

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)

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

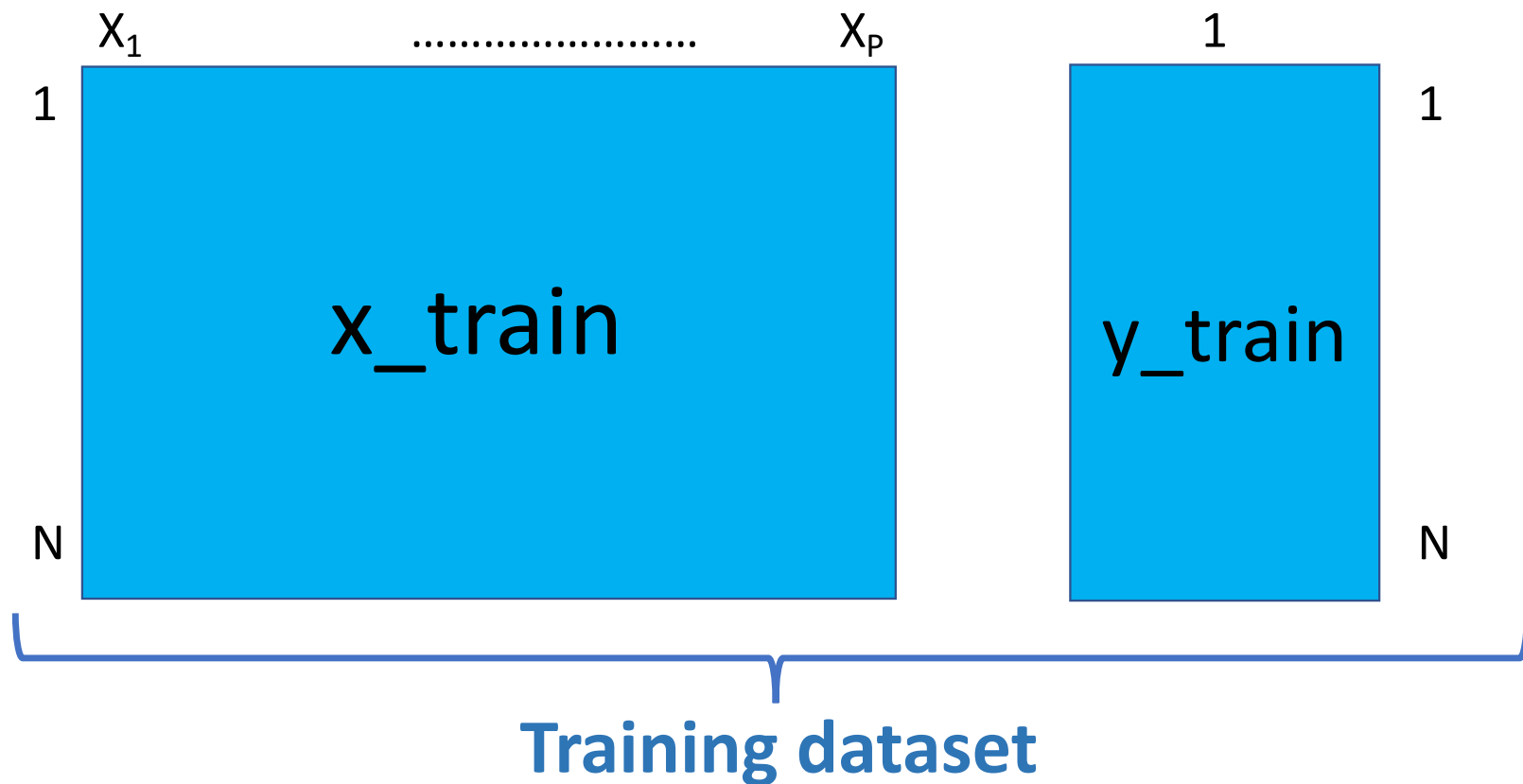
Modelling approach 1: Try to find the true  $f(x)$

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

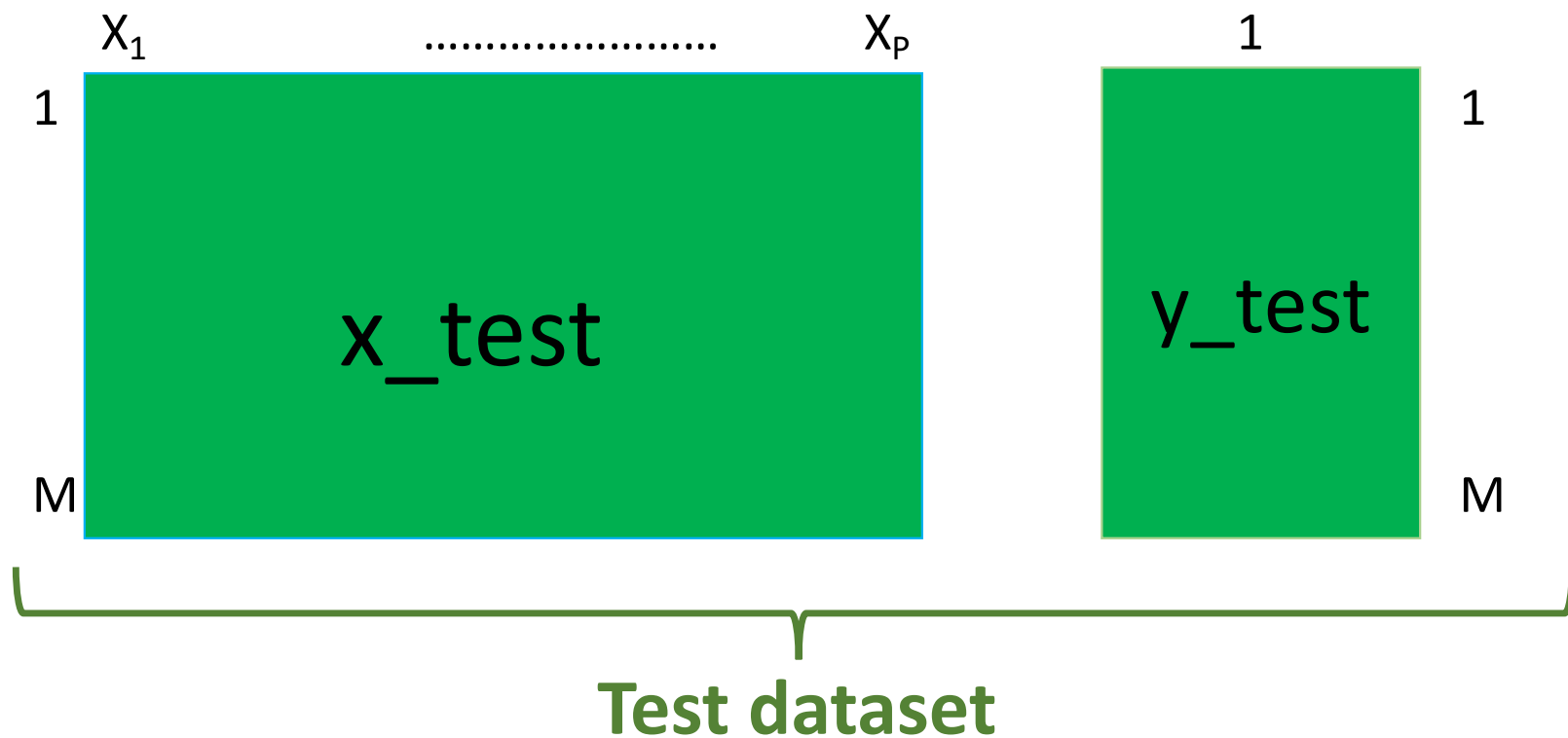
## Two main steps

- Step 1: Training
- Step 2: Test

**Step 1:** Train an algorithm predicting  $Y$  as a function of  $X_1, \dots, X_p$  using a **training dataset**

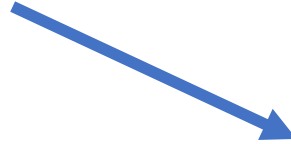
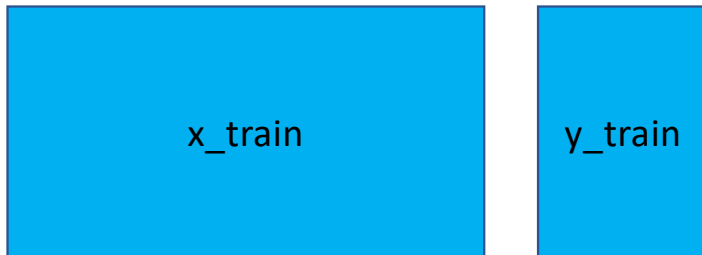


