2021

Towards machine learning; Trees and forests

David Makowski

Artificial intelligence

Machine learning

Artificial intelligence

Machine learning

Supervised learning

Objective: « Learning a function that maps an input to an output based on examples of input-output pairs »

Statistical Modeling: The Two Cultures (Breiman, 2001)

$$y = f(x) + e$$

Modelling approach 1: Try to find the true $f(x)$

Modelling approach 2: Predict y from x as accurately as possible

Statistical Modeling: The Two Cultures (Breiman, 2001)

$$y = f(x) + e$$

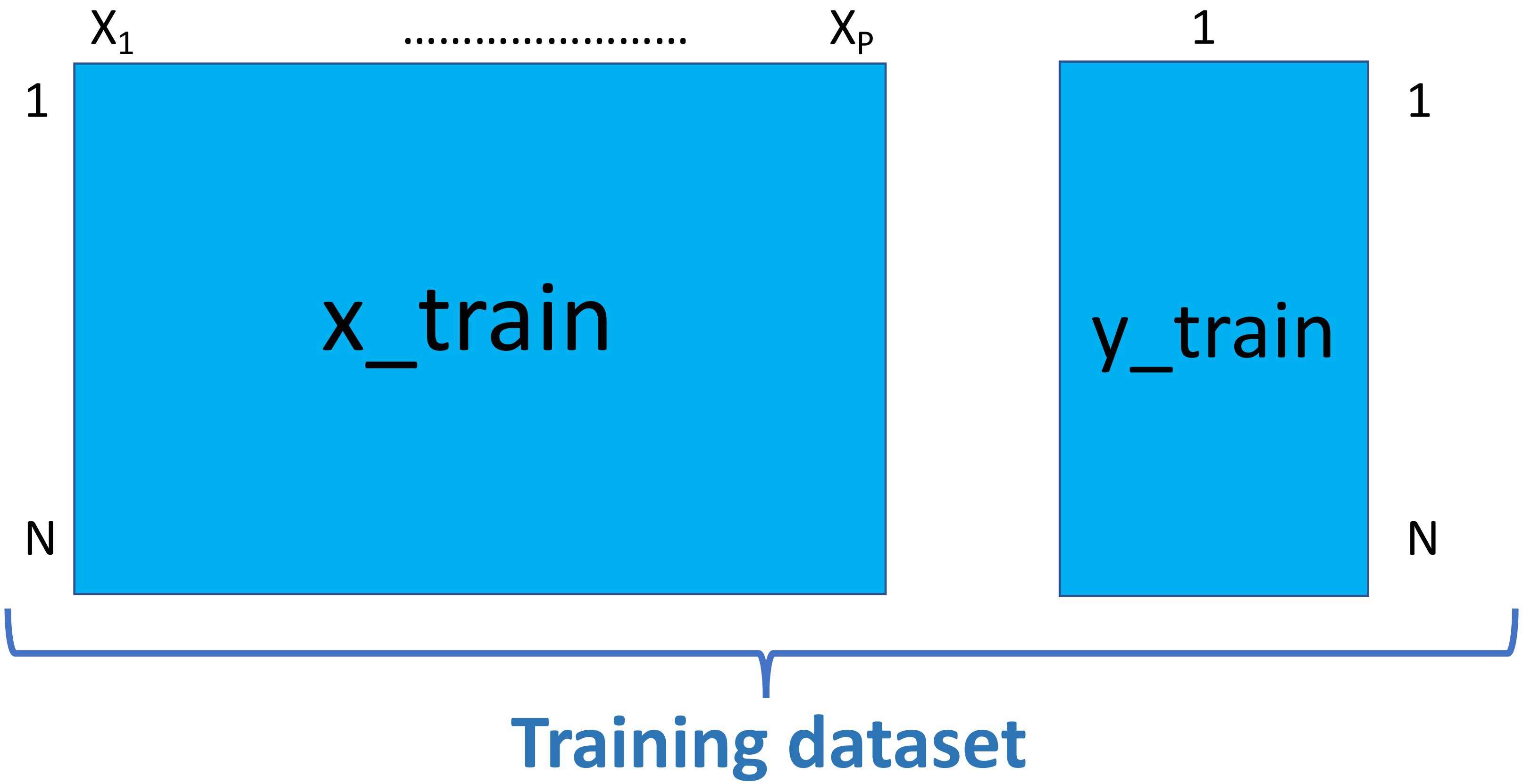
Modelling approach 1: Try to find the true $f(x)$

