In [10]:
```
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt
import seaborn as sns
import pprint

from scipy import stats as stats
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score,GridSea
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import plot_confusion_matrix, confusion_matrix, plot_roc
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import export_graphviz

import pickle

import requests
from bs4 import BeautifulSoup as bs
from bs4 import Comment

import regex
import datetime

from tqdm import tqdm
```

Flags for purposes of running code when certain data has already been collected and/or final models have been trained.

In [159…
```
__DATA_COLLECTED__=True
__MODELS_TRAINED__=False
```

Data Uploading & Cleaning

Due to the unusual and shortened schedule in 2020, chose to drop. Similarly for the not-yet-completed 2021 season.

Odds Data (Already Downloaded from Website)

In [2]:
```
years=['2019','2018','2017','2016','2015']
data={}
```

In [3]:
```
for yr in years:
    data[yr]=pd.read_excel('../Raw Data/'+yr+'.xlsx')
```

In [4]:
```
for yr,df in data.items():
    df_clean=df.drop(['Rot','1st','2nd','3rd','4th','5th','6th','7th','8th','
    df_clean['year']=int(yr)
    df_clean['day']=df_clean.Date.apply(lambda x:x%100)
    df_clean['month']=df_clean.Date.apply(lambda x:int(np.round(x,-2)/100))
    df_clean['date']=df_clean.apply(lambda r:datetime.datetime(r.year,r.month
    df_clean.drop(['Date','day'],axis=1,inplace=True)
    data[yr]=df_clean.rename({'Open OU':'OpenOU','Close OU':'CloseOU','Unname
```

In [5]:
```
all_data=pd.concat(data.values(),ignore_index=True)
```

In [6]:
```
patt=r'\-[A-Z]{1}'
all_data.Pitcher=all_data.Pitcher.str.replace(patt,'')
```

Cleaning Team Labels

Checking team labels in the odds data, note a couple of issues to fix.

In [7]:
```
all_data.Team.value_counts()
```

Out[7]:
```
COL    811
MIL    811
CUB    811
SFO    810
CIN    810
HOU    810
MIN    810
WAS    810
BAL    810
ARI    810
TEX    810
OAK    810
SEA    810
PIT    810
PHI    810
NYM    810
LAA    810
TOR    810
SDG    809
TAM    809
KAN    809
BOS    809
ATL    809
CWS    809
NYY    809
STL    808
MIA    808
CLE    808
DET    807
LOS    485
LAD    326
Name: Team, dtype: int64
```

First, some inconsistencent labeling (LOS v LA, e.g.).

In [8]:
```
all_data.Team.replace({'CUB':'CHC','LOS':'LAD','SFG':'SFO','BRS':'BOS'}).valu
```

Out[8]:
```
COL    811
CHC    811
MIL    811
LAD    811
SFO    810
CIN    810
HOU    810
MIN    810
WAS    810
BAL    810
ARI    810
TEX    810
OAK    810
PIT    810
PHI    810
SEA    810
LAA    810
TOR    810
NYM    810
SDG    809
KAN    809
TAM    809
BOS    809
NYY    809
ATL    809
CWS    809
STL    808
MIA    808
CLE    808
DET    807
Name: Team, dtype: int64
```

In [9]:
```
all_data.Team=all_data.Team.replace({'CUB':'CHC','LOS':'LAD','SFG':'SFO','BRS
```

Second, need to adjust labels to be consistent with official labels.

In [10]:
```
all_data.Team=all_data.Team.replace({
                                     'KAN':'KCR',
                                     'TAM':'TBR',
                                     'SDG':'SDP',
                                     'WAS':'WSN',
                                     'SFO':'SFG',
                                     'CWS':'CHW'
                                     })
```

Home & Away Labels

There are four (two games) rows which have neither. Upon further expection, this comes from games played outside of the US. Links:

1) https://www.baseball-reference.com/boxes/OAK/OAK201903200.shtml

2) https://www.baseball-reference.com/boxes/OAK/OAK201903210.shtml

In [16]:
```
all_data.VH.value_counts()
```

Out[16]:
```
V    12144
H    12144
Name: VH, dtype: int64
```

In [17]:
```
all_data[all_data.VH=='N']
```

Out[17]:

VH	Team	Pitcher	Final	OpenOU	Open_Odds	CloseOU	Close_Odds	year	month	date

In [13]:
```
all_data.at[0,'VH']='V'
all_data.at[1,'VH']='H'
all_data.at[2,'VH']='V'
all_data.at[3,'VH']='H'
```

In [14]:
```
all_data.head()
```

Out[14]:

	VH	Team	Pitcher	Final	OpenOU	Open_Odds	CloseOU	Close_Odds	year	month
0	V	SEA	MGONZALES	9	8.5	-110	8.5	-120	2019	3
1	H	OAK	MFIERS	7	8.5	-110	8.5	100	2019	3
2	V	SEA	YKIKUCHI	5	9.0	-110	9.0	-130	2019	3
3	H	OAK	MESTRADA	4	9.0	-110	9.0	110	2019	3
4	V	NYM	JDEGROM	2	6.5	-110	6.5	-105	2019	3

In [15]:
```
all_data.VH.value_counts()
```

Out[15]:
```
V    12144
H    12144
Name: VH, dtype: int64
```

Missing Data & Outliers

Looking at the runs column, there is a single missing observation.

In [18]:
```
all_data.Final.value_counts()
```

Out[18]:
```
3     3313
2     3207
4     3137
5     2779
1     2571
6     2151
7     1677
0     1457
8     1267
9      831
10     650
11     444
12     292
13     185
14     124
15      73
16      48
17      34
19      16
18      13
21       7
20       4
22       2
23       2
NL       2
25       1
24       1
Name: Final, dtype: int64
```

In [19]:
```
all_data[all_data.Final=='NL']
```

Out[19]:

	VH	Team	Pitcher	Final	OpenOU	Open_Odds	CloseOU	Close_Odds	year	month	
16276	V	NYM	SMATZ	NL	8.0	100	7.5	-110	2016	6	2
16277	H	PIT	JNIESE	NL	8.0	-120	7.5	-110	2016	6	2

It is easy to fill in this infusing the very reliable baseball-reference resource.
https://www.baseball-reference.com/boxes/PIT/PIT201606071.shtml

In [20]:
```
all_data.at[16276,'Final']=3
all_data.at[16277,'Final']=1
```

In [21]:
```
all_data.Final.value_counts()
```

Out[21]:
```
3     3314
2     3207
4     3137
5     2779
1     2572
6     2151
7     1677
0     1457
8     1267
9      831
10     650
11     444
12     292
13     185
14     124
15      73
16      48
17      34
19      16
18      13
21       7
20       4
22       2
23       2
24       1
25       1
Name: Final, dtype: int64
```

In [25]:
```
#Convert column into integers.
all_data.Final=all_data.Final.astype(int)
```

There are some outrageous values that cannot be explained. Since there is no other resource to check for the correct value, replace the line @ close with the line @ open.

In [26]:
```
all_data.CloseOU.value_counts()
```

Out[26]:
```
8.5      4710
9.0      4402
8.0      3596
7.5      3334
9.5      2466
7.0      2004
10.0     1172
10.5      902
6.5       550
11.0      470
11.5      226
6.0       138
12.0      108
12.5       76
13.0       56
13.5       30
5.5        16
14.0       14
14.5        6
102.0       2
15.0        2
104.0       2
5.0         2
101.0       2
100.0       2
Name: CloseOU, dtype: int64
```

In [27]:
```
all_data[all_data.CloseOU > 20]
```

Out[27]:

	VH	Team	Pitcher	Final	OpenOU	Open_Odds	CloseOU	Close_Odds	year	m
5872	V	DET	MFULMER	6	9.5	-110	104.0	-110	2018	
5873	H	TEX	MMOORE	7	9.5	-110	104.0	-110	2018	
6740	V	STL	MWACHA	6	8.5	-120	102.0	-110	2018	
6741	H	CIN	LCASTILLO	4	8.5	100	102.0	-110	2018	
8842	V	SEA	ERAMIREZ	3	8.0	-110	101.0	-110	2018	
8843	H	SDP	JLUCCHESI	8	8.0	-110	101.0	-110	2018	
9478	V	KCR	JJUNIS	4	8.5	-110	100.0	-110	2018	
9479	H	DET	JZIMMERMANN	5	8.5	-110	100.0	-110	2018	

In [28]:
```
all_data.CloseOU=all_data.apply(lambda r:r.OpenOU if r.CloseOU>20 else r.Clos
```

In [29]:
```
all_data.CloseOU.value_counts()
```

Out[29]:
```
8.5     4714
9.0     4402
8.0     3598
7.5     3334
9.5     2468
7.0     2004
10.0    1172
10.5     902
6.5      550
11.0     470
11.5     226
6.0      138
12.0     108
12.5      76
13.0      56
13.5      30
5.5       16
14.0      14
14.5       6
5.0        2
15.0       2
Name: CloseOU, dtype: int64
```

In [31]:
```
#Convert this column into floats
all_data.CloseOU=all_data.CloseOU.astype(float)
```

Consolidating Data: One Observation for each Game in Date Range.

The original data is a bit confusing. For each game, there are two rows in the dataset: one containing data for the Road team and one for the Home Team. However, the over-under variable in both rows refers to the total runs scored.

As a result, need to consolidate each row-pair into a single row with both Home and Away data. Because the paired rows are adjacent, it is easy to combine. This was done for each row--even though it adds significant execution time--and removed duplicates to be safe.

In [36]:
```
if __DATA_COLLECTED__:
    all_data=pd.read_pickle('../Primary Data/Bench_1.pkl')
    pass
else:
    all_data['Home_Pitcher']=''
    all_data['Away_Pitcher']=''
    all_data['Home_Team']=''
    all_data['Away_Team']=''
    all_data['Home_Score']=0
    all_data['Away_Score']=0
    for i,row in tqdm(all_data.iterrows()):
        if row.VH=='V':
            all_data.loc[i,'Away_Pitcher']=all_data.loc[i,'Pitcher']
            all_data.loc[i,'Away_Team']=all_data.loc[i,'Team']
            all_data.loc[i,'Away_Score']=all_data.loc[i,'Final']
            all_data.loc[i,'Home_Score']=all_data.loc[i+1,'Final']
            all_data.loc[i,'Home_Team']=all_data.loc[i+1,'Team']
            all_data.loc[i,'Home_Pitcher']=all_data.loc[i+1,'Pitcher']
            pass
        elif row.VH=='H':
            all_data.loc[i,'Away_Pitcher']=all_data.loc[i-1,'Pitcher']
            all_data.loc[i,'Away_Team']=all_data.loc[i-1,'Team']
            all_data.loc[i,'Away_Score']=all_data.loc[i-1,'Final']
            all_data.loc[i,'Home_Score']=all_data.loc[i,'Final']
            all_data.loc[i,'Home_Team']=all_data.loc[i,'Team']
            all_data.loc[i,'Home_Pitcher']=all_data.loc[i,'Pitcher']
            pass
        else:
            pass
    all_data.to_pickle('../Primary Data/Bench_1.pkl')
    pass
all_data
```

```
24288it [02:40, 151.00it/s]
```

Out[36]:

	VH	Team	Pitcher	Final	OpenOU	Open_Odds	CloseOU	Close_Odds	year	mo
0	V	SEA	MGONZALES	9	8.5	-110	8.5	-120	2019	
1	H	OAK	MFIERS	7	8.5	-110	8.5	100	2019	
2	V	SEA	YKIKUCHI	5	9.0	-110	9.0	-130	2019	
3	H	OAK	MESTRADA	4	9.0	-110	9.0	110	2019	
4	V	NYM	JDEGROM	2	6.5	-110	6.5	-105	2019	
...	...	...	...	...	...	...	...	...	...	...
24283	H	CHW	FMONTAS	0	8.5	-110	7.5	-130	2015	
24284	V	HOU	MCCULLERS	3	8.5	-105	8.5	-105	2015	
24285	H	ARI	RRAY	5	8.5	-115	8.5	-115	2015	
24286	V	STL	JLACKEY	0	7.0	105	7.0	105	2015	
24287	H	ATL	SMILLER	6	7.0	-125	7.0	-125	2015	

24288 rows × 17 columns

In [40]:
```
all_data=all_data.drop_duplicates(subset=['Away_Pitcher','Home_Pitcher','date
all_data=all_data.copy()
```

Dropping and Create Features

Dropping certain columns that are definetly not going to be used or are duplicative of other columns.

In [42]:
```
all_data.drop(['VH','Team','Pitcher','Final','OpenOU','Open_Odds'],axis=1,inp
all_data
```

Out[42]:

	CloseOU	Close_Odds	year	month	date	Home_Pitcher	Away_Pitcher	Home_Team
0	8.5	-120	2019	3	2019-03-20	MFIERS	MGONZALES	OAK
1	9.0	-130	2019	3	2019-03-21	MESTRADA	YKIKUCHI	OAK
2	6.5	-105	2019	3	2019-03-28	MSCHERZER	JDEGROM	WSN
3	8.5	-120	2019	3	2019-03-28	JCHACIN	MMIKOLAS	MIL
4	8.0	-105	2019	3	2019-03-28	ANOLA	JTEHERAN	PHI
...	...	...	...	...	...	...	...	...
12139	7.5	-115	2015	10	2015-10-04	VNUNO	CBASSITT	SEA
12140	7.5	-120	2015	10	2015-10-04	RNOLASCO	JCUETO	MIN
12141	7.5	110	2015	10	2015-10-04	FMONTAS	DNORRIS	CHW
12142	8.5	-105	2015	10	2015-10-04	RRAY	MCCULLERS	ARI
12143	7.0	105	2015	10	2015-10-04	SMILLER	JLACKEY	ATL

12144 rows × 11 columns

Create Total Runs; OVER Dummy Variable; and a key value.

To build the key value, start with date + Home Team (later date sources will use a similar key). However, because of double-headers, this does not create unique key values. Next, tried date combined with the runs scored by the home and away team. This works well with one exception where both games in a double header ended in the same score. These observations' key-values were manually adjusted.

