Machine Learning applied to NBA games data

Hugo Cornet, Pierre-Emmanuel Diot, Guillaume Le Halper, Djawed Mancer

Contents

Introduction

In the field of sports, predictive analysis has played an important role for the past two decades. Some say the starting point has been the team's analytical, evidence-based approach of the baseball manager Billy Beane whose goal was to assemble a competitive baseball team despite a small budget. At the same time, machine learning has become a powerful tool to predict various outcomes. That is why we choose the kaggle data set related to the results of all NBA games from 2014 to 2018.

Our goal

The objective of our project is to predict the odds of winning for each team given the games' features. The target variable used in the models is `Win`.

Our method

We trained several machine learning models, then compared their accuracy in order to keep the one which best predicts the outcome of the NBA games.

The NBA database

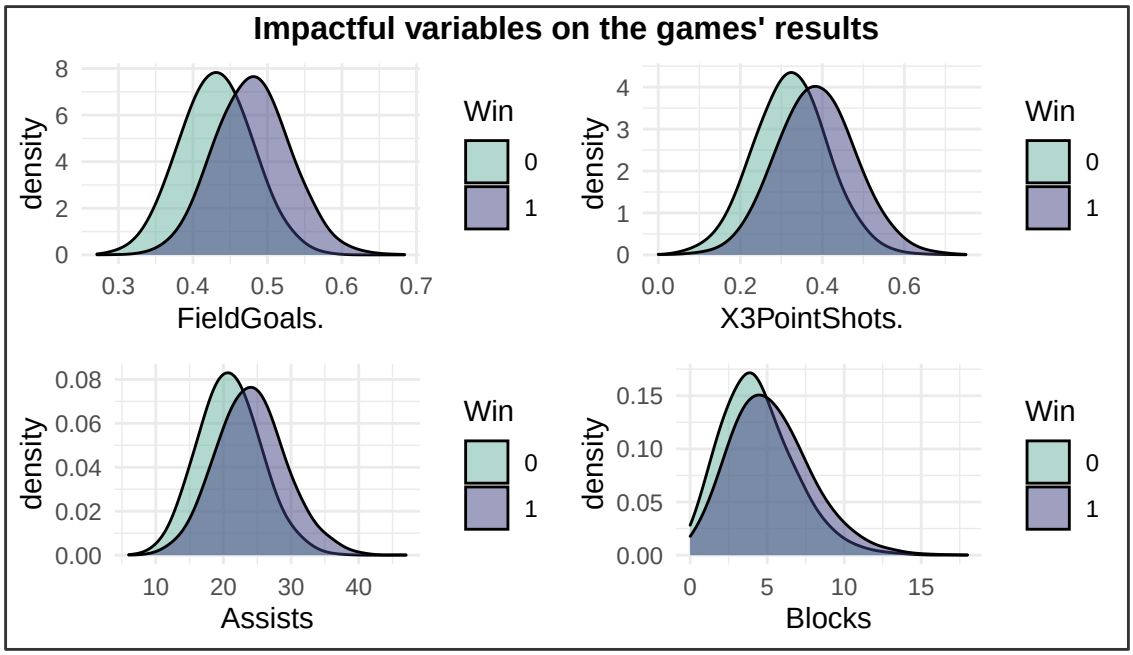
Our database was initially composed of 41 variables which were strongly linked together. The very first step is to remove the `TeamPoints` and `OpponenentPoints` variables to ensure predictions are not too trivial. Thanks to a brief descriptive analysis we decide to remove the correlated variables in order solely to keep the core information. For instance, there were both the variables `FieldGoals` and `FieldGoalsAttempted` plus the ratio between the two latter namely `FieldGoals.`. We decide only to preserve the ratio variable and we apply the same process to each feature. We have the same set of predictors for both teams and opponents. The new database is now composed of 24 variables and 9840 observations.

Table 1: Overview of the NBA data set

Date	Game	Team	Opponent	Home	Win	FieldGoals.	X3PointShots.	FreeThrows.
2014-10-29	1	ATL	TOR	0	0	0.50	0.59	0.53
2014-11-01	2	ATL	IND	1	1	0.51	0.35	0.76
2014-11-05	3	ATL	SAS	0	0	0.41	0.32	0.73
2014-11-07	4	ATL	CHO	0	0	0.46	0.39	0.77
2014-11-08	5	ATL	NYK	1	1	0.41	0.41	0.78
2014-11-10	6	ATL	NYK	0	1	0.38	0.37	0.96

Descriptive statistics

Here are the top four features which influence the most the probability of winning an NBA game. Emerges a Loss/Win dichotomy from the features represented below. These variables are most likely to be good predictors for machine learning models.



Train/test split

In the data set each row is duplicated because information about each game pops up twice. For example, you can find the following two observations which are linked to the same game.

Table 2: An example of duplicated rows in the database

	Date	Game	Team	Opponent	Home	Win	FieldGoals.	Opp.FieldGoals.
51	2015-02-06	51	ATL	GSW	1	1	0.49	0.46
786	2015-02-06	48	GSW	ATL	0	0	0.46	0.49

The main stake here is to prevent one pair of observations to be divided during the train/test split step. If it is the case, the model will be tested on data it has already seen during the training process. The method we use to avoid this issue is to randomly select 70% of the dates we have in our data. Then we build the train data set using these 70% dates and the test data set is associated to the remaining dates.

Complete data set

This database is made of all the variables we keep and will be used for decision trees, random forests, bagging and boosting.

Reduced data set

Performance Index Rating (PIR) is a basketball statistical formula that is used to measure one team's efficiency during games. It is the difference between positive and negative games' statistics.

$$\text{PIR} = (\text{TeamPoints} + \text{TotalRebounds} + \text{Assists} + \text{Steals} + \text{Blocks} + \text{OppTotalFouls}) -$$
$$(\text{FieldGoalsAttempted} - \text{FieldGoals} + \text{FreeThrowsAttempted} - \text{FreeThrows} + \text{Turnovers} + \text{OppBlocks} + \text{TotalFouls})$$

We use all of these features to build both the `PIR` and `OppPIR` variables which lead us to construct a reduced data set. The latter shall be employed for the same models as the complete database.

