CLASSIFICATION WITH DECISION TREES

ON "KOBE BRYANT SHOT SELECTION" DATASET

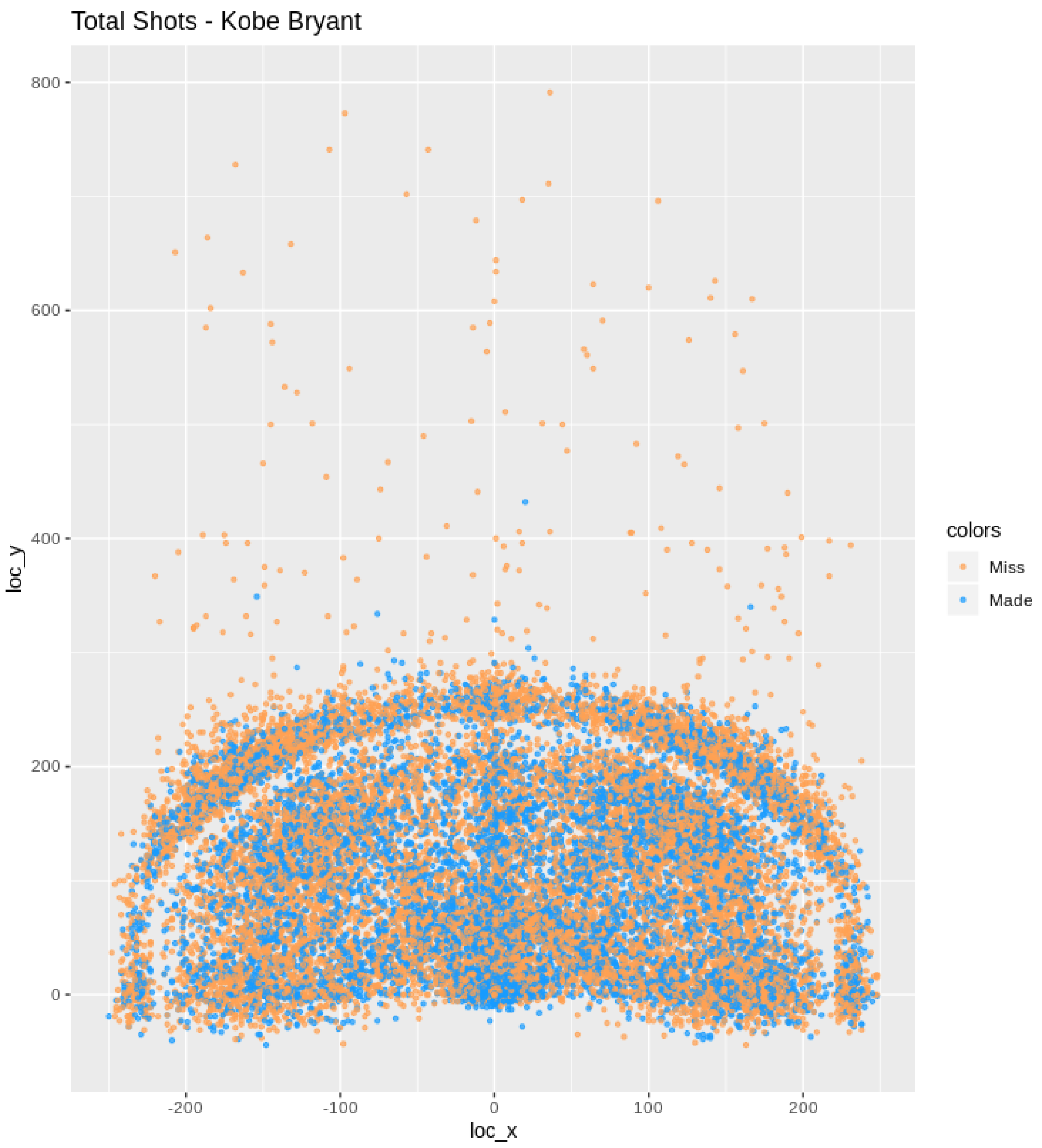
Table of Contents

Achievements:

- 5x NBA champion.
- 1x NBA Mvp.
- 11x All-NBA First Team.
- 2x NBA scoring champion.
- Third ever for points scored.
- 33,643 points (25.0 ppg).
- 1,346 games played.

