# Baseball Case Study



Problem Definition.

It is a sports case study where data for 2014 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. It will help the team management to replace the team member.

# About Data

R: Runs,

AB: At Bats

H: Hits

2B: Doubles

3B: Triples

HR: Homeruns

BB: Walks

SO: Strikeouts

SB: Stolen Bases

RA: Runs Allowed

ER: Earned Runs

ERA: Earned Run Average (ERA)

CG: Shutouts

SV: Saves,

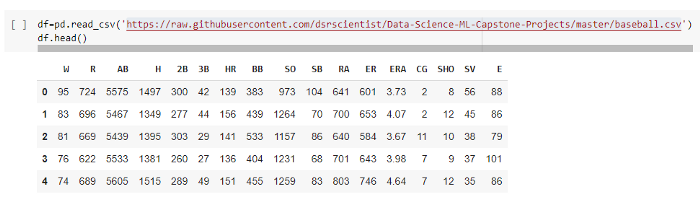
SV: Complete Games

E: Errors

W: Win

**Data Analysis and EDA Concluding Remark.**

[Importing data](https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/baseball.csv)(click to get the data)



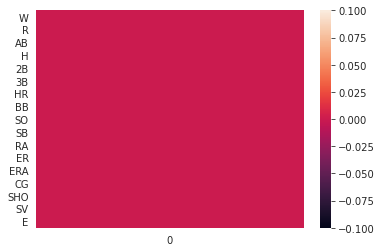
The shape of the data frame.

df.shape

(30, 17)

checking for null value

sns.heatmap(df.isnull().sum().to\_frame())



Basic information about dataset

df.info()

# 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

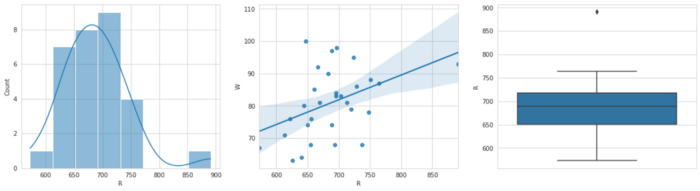
16 E 30 non-null int64

No need to change the datatype they are appropriate so we can skip the step of data conversion Now directly jumps to Visualization

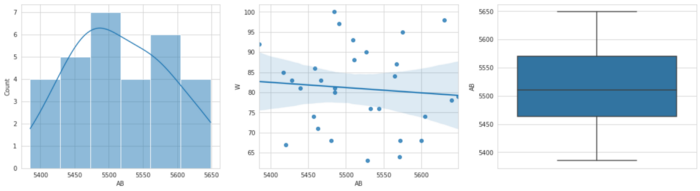
**Visualization**

Now, let’s visualize the distribution of continuous features

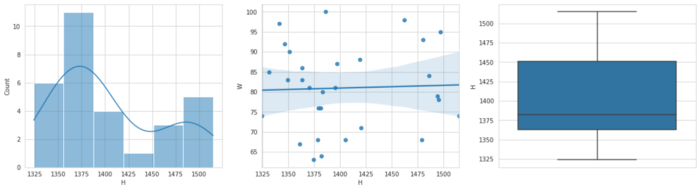
def pplot(df,i):plt.figure(figsize=(20,5))plt.subplot(1,3,1)sns.histplot(x=i,data=df,kde=True)plt.subplot(1,3,2)sns.regplot(x=i,y=’W’,data=df)plt.subplot(1,3,3)sns.boxplot(y=i,data=df)plt.show()



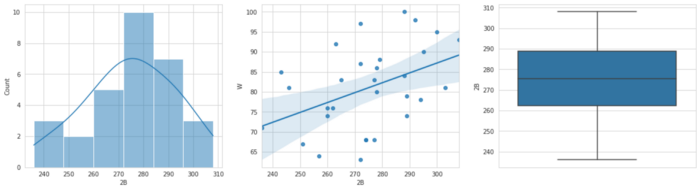
Run and win are linearly correlated. And outliers are present between 850 and 900



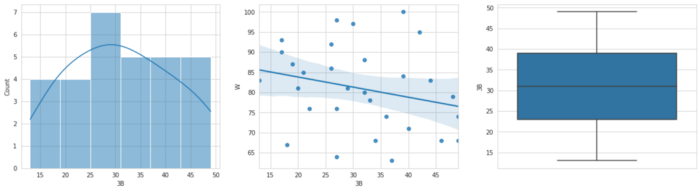
At balls is very weekly related to Wins With no outliers.



