Train.corr()
sns.heatmap(vg_corr,
            xticklabels = vg_corr.columns.values,
            yticklabels = vg_corr.columns.values,
            annot = True);
```



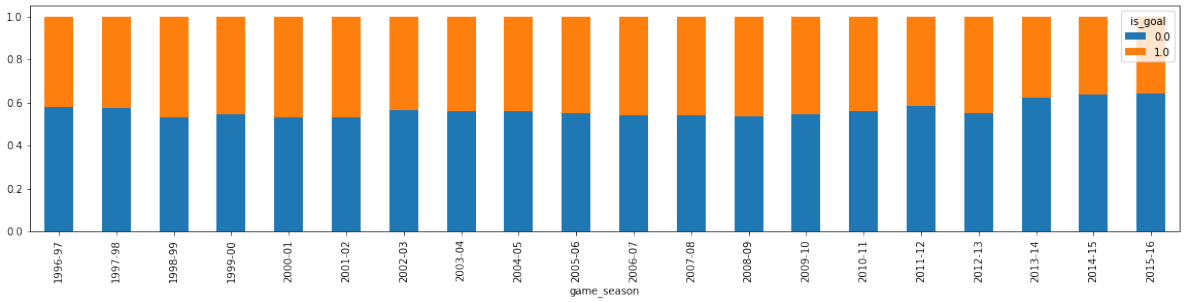
In [33]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['power_of_shot'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



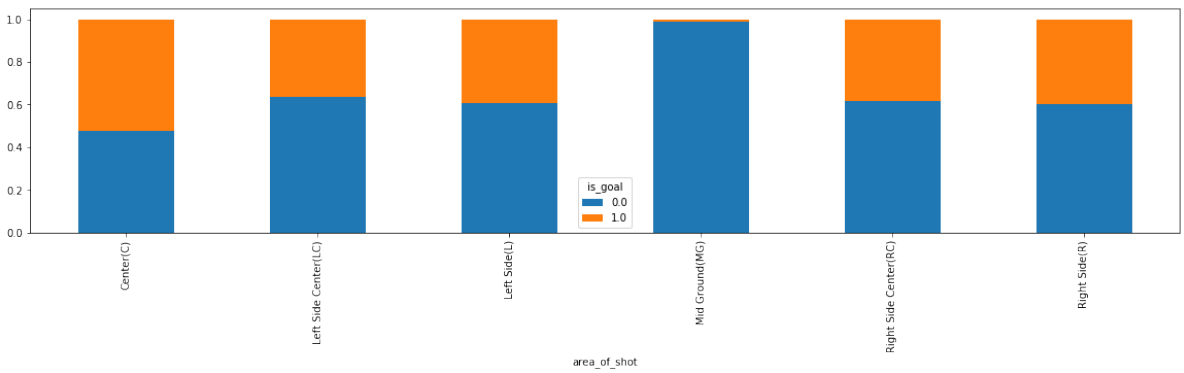
In [34]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['knockout_match'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



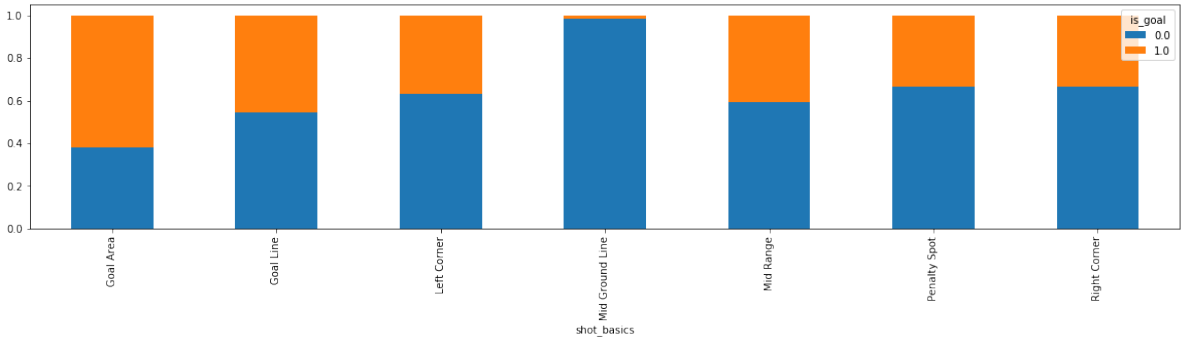
In [35]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['game_season'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



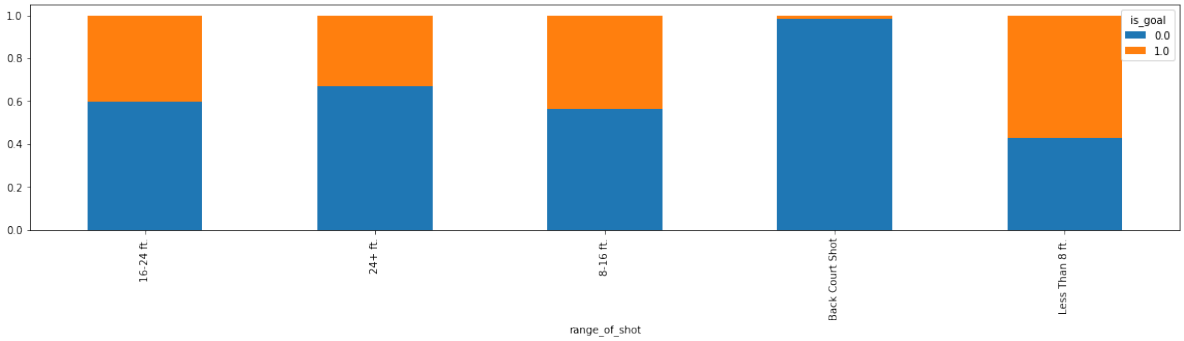
In [36]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['area_of_shot'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



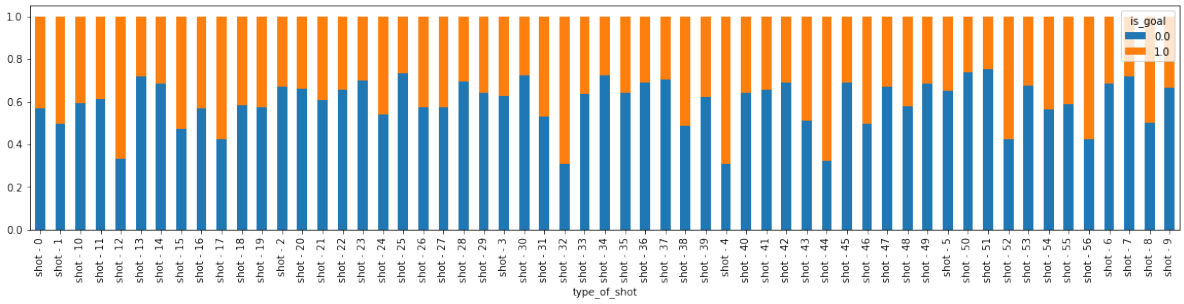
In [37]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['shot_basics'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



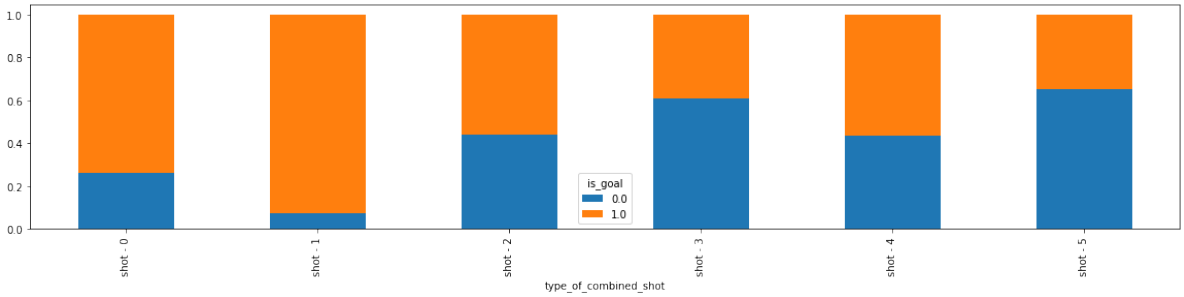
In [38]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['range_of_shot'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



In [39]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['type_of_shot'],Train['is_goal'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



In [40]:
```
# categorical vs Categorical
#Train.plot.bar(stacked=True)
Married=pd.crosstab(Train['type_of_combined_shot'],Train['is_goal'])
)
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(20,4))
plt.show()
```



In [41]:
```
# Renaming Columns which are repeating with Different Names for ana
lyzation
Train.rename(columns={'remaining_min.1': 'remaining_minONE', 'power
_of_shot.1': 'power_of_shotONE' ,'knockout_match.1': 'knockout_matc
hONE','remaining_sec.1': 'remaining_secONE', 'distance_of_shot.1':'
distance_of_shotONE'}, inplace=True)
```

In [42]:
```
Train.describe()
```

Out[42]:

	Unnamed: 0	match_event_id	location_x	location_y	remaining_min	power_o
count	30697.000000	29134.000000	29236.000000	29157.000000	29135.000000	29211.0
mean	15348.000000	249.576028	7.383876	91.126933	4.883233	2.5
std	8861.604943	150.186019	110.263049	87.676395	3.452533	1.1
min	0.000000	2.000000	-250.000000	-44.000000	0.000000	1.0
25%	7674.000000	111.000000	-68.000000	4.000000	2.000000	1.0
50%	15348.000000	254.000000	0.000000	74.000000	5.000000	3.0
75%	23022.000000	369.000000	95.000000	160.000000	8.000000	3.0
max	30696.000000	659.000000	248.000000	791.000000	11.000000	7.0

In [43]:
```
# Function to preprocess home/away data
def preprocess_home_away(x):
    try:
        t = x.split(' ')
    except:
        return None
    return t[0]+' vs. '+t[2]
```

In [44]:
```
Train['home/away'] = Train['home/away'].apply(lambda x: preprocess_
home_away(x))
```

Missing values Handling

In [45]:
```
Train.shape
```

Out[45]: (30697, 28)

In [46]:
```
for i in range(len(Train)):
