BASEBALL CASE STUDY

Problem Statement:

This dataset utilizes data from 2014 Major League Baseball seasons in order to develop an algorithm that predicts the number of wins for a given team in the 2015 season based on several different indicators of success. There are 16 different features that will be used as the inputs to the machine learning and the output will be a value that represents the number of wins.

Input Features Used:
Runs, At Bats, Hits, Doubles, Triples, Homeruns, Walks, Strikeouts, Stolen Bases, Runs Allowed, Earned Runs, Earned Run Average (ERA), Shutouts, Saves, Complete Games and Errors

Output of Project:
Number of Predicted Wins (W)

Data Analysis:

First the dataset for 2014 Major League Baseball was imported to continue with the analysis.

	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
5	93	891	5509	1480	308	17	232	570	1151	88	670	609	3.80	7	10	34	88
6	87	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48	93
7	81	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43	77
8	80	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60	95
9	78	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40	97
10	88	751	5511	1419	279	32	172	503	1233	101	733	680	4.24	5	9	45	119
11	86	729	5459	1363	278	26	230	486	1392	121	618	572	3.57	5	13	39	85
12	85	661	5417	1331	243	21	176	435	1150	52	675	630	3.94	2	12	46	93

The size of complete dataset was (30,17)

Ie. 30 Rows and 17 Columns. Each column carrying a specific kind of value related to the sport and 30 records.

Description of Variables in Dataset is mentioned below:

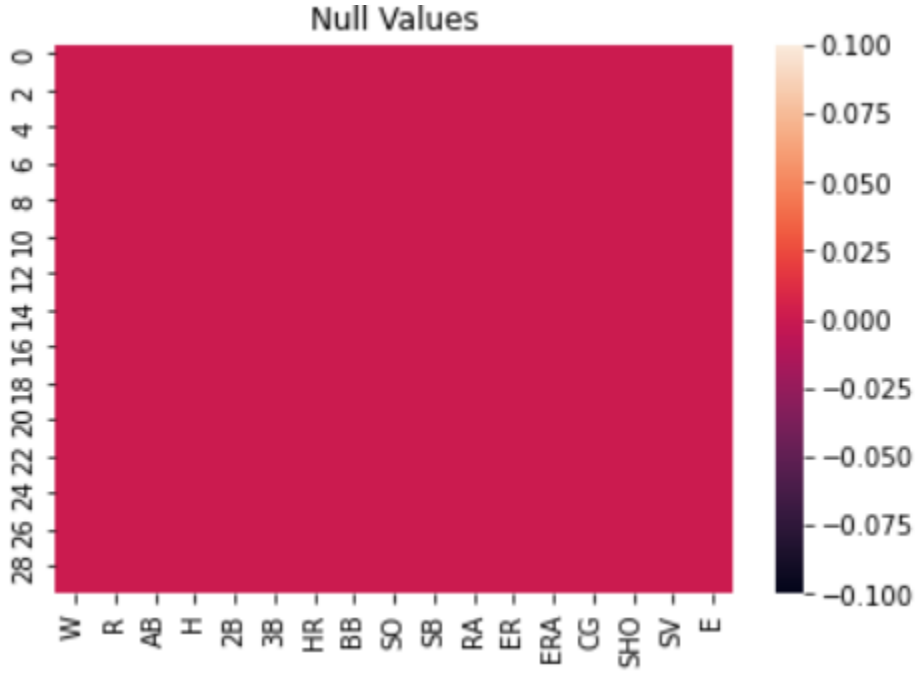
1. W - No. of Predicted Wins
2. R - Runs scored: times reached home plate legally and safely.
3. AB - At bats per home run: at bats divided by home runs
4. H - Hits allowed: total hits allowed
5. 2B - Double: hits on which the batter reaches second base safely without the contribution of a fielding error.
6. 3B – Triple: hits on which the batter reaches third base safely without the contribution of a fielding error.
7. HR - Home runs allowed: total home runs allowed
8. BB - Base on balls (also called a "walk"): hitter not swinging at four pitches called out of the strike zone and awarded first base.
9. SO - Strike out. number of times that a third strike is taken or swung at and missed, or bunted foul. Catcher must catch the third strike or batter may attempt to run to first base.
10. SB - Stolen base: number of bases advanced by the runner while the ball is in the possession of the defense
11. RA - Plate appearances per strikeout: number of times a batter strikes out to their plate appearance
12. ER - Earned run: number of runs that did not occur as a result of errors or passed balls
13. ERA - Earned run average: total number of earned runs (see "ER" above), multiplied by 9, divided by innings pitched.
14. CG - Complete game: number of games where player was the only pitcher for their team.
15. SHO - Shutout: number of complete games pitched with no runs allowed
16. SIERA – Skill-Interactive Earned Run Average: another advanced stat that measures pitching. SIERA builds on FIP and xFIP by taking a deeper look at what makes pitchers better.
17. SV - Save: number of games where the pitcher enters a game led by the pitcher's team, finishes the game without surrendering the lead, is not the winning pitcher, and either (a) the lead was three runs or fewer when the pitcher entered the game; (b) the potential tying run was on base, at bat, or on deck; or (c) the pitcher pitched three or more innings
18. E - Errors: number of times a fielder fails to make a play he should have made with common effort, and the offense benefits as a result