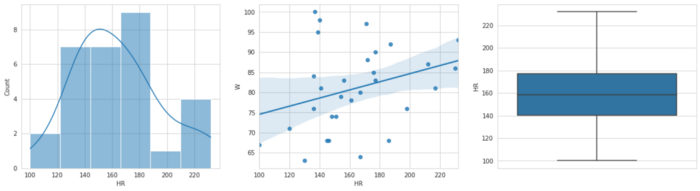
Hits are also very weekly related to winning with no outliers



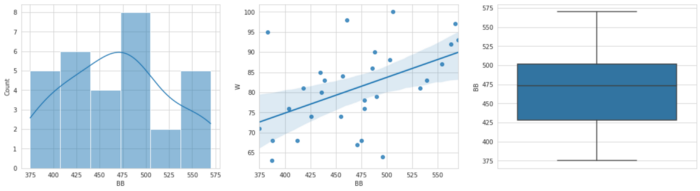
Doubles are linearly related with Wins, with no outlier, and data is left-skewed



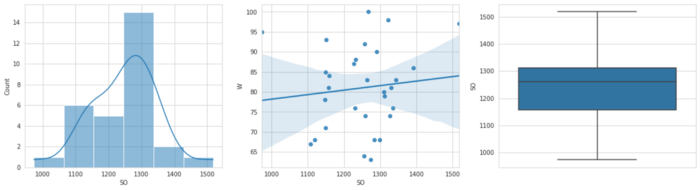
Triple has very weak negative correlations with win with no outlier and data is right-skewed



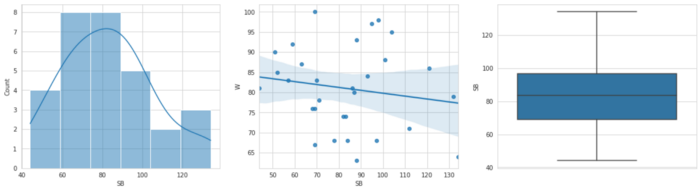
The home run has a very low correlation with wins with no lot of outliers and data is right-skewed



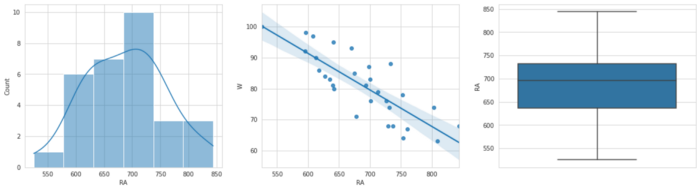
Walk is lightly correlated to win with no outlier



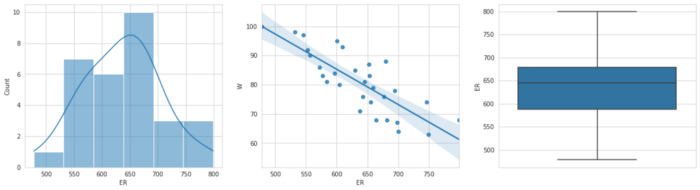
Strikeout is not correlated to win with no outlier and the dataset is left-skewed.



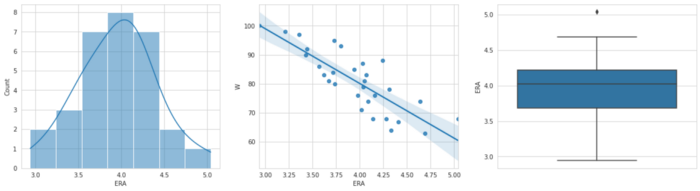
Stolen Bases is not correlated to win with no outlier



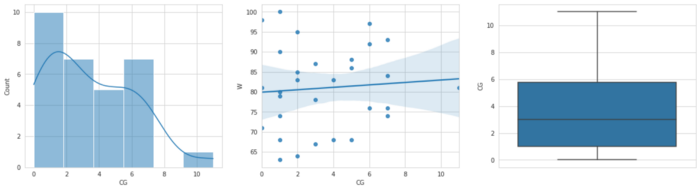
Stolen Bases is not correlated to win with no outlier



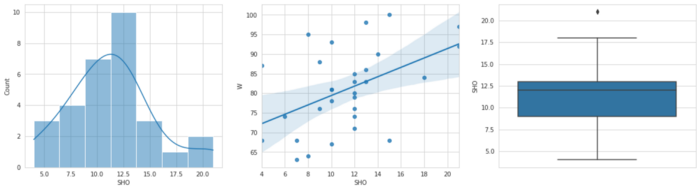
Runs Allowed is highly correlated to win with no outliers



