Udacity Machine Learning Engineer Nanodegree

Capstone Project Report: MLB Wins Prediction

By John K. Hancock

Username: jkhancock@gmail.com

August 25, 2018

I. Definition

Project Overview

Major League Baseball ("MLB") is an American professional sports league where teams invest billions of dollars every year with the goal of winning a championship. MLB has also been at the vanguard of using data to make informed decisions regarding evaluating players, how much to spend on payrolls, and how to win championships. Organizations like MIT's Sloan Conference and the Society for American Baseball Research sponsor and publish articles promoting the use of statistical analysis in sports.

One of the more studied aspects is predicting wins. Wins are important to a team since in order to win a championship, teams have to make the playoffs, and in order to do that they have to win more games than other teams. The International Journal of Computer Science in Sport published an article by C. Soto Valera, "Predicting Win-Loss Outcomes in MLB Regular Season Games – A Comparative Study Using Data Mining Methods."[1]

The problem domain of this project is to understand what features are important to an MLB team that will lead to wins. This project does that by building supervised machine learning regression models that can predict wins based on historical data, learn the most important features for winning, and identify players that possess these critical features.

This project uses datasets compiled by the organization, FanGraphs.com (www.fangraphs.com). Fangraphs.com is website operated by FanGraphs, Inc. Fangraphs compiles historical statistical data for the entire history of Major League Baseball. In addition, it creates and records advanced baseball metrics outside of the established statistics. FanGraphs is well established as a chronicler and compiler of baseball statistics. It has partnership deals with ESPN and SB

[1] International Journal of Computer Science in Sport, Vol 15, Issue 2, 2016 (link: https://content.sciendo.com/view/journals/ijcss/15/2/article-p91.xml)

Nation. For more information, please see their Wikipedia entry, https://en.wikipedia.org/wiki/Fangraphs.

For the purpose of this academic exercise, I purchased a year's membership to the site and was able to download the datasets needed for this project. Datasets for this project are located in the folder: **CAPSTONE FINAL PROJECT\00_Data_Collection_and_Wrangling\Data\Revised Statistics**

Problem Statement

Imagine that you are a new general manager ("GM") of a major league baseball team, and you have been hired with the specific task of getting your new team into the playoffs. Now, as GM you know that to make the playoffs the team needs to win more games than the other teams.

But how does the team actually accomplish this? What does the team need to do in order to win? The GM has tasked his Data Scientist ("DS") to answer this question.

The DS reports that there is a wealth of statistical information about teams, players in MLB going back decades. This statistical information can be used to build regression models that can predict wins. Additionally, these models can tell us which team statistics carry the most weight in for these predictions. With that information, the team can then identify players that are best at providing the team with these features.

For example, we build a regression model that creates a function that maps statistical data (independent features) to wins (the dependent variable). From this function, we can see the weights given to each of these independent features. For instance, the model may say that a player's On-Base Percentage ("OBP" – the number of times a hitter gets on base divided by the number of times he goes to the plate) is more important to wins than Stolen Bases. Using that information, the GM will look target players that have high OBP over those players with a high number of SBs.

Metrics

Given that the independent variables used to make the predictions on Wins will mostly be numerically, continuous data as well as the predictions will also be continuous, the project will use a supervised machine learning and deep neural network linear regression models. The most useful metrics to evaluate these regression models are with Mean Square Error ("MSE"), Mean Absolute Error ("MAE"), R-square, and Adjusted R-square.

The project will use MSE over MAE as the metric. "[MSE] represents the sample standard deviation of the differences between predicted values and observed values (called residuals) …. MAE is the average of the absolute difference between the predicted values and observed value. The MAE is a linear score which means that all the individual differences are weighted equally in the average."[2] MSE punishes large errors while MAE is steadier. MAE is robust to outliers. The justification for using MSE over MAE is that we want to punish and minimize errors as much as possible. The faster and better the models get at minimizing error, the better the models are at learning what variables are needed for wins.

The other metrics that will be used to evaluate the models are R-squared and Adjusted R-squared. Both metrics "explain how well your selected independent variable(s) explain the variability in your dependent variable(s)"[3]. It is the proportion of the variance in the dependent variable (Wins) that is predictable from the independent variables (features). In sum, it provides a statistic that evaluates how well the model fits the data. "The adjusted R-squared increases only if the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected by chance. "[4]

These metric, MSE, R-Square, and Adjusted R-Square are being used to evaluate the current model and make future comparisons if the model is changed in the future. For example, if more, or even all variables are fed into the model, then these metrics will give us a baseline comparison.

II. Analysis

Data Exploration

Data Source: FanGraphs.com

Fangraphs.com is website operated by FanGraphs, Inc. Fangraphs compiles historical statistical data for the entire history of Major League Baseball. In addition, it creates and records advanced baseball metrics outside of the established statistics. FanGraphs is well established as a chronicler and compiler of baseball statistics. It has partnership deals with ESPN and SB Nation. (Link to website: https://www.fangraphs.com/) (Wikipedia: https://en.wikipedia.org/wiki/Fangraphs). For the purpose of this academic exercise, I purchased a year's membership to the site and was able to download the needed datasets.

[2] Alvira Swalin, https://www.kdnuggets.com/2018/04/right-metric-evaluating-machine-learning-models-1.html
[3] Id.
[4] The Minitab Blog, "http://blog.minitab.com/blog/adventures-in-statistics-2/multiple-regession-analysis-use-adjusted-r-squared-and-predicted-r-squared-to-include-the-correct-number-of-variables"

Data Collection

Fangraphs has an interface for members wherein you are able to download individual and team statistics covering a single season or multiple years. You are able to download and export the data to a csv file. FanGraphs compiles a massive amount of statistical data from standard metrics, Earned Run Average, Home Runs, Runs Batted In, Error, Double Play to more advanced measures such as Weighted Runs Created, Ultimate Base Running, and Ultimate Zone Rating. On the site, you can generate a custom report for these statistics Fielding, Hitting, and Pitching which is exactly how I collected the data.

I created three custom reports for Fielding, Hitting, and Pitching statistics for the past 20 years of MLB. I used the cutoff year of 1998 since that was the last year that MLB expanded by adding two additional teams. Data from any other previous era would not be consistent with this current era and may affect the model. In order for the model to make accurate predictions about Wins, it must reflect the current state of the game.

Each custom report contains every statistic compiled by FanGraphs on the site. I then downloaded a report for every year from 1998 to 2017 for Fielding, Hitting, and Pitching. There are 72 different statistical categories for each year of Fielding statistics, 304 different statistical categories for each year of Hitting statistics, and 320 different statistical categories for each year of Pitching statistics. The downloaded datasets may be found in the folder path: 00_Data Collection and Wrangling\Data\Revised Statistics.