Now we will check the basic description of the Dataset to help analyse it:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER
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
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
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
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
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
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
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
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

ERA	CG	SHO	SV	E
30.000000	30.000000	30.000000	30.000000	30.000000
3.956333	3.466667	11.300000	43.066667	94.333333
0.454089	2.763473	4.120177	7.869335	13.958889
2.940000	0.000000	4.000000	28.000000	75.000000
3.682500	1.000000	9.000000	37.250000	86.000000
4.025000	3.000000	12.000000	42.000000	91.000000
4.220000	5.750000	13.000000	46.750000	96.750000
5.040000	11.000000	21.000000	62.000000	126.000000

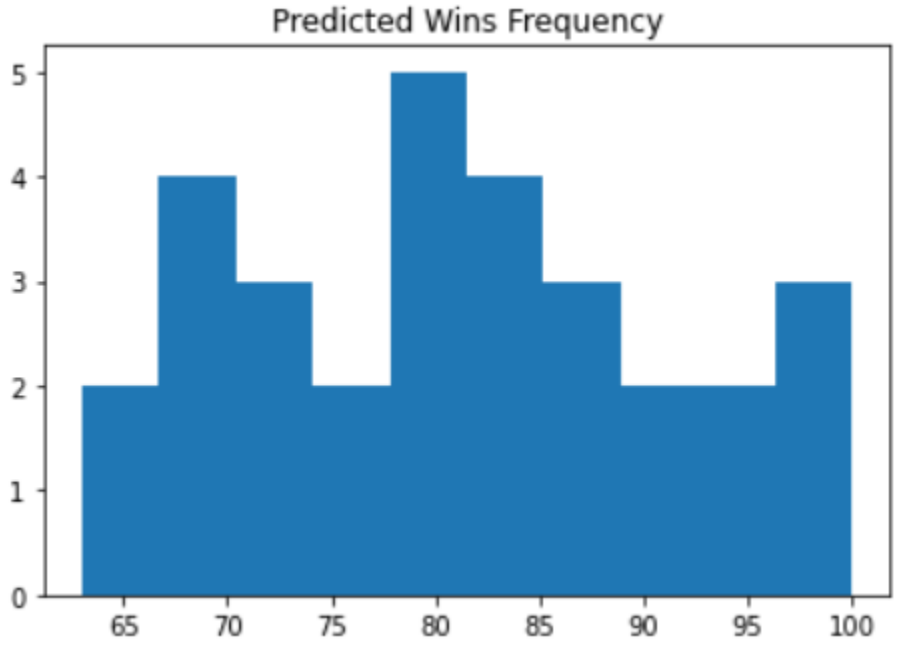
Checked for Any Null Values in Dataset:



No Null Values were found so we can proceed.