Table 3: Overview of the training reduced data set

Date	Game	Team	Opponent	Home	Win	PIR	OppPIR	TeamGroup	OppGroup
2014-10-29	1	ATL	TOR	0	0	108	131	3	4
2014-11-01	2	ATL	IND	1	1	128	88	3	2
2014-11-05	3	ATL	SAS	0	0	85	122	3	4
2014-11-07	4	ATL	CHO	0	0	111	142	3	3
2014-11-08	5	ATL	NYK	1	1	119	95	3	1
2014-11-10	6	ATL	NYK	0	1	107	74	3	1

Data set obtained with PCA

Since we want to perform LDA, QDA and Logistic regression and given the amount of predictors and the collinearity between them, we find it relevant to process principal component analysis on our data. We then go from 22 quantitative variables to 8 principal axes which summarize the main information contained in the variables.

Table 4: Overview of the training data set obtained with PCA

Date	Game	Team	Opponent	Home	Win	TeamGroup	OppGroup	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8
2014-10-29	1	ATL	TOR	0	0	3	4	-0.45	-1.78	0.69	-0.52	2.23	1.38	2.34	0.30
2014-11-01	2	ATL	IND	1	1	3	2	0.76	-1.08	0.56	2.23	-0.23	-0.63	0.16	0.90
2014-11-05	3	ATL	SAS	0	0	3	4	-0.82	0.63	1.45	2.13	-0.85	1.08	0.04	0.00
2014-11-07	4	ATL	CHO	0	0	3	3	0.31	0.42	2.24	1.32	0.64	-1.61	2.88	0.84
2014-11-08	5	ATL	NYK	1	1	3	1	-0.33	1.30	-0.83	1.68	-1.26	-0.51	1.22	0.46
2014-11-10	6	ATL	NYK	0	1	3	1	-0.21	-0.05	-0.28	0.51	-0.13	-1.46	-0.72	-0.31

LDA, QDA, logistic regression

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are two classification techniques which aim at building a projected discriminant axis that will separate the classes of a categorical variable. The major difference between these two classifiers is LDA relies on the strong assumption that the predictors have the same variance-covariance matrix in each class of the target variable, which is not the case for QDA. The selection between LDA and QDA is linked to the bias/variance trade-off since LDA is less variable than QDA but has a more important bias.

As for logistic regression, it is part of the generalized linear models family and is used to predict a binary dependent variable by means of quantitative and/or categorical predictors. It relies on the maximum likelihood estimator whose goal is to find the parameters that best fit the data.

In our case, the target variable will be `Win` and the predictors the eight principal components obtained by PCA. In that way, the variables' collinearity issue can be avoided which is useful when training LDA, QDA and logistic regression models. That is why the three models are trained on the PCA data, then tested on the test data expressed in the PCA change-of-basis matrix.

Probabilistic approach

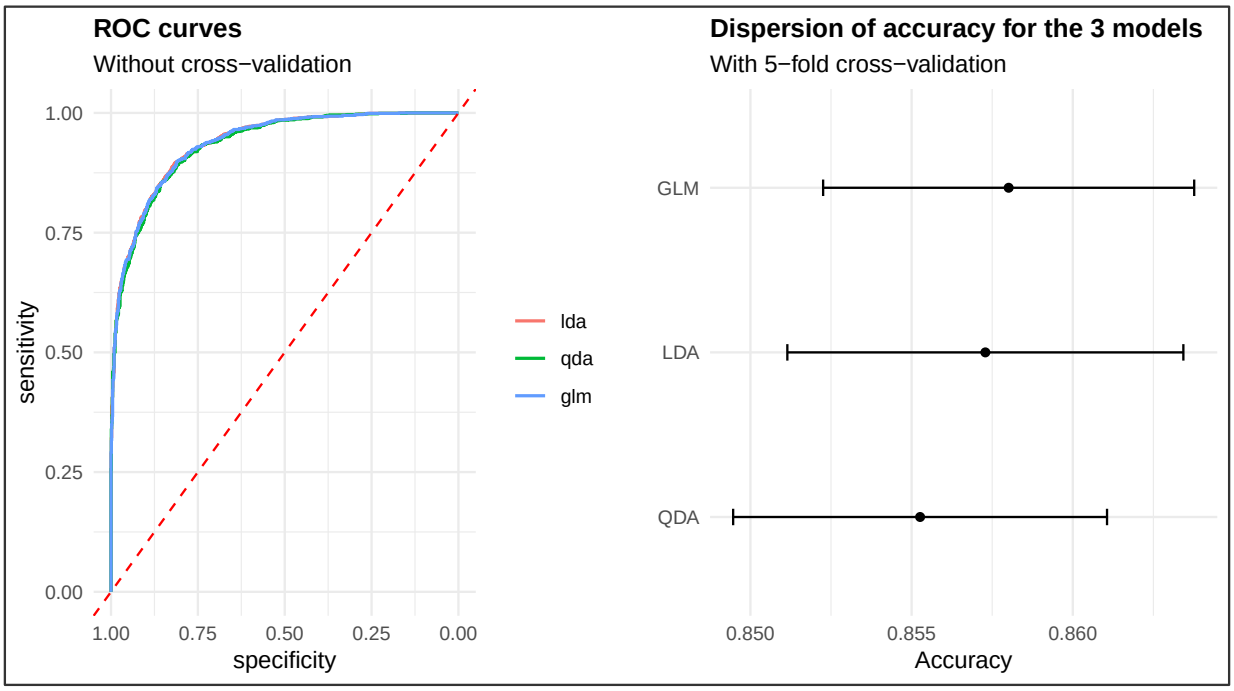
The first step of our method is to compare the predictions we can get with LDA, QDA and logistic regression. On the one hand we perform three models using the classic train/test split process with a 70%-train set. On the other hand we process a 5-fold cross-validation in order to get more accurate results.

The following table and plots show how close the results are. Cross-validation has slightly improved the accuracy of the three models but not in a significant way. The best model we obtain is the cross-validated logistic regression and it will be applied on the test data set.