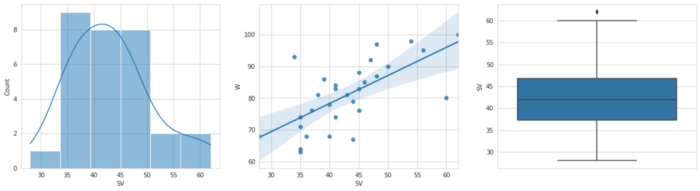
Earned Runs is highly correlated to win with no outliers



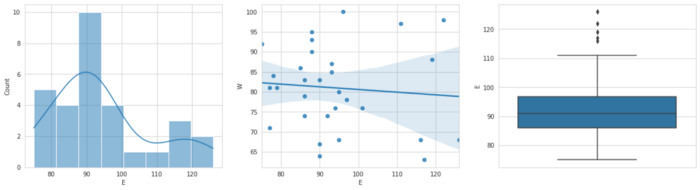
Earned Run Average (ERA) is not correlated to win with no outliers and data is right-skewed.



Shutouts are highly correlated to win with no outliers and data is right-skewed.



Saves are lightly correlated to win with no outliers-skewed.



Errors are not correlated to win, the data has outliers and data is right-skewed.

Now when numerical data are analyzed it’s time for some stories.

**Storytelling:**

The number of runs, home runs, doubles, Saves, Shutouts, and Walks are highly positively linearly correlated.

Stolen Bases, Runs Allowed, Earned Runs are highly negative linearly correlated.

The remaining features have less to no linear correlation with no of Wins

The dataset set has too much randomness. And only has sets of data (too little to form a suitable Decision tree)

**Pre-Processing Pipeline.**

As all the data in the set have continuous value the preprocessing steps consist of data transformation (standardization) and scaling (normalization).

**Standardization**is a method to convert the data set into a normal distribution. There are many methods to do it.

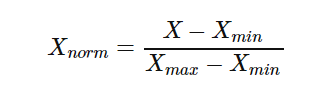
1. Log-transformation — if the data set is highly rightly skewed then it is best to use Log-transformation
2. Square root transformation — if data is a little bit right-skewed then it is best to use square root transform
3. Cube root transformation
4. Reciprocal transformation
5. Box-cox transformation
6. Power transformation

And much more mathematical transformation is there to standardize the data.

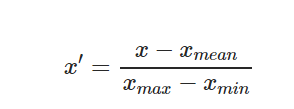
**Normalization**is a mathematical technic that scales the data into some range it is required for some machine learning algorithms such as Linear regression, KNN, etc.

Normalization minimizes the difference between the low valve and high value which helps in better prediction. Some Normalization methods are mentioned below

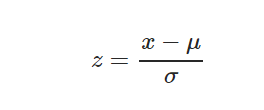
1. Min-Max-Scaler- it converter the dataset into the range of 0 to 1



2. Mean Normalization- it converts the data set into the range of -1 ti 1 and the mean is 0.



3. Standard scaling(z-score)- its converts data set as mean=0, standard deviation as 1.



**Outliers**It can be defined as the odd one out means the data in the dataset having some odd values. It can be too low or too high. There are two main methods to remove outliers:

1. **Z-score method**- it covert the dataset into z-sore and then check if absolute of z\_score is greater than 3 then remove those data. In a standard normal distribution, 99.73% of data should lie within 3 standard deviations i.e 3(the standard, deviation is 1).

z=np.abs(stat.zscore(df\_z[num]))a=int(df\_z.size)df\_z = df\_z[(z>3).all(axis=1)]

**2. IQR method-**IQR stands for the interquartile reason it uses percentile for outlier detection 50 percentile represents the median, 25 percentile represents the Q1, 75 percentile represents Q2.

