Project Proposal

Predicting MLB Player Performance Using Decision Trees

Presented By:

Kurt Ruegg (Lead)

Srujana Paluri

Amy Chou

Advisor: Prof. Ming-Hwa Wang

COEN 281 - Data Mining

1. Introduction

1.1 Objective

In this project, we want to predict the performance of Major League Baseball (MLB) players in the future. We aim to factor in all the complex variables that may impact how a player will perform, and determine what their baseball statistics will be for a future date.
In order to achieve our goal, we will use a decision tree. A decision tree is well suited to this problem because it can determine and apply rules in order of importance, and is extremely accurate and adaptable. Decision trees can easily grow with data and can also be easily combined with other techniques for even further accuracy.

1.2 Problem

Predicting the performance of players in sports is a problem that is very complex and lucrative. There is no doubt that there is a lot of money in sports these days, whether it is as a business investor, a manager, a player, or even a spectator. By being able to predict performance for any particular day, investors or managers would better be able to properly manage their teams and be ready for what is ahead of them. With fantasy sports betting on the rise, even the average person can have something to gain by solving this problem.
There are many factors that go into how a particular player may perform on a particular day. These factors can be anything from the day of the week to the weather conditions on any particular day. There may even be a relationship between how one player performs and how another player (especially of the opposing team) performs. The relationship between all the factors and the outcome is not completely known. In this project, we would like to build a solution that allows the user to factor in all the complexities that may have some impact on the outcome, and predict the performance of players in the future.

1.3 Project Relation to Class

This project relates to our Data Mining class since we will be mining large amounts of baseball data in order to determine the performance of players. We are not sure exactly what we are looking for or which factors influence the performance of players. We only have a high level idea of the results we are looking for, so data mining is well suited to this problem.
We will be using a decision tree as the primary tool in this project, which is one of the many topics discussed in our Data Mining class.

1.4 Area or Scope of Investigation

We will build a system that incorporates multiple techniques for optimal performance and accuracy.

- Proper data formatting and pre-processing
- Ensembling multiple Decision Tree models for better performance
    - Gradient-Boosted Decision Tree
    - C4.5 Classification Decision Tree
- Performance analysis and optimization

2. Theoretical Bases and Literature Review

2.1 Definition of the Problem

Data mining can be used to solve many problems today. Available datasets such as baseball statistics over time can be data mined to obtain accurate predictions of how the data will look like in the future. Our problem is to use a large amount of baseball datasets over the span of five years in order to determine the performance of players in the future. This way, coaches, players and business investors can adequately prepare for the future and make the best decisions for their teams. Also, people placing bets in fantasy baseball games can optimize their bets so that they have the best chance to win.

2.2 Theoretical Background of the Problem

Many modern business organizations accumulate large amounts of data from their day-to-day activities. This data is stored in databases, data warehouses, and other various information storage schemes. This data is important and it has to be processed in order to get the useful information. Data Mining technique is used to analyze and extract the useful information from the data. It helps to analyse ,catagorise and to find the relationships among the data using various algorithms such as classification ,clustering, regression,asscociation rule mining and anamoly detections.Classification is used for prediction.In this paper we use Decision trees one of the classification algorithm to predict the performance of a baseball player.

In Classification, the system is trained with set of data and this data consists of multiple records each having multiple attributes. The input data used to train the system is called the training data. Each record will have a class label. The objective is to analyse the input data to develop the model for each class using the attributes.Whereas this model is used to to test the data for which the class labels are unknown.

Decision tree begins with the root node and branches out into internal node or leaf node. Each branch represents data attributes that fall within the range of values. data points travel through the internal nodes to reach the final node which is known as leafnode. At leaf node the data point is assigned to class label. Decision trees are simple, inexpensive and can build the models quickly even though the training set is very large.

2.3 Related Research to Solve the Problem

Sports data has been well-studied over the years. In fact, baseball data has been so well-studied that the term "sabermetrics" has been coined as meaning "the search for objective knowledge about baseball." The most common solution that has been applied to sports predictions is simply using statistical methods. The first method that is common is using linear regression. Linear regression just uses the dataset to create a model and try to fit it to an equation. Then they can use that model to predict the behavior at a future time.
The second method that is common is Support Vector Machines (SVMs). SVMs are supervised learning models that take a training dataset and creates a hyperplane model. Then, as new data examples come in, it clusters the new example into one of the categories. The SVM can generate both linear and nonlinear classifiers. For sports prediction applications, SVM is often used to rank the performance of the players, and to give the ranking predictions.
Decision trees have also been used experimentally to predict sports results. One person used a decision tree model to predict the winner of the Stanley Cup 2011 Western Conference. They got to a conclusion where if the Vancouver Canucks restricted Tampa Bay to less than 2.5 goals, then they had a 93% chance to win.

