Introduction to Machine Learning project: Predicting NBA games' outcomes

Enrico Malcapi[1], Gabriel Costanzo[2], and Francesco Speciale[3]

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

Course of AA 2022-2023 - Data Science and Scientific Computing

1 Problem statement

The aim of our project is to build a ML model to predict the outcome of a basketball match from the NBA regular season.
Assuming a dependence between the outcome of the match and the statistics scored by the two competitors in the previous games, the learning phase should give a model that will predict the winner of the match from an input that considers the recent statistics of both teams. Defining $\vec{x_0}$ and $\vec{x_1}$ the statistics of the home team and away team, our problem can be considered as a binary classification with the outcome prediction $y = 1$ if the home team wins , $y = 0$ otherwise.
Since our dataset is provided with the labels (the outcome of a match) we decided to use supervised machine learning and since we are facing a binary classification problem we used as assessment the AUC index as in most of the procedures.

2 Designing phase

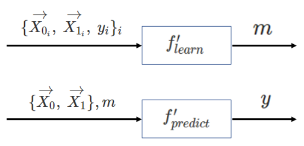
2.1 Data gathering and pre-processing

Data were downloaded from Kaggle's repository "NBA Games Data" by Nathan Lauga[1]. We collected all games outcomes (a binary categorical variable, "home team wins") and the raw statistics scored by every player of every team in all regular season games from the beginning of the 2008/2009 season (the first one for which data are complete and consistent) until the end of 2018/2019 one, before the COVID-19 pandemic disrupted the course of the following season. The dataset is quite balanced, as the percentage of the home team who won

games is around 59%.
Consulting the famous basketball statistics site "Basketball Reference"[3] and the blog "Hack a stat"[4], we selected and built a series of advanced and more meaningful team statistics by aggregating those of the individual players in each game. In order to highlight the contribution to the result of the most important players, some statistics were computed by aggregating the stats of only the 6 most used players (a representation of the five starters plus the first substitute plus the so-called sixth man[6]). In addition to these advanced stats, winning percentages both overall (last 40 games), home games only (last 20), away games only (last 20), and a binary variable set to 1 if the match is a back-to-back (second game played in last 48 hours, it could affect team's fatigue) were added as the last variables.
The final model we realised has 34 numerical inputs and a categorical output "home team wins". The inputs represent the differences between the averages of each statistic calculated for the home team, in the last 40 games before the one considered, and the averages of the same statistics for the away team. We used differences instead of keeping the stats for both teams because we wanted to reduce the complexity of the task, after a computationally-heavy pre-processing and we believed that what really matters to win a match is performing better than your opponent rather than performing well or badly in absolute terms.
The choice of a time-window of 40 is given to the singular structure of the league and its calendar. We thought it reasonable to assume a 40-game window over which to calculate averages, because we believe this is a good approximation of the shortest series of games for all teams to have faced each other at least once and have proved their consistency in a balanced series of matches. We made this choice also when the two teams have just played $GP < 40$ games in the current season. In that case, the remaining $40 - GP$ most recent matches from the previous season are taken into account. In the calculation of the averages and the winning percentages, the original numbers are weighted according to their distance from the match in question, so that the current shape of the teams at the time of the match is still considered. The weights are linearly equi-spaced from 0.75 (for the 40th most recent game) to 1 (last game) to keep the importance of all the games and avoid game-fixtures dependency too much.

2.2 Learning Techniques

With the definitions adopted in the Problem statement section, our learning technique can be represented by the scheme in Figure 1. Since we are facing a binary classification problem, using the accuracy would be inappropriate and in order to use an index that is free from the choice of the threshold, we decided to use the AUC index for the assessment. We chose to compare 4 different learning techniques (Random Forest, Tree, Support Vector Machine,k-nearest-neighbor) and searched for the best parameters with a grid-search. Once found the best parameters for each learning technique, we decided to compare the predictions using a 10-Cross validation in order to evaluate the AUC. Since for evaluating the AUC index you need a probabilistic output for the SVM, we used an implemented method that evaluates a probabilistic output from SVM using statistical techniques[2].