In [43]:
```
all_data['Total_Runs']=all_data['Home_Score']+all_data['Away_Score']
all_data['OVER']=all_data.apply(lambda r:r.Total_Runs>r.CloseOU,axis=1)
all_data
```

Out[43]:

	CloseOU	Close_Odds	year	month	date	Home_Pitcher	Away_Pitcher	Home_Team
0	8.5	-120	2019	3	2019-03-20	MFIERS	MGONZALES	OAK
1	9.0	-130	2019	3	2019-03-21	MESTRADA	YKIKUCHI	OAK
2	6.5	-105	2019	3	2019-03-28	MSCHERZER	JDEGROM	WSN
3	8.5	-120	2019	3	2019-03-28	JCHACIN	MMIKOLAS	MIL
4	8.0	-105	2019	3	2019-03-28	ANOLA	JTEHERAN	PHI
...	...	...	...	...	...	...	...	...
12139	7.5	-115	2015	10	2015-10-04	VNUNO	CBASSITT	SEA
12140	7.5	-120	2015	10	2015-10-04	RNOLASCO	JCUETO	MIN
12141	7.5	110	2015	10	2015-10-04	FMONTAS	DNORRIS	CHW
12142	8.5	-105	2015	10	2015-10-04	RRAY	MCCULLERS	ARI
12143	7.0	105	2015	10	2015-10-04	SMILLER	JLACKEY	ATL

12144 rows × 13 columns

In [44]:
```
all_data['KEY']=all_data.date.apply(lambda d:str(d).replace(' 00:00:00','-'))
```

In [45]:
```
all_data[all_data.KEY.duplicated(keep=False)]
```

Out[45]:

	CloseOU	Close_Odds	year	month	date	Home_Pitcher	Away_Pitcher	Home_Team
303	10.0	-105	2019	4	2019-04-20	ACOBB	MPEREZ	BAL
306	8.5	100	2019	4	2019-04-20	CKLUBER	JTEHERAN	CLE
307	8.0	-115	2019	4	2019-04-20	TBAUER	UNDECIDED	CLE
308	9.0	-120	2019	4	2019-04-20	DSTRAILY	JBERRIOS	BAL
341	9.0	-115	2019	4	2019-04-23	HVELAZQUEZ	STURNBULL	BOS
...	...	...	...	...	...	...	...	...
12125	7.5	105	2015	10	2015-10-03	AHARANG	TKOEHLER	PHI
12126	7.5	-110	2015	10	2015-10-03	WCHEN	INOVA	BAL
12127	7.0	107	2015	10	2015-10-03	SYNDERGARD	GGONZALEZ	NYM
12128	7.5	100	2015	10	2015-10-04	MWISLER	LLYNN	ATL
12143	7.0	105	2015	10	2015-10-04	SMILLER	JLACKEY	ATL

272 rows × 14 columns

In [46]:
```
all_data.KEY=all_data.apply(lambda r:r.KEY+str(r.Home_Score)+str(r.Away_Score
all_data[all_data.KEY.duplicated(keep=False)]
```

Out[46]:

	CloseOU	Close_Odds	year	month	date	Home_Pitcher	Away_Pitcher	Home_Team	A
8147	7.5	-120	2016	6	2016-06-07	JNIESE	SMATZ	PIT	
8162	7.5	100	2016	6	2016-06-07	JNICASIO	JDEGROM	PIT	

In [47]:
```
all_data.at[8147,'KEY']=all_data.at[8147,'KEY']='A'
all_data.at[8162,'KEY']=all_data.at[8162,'KEY']='B'
```

In [48]:
```
all_data
```

Out[48]:

	CloseOU	Close_Odds	year	month	date	Home_Pitcher	Away_Pitcher	Home_Team
0	8.5	-120	2019	3	2019-03-20	MFIERS	MGONZALES	OAK
1	9.0	-130	2019	3	2019-03-21	MESTRADA	YKIKUCHI	OAK
2	6.5	-105	2019	3	2019-03-28	MSCHERZER	JDEGROM	WSN
3	8.5	-120	2019	3	2019-03-28	JCHACIN	MMIKOLAS	MIL
4	8.0	-105	2019	3	2019-03-28	ANOLA	JTEHERAN	PHI
...	...	...	...	...	...	...	...	...
12139	7.5	-115	2015	10	2015-10-04	VNUNO	CBASSITT	SEA
12140	7.5	-120	2015	10	2015-10-04	RNOLASCO	JCUETO	MIN
12141	7.5	110	2015	10	2015-10-04	FMONTAS	DNORRIS	CHW
12142	8.5	-105	2015	10	2015-10-04	RRAY	MCCULLERS	ARI
12143	7.0	105	2015	10	2015-10-04	SMILLER	JLACKEY	ATL

12144 rows × 14 columns

In [49]:
```
all_data.to_pickle('../Base DFs/Odds_Data.pkl')
```

Scrape Game Info from Baseball-Reference

In [51]:
```
TEAMS={}
with open('../Teams.txt') as f:
    for r in f.readlines():
        t=r.split(',')
        TEAMS[t[0]]=t[1]

YRs=['2019','2018','2017','2016','2015']
odds_data=pd.read_pickle('../Base DFs/Odds_Data.pkl')
```

Game Matchup Data

Baseball-Reference has a page with the entire MLB schedule in any given year containing: links to the game's box score, each participating team's homepage, the score and date. Keeping the last two for connecting to odds_data's KEY value.

In [52]:
```
s_patt=r'([A-Za-z .\']+) \(([0-9]{1,2})\)\s{0,2}\@ ([A-Za-z .\']+) \(([0-9]{1
s_rgx=regex.compile(s_patt)
d_patt=r'[A-Z]{3}([0-9]{8})[0-9]{1}\.shtml'
d_rgx=regex.compile(d_patt)
```

In [53]:
```
game_data=[]
for yr in YRs:
    link='https://www.baseball-reference.com/leagues/MLB/'+yr+'-schedule.shtm
    r=requests.get(link)
    soup=bs(r.content)
    table=soup.find('span',attrs={'data-label':'MLB Schedule'}).parent.find_n
    games=table.find_all('p',attrs={'class':'game'})
    for g in tqdm(games):
        s=' '.join(g.stripped_strings)
        info=s_rgx.search(s).groups()
        links=g.find_all('a')
        l=links[-1]['href']
        d={'Away_Team':links[0]['href'],
           'Home_Team':links[1]['href'],
           'Away_Score':info[1],
           'Home_Score':info[3],
           'link':l,
           'date':pd.to_datetime(d_rgx.search(l).groups()[0],format='%Y%m%d')
          }
        game_data.append(d)
```

```
100%|██████████| 2429/2429 [00:00<00:00, 5364.31it/s]
100%|██████████| 2431/2431 [00:00<00:00, 4541.65it/s]
100%|██████████| 2430/2430 [00:00<00:00, 5151.93it/s]
100%|██████████| 2428/2428 [00:00<00:00, 5277.47it/s]
100%|██████████| 2429/2429 [00:00<00:00, 5182.46it/s]
```

In [54]:
```
len(game_data)
```

Out[54]:
```
12147
```

In [55]:
```
games_meta=pd.DataFrame(game_data)
games_meta
```

Out[55]:

	Away_Team	Home_Team	Away_Score	Home_Score	
0	/teams/SEA/2019.shtml	/teams/OAK/2019.shtml	9	7	/boxes/OAK/OAK
1	/teams/SEA/2019.shtml	/teams/OAK/2019.shtml	5	4	/boxes/OAK/OAK
2	/teams/PIT/2019.shtml	/teams/CIN/2019.shtml	3	5	/boxes/CIN/CIN
3	/teams/CHW/2019.shtml	/teams/KCR/2019.shtml	3	5	/boxes/KCA/KCA
4	/teams/ARI/2019.shtml	/teams/LAD/2019.shtml	5	12	/boxes/LAN/LAN
...	...	...	...	...	
12142	/teams/CIN/2015.shtml	/teams/PIT/2015.shtml	0	4	/boxes/PIT/PIT
12143	/teams/OAK/2015.shtml	/teams/SEA/2015.shtml	2	3	/boxes/SEA/SEA
12144	/teams/COL/2015.shtml	/teams/SFG/2015.shtml	7	3	/boxes/SFN/SFN
12145	/teams/TOR/2015.shtml	/teams/TBR/2015.shtml	3	12	/boxes/TBA/TBA
12146	/teams/LAA/2015.shtml	/teams/TEX/2015.shtml	2	9	/boxes/TEX/TEX

12147 rows × 6 columns

In [79]:
```
patt=r'\/teams\/([A-Z]{3})\/[0-9]{4}\.shtml'
games_meta.Away_Team=games_meta.Away_Team.str.extract(patt)
games_meta.Home_Team=games_meta.Home_Team.str.extract(patt)
games_meta
```

Out[79]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	d
0	SEA	OAK	9	7	/boxes/OAK/OAK201903200.shtml	20
1	SEA	OAK	5	4	/boxes/OAK/OAK201903210.shtml	20 03
2	PIT	CIN	3	5	/boxes/CIN/CIN201903280.shtml	20
3	CHW	KCR	3	5	/boxes/KCA/KCA201903280.shtml	20
4	ARI	LAD	5	12	/boxes/LAN/LAN201903280.shtml	20
...	...	...	...	...	...	
12142	CIN	PIT	0	4	/boxes/PIT/PIT201510040.shtml	20
12143	OAK	SEA	2	3	/boxes/SEA/SEA201510040.shtml	20
12144	COL	SFG	7	3	/boxes/SFN/SFN201510040.shtml	20
12145	TOR	TBR	3	12	/boxes/TBA/TBA201510040.shtml	20
12146	LAA	TEX	2	9	/boxes/TEX/TEX201510040.shtml	20

12147 rows × 7 columns

Create the KEY value in the same way as the Odds Data.

In [80]:
```
games_meta['KEY']=games_meta.date.apply(lambda d:str(d).replace(' 00:00:00','
games_meta.KEY=games_meta.apply(lambda r:r.KEY+str(r.Home_Score)+str(r.Away_S
games_meta[games_meta.KEY.duplicated(keep=False)]
```

Out[80]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	date
8158	NYM	PIT	1	3	/boxes/PIT/PIT201606071.shtml	2016-06-07
8159	NYM	PIT	1	3	/boxes/PIT/PIT201606072.shtml	2016-06-07

In [82]:

```
games_meta.at[8158,'KEY']=games_meta.at[8158,'KEY']='A'
games_meta.at[8159,'KEY']=games_meta.at[8159,'KEY']='B'
games_meta
```

Out[82]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	d
0	SEA	OAK	9	7	/boxes/OAK/OAK201903200.shtml	20
1	SEA	OAK	5	4	/boxes/OAK/OAK201903210.shtml	20 03
2	PIT	CIN	3	5	/boxes/CIN/CIN201903280.shtml	20
3	CHW	KCR	3	5	/boxes/KCA/KCA201903280.shtml	20
4	ARI	LAD	5	12	/boxes/LAN/LAN201903280.shtml	20
...	...	...	...	...	...	
12142	CIN	PIT	0	4	/boxes/PIT/PIT201510040.shtml	20
12143	OAK	SEA	2	3	/boxes/SEA/SEA201510040.shtml	20
12144	COL	SFG	7	3	/boxes/SFN/SFN201510040.shtml	20
12145	TOR	TBR	3	12	/boxes/TBA/TBA201510040.shtml	20
12146	LAA	TEX	2	9	/boxes/TEX/TEX201510040.shtml	20

12147 rows × 7 columns

Done with this data (except for merging) so save out.

In [83]:
```
games_meta.to_pickle('../Base DFs/Games_Meta.pkl')
```

On merge, note the loss of data. This is because the score reported in the odds data does not match what was collected from Baseball-Reference. Given the size of the dataset, it makes sense to drop these observations (31 in total.)

In [85]:
```
game_data=pd.merge(games_meta,odds_data,on='KEY',how='inner')
game_data=game_data.drop(['Away_Score_y','Home_Score_y','Away_Team_y','Home_T
game_data=game_data.rename({'Away_Score_x':'Away_Score','Home_Score_x':'Home_
game_data=game_data.astype({'Away_Score':'int64','Home_Score':'int64','Close_
game_data.CloseOU=game_data.CloseOU.astype(float)
game_data['Total_Runs']=game_data.Away_Score+game_data.Home_Score
game_data
```

Out[85]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	d
0	SEA	OAK	9	7	/boxes/OAK/OAK201903200.shtml	20
1	SEA	OAK	5	4	/boxes/OAK/OAK201903210.shtml	20 03
2	PIT	CIN	3	5	/boxes/CIN/CIN201903280.shtml	20
3	CHW	KCR	3	5	/boxes/KCA/KCA201903280.shtml	20
4	ARI	LAD	5	12	/boxes/LAN/LAN201903280.shtml	20
...	...	...	...	...	...	
12111	CIN	PIT	0	4	/boxes/PIT/PIT201510040.shtml	20
12112	OAK	SEA	2	3	/boxes/SEA/SEA201510040.shtml	20
12113	COL	SFG	7	3	/boxes/SFN/SFN201510040.shtml	20
12114	TOR	TBR	3	12	/boxes/TBA/TBA201510040.shtml	20
12115	LAA	TEX	2	9	/boxes/TEX/TEX201510040.shtml	20

12116 rows × 15 columns

Some Additional Games Data

In [62]:
```
base='https://www.baseball-reference.com/teams/{t}/{y}-schedule-scores.shtml'
stats=['boxscore','day_or_night','attendance','cli']
game_stats={}
```

In [67]:
```
if __DATA_COLLECTED__:
    with open('../Primary Data/Bench_2.pkl','rb') as f:
        game_stats=pickle.load(f)
    pass
else:
    for yr in YRs:
        for team in tqdm(TEAMS.keys()):
            link=base.format_map({'t':team,'y':yr})
            r=requests.get(link)
            soup=bs(r.content)
            table=soup.find('table',attrs={'id':'team_schedule'}).tbody
            games=table.find_all('tr')
            i=1
            for g in games:
                info=g.find_all('td',attrs={'data-stat':stats})
                if(info):
                    link=info[0].a['href']
                    d=dict(zip(stats[1:],[col.text for col in info[1:]]))
                    d['link']=link
                    game_stats[team+'_'+yr+'_'+str(i)]=d
                    i+=1
    with open('../Primary Data/Bench_2.pkl','wb') as f:
        pickle.dump(game_stats,f)
    pass
```

```
100%|██████████| 30/30 [00:12<00:00,  2.37it/s]
100%|██████████| 30/30 [00:12<00:00,  2.41it/s]
100%|██████████| 30/30 [00:13<00:00,  2.27it/s]
100%|██████████| 30/30 [00:15<00:00,  1.99it/s]
100%|██████████| 30/30 [00:14<00:00,  2.03it/s]
```

In [68]:
```
#Put into dataframe & save out properly.
game_logs=pd.DataFrame.from_dict(game_stats,orient='index')
game_logs.to_pickle('../Base DFs/Games_Info_I.pkl')
```

In [86]:
```
game_data=pd.merge(game_data,game_logs.drop_duplicates(['link']),left_on=['li
game_data
```

Out[86]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	d
0	SEA	OAK	9	7	/boxes/OAK/OAK201903200.shtml	20
1	SEA	OAK	5	4	/boxes/OAK/OAK201903210.shtml	20 03
2	PIT	CIN	3	5	/boxes/CIN/CIN201903280.shtml	20
3	CHW	KCR	3	5	/boxes/KCA/KCA201903280.shtml	20
4	ARI	LAD	5	12	/boxes/LAN/LAN201903280.shtml	20
...	...	...	...	...	...	
12111	CIN	PIT	0	4	/boxes/PIT/PIT201510040.shtml	20
12112	OAK	SEA	2	3	/boxes/SEA/SEA201510040.shtml	20
12113	COL	SFG	7	3	/boxes/SFN/SFN201510040.shtml	20
12114	TOR	TBR	3	12	/boxes/TBA/TBA201510040.shtml	20
12115	LAA	TEX	2	9	/boxes/TEX/TEX201510040.shtml	20

12116 rows × 18 columns

Refer to Baseball-Reference's box score page to collect the name and link to both pitchers' player's page. The boxscore link is then used to merge the resulting dataframe with the other dataframes.

