Analyzing the Dataset of Kobe Bryant Shot Selection

Mingyi Xue*
School of Chemistry and Chemical Engineering, Nanjing University

Wangqian Miao
Kuang Yaming Hornors School, Biophysics, Nanjing University

Rui Wang
School of Business, Nanjing University

Instructor: Prof. Cho-Jui Hsieh
Department of Computer Science & Statistics, University of California, Davis

Abstract

In the real world application of machine learning, it is always difficult to apply the algorithms from books definitely. Most of the time, we find that our machine learning method does not work well without data pretreatment and feature engineering.

Using 20 years of data on Kobe Bryant's swishes and misses in NBA, we will predict which shots can find the bottom of the net. By this dataset from Kaggle, we practice feature engineering with classification basic techniques like logistic regression , support vector machine and neural networks, and the up-to-date techniques like XGBoost. At last, we give the conclusion of our job concretely.

1 Introduction

When applying machine learning methods in the real world application, we always pretreat dataset, remove outliers, fill in missing values, create dummy variables, split dataset into train/dev/test sets and so on before we use the algorithms, especially in the case of that the dataset contains a lot of catogorical variables. However, nowadays, some algorithms can save us a lot time from feature engineering (for example, XGBoost method).

This dataset describes more than 30000 shots in Kobe's long career in Lakers, including different kinds of information about the shots and the game. Our task is to make the classification (the shot scores or not) as accurate as possible. In this paper, firstly, we transform the dataset into dataframe and then mine the most important information from the dataset through the plots. As a first try, we remove redundancy of multiple features to create a new dataset and then apply basic machine learning methods including logistic regression, SVM, neural networks on the set. Next, we tend to do more feature engineering with PCA and XGBoost to make our data matrix contain more accurate and focused infomation. At last, we will compare different algorithms on the updated data matrix.

2 Data Summary and Data Preprocessing

2.1 Data Summary

Kobe dataset has 30697 rows, 22 variables, among which `shot_made_flag` is the result whether he made the shot or not. Table 1 gives us a brief review of the features in our dataset. It is obvious

*Three authors are all exchange students from Nanjing University.

that there exists so many catogorical variables in our dataset and that is why our task is so difficult.

Name	**Variable Kind**	**Name**	**Variable Kind**
`action_type`	category	`seconds_remaining`	int64
`combined_shot_type`	category	`shot_distance`	int64
`game_event_id`	category	`shot_made_flag`	response
`game_id`	category	`shot_type`	category
`lat`	float64	`shot_zone_area`	category
`loc_x`	int64	`shot_zone_basic`	category
`loc_y`	int64	`shot_zone_range`	category
`lon`	float64	`team_name`	category
`minutes_remaining`	int64	`matchup`	category
`period`	int64	`opponent`	category
`playoffs`	category	`season`	category

Table 1: A summary table for the variables

There are 5000 rows with missing feature `shot_made_flag` because these rows are acted as test set on Kaggle competition. Since we needn't know the labels of these 5000 rows, we picked the rest 25697 rows and split them randomly into 70/30 as training set and test set.

At a glance of the dataset, it is easy to find that most of the variables are categorical, so we take a further step to see the number of categories in each variable and whether different categories have a significant effect on the average probability of $Y = 1$ (Kobe made the shot) in Figure 1. Information on categorical variables are listed as follows.

Name	**Number of categories**	**Name**	**Number of categories**
`action_type`	57	`shot_zone_area`	6
`shot_zone_basic`	7	`opponent`	33
`shot_zone_range`	5	`combined_shot_type`	6
`team_name`	1	`season`	20
`matchup`	74	`shot_type`	2

Table 2: A summary table for the categorical variables

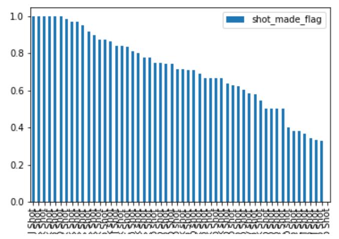
We can conclude from tables and plots above that the average probability of shot score varies in different categories and categories are in an acceptable number. So creating dummy variables (or one hot matrix) for each variable after feature engineering is doable.