Modelling approach 2: Predict y from x as accurately as possible

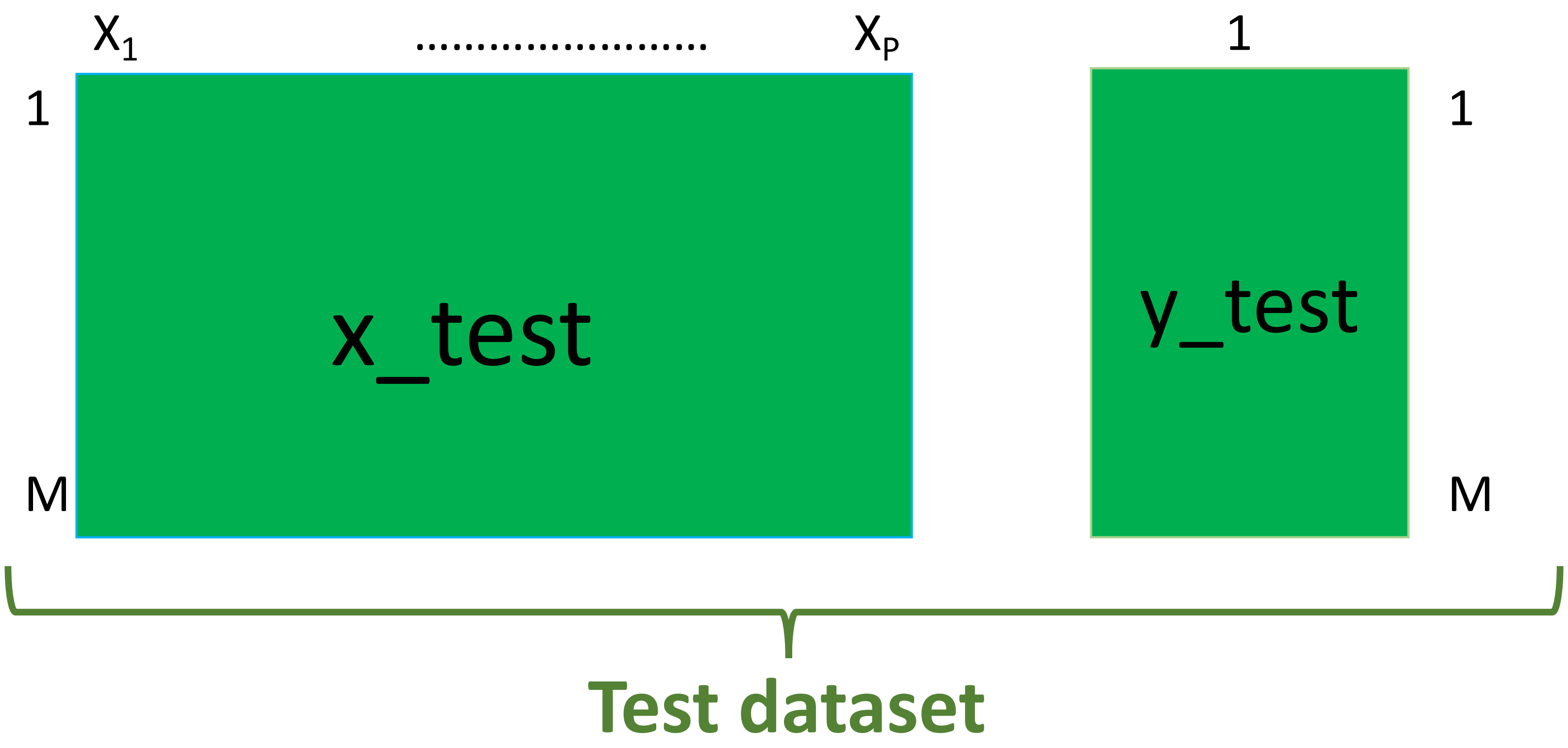
Two main steps

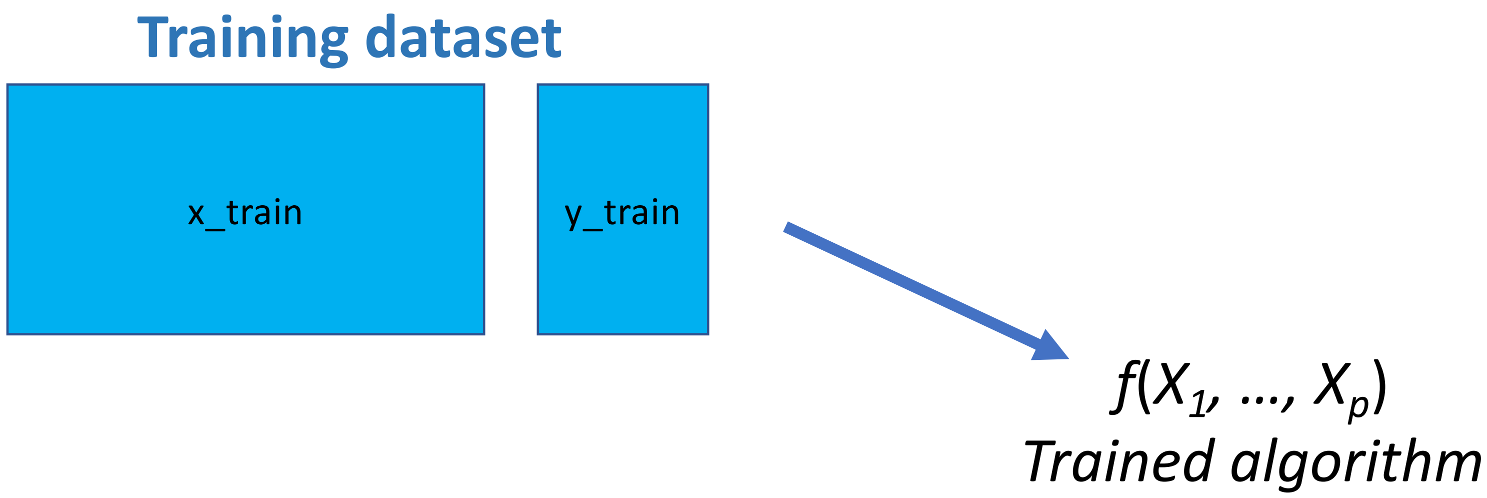
- Step 1: Training
- Step 2: Test

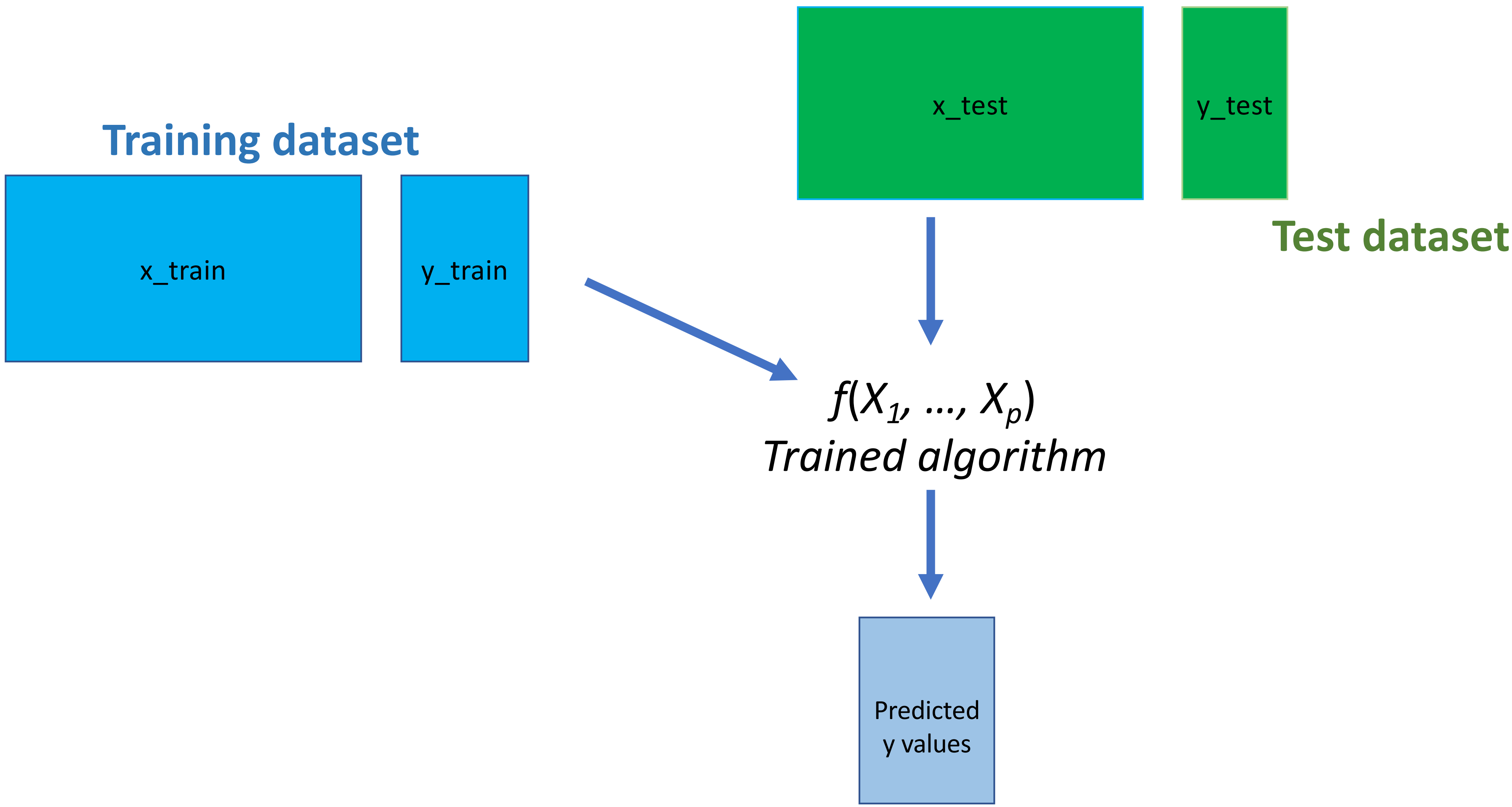
Step 1: Train an algorithm predicting Y as a function of X_1, …, X_P using a **training dataset**

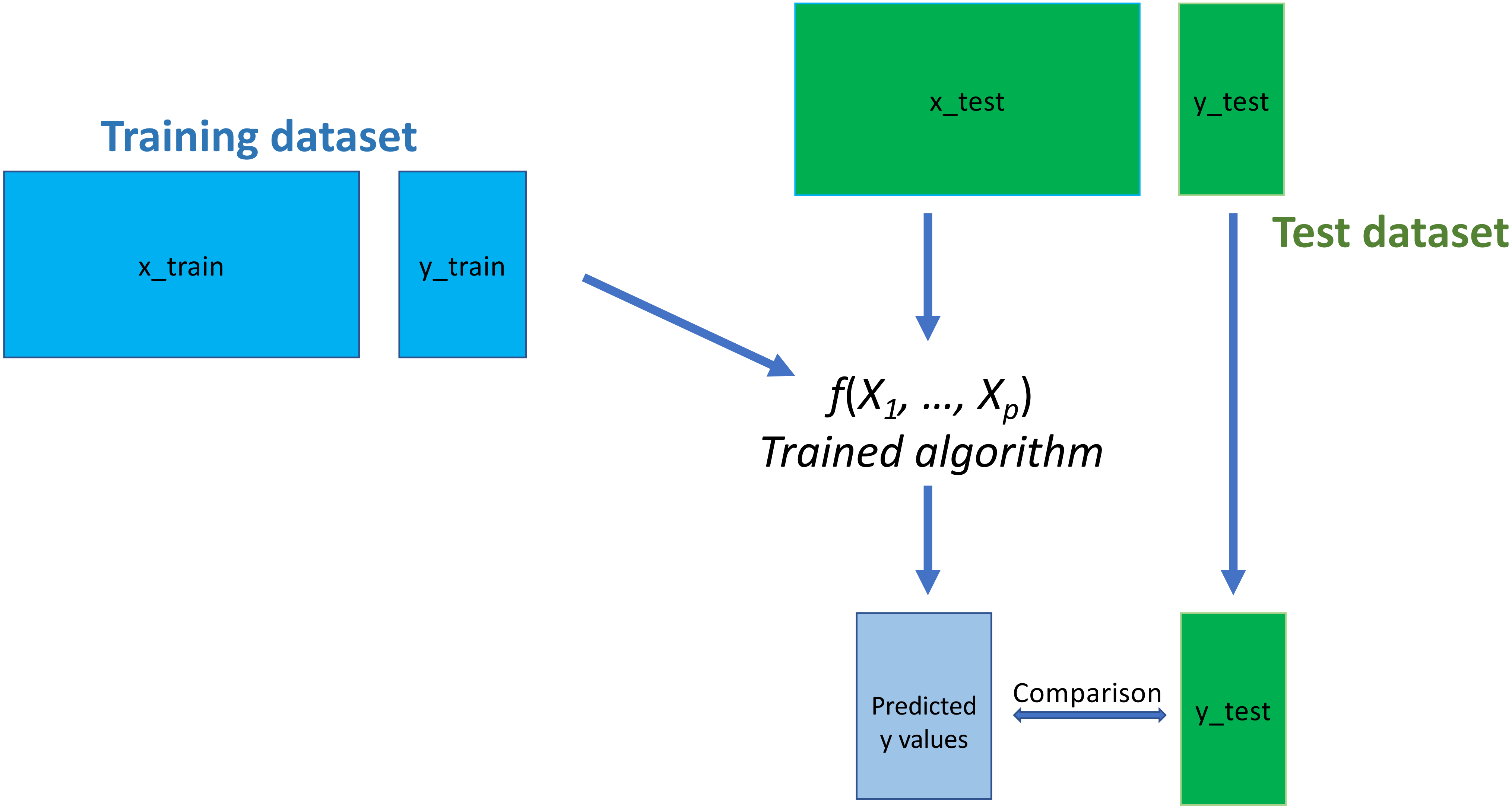


Step 2: Assess the predictive capability of the trained algorithm using a **test dataset**