In [ ]:
```
if __DATA_COLLECTED__:
    with open('../Primary Data/Bench_3.pkl','rb') as f:
        more_data=pickle.load(f)
    pass
else:
    base='https://www.baseball-reference.com'
    more_data={}
    for game in tqdm(game_data.link):
        try:
            D={}
            link=base+game
            r=requests.get(link)
            soup=bs(r.content)
            X=soup.find('span',attrs={'data-label':'Pitching Lines and Info'}
            lineups=bs(X.find_all(text=lambda text:isinstance(text,Comment))[
            pitcher=lineups[0].tbody.tr.th.a
            D['Away_Pitch']=(pitcher.text,pitcher['href'])
            pitcher=lineups[1].tbody.tr.th.a
            D['Home_Pitch']=(pitcher.text,pitcher['href'])
            X=soup.find('span',attrs={'data-label':'Other Info'}).parent.pare
            other=bs(X.find_all(text=lambda text:isinstance(text,Comment))[0]
            D['other']=other.text.strip()
            more_data[game]=D
        except:
            print(game)
            pass
        pass
    with open('../Primary Data/Bench_3.pkl','wb') as f:
        pickle.dump(more_data,f)
    pass
###
```

In [89]:
```
game_detail=pd.DataFrame.from_dict(more_data,orient='index')
game_detail.to_pickle('../Base DFs/Games_Info_II.pkl')
game_detail
```

Out[89]:

	Away_Pitch	Home_Pitch	
/boxes/OAK/OAK201903200.shtml	(Marco Gonzales, /players/g/gonzama02.shtml)	(Mike Fiers, /players/f/fiersmi01.shtml)	U Hl Nelsc Gib
/boxes/OAK/OAK201903210.shtml	(Yusei Kikuchi, /players/k/kikucyu01.shtml)	(Marco Estrada, /players/e/estrama01.shtml)	U H Well Bark
/boxes/CIN/CIN201903280.shtml	(Jameson Taillon, /players/t/taillja01.shtml)	(Luis Castillo, /players/c/castilu02.shtml)	U H We Eric C

/boxes/KCA/KCA201903280.shtml	(Carlos Rodon, /players/r/rodonca01.shtml)	(Brad Keller, /players/k/kellebr01.shtml)	U HP Mea Ron
/boxes/LAN/LAN201903280.shtml	(Zack Greinke, /players/g/greinza01.shtml)	(Hyun Jin Ryu, /players/r/ryuhy01.shtml)	U HP Gorr B
...	...	...	
/boxes/PIT/PIT201510040.shtml	(Josh A. Smith, /players/s/smithjo07.shtml)	(J.A. Happ, /players/h/happja01.shtml)	U H Well Ti
/boxes/SEA/SEA201510040.shtml	(Chris Bassitt, /players/b/bassich01.shtml)	(Vidal Nuno III, /players/n/nunovi01.shtml)	U HP Esta Hic
/boxes/SFN/SFN201510040.shtml	(Christian Bergman, /players/b/bergmch01.shtml)	(Matt Cain, /players/c/cainma01.shtml)	U HP - Torre Fl
/boxes/TBA/TBA201510040.shtml	(Mark Buehrle, /players/b/buehrma01.shtml)	(Matt Moore, /players/m/moorema02.shtml)	U Marq I
/boxes/TEX/TEX201510040.shtml	(Garrett Richards, /players/r/richaga01.shtml)	(Cole Hamels, /players/h/hamelco01.shtml)	U HP Cede 1B

12116 rows × 3 columns

In [90]:
```
game_data=pd.merge(game_data,game_detail,how='left',right_index=True,left_on=
print(game_data.columns)
game_data
```

```
Index(['Away_Team', 'Home_Team', 'Away_Score', 'Home_Score', 'link', 'date',
       'KEY', 'CloseOU', 'Close_Odds', 'year', 'month', 'Home_Pitcher',
       'Away_Pitcher', 'OVER', 'Total_Runs', 'day_or_night', 'attendance',
```

```
 'cli', 'Away_Pitch', 'Home_Pitch', 'other'],
dtype='object')
```

Out[90]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	d
0	SEA	OAK	9	7	/boxes/OAK/OAK201903200.shtml	20
1	SEA	OAK	5	4	/boxes/OAK/OAK201903210.shtml	20 03
2	PIT	CIN	3	5	/boxes/CIN/CIN201903280.shtml	20
3	CHW	KCR	3	5	/boxes/KCA/KCA201903280.shtml	20
4	ARI	LAD	5	12	/boxes/LAN/LAN201903280.shtml	20
...	...	...	...	...	...	
12111	CIN	PIT	0	4	/boxes/PIT/PIT201510040.shtml	20
12112	OAK	SEA	2	3	/boxes/SEA/SEA201510040.shtml	20
12113	COL	SFG	7	3	/boxes/SFN/SFN201510040.shtml	20
12114	TOR	TBR	3	12	/boxes/TBA/TBA201510040.shtml	20

12115	LAA	TEX	2	9	/boxes/TEX/TEX201510040.shtml	20

12116 rows × 21 columns

By unpacking the Home_ and Away*Pitch columns, can get rid of the unreliable Home* and Away_Pitcher columns from the Odds data.

In [91]:
```
game_data['Away_Pitcher_Name']=game_data.Away_Pitch.apply(lambda t:t[0])
game_data['Away_Pitcher_ID']=game_data.Away_Pitch.apply(lambda t:t[1])

game_data['Home_Pitcher_Name']=game_data.Home_Pitch.apply(lambda t:t[0])
game_data['Home_Pitcher_ID']=game_data.Home_Pitch.apply(lambda t:t[1])
game_data.Home_Pitcher=game_data.Home_Pitch
game_data.Away_Pitcher=game_data.Away_Pitch
game_data.drop(['Away_Pitch','Home_Pitch'],inplace=True,axis=1,errors='ignore
```

In [92]:
```
print(game_data.columns)
game_data
```

```
Index(['Away_Team', 'Home_Team', 'Away_Score', 'Home_Score', 'link', 'date',
       'KEY', 'CloseOU', 'Close_Odds', 'year', 'month', 'Home_Pitcher',
       'Away_Pitcher', 'OVER', 'Total_Runs', 'day_or_night', 'attendance',
       'cli', 'other', 'Away_Pitcher_Name', 'Away_Pitcher_ID',
       'Home_Pitcher_Name', 'Home_Pitcher_ID'],
      dtype='object')
```

Out[92]:

	Away_Team	Home_Team	Away_Score	Home_Score	link	d
0	SEA	OAK	9	7	/boxes/OAK/OAK201903200.shtml	20
1	SEA	OAK	5	4	/boxes/OAK/OAK201903210.shtml	20 03
2	PIT	CIN	3	5	/boxes/CIN/CIN201903280.shtml	20

3	CHW	KCR	3	5	/boxes/KCA/KCA201903280.shtml	20
4	ARI	LAD	5	12	/boxes/LAN/LAN201903280.shtml	20
...	...	...	...	...	...	
12111	CIN	PIT	0	4	/boxes/PIT/PIT201510040.shtml	20
12112	OAK	SEA	2	3	/boxes/SEA/SEA201510040.shtml	20
12113	COL	SFG	7	3	/boxes/SFN/SFN201510040.shtml	20
12114	TOR	TBR	3	12	/boxes/TBA/TBA201510040.shtml	20
12115	LAA	TEX	2	9	/boxes/TEX/TEX201510040.shtml	20

12116 rows × 23 columns

In [93]:
```
game_data.to_pickle('../Merged DFs/Games_AllData.pkl')
```

Collect Annual Team Stats from Baseball Reference (Already Downloaded)

Since the data ranges from 2015-2019, we only need to grab these stats from 2014-18.

For Batting, Adv. Batting, Pitching, Starting/Relief Pitching and Fielding, the process was the same: 1) Load the statistics for each year and combine.

2) Map the team label to match with the dataframe built in the above section.

3) Adjust &/Or Calculate Features: Making sure that data is per game; combine features when possible; drop columns that either do not provide additional information to the number of runs scored/given up.

4) Create multindex w/ Team & Year.

In [11]:
```
TEAMS={}
with open('../Teams.txt') as f:
    for r in f.readlines():
        t=r.split(',')
        TEAMS[t[1].strip()]=t[0]
TEAMS['Los Angeles Angels of Anaheim']='LAA' #Angels have multiple names (Ana
```

In [12]:
```
team_Batting_splits={}

for yr in ['2014','2015','2016','2017','2018']:
    with open('../Raw Data/MLB_'+yr+'.txt') as f:
        keys=f.readline()[:-3].split(',')
        for l in f.readlines():
            d=dict(zip(keys,l[:-3].split(',')))
            d['Yr']=yr
            team_Batting_splits[d['Tm']+yr]=d
```

In [13]:
```
team_Batting_splits=pd.DataFrame.from_dict(team_Batting_splits,orient='index'
team_Batting_splits=team_Batting_splits.dropna()
```

In [14]:
```
team_Batting_splits=team_Batting_splits.reset_index().drop(['index','R/G','G'
team_Batting_splits=team_Batting_splits.set_index(['Tm','Yr'],drop=True).sort
print(team_Batting_splits.columns)
team_Batting_splits
```

```
Index(['#Bat', 'BatAge', 'PA', 'AB', 'R', 'H', '2B', '3B', 'HR', 'RBI', 'SB',
       'CS', 'BB', 'SO', 'BA', 'OBP', 'SLG', 'OPS', 'OPS+', 'TB', 'GDP', 'HBP'
,
       'SH', 'SF', 'IBB', 'LO'],
      dtype='object')
```

Out[14]:

Tm	Yr	#Bat	BatAge	PA	AB	R	H	2B	3B	HR	RBI	...	SLG	OPS	OPS+
ARI	**2014**	52	27.6	6089	5552	615	1379	259	47	118	573	...	.376	.678	88
	2015	50	26.5	6276	5649	720	1494	289	48	154	680	...	.414	.738	97
	2016	50	26.7	6260	5665	752	1479	285	56	190	709	...	.432	.752	93
	2017	45	28.3	6224	5525	812	1405	314	39	220	776	...	.445	.774	94
	2018	49	29.2	6157	5460	693	1283	259	50	176	658	...	.397	.707	86
...	**...**	...	...	...	...	...	...	...	...	...	...	...	...	...	...
WSN	**2014**	40	28.7	6216	5542	686	1403	265	27	152	635	...	.393	.714	96
	2015	44	28.4	6117	5428	703	1363	265	13	177	665	...	.403	.724	96
	2016	43	28.8	6201	5490	763	1403	268	29	203	735	...	.426	.751	96
	2017	49	29.2	6214	5553	819	1477	311	31	215	796	...	.449	.782	99
	2018	53	27.6	6288	5517	771	1402	284	25	191	737	...	.419	.753	98

150 rows × 26 columns

In [15]:

```
team_Batting_splits=team_Batting_splits.astype(float).apply(lambda x:x/162)
for stat in ['#Bat','BatAge','H','BA','OBP','SLG','OPS','OPS+']:
    team_Batting_splits[stat]=162*team_Batting_splits[stat]
team_Batting_splits
```

Out[15]:

Tm	Yr	#Bat	BatAge	PA	AB	R	H	2B	3B	
ARI	**2014**	52.0	27.6	37.586420	34.271605	3.796296	1379.0	1.598765	0.290123	0.72
	2015	50.0	26.5	38.740741	34.870370	4.444444	1494.0	1.783951	0.296296	0.95
	2016	50.0	26.7	38.641975	34.969136	4.641975	1479.0	1.759259	0.345679	1.17
	2017	45.0	28.3	38.419753	34.104938	5.012346	1405.0	1.938272	0.240741	1.35
	2018	49.0	29.2	38.006173	33.703704	4.277778	1283.0	1.598765	0.308642	1.08
...	**...**	...	...	...	...	...	...	...	...	
WSN	**2014**	40.0	28.7	38.370370	34.209877	4.234568	1403.0	1.635802	0.166667	0.93
	2015	44.0	28.4	37.759259	33.506173	4.339506	1363.0	1.635802	0.080247	1.09
	2016	43.0	28.8	38.277778	33.888889	4.709877	1403.0	1.654321	0.179012	1.25
	2017	49.0	29.2	38.358025	34.277778	5.055556	1477.0	1.919753	0.191358	1.32
	2018	53.0	27.6	38.814815	34.055556	4.759259	1402.0	1.753086	0.154321	1.17

150 rows × 26 columns

In an effort to consolidate the # of features: create steal attempts (see EDA section), sacs (for scoring runs, the difference between sac flies & sac hits, there is little difference), and free bases (the majority will be from walks anyways).

Then drop the components of these new features as well as some more complex statistics that are basically a function of other variables.

In [16]:
```
team_Batting_splits['Steal Att.']=team_Batting_splits.CS+team_Batting_splits.
team_Batting_splits['Sacs']=team_Batting_splits.SH+team_Batting_splits.SF
team_Batting_splits['Free_Bases']=team_Batting_splits.BB+team_Batting_splits.
```

In [17]:
```
team_Batting_splits=team_Batting_splits.drop(['CS','SB','SH','SF','BB','IBB',
team_Batting_splits=team_Batting_splits.drop(['OBP','OPS','SLG','TB'],axis=1,
team_Batting_splits
```

Out[17]:

		#Bat	BatAge	PA	R	H	2B	3B	HR	
Tm	**Yr**									
ARI	**2014**	52.0	27.6	37.586420	3.796296	1379.0	1.598765	0.290123	0.728395	3.537
	2015	50.0	26.5	38.740741	4.444444	1494.0	1.783951	0.296296	0.950617	4.197
	2016	50.0	26.7	38.641975	4.641975	1479.0	1.759259	0.345679	1.172840	4.376
	2017	45.0	28.3	38.419753	5.012346	1405.0	1.938272	0.240741	1.358025	4.790
	2018	49.0	29.2	38.006173	4.277778	1283.0	1.598765	0.308642	1.086420	4.061
...	**...**	...	...	...	...	...	...	...	...	
WSN	**2014**	40.0	28.7	38.370370	4.234568	1403.0	1.635802	0.166667	0.938272	3.919
	2015	44.0	28.4	37.759259	4.339506	1363.0	1.635802	0.080247	1.092593	4.104
	2016	43.0	28.8	38.277778	4.709877	1403.0	1.654321	0.179012	1.253086	4.537
	2017	49.0	29.2	38.358025	5.055556	1477.0	1.919753	0.191358	1.327160	4.913
	2018	53.0	27.6	38.814815	4.759259	1402.0	1.753086	0.154321	1.179012	4.549

150 rows × 17 columns

In [18]:
```
team_Batting_splits.to_pickle('../Base DFs/team_Batting_years.pkl')
```

In [71]:
```
team_StartP_splits=pd.read_excel('../Raw Data/'+'2014'+'_StartingPitching.xls
team_StartP_splits['Year']='2014'
for yr in ['2015','2016','2017','2018']:
    d=pd.read_excel('../Raw Data/'+yr+'_StartingPitching.xls')
    d['Year']=yr
    team_StartP_splits=team_StartP_splits.append(d.loc[0:29])
team_StartP_splits.Tm=team_StartP_splits.Tm.map(TEAMS)
```

In [72]:
```
print(team_StartP_splits.columns)
team_StartP_splits
```

```
Index(['Tm', 'G', 'GS', 'Wgs', 'Lgs', 'ND', 'Wchp', 'Ltuf', 'Wtm', 'Ltm',
       'tmW-L%', 'Wlst', 'Lsv', 'CG', 'SHO', 'QS', 'QS%', 'GmScA', 'Best',
       'Wrst', 'BQR', 'BQS', 'sDR', 'lDR', 'RS/GS', 'RS/IP', 'IP/GS', 'Pit/GS'
,
       '<80', '80-99', '100-119', '≥120', 'Max', 'Year'],
      dtype='object')
```

Out[72]:

	Tm	G	GS	Wgs	Lgs	ND	Wchp	Ltuf	Wtm	Ltm	...	RS/GS	RS/IP	IP/GS	Pit/GS
0	ARI	162	162	41	69	52	13	17	64	98	...	3.80	3.30	5.80	91
1	ATL	162	162	58	60	44	9	31	79	83	...	3.50	3.40	6.30	98
2	BAL	162	162	68	45	49	26	12	96	66	...	4.40	4.20	5.90	98
3	BOS	162	162	50	64	48	8	15	71	91	...	3.90	3.80	6.00	98
4	CHC	162	162	50	63	49	10	13	73	89	...	3.80	3.70	5.70	94
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25	STL	162	162	59	46	57	15	9	88	74	...	4.70	4.40	5.50	91
26	TBR	162	162	35	36	91	12	7	90	72	...	4.50	3.80	3.90	63
27	TEX	162	162	43	68	51	20	10	67	95	...	4.60	4.20	5.20	86
28	TOR	162	162	39	67	56	11	10	73	89	...	4.40	4.00	5.20	89
29	WSN	162	162	55	53	54	13	18	82	80	...	4.80	4.50	5.70	94

150 rows × 34 columns

Dropping columns that do not directly relate to giving up runs in future games (i.e. # of wins provides no additional inforamtion to allowing runs that runs scored, etc might provide.)