Upper IQR=1.5\*(Q2-Q1)+Q2

Lower IQR=Q1–1.5\*(Q2-Q1)

Remove any data outside upper IQR and lower IQR

Q1 = np.percentile(df\_boston[‘DIS’], 25,interpolation = ‘midpoint’)Q3 = np.percentile(df\_boston[‘DIS’], 75,interpolation = ‘midpoint’)IQR = Q3 — Q1print(“Old Shape: “, df\_boston.shape)# Upper boundupper = np.where(df\_boston[‘DIS’] >= (Q3+1.5\*IQR))# Lower boundlower = np.where(df\_boston[‘DIS’] <= (Q1–1.5\*IQR))‘’’ Removing the Outliers ‘’’df\_boston.drop(upper[0], inplace = True)df\_boston.drop(lower[0], inplace = True)

**Data Reduction**— This dataset has continuous independent value and dependent value. There are many technics for Data reduction

1. **Pearson r correlation:** Pearson r correlation is the most widely used correlation statistic to measure the degree of the relationship between linearly related variables.

sns.heatmap(pt1.corr(),annot=True,cmap=’PiYG’)

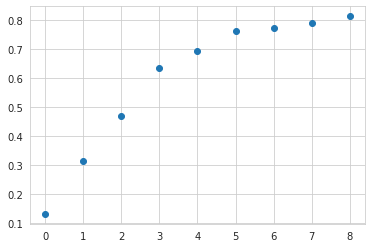


**2. ANOVA:**Analysis of variance (ANOVA) is an analysis tool used in statistics that splits an observed aggregate variability found inside a data set into two parts: systematic factors and random factors. The systematic factors have a statistical influence on the given data set, while the random factors do not. Analysts use the ANOVA test to determine the influence that independent variables have on the dependent variable in a regression study**.**

**from sklearn.feature\_selection import SelectKBestfrom sklearn.feature\_selection import f\_classifs = SelectKBest(f\_classif, k=15)s.fit(x,y)anova=pd.DataFrame([s.scores\_,s.pvalues\_],columns=x.columns).T.sort\_values(by=0)**

**3. R-squared (R2):**It is a statistical measure that represents the proportion of the variance for a dependent variable that is explained by an independent variable. R-squared explains to what extent the variance of the dependent variable explains the variance of the dependent variable. So, if the R2 of a model is 0.50, then approximately half of the observed variation can be explained by the model’s inputs.

cv\_results = cross\_validate(lr(normalize=True), x, y, cv=3,scoring=’r2')lr\_reg=cv\_results[‘test\_score’].mean()r2=[]r2.append(lr\_reg)j=1k=1least\_r21=-10while j==1:for i in x.columns:demo\_x=x.drop(i,axis=1)cv\_results = cross\_validate(lr(normalize=True), demo\_x, y, cv=3,scoring=’r2')lr\_reg1=cv\_results[‘test\_score’].mean()if lr\_reg1>lr\_reg:least=ileast\_r2=lr\_reg1lr\_reg=lr\_reg1if least\_r21<least\_r2:print(str(least) + ‘is having r2 score of’ + str(least\_r2))least\_r21=least\_r2x.drop(least,axis=1,inplace=True)r2.append(lr\_reg)k=k+1else:j=0plt.scatter(range(k),r2)



R-square Vs Variables droped

In the above graph, it is clear that removing some variable R-square increases significantly which means randomness in the dataset has decreased

Pre-Processing pipeline is a step-by-step process of pre-processing required for a dataset for making it more understandable and recognizable by the computer to perform machine learning.

The whole step-by-step process is packed in a set that can be created in function, class, or using pipeline from sklearn library.

preprocessing helps in reducing the noise or randomness from data. It consists of the following elements

1. **Data Cleaning:-**removing the Null value
2. **Data Integration:-** integrates data from a multitude of sources into a single data warehouse.
3. **Data Transformation:-**Transforming data such as standardization, normalization, encoding, etc
4. **Data Reduction:-**removing redundant data.

def pipeline\_viz(df):num,cat=num\_cat(df)num\_plot(df,num)df[num]=pow\_tran(df,num)df[num]=stan\_sc(df,num)df[num]=z\_outlier(df,num)df\_new=dimention\_reduction(df,num)return (df\_new)

Enable function step by step prepossessing functions are called to make a pipeline first function segregates numerical and categorical data, the second function plots the numerical data for the analysis, The third function standardizes the numerical data, the Forth function normalizes the numerical data, the fifth function to remove the outlier from the numerical data set and the last function is used for dimension reduction This function is the pipeline for preprocessing and visualization.

