**Introduction:**

We decided to do a project centered around the game of baseball. Our goal was to predict the rounds in which each MLB playoff team lost, from 1995-2016.

Here is a quick overview of the game of baseball:

Baseball is a **“bat-and-ball”** game played between two teams of nine payers each, who take turns **batting** and **fielding**. The batting team attempts to score runs by hitting a ball that is thrown by the **pitcher** with a bat swung by the **batter**, then running counter-clockwise around a series of 4 bases: first base, second base, third base, and home plate. A run is scored when a player advances around the bases and returns to home plate and touches the base. Players on the batting team take turns hitting against the pitcher of the fielding team, which tries to prevent runs by getting hitters “out” in any several ways. A player on the batting team who reaches a base safely can later attempt to advance to subsequent bases during teammates’ turns batting, such as a “hit” or by other means. The teams switch between batting and fielding whenever the fielding team records three outs. One turn batting for both teams, beginning with the vising team, constitutes one “inning”. A game is composed of nine innings, and the team with the greater number of runs at the end of the game wins. (In *Wikipedia*. Retrieved December 8th, 2016.)

In Major League Baseball, there are two leagues: 1) The National League and, 2) The American League. Each league has three divisions: 1) East, 2) Central, and 3) West. Every team plays a 162-game regular season. At the end of the regular season, the leaders of each division are guaranteed a playoff spot, totaling 6 playoff teams. The two teams with the next best records in each league play in a single-elimination “wild card” game, adding two more playoff teams. The respective playoff rounds that each league competes in individually are 1) the Division Series and, 2) the Championship Series. The National League Championship Series winner and the American League Championship Series winner will face off against each other in the **“World Series”.** The Division Series is out of 5 games, so the winner must win 3 games. The Championship Series and World Series are out of 7 games, so the winners must win 4 games in these rounds. Therefore, a team must win 11 playoff games to win the world series.

Baseball has been a pioneer in the use of analytics to predict performance in sports. Theo Epstein, the General Manager of the Chicago Cubs and former General Manager of the Boston Red Sox has been using analytics to operate his teams for the past few decades, and it has paid off in a huge way, winning the World Series with both teams. He is now one of the most legendary front-office managers in all of sports. We decided to follow his path and create a World Series Prediction Model.

We tested multiple predictive modeling techniques such as linear regression, logistic regression, and KNN – Nearest Neighbor to predict the number of playoff games won by each playoff team. The team that was closest to 11 playoff wins was deemed the “World Series Winner”. The team with the 2nd highest number of playoff wins was deemed the “Championship Series Winner”, and the team with the third highest number of playoff wins was deemed the “Division Series Winner.” We decided to use data from the past 20 years, including 1995, because the “Wild Card” playoff system began in 1995 which changed the number of playoff teams from six to eight. The modeling technique with the best accuracy score was KNN – Nearest Neighbor. Below, you can see a distribution of the playoff teams of the past 20 years, and the playoff rounds in which lost/won.

# 

# Data

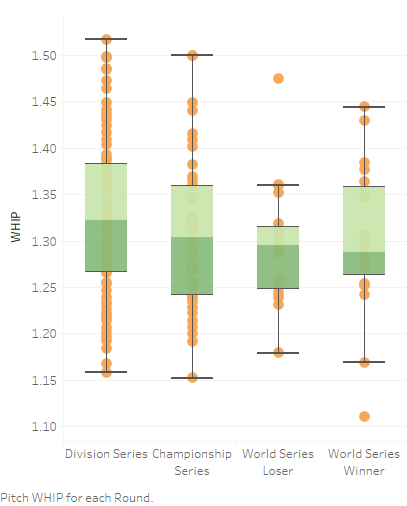
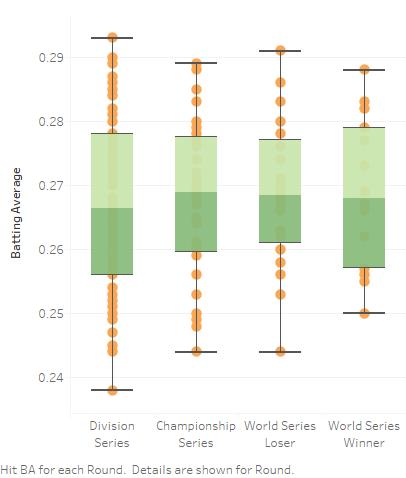
Data for our project consisted of team statistics for each of the eight playoff teams from the 1995 – 2016 major league baseball seasons. Due to a significant rule change after the 1994 season we decided to include seasons after 1995 for consistency. This resulted in 176 rows of data and 63 columns. Two additional columns were manually created for our dataset: 1) a binary column of 1’s and 0’s indicating if a team won a world series in a particular season 2) a categorical column representing how far a team progressed in the playoffs.

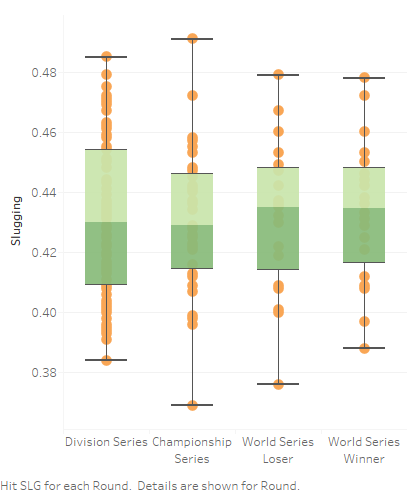
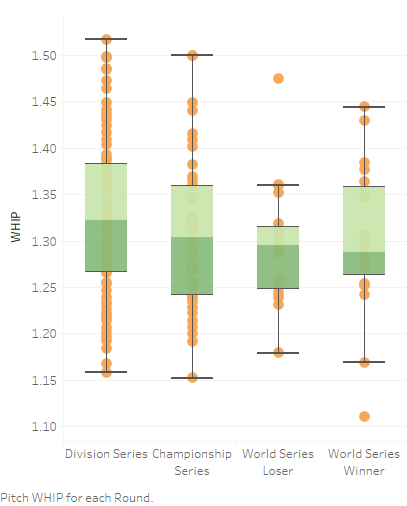
All data were gathered from Baseball-reference.com; a free website which hosts baseball statistics dating back to the 1800’s. Baseball-reference.com is designed for easy data export. All data are stored in downloadable CSV format. CSV files were downloaded for all teams from the 1995 – 2016 seasons and filtered to only include teams which made the playoffs.

Baseball statistics are largely split into two main categories: pitching and hitting. Pitching can be thought of as more “defensive” statistics and hitting equates to “offensive” statistics. Several statistics which we intuitively believed contributed to playoff success were part of our initial data exploration. We created boxplots using Tableau for on base percentage (OBP), batting average (BA), slugging (SLG), and walks + hits by innings pitched (WHIP). Definitions for each statistic are below.

* On base percentage: the number of times that a team reached based divided by the total number of at bats
* Batting Average: the total number of hits for a team divided by their number of at bats
* Slugging: the percent of hits which results in a double, triple, or home run
* Walks + Hits divided by Innings Pitched: the number of walks plus hits divided by the total number of innings pitched

Not surprisingly, the boxplots revealed that hitting and pitching statistics generally improved as teams progressed further in the playoffs.

Once the dataset was loaded into Python, we continued our analysis and created algorithms using packages like Seaborn, Matplotlib, Py4j, Pyspark, Atexit, and Train\_test\_split.

# Variable Selection

**Recursive Feature elimination**

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), recursive feature elimination ([**RFE**](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE)) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and weights are assigned to each one of them. Then, features whose absolute weights are the smallest are pruned from the current set features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

**ExtraTrees Classifier**

An extremely randomized tree classifier.Extra-trees differ from classic decision trees in the way they are built. When looking for the best split to separate the samples of a node into two groups, random splits are drawn for each of the max\_features randomly selected features and the best split among those is chosen. When max\_features is set 1, this amounts to building a totally random decision tree.Warning: Extra-trees should only be used within ensemble methods.

**Random Forest Regressor**

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

**Compare output of each method**

**RFE output –**

Features sorted by their rank:

[(1, 'hit\_OBP'), (2, 'hit\_OPS'), (3, 'hit\_SLG'), (4, 'hit\_BA'), (5, 'pitch\_WHIP'), (6, 'pitch\_BB9'), (7, 'pitch\_H9'), (8, 'pitch\_HR9'), (9, 'pitch\_FIP'), (10, 'pitch\_WL'), (11, 'hit\_RG'), (12, 'pitch\_ERA'), (13, 'pitch\_SOW'), (14, 'pitch\_SO9'), (15, 'pitch\_W'), (16, 'pitch\_GS'), (17, 'pitch\_G'), (18, 'hit\_G'), (19, 'pitch\_RAG'), (20, 'pitch\_PAge'), (21, 'pitch\_CG'), (22, 'pitch\_GF'), (23, 'hit\_BatAge'), (24, 'pitch\_ER'), (25, 'pitch\_H'), (26, 'pitch\_BB'), (27, 'pitch\_IP'), (28, 'pitch\_cSho'), (29, 'pitch\_BK'), (30, 'hit\_H'), (31, 'hit\_AB'), (32, 'hit\_SF'), (33, 'hit\_SH'), (34, 'hit\_PA'), (35, 'pitch\_L'), (36, 'pitch\_SO'), (37, 'hit\_LOB'), (38, 'pitch\_LOB'), (39, 'pitch\_BF'), (40, 'pitch\_R\_allowed'), (41, 'hit\_TB'), (42, 'hit\_RBI'), (43, 'pitch\_tSho'), (44, 'hit\_HR'), (45, 'pitch\_numP'), (46, 'hit\_BB'), (47, 'hit\_HBP'), (48, 'hit\_GDP'), (49, 'year'), (50, 'hit\_CS'), (51, 'pitch\_WP'), (52, 'pitch\_HBP'), (53, 'hit\_R\_scored'), (54, 'hit\_2B'), (55, 'pitch\_HR'), (56, 'pitch\_ERAplus'), (57, 'pitch\_IBB'), (58, 'hit\_3B'), (59, 'hit\_OPS+'), (60, 'hit\_SB'), (61, 'hit\_SO'), (62, 'pitch\_SV'), (63, 'hit\_numBat'), (64, 'hit\_IBB')]

**ExtraTrees Classifier output -**

[(0.027300000000000001, 'hit\_BatAge'), (0.026800000000000001, 'hit\_3B'), (0.0253, 'pitch\_H9'), (0.025100000000000001, 'pitch\_WP'), (0.024899999999999999, 'hit\_SH'), (0.0246, 'pitch\_W'), (0.024, 'hit\_BA'), (0.023900000000000001, 'hit\_LOB'), (0.022599999999999999, 'pitch\_PAge'), (0.022499999999999999, 'pitch\_HBP'), (0.022499999999999999, 'hit\_TB'), (0.0224, 'hit\_2B'), (0.020899999999999998, 'hit\_BB'), (0.0201, 'pitch\_SOW'), (0.019599999999999999, 'pitch\_tSho'), (0.0189, 'pitch\_WHIP'), (0.0184, 'hit\_GDP'), (0.018100000000000002, 'pitch\_CG'), (0.017899999999999999, 'pitch\_R\_allowed'), (0.017600000000000001, 'pitch\_IBB'), (0.017600000000000001, 'hit\_IBB'), (0.017500000000000002, 'pitch\_SV'), (0.0172, 'pitch\_BK'), (0.017000000000000001, 'hit\_OPS+'), (0.016799999999999999, 'pitch\_BB'), (0.016400000000000001, 'pitch\_numP'), (0.016400000000000001, 'pitch\_H'), (0.016299999999999999, 'hit\_HBP'), (0.0161, 'pitch\_ERA'), (0.0161, 'hit\_HR'), (0.015900000000000001, 'pitch\_ERAplus'), (0.015699999999999999, 'pitch\_LOB'), (0.015699999999999999, 'hit\_OPS'), (0.015599999999999999, 'pitch\_SO'), (0.015299999999999999, 'pitch\_BB9'), (0.014500000000000001, 'pitch\_HR9'), (0.014500000000000001, 'pitch\_ER'), (0.0144, 'hit\_CS'), (0.014, 'hit\_SO'), (0.013299999999999999, 'pitch\_RAG'), (0.0132, 'hit\_RBI'), (0.0126, 'hit\_AB'), (0.0123, 'hit\_SB'), (0.0121, 'hit\_numBat'), (0.012, 'pitch\_SO9'), (0.012, 'pitch\_BF'), (0.012, 'hit\_RG'), (0.011599999999999999, 'hit\_SLG'), (0.0115, 'year'), (0.0115, 'hit\_H'), (0.0112, 'hit\_OBP'), (0.010500000000000001, 'pitch\_IP'), (0.010500000000000001, 'hit\_G'), (0.010200000000000001, 'pitch\_GF'), (0.0097999999999999997, 'pitch\_FIP'), (0.0092999999999999992, 'pitch\_G'), (0.0088000000000000005, 'hit\_SF'), (0.0086, 'pitch\_cSho'), (0.0086, 'hit\_PA'), (0.0077999999999999996, 'pitch\_WL'), (0.0077999999999999996, 'hit\_R\_scored'), (0.0071999999999999998, 'pitch\_L'), (0.0068999999999999999, 'pitch\_HR'), (0.0025000000000000001, 'pitch\_GS')]

**Random Forest Regressor output –**

Features sorted by their score:

[(0.077200000000000005, 'pitch\_HBP'), (0.061400000000000003, 'pitch\_PAge'), (0.059400000000000001, 'hit\_BB'), (0.0511, 'hit\_3B'), (0.039399999999999998, 'pitch\_WP'), (0.039100000000000003, 'pitch\_IBB'), (0.039100000000000003, 'hit\_BatAge'), (0.037499999999999999, 'hit\_CS'), (0.033700000000000001, 'hit\_SO'), (0.030099999999999998, 'pitch\_ER'), (0.0293, 'pitch\_WHIP'), (0.027400000000000001, 'pitch\_SV'), (0.0218, 'hit\_SH'), (0.020199999999999999, 'pitch\_IP'), (0.020199999999999999, 'pitch\_ERAplus'), (0.02, 'hit\_R\_scored'), (0.019199999999999998, 'hit\_SB'), (0.0189, 'pitch\_H'), (0.018499999999999999, 'pitch\_HR'), (0.017899999999999999, 'pitch\_H9'), (0.0178, 'pitch\_W'), (0.017500000000000002, 'pitch\_FIP'), (0.0154, 'hit\_IBB'), (0.0147, 'hit\_HR'), (0.0143, 'hit\_SLG'), (0.012800000000000001, 'pitch\_BF'), (0.012200000000000001, 'pitch\_SO'), (0.0117, 'hit\_H'), (0.0114, 'hit\_OPS+'), (0.0111, 'pitch\_WL'), (0.010699999999999999, 'pitch\_BK'), (0.0104, 'pitch\_SO9'), (0.010200000000000001, 'hit\_RBI'), (0.0101, 'pitch\_LOB'), (0.0092999999999999992, 'pitch\_tSho'), (0.0091000000000000004, 'pitch\_BB'), (0.0088000000000000005, 'year'), (0.0088000000000000005, 'pitch\_G'), (0.0086999999999999994, 'hit\_GDP'), (0.0083000000000000001, 'hit\_TB'), (0.0083000000000000001, 'hit\_SF'), (0.0082000000000000007, 'hit\_LOB'), (0.0077000000000000002, 'hit\_PA'), (0.0068999999999999999, 'pitch\_ERA'), (0.0064000000000000003, 'hit\_G'), (0.0061999999999999998, 'pitch\_numP'), (0.0061999999999999998, 'hit\_HBP'), (0.0047999999999999996, 'hit\_numBat'), (0.0041999999999999997, 'hit\_AB'), (0.0041000000000000003, 'hit\_OBP'), (0.0040000000000000001, 'hit\_2B'), (0.0032000000000000002, 'pitch\_SOW'), (0.0030000000000000001, 'pitch\_BB9'), (0.0022000000000000001, 'pitch\_R\_allowed'), (0.0018, 'pitch\_HR9'), (0.0018, 'hit\_BA'), (0.0012999999999999999, 'pitch\_L'), (0.0011999999999999999, 'hit\_OPS'), (0.0011000000000000001, 'pitch\_CG'), (0.001, 'pitch\_RAG'), (0.00080000000000000004, 'hit\_RG'), (0.00040000000000000002, 'pitch\_cSho'), (0.00040000000000000002, 'pitch\_GF'), (0.0, 'pitch\_GS')]

**Algorithm**

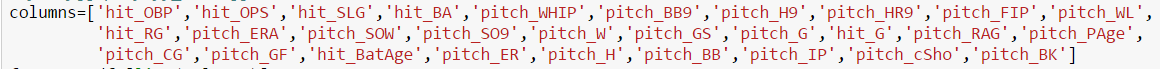
1). Logistic Regression

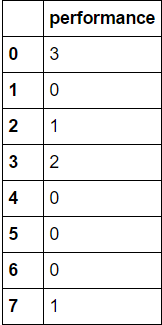
2). KNN Model

Logistic Regression

We imported logistic regression from scikit learn package.

Input Variables- We used the top 29 features as input variables using Recursive Feature Elimination



Output - We divided the team performance into four categories [0,1,2,3,4] where

0->implies team lost in Division Series

1-> team lost in Champions Series

2-> team is World Series loser

3->Team won the World series

Step1

The data was split into training and test variables as

x\_train->It contains input variables data from year 1995-2015 for all playoff teams

y\_train->It contains output data which has performance of all teams from year 1995-2015

x\_test->Input variable data for year 2016

y\_test-> performance of all teams in year 2016

Step 2

Fit the training input(x\_train) and output data(y\_train) into logistic regression model to train the algorithm

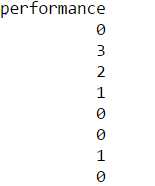
Step 3

Predict the output performance(y\_pred) of all eight teams for year 2016 with input(x\_test)

Step 4

Compare the predicted output performance(y\_pred) with actual performance(y\_test) to get various parameters like accuracy, precision, recall and F1 score.

Conclusion- We were able to predict the world series winner correctly using this model.



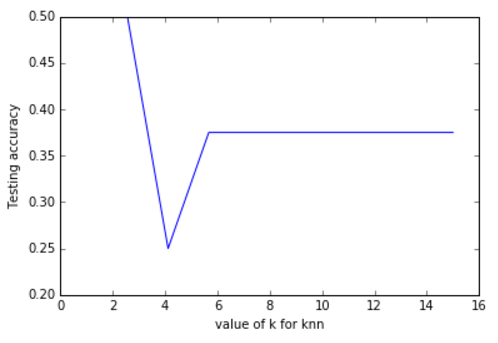
Actual Performance Predicted Performance

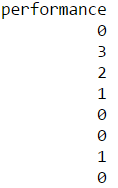
Accuracy-62%, Precision-52.5%, Recall-56.25% and F1 Score=54.166%

KNN Model

We applied the K Nearest Neighbor classifier on above data while splitting into same x\_train, y\_train, x\_test and y\_test variables

On plotting the graph between k-nearest neighbor and Testing Accuracy we found that accuracy is higher when k=1 or 2



The output result of KNN model is



Predicted Performance

Actual Performance

Accuracy-50%, Precision-50%, Recall-50% and F1 Score=50%

Conclusion- We are able to predict some of the teams which were world series loser, Division series loser, Champion series loser but the model has a shortcoming of not able to predict the world series winner for year 2016.

Some other variations we tried on dataset

1). We applied the KNN and Logistic Regression by dividing the training and test data set into 60:40 ratio to predict the performance of team for multiple years

KNN Output

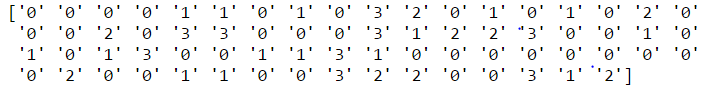
Accuracy 0.342857142857

Precision 0.294429050356

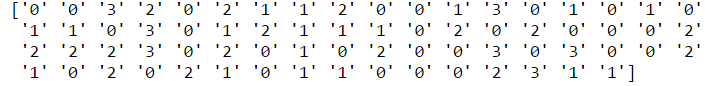
Recall 0.274774774775

F1 score 0.27946969697

Actual Performance



Predicted Performance



Logistic Regression Output

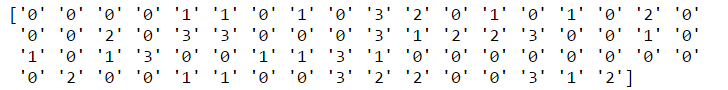
Accuracy 0.342857142857

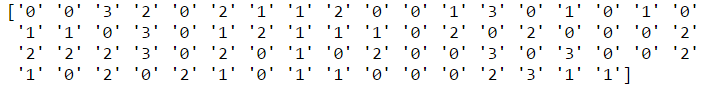
Precision 0.294429050356

Recall 0.274774774775

F1 score 0.27946969697

Actual Performance



Predicted Performance

2). KNN and Logistic Regression with K-mean cross validation by splitting Training and Test data into 60:40

We applied the KNN and logistic Regression using k-mean cross validation to check the accuracy and precision

KNN output-> Accuracy=50.95% Precision=27.87%

Logistic Regression -> Accuracy=49.20% Precision=24.20%

# Conclusions

Overall, we were somewhat satisfied with our model’s results. Given the inherent randomness in sports – and baseball in particular – being able to mathematically predict the world series winner is no small feat.

While we were able to correctly predict the world series winner in a few seasons, our model was not without flaws. In certain cases, the bulk of the model’s accuracy was achieved by predicting every team to lose in the first round of the playoffs. Since 4 out of 8 teams lose in the first round, predicting all 8 teams to lose guarantees an “accuracy” of 50%. This result was typical of the KNN algorithm but not as prevalent with logistic regression. A possible remedy for this situation would be to incorporate some sort of linearity constraint which forces the model to choose a winner and loser in each round of the playoffs.

The next natural step for our model would be to incorporate more data. Team statistics such as SLG, WHIP, and BA are certainly valid predictors for playoff performance, but there are myriad other factors which contribute to a team’s chances of winning a world series. Several extensions for our model include:

* Each team’s expected performance against their opponent
* Strength of schedule during the regular season
* Recent performance for the past ‘n’ regular season games before the playoffs
* Home field advantage
* Individual player analysis

We plan to further investigate each of these factors and improve our model in the coming weeks.