

Design and Implementation of Analytics System:

Analyzing Baseball Data & Predict the Winners of 2020 Series

Project Background:

Baseball has been referred to as America's national pastime. Since its start in the mid-19th century, the game has grown in popularity across the globe. Many professional leagues have come and gone since the game began, with Major League Baseball (MLB) becoming the premier league. MLB held its first game in 1869 and continues to play games to this day. Financially, the MLB generates a great deal of money, and according to Forbes annual valuation of the MLB, the league generated \$10.5 billion in revenue during the 2019 season. However, this year is different. Due to coronavirus, the 2020 season has been vastly changed.

The season's opening day was pushed back by three months and the number of games to be played is now at 60 versus the usual 162 game season. Additionally, fans are not allowed to be in the stadiums when the games are played. Through September 2nd, 41 games have been postponed due to Covid-19 concerns. With all these changes affecting the game and its players, we want to see if we can accurately predict several data points for the 2020 MLB season.

Forbes annual valuation URL:

<https://www.forbes.com/sites/mikeozanian/2020/04/09/despite-lockdown-mlb-teams-gain-value-in-2020/#711e0e162010>

Scope:

Baseball is a game of trends. With normal seasons being 162 games, where each team plays most other teams across multiple series, it's normal to get a large sample size of regular season data which can be used to predict post season success. While the post season matchups are much shorter in length and they can be subject to short term variance and predicted winners could end up losing, the teams that are believed to be the best regular season teams are likely to be the best post season teams. However, 2020 has changed that. With MLB having only 37% of their regular games, it might be much more difficult to predict the post seasons successes this year. Additionally, MLB has been considering a reduction in the number of games played for a while now. Perhaps they could see this unique season as a working hypothesis to determine the effect that fewer games would have. Many different groups of people would be interested in answering these questions: Major League Baseball, individual baseball teams, casinos, even personal sports betters would be very interested in the outcome of the 2020 MLB season, and what prediction power the shortened season has.

Research Questions:

- Who will win the 2020 Cy Young for each league?
- Who will win the 2020 MVP Winners for each league?
- Who will win the National League? American League?
- Who will win the World Series in 2020?

Tools & Technologies:

- Scripting – R, Python / Jupiter-Notebooks
- ETL – KNIME / API for Current Year Data
- MongoDB for Datastore (NoSQL scalable database)
- Data Visualization – Tableau

Possible source of 2020 stats <https://www.baseball-reference.com/leagues/MLB/2020.shtml>

Data:

The data for the project is primarily taken from the Lahman baseball dataset which is publicly available. The Lahman Baseball Dataset is used frequently in data analysis as it contains pitching, hitting, and fielding statistics for Major League Baseball from 1871 through 2017. It includes data from the two current leagues (American and National), the four other "major" leagues (American Association, Union Association, Players League, and Federal League), and the National Association of 1871-1875.

The database was created by Sean Lahman, who pioneered the effort to make baseball statistics freely available to the public. What started as a one-man effort in 1994 has grown tremendously, and now a team of researchers have collected their efforts to make this the largest and most accurate source for baseball statistics available anywhere.

The raw data consists of 26 *.csv files as shown in Table 1. Each *.csv contains between 4 and 56 attributes and records which range from 53 to 143,047 (see Table 1). The table can be joined via inner joins to provide additional details when needed.

Table Design:

The design follows these general principles. Each player is assigned a unique number (playerID). All the information relating to that player is tagged with his playerID. The playerIDs are linked to names and birthdates in the MASTER table.

The database is comprised of the following main tables (See Table 1 for attribute and records counts):

- People - Player names, DOB, and biographical info
- Batting - batting statistics
- Pitching - pitching statistics
- Fielding - fielding statistics

It is supplemented by these tables (see Table 1 for attribute and records counts):

- AllStarFull - All-Star appearances
- Appearances - details on the positions a player appeared at
- AwardsManagers - awards won by managers
- AwardsPlayers - awards won by players
- AwardsShareManagers - award voting for manager awards
- AwardsSharePlayers - award voting for player awards
- BattingPost - post-season batting statistics
- CollegePlaying - list of players and the colleges they attended
- FieldingOF - outfield position data
- FieldingOFsplit - LF/CF/RF splits
- FieldingPost - post-season fielding data
- HallofFame - Hall of Fame voting data

- HomeGames - Number of home games played by each team in each ballpark
- Managers - managerial statistics
- ManagersHalf - split season data for managers
- Parks - list of major league ballparks
- PitchingPost - post-season pitching statistics
- Salaries - player salary data
- Schools - list of colleges that players attended
- SeriesPost - post-season series information
- TeamFranchises - franchise information
- Teams - yearly stats and standings
- TeamsHalf - split season data for teams

Table 1: Lahman Baseball Dataset

File	Attributes/Variables	Records
AllstarFull.csv	8	5376
Appearances.csv	21	107,358
AwardsManagers.csv	6	180
AwardsPlayers.csv	6	6,237
AwardsShareManagers.csv	7	426
AwardsSharePlayers.csv	7	6,880
Batting.csv	22	107,430
BattingPost.csv	23	14,751
CollegePlaying.csv	3	17,351
Fielding.csv	18	143,047
FieldingOF.csv	6	12,029
FieldingOFsplit.csv	18	33,280
HallOfFame.csv	8	4,192
HomeGames.csv	9	3,109
Managers.csv	10	3,537
ManagersHalf.csv	10	94
Parks.csv	6	256
People.csv	50	1,624
Pitching.csv	56	47,629
PitchingPost.csv	56	5,799
Salaries.csv	5	26,429
Schools.csv	5	1,208
SeriesPost.csv	9	344
Teams.csv	48	2,926
TeamsFranchises.csv	4	121
TeamHalf.csv	10	53

Additional Data Sources:

Additional data will be obtained from the following sources:

- mlb.com (Major League Baseball) - <https://www.mlb.com/>
- Baseball-Reference.com - <https://www.baseball-reference.com/>
- Retrosheet.com - <https://www.retrosheet.org/>

The Major League Baseball website provides copious statistical data, sortable and printable, updated instantly as

games progress. The main attraction of the MLB website is that it provides PITCHf/x data. That is, for every pitch thrown by any pitchers in MLB, they'll tell you the type of pitch, where it crossed the plate, and how much it broke vertically and horizontally. As a result, and not surprisingly, much of the groundbreaking research in sabermetrics has to do with pitch analysis.

Baseball-Reference.com (B-R) is like mlb.com but provides historical data. Not only can we get the regular Bill-Terry's-batting-average data, but you also get a large selection of sabermetric stats, breakdowns by tens of different criteria (left/right, day/night, April/September, and so on), and the ability to manipulate the data in ways that other websites don't allow.

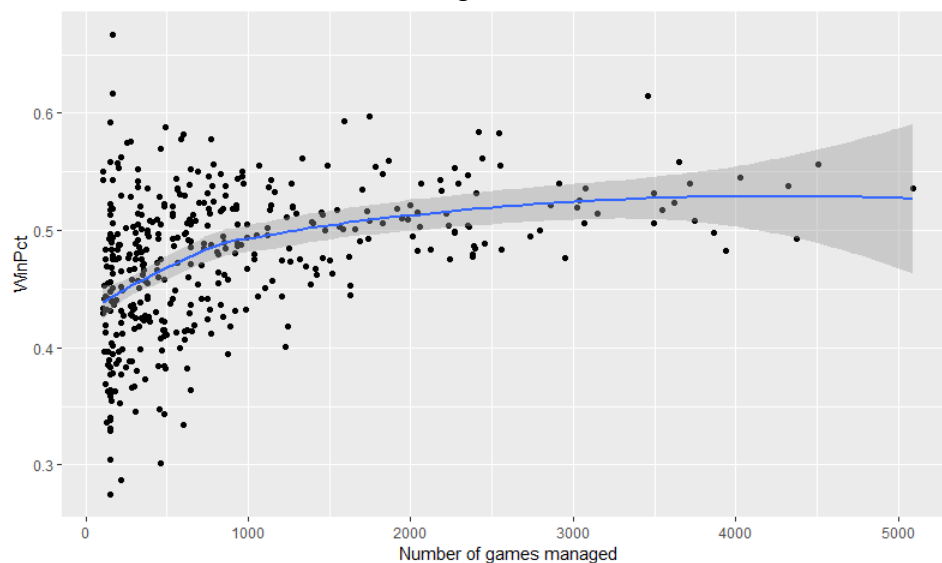
Data Acquisition:

MongoDB is a type of NoSQL database, called a document store. The data is stored in a JSON-like array in a format called BSON (binary JSON), which allows for quick and scalable (baseball related) queries. Java Script Object Notation (JSON) are name/value pairs that can be nested to store complex data. Another way of thinking of a MongoDB is that they are akin to a multidimensional array. Like many NoSQL databases, the schema structure is dynamic. This means that the descriptors of a dataset can be added, removed, or not even required for records to be stored into a given database.

Loading and accessing the data was made relatively easy with the mongolite R Package. Once data is uploaded to MongoDB, mongolite allows for a wide range of R function to perform data wrangling and visualization operations. Table 2 shows the results of a simple query of players and Figure 1 is an example plot of win percent (WinPct feature) versus the Number of games managed.

Table 2			
	PlayerID <chr>	NameFirst <chr>	NameLast <chr>
1	aardsda01	David	Aardsma
2	aaronha01	Hank	Aaron
3	aaronto01	Tommie	Aaron
4	aasedo01	Don	Aase
5	abadan01	Andy	Abad
6	abadfe01	Fernando	abad

Figure 1



Data Acquisition via Web:

As stated above, we will be accessing one to several websites to gather current baseball statistics from the 2020 season. The Lahman dataset is an excellent reference, but to gather current data we need to scrape the web using R. The rvest package in R is a great tool for scraping the necessary statistics from html web pages to test our models. The package itself was released in 2014 and is widely used and very stable.

In addition to rvest, we can use the Selector Gadget extension in Chrome to assist with selecting the correct fields from mlb.com. The Selector Gadget is an open-source extension that helps with CSS selection. The benefit of this tool is that you do not need to know html or CSS to select the correct data for our database.

For example, as we work on predicting the Cy Young winner for 2020, current pitching statistics will be vital to our models. Rvest will allow us to easily gather statistics from mlb.com, such as the ones shown in the image below. We must determine the fields we want and build out the necessary tables in R to hold the variables before they are moved to MongoDB or used in our test models. By using rvest and Selector Gadget, we essentially point and click on the data we need, such as ERA, Wins, Games Started, etc.

Picture 1

Hitting

Pitching

2020

Regular Season

MLB

Year to Date

Select Player Pool

Select a Split

Standard

Expanded

PLAYER	TEAM	W	L	ERA	G	GS	CG	SHO	SV	SVO	IP	H	R	ER	HR	HB	BB	SO		
1	Shane Bieber P	CLE	8	1	1.74	11	11	0	0	0	0	72.1	44	14	14	7	1	18	112	C
2	Corbin Burnes P	MIL	4	0	1.77	11	8	0	0	0	0	56.0	31	12	11	1	3	22	83	C
3	Trevor Bauer P	CIN	4	4	1.80	10	10	2	2	0	0	65.0	37	16	13	9	3	16	88	C
4	Yu Darvish P	CHC	7	2	1.86	10	10	0	0	0	0	63.0	47	14	13	4	2	12	79	C
5	Max Fried P	ATL	7	0	1.96	10	10	0	0	0	0	55.0	39	12	12	0	4	19	50	1
6	Dallas Keuchel P	CWS	6	2	2.04	10	10	0	0	0	0	57.1	48	14	13	2	0	15	39	1
7	Jacob deGrom P	NYM	4	1	2.09	10	10	0	0	0	0	56.0	38	16	13	4	0	14	80	C
8	Dinelson Lamet P	SD	3	1	2.10	11	11	0	0	0	0	60.0	36	15	14	5	4	18	80	C
9	Clayton Kershaw P	LAD	6	2	2.15	9	9	0	0	0	0	54.1	33	14	13	6	1	8	59	C
10	Kenta Maeda P	MIN	5	1	2.52	10	10	0	0	0	0	60.2	36	17	17	8	0	10	71	C
11	Lance Lynn P	TEX	6	2	2.53	12	12	1	0	0	0	78.1	52	24	22	10	6	23	84	C
12	Chris Bassitt P	OAK	5	2	2.57	10	10	0	0	0	0	56.0	51	18	16	6	2	17	49	1
13	Zack Wheeler P	PHI	4	0	2.62	9	9	0	0	0	0	58.1	55	19	17	3	5	9	38	1
14	Zach Davies P	SD	7	3	2.69	10	10	0	0	0	0	60.1	45	23	18	7	0	16	55	1
15	Adam Wainwright P	STL	5	1	2.87	8	8	2	0	0	0	53.1	40	19	17	7	2	12	44	C
16	Carlos Carrasco P	CLE	3	4	2.90	11	11	0	0	0	0	62.0	52	20	20	7	2	24	74	1

Datasets:

We have collected two separate datasets. The first data set is the Lahman dataset that is available in R through the Lahman package. The complete dataset is shown in Appendix A. This dataset will be providing all the historical data that we're using to train our models. It is a complete dataset of every baseball statistic that is currently used in baseball analytics. Our plan is using the past five years of data to train our models.

The second dataset has been downloaded from baseball-reference.com. There are a total of six different csv files associated with this dataset; the complete dataset is shown in Appendix B. The baseball-reference.com dataset contains data from the current 2020 season and is being used to test our models. Our initial plan was to use the baseball package currently available in R to scrape the baseball-reference site, however several variables that are required for our models were not being collected using that package. As an alternative, we downloaded the available csv files from the site as they contained the required variables.

Both sets of data are complete and already cleaned as they are official statistics provided by Major League Baseball (MLB). Several variables have null values as that variable doesn't apply to a specific player; but overall, the data is complete and ready to use. We will be keeping the missing values as they are. Outliers in the dataset are rare, as it takes a special player/season to outperform his peers. With that in mind, our first model attempts to predict who will win the Cy Young award, given to the best pitchers in baseball. One pitcher from the American League (AL) and one pitcher from the National League (NL) are each awarded the Cy Young each season.

The most direct approach is to calculate the Cy Young points for each pitcher. The Cy Young point formula will create a derived variable for us to use. All the required variables are available in both the Lahman dataset and the baseball-reference dataset. The necessary variables are shown below:

- Innings Pitched (IP)
- Earned Runs Allowed (ER)
- Strikeouts (SO)
- Saves (SV)
- Shutouts (SHO)
- Wins (W)
- Losses (L)

The formula to calculate Cy Young points is straightforward and is shown below:

$$\text{Cy Young Points (CYP)} = ((5 * \text{IP} / 9) - \text{ER}) + (\text{SO} / 12) + (\text{SV} * 2.5) + \text{Shutouts} + ((\text{W} * 6) - (\text{L} * 2))$$

While the Cy Young award focuses on an exceptional pitching season by a player, we are also working on a way to predict total wins for each team. Our objective is to build a multiple linear regression model using the Lahman data to predict the number of wins for the 2020 season. We'll use the baseball-reference dataset to test our model. All the following variables are present in both datasets.

Table 3

Variable Name	Definition
H	Base Hits by Batters (1B, 2B, 3B, HR)
2B	Doubles by Batters (2B)
3B	Triples by Batters (3B)
HR	Homeruns by Batters (HR)
BB	Walks by Batters
SO	Strikeouts by Batters
SB	Stolen Bases
CS	Caught Stealing
HBP	Batters hit by pitch

E	Errors
DP	Double Plays
H	Hits allowed
HR	Homeruns allowed
BB	Walks by Pitchers
SO	Strikeouts by Pitchers
W	Number of Wins by pitcher

Between these two models we have 21 required variables due to overlap of 2 variables. In addition, our last two research questions ask if we can predict the league winner (AL or NL) and the World Series Winner. To do this, we need to combine the above player data with win/loss data from the teams in which they played. This was done by joining the player data with data from the Teams table based on playerID, yearID, and teamID. The following list are variables included in the uncleaned dataset due to availability in both datasets.

