In [1]:
```
import pandas as pd
import numpy as np
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\Downloads\\archive\\games.csv")
a
```

Out[2]:

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON
0	2022-12-22	22200477	Final	1610612740	1610612759	202
1	2022-12-22	22200478	Final	1610612762	1610612764	202
2	2022-12-21	22200466	Final	1610612739	1610612749	202
3	2022-12-21	22200467	Final	1610612755	1610612765	202
4	2022-12-21	22200468	Final	1610612737	1610612741	202
...	...	...	...	...	...	.
26646	2014-10-06	11400007	Final	1610612737	1610612740	201
26647	2014-10-06	11400004	Final	1610612741	1610612764	201
26648	2014-10-06	11400005	Final	1610612747	1610612743	201
26649	2014-10-05	11400002	Final	1610612761	1610612758	201
26650	2014-10-04	11400001	Final	1610612748	1610612740	201

26651 rows × 21 columns

In [3]:
```
a.head()
```

Out[3]:

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TE
0	2022-12-22	22200477	Final	1610612740	1610612759	2022	
1	2022-12-22	22200478	Final	1610612762	1610612764	2022	
2	2022-12-21	22200466	Final	1610612739	1610612749	2022	
3	2022-12-21	22200467	Final	1610612755	1610612765	2022	
4	2022-12-21	22200468	Final	1610612737	1610612741	2022	

5 rows × 21 columns

In [4]:
```
a.tail()
```

Out[4]:

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON
26646	2014-10-06	11400007	Final	1610612737	1610612740	201

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON
26647	2014-10-06	11400004	Final	1610612741	1610612764	201
26648	2014-10-06	11400005	Final	1610612747	1610612743	201
26649	2014-10-05	11400002	Final	1610612761	1610612758	201
26650	2014-10-04	11400001	Final	1610612748	1610612740	201

5 rows × 21 columns

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

Out[5]:

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home
count	2.665100e+04	2.665100e+04	2.665100e+04	26651.000000	2.665100e+04	26552.00000
mean	2.175487e+07	1.610613e+09	1.610613e+09	2012.113879	1.610613e+09	103.45589
std	5.570189e+06	8.638670e+00	8.659299e+00	5.587031	8.638670e+00	13.28337
min	1.030000e+07	1.610613e+09	1.610613e+09	2003.000000	1.610613e+09	36.00000
25%	2.070001e+07	1.610613e+09	1.610613e+09	2007.000000	1.610613e+09	94.00000
50%	2.120076e+07	1.610613e+09	1.610613e+09	2012.000000	1.610613e+09	103.00000
75%	2.180005e+07	1.610613e+09	1.610613e+09	2017.000000	1.610613e+09	112.00000
max	5.210021e+07	1.610613e+09	1.610613e+09	2022.000000	1.610613e+09	168.00000

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26651 entries, 0 to 26650
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   GAME_DATE_EST     26651 non-null  object 
 1   GAME_ID           26651 non-null  int64  
 2   GAME_STATUS_TEXT  26651 non-null  object 
 3   HOME_TEAM_ID      26651 non-null  int64  
 4   VISITOR_TEAM_ID   26651 non-null  int64  
 5   SEASON            26651 non-null  int64  
 6   TEAM_ID_home      26651 non-null  int64  
 7   PTS_home          26552 non-null  float64
 8   FG_PCT_home       26552 non-null  float64
 9   FT_PCT_home       26552 non-null  float64
 10  FG3_PCT_home      26552 non-null  float64
 11  AST_home          26552 non-null  float64
 12  REB_home          26552 non-null  float64
 13  TEAM_ID_away      26651 non-null  int64  
 14  PTS_away          26552 non-null  float64
 15  FG_PCT_away       26552 non-null  float64
 16  FT_PCT_away       26552 non-null  float64
 17  FG3_PCT_away      26552 non-null  float64
 18  AST_away          26552 non-null  float64
 19  REB_away          26552 non-null  float64
 20  HOME_TEAM_WINS    26651 non-null  int64  
```