    if(type(Train.loc[i ,'type_of_combined_shot']) == float):
        Train.loc[i ,'type_of_combined_shot'] = 'shot - ' + str(int
(int(Train.loc[i ,'type_of_shot'].split('-')[1])/10))
```

In [47]:
```
for i in range(len(Train)):
    if(type(Train.loc[i ,'type_of_shot']) == float):
        Train.loc[i ,'type_of_shot'] = 'shot - ' + str(int(Train.lo
c[i ,'type_of_combined_shot'].split('-')[1])*10+5)
```

In [48]:
```
# splitting the numerical feature i.e. 45 from shot-45
Train['type_of_shot'] = Train['type_of_shot'].apply(lambda x : int(
x.split('-')[1]))
```

In [49]:
```
for i in range(len(Train)):
    if(np.isnan(Train.loc[5 ,'power_of_shot'])):
        Train.loc[i ,'power_of_shot'] = int(Train.loc[i ,'power_of_
shotONE']/10)
```

In [50]:
```
Train['power_of_shot'].fillna(Train['power_of_shot'].mode()[0], inp
lace=True)
```

In [51]:
```
for i in range(len(Train)):
    if(np.isnan(Train.loc[5 ,'remaining_min'])):
        Train.loc[i ,'remaining_min'] = int(Train.loc[i ,'remaining
_minONE']/10)
```

In [52]:
```
Train['remaining_min'].fillna(Train['remaining_min'].mode()[0], inp
lace=True)
```

In [53]:
```
# filling knockout_match
```

In [54]:
```
lis_knockout = Train.index.tolist()# put index in a list
for i in range(len(lis_knockout)):
    if(np.isnan(Train.loc[lis_knockout[i],'knockout_match'])):
        Train.loc[lis_knockout[i],'knockout_match']= Train.loc[lis_
knockout[i], 'knockout_matchONE']
```

In [55]:
```
# now filling remaining missing values with median/mode
Train['knockout_match'].fillna(Train['knockout_match'].mode()[0], i
nplace=True)
```

In [56]:
```
# distance_of_shot
lis_knockout = Train.index.tolist()# put index in a list
for i in range(len(lis_knockout)):
    if(np.isnan(Train.loc[lis_knockout[i],'distance_of_shot'])):
        Train.loc[lis_knockout[i],'distance_of_shot']= round(Train.
loc[lis_knockout[i], 'distance_of_shotONE'])
```

In [57]:
```
Train['distance_of_shot'].fillna(Train['distance_of_shot'].median()
, inplace=True)
```

In [58]:
```
# filling  remaining_sec
```

In [59]:
```
lis_knockout = Train.index.tolist()# put index in a list
for i in range(len(lis_knockout)):
    if(np.isnan(Train.loc[lis_knockout[i],'remaining_sec'])):
        Train.loc[lis_knockout[i],'remaining_sec']= Train.loc[lis_k
nockout[i], 'remaining_secONE']
```

In [60]:
```
Train['remaining_sec'].fillna(Train['remaining_sec'].median(), inpl
ace=True)
```

In [61]:
```
Train['location_x'].fillna(Train['location_x'].mean(), inplace=True
)
```

In [62]:
```
Train['home/away'].fillna(Train['home/away'].mode()[0], inplace=Tru
e)
```

In [63]:
```
Train['area_of_shot'].fillna(Train['area_of_shot'].mode()[0], inpla
ce=True)
```

In [64]:
```
Train['team_name'].fillna(Train['team_name'].mode()[0], inplace=Tru
e)
```

In [65]:
```
Train['location_y'].fillna(Train['location_y'].median(), inplace=Tr
ue)
```

In [66]:
```
Train[['year','month','date']] = Train.date_of_game.str.split('-',
expand=True)