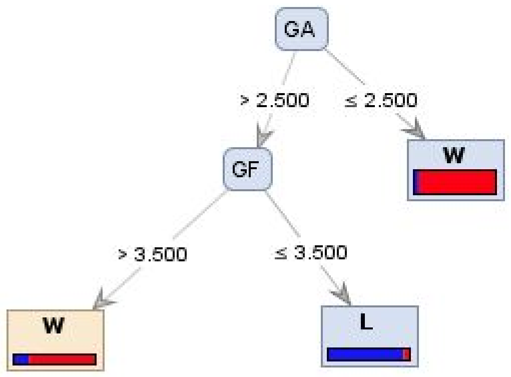


Figure 1

However, the usage of decision trees to predict sports performance has been limited, as many over methods such as linear regression and SVMs have been dominant. Most of the applications of decision trees to sports predictions has been simple, so we have room to improve. By using gradient-boosted decision trees as well as ensembling multiple models, we can come up with a more accurate method of prediction.

2.4 Advantages of Related Research

The research that has been done in the past has been a step forward for sports predictive purposes. They have given some good results and have given some insight into the ways that datasets may be analyzed. They have also given insight into the relationships that variables may have. By using regression or SVMs, they have shown with decent results that there is a lot of potential when it comes to modeling and predicting future sports performance. For example, there was a company called "Advanced NFL Stats" was able to find through regression techniques that passing efficiency was by far the most important factor when predicting wins. SVMs are very good for clustering or ranking.

2.5 Disadvantages of Related Research

Decision trees are much more useful than the classic techniques such as regression and SVMs when it comes to predicting future sports performance. The relationships between different variables in sports are very complex and regression generally cannot recognize the relationship between different variables quite as well as decision trees. Regression also has a problem that it is difficult to determine whether there is simply correlation or whether there is causation. Decision trees are better at discarding information that is essentially useless.
The decision trees that have been generated for sports predictive purposes in the past are for the most part very simple. They do not use multiple methods or utilize boosting efficiently. In our case, we can combine more methods in order to get a more desirable result.

2.6 Our Solution to Solve the Problem

We will combine multiple techniques in order to generate decision tree models. We will generate multiple decision tree models and ensemble them by comparing the errors and taking the weighted average. The two decision tree models that we will be using are a gradient-boosted decision tree model and the C4.5 decision tree model. By combining multiple trees that effective on their own, we can get an even better result than we would with just one. Also, by using a gradient-boosted model, that itself will generate multiple trees that act as one in order to improve the performance. The C4.5 model will prune efficiently in order to avoid overfitting. By combining these two methods, we believe we can obtain a good result.

2.7 How Our Solution Differs from Others

Our solution combines multiple methods that have only been used on their own. We want to see how combining these different models can improve the result. Furthermore, if we have more time, we will incorporate other types of models that are highly effective as well. By combining methods, we can get the advantages of multiple types of decision tree models.

2.8 Why Our Solution is Better

Our solution incorporates multiple techniques which have been used only on their own for the most part. These different models all have their own advantages and by comparing the errors and then taking the weighted averages we can get some of the advantages from all the different models. We will also optimize these trees to effective prune data. We may make more optimizations also as we see fit.

3. Hypothesis

3.1 Positive Hypothesis

We hypothesize that we will be able to create a decision tree model that will be able to predict various baseball stats as well as or better than most human experts.

3.2 Anticipated Result

We anticipate that we will be able to create a model that will give us meaningful predictions for baseball statistics.

4. Methodology

4.1 Input Data

The data will be scraped from the website baseballreference.com and will consist of sets of all batting and pitching data from the 2001 season to the 2015 season. This data will be divided into 80% percent training data and 20% cross validation data, with the cross validation data being the last 20% of the games chronologically. We also included weather data that was gathered from weatherunderground.com

We will use a number of features to help predict the following player statistics:
Number of Home Runs
Number of Singles
Number of Doubles
Number of Triples

In our datasets we have the following information for batters:
Date: day of the game
Team: team the player is on
Opp: team the player is facing

Result: result of game (Win or Loss)
PA: number of plate appearances
AB: number of “At Bats”
R: number of runs scored
H: number of hits
2B: number of doubles
3B: number of triples
HR: number of home runs
RBI: number of “Runs Batted In”
BB: number of walks
IBB: number of “Intentional Walks”
SO: number of strike outs
HBP: number of “Hit By Pitch Advances”
SH: number of sacrifice hits
SF: number of sacrifice flies
ROE: number of base hits due to error
GDP: number of ground plays doubled into
SB: number of stolen bases
CS: number of times thrown out while stealing

