NBA PLAYER PREDICTION UNDER SUPERVISED LEARNING

Cheng Huang, Mangyue Sun, Yue Cheng, and Zhongnan Su

ABSTRACT

The objective of this report is to display the method used to classify basketball players in terms of their skills and talents. Three classification algorithms will be discussed below, including k-nearest neighbor, random forest, and support vector machine. The task is a typical supervised learning example problem where the data is assumed to be separated under three labels.

1. INTRODUCTION

Basketball has always been one of the most popular sports all over the world, and NBA, the men pro basketball league in the US, has never lost its popularity internationally. The NBA players exhibit their talents and skills every game night, and their contribution to the teams is recorded extensively and comprehensively by the official stats people. The statistics that shows their contribution includes points, rebounds, assists, field-goal percentage, and turnovers etc. Although fans rank their favorite players via various aspects apart from the stats, it is the most accurate and objective way to differentiate players by their talents and skills. The goal of this project is to separate NBA players from 1991 to 2017 into 3 different classes according to the five positions they played or play. The reason for separating players into five positions before classifying the data is that different positions have different specific talents and gifts illustrated by their stats. The players are set to be classified into 3 groups labeled as 'Star Players', 'Good Players', and 'Mediocre Players'. The star players are the players who are rated 7.3 or above on their Win Share values, while players with values greater than 3 and less than or equal to 7.3 are deemed as 'Good Players'; Otherwise, players are 'Not Heard of'. Since the problem definition suggests a supervised learning task, the classification algorithm of K nearest neighbor(KNN), random forest, and support vector machine(SVM) will be discussed, and classification results will eventually be compared.

2. ALGORITHM DESCRIPTION

1. KNN(k–Nearest-Neighbor)

The k-Nearest-Neighbor Rule classifies x by assigning it the label most frequently represented among the k nearest samples; in other words, a decision is made by examining the labels on the k nearest neighbors and taking a vote.

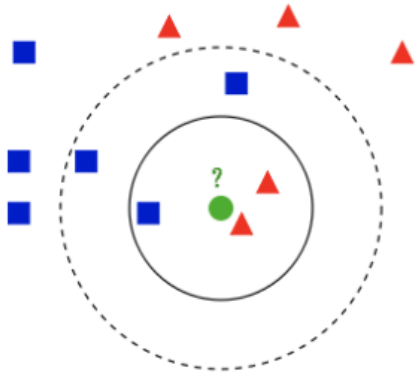

Figure 1 KNN theory

If k = 3 (solid line circle) the test sample (green circle) is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) green circle is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle).

Choice of k is very critical – A small value of k means that noise will have a higher influence on the result. A large value makes it computationally expensive and defeats the basic philosophy behind KNN (that points that are near might have similar densities or classes). A simple approach to select k is set k= n.

The nearest-neighbor classifier relies on a metric or "distance" function between patterns. the Euclidean formula for distance in d dimensions:

$$D(a,b) = (\sum_{k=1}^{d} (a_k - b_k)^2)^{\frac{1}{2}}$$

the Cosine similarity for comparison:

$$cos\ (\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

2. Support Vector Machine

1. fundamental method and theory of SVM

Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier and the main method for the SVMs algorithm, can be turned into a constrained optimization problem on the margin distance:

$$L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{j=1}^{l} a_i a_j y_i y_j (X_i \cdot X_j)$$

$$s.t. \sum_{i=1}^{l} a_i y_i = 0 \;\&\; a_i \geq 0$$

The claim is that this function will be maximized if we give non-zero values to a's that correspond to the support vectors, those that 'matter' in fixing the maximum width margin.

As our dataset is comprised of more than 10 features even after dimension reduction. Therefore, we choose SVM because it allows the original finite-dimensional space be mapped into a much higher-dimensional space, which makes the separation easier in that space. SVM relies on preprocessing the data to represent patterns in a high dimension. With an appropriate nonlinear mapping φ() to a sufficiently high dimension, data from two or three categories can always be separated by a hyperplane.