**Step 2:** Assess the predictive capability of the trained algorithm using a **test dataset**

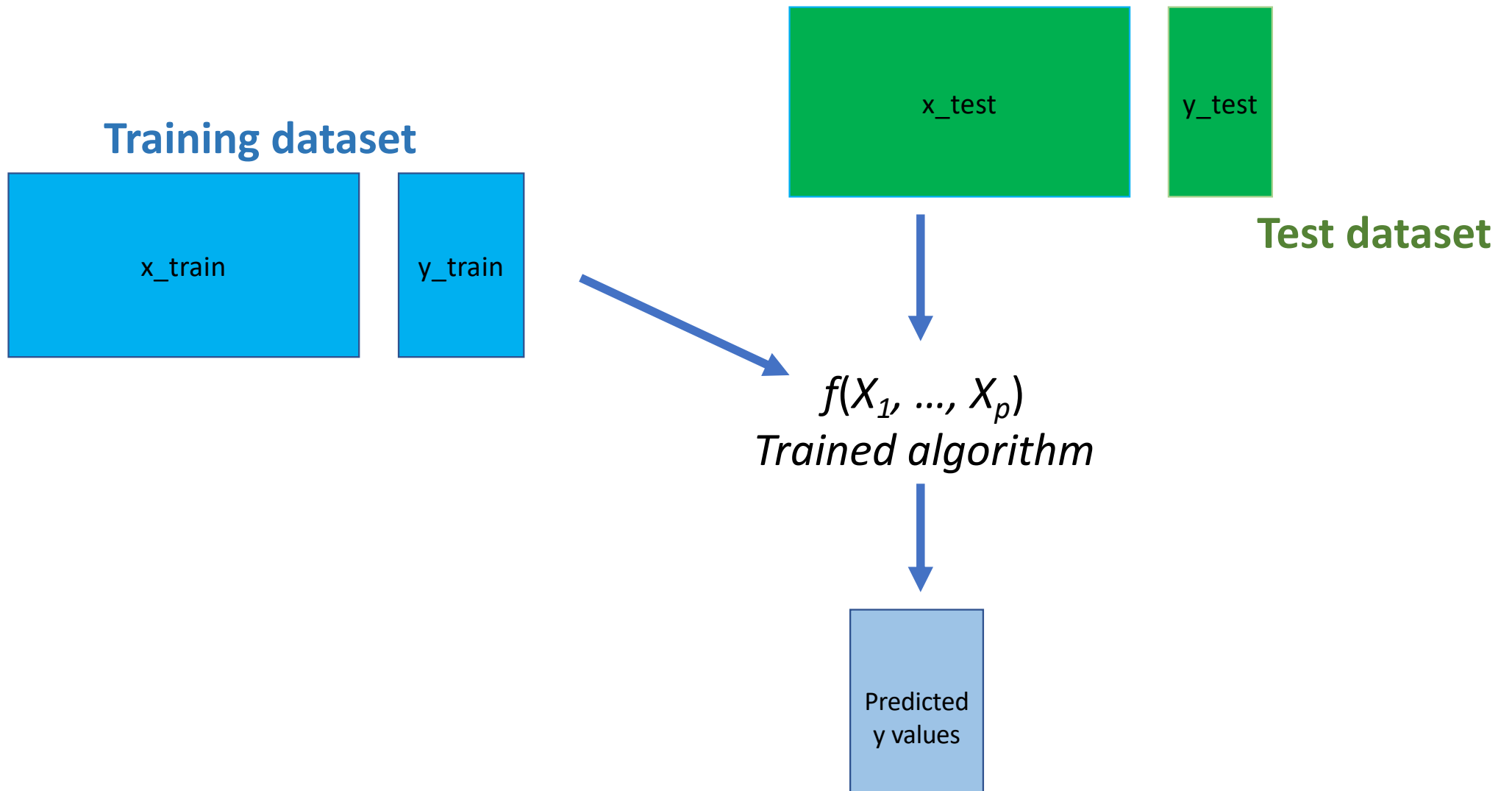


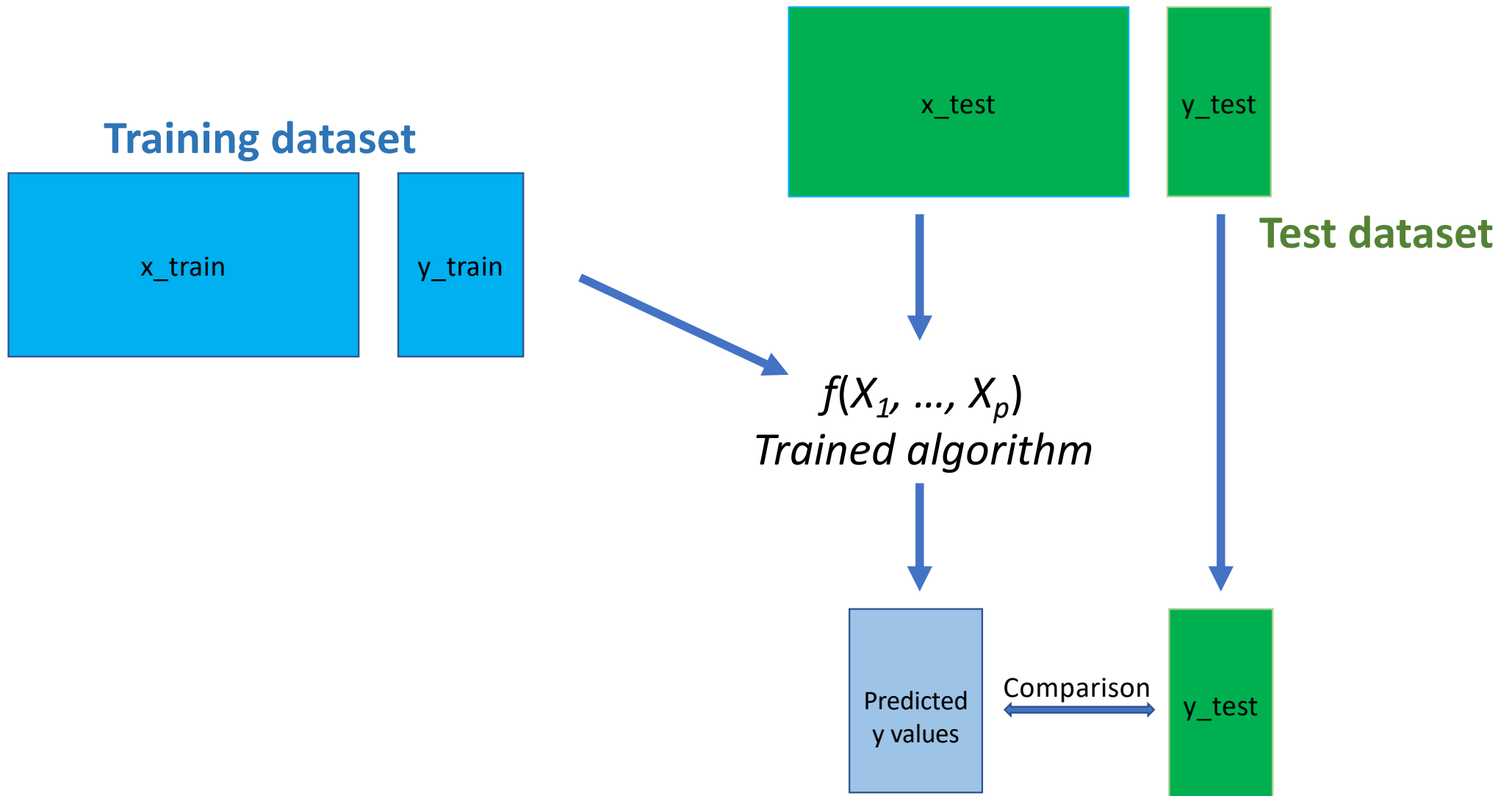


## Training dataset

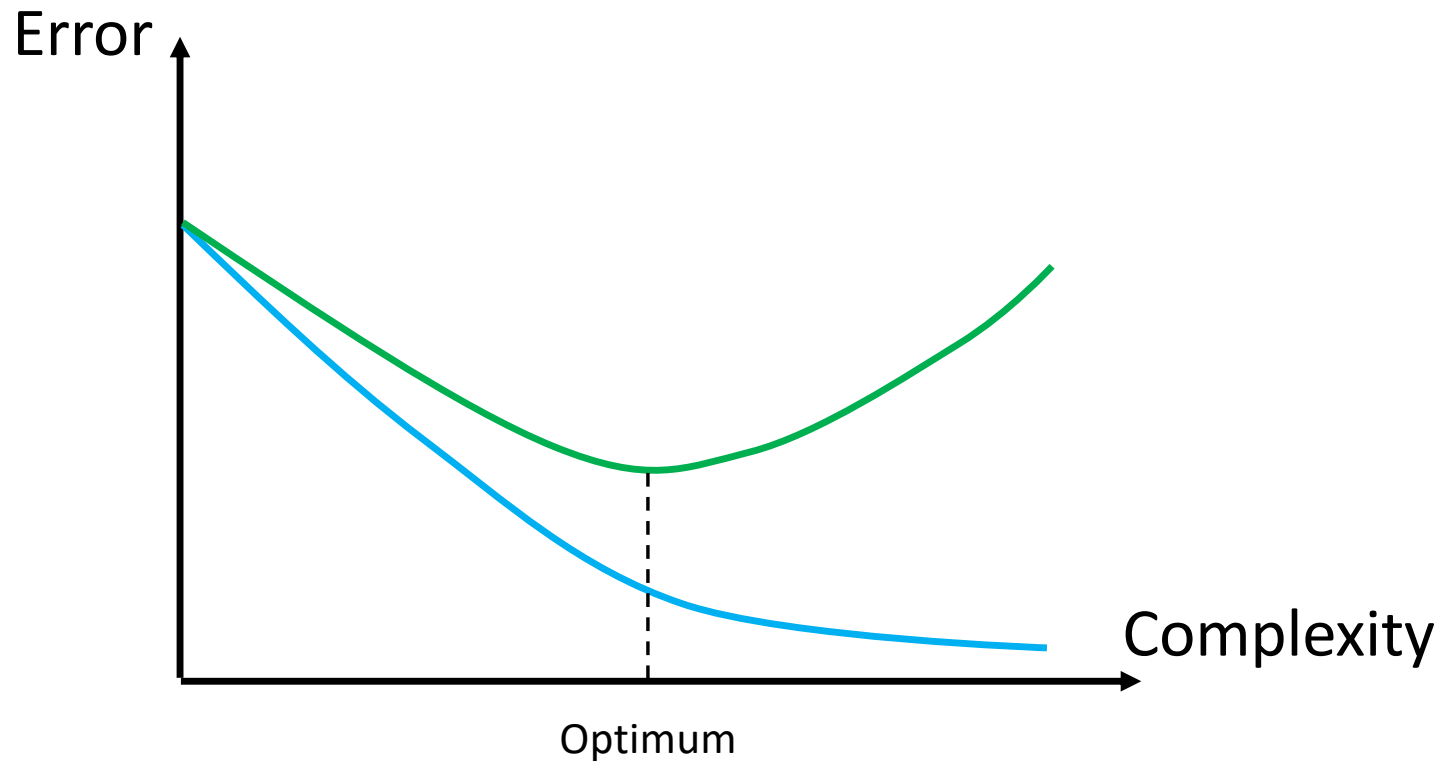


$f(X_1, \dots, X_p)$   
*Trained algorithm*

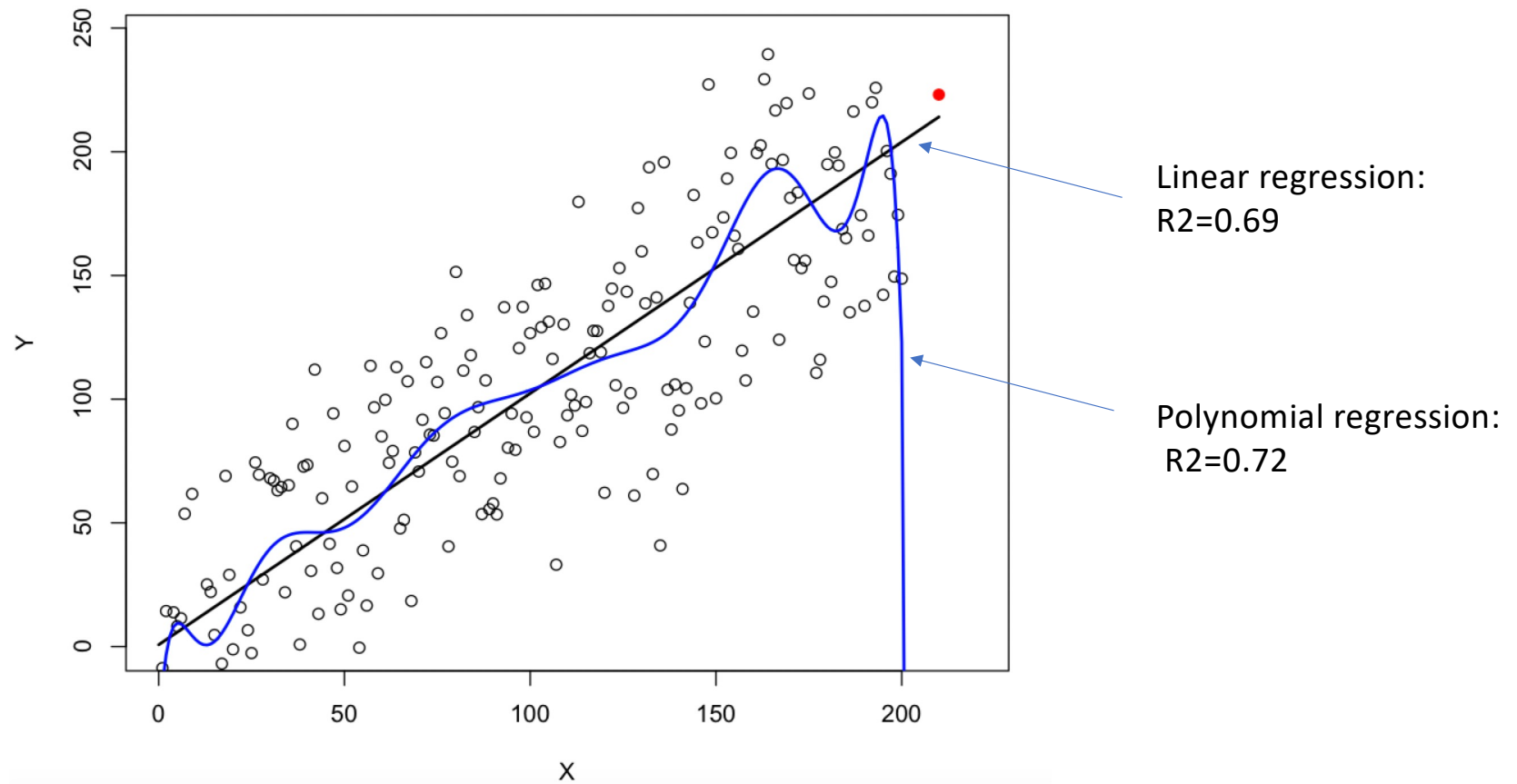




Model testing should be taken seriously to avoid risk of overfitting