Data Inspection

In order to combine the datasets into one overall dataset, I did the following:

First, I copied the datasets for fielding, hitting, and pitching into their own folders by Years. See the directory: 00_Data Collection and Wrangling\Data\Revised Statistics\Years\

Next, after a cursory inspection of the datasets showed that the names for some of the statistics were the same for all three types, fielding, hitting, and pitching. For example, the strikeouts metric, "SO", is the same for both hitting and pitching, but it has two very different meanings. In order to distinguish these features, I had to prepend an identifier. For fielding, I prepended, "FIELD". For pitching, I used "PITCH", and for hitting, I used "HIT_".

Finally, all datasets were combined into one dataset, CUMULATIVE_MLB_STATS_1998_to_2017.csv, and written out to a csv file of the same name.

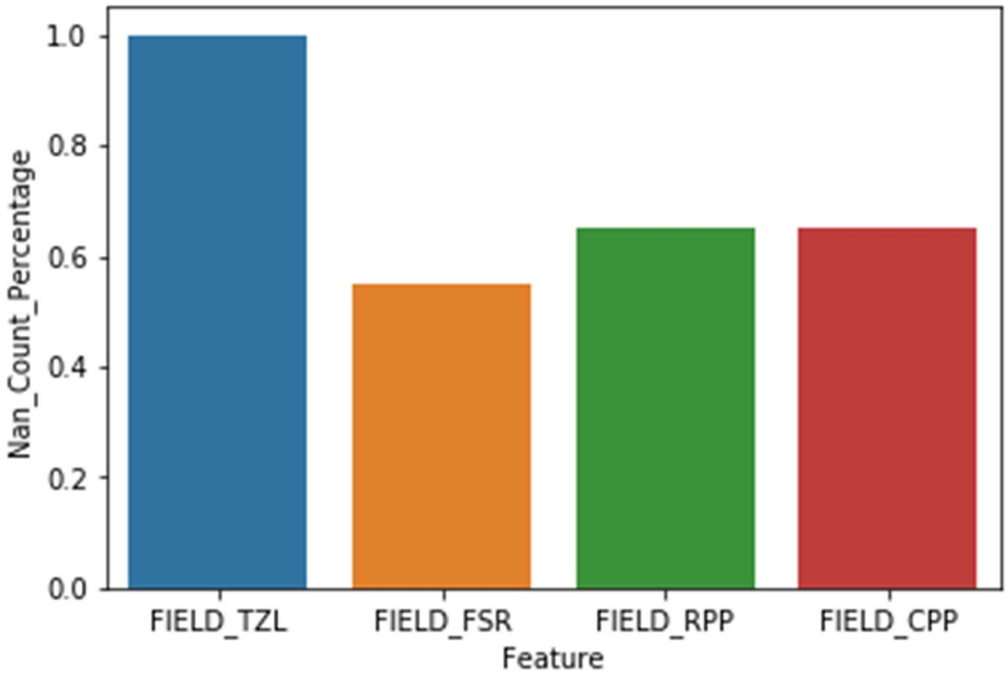
Data Wrangling and Cleaning

The shape of the combined datasets shows 600 observations and 697 features. The head function shows an additional column, "Unnamed". When looking at the column values, I noticed that there were duplicates as some of the column headings have a ".1" suffix which indicates a duplicate. The utility function, removeDuplicateFeatures (imported from the utility file: “capstoneutils.py”) removes all of the duplicate column names from the dataframe.

Even after duplicates are removed, there are still 634 features in this dataset. This is due to the fact that Fangraphs maintains a huge library of metrics for every team and player. Some of these features, e.g. Wins Above Replacement ("WAR") are more of a measure for an individual player, not a team. A complete glossary of each statistic is available on the Fangraphs site at this link: https://www.fangraphs.com/library/

For this project, I chose 137 features which are more team oriented features. The complete list that I chose is listed in the file, “Selected Features.txt” located in the folder: 00_Data Collection.

Next, I want to remove any features that have NaN or missing values. There are 138 features in this dataset and NaN or null values may affect the model's ability to accurately predict wins. First, we need to know the percentage of missing data for each feature. The graphic below shows those features which have greater than 20% NaN values. This makes intuitive sense that the Fielding features would have the higher NaN counts since it's difficult to measure fielding statistics unless it is being observed in real time. Pitching and hitting statistics can be gleaned from the box score.



With all the NaN and null values removed, I wrote out the final dataset and the number of features were reduced to 67.

Reflections on Data Exploration

I made some major choices about the data. First, I culled the data by removing features that are more player oriented than team oriented. Next, I cleaned up data that had the '%' sign. Finally, I removed all features that had NaN values. Some, or even all, of the choices that I made may affect the model's performance. However, given the steps laid out in this section, it should be fairly simple to reverse or add another step which will lead to a different final data set. Additionally, I saved off the datasets after every step. Each of these save datasets could be substituted in building the model.

Data Exploration

The Pythagorean Theorem of Baseball

After printing the summary statistics for the Fielding, Hitting, and Pitching statistics, I did not find anything unusual in the data worth mentioning, but I did want to explore the Pythagorean Theorem of Baseball which is a creation of one of baseball's statistical pioneers, Bill James. James related the number of runs a team has scored and surrendered to its actual winning percentage, based on the idea that runs scored compared to runs allowed is a better indicator of a team's (future) performance than a team's actual winning percentage.[5] The formula is:

1 / (1 + (Runs Allowed / Runs Scored)^2)

The minimum number of runs allowed by a team over the past 20 years is 525 and that team's win total was 100. That same team scored 891 runs for a run differential of 366. According to the Pythagorean Theorem of Baseball, the team's win total should have been 120.25.

The maximum number of runs scored by a team over the past 20 years is 1009 and that team's win total was 103. That same team allowed 661 runs for a run differential of 348. According to the Pythagorean Theorem of Baseball, the team's win total should have been 113.35.

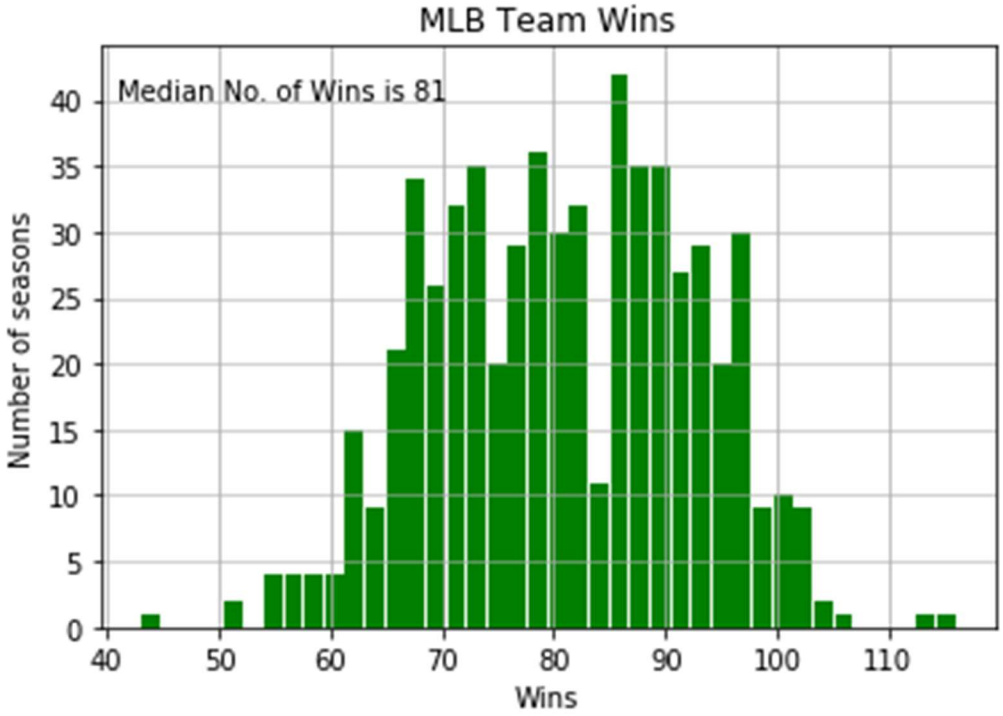
The results show that even with this traditional wins prediction heuristic there is variance between what the formula predicts and the actual results.