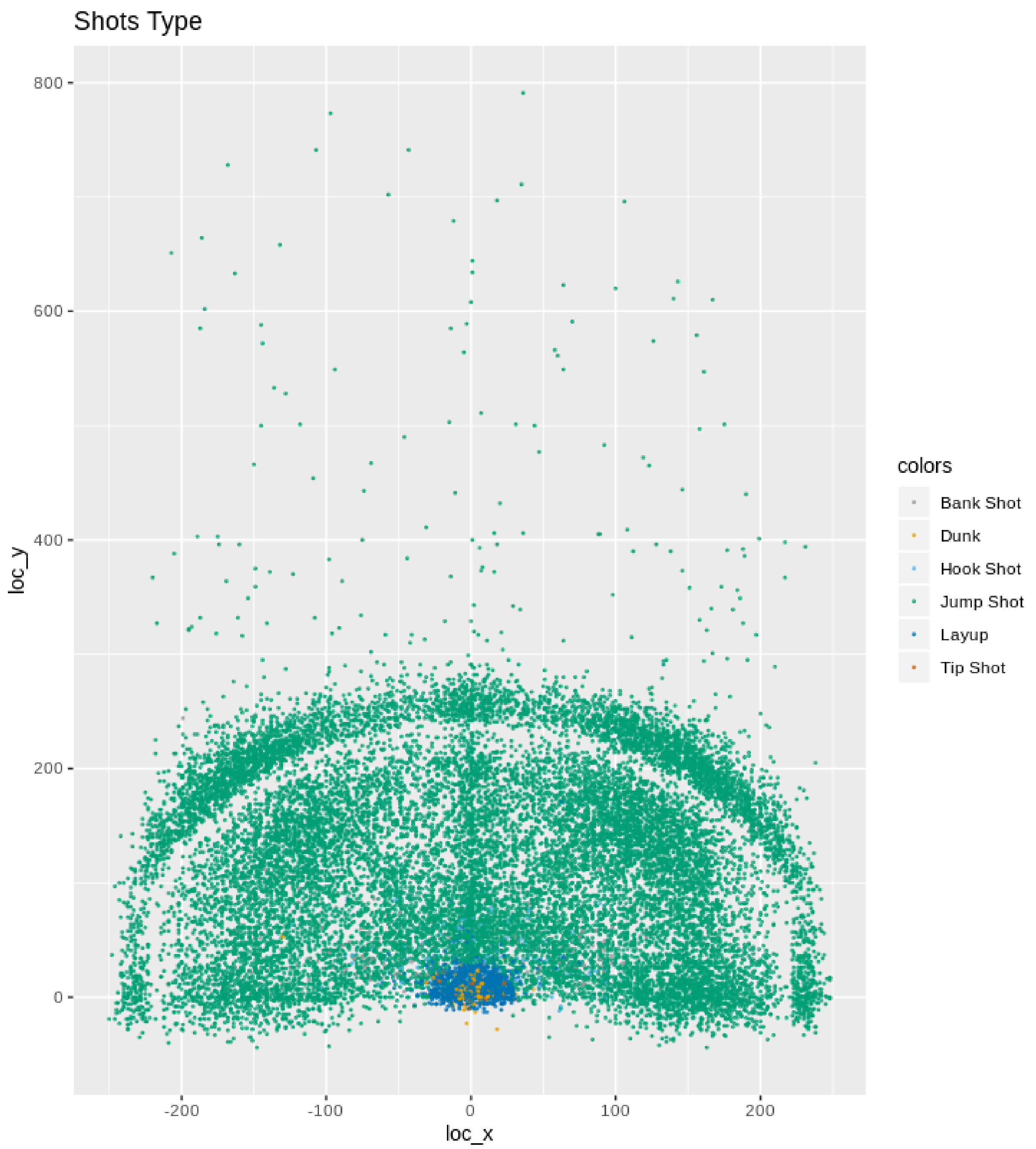
The objective is to predict the qualitative response:

- Shot Made
- Shot Missed



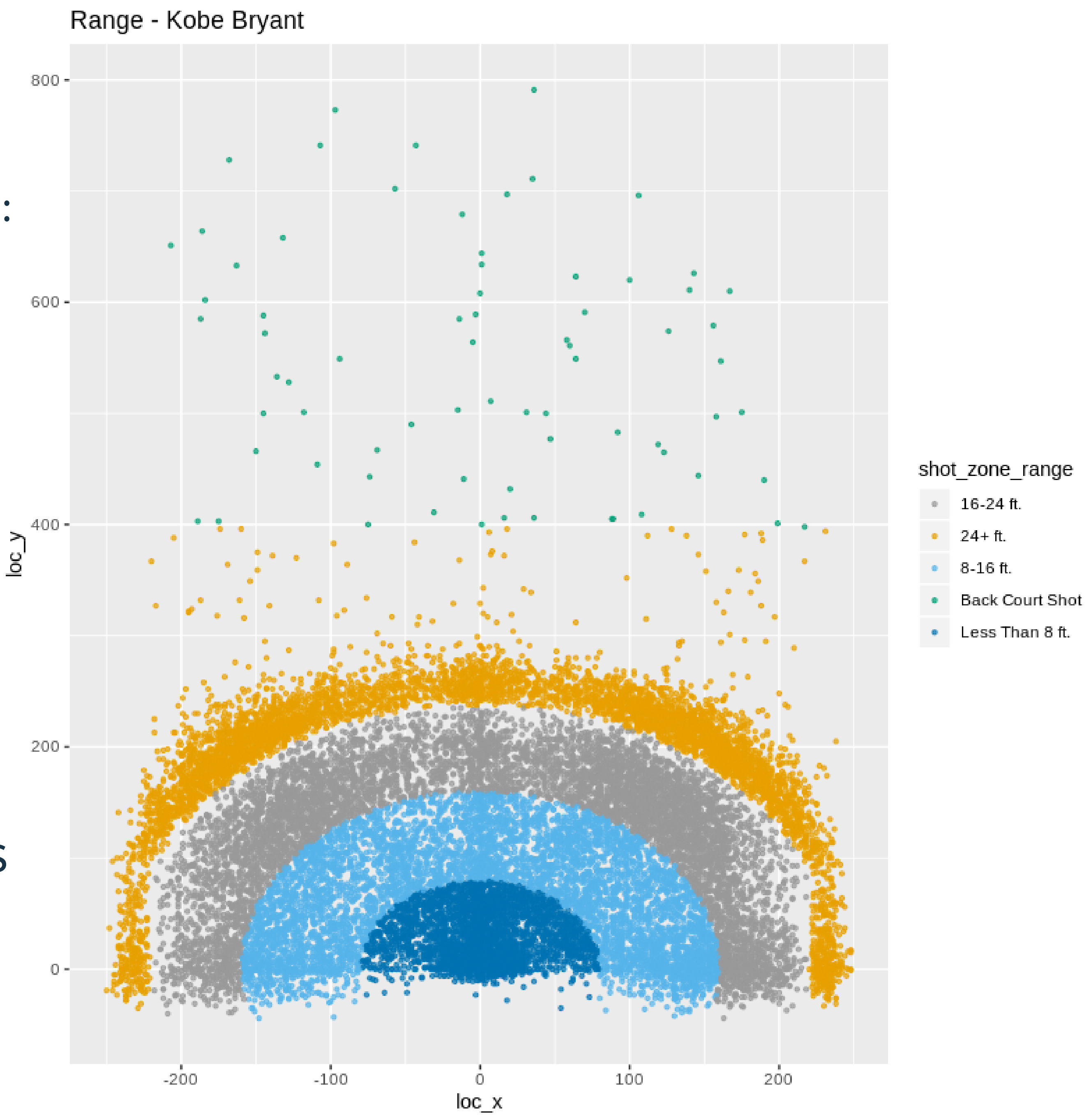
Shooting % per type:
- Bank Shot = 79.17%
- Dunk = 92.80%
- Hook Shot = 53.54%
- Jump Shot = 39.11 %
- Layup = 56.51 %
- Tip Shot = 34.87 %

To notice:
The number of shot types used increases as the player approaches the basket.