def num\_cat(df):num=[]cat=[]count=df.nunique()for i in df.columns:if count[i]>5:num.append(i)else:cat.append(i)return(num,cat)from sklearn.preprocessing import power\_transform as PTdef pow\_tran(df,num):pt=pd.DataFrame()for i in num:if df[i].min()<=0:pt1=(df[i]-df[i].min()+0.0001)else:pt1=df[i]pt=pd.concat([pt,pd.DataFrame(pt1)],axis=1)pt1=PT(pt)pt1=pd.DataFrame(pt1,columns=num)return(pt1)def num\_plot(df,num):for i in num:plt.figure(figsize=(20,5))plt.subplot(1,3,1)sns.histplot(x=i,data=df,kde=True,hue=’Churn’)plt.subplot(1,3,2)sns.regplot(x=i,y=’W’data=df,kde=True,hue=’Churn’)plt.subplot(1,3,3)sns.boxplot(y=i,data=df)plt.show()def z\_outlier(df,num):df\_z=df[num]z=np.abs(stat.zscore(df\_z))a=int(df\_z.size)df\_z = df\_z[(z>3).all(axis=1)]print(‘Percent of data retained = ‘+ str(int(df\_z.size)/a))return(df\_z)def stan\_sc(df,num):ss=StandardScaler()x=df[num].to\_numpy()x1=pd.DataFrame(ss.fit\_transform(x),columns=num)return(x1)

**Building Machine Learning Models.**

Machine learning is as good as preprocessing. Now let’s take a look into some algorithms of machine learning for classification.

1. Linear regression — it is an algorithm that minimizes the loss function i.e summation of square error.
2. Decision Tree — It is a tree-based algorithm that uses the gini or entropy method to generate the tree.
3. Random forest- It is a multiple trees algorithm that uses bootstrap sampling and bagging for classification.
4. Extreme Gradient boosting — it is a boosting algorithm that uses a multi-decision tree.
5. Support vector machine- it constructs Hyperplane and marginal plane and distance between them is optimized.

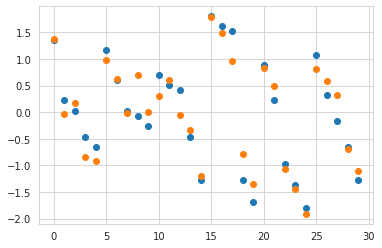
There are multiple machine learning algorithms to select from. Based on the quite best method to select the algorithm having the minimum difference between predicted score and cross-validation score.

def clf(model,x,y,x\_test,y\_test):lr = model.fit(x, y)y\_pred=lr.predict(x\_test)print(confusion\_matrix(y\_test,y\_pred))print(accuracy\_score(y\_test,y\_pred))print(classification\_report(y\_test,y\_pred))lr\_acc=f1\_score(y\_test,y\_pred)cv\_results = cross\_validate(model, x, y, cv=10)lr\_score=cv\_results[‘test\_score’].mean()return(lr\_acc,lr\_score)lr\_acc,lr\_score=clf(LinearRegression(),x\_train\_scaler,y\_train,x\_test\_scaler,y\_test)rfc\_acc,rfc\_score=clf(RFR(),x\_train\_scaler,y\_train,x\_test\_scaler,y\_test)xgc\_acc,xgc\_score=clf(XBR(),x\_train\_scaler,y\_train,x\_test\_scaler,y\_test)etc\_acc,etc\_score=clf(ETR(),x\_train\_scaler,y\_train,x\_test\_scaler,y\_test)svc\_acc,svc\_score=clf(SVC(),x\_train\_scaler,y\_train,x\_test\_scaler,y\_test)score=[lr\_score,rfc\_score,xgc\_score,etc\_score,svc\_score]error=[lr\_acc,rfc\_acc,xgc\_acc,etc\_acc,svc\_acc]name=[‘LR’,’RFR’,’XGR’,’ETR’,’SVR’]diff=[]for i in range(5):diff.append(score[i]-error[i])pd.DataFrame([name,score,error,diff]).T

Now after selecting the best algorithm now we should do Hyperparameter tuning for extreme gradient boosting.

1. Random search cv:- It randomly Search the grid for the best scoring. Processing time is less but the accuracy is not that great
2. Grid search cv:- it searches the grid one by one for the best scoring. Processing time is more but the accuracy is great

**from sklearn.model\_selection import GridSearchCVridge\_params = {‘alpha’:[1,2,3,4,5,6,7,8,9,10]}xg\_grid = GridSearchCV(Ridge(), ridge\_params, cv=3)xg\_grid.fit(x\_train\_scaler, y\_train)print(‘Best score:’, xg\_grid.best\_score\_)print(‘Best score:’, xg\_grid.best\_params\_)print(‘Best score:’, xg\_grid.best\_estimator\_)**



Y\_actual Vs Predicted

**Concluding Remarks.**

As discussed in EDA and storytelling the randomness of the dataset should be reduced by reducing the number of variables. Which was done by R-square dimension reduction.

The root means square is 0.08 and the R-square value is 91.2.