Train['year'].fillna(Train['year'].mode()[0], inplace=True)
Train['month'].fillna(Train['month'].mode()[0], inplace=True)
Train['date'].fillna(Train['date'].mode()[0], inplace=True)
```

In [67]:
```
Train['match_event_id'].fillna(Train['match_event_id'].median(), in
place=True)
```

In [68]:
```
Train['range_of_shot'].fillna(Train['range_of_shot'].mode()[0], inp
lace=True)
```

In [69]:
```
Train['lat/lng'].fillna(Train['lat/lng'].mode()[0], inplace=True)
```

In [70]:
```
Train['shot_basics'].fillna(Train['shot_basics'].mode()[0], inplace
=True)
```

In [71]:
```
Train['game_season'].fillna(Train['game_season'].mode()[0], inplace
=True)
```

In [72]:
```
Train=Train.drop(['remaining_minONE','power_of_shotONE','knockout_m
atchONE','remaining_secONE','distance_of_shotONE'], axis=1)
```

In [73]:
```
Train=Train.drop(['date_of_game'], axis=1)
```

In [74]:
```
# checking missing data percentage in train data
total = Train.isnull().sum().sort_values(ascending = False)
percent = (Train.isnull().sum()/Train.isnull().count()*100).sort_va
lues(ascending = False)
missing_TrainData  = pd.concat([total, percent], axis=1, keys=['Tot
al', 'Percent'])
missing_TrainData.head(30)
```

Out[74]:

	Total	Percent
is_goal	6268	20.418933
shot_id_number	1563	5.091703
date	0	0.000000
area_of_shot	0	0.000000
match_event_id	0	0.000000
location_x	0	0.000000
location_y	0	0.000000
remaining_min	0	0.000000
power_of_shot	0	0.000000
knockout_match	0	0.000000
game_season	0	0.000000
remaining_sec	0	0.000000
distance_of_shot	0	0.000000
shot_basics	0	0.000000
month	0	0.000000
range_of_shot	0	0.000000
team_name	0	0.000000
home/away	0	0.000000
lat/lng	0	0.000000
type_of_shot	0	0.000000
type_of_combined_shot	0	0.000000
match_id	0	0.000000
team_id	0	0.000000
year	0	0.000000
Unnamed: 0	0	0.000000

In [75]:
```
Train.head()
```

Out[75]:

	Unnamed: 0	match_event_id	location_x	location_y	remaining_min	power_of_shot	knock
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	0.0	2.0	

5 rows × 25 columns

In [76]:
```
Train.columns
```

Out[76]:
```
Index(['Unnamed: 0', 'match_event_id', 'location_x', 'location_y',
       'remaining_min', 'power_of_shot', 'knockout_match', 'game_s
eason',
       'remaining_sec', 'distance_of_shot', 'is_goal', 'area_of_sh
ot',
       'shot_basics', 'range_of_shot', 'team_name', 'home/away',
       'shot_id_number', 'lat/lng', 'type_of_shot', 'type_of_combi
ned_shot',
       'match_id', 'team_id', 'year', 'month', 'date'],
      dtype='object')
```

In [77]:
```
Train.shape
```

Out[77]:
```
(30697, 25)
```

In [78]:
```
# Delete columns at index 1 & 2
Train = Train.drop([Train.columns[0]] ,  axis='columns')
```

In [79]:
```
Train.head()
```

Out[79]:

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

5 rows × 24 columns

In [80]:
```
# drop team_name and team_id as they are same throughout individual
ly.
Train = Train.drop(['team_name','team_id'] ,  axis=1)
```

In [81]:
```
Train.columns
```

Out[81]:
```
Index(['match_event_id', 'location_x', 'location_y', 'remaining_mi
n',
       'power_of_shot', 'knockout_match', 'game_season', 'remainin
g_sec',
       'distance_of_shot', 'is_goal', 'area_of_shot', 'shot_basics
',
       'range_of_shot', 'home/away', 'shot_id_number', 'lat/lng',
       'type_of_shot', 'type_of_combined_shot', 'match_id', 'year'
, 'month',
       'date'],
      dtype='object')
```

In [82]:
```
# Label Encoding game_season, area_of_shot, shot_basics, range_of_s
hot, home/away, lat/lng, type_of_combined_shot
# respectively
labelencoder = LabelEncoder()