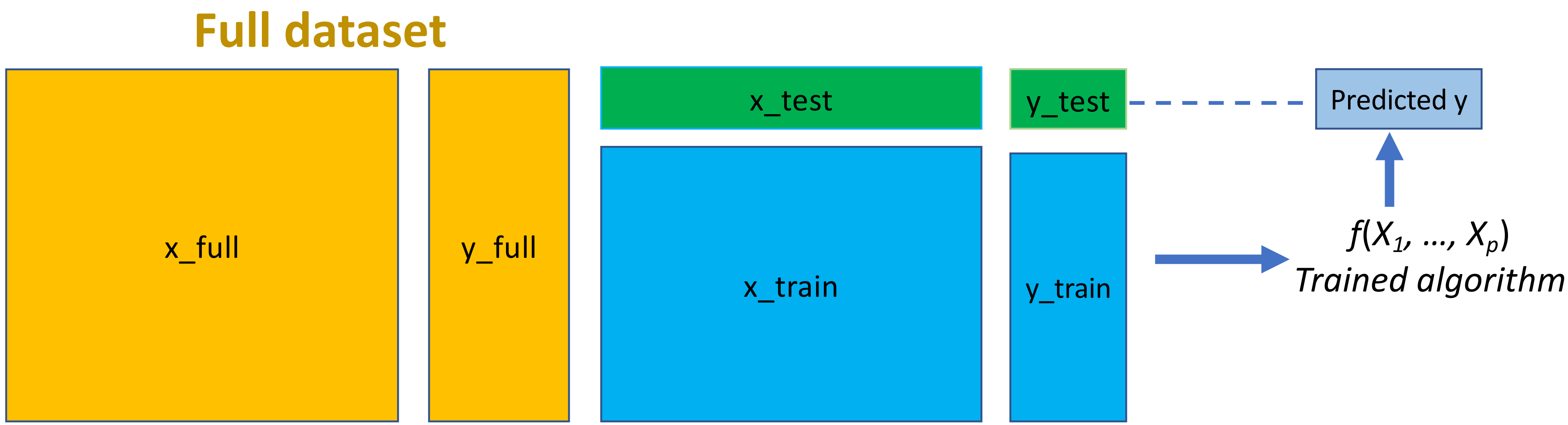

Training dataset
x_train
y_train
f(X1, ..., Xp)
Trained algorithm

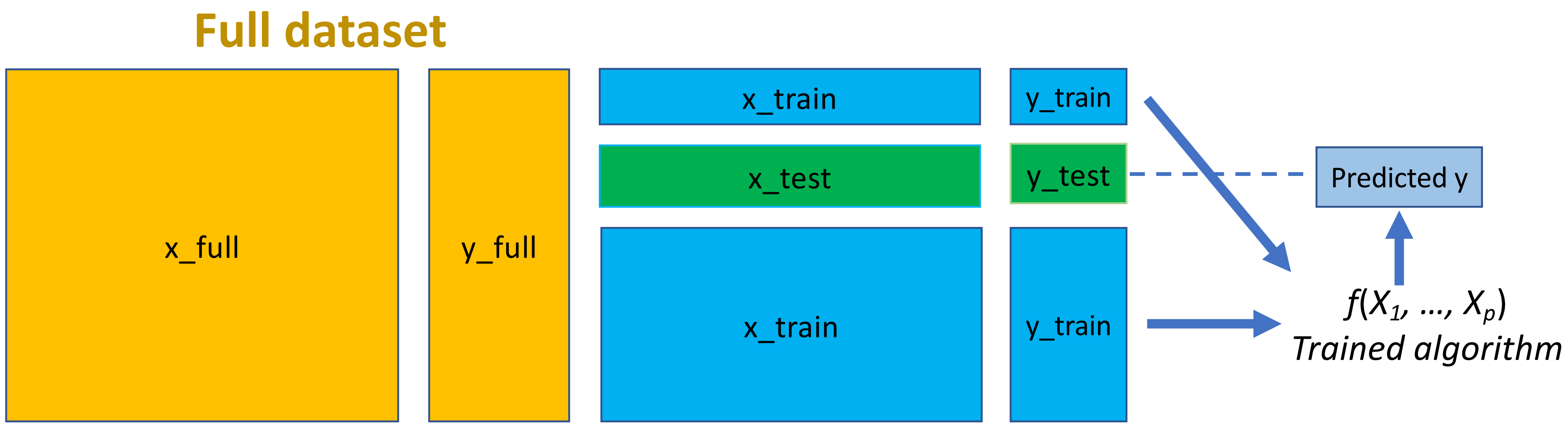

x_test
y_test
Test dataset
Training dataset
x_train
y_train
$f(X_1, \ldots, X_p)$
Trained algorithm
Predicted
y values

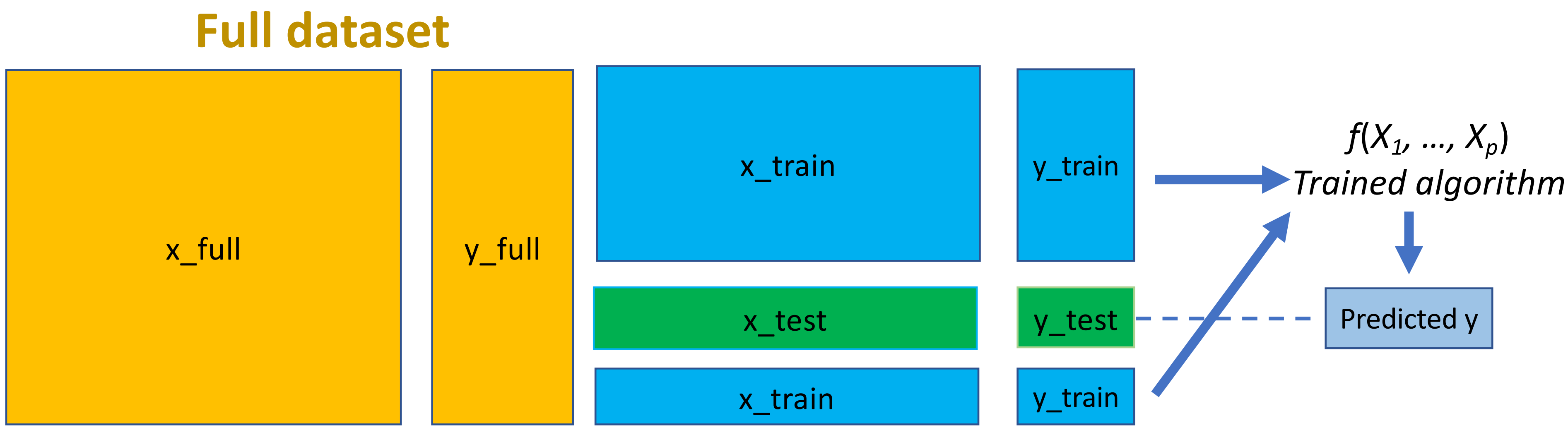


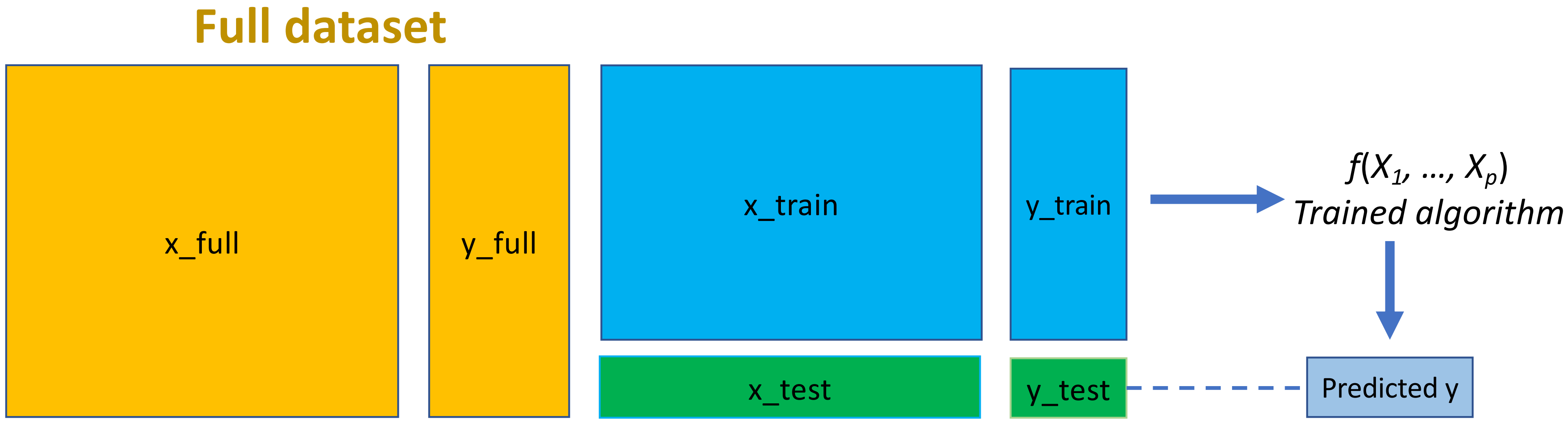
Training dataset
x_train
y_train
x_test
y_test
Test dataset
$f(X_1, \ldots, X_p)$
Trained algorithm
Predicted
y values
Comparison
y_test