A Look at MLB Wins

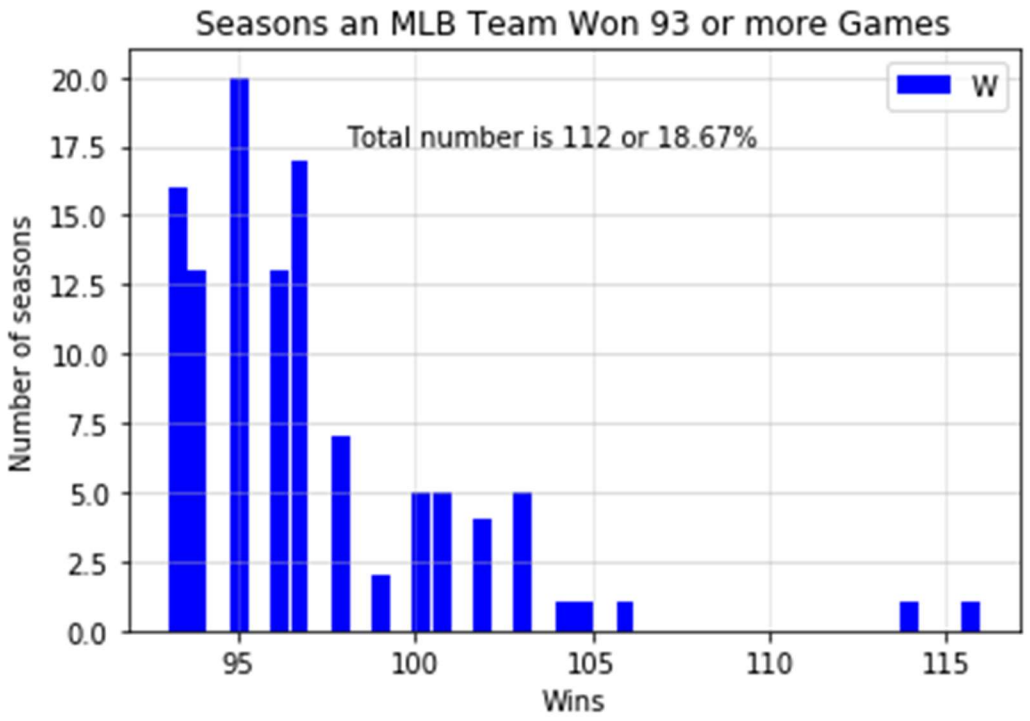
The DS chose to look at data for the past 20 years of MLB which is the last year that the league expanded. After importing the final dataset from the collection and data wrangling section of the project, there are a total of 600 observations and 67 features which are all continuous

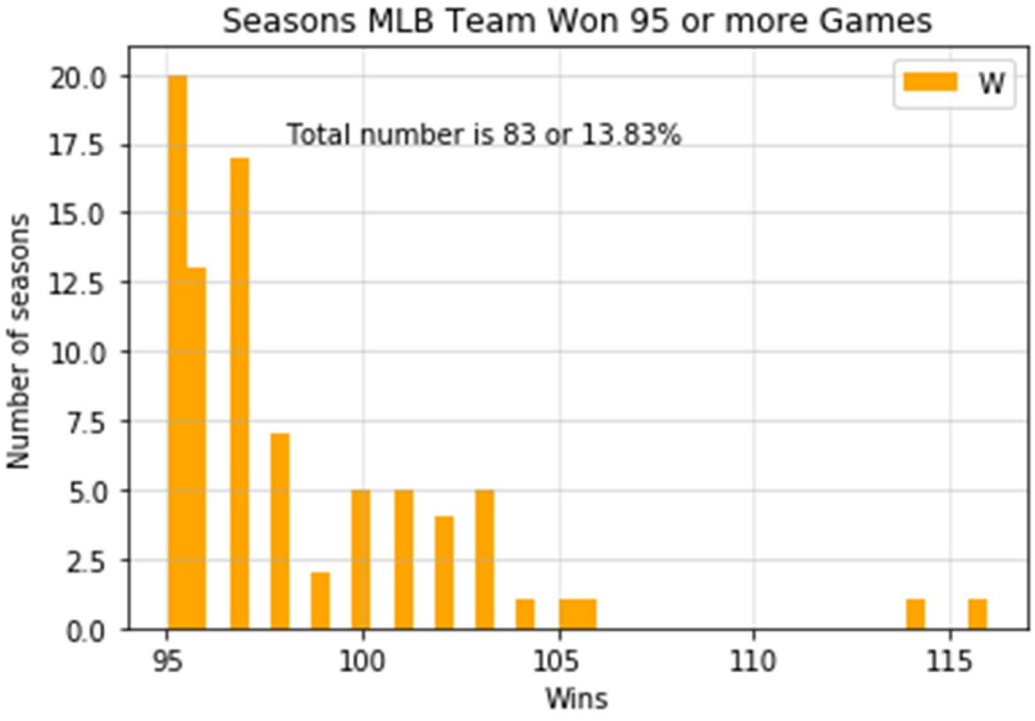
[5] Baseball Reference.com https://www.baseball-reference.com/bullpen/Pythagorean_Theorem_of_Baseball

