**Data Analytics to predict the wins of Base Ball League in 2014**

Baseball is a game which is played between two groups of nine players each. These two teams alternate batting and fielding. The batting team attempts to attain runs by taking turns batting a ball that's thrown by the pitcher of the fielding team, then running counter-clockwise around a series of 4 bases: first, second, third, and residential plate. The fielding team tries to stop runs by getting hitters or base runners call at any of several ways and a run (R) is scored when a player advances round the bases and returns to home base . A player on the batting team who reaches base safely will plan to advance to subsequent bases during teammates’ turns batting, like on successful (H), stolen base (SB), or by other means.

In this project, we will test out several machine learning models from sklearn to predict the number of games that a Major-League Baseball team won in that season, based on the teams statistics and other variables from the dataset provided.

We can build a model using historic data from previous seasons to forecast the upcoming Major Baseball League. Using the provided statistic datasets, we can easily generate indicators like average wins per annum over the previous five years, and other such factors, to form a highly accurate model.

**Problem Definition**

To recognize the purpose of the problem and the prediction target, we must define the project objectives appropriately. Therefore, to proceed with an analytical approach, we must acknowledge the obstacles first. In this project, we have used the machine learning model to predict the number of wins a baseball team by using the provided dataset.

**For using using the provided baseball dataset regarding our prediction we need to follow the given steps:**

1. ***Importing the data***
2. ***Cleaning and preparing the data***
3. ***Exploring and visualizing the data***
4. ***Splitting dataset into train\_set and test\_set***
5. ***Selecting error metric: mean absolute error vs mean squared error***
6. ***Using varous algorithms:***

***a. LinearRegression***

***b. Lasso***

***c. KNeighborsRegressor***

***d. AdaBoostRegressor***

***e. DecisionTreeRegressor***

1. ***Using GridSearchCV for best parameter***
2. ***Find the cross val score***

**Importing the data**

Importing of data is done for the use of Python Standard Library that has provided the functions to interact and concluded. Python has bundle of libraries for different tasks. Importing is not only a matter of using external libraries, but it also allows you to keep your code clean and organized.

**Cleaning and preparing the data**

After importing the dataset, it has been read and null value has been analysed. Heatmap has been used to find all the null values in the dataset. Even null values are found in the dataframe.

We have eliminated the rows where the columns have a small number of null values, which has given us to lose approx… five percent of our data. Meanwhile, we need to predict baseball wins, runs scored and runs permitted are extremely correlated with the target.

So we want data in those columns to be very accurate.

Thus, for better off we need to keep the rows and filling the null values with the median value from each of the columns to get accuracy.

As for example; we have found the null value using heatmap:



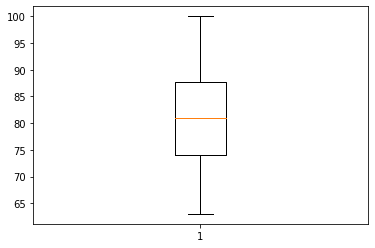
Apart from replacing the data with the median values, some of the columns have been dropped to get the maximum accuracy level.

**Exploring and visualizing the data**

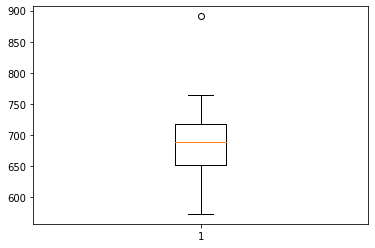
**Handling data outliers and skewness**

We have thoroughly checked distribution of column and relationship between others. Then, accordingly we have dandled the data outliers, worked out for correlation columns and skewness within it. We have used boxplot and found many outliers in columns.