Cross-validation is used when no independent test dataset is available


Full dataset
x_full
y_full
x_test
y_test
Predicted y
x_train
y_train
$f(X_1, \ldots, X_p)$
Trained algorithm

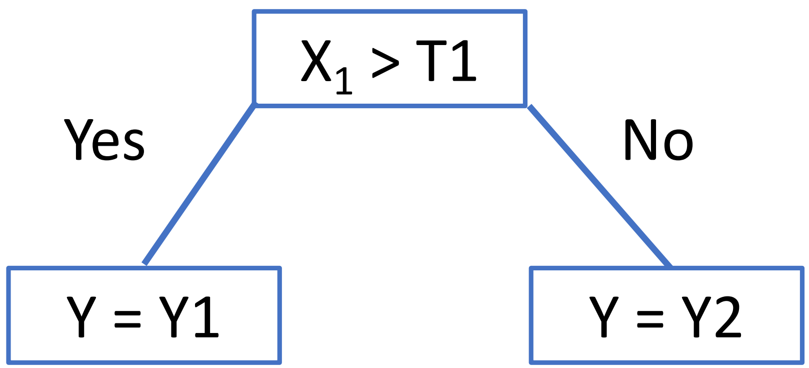

Full dataset
x_full
y_full
x_train
y_train
x_test
y_test
Predicted y
x_train
y_train
$f(X_1, \ldots, X_p)$
Trained algorithm


Full dataset
x_full
y_full
x_train
y_train
x_test
y_test
x_train
y_train
f(X1, …, Xp)
Trained algorithm
Predicted y


Full dataset
x_full
y_full
x_train
y_train
x_test
y_test
$f(X_1, \ldots, X_p)$
Trained algorithm
Predicted y

Tree

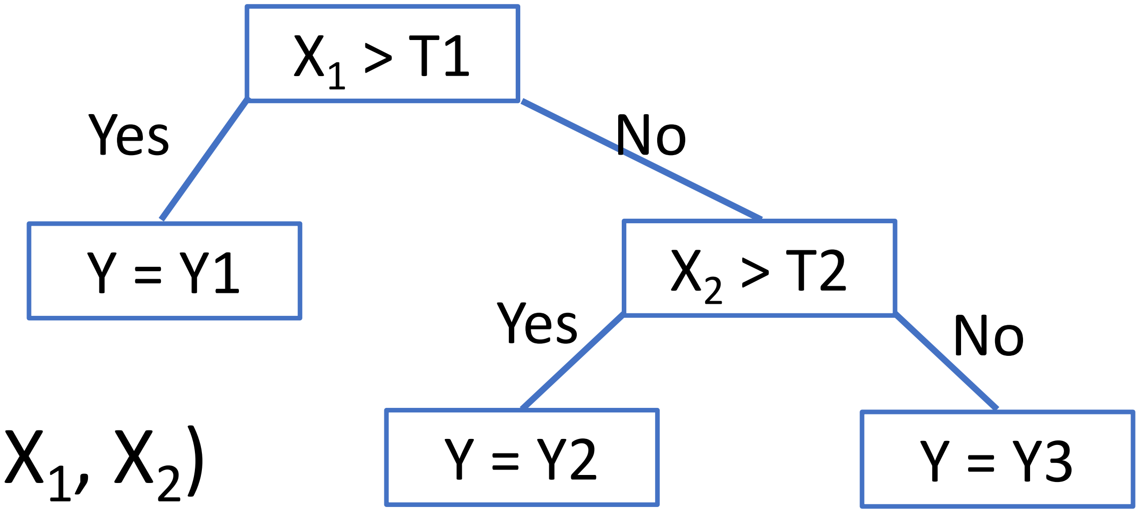
Tree = Model based on a series of splitting rules



$$Y = f(X_1)$$

Tree

Tree = Model based on a series of splitting rules



$Y=f(X_1, X_2)$

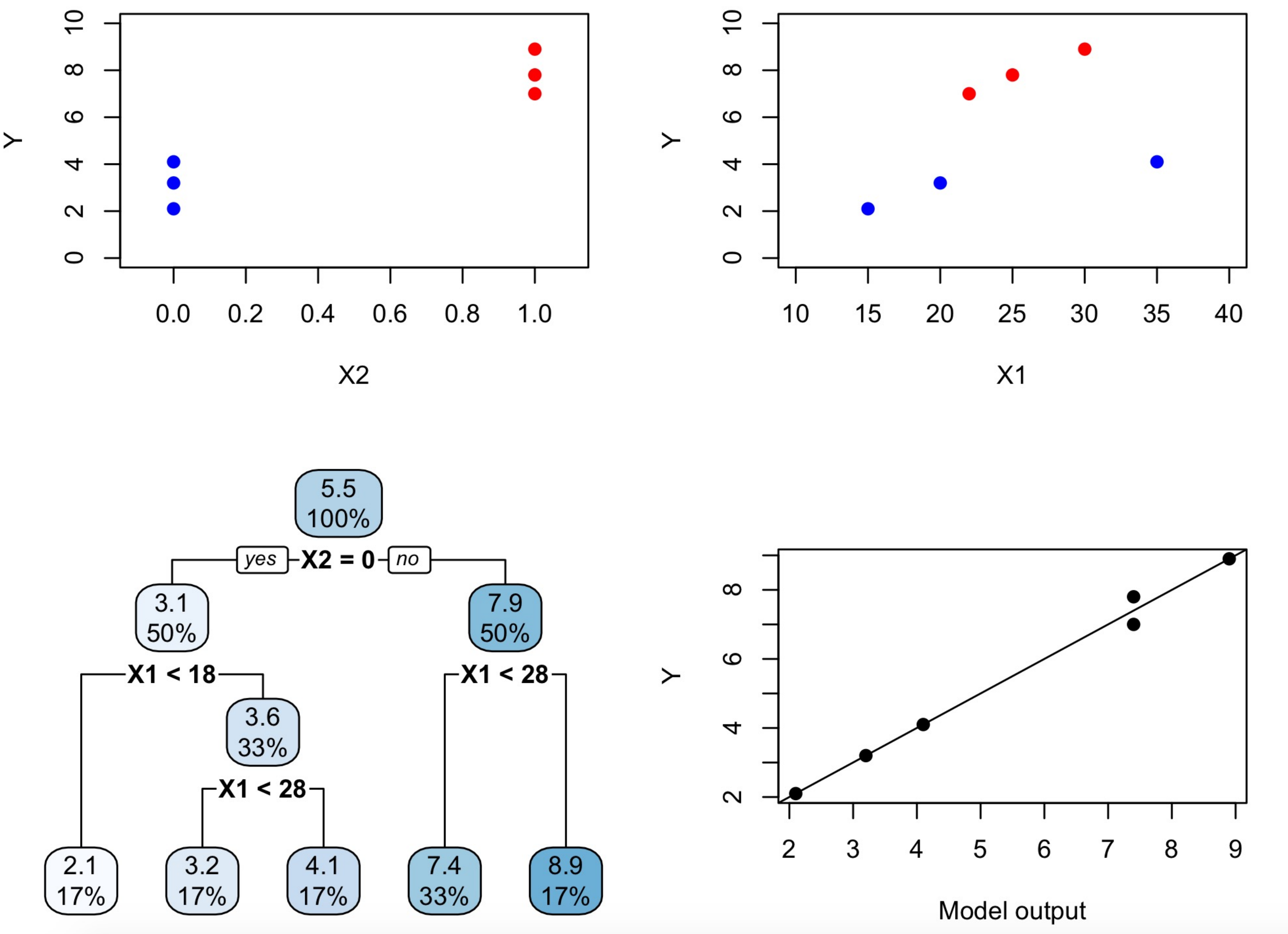
Tree

- Training is based on the optimization of a criterion measuring the level of « purity » of each terminal node (Gini) or its accuracy (MSE).
- The tree is pruned in order to keep it relatively simple. The level of pruning is optimized by cross-validation