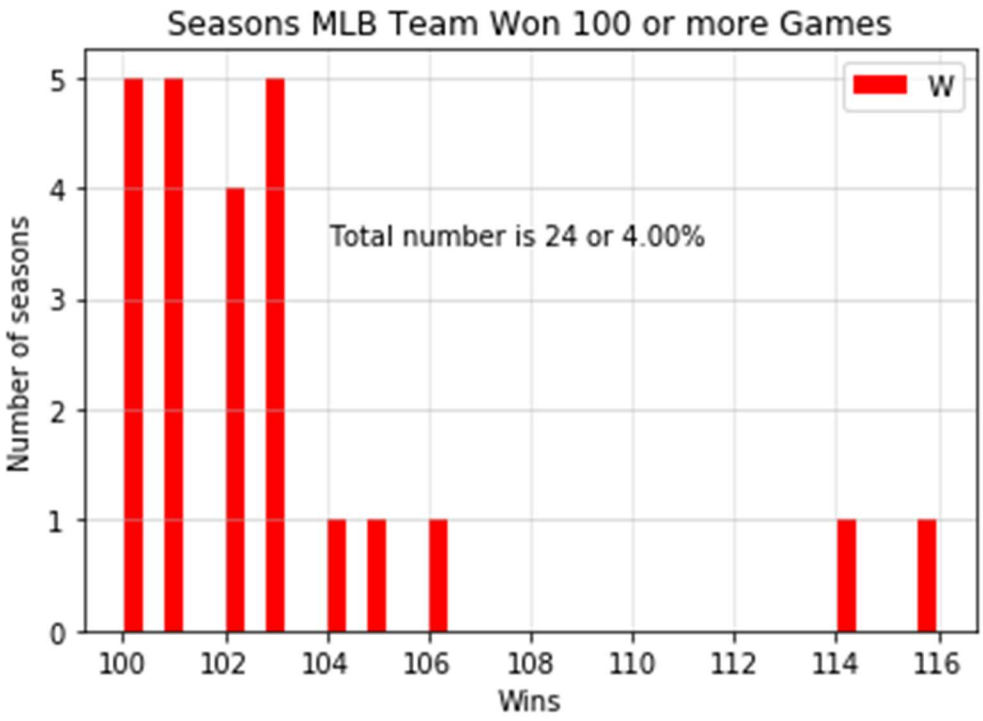
data., dtypes: float64(27), int64(40), and there are no categorical features. As previously stated, the objective of this project is to build a model that can predict the number of wins and lists the most important features that are important to winning. Over the past 20 years, the average and median number of wins is 81 games, and the standard deviation is 11.52.



The best chance for the team to make the playoffs is to win at least on standard deviation above the mean or roughly 93 games. The probability of winning 93 games or more is 18.67%. The probability of winning more than 95 games is 14%, and the probability of winning more than 100 games is just 4%.


Seasons an MLB Team Won 93 or more Games
W
Total number is 112 or 18.67%
Number of seasons
Wins
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
95
100
105
110
115


Seasons MLB Team Won 95 or more Games
W
Total number is 83 or 13.83%
Number of seasons
Wins
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
95
100
105
110
115



Algorithms and Techniques

Benchmark

The models will be evaluated by measuring what they predict to the actual wins. In section, 2.0 of the iPython notebook, the dependent feature, “W”, from the independent features and placing it into its own dataframe, “Wins”. The second dependent feature, “L” (or Losses) was also dropped. The independent variables were placed into their dataframe called, “features”. The models will use the features dataframe to predict wins which will then be compared to the actual wins in the Wins dataframe.

III. Methodology

The goal of this project is to build models that can predict wins. Given that the data is comprised of 67 independent, continuous numerical features, the project will build linear models to make predictions. Prior to building the models, the data will have to be pre-processed so that certain observations cannot distort the outcome of the model. Additionally, the models will tell us which features have the highest weights in regards to the prediction. Finally, the project will use these features to build hierarchical clusters of the players for the team to target. These players will possess those features that the team needs to create more wins.

The step-by-step process Is as follows:

a. Pre-Processing
    1. Remove outliers
    2. Detect and Correct Skewness of the features
    3. Scale the data
    4. Select the Best Features
b. Modeling
    1. Build a linear regression model using LASSO Regression
    2. Build a linear regression model using a Deep Neural Network
c. Hierarchical cluster analysis to target players

a. Pre-Processing

1. Remove Outliers

The first step was to identify any outliers in the data. Outliers are observations that are abnormally distant from the other observations. To identify the outliers, I used the "Tukey's Method for identifying outliers: An outlier step is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal." (citation: Udacity MLND course on Unsupervised Learning. Customer Segments project.) (Turkey's Method: http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/).

The function, detect_outliers, in the utility file, helpers.py, was applied to features dataset. The function returned outliers for each feature. Some features had more outliers than others. For example, the feature, FIELD_DP, has 13 observations that are outliers. The function populated a Collection object called “cnt”. The collection lists observations in the dataset and the number of outliers that it has.

[**Key Decision**] Observations that had more than 4 outliers were removed from the features dataset. The justification was that these features may bias the model. For example, observation 59 is an outlier across 11 different pitching features.

```
Counter({59: 11,
         30: 8,
         570: 7,
         0: 6,
         573: 5,
         598: 5,
```

My decision to limit the number of outliers to remove is arbitrary. In fact, one could make an argument to include all outliers as well since it is legitimate data that wasn't recorded in error.

2. Detect and Correct Skewness of the Features

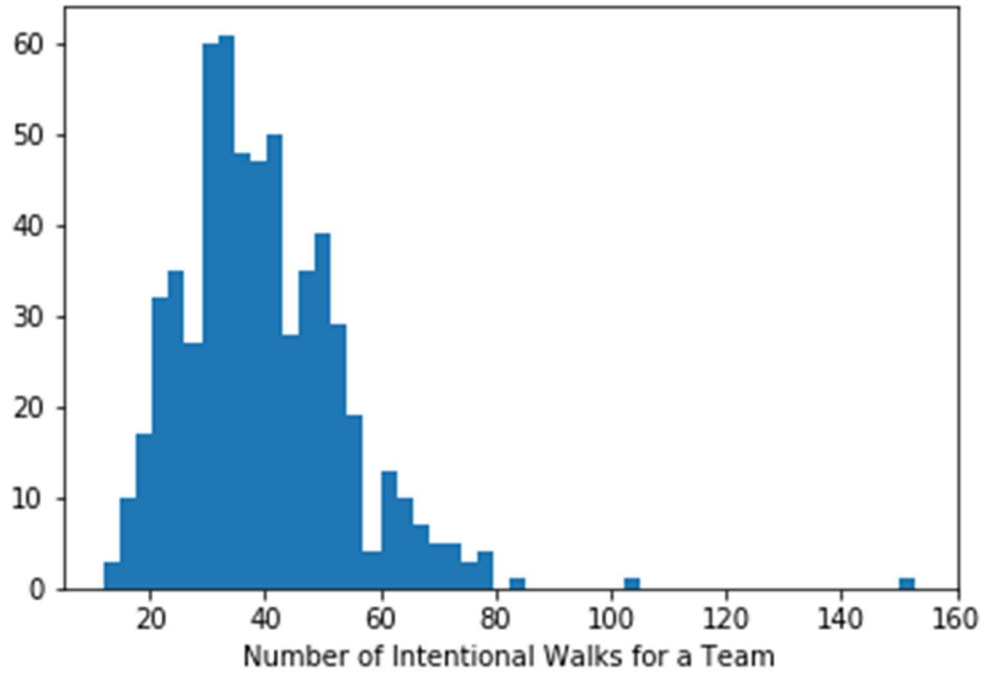
Skewness is a measure of symmetry, or lack thereof, in a dataset while Kurtosis is a measure of how large the tails are compared to a normal distribution. If both skew and kurtosis are 0, then we can safely say that the data is normally distributed. "The rule of thumb seems to be that:

> If the skewness is between -0.5 and 0.5, the data are fairly symmetrical
> If the skewness is between -1 and – 0.5 or between 0.5 and 1, the data are moderately skewed
> If the skewness is less than -1 or greater than 1, the data are highly skewed"[6]

In section 4.0, I detected three features where the skew value were either less than -1 or greater than 1:

{'FIELD_PB': [1.4430653832375777], 'OFF_IBB': [1.433840489666504], 'PITCH_CG': [1.1213488048517393]}

A plot of one of these features, OFF_IBB (Number of Intentional Walks for a Team), is in the chart below:



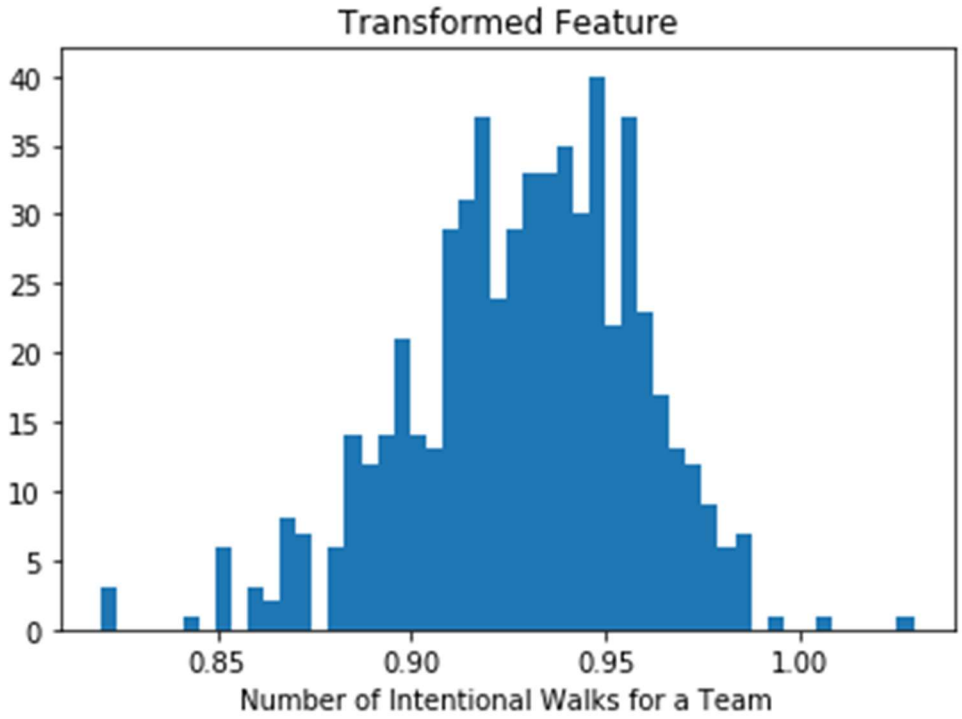
[6] https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics

Clearly, this feature is not normally distributed which may impact the model.

[**Key decision**] In section 4.1, I transformed the skewed features of the dataset identified above by taking the log of the feature + 1.

For example,

Below are the results of the dataset after it has been transformed:



In the next code block, I printed out the skew and kurtosis values of each feature.

3. Scale the Data

The final pre-processing step was scaling the features to ensure that they have the properties of a standard normal distribution. "Standardizing the features so that they are centered around 0 with a standard deviation of 1 is not only important, if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms. Intuitively, we can think of gradient descent as a prominent example (an optimization algorithm often used in logistic regression, SVMs, perceptrons, neural networks etc.); with features being

on different scales, certain weights may update faster than others since the feature values play a role in the weight updates."[7]

[**key decision**] For this project, I applied sklearn's MinMaxScaler. "In this approach, the data is scaled to a fixed range - usually 0 to 1. The cost of having this bounded range - in contrast to standardization - is that we will end up with smaller standard deviations, which can suppress the effect of outliers."[8]

In sum, I scaled the numerical features of this dataset because of the magnitude in units and ranges. For example, the numerical feature for a team's number of hits is going to be of greater magnitude than it's weighted on base average simply because one is a whole number and the other is a percentage. The model needs to see each one equally on a scale.

4. Select Best Features

At this point, I still have 65 features, and I needed a way to reduce them before I fed the data into the models. I imported sklearn's SelectKBest library which scores the best features in the dataset. I used the score_func parameter, f_regression.

[**Key Decision**] I culled out the features whose score was greater than 50. This left 40 features to feed into the model. This was a somewhat arbitrary decision that I made in an effort to find the best combination of features.

b. Modeling

Prior to building the models, the features and Wins dataframes were separated into test, train splits. The models will be designed and fitted against the training sets and then evaluated against the test set. Datasets are split in this regard so that the models don't over-learn on the same dataset.

Model One: LASSO Regression

For this project, I chose to do a Supervised Machine Learning linear regression model. Linear Regression works best for this project since we are taking a set of inputs (features) to make a prediction on the outputs (Wins) not categorization. "The linear equation assigns one scale factor to each input …called a coefficient… Learning a linear regression model means estimating

[7] About Feature Scaling and Normalization– and the effect of standardization for machine learning algorithms by Sebastian Raschka link: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-min-max-scaling

[8] Id

the values of the coefficients used in the representation with the data that we have available."[9] For this project we want to know the weights that the model assigns to each input or feature in order to give us insight on selecting players.

Part of the learning process for the model entails understanding the weight of the coefficient for each feature. One technique to do this is L1 Regularization. "[L1] both minimizes the sum of the squared error of the model on the training data (using ordinary least squares) but also reduces the complexity of the model (like the number or absolute size of the sum of all coefficients in the model)."[10]

Least Absolute Selection and Shrinkage Operator or "LASSO" regression model was the first choice for this project given the high number of features in the dataset and the low number of observations. "LASSO is a regression analysis method that performs both variable selection and L1 regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces."[11] LASSO causes regression coefficients for some variables to shrink towards zero, thereby creating a simpler model that does not include those coefficients.

In the Lasso regression model imported from sklearn, uses Regularization to penalize model complexity. The lambda parameter penalizes the coefficients so that the model does not over fit. In sum, due to the high number of features, the model will reduce the coefficients of most of them to zero in order to create a simpler, more accurate prediction model. (**See section 6.0 of the Python Notebook for the implementation of this model.**)

Refinement

[**Key Decision**] The key parameter of the model were the alphas that were used. As stated earlier, Regularization is used to shrink large coefficients of the features to reduce over fitting. The alpha terms are constants that the L1 regularization are multiplied by. A range of alphas values (from 0.005 to 5000000000) were used in this model.

This alpha parameter along with the LASSO and the number of folds parameters were fed into the GridSearchCV to create the classifier, "clf". which constructs different combinations of the parameters to find the best model. I tried different ranges of alphas and different numbers of

[9] "Linear Regression for Machine Learning" by Jason Brownlee Link: https://machinelearningmastery.com/linear-regression-for-machine-learning/
[10] Id
[11] https://en.wikipedia.org/wiki/Lasso_(statistics)

folds, and the present combination of parameters gave me the best Mean Square Error and R-square scores.