Table 5: Predicition global error (in percent)

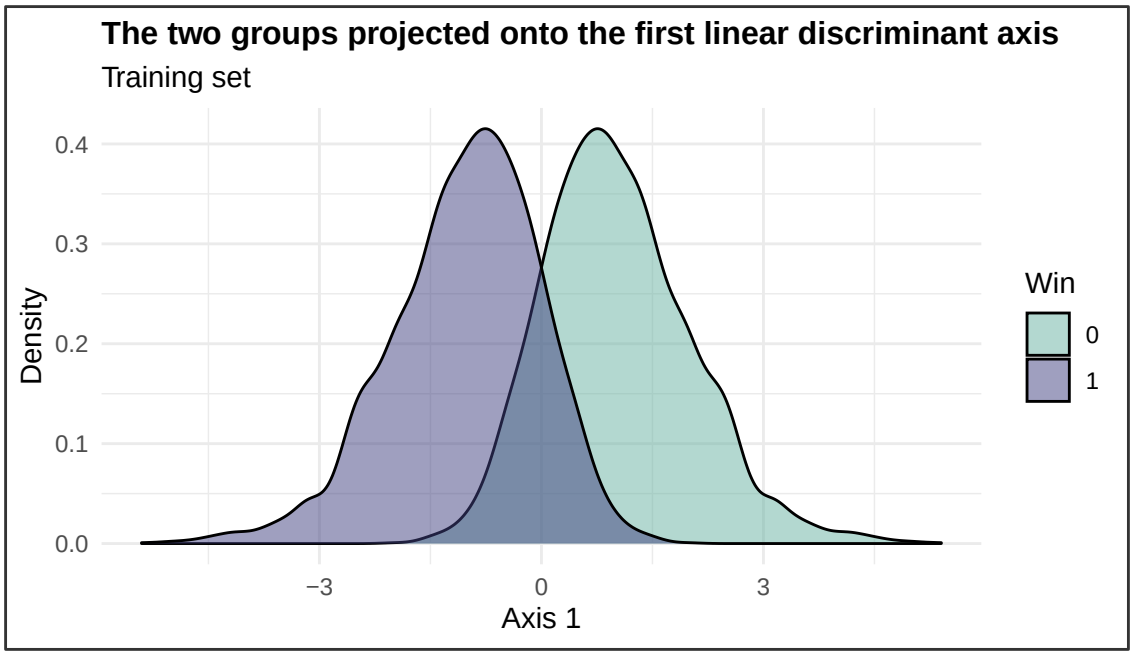
	LDA	QDA	GLM
Without cv	14.49	14.58	14.66
With 5-fold cv	14.42	14.55	14.26

The ROC curves displayed below illustrate how badly the trained models are close in terms of prediction. The second plot confirms that the cross-validated logistic regression is the best model.



Geometric approach for LDA

The second stage of our prediction process is to carry out a linear discriminant analysis by hand. The goal is to visualize how the linear discriminant ought to separate the two modalities of the `Win` variable, namely "0" or "1". This linear axis is computed by maximizing the inertia between the "0" and "1" classes (inter-class inertia). It helps to get two distinct groups as does our data. Once the axis identified, the natural rule of affectation is to determine the euclidian distance between an observation from the test data and the center of gravity of each group projected onto this axis. Finally the new point is assigned in the group in where the distance is the smallest.



When we test the geometric LDA model on the test data, we realize it performs better than the probabilistic LDA model, meaning the geometric affectation rule works better for our data than the probabilistic one.

Table 6: Two ways of performing LDA (test error)

	Probabilistic	Geometric
% error	14.49	13.61

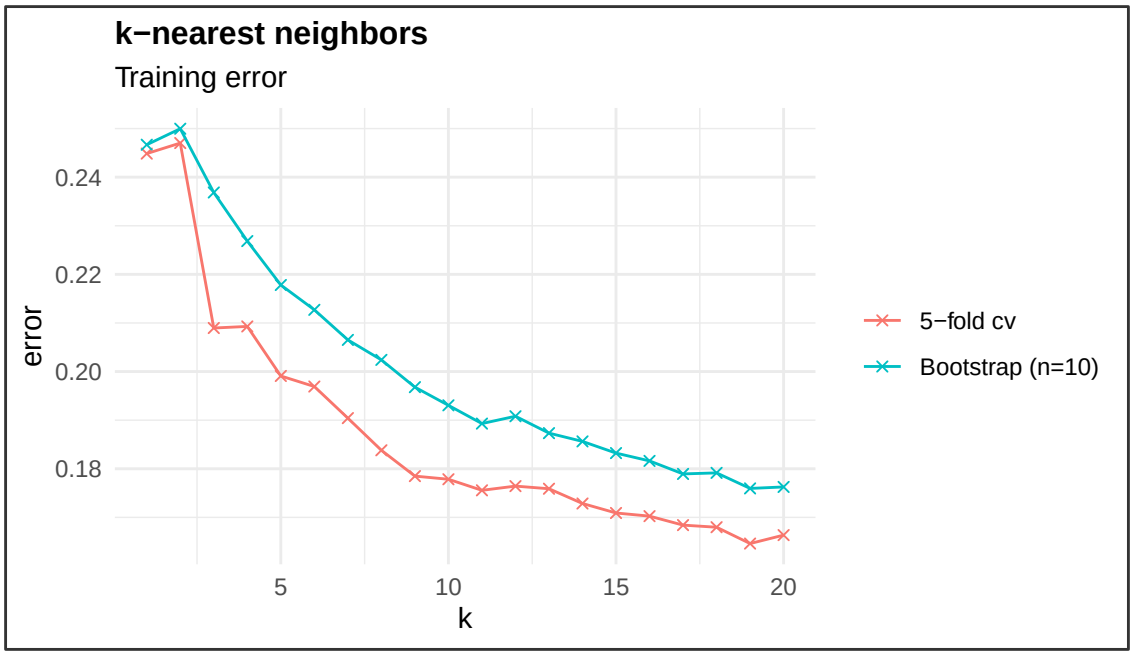
k-nearest neighbors

k-nearest neighbors (k-nn) is a "non-parametric" standard classification algorithm that estimates the class of a new data by looking at the majority class of the k closest neighboring data.

