**Baseball Case Study**

A picture containing sky, outdoor, building, grass

Description automatically generated

It is a case study of very famous sport i.e. Baseball. In this we have used a 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. It will help the management of the team to make changes in the team as per the predicted results.

**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 Raw Data for our Model [https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/baseball.cs](https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/baseball.csv)v

Import Important libraries:

**import** pandas **as** pd  
**import** numpy **as** np  
**import** seaborn **as** sns  
**import** scipy  
**import** sklearn  
**import** warnings  
warnings**.**filterwarnings('ignore')

Table

Description automatically generated

In the above data, we have following attributes. In this we have to predict Wins (W) based on 16 different attributes:

Here our target/output is W i.e. win and rest are inputs. 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 Frame Shape

df**.**shape**(30, 17)**

Dataset has 30 rows and 17 columns.

Now, Checking for data types

df**.**dtypesW int64  
R int64  
AB int64  
H int64  
2B int64  
3B int64  
HR int64  
BB int64  
SO int64  
SB int64  
RA int64  
ER int64  
ERA float64  
CG int64  
SHO int64  
SV int64  
E int64  
dtype: object

In the above inputs and target, have integer and float values. So, we can proceed easily as we don’t have any object datatype.

Now, Checking for Null value in data.

First using info logic:

df**.**info()<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

Now using heatmap for visualizing null values:

sns**.**heatmap(df**.**isnull())

Chart, shape

Description automatically generated

We have seen null values using heatmap. So, no null values are present in the data. We will check it using sum also.

df**.**isnull()**.**sum()

Table

Description automatically generated

So, finally we can conclude that no null values are present.

Description of Data

df**.**describe()

Graphical user interface, application

Description automatically generated with medium confidence

Table, letter

Description automatically generated with medium confidence

As we can see from the above data, that count is equal in all columns, hence no missing data is present. In columns AB,H,HR and E mean is higher than median. It means data is right skewed in these columns. In columns HR,BB,SO,SB,RA,ER there is large gap between 75th and max, there are chances that some outliers may present. High Standard Deviation (SD) in column R,AB,H,BB,SO,RA,ER. It means data is spreaded.

**Outliers Visualization:**

We will use boxplot type of visualization to see the outliers.

df**.**plot(kind**=**'box',subplots**=True**,layout**=**(3,6),figsize**=**(10,10))

A screenshot of a computer

Description automatically generated with low confidence

A picture containing calendar

Description automatically generated

we can see from above Boxplots that some outliers are present in the data.

**Skewness Visualization:**

df**.**skew()

A picture containing text

Description automatically generated

**import** matplotlib.pyplot **as** plt  
df['R']**.**plot**.**hist()

Chart, histogram

Description automatically generated

sns**.**distplot(df['R'],bins**=**10)

Chart, histogram

Description automatically generated

In R column as we can see from dist and hist plot data is right skewed.

Now, we will see combinely for other columns as well using hist plot and pairplot.

**import** matplotlib.pyplot **as** plt  
df**.**hist(bins**=**50, figsize**=**(15,15))  
plt**.**show()

Chart, diagram

Description automatically generated with medium confidence

sns**.**pairplot(df)

Chart, scatter chart

Description automatically generated

As, we can see from above plots that data is skewed for almost all the columns. Majorly is right skewed, we will try to minimize the skewness during our data processing.

**Correlation between the columns:**

df**.**corr()

A picture containing text, window

Description automatically generated

We will see the above result using heatmap visualization as below:

plt**.**figure(figsize**=**(10,10))  
sns**.**heatmap(df**.**corr(),cmap**=**'coolwarm',annot**=True**,annot\_kws**=**{'size': 10})

Chart, treemap chart

Description automatically generated

From above heatmap, we can see that output is in good correlation with many inputs but for some inputs it is not in good correlation. The inputs such as Shutouts, Walks, runs, Saves, Shutouts, home runs and doubles are positively correlated. Earned Runs, Stolen Bases and Runs Allowed are negatively correlated. The dataset has lot of randomness present.

**Pre-Processing Pipeline:**

As we have already checked that all the data present the dataset is of continuous or integer nature, so we can proceed to the pre processing of the data.