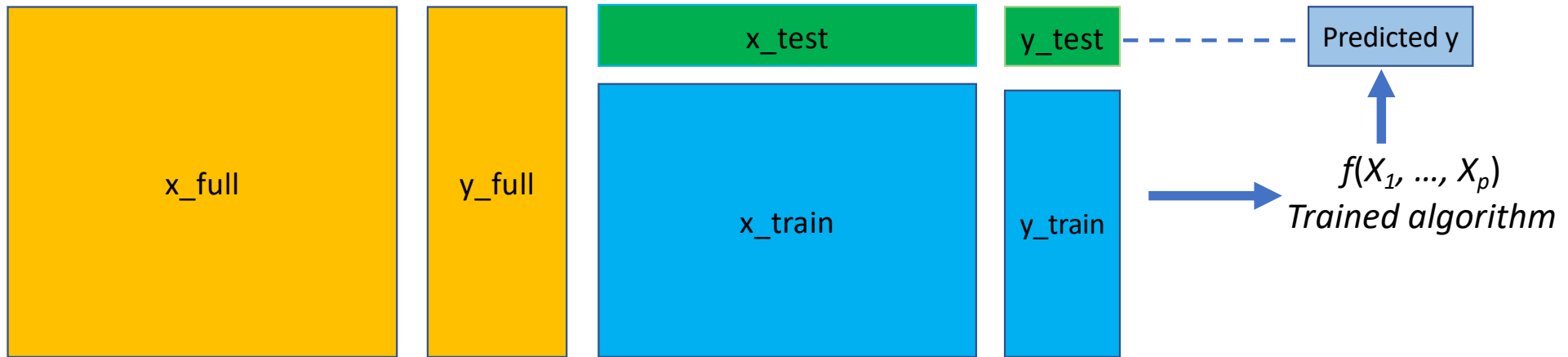


Model testing should be taken seriously to avoid risk of overfitting

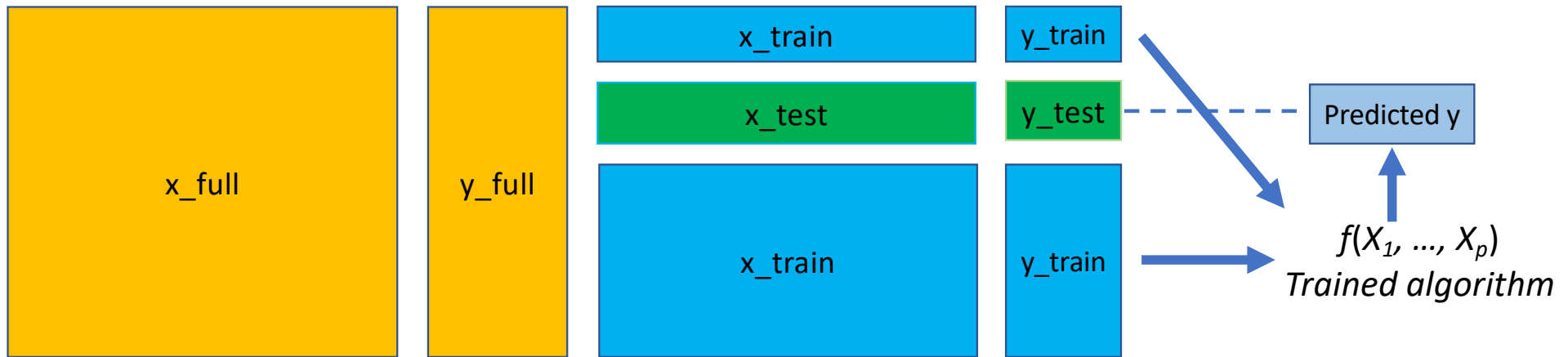


Cross-validation is used when no independent test dataset is available

## Full dataset

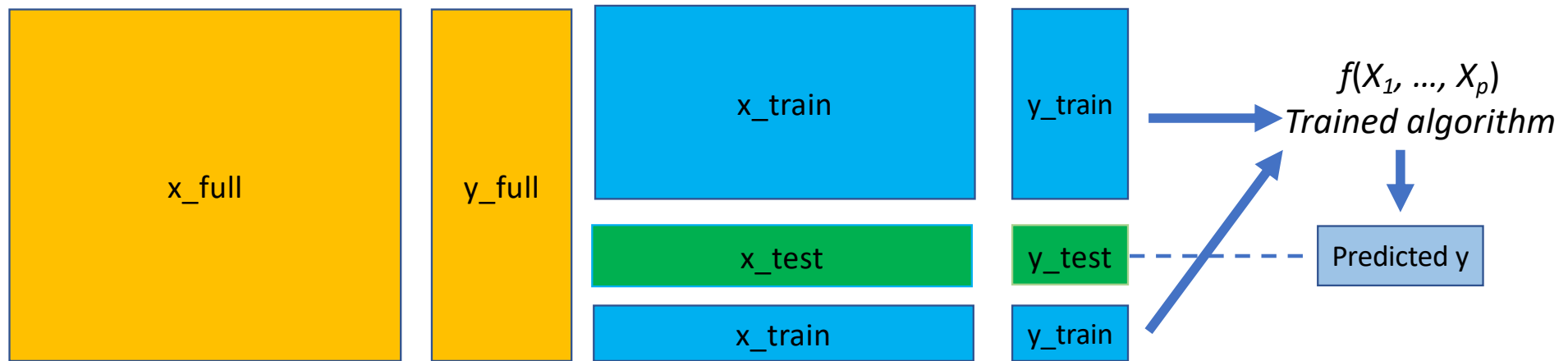


## Full dataset

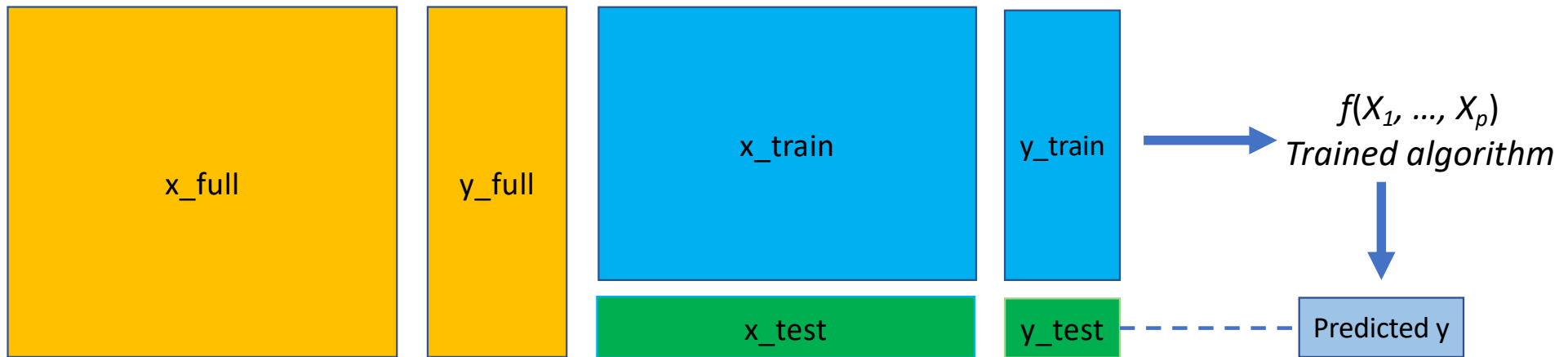




## Full dataset

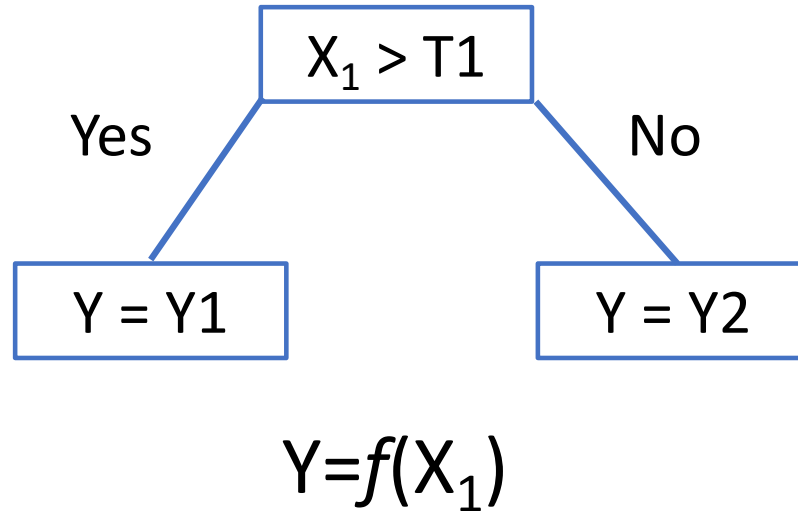