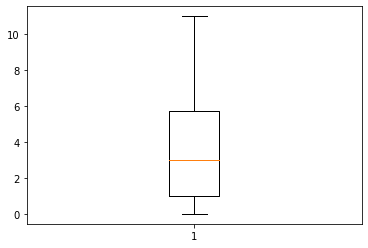
Some of them with their graphical representations are as follows;



Graphical representation of BoxPlot for W



Graphical representation of BoxPlot for R

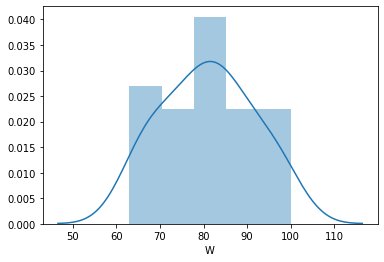


Graphical representation of BoxPlot for CG

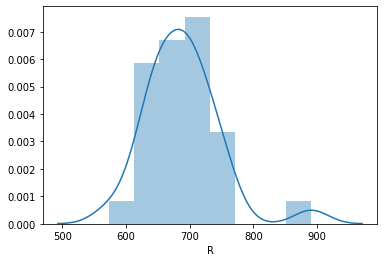
After using boxplot, many outliers have been found in columns.

**Skewness subject wise**

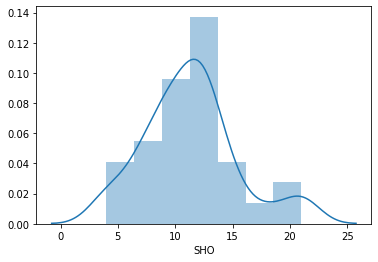
We have Checked the distribution of columns data and found the data skewed is positive. The graphical representation of some of the data are as follows;



Graphical representation of Skewness for W



Graphical representation of Skewness for R

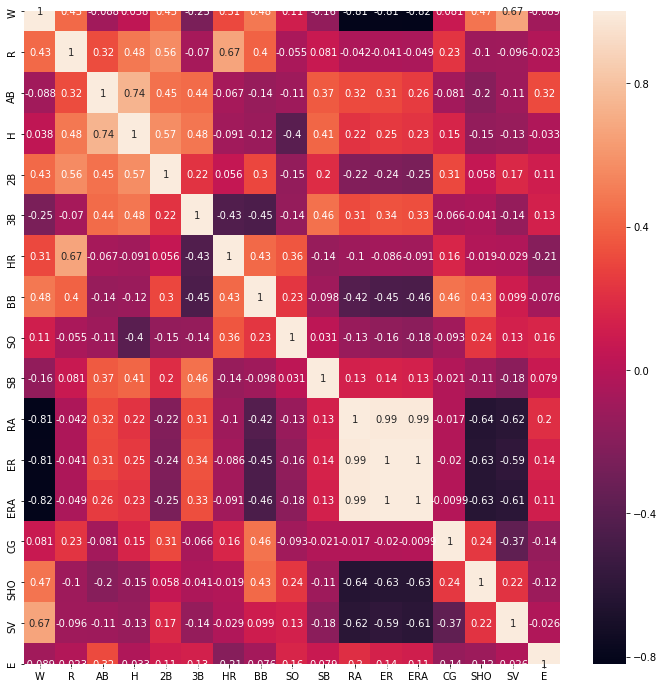


Graphical representation of Skewness for SHO

According to the graph we have found the data are alleviating and deteriorating which shows the skewness of data.

**Checking Correlation columns**

We have used the heatmap to check the correlation between the data and we found there are some columns is highly correlated (AB, E, RA).



**Splitting dataset into train\_set and test\_set**

We have splitted the data to prepare the machine for its learning to provide the output. This will help the machine to provide the output on the basis of trained dataset.

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=47,test\_size=0.20)

lg.fit(x\_train,y\_train)

y\_pred=lg.predict(x\_test)

print('RMSE is: ',np.sqrt(mean\_squared\_error(y\_test,y\_pred)))

print('MAE is: ',mean\_absolute\_error(y\_test,y\_pred))

print('r2\_score is:',r2\_score(y\_test,y\_pred))

print('cross val score Mean:',cross\_val\_score(ada\_reg,x,y,cv=5,scoring='r2').mean())

print('cross val score STD:',cross\_val\_score(ada\_reg,x,y,cv=5,scoring='r2').std())

print('\n')

print(y\_pred)

print(y\_test)

**Using various algorithms:**

1. **LinearRegression**

**Linear Regression** is generally the first machine learning **algorithm** that every data scientist comes across. It is the simplest model, but everyone needs to master it as Linear Regression lays the foundation for other machine learning **algorithms**. Its use has a main advantage i.e.; the ease of interpreting the results.

