

# Kobe Bryant Shot Selection

*Kevin Siswandi*

*March 3, 2017*

## Introduction

This report analysed the Kaggle playground challenge: [Kobe Bryant's shot selection](#). The data contained the location and circumstances of every field goal attempted by Kobe Bryant during his 20-year career. The task was to predict whether the basket went in (shot\_made\_flag). Kaggle had removed 5000 of the shot\_made\_flags (represented as missing values in the csv file). These were the test set shots for which one must submit a prediction.

Various models were run for this project:

1. XGBOOST in Python
2. XGBOOST in R
3. TensorFLOW DNN in Python
4. sklearn (random forest, logistic regression, linear discriminant analysis, K-NN, decision tree, naive bayes, extra tree, adaboost, and gbm) in Python

This analysis resulted in a score of 0.60126 LB (99th out of 1117 teams on the leaderboard). The script that produced the final score had been hosted on Kaggle kernel: <https://www.kaggle.com/kevins/kobe-bryant-shot-selection/fork-of-notebook55e4f6ba6a>

Submission and Description	Private Score	Public Score	Use for Final Score
<b>Fork of Notebook55e4f6ba6a</b> (version 17/17) 9 hours ago by <a href="#">kesis</a> From "Fork of Notebook55e4f6ba6a" Script		0.60126	<input type="checkbox"/>

Figure 1: Final submission score

96	new	asdasd		0.60125	15	9mo
97	new	dikdoogi		0.60125	5	9mo
98	new	Suwon & Chungkeun	 	0.60125	16	9mo
99	 160	ShuaiWang		0.60128	25	9mo
100	new	asdfasd		0.60130	3	9mo

Figure 2: This score would have taken over the 99th position on the leaderboard

## Main scripts

The script that was used to generate the final score had been hosted on Kaggle kernel: <https://www.kaggle.com/kevins/kobe-bryant-shot-selection/fork-of-notebook55e4f6ba6a>

Other scripts in `code` directory:

- `xgboost.R` (ran locally) – basic xgboost in R.
- `show-me-your-best-model.ipynb` (ran on Kaggle kernel) – a forked [public notebook](#) containing exploratory data analysis, feature selection, and various sklearn models.
- `models_war.py` (ran on Kaggle kernel) – various sklearn models (made into a script from “show-me-your-best-model.ipynb”).
- `tensorflow.ipynb` (ran using GCP docker) – tensorflow deep neural network classifier.
- `temporal_spatial_eda.ipynb` (ran on Kaggle kernel) – a forked [public notebook](#) containing exploratory data analysis results related to Kobe’s psychology into the game.

## Feature selection

Based on insights gleaned from the exploratory analysis of the data, I decided that the following features should be dropped:

- `shot_id` (used as index, not included as a feature)
- `team_name` (only one category)
- `team_id` (only one category)
- `game_event_id` (unique within a game, not related to shots made)
- `game_id` (redundant information already contained in `matchup/opponent`)
- `lon` (correlated with `loc_x`)
- `lat` (correlated with `loc_y`)

## Model selection

For model selection, I built on [a public script](#) that explored multiple models in scikit-learn including Logistic Regression, Linear Discriminant Analysis, K-Nearest-Neighbor, Decision Tree, Naive Bayes, Random Forest, Extra Trees, Adaboost, and GBM. Interestingly, the models performed quite well on cross-validation (~0.6xx), but when I submitted the output, the results were very poor (~1.0 logloss).

At first, I thought it was the feature selection process that introduced the overfitting (as it involved selecting top 20 features according to Random Forest Classifier and Recursive Feature Elimination using Logistic Regression Classifier), but it became evident that this overfitting had to do with the [data leakage](#) after I reproduced [a similar score](#) even after removing the automated feature selection process.