Bearing in mind that the only parameter to set is k - the number of neighbors to consider - it is relevant to tune this parameter using cross-validation and bootstrap. After having set the maximum number of neighbors at 20, both methods identify 19 as the optimal value of k. Once again the k-nn classifier is trained on PCA data as it is very sensible to variable correlation.

Table 7: Best k-nn models with cross-validation and bootstrap (training set)

	k	error	dispersion
5-fold cv	19	0.16	0.02
Bootstrap (n=10)	19	0.18	0.00



We will test this 19-nearest neighbors model on the test data set at the end of the study so as to pick the best model out of all models.

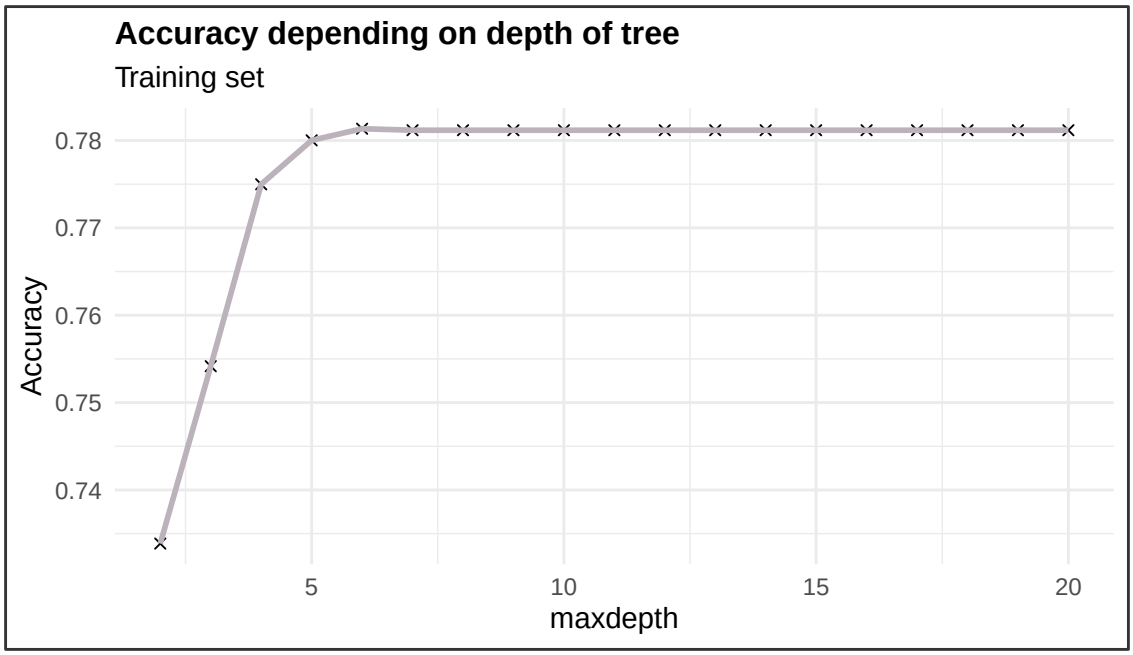
Decision trees

A decision tree is a machine learning algorithm which segments the predictor space into a number of high dimensional rectangles. A classification tree assigns each observation to the most commonly occurring class of training.

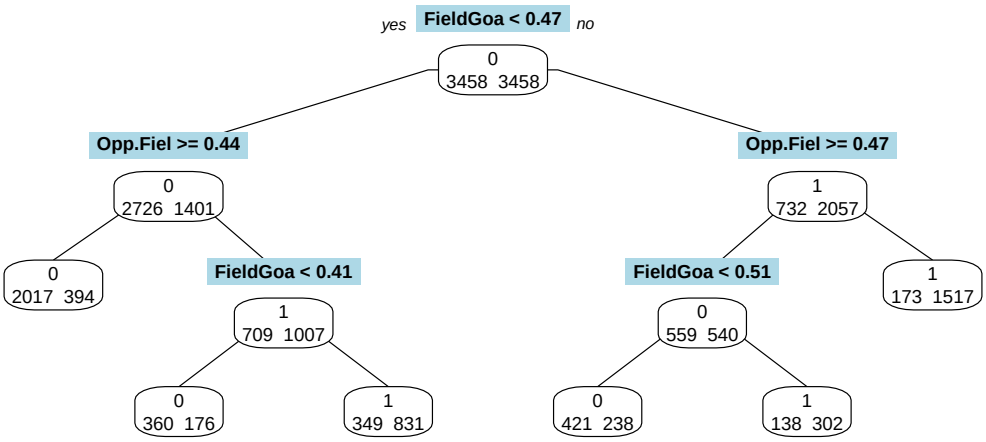
As explained in the "Train/Test Split" section we perform decision trees on two data sets: one with all the variables and another one with the `PIR` and `OppPIR` features.

