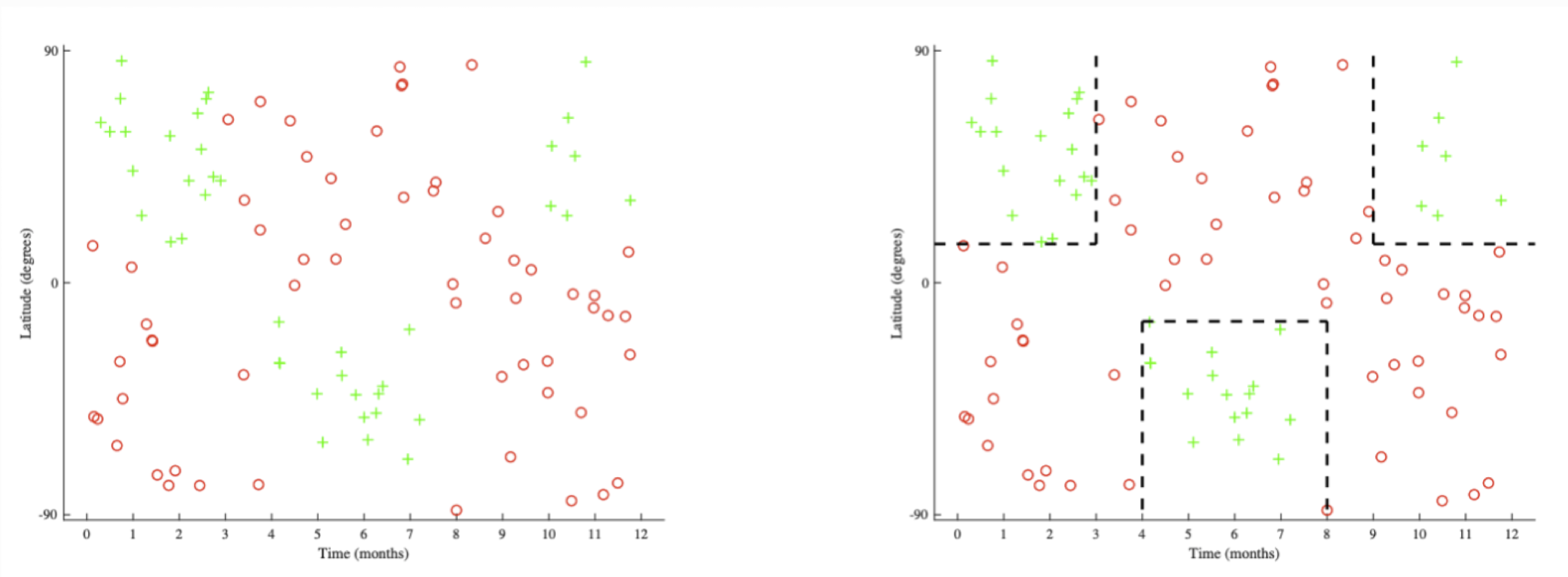
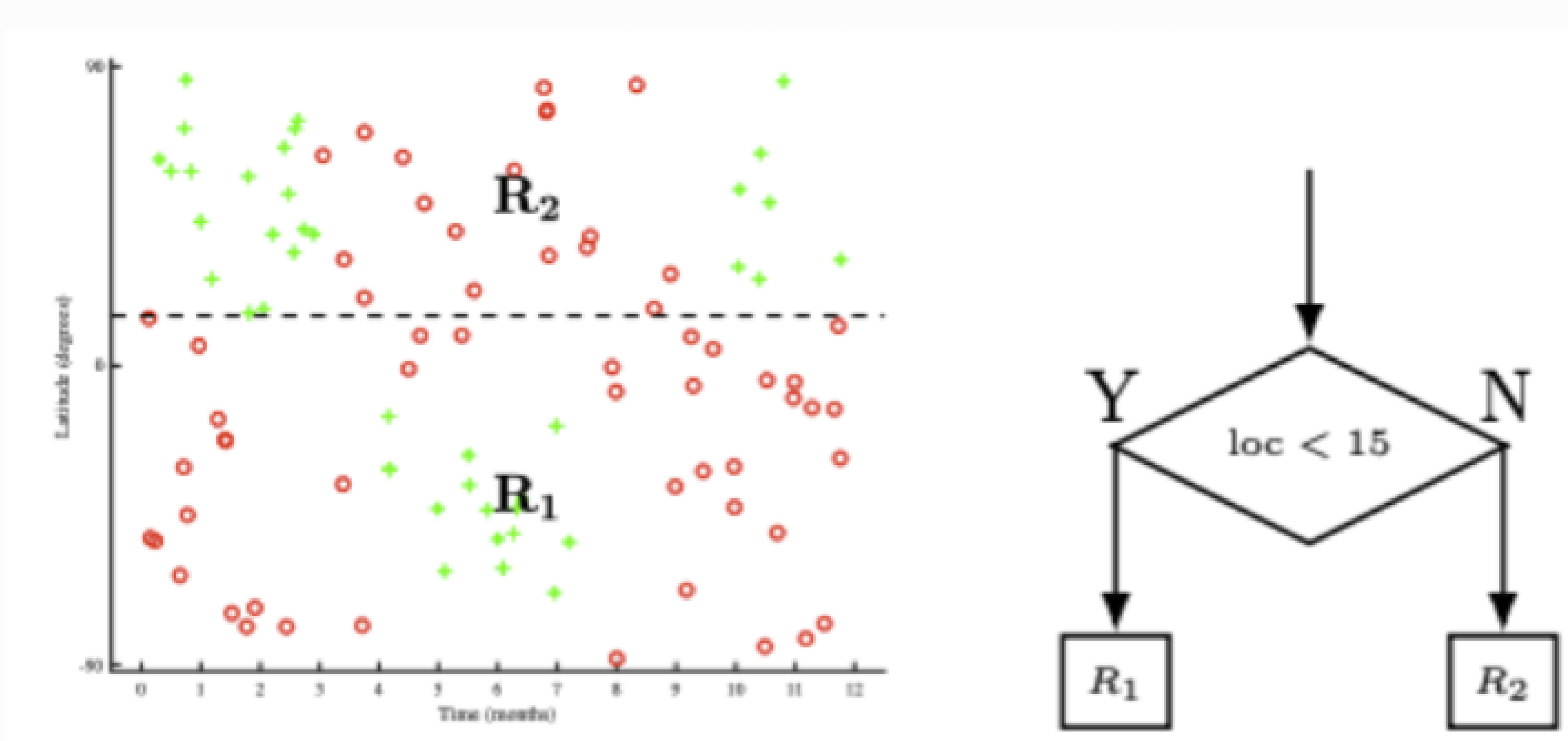
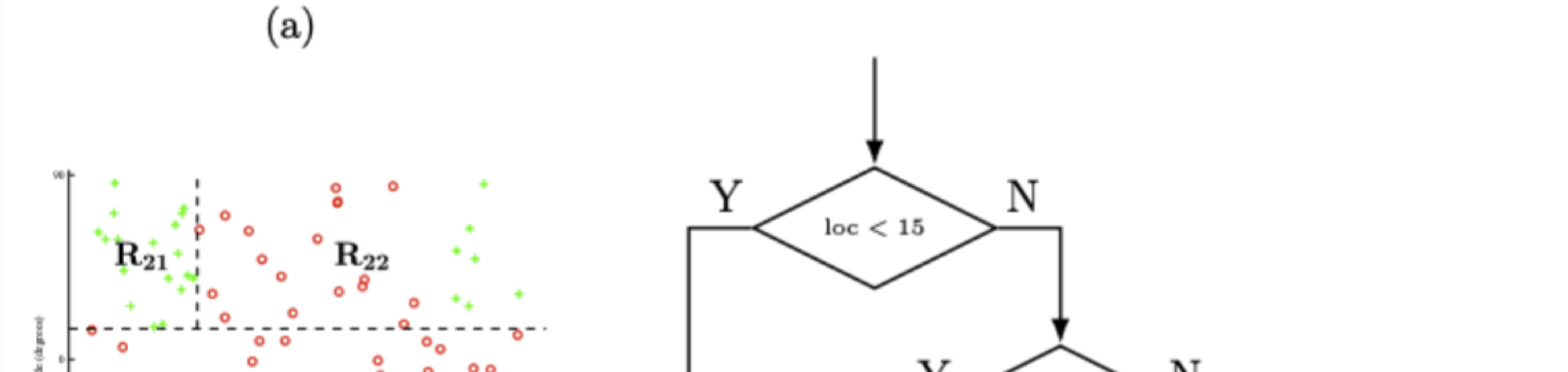
Decision tree is a non-linear non-parametric supervised machine learning algorithm. It uses greedy, top-down, recursive partitioning method. Decision tree is easy to understand & very intuitive. Let's see details of decision tree with an example dataset. We want to make a classifer which will predict in any given time & location can we ski there.

