**Baseball Case Study**

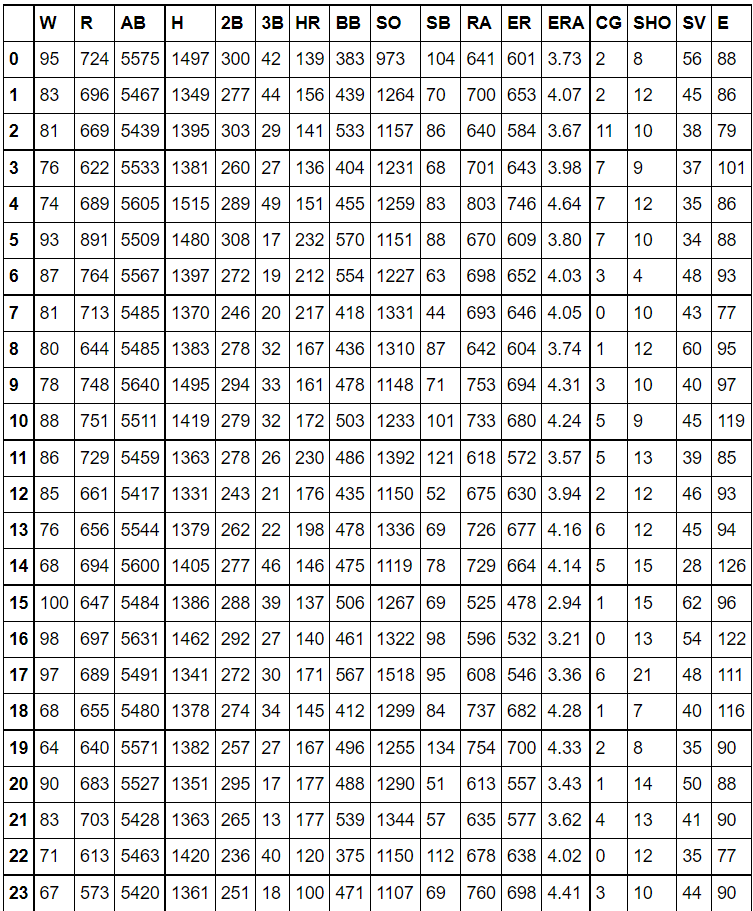
1. Problem Definition

The goal of this project is to develop a model that can accurately predict the number of wins for a given team in 2015 season. This dataset utilizes data from 2014 Major League Baseball seasons. Model has to predict the number of wins for a given team based on several different indicators of success. There are 16 different attributes in the given dataset that will be used to determine number of wins for a given team. In this project, you’ll test out several machine learning models from sklearn to predict the number of games that a Major-League Baseball team will win, based on the team’s statistics and other variables from that season.

2. Data Analysis

2.1 Data Import:

* The dataset is in csv format, we shall import the dataset using ‘read\_csv’ function.
* Once the dataset is imported and converted into a data frame, store the data frame and print it, to analyse the datapoints in rows and columns.



* Our project’s main objective is to predict number of wins (W) for a given team in the 2015 season based on data from 2014 Major League Baseball seasons with the help of several different attributes.
* As per the official records, description of each column in the Data frame is as follows:

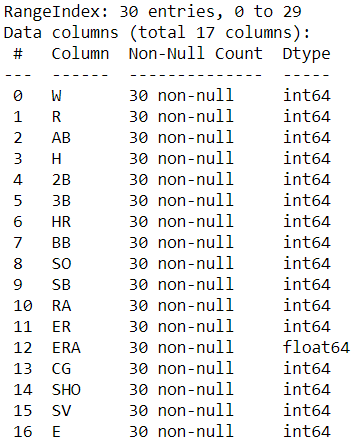
Output/Target Variable:

* + W – Win: number of games where pitcher was pitching while their team took the lead and went on to win, also the starter needs to pitch at least 5 innings of work.