Shooting % per range:
- <8ft. = 57.31%
- 8-16ft = 43.56%
- 16-24ft = 40.18%
- >24ft = 33.25%
- Back Court = 1.39%

To notice:
Shooting % decreases with range.



"Home/Away"

Home game shooting % = 45.65
Away game shooting % = 43.64

"Time Remaining"

Decrease in shooting percentage in the last minute of every quarter:

45.54% 38.05%

"Season"

Might be useful to capture the aging in the player's performance.

DECISION TREES

Non-parametric supervised learning tool for both regression and classification problems.

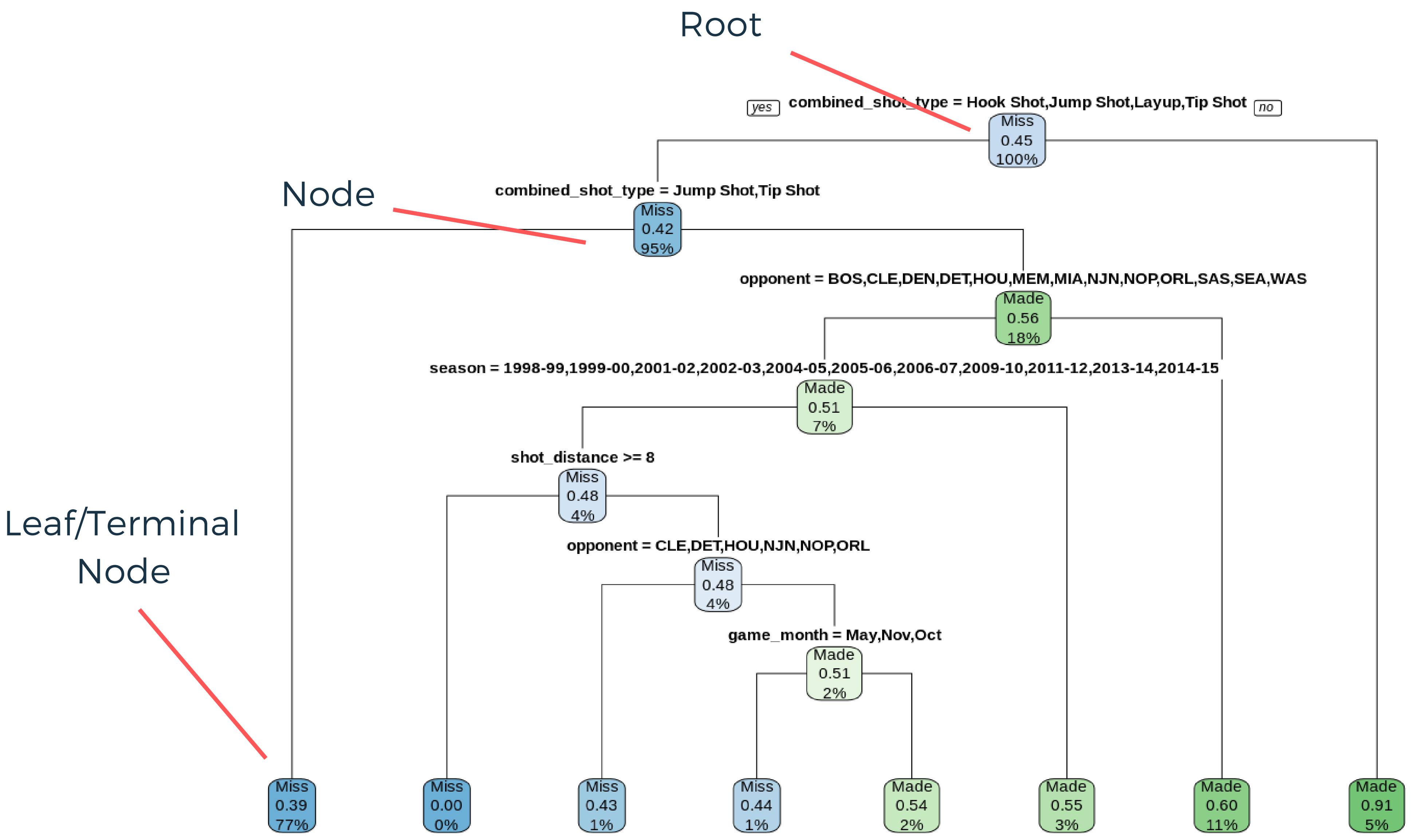
+

- Easy to explain and to interpret.
- Can be displayed graphically.
- Can handle qualitative predictors without creating dummy variables.

−

- Not the best level of predictive accuracy.
- Small change in the data could lead to large change in the final tree.

HOW DOES A DECISION TREE LOOK LIKE?


Root
yes
combined_shot_type = Hook Shot,Jump Shot,Layup,Tip Shot
no
Miss 0.45 100%
Node
combined_shot_type = Jump Shot,Tip Shot
Miss 0.42 95%
opponent = BOS,CLE,DEN,DET,HOU,MEM,MIA,NJN,NOP,ORL,SAS,SEA,WAS
Made 0.56 18%
season = 1998-99,1999-00,2001-02,2002-03,2004-05,2005-06,2006-07,2009-10,2011-12,2013-14,2014-15
Made 0.51 7%
shot_distance >= 8
Miss 0.48 4%
opponent = CLE,DET,HOU,NJN,NOP,ORL
Miss 0.48 4%
game_month = May,Nov,Oct
Made 0.51 2%
Leaf/Terminal Node
Miss 0.39 77%
Miss 0.00 0%
Miss 0.43 1%
Miss 0.44 1%
Made 0.54 2%
Made 0.55 3%
Made 0.60 11%
Made 0.91 5%

- Regarding a classification tree the predicted class for an observation is given by the most commonly occurring class of training observations in the region to which it belongs.

- A recursive binary splitting technique is used to grow the tree. It starts with a first binary split and then it continues untile no further splits can be made.