We also have the following information on pitchers:
IP: innings pitched
H: number of hits allowed
R: number of runs allowed
ER: number of “Earned Runs” allowed
BB: number of walks allowed
SO: number of strike outs
HR: number of home runs allowed
UER: number of “Unearned Runs” allowed
Pit: number of pitches
Str: number of strikes

The above data cannot be used to predict our output variables because they are unknowns at the time of prediction. However, the variables associated with previous time events can be used. Some examples of features we plan to use:
Average Number of Hits in Last 10 Games
Average Number of Runs in Last 10 Games
Average Number of RBIs in Last 10 Games
Average Number of Hits Allowed by Opponent in Last 10 Games
Average Number of Hits Allowed by Opposing Pitcher
Min Temperature For Game Day

Max Temperature For Game Day
Mean Temperature For Game Day
Precipitation on Game Day
Home/Away
Park Team is Playing At

4.2 Problem Solution

4.2.1 Algorithm Design

Our algorithm will consist of ensembling two different decision tree models, the C4.5 decision tree model and the Xgboost Gradient Boosted Decision Tree model. First, we will create our two models by using our training set to construct our model. Next we will use our cross validation set to choose the best model. Once we have separately created the best models of each type, we will then ensemble the two into a single model. To do this we will measure the cross validation error for different weightings of the two models and select the weighting that minimizes the error.

4.2.2 Languages Used

We will use Python to implement both the Xgboost Gradient Boosted Tree model and C4.5 decision tree model.

4.2.1 Tools Used

Within Python, we will use the following library:
pandas
numpy
scikit-learn
Xgboost

Within R, the following libraries were used:
ggplot2
weatherData

4.3 How to Generate Output

To predict a statistic for a particular player and game, we simply need to create a feature set for a our player and feed it into our model.

4.4 How to Test Against Hypothesis

For now we can only test against our cross validation set, but once the 2016 MLB season begins on April 3, we can test our model in real time.

There are many pundits and models that make predictions on baseball players during the season. We could compare our model results to the predictions made from these other sources to test the efficacy of our algorithm.