A « toy » example

	X1	X2	Y
1	15	0	2.1
2	20	0	3.2
3	22	1	7.0
4	25	1	7.8
5	30	1	8.9
6	35	0	4.1

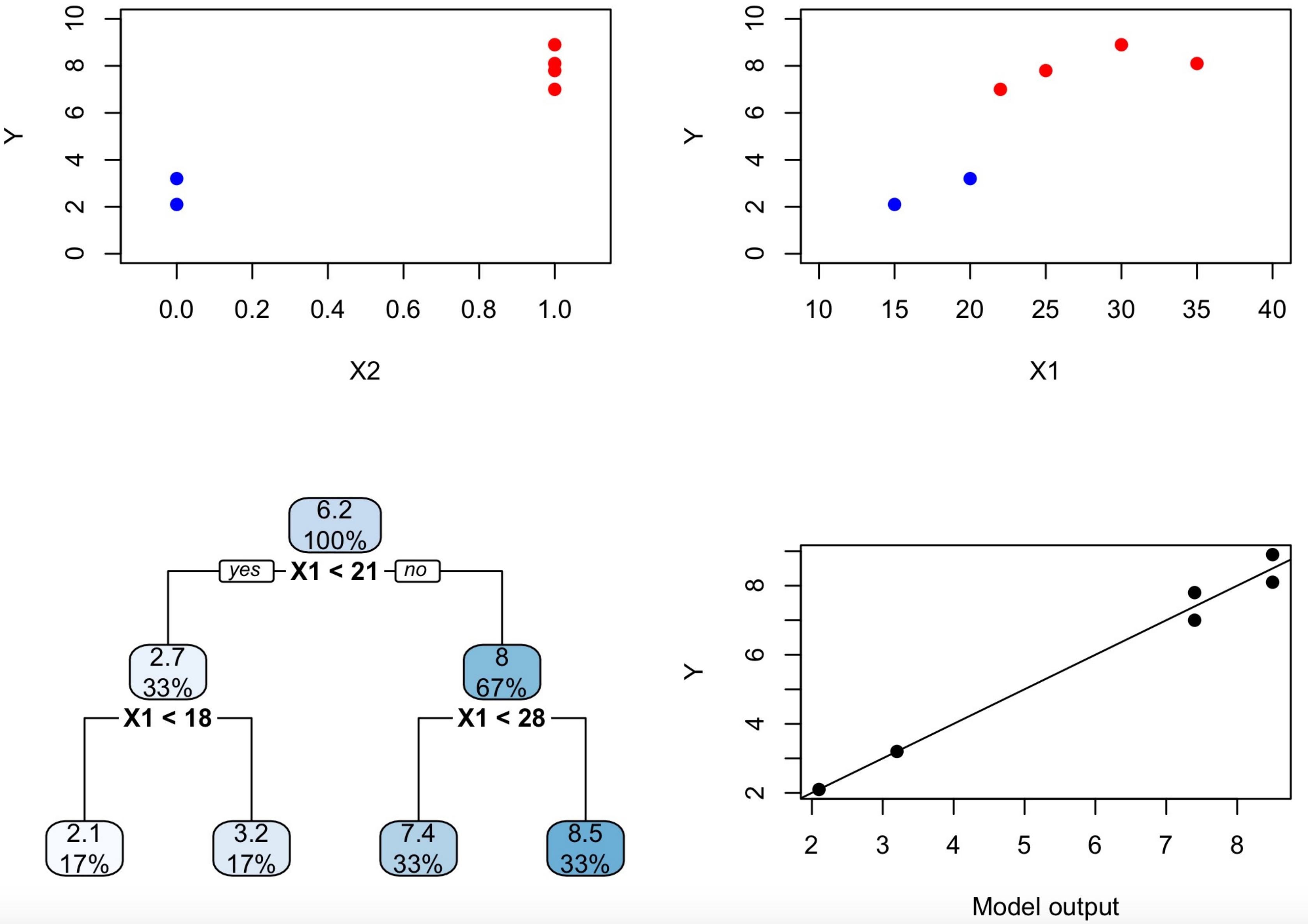
```r
library(rpart)
library(rpart.plot)

Model<-rpart(Y~X1+X2, data=Training, control=rpart.control(minsplit = 2))

rpart.plot(Model)
```



Y
X2
0.0
0.2
0.4
0.6
0.8
1.0
X1
10
15
20
25
30
35
40
5.5
100%
yes
X2 = 0
no
3.1
50%
7.9
50%
X1 < 18
X1 < 28
3.6
33%
X1 < 28
2.1
17%
3.2
17%
4.1
17%
7.4
33%
8.9
17%
Model output

Instability of trees

	X1	X2	Y		
1	15	0	2.1		
2	20	0	3.2		
3	22	1	7.0		
4	25	1	7.8		
5	30	1	8.9		
6	35	**0**	**4.1**	**1**	**8.1**


Y
X2
0.0 0.2 0.4 0.6 0.8 1.0
0 2 4 6 8 10
Y
X1
10 15 20 25 30 35 40
0 2 4 6 8 10
6.2
100%
yes
X1 < 21
no
2.7
33%
X1 < 18
8
67%
X1 < 28
2.1
17%
3.2
17%
7.4
33%
8.5
33%
Y
Model output
2 3 4 5 6 7 8
2 4 6 8

How to reduce the instability and improve the accuracy of the predictions?

- Bagging
- Boosting

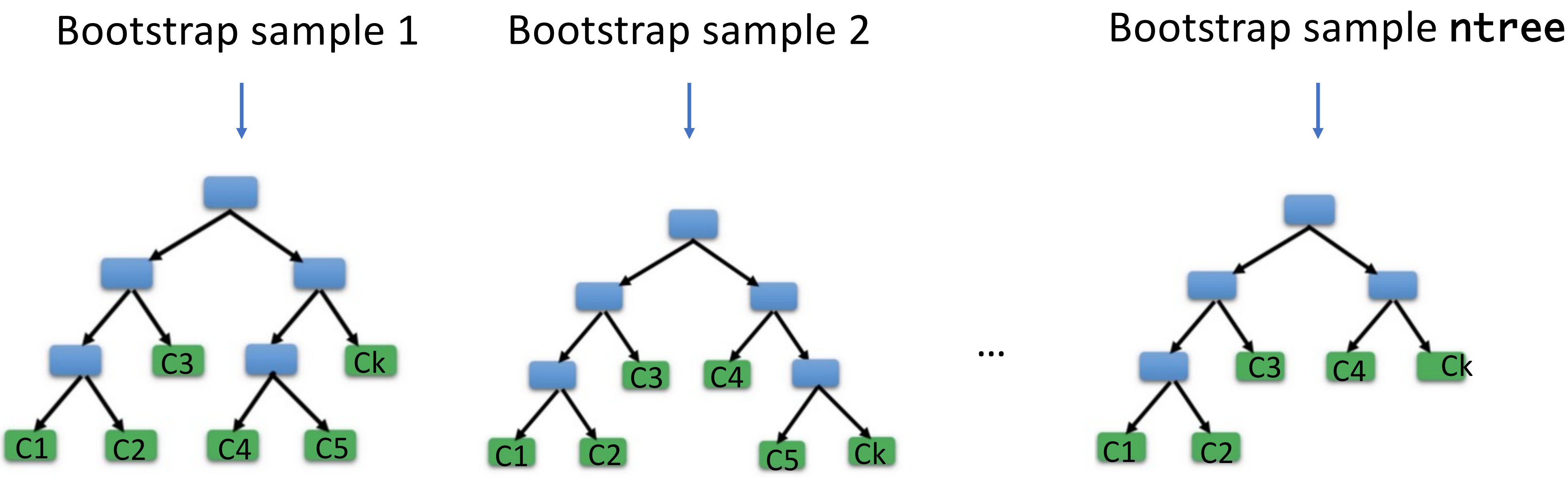
Bagging

- Resample *K* datasets from the original training dataset (bootstrap)
- Train a tree using each of the *K* datasets (and a subset of inputs)
- Average the *K* resulting trees

Hyper-parameters:

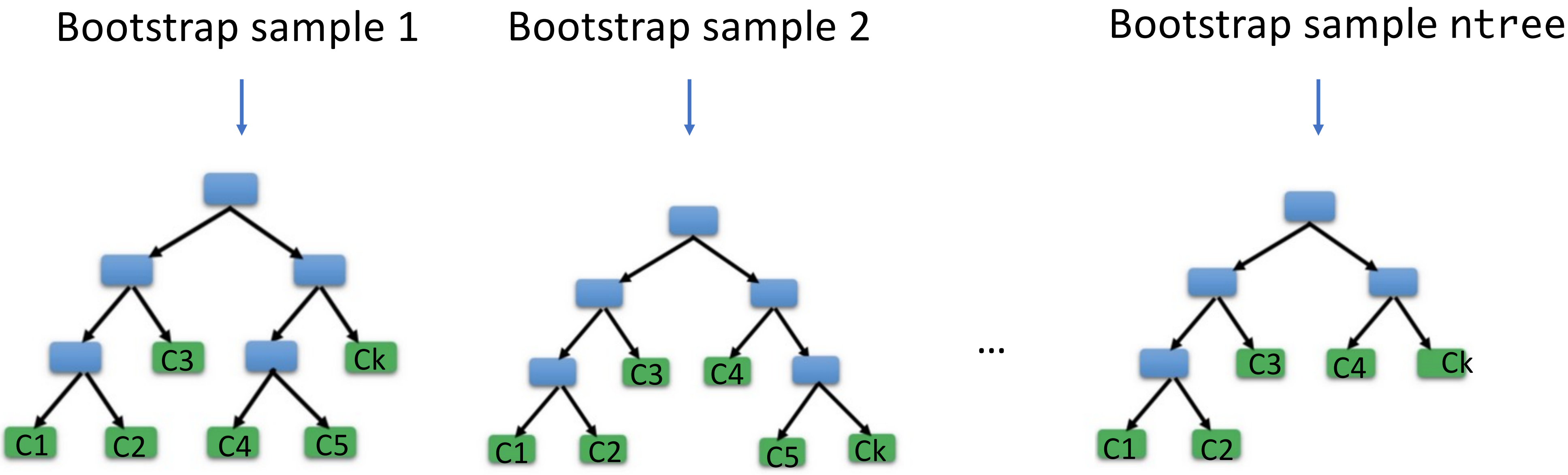
- Value of *K*,
- number of inputs (features) tested at each node of each tree.