$\{\overrightarrow{X_{0_i}}, \overrightarrow{X_{1_i}}, y_i\}_i$
f'_{learn}
m
$\{\overrightarrow{X_0}, \overrightarrow{X_1}\}, m$
$f'_{predict}$
y

Figure 1: Representation of the learning technique

3 Implementation

3.1 Hyperparameter Tuning

For each learning technique, we evaluate a grid search in order to find the best parameters as follows:

- Tree: we compared 25 values of n_{min} from a range $[500, 3000]$ with a step of 100 and the two different criteria Gini and cross-entropy using cross-validation with 5 folders and the AUC index for the assessment. The range for the n_{min} parameter was chosen by inspecting the error on the test dataset for different values of such parameter.
- KNN: we compared 16 values of K from a range $[200, 1700]$ with a step of 100 and the two different metrics Manhattan and Euclidean using cross-validation with 5 folders and the AUC index for assessment. The range for K was chosen in the same way as the tree.
- Random Forest: we compared 5 values for n_tree (100,400,600,700,900) and the two different criteria Gini and cross-entropy using cross-validation with 5 folders and the AUC index for the assessment;
- SVM: we compared 5 values of C (0.1,1,10,100) with values of γ (1,0.1,0.001,0.0001) using the standard Gaussian Kernel and cross-validation with 5 folders. For the assessment index in the grid search we didn't use the AUC because producing a probabilistic output has a very high time cost. So, we decided to go for balanced accuracy as assessment index.

Results are reported in Table 1.

3.2 Assessment

After we found the best parameter with the grid search, we evaluated the AUC index in a 10-Cross Validation for each model and then reported the mean and standard deviation(Table 1). Moreover, we visualized a box plot of such results(Figure 2).

4 Results

Model	Grid search results	Assessment results
SVM	$C = 1, \ \gamma = 0.001$	$AUC = 0.686 \pm 0.012$
Random forest	$n_{tree} = 900$, imp.= cross-entropy	$AUC = 0.710 \pm 0.009$
KNN	$n_{neighbors} = 1400$, metric = Manhat.	$AUC = 0.712 \pm 0.009$
Decision Tree	$n_{min} = 1800$, imp. = cross-entropy	$AUC = 0.705 \pm 0.011$

Table 1: Result of the hyperparameter tuning

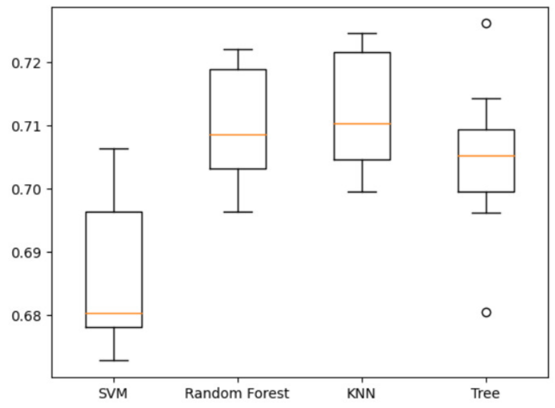


Figure 2: Box plot of the AUC index of the 4 learning techniques

As we can see from the table and the boxplot, KNN is the best learning technique in terms of effectiveness, with the best AUC overall. However, it has also the most expensive prediction. Furthermore, Random Forest effectiveness is really close to KNN's with much more efficiency. Therefore, considering that this ML system would be used for the prediction of future matches and that this would require a not inconsiderable amount of data processing for each match (it would be necessary to collect and aggregate data from teams' last 40 games as explained previously) we believe that the use of Random Forest is preferable in order not to further burden the system.
For Random Forest we also evaluated the following indexes: $FPR = 0.53 \pm 0.02, TPR = 0.80 \pm 0.01, Acc = 0.67 \pm 0.01$. These results suggest increasing classifier's threshold to 0.585 to get an EER of 0.36. Even though this results could seem poor, they're not far from the state of the art. NBA games' outcome is a highly randomic event [5].

References

[1] Nathan Lauga. *NBA Games*. URL: https://www.kaggle.com/datasets/nathanlauga/nba-games.

[2] John C. Platt. "Probabilistic outputs for SVMs and comparisons to regularized likelihood methods". In: *Microsoft Research* (1999).

[3] Basketball reference. *Glossary*. URL: https://www.basketball-reference.com/about/glossary.html.

[4] Hack a stat. *Learn a stat section*. URL: https://hackastat.eu/learn-a-stat/.

[5] Josh Weiner. *Predicting the outcome of NBA games with Machine Learning*. URL: https://towardsdatascience.com/predicting-the-outcome-of-nba-games-with-machine-learning-a810bb768f20.

[6] Wikipedia. *Sixth man*. URL: https://en.wikipedia.org/wiki/Sixth_man.