Classification of baseball players based on machine learning algorithms

Andy Fang, Dehao Liu, Yanglong Lu
{dehao.liu, ylu381, afang3}@gatech.edu

Abstract

Over the years, fantasy sports has become an increasingly popular pastime and in recent years have seen the rise to sport gambling websites. Here, fans are able to "bet" on certain players whom they predict will outperform the others in said statistics and ultimately allow the fans make real or virtual currency. Our goal is to identify the high-performance baseball players for the person who drafts them into their team. Our dataset is the Sean Lahman baseball database that contains patting, pitching, and fielding statistics in 2016. Supervised and unsupervised machine learning algorithms are applied to the baseball dataset and some good results are obtained.

1 Introduction

We are looking at data related to fantasy baseball. Our goal is to identify the players that will earn the most points for the person who drafts him into their team. Our dataset is the Sean Lahman baseball database that contains patting, pitching, and fielding statistics until 2016.

Over the years, fantasy sports has become an increasingly popular pastime and in recent years have seen the rise to sport gambling websites. Here, fans are able to "bet" on certain players whom they predict will outperform the others in said statistics and ultimately allow the fans make real or virtual currency.

Our ultimate research goal is to find the players that will earn the most points for the person who drafts him into their team. In pursuit of this goal, there are various levels of granularity we could choose for our predictions:

Classify the players into two groups based on the prediction of their level of earned points.

Identify the best player based on the predicted earned points

In this project, we will choose the first level to classify the players into two groups. Besides, we will look into the improved performance on prediction after some data preprocessing. We also want to evaluate different machine learning algorithm based on their own performance.

http://www.cbssports.com/ provides a way to score the points earned by the player for batting and pitching categories. With the batting and pitching datasets, we can calculate points earned by each player. The points corresponding to each player are filled in the master dataset which contain the

information and background of players. There are more than 20 features in the master dataset, so we will do the feature selection to reduce the dimension of the feature space. Because the dataset is large, we can do the clustering to separate data points if necessary.

2 Related Work

Nikolai Yakovenko studied the performance of pitchers season by season. He determined that there is a baseline accuracy metric given by a 3 year regression of player performance and player age. He attempted to improve upon this model by taking into account things such as surgeries, strikeout statistics, injuries, and frequencies of different pitch types. His project is similar to ours in that it attempts to predict player performance. However, his project is different from ours in that his goal was to project performance based on previous statistics, and ours seeks to separate high performers, medium, and low performers from each other based solely on biographical information.

Jensen, McShane, and Wyner used a hierarchical Bayesian model to examine variables such as home run totals, at bat total, age, home ballpark, and position. This differs from our approach again in the manner that we are classifying players based solely on demographics, and also in that they are, again, making projections for future seasons based on already established players. We are trying to see if new players' performance can be predicted based on their biographical data.

3 Approach and implementation

Our goal is to attempt to seperate "high performing" major league baseball players from "low perfoming" players based on biographical data such as: age, state of origin, height, weight, handed-ness, and number of years active. Our dataset comes from the Lahman Baseball Database. Specifically, we used the biographical data in the master file with the performance data in the batting file.

We are using 2 clustering algorithms and 5 classification algorithms to try to see if we can classify certain players as high and low performers based on their biographical data. In order to preprocess our data, we marked and cleaned missing values, randomized the dataset, made numeric to nominal conversions, and did some feature selection. We examined only players that played in the 2016 season in order to have the most relevant data without having too much for our algorithms to process.

We are using WEKA to implement the following algorithms: K-Means Clustering, Expectation Maximization, Decision Tree, Neural Networks, Boosting, SVM, KNN.

4 Data description and preprocessing

The dataset is extracted from Lahman's baseball database http://www.seanlahman.com/baseball-archive/statistics/. The dataset includes the comprehensive statistic information of batting player in 2016, like biographic information, salaries, awards, performance and so on. http://www.cbssports.com/ provides a way to calculate the points earned by the player for batting categories. Based on the earned points, batting players are divided into two performance groups, namely high and low, represented by number 2 and 1. The original dataset is consisted of 2 classes with 17 attributes and 1275 instances. The explanation of 17 features is listed in

Table 1.