2.2 Data Preparation and Preprocessing

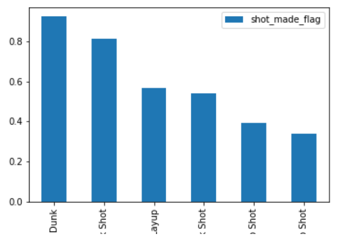
In the firsthand, `shot_id`, `game_event_id` and `game_id` are all ordinal, subjective features created by people, so we removed them.

Secondly, we checked whether `lat/lon` contain almost the same infomation as `loc_x/loc_y` since they both recorded infomation on position.

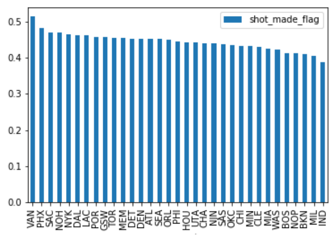
It is obvious from the plot above that `lat` has a perfect negative linear relationship with `loc_y` and `lon` has a perfect positive linear relationship with `loc_x`, so we can remove features `lat` and `lon` from the dataset.



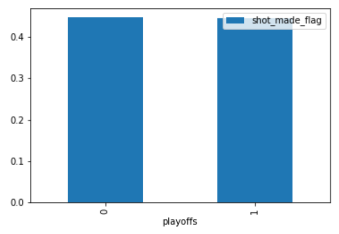
(a) action type



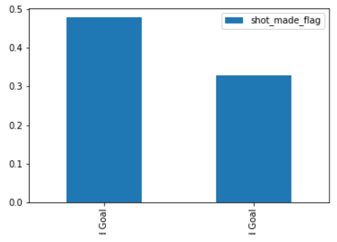
(b) combined shot type



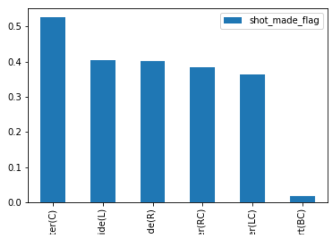
(c) opponent



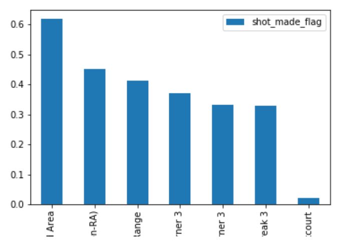
(d) playoffs



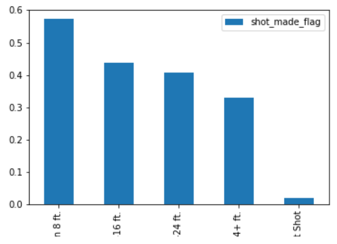
(e) shot type



(f) shot zone area

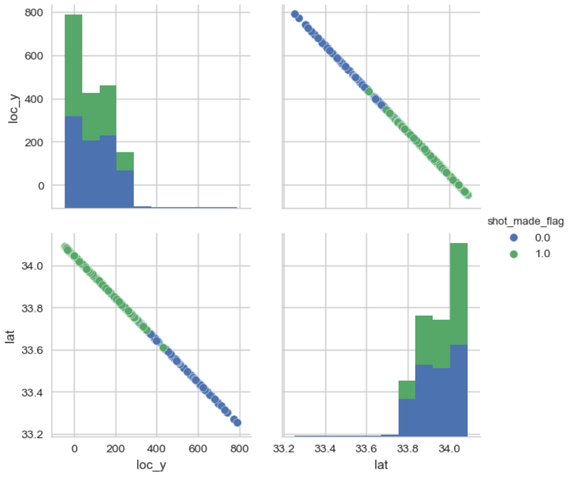


(g) shot zone basic

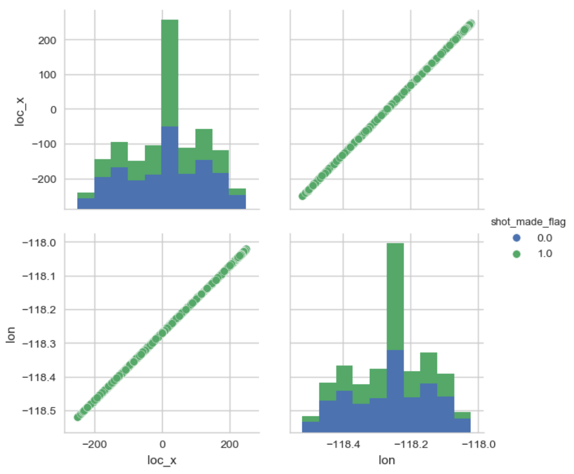


(h) shot zone range

Figure 1: Barplots of mean value of categories



(a) Correlation between `lat` and `loc_y`



(b) Correlation between `lon` and `loc_x`

Figure 2: Correlation between location and x,y

Then we noticed that game time both in large scale (date) or small scale (time) is an important factor in the dataset. In order to simplify data structure, we broke the feature `game_date` into three orthogonal features `game_year`, `game_month` and `game_day`. Besides, we combined `minutes_remaining` and `seconds_remaining` into `timeRemaining`.

Furthermore, we plot the scatter plots to see how `loc_x/loc_y` can indicate zone features of the basketball field. It is safe to remove all indomation about zone, including `shot_zone_area`, `shot_zone_basic` and `shot_zone_range` according to the plot below.