In [73]:

```
team_StartP_splits.drop(['G','GS','Wgs','Lgs','ND','Wtm','Ltm','tmW-L%','Wlst
                         'QS','Best','Wrst','BQR','BQS','sDR','lDR','RS/GS',
                         'RS/IP','<80','80-99','100-119','≥120','Max'
                        ]
                        ,axis=1,inplace=True,errors='ignore')
team_StartP_splits.drop(['GmScA'],axis=1,inplace=True,errors='ignore')  #Due
print(team_StartP_splits.columns)
team_StartP_splits
```

```
Index(['Tm', 'Wchp', 'Ltuf', 'CG', 'SHO', 'QS%', 'IP/GS', 'Pit/GS', 'Year'], d
type='object')
```

Out[73]:

	Tm	Wchp	Ltuf	CG	SHO	QS%	IP/GS	Pit/GS	Year
0	ARI	13	17	2	1	0.46	5.80	91	2014
1	ATL	9	31	5	2	0.68	6.30	98	2014
2	BAL	26	12	3	2	0.48	5.90	98	2014
3	BOS	8	15	3	2	0.54	6.00	98	2014
4	CHC	10	13	1	1	0.49	5.70	94	2014
...	...	...	...	...	...	...	...	...	...
25	STL	15	9	1	1	0.42	5.50	91	2018
26	TBR	12	7	0	0	0.24	3.90	63	2018
27	TEX	20	10	1	0	0.31	5.20	86	2018
28	TOR	11	10	0	0	0.35	5.20	89	2018
29	WSN	13	18	2	1	0.46	5.70	94	2018

150 rows × 9 columns

In [74]:

```
for stat in ['Wchp','Ltuf','CG','SHO']:
    team_StartP_splits[stat]=team_StartP_splits[stat]/162
team_StartP_splits
```

Out[74]:

	Tm	Wchp	Ltuf	CG	SHO	QS%	IP/GS	Pit/GS	Year
0	ARI	0.08	0.10	0.01	0.01	0.46	5.80	91	2014
1	ATL	0.06	0.19	0.03	0.01	0.68	6.30	98	2014
2	BAL	0.16	0.07	0.02	0.01	0.48	5.90	98	2014
3	BOS	0.05	0.09	0.02	0.01	0.54	6.00	98	2014
4	CHC	0.06	0.08	0.01	0.01	0.49	5.70	94	2014
...	...	...	...	...	...	...	...	...	...
25	STL	0.09	0.06	0.01	0.01	0.42	5.50	91	2018
26	TBR	0.07	0.04	0.00	0.00	0.24	3.90	63	2018
27	TEX	0.12	0.06	0.01	0.00	0.31	5.20	86	2018
28	TOR	0.07	0.06	0.00	0.00	0.35	5.20	89	2018
29	WSN	0.08	0.11	0.01	0.01	0.46	5.70	94	2018

150 rows × 9 columns

In [75]:
```
team_StartP_splits=team_StartP_splits.set_index(['Tm','Year']).sort_index(0)
team_StartP_splits=team_StartP_splits.rename(lambda s:'Start_'+s,axis=1)
team_StartP_splits
```

Out[75]:

Tm	Year	Start_Wchp	Start_Ltuf	Start_CG	Start_SHO	Start_QS%	Start_IP/GS	Start_Pit/(
ARI	**2014**	0.08	0.10	0.01	0.01	0.46	5.80	
	2015	0.08	0.07	0.01	0.01	0.43	5.50	
	2016	0.06	0.05	0.01	0.01	0.38	5.50	
	2017	0.09	0.07	0.01	0.01	0.51	5.80	
	2018	0.07	0.07	0.01	0.01	0.48	5.70	
...	**...**	...	...	...	...	...	...	
WSN	**2014**	0.04	0.11	0.03	0.02	0.65	6.20	
	2015	0.07	0.11	0.02	0.02	0.56	6.00	
	2016	0.07	0.07	0.01	0.00	0.57	5.90	
	2017	0.06	0.10	0.02	0.01	0.61	6.00	
	2018	0.08	0.11	0.01	0.01	0.46	5.70	

150 rows × 7 columns

In [76]:
```
team_StartP_splits.to_pickle('../Base DFs/team_StartingP_years.pkl')
```

In [124…
```
team_ReliefP_splits=pd.read_excel('../Raw Data/'+'2014'+'_ReliefPitching.xls'
team_ReliefP_splits['Year']='2014'
for yr in ['2015','2016','2017','2018']:
    d=pd.read_excel('../Raw Data/'+yr+'_ReliefPitching.xls')
    d['Year']=yr
    team_ReliefP_splits=team_ReliefP_splits.append(d.loc[0:29])
team_ReliefP_splits.Tm=team_ReliefP_splits.Tm.map(TEAMS)
```

In [125…
```
print(team_ReliefP_splits.columns)
team_ReliefP_splits
```

```
Index(['Tm', 'G', 'GR', 'GF', 'Wgr', 'Lgr', 'SVOpp', 'SV', 'BSv', 'SV%',
       'SVSit', 'Hold', 'IR', 'IS', 'IS%', '1stIP', 'aLI', 'LevHi', 'LevMd',
       'LevLo', 'Ahd', 'Tie', 'Bhd', 'Runr', 'Empt', '>3o', '<3o', 'IPmult',
       '0DR', 'Out/GR', 'Pit/GR', 'Year'],
      dtype='object')
```

Out[125…

	Tm	G	GR	GF	Wgr	Lgr	SVOpp	SV	BSv	SV%	...	Bhd	Runr	Empt	>3o	<3o
0	ARI	162	488	160	23	29	56	35	21	0.63	...	229	127	361	83	143
1	ATL	162	472	157	21	23	67	54	13	0.81	...	187	136	336	63	158
2	BAL	162	479	159	28	21	72	53	19	0.74	...	150	148	331	125	135
3	BOS	162	493	159	21	27	54	36	18	0.67	...	213	143	350	89	150
4	CHC	162	537	161	23	26	58	37	21	0.64	...	226	136	401	90	153
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25	STL	162	565	161	29	28	65	43	22	0.66	...	231	166	399	109	175
26	TBR	162	553	162	55	36	74	52	22	0.70	...	129	174	379	204	157
27	TEX	162	506	161	24	27	56	42	14	0.75	...	232	143	363	130	139
28	TOR	162	590	162	34	22	60	39	21	0.65	...	317	164	426	116	162
29	WSN	162	563	160	27	27	54	40	14	0.74	...	232	135	428	75	177

150 rows × 32 columns

In [126…

```
team_ReliefP_splits.drop(['G','GR','GF','Wgr','Lgr','SVOpp','SV',
                          'SV%','Hold','IR','IS','1stIP','LevHi',
                          'LevMd','LevLo','Ahd','Tie','Bhd','Runr',
                          'Empt','>3o','<3o','Out/GR','Pit/GR'
                         ]
                         ,axis=1,inplace=True,errors='ignore')
team_ReliefP_splits.drop(['IPmult'],axis=1,inplace=True,errors='ignore') #Due
print(team_ReliefP_splits.columns)
team_ReliefP_splits
```

```
Index(['Tm', 'BSv', 'SVSit', 'IS%', 'aLI', '0DR', 'Year'], dtype='object')
```

Out[126…

	Tm	BSv	SVSit	IS%	aLI	0DR	Year
0	ARI	21	134	0.33	1.02	94	2014
1	ATL	13	150	0.24	1.08	122	2014
2	BAL	19	171	0.25	1.05	92	2014
3	BOS	18	123	0.30	1.00	108	2014
4	CHC	21	144	0.30	1.01	104	2014
…	…	…	…	…	…	…	…
25	STL	22	156	0.31	1.02	105	2018
26	TBR	22	206	0.32	1.03	122	2018
27	TEX	14	147	0.34	0.92	71	2018
28	TOR	21	149	0.31	0.97	111	2018
29	WSN	14	138	0.24	0.97	126	2018

150 rows × 7 columns

In [127…

```
for stats in ['BSv','SVSit','0DR']:
    team_ReliefP_splits[stats]=team_ReliefP_splits[stats]/162
```

In [128…

```
team_ReliefP_splits=team_ReliefP_splits.set_index(['Tm','Year']).sort_index(0
team_ReliefP_splits=team_ReliefP_splits.rename(lambda s:'Relief_'+s,axis=1)
team_ReliefP_splits
```

Out[128…

Tm	Year	Relief_BSv	Relief_SVSit	Relief_IS%	Relief_aLI	Relief_0DR
ARI	**2014**	0.13	0.83	0.33	1.02	0.58
	2015	0.13	0.97	0.32	0.99	0.64
	2016	0.14	0.81	0.37	1.01	0.80
	2017	0.12	1.00	0.24	0.99	0.72
	2018	0.17	1.17	0.32	1.06	0.88
…	**…**	…	…	…	…	…
WSN	**2014**	0.10	0.97	0.29	0.98	0.41
	2015	0.14	0.88	0.31	0.99	0.53
	2016	0.09	1.02	0.25	1.01	0.73
	2017	0.11	0.98	0.26	0.99	0.57
	2018	0.09	0.85	0.24	0.97	0.78

150 rows × 5 columns

In [129…
```
team_ReliefP_splits.to_pickle('../Base DFs/team_ReliefP_years.pkl')
```

In [31]:
```
team_AdvBatting_splits=pd.read_excel('../Raw Data/'+'2014'+'_AdvBatting.xls')
team_AdvBatting_splits['Year']='2014'
for yr in ['2015','2016','2017','2018']:
    d=pd.read_excel('../Raw Data/'+yr+'_AdvBatting.xls')
    d['Year']=yr
    team_AdvBatting_splits=team_AdvBatting_splits.append(d.loc[0:29])
team_AdvBatting_splits.Tm=team_AdvBatting_splits.Tm.map(TEAMS)
```

In [32]:
```
print(team_AdvBatting_splits.columns)
team_AdvBatting_splits
```

```
Index(['Tm', 'R/G', 'Outs', 'RC', 'RC/G', 'AIR', 'BAbip', 'BA', 'lgBA', 'OBP',
       'lgOBP', 'SLG', 'lgSLG', 'OPS', 'lgOPS', 'OPS+', 'OWn%', 'BtRuns',
       'BtWins', 'TotA', 'SecA', 'ISO', 'PwrSpd', 'Year'],
      dtype='object')
```

Out[32]:

	Tm	R/G	Outs	RC	RC/G	AIR	BAbip	BA	lgBA	OBP	...	lgOPS	OPS+	OWn%
0	ARI	3.80	4413	651	4.0	96	0.293	0.248	0.259	0.302	...	0.722	88	0.472
1	ATL	3.54	4386	630	3.9	93	0.298	0.241	0.254	0.305	...	0.709	88	0.460
2	BAL	4.35	4365	742	4.6	95	0.296	0.256	0.257	0.311	...	0.717	104	0.526
3	BOS	3.91	4431	674	4.1	95	0.297	0.244	0.258	0.316	...	0.720	91	0.475
4	CHC	3.79	4425	661	4.0	96	0.296	0.239	0.259	0.300	...	0.722	89	0.478
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25	STL	4.69	4347	759	4.7	99	0.294	0.249	0.250	0.321	...	0.734	99	0.512
26	TBR	4.42	4311	766	4.8	97	0.317	0.258	0.246	0.333	...	0.726	105	0.512
27	TEX	4.55	4351	727	4.5	108	0.292	0.240	0.261	0.318	...	0.766	89	0.484
28	TOR	4.38	4331	728	4.5	98	0.286	0.244	0.248	0.312	...	0.729	102	0.486
29	WSN	4.76	4333	829	5.2	107	0.297	0.254	0.260	0.335	...	0.763	98	0.554

150 rows × 24 columns

In [33]:
```
team_AdvBatting_splits.drop(['R/G','Outs','RC','RC/G','BA','lgBA','OBP',
                             'lgOBP','SLG','lgSLG','OPS','lgOPS','OPS+',
                             'BtWins','TotA','SecA','ISO'
                            ]
                            ,axis=1,inplace=True,errors='ignore')
```

In [34]:
```
team_AdvBatting_splits=team_AdvBatting_splits.set_index(['Tm','Year']).sort_i
print(team_AdvBatting_splits.columns)
team_AdvBatting_splits
```

```
Index(['AIR', 'BAbip', 'OWn%', 'BtRuns', 'PwrSpd'], dtype='object')
```

Out[34]:

Tm	Year	AIR	BAbip	OWn%	BtRuns	PwrSpd
ARI	**2014**	96	0.293	0.472	-101.3	99.5
	2015	103	0.316	0.540	-28.2	142.2
	2016	111	0.315	0.522	-72.0	159.2
	2017	116	0.306	0.533	-57.9	140.3
	2018	106	0.286	0.480	-116.8	109.1
...	**...**	...	...	...	...	...
WSN	**2014**	98	0.303	0.533	-28.1	121.4
	2015	101	0.300	0.516	-30.7	86.2
	2016	107	0.293	0.529	-31.4	151.6
	2017	112	0.311	0.548	-12.4	143.8
	2018	107	0.297	0.554	-15.7	146.6

150 rows × 5 columns

In [35]:
```
team_AdvBatting_splits.to_pickle('../Base DFs/team_AdvBatting_years.pkl')
```

In [36]:
```
team_Fielding_splits=pd.read_excel('../Raw Data/'+'2014'+'_Fielding.xls').loc
team_Fielding_splits['Year']='2014'
for yr in ['2015','2016','2017','2018']:
    d=pd.read_excel('../Raw Data/'+yr+'_Fielding.xls')
    d['Year']=yr
    team_Fielding_splits=team_Fielding_splits.append(d.loc[0:29])
team_Fielding_splits.Tm=team_Fielding_splits.Tm.map(TEAMS)
```

In [37]:
```
print(team_Fielding_splits.columns)
team_Fielding_splits
```

```
Index(['Tm', '#Fld', 'RA/G', 'DefEff', 'G', 'GS', 'CG', 'Inn', 'Ch', 'PO', 'A'
,
       'E', 'DP', 'Fld%', 'Rtot', 'Rtot/yr', 'Rdrs', 'Rdrs/yr', 'Rgood',
       'Year'],
      dtype='object')
```

Out[37]:

	Tm	#Fld	RA/G	DefEff	G	GS	CG	Inn	Ch	PO	A	E	DP	Fld%	Rt
0	ARI	52	4.58	0.673	162	1458	1209	12999	6079	4333	1645	101	147	0.983	
1	ATL	39	3.69	0.687	162	1458	1191	13095	5977	4365	1527	85	143	0.986	
2	BAL	43	3.66	0.706	162	1458	1124	13152	6075	4384	1604	87	156	0.986	
3	BOS	55	4.41	0.685	162	1458	1136	13191	6099	4397	1610	92	155	0.985	
4	CHC	48	4.36	0.680	162	1458	1120	13170	6161	4390	1668	103	137	0.983	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
25	STL	49	4.27	0.685	162	1458	1031	13098	6039	4366	1540	133	151	0.978	-
26	TBR	54	3.99	0.708	162	1458	1101	13035	5913	4345	1483	85	136	0.986	
27	TEX	50	5.23	0.682	162	1458	1149	12879	5971	4293	1558	120	168	0.980	
28	TOR	63	5.14	0.678	162	1458	1116	12903	5849	4301	1447	101	138	0.983	-
29	WSN	53	4.21	0.703	162	1458	1077	13014	5765	4338	1363	64	115	0.989	

150 rows × 20 columns

In [38]:
```
team_Fielding_splits.drop(['#Fld','RA/G','G','GS','Inn','Ch',
                           'PO','A','E','DP','Rtot','Rdrs','Rdrs/yr'
                          ],
                          axis=1,inplace=True,errors='ignore')
```

In [39]:
```
team_Fielding_splits=team_Fielding_splits.set_index(['Tm','Year']).sort_index
team_Fielding_splits
```

Out[39]:

Tm	Year	DefEff	CG	Fld%	Rtot/yr	Rgood
ARI	**2014**	0.673	1209	0.983	0	-1
	2015	0.693	1107	0.986	3	9
	2016	0.665	1105	0.983	-3	11
	2017	0.687	1090	0.982	0	3
	2018	0.698	1119	0.988	3	14
...	**...**	...	...	...	...	...
WSN	**2014**	0.691	1167	0.984	3	5
	2015	0.685	1166	0.985	0	-2
	2016	0.700	1147	0.988	4	3
	2017	0.698	1118	0.985	1	-8
	2018	0.703	1077	0.989	0	4

150 rows × 5 columns

In [40]:
```
team_Fielding_splits.to_pickle('../Base DFs/team_Fielding_years.pkl')
```

In [140…
```
team_Pitching_splits=pd.read_excel('../Raw Data/'+'2014'+'_Pitching.xls').loc
team_Pitching_splits['Year']='2014'
for yr in ['2015','2016','2017','2018']:
    d=pd.read_excel('../Raw Data/'+yr+'_Pitching.xls')
    d['Year']=yr
    team_Pitching_splits=team_Pitching_splits.append(d.loc[0:29])
team_Pitching_splits.Tm=team_Pitching_splits.Tm.map(TEAMS)
```

In [141…
```
print(team_Pitching_splits.columns)
team_Pitching_splits
```

```
Index(['Tm', '#P', 'PAge', 'RA/G', 'W', 'L', 'W-L%', 'ERA', 'G', 'GS', 'GF',
       'CG', 'tSho', 'cSho', 'SV', 'IP', 'H', 'R', 'ER', 'HR', 'BB', 'IBB',
       'SO', 'HBP', 'BK', 'WP', 'BF', 'ERA+', 'FIP', 'WHIP', 'H9', 'HR9',
       'BB9', 'SO9', 'SO/W', 'LOB', 'Year'],
      dtype='object')
```

Out[141…

	Tm	#P	PAge	RA/G	W	L	W-L%	ERA	G	GS	...	ERA+	FIP	WHIP	H9	HR9
0	ARI	25	28.00	4.58	64	98	0.40	4.26	162	162	...	88	3.83	1.34	9.10	1.00
1	ATL	20	27.30	3.69	79	83	0.49	3.38	162	162	...	106	3.47	1.26	8.50	0.70
2	BAL	20	27.70	3.66	96	66	0.59	3.43	162	162	...	115	3.96	1.24	8.30	0.90
3	BOS	25	29.90	4.41	71	91	0.44	4.01	162	162	...	100	3.93	1.32	9.00	0.90
4	CHC	27	28.00	4.36	73	89	0.45	3.91	162	162	...	97	3.51	1.30	8.60	0.70
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25	STL	30	26.70	4.27	88	74	0.54	3.85	162	162	...	101	3.97	1.34	8.40	0.90
26	TBR	31	27.10	3.99	90	72	0.56	3.74	162	162	...	110	3.82	1.20	7.70	1.00
27	TEX	32	30.40	5.23	67	95	0.41	4.92	162	162	...	97	4.79	1.40	9.50	1.40
28	TOR	36	29.30	5.14	73	89	0.45	4.85	162	162	...	88	4.53	1.41	9.30	1.30
29	WSN	31	30.20	4.21	82	80	0.51	4.04	162	162	...	106	4.15	1.25	8.20	1.20

150 rows × 37 columns

In [142…

```
team_Pitching_splits.drop(['W','L','W-L%','ERA','G','GS','GF','SV',
                           'H','R','ER','HR','BB','IBB','HBP','BK',
                           'WP','ERA+','WHIP','BB9','SO9','CG'
                          ]
                          ,axis=1,inplace=True,errors='ignore')
team_Pitching_splits.drop(['SO/W'],axis=1,inplace=True,errors='ignore') #Due
```

In [143…

```
for stat in ['tSho','cSho','IP']:
    team_Pitching_splits[stat]=team_Pitching_splits[stat]/162
team_Pitching_splits=team_Pitching_splits.set_index(['Tm','Year']).sort_index
team_Pitching_splits=team_Pitching_splits.rename(lambda s:'All_'+s,axis=1)
team_Pitching_splits
```

Out[143…

Tm	Year	All_#P	All_PAge	All_RA/G	All_tSho	All_cSho	All_IP	All_SO	All_BF	All_FIP	A
ARI	**2014**	25	28.00	4.58	0.02	0.01	8.91	1278	6162	3.83	
	2015	27	27.10	4.40	0.07	0.01	9.05	1215	6257	4.21	
	2016	29	26.40	5.49	0.04	0.01	8.96	1318	6437	4.50	
	2017	23	28.70	4.07	0.07	0.01	8.90	1482	6072	3.80	
	2018	30	29.60	3.98	0.06	0.01	9.03	1448	6139	3.91	
…	**…**	…	…	…	…	…	…	…	…	…	
WSN	**2014**	18	28.30	3.43	0.12	0.02	9.08	1288	6020	3.17	
	2015	26	28.60	3.92	0.08	0.02	8.85	1342	5975	3.45	
	2016	24	29.10	3.78	0.07	0.00	9.01	1476	6036	3.58	
	2017	24	30.10	4.15	0.03	0.01	8.93	1457	6068	3.99	
	2018	31	30.20	4.21	0.04	0.01	8.93	1417	6087	4.15	

150 rows × 12 columns

In [144…
```
team_Pitching_splits.to_pickle('../Base DFs/team_Pitching_years.pkl')
```

In [145…
```
stat_DFs=[team_Batting_splits,
          team_AdvBatting_splits,
          team_Pitching_splits,
          team_StartP_splits,
          team_ReliefP_splits,
          team_Fielding_splits
         ]
annual_team_stats=pd.concat(stat_DFs,axis=1)
```

In [146…
```
annual_team_stats.to_pickle('../Merged DFs/team_annual_statistics.pkl')
```

Combine the annual data and game info to create the final dataset (FOR NOW).

In [147…
```
game_data=pd.read_pickle('../Merged DFs/Games_AllData.pkl')
```

In [148…
```
Away_Stats=game_data.apply(lambda r:annual_team_stats.loc[(r.Away_Team,str(r.
Home_Stats=game_data.apply(lambda r:annual_team_stats.loc[(r.Home_Team,str(r.
```

In [149…
```
stats=pd.merge(Away_Stats,Home_Stats,left_index=True,right_index=True,suffixe
```

In [150…
```
stats
```

Out[150…

	#Bat_A	BatAge_A	PA_A	R_A	H_A	2B_A	3B_A	HR_A	RBI_A	SO_A	…	Relief
0	53.00	29.80	37.57	4.18	1,402.00	1.58	0.20	1.09	3.98	7.54	…	
1	53.00	29.80	37.57	4.18	1,402.00	1.58	0.20	1.09	3.98	7.54	…	
2	48.00	27.80	37.44	4.27	1,381.00	1.79	0.23	0.97	4.10	7.59	…	
3	51.00	26.50	37.48	4.05	1,332.00	1.60	0.25	1.12	3.94	9.84	…	
4	49.00	29.20	38.01	4.28	1,283.00	1.60	0.31	1.09	4.06	9.01	…	
…	…	…	…	…	…	…	…	…	…	…	…	
12111	45.00	29.00	36.90	3.67	1,282.00	1.57	0.12	0.81	3.47	7.73	…	
12112	45.00	29.60	38.55	4.50	1,354.00	1.56	0.20	0.90	4.23	6.81	…	
12113	49.00	27.30	38.05	4.66	1,551.00	1.90	0.25	1.15	4.45	7.91	…	
12114	55.00	29.20	38.07	4.46	1,435.00	1.74	0.15	1.09	4.26	7.10	…	
12115	54.00	29.30	38.80	4.77	1,464.00	1.88	0.19	0.96	4.50	7.81	…	

12116 rows × 102 columns

In [151…
```
dataset=pd.merge(game_data[['OVER','CloseOU','cli']],stats,left_index=True,ri
```

In [152…
```
dataset.to_pickle('../Merged DFs/Final_Database.pkl')
```

EDA

Team Stats Analysis

The team statistics data is the only quantitative data we have but given the # of these features, we are really limited to doing some preliminary exploration.

Pitching & Fielding

In [108…
```
pd.options.display.float_format="{:,.2f}".format
APitching=pd.read_pickle('../Base DFs/team_Pitching_years.pkl')
SPitching=pd.read_pickle('../Base DFs/team_StartingP_years.pkl')
RPitching=pd.read_pickle('../Base DFs/team_ReliefP_years.pkl')
Fielding=pd.read_pickle('../Base DFs/team_Fielding_years.pkl')
Pitching=pd.concat([APitching,SPitching,RPitching],axis=1)
Defense=pd.concat([Pitching,Fielding],axis=1)
```

In [109…

```
print(APitching.describe(),end='\n\n')
print(SPitching.describe(),end='\n\n')
print(RPitching.describe(),end='\n\n')
print(Fielding.describe())
```

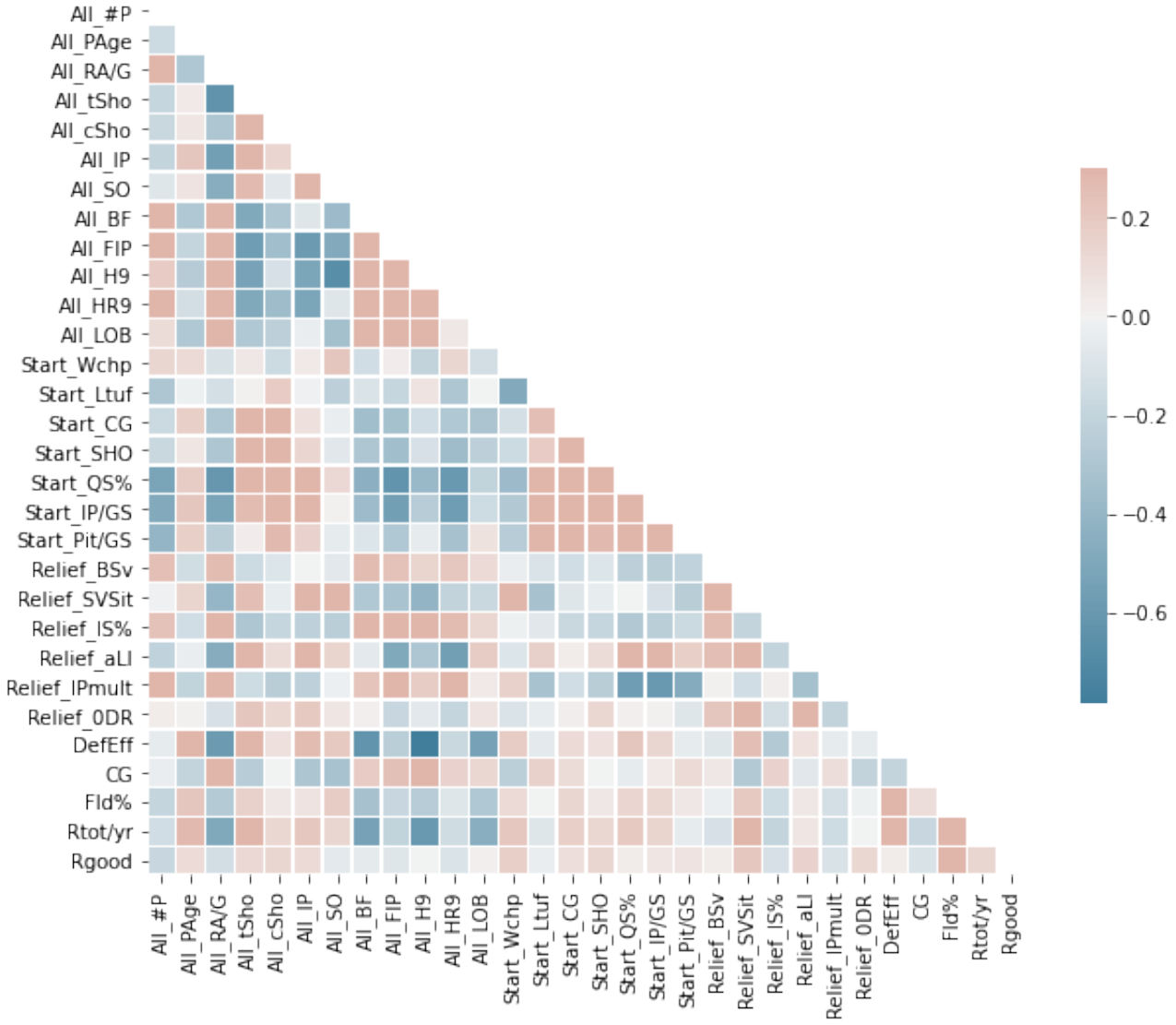
```
       All_#P  All_PAge  All_RA/G  All_tSho  All_cSho  All_IP   All_SO  \
count  150.00    150.00    150.00    150.00    150.00  150.00   150.00
mean    27.47     28.42      4.38      0.06      0.01    8.93 1,301.20
std      4.30      1.03      0.52      0.03      0.01    0.08   126.01
min     18.00     26.30      3.24      0.01      0.00    8.77 1,031.00
25%     24.00     27.72      3.99      0.04      0.00    8.87 1,215.25
50%     27.00     28.40      4.35      0.06      0.01    8.92 1,289.00
75%     31.00     29.00      4.75      0.07      0.01    8.99 1,376.25
max     41.00     31.70      5.52      0.14      0.03    9.19 1,687.00

        All_BF  All_FIP  All_H9  All_HR9  All_LOB
count   150.00   150.00  150.00   150.00   150.00
mean  6,150.47     4.08    8.68     1.09 1,099.90
std     106.72     0.42    0.59     0.20    53.67
min   5,866.00     3.17    6.90     0.70   966.00
25%   6,079.75     3.80    8.30     1.00 1,061.75
50%   6,154.00     4.06    8.70     1.10 1,107.50
75%   6,217.75     4.36    9.10     1.20 1,136.50
max   6,437.00     5.24   10.10     1.60 1,215.00

       Start_Wchp  Start_Ltuf  Start_CG  Start_SHO  Start_QS%  Start_IP/GS  \
count      150.00      150.00    150.00     150.00     150.00       150.00
mean         0.08        0.08      0.02       0.01       0.47         5.66
std          0.03        0.03      0.01       0.01       0.09         0.33
min          0.02        0.03      0.00       0.00       0.24         3.90
25%          0.06        0.06      0.01       0.00       0.41         5.50
50%          0.07        0.08      0.01       0.01       0.46         5.70
75%          0.10        0.10      0.02       0.01       0.53         5.90
max          0.16        0.19      0.07       0.03       0.68         6.30

       Start_Pit/GS
count        150.00
mean          92.23
std            4.46
min           63.00
25%           90.00
50%           92.00
75%           95.00
max          102.00

       Relief_BSv  Relief_SVSit  Relief_IS%  Relief_aLI  Relief_IPmult  \
count      150.00        150.00      150.00      150.00         150.00
mean         0.12          0.91        0.30        0.99           0.76
std          0.03          0.14        0.04        0.04           0.17
min          0.05          0.58        0.20        0.86           0.44
25%          0.10          0.82        0.28        0.96           0.64
50%          0.12          0.90        0.30        0.99           0.74
75%          0.14          1.00        0.32        1.01           0.86
max          0.20          1.35        0.40        1.10           1.44

       Relief_0DR
```

```
count      150.00
mean         0.65
std          0.12
min          0.36
25%          0.57
50%          0.63
75%          0.73
max          0.91
```

	DefEff	CG	Fld%	Rtot/yr	Rgood
count	150.00	150.00	150.00	150.00	150.00
mean	0.69	1,105.69	0.98	0.03	-0.25
std	0.01	56.46	0.00	3.03	5.20
min	0.66	910.00	0.98	-8.00	-14.00
25%	0.68	1,077.50	0.98	-2.00	-4.00
50%	0.69	1,114.00	0.98	0.00	-0.50
75%	0.70	1,143.75	0.99	2.00	3.00
max	0.73	1,234.00	0.99	9.00	14.00

In [116…

```
f, ax = plt.subplots(figsize=(10, 10))
corr = Defense.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
cmap = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5});
plt.savefig('../Images/CorrHeat_Defensive.jpg')
```



Some of the negative correlations between a starter's game score may be of concern with respect to autocorrel.

