# **Predicting the outcome of Major League Baseball**



Major League Baseball (MLB) is the highest level of professional baseball in the world and accounts for some of the most popular international sporting events. Many scholars have conducted research on predicting the outcome of MLB matches. The accuracy in predicting the results of baseball games is low. Therefore, deep learning and machine learning methods were used to build models for predicting the outcomes (win/loss) of MLB matches and investigate the differences between the models in terms of their performance.

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:

R-Run

AB=At Bats

H=Hits

2B= Doubles

3B=Triples

HR=Homeruns

BB= Base on balls/Walk

SO=Strikeouts

SB= Stolen Bases

RA= Runs Allowed

ER= Earned Runs

Earned Run Average (ERA)

SHO=Shutouts

SV=Saves

CG=Complete Games

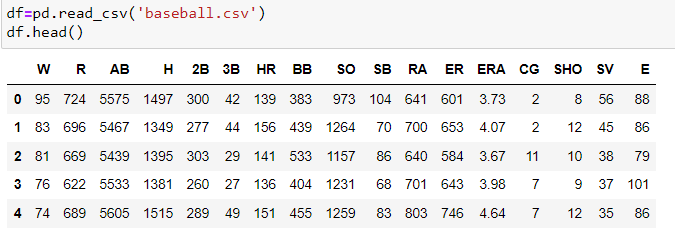
E=Errors

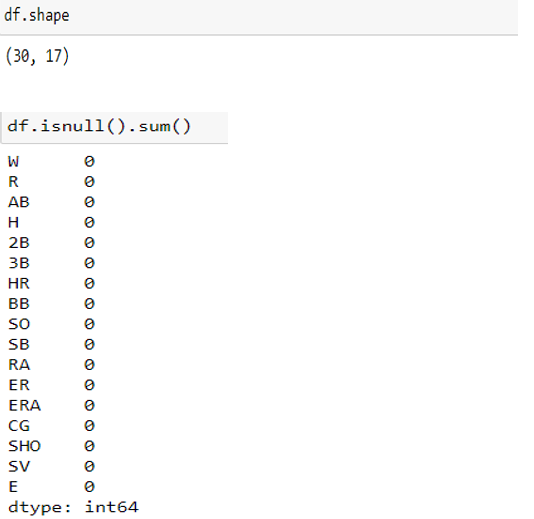
Output Feature:

Number of predicted wins (W)

# **Data Preprocessing**

We use pandas to read the dataset and pre-process it. There aren’t any rows with all missing values or duplicates (this rarely happens with real-world datasets). There are totally 30 rows and 17columns in this dataset.

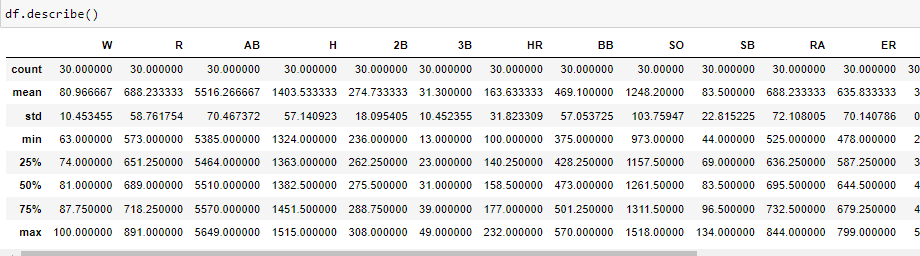




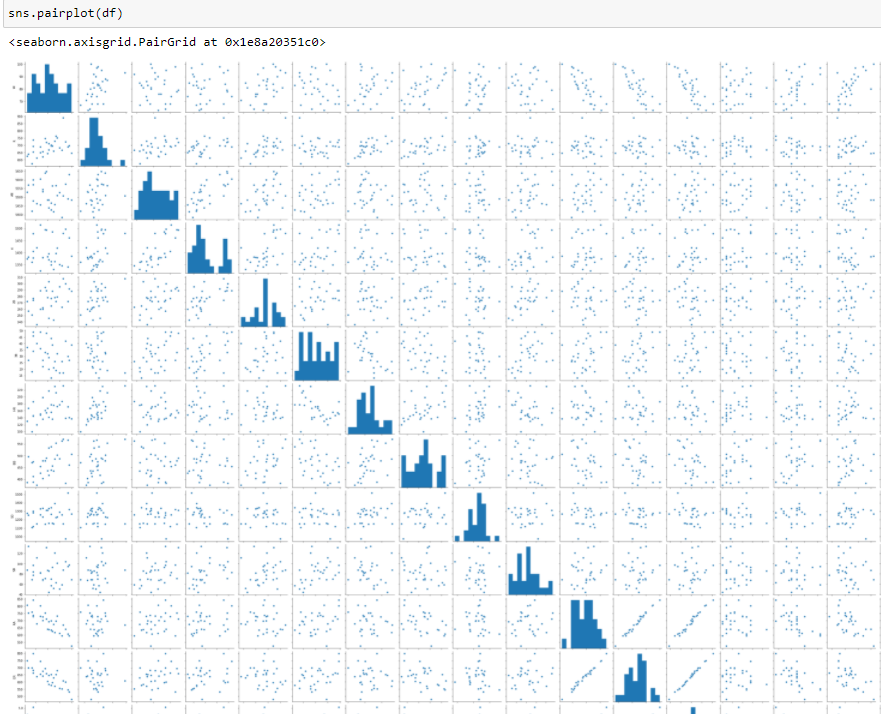
## **Exploratory Data Analysis**

## Features distribution:

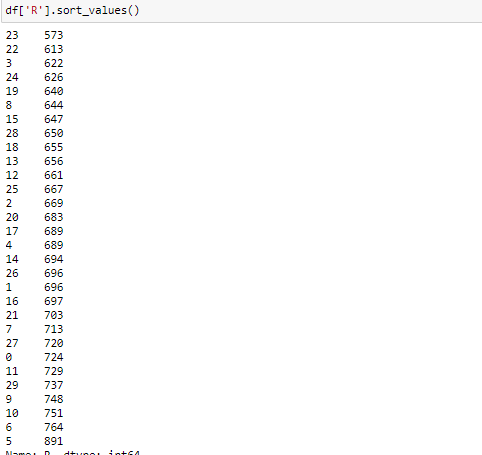
## Standard deviations are important here because the shape of a normal curve is determined by its mean and standard deviation. The mean tells you where the middle, highest part of the curve should go.



At first glance, there aren’t any outliers in the data. No data point is disconnected from distribution or too far from the mean value. To confirm that we would need to plot the data.



We can see that there are few outliers present in the data. One particular value acts an outlier for most of the values. So let’s drop that row.



We can see that row 5 acts as an outlier. Let’s drop row 5



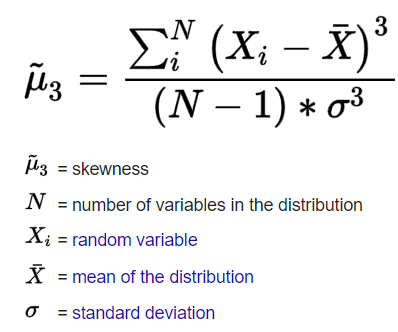
* Correlation

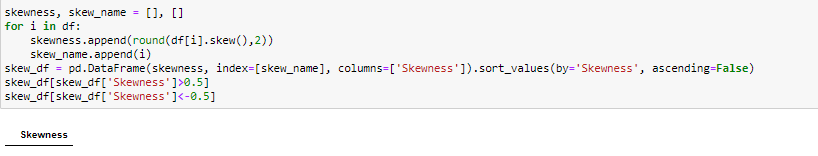
Correlation coefficients are used to measure the strength of the relationship between two variables. This measures the strength and direction of a linear relationship between two variables. Values always range between -1 (strong negative relationship) and +1 (strong positive relationship).



1. We observe that Save (SV) is highly correlated with win (W).
2. ER (Earned Runs), ERA (Earned Run Average) and RA (Runs Allowed) are negatively correlated with W.
3. We can see that RA, ERA, ER are highly correlated with each other, So we shall drop RA and ERA.
4. AB (At Bats) is highly correlated with H (Hits).

Let’s check the skewness of the dataset. Skewness is a measure of the symmetry of a distribution. The highest point of a distribution is its mode. Negative values for the skewness indicate data that are skewed left and positive values for the skewness indicate data that are skewed right.



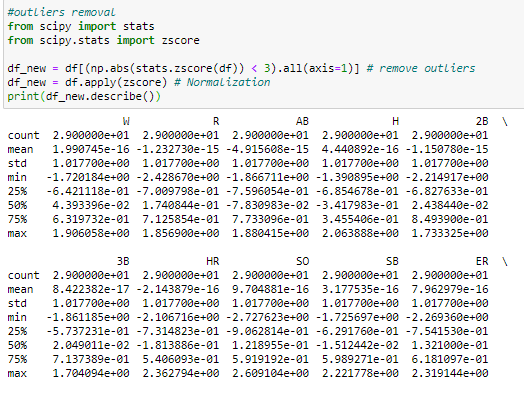


If skewness is between -1 and -0.5 or between 0.5 and 1, the distribution is moderately skewed. If skewness is between -0.5 and 0.5, the distribution is approximately symmetric. We can see that skewness of values in our dataset is not less than -0.5 and greater than 0.5. So distribution of our data is approximately symmetric.

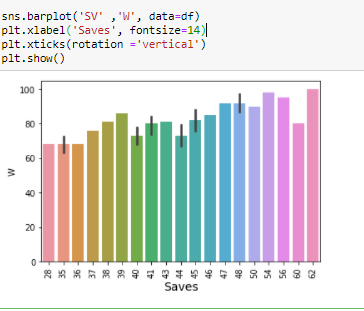
We shall remove further outliers of the data using zscore.

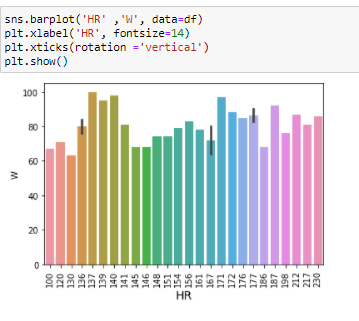
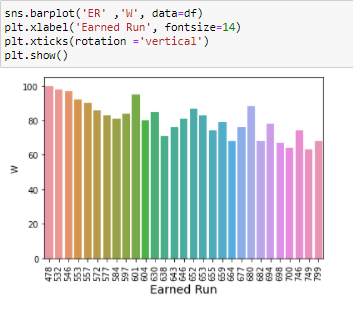
**Simply put, a z-score (also called a**standard score**) gives you an idea of how far from the**[mean](https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-median-mode/)**a data point is.**But more technically it’s a measure of how many [standard deviations](https://www.statisticshowto.com/probability-and-statistics/standard-deviation/) below or above the [population mean](https://www.statisticshowto.com/population-mean/)a [raw score](https://www.statisticshowto.com/raw-score/) is.

A z-score can be placed on a [**normal distribution**](https://www.statisticshowto.com/probability-and-statistics/normal-distributions/) curve. Z-scores range from -3 standard deviations (which would fall to the far left of the normal distribution curve) up to +3 standard deviations (which would fall to the far right of the normal distribution curve).



Let’s check the relationship between few variables with target variable W (Win).



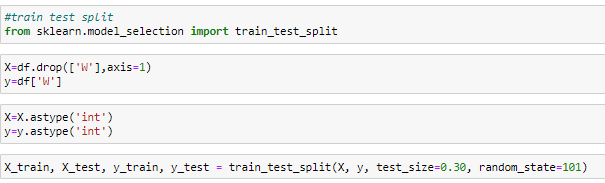


From the above plots we can observe that

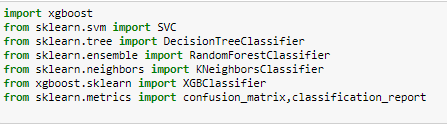
1. Winning chances are high when the saves are above 47.
2. As the runs earned increases the chances of winning decreases slightly.
3. There is a slight linearity between Homeruns and Win.

## Training the model

We split the dataset to train (70% samples) and test (30% samples).  
We train (fit) the data and make predictions. With classification\_report we calculate precision and recall with actual and predicted values.



Let’s import machine learning models and few libraries.



Now we predict the output variable using ML models. We are using below models for predicting the output.

1. SVC:

SVC is a nonparametric clustering algorithm that does not make any assumption on the number or shape of the clusters in the data. In our experience it works best for low-dimensional data, so if your data is high-dimensional, a Preprocessing step, e.g. using principal component analysis, is usually required.

1. KNeighbours Classifier:

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K-NN algorithm.

1. Decision Tree

The decision tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for regression problem.The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

1. Random Forest

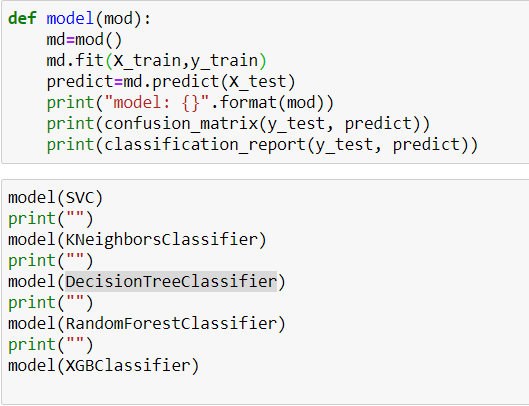
Random forests (RF) are basically a bag containing n Decision Trees (DT) having a different set of hyper-parameters and trained on different subsets of data. Let’s say I have 100 decision trees in my Random Forest .These decision trees have a different set of hyper-parameters and a different subset of training data, so the decision or the prediction given by these trees can vary a lot. Let’s consider that I have somehow trained all these 100 trees with their respective subset of data. Now I will ask all the hundred trees in my bag that what is their prediction on my test data. Now we need to take only one decision on one example or one test data, we do it by taking a simple vote. We go with what the majority of the trees have predicted for that example

1. XGBoost

It is also known as “Extreme Gradient Boosting”. It carries out the gradient boosting decision tree algorithm. It has several different names like gradient boosting, gradient boosting machine, etc.

Boosting is nothing but ensemble techniques where previous model errors are resolved in the new models. These models are added straight until no other improvement is seen. One of the best examples of

such an algorithm is the AdaBoost algorithm. Gradient boosting is a method where the new models are created that computes the error in the previous model and then leftover is added to make the final prediction



Above code shows that we have defined a function and models as its argument. We will fit the model to training dataset and later predict the outcome of test dataset. We then print confusion matrix and classification report for better understanding of predicted outcome.

Confusion Matrix:

A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.

Classification Report:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False? More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives.

* Precision:

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

**TP – True Positives**

**FP – False Positives**

**Precision – Accuracy of positive predictions.**

**Precision = TP/(TP + FP)**

* Recall:

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

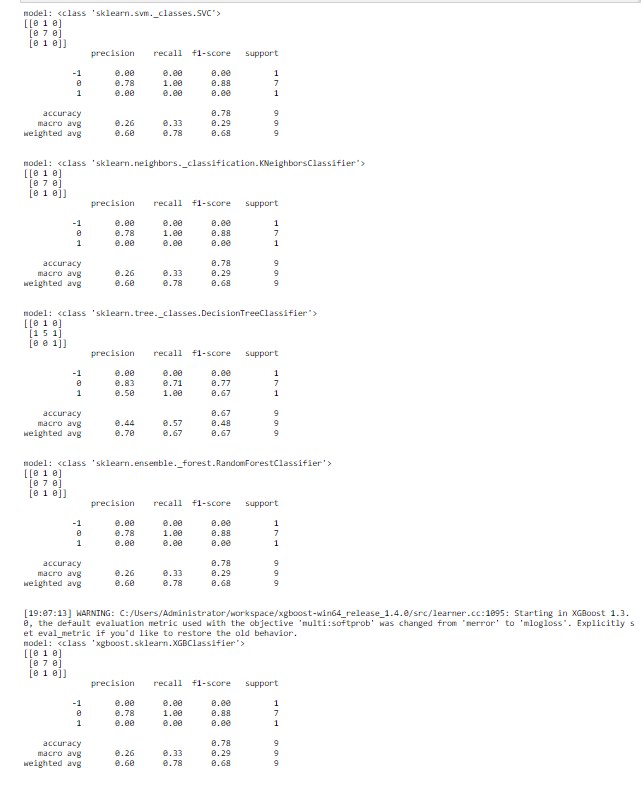
**FN – False Negatives**

**Recall = TP/(TP+FN)**

* F1 score:

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

**F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)**



We are using five Machine Learning models. Among which all the models expect decision tree predicts 78% accuracy and decision tree predicts 67%. Let’s check the same after hyper parameter tuning. Here we use GridsearchCV method for hyper parameter tuning.

GridSearchCV:

 It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

GridSearchCV is a function that comes in Scikit-learn’s (or SK-learn) model selection package. So an important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

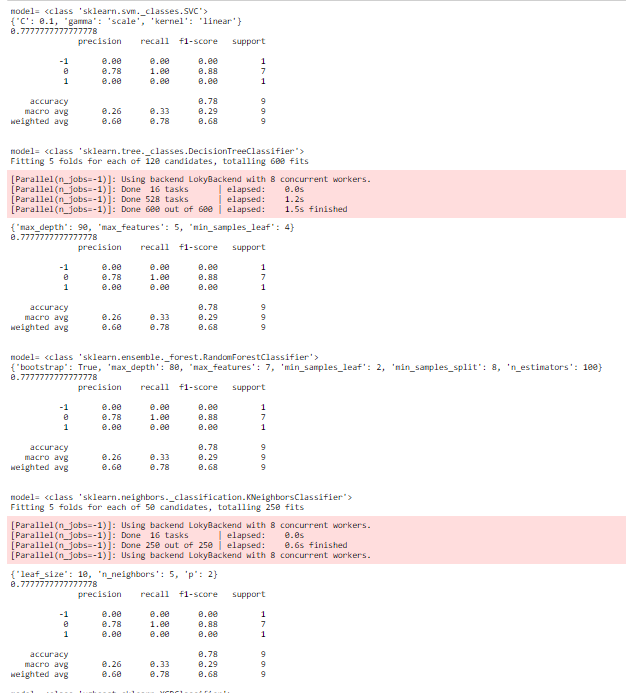
We pass predefined values for hyperparameters to the GridSearchCV function. We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take. As shown below in the code.

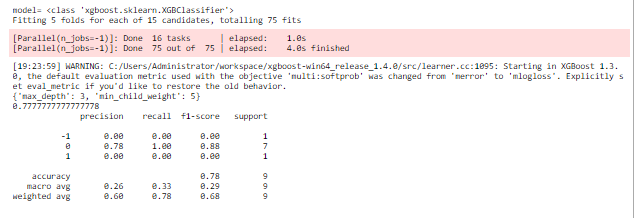
Here C, gamma and kernels are some of the hyperparameters of an SVM model. Note that the rest of the hyperparameters will be set to their default values

GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.



We can observe that all the models after parameter tuning gives an accuracy score of 78%.





Model accuracy is defined as the number of classifications a model correctly predicts divided by the total number of predictions made. It's a way of assessing the performance of a model.

## **Conclusion:**

We predicted the outcomes of MLB matches by collecting the match data of teams in the 2014 season and using Random Forest, KNN, Decision tree, SVM and Xgboost prediction models. The prediction accuracies of these models were then compared. We investigated whether the selection of input variables and the characteristics before and after feature selection had an impact on the prediction accuracy. The prediction results indicated that Random Forest, KNN, Decision tree, SVM and Xgboost models achieved higher prediction accuracies. The three prediction models proposed in this study all achieved high prediction performance and can thus be used to provide some reference information for fans, team managers, and baseball enthusiasts.