Table 4

Variable Name	Definition
PlayerID	ID designation of player
NameLast	Last Name of Player
NameFirst	First Name of Player
batting_H	Base Hits by Batters (1B, 2B, 3B, HR)
batting_X2B	Doubles by Batters (2B)
batting_X3B	Triples by Batters (3B)
batting_HR	Homeruns by Batters (HR)
batting_BB	Walks by Batters
batting_SO	Strikeouts by Batters
batting_SB	Stolen Bases
batting_CS	Caught Stealing
batting_HBP	Batters hit by pitch
fielding_E	Fielding Errors
fielding_DP	Double Plays
pitching_H	Hits allowed
pitching_HR	Homeruns allowed
pitching_BB	Walks by Pitchers
pitching_SO	Strikeouts by Pitchers
pitching_W	Number of Wins by pitcher
pitching_L	Number of Losses by pitcher
pitching_SHO	Shutouts by pitcher
pitching_SV	Saves by pitcher
pitching_IP	Innings Pitched by pitcher
pitching_R	Runs allowed by pitcher
YearID	Year of data entry
TeamID	ID of team of data entry
team_W	Team wins
team_L	Team Losses

DivWin	Did they win the division (Y or N)
LgWin	Did they win the league (Y or N)
WSWin	Did they win the World Series (Y or N)

The following graphs, generated using the **DataExplorer** package in R, provide insight and detail into the uncleaned.

Figure 2 – Basic Statistics

Name	Value
Rows	140,703
Columns	32
Discrete columns	7
Continuous columns	25
All missing columns	0
Missing observations	1,105,448
Complete Rows	33,863
Total observations	4,502,496

Figure 3 - Percentages

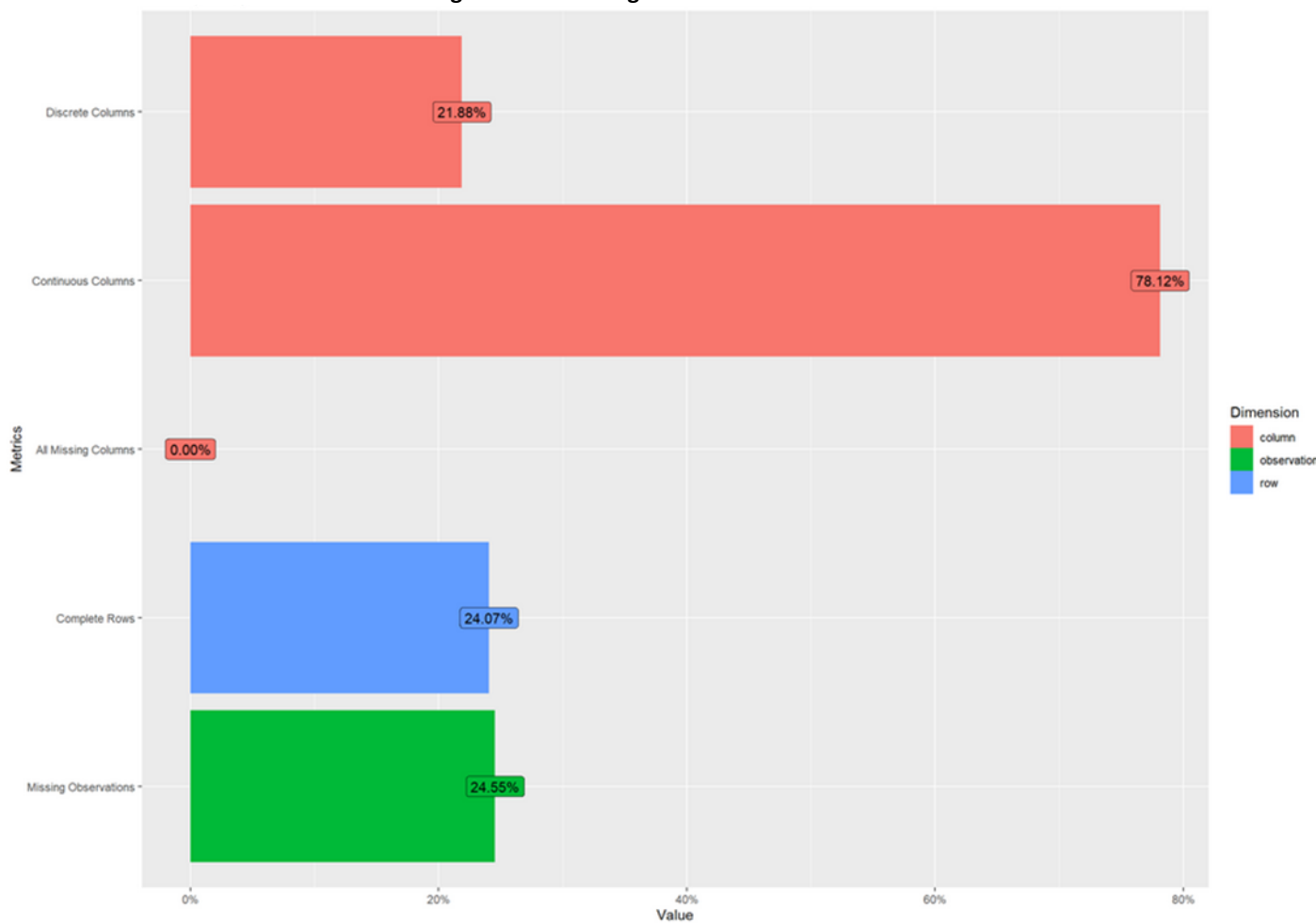


Figure 4 – Feature types and their categories (if available)

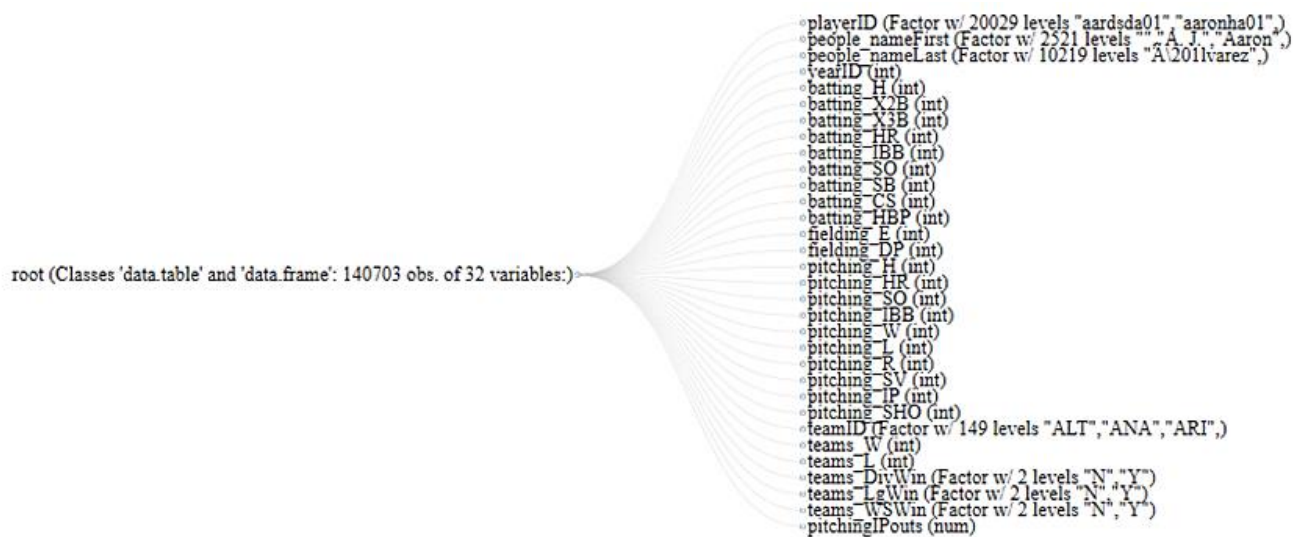


Figure 5 – Missing Data Profile

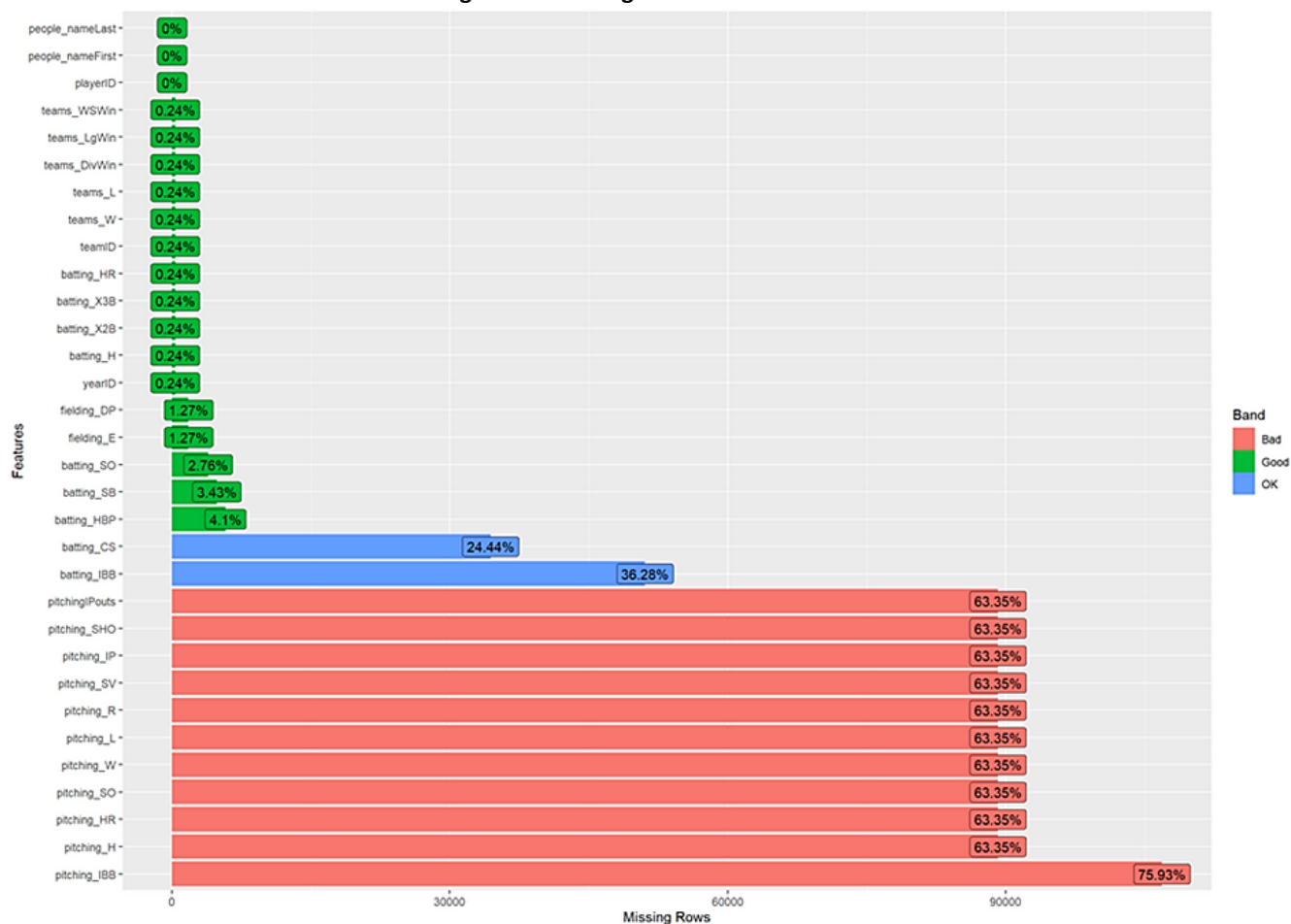


Figure 6 – Univariate Distribution (using Histogram)

