Predicting NBA Shots

Brett Meehan

bmeehan2@stanford.edu

Stanford University

Predicting

The purpose of this project was to predict the success of failure of a shot made by an NBA player based on information such as the shot distance, closest defender distance, time remaining on shot clock, etc. (See Data section). The machine learning algorithms I used for classification were logistic regression, SVMs, Naive Bayes, neural networks, random forests, and boosting. Algorithm such as logistic regression, SVMs, Naive Bayes, and neural networks performed relatively poorly on the dataset, while random forests and boosting performed better. Boosting was the most accurate algorithm used, with a maximum accuracy of 68% on the test set.

Data

My project's data came from Kaggle (link: kaggle.com/dansbecker/nba-shot-logs), and originally from the now-defunct NBA statistics API. Each row represents a shot attempt by a player from the 2014-2015 NBA season, along with information such as the shot outcome, distance data, and game data.

Raw features: GAME_ID, MATCHUP, LOCATION, W/L, FINAL_MARGIN, SHOT_NUMBER, PERIOD, GAME_CLOCK, SHOT_CLOCK, DRIBBLES, TOUCH_TIME, SHOT_DIST, PTS_TYPE, SHOT_RESULT, CLOSEST_DEFENDER, CLOSEST_DEFENDER_PLAYER_ID, CLOSE_DEF_DIST, FGM, PTS, player_name, player_id

Features

Most of the features I used as inputs to the algorithms were the same as the raw features. I removed "PTS"(points made) and "FGM"(field goals made) in every case since they were perfect predictors of whether a shot attempt succeeded or failed. For algorithms such as logistic regression, SVMs, and neural networks, I removed categorical data since I believed it would perform poorly with these types of models and better with models that use decision trees such as random forests and boosting. I also sometimes created a linear time variable that counts the total elapsed game time instead of using the categorical "PERIOD" and downward-counting "GAME_CLOCK" variables.

Models

MODEL 1: Logistic Regression

Logistic regression with the response variable in {-1,1} was one of the simpler models I used. I performed gradient descent on the following loss function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\log(1+e^{-y^{(i)}\theta^T x^{(i)}})$$

MODEL 2: SVM

I used an SVM to generate more latent features in order to try to deal with high bias. The optimization for the SVM can be described by the following function:

$$\min_{\gamma,w,b}\frac{1}{2}\|w\|^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)}+b)\geq 1, i=1,\ldots,m$$

MODEL 3: Neural Networks

I used two 1-layer neural networks in this project, one with sigmoid activations in the hidden layer, and one with RELU activations in the hidden layer. Since the output activation was the sigmoid function, both networks performed backpropagation on the log loss function:

$$\frac{1}{m}\sum_{i=1}^{m}-y^{(i)}\log p^{(i)}-(1-y^{(i)})\log(1-p^{(i)})$$

MODEL 4: Naive Bayes

I used multinomial Naive Bayes with and without binning continuous variables. To make a prediction for an example in the test set, I used Bayes' rule:

$$p(y=1|x)=\frac{p(x|y=1)p(y=1)}{p(x)}$$

MODEL 5: Random Forest

The random forest algorithm I used was simply Matlab's TreeBagger function with 80 trees.

MODEL 6: Boosting

In my boosting setup, I used the XGBoost algorithm created by Tianqi Chen with 1 estimator.

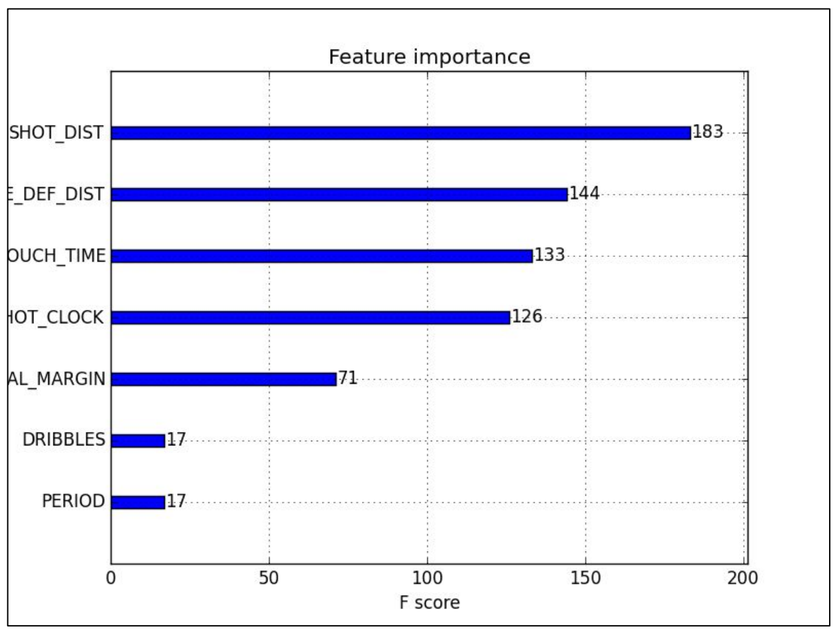
Results

Boosting, with an accuracy of 68%, gave the optimal results on the test set. Random forests came in second place, with an accuracy of 61%.

	Training set size	Test set size	Training Error	Test Error
Logistic Regression	40000	5000	0.40	0.41
SVM	40000	5000	0.43	0.45
Sigmoid-only neural network	100000	5000	0.44	0.45
Sigmoid/RELU neural network	100000	5000	0.47	0.45
Naive Bayes	100000	5000	0.43	0.46
Random Forest	100000	5000	0.37	0.39
Boosting	100000	6404	0.31	0.32

Discussion

Algorithms like logistic regression, SVMs, and neural networks performed poorly in my tests. I believe that this poor performance is due to the fact that I had to remove some categorical data for these algorithms, as well as the limited information inherent in this dataset. Naive Bayes performed poorly most likely due to the limitations of the Naive Bayes assumption, as well as the simplicity of the model. In my milestone, I predicted that random forests would be the optimal classification method due to their natural handling of categorical variables. Random forests turned out to be the second best method of classification, second only to boosting. Boosting has the added advantage of being able to reduce bias. Overall, ensemble decision tree methods performed best on this challenging dataset with limited information.



Feature importance for XGBoost algorithm.

Future

If I had more time, I would try another machine learning algorithm that was apparently used with success in a similar problem by Wright et al[3].: factorization machines. I would also try to gather more spatial data since, as Chang et al.[1] demonstrate, there appears to be valuable information in the two-dimensional coordinates of players on the court. In addition, the most important features for boosting appear to be spatial, as shown in the figure above. Based on Wang and Zemel's[2] success in classifying offensive plays using neural networks on pure player coordinate data, I would then experiment with more complex neural networks on this new spatial data.

References

1. **Y. Chang, R. Maheswaran, J. Su, S. Kwok, T. Levy, A. Wexler, K. Squire. Quantifying Shot Quality in the NBA. *MIT Sloan Sports Analytics Conference*, 2014.**
2. **K. Wang and R. Zemel. Classifying NBA Offensive Plays Using Neural Networks. *MIT Sloan Sports Analytics Conference*, 2016.**
3. **R. Wright, J. Silva, and I. Kaynar-Kabul. Shot Recommender for NBA Coaches. *KDD Workshop on Large-Scale Sports Analytics*, 2016.**