Batting

In [100…
```
Base_Batting=pd.read_pickle('../Base DFs/team_batting_years.pkl')
Adv_Batting=pd.read_pickle('../Base DFs/team_AdvBatting_years.pkl')
Batting=pd.concat([Base_Batting,Adv_Batting],axis=1)
```

In [101…
```
Base_Batting.describe()
```

Out[101…

	#Bat	BatAge	PA	R	H	2B	3B	HR	RBI	SO	BA
count	150.00	150.00	150.00	150.00	150.00	150.00	150.00	150.00	150.00	150.00	150.00
mean	49.51	28.33	37.97	4.38	1,394.73	1.70	0.18	1.09	4.17	8.03	0.25
std	4.78	1.07	0.58	0.44	66.98	0.15	0.06	0.23	0.42	0.74	0.01
min	39.00	25.40	36.45	3.30	1,199.00	1.37	0.03	0.59	3.09	6.01	0.23
25%	46.00	27.70	37.49	4.04	1,349.25	1.59	0.14	0.91	3.85	7.56	0.25
50%	49.00	28.30	37.95	4.35	1,396.00	1.69	0.18	1.09	4.14	8.03	0.25
75%	52.00	28.90	38.41	4.66	1,435.75	1.78	0.21	1.28	4.44	8.55	0.26
max	64.00	32.80	39.31	5.53	1,598.00	2.19	0.35	1.65	5.27	9.84	0.28

In [102…

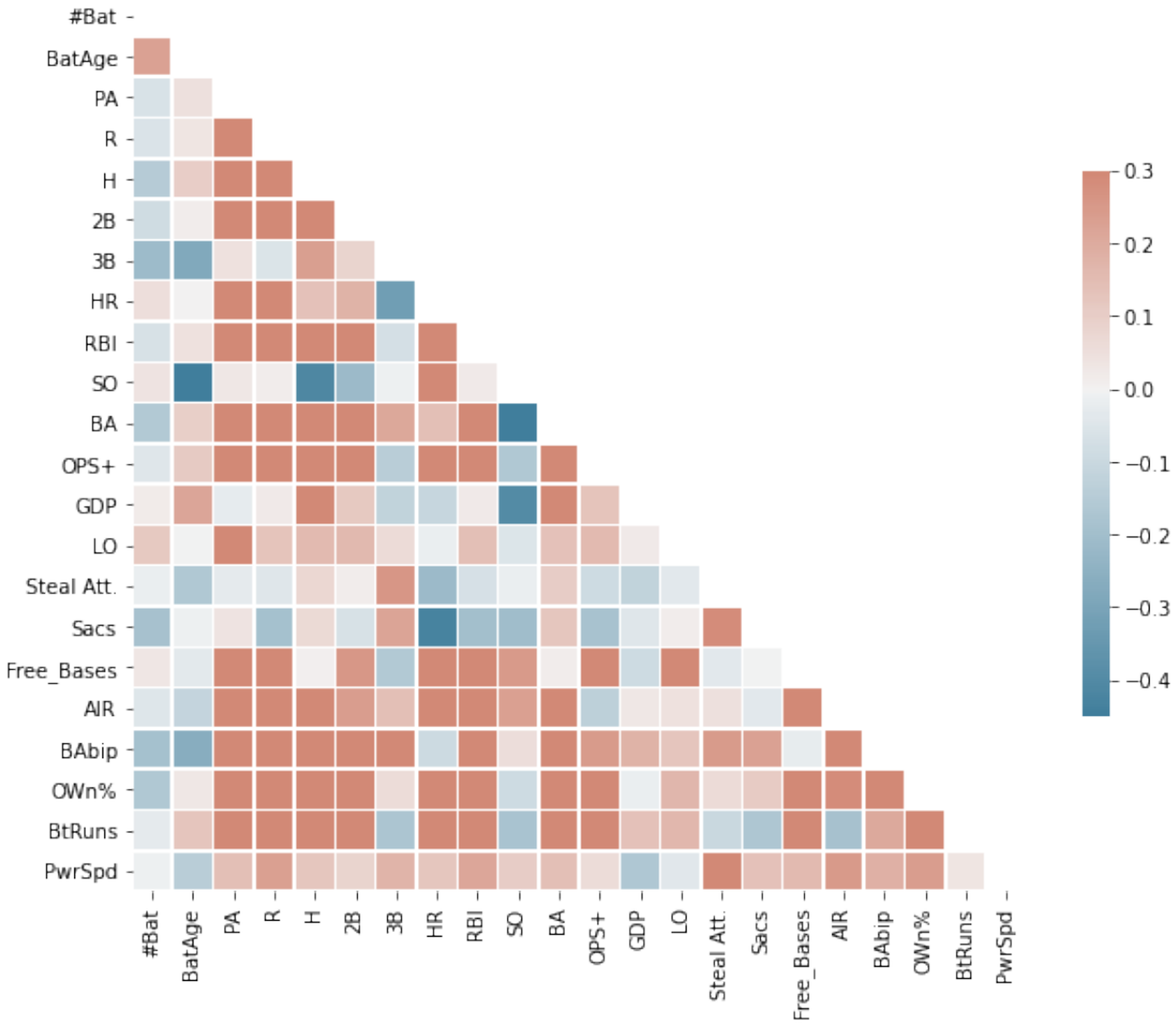
```
Adv_Batting.describe()
```

Out[102…

	AIR	BAbip	OWn%	BtRuns	PwrSpd
count	150.00	150.00	150.00	150.00	150.00
mean	100.58	0.30	0.50	-24.26	110.84
std	6.28	0.01	0.04	59.53	25.18
min	88.00	0.28	0.41	-151.60	35.30
25%	96.00	0.29	0.47	-66.38	94.08
50%	100.00	0.30	0.50	-26.90	108.10
75%	104.00	0.30	0.53	8.75	126.83
max	123.00	0.33	0.59	200.00	187.30

In [106…

```
f, ax = plt.subplots(figsize=(10, 10))
corr = Batting.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
cmap = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5});
plt.savefig('../Images/CorrHeat_Offensive.jpg')
```



Definetly no autocorrelation to worry about in this case.

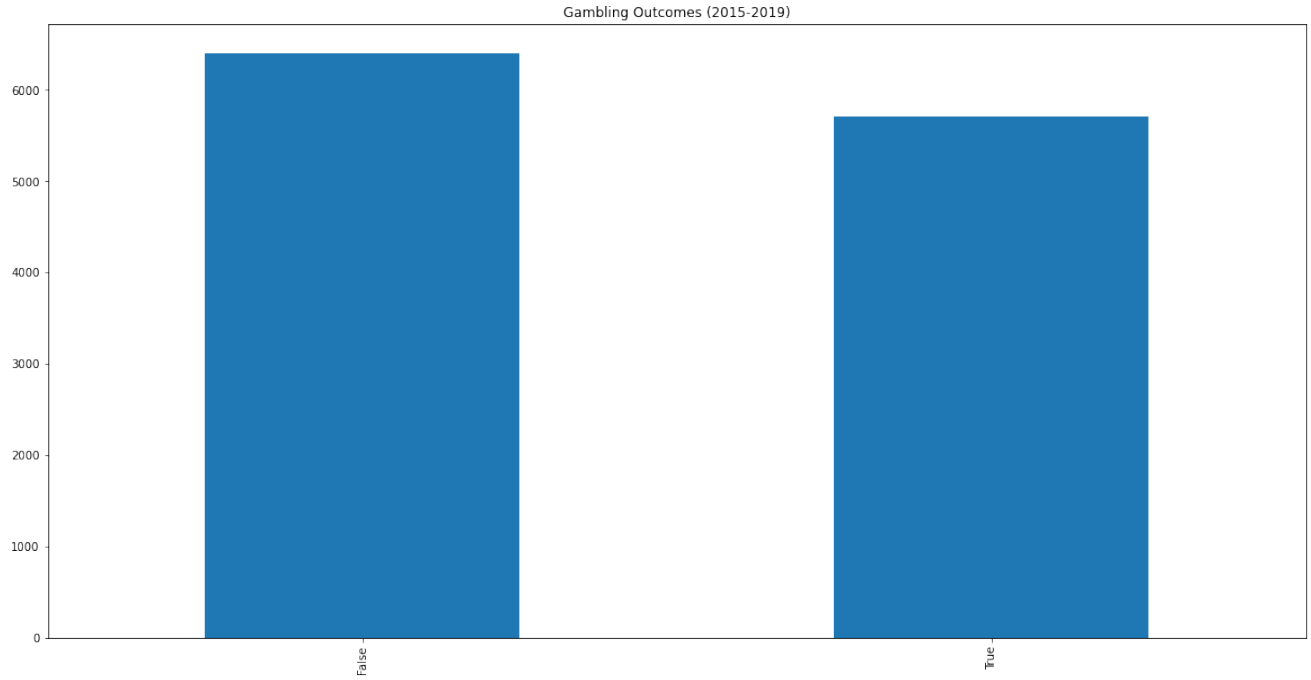
Additional Analysis w/ Game Data

It might be worth looking at info we might glean from the game information

In [86]:
```
game_data=pd.read_pickle('../Merged DFs/Games_AllData.pkl')
```

Target Variable-Balanced

In [115…
```
odds_split=game_data.OVER.value_counts()
odds_split
fig,ax=plt.subplots(figsize=(20,10))
ax.set_title('Gambling Outcomes (2015-2019)')
odds_split.plot.bar(ax=ax);
plt.savefig('../Images/Balance_Check.jpg')
```



Apprears well-balanced. False outcomes should be more likely since they include observations which neither the over nor under hit.

Pitching Experience

Value of O/U Line

In [94]:
```
by_line=game_data.groupby(['CloseOU']).OVER.agg(['sum','count','mean'])
by_line
```

Out[94]:

CloseOU	sum	count	mean
5.00	0	1	0.00
5.50	4	8	0.50
6.00	29	69	0.42
6.50	154	275	0.56
7.00	443	999	0.44
7.50	883	1661	0.53
8.00	847	1795	0.47
8.50	1118	2353	0.48
9.00	955	2198	0.43
9.50	552	1229	0.45
10.00	276	584	0.47
10.50	229	450	0.51
11.00	109	235	0.46
11.50	53	113	0.47
12.00	26	54	0.48
12.50	13	38	0.34
13.00	11	28	0.39
13.50	6	15	0.40
14.00	2	7	0.29
14.50	1	3	0.33
15.00	1	1	1.00

There is no evidence that the OVER hitting has any relationship to where the line was set.

By Team &/Or By Time

In [95]:
```
by_ATeam=game_data.groupby(['Away_Team']).OVER.agg(['sum','count','mean'])
by_HTeam=game_data.groupby(['Home_Team']).OVER.agg(['sum','count','mean'])
by_Team=by_ATeam+by_HTeam
by_Team['mean']=by_Team['sum']/by_Team['count']
by_Team
```

Out[95]:

	sum	count	mean

Away_Team			
ARI	387	808	0.48
ATL	392	808	0.49
BAL	376	806	0.47
BOS	400	809	0.49
CHC	363	810	0.45
CHW	375	806	0.47
CIN	391	808	0.48
CLE	366	807	0.45
COL	362	811	0.45
DET	388	805	0.48
HOU	367	809	0.45
KCR	376	808	0.47
LAA	359	810	0.44
LAD	377	810	0.47
MIA	393	805	0.49
MIL	368	808	0.46
MIN	395	809	0.49
NYM	394	804	0.49
NYY	388	805	0.48
OAK	395	810	0.49
PHI	377	807	0.47
PIT	388	809	0.48
SDP	404	807	0.50
SEA	389	807	0.48
SFG	374	809	0.46
STL	373	806	0.46
TBR	382	806	0.47
TEX	376	809	0.46
TOR	369	809	0.46
WSN	380	807	0.47

There is no evidence that the oddsmakers have any difficulty setting the line for a particular team over the entire period of observation.

In [112…
```
annual_splits=game_data.groupby(['year']).OVER.agg(['sum','count','mean'])
monthly_splits=game_data.groupby(['month']).OVER.agg(['sum','count','mean'])
print(annual_splits)
print(monthly_splits)
```

```
       sum  count  mean
year
2015  1174   2422  0.48
2016  1151   2420  0.48
2017  1129   2424  0.47
2018  1122   2427  0.46
2019  1136   2423  0.47
        sum  count  mean
month
3        42     92  0.46
4       886   1812  0.49
5      1001   2099  0.48
6       956   2020  0.47
7       833   1868  0.45
8      1000   2096  0.48
9       958   2025  0.47
10       36    104  0.35
```

There is no evidence that oddsmakers struggle with setting the line in any given month. This is somewhat surprising as one might expect that earlier in the season, it is harder to make proper estimations.

Similarly, there is no evidence that oddsmakers struggled to set proper lines in any particular year.

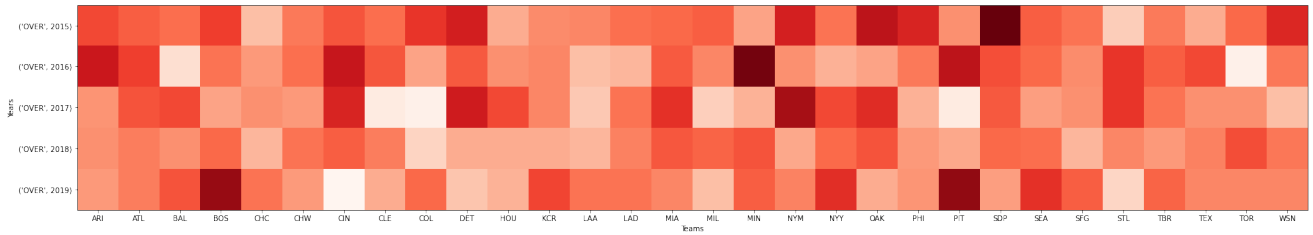
In [97]:
```
Hteams_years=game_data.pivot_table(['OVER'],index=['Home_Team'],columns=['yea
Ateams_years=game_data.pivot_table(['OVER'],index=['Away_Team'],columns=['yea
teams_years=(Ateams_years+Hteams_years)/2
teams_years.index.name='Team'
teams_years
```

Out[97]:

	OVER				
year	**2015**	**2016**	**2017**	**2018**	**2019**
Team					
ARI	0.50	0.53	0.45	0.46	0.45
ATL	0.49	0.51	0.49	0.47	0.47
BAL	0.48	0.40	0.50	0.46	0.49
BOS	0.51	0.48	0.44	0.48	0.57
CHC	0.43	0.45	0.46	0.43	0.48