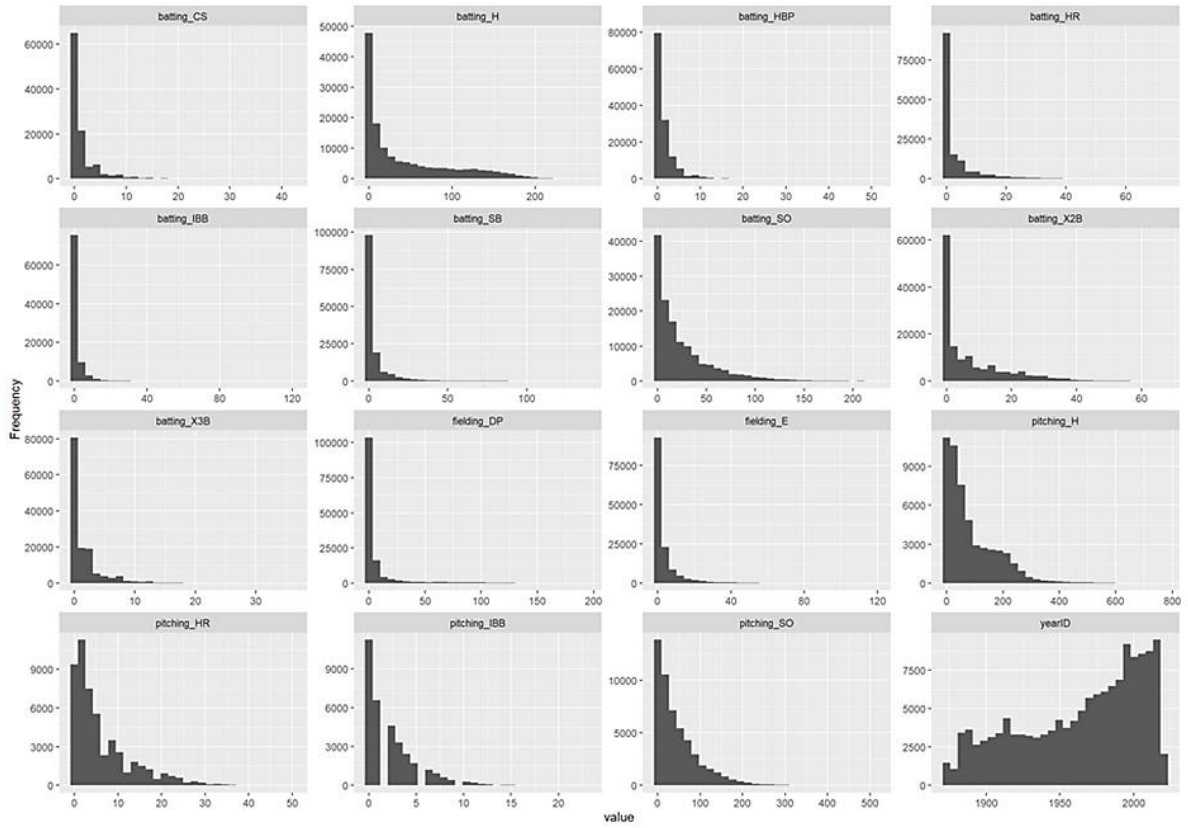


Figure 6 – Univariate Distribution (using Histogram) - Continued

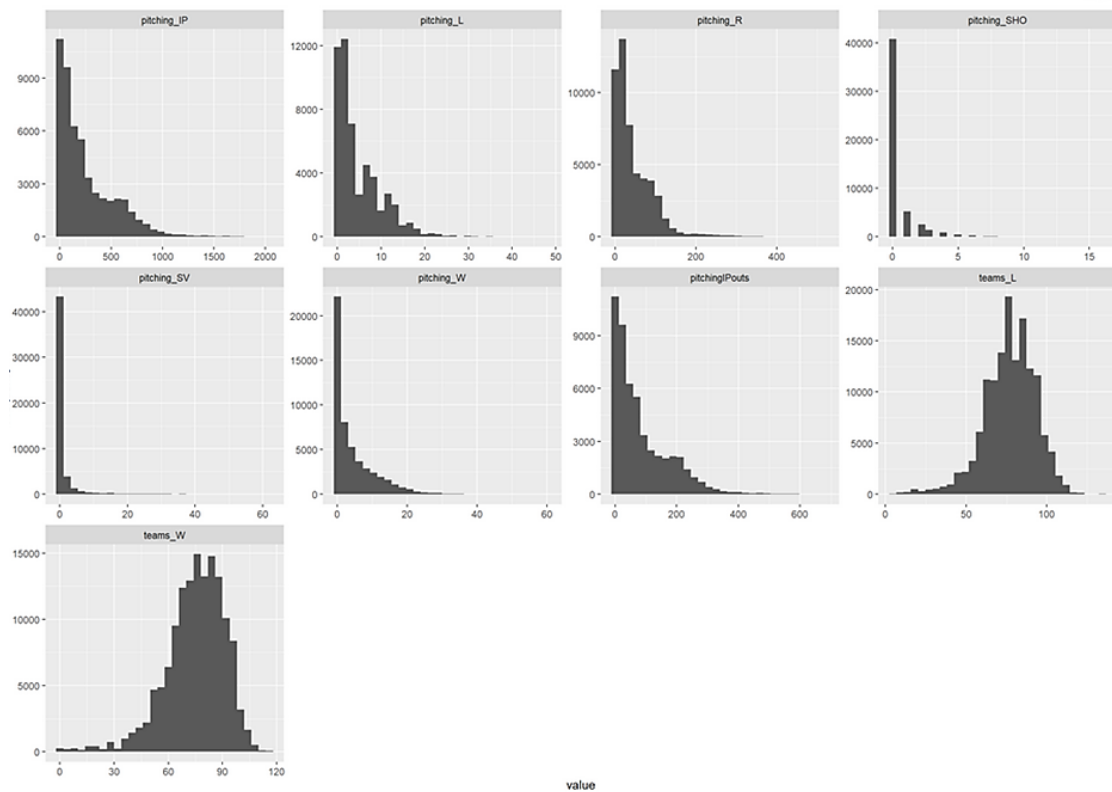
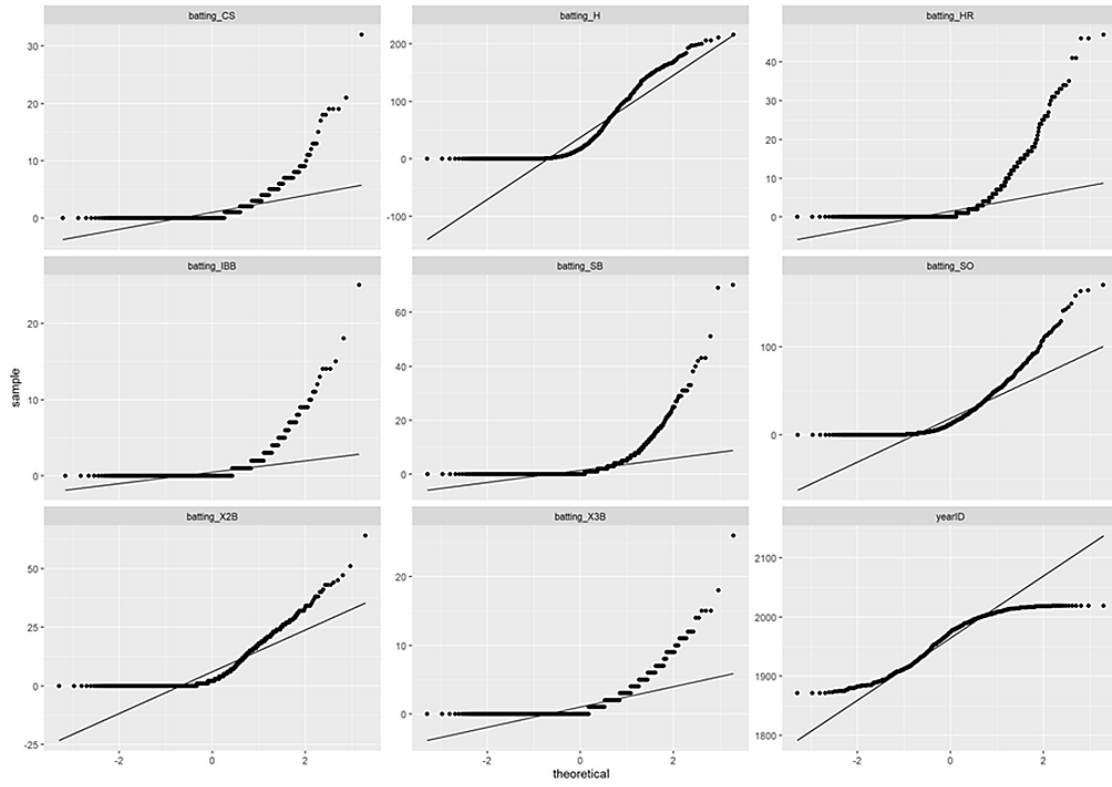


Figure 7 – QQ Plot

QQ Plot



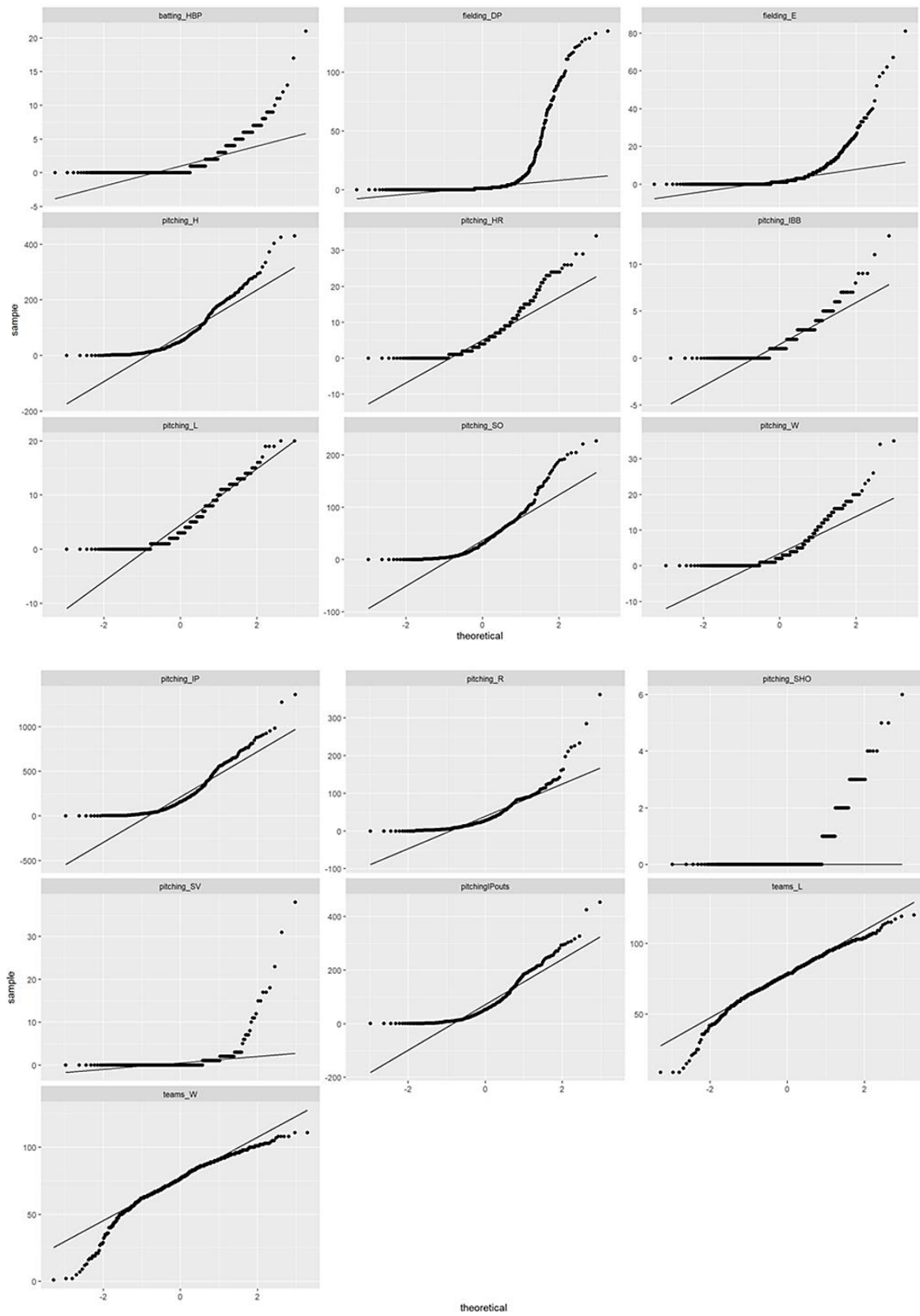
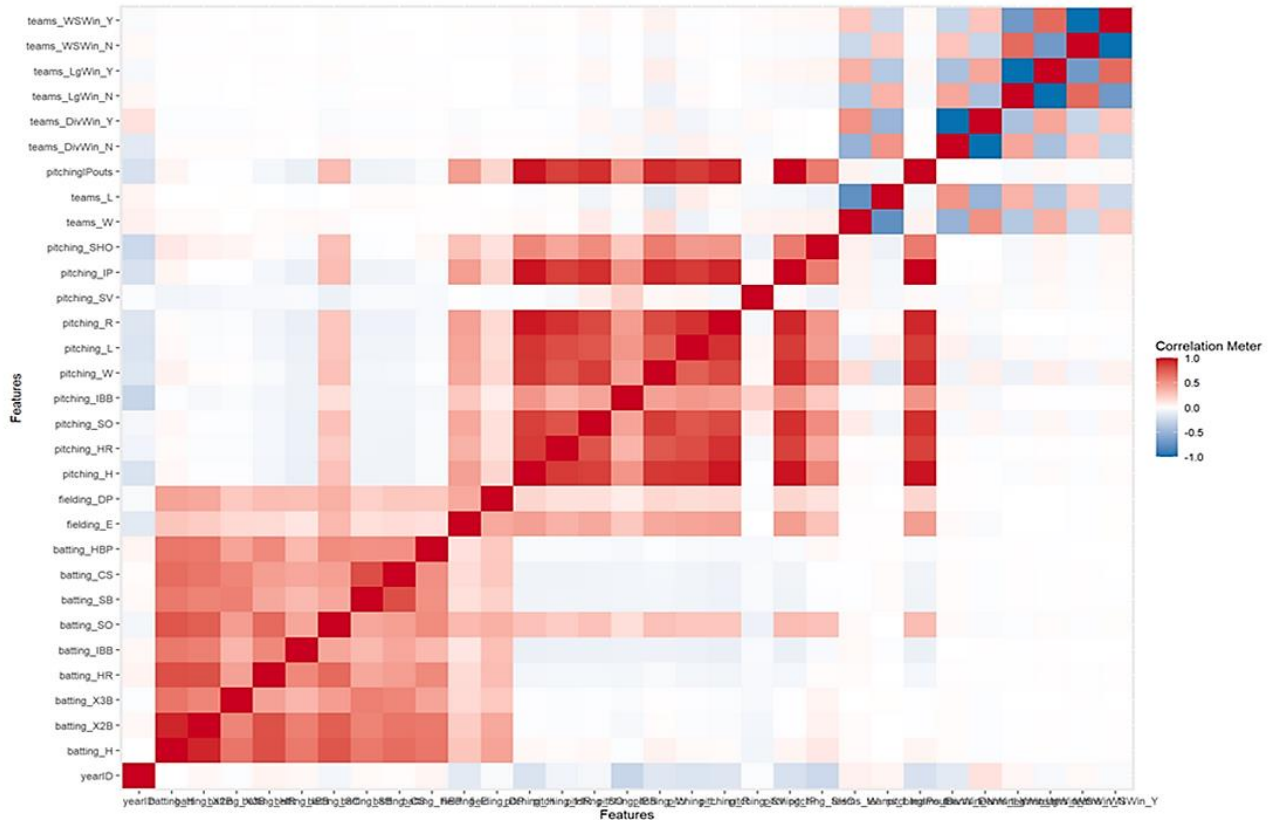


Figure 8 – Correlation Analysis



The uncleaned dataset contains 140,173 rows and 32 columns for a total of 4,502,496 observations. Of variables included, 7 are discrete and 25 are continuous. After the data is joined together, about 25% of the data is missing which will need to be cleaned and fixed before analysis can begin. Initial analysis of the histograms from the uncleaned dataset suggests that most of the variables are heavily skewed in one direction or another. In addition, an initial look at the quantile plots of each of the numeric variables suggests that most of the variables are not normal and that will need to be addressed as well before modeling can begin. Finally, the correlation plot of the uncleaned data shows a few things. Much of the batting variables show some correlation with each other. The same is true for the pitching variables and the team variables.

Now that the dataset has been fully defined, it is now time to select the necessary attributes to be included in the data model. To do this, we will use the backward elimination method. In this, a model will be generated using the entire set of data, and there will be a p-value generated for each attribute. Attributes are then dropped from the dataset, starting with the attribute with the highest p-value greater than 5%, and the p-values of the remaining attributes are then recalculated. Once all the appropriate attributes have been removed, the R-squared value of the original model will be compared to the new models' R-squared value. If there wasn't a steep drop off in this value, it was acceptable to have eliminated the above attributes.

Team Wins – Variable Selection:

Figure 9

```

Call:
lm(formula = Tm_W ~ ., data = baseball)

Residuals:
    Min       1Q   Median       3Q      Max
-21.2662  -3.7386   0.0546   3.9559  17.2501

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  24.7742153   2.0502554   12.083 < 2e-16 ***
Tm_Batting_H    0.0774767   0.0023005   33.678 < 2e-16 ***
Tm_Batting_2B  -0.0406723   0.0049720   -8.180 4.97e-16 ***
Tm_Batting_3B    0.0569798   0.0119989    4.749 2.19e-06 ***
Tm_Batting_HR    0.0879498   0.0051704   17.010 < 2e-16 ***
Tm_Batting_BB    0.0498710   0.0018455   27.023 < 2e-16 ***
Tm_Batting_SO    0.0006357   0.0013649    0.466 0.64146
Tm_Batting_SB    0.0211227   0.0046719    4.521 6.51e-06 ***
Tm_Batting_CS    0.0136796   0.0111573    1.226 0.22032
Tm_Batting_HBP   0.0217694   0.0103426    2.105 0.03543 *
Tm_Fielding_E   -0.0186472   0.0065780   -2.835 0.00463 **
Tm_Fielding_DP   0.0262573   0.0027285    9.623 < 2e-16 ***
Tm_Pitching_H   -0.0494205   0.0019849  -24.898 < 2e-16 ***
Tm_Pitching_HR  -0.1090372   0.0060673  -17.971 < 2e-16 ***
Tm_Pitching_BB  -0.0450425   0.0019813  -22.734 < 2e-16 ***
Tm_Pitching_SO   0.0122679   0.0013354    9.187 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.728 on 2006 degrees of freedom
(903 observations deleted due to missingness)
Multiple R-squared:  0.8052,    Adjusted R-squared:  0.8038
F-statistic: 552.9 on 15 and 2006 DF,  p-value: < 2.2e-16

```

“Tm_Batting_SO” is the attribute with the highest p-value at 0.64146. Once that is removed and the model updated, we get the following results:

Figure 10

```

Call:
lm(formula = Tm_W ~ Tm_Batting_H + Tm_Batting_2B + Tm_Batting_3B +
    Tm_Batting_HR + Tm_Batting_BB + Tm_Batting_SB + Tm_Batting_CS +
    Tm_Batting_HBP + Tm_Fielding_E + Tm_Fielding_DP + Tm_Pitching_H +
    Tm_Pitching_HR + Tm_Pitching_BB + Tm_Pitching_SO, data = baseball)

Residuals:
    Min       1Q   Median       3Q      Max
-21.2405  -3.7421   0.0529   3.8992  17.2917

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  24.892095   2.034175  12.237 < 2e-16 ***
Tm_Batting_H    0.076970   0.002026  37.985 < 2e-16 ***
Tm_Batting_2B  -0.040490   0.004956  -8.171 5.37e-16 ***
Tm_Batting_3B   0.057429   0.011958   4.803 1.68e-06 ***
Tm_Batting_HR   0.088632   0.004957  17.879 < 2e-16 ***
Tm_Batting_BB   0.049818   0.001842  27.051 < 2e-16 ***
Tm_Batting_SB   0.021517   0.004594   4.684 3.00e-06 ***
Tm_Batting_CS   0.013420   0.011141   1.205 0.22852
Tm_Batting_HBP  0.022283   0.010282   2.167 0.03034 *
Tm_Fielding_E  -0.018592   0.006576  -2.827 0.00474 **
Tm_Fielding_DP  0.026485   0.002684   9.868 < 2e-16 ***
Tm_Pitching_H  -0.049088   0.001852 -26.507 < 2e-16 ***
Tm_Pitching_HR -0.108873   0.006056 -17.978 < 2e-16 ***
Tm_Pitching_BB -0.045054   0.001981 -22.745 < 2e-16 ***
Tm_Pitching_SO  0.012728   0.000899  14.157 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.726 on 2007 degrees of freedom
(903 observations deleted due to missingness)
Multiple R-squared:  0.8052,    Adjusted R-squared:  0.8038
F-statistic: 592.6 on 14 and 2007 DF,  p-value: < 2.2e-16

```

“Tm_Batting_CS” is the attribute with the next highest p-value at 0.22852. With that removed and the model updated, we get the following:

Figure 11

```

Call:
lm(formula = Tm_W ~ Tm_Batting_H + Tm_Batting_2B + Tm_Batting_3B +
    Tm_Batting_HR + Tm_Batting_BB + Tm_Batting_SB + Tm_Batting_HBP +
    Tm_Fielding_E + Tm_Fielding_DP + Tm_Pitching_H + Tm_Pitching_HR +
    Tm_Pitching_BB + Tm_Pitching_SO, data = baseball)

Residuals:
    Min       1Q   Median       3Q      Max
-24.9671  -4.0852   0.0519   4.0368  20.0357

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  30.4507016   1.7807309   17.100 < 2e-16 ***
Tm_Batting_H    0.0787458   0.0017846   44.126 < 2e-16 ***
Tm_Batting_2B  -0.0297595   0.0043743   -6.803 1.25e-11 ***
Tm_Batting_3B    0.0516104   0.0096655    5.340 1.01e-07 ***
Tm_Batting_HR    0.0760903   0.0049304   15.433 < 2e-16 ***
Tm_Batting_BB    0.0502698   0.0016942   29.671 < 2e-16 ***
Tm_Batting_SB    0.0241238   0.0026176    9.216 < 2e-16 ***
Tm_Batting_HBP   0.0100469   0.0084396    1.190  0.234
Tm_Fielding_E  -0.0324615   0.0027547  -11.784 < 2e-16 ***
Tm_Fielding_DP   0.0281768   0.0024207   11.640 < 2e-16 ***
Tm_Pitching_H   -0.0528769   0.0014962  -35.341 < 2e-16 ***
Tm_Pitching_HR  -0.1227229   0.0056688  -21.649 < 2e-16 ***
Tm_Pitching_BB  -0.0464308   0.0017761  -26.143 < 2e-16 ***
Tm_Pitching_SO   0.0139306   0.0008208   16.972 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.215 on 2728 degrees of freedom
(183 observations deleted due to missingness)
Multiple R-squared:  0.8089,    Adjusted R-squared:  0.808
F-statistic: 888.4 on 13 and 2728 DF,  p-value: < 2.2e-16

```

“Tm_Batting_HBP” is the next attribute with the highest p-value, at 0.234. After removing that attribute and updating the model, we get the following:

Figure 12


```

Call:
lm(formula = Tm_W ~ Tm_Batting_H + Tm_Batting_2B + Tm_Batting_3B +
    Tm_Batting_HR + Tm_Batting_BB + Tm_Batting_SB + Tm_Fielding_E +
    Tm_Fielding_DP + Tm_Pitching_H + Tm_Pitching_HR + Tm_Pitching_BB +
    Tm_Pitching_SO, data = baseball)

Residuals:
    Min       1Q   Median       3Q      Max
-26.7960  -4.2613  -0.0012   4.1449  21.0250

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  16.6127023   1.2143988   13.680 < 2e-16 ***
Tm_Batting_H    0.0829049   0.0017344   47.801 < 2e-16 ***
Tm_Batting_2B  -0.0382262   0.0042983   -8.893 < 2e-16 ***
Tm_Batting_3B   0.0536708   0.0095833    5.600 2.35e-08 ***
Tm_Batting_HR   0.0726048   0.0049777   14.586 < 2e-16 ***
Tm_Batting_BB   0.0524947   0.0016936   30.996 < 2e-16 ***
Tm_Batting_SB   0.0274084   0.0024544   11.167 < 2e-16 ***
Tm_Fielding_E  -0.0303240   0.0024542  -12.356 < 2e-16 ***
Tm_Fielding_DP   0.0300713   0.0024425   12.312 < 2e-16 ***
Tm_Pitching_H  -0.0484911   0.0013984  -34.677 < 2e-16 ***
Tm_Pitching_HR  -0.1310496   0.0056059  -23.377 < 2e-16 ***
Tm_Pitching_BB  -0.0452234   0.0017498  -25.845 < 2e-16 ***
Tm_Pitching_SO   0.0169493   0.0007583   22.352 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.321 on 2788 degrees of freedom
(124 observations deleted due to missingness)
Multiple R-squared:  0.8467,    Adjusted R-squared:  0.8461
F-statistic: 1283 on 12 and 2788 DF,  p-value: < 2.2e-16

```

We now have no attributes that have a p-value of greater than 0.05. Therefore, there is no need to further remove any additional attributes from the dataset. Now, looking at the R-squared values, the original model has an R-squared value of 0.8052, with the final model having an R-squared value of 0.8467. This would imply that the final model is somewhat a better fit for the data than the original dataset. But it's not a significant improvement. The important thing to note here is that the R-squared value didn't significantly drop. This means that our removal of the Tm_Batting_SO, Tm_Batting_CS, and Tm_Batting_HBP attributes from the dataset was appropriate.

Team Wins – Variable Transformation:

most of the attributes in the dataset are heavily skewed in one direction, i.e., most baseball players have under 10 homeruns and only a small number of players have 40+ homeruns in each season. Therefore, it will be necessary to perform some sort of transformation on the skewed attributes to normalize the data.

To do so, the remaining dataset will be compared to the log-based version of the dataset, along with a square root version as well. Here's a snapshot of one attribute that has been transformed:

Figure 13

	baseball_pruned.Tm_Pitching_HR	log_normalization.Tm_Pitching_HR	square_root_normalization.Tm_Pitching_HR
1	2	0.6931472	1.189207
2	6	1.7917595	1.565085
3	13	2.5649494	1.898829
4	5	1.6094379	1.495349
5	7	1.9459101	1.626577
6	3	1.0986123	1.316074

Here's the distributions of each dataset presented with histograms:

Figure 14 – Original Dataset

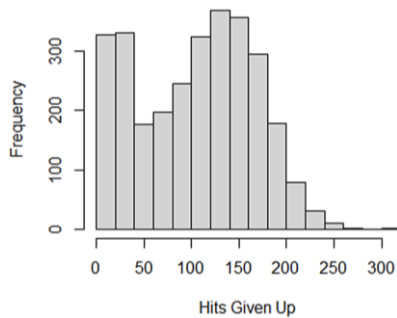


Figure 15 – Log Transformation

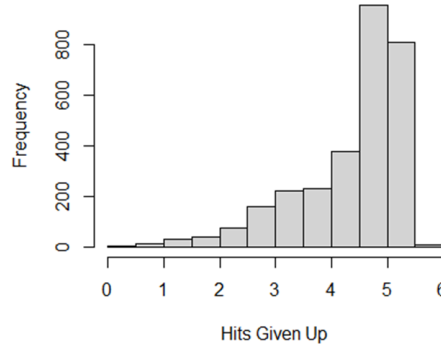
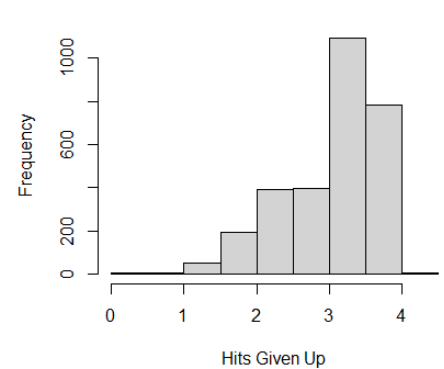


Figure 16 – Square-Root Transformation



After reviewing the distributions, it appears as though the square-root transformation does the best job in normalizing the dataset, although it is not perfect, it becomes unimodal along with a gradual decline on the left side of the plot, and a steeper decline towards the right. It does appear to be slightly skewed as well. But overall, it appears to be a notable improvement.

After reviewing the other attributes, it appears though some of the attributes will be normalized via the square-root method, some will be transformed via the log method, and some will remain in their original state. The important takeaway from this lesson's submission is that each attribute should be reviewed individually. You cannot apply a blanket solution to every attribute within the model and expect to see improvements to the results.

In addition to the selection and transformations steps detailed above, a second method was employed for feature selection and transformation to find the best possible data set for the MVP and winner models. This method uses the Boruta algorithm to determine feature importance. The Boruta algorithm is based on the Random Forest model. The algorithm generates a random dataset using the features you give it and builds a model based on those features. It then builds another model based on the actual dataset and compares the features to see which performed better. If the real data performs better, it is a hit. This process is repeated many times until enough hits or failures are generated to determine feature importance (Mazzanti). In addition to the algorithm, each variable was transformed to a scale of zero to one because the variables have a large range of values.

To begin, a unique dataset was created to answer each of the four research questions. A unique data set is needed because each question relates to different aspects of baseball and needs different data. To predict the MVP winners, only data after 1969 can be used because it was this year that the modern National League and American League both existed. In addition, only the players that participate in the NLCS and ALCS are eligible, so it does not make sense to look at every player but only the ones that were in their respective series'. The same logic is applied to predicting the National League winner, the American League winner, and the World Series winner. As stated above, the current Leagues are only 40 years old so only the past 40 years of data can be used for those predictions. Moreover, this same data must be used for predicting the World Series winner. The current baseball system uses the victors of the NLCS and ALCS as competitors, so any data gathered before this time is void because it was generated under the same circumstances. In addition, these datasets except for the World Series prediction must be mutually exclusive as a team in the American League cannot win the NCLS and vice versa. After these considerations, removing all the missing data, and converting any categorical variables into numeric ones using dummy variables, I arrived at the following dimensions for the datasets:

Table 5

Dataset	Number of Variables	Number of Entries
American League MVP	183	29763

National League MVP	187	31314
American League Championship	58	691
National League Championship	56	689
World Series	78	1380

After the algorithm was ran on the datasets, the following graphs were generated showing the importance of each feature. Green shows the important features while red shows the rejected features. The features in yellow are features that were still undecided when the algorithm terminated at 100 iterations. For initial modeling these yellow features will be ignored however the Boruta algorithm will be left to run for longer when the final modeling is completed. The superseding tables lists the variables to be chosen for each model respectively.

Figure 17 – AL (American League) MVP Dataset

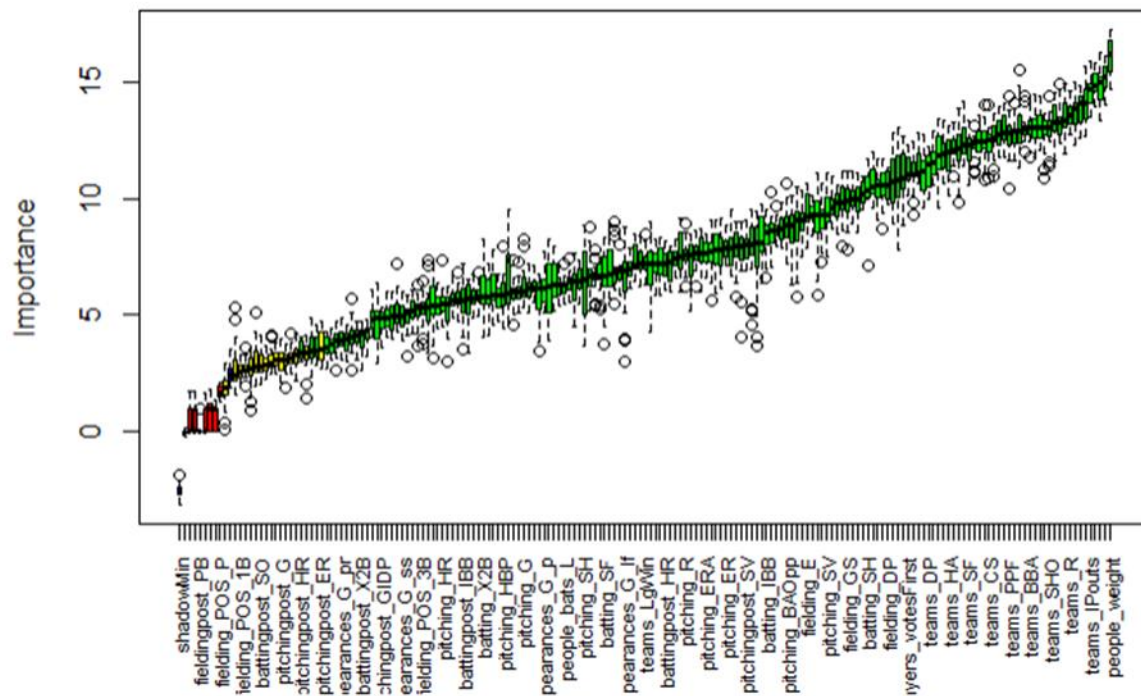


Figure 18 – NL (National League) MVP Dataset

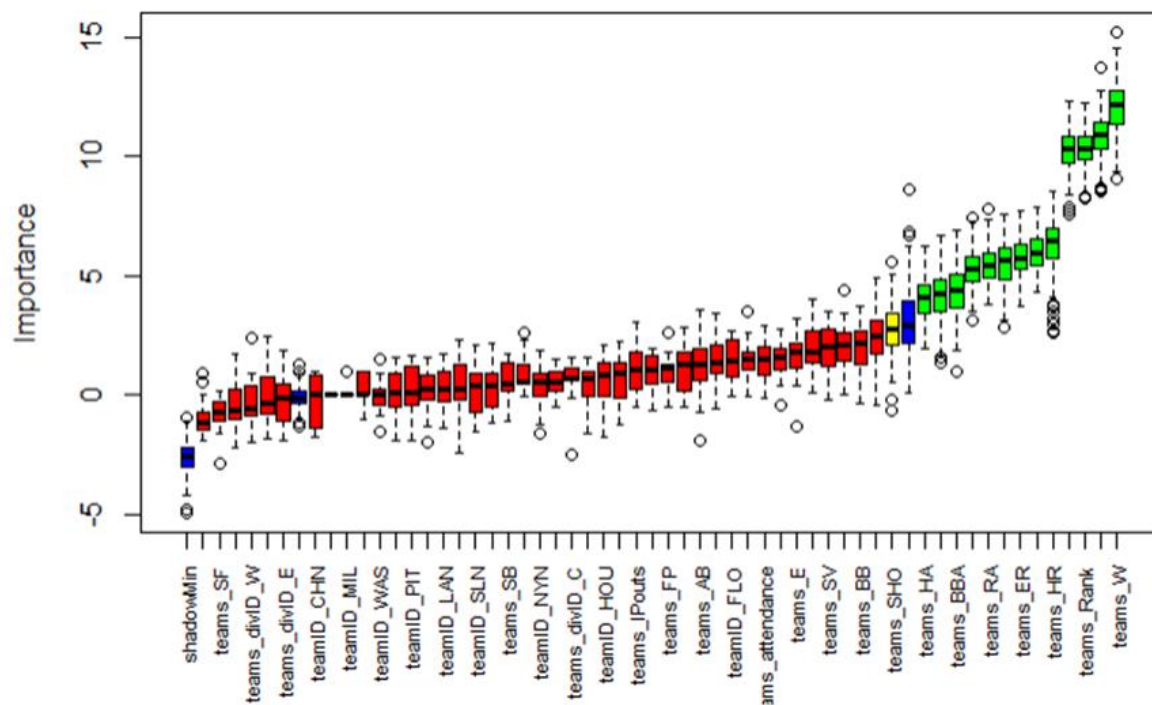


Figure 21 – World Series Dataset

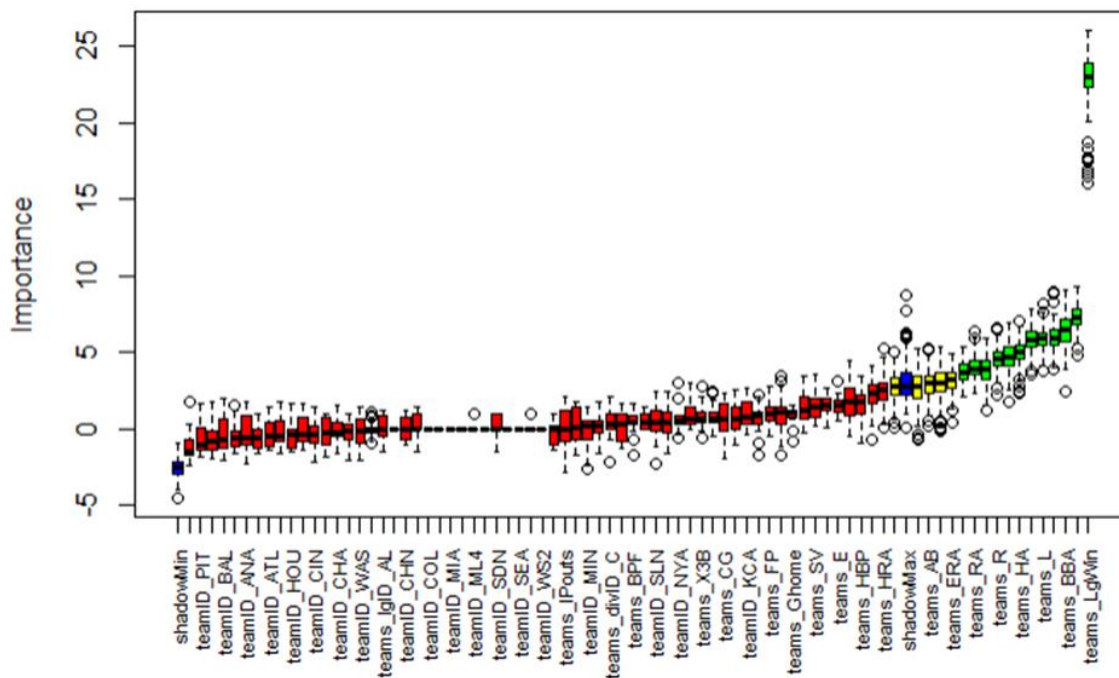


Table 6

Dataset	Selected Features
American League MVP	159 Confirmed Features

National League MVP	162 Confirmed Features
American League Championship (10)	Team Rank, Team Wins, Team Losses, Division Win (Binary), Team Runs, Team Hits, Runs scored by opponent teams, Team Earned Runs Allowed, Team Earned Run Average, Hits Allowed
National League Championship (13)	Team Rank, Team Wins, Team Losses, Division Win (Binary), Team Runs, Team Homeruns, Team Batters Caught by Pitch, Runs Scored by Opponent Teams, Team Earned Runs Allowed, Team Earned Run Average, Hits Allowed, Homeruns Allowed by opponent teams, Team walks allowed
World Series (11)	Team Rank, Team Wins, Team Losses, Division Win (Binary), League Win (Binary), Team Runs, Team Hits, Runs scored by opponent teams, Team Earned Runs Allowed, Hits Allowed, Team walks allowed, Florida Marlins (Binary)

To model these predictions, a couple different models will be employed to generate more accurate results. The models used are Neural Networks and Random Forests. To accompany these models, 10-fold, 3-repeated cross-validation and using a 70/30 training/test split will be used.