Using the complete data set

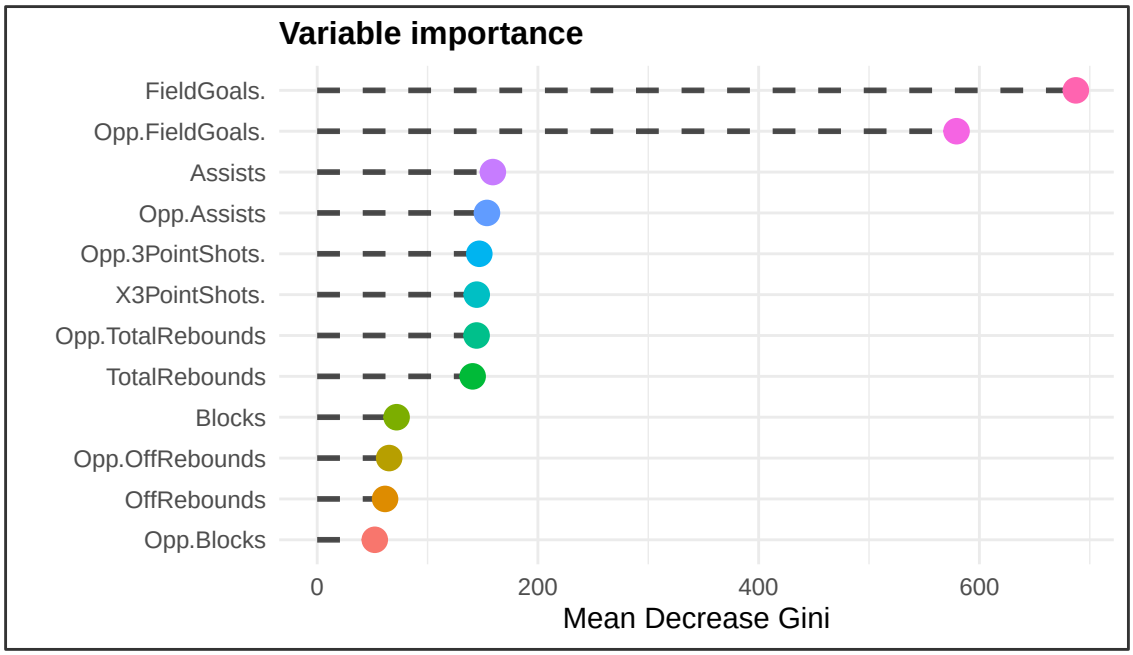
Even though the `TeamPoints` and `OpponentPoints` variables have been removed from the database, there still are two variables strongly impacting the odds of winning, specifically `FieldGoals.` and `Opp.FieldGoals.`. The two latter play an important role in the classification tree building because they suffice to obtain homogeneous nodes quickly enough. As a consequence the tree we get is of maximal depth equal to 6. This optimal number of splits has been decided after a 5-fold cross-validation in which the depth of tree parameter was tuned.



Best decision tree obtained after 5−fold cross−validation

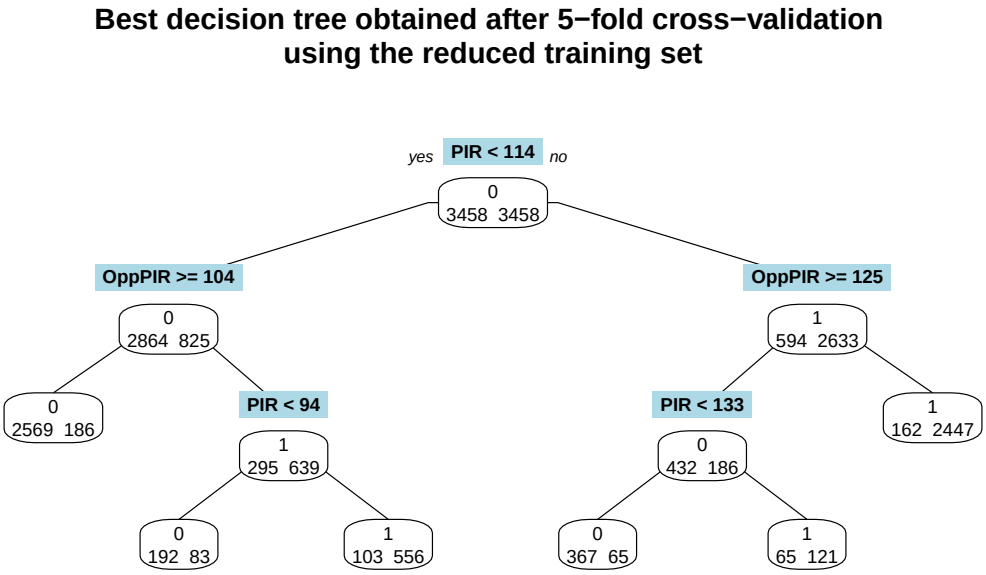


When training a decision tree, it can be relevant to evaluate the predictor importance in the tree building. It can be measured with the Mean Decrease Gini (MDG) metric. MDG weighs the decrease in impurity of nodes resulting from the split of the considered variable. In our case, the `FieldGoals.` and `Opp.FieldGoals.` contribute to the most important MDG. This result is not striking since these two features represent the shooting ratio, free throws aside, for the team and its opponent during a game. It is strongly linked to the `Win` target variable.



Using the reduced data set

We use the same tree building method as for the previous tree on the data set containing the `PIR` and `OppPIR` features to check whether or not these two variables can be considered good predictors of the odds of winning a basketball game. After the tuning stage, we obtain a tree with maximal depth equal to 5 and whose nodes are only made of `PIR` and `OppPIR`. These latter being partly constructed with `FieldGoals.` and `Opp.FieldGoals.`, it is not abnormal getting a similar tree as the previous one.



Comparing the two decision trees

Performance Index Rating being a combination of several game related variables, including `FieldGoals.` and `Opp.FieldGoals.`, the nodes of the second classification tree are better information carriers than those of the first tree. Hence there is nothing abnormal to get better predictions with the tree trained on the reduced data set. It is the one we will keep for the final models comparison taking into account it predicts twice as well as the other tree.

Table 8: Test error obtained with the two trees

	Complete data set	Reduced data set
% error	21.89	9.92

Bagging and random forests

Which database to choose ?

Even though we get very accurate predictions on the reduced data set with the classification tree, we think it is possible to obtain better yet predictions on the complete data set. To do so, we will use both the bagging and random forest algorithms. The principle of these two methods is to combine a large number of trees to improve prediction accuracy at the expense of some loss interpretation. They can be perceived as "black boxes".

The initial step is to train three different models for both the complete and the reduced data set. We carry out a random forest, a bagging and a forest of stumps through 1000 iterations. The goal here is to determine which one best fits the data.

Out-Of-Bag (OOB) stands for the non-selected observations during the training stage of the bagging algorithm. Due to the fact that we perform a 1000-bootstrap, there are about 333 not picked observations for each iteration. These unseen observations are then used for the testing part in order to make predictions. OOB error corresponds to the prediction error on the OOB samples.