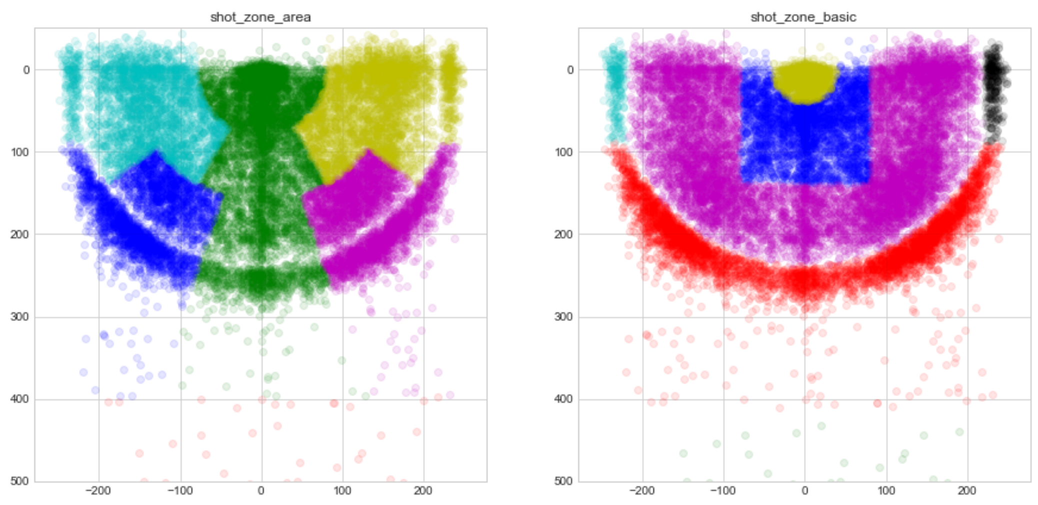


Figure 3: Plot of shot zones (different colors represent different regions)

Due to Kobe Bryant's staying in the same team for his all 20 years, we can remove `team_id` and `team_name` since these two features are consistent in the whole dataset.

Last but not least, it is interesting to find that matchup contains the infomation of opponent but more than that. We broke `matchup` down to `opponent` and `homeGame`, which indicate whether a game is hosted at home or away.

In conclusion, we created dummy variables for all categorical features and got totally 117 features including labels of `shot_made_flag`.

3 Basic Supervised Learning Methods

After we clean the data and do the basic preprocessing, we will apply some basic classification algorithms. In this section, the behaviour of logistic regression, Support Vector Machine and Neural Networks will be discussed when we choose different parameters, kernel functions, and depth of our Nerual Networks.

3.1 Logistic Regression

Firstly, Logistic Regression is the basic classification algorithm in machine learning, especially useful for binomial classification. We try to find a linear relationship between $\text{logit}(\pi)$ and explanatory variables, where $\text{logit}(\pi) = \ln(\frac{\pi}{1-\pi})$ and π is the probability of $Y = 1$.

In Logistic Regression, there are several important parameters such as how we choose the cutoff value (in common, we choose 0.5 to tell the difference between 0 and 1), the regularization term and the parameters for the regularization term.

The value of cutoff is quite significant when we use logistic regression. Different cutoffs will give us different error matrixes. As shown in Figure 4, authors find the best cutoff is 0.55.