For the Cy Young prediction, we believe that the best features to use for prediction are in the formula detailed in earlier. Not only does external research show that this formula is fairly accurate, but our own attempts at modeling also using a larger dataset did not produce good results (Castrovince). An SVM model and an Ada Boost model were constructed to see if another approach could be used to predict the Cy Young winner. Using the Lahman dataset pitching statistics from 1957 onward, when the Cy Young was first awarded, a broad approach was utilized to see if other models could successfully predict the Cy Young winner. The models were run with an 80/20 split and then without splitting as the number of Cy Young winners is very low compared to the number of pitchers.

To generate the SVM model utilizing an 80/20 split; we rescaled all the pitching statistics from 1957 onward using a 0 to 1 scale. PlayerID, teamID, stint, and yearID were all excluded from the data set. The SVM model was then run to see if utilizing all available pitching statistics could successfully predict the Cy Young winner.

SVM Results

As shown in the Testing confusion matrix below, the SVM model was unable to predict a Cy Young winner in any instance. This is most likely due to the sheer number of pitchers that did not win the CY Young, as it's only award to two pitchers in MLB each year.

Table 7

Actual	Non-winner	CY winner	Error
Non-winner	5,487	0	
CY winner	9	0	

The Precision/Recall Plot below highlights the unreliable nature of the SVM model. While the Lift chart and the ROC Curve also highlight the fact that SVM model is not a good model to use when trying to determine the Cy Young winner.

Figure 22 – SVM Precision/Recall

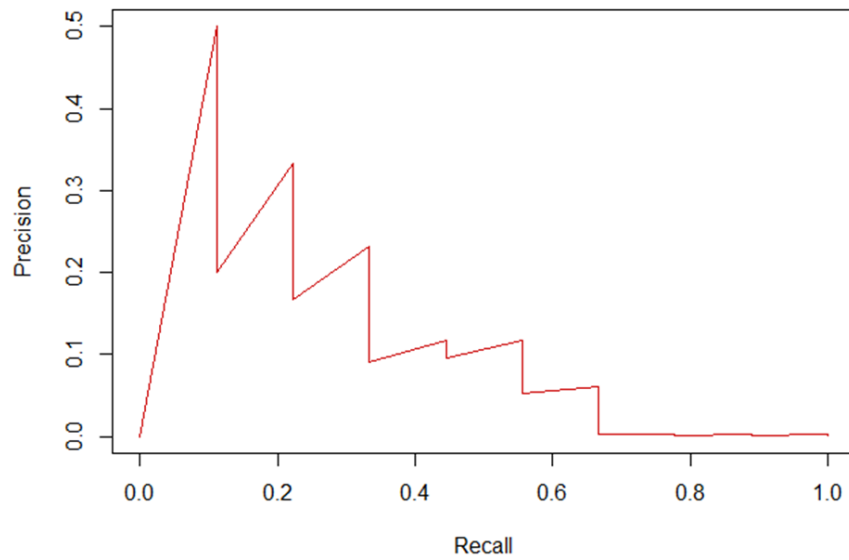


Figure 23 – SVM True Positive/False Positive Rate

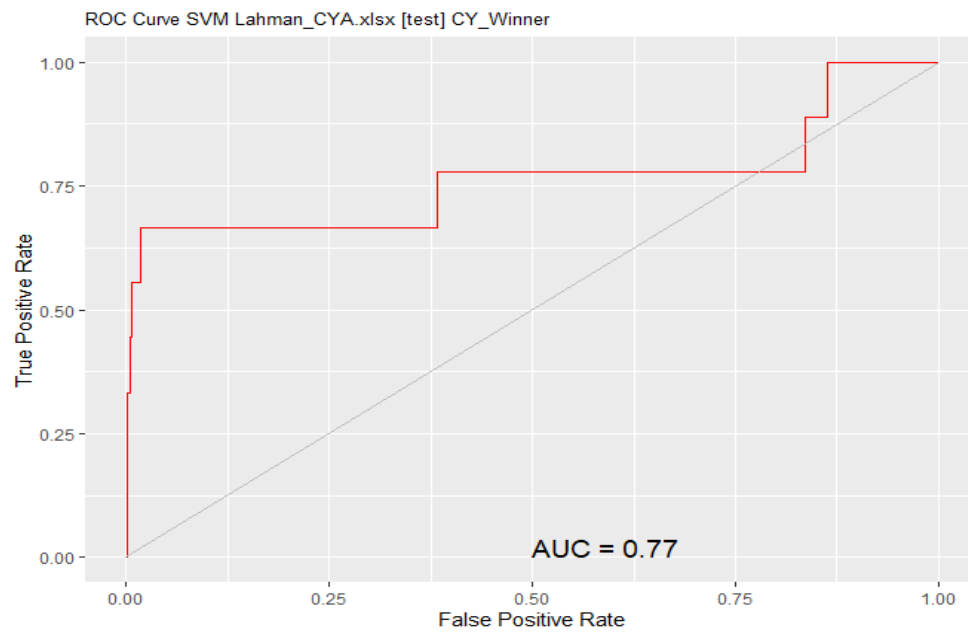
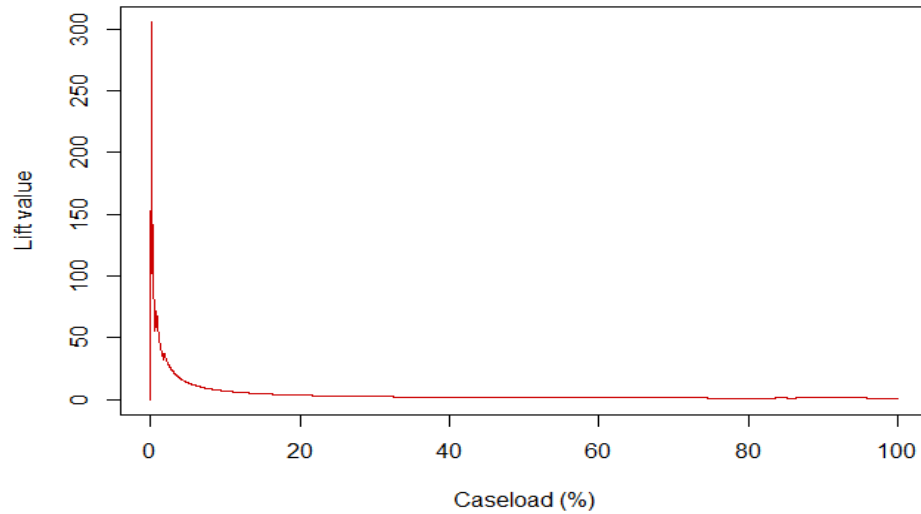


Figure 24 – SVM Life Value Plot



We reran the SVM model without splitting the data to see if better results could be achieved. Here are the results without splitting into a training/testing set. As you can see in the error matrix below, Cy Young winners were predicted this time; but the model achieved a high error rate (69.2) in predicting CY winners.

Table 8

Actual	Non-winner	CY winner	Error
Non-winner	27,225	0	
CY winner	36	16	69.2

Figure 25 – SVM Precision/Recall

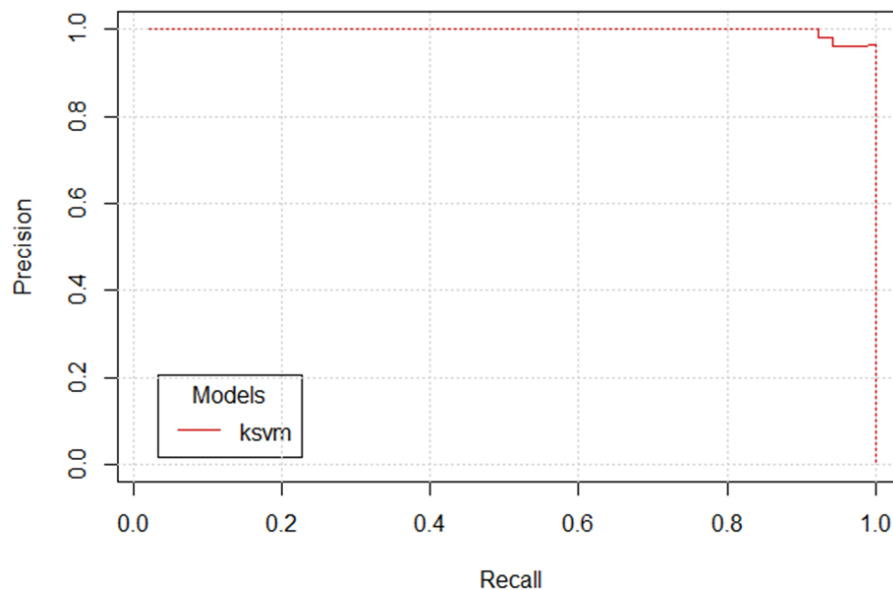


Figure 26 – SVM True Positive/False Positive Rate

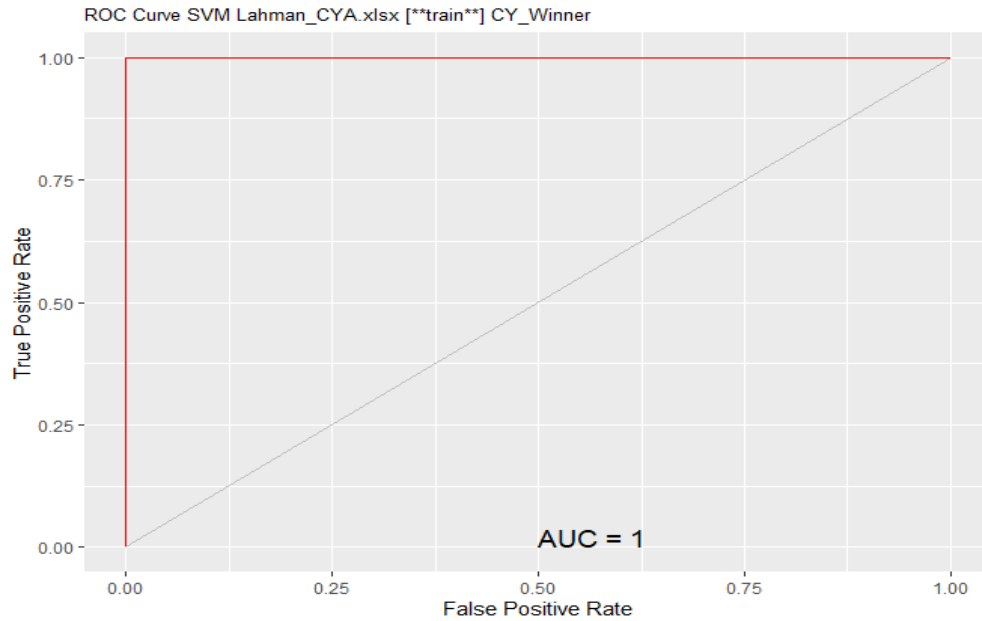
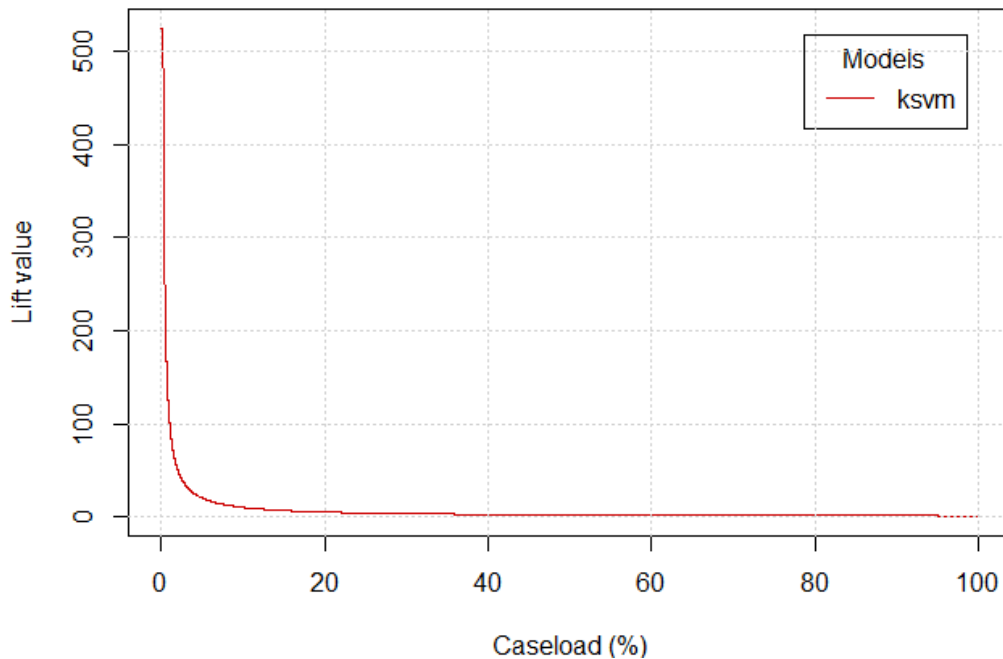


Figure 27 – SVM Life Value Plot



In addition to the SVM model, an Ada Boost model was run as well. To generate the Ada Boost model utilizing an 80/20 split; all the pitching statistics from 1957 onward were rescaled using a 0 to 1 scale. PlayerID, teamID, stint, and yearID were all excluded from the data set. The Ada Boost model was then run with a max depth of 6 and 2 iterations to see if utilizing all the pitching statistics could successfully predict the Cy Young winner. Additionally, variables with no values were removed from the data set as well.

Ada Boost Results

As shown in the Testing confusion matrix below, the Ada Boost model had a high error rate (77.8) when attempting to predict Cy Young winners. Again, this is most likely due to the sheer number of pitchers that did not win the Cy Young, as it's only award to two pitchers in MLB each year.