labelencoder.fit(Train.iloc[:,6].values)
Train.iloc[:,6]=labelencoder.transform(Train.iloc[:,6])


labelencoder.fit(Train.iloc[:,10].values)
Train.iloc[:,10]=labelencoder.transform(Train.iloc[:,10])

labelencoder.fit(Train.iloc[:,11].values)
Train.iloc[:,11]=labelencoder.transform(Train.iloc[:,11])

labelencoder.fit(Train.iloc[:,12].values)
Train.iloc[:,12]=labelencoder.transform(Train.iloc[:,12])

labelencoder.fit(Train.iloc[:,13].values)
Train.iloc[:,13]=labelencoder.transform(Train.iloc[:,13])

labelencoder.fit(Train.iloc[:,15].values)
Train.iloc[:,15]=labelencoder.transform(Train.iloc[:,15])

labelencoder.fit(Train.iloc[:,17].values)
Train.iloc[:,17]=labelencoder.transform(Train.iloc[:,17])
```

In [83]:
```
Train.head()
```

Out[83]:

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

5 rows × 22 columns

In [84]:
```
Train.shape
```

Out[84]:
```
(30697, 22)
```

In [85]:
```
# Filling shot_id_number with previous value + 1 (Series of Natural Numbers)
lis = Train[Train['shot_id_number'].isnull()].index.tolist()
for i in range(len(lis)):
    Train.loc[lis[i],'shot_id_number']= Train.loc[lis[i]-1, 'shot_id_number']+1
```

In [86]:
```
# checking missing data percentage in train data
total = Train.isnull().sum().sort_values(ascending = False)
percent = (Train.isnull().sum()/Train.isnull().count()*100).sort_values(ascending = False)
missing_TrainData  = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_TrainData.head(30)
```

Out[86]:

	Total	Percent
is_goal	6268	20.418933
date	0	0.000000
month	0	0.000000
location_x	0	0.000000
location_y	0	0.000000
remaining_min	0	0.000000
power_of_shot	0	0.000000
knockout_match	0	0.000000
game_season	0	0.000000
remaining_sec	0	0.000000
distance_of_shot	0	0.000000
area_of_shot	0	0.000000
shot_basics	0	0.000000
range_of_shot	0	0.000000
home/away	0	0.000000
shot_id_number	0	0.000000
lat/lng	0	0.000000
type_of_shot	0	0.000000
type_of_combined_shot	0	0.000000
match_id	0	0.000000
year	0	0.000000
match_event_id	0	0.000000

In [87]:
```
lis2 = Train[Train['is_goal'].isnull()].index.tolist()
#for i in range(len(lis2)):
#    Test1=Train.loc[lis2[i],:]
```

In [88]:
```
#Test1.count()
```

'match_event_id','location_x','location_y','remaining_min','power_of_shot','knockout_match',
'game_season','remaining_sec','distance_of_shot','area_of_shot','shot_basics','range_of_shot','team_nam
'home/away','shot_id_number','lat/lng','type_of_shot','type_of_combined_shot','match_id', 'team_id'

In [89]:
```
df3=Train
```

In [90]:
```
df4=df3.iloc[lis2,:]
```

In [91]:
```
df4.head()# contains rowws where is_goal is missing
```

Out[91]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match
0	10.0	167.0	72.0	10.0	1.0	0.0
7	254.0	1.0	28.0	8.0	3.0	0.0
16	100.0	0.0	0.0	0.0	1.0	0.0
19	249.0	0.0	0.0	10.0	3.0	0.0
21	265.0	134.0	127.0	9.0	3.0	0.0

5 rows × 22 columns

In [92]:
```
df4.to_csv('Test.csv')
```

In [93]:
```
TestDataPath = 'Test.csv'
# Loading the Training and Test Dataset
Test = pd.read_csv(TestDataPath)
Test_original=Test.copy()
```

In [94]:
```
Test.head()
```

Out[94]:

	Unnamed: 0	match_event_id	location_x	location_y	remaining_min	power_of_shot	knock
0	0	10.0	167.0	72.0	10.0	1.0	
1	7	254.0	1.0	28.0	8.0	3.0	
2	16	100.0	0.0	0.0	0.0	1.0	
3	19	249.0	0.0	0.0	10.0	3.0	
4	21	265.0	134.0	127.0	9.0	3.0	

5 rows × 23 columns

In [95]:
```
# Delete columns at index 1 & 2
Test = Test.drop([Test.columns[0]] ,  axis='columns')
```

In [96]:
```
# checking missing data percentage in train data
total = Test.isnull().sum().sort_values(ascending = False)
percent = (Test.isnull().sum()/Test.isnull().count()*100).sort_valu
es(ascending = False)
missing_TrainData  = pd.concat([total, percent], axis=1, keys=['Tot
al', 'Percent'])
missing_TrainData.head(30)
```

Out[96]:

	Total	Percent
is_goal	6268	100.0
date	0	0.0
month	0	0.0
location_x	0	0.0
location_y	0	0.0
remaining_min	0	0.0
power_of_shot	0	0.0
knockout_match	0	0.0
game_season	0	0.0
remaining_sec	0	0.0
distance_of_shot	0	0.0
area_of_shot	0	0.0
shot_basics	0	0.0
range_of_shot	0	0.0
home/away	0	0.0
shot_id_number	0	0.0
lat/lng	0	0.0
type_of_shot	0	0.0
type_of_combined_shot	0	0.0
match_id	0	0.0
year	0	0.0
match_event_id	0	0.0

In [97]:
```
X=Train
#Y=Train
```

In [98]:
```
X.head()
```

Out[98]:

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

5 rows × 22 columns

In [99]:
```
X.columns
```

Out[99]:
```
Index(['match_event_id', 'location_x', 'location_y', 'remaining_mi
n',
       'power_of_shot', 'knockout_match', 'game_season', 'remainin
g_sec',
       'distance_of_shot', 'is_goal', 'area_of_shot', 'shot_basics
',
       'range_of_shot', 'home/away', 'shot_id_number', 'lat/lng',
       'type_of_shot', 'type_of_combined_shot', 'match_id', 'year'
, 'month',
       'date'],
      dtype='object')
```

In [100]:
```
#X['date_of_game'].dt.strftime("%Y%m%d").astype(int)
#X['date_of_game']
#X['date_of_game'].str.replace("-","")
#X['date_of_game'] = pd.to_numeric(X.date_of_game.str.replace('-','
'))
#Test['date_of_game'] = pd.to_numeric(Test.date_of_game.str.replace
('-',''))
```

In [101]:
```
X.head()
```

Out[101]:

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

5 rows × 22 columns

In [102]:
```
#X=pd.DataFrame(X)
#X=X.drop(['game_season'], axis=1)
#Test=Test.drop(['game_season'], axis=1)
```

In [103]:
```
X.head()
```

Out[103]:

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

5 rows × 22 columns

In [104]:
```
Train.head()
```

Out[104]:

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

5 rows × 22 columns

In [105]:
```
# checking missing data percentage in train data
total = X.isnull().sum().sort_values(ascending = False)
percent = (X.isnull().sum()/X.isnull().count()*100).sort_values(asc
ending = False)
missing_TrainData  = pd.concat([total, percent], axis=1, keys=['Tot
al', 'Percent'])
missing_TrainData.head(30)
```

Out[105]:

	Total	Percent
is_goal	6268	20.418933
date	0	0.000000
month	0	0.000000
location_x	0	0.000000
location_y	0	0.000000
remaining_min	0	0.000000
power_of_shot	0	0.000000
knockout_match	0	0.000000
game_season	0	0.000000
remaining_sec	0	0.000000
distance_of_shot	0	0.000000
area_of_shot	0	0.000000
shot_basics	0	0.000000
range_of_shot	0	0.000000
home/away	0	0.000000
shot_id_number	0	0.000000
lat/lng	0	0.000000
type_of_shot	0	0.000000
type_of_combined_shot	0	0.000000
match_id	0	0.000000
year	0	0.000000
match_event_id	0	0.000000

In [106]:
```
# dropping is_goal in X i.e Train dataset wherever it is null
X_demo=X
```

Dropping the rows where there is not NULL Value or missing value

In [107]:
```
import numpy as np

