ZS Data Science Challenge

Data Preparation -

Column Renaming

- If 2 columns had same, "_1" was added as suffix to second column
- All "-", "." And "/" in column name were replaced by "_"

Observation - Game Id has no missing data

Observation - Multiple columns of same name such as 'distance_of_shot'

- Both columns with same name have missing values
- Second column of the 2 has noisy data in form of decimal numbers identified by looking at variance
- Used values from second column that are not decimals to fill data in first column

Observation - Missing values are present in most features.

Handling Missing Data

- Visualise variances for continuous variables and histograms for categorical attributes
- For Columns with same names, some missing values were filled from second column that were not missing and were not decimal values
- Feature wise analysis on how missing values were handled
    1. **) match_event_id :** Processed based on previous and next row. If game_id of previous row is same as that of the row with the missing value then assign match_event_id as 1 + previous_match_id.
       Similar condition for next row.
       Remaining missing values are padded same as previous values.
    2.) **location_x :** Missing value is found using LinearRegression in which the training features were ['location_y', 'distance_of_shot', 'lower_range', 'upper_range']

3.) **location_y :** Missing value is found using LinearRegression in which the training features were [‘location_x’, ‘distance_of_shot’, ‘lower_range’, ‘upper_range’]
4.) **remaining_min** :
5.) **power_of_shot** : Missing Values were padded
6.) **game_season** : missing Values were padded
7.) **distance_of_shot** : missing values were replaced by mean
8.) **area_of_shot** : missing values were replaced by mode
9.) **shot_basics** : missing values were replaced by mode
10.) **lower_range** :missing values were replaced by mode
11.) **upper_range** :missing values were replaced by mode
12.) **date_of_game**:missing Values were padded
13.) **home**: missing Values were padded
14.) **lat_lng** : missing Values were padded
15.) **type_of_shot** : missing values were assigned a unique value which was not in the column
16.) **type_of_combined_shot** : missing values were assigned a unique value which was not in the column

EDA

Problem

We need to provide a probability to the dependent variable.

We can reduce this to a classification problem in which the dependent variable is binary. Classification algorithms have various probability thresholds for classification. We need to return these probabilities.

Data

Numerical Data -
[match_event_id, location_x, remaining_min, location_y, remaining_sec, distance_of_shot
]
Categorical Data-
[power_of_shot, knockout_match, area_of_shot, shot_basics, range_of_shot, home, lat_lng, type_of_shot, type_of_combined_shot
]

Dealing with missing data

Outliers

Feature Engineering

Transforming/Changing Column Values

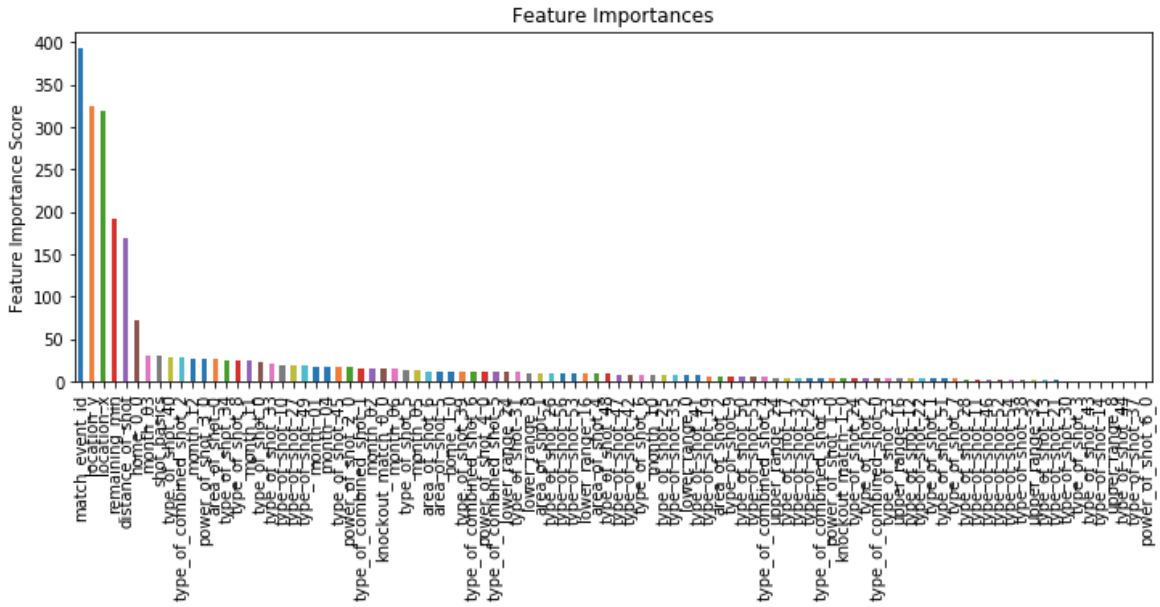
- **‘range_of_shot'** is split into **‘lower_range’** and **‘upper_range’**
- **date_of_game** is split into year, month, day
- **home/away** is transformed to **‘home’** which is boolean variable having values 1 or 0