Table 9

Actual	Non-winner	CY winner	Error
Non-winner	5,486	1	
CY winner	7	2	77.8

Figure 28 – AdaBoost Precision/Recall

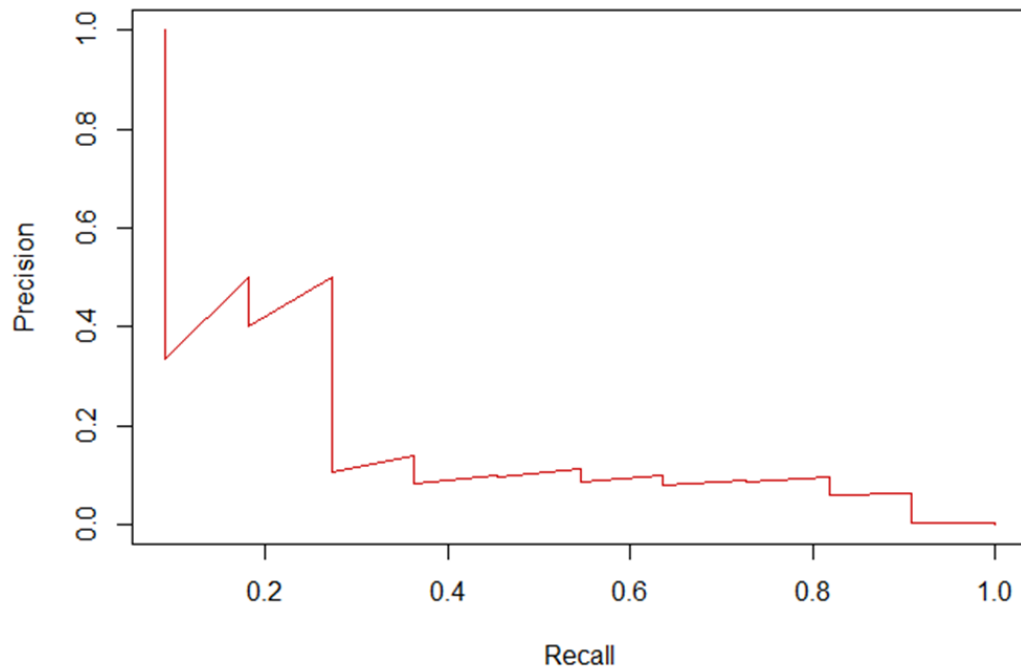


Figure 29 – AdaBoost True Positive/False Positive Rate

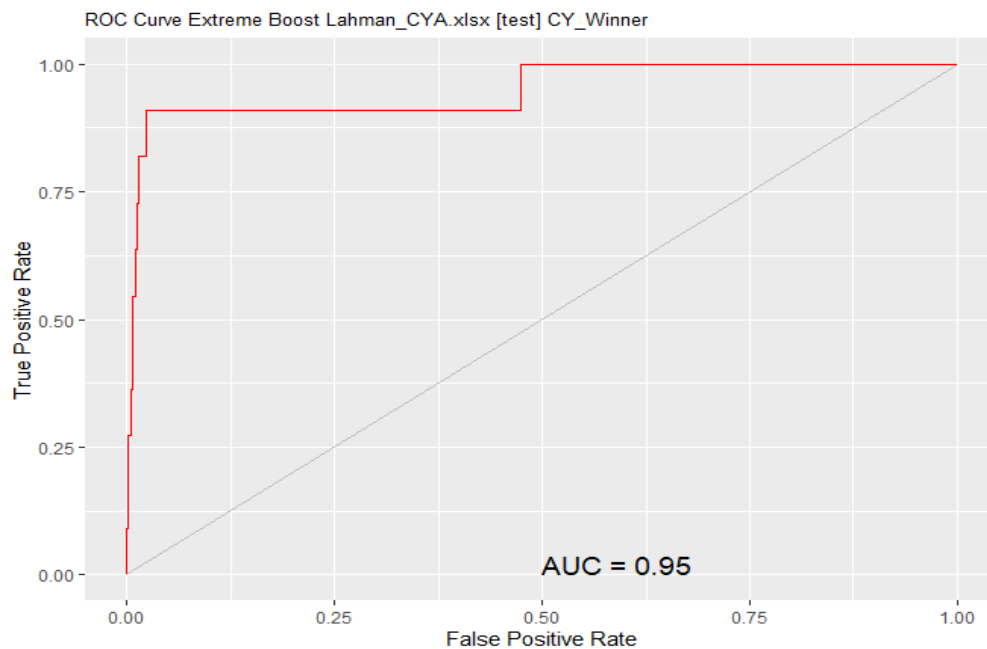
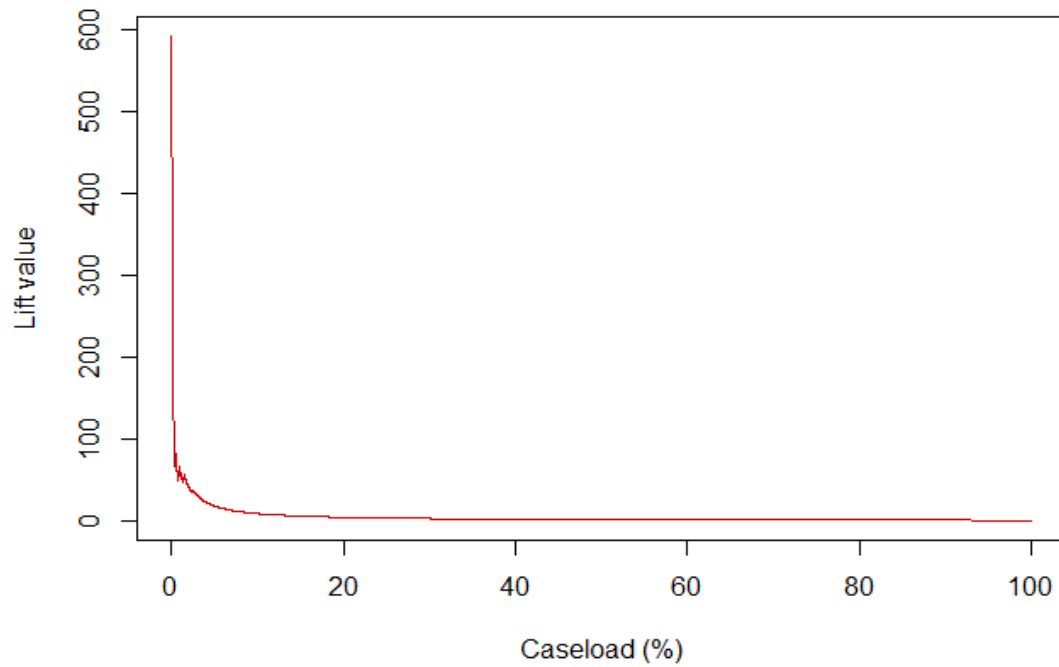


Figure 30 – AdaBoost Lift Value Chart



I reran the Ada model without splitting the data to see if better results could be achieved. Here are the results without splitting into a training/testing set. While the error rate improved from the 80/20 split, the model still achieved a high error rate (69.4) in predicting CY winners.

Table 10

Actual	Non-winner	CY winner	Error
Non-winner	32,432	6	
CY winner	43	19	69.4

Figure 31 – AdaBoost Precision/Recall

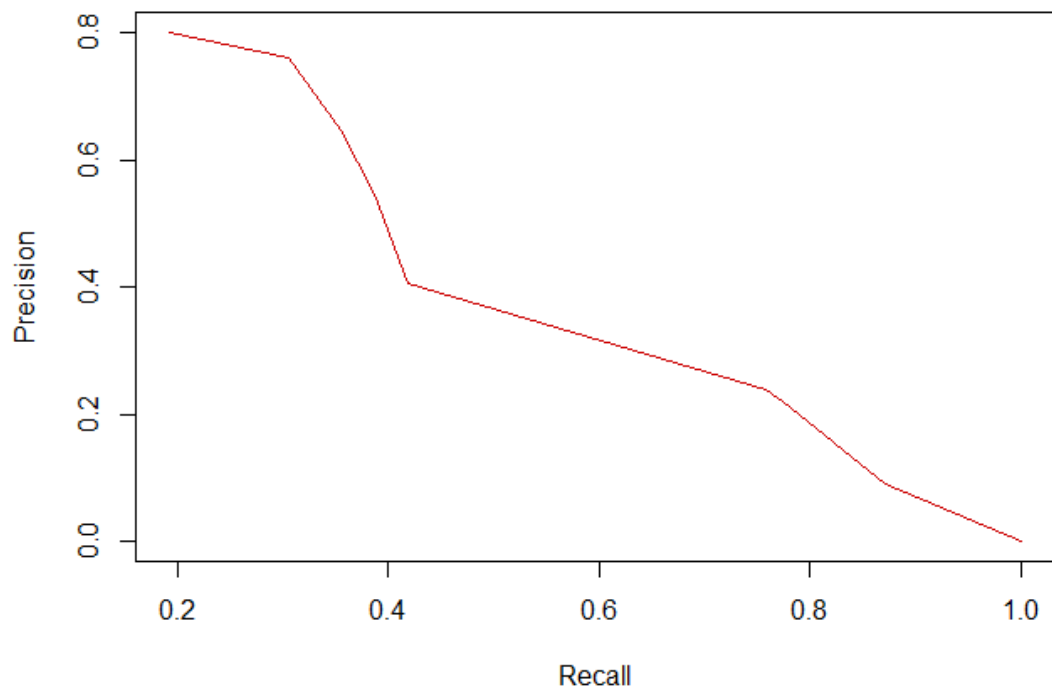


Figure 32 – AdaBoost True Positive/False Positive Rate

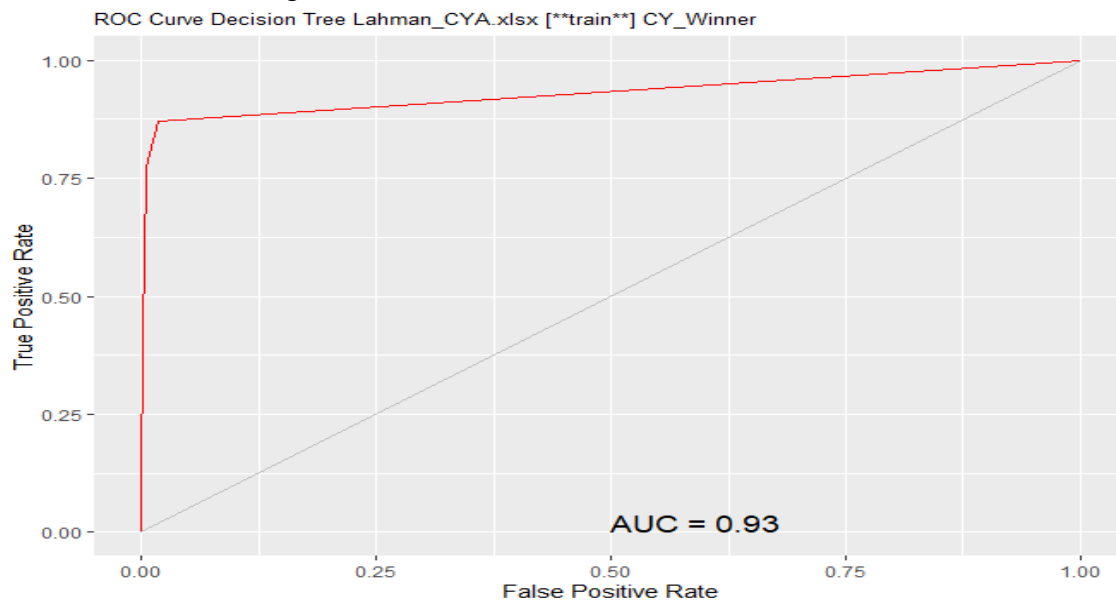
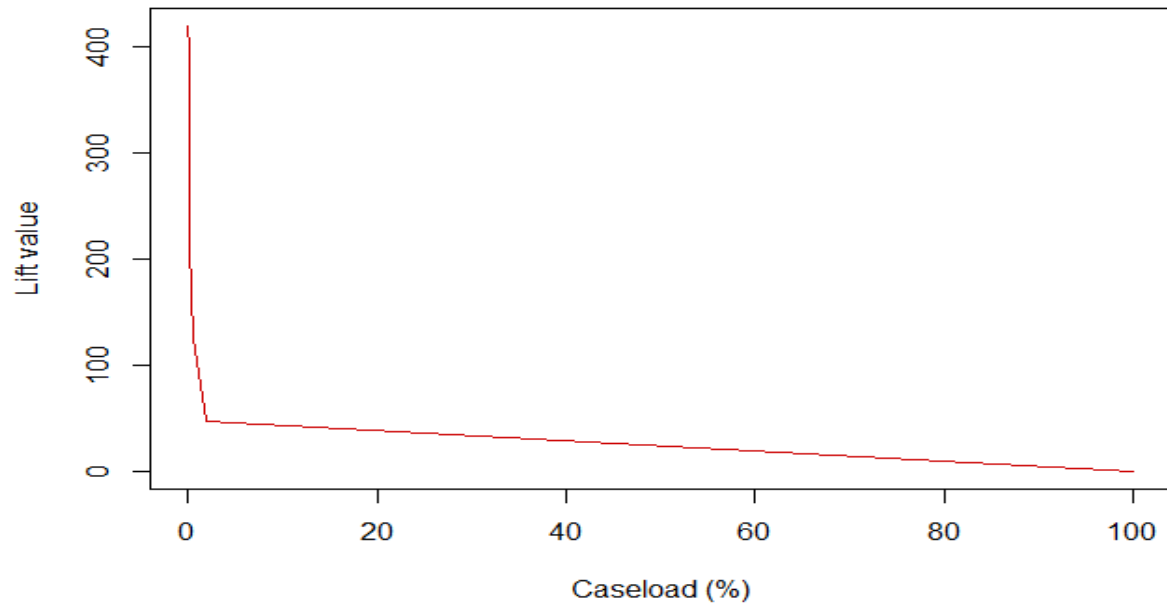


Figure 33 – AdaBoost Lift Value Chart



Overall Results

Overall, the SVM and Ada Boost models proved to not be good models to use when predicting the Cy Young winner. They were unable to predict the winner with any accuracy, having error rates above 69%. The formula approaches the group is utilizing is a far better approach than either the SVM or Ada Boost approaches.

The following table summarizes our attempts at generating predictive models to answer the research questions detailed at the beginning of this paper. Accuracy was the main metric we decided to use to evaluate model fitness. All our models are classification models, so we selected accuracy.

Table 11 – Overall Results of Random Forest and Neural Network

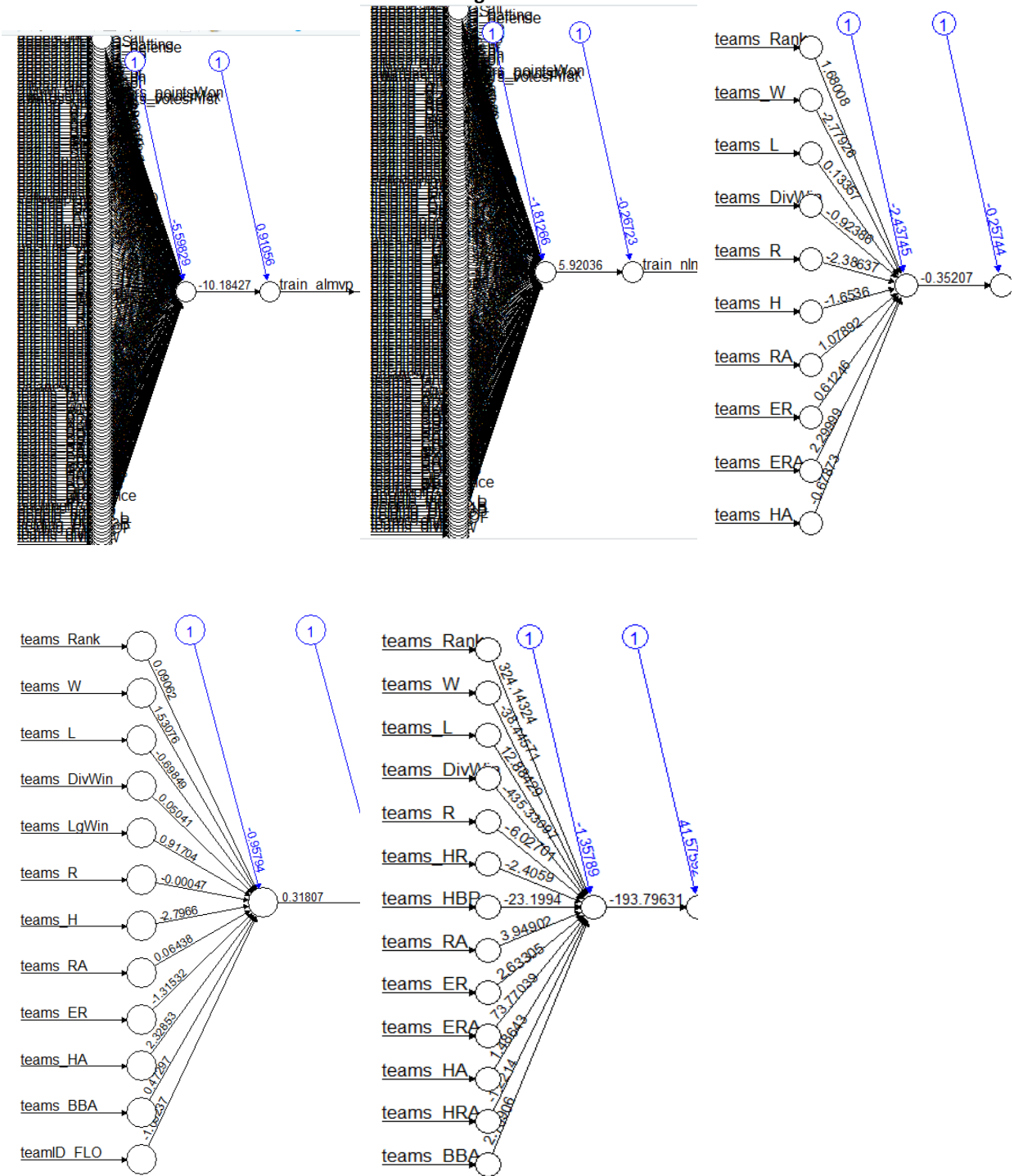
<u>What does the model predict?</u>	<u>Modeling Method</u>	<u>Validation Method</u>	<u>Train/Test or Average Model Accuracy</u>
American League MVP	Neural Network	70/30 Split	.6919 / .7059
American League MVP	Random Forest Classification	70/30 Split	1 / .9941
American League MVP	Neural Network	10-Fold, Cross-Validation	.6946
American League MVP	Random Forest Classification	10-Fold, Cross-Validation	.99
National League MVP	Neural Network	70/30 Split	.5060 / .4984
National League MVP	Random Forest Classification	70/30 Split	1 / .9923
National League MVP	Neural Network	10-Fold, Cross-Validation	.5686
National League MVP	Random Forest Classification	10-Fold, Cross-Validation	.99