X_demo = X_demo[np.isfinite(X_demo['is_goal'])]
#Test = Test[np.isfinite(Test['is_goal'])]
```

In [108]:
```
Test.head()
```

Out[108]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match
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

In [109]:
```
X_demo.head()
```

Out[109]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match
1	12.0	-157.0	0.0	10.0	1.0	0.0
2	35.0	-101.0	135.0	7.0	1.0	0.0
3	43.0	138.0	175.0	6.0	1.0	0.0
4	155.0	0.0	0.0	0.0	2.0	0.0
5	244.0	-145.0	-11.0	9.0	3.0	0.0

5 rows × 22 columns

In [110]:
```
# checking missing data percentage in train data
total = X_demo.isnull().sum().sort_values(ascending = False)
percent = (X_demo.isnull().sum()/X_demo.isnull().count()*100).sort_
values(ascending = False)
missing_TrainData  = pd.concat([total, percent], axis=1, keys=['Tot
al', 'Percent'])
missing_TrainData.head(30)
```

Out[110]:

	Total	Percent
date	0	0.0
month	0	0.0
location_x	0	0.0
location_y	0	0.0
remaining_min	0	0.0
power_of_shot	0	0.0
knockout_match	0	0.0
game_season	0	0.0
remaining_sec	0	0.0
distance_of_shot	0	0.0
is_goal	0	0.0
area_of_shot	0	0.0
shot_basics	0	0.0
range_of_shot	0	0.0
home/away	0	0.0
shot_id_number	0	0.0
lat/lng	0	0.0
type_of_shot	0	0.0
type_of_combined_shot	0	0.0
match_id	0	0.0
year	0	0.0
match_event_id	0	0.0

In [111]:
```
X=X_demo
```

In [112]:
```
X.to_csv('v1.csv')
```

In [113]:
```
X.head()
```

Out[113]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match
1	12.0	-157.0	0.0	10.0	1.0	0.0
2	35.0	-101.0	135.0	7.0	1.0	0.0
3	43.0	138.0	175.0	6.0	1.0	0.0
4	155.0	0.0	0.0	0.0	2.0	0.0
5	244.0	-145.0	-11.0	9.0	3.0	0.0

5 rows × 22 columns

Making y DataFrame for Model Feeding

In [114]:
```
y=pd.DataFrame(X['is_goal'])
```

In [115]:
```
y.head()
```

Out[115]:

	is_goal
1	0.0
2	1.0
3	0.0
4	1.0
5	0.0

In [116]:
```
y.shape
```

Out[116]: (24429, 1)

In [117]:
```
X.head()
```

Out[117]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match
1	12.0	-157.0	0.0	10.0	1.0	0.0
2	35.0	-101.0	135.0	7.0	1.0	0.0
3	43.0	138.0	175.0	6.0	1.0	0.0
4	155.0	0.0	0.0	0.0	2.0	0.0
5	244.0	-145.0	-11.0	9.0	3.0	0.0

5 rows × 22 columns

In [118]:
```
X['is_goal'].head()
```

Out[118]:
```
1    0.0
2    1.0
3    0.0
4    1.0
5    0.0
Name: is_goal, dtype: float64
```

In [119]:
```
# Drop is_goal from both datasets

X=X.drop(['is_goal'],axis=1)
Test=Test.drop(['is_goal'],axis=1)
```

In [120]:
```
X.head()
```

Out[120]:

	match_event_id	location_x	location_y	remaining_min	power_of_shot	knockout_match
1	12.0	-157.0	0.0	10.0	1.0	0.0
2	35.0	-101.0	135.0	7.0	1.0	0.0
3	43.0	138.0	175.0	6.0	1.0	0.0
4	155.0	0.0	0.0	0.0	2.0	0.0
5	244.0	-145.0	-11.0	9.0	3.0	0.0

5 rows × 21 columns

In [121]:
```
# Now i have X and y values for train teest split
```

In [122]:
```
X.columns
```

Out[122]:
```
Index(['match_event_id', 'location_x', 'location_y', 'remaining_mi
n',
       'power_of_shot', 'knockout_match', 'game_season', 'remainin
g_sec',
       'distance_of_shot', 'area_of_shot', 'shot_basics', 'range_o
f_shot',
       'home/away', 'shot_id_number', 'lat/lng', 'type_of_shot',
       'type_of_combined_shot', 'match_id', 'year', 'month', 'date
'],
      dtype='object')
```

In [123]:
```
Test.columns
```

Out[123]:
```
Index(['match_event_id', 'location_x', 'location_y', 'remaining_mi
n',
       'power_of_shot', 'knockout_match', 'game_season', 'remainin
g_sec',
       'distance_of_shot', 'area_of_shot', 'shot_basics', 'range_o
f_shot',
       'home/away', 'shot_id_number', 'lat/lng', 'type_of_shot',
       'type_of_combined_shot', 'match_id', 'year', 'month', 'date
'],
      dtype='object')
```

In [124]:
```
X.shape
```

Out[124]:
```
(24429, 21)
```

In [125]:
```
Test.shape
```

Out[125]:
```
(6268, 21)
```

In [126]:
```
X1=X
Test1=Test
```

In [127]:
```
X1.shape
```

Out[127]:
```
(24429, 21)
```

Prediction

In [128]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X1,y,test_size=0.2
,random_state=0)
```

In [129]:
```
from sklearn.preprocessing import StandardScaler
sc_X=StandardScaler()
X_train_ = sc_X.fit_transform(X_train)
X_test_ = sc_X.transform(X_test)
Test_ = sc_X.transform(Test)
```

```
/Users/shubham/anaconda3/lib/python3.7/site-packages/sklearn/prepr
ocessing/data.py:625: DataConversionWarning: Data with input dtype
int64, float64, object were all converted to float64 by StandardSc
aler.
  return self.partial_fit(X, y)
/Users/shubham/anaconda3/lib/python3.7/site-packages/sklearn/base.
py:462: DataConversionWarning: Data with input dtype int64, float6
4, object were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:4: DataConversionWarning: Data with input dtype int64, fl
oat64, object were all converted to float64 by StandardScaler.
  after removing the cwd from sys.path.
/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:5: DataConversionWarning: Data with input dtype int64, fl
oat64 were all converted to float64 by StandardScaler.
  """
```

In [130]:
```
Test.shape
```

Out[130]: (6268, 21)

In [131]:
```
X_train['year'] = X_train['year'].astype(int)
X_train['month'] = X_train['month'].astype(int)
X_train['date'] = X_train['date'].astype(int)
```

```
/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy

/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doin
g imports until
```