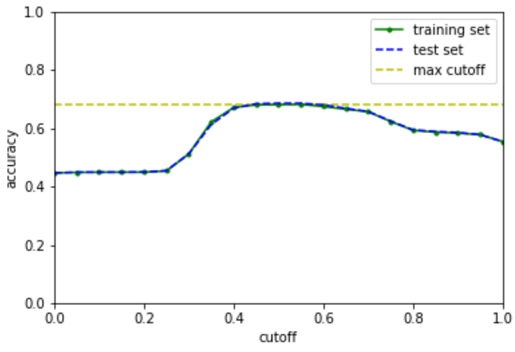
We fixed the cutoff value and changed the hyperparameters for the regularization term in this problem. Results are as follows.

Cutoff Value	**Regularization**	λ	**Train Accuracy**	**Test Accuracy**
0.50[1]	ℓ_2	1.0	44.65%	44.47%
0.55[2]	ℓ_2	1.0	68.35%	68.50%
0.55	ℓ_2	10.0	63.44%	64.10%
0.55	ℓ_1	1.0	67.43%	66.43%

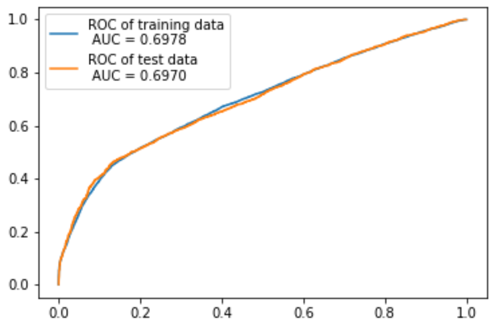
[1] Implemented by authors

[2] Implemented with `sklearn.linear_model.LogisticRegression`

Table 3: Table for Logistic Regression behaviour



(a) Plot of accuracy vs. cutoff



(b) ROC

Figure 4: The behaviour of Logistic Regression

From the experiment above we had some basic observation as following.

- The cutoff value will significantly influence the behaviour of Logistic Regression and in this expriment, the best cutoff is 0.55. However, there may be overfitting because the test accuracy is higher than train accuracy.
- ℓ_2 regularization and $\lambda = 1$ is a good choice for this problem.
- AUC is between $(0.60, 0.70)$ and the Logistic Regression model is moderately fit for this task.

3.2 Support Vector Machine

SVM is a fast and dependable classification algorithm that performs very well with a limited amount of data and that is why we choose it as our second algorithm.

In this task we just ran different kernel functions with the help of sklearn due to the authors are not familiar with the SVM algorithm (We just use SVM for reference) and test the performance of SVM on our dataset. The results are shown in Table 4.

It is obvious that the linear kernel gives us the best results which is close to the accuracy of Logistic Regression.

Kernel	**Time consuming**(s)	**Train Accuracy**	**Test accuracy**
`rbf`	91.83	67.30%	67.98%
`linear`	74.96	68.07%	68.46%
`poly`	85.56	55.40%	55.31%

Table 4: Table for SVM behaviour

3.3 Neural Networks

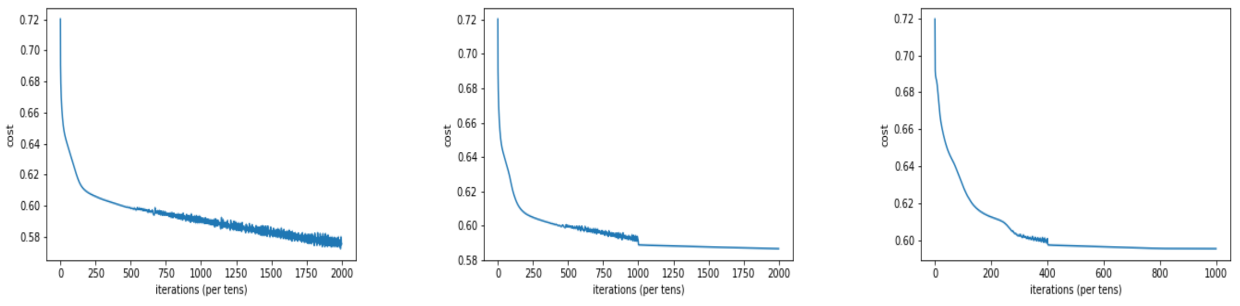
Nowadays, Neural Network is a popular way to realize classification, especially in computer vision. In this task, we hope to use Neural Networks to find whether there exists some non-linear pattern in our data.

