Introduction to Machine Learning: Basketball project

Giacomo Davide[1], Lorenzo Tomada[2], and Giovanni Tracogna[3]

[1] problem statement ***, solution design ***, solution development ***, data gathering *, writing ***
[2] problem statement ***, solution design ***, solution development ***, data gathering *, writing ***
[3] problem statement ***, solution design ***, solution development ***, data gathering *, writing ***

Course of AA 2023-2024 - Mathematics (SM34)

1 Problem statement

Sport analytics provides insights into player performance, team strategies and game dynamics, helping teams optimize their performance. In particular, our goal is to predict the number of three-point field goals (also called *triples* or *three-pointers*) scored by a given player in a game.

Our knowledge of basketball suggests that the highly situational nature of this parameter can hardly be explicitly captured by a human-designed scheme. Therefore, the aim of this report is to design a machine learning system that provides a reasonable solution to the problem.

The problem statement then reads as follows: given an NBA player and a game, predict the number of three-point field goals scored by the player in that game.

Formally, we define $X = X_1 \times X_2 = \{(x_1, x_2) : x_1 \text{ is a player}, x_2 \text{ is a game}\}$ and $Y = \mathbb{R}^+$ [1]; we then want to learn an $f_{\text{predict}} : X \to Y$.

However, in order to make f_{learn} and f_{predict} executable on a machine, we introduce some pre-processing steps (see Subsection 1.2), meaning that we employ some $f_{\text{pre-proc}} : X \to X'$ which allows to learn an $f_{\text{predict}} : X' \to Y$. Here X' is obtained after our feature engineering procedure[2].

[1] We could possibly round the prediction to obtain an integer result, as the number of three-pointers which can be scored is a non-negative integer. In that case, $Y = \mathbb{N}$.

[2] In this work we considered feature engineering as a part of the problem statement and not of the design of the ML system, since X is not formed by objects which are completely digital (consistently with what was done in the lecture notes [1], slide 56/366).

1.1 Data exploration

The dataset at our disposal contains information about NBA games played in a timespan ranging from 2004 to 2019, including both team-related stats (e.g. the final NBA ranking per season) and player-related stats (e.g. the number of fouls committed per game, the minutes played and the number of baskets scored).

In particular, one of the columns shows the number of three-point field goals scored by each player during each game (the variable we aim to predict). The huge size of the dataset (≈ 500000 rows) provides us with plenty of examples, suggesting the use of supervised machine learning. Thus, we are dealing with a supervised regression problem.

The ML system we designed is described in section 2.

1.2 Feature engineering

We design $f_{\text{pre-proc}} : X \to X'$, where $f_{\text{pre-proc}}(x_1, x_2) = x' = (x'_1, \ldots, x'_8) \in X'$, in which

i) x'_1 is the ID associated with the player x_1;
ii) x'_2 is the ID associated with the team in which x_1 plays;
iii) x'_3 is the ID associated with the opposing team of x'_2 during the game x_2;
iv) x'_4 is the date difference (in days) between x_2 and the previous game played by x_1;
v) x'_5 is the season in which x_2 is played;
vi) x'_6 is the winrate of x_1 taking into account the three games preceding x_2;
vii) x'_7 is the average number of three-pointers scored by x_1 in all the games preceding x_2;
viii) x'_8 is the location in which x_2 is played (home or away, from the perspective of x_1).

All the numerical ID's are converted to strings to preserve their unordered nature, thus $X' = (A^*)^3 \times \mathbb{N}^2 \times [0, 1] \times [0, +\infty) \times \{\text{home}, \text{away}\}$, where $A =$ UTF-16 and A^* denotes the powerset with duplicates.

The choice of these particular features was led by the fact that all this data can easily be extracted from the dataset at our disposal. Moreover, all these features are widely used in real-life scenarios in order to predict the outcome of a game, suggesting that they could also be meaningful in the prediction of player-related stats.

Since all the information needed to perform predictions is available before the game is played, no information leakage is caused by our choice of features.

1.3 Assessment and performance indices

In this work, the main axis of assessment is effectiveness, with some hints of explainability.

The error indices we use to measure effectiveness are Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Square Error (RMSE)[3].

Regarding explainability, we value the degree to which we can identify strong predictors (if present).

2 Proposed solution

We implement two different learning techniques, namely Random Forest (RF) and k-Nearest Neighbors (kNN), to later on pick the one performing better in terms of effectiveness and use it to draw our conclusions after the experimental evaluation.