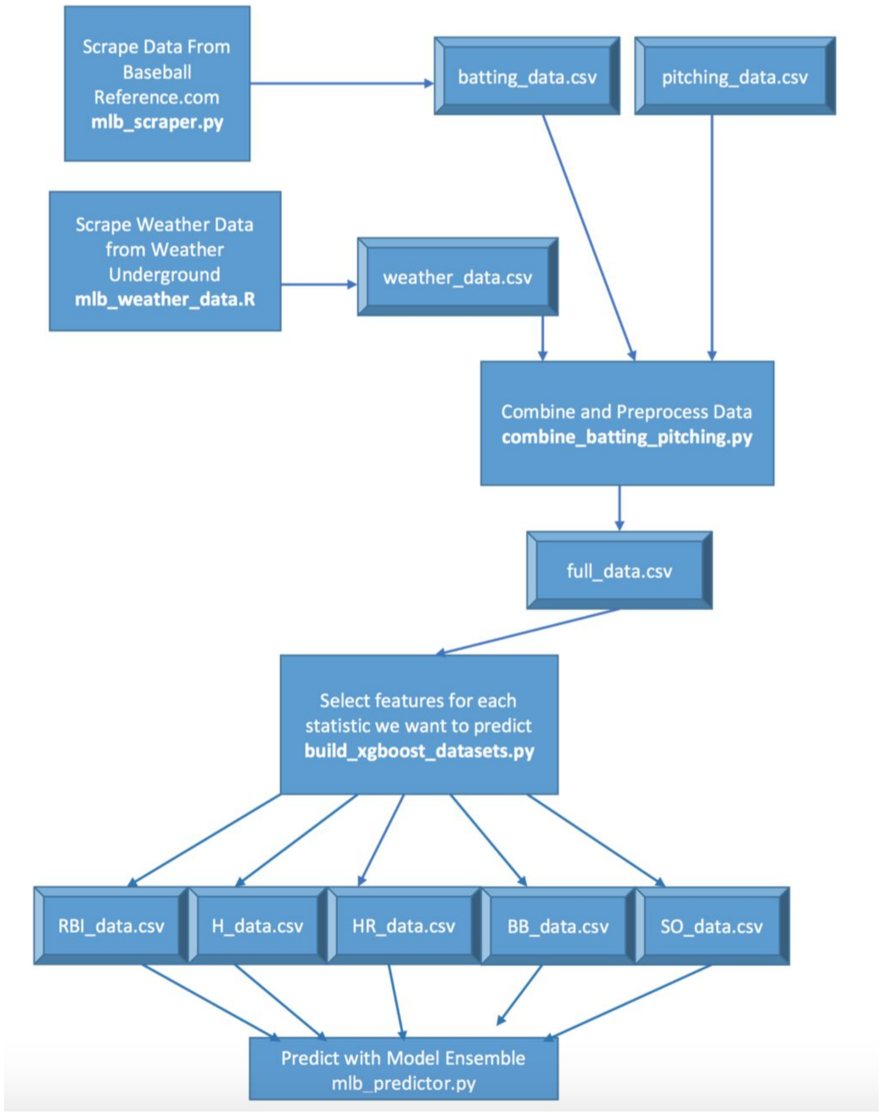
5. Implementation

5.1 Code Architecture

The code architecture is as follows:

1. Perform data scraping to get the raw data
2. Import necessary libraries (numpy, sklearn, pandas)
3. Load in raw data (multiple files)
4. Pre-process data
    a. Perform error checking on data
    b. Convert data to format that is consumable by the libraries
    c. Perform rolling averages
    d. Aggregate pitching, batting and weather data to single source
5. Split data into 70% training set and 30% testing set
6. Use training data to create three models:
    a. Xgboost Gradient Boosted Tree
    b. CART (Classification and Regression Trees Algorithm) Regression Tree with sklearn DecisionTreeRegressor library
    c. SVR (Support Vector Regression)
7. Use the testing data to compute RMSE (Root Mean Square Error)
8. Compute weighted average for the three models based on the errors
9. Used the ensembled model to output the prediction based on input to be predicted

5.2 Design Document and Flowchart


Scrape Data From Baseball Reference.com mlb_scraper.py
batting_data.csv
pitching_data.csv
Scrape Weather Data from Weather Underground mlb_weather_data.R
weather_data.csv
Combine and Preprocess Data combine_batting_pitching.py
full_data.csv
Select features for each statistic we want to predict build_xgboost_datasets.py
RBI_data.csv
H_data.csv
HR_data.csv
BB_data.csv
SO_data.csv
Predict with Model Ensemble mlb_predictor.py

5.2.1 Overview

The high level design flow is as below:

1. Identify goals and collect data.
2. Investigate tools, libraries and resources to be used.
3. Create working POC using small pieces.
4. Test and optimize models.
5. Discuss and adjust as necessary.

5.2.2 Data Preprocessing

One module did all the scraping of the data from baseballreference.com and weatherunderground.com. The files were quite large and both websites put limits on the amount of data that can be displayed at a single time, so the datasets were initially downloaded as a large number of small chunks.

A second module combined all the chunks into a cohesive dataset and did all the processing. Preprocessing included:

- Setting all NA values to -999
- Calculating the rolling average of various stats
- Turning home and date into numerics
- Lining up pitching data with each training example
- One-hot encoding park playing at

We also had to convert the data to a format that was interpretable by the libraries which we chose to use. We utilized a library called pandas that helped with data preprocessing. It reads in the csv files and then turns them into data frames which can be easily manipulated and modified.
The output of this stage was a set of combined, sanitized data sets.
The test and the training data were split into separate files, as well as the training points and the labels. The output of this stage looks as below:

- final_RBI_train.csv
- final_RBI_train_label.csv
- final_RBI_test.csv
- final_RBI_test_label.csv

5.2.3 Xgboost (Gradient-Boosted Decision Tree)

The training set was used with the Xgboost library to create a Gradient-Boosted Decision Tree. This library had many parameters that we could tweak in order to get desired results. We adjusted the parameters to reduce overfitting as well as get a reasonable run-time. By incorporating the gradient-boosted decision tree, we get the benefit of ensembling from a forest at the cost of potentially overfitting. We ran the test data set on the model and computed the RMSE. The list of parameters and an explanation of what they do:

Eta : the weight of each individual tree we add to the model. By decreasing eta, we make the boosting process more conservative and thereby prevent overfitting

Gamma : the minimum loss that we will allow to split a node. Increasing this value prevent overfitting

Max_depth : the maximum depth of any individual tree in the ensemble

Min_child_weight : the minimum total weight value of examples allowed in a node. A larger value makes the model more conservative and decreases overfitting.