In [132]:
```
Test['year'] = Test['year'].astype(int)
Test['month'] = Test['month'].astype(int)
Test['date'] = Test['date'].astype(int)
```

In [133]:
```
X_test['year'] = X_test['year'].astype(int)
X_test['month'] = X_test['month'].astype(int)
X_test['date'] = X_test['date'].astype(int)
```

```
/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy

/Users/shubham/anaconda3/lib/python3.7/site-packages/ipykernel_lau
ncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doin
g imports until
```

In [134]:
```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(solver='lbfgs',random_state=0,C=10)
classifier.fit(X, y)
```

```
/Users/shubham/anaconda3/lib/python3.7/site-packages/sklearn/utils
/validation.py:761: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y t
o (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[134]:
```
LogisticRegression(C=10, class_weight=None, dual=False, fit_interc
ept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=0, solver='lbfgs
',
          tol=0.0001, verbose=0, warm_start=False)
```

Fitting Logistic Regression to the Training set

In [135]:
```
y_pred=classifier.predict_proba(X_test)
```

In [136]:
```
y_pred1=classifier.predict_proba(Test)
```

In [137]:
```
preds=y_pred1[:,1]
```

In [138]:
```
preds = np.round(preds,2)
```

Extracting corresponding shot_id_number from sample submission file as dataset size is different for sample_submission.csv and Test.csv

In [139]:
```
ans=pd.DataFrame(y_pred1)
```

In [140]:
```
ans.head()
```

Out[140]:

	0	1
0	0.540402	0.459598
1	0.540402	0.459598
2	0.540402	0.459598
3	0.540402	0.459598
4	0.540402	0.459598

In [141]:
```
# Delete columns at index 1 & 2
ans = ans.drop([ans.columns[0]] ,  axis='columns')
```

In [142]:
```
ans.head()
```

Out[142]:

	1
0	0.459598
1	0.459598
2	0.459598
3	0.459598
4	0.459598

In [143]:
```
ans.columns = ['is_goal']
```

In [144]:
```
ans.head()
```

Out[144]:

	is_goal
0	0.459598
1	0.459598
2	0.459598
3	0.459598
4	0.459598

In [145]:
```
# map with shot_id_number
```

In [146]:
```
G=Test_original
```

In [147]:
```
G.head()
```

Out[147]:

	Unnamed: 0	match_event_id	location_x	location_y	remaining_min	power_of_shot	knock
0	0	10.0	167.0	72.0	10.0	1.0	
1	7	254.0	1.0	28.0	8.0	3.0	
2	16	100.0	0.0	0.0	0.0	1.0	
3	19	249.0	0.0	0.0	10.0	3.0	
4	21	265.0	134.0	127.0	9.0	3.0	

5 rows × 23 columns

In [148]:
```
shot_id_list=G['shot_id_number'].to_list()
```

In [149]:
```
shot_id_list
```

Out[149]:
```
[1.0,
 8.0,
 17.0,
 20.0,
 22.0,
 33.0,
 34.0,
 35.0,
 36.0,
 37.0,
 38.0,
 45.0,
 50.0,
 55.0,
```

```
60.0,
62.0,
66.0,
67.0,
71.0,
72.0,
76.0,
80.0,
85.0,
86.0,
87.0,
92.0,
95.0,
97.0,
104.0,
113.0,
122.0,
123.0,
126.0,
133.0,
134.0,
141.0,
144.0,
150.0,
152.0,
153.0,
156.0,
159.0,
165.0,
172.0,
182.0,
192.0,
193.0,
196.0,
202.0,
205.0,
210.0,
211.0,
228.0,
250.0,
257.0,
260.0,
262.0,
264.0,
267.0,
279.0,
282.0,
293.0,
299.0,
300.0,
301.0,
317.0,
327.0,
```

```
328.0,
335.0,
346.0,
348.0,
360.0,
365.0,
379.0,
382.0,
390.0,
391.0,
392.0,
397.0,
417.0,
418.0,
420.0,
428.0,
429.0,
434.0,
439.0,
440.0,
451.0,
456.0,
459.0,
472.0,
480.0,
491.0,
497.0,
509.0,
511.0,
523.0,
533.0,
544.0,
546.0,
552.0,
555.0,
558.0,
560.0,
568.0,
571.0,
573.0,
576.0,
578.0,
591.0,
596.0,
597.0,
601.0,
605.0,
617.0,
625.0,
628.0,
634.0,
646.0,
649.0,
```

```
651.0,
654.0,
656.0,
657.0,
661.0,
669.0,
670.0,
680.0,
693.0,
694.0,
699.0,
711.0,
715.0,
718.0,
722.0,
723.0,
725.0,
734.0,
735.0,
737.0,
746.0,
747.0,
764.0,
766.0,
767.0,
771.0,
773.0,
774.0,
783.0,
784.0,
789.0,
792.0,
804.0,
809.0,
812.0,
813.0,
817.0,
818.0,
820.0,
825.0,
826.0,
829.0,
833.0,
855.0,
859.0,
862.0,
870.0,
876.0,
880.0,
886.0,
889.0,
891.0,
897.0,
```

```
902.0,
903.0,
908.0,
909.0,
920.0,
922.0,
924.0,
932.0,
934.0,
937.0,
938.0,
941.0,
943.0,
947.0,
950.0,
952.0,
954.0,
962.0,
963.0,
975.0,
979.0,
981.0,
983.0,
985.0,
989.0,
995.0,
998.0,
1008.0,
1019.0,
1023.0,
1026.0,
1029.0,
1034.0,
1036.0,
1038.0,
1039.0,
1044.0,
1049.0,
1051.0,
1052.0,
1062.0,
1064.0,
1066.0,
1087.0,
1092.0,
1094.0,
1098.0,
1099.0,
1102.0,
1103.0,
1118.0,
1131.0,
1133.0,
```

```
1135.0,
1138.0,
1139.0,
1141.0,
1142.0,
1151.0,
1152.0,
1155.0,
1157.0,
1165.0,
1167.0,
1172.0,
1174.0,
1178.0,
1180.0,
1200.0,
1202.0,
1209.0,
1227.0,
1228.0,
1229.0,
1237.0,
1241.0,
1248.0,
1256.0,
1277.0,
1279.0,
1288.0,
1290.0,
1293.0,
1295.0,
1308.0,
1315.0,
1319.0,
1320.0,
1321.0,
1322.0,
1331.0,
1334.0,
1343.0,
1348.0,
1349.0,
1351.0,
1373.0,
1375.0,
1380.0,
1382.0,
1385.0,
1387.0,
1394.0,
1399.0,
1403.0,
1409.0,
```