With the help of tensorflow, we can build our own neural networks easily. However, so many hyperparameters exist in neural networks, like learning rate, mini-batch size, number of layers and neurons in each layer, activation fuction and so on. We do some tests to see how they will affect our models' behaviour.

In this problem, we firstly transform the 0/1 label to one hot matrix. We applied ReLU to hidden layers and softmax to output layers. Mini-batch size equals the whole dataset which means we used normal gradient descent algorithm, and the initial learning rate (denoted as α) is 0.1.

Depth	**Neurons**	**iterations**	**decrease in α**	**Train Accuracy**	**Test Accuracy**
3	[25,12,2]	10000	no	69.83%	65.84%
3	[25,12,2]	10000	yes	69.18%	68.65%
5	[32,16,8,4,2]	5000	yes	69.00%	67.84%

Table 5: Table for Neural Networks behaviour



(a) No decrease in α (b) $\alpha \times 0.1$ every 5000 iteration (c) $\alpha \times 0.1$ every 2000 iteration

Figure 5: Importance of decrease in learning rate

Here are several findings.

- A decrease in learning rate is important in neural network because large steps will lead to increasing oscillation as iteration goes on.
- For this dataset, since the amount of data is not large enough, there seems no significant difference in training effects for shallow or deep neural networks.
- Usually deeper neural networks need more time to complete each iteration (That's why our attempt to train a 10-layer neural network failed due to limited time and computing resources).

- We prefer to train the 3-layer network with more iterations to reach higher accuracy in a shoter time.

3.4 Algorithm Comparison

As shown above, we applied three different classification algorithms. In Table 6, we comprare the model performace of them.

Algorithm	**Time consuming**(s)	**Train Accuracy**	**Test Accuracy**
Logistic Regression	0.5305	68.35%	68.50%
SVM	74.96	68.07%	68.46%
Neural Networks	1183.87	69.18%	68.65%

Table 6: Algorithms compaison

The conclusion is as following.

- Three different methods give us almost the same accuracy.
- Neural Networks take much more time than other algorithms.
- It seems that the non-linear pattern in our data is not obvious.

4 Dimension Reduction and Feature Engineering

Now, our dataset contains 117 features (including dummy variables) which may contain some trivial information, for example, some categories just have a couple rows and the truth is that the data matrix is a large sparce matrix (We have so many dummy variables). Dimension reduction or important feature selection may help us know more about the most important features in the dataset. In this section, we apply PCA which is unsupervised and XGBoost which is supervised to deal with the problem.

4.1 Dimension Reduction with PCA

Principal Component Analysis (PCA) is a statistical procedure that extracts the most important features of a dataset.

Firstly, we normalized each column of the 117-feature dataset to mean 0, standard deviation of 1 normal distribution by $\frac{X-\bar{X}}{SE\{X\}}$. Then we computed the correlation matrix of the normalized dataset which is denoted as $\boldsymbol{C}$. Furthermore, we applied sklearn.decomposition.PCA on the matrix $\boldsymbol{C}$ to find columns in the normalized data which can explain 99.9% of the total variance. Finally, 4 columns remained as the compacted dataset and we applied the same classification algorithm discussioned above on it.

Algorithm	**Time consuming**(s)	**Train Accuracy**	**Test Accuracy**
Logistic Regression	–	–	57.78%
SVM	86.74	–	59.88%
Neural Networks	468.92	60.48%	60.11%

Table 7: Algorithms comparisons after PCA

- Though the accuracy of compacted data is around 60%, which is a little far away from the accuracy of 68% of the whole dataset, taking the fact of only 4 columns into consideration, these columns have contributed to half of the probability gain in prediction compared to the 50% probability abtained by guessing based on a coin-flip.
- PCA is efficient in data compact, but there is still a problem in this project that most of the variables are dummy variables which is binomially distributed and cannot be normalized in the same way as numeric variables. Instead, standardized pearson residuals are recmmended for those dummy variables.