2. Kernel selection and parameter selection

A kernel method enables SVM to operate in a high-dimensional, *implicit* feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products over all pairs of data in the feature space. The most popular kernel functions using in SVM are as follows:

$$\mathrm{K}(\mathrm{X_i}, \mathrm{X_j}) = \begin{cases} X_i, X_j & Linear \\ (\gamma X_i \cdot X_j + C)^d & Polynomial \\ \exp\left(-\gamma |X_i - X_j|^2\right) & RBF \\ \tanh(\gamma X_1 \cdot X_j + C) & Sigmold \end{cases}$$

The effectiveness of SVM depends on the selection of kernel and the kernel's parameters, including gamma, cost coefficient, etc. For our project, each combination of parameter choices is checked using cross validation(CV), which is a standard technique for adjusting hyper parameters of predictive models. In K-fold CV, the available dataset S is divided into K subsets S1, ..., SK. Each subset will be randomly selected to be the test dataset, and the rest of the fold data will be the training dataset. Here in our project, we choose k equals to 10 and applied a 10-fold CV. Hence, the kernel parameters with best cross-validation accuracy are picked. The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters.

3. Random Forest

Random forests algorithm is an optimized machine learning method for classification, regression and other learning tasks compared to the traditional decision tree. By using the concept of bagging, choosing random feature to create several trees and taking each trees' classification result into account, the random forest have an unexcelled accuracy among current algorithms, and it can also give which feature is more important during the whole classification process.

During the classification process, each tree is constructed following these steps:

1. Suppose we have N training cases and M features are to take into consideration.
2. Choosing m (m<<M) features randomly to use for the construction of a single decision tree.
3. For each node of the tree, calculate the Gini impurity of each variable to decide the feature to choose for the best split. By taking all feature derived from the m random variables into account, each tree will be fully grown.
4. Calculate the best split of each individual tree.

When the forest is constructed, feature attributes of the predicting sample will be applied to each tree model among the existing random forest, and the prediction will be given by the average votes of all trees.

3. Simulation Results

1. PCA

Because the players' data we used ranges from 1991 to 2017, so it's a large capacity of data with more than 14000 rows and 36 columns (35 feature, 1 label)). We decided to use Win Share as labels to decide what types of players fall in to which levels (WS≤3: label=0 mediocre players,3＜ WS≤7.3: label=1 good players, WS ＞ 7.3:label=2 star players). If we directly use these data without reducing its dimension, it will cost us a lot of time. Thus, to find the

desired number of eigenvectors that can be used to form the Eigen-space, the percentages of original data versus data dimension is plotted.

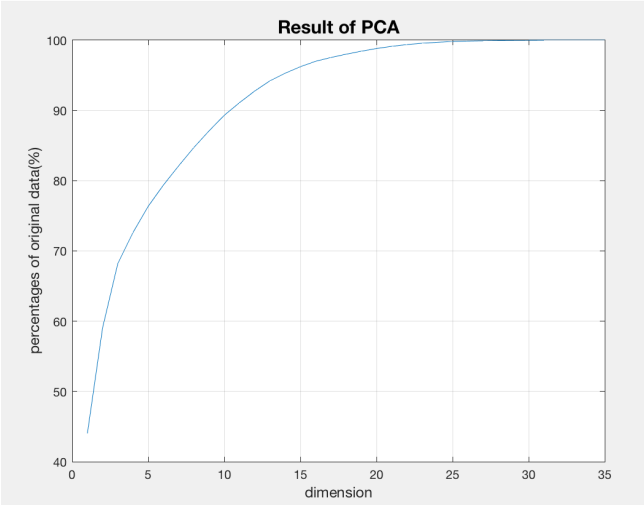

Figure 2 Result of PCA

According to the result, we find that when we reduce the dimension from 35 to 10, 90% of the original d ata is still accounted for. Therefore, considering the tradeoff between accuracy and time consumption, we decided to use 10-dimensional data for classification.

2. KNN Simulation Results:

1. Data Processing

The most important thing in kNN is to choose a suitable value of K. A small value of k means that the noise will have a higher influence on the result, and a large value makes it computationally expensive. As we mentioned before, a simple approach to select k is to set k= n. We have 14000+ samples, so we should set K=120. To choose a suitable value of k, we use cross validation to test K values and to adjust the accuracy.