- Each node is presented as an "if/then" rule.

When is the data splitted?

The data is splitted accorting to one selected criterion.

Classification Error Rate

The fraction of training observations in the region m that do not belong to the most common class k.

$$E = 1 - \max_{k}(\hat{p}_{mk})$$

Where p̂ represents the proportion of training observations in the mth region that are from the kth class.

Although it seems like the natural alternative to the RSS, used for regression trees, it turns out that it is not sensible enough for tree-growing and two other criteria are preferable.

Gini Index

The Gini Index is a measure of the total variance across the K classes. It can also be seen as a measure of node purity, because a small value indicates that a node contains predominantly observations for a single class.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Therefore, in this case, the data is splitted if the split produces an improvement in the node purity; so if the Gini Index decreases.

Entropy

Alternative to Gini Index, The cross-entropy takes on a value near zero if the p̂'s are all near 0 or near 1. Therefore, like the Gini index, the cross-entropy takes on a small value if the mth node is pure.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$$

Regarding the model of this presentation the Gini Index is the chosen citerion to split the data.

```
rpmodel <- rpart(shot_made_flag ~ action_type +
shot_distance + home_away + season + opponent +
time_remaining + shot_zone_basic + shot_zone_range +
shot_zone_area, data=data.train, control=rpart.control
(minsplit=20, minbucket=5, cp=0.001, xval=10),
parms=list(split="gini"))
```

Where:
- minsplit = minimum number of observations that has to exist in a node in order for a split to be attempted.
- minbucket = minimum number of observations in any terminal node.
- xval= number of cross-validations.

Output:

Variable importance:

action_type (68)	shot_zone_area (1)
shot_distance (11)	opponent (1)
shot_zone_basic (10)	season (1)
shot_zone_range (6)	

Root node error: 9172/20558 = 0.44615
This is the error rate for a single node tree, that is, if the tree was pruned to node 1.

n= 20558

DECISION TREES- MODEL OUTPUT -2

```
CP          nsplit  rel error  xerror    xstd
0.2854339     0     1.00000    1.00000   0.0077708
0.0017444     1     0.71457    0.71653   0.0072902
0.0012538     5     0.70759    0.71729   0.0072923
0.0011993     7     0.70508    0.71773   0.0072935
0.0010000     8     0.70388    0.71609   0.0072891
```

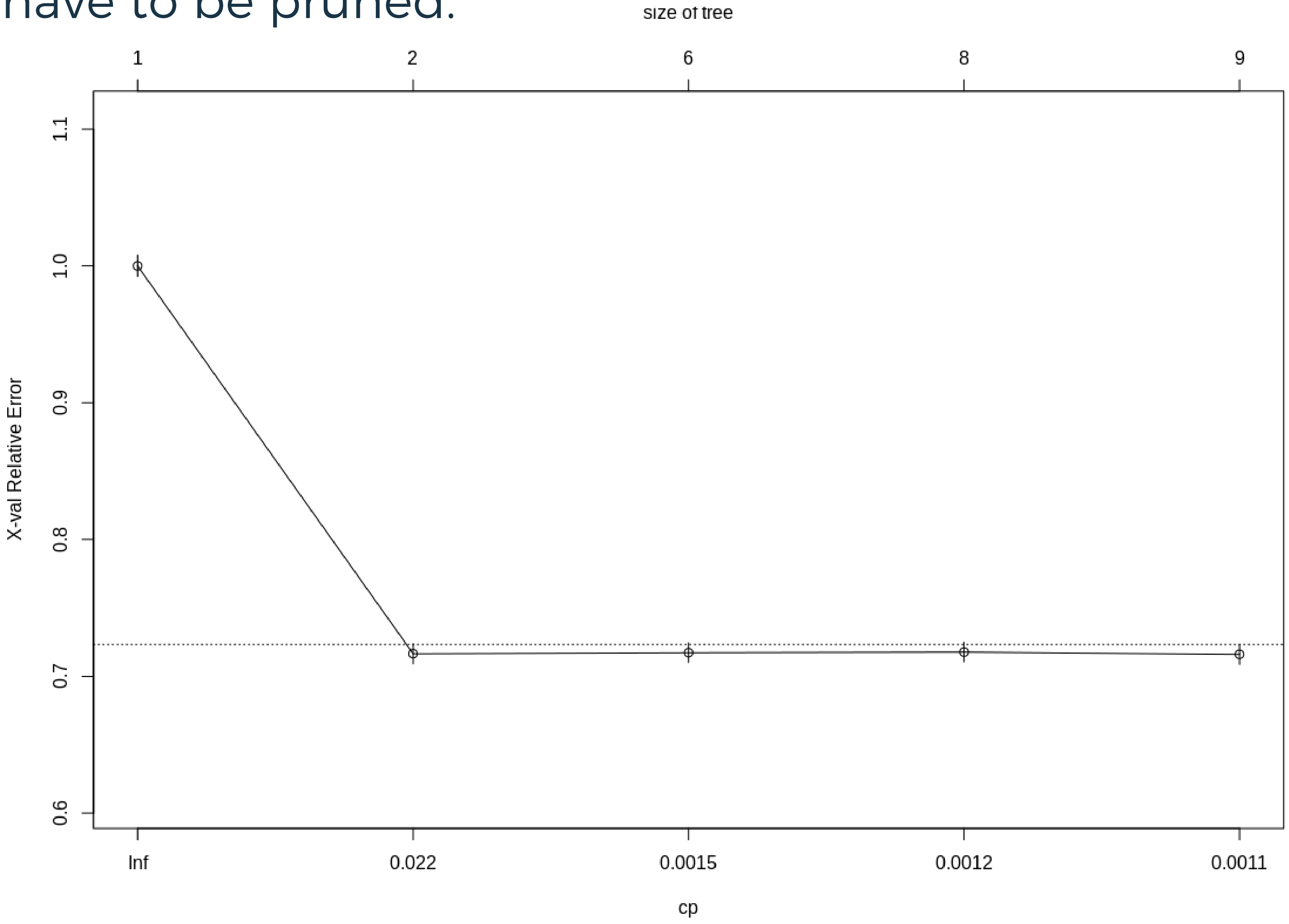
- rel error = error for predictions of the data used to estimate the model.
- x error = the cross-validation error
- x std = the standard deviation of error across the cross-validation sets.