Model Two: Linear Regression with Deep Neural Network

[**Key Decision**] The next model that I used to predict wins was a Deep Learning Neural Network. Deep Learning is an area of machine learning that mimics animal brains by progressively improving on its ability to learn a task through the use of examples. Input is passed into a network of layers called neurons. Each input is assigned which yields an output. "The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output."[12] (). "[E]xposed to enough of the right data, deep learning is able to establish correlations between present events and future events. It can run regression between the past and the future."[13]

For this model, I built 4 layers with one final output layer. In the first four layers, I used a relu function as the activation function.

Refinement

[**Key Decision**] For a classification model, a Deep Neural Network is used to classify the data as belonging to one class or another. However, for this project, we are trying to predict a number, Wins, not whether something belongs to a class or not. For this model, the deep neural network was constructed with one input layer, 4 hidden layers, and one output layer. All of the layers used the ReLu function, with the exception of the output layer. I used more layers, different activation functions, and different units, and this current architecture worked the best.

c. Hierarchical cluster analysis to target players

For this project, we want hierarchical clusters of the players since the goal is to target players whose statistics are a better match for the features that we have discovered in the models. Hierarchical clustering starts out with each observation as its own cluster. It then identifies each cluster that is closest together and merges similar clusters until all clusters are merged into one cluster.

[12] https://en.wikipedia.org/wiki/Deep_learning
[13] https://skymind.ai/wiki/neural-network

[**Key Decision**] For the cluster analysis, I focused only on the hitters data due to time constraints.

[**Key Decision**] I imported player data from Fangraphs.com and removed outlier observations from the dataset and scaled the data.

Next, I created a Hitters Dendogram which showed two distinct clusters. Using sklearn's AgglomerativeClustering hierarchical clustering algorithm, I chose the parameters number of clusters at 2, affinity at Euclidean, and the linkage as complete.

Refinement

I chose "complete" as the linkage parameter because I wanted the maximum distance between sets of observations. I tried different parameters for affinity and linkage, but this architecture gave me the best results.

IV. Results

Model Evaluation and Validation

The R-Square value for Model One, the LASSO regression model, was 0.76 on the test data and 0.72 on the training data, and the R-Square value for Model Two, the Deep Neural Network was 0.72. Both R-square values are close to 1, we can say that both models do well to explain that the proportion of the variability of the dependent variable (Wins) can be explained by the independent features. These model account for 76% and 72% of the variance. If the R square was 1, it would account for 100% of the variance. The Adjusted R-Square scores for Models One and Two respectively are: 0.7656 and 0.7615. These scores can be used to compare if additional features are added to the model.

However, both models have high MSE values. For Model One, the MSE was 32.098 on the test data, and for Model Two, the MSE was higher at 37.64 which means that there is a high spread around the regression line which means that there is a range of uncertainty.

Justification

Earlier in the project, I discussed baseball's rule of thumb for predicting wins, the Pythagorean Theorem. The minimum number of runs allowed by a team over the past 20 years is 525 and that team's win total was 100. That same team scored 891 runs for a run differential of 366. According to the Pythagorean Theorem of Baseball, the team's win total should have been 120.25.

For the team that allowed the least number of runs, the LASSO model predicted 99 wins while the Deep Neural Networks model predicted 99 wins. The two models were more accurate than baseball's rule of thumb.

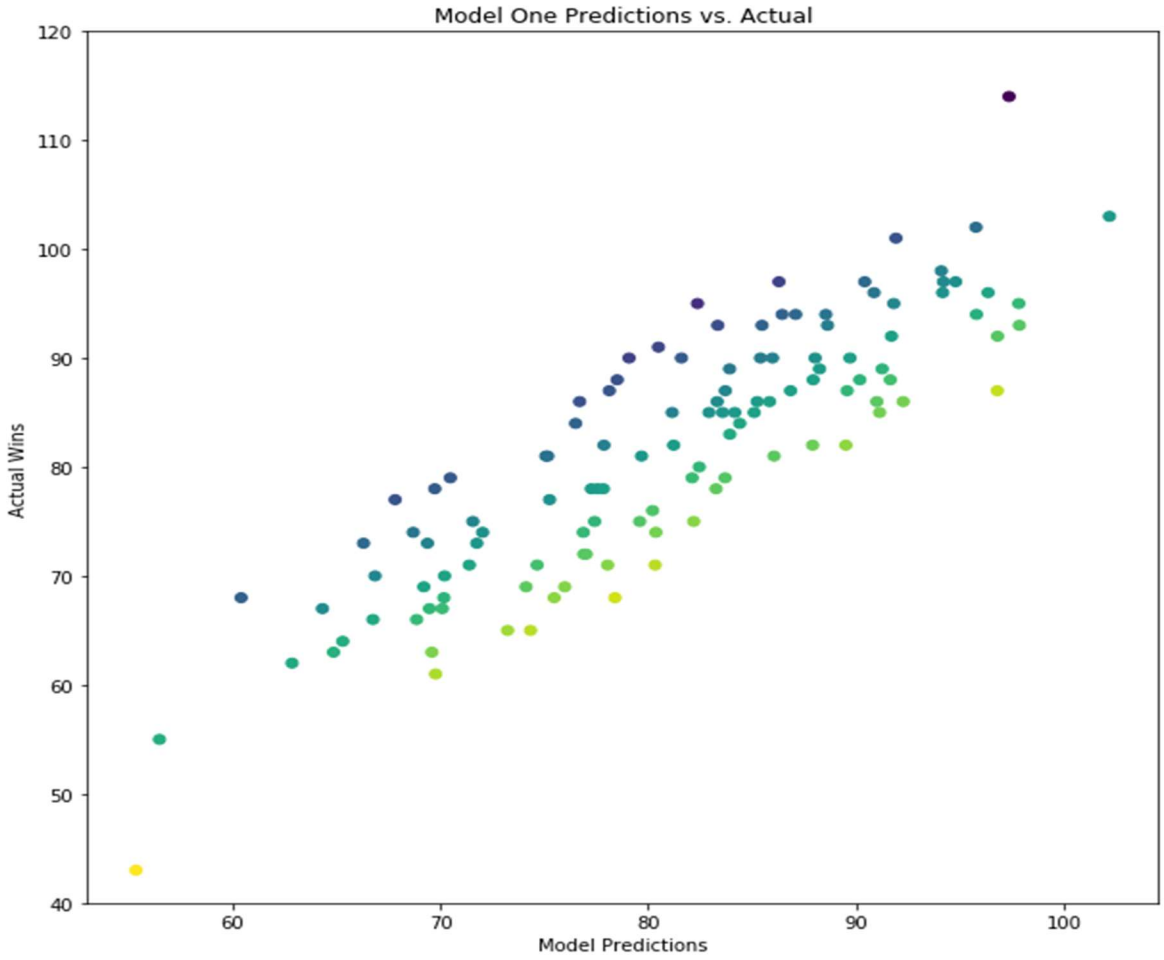
I was unable to compare the results of the model where the team scored the most runs. That observation was removed as an outlier. I tried running that observation through the workflow, but when I scaled the features, it reduced everything to 0.