Pre-Processing of data consists of steps below steps:

**Multicollinearity using VIF**- When there is a correlation and linear relationship b/w one or more independent inputs. It creates a trouble in the multiple regression as all the inputs are persuading each other. So, we will remove it using Variation Inflation Factor (VIF).

**from** statsmodels.stats.outliers\_influence **import** variance\_inflation\_factorx**=**df**.**iloc[:,1:]  
x**.**head(4)

Text, table

Description automatically generated

y**=**df**.**iloc[:,**-**17]  
y**.**head(4)

A picture containing text

Description automatically generated

**def** vif\_calc():  
 vif**=**pd**.**DataFrame()  
 vif['features']**=**x**.**columns  
 vif['VIF Factor']**=**[variance\_inflation\_factor(x**.**values,i) **for** i **in** range(x**.**shape[1])]  
 print(vif)vif\_calc()

Text

Description automatically generated with medium confidence

From above calculation, we can see that ER, ERA column has high value. So we will drop ER column as in comparison between two, it is contributing less to the output. Same as between AB and H, we will drop AB column to remove multicollinearity.

df**.**drop(['ER','AB'],axis**=**1,inplace**=Tru**

Again we will see the VIF to see the multicollinearity.

**def** vif\_calc():  
 vif**=**pd**.**DataFrame()  
 vif['features']**=**x**.**columns  
 vif['VIF Factor']**=**[variance\_inflation\_factor(x**.**values,i) **for** i **in** range(x**.**shape[1])]  
 print(vif)

Text

Description automatically generated

From above VIF, we can see that RA, ERA column has high value. So we will drop RA column as in comparison between two, it is contributing less to the output.

df**.**drop(['RA'],axis**=**1,,)

**Removing Outliers using Z-score:**

**Outliers**These are the unusual values in the data set, these are far from the mean. They can cause tests to either miss important findings or mislead real results. Here we will use Z-score method to remove outliers.

**Z-score method**- it changes the dataset into z-score and then verify if absolute of z-score is larger than 3, then it will remove all those data having value more than 3. It tries to normalized the data as in a normal distribution, data lies within 3 standard deviations.

**from** scipy.stats **import** zscore  
**import** numpy **as** np  
z**=**np**.**abs(zscore(df))  
z

A picture containing text, newspaper

Description automatically generated

threshold**=**3  
print(np**.**where(z**>**3))



df\_new**=**df[(z**<**3)**.**all(axis**=**1)]df\_new**.**head()

Text, table

Description automatically generated

df**.**shape



df\_new**.**shape



loss\_percentage**=**(30**-**29)**/**30**\***100  
print(loss\_percentage)



df**=**df\_new  
df**.**shape



x**=**df**.**iloc[:,1:]  
x**.**head(4)

Table

Description automatically generated

y**=**df**.**iloc[:,**-**14]  
y**.**head(4)

**Standardization-**It is a method to transform the dataset into a normal distribution. There are several methods to do it.

1. Log-transformation- It can be used if data is highly right skewed.
2. Square root transformation- It can be used if data is little bit right-skewed
3. Cube root transformation
4. Reciprocal transformation
5. Box-cox transformation
6. Power transformation

Here we will use the Power Transform method using yeo-johnson. It is very useful when data has negative values also.

**from** sklearn.preprocessing **import** power\_transformx**=**power\_transform(x,method**=**'yeo-johnson')  
x

So, now skewness is removed or we can say minimized. We can see it by plotting the hist plots.

Chart

Description automatically generated

**Normalization- It**is a mathematical technique which scales the data into the range which is required for many ML algorithms. It tries to minimize the difference b/w the low and high value, which helps in better estimation of the desired output.

Here we will use Standard scaling technique to convert the data set as mean=0 and standard deviation =1.

**from** sklearn.preprocessing **import** StandardScaler  
sc**=**StandardScaler()  
x**=**sc**.**fit\_transform(x)  
x

**Building Machine Learning Models:**

As output is of continuous type, so we will use the ML models as per the output values.