1. **Lasso**

Lasso regression may be a sort of rectilinear regression that uses shrinkage. Just like the mean, shrinkage is where data values are shrunk towards the central point. The lasso procedure boosts simple, thin models (i.e. models with fewer parameters). This regression is well-suited for models showing high levels of muticollinearity or once you want to automate certain parts of model.

1. **KNeighborsRegressor**

KNeighborRegressor is a type of regression based on k-nearest neighbors. Local interpolation of the targets associated of the nearest neighbors in the training set to predict the target.

1. **AdaBoostRegressor**

An AdaBoost regressor is a meta-estimator that originates by fitting a regressor on the unique dataset and after that it fits extra copies of the regressor on the selected dataset but where the weights of instances are accustomed conferring the error of the current prediction.

**e. DecisionTreeRegressor**

Decision trees regression normally uses the mean squared error (MSE) to decide to split a node in two or more sub-nodes. Suppose we are doing a binary tree the algorithm first will pick a value, and split the data into two subset. For each subset, we have to calculate the MSE separately.

**Using GridSearchCV for best parameter**

GridSearchCV used to find the best parameter for different algoriths to retrieve the best output. As for example;

neighbors={'n\_neighbors':range(1,20)}

knr=KNeighborsRegressor()

gknr=GridSearchCV(knr,neighbors,cv=10)

gknr.fit(x,y)

gknr.best\_params\_

{'n\_neighbors': 6}

We have used in the above algorithm KNeighborRegressor to find the nearest neighborhood. In the above example we have found “6th” as the nearest neighbor.

Similarly, we have used in various algoriths like LASSO to find the alpha value and in AdaBoostRegression to provide the various parameter base\_estimator, learning rate & n estimator etc. to improvise the AdaBoost algorithm.

**GridSearch used in AdaBoostRegressor in the below;**

{'base\_estimator': Lasso(alpha=0.001, copy\_X=True, fit\_intercept=True, max\_iter=1000,

normalize=False, positive=False, precompute=False, random\_state=None,

selection='cyclic', tol=0.0001, warm\_start=False),

'learning\_rate': 1,

'n\_estimators': 100}

**Find the cross val score**

We have used cross val score in each & every algorithm to analyse the effectiveness of our model’s performance. Our model delivered a positive result on validation data in LASSO algorithm.

lsreg=Lasso(alpha=0.001)

r\_state=maxr2\_score(lsreg,x,y)

print('Mean r2 score for Lasso Regression',cross\_val\_score(lsreg,x,y,cv=5,scoring='r2').mean())

print('stander deviation in r2 score for Lasso Regression',cross\_val\_score(lsreg,x,y,cv=5,scoring='r2').std())

max r2 score corrasponding to 47 is 0.9933581573456538

Mean r2 score for Lasso Regression 0.34933865479216253

stander deviation in r2 score for Lasso Regression 0.6225490432857381

Form the above, we have found the positive values for LASSO algorithm.

Even the mean squared error & mean absolute error are less in comparison to other algorithm, thus we have finalized the LASSO as the best performing algorithm for providing the highest level of accuracy for machine learning.

**Conclusion:**

Depending upon the different features that were used as the inputs to the machine learning to predict the number of wins a baseball team. We have used the EDA process to find out the correlated columns and data processing. And later we have used the different algorithms like linear regression, KNeighborsRegressor, LASSO & AdaBoostRegressor. Further, we have used the GridSearchCV parameter to get the best result for the algorithm.

After using all the above approaches we have concluded that LASSO provides the highest level of accuracy for machine learning i.e., approx... 99% regarding the baseball league for the year 2014.