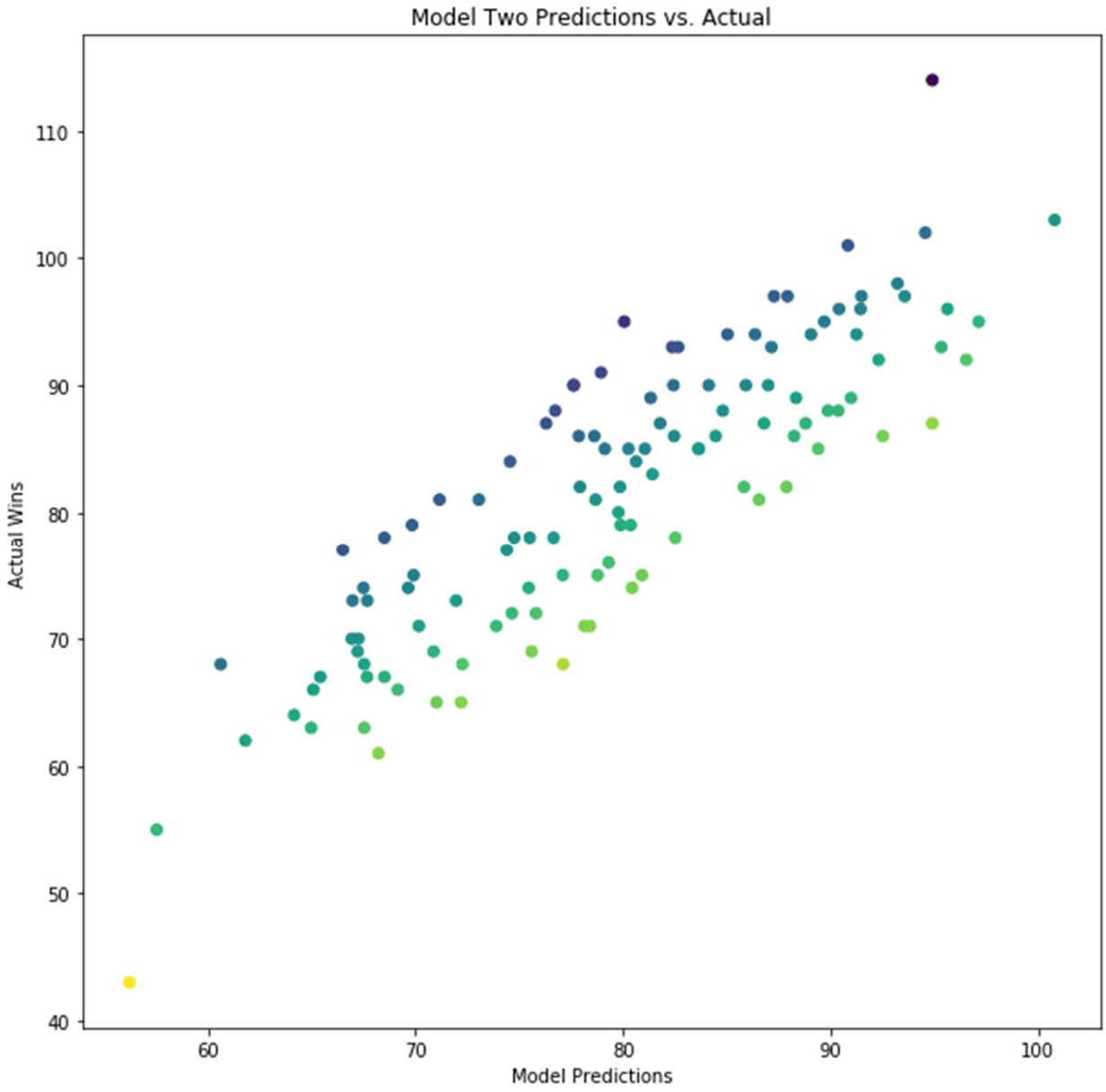
Though the models do seem to be able to predict the number of wins better than the heuristic, I cannot say with complete certainty that the models should not be relied upon to predict wins in MLB. When the models were tested on 2018 data, it did not perform well. However, that may be due to the current season being only 83% complete. The models were trained on complete seasons. With that being said, the approach is sound. From the data collection, wrangling, pre-processing, to the modeling, the workflow does show that the models may give the team better insight into what it takes to win in MLB.

VI. Conclusions

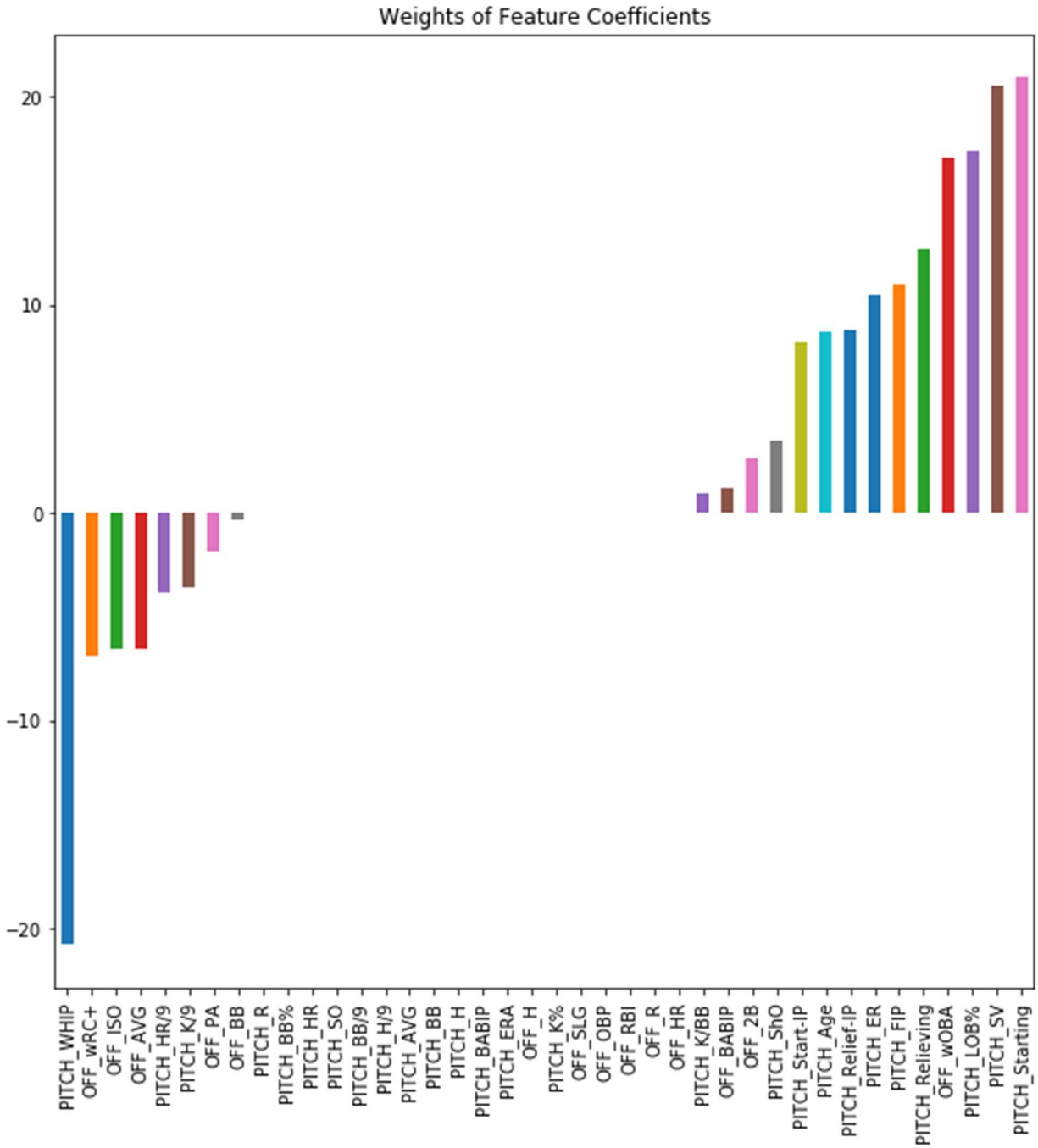
The plots below show how the models performed against the actual data. The yellow color dots show where the model predicted more wins than the actual total. The purple color dots show where the model predicted less wins than the actual total. The green color dots show where