```
dtypes: float64(12), int64(7), object(2)
memory usage: 4.3+ MB
```

In [7]:
```
a.isna().sum()
```

Out[7]:
```
GAME_DATE_EST        0
GAME_ID              0
GAME_STATUS_TEXT     0
HOME_TEAM_ID         0
VISITOR_TEAM_ID      0
SEASON               0
TEAM_ID_home         0
PTS_home            99
FG_PCT_home         99
FT_PCT_home         99
FG3_PCT_home        99
AST_home            99
REB_home            99
TEAM_ID_away         0
PTS_away            99
FG_PCT_away         99
FT_PCT_away         99
FG3_PCT_away        99
AST_away            99
REB_away            99
HOME_TEAM_WINS       0
dtype: int64
```

In [8]:
```
a.fillna(35,inplace=True)
```

In [9]:
```
a.isna().sum()
```

Out[9]:
```
GAME_DATE_EST       0
GAME_ID             0
GAME_STATUS_TEXT    0
HOME_TEAM_ID        0
VISITOR_TEAM_ID     0
SEASON              0
TEAM_ID_home        0
PTS_home            0
FG_PCT_home         0
FT_PCT_home         0
FG3_PCT_home        0
AST_home            0
REB_home            0
TEAM_ID_away        0
PTS_away            0
FG_PCT_away         0
FT_PCT_away         0
FG3_PCT_away        0
AST_away            0
REB_away            0
HOME_TEAM_WINS      0
dtype: int64
```

In [10]:
```
list(a)
```

Out[10]:
```
['GAME_DATE_EST',
 'GAME_ID',
 'GAME_STATUS_TEXT',
 'HOME_TEAM_ID',
```