## Full dataset



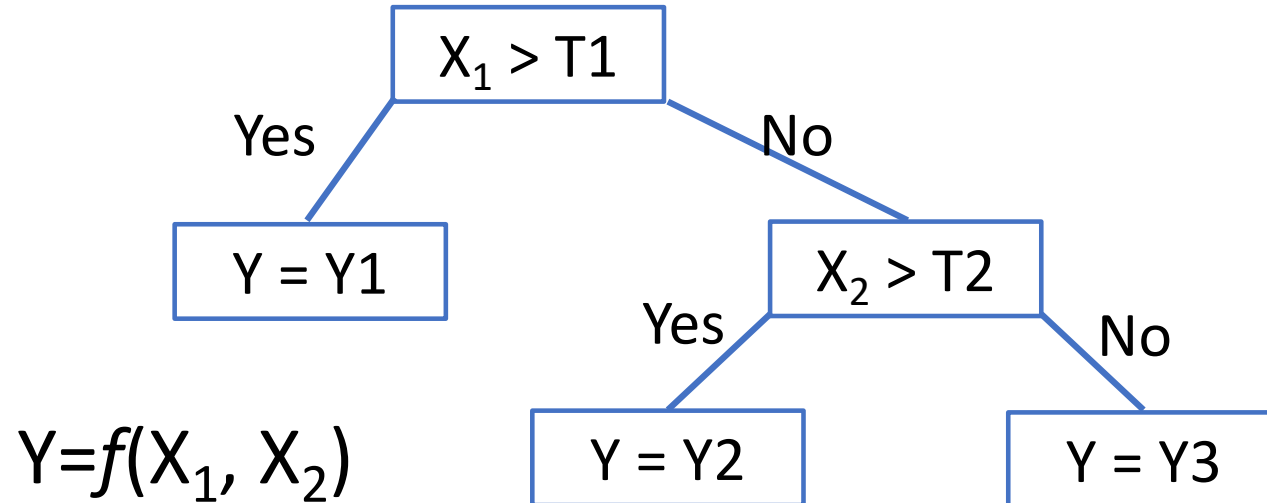
# Tree

Tree = Model based on a series of splitting rules



# Tree

Tree = Model based on a series of splitting rules



# 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

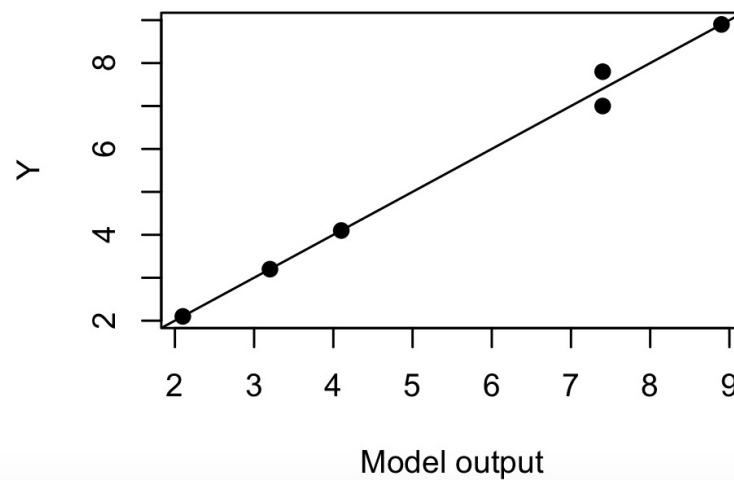
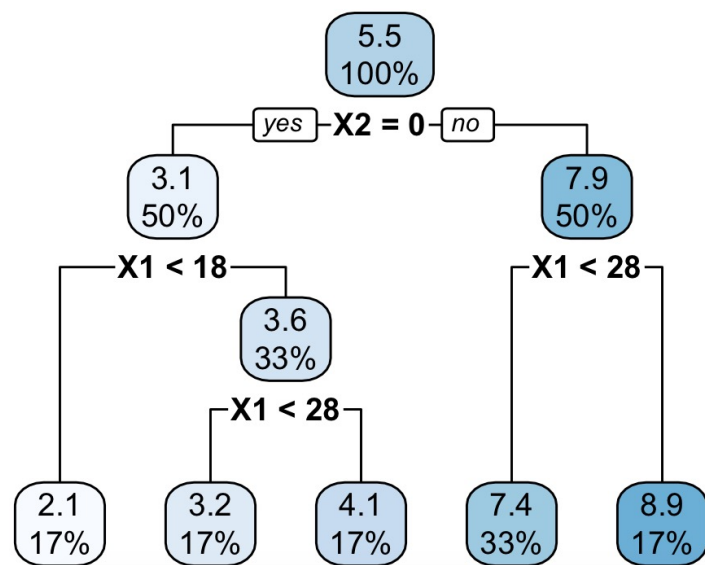
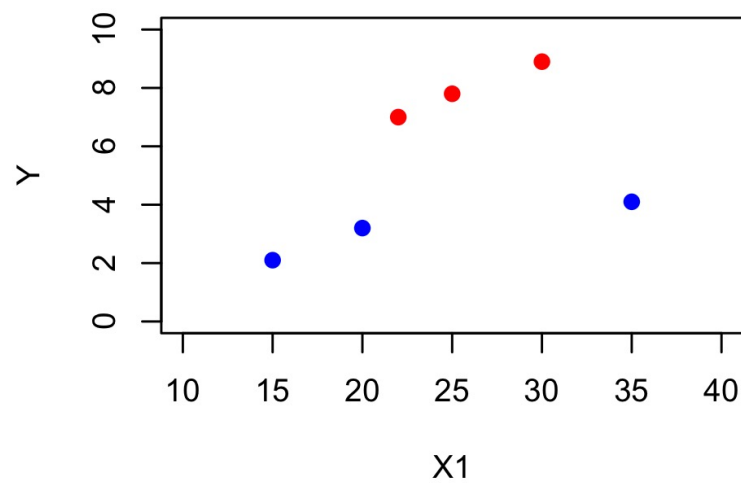
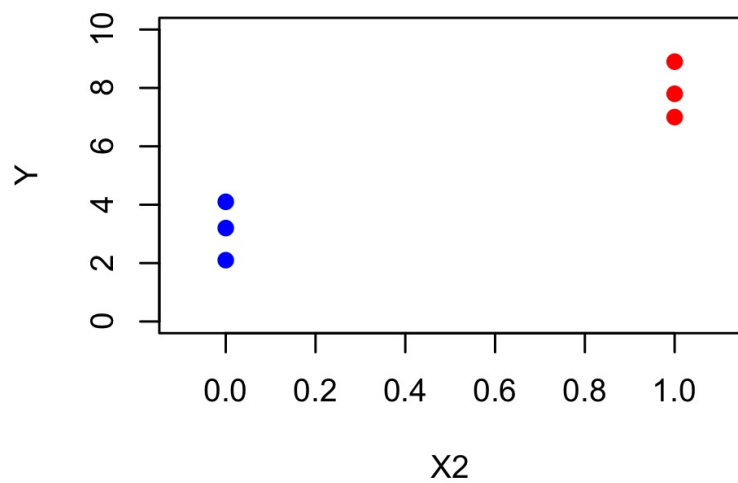
## A « toy » example