Training



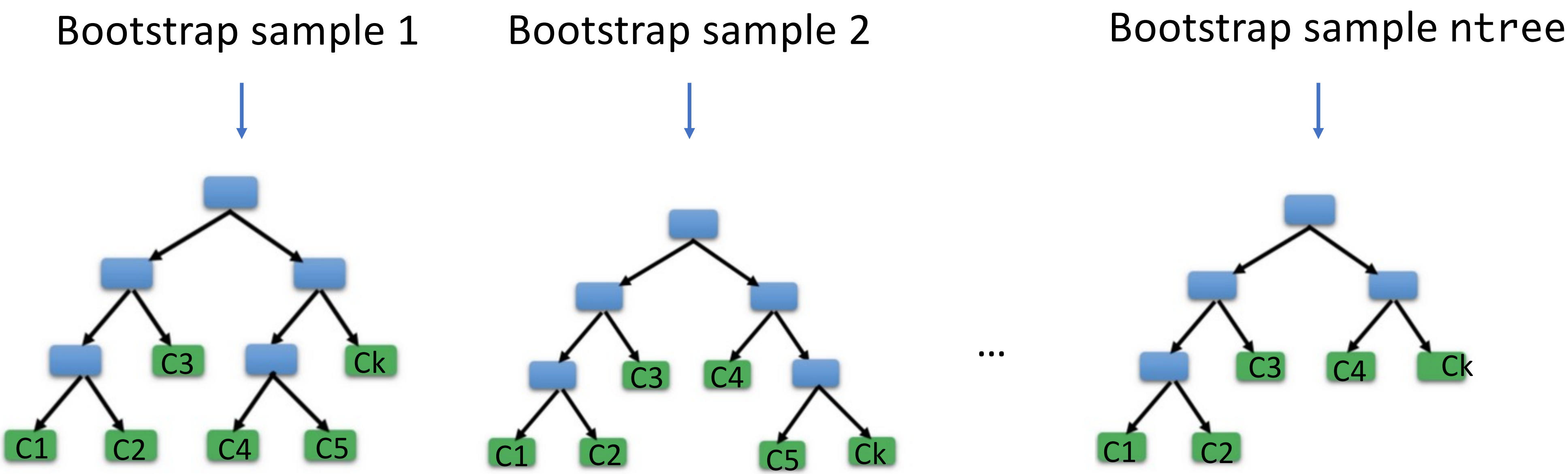
- We train **`ntrees` trees** from the training dataset
- Each tree is trained from a bootstrap sample of data
- The number `ntrees` needs to be optimized (often, 500 is enough)

Training



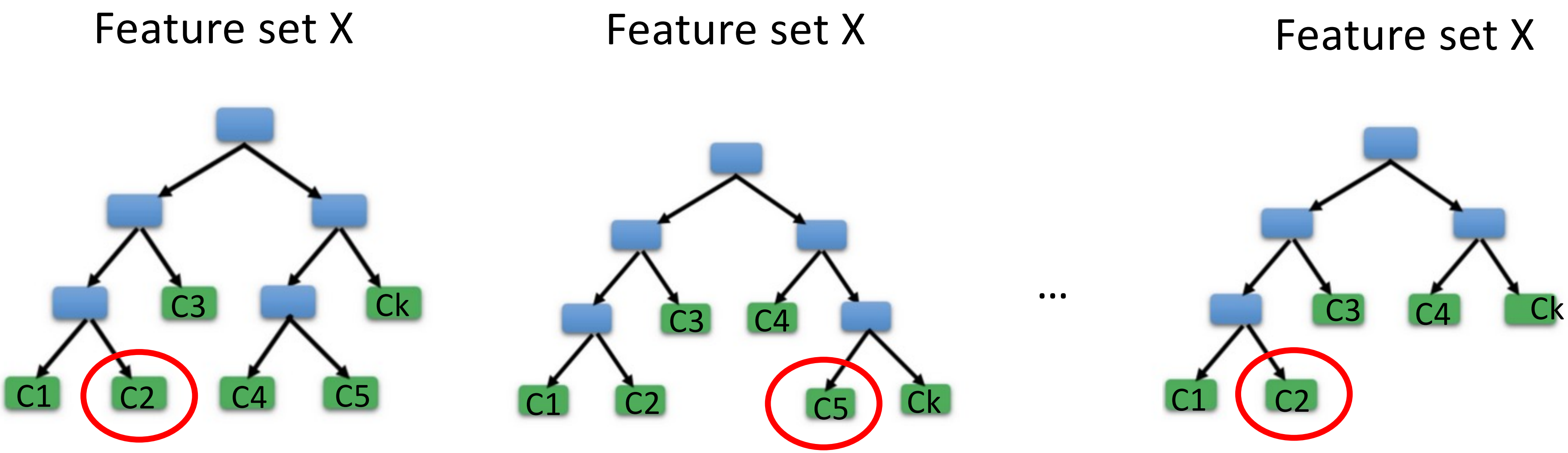
- Each terminal node includes N data (N=10 by default)
- Each terminal node returns one category
- The category of each node corresponds to the majority of the data of this node

Training



- Each split of each tree is based on one of the features (inputs) available
- This feature is selected to minimize the Gini impurity or MSE
- At each split, **only `mtry` features** are considered, randomly chosen among the whole set of features available
- The value of **`mtry`** is either set to its default value or optimized by cross-validation

Prediction with a forest



- For each X, get **`ntrees`** (ex:500) categories: c2, c5, c2, c1, ….
- Compute the proportions of each of the categories
- Prediction=category with the highest probability

Useful R packages

rpart	Trees
randomForest	Random forest
ranger	Fast implementation of random forest

Algorithm: fast implementation of random forest

- Not practical to use the standard `randomForest` R function due to the large size of the dataset
- Use of the `ranger` package

ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R

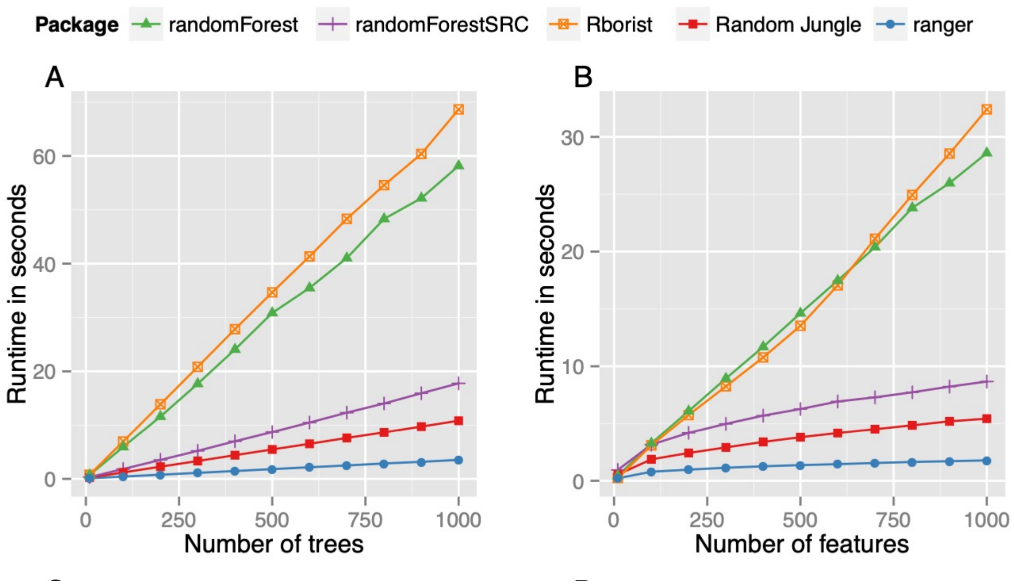
Marvin N. Wright
Universität zu Lübeck

Andreas Ziegler
Universität zu Lübeck,
University of KwaZulu-Natal

10.18637/jss.v077.i01

Fast implementation of random forest

- Not practical to use the standard `randomForest` R function due to the large size of the dataset
- Use of the `ranger` package