Table 1. Feature descriptions

1. birthyear: Year player was born	2. debut: Date that player made first major league appearance
3. birthMonth: Month player was born	4. salary: Salary
5. birthday: Day player was born	6. awardID: Name of award won
7. birthCountry: Country where player was born	8. AllStar: Attend allstar matches
9. birthState: State where player was born	10. teamID: Team
11. weight: Player's weight in pounds	12. lgID: League
13. height: Player's height in inches	14. G_all: Total games played
15. bats: Player's batting hand (left, right, or both)	16. G_batting: Games in which player batted
17. throws: Player's throwing hand (left or right)	

The dataset includes missing values in 'birthState' and 'salary' attributes which are marked as "?". The data with missing values in 'birhtState' attribute are deleted and the missing values in 'salary' attribute are replaced with the mean of all salaries. Some features with too many distinct values such as birthState, birthCity, and playerID are removed. There are 16 features used for prediction.

5 Experimental Results

In the section, we present the experimental results of supervised and unsupervised machine learning algorithms and provide some detailed analysis.

5.1 Decision Tree

For the decision tree algorithm analysis, J48 in Weka is used to compute the result. It uses the on-line pruning algorithm and use information gain to split attributes. The confidence factor is chosen as the variable to analyze the performance of J48 in Weka. It is stated in Weka that the confidence factor is used for pruning and as the confidence factor decreases, there will be more pruning. Some parameters about the structure of the decision tree is specified. For example, the minimum number of instances per leaf is set to be 1. There is another option for pruning used in decision tree such as the reduced error pruning. The predictions from two types of pruning are also compared.

k-fold cross validation method is used to train the dataset and test the model. The value of k is selected as 10.

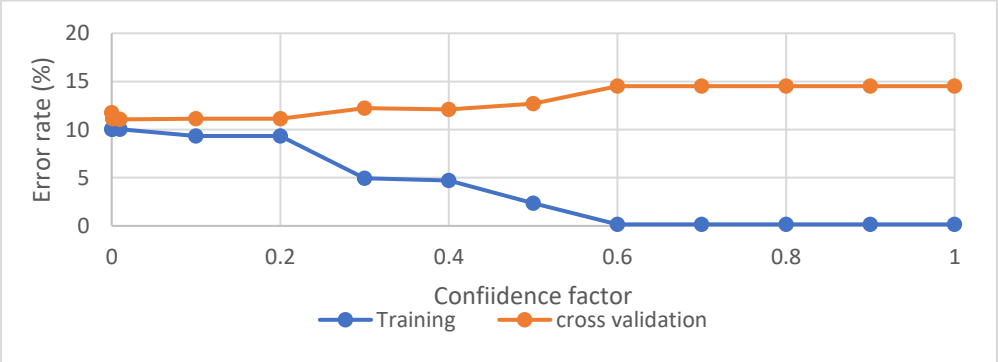


Figure 1. Complexity curve for decision tree algorithm with baseball dataset.

In Figure 1, the training error keeps decreasing as the confidence factor increase. The training error approaches to 0 after confidence factor is 0.6. The training time also dramatically increases after 0.6, because less pruning used leads to the structure of the decision tree complex. The cross validation error decreases before confidence factor as 0.001 and increase after that, which indicates the overfitting happens when the confidence factor is greater than 0.6. The decision tree biases to the short tree and when confidence is 0.001, the number of leaves is 23 and the size of the tree is 31. The tree is simple at this point and gives the best prediction accuracy, which is 11.0588%. When the reduced error pruning is used, the prediction error is 11.8431%, which is higher than using on-line pruning. If no pruning is used, the prediction error is 14.5882%. Therefore, pruning can be used to improve the prediction accuracy and the on-line pruning used in J48 is the most appropriate one in this case. The training time is 0.02s. From the results, it is found that decision tree can handle this prediction problem very well.