Feature Contribution

	Specs	Score
16	upper_range	210.617427
15	lower_range	210.617427
9	shot_basics	173.074551
8	area_of_shot	82.023326
7	distance_of_shot	77.158125
2	location_y	35.575411
0	match_event_id	4.725222
17	home	4.008968
3	remaining_min	3.917512
4	power_of_shot	3.793183
6	game_season	1.291374
12	type_of_combined_shot	0.763052
10	lat_lng	0.684391
11	type_of_shot	0.271351
13	month	0.188948
14	year	0.066180
1	location_x	0.052376
5	knockout_match	0.000766

'''
learning_rate = 0.1
Estimators = 149,
Accuracy : 0.6602
AUC Score (Train): 0.727088

learning_rate = 0.5
Estimators = 63,



Accuracy : 0.6528
AUC Score (Train): 0.695635

learning_rate = 0.3
Estimators = 77
Accuracy : 0.6568
AUC Score (Train): 0.710403

'''

({'mean_fit_time': array([25.47543125, 29.29744964, 37.45478344, 51.23701358,
45.63270326,
       61.34791794, 63.19848642, 56.18198481, 58.7547133 , 59.78487515,
       60.19948406, 55.76035213]),
  'std_fit_time': array([0.53580202, 6.02017352, 0.63569917, 1.8163702 ,
12.13455724,

```
        9.0683577 ,  2.39844681,  2.17152607,  3.73682286,  2.62456035,
        1.12735729,  4.10586478]),
 'mean_score_time': array([0.06946297, 0.07501221, 0.17372174, 0.20709124,
0.25987864,
        0.15443721, 0.26897616, 0.21223593, 0.18476658, 0.1734283 ,
        0.20747576, 0.14157748]),
 'std_score_time': array([0.0065865 , 0.00747546, 0.07382257, 0.08597596,
0.1022942 ,
        0.03309303, 0.11645871, 0.06307146, 0.0392818 , 0.02075738,
        0.06637969, 0.03079519]),
 'param_max_depth': masked_array(data=[3, 3, 3, 5, 5, 5, 7, 7, 7, 9, 9, 9],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_min_child_weight': masked_array(data=[1, 3, 5, 1, 3, 5, 1, 3, 5, 1, 3, 5],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'max_depth': 3, 'min_child_weight': 1},
  {'max_depth': 3, 'min_child_weight': 3},
  {'max_depth': 3, 'min_child_weight': 5},
  {'max_depth': 5, 'min_child_weight': 1},
  {'max_depth': 5, 'min_child_weight': 3},
  {'max_depth': 5, 'min_child_weight': 5},
  {'max_depth': 7, 'min_child_weight': 1},
  {'max_depth': 7, 'min_child_weight': 3},
  {'max_depth': 7, 'min_child_weight': 5},
  {'max_depth': 9, 'min_child_weight': 1},
  {'max_depth': 9, 'min_child_weight': 3},
  {'max_depth': 9, 'min_child_weight': 5}],
 'split0_test_score': array([0.65859375, 0.65832107, 0.65746317, 0.65620218,
0.65849361,
        0.65713478, 0.65729401, 0.65315332, 0.65398561, 0.65681114,
        0.65669489, 0.65556753]),
 'split1_test_score': array([0.63414827, 0.6327172 , 0.63272449, 0.63118234,
0.62533873,
        0.63347123, 0.6273156 , 0.62885936, 0.62369662, 0.62028901,
        0.62614652, 0.62337815]),
 'split2_test_score': array([0.64755535, 0.64630454, 0.64629029, 0.6459675 ,
0.65117671,
        0.65230492, 0.64069436, 0.64240812, 0.64426925, 0.62937556,
        0.64228077, 0.63702196]),
```

```
  'split3_test_score': array([0.64026719, 0.63931568, 0.64157998, 0.63952163,
0.63981467,
        0.6418954 , 0.63134946, 0.63694565, 0.63485957, 0.622734  ,
        0.62925185, 0.63085692]),
  'split4_test_score': array([0.65036493, 0.64987004, 0.65032812, 0.64807278,
0.64862052,
        0.64793757, 0.64963812, 0.6510551 , 0.64363405, 0.64543547,
        0.6430769 , 0.64304865]),
  'mean_test_score': array([0.6461859 , 0.6453057 , 0.64567721, 0.64418929,
0.64468885,
        0.64654878, 0.64125831, 0.64248431, 0.64008902, 0.63492904,
        0.63949019, 0.63797464]),
  'std_test_score': array([0.00840512, 0.00878159, 0.00831235, 0.00840858,
0.01136931,
        0.0082438 , 0.01113793, 0.00898452, 0.01019034, 0.01402214,
        0.01095016, 0.01095338]),
  'rank_test_score': array([ 2,  4,  3,  6,  5,  1,  8,  7,  9, 12, 10, 11], dtype=int32)},
 {'max_depth': 5, 'min_child_weight': 5},
 0.6465487780417231)
```

```
({'mean_fit_time': array([28.97310758, 29.19032226, 26.44795671, 32.02436671,
30.11715417,
        31.42415733, 34.43228965, 36.22571115, 30.53552098]),
  'std_fit_time': array([1.03048461, 2.03615444, 0.25609932, 0.95212453,
0.30985652,
        1.70290915, 1.46366673, 1.47890386, 6.02755877]),
  'mean_score_time': array([0.08572855, 0.08006516, 0.07456479, 0.09054503,
0.08925452,
        0.08765597, 0.09144964, 0.11746426, 0.07450438]),
  'std_score_time': array([0.01973125, 0.00963097, 0.00436722, 0.00995394,
0.0104075 ,
        0.0154485 , 0.00340498, 0.05036184, 0.01114788]),
  'param_max_depth': masked_array(data=[4, 4, 4, 5, 5, 5, 6, 6, 6],
               mask=[False, False, False, False, False, False, False, False,
                     False],
        fill_value='?',
             dtype=object),
  'param_min_child_weight': masked_array(data=[4, 5, 6, 4, 5, 6, 4, 5, 6],
               mask=[False, False, False, False, False, False, False, False,
                     False],
```