	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

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


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

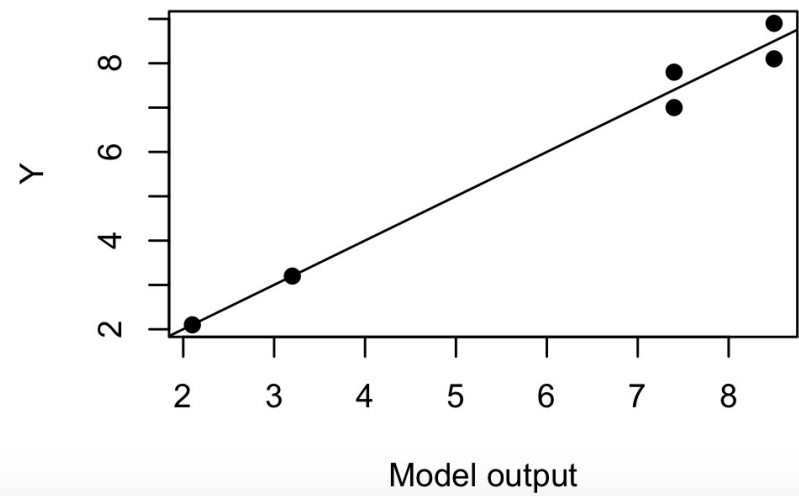
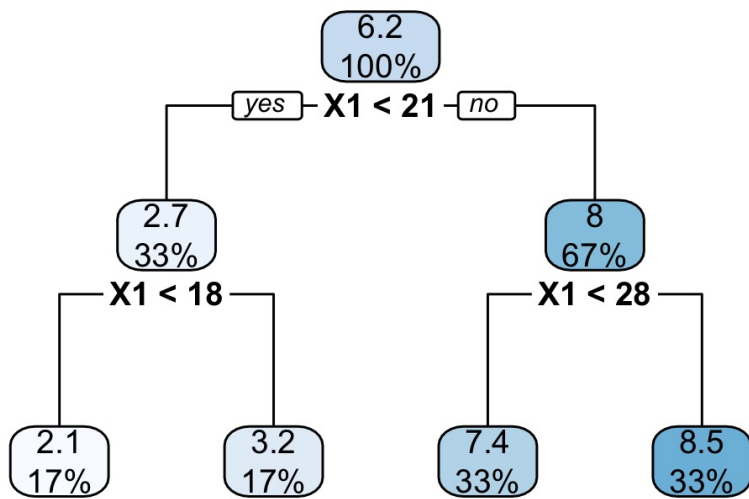
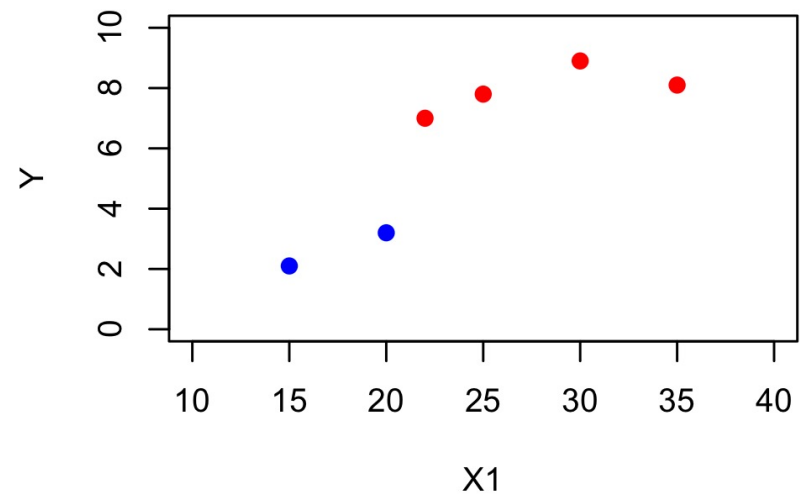
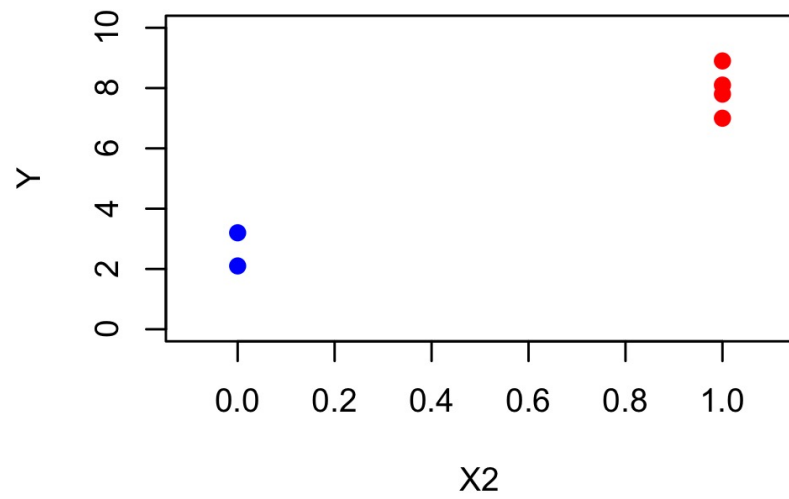
rpart.plot(Model)
```





# 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	<b>0</b>	<b>4.1</b>		<b>1      8.1</b>



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

- Bagging
- Boosting

# Bagging

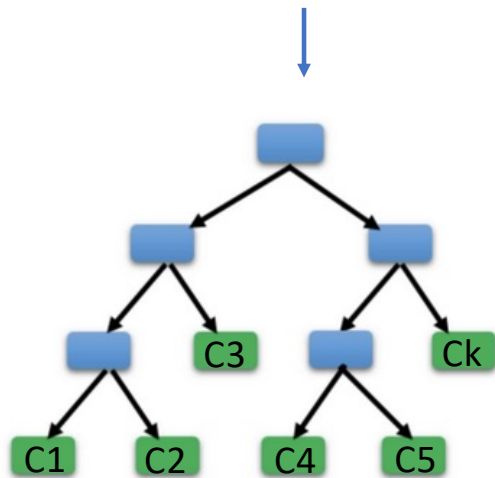
- Resample  $K$  datasets from the original training dataset (bootstrap)
- Train a tree using each of the  $K$  datasets (and a subset of inputs)
- Average the  $K$  resulting trees

Hyper-parameters:

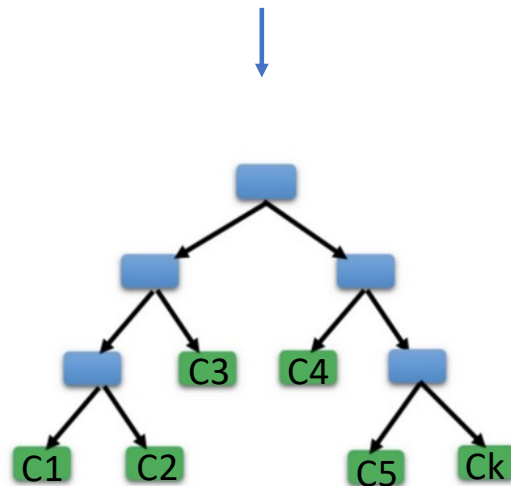
- Value of  $K$ ,
- number of inputs (features) tested at each node of each tree.

# Training

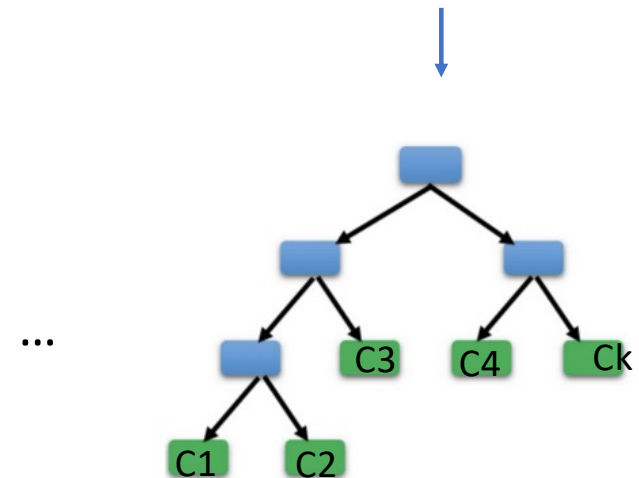
Bootstrap sample 1



Bootstrap sample 2



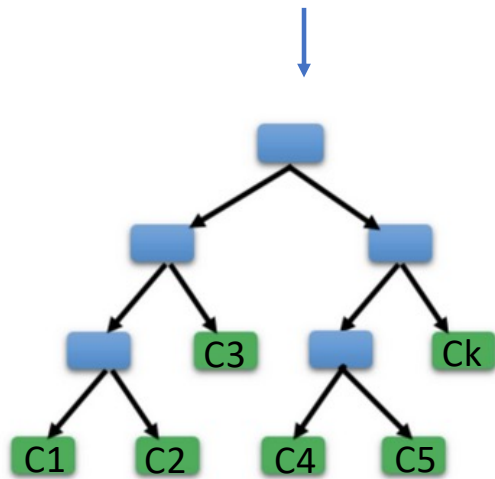
Bootstrap sample  $ntree$



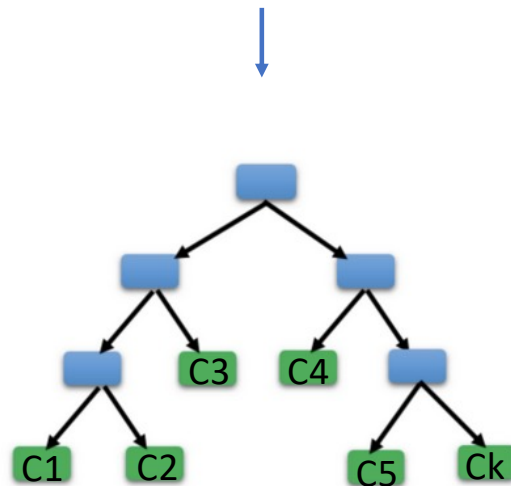