**from** sklearn.linear\_model **import** LinearRegression  
**from** sklearn.metrics **import** mean\_squared\_error,mean\_absolute\_error  
**from** sklearn.metrics **import** r2\_score  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.linear\_model **import** Lasso,Ridge  
**from** sklearn.linear\_model **import** ElasticNet  
**from** sklearn.linear\_model **import** ElasticNet



x\_train**.**shape



x\_test**.**shape



y\_train**.**shape



Firstly we will use linear regression model and see the outcome.

**Linear Regression Model:**It is an algorithm which minimize the summation of square error i.e. loss function.

lr**=**LinearRegression()  
lr**.**fit(x\_train,y\_train)  
lr**.**score(x\_train,y\_train)



pred**=**lr**.**predict(x\_test)  
print('Predicted result price:',pred)  
print('actual price',y\_test)  
print('error:')  
print('Mean absolute error:',mean\_absolute\_error(y\_test,pred))  
print('Mean squared error:',mean\_squared\_error(y\_test,pred))  
print('Root Mean Squared Error:',np**.**sqrt(mean\_squared\_error(y\_test,pred)))

Text, letter

Description automatically generated

print(r2\_score(y\_test,pred))



Here in linear regression model, score is coming out as 96.37% and r2 score 82.22%, we will use other models to further check the predicted score and r2 score values.

**Lasso Model:**Lasso (least absolute shrinkage and selection operator) is a regression analysis method which accomplishes both regularization and variable selection to increase the prediction correctness and interpretability of the resulting statistical model.

ls**=**Lasso(alpha**=**0.0001)  
ls**.**fit(x\_train,y\_train)  
ls**.**score(x\_train,y\_train)



predls**=**ls**.**predict(x\_test)  
print('Predicted result price:',predls)  
print('actual price',y\_test)  
print('error:')  
print('Mean absolute error:',mean\_absolute\_error(y\_test,predls))  
print('Mean squared error:',mean\_squared\_error(y\_test,predls))  
print('Root Mean Squared Error:',np**.**sqrt(mean\_squared\_error(y\_test,predls)))

Text, letter

Description automatically generated

print(r2\_score(y\_test,predls))



In Lasso model, score is coming out as 96.37% and r2 score 87.88%.

**Ridge Regression Model:**It is a method used for evaluating multiple regression where the data undergoes multicollinearity. It evades overfitting, it works just like the linear regression, but it just add up an additional term (α) which assists in the reduce the overfitting.

rd**=**Ridge(alpha**=**0.0001)  
rd**.**fit(x\_train,y\_train)  
rd**.**score(x\_train,y\_train)



predrd**=**rd**.**predict(x\_test)  
print('Predicted result price:',predrd)  
print('actual price',y\_test)  
print('error:')  
print('Mean absolute error:',mean\_absolute\_error(y\_test,predrd))  
print('Mean squared error:',mean\_squared\_error(y\_test,predrd))  
print('Root Mean Squared Error:',np**.**sqrt(mean\_squared\_error(y\_test,predrd)))

Text, letter

Description automatically generated

print(r2\_score(y\_test,predrd))



In Ridge model, score is coming out as 96.37% and r2 score 82.22%.

**ElasticNet Model:**

enr**=**ElasticNet(alpha**=**0.0001)  
enr**.**fit(x\_train,y\_train)  
enr**.**score(x\_train,y\_train)



predenr**=**enr**.**predict(x\_test)  
print('Predicted result price:',predenr)  
print('actual price',y\_test)  
print('error:')  
print('Mean absolute error:',mean\_absolute\_error(y\_test,predenr))  
print('Mean squared error:',mean\_squared\_error(y\_test,predenr))  
print('Root Mean Squared Error:',np**.**sqrt(mean\_squared\_error(y\_test,predenr)))

Graphical user interface, text

Description automatically generated

print(r2\_score(y\_test,predenr))



In ElasticNet r2 score is 82.24%, which is lesser than lasso i.e is 87.88%.

Here best r2 score is of Lasso model, we will try to improve it by by finding best alpha value using GridsearchCV.

**Cross Validation:**

We will use the Cross Validation technique to find the scores and match it with the already calculated scores of all above models. It is to see that our model is working fine, it not overfitting or underfitting.