5.2 Neural Networks

In the neural network machine learning algorithm, We modified numbers of layers and number of nodes. These hyperparameters are important to the neural network because hidden layers would determine the accuracy of the predicted outcome. Then the learning rate was changed to investigate its effects on the efficiency of the algorithm. To test changing numbers of layers and number of nodes, we have changed numbers of nodes first then tried to increase the layers in the learning algorithm. We generated the outcome in the chart format instead of the graph format simply because it is easier and more reasonable to put the outcome in chart form due to changes in two attributes as shown in Table 2 and Table 3.

Table 2. Complexity chart of neural networks algorithm with baseball dataset

Layers	Cross validation	Training	Training time
2	14.7451	13.1765	3.26
4	15.5294	6.5098	6.08
6	16.0784	22.7451	8.31
8	16.3137	11.2157	10..99
10	16.8627	8.2353	13.64
2,2	16.1569	12.0784	2.95
2,4	15.8431	12.3922	3.15
4,2	15.2157	8.9412	5.06
4,4	15.3725	9.9608	5.25
4,6	14.7451	10.1961	5.85
4,8	15.9216	9.4118	5.88

Table 3. Effect of learning rate on the accuracy and efficiency of NN algorithm when layers=4,6

Learning rate	Cross validation	Training	Training time
0.1	16.8627	8.9412	5.866
0.3	14.7451	10.1961	5.85

0.5	14.4314	8.7843	6
0.7	17.2549	13.8039	5.61

From the complexity chart, increasing the number of layers does not necessary increase the accuracy. Increasing the number of nodes will increase the accuracy and training time as well. Though the point of 2 and point of 4,6 has same cross validation error, however, the training error at the 4,6 point is lower. After the 4,6 point, overfitting occurs. We chose Layers=4,6 as the hyper parameter for the neural network algorithm with the baseball dataset since it has highest accuracy and relatively low training time. As learning rate increases, the error rate decreases first and then increases, while training time does not change too much. This is because, as the learning rate increases, more amounts of weights are updated and it is not easy to converge. Therefore, we chose a learning rate of 0.5 as the hyper parameter for the neural network algorithm with the baseball dataset.

5.3 Boosting

For the Adaptive Boosting algorithm, the J48 decision tree with the same hyper parameters as the decision tree part is used as the classifier. The number of iterations is used as the variable, because the algorithm relies on weights made in each iteration and weights are placed on the misclassified categories in the next run.

The value of k is selected to be 10 for the k-fold cross validation after testing several values.

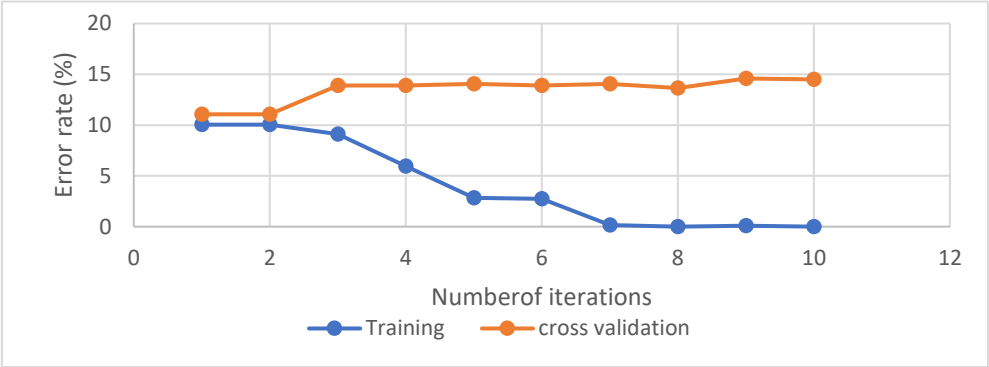
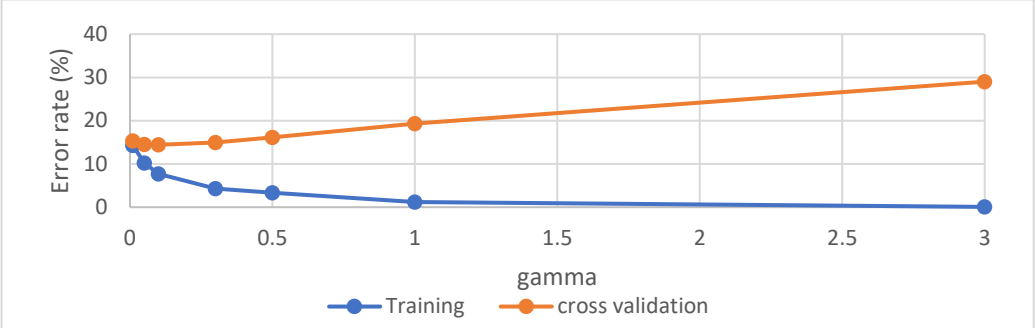