- We train  **$ntrees$  trees** from the training dataset
- Each tree is trained from a bootstrap sample of data
- The number  $ntrees$  needs to be optimized (often, 500 is enough)

# Training

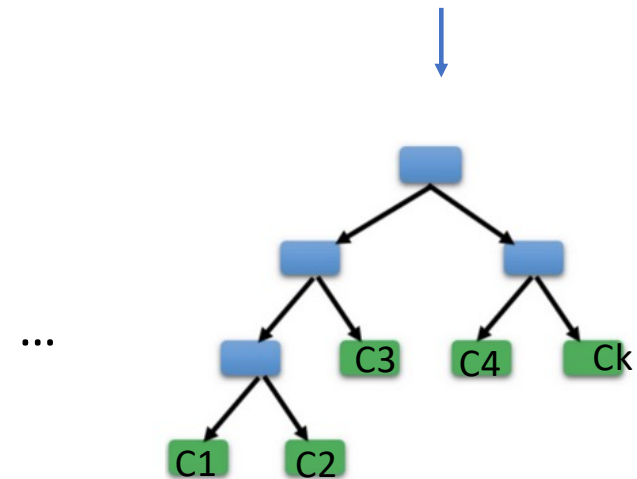
Bootstrap sample 1



Bootstrap sample 2



Bootstrap sample ntree

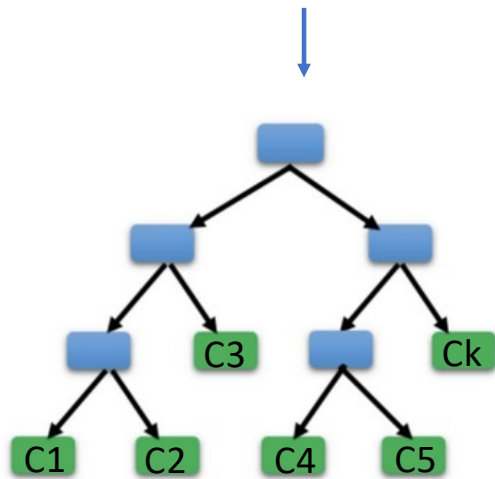


...

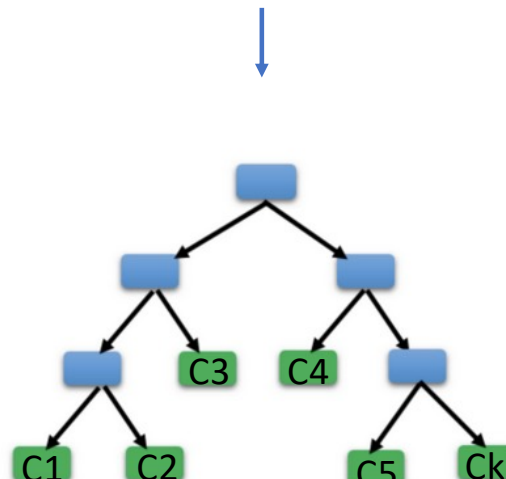
- Each terminal node includes N data (N=10 by default)
- Each terminal node returns one category
- The category of each node corresponds to the majority of the data of this node

# Training

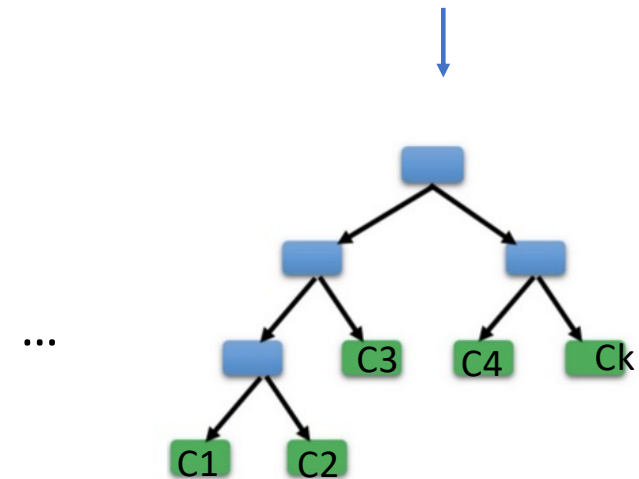
Bootstrap sample 1



Bootstrap sample 2

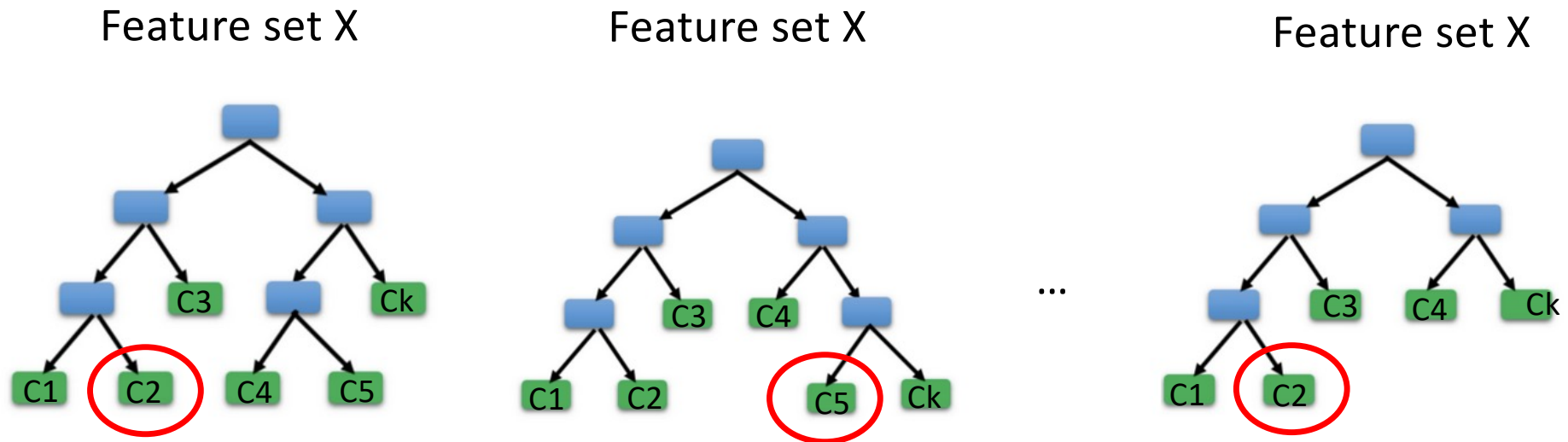


Bootstrap sample ntree



- 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**

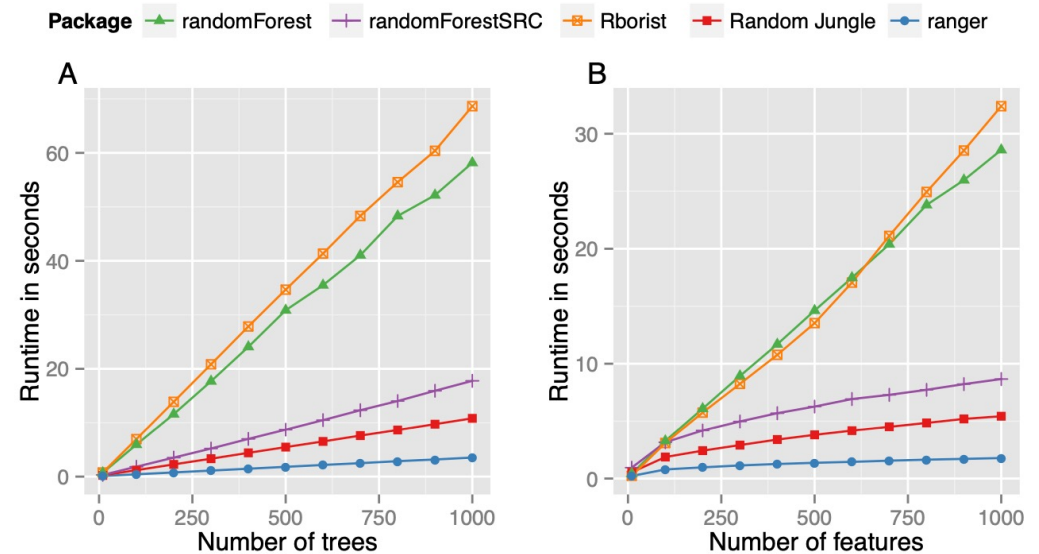
Marvin N. Wright  
Universität zu Lübeck

Andreas Ziegler  
Universität zu Lübeck,  
University of KwaZulu-Natal

[10.18637/jss.v077.i01](https://doi.org/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](https://github.com/davemakowski/TP_machinelearning)