Input/Independent Variable:

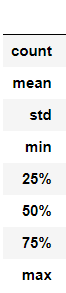
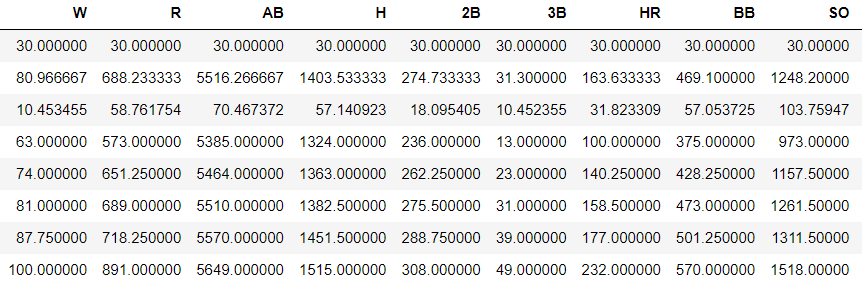
* + R – Runs: Runs scored times reached home plate legally and safely.
  + AB – At bat: plate appearances, not including bases on balls, being hit by pitch, sacrifices, interference, or obstruction.
  + H – Hits allowed: total hits allowed.
  + 2B – Double: hits on which the batter reaches second base safely without the contribution of a fielding error.
  + 3B – Triple: hits on which the batter reaches third base safely without the contribution of a fielding error.
  + HR – At bats per home run: at bats divided by home runs.
  + BB – Base on balls (also called a "walk"): hitter not swinging at four pitches called out of the strike zone and awarded first base.
  + SO – Plate appearances per strikeout: number of times a batter strikes out to their plate appearance.
  + SB – Stolen base: number of bases advanced by the runner while the ball is in the possession of the defence.
  + RA – Run average: number of runs allowed times nine divided by innings pitched.
  + ER – Earned run: number of runs that did not occur as a result of errors or passed balls.
  + ERA – Earned run average: total number of earned runs (see "ER" above), multiplied by 9, divided by innings pitched.
  + CG – Complete game: number of games where player was the only pitcher for their team
  + SHO – Shutout: number of complete games pitched with no runs allowed.
  + SV – Save: number of games where the pitcher enters a game led by the pitcher's team.
  + 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.

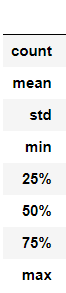
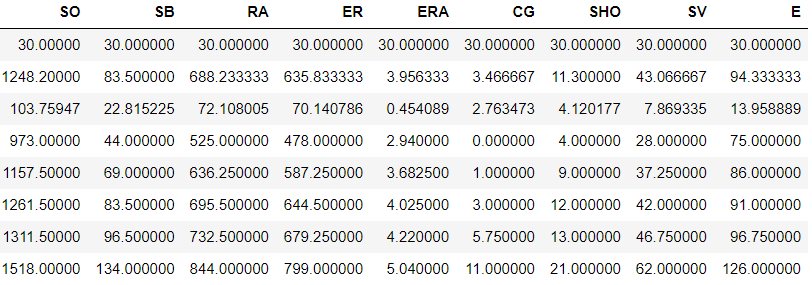
2.2 Data Analysis



* After executing the ‘info’ command we obtain the above output and following is our observation:
* Number of rows in dataset are: 30
* Number of columns in dataset are:17
* Dataset contains any null values: False
* All columns are of type Numeric.
* Only column ERA is of Float datatype
* Rest all columns are of Integer datatype.
* For further analyses of the dataset, we require Statistical Summary of each column such as mean

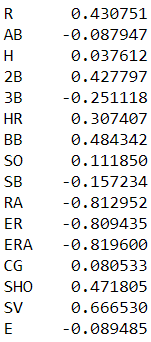
value, median value, max value, min value, standard deviation value.

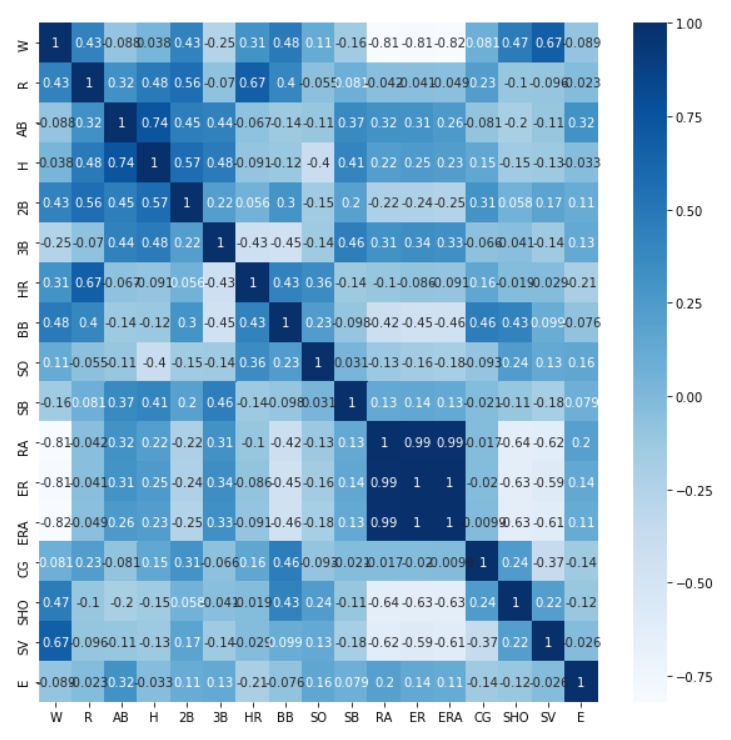
 

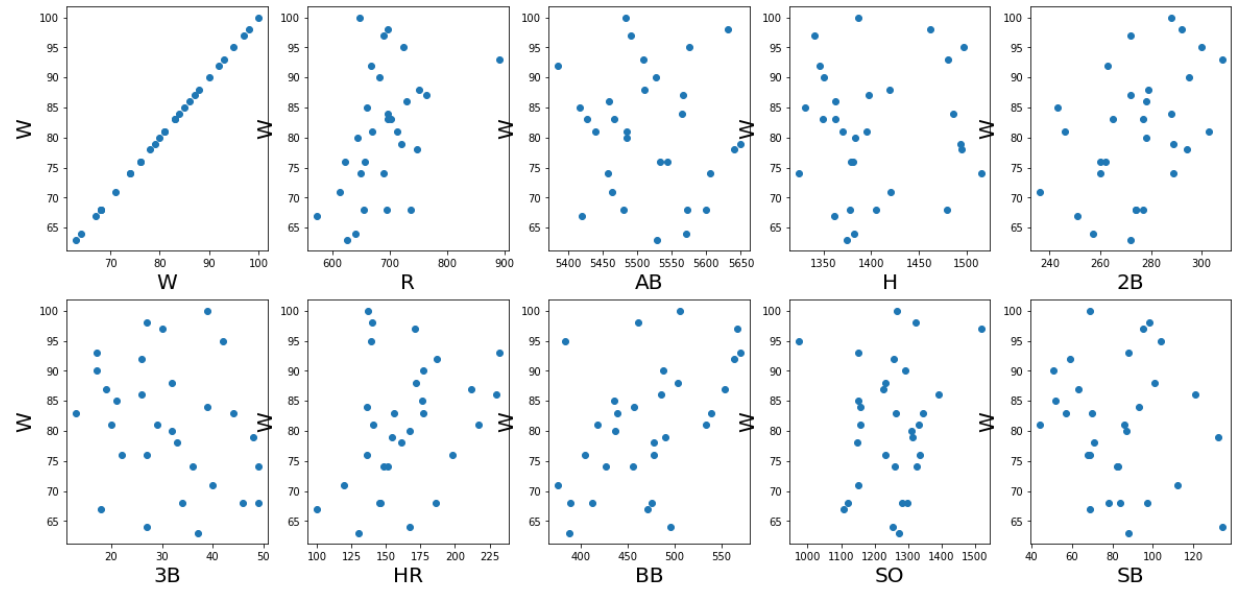
* Count value represents total number of non-null rows present in Dataset.
* Describe function provides us lower, 50 and upper percentiles.
* In statistics, a percentile is a score *at or below which* a given percentage falls.
* For example, the 50th percentile is the score below which 50% of the scores in the distribution may

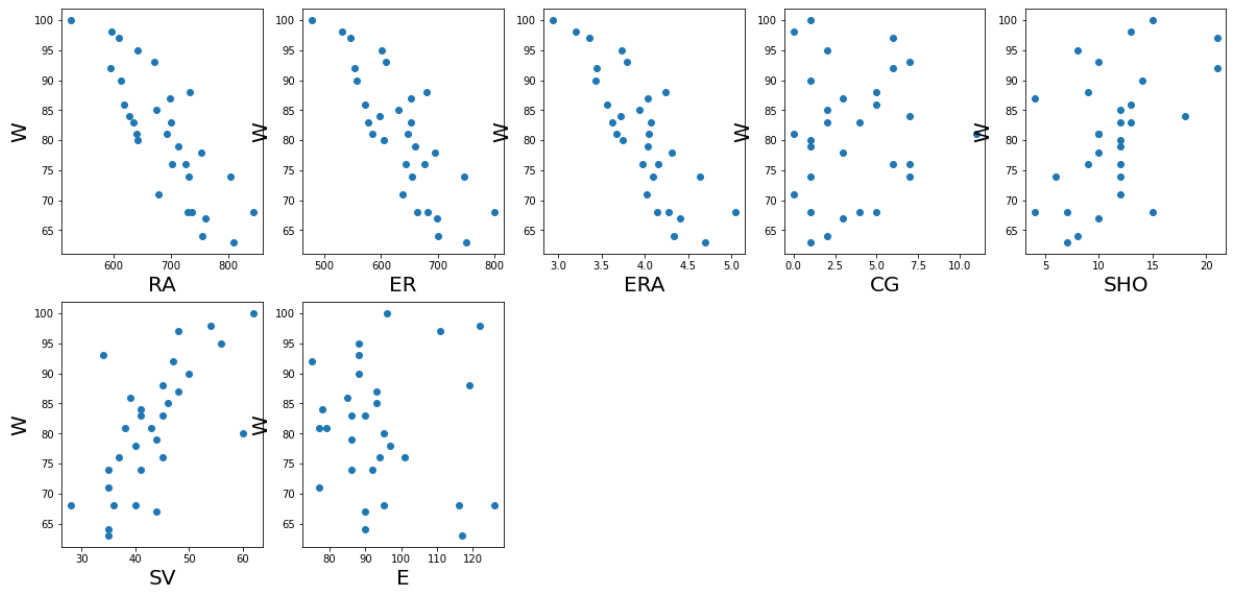
be found.

* Next Step is to check and analyse the correlation of input features w.r.t output column. 
* We can deduce which columns are highly correlated to the target column and which columns are least correlated. We perform these analyses to determine which columns can dropped based on their correlation w.r.t target variable.
* After analysis, we can conclude that column E & column AB are the least correlated and they would be of least significance while training the model.
* We need to further analyse our data to confirm our theory.
* Visualising correlation of the entire dataset using Heatmap.



* Check the graphical representation of input feature column w.r.t target variable to analyse their relation.





3. Exploratory data analysis

### Columns such as SV, SHO, BB, HR, R, 2B have positive correlation (Positive correlation is when the scatter plot takes a generally upward trend).

### The datapoints of columns SO, SB, AB, H, E, CG, 3B are just scattered w.r.t target variable.

### Remaining have negative correlation. (Negative correlation is when the scatter plot takes a generally downward trend).

### According to the graphs many columns have very low correlation with Number of predicted wins.

### With the help of correlation analyses, heatmap and graphical analysis we can conclude that SO, SB, AB, H, E, CG, 3B columns have least correlation & they can be dropped for the purpose of training the model.

### Check distribution plot of each column to determine whether a column has datapoints representing normal distribution or bimodal distribution or rectangular distribution plot.

### 

### 

### By observing the above plots, we obtain the following findings:

### All columns have normal distribution plots.

### There might be some outliers present in R, SHO, SV & ERA columns.

### As we already know there is a possibility of outliers existing in the dataset, we should cross check for outliers using “boxplot” command

### Boxplot for column W Boxplot for column R

### 

### Boxplot for column 2B Boxplot for column HR

### 

### Boxplot for column BB Boxplot for column RA

### 

### Boxplot for column ER Boxplot for column ERA

### 

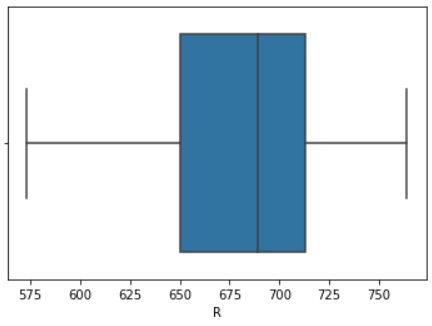
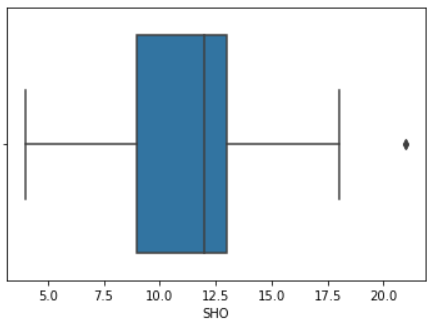
### Boxplot for column SHO Boxplot for column SV

### 

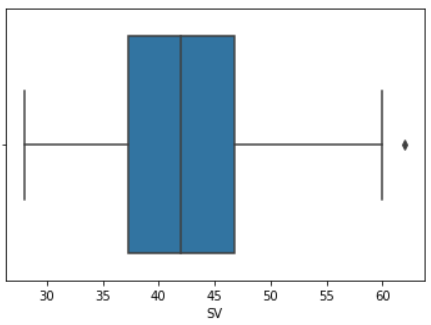
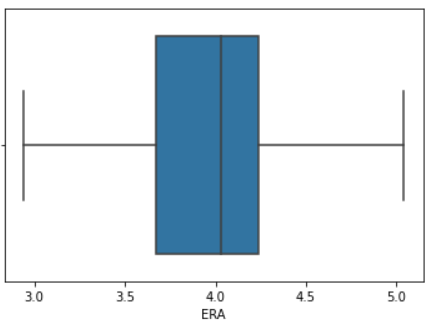
### We can confirm our thesis about outliers being present in R, SHO, SV & ERA columns.

* We need to remove those outliers, otherwise it will hamper our model’s performance.
* There are 2 ways to remove outliers:
  + Z-score method
  + IQR method
* Try the Z-score method and recheck the columns for any outliers’ present.

Boxplot for column R Boxplot for column SHO

Boxplot for column SV Boxplot for column ERA

* From above boxplots we can observe that outliers of columns R & ERA are removed, while outliers are still present in columns SV & SHO.
* Check shape of dataset before & after removal of outliers by comparing their shape.
  + Dataset size before z score: 30 Rows
  + Dataset size after z score: 29 rows
* Amount of data loss occurred after removing outliers Z-Score is 3.34% which is less than 10%.
* Upon further iteration of z-score no more outliers were removed and using IQR method leads to data loss of more than 10%

EDA Concluding Remark

* SO, SB, AB, H, E, CG, 3B columns have least correlation & they can be dropped for the purpose of training the model.

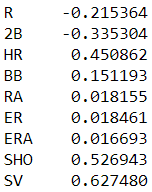
### All columns have normal distribution plots.

### The presence of some outliers is visible in R, SHO, SV & ERA columns.

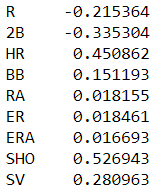
### Using Z-Score we were able to remove some outliers, while suffering data loss of 3.34%.

4. Pre-Processing Pipeline

* To train the model we need to split target variable and input features.
* To avoid biasing issue, we need to check for biasing using skew method of input features and remove if required.



* Skewness is present if skew value of a column is more than 0.55.
* From above table we can see biasing exists in column SV.
* Remove biasing of specific column by replacing its data points by cube root of its data points.
* Check for biasing again.



* All biasing has been removed, now there won’t be any biasing issue while training the model.
* Perform feature scaling using standard scaler algorithm.
* The objective of performing feature Scaling is to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing.
* Multicollinearity occurs when independent variables in a regression model are correlated. This correlation is a problem because independent variables should be independent. If the degree of correlation between variables is high enough, it can cause problems when you fit the model and interpret the results. Therefore, after performing feature scaling check for multi collinearity among columns.



* Column VIF shows the multi collinearity of each feature column.

### Multicollinearity only affects the coefficients and p-values, but it does not influence the model’s ability to predict the dependent variable.

### Levels of multicollinearity & what does it signify.

### VIF ~ 1: Negligible

### 1<VIF<5: Moderate

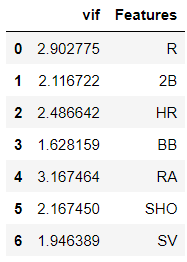
### VIF>5: Extreme

### Since ERA is derived from ER we can drop ERA as they are almost same

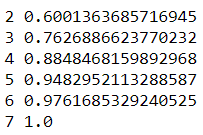
### Runs Allowed is the amount of runs that score against a pitcher. This includes earned runs and unearned runs.

### Meaning runs allowed RA is derived using earned runs ER. drop ER

* We drop the columns highest multicollinearity until it all columns have multicollinearity value that doesn’t exceed 5.



* Upon iterating, we solved the multicollinearity issue.
* The **Principal Component Analysis** is a popular unsupervised learning technique for reducing the dimensionality of data.
* It increases interpretability yet, at the same time, it minimizes information loss.
* It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D.
* PCA helps in finding a sequence of linear combinations of variables.
* Perform PCA to reduce dimensionality to optimize model performance.



* After running PCA we can observe the number of dimensions along with the info loss occurring.
* We can observe that when data dimension is reduced to 5 dimensions info loss of only 5% occurs and data retention of approximate 95% is obtained. Hence fit the data to 5 dimensions using PCA.

5. Building Machine Learning Models

### Now we will start with model selection and fine-tuning process.

### First, we need to find the most optimum model.

### We shall evaluate model on r2 score & cv score.

### The types of models through which we need to iterate are:

### Gradient Boosting Regressor

### NuSVR

### Linear Regression

### Ridge

### RidgeCV

### Bayesian Ridge

### SGD Regressor

### SVR

### AdaBoost Regressor

### Linear SVR

### KNeighbors Regressor

### RandomForest Regressor

### Bagging Regressor

### Decision Tree Regressor

### LGBM Regressor

### XGBRF Regressor

### XGB Regressor

### After iterating through all the above algorithms, we obtained the following r2 accuracy, cv score, mean squared error, mean absolute error and the difference between r2 score and cv score.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | CV score | R2 score | difference | MSE | MAE |
| Gradient-Boosting Regressor | 30.66% | 72.741% | 42.076% | 4.77 | 5.12 |
| NuSVR | -59.17% | 18.54% | 77.71% | 7.12 | 8.85 |
| Linear Regression | 62.89% | 83% | 18% | 3.23 | 4.04 |
| Ridge | 65% | 83% | 18% | 3.23 | 4.04 |
| RidgeCV | 62.11% | 83% | 20.86% | 3.23 | 4.04 |
| Bayesian Ridge | 65.50% | 82.84% | 17.34% | 3.30 | 4.06 |
| SGD Regressor | 63.65% | 82.25% | 18.6% | 3.34 | 4.14 |
| SVR | -45.86% | 25.15% | 71% | 7.37 | 8.5 |
| AdaBoost Regressor | 50.9% | 67.45% | 16.54% | 5.30 | 5.60 |
| Linear SVR | -68.13% | -35.45% | 35.68% | 58.61 | 59.26 |
| KNeighbors Regressor | 34.56% | 83.62% | 49.06% | 3.4 | 3.97 |
| RandomForest Regressor | 46% | 62.75% | 16.76% | 5.66 | 5.98 |
| Bagging Regressor | 46.13% | 54.25% | 8.12% | 6.434 | 6.639 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Decision TreeRegressor | -8.76% | 77.34% | 86.09% | 4.16 | 4.67 |
| LGBM Regressor | -68.75% | 00.50% | 68.25% | 8.56 | 9.83 |
| XGBRF Regressor | 50% | 68% | 18.88% | 5.38 | 5.47 |
| XGB Regressor | 38% | 72.4% | 34.16% | 4.77 | 5.15 |

### After analysing above model accuracies, we can conclude that:

### Independent model does not have high accuracy.

### CV score is very low.

### Difference between cv score and r2 score is too high.

### Top performing models are:

### Bayesian Ridge

### XGBRF Regressor

### RandomForest Regressor

### Perform fine tuning of all these models and find the best parameters to be used for the model by using GridsearchCV algorithm.

### Best parameters for Random Forest Regressor are:

* + 'criterion': 'mse'
  + 'max\_features': 'sqrt'
  + 'n\_estimators': 10

### Best parameters for Bayesian Ridge are:

* + 'compute\_score': True
  + 'fit\_intercept': True
  + 'n\_iter': 1

### Best parameters for XGBRF Regressor are:

* + 'max\_depth': 7
  + 'n\_estimators': 45
  + 'reg\_lambda': 0.25

### Now we need to find the most optimum random state in train test split for Bayesian Ridge model to get best score, in this case best random state is 141

### As we already concluded that an individual model is unable to perform, hence stack all the models, to boost their performance using StackCV Regressor.

### Stack all the best models to improve accuracy.

### 

### By running the above code, we are going to split the dataset by keeping 20% for testing and 80% for training

### Using StackCV Regressor we combine the fine-tuned models to boost the total performance.

### Lastly, we need to check if overfitting exists by obtaining the prediction score of train & test dataset.

### Obtained train & test score is as follows:

### r2 score test 0.9013313377866263

### r2 score train 0.8749523801687697

### From the above we can observe that there is not much difference between train and test accuracy confirming that model has no underfitting/overfitting issue and gives us the best accuracy

### Now, obtain all the metrics of the currently trained model:

* Co-efficient of determination is: 0.9013313377866263
* mean absolute error: 3.536776246442384
* mean squared error: 13.484717169161064
* root mean square error: 3.6721542953913393
* cross validation score 0.2913070147795927
* std err 0.9886496307325187

### Now test the model on whole dataset by putting target variable ‘W’ column & predicted “W” column side by side to observe the difference between the actual and predicted datapoints.

### Round the values of column ‘Wpred’ to integer.

### 

### We also added a column “difference” which is obtained by subtracting actual data points in W with predicted data points in Wpred.

### Observing above data frame we can conclude that there is only slight difference between actual and predicted values.

### 

### A Regression plot is used to check how close the actual and predicted values are by checking the relation.

### As we can see the predicted and actual value have a strong linear relation, meaning the error datapoints lie very close the linear line.

### Plot appropriate graph to check how close actual and predicted values are.

### Red star indicates the actual value, while blue dots represent predicted value.

### As we can observe there is very less difference between the actual and predicted plots, which indicates the high accuracy of our model.

### Further analysing distribution plot of actual and predicted values.

### 

### From above plot we can observe the difference in distribution between actual and predicted values.

### Finally, we save the model using “joblib” library which can be reused for further prediction.

6. Concluding remarks

We were able to build a model having 90% accuracy in predicting the number of wins for a given team for the season of 2015. An individual model was not sufficient enough to obtain high accuracy, hence we built a model having a combination of Bayesian Ridge, XGBRF regressor & Random forest. We also observed the difference between actual and predicted value. By further visualizing of datapoints we can conclude that the model is accurately able to predict number of wins of a team for 2015 season. The saved model can be loaded and used again to predict number of wins. Model’s accuracy can increase if more training data is provided.

### Click the link below to go through the jupyter notebook:

### [Baseball](https://github.com/genos1998/datatrained-project/blob/main/Baseball.ipynb)