4.2 Feature Engineering with XGBoost

XGBoost is short for "Extreme Gradient Boosting", where the term "Gradient Boosting" is proposed in the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. XGBoost is based on this original model which is applied by many data scientists in Kaggle to realize feature engineering. XGBoost is an ensemble method and realized by C++ and OpenMP. We get the final results with 4 thread and 500 iterations in just few seconds.

Through XGBoost, we can find the interesting features in our dataset as shown in the following figure and then pick them up to make our model more specific and accurate.

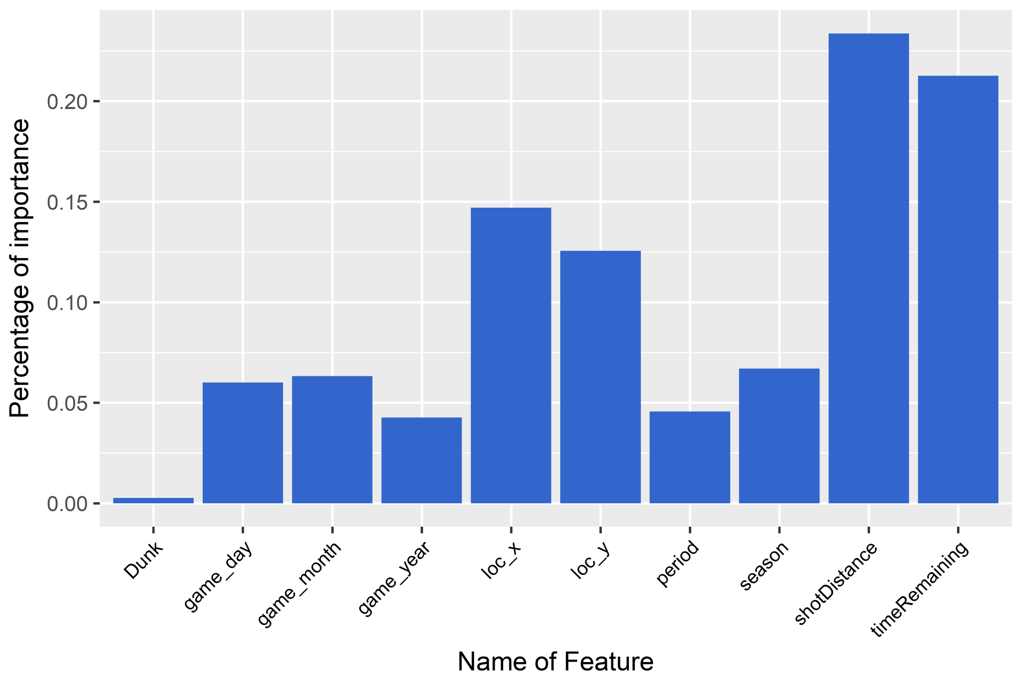


Figure 6: Ten most important features

XGBoost gives us a more smaller and denser data matrix and we applied the algorithms on the new dataset again, as shown in Table 8. It seems that the 10 most important features explains the most important behaviour of our dataset.

Algorithm	**Time consuming**(s)	**Train Accuracy**	**Test Accuracy**
Logistic Regression	0.2325	60.32%	61.23%
SVM	19.22	58.90%	58.91%
Neural Networks	273.42	60.41%	60.92%

Table 8: Algorithms comparison after XGBoost

5 Conclusion

One important characteristic of our dataset is that the amount of data is not big enough to behave differently with diffenrent classification algorithms. That's a major reason of why logistic regression sometimes even works better than neural networks and why XGBoost does not seem to revolutionize prediction of the dataset a lot.

We suppose to get vote results from three basic classification prediction and expect to see an improvement in accuracy as a followup step.

6 Acknowledgement

This work is supervised by Prof. Cho-Jui Hsieh who gives us a bunch of useful advice. The authors learn a lot from the following websites and tutorials. Without these generous people, we cannot finish this difficult job. Thanks a lot!

1. Prof. Cho-Jui Hsieh's lecture notes on STA141C.
2. Andrew Ng's course on Coursera: Deep Learning Specialization.
3. Scikit-learn tutorial: `http://scikit-learn.org/stable/documentation.html`.
4. Kaggle competition discussion: `https://www.kaggle.com/c/kobe-bryant-shot-selection`.
5. XGBoost tutorial: `https://xgboost.readthedocs.io/en/latest/R-package/index.html`.