The complexity parameter (cp) in rpart is the amount by which splitting a node improves the relative error. When the model can only be improved by less than the defualt cp value, the tree building stops. It's based on the cost complexity of the model defined as:

$$\sum_{TerminalNodes} Misclass_i + \lambda * (Splits)$$

For a tree, the complexity parameter is given by the misclassification at every terminal node, then the number of splits is multiplied by a penalty term (lambda) and added to the total misclassification. The lambda is determined through cross-validation and not reported in R.
So basically the cp value is a stopping parameter.

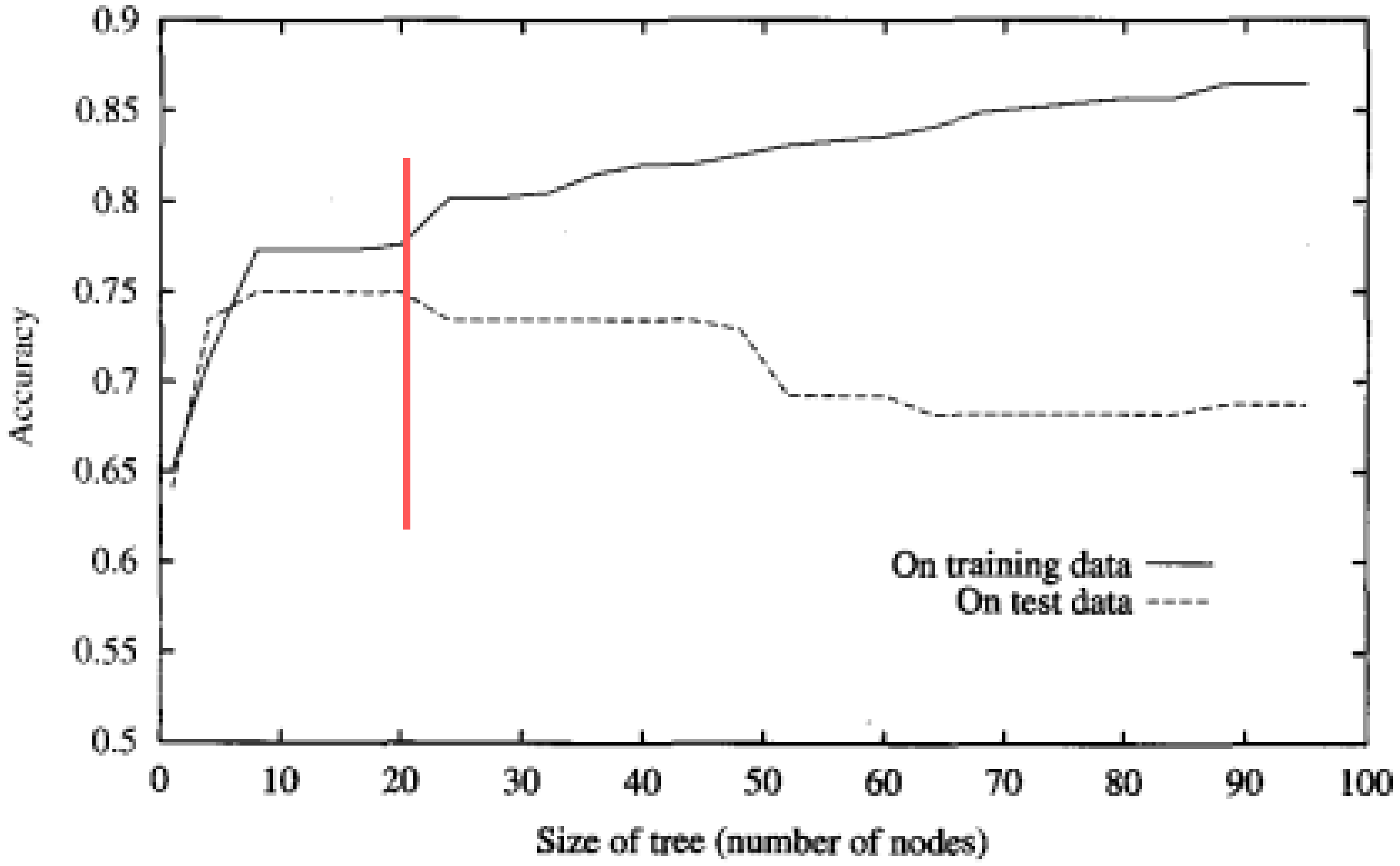
Since the cross validation error is slightly decreasing the tree doesn't have to be pruned.



Pruning

In this case a good choice could be to cut the tree at size = 20 in order to avoid overfitting.