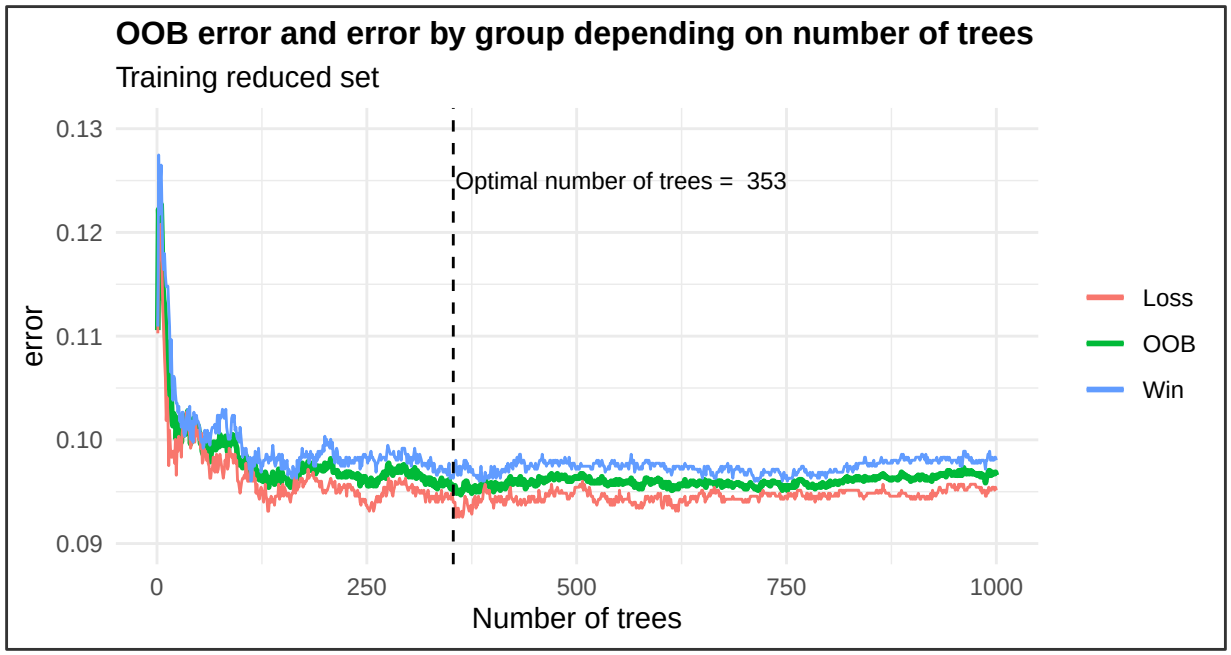
Two core points arise from the following table. On the one hand, linking ensemble methods to decision trees has drastically reduced the error rate for the complete database. On the other hand, using the reduced data set to train the models still gives better predictions. That is why it will be the one used for the tuning process.

Table 9: Comparison of percent OOB error rates for the the two data sets

	Complete data set	Reduced data set
Random forest	11.36	9.47
Bagging	12.09	10.25
Forest of stumps	11.00	10.19

Tuning the model

First of all, let us choose the optimal number of trees to iterate over. From the following plot we see the OOB error tends to about 9% from roughly 250 trees. More precisely this optimal number of trees to build is the number of trees giving an OOB error equal to the minimal OOB error with a 0.05% fluctuation.

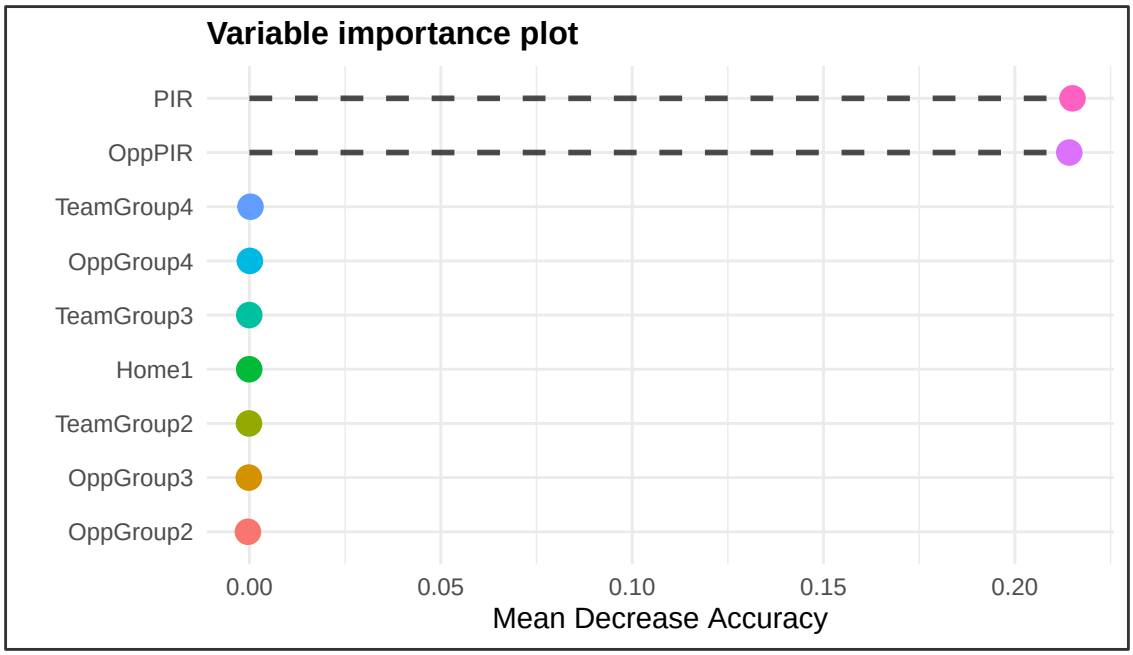


We then use the `ranger` method so as to find the best parameters for the random forest. We end up with a final model with an OOB prediction error of 8.85% and whose attributes are shown in the following table.