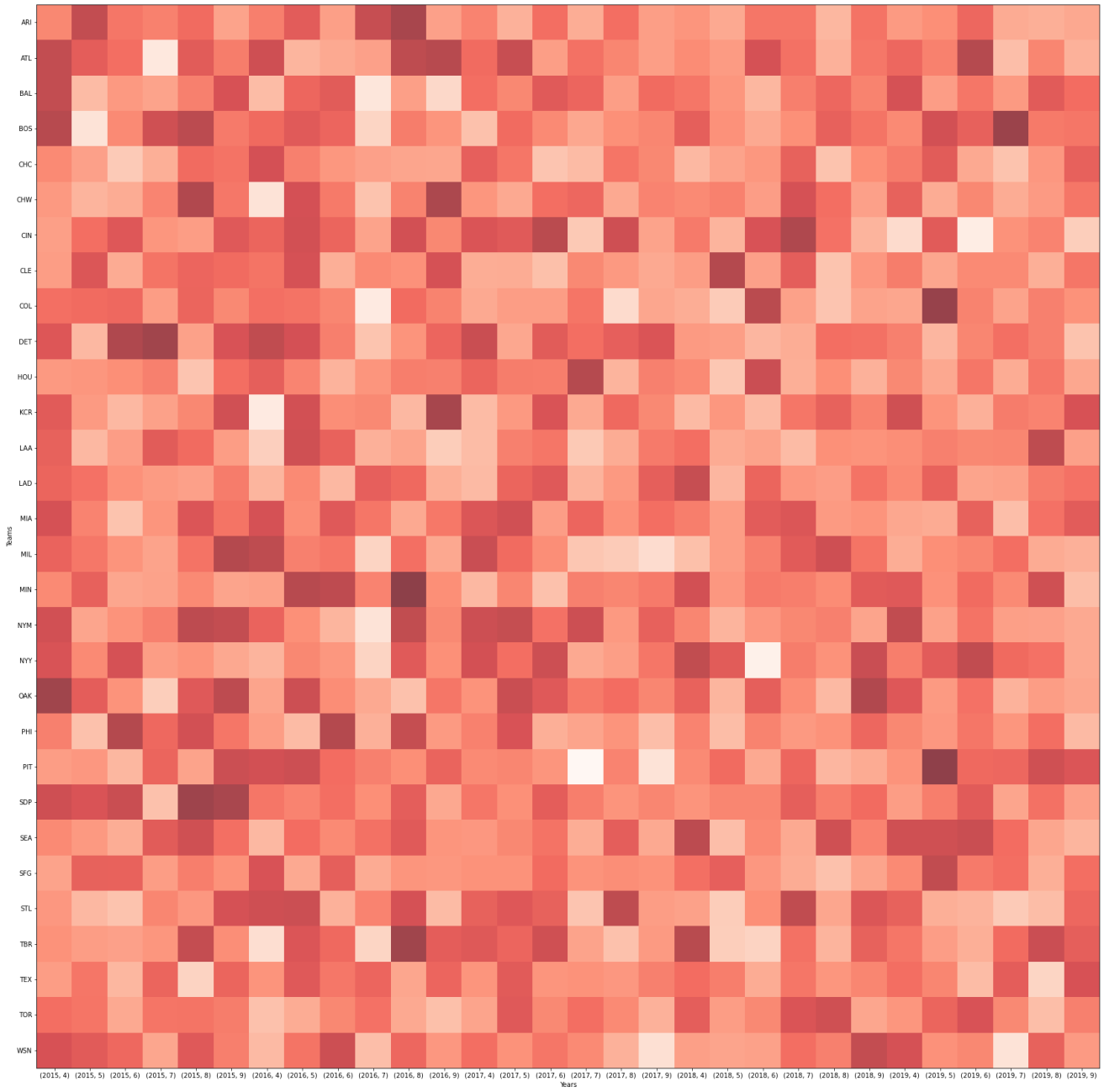
CHW	0.47	0.48	0.45	0.48	0.45
CIN	0.49	0.54	0.52	0.49	0.38
CLE	0.48	0.49	0.39	0.47	0.44
COL	0.51	0.44	0.38	0.41	0.48
DET	0.53	0.49	0.53	0.44	0.42
HOU	0.44	0.46	0.50	0.44	0.43
KCR	0.46	0.46	0.46	0.44	0.50
LAA	0.46	0.43	0.42	0.43	0.48
LAD	0.48	0.43	0.48	0.47	0.48
MIA	0.48	0.49	0.52	0.49	0.46
MIL	0.49	0.46	0.41	0.48	0.43
MIN	0.44	0.58	0.43	0.49	0.49
NYM	0.53	0.46	0.56	0.44	0.47
NYY	0.48	0.43	0.50	0.48	0.52
OAK	0.54	0.44	0.52	0.49	0.44
PHI	0.52	0.47	0.43	0.45	0.45
PIT	0.46	0.54	0.39	0.44	0.57
SDP	0.59	0.50	0.49	0.48	0.45
SEA	0.49	0.48	0.45	0.48	0.52
SFG	0.48	0.46	0.46	0.43	0.49
STL	0.42	0.51	0.51	0.46	0.41
TBR	0.47	0.49	0.48	0.45	0.48
TEX	0.44	0.50	0.46	0.47	0.46
TOR	0.48	0.38	0.46	0.50	0.46
WSN	0.52	0.47	0.43	0.47	0.46

In [114…
```
fig,ax=plt.subplots(figsize=(30,30))
indX=np.arange(teams_years.shape[0])
indY=np.arange(teams_years.shape[1])
im=ax.imshow(teams_years.transpose(),cmap='Reds')
ax.set_xlabel('Teams');
ax.set_ylabel('Years');
ax.set_xticks(indX);
ax.set_yticks(indY);
ax.set_yticklabels(teams_years.columns);
ax.set_xticklabels(teams_years.index);
plt.savefig('../Images/Overs_by_TeamYear.jpg')
```



In [154…
```
away_monthly_splits=game_data.groupby(['Away_Team','year','month'])['OVER'].m
home_monthly_splits=game_data.groupby(['Home_Team','year','month'])['OVER'].m
away_monthly_splits.index.names=['Team','year','month']
home_monthly_splits.index.names=['Team','year','month']
monthly_team_splits=(away_monthly_splits+home_monthly_splits)/2
monthly_pivot=monthly_team_splits.unstack().unstack().swaplevel('month','year
monthly_pivot=monthly_pivot.sort_index(axis=1,level=0).dropna(axis=1)
```

In [156…
```
fig,ax=plt.subplots(figsize=(30,30))
indX=np.arange(monthly_pivot.shape[1])
indY=np.arange(monthly_pivot.shape[0])
im=ax.imshow(monthly_pivot,cmap='Reds',alpha=0.75)
ax.set_ylabel('Teams');
ax.set_xlabel('Years');
ax.set_xticks(indX);
ax.set_yticks(indY);
ax.set_xticklabels(monthly_pivot.columns);
ax.set_yticklabels(monthly_pivot.index);
plt.savefig('../Images/Overs_by_TeamMonth.jpg')
```



The heatmaps above suggest that over a single year or given month (or other short-term period) that oddsmakers struggle to properly set the line for a specific team.

Modeling

Initial Setup

In [164…

```
class ModelWithCV():
    '''Structure to save the model and more easily see its crossvalidation'''

    def __init__(self, model, model_name, X, y, cv_now=True):
```

```
        self.model = model
        self.name = model_name
        self.X = X
        self.y = y
        # For CV results
        self.cv_results = None
        self.cv_mean = None
        self.cv_median = None
        self.cv_std = None
        #
        if cv_now:
            self.cross_validate()

    def cross_validate(self, X=None, y=None, kfolds=10):
        '''
        Perform cross-validation and return results.

        Args:
          X:
            Optional; Training data to perform CV on. Otherwise use X from ob
          y:
            Optional; Training data to perform CV on. Otherwise use y from ob
          kfolds:
            Optional; Number of folds for CV (default is 10)
        '''

        cv_X = X if X else self.X
        cv_y = y if y else self.y

        self.cv_results = cross_val_score(self.model, cv_X, cv_y, cv=kfolds)
        self.cv_mean = np.mean(self.cv_results)
        self.cv_median = np.median(self.cv_results)
        self.cv_std = np.std(self.cv_results)


    def print_cv_summary(self):
        cv_summary = (
        f'''CV Results for `{self.name}` model:
            {self.cv_mean:.5f} ± {self.cv_std:.5f} accuracy
        ''')
        print(cv_summary)


    def plot_cv(self, ax):
        '''
        Plot the cross-validation values using the array of results and given
        Axis for plotting.
        '''
        ax.set_title(f'CV Results for `{self.name}` Model')
        # Thinner violinplot with higher bw
        sns.violinplot(y=self.cv_results, ax=ax, bw=.4)
        sns.swarmplot(
                y=self.cv_results,
```

```
                color='orange',
                size=10,
                alpha= 0.8,
                ax=ax
        )

        return ax
```

Prep Data for Training

In [160…
```
dataset=pd.read_pickle('../Merged Dfs/Final_Database.pkl')
X=dataset.drop('OVER',axis=1)
y=dataset.OVER
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1988)
```

In [161…
```
scaler=StandardScaler()
scaler.fit(X_train)
X_train=pd.DataFrame(scaler.transform(X_train),columns=X_train.columns,index=
X_test=pd.DataFrame(scaler.transform(X_test),columns=X_test.columns,index=X_t
```

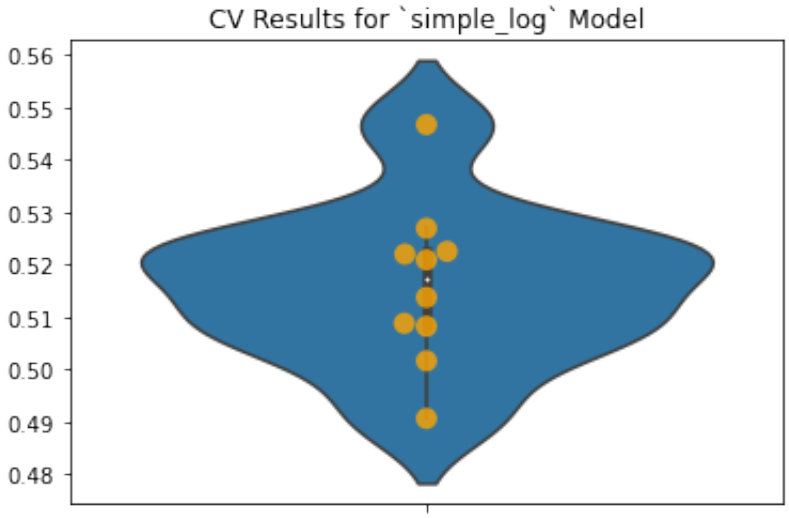
Train Models

Logistic Regression

Simple w/ Cross Validation

In [165…
```
if (__MODELS_TRAINED__&__DATA_COLLECTED__):
    with open('../Models/SimpleCV_Log.pkl','rb') as f:
        CV_log_model=pickle.load(f)
    pass
else:
    CV_log_results=ModelWithCV(LogisticRegression(random_state=1988,max_iter=
    CV_log_model=CV_log_results.model
    CV_log_model.fit(X_train,y_train)
    with open('../Models/SimpleCV_Log.pkl','wb') as f:
        pickle.dump(CV_log_model,f)
    pass
```

In [166…
```
fig,ax=plt.subplots()
ax=CV_log_results.plot_cv(ax)
plt.plot();
plt.savefig('../Images/SimpleCVLog-Results.jpg')
```



In [402…
```
pprint.pprint(dict(zip(X_train.columns.values,np.round(CV_log_model.coef_[0],
```

```
{'#Bat_A': 0.05,
 '#Bat_H': -0.0,
 '2B_A': 0.07,
 '2B_H': 0.09,
 '3B_A': 0.06,
 '3B_H': 0.09,
 'AB_A': 0.39,
 'AB_H': -0.45,
 'AIR_A': -0.2,
 'AIR_H': -0.23,
 'All_#P_A': -0.01,
 'All_#P_H': -0.06,
 'All_BF_A': -0.22,
 'All_BF_H': -0.66,
 'All_FIP_A': 0.15,
 'All_FIP_H': 0.52,
 'All_H9_A': -0.01,
 'All_H9_H': 0.03,
 'All_HR9_A': -0.1,
 'All_HR9_H': -0.13,
 'All_IP_A': 0.09,
 'All_IP_H': 0.26,
 'All_LOB_A': 0.11,
 'All_LOB_H': 0.24,
 'All_PAge_A': -0.01,
 'All_PAge_H': 0.03,
 'All_RA/G_A': 0.05,
 'All_RA/G_H': 0.17,
 'All_SO/W_A': 0.03,
 'All_SO/W_H': -0.03,
 'All_SO_A': 0.07,
 'All_SO_H': 0.0,
 'All_cSho_A': 0.01,
 'All_cSho_H': 0.01,
 'All_tSho_A': 0.02,
 'All_tSho_H': -0.01,
 'BA_A': 0.35,
```

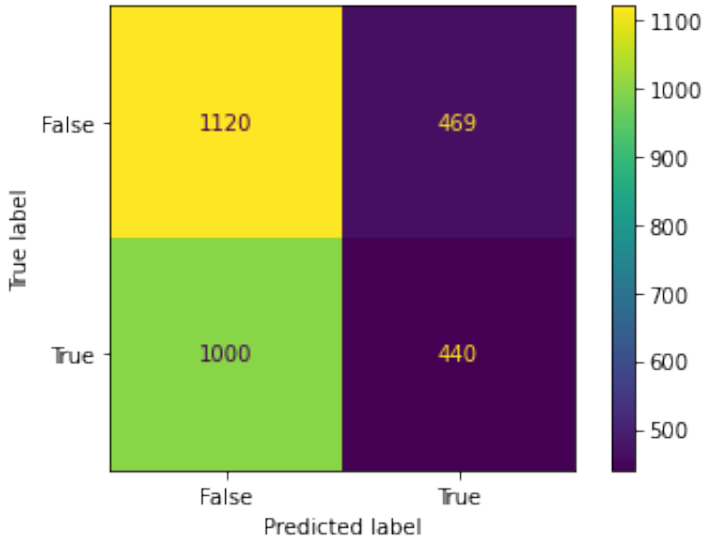
```
'BA_H': 0.12,
'BAbip_A': -0.39,
'BAbip_H': -0.61,
'BatAge_A': 0.04,
'BatAge_H': 0.0,
'BtRuns_A': -0.51,
'BtRuns_H': 0.63,
'CG_A': 0.04,
'CG_H': 0.03,
'CloseOU': -0.1,
'DefEff_A': -0.09,
'DefEff_H': -0.18,
'Fld%_A': 0.01,
'Fld%_H': -0.02,
'Free_Bases_A': 0.58,
'Free_Bases_H': -0.11,
'GDP_A': -0.03,
'GDP_H': 0.01,
'HR_A': -0.07,
'HR_H': 0.03,
'H_A': 0.33,
'H_H': 1.06,
'LO_A': 0.03,
'LO_H': 0.02,
'OPS+_A': 0.48,
'OPS+_H': -1.11,
'OWn%_A': -0.09,
'OWn%_H': 0.14,
'PA_A': -0.71,
'PA_H': 0.25,
'PwrSpd_A': 0.04,
'PwrSpd_H': 0.05,
'RBI_A': -0.49,
'RBI_H': -0.3,
'R_A': 0.38,
'R_H': 0.08,
'Relief_0DR_A': -0.02,
'Relief_0DR_H': -0.04,
'Relief_BSv_A': -0.0,
'Relief_BSv_H': 0.02,
'Relief_IPmult_A': -0.01,
'Relief_IPmult_H': -0.0,
'Relief_IS%_A': -0.01,
'Relief_IS%_H': -0.01,
'Relief_SVSit_A': -0.01,
'Relief_SVSit_H': 0.0,
'Relief_aLI_A': 0.02,
'Relief_aLI_H': -0.06,
'Rgood_A': -0.01,
'Rgood_H': -0.05,
'Rtot/yr_A': 0.05,
'Rtot/yr_H': -0.03,
'SO_A': 0.34,
'SO_H': 0.47,
'Sacs_A': 0.08,
'Sacs_H': -0.05,
'Start_CG_A': -0.04,
```

```
 'Start_CG_H': -0.03,
 'Start_GmScA_A': -0.07,
 'Start_GmScA_H': 0.15,
 'Start_IP/GS_A': -0.12,
 'Start_IP/GS_H': -0.02,
 'Start_Ltuf_A': -0.0,
 'Start_Ltuf_H': -0.06,
 'Start_Pit/GS_A': -0.0,
 'Start_Pit/GS_H': -0.06,
 'Start_QS%_A': 0.11,
 'Start_QS%_H': 0.09,
 'Start_SHO_A': 0.01,
 'Start_SHO_H': 0.01,
 'Start_Wchp_A': -0.04,
 'Start_Wchp_H': 0.03,
 'Steal Att._A': -0.08,
 'Steal Att._H': -0.04}
```

In [167…

```
plot_confusion_matrix(CV_log_model,X_test,y_test);
plt.savefig('../Images/SimpleCVLog-Confusion.jpg')
print(classification_report(y_test,CV_log_model.predict(X_test)))
```

```
              precision    recall  f1-score   support

       False       0.53      0.70      0.60      1589
        True       0.48      0.31      0.37      1440

    accuracy                           0.52      3029
   macro avg       0.51      0.51      0.49      3029
weighted avg       0.51      0.52      0.49      3029
```



The logistic model is somehow even worse than just a dummy guessing all False. It is also noteworthy that the majority of the accuracy is coming from correctly guessing False observations. This imbalance may be a detriment to a gambler who may be only concerned with the outcome when the bets (i.e. when the model predicts True and might suffer from too many false negativeese, even if it benefits the model's accuracy.