**from** sklearn.model\_selection **import** cross\_val\_scorescore**=**cross\_val\_score(lr,x,y,cv**=**8)  
print(score)  
print (score**.**mean())  
print (score**.**std())

Text

Description automatically generated

score**=**cross\_val\_score(ls,x,y,cv**=**8)  
print(score)  
print (score**.**mean())  
print (score**.**std())

Text

Description automatically generated

score**=**cross\_val\_score(rd,x,y,cv**=**8)  
print(score)  
print (score**.**mean())  
print (score**.**std())

Text

Description automatically generated

From the above observations, we can see that our scores are OK. We can now proceed further to do the Hyperparameter tuning and import the outcome.

**Hyperparameter tuning:**

Her we will select our 2 best model i.e. Lasso and Ridge and try to improve its outcome.

**from** sklearn.model\_selection **import** GridSearchCVgrid**=** GridSearchCV(estimator**=**modelrd,param\_grid**=**alphavalue)alphavalue**=**{'alpha':[1,0.1,0.01,0.001,0.0001,0]}  
modelrd**=**Ridge()  
modells**=**Lasso()grid**.**fit(x,y)  
print(grid)  
print(grid**.**best\_score\_)  
print(grid**.**best\_estimator\_**.**alpha)  
print(grid**.**best\_params\_)

Text

Description automatically generated

grid1**=** GridSearchCV(estimator**=**modells,param\_grid**=**alphavalue)  
grid1**.**fit(x,y)  
print(grid1)  
print(grid1**.**best\_score\_)  
print(grid1**.**best\_estimator\_**.**alpha)  
print(grid1**.**best\_params\_)

Text

Description automatically generated

By using gridsearchCV, alpha value 1 comes out to be best value for Lasso and Ridge . So, we will predict the result using it.

**Lasso Model using alpha=1:**

ls1**=**Lasso(alpha**=**1)  
ls1**.**fit(x\_train,y\_train)  
ls1**.**score(x\_train,y\_train)



predls1**=**ls1**.**predict(x\_test)  
print('Predicted result price:',predls1)  
print('actual price',y\_test)  
print('error:')  
print('Mean absolute error:',mean\_absolute\_error(y\_test,predls1))  
print('Mean squared error:',mean\_squared\_error(y\_test,predls1))  
print('Root Mean Squared Error:',np**.**sqrt(mean\_squared\_error(y\_test,predls1)))

Text

Description automatically generated

print(r2\_score(y\_test,predls1))



r2 score of Lasso remains the same after using alpha=1 also, so we will use any of the lasso model.

**Ridge Model using alpha=1:**

rd1**=**Ridge(alpha**=**1)  
rd1**.**fit(x\_train,y\_train)  
rd1**.**score(x\_train,y\_train)



predrd1**=**rd1**.**predict(x\_test)  
print('Predicted result price:',predrd1)  
print('actual price',y\_test)  
print('error:')  
print('Mean absolute error:',mean\_absolute\_error(y\_test,predrd1))  
print('Mean squared error:',mean\_squared\_error(y\_test,predrd1))  
print('Root Mean Squared Error:',np**.**sqrt(mean\_squared\_error(y\_test,predrd1)))

Text, letter

Description automatically generated

print(r2\_score(y\_test,predrd1))



r2 score of ridge remains the same after using alpha=1 also.

As from the above models, we can see that r2 score of lasso model is best. i.e 87. So, we will use this model for prediction.

**Concluding Remarks:**

We have started our baseball case study project by importing several libraries and imported the dataset from GitHub. Studying the various important details like what is the problem type and observing the datatypes of all the columns i.e. how many columns having integer, float and object type values.

As per statistic observations we found that all our columns are of numeric type. We have also noticed that there were outliers and skewness present in the data. We have applied standard scaler technique to scale the variables. We have also noticed that there were no Null Value (NaN) present in our data. During this procedure we have used matplotlib as well as seaborn for various visualizations and heatmaps. Subsequently we have started training of distinct machine learning models. We have utilized cross validation on all the regression models and then try to tune the model by using Hyperparameter Tuning.

We can also use other ML models such as DecisionTree Regressor, RandomForest Regressor, Support vector machine and Extreme Gradient boosting to predict the result.