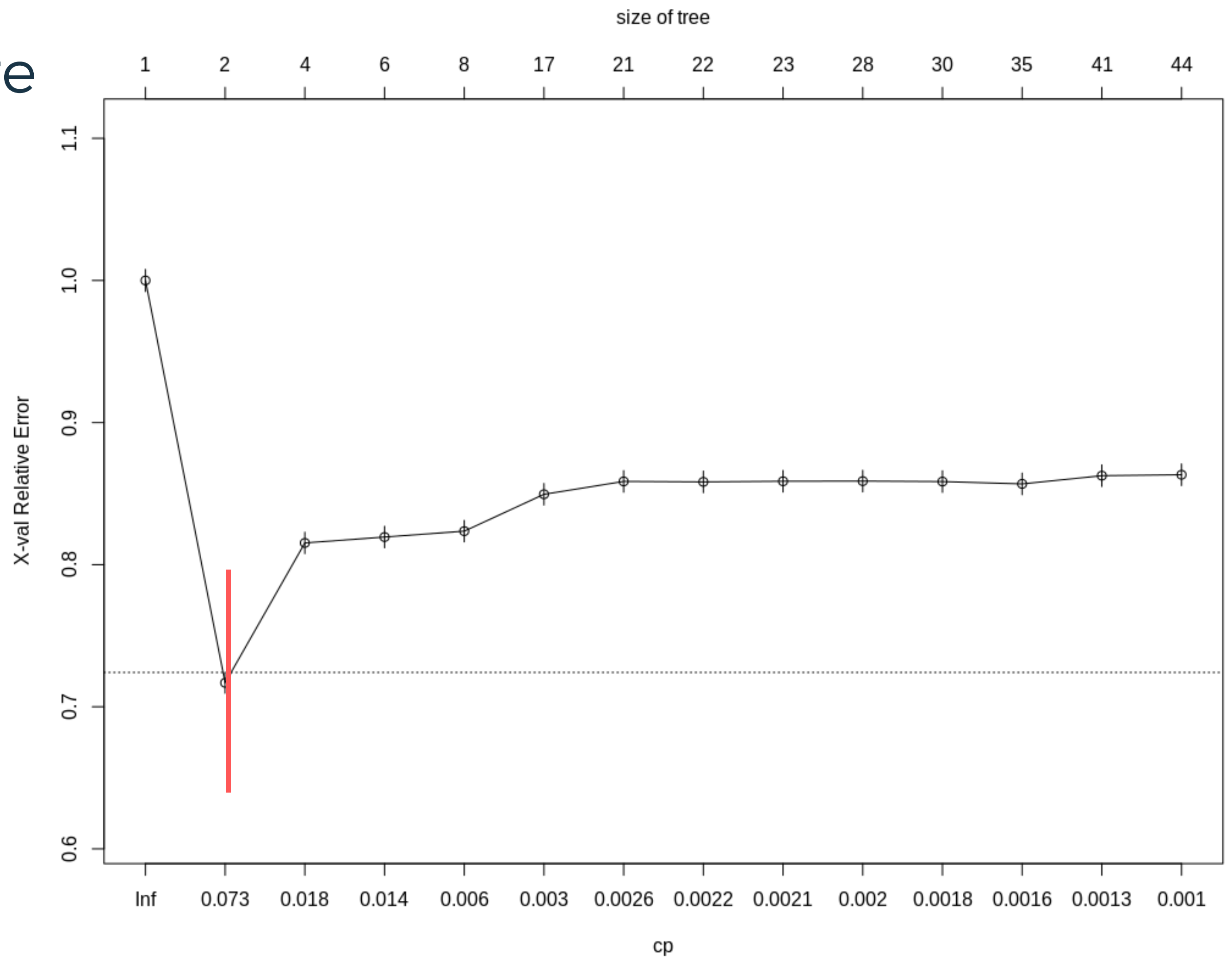
How to spot overfitting though?



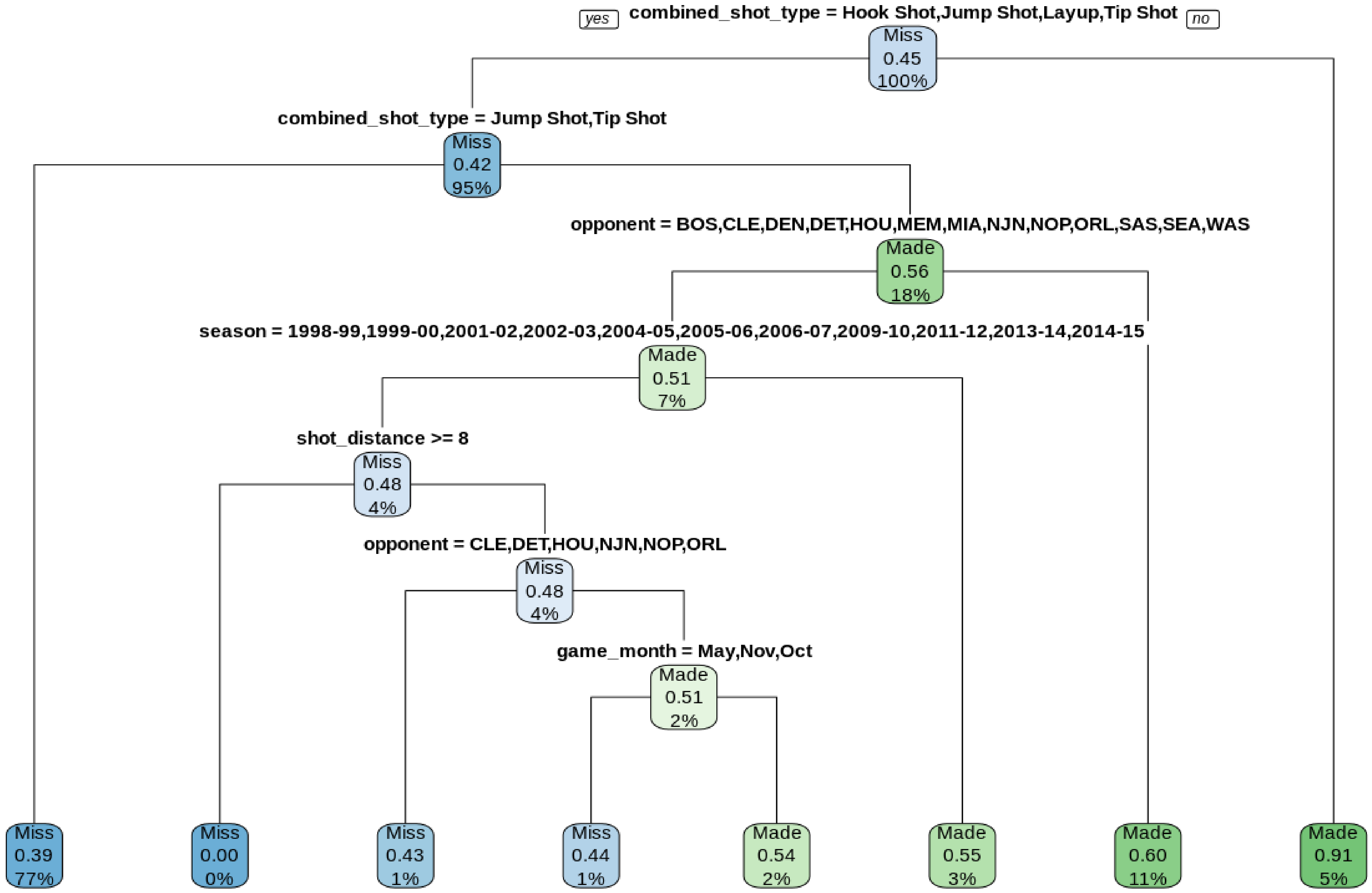
In the case of a decision tree, overfitting can be spotted by looking at the CP table. Infact, if the model overfit, there will be a decreasing "rel error" facing an increasing "x error".