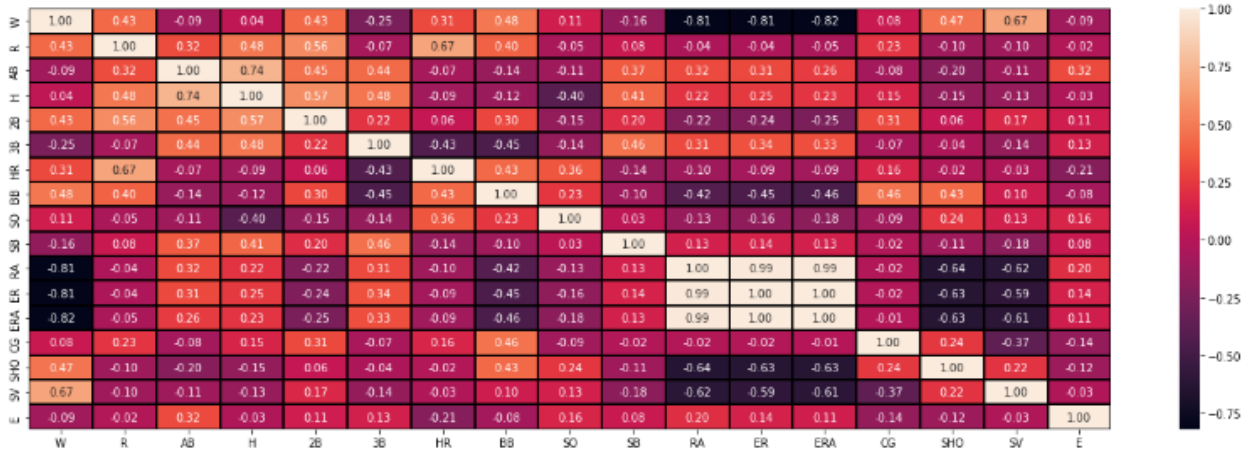
EDA Concluding Remarks

Now we check the Ffrequency of Predicted Wins Values in the Dataset.



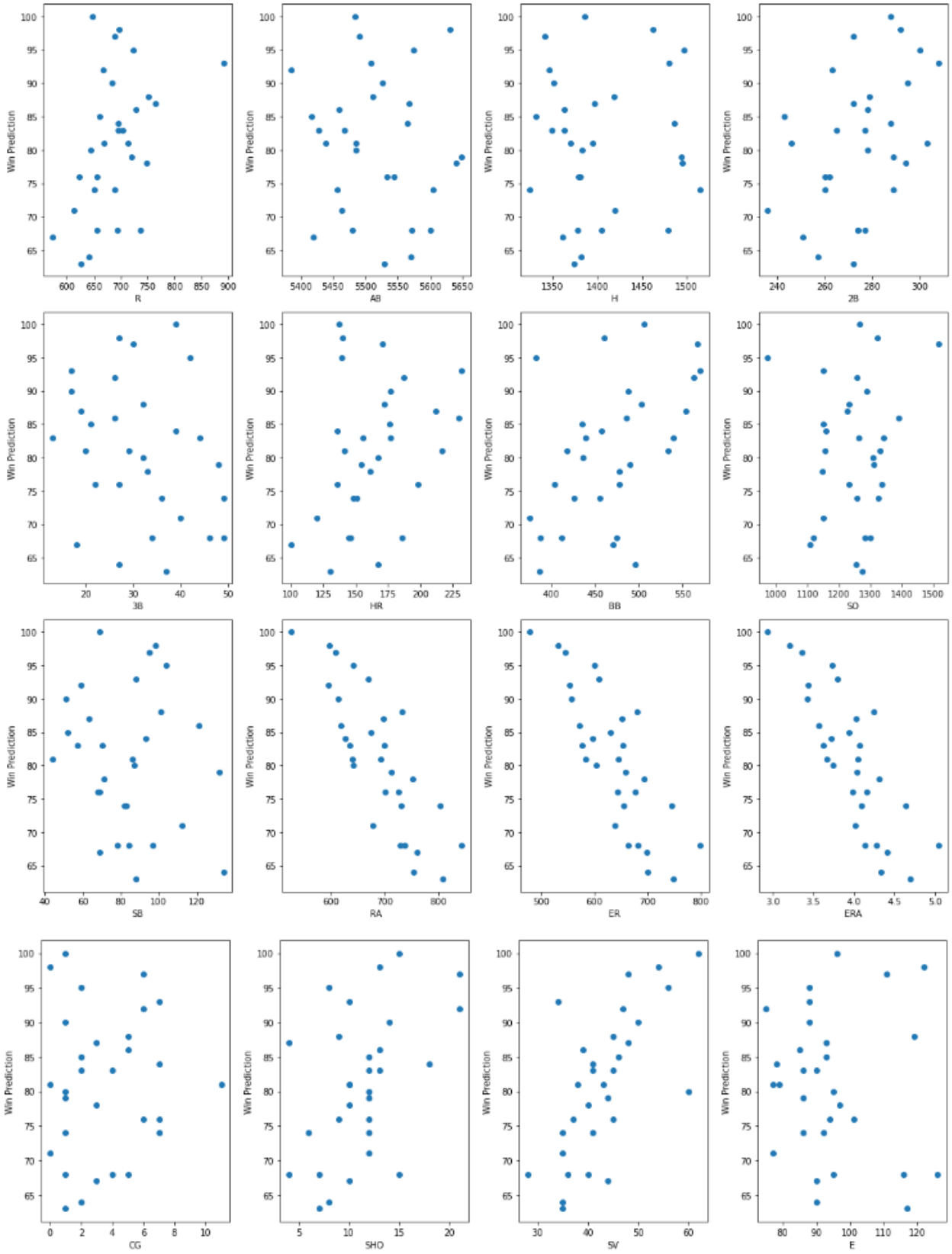
Values are randomly distributed and there is no bias in the dataset.