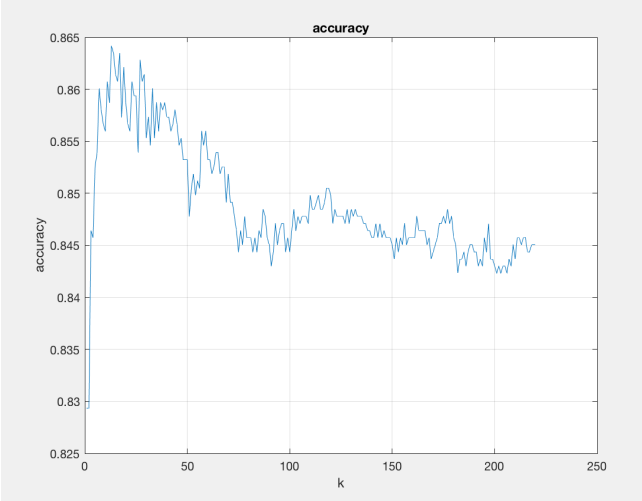

Figure 3 accuracy of different K value

We can see from the figure 3, the curve tends to be flattened out when k is greater than 80. Therefore, the value of K is decided to be 120, which is the square root of 14000.

2. Prediction

After we decided value of K, we select 50 row entries as test data. We put data into 'test_data.csv' and let the algorithm give us the result of classification. Then we compare with the true label to calculate accuracy.

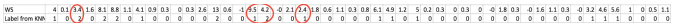

Figure 4 result of KNN

In figure 4, the first row is WS (WS≤3: label=0 mediocre players,3 < WS≤7.3: label=1 good players, WS > 7.3:label=2 star players. The second row is the classification result calculated by the KNN algorithm. Through all the forenamed results, we can see there are four incorrect results among 50 datasets. Therefore, the accuracy is 80%.

3. SVM Simulation Results

Among the matlab libraries being used to apply SVM, the libsvm (developed by Chih-Chung Chang and Chih-Jen L) is the most popular one. According to the parameter setting in libsvm, we had to choose the appropriate kernel to implement the prediction. We did the three-class classification based on four optional kernel functions given. As the accuracy of each kernel suggested, the SVM with RGB gave the best performance compared to others. Therefore, we choose RGB for the following training and testing.

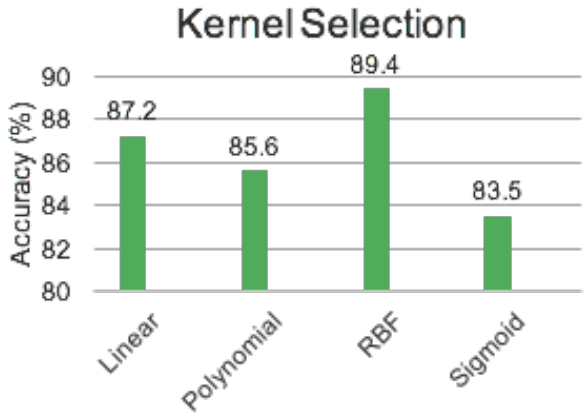

Figure 5 Accuracy of Kernel Functions

4. Random Forest Simulation Result

The RandomForestClassifier function in the scikit-learn toolbox can generate a random forest classifier for predicting the NBA player. By random choosing the 95% of the dataset as training set, the classifier can be well trained, and the trained classifier can give a chart showing the relationship between the accuracy and two parameters, the n_estimators and max_feature.