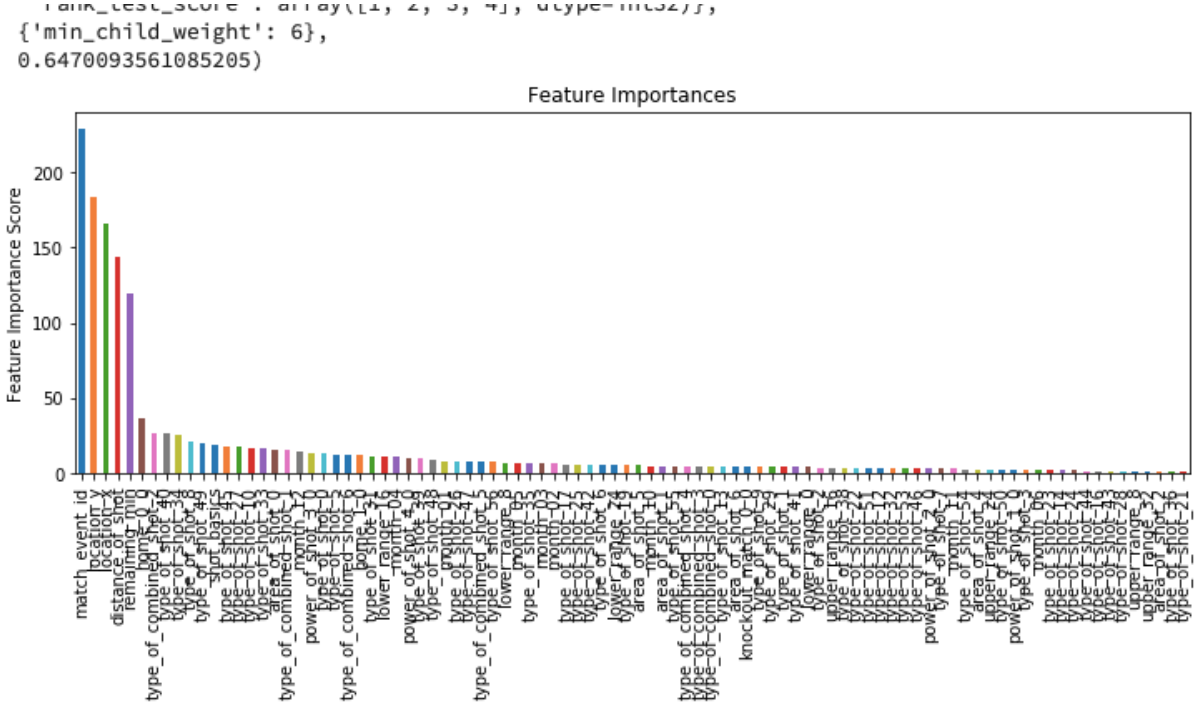
```
        fill_value='?',
            dtype=object),
  'params': [{'max_depth': 4, 'min_child_weight': 4},
   {'max_depth': 4, 'min_child_weight': 5},
   {'max_depth': 4, 'min_child_weight': 6},
   {'max_depth': 5, 'min_child_weight': 4},
   {'max_depth': 5, 'min_child_weight': 5},
   {'max_depth': 5, 'min_child_weight': 6},
   {'max_depth': 6, 'min_child_weight': 4},
   {'max_depth': 6, 'min_child_weight': 5},
   {'max_depth': 6, 'min_child_weight': 6}],
  'split0_test_score': array([0.65867667, 0.65922908, 0.65980717, 0.65839653,
0.65713478,
        0.65443864, 0.65482757, 0.65610399, 0.65871678]),
  'split1_test_score': array([0.6346186 , 0.63633253, 0.6346175 , 0.62868995,
0.63347123,
        0.63285973, 0.62797653, 0.62974795, 0.62981155]),
  'split2_test_score': array([0.65055893, 0.65201824, 0.64821221, 0.65244702,
0.65230492,
        0.6481181 , 0.64482479, 0.64633989, 0.64538304]),
  'split3_test_score': array([0.64389067, 0.64088496, 0.64260806, 0.64157634,
0.6418954 ,
        0.6419311 , 0.63852366, 0.63771689, 0.63855113]),
  'split4_test_score': array([0.65062086, 0.65179845, 0.64980184, 0.64903796,
0.64793757,
        0.64896959, 0.64640811, 0.64936184, 0.64703949]),
  'mean_test_score': array([0.64767315, 0.64805265, 0.64700936, 0.64602956,
0.64654878,
        0.64526343, 0.64251213, 0.64385411, 0.6439004 ]),
  'std_test_score': array([0.00803521, 0.00828742, 0.00831816, 0.01023396,
0.0082438 ,
        0.00736241, 0.00893743, 0.00920025, 0.00957797]),
  'rank_test_score': array([2, 1, 3, 5, 4, 6, 9, 8, 7], dtype=int32)},
 {'max_depth': 4, 'min_child_weight': 5},
 0.6480526533601416)



[139]  train-auc:0.70788+0.00139126 test-auc:0.65114+0.00296943

Model Report
Accuracy : 0.6499
AUC Score (Train): 0.702410
Out[51]:
```