Here is an extreme case example where the cross validation error starts rising at size = 2, while the relative error decreases.

nsplit	rel error	xerror
0	1.00000	1.00000
1	0.71276	0.71460
3	0.68049	0.82925
5	0.65307	0.83070
7	0.64115	0.83710
9	0.63020	0.85105
11	0.62438	0.85406
12	0.62177	0.85958
13	0.61925	0.85958
17	0.60926	0.86268



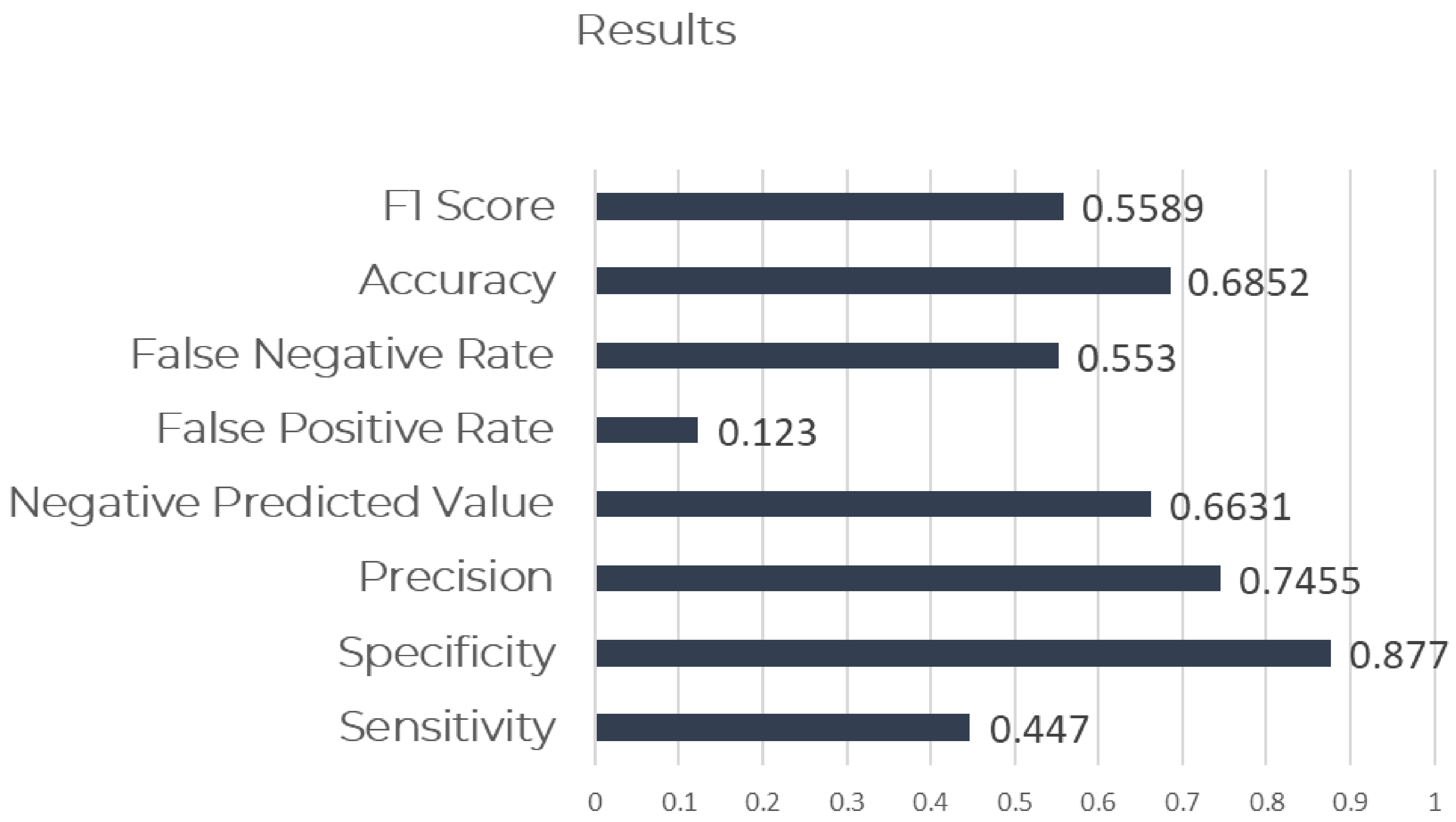
The only difference of the following plot with the model actually used in the prediction is the use of "combined_shot_type" intead of "action_type". Because although "action_type" makes the model more accurate, it also makes the plot bigger since the predictor contains 57 factors.
I suggest then using "combined_shot_type" for visualization purposes and using "action_type" for prediction, in order to exploit this predictor capacity of embedding a lot of information about the taken shot.


yes
combined_shot_type = Hook Shot,Jump Shot,Layup,Tip Shot
no
Miss 0.45 100%
combined_shot_type = Jump Shot,Tip Shot
Miss 0.42 95%
opponent = BOS,CLE,DEN,DET,HOU,MEM,MIA,NJN,NOP,ORL,SAS,SEA,WAS
Made 0.56 18%
season = 1998-99,1999-00,2001-02,2002-03,2004-05,2005-06,2006-07,2009-10,2011-12,2013-14,2014-15
Made 0.51 7%
shot_distance >= 8
Miss 0.48 4%
opponent = CLE,DET,HOU,NJN,NOP,ORL
Miss 0.48 4%
game_month = May,Nov,Oct
Made 0.51 2%
Miss 0.39 77%
Miss 0.00 0%
Miss 0.43 1%
Miss 0.44 1%
Made 0.54 2%
Made 0.55 3%
Made 0.60 11%
Made 0.91 5%