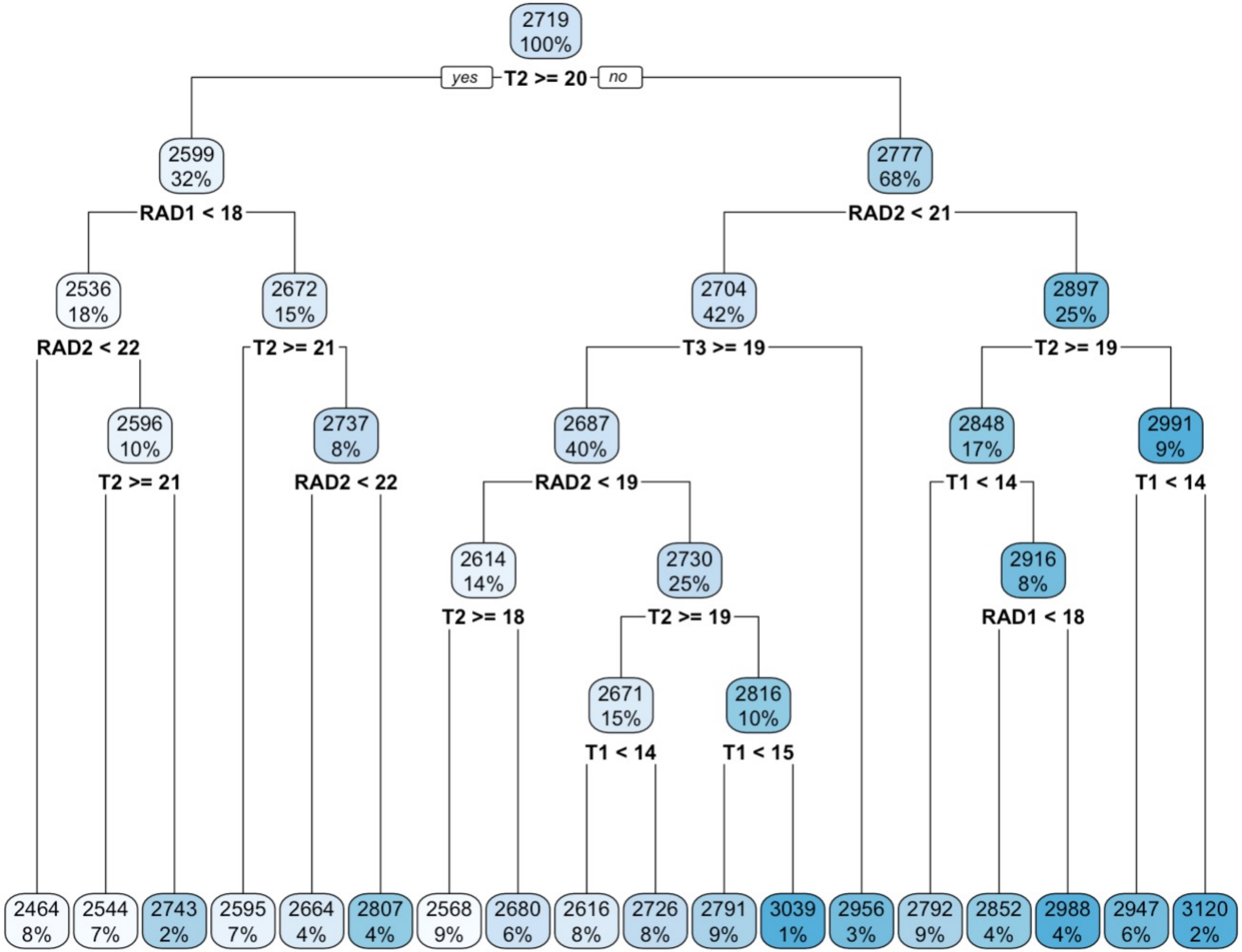


[10.18637/jss.v077.i01](https://doi.org/10.18637/jss.v077.i01)

Example: maize biomass

https://github.com/davemakowski/TP_machinelearning

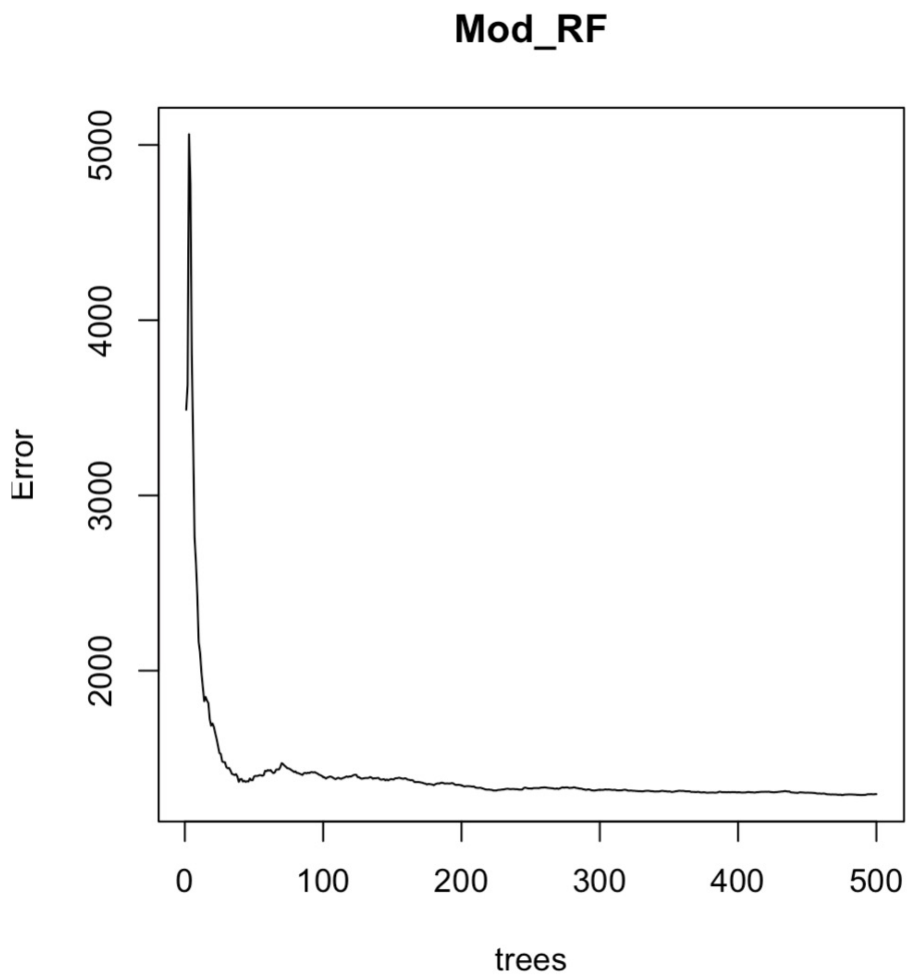
```
####Regression tree
library(rpart)
library(rpart.plot)

Mod_tree<-rpart(B~T1+T2+T3+RAD1+RAD2+RAD3,data=DataSet)
print(Mod_tree)
#dev.new()
par(mfrow=c(1,1))
rpart.plot(Mod_tree)
```


2719
100%
yes
T2 >= 20
no
2599
32%
RAD1 < 18
2777
68%
RAD2 < 21
2536
18%
RAD2 < 22
2672
15%
T2 >= 21
2704
42%
T3 >= 19
2897
25%
T2 >= 19
2596
10%
T2 >= 21
2737
8%
RAD2 < 22
2687
40%
RAD2 < 19
2848
17%
T1 < 14
2991
9%
T1 < 14
2614
14%
T2 >= 18
2730
25%
T2 >= 19
2916
8%
RAD1 < 18
2671
15%
T1 < 14
2816
10%
T1 < 15
2464
8%
2544
7%
2743
2%
2595
7%
2664
4%
2807
4%
2568
9%
2680
6%
2616
8%
2726
8%
2791
9%
3039
1%
2956
3%
2792
9%
2852
4%
2988
4%
2947
6%
3120
2%