({'mean_fit_time': array([28.53550501, 28.81774368, 27.9477304 , 29.96793151]),
  'std_fit_time': array([0.31587512, 0.49189319, 0.32286106, 1.19242715]),
  'mean_score_time': array([0.07632217, 0.07910891, 0.08253827, 0.06578536]),
  'std_score_time': array([0.00675621, 0.00389049, 0.01364659, 0.01018823]),
  'param_min_child_weight': masked_array(data=[6, 8, 10, 12],
               mask=[False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'min_child_weight': 6},
   {'min_child_weight': 8},
   {'min_child_weight': 10},
   {'min_child_weight': 12}],
  'split0_test_score': array([0.65980717, 0.66010936, 0.65650505, 0.6576631]),
  'split1_test_score': array([0.6346175 , 0.63188329, 0.6321826 , 0.63205185]),
  'split2_test_score': array([0.64821221, 0.64939079, 0.65018323, 0.64607001]),
  'split3_test_score': array([0.64260806, 0.64051036, 0.63973157, 0.64014068]),
  'split4_test_score': array([0.64980184, 0.65209348, 0.65029783, 0.64904237]),
  'mean_test_score': array([0.64700936, 0.64679746, 0.64578006, 0.6449936]),
  'std_test_score': array([0.00831816, 0.00973591, 0.00867524, 0.00859194]),
  'rank_test_score': array([1, 2, 3, 4], dtype=int32)},
 {'min_child_weight': 6},
 0.6470093561085205)



GAMMAGAM

({'mean_fit_time': array([28.59535127, 27.10575109, 28.10539389, 28.15843081,
25.84967532]),
  'std_fit_time': array([0.63246245, 0.43370455, 1.21017517, 0.69832044,
5.40973399]),
  'mean_score_time': array([0.07324219, 0.07971759, 0.09245124, 0.07482023,
0.06453662]),