Subsample : the ratio of training examples subsampled to build each tree. Once again, this technique decreases overfitting

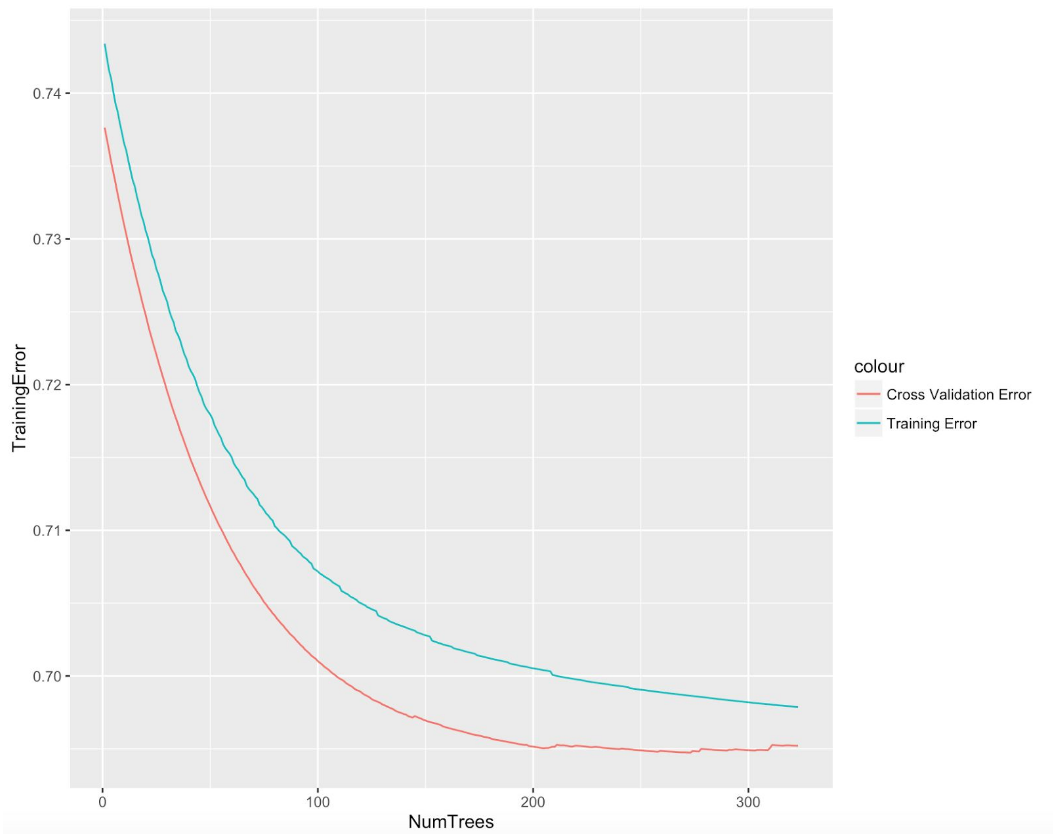
Colsample_bylevel : subsample ratio of columns used to build each tree.

Lambda : regularization constant using during individual tree building process

The following graph shows an ensemble model being built with the following parameters:

'bst:max_depth':5,
'bst:eta':0.01,
'gamma':0.7,
'lambda':0.7,
'subsample':0.7,
'colsample_bytree':0.7

As the number of trees increases, the error decreases. However at a certain point, the error begins to go back up. Once the cross validation error begins to increase, the building stops and we choose the ensemble that produced the lowest RMSE score



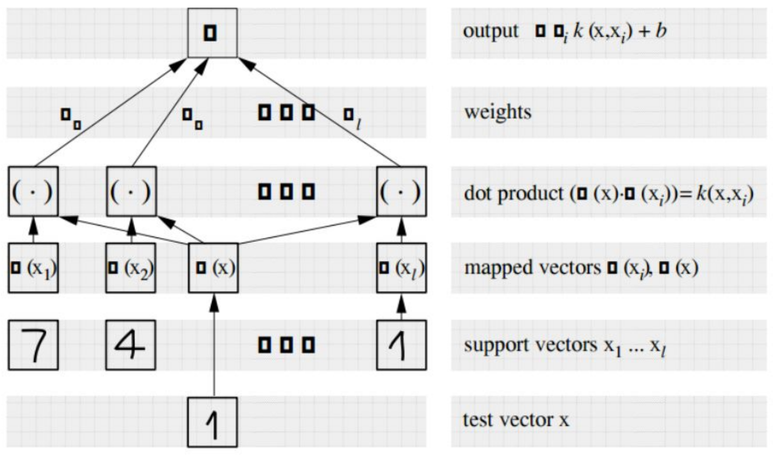
5.2.4 Sklearn DecisionTreeRegressor (CART - Classification and Regression Trees)

We utilized the Sklearn DecisionTreeRegressor library which creates a tree based on the CART (Classification and Regression Trees) algorithm. This algorithm is very similar to the C4.5 algorithm which we described above. It chooses the most important features first and subsequently the lesser important ones. Again, while using the library, we kept adjusting the parameters to get better results. To avoid overfitting, we set a minimum weight and amount of samples that a feature must have in order to split, or that a leaf must have in order to be a leaf. We used the regression tree to predict the values. We ran the test data set on the model and computed the RMSE.
Also the pydot library and graphviz were used in order to draw the decision tree for visualization purposes. This gave us more of an idea what the tree we were creating actually looked like, and what it was actually doing.