Now we check the correlation between the different variables in dataset to check if all values are actually independent or not.

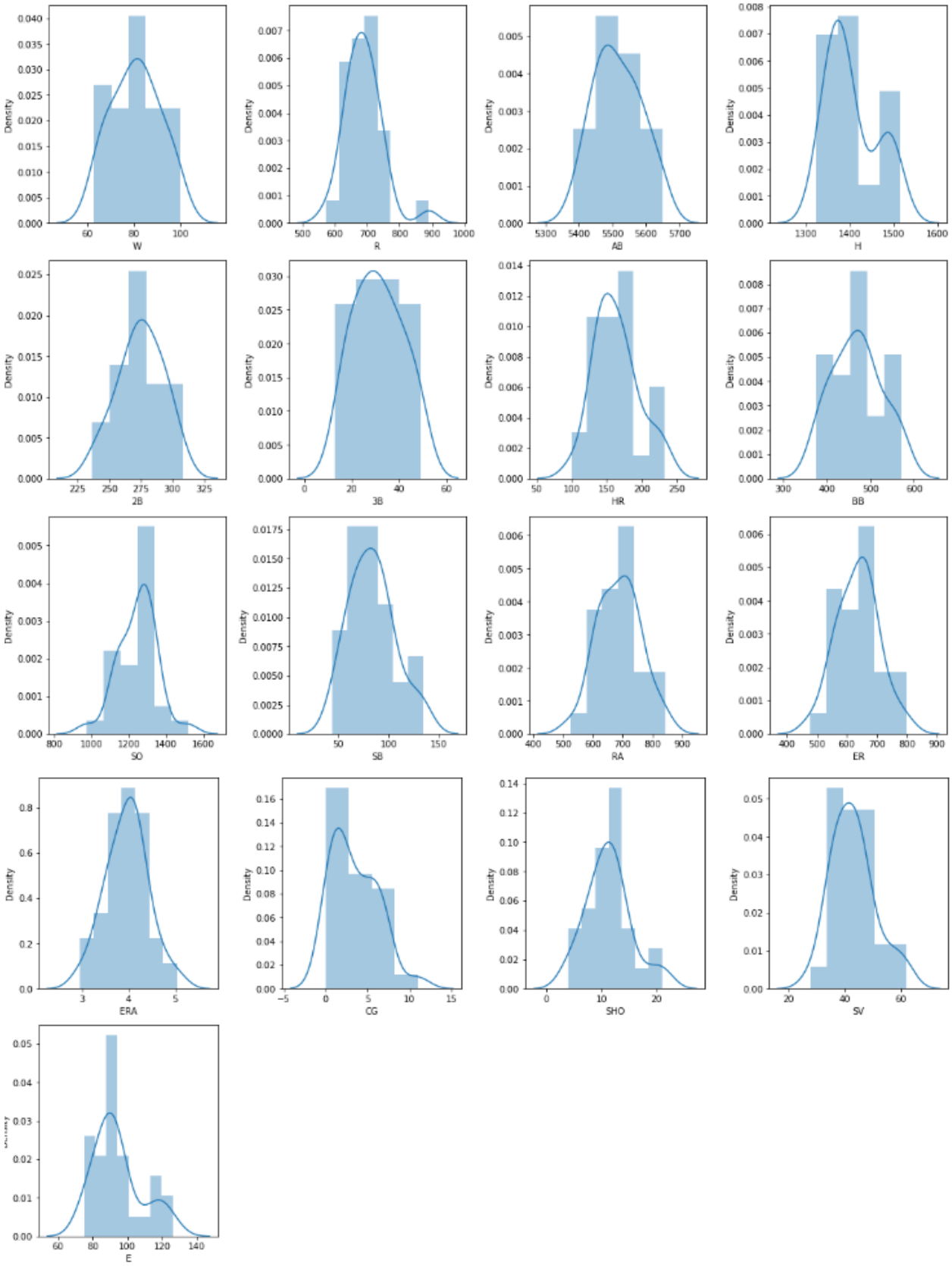


Correlation of dataset is checked through the heatmap as mentioned above.