Table 10: Tuned parameters for random forest

	mtry	splitrule	min.node.size	ntrees
Optimal values	5	gini	50	353

The following plot depicts the Mean Decrease Accuracy (MDA) which is the average difference in out-of-bag error before and after the permutation over the forest. One can note the `PIR` and `OppPIR` features stand out from the other ones. This result is linked to the variable importance plot of the previous decision tree. Here `PIR` and `OppPIR` replace `FieldGoals.` and `Opp.FieldGoals.` Inasmuch as we perform the random forest algorithm on the reduced training set, the explanation for the importance of these two variables is the same as before. The odds of winning an NBA game are strongly related to the teams' global efficiency. This plot confirms us in our decision to use the PIR statistics as a predictor of the `Win` target variable.



Boosting

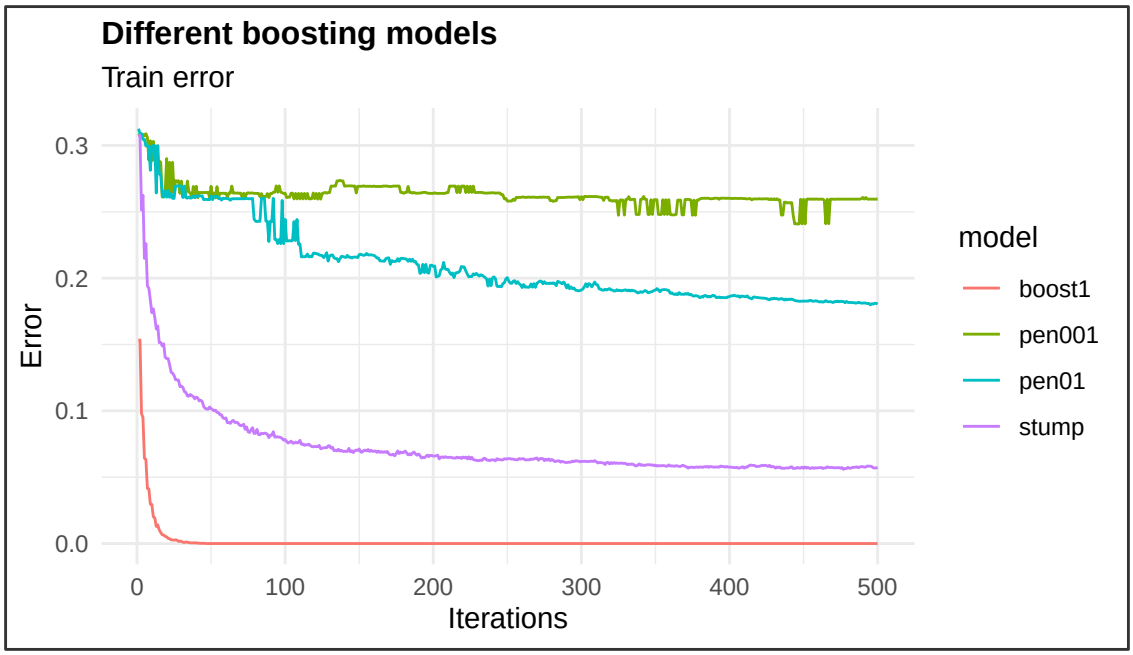
Tuning parameters

Boosting method differs from bagging and random forests to the extent where it is a sequential learning approach, meaning it iterates over M models using the same training set instead of using bootstrapping. At each step of the boosting procedure, each observation is being assigned a weight which is more important for individuals associated with misclassified predictions. As the algorithm goes along, the training prediction error tends to a very low value.

When training boosted decision trees on both the complete and the reduced training set, we come to the conclusion that the former gives a lower train error through a smaller number of iterations. Therefore, the subsequent results are related to the complete data set.

We train several boosting configurations, each of them with 500 iterations but different parameters:

- Boosting with maxed-size-trees,
- Boosting with stumps,
- Boosting with stumps and shrinkage parameter equal to 0.01 to slow down the training process,
- Boosting with stumps and shrinkage parameter equal to 0.001.



From the previous plot, one notices the non-penalized boosting is way faster than the boosting with stumps and/or shrinkage parameter (ν). This result is quite obvious since ν aims to diminish the model coefficient α_m which represents the learning pace. Mathematically it is defined as follows:

$$\widetilde{\alpha_m} = \nu\alpha_m \ , \nu < 1$$

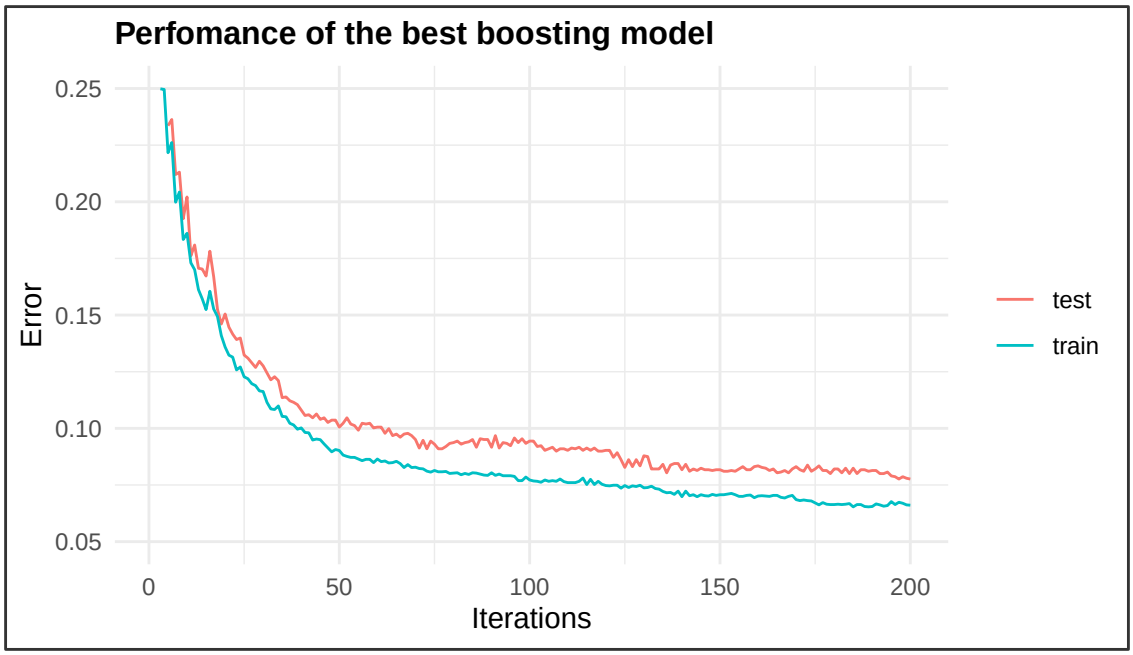
Tuned boosting

Once again we use the `caret` library with a view of tuning the boosting's parameters namely the trees' maximal depth, the number of trees to train and the shrinkage parameter ν.