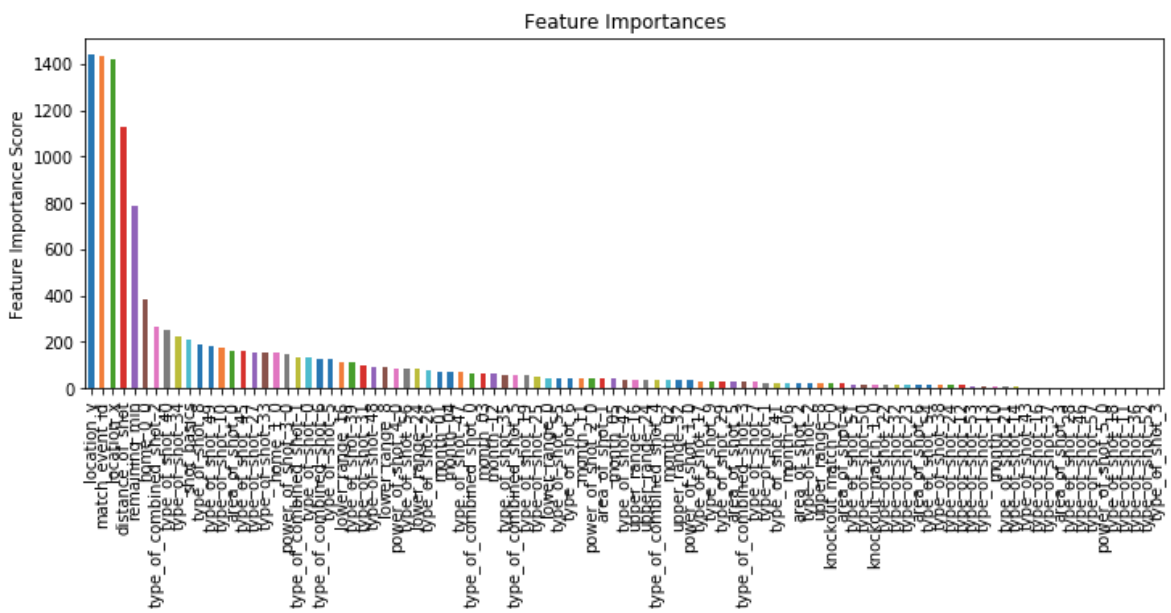
GAMMA

  'std_score_time': array([0.00088793, 0.01010845, 0.02712733, 0.0046064 ,
0.01475829]),
  'param_gamma': masked_array(data=[0.0, 0.1, 0.2, 0.3, 0.4],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'gamma': 0.0},
   {'gamma': 0.1},
   {'gamma': 0.2},
   {'gamma': 0.3},
   {'gamma': 0.4}],
  'split0_test_score': array([0.65980717, 0.65982956, 0.65942715, 0.65925146,
0.65944902]),
  'split1_test_score': array([0.6346175 , 0.63361613, 0.63360961, 0.63426426,
0.63465065]),
  'split2_test_score': array([0.64821221, 0.64870535, 0.64778827, 0.6484782 ,
0.64811454]),
  'split3_test_score': array([0.64260806, 0.6426602 , 0.64269971, 0.64438414,
0.64269301]),
  'split4_test_score': array([0.64980184, 0.64978267, 0.64997557, 0.64997608,
0.65126581]),
  'mean_test_score': array([0.64700936, 0.64691878, 0.64670006, 0.64727083,
0.64723461]),
  'std_test_score': array([0.00831816, 0.00863919, 0.00850005, 0.00812177,
0.00830902]),
  'rank_test_score': array([3, 4, 5, 1, 2], dtype=int32)},
 {'gamma': 0.3},
 0.6472708288736031)

```
xgb4 = XGBClassifier(
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=4,
 min_child_weight=5,
 gamma=0.3,
 subsample=0.9,
 colsample_bytree=0.55,
 reg_alpha=0.1,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb4, train, predictors)
```

[1154]train-auc:0.700428+0.000768312 test-auc:0.650449+0.00381668

Model Report
Accuracy : 0.6485
AUC Score (Train): 0.692643



[[1157 197]
 [708 381]]
 0.6295538272615636

 [[10648 1548]
 [6075 3715]]
 0.653279359592468

Precision Recall Score = 0.5204261614465217

Classification Report

	precision	recall	f1-score	support
0.0	0.62	0.85	0.72	1354
1.0	0.66	0.35	0.46	1089
accuracy			0.63	2443
macro avg	0.64	0.60	0.59	2443
weighted avg	0.64	0.63	0.60	2443