```
'VISITOR_TEAM_ID',
'SEASON',
'TEAM_ID_home',
'PTS_home',
'FG_PCT_home',
'FT_PCT_home',
'FG3_PCT_home',
'AST_home',
'REB_home',
'TEAM_ID_away',
'PTS_away',
'FG_PCT_away',
'FT_PCT_away',
'FG3_PCT_away',
'AST_away',
'REB_away',
'HOME_TEAM_WINS']
```

In [11]:
```
b=a.drop(['GAME_DATE_EST'],axis=1)
```

In [12]:
```
b
```

Out[12]:

	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home
0	22200477	Final	1610612740	1610612759	2022	1610612740
1	22200478	Final	1610612762	1610612764	2022	1610612762
2	22200466	Final	1610612739	1610612749	2022	1610612739
3	22200467	Final	1610612755	1610612765	2022	1610612755
4	22200468	Final	1610612737	1610612741	2022	1610612737
...	...	...	...	...	...	...
26646	11400007	Final	1610612737	1610612740	2014	1610612737
26647	11400004	Final	1610612741	1610612764	2014	1610612741
26648	11400005	Final	1610612747	1610612743	2014	1610612747
26649	11400002	Final	1610612761	1610612758	2014	1610612761
26650	11400001	Final	1610612748	1610612740	2014	1610612748

26651 rows × 20 columns

In [13]:
```
b.groupby(['HOME_TEAM_WINS']).count()
```

Out[13]:

HOME_TEAM_WINS	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TE
0	11006	11006	11006	11006	11006	
1	15645	15645	15645	15645	15645	

In [14]:
```
c=pd.get_dummies(b,dtype=int)
c
```

Out[14]:

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_h
0	22200477	1610612740	1610612759	2022	1610612740	126.0	
1	22200478	1610612762	1610612764	2022	1610612762	120.0	
2	22200466	1610612739	1610612749	2022	1610612739	114.0	
3	22200467	1610612755	1610612765	2022	1610612755	113.0	
4	22200468	1610612737	1610612741	2022	1610612737	108.0	
...	...	...	...	...	...	...	
26646	11400007	1610612737	1610612740	2014	1610612737	93.0	
26647	11400004	1610612741	1610612764	2014	1610612741	81.0	
26648	11400005	1610612747	1610612743	2014	1610612747	98.0	
26649	11400002	1610612761	1610612758	2014	1610612761	99.0	
26650	11400001	1610612748	1610612740	2014	1610612748	86.0	

26651 rows × 20 columns

In [15]:
```
cor=b.corr()
cor
```

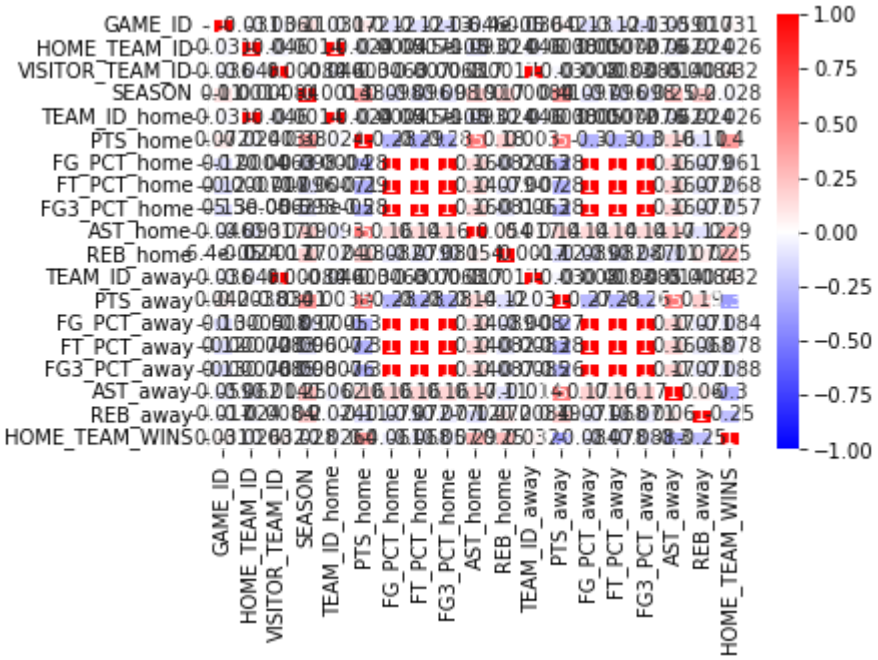
Out[15]:

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_ho
GAME_ID	1.000000	-0.030770	-0.036014	0.105800	-0.030770	0.072
HOME_TEAM_ID	-0.030770	1.000000	-0.046277	-0.001401	1.000000	-0.023
VISITOR_TEAM_ID	-0.036014	-0.046277	1.000000	0.000840	-0.046277	0.003
SEASON	0.105800	-0.001401	0.000840	1.000000	-0.001401	0.376
TEAM_ID_home	-0.030770	1.000000	-0.046277	-0.001401	1.000000	-0.023
PTS_home	0.072393	-0.023525	0.003005	0.376239	-0.023525	1.000
FG_PCT_home	-0.124983	-0.000401	-0.006303	-0.098249	-0.000401	-0.282
FT_PCT_home	-0.123637	-0.000707	-0.006980	-0.095996	-0.000707	-0.290
FG3_PCT_home	-0.125502	-0.000055	-0.006300	-0.098188	-0.000055	-0.278
AST_home	-0.046225	-0.093471	0.017023	0.190400	-0.093471	0.529
REB_home	-0.000064	-0.023688	0.001685	0.165785	-0.023688	0.177
TEAM_ID_away	-0.036014	-0.046277	1.000000	0.000840	-0.046277	0.003
PTS_away	0.042111	-0.003766	-0.030000	0.405771	-0.003766	0.531
FG_PCT_away	-0.125343	0.000501	-0.007980	-0.097405	0.000501	-0.296
FT_PCT_away	-0.123654	0.000723	-0.008333	-0.095820	0.000723	-0.295
FG3_PCT_away	-0.125500	0.000758	-0.008478	-0.098426	0.000758	-0.296

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_h
AST_away	-0.058826	-0.061976	-0.013995	0.247919	-0.061976	0.155
REB_away	-0.017368	-0.023927	-0.008369	0.199351	-0.023927	-0.113
HOME_TEAM_WINS	0.030804	-0.026123	0.032002	-0.027849	-0.026123	0.398

In [16]:
```
import seaborn as sb
import matplotlib.pyplot as plt
sb.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidth=-5,cmap='bwr')
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
y=c['HOME_TEAM_WINS']
y
```

Out[17]:
```
0        1
1        1
2        1
3        1
4        0
        ..
26646    1
26647    0
26648    1
26649    1
26650    0
Name: HOME_TEAM_WINS, Length: 26651, dtype: int64
```

In [18]:
```
x=c.drop(['HOME_TEAM_WINS'],axis=1)
x
```

Out[18]:

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_h
0	22200477	1610612740	1610612759	2022	1610612740	126.0	
1	22200478	1610612762	1610612764	2022	1610612762	120.0	

	GAME_ID	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_I
2	22200466	1610612739	1610612749	2022	1610612739	114.0	
3	22200467	1610612755	1610612765	2022	1610612755	113.0	
4	22200468	1610612737	1610612741	2022	1610612737	108.0	
...	...	...	...	...	...	...	
26646	11400007	1610612737	1610612740	2014	1610612737	93.0	
26647	11400004	1610612741	1610612764	2014	1610612741	81.0	
26648	11400005	1610612747	1610612743	2014	1610612747	98.0	
26649	11400002	1610612761	1610612758	2014	1610612761	99.0	
26650	11400001	1610612748	1610612740	2014	1610612748	86.0	

26651 rows × 19 columns

In [19]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.35,random_state=33)
```

In [20]:
```
from sklearn.linear_model import LogisticRegression
cls=LogisticRegression()
cls.fit(x_train,y_train)
```

Out[20]:
```
LogisticRegression()
```

In [21]:
```
ypred=cls.predict(x_test)
ypred
```

Out[21]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [22]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,ypred)
```

Out[22]:
```
array([[   0, 3840],
       [   0, 5488]], dtype=int64)
```

In [23]:
```
from sklearn.metrics import accuracy_score
accuracy_score(ypred,y_test)
```

Out[23]:
```
0.5883361921097771
```

In [24]:
```
#(from sklearn.model_selection import GridSearchCV
#from sklearn.linear_model import Ridge
#ridge=Ridge()
#alpha=[1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
#parameters={'alpha':alpha}
#reg=GridSearchCV(ridge,parameters)
#reg.fit(x_train,y_train)
```

In [25]:
```
#reg.best_params_
```

In [26]:
```
#ridge=Ridge(1)
#ridge.fit(x_train,y_train)
#y_pred=ridge.predict(x_test)
#y_pred
```

In [27]:
```
#r2_score(y_test,y_pred)
```

In [28]:
```
#from sklearn.model_selection import GridSearchCV
#from sklearn.linear_model import Lasso
#lasso=Lasso()
#alpha=[1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
#parameters={'alpha':alpha}
#reg=GridSearchCV(lasso,parameters)
#reg.fit(x_train,y_train)
```

In [29]:
```
#reg.best_params_
```

In [30]:
```
#lasso=Lasso(1e-08)
#lasso.fit(x_train,y_train)
#y_pred=lasso.predict(x_test)
#y_pred
```

In [31]:
```
#r2_score(y_test,y_pred)
```

Random forest

In [32]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
reg=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['gini','entropy']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[32]:
```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [33]:
```
rfc_reg.best_params_
```

Out[33]:
```
{'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 150}
```

In [34]:
```
reg=RandomForestClassifier(n_estimators=150,criterion='gini',max_depth=10)
reg.fit(x_train,y_train)
```

Out[34]:
```
RandomForestClassifier(max_depth=10, n_estimators=150)
```

In [35]:
```
ypred=reg.predict(x_test)
ypred
```

Out[35]:
```
array([1, 0, 0, ..., 1, 1, 1], dtype=int64)
```

In [36]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,ypred)
```

Out[36]:
```
0.9669811320754716
```

In [ ]:
```

```