American League Winner	Neural Network	70/30 Split	.5714 / .3333
American League Winner	Random Forest Classification	70/30 Split	1 / .9333
American League Winner	Neural Network	10-Fold, Cross-Validation	.5133
American League Winner	Random Forest Classification	10-Fold, Cross-Validation	.9167
National League Winner	Neural Network	70/30 Split	.9714 / .9333
National League Winner	Random Forest Classification	70/30 Split	.9857 / .9333
National League Winner	Neural Network	10-Fold, Cross-Validation	.8400
National League Winner	Random Forest Classification	10-Fold, Cross-Validation	.9133
World Series Winner	Neural Network	70/30 Split	.5142 / .4667
World Series Winner	Random Forest Classification	70/30 Split	.9857 / 1
World Series Winner	Neural Network	10-Fold, Cross-Validation	.5233
World Series Winner	Random Forest Classification	10-Fold, Cross-Validation	.9900
CY Young Award Winner	Nearest Neighbor	10-Fold, Cross-Validation	.7222
CY Young Award Winner	Support Vector Machine	10-Fold, Cross-Validation	.6945
CY Young Award Winner	CART	10-Fold, Cross-Validation	.7963
CY Young Award Winner	Random Forest Classification	10-Fold, Cross-Validation	.8148

The first thing that stands out is the huge discrepancy the neural network models and the random forest models for the MVP / Winner models. The data sets used for the neural network and random forest modes were identical respective to what they were predicting however each of models were predicting no “winning” responses regardless of which dataset was used. The reason for this is over fitting and how each model uses the data set it is given. Initially the data sets for these predictions were massively under-sampled with a positive result occurring less than five percent of the time. To correct this, a new data set was generated with all of the entries with positive results and an equal amount of randomly selected entries with a negative result. This resulted in a balanced data set. For the neural network, this meant a much more valid model than what was created from the unbalanced data set with accuracy results of around .5 for each model. However, the random forest model yields accuracy of around .99. This is an interesting result because this time I do not believe the random forest is overfitting. The training and test sets are extremely close together which means the model is not memorizing the training data. In addition, the model was run using the unbalanced dataset and again yielded accuracy results of .98. Overall, the random forest performed significantly better than the neural network in all cases. Another important thing to note is the accuracy of the Cy Young Award predictions. These models were also generated using cross-validation and a balanced data in the same way as the neural network models. However, they are much more accurate with the highest being .81. This speaks to the effectiveness of doing outside research when modeling. The neural networks were generated purely using variable selection methods learned in class where the Cy Young models were generated using a mixture of traditional variable selection and research to choose specific predictors.

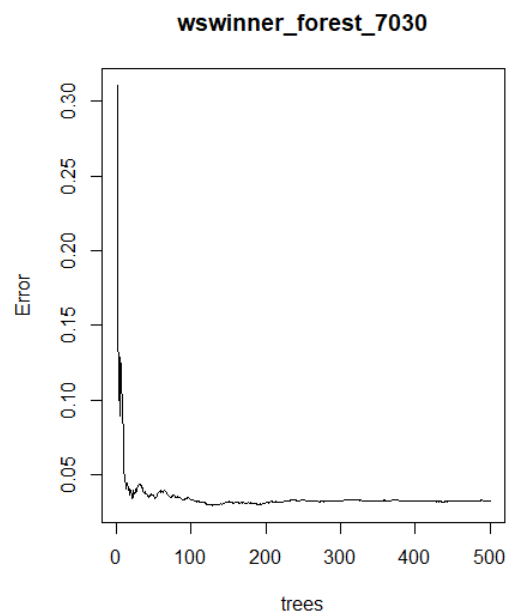
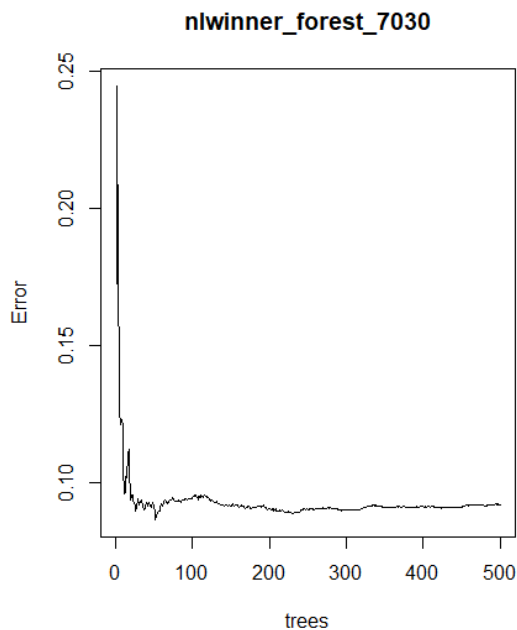
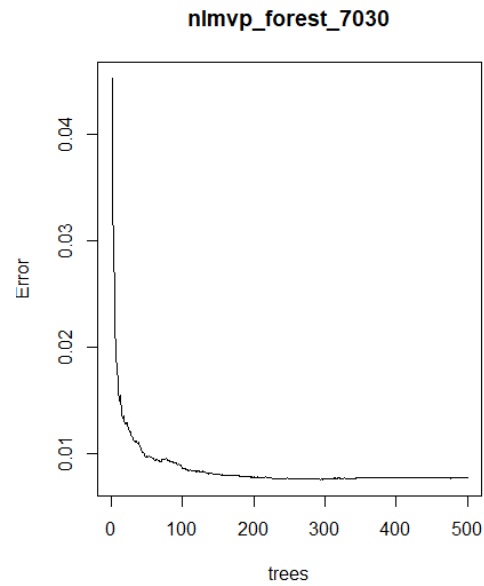
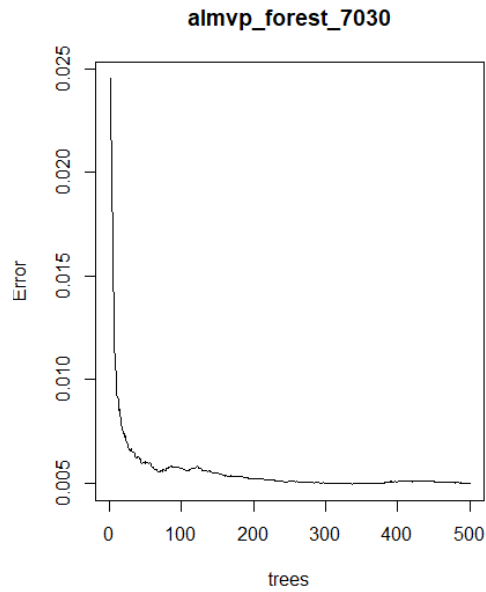
Neural Network Plots

Figure 34



Random Forest Classification Plots

Figure 35



Who will win the World Series in 2020?

The first step in predicting the 2020 world Series winner was to simulate the 2020 season to obtain the teams "talent" and then plug the result into the Bradley-Terry (BT) model. Given the 30 professional teams who talents (quantitative ability) denoted by a_1, \dots, a_{30} . The probability team i defeats team j is given by

This is a special case of a logistic model which is a common regression model where the response variable is binary (0 or 1, lose or win)

Model Setup

The date was extracted from the Group 6 MongoDB. Based upon the data I created a new variable Win that is 1 or 0 depending on if the home team wins (1) or loses (0). Then using the summarize function in the **dplyr** R package, I found the number of homes wins and away wins for each matchup of two teams. Figure 36 show the first few rows of this data frame.

Figure 36

HomeTeam <chr>	VisitingTeam <chr>	home.wins <dbl>	away.wins <dbl>
ANA	BAL	1	3
ANA	BOS	1	2
ANA	CHA	3	1
ANA	CIN	2	0
ANA	CLE	0	3
ANA	DET	1	2
6 rows			

ANA: Los Angeles Angels BAL: Baltimore Orioles

Fitting the Model

With the data frame from Figure 1 we can use the **Bradley-Terry** model using the **Btm** function in the **BradleyTerry2** package, which calculates the seasonal talents. The resulting talents are show in Figure 37. The output gives the estimates of the team talents a_1, \dots, A_{30} . For example, the team talent for the Los Angeles Dodgers is 0.90295 and for the Tampa Bay Rays is 0.49568. A modification to the model would allow for calculating the effect of a home field advantage, but since neither team will be playing in their respective hometowns a home field advantage will not be used in this model.

Figure 37

```
Bradley Terry model fit by glm.fit

Call: Btm(outcome = cbind(home.wins, away.wins), player1 = HomeTeam,
  player2 = VisitingTeam, id = "team", data = d)

Coefficients:
teamARI  teamATL  teamBAL  teamBOS  teamCHA  teamCHN  teamCIN  teamCLE  teamCOL
 0.41294  0.65638 -0.49333  0.21116 -0.11815  0.39001  0.19483  0.34739  0.10658
teamDET  teamHOU  teamKCA  teamLAN  teamMIA  teamMIL  teamMIN  teamNYA  teamNYN
-0.75924  0.80972 -0.44267  0.90295 -0.24458  0.51958  0.55234  0.66881  0.43477
teamOAK  teamPHI  teamPIT  teamSDN  teamSEA  teamSFN  teamSLN  teamTBA  teamTEX
 0.56156  0.30207  0.03835  0.06279 -0.11521  0.23816  0.54518  0.49568  0.12452
teamTOR  teamWAS
-0.17666  0.56317

Degrees of Freedom: 540 Total (i.e. Null); 511 Residual
Null Deviance: 778.8
Residual Deviance: 607.3 AIC: 1477
```

Predicting the World Series Winner

The winner of the word series is the first team to win 4 of 7 games. Using the BT model for the first game we find

that the Los Angeles Dodgers are favored to win, see Figure 38. After each game talents are recalculated and then reused to predict the winner of a particular game. Figure 38 show the game predictions and results. The model predicted that the Los Angeles Dodgers should have defeated the Tampa Bay Rays in the first four games, but Tampa Bay was able to win two.

Figure 38

Game	$P_{Dodgers}$	P_{Rays}	P Winner	Actual Winner	Results	Series
Gm 1	0.60	0.40	LA Dodgers	LA Dodgers	LAD 8, TB 3	1 - 0
Gm 2*	0.56	0.44	LA Dodgers	TB Rays	TB 6, LAD 4	1 - 1
Gm 3	0.54	0.46	LA Dodgers	LA Dodgers	LAD 6, TB 2	2 - 1
Gm 4*	0.57	0.43	LA Dodgers	TB Rays	TB 8, LAD 7	2 - 2
Gm 5	0.56	0.44	LA Dodgers	LA Dodgers	LAD 4, TB 2	3 - 2
Gm 6	0.59	0.41	LA Dodgers	LA Dodgers	LAD 3, TB 1	4 - 2

Table 12 is the final worlds series prediction confusion matrix.

Table 12 – Confusion Matrix LA Dodges

N = 6	Predicted Loses	Predicted Wins
Actual Loses	TN = 0	FP = 2
Actual Wins	FN = 0	TP = 4

Model accuracy:

$$(4 + 0)/6 = 0.67$$

$$(TP + TN)/total =$$

Misclassification Rate (Error Rate)

$$(2 + 0)/6 = 0.33$$

$$(FP + FN)/total =$$

True Positive Rate: When the was a LA Dodger win, how often did the model predict a win

$$4/4 = 1.00$$

$$TP/actual\ yes =$$

False Positive Rate: When the LA Dodges actual lose, how often did the model predict a loss.

$$2/2 = 1.00$$

$$FP/actual\ no =$$

Who will win the 2020 Cy Young Award?

To predict the 2020 Cy Young Award, I started with 35 variables as shown in figure below:

Figure 39

##	[1]	"Rk"	"Name"	"Age"	"Tm"	"Lg"	"W"	"L"	"W-L&"	"ERA"	"G"
##	[11]	"GS"	"GF"	"CG"	"SHO"	"SV"	"IP"	"H"	"R"	"ER"	"HR"
##	[21]	"BB"	"IBB"	"SO"	"HBP"	"BK"	"WF"	"BF"	"ERA+"	"FIP"	"WHIP"
##	[31]	"H9"	"HR9"	"BB9"	"SO9"	"SO/W"					

To this group of variables, I added one additional variable – CYP or the Cy Young Points using the following formula:

$$CYP = \left(\left(5 * \frac{IP}{9} \right) - ER \right) + \left(\frac{SO}{12} \right) + (CV * 2.5) + Shutouts + ((W * 6) - (L * 2)) + VB$$

Where:

- IP: Innings pitched
- ER: Earned runs
- SV: Saves
- SO: Shutouts
- W-L: Wins-Losses
- VB: Victory Bonus is a 12-point bonus awarded for leading your team to the division championship. I did not include this variable as at the time as I didn't know who the NL/AL winners were.

Predictive Models

Four models were revied to determine which was the most accurate for predicting the 2020 Cy Young Award. Those models were the

- K-Nearest Neighbor
- Support Vector Machine
- Regression Tree
- Random Forest

Before any analysis was performed the data was split the data into two data sets; a training and a testing dataset, based upon an 80/20 split. The dataset was large enough to perform cross validation using an 80/20 split of training and testing data. An arbitrary seed was set to assure reproducibility, and the training and testing sets were created using **createDataPartition** from the **caret** package.

A random forest model was built using all the prediction variables and the CYP variable as the categorical outcome. A random forest was attempted with the *train* function from the *caret* package and with the **randomForest** function from the randomForest package. The *train* function run for ~75 minutes and the function never completed. The randomForest method ran in under ten minutes, and thus the randomForest method was used to build the random forest. The final variables selected are shown in Figure 40.

Figure 40

```
## Overall
## CYA -3.684426
## GS 1026.493739
## IPouts 4104.595656
## ER 2378.556425
## SO 640.842521
## SV 323.992481
## SHO 375.788504
## W 7309.027335
## L 392.357428
```

Over-Under-Sampling

The training dataset contained 12-percent of Cy Young winners and 88-percent no winners. This indicated a severely imbalanced data set which would adversely affect the model. So, two new training datasets were created one an under-sampled dataset and the other an over-sampled dataset. So, 16 models were run and compared to determine the best and most accurate model.

Predictions

Of the 16 models the most accurate models were the over- and under- sampled regression tree models which had an accuracy of 0.83 (95% CI 0.67 - 0.94), See Figure 41.