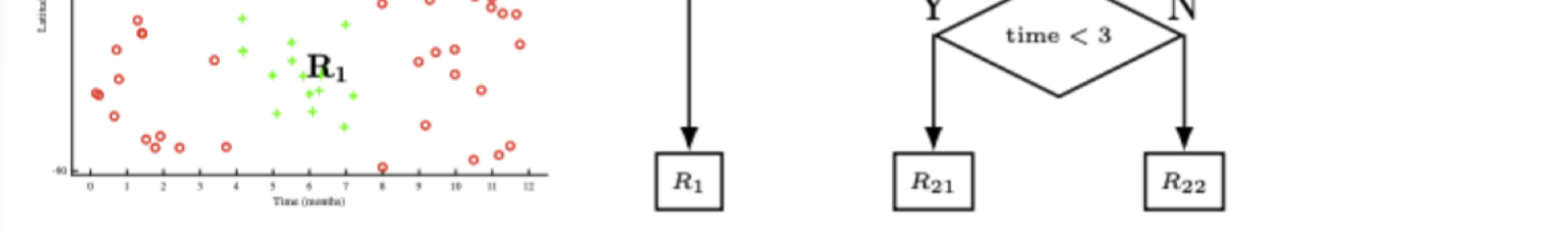


We will pick the region with possible sking. And we will do that by a greedy recursive approach. This can be done by asking question.

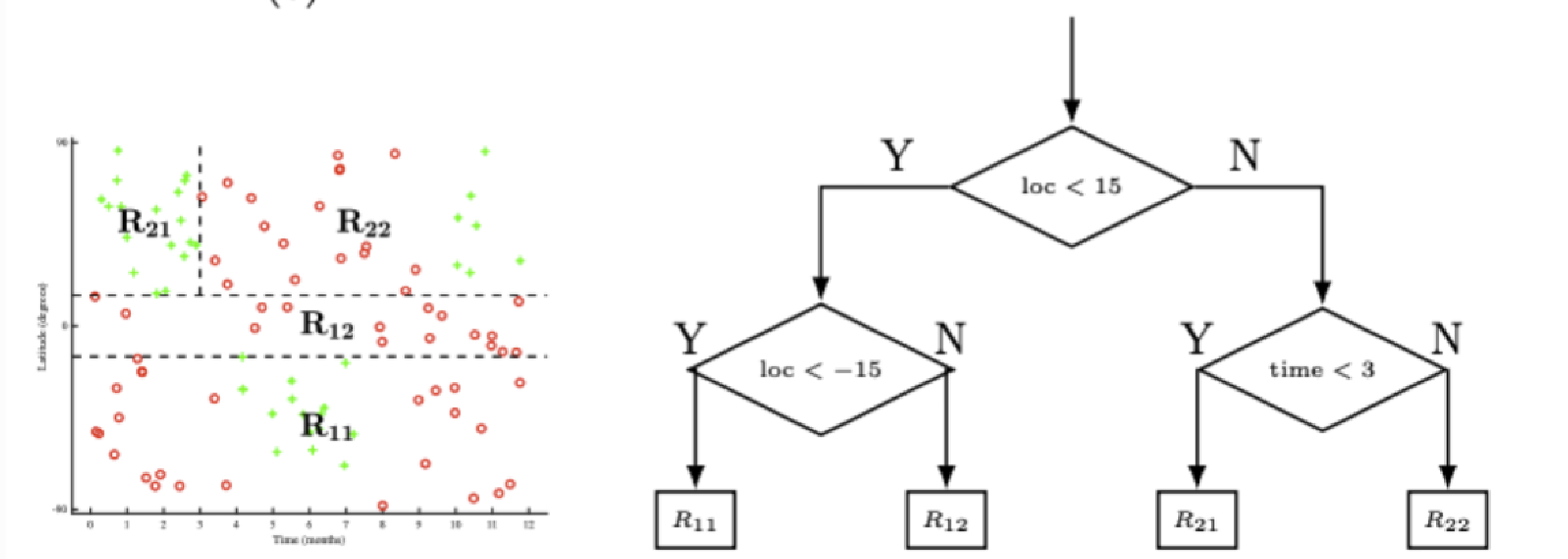


(a)





(b)



(c)

Formally, given a parent region R_p, a feature index j, and a threshold $t \in \mathbb{R}$, we obtain two child region R_1 & R_2 :

$$R_1 = \{X \mid X_j < t;\ X \in R_p\}$$
$$R_2 = \{X \mid X_j >= t;\ X \in R_p\}$$

$t = 15$
$j =$ location
$R_p =$ whole dataset

Which feature value & threshold needs to be selected?

→ Depends on for which values loss decreases most.