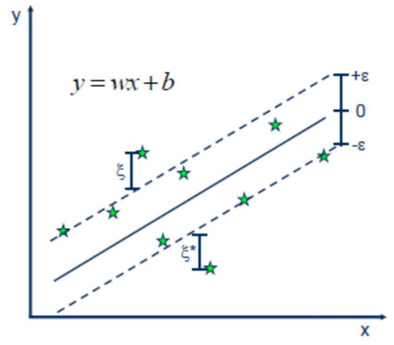
5.2.5 Sklearn SVM (Support Vector Regression)

We used the Sklearn SVR library in order to build a Support Vector Regression. The implementation is similar to the SVM (Support Vector Machine), but applied on a regression problem. For classification problem the Support vector regression uses the same principals as SVM only with minor differences. As the output is real number it is very difficult to predict the information which will have the many number of possibilities hence the algorithm is very complicated.The main idea is to minimize the error , individualizing the hyperplane which maximizes the margin. However a some part of the error is tolerated.

Architecture of regression Machine:


output ◘ ◘$_i$ k (x,x$_i$) + b
weights
dot product (◘ (x)·◘ (x$_i$))= k(x,x$_i$)
mapped vectors ◘ (x$_i$), ◘ (x)
support vectors x$_1$ ... x$_l$
test vector x
7
4
1
1

The input pattern is mapped into feature space by a map function ɸ. Then the dot products are computed with the images of training patterns of ɸ. Which corresponds to evaluating the kernel function. Then the dot products are added up using the weights (∝i-∝i *) which are subsets of training pattern . This plus a constant term b is yield to final prediction output.



The basic idea here is to generate the function with at most epsilon ε deviation from actually obtained targets yi for all training data and same time as flat as possible. The errors within epsilon are not considered.

There are two different types of SVR s

1. LInear
2. Non-Linear

Linear SVR

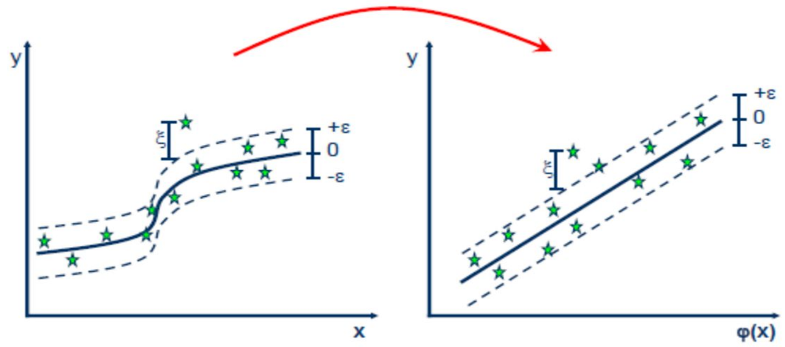
$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$

Nonlinear SVR

The kernel function transform the data into higher feature space in order to make the linear separation possible.

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle \varphi(x_i), \varphi(x) \rangle + b$$

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b$$



Kernel Functions:

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\|\mathrm{x}_i - \mathrm{x}_j\|^2}{2\sigma^2} \right)$$

We applied the linear, polynomial and RBF kernels to the training set to see which kernel would fit the problem the best. For our problem, we concluded that the RBF worked the best, followed by the polynomial . We ran the test data set on the model and computed the RMSE.

5.2.6 Ensembling Multiple Models

We used the RMSE that was computed for each model in order to weight the models and create a weighted average for the total prediction.
The error of ensembling the models was much better than any of the models on their own.

6. Data Analysis and Discussion

6.1 Output Generation

We generated the following outputs using the ensembled predictor:

- Prediction of given label (for ex: RBI's or Home Runs)
- Root Mean Squared Errors for cross validations

We did an in depth analysis on our models that were built to predict
Model 1 RMSE: 0.694444
Model 2 RMSE: 0.697050271149
Combined RMSE: 0.692880678494

6.2 Output Analysis

The Root Mean Squared Error (RMSE) of the combined model was less than the RMSE of any of the individual models. This is a result of the fact that decision trees are best implemented as an ensemble of trees to provide generality and to prevent overfitting.