Figure 41

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction No Yes
## No 25 1
## Yes 5 5
##
## Accuracy : 0.8333
## 95% CI : (0.6719, 0.9363)
## No Information Rate : 0.8333
## P-Value [Acc > NIR] : 0.6067
##
## Kappa : 0.5263
##
## McNemar's Test P-Value : 0.2207
##
## Sensitivity : 0.8333
## Specificity : 0.8333
## Pos Pred Value : 0.9615
## Neg Pred Value : 0.5000
## Prevalence : 0.8333
## Detection Rate : 0.6944
## Detection Prevalence : 0.7222
## Balanced Accuracy : 0.8333
##
## 'Positive' Class : No
##
```

Predicting the Cy Young Award winner using python:

Just like the predictive model in R, it started with 35 variables but ended up using only 7 variables for prediction based on the CYP formula which was discussed in the previous section.

Predictive Models

Seven models were used to determine the most accurate model for predicting the 2020 Cy Young Award:

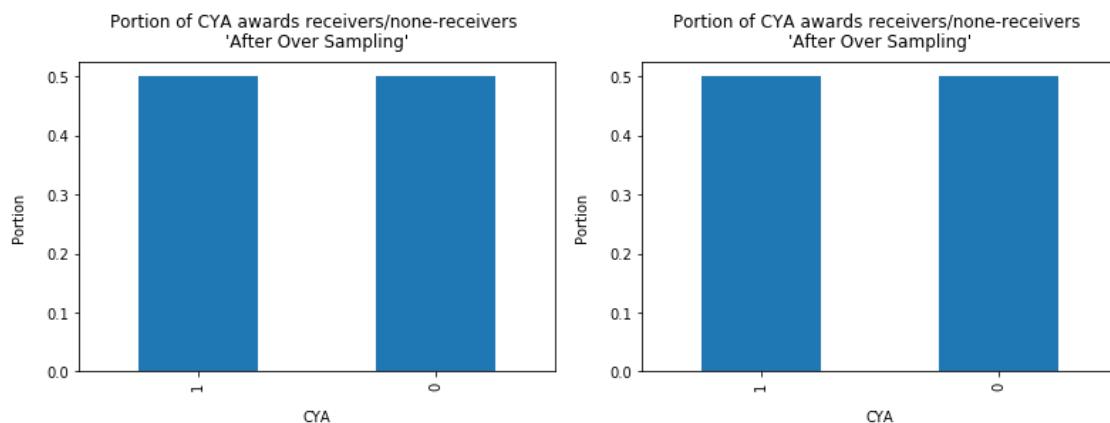
- Logistic Regression
- Naïve Bayes
- Decision Tree
- Neural Network
- Support Vector Machine
- Regression Tree
- AdaBoost Classifier

The data was large enough to perform cross validation, thus, it was split into training and test datasets with the 80 to 20 ratios. All these was performed using model selection module in scikit-learn library in python.

Over-Under-Sampling

The training dataset contained 15-percent of Cy Young winners and 85-percent none-winners. This indicated a severely imbalanced data set which would adversely affect the model. Two datasets were created: one with over sampled data and the other one with the under sampled data and all the seven prediction models were ran under the two different datasets but to have a tidy and concise document only the over-sampled results were attached to the document which had more accurate results in general.

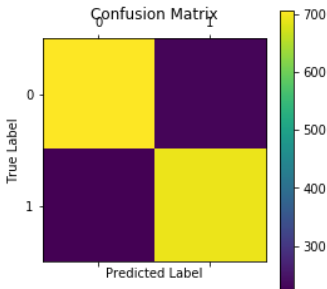
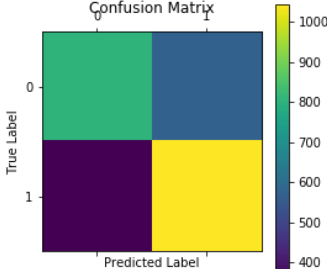
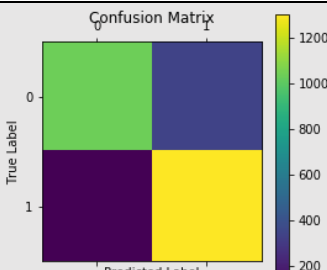
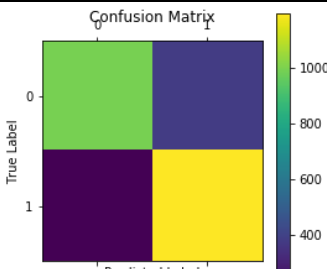
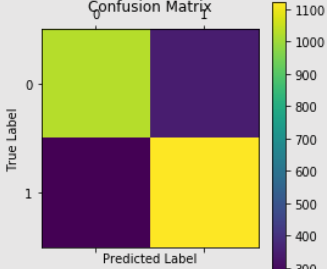
Figure 42

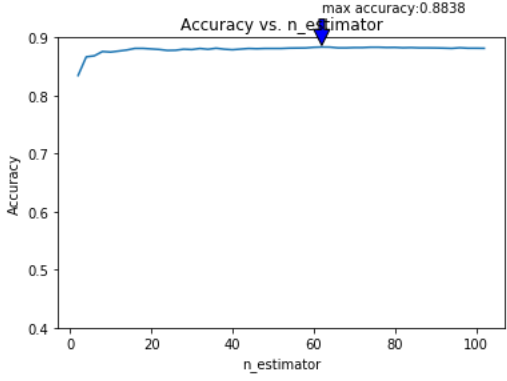
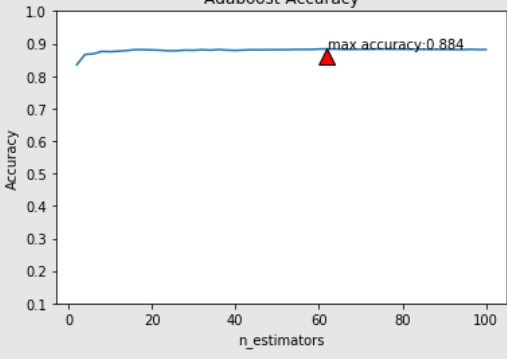


Predictions

As it is stated in table below the most accurate models are the Random Forest and AdaBoost classifier models with 88% percent of accuracy in each.

Table 13 – Summary

Model Name	Accuracy	Precision		Recall		f1-Score		Confusion Matrix Plot																								
		0	1	0	1	0	1																									
Logistic Regression	0.76	.76	.75	.75	.76	.76	.75																									
Naïve Bayes	0.67	.69	.65	.58	.74	.63	.69																									
Decision Tree	0.84 <div><table><tr><th></th><th>variable</th><th>importance</th></tr><tr><td>0</td><td>GS</td><td>0.038326</td></tr><tr><td>1</td><td>IP</td><td>0.081972</td></tr><tr><td>2</td><td>ER</td><td>0.170722</td></tr><tr><td>3</td><td>SO</td><td>0.159170</td></tr><tr><td>4</td><td>SV</td><td>0.098595</td></tr><tr><td>5</td><td>W</td><td>0.386462</td></tr><tr><td>6</td><td>L</td><td>0.064752</td></tr></table></div>		variable	importance	0	GS	0.038326	1	IP	0.081972	2	ER	0.170722	3	SO	0.159170	4	SV	0.098595	5	W	0.386462	6	L	0.064752	.91	.80	.76	.93	.83	.86	
	variable	importance																														
0	GS	0.038326																														
1	IP	0.081972																														
2	ER	0.170722																														
3	SO	0.159170																														
4	SV	0.098595																														
5	W	0.386462																														
6	L	0.064752																														
Neural Network	0.79	.83	.76	.72	.85	.77	.80																									
Support Vector Machine	0.77	.79	.76	.75	.80	.77	.78																									

Random Forest	0.883	
AdaBoost Classifier	0.884	

Predicting the MVP (Most Valuable Person) Award winner using python:

The Most Valuable Player Award is given annually to one player in each league. The award began in 1911 as the Chalmers Award, honoring the "most important and useful player to the club and to the league". This award was discontinued in 1914. From 1922 to 1928 in the American League and from 1924 to 1929 in the National League, an MVP award was given to "the baseball player who is of the greatest all-around service to his club". Prior winners were not eligible to win the MVP award again during this time. The current incarnation of the MVP award was established in 1931. The Baseball Writers' Association of America (BBWAA) votes on the MVP award at the conclusion of each season before the postseason starts.

The dataset used for modeling was the batting dataset which has the variables listed below:

- H: Hits
- BB: Walks by batters
- HBP: Batters hit by pitch
- AB: At Bats
- SF: Sacrifice flies
- H, 2B, 3B: Hits, Doubles, Triples
- HR: Homeruns
- RBI: Runs batted in
- SB: Stolen Bases;

To perform the modeling some variables had to be generated according to the baseball reference website:

- BA: Hits/At Bats (H/AB); according to baseball reference website;
- OBP: $(H + BB + HBP) / (AB + BB + HBP + SF)$; according to baseball reference website;
- SLG: $(H - X2B - X3B - HR + 2 * X2B + 3 * X3B + 4 * HR) / AB$; according to Max Marchi and Jim Albert article;

Also, MVP variable which is an indicator that a certain player has won the award or not was added to the dataset based on the “awards” dataset. Also, several unused variables were dropped that resulted the final table to be like this:

Figure 43

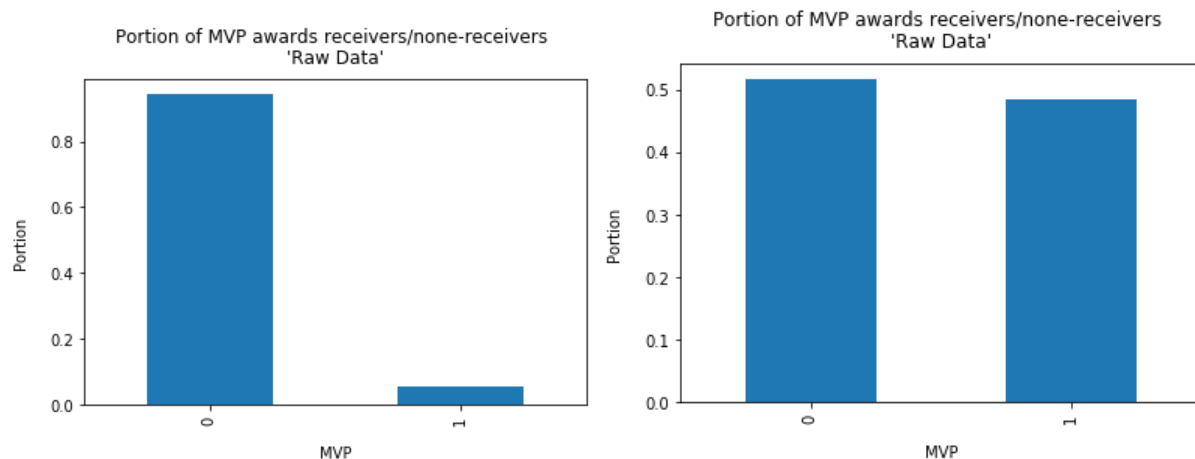
	playerID	yearID	BB	HR	RBI	SB	MVP	BA	SLG	OBP
36914	aaronha01	1954	28	13	69.0	2.0	0	0.279915	0.446581	0.322068
36915	aberal01	1954	2	0	3.0	0.0	0	0.128205	0.128205	0.166667
36916	abramca01	1954	10	0	2.0	0.0	1	0.142857	0.214286	0.307692
36917	abramca01	1954	72	6	25.0	1.0	1	0.293144	0.420804	0.400000
36918	adamsbo03	1954	55	3	23.0	2.0	0	0.269231	0.387179	0.362222

Just like the Cy Young award dataset, the MVP data was large enough to perform cross validation, thus, it was split into training and test datasets with the 80 to 20 ratios using the same techniques.

Handling the imbalanced dataset:

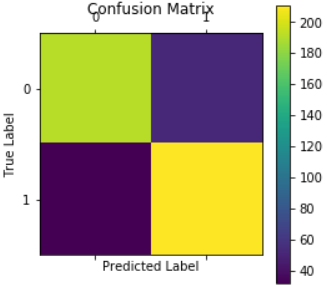
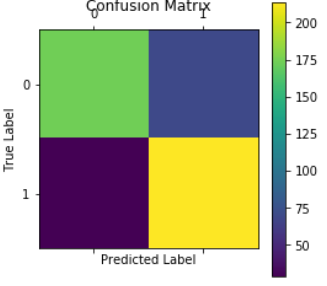
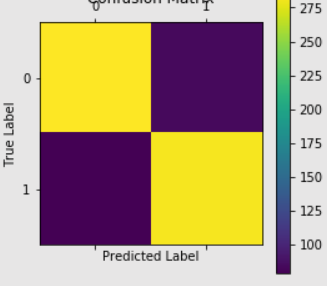
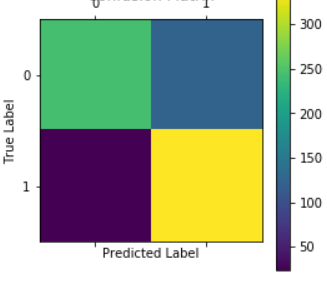
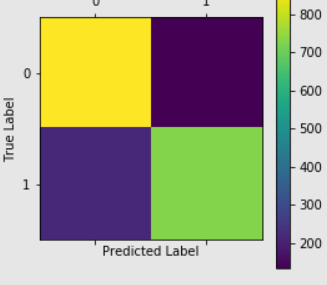
The training dataset contained 5-percent of MVP award winners and 95-percent none-winners. This indicated a severely imbalanced data set which would adversely affect the model. This problem handled by simply filtering the dataset based on variables such as: Year ID, BA, SLG, OBP, and HR which resulted in a perfectly balanced dataset with the ratio of 51 to 48.

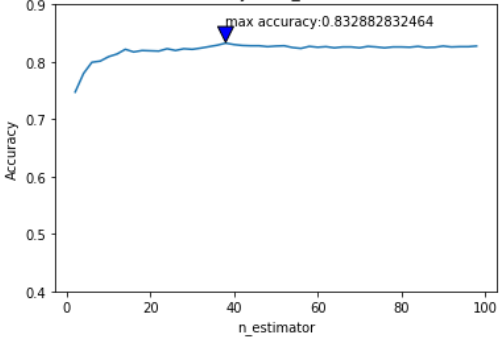
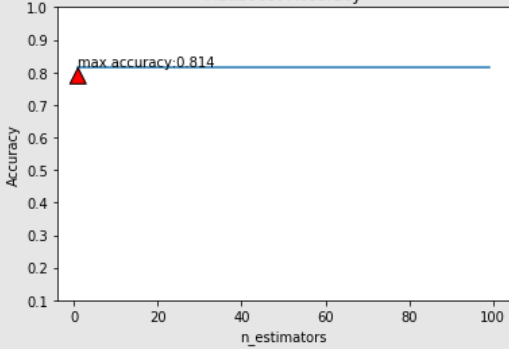
Figure 44



Predictions

In general MVP awards prediction models performed better than Cy Young models. The highest accuracy achieved with Logistic Regression and Random Forest models with the accuracy of 83% for each.

Model Name	Accuracy	Precision		Recall		f1-Score		Confusion Matrix Plot																					
		0	1	0	1	0	1																						
Logistic Regression	0.83	.86	.80	.79	.87	.82	.83																						
Naïve Bayes	0.80	.86	.75	.71	.88	.78	.81																						
Decision Tree	0.78 <div><table><thead><tr><th></th><th>variable</th><th>importance</th></tr></thead><tbody><tr><td>0</td><td>BA</td><td>0.131373</td></tr><tr><td>1</td><td>OBP</td><td>0.065986</td></tr><tr><td>2</td><td>SLG</td><td>0.111126</td></tr><tr><td>3</td><td>HR</td><td>0.057238</td></tr><tr><td>4</td><td>RBI</td><td>0.578903</td></tr><tr><td>5</td><td>SB</td><td>0.055374</td></tr></tbody></table></div>		variable	importance	0	BA	0.131373	1	OBP	0.065986	2	SLG	0.111126	3	HR	0.057238	4	RBI	0.578903	5	SB	0.055374	.78	.77	.77	.78	.78	.78	
	variable	importance																											
0	BA	0.131373																											
1	OBP	0.065986																											
2	SLG	0.111126																											
3	HR	0.057238																											
4	RBI	0.578903																											
5	SB	0.055374																											
Neural Network	0.80	.91	.73	.66	.93	.77	.82																						
Support Vector Machine	0.82	.80	.84	.87	.77	.83	.81																						

Random Forest	0.83	<p>Accuracy vs. n_estimator</p>  <p>max accuracy:0.832882832464</p>
AdaBoost Classifier	0.81	<p>Adaboost Accuracy</p>  <p>max accuracy:0.814</p>