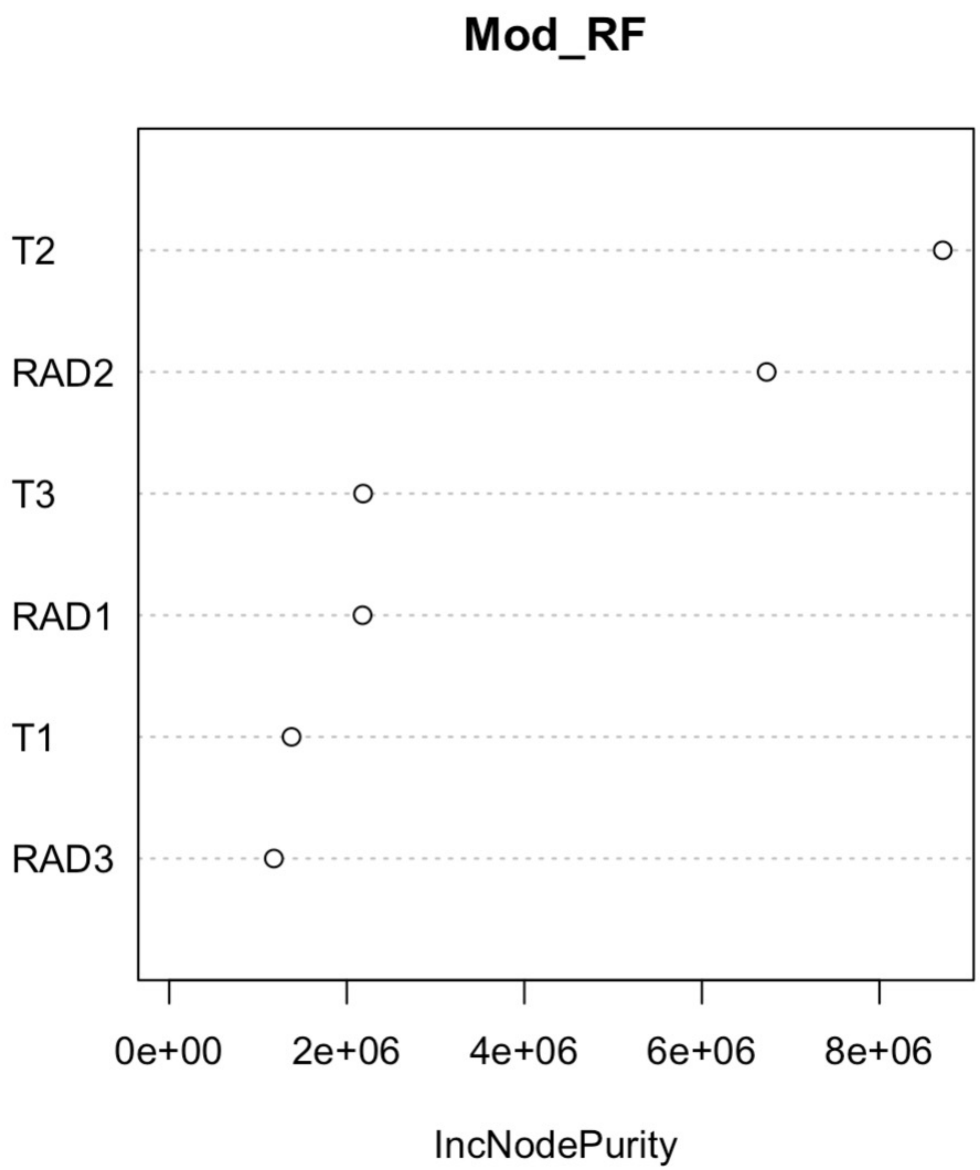
```
library(randomForest)
Mod_RF<-randomForest(B~T1+T2+T3+RAD1+RAD2+RAD3,data=DataSet,ntree=500, mtry=6)
Mod_RF
```

```
Call:
 randomForest(formula = B ~ T1 + T2 + T3 + RAD1 + RAD2 + RAD3,      data = DataSet, ntree = 500, mtry = 6)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 1295.703
                    % Var explained: 96.08
> 
```

plot(Mod_RF)



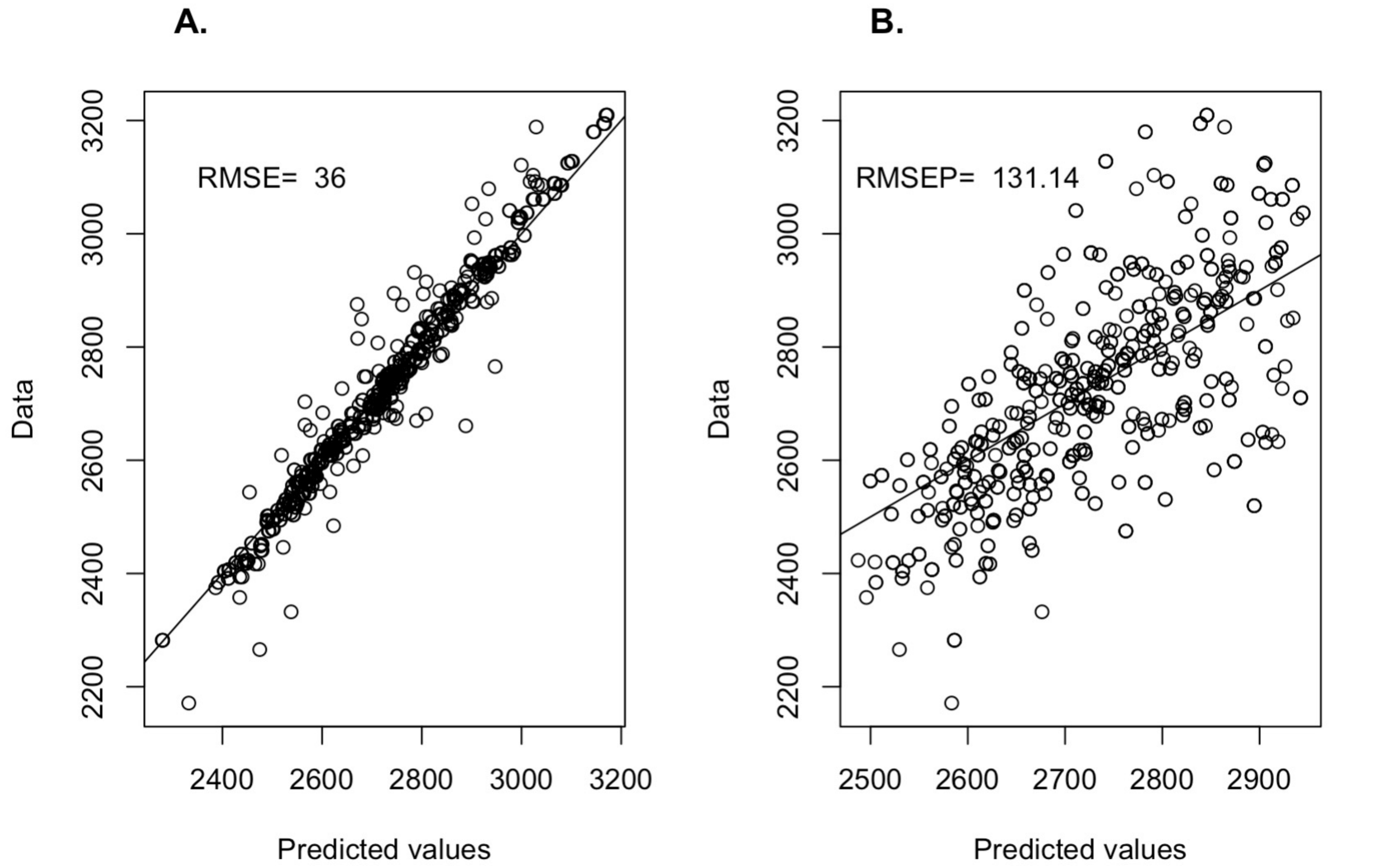
varImpPlot(Mod_RF,type=2)



```
RMSE_rf<-sqrt(mean((DataSet$B-predict(Mod_RF))^2))
RMSE_rf

#Cross-validation
B_pred_rf<-rep(NA,length(DataSet$B))

List_year<-unique(DataSet$Year)

for (i in 1:length(List_year))
{
Training_i<-DataSet[DataSet$Year!=List_year[i],]
Test_i<-DataSet[DataSet$Year==List_year[i],]
Mod_i<-randomForest(B~T1+T2+T3+RAD1+RAD2+RAD3,data=Training_i, ntree=200)
B_rf_i<-predict(Mod_i, newdata=Test_i)
B_pred_rf[DataSet$Year==List_year[i]]<-B_rf_i
}

RMSEP_rf<-sqrt(mean((DataSet$B-B_pred_rf)^2))
RMSEP_rf
```

```
> RMSE_rf
[1] 35.99588

> RMSEP_rf
[1] 131.1415
```

```
par(mfrow=c(1,2))
plot(predict(Mod_RF),DataSet$B, xlab="Predicted values", ylab="Data")
abline(0,1)
title("A.                                         ")
text(2500,3100,paste("RMSE= ", round(RMSE_rf, digits=2)))
plot(B_pred_rf,DataSet$B, xlab="Predicted values", ylab="Data")
abline(0,1)
title("B.                                         ")
text(2600,3100,paste("RMSEP= ", round(RMSEP_rf, digits=2)))
```


A.
RMSE= 36
Data
Predicted values
B.
RMSEP= 131.14
Data
Predicted values
2200
2400
2600
2800
3000
3200
2500
2600
2700
2800
2900

Main challenges in machine learning projects

- Choose a relevant question (Which Y? Which X?)
- Find reliable data
- Calibrate the hyper-parameters
- Assess prediction accuracy without bias
- Optimize computation time
- Vizualisation of output responses