Table 11: Tuned parameters for boosted decision trees

	iter	maxdepth	nu
Optimal values	200	1	1

The final model we get with the boosting algorithm gives a 6.4% train error throughout 193 iterations. We will apply the final model to the test data set in the following part.



Choosing the best model

Summary of our models' performance

After having trained different machine learning techniques on the NBA data set, the final and most important step of our study protocol is to pick the model which gives the most accurate results. Here we have to choose between three families of models namely those applied on the PCA data, those trained on the reduced data set (with the PIR feature) and the boosted trees trained on the complete data. From the following table one can conclude the boosted trees best predict the `Win` target variable, with a roughly 7.76% test error.

Table 12: Test error for the best models

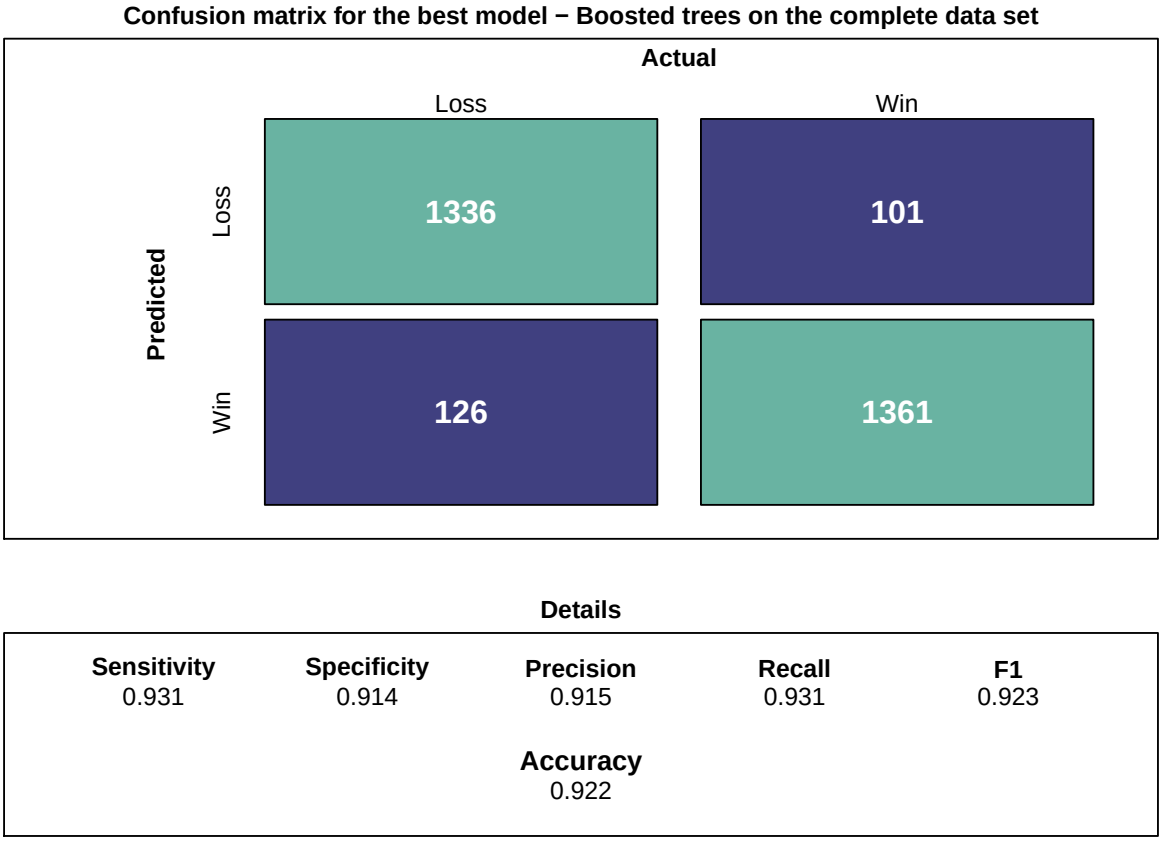
	PCA data		With PIR		Complete data set
	Logistic regression	19-nn	Decision tree	Random forest	Boosted trees
% error	13.54	16.01	9.92	8.55	7.76

The best model

Now the best model identified, let us explore its performance by studying some of its associated metrics. Sensitivity (respectively specificity) measures the proportion of correctly identified wins (respectively losses) and is equivalent to the true positive (respectively negative) rate. Precision corresponds to the proportion of correct "Win" predictions among all the "Win" predictions. As for recall it represents the ability of the model to detect all the actual "Win" outcomes. The F1-score is the harmonic mean between precision and recall. The closer to 1, the more accurate model. Mathematically we have:

$$F_1 = \left(\frac{\text{precision}^{-1} + \text{recall}^{-1}}{2} \right)^{-1}$$

Finally a 92.2% accuracy means the final model classifies a game outcome correctly roughly 92% of the time.



Saving the best model

Let us apply the final boosted trees to the entire data set, then save it in order to use it on future data.

Concluding remarks

Although we managed to build a very accurate final model to predict the outcomes of the NBA games, we are conscious how complicated it could be to forecast unseen data. Now that every predictor is temporally associated to the target variable, the only way to apply our results would be processing it live. In this fashion, it would be a difficult task in terms of data collection. Nonetheless one solution could be to assess the teams' current condition before each game. It could take into account both the players' intrinsic level and ongoing momentum. We would end up with two aggregate variables to utilize as predictors of the '`Win` target variable.