Test results

Confusion Matrix:

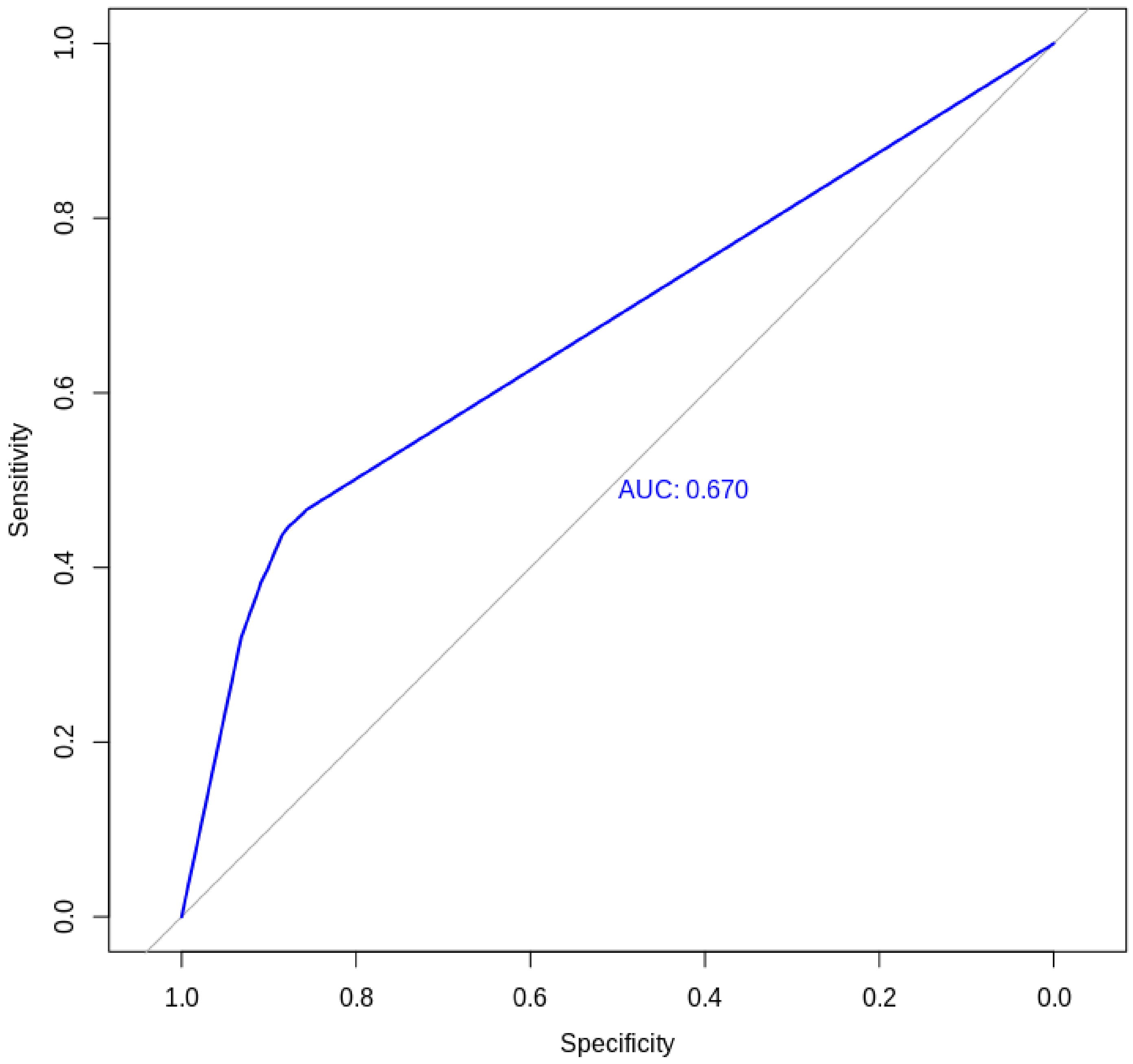
	Made	Miss	Sum
Predicted Made	1025	350	1375
Predicted Miss	1268	2496	3764
Sum	2293	2846	5139


Results
FI Score
0.5589
Accuracy
0.6852
False Negative Rate
0.553
False Positive Rate
0.123
Negative Predicted Value
0.6631
Precision
0.7455
Specificity
0.877
Sensitivity
0.447
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Remarks on confusion matrix

The main issue, as stated by the low sensitivity and higher precision, is that the model misses a lot of positives (high FN) but the predicted ones are indeed positive (low FP). Thus a possible improvement of the model could be the recognition of false negatives.

ROC Curve and AUC



The ROC Curve confirms the low sensitivity which is drag down by the high number of FN.

A possible development of the project could be the exploitation of methods such as bagging, random forest and boosting in order to attempt an improvement in the prediction performance of the model.
In addition it could be helpful to find some data regarding defence (the position of the defender...) since this dataset doesn't have variables explaining it.

Thank you all for the attention.

- L. Breiman, J.H. Friedman, R.A. Olshen, , and C.J Stone. Classification and Regression Trees. Wadsworth, Belmont, Ca, 1983.
- Danesi Ivan Luciano, Perugini Federica, Data Exploration for Data Science, EDUCatt, Milano, 2018.
- Eric Eaton, Classification: Decision Trees & Overfitting, Georgia Tech, 2016.
- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, An Introduction to Statistical Learning, Springer, London, 2017.
- Kaggle, https://www.kaggle.com/c/kobe-bryant-shot-selection/data, Dataset, 2016.
- Terry Therneau, Beth Atkinson, Brian Ripley, Package 'rpart', CRAN, 2018.
- Terry Therneau, Beth Atkinson, An Introduction to Recursive Partitioning Using the RPART, routines, Mayo Foundation February, 2018.
- Papadopoulos Aris, https://towardsdatascience.com/nba-data-science-93e0314bb45e, 2018.
- http://www.learnbymarketing.com/tutorials/rpart-decision-trees-in-r/, 2017.