The output is the expected value of a statistic for a particular player and game. The results and the RMSE scores are within the acceptable range for predicting statistics

6.3 Compare Output Against Hypothesis

Our RMSE score is the best indicator of how well our model is doing at predicting the statistics we are interested in. We also looked at the output of data. Below is an example of some of the predicted values our model output and the actual values:

Actual	**Predicted**	**Actual**	**Predicted**
2	0.55586201	1	0.463624209
0	0.049382836	0	0.387319535
1	0.527879298	0	0.415851891
0	0.294905454	0	0.387146711
2	0.466825664	1	0.471220911
0	0.535081863	0	0.301405311

6.4 Abnormal Case Explanation

There were missing values in the set which may have caused some of our results to be slightly impacted. We may have gotten better results had the data set been more sanitary and complete.

Another abnormality that occurred was the training error was greater than the cross validation error for many of the Xgboost models. This usually is not the case in most examples of boosted tree construction. The reason for this could be that the model was actually being underfit, which means the parameters were set too aggressively to prevent overfitting. Another possible solution could have been to increase the number of training examples.

It was also noted that predicting the HR (home run) statistic produced a much worse RMSE than the other categories (1B, BB, SB, SO). This is likely due to the high variability of this particular statistic, as opposed to any failure in our model. Home runs are much more prone to random variation because they are such an extreme event. Our other categories are much more common occurrences and therefore have less game to game variability.

6.5 Statistic Regression

We used RMSE as our verification statistic. Minimizing this error metric can be shown to produce the optimal scores for a regression model.

6.6 Discussion

The results that we produced were good but definitely has room for improvement. There is always room to add more features that have predictive power to make the models better, which makes the Decision Tree building process a never ending one. We will be able to test the models on real data in April when the 2016 MLB season begins.

7. Conclusion

7.1 Summary and Conclusions

The three models produced adequate results on their own, averaging a relatively low RMSE for each data set. Through these advanced algorithms and including pruning and other measures for overfitting, we were able to generate trees as well as a SVR that performed pretty well on their own. This shows that these machine learning techniques can be applied well to predicting sports data. By ensembling multiple methods, we were able to create an even more accurate model.

7.2 Recommendations for Future Studies

Though our model performed relatively well, there is still a lot of work to be done here. The error of our model can still be reduced quite a bit. This is a very complex problem, and since the data is so specific, sometimes we may be vulnerable to overfitting. Though we did go through various methods to reduce the overfitting, there is still more research to be done.

8. Bibliography

http://cs229.stanford.edu/proj2012/Bookman-PredictingFantasyFootball.pdf
http://cs229.stanford.edu/proj2013/ChengDadeLipmanMills-PredictingTheBettingLineInNBAGames.pdf

http://www.simafore.com/blog/bid/58159/Predictive-Analytics-in-Sport-Stanley-Cup-2011-Western-Conference

Du et al., Building Decision Tree Classifier on Private Data, IEEE International Conference on Data Min-Ing. 2002.

Zhao et al., Cost-Sensitive Decision Tree With Probabilistic Pruning Method, IEEE. 2014.

Fekri et al., Cross Split Decision Trees For Pattern Classification, IEEE. 2015.