```
1413.0,
1425.0,
1428.0,
1434.0,
1444.0,
1448.0,
1461.0,
1466.0,
1476.0,
1478.0,
1485.0,
1491.0,
1494.0,
1495.0,
1500.0,
1501.0,
1503.0,
1507.0,
1511.0,
1519.0,
1524.0,
1528.0,
1529.0,
1533.0,
1538.0,
1539.0,
1545.0,
1546.0,
1550.0,
1555.0,
1556.0,
1558.0,
1559.0,
1561.0,
1563.0,
1565.0,
1574.0,
1581.0,
1583.0,
1591.0,
1594.0,
1605.0,
1611.0,
1614.0,
1617.0,
1620.0,
1626.0,
1628.0,
1630.0,
1632.0,
1634.0,
1643.0,
1651.0,
```

```
1654.0,
1655.0,
1658.0,
1659.0,
1665.0,
1675.0,
1680.0,
1690.0,
1696.0,
1714.0,
1718.0,
1721.0,
1726.0,
1728.0,
1734.0,
1735.0,
1740.0,
1744.0,
1747.0,
1752.0,
1763.0,
1764.0,
1765.0,
1769.0,
1770.0,
1785.0,
1787.0,
1788.0,
1793.0,
1794.0,
1800.0,
1812.0,
1825.0,
1826.0,
1834.0,
1843.0,
1844.0,
1848.0,
1849.0,
1860.0,
1863.0,
1875.0,
1883.0,
1890.0,
1892.0,
1893.0,
1897.0,
1905.0,
1908.0,
1931.0,
1940.0,
1943.0,
1947.0,
```

```
1965.0,
1967.0,
1968.0,
1972.0,
1982.0,
1985.0,
2005.0,
2010.0,
2021.0,
2024.0,
2027.0,
2028.0,
2035.0,
2036.0,
2043.0,
2047.0,
2049.0,
2055.0,
2065.0,
2066.0,
2070.0,
2077.0,
2081.0,
2085.0,
2089.0,
2095.0,
2097.0,
2100.0,
2108.0,
2112.0,
2113.0,
2120.0,
2121.0,
2122.0,
2139.0,
2141.0,
2144.0,
2146.0,
2148.0,
2149.0,
2157.0,
2158.0,
2159.0,
2168.0,
2176.0,
2178.0,
2181.0,
2184.0,
2187.0,
2188.0,
2190.0,
2193.0,
2194.0,
```

```
2197.0,
2202.0,
2204.0,
2211.0,
2214.0,
2215.0,
2217.0,
2222.0,
2223.0,
2226.0,
2230.0,
2233.0,
2240.0,
2246.0,
2247.0,
2250.0,
2254.0,
2257.0,
2259.0,
2262.0,
2263.0,
2266.0,
2305.0,
2310.0,
2313.0,
2319.0,
2320.0,
2341.0,
2345.0,
2350.0,
2356.0,
2360.0,
2370.0,
2374.0,
2384.0,
2386.0,
2397.0,
2398.0,
2411.0,
2414.0,
2435.0,
2439.0,
2446.0,
2458.0,
2465.0,
2466.0,
2470.0,
2490.0,
2495.0,
2501.0,
2507.0,
2533.0,
2537.0,
```

```
2538.0,
2541.0,
2542.0,
2544.0,
2554.0,
2555.0,
2565.0,
2572.0,
2573.0,
2575.0,
2586.0,
2588.0,
2594.0,
2617.0,
2618.0,
2620.0,
2623.0,
2631.0,
2640.0,
2642.0,
2647.0,
2663.0,
2664.0,
2676.0,
2680.0,
2681.0,
2690.0,
2691.0,
2693.0,
2695.0,
2704.0,
2708.0,
2709.0,
2717.0,
2718.0,
2722.0,
2728.0,
2732.0,
2743.0,
2745.0,
2752.0,
2756.0,
2757.0,
2761.0,
2769.0,
2771.0,
2773.0,
2774.0,
2776.0,
2781.0,
2793.0,
2799.0,
2800.0,
```

```
2803.0,
2811.0,
2818.0,
2824.0,
2828.0,
2829.0,
2830.0,
2842.0,
2847.0,
2848.0,
2868.0,
2870.0,
2885.0,
2886.0,
2889.0,
2902.0,
2903.0,
2904.0,
2918.0,
2919.0,
2922.0,
2923.0,
2931.0,
2945.0,
2946.0,
2955.0,
2957.0,
2960.0,
2963.0,
2966.0,
2972.0,
2973.0,
2974.0,
2981.0,
2982.0,
2984.0,
2998.0,
3016.0,
3021.0,
3023.0,
3025.0,
3026.0,
3029.0,
3034.0,
3037.0,
3038.0,
3040.0,
3051.0,
3062.0,
3069.0,
3071.0,
3076.0,
3079.0,
```

```
3081.0,
3082.0,
3084.0,
3091.0,
3095.0,
3098.0,
3104.0,
3105.0,
3108.0,
3114.0,
3118.0,
3122.0,
3123.0,
3132.0,
3138.0,
3141.0,
3156.0,
3162.0,
3167.0,
3170.0,
3171.0,
3172.0,
3176.0,
3180.0,
3189.0,
3193.0,
3198.0,
3203.0,
3205.0,
3206.0,
3215.0,
3217.0,
3222.0,
3228.0,
3229.0,
3232.0,
3233.0,
3235.0,
3238.0,
3256.0,
3264.0,
3265.0,
3273.0,
3275.0,
3285.0,
3289.0,
3294.0,
3306.0,
3311.0,
3315.0,
3318.0,
3322.0,
3324.0,
```

```
3327.0,
3334.0,
3341.0,
3345.0,
3346.0,
3352.0,
3368.0,
3369.0,
3371.0,
3372.0,
3375.0,
3384.0,
3386.0,
3387.0,
3393.0,
3406.0,
3411.0,
3412.0,
3414.0,
3417.0,
3420.0,
3423.0,
3436.0,
3437.0,
3438.0,
3442.0,
3443.0,
3446.0,
3451.0,
3461.0,
3473.0,
3477.0,
3489.0,
3491.0,
3510.0,
3515.0,
3527.0,
3530.0,
3533.0,
3540.0,
3545.0,
3553.0,
3569.0,
3582.0,
3604.0,
3605.0,
3609.0,
3616.0,
3624.0,
3628.0,
3630.0,
3641.0,
3642.0,
```