Plotting Win Prediction against other independent variables to visualise dataset:



Now we will plot density of each independent value to make observations:



Pre-processing Pipeline

For Pre-processing of dataset we have used scaler transformation technique:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
x_scaled
```

```
array([[ 0.61907797,  0.84773126,  1.66368512,  1.42017307,  1.04119304,
        -0.7872986 , -1.53490242, -2.69762957,  0.91388329, -0.66623393,
        -0.50511008, -0.50695519, -0.5398062 , -0.81462859,  1.67160651,
        -0.46146963],
       [ 0.13443166, -0.71109353, -0.97068076,  0.12740339,  1.23580856,
        -0.24396668, -0.5365919 ,  0.15487844, -0.60182558,  0.16597077,
         0.24892985,  0.25459752, -0.5398062 ,  0.1728    ,  0.24987932,
        -0.60719689],
       [-0.33290586, -1.1152333 , -0.15189137,  1.58879521, -0.22380785,
        -0.7233772 ,  1.13914361, -0.89398189,  0.11144918, -0.68033909,
        -0.75162313, -0.64134684,  2.77264091, -0.32091429, -0.65485616,
        -1.11724227],
       [-1.14641931,  0.24152162, -0.40108814, -0.82812203, -0.41842337,
        -0.8831807 , -1.16053598, -0.16860185, -0.69098493,  0.18007593,
         0.10392217,  0.05301004,  1.3004422 , -0.56777144, -0.78410408,
         0.48575751],
       [ 0.01327008,  1.28073815,  1.98408098,  0.80189192,  1.72234737,
        -0.40377019, -0.25136033,  0.10586628, -0.02228984,  1.61880269,
         1.59750126,  1.53131824,  1.3004422 ,  0.1728    , -1.04259994,
```

Now we will split the dataset into train and test samples.

```
# splitting dataset in test and train sets
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size=0.25,random_state=5)
y_train.head()
```

```
26    84
21    83
28    74
0     95
17    97
Name: W, dtype: int64
```

The ratio we have used for splitting the dataset into test adn train dataset is 3:1.
75% of dataset will be used to train the prediction model and 25% will be used as test sample to crosscheck the model.

Building Machine Learning Models

Now for making prediction model, we are going to use Ridge Regression technique as the dataset is overfitting and there are many independent variables which are affecting the No. of Predicted Wins.

```
from sklearn.linear_model import Ridge,RidgeCV
```

```
ridgecv = RidgeCV(alphas=np.arange(0.001,0.1,0.01),normalize=True)
ridgecv.fit(x_train,y_train)
```

```
RidgeCV(alphas=array([0.001, 0.011, 0.021, 0.031, 0.041, 0.051, 0.061, 0.071, 0.081,
       0.091]),
        normalize=True)
```

```
ridgecv.alpha_
```

```
0.09099999999999998
```

```
# making ridge regression model as data is overfitting and many variables influence wins (W)
ridge_model = Ridge(alpha=ridgecv.alpha_)
ridge_model.fit(x_train,y_train)
```

```
Ridge(alpha=0.09099999999999998)
```

Now that we have made the train sets of x and y from splitting the dataset in earlier steps, we have trained the regression model.

Concluding Remarks

After training the model, we can check the accuracy of the model by comparing it to the test set.

```
# checking score of model made for win prediction
ridge_model.score(x_test,y_test)
```

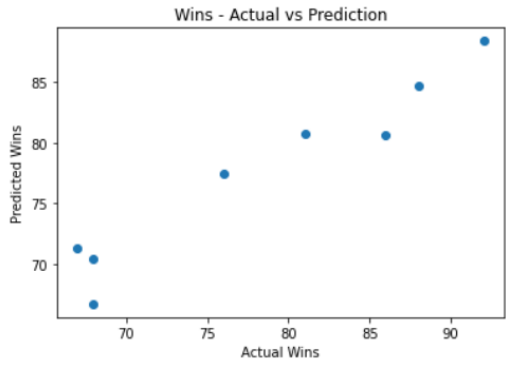
```
0.8825915609588604
```

As we can see from the above image that the model we have trained gives us **88.25% Accuracy**.

Now we plot the prediction value vs the actual value in dataset:

```
y_pred = ridge_model.predict(x_test)
```

```
plt.scatter(y_test,y_pred)
plt.xlabel('Actual Wins')
plt.ylabel('Predicted Wins')
plt.title('Wins - Actual vs Prediction')
plt.show()
```



Finally after making prediction model and checking its accuracy, we are satisfied with the results. Therefore we can now save the model in the form of a pickle file:

```
# creating pickle file to save model
filename= 'baseball_final_model.pickle'
pickle.dump(ridgecv,open(filename,'wb'))
```

```
loaded_model = pickle.load(open(filename,'rb'))
```

END