In particular, for both techniques, the learning phase is preceded by hyperparameter tuning to further improve the effectiveness. During this step, we adopt MSE as efficiency index for the comparison. This choice is arbitrary but justified by its widespread use in similar scenarios. The learning-test division procedure which is used is the same introduced in Subsection 3.1.

Regarding RF, the hyparameter values which we tune using grid search are the number of trees n_{trees} and the number of variables retained by each tree p. We selected the subsets $P_1' = \{50, 100, 500\}$ and $P_2' = \{2, 3, 4\}$ as ranges containing the default values $500 \in P_1'$ and $\lceil \sqrt{p} \rceil = \lceil \frac{1}{3} p \rceil = 3 \in P_2'$.

As for kNN, we consider the number of neighbours k and the distance used on X'.[4] We define $\tilde{P}_1' = \{1, 5, 10, 50, 150\}$ and $\tilde{P}_2' = \{\text{Manhattan, Euclidean}\}$ for the same reason.

The resulting hyperparameters are the ones used to assess the two learning techniques.

3 Experimental evaluation

3.1 Data and procedure

Regarding the methodology used for assessment, we employed cross-fold validation (CV) with $k = 10$ folds because of its robustness with respect to data and to test the generalization ability of our techniques.

To complete the evaluation procedure, we compared our results to a baseline, namely the dummy regressor, following the criteria introduced in section 1.3. Recall that the dummy regressor, independently on the input $x' \in X'$, always returns the average of the values of three-pointers scored (computed with respect to the learning set).

[3] Notice that Mean Absolute Percentage Error (MAPE) cannot be used here as it leads to divisons by zero.

[4] Note that, due to scikit-learn technicalities, we are forced to convert all categorical values to numbers. This has been done using scikit-learn's label encoder since one-hot encoding would be too computationally demanding.

3.2 Results

Table 1 and Figure 1 show the numerical results of the evaluation procedure described in Subsection 3.1.

Model	Mean			Standard deviation		
	MAE	**MSE**	**RMSE**	**MAE**	**MSE**	**RMSE**
RF	0.664	1.050	1.024	0.002	0.013	0.006
kNN	0.683	1.077	1.038	0.004	0.011	0.005
Dummy Regressor	0.947	1.557	1.248	0.003	0.019	0.008

Table 1: Values for the assessment indices of our techniques.

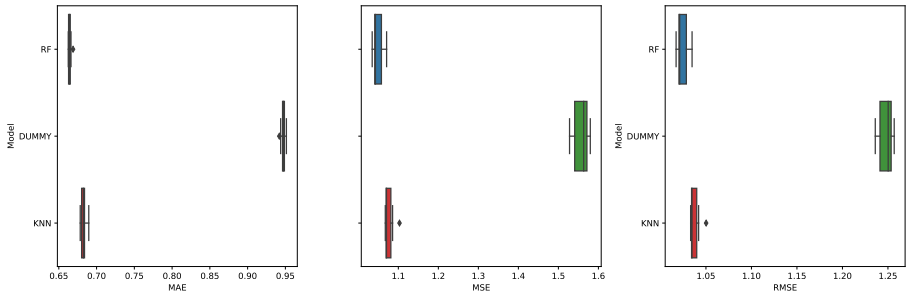

Figure 1: Boxplots of the error indices

According to all our error indices, kNN and RF largely outperform the Dummy Regressor in terms of effectiveness, with RF being slightly better than kNN.

Overall, RF proves to be better as it also guarantees a decent level of explainability. Indeed, mean RSS decrease[5] highlights the predictive value of average of triples (variable x'_7), as shown in Table 2.

Statistic	**Player's team**	**Player**	**Opposing team**	**Date difference**	**Winrate**	**Average triples**	**Location**	**Season**
Mean	0.055922	0.091323	0.148123	0.058041	0.049076	0.505702	0.028663	0.063151
Std. dev.	0.000176	0.000193	0.000219	0.000287	0.000368	0.000476	0.000133	0.000261

Table 2: Mean RSS decrease.

In conclusion, we are reasonably satisfied with the degree to which our techniques were able to capture the behaviour of the real system in terms of effectiveness and (concerning RF) explainability.

[5]The choice of mean RSS decrease instead of feature ablation descends from technical limitations of scikit-learn. Notice also that the scikit-learn utility devoted to this computation internally scales the values in such a way that they add up to 1.

References

[1] Eric Medvet. *Introduction to Machine Learning and Evolutionary Robotics*. 2023-2024.