the model predicted the same or nearly the same as the actual total.

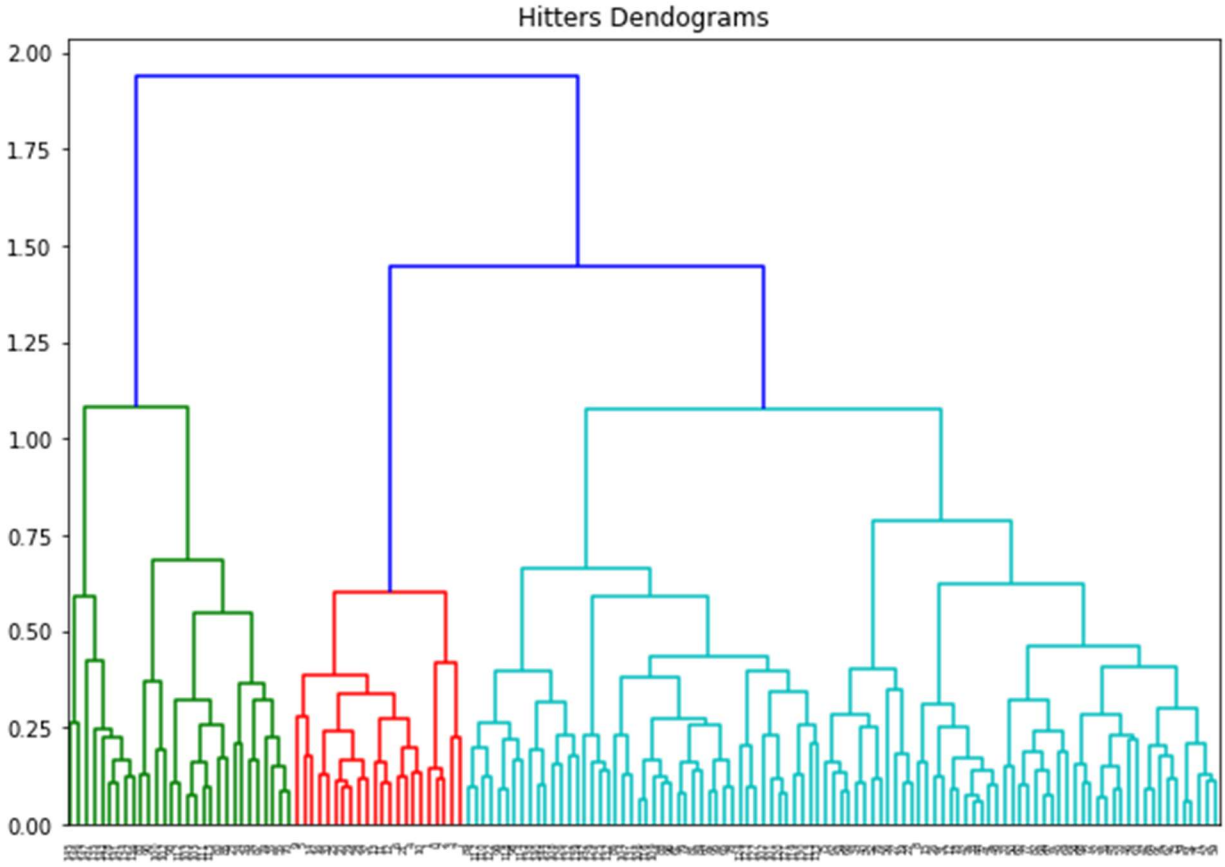



Model Two Predictions vs. Actual
Actual Wins
Model Predictions
110
100
90
80
70
60
50
40
60
70
80
90
100

The visualization below shows the most important features:



Using the Dendrogram below, I was able to cluster the hitters into two hierarchical categories, 0 and 1.



From this process, I was able to identify a player that the team should target that will help the team increase wins.

Based on the Wins predictions model and hierarchical clustering, one player that the team should target is Nolan Arenado, 3B Colorado Rockies. He is a cluster 0 player, and his performance statistics are in line with the metrics that lead to wins.

Reflection

The process started with collecting data from the FanGraphs website. I selected every hitting, pitching, and fielding statistic that the site had for the year 1998 to 2017. I then had to combine all of the years into one dataset. Next, I had to clean the data by removing the percentage sign and ensuring that all of the values were continuous. The pre-processing steps are listed below.

d. Pre-Processing
    1. Remove outliers
    2. Detect and Correct Skewness of the features
    3. Scale the data
    4. Select the Best Features
e. Modeling
    1. Build a linear regression model using LASSO Regression
    2. Build a linear regression model using a Deep Neural Network
f. Hierarchical cluster analysis to target players

When I started this project, I collected over 600 features, and my primary goal was to pare down the features to the best features for the model. I made some arbitrary decisions such as step 4 of the pre-processing where I limited the number of features that I fed into the model only where the score of the feature should be greater than 50. It's fair to say that I should have lowered the score and feed in more features into the model.

Improvement

1. Go back further in time past 1998 and get more data. I may have limited myself without a strong reason for doing so.
2. Feed more features into the model. Maybe use all of them to see if that improves performance.
3. Leave outliers in the data. For this type of analysis, there may not be any reason to drop outliers. However, when I did put the outliers back in, the model's performance declined.
4. Run the process through other sets of baseball data and see how the model performs.