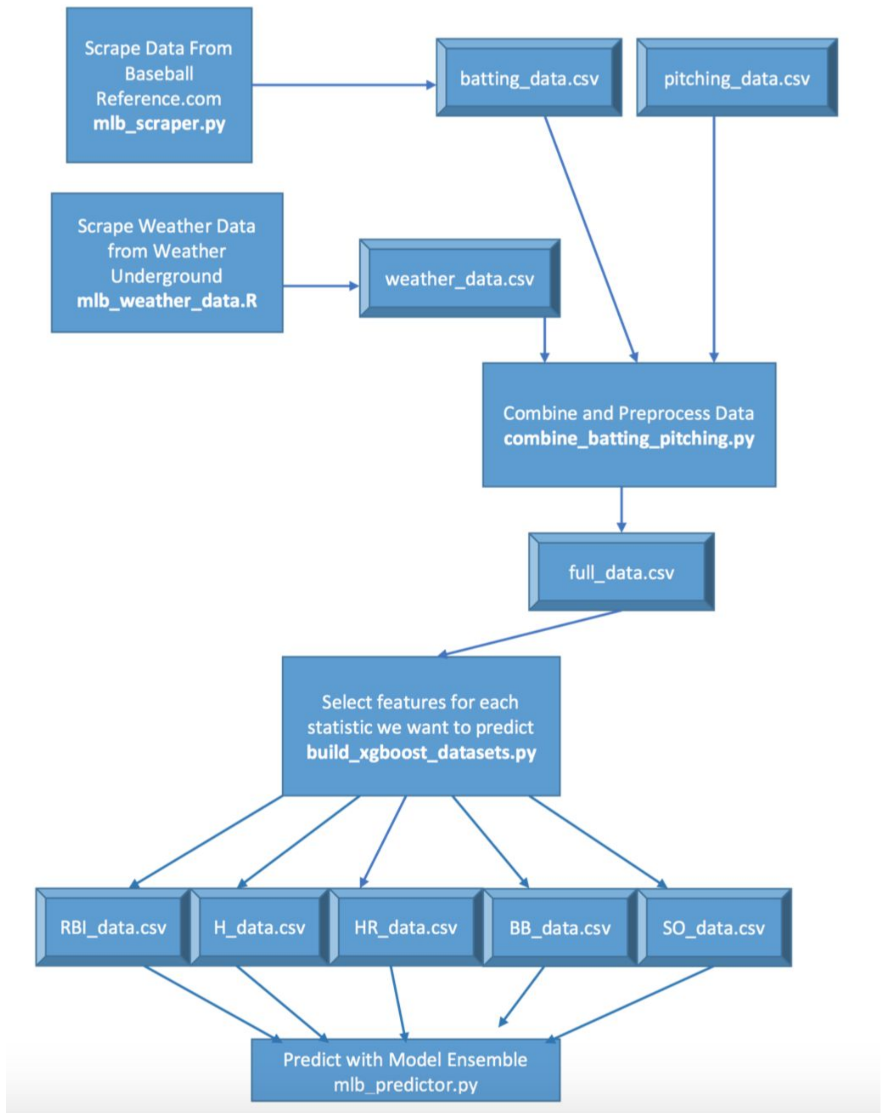
Friedman H. Jerome, Greedy Function Approximation: A Gradient Boosted Machine, IMS Lecture. 2001.

Freund et al., A Short Introduction to Boosting, AT&T Research Laboratory, 1999.

Chen et al., Research on Prediction Method for Pivotal Indicator of Hospital Medical Quality Using Decision Tree, IEEE. 2015.

9. Appendices

9.1 Program Flowchart

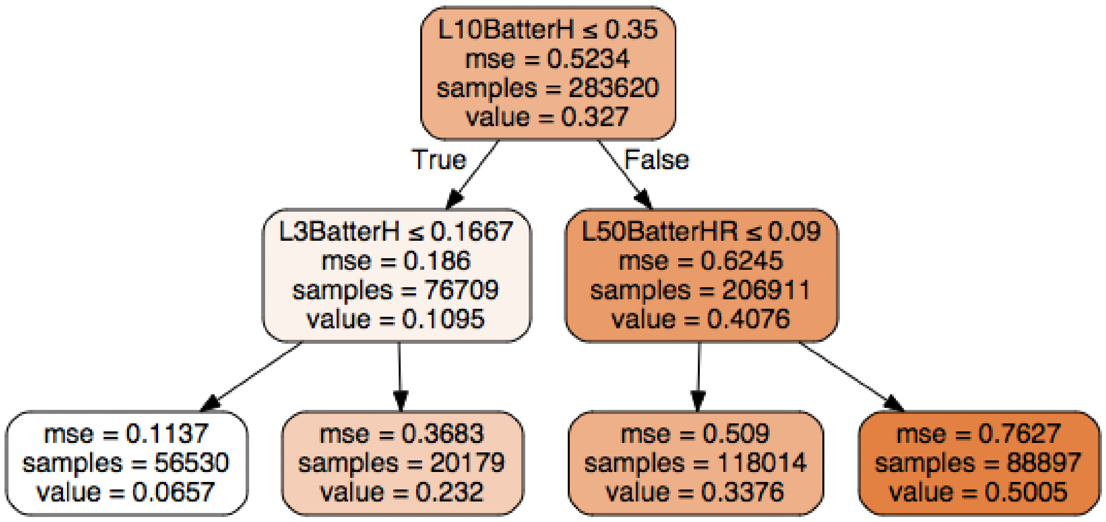

Scrape Data From Baseball Reference.com
mlb_scraper.py
batting_data.csv
pitching_data.csv
Scrape Weather Data from Weather Underground
mlb_weather_data.R
weather_data.csv
Combine and Preprocess Data
combine_batting_pitching.py
full_data.csv
Select features for each statistic we want to predict
build_xgboost_datasets.py
RBI_data.csv
H_data.csv
HR_data.csv
BB_data.csv
SO_data.csv
Predict with Model Ensemble
mlb_predictor.py

9.2 Program Source with Documentation

mlb_predictor.py

9.3 Input/Output Listing

Simplified Decision Tree (Full one was much larger):



9.4 Other Related Material

Sklearn Library: http://scikit-learn.org/stable/modules/tree.html