Therefore, I had to use another strategy for this challenge: XGBOOST was the best next thing to do. Using [this starter script: XGBOOST in R](#) as a starting template, I modified the feature selection and used one-hot-encoding to handle the categorical variables (c.f. `xgboost.R`) and managed to get ~0.607 logloss (local CV and LB). Furthermore, doing my own feature selection and using one-hot encoding for the categorical variables (the starter script simply converted the categorical variables to ordered integers) gave me 0.606 on LB!

Using tensorflow in Python (c.f. `tensorflow.ipynb`), I also managed to achieve ~0.66 logloss on LB, but the training time was too expensive as it exceeded Kaggle’s kernel limit with only 4 hidden layers (~50 neurons each) so I had to use GCP docker on my laptop.

Due to the accuracy and efficiency of XGBOOST, I concluded that the best strategy for this competition would be to do advanced feature engineering and feed the engineered features to XGBOOST. Using this strategy, I managed to improve my LB score from 0.606 (basic XGBOOST) to ~0.604 then ~0.603 with progressively more features added. It eventually reached 0.601 with even more features and parameter tuning.

## Feature Engineering

### Time remaining

- combined `seconds_remaining` and `minutes_remaining` into a single variable `time_remaining`
- created a flag indicating whether it's last 5 seconds of the period (it has been shown that Kobe's last-minute shots tend to be more desperate)
- created two additional variables, `seconds_from_period_start` and `seconds_from_game_start`, from combination of the variable `time_remaining` and the variable `period`.

### Home game vs Away game

- created a flag of whether it's home game for LA Lakers. This was done using the information from `matchup`: it's home game if it was written like "LA vs X", away if it was like "LA @ X".

### Extraction of temporal features

- extracted the year and month of game date because this may provide some seasonal information on Kobe's performance.
- extracted the day of the week and the day of the year. Note: As day of week and time of year are cyclical, I experimented by converting them to radian and take the sine/cosine values (c.f. `last_shot_feature.ipynb`); but these were not included in the final `xgboost` model because this radial transformation did not improve `xgboost`'s performance).

### Level reduction of categorical variables

- replaced the 20 least-common action types with 'Other'.

### One-hot encoding

- converted each unique level of categorical variables (both raw and engineered) into a column of binary (0/1).

In all instances of the models (`sk-learn`, `xgboost`, and `tensorflow`), the following (raw) categorical variables were converted to columns of 0/1 with one-hot encoding:

1. `action_type`
2. `combined_shot_type`
3. `period`
4. `shot_type`
5. `shot_zone_area`
6. `shot_zone_basic`
7. `shot_zone_range`
8. `opponent`

### Shot location clusters

- used Gaussian mixture models to cluster the locations of shot attempts, based on `loc_x` and `loc_y`. Following `temporal_spatial_eda.ipynb`, 13 clusters were specified for the clustering. However, this shot location cluster was not included in the final `XGBOOST` model because it did not improve the final score.

## Last shot status

- created a flag on whether the last shot that Kobe attempted was made or missed (cf `last_shot_feature.ipynb`). This feature did not improved the final score of xgboost.

## Future work

This prediction could be improved in many ways given the time and computational resources. Aside from hyperparameter tuning of the xgboost model (due to time constraint I had to skip the grid search process and resort to hand-tuning), there were other potential things to try:

- feature interaction using tensorflow's `cross_column`
- hyperparameter tuning of tensorflow's deep NN
- ensemble of xgboost and DNNclassifier
- exploring the data leak

## Conclusion

A single model XGBoost model without much feature engineering performed very well. XGBoost was really efficient in identifying non-linear interactions between features that not much feature engineering was required to get a good score (in fact several features that I generated ended up not used because they did not improve the final score by XGBoost). Using a single XGBoost model with feature engineering and hand-tuned parameters, a final score of  $\sim 0.601$  was achieved on the leaderboard, which translated to the 99th position out of 1117 teams (top 10%).

## References

I would like to credit the discussion forums and public kernels in this competition; some notebooks in the public kernels would take significant effort to produce – without them it wouldn't be possible for me to produce the results in this report within such a short time.