```
####Regression tree
```

```
library(rpart)
```

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

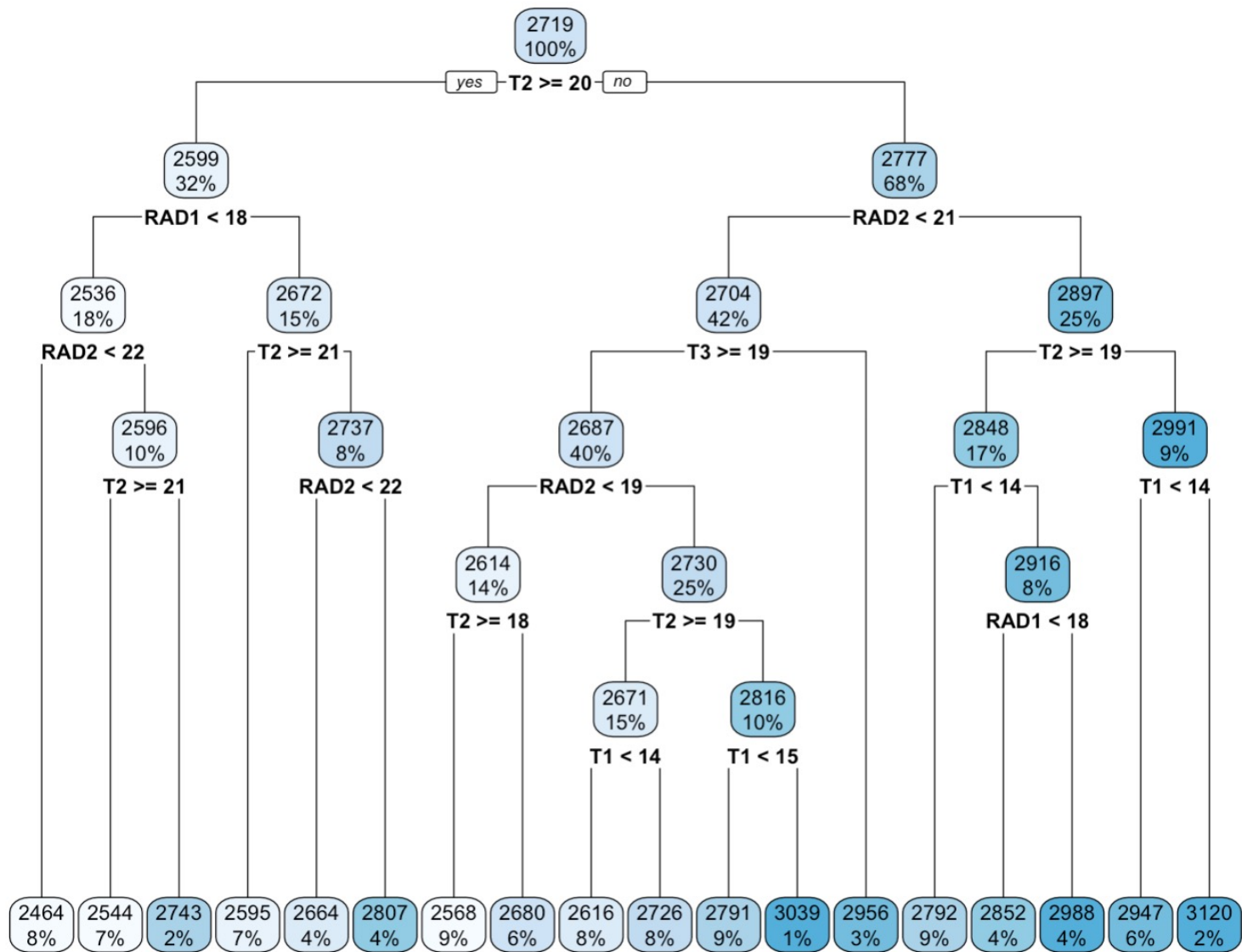
```
Mod_tree<-rpart(B~T1+T2+T3+RAD1+RAD2+RAD3,data=DataSet)
```

```
print(Mod_tree)
```

```
#dev.new()
```

```
par(mfrow=c(1,1))
```

```
rpart.plot(Mod_tree)
```



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

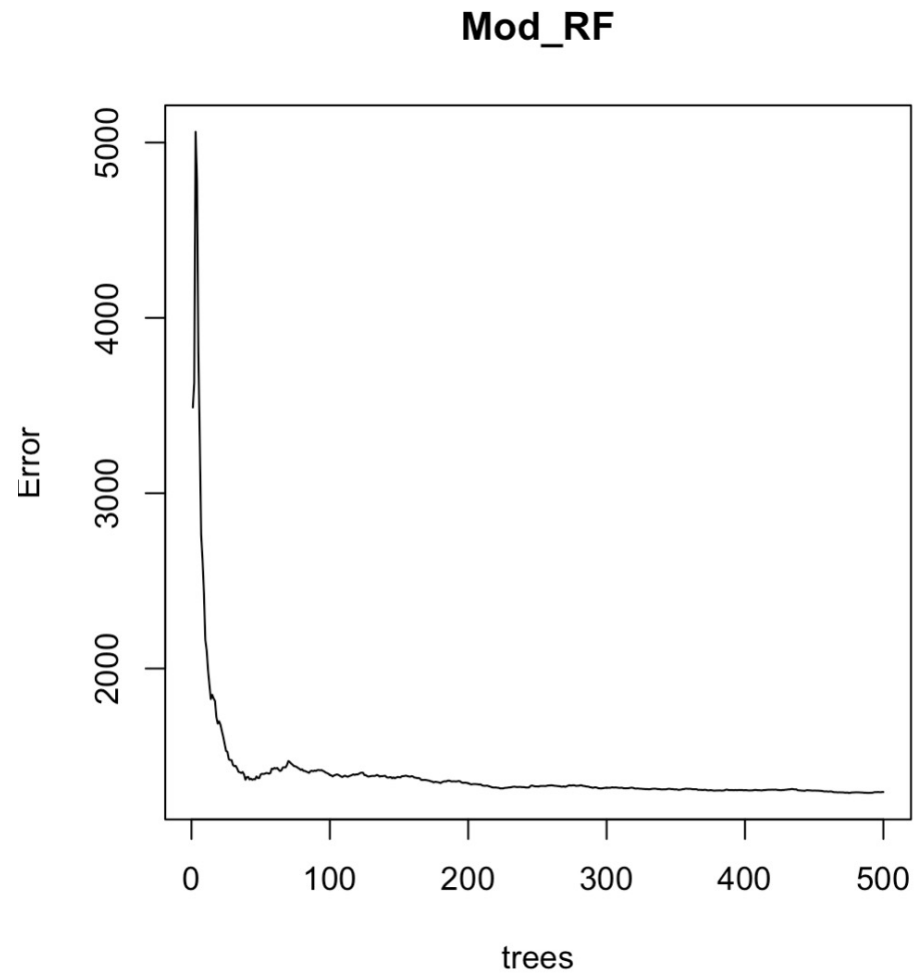
Call:

```
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
```

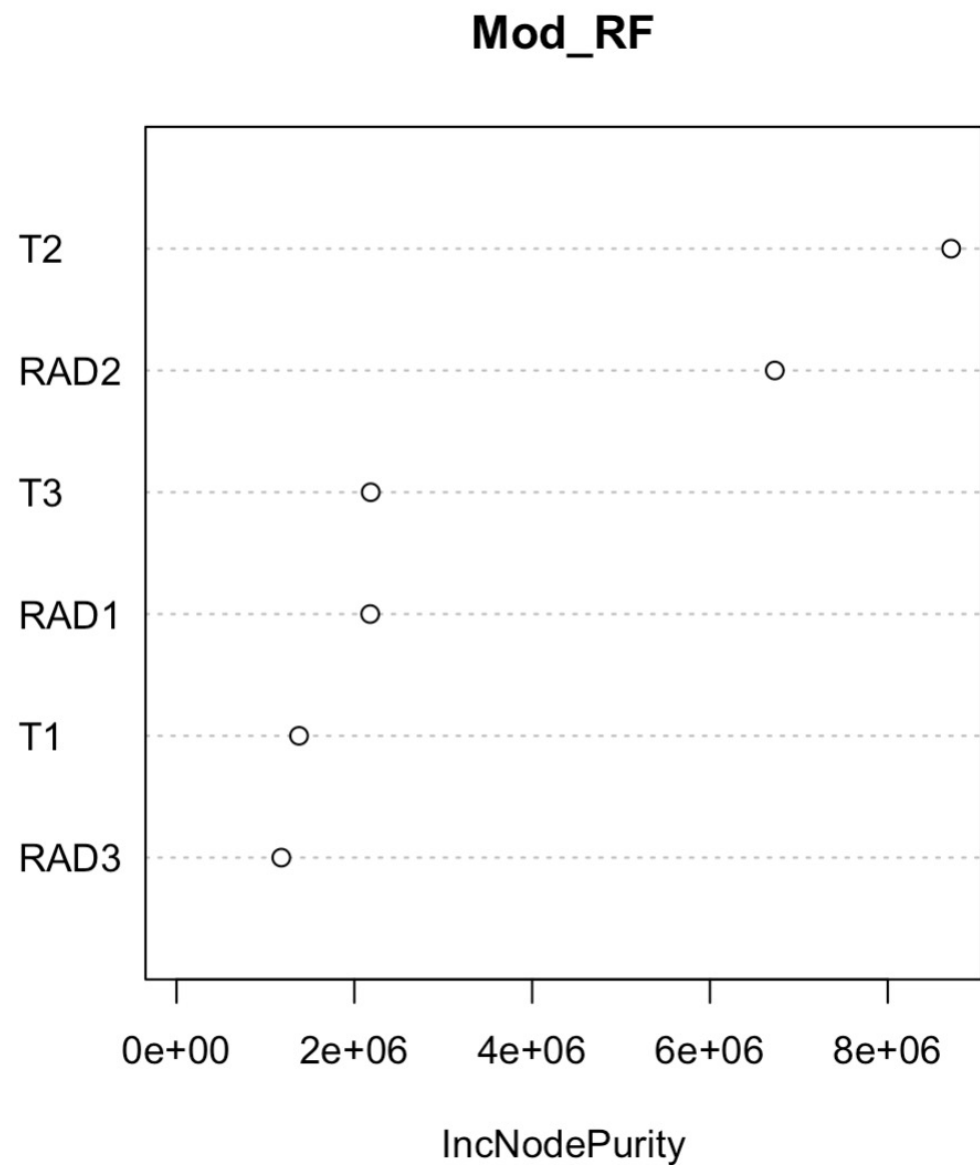
> |

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
}
```

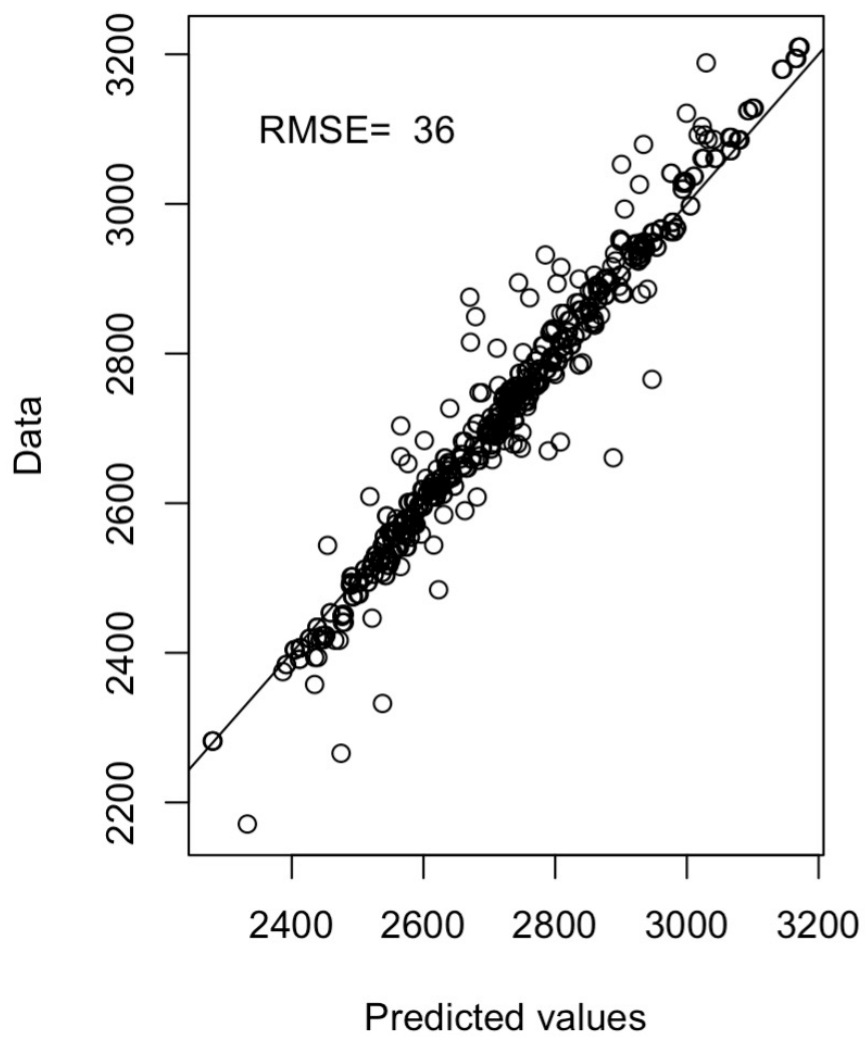
```
RMSEP_rf<-sqrt(mean((DataSet$B-B_pred_rf)^2))
RMSEP_rf
```

```
> RMSE_rf
[1] 35.99588
```

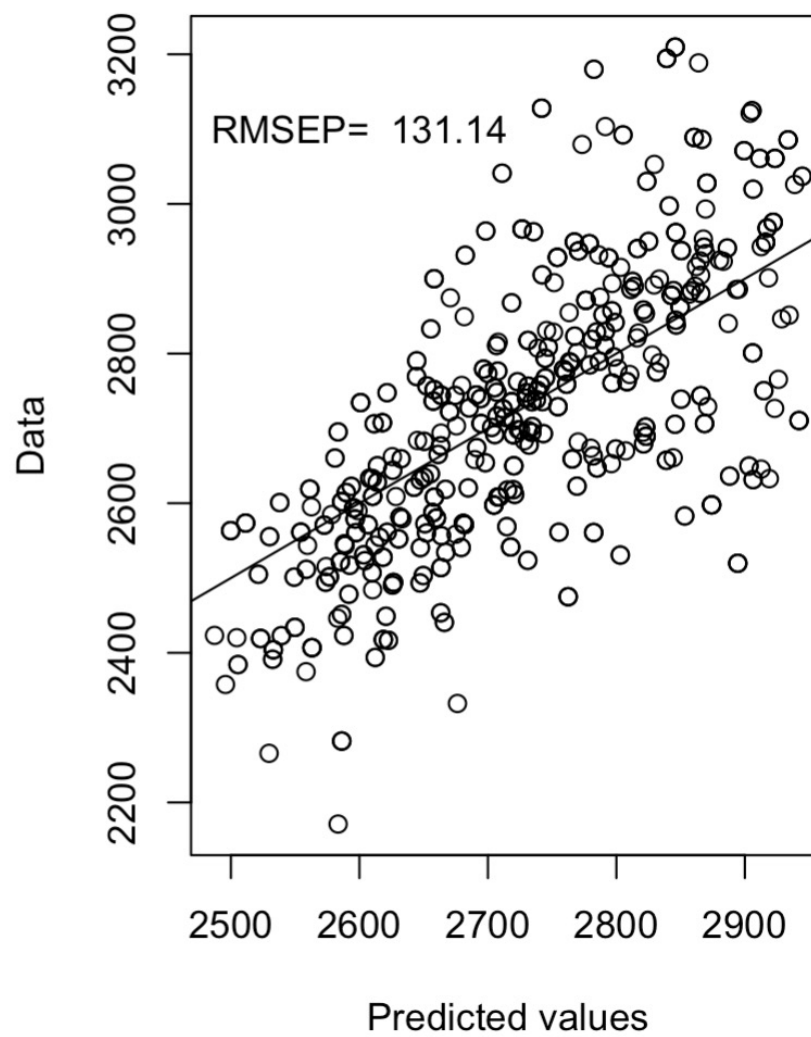
```
> RMSEP_rf
[1] 131.1415
```

```
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.**



**B.**



# 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