Figure 2. Complexity curve for boosting algorithm with baseball dataset.

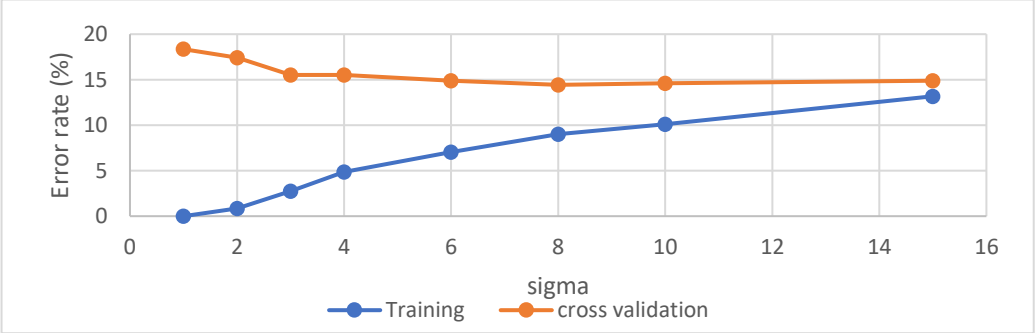
In Figure 2, it is found that as more iterations used, the training error keeps decreasing and approaches to 0 after 7 iterations, because each iteration in boosting is based on the misclassifications in the previous iteration and training performance can be always improved. However, the cross validation error does not get improved as more iterations used. The number of iterations which gives the best prediction is 1, which is equivalent to the decision tree algorithm. An overfitting is happening when more iterations are used.

5.4 SVM

For the SVM algorithm, the complexity parameter is firstly determined, because it trades off misclassification of training examples against simplicity of decision surface. A low complexity parameter makes the decision surface smooth, while a high complexity parameter aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. Then, two kernels are used, such the RBF kernel with gamma as the variable and the Puk kernel with sigma as the variable. In this case, the complexity parameter is selected at 10 after trying several values.



(a)



(b)

Figure 3. (a) SVM using RBF kernel with gamma as variable; (b) Puk with sigma as variable

In Figure 3(a), graph shows the SVM algorithm using RBF kernel with the gamma as the variable. The gamma defines how far the influence of a single training example reaches, with low values meaning 'far' and the high values meaning 'close'. The gamma can be seen as the inverse of the radius of influence of samples selected by the model. In this dataset, the prediction error decreases as the gamma increases before 0.1 and the prediction error decreases after that. The training error decreases as the gamma increases. Therefore, gamma is selected as 0.1 and the prediction error is 14.4314%. The graph in Figure 3(b) shows the SVM algorithm using Puk kernel with the sigma as the variable. The sigma=8 is selected for the Puk kernel, because it has the lowest prediction error. The prediction error is 14.4314% which is the same as the best prediction using RBF kernel.

5.5 KNN

In the creation of KNN algorithm, I have chosen number of nearest neighbors as the variable. Number of nearest neighbors would likely to be the most influential factor in this algorithm because in this lazy learning algorithm, it will always attempt to look up k number of nearest data to classify the data that is passed in. Figure 4 shows the complexity curve for KNN algorithm with baseball dataset. As the number of nearest neighbors increases, cross validation error decreases first and then increases while the training error increases first and then decreases. The training error is always zero when k equals 1, because the nearest neighbor is the data point itself. From KNN, lazy learning algorithm, the value of k = 60 fits best with this dataset since the gap between cross validation and training error is smallest. The training time of KNN is always 0 and the error rate is larger than any other supervised machine learning algorithm. This is because KNN is a kind of lazy algorithm thus its accuracy is low and efficiency is high.

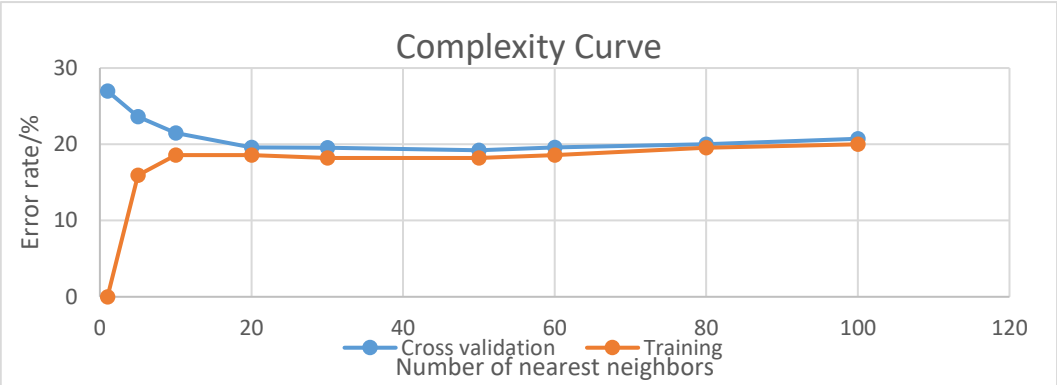


Figure 4. Complexity curve for KNN algorithm with baseball dataset.

5.6 Dimension reduction

For dimension reduction, we used InfoGainAttributeEval in Weka to select those attributes with larger InfoGain. The rank of attributes is list in Table 4.

Table 4. Ranked attributes according to InfoGain with baseball dataset

InfoGain	AttributeID	Attribute
0.54403	16	G_batting
0.39246	15	G_all
0.06517	6	height
0.04707	10	salaries
0.03482	11	awardID
0.03223	7	bats
0.02561	4	birthCountry
0.01696	8	throws
0.01289	9	debut
0.01027	12	AllStar
0.00847	13	teamID
0.00766	5	weight
0.00215	14	lgID
0	3	birthDay
0	2	birthMonth
0	1	birthYear

According to the rank of attributes, we remove those attributes with 0 InfoGain, namely birthday, birthMonth and birthYear.

5.7 Clustering

For clustering, we used both the K-Means algorithm and also the Expectation Maximization algorithm.

In order to find the best value of K for K-Means, we examined the sum of squared error value as K changed. The error as the number of clusters changes is displayed in figure 5.

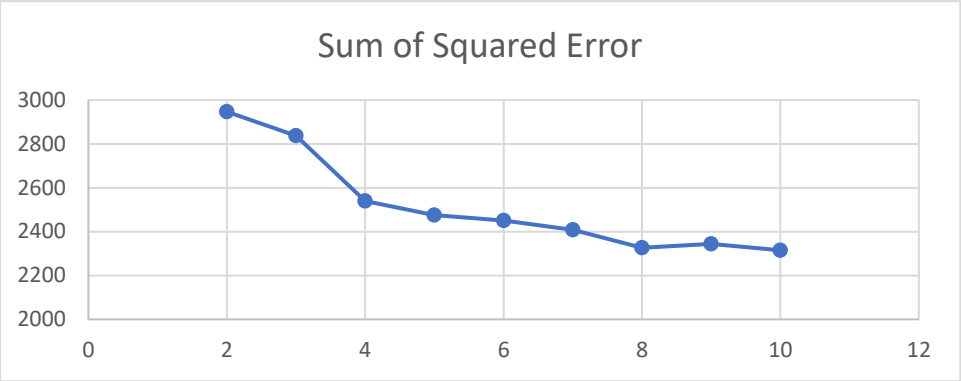


Figure 5. Error vs number of clusters

Using the elbow method, we determined that the optimal number of clusters for the K-Means method was 4. The rate of change of sum of squared error decreases after that point.