Grid Search

In [168…
```
param_grid={
    'max_iter':[500,1000,1500,5000],
    'penalty':['l1','l2','none'],
    'solver':['liblinear','lbfgs'],
}
```

In [169…
```
grid_logistic=GridSearchCV(LogisticRegression(random_state=1988),param_grid,c
```

In [170…
```
if (__MODELS_TRAINED__&__DATA_COLLECTED__):
    with open('../Models/Grid_Log.pkl','rb') as f:
        grid_logistic_model=pickle.load(f)
    pass
else:
    grid_logistic.fit(X_train,y_train)
    grid_logistic_model=grid_logistic.best_estimator_
    with open('../Models/Grid_Log.pkl','wb') as f:
        pickle.dump(grid_logistic_model,f)
    pass
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   45.9s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:  2.6min finished
```

In [171…
```
print(grid_logistic.best_score_)
print(grid_logistic.best_params_)
```

```
0.5166735385701475
{'max_iter': 1000, 'penalty': 'none', 'solver': 'lbfgs'}
```

By getting rid of the regularization (penalty parameter set to None), we improve--but only slightly--the model's performance on the training data.

What do the coefficients look like?

In [172…
```
pprint.pprint(dict(zip(X_train.columns.values,np.round(grid_logistic_model.co
```

```
{'#Bat_A': 0.05,
 '#Bat_H': 0.0,
 '2B_A': 0.12,
 '2B_H': 0.02,
 '3B_A': 0.07,
 '3B_H': 0.05,
 'AIR_A': -0.44,
 'AIR_H': 0.1,
 'All_#P_A': -0.02,
 'All_#P_H': -0.05,
 'All_BF_A': -1.45,
 'All_BF_H': -4.09,
 'All_FIP_A': 0.14,
 'All_FIP_H': 0.55,
```

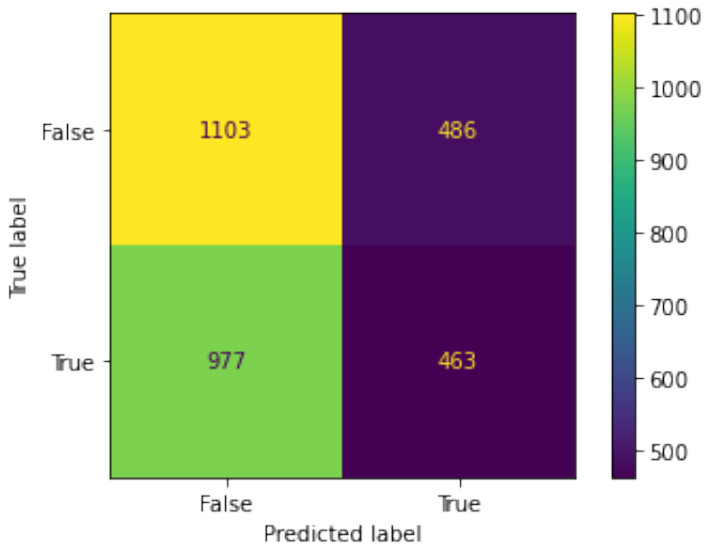
```
'All_H9_A': -0.06,
'All_H9_H': 0.06,
'All_HR9_A': -0.07,
'All_HR9_H': -0.14,
'All_IP_A': 0.57,
'All_IP_H': 1.51,
'All_LOB_A': 0.73,
'All_LOB_H': 1.95,
'All_PAge_A': 0.0,
'All_PAge_H': 0.02,
'All_RA/G_A': 1.03,
'All_RA/G_H': 2.75,
'All_SO_A': 0.03,
'All_SO_H': 0.05,
'All_cSho_A': 0.01,
'All_cSho_H': 0.01,
'All_tSho_A': 0.03,
'All_tSho_H': 0.0,
'BA_A': -0.06,
'BA_H': 0.14,
'BAbip_A': -0.52,
'BAbip_H': -0.73,
'BatAge_A': 0.04,
'BatAge_H': -0.0,
'BtRuns_A': -0.96,
'BtRuns_H': 1.14,
'CG_A': 0.05,
'CG_H': 0.02,
'CloseOU': -0.11,
'DefEff_A': -0.15,
'DefEff_H': -0.14,
'Fld%_A': 0.02,
'Fld%_H': -0.03,
'Free_Bases_A': 0.6,
'Free_Bases_H': 0.06,
'GDP_A': -0.04,
'GDP_H': 0.01,
'HR_A': -0.01,
'HR_H': -0.26,
'H_A': 1.26,
'H_H': 0.83,
'LO_A': 0.04,
'LO_H': 0.03,
'OPS+_A': 0.69,
'OPS+_H': -1.3,
'OWn%_A': -0.12,
'OWn%_H': 0.23,
'PA_A': -0.58,
'PA_H': -0.21,
'PwrSpd_A': 0.07,
'PwrSpd_H': 0.0,
'RBI_A': -0.63,
'RBI_H': -0.38,
'R_A': 0.53,
'R_H': 0.21,
'Relief_0DR_A': -0.03,
'Relief_0DR_H': -0.04,
```

```
'Relief_BSv_A': 0.0,
'Relief_BSv_H': 0.02,
'Relief_IS%_A': -0.01,
'Relief_IS%_H': 0.01,
'Relief_SVSit_A': -0.01,
'Relief_SVSit_H': 0.02,
'Relief_aLI_A': 0.01,
'Relief_aLI_H': -0.05,
'Rgood_A': -0.0,
'Rgood_H': -0.05,
'Rtot/yr_A': 0.05,
'Rtot/yr_H': -0.05,
'SO_A': 0.44,
'SO_H': 0.57,
'Sacs_A': 0.05,
'Sacs_H': 0.04,
'Start_CG_A': -0.07,
'Start_CG_H': -0.01,
'Start_IP/GS_A': -0.08,
'Start_IP/GS_H': -0.04,
'Start_Ltuf_A': 0.01,
'Start_Ltuf_H': -0.08,
'Start_Pit/GS_A': -0.03,
'Start_Pit/GS_H': -0.04,
'Start_QS%_A': 0.08,
'Start_QS%_H': 0.17,
'Start_SHO_A': 0.01,
'Start_SHO_H': 0.01,
'Start_Wchp_A': -0.05,
'Start_Wchp_H': 0.04,
'Steal Att._A': -0.11,
'Steal Att._H': 0.0,
'cli': -0.03}
```

In [173…

```
print(classification_report(y_test,grid_logistic_model.predict(X_test)))
plot_confusion_matrix(grid_logistic_model,X_test,y_test);
plt.savefig('../Images/GridLog-Confusion.jpg')
```

```
                precision    recall  f1-score   support

         False       0.53      0.69      0.60      1589
          True       0.49      0.32      0.39      1440

      accuracy                           0.52      3029
     macro avg       0.51      0.51      0.49      3029
  weighted avg       0.51      0.52      0.50      3029
```



Random Forest

In [404…

```
param_grid = {
    'n_estimators': [300,400,500,600,700,800],
    'min_samples_leaf':[25,50,75,80,90],
    'max_leaf_nodes':range(50,100,5),
    'max_features' :['auto', 5,6]
}
```

In [405…

```
grid_forest=GridSearchCV(RandomForestClassifier(), param_grid, cv=5, scoring=
```

In [407…

```
if (__MODELS_TRAINED__&__DATA_COLLECTED__):
    with open('../Models/Grid_Log.pkl','rb') as f:
        grid_forest_model=pickle.load(f)
    pass
else:
    grid_forest.fit(X_train,y_train)
    grid_logistic_model=grid_logistic.best_estimator_
    with open('../Models/Grid_Forest.pkl','wb') as f:
        pickle.dump(grid_lforest_model,f)
    pass
```

```
Fitting 5 folds for each of 900 candidates, totalling 4500 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  2.4min
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed: 10.3min
[Parallel(n_jobs=-1)]: Done 434 tasks      | elapsed: 23.4min
[Parallel(n_jobs=-1)]: Done 784 tasks      | elapsed: 42.5min
[Parallel(n_jobs=-1)]: Done 1234 tasks      | elapsed: 66.8min
[Parallel(n_jobs=-1)]: Done 1784 tasks      | elapsed: 89.4min
[Parallel(n_jobs=-1)]: Done 2434 tasks      | elapsed: 110.2min
[Parallel(n_jobs=-1)]: Done 3184 tasks      | elapsed: 135.0min
[Parallel(n_jobs=-1)]: Done 4034 tasks      | elapsed: 165.5min
[Parallel(n_jobs=-1)]: Done 4500 out of 4500 | elapsed: 182.5min finished
```

Out[407…

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_features': ['auto', 5, 6],
                         'max_leaf_nodes': range(50, 100, 5),
                         'min_samples_leaf': [25, 50, 75, 80, 90],
                         'n_estimators': [300, 400, 500, 600, 700, 800]},
             scoring='accuracy', verbose=1)
```

In [ ]:

```
grid_forest_model.score(X_test,y_test)
print(classification_report(y_test,grid_forest_model.predict(X_test)))
plot_confusion_matrix(grid_forest_model,X_test,y_test);
plt.savefig('../Images/Forest-Confusion.jpg')
```

In [175…

```
pprint.pprint(dict(zip(X_test.columns,grid_forest_model.feature_importances_)
```

```
{'#Bat_A': 0.008772766589135048,
 '#Bat_H': 0.0053278449323147705,
 '2B_A': 0.01010743791856066,
 '2B_H': 0.0108974452362617,
 '3B_A': 0.009548752228716685,
 '3B_H': 0.011710132351576481,
 'AIR_A': 0.011581759207467903,
 'AIR_H': 0.007509543100647016,
 'All_#P_A': 0.005550056912706601,
 'All_#P_H': 0.00843307217775191,
 'All_BF_A': 0.010657201118375339,
 'All_BF_H': 0.006763314545489442,
 'All_FIP_A': 0.011957433937879312,
 'All_FIP_H': 0.002784565393208734,
 'All_H9_A': 0.008894069082619409,
 'All_H9_H': 0.01207851730002184,
 'All_HR9_A': 0.004782416723031912,
 'All_HR9_H': 0.01130479345673314,
 'All_IP_A': 0.009586190933917898,
 'All_IP_H': 0.009343615266451383,
 'All_LOB_A': 0.011774754042182398,
 'All_LOB_H': 0.010888535962572799,
 'All_PAge_A': 0.007780398723140299,
 'All_PAge_H': 0.010062450339605193,
 'All_RA/G_A': 0.010106459869155919,
 'All_RA/G_H': 0.011938208465969987,
 'All_SO_A': 0.012994754335460998,
 'All_SO_H': 0.009923212070707121,
 'All_cSho_A': 0.0026908213188208082,
```

```
'All_cSho_H': 0.00948843491917227,
'All_tSho_A': 0.007950010643085958,
'All_tSho_H': 0.012454694181189723,
'BA_A': 0.00758044617703022,
'BA_H': 0.009088633232501498,
'BAbip_A': 0.007970482071315784,
'BAbip_H': 0.010148644981680154,
'BatAge_A': 0.010875569214530745,
'BatAge_H': 0.009430705242092604,
'BtRuns_A': 0.011876056633978555,
'BtRuns_H': 0.010326568648770598,
'CG_A': 0.009148224197861075,
'CG_H': 0.007589515035646812,
'CloseOU': 0.013392491498286797,
'DefEff_A': 0.015732754128038918,
'DefEff_H': 0.005843262668966968,
'Fld%_A': 0.010645241253596885,
'Fld%_H': 0.010019236223083903,
'Free_Bases_A': 0.011681708991967927,
'Free_Bases_H': 0.01009429871505305,
'GDP_A': 0.00956038566156613,
'GDP_H': 0.009462258574589478,
'HR_A': 0.011033866716850116,
'HR_H': 0.009206432621970906,
'H_A': 0.011348667496927735,
'H_H': 0.0074693942569863695,
'LO_A': 0.0071983615888626664,
'LO_H': 0.01226715895004384,
'OPS+_A': 0.008116315895162601,
'OPS+_H': 0.009284719661824186,
'OWn%_A': 0.009424317027402259,
'OWn%_H': 0.010689615738319792,
'PA_A': 0.010970004843857486,
'PA_H': 0.00789957497888079,
'PwrSpd_A': 0.011196867609526621,
'PwrSpd_H': 0.00783876633931492,
'RBI_A': 0.009090444148027704,
'RBI_H': 0.012020383825130244,
'R_A': 0.009862674583953868,
'R_H': 0.007707510777783736,
'Relief_0DR_A': 0.007853710210103752,
'Relief_0DR_H': 0.004671689509873115,
'Relief_BSv_A': 0.005042559043191015,
'Relief_BSv_H': 0.0059272414805949985,
'Relief_IS%_A': 0.007445425918445963,
'Relief_IS%_H': 0.009825075620848995,
'Relief_SVSit_A': 0.0073566948486700956,
'Relief_SVSit_H': 0.002704319836394598,
'Relief_aLI_A': 0.00997219400934001,
'Relief_aLI_H': 0.005989665697495316,
'Rgood_A': 0.011572311945348926,
'Rgood_H': 0.00995895389527901,
'Rtot/yr_A': 0.00917874345088461,
'Rtot/yr_H': 0.008201649259836977,
'SO_A': 0.010122831804488468,
'SO_H': 0.010918153080909192,
'Sacs_A': 0.009925762005594804,
```

```
 'Sacs_H': 0.00871601043977288,
 'Start_CG_A': 0.006802609722029547,
 'Start_CG_H': 0.0038512379012929617,
 'Start_IP/GS_A': 0.008396059977067105,
 'Start_IP/GS_H': 0.0064871599137779675,
 'Start_Ltuf_A': 0.007579387195020896,
 'Start_Ltuf_H': 0.0069622522099591876,
 'Start_Pit/GS_A': 0.006247895881135394,
 'Start_Pit/GS_H': 0.007232731148133386,
 'Start_QS%_A': 0.0020638277205715988,
 'Start_QS%_H': 0.011512675603576635,
 'Start_SHO_A': 0.004700062902261075,
 'Start_SHO_H': 0.009636183081329937,
 'Start_Wchp_A': 0.00998891387113603,
 'Start_Wchp_H': 0.01008973324476143,
 'Steal Att._A': 0.011344210713306851,
 'Steal Att._H': 0.008701979916756473,
 'cli': 0.006445916570546439}
```

In [176…
```
grid_forest_model.get_params()
```

Out[176…
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 5,
 'max_leaf_nodes': 50,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 80,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 700,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

In [179…
```
#Lets look @ an example tree
example_tree=grid_forest_model.estimators_[5]
export_graphviz(example_tree,out_file='../Images/tree_example.dot')
```