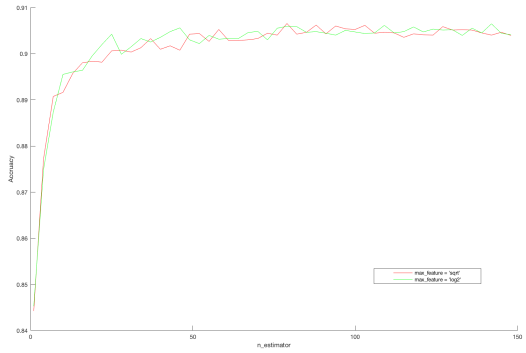

figure 6 Accuracy of different numbers of estimators

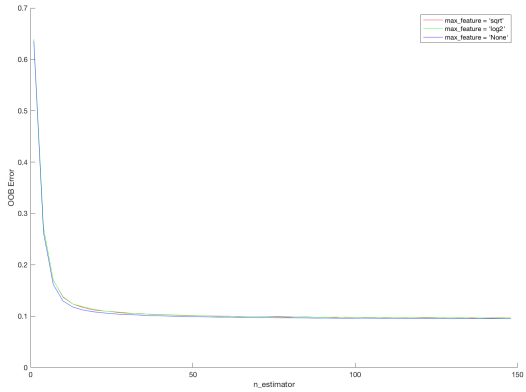

figure 7 Out-of-bag error rates

The n_estimator represents the number of decision trees in the forest, and the max_feature represents the split. Since there are only 35 features to describe a certain basketball player, there's little change when the max_feature parameter is altered.

The importance of each feature can also be generated thought the random forest classification. According to the result, the MP (Minute played), FG (Field Goal) and the overall points are the top 3 important features to consider the performance of the player.

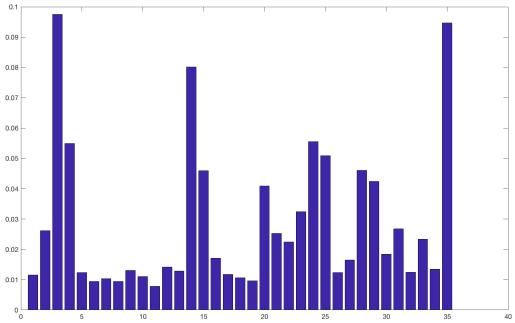

figure 8 Importance of each features

Random forest can also be used in regression, which can predict the winning share by the player's performance. The mean square error of test samples is 0.7.

4. Discussion

Based on the results we obtained by applying different classification algorithms to predict the performance of NBA players, we chose the accuracy of 10-fold cross validation as the final criteria to evaluate a certain algorithm. Compared to others, the random forest method has the best performance by using the concept of ensemble learning, which leads to better prediction result compared to any single classify system. Enlightened by the analysis and results, many other problems with large amount of features and historical data could also be solved by using such analytical method when requiring classification and prediction, such as stock price prediction and movie rating prediction.

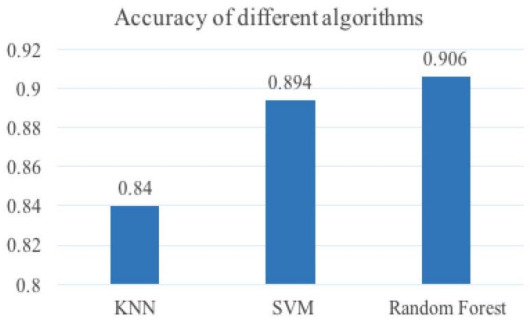

figure 9 Accuracy of different algorithms

5. CONCLUSION

This paper discusses some popular machine learning algorithms including PCA, KNN, SVM and Random Forest and concludes that using PCA with Random Forest will get the most accurate prediction result. Moreover, the Random Forest can also be used as regression which can predict the winning share of a certain NBA player. The classification be easily used to separate players into three group, which helps the GMs and player scouts decide which players will be the best fit for their teams. In addition, the algorithm can be used to classify college basketball players. Therefore, their performance can be well tracked well before the day of NBA Draft held each summer.

6. REFERENCE

[1] Joachims, Thorsten. "Text categorization with Support Vector Machines: Learning with many relevant features." European Conference on Machine Learning Springer, Berlin, Heidelberg, 1998:137-142.

[2] Durgesh, K. SRIVASTAVA, and B. Lekha. "Data classification using support vector machine." Journal of Theoretical and Applied Information Technology 12.1 (2010): 1-7.

[3] cn.mathworks.com. (2017). *Classify data using nearest neighbor method - MATLAB knnclassify - MathWorks United Kingdom*. [online] Available at: https://cn.mathworks.com/help/bioinfo/ref/knnclassify.html [Accessed 10 Dec. 2017].

[4] Scikit-learn.org. (2017). *scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation*. [online] Available at: http://scikit-learn.org/stable/index.html

[5] Ramón Díaz-Uriarte, and Sara Alvarez de Andrés. "Gene selection and classification of microarray data using random forest." *BMC Bioinformatics.* 7.1(2006):3.

[6] E. M. Mirkes, KNN and Potential Energy: applet. University of Leicester, 2011.