We used a similar method of finding the best number of clusters for the Expectation Maximization algorithm. We examined the effect of the number of clusters vs the log likelihood of the model. This is displayed below in figure 6.

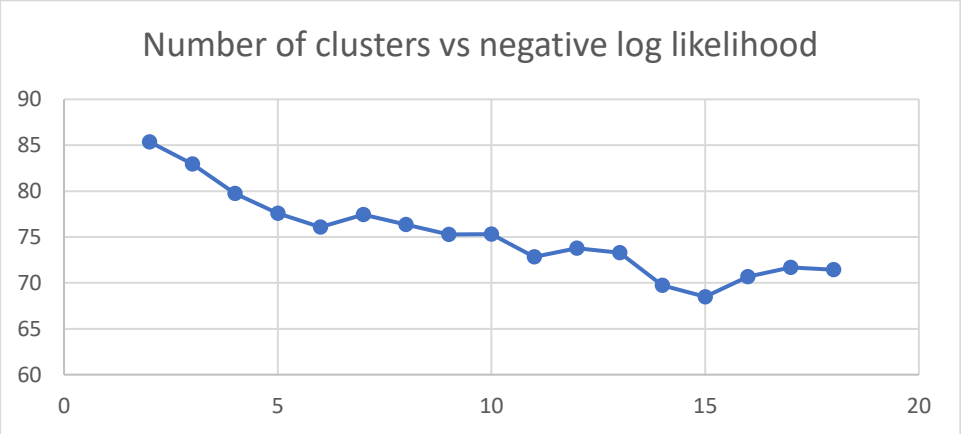


Figure 6. Clusters vs Log Likelihood

The “elbow” here appears when the number of clusters equals 6. The rate of decrease of negative log likelihood changes after k = 6.

5.8 After dimension reduction and clustering

Dimensional reduction is a useful method to improve the training efficiency especially for the neural network, which requires more training time. Therefore, neural networks is applied to the dataset after dimension reduction and clustering. The number of clusters is 4. The comparison of accuracy and efficiency with different operations is listed in Table 5. After dimension reduction, the cross-validation error increases and training errors decrease. At the same time, training time decreases since some non-determined attributes are removed. After dimension reduction and clustering, both cross validation error and training error increases. However, compared with dimension reduction only, the cross validation error decreases and training error increases. Therefore, dimension reduction can improve efficiency and decrease accuracy. While clustering will improve the accuracy and efficiency a little bit under this situation.

Table 5. Comparison of accuracy and efficiency after dimension reduction and clustering

	Cross validation error/%	Training error/%	Training time/Second
NN	14.4314	8.7843	6
InfoGain-NN	15.2941	8.7059	4.88
InfoGain-Kmeans-NN	14.7451	13.3333	4.73
InfoGain-EM-NN	14.7451	13.3333	4.75

6 Conclusion

In this paper, 5 supervised machine learning algorithms and 3 unsupervised machine learning algorithms are applied to the baseball dataset for classification of baseball players. Decision tree and Boosting algorithms achieve best performance. This is most likely because there are many nominal features in the dataset. Therefore, Decision tree can handle nominal features better than other algorithms. Using InfoGain to select attributes and reduce dimension can improve the efficiency though it will reduce the accuracy a little bit. Clustering will improve the accuracy and efficiency a little bit under this situation. To improve the accuracy of prediction in the future, we could add more features and make the dataset more balanced.

References

[1] Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, pages 3084–3092, 2013.

[2] Shane T. Jensen, Blakeley B. McShane, Abraham J. Wyner. Heirarchial Bayesian modeling of hitting performance in baseball. In *Bayesian Analysis*, Vol4, Number 4, 631-652

[3] Nikolai Yakovenko. Machine Learning for Baseball. https://medium.com/@Moscow25/machine-learning-for-baseball-my-story-84dbe075e4b1