```
3648.0,
3650.0,
3651.0,
3652.0,
3660.0,
3664.0,
3670.0,
3672.0,
3680.0,
3682.0,
3685.0,
3686.0,
3689.0,
3691.0,
3696.0,
3698.0,
3705.0,
3712.0,
3713.0,
3719.0,
3721.0,
3723.0,
3724.0,
3725.0,
3727.0,
3733.0,
3734.0,
3736.0,
3742.0,
3761.0,
3765.0,
3770.0,
3774.0,
3777.0,
3778.0,
3780.0,
3781.0,
3783.0,
3784.0,
3788.0,
3790.0,
3794.0,
3795.0,
3796.0,
3801.0,
3802.0,
3806.0,
3808.0,
3813.0,
3814.0,
3818.0,
3824.0,
3836.0,
```

```
3853.0,
3855.0,
3858.0,
3861.0,
3866.0,
3869.0,
3870.0,
3874.0,
3876.0,
3878.0,
3884.0,
3890.0,
3900.0,
3904.0,
3906.0,
3907.0,
3908.0,
3912.0,
3913.0,
3914.0,
3918.0,
3923.0,
3924.0,
3925.0,
3927.0,
3955.0,
3964.0,
3966.0,
3969.0,
3974.0,
3976.0,
3984.0,
3991.0,
3993.0,
4009.0,
4011.0,
4016.0,
4020.0,
4024.0,
4033.0,
4035.0,
4041.0,
4045.0,
4052.0,
4055.0,
4056.0,
4057.0,
4067.0,
4069.0,
4077.0,
4079.0,
4094.0,
4095.0,
```

```
4096.0,
4100.0,
4119.0,
4129.0,
4130.0,
4131.0,
4132.0,
4134.0,
4135.0,
4136.0,
4137.0,
4138.0,
4143.0,
4145.0,
4148.0,
4156.0,
4163.0,
4166.0,
4169.0,
4170.0,
4174.0,
4180.0,
4190.0,
4195.0,
4199.0,
4201.0,
4203.0,
4210.0,
4211.0,
4213.0,
4225.0,
4235.0,
4236.0,
4242.0,
4244.0,
4245.0,
4250.0,
4252.0,
4254.0,
4263.0,
4264.0,
4268.0,
4270.0,
4274.0,
4282.0,
4283.0,
4291.0,
4304.0,
4306.0,
4309.0,
4310.0,
4313.0,
4314.0,
```

```
4315.0,
4321.0,
4324.0,
4325.0,
4328.0,
4329.0,
4332.0,
4337.0,
4342.0,
4344.0,
4352.0,
4355.0,
4361.0,
4366.0,
4371.0,
4372.0,
4376.0,
4381.0,
4384.0,
4387.0,
4389.0,
4390.0,
4405.0,
4406.0,
4407.0,
4409.0,
4410.0,
4413.0,
4424.0,
4428.0,
4434.0,
4452.0,
4458.0,
4460.0,
4461.0,
4470.0,
4480.0,
4496.0,
4503.0,
4507.0,
4512.0,
4518.0,
4519.0,
4522.0,
4525.0,
4537.0,
4546.0,
4547.0,
4553.0,
4556.0,
4558.0,
4563.0,
4568.0,
```

```
4574.0,
4578.0,
4587.0,
4592.0,
4594.0,
4610.0,
4615.0,
4617.0,
4619.0,
4620.0,
4625.0,
4626.0,
4635.0,
4636.0,
4641.0,
4647.0,
4650.0,
4653.0,
4658.0,
4660.0,
4665.0,
4668.0,
4669.0,
4670.0,
4674.0,
4675.0,
4676.0,
4679.0,
4680.0,
4686.0,
4690.0,
4699.0,
4711.0,
4713.0,
4718.0,
4721.0,
4722.0,
4724.0,
4730.0,
4731.0,
4733.0,
4735.0,
4739.0,
4743.0,
4745.0,
4746.0,
4747.0,
4748.0,
4761.0,
4762.0,
4763.0,
4769.0,
4771.0,
```

```
 4774.0,
 4784.0,
 4794.0,
 4797.0,
 4801.0,
 4804.0,
 4820.0,
 4833.0,
 4834.0,
 4851.0,
 4859.0,
 4860.0,
 4863.0,
 4873.0,
 4881.0,
 4892.0,
 4897.0,
 4908.0,
 4911.0,
 4915.0,
 4922.0,
 4929.0,
 4936.0,
 4939.0,
 4948.0,
 4951.0,
 4956.0,
 4961.0,
 4964.0,
 4981.0,
 4991.0,
 4992.0,
 ...]
```

In [150]:
```
shot_id_list=pd.DataFrame(shot_id_list)
```

In [151]:
```
shot_id_list.head()
```

Out[151]:

	0
0	1.0
1	8.0
2	17.0
3	20.0
4	22.0

In [152]:
```
shot_id_list.columns=['shot_id_number']
```

In [153]:
```
shot_id_list.head()
```

Out[153]:

	shot_id_number
0	1.0
1	8.0
2	17.0
3	20.0
4	22.0

In [154]:
```
shot_id_list.set_index('shot_id_number',inplace=True)
```

In [155]:
```
shot_id_list.head()
```

Out[155]:

shot_id_number
1.0
8.0
17.0
20.0
22.0

In [156]:
```
#generating submission csv
submission = pd.DataFrame({'shot_id_number':shot_id_list.index,'is_goal':preds})
#save the file to your directory
submission.to_csv('submission_prob.csv',index=False)
```

In [157]:
```
submission.head()
```

Out[157]:

	shot_id_number	is_goal
0	1.0	0.46
1	8.0	0.46
2	17.0	0.46
3	20.0	0.46
4	22.0	0.46

In [158]:
```
submission.set_index('shot_id_number',inplace=True)
```

In [159]:
```
submission.head()
```

Out[159]:

shot_id_number	is_goal
1.0	0.46
8.0	0.46
17.0	0.46
20.0	0.46
22.0	0.46

In [160]:
```
sample_submission.head()
```

Out[160]:

	shot_id_number	is_goal
0	1	0.1
1	8	0.1
2	17	0.1
3	20	0.1
4	33	0.1

In [161]:
```
sample_submission=pd.read_csv('sample_submission.csv')
sample_submission.set_index('shot_id_number',inplace=True)
```

In [162]:
```
sample_submission.head()
```

Out[162]:

shot_id_number	is_goal
1	0.1
8	0.1
17	0.1
20	0.1
33	0.1

In [163]:
```
a = sample_submission.index.values.tolist()
```

In [164]:
```
final_sub_df = submission.loc[a,:]
```

In [167]:
```
final_sub_df.to_csv('FINAL_Shubham_Sunwalka_10_01_1997_submission_2.csv',index = True)
```

In [168]:
```
len(final_sub_df)
```

Out[168]: 5000

The End

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: