Import required libraries

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warning
warnings.filters('ignore')
```

Loading the dataset

In [2]:

```
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/baseball.csv")
df.head()
```

Out[2]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86

In [3]:

```
df.shape
```

Out[3]:

```
(30, 17)
```

In [4]:

```
df.columns    # This will print the names of all columns
```

Out[4]:

```
Index(['W', 'R', 'AB', 'H', '2B', '3B', 'HR', 'BB', 'SO', 'SB', 'RA', 'ER',
       'ERA', 'CG', 'SHO', 'SV', 'E'],
      dtype='object')
```

In [5]:

```
df.head()    # Will give you first 5 recordsdf.head()
```

Out[5]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86

In [6]:

```
df.tail()
```

Out[6]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
25	92	667	5385	1346	263	26	187	563	1258	59	595	553	3.44	6	21	47	75
26	84	696	5565	1486	288	39	136	457	1159	93	627	597	3.72	7	18	41	78
27	79	720	5649	1494	289	48	154	490	1312	132	713	659	4.04	1	12	44	86
28	74	650	5457	1324	260	36	148	426	1327	82	731	655	4.09	1	6	41	92
29	68	737	5572	1479	274	49	186	388	1283	97	844	799	5.04	4	4	36	95

In [7]:

```
df.info() # This will print the last n rows of the Data Frame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   W       30 non-null     int64  
 1   R       30 non-null     int64  
 2   AB      30 non-null     int64  
 3   H       30 non-null     int64  
 4   2B      30 non-null     int64  
 5   3B      30 non-null     int64  
 6   HR      30 non-null     int64  
 7   BB      30 non-null     int64  
 8   SO      30 non-null     int64  
 9   SB      30 non-null     int64  
 10  RA      30 non-null     int64  
 11  ER      30 non-null     int64  
 12  ERA     30 non-null     float64
 13  CG      30 non-null     int64  
 14  SHO     30 non-null     int64  
 15  SV      30 non-null     int64  
 16  E       30 non-null     int64  
dtypes: float64(1), int64(16)
memory usage: 4.1 KB
```

In [10]:

```
df.describe()
```

Out[10]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.00000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000
mean	80.966667	688.233333	5516.266667	1403.533333	274.733333	31.300000	163.633333	469.100000	1248.20000	83.500000	688.233333	635.833333	3.956333	3.466667	11.300000	43.066667
std	10.453455	58.761754	70.467372	57.140923	18.095405	10.452355	31.823309	57.053725	103.75947	22.815225	72.108005	70.140786	0.454089	2.763473	4.120177	7.869335
min	63.000000	573.000000	5385.000000	1324.000000	236.000000	13.000000	100.000000	375.000000	973.00000	44.000000	525.000000	478.000000	2.940000	0.000000	4.000000	28.000000
25%	74.000000	651.250000	5464.000000	1363.000000	262.250000	23.000000	140.250000	428.250000	1157.50000	69.000000	636.250000	587.250000	3.682500	1.000000	9.000000	37.250000
50%	81.000000	689.000000	5510.000000	1382.500000	275.500000	31.000000	158.500000	473.000000	1261.50000	83.500000	695.500000	644.500000	4.025000	3.000000	12.000000	42.000000
75%	87.750000	718.250000	5570.000000	1451.500000	288.750000	39.000000	177.000000	501.250000	1311.50000	96.500000	732.500000	679.250000	4.220000	5.750000	13.000000	46.750000
max	100.000000	891.000000	5649.000000	1515.000000	308.000000	49.000000	232.000000	570.000000	1518.00000	134.000000	844.000000	799.000000	5.040000	11.000000	21.000000	62.000000

Checking the NULL values

In [11]:

```
df.isnull().sum()
```

Out[11]:

```
W      0
R      0
AB     0
H      0
2B     0
3B     0
HR     0
BB     0
SO     0
SB     0
RA     0
ER     0
ERA    0
CG     0
SHO    0
SV     0
E      0
dtype: int64
```

In [12]:

```
df.isnull()
```

Out[12]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
12	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
15	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
16	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
21	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
22	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
23	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
24	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
25	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
26	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
27	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
28	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
29	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

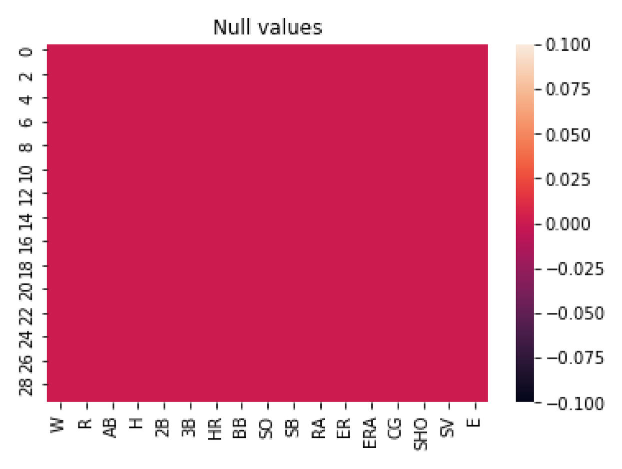
Graphical Analysis

In [13]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.isnull())
plt.title('Null values')
plt.show
```

Out[13]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



correlation

In [14]:

```
corr_matrix=df.corr()
corr_matrix
```

Out[14]:

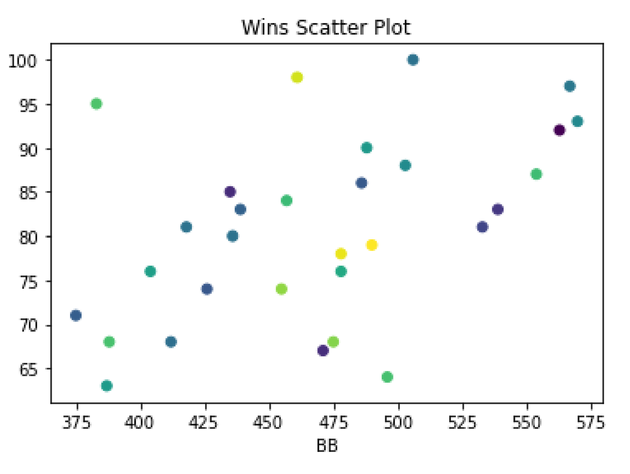
	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
W	1.000000	0.430751	-0.087947	0.037612	0.427797	-0.251118	0.307407	0.484342	0.111850	-0.157234	-0.812952	-0.809435	-0.819600	0.080533	0.471805	0.666530	-0.089485
R	0.430751	1.000000	0.319464	0.482856	0.560084	-0.070072	0.671283	0.402452	-0.054726	0.081367	-0.041623	-0.041245	-0.049281	0.232042	-0.103274	-0.096380	-0.023262
AB	-0.087947	0.319464	1.000000	0.739122	0.453370	0.435422	-0.066983	-0.136414	-0.106022	0.372618	0.316010	0.309686	0.255551	-0.080876	-0.197321	-0.106367	0.316743
H	0.037612	0.482856	0.739122	1.000000	0.566847	0.478694	-0.090855	-0.118281	-0.398830	0.413444	0.224324	0.252489	0.231172	0.147955	-0.145559	-0.130371	-0.033173
2B	0.427797	0.560084	0.453370	0.566847	1.000000	0.220490	0.056292	0.302700	-0.150752	0.195027	-0.218160	-0.235531	-0.254854	0.306675	0.057998	0.171576	0.105754
3B	-0.251118	-0.070072	0.435422	0.478694	0.220490	1.000000	-0.430915	-0.454949	-0.141196	0.457437	0.314125	0.340225	0.330951	-0.065898	-0.041396	-0.142370	0.126678
HR	0.307407	0.671283	-0.066983	-0.090855	0.056292	-0.430915	1.000000	0.425691	0.359923	-0.136567	-0.103903	-0.085922	-0.090917	0.156502	-0.019119	-0.028540	-0.207597
BB	0.484342	0.402452	-0.136414	-0.118281	0.302700	-0.454949	0.425691	1.000000	0.233652	-0.098347	-0.416445	-0.452663	-0.459832	0.462478	0.426004	0.099445	-0.075685
SO	0.111850	-0.054726	-0.106022	-0.398830	-0.150752	-0.141196	0.359923	0.233652	1.000000	0.030968	-0.129745	-0.161612	-0.180368	-0.093418	0.237721	0.126297	0.155133
SB	-0.157234	0.081367	0.372618	0.413444	0.195027	0.457437	-0.136567	-0.098347	0.030968	1.000000	0.132290	0.143068	0.126063	-0.020783	-0.106563	-0.183418	0.079149
RA	-0.812952	-0.041623	0.316010	0.224324	-0.218160	0.314125	-0.103903	-0.416445	-0.129745	0.132290	1.000000	0.991018	0.986674	-0.016659	-0.636862	-0.616224	0.198996
ER	-0.809435	-0.041245	0.309686	0.252489	-0.235531	0.340225	-0.085922	-0.452663	-0.161612	0.143068	0.991018	1.000000	0.997248	-0.020221	-0.630192	-0.589663	0.136921
ERA	-0.819600	-0.049281	0.255551	0.231172	-0.254854	0.330951	-0.090917	-0.459832	-0.180368	0.126063	0.986674	0.997248	1.000000	-0.009856	-0.630833	-0.607005	0.113137
CG	0.080533	0.232042	-0.080876	0.147955	0.306675	-0.065898	0.156502	0.462478	-0.093418	-0.020783	-0.016659	-0.020221	-0.009856	1.000000	0.241676	-0.367766	-0.140047
SHO	0.471805	-0.103274	-0.197321	-0.145559	0.057998	-0.041396	-0.019119	0.426004	0.237721	-0.106563	-0.636862	-0.630192	-0.630833	0.241676	1.000000	0.221639	-0.115716
SV	0.666530	-0.096380	-0.106367	-0.130371	0.171576	-0.142370	-0.028540	0.099445	0.126297	-0.183418	-0.616224	-0.589663	-0.607005	-0.367766	0.221639	1.000000	-0.025636
E	-0.089485	-0.023262	0.316743	-0.033173	0.105754	0.126678	-0.207597	-0.075685	0.155133	0.079149	0.198996	0.136921	0.113137	-0.140047	-0.115716	-0.025636	1.000000

EDA Analysis

In [18]:

```
# Plotting scatter graph of Year vs. Wins
plt.scatter(df['BB'], df['W'], c=df['AB'])
plt.title('Wins Scatter Plot')
plt.xlabel('BB')
plt.ylabel('AB')

plt.show()
```



Conclusion

The prediction accuracies of these models were then compared. We investigated whether the selection of input variables and the characteristics before and after feature selection had an impact on the rediction accuracy The three prediction models proposed in this study all achieved high prediction predict game outcomes for other sports, however, their prediction performance cannot be verified without experimental simulation I am using several model to get best outcome.

THANK YOU