Loss function :-

$$\text{Loss of parent region } R_p = \frac{|R_1| L(R_1) + |R_2| L(R_2)}{|R_1| + |R_2|}$$

So we want to select feature & threshold that will maximize our decrease in loss:

$$\text{maximize} \quad |R_1| L(R_1) + |R_2| L(R_2)$$

maximize $L(R_p) - \frac{}{|R_1| + |R_2|}$

There are couple of loss function that can be used :

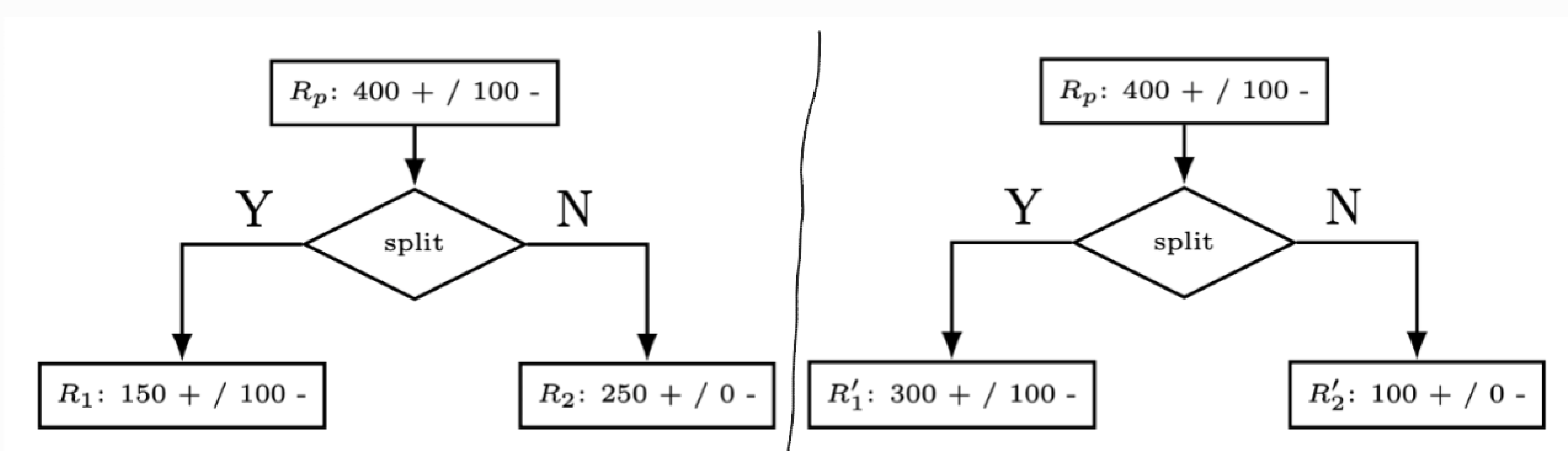
1) Classification
2) Cross entropy
3) Gini

1) Classification :-

$$L_{misclass}(R) = 1 - \max_c (\hat{P}_c)$$

Here, $\hat{P}_c$ = Proportion of example of class c in region R.

We will spit



$\hat{P}_c\{c=+\} = \frac{150}{250} = \frac{3}{5}$.

$\hat{P}_c\{c=-\} = \frac{100}{250} = \frac{2}{5}$

So for R_1, $\hat{P}_c = \frac{3}{5}$,

$$L(R_1) = 1 - \max_c (\hat{P}_c)$$
$$= 1 - \frac{3}{5} = \frac{2}{5}$$

R_2, $\hat{P}_c = 1$

$L(R_2) = 1 - \max(\hat{P}_c)$

$\hat{P}_c\{c=+\} = \frac{300}{400} = \frac{3}{4}$

$\hat{P}_c\{c=-\} = \frac{100}{400} = \frac{1}{4}$.

$$L(R_1) = 1 - \max_c (\hat{P}_c)$$
$$= 1 - \frac{3}{4}$$
$$= \frac{1}{4}.$$

$L(R_2) = 1 - \max(\hat{P}_c)$

$= 1 - 1 = 0$

$$L_{misclass} = \frac{|R_1| L(R_1) + |R_2| L(R_2)}{|R_1| + |R_2|}$$

$$= \frac{250 \times \frac{2}{5} + 250 \times 0}{250 + 250}$$

$$= \frac{250 \times \frac{2}{5}}{500}$$

$$= 0.2$$

$= 1 - 1 = 0$

$$L_{misclass} = \frac{|R_1| L(R_1) + |R_2| L(R_2)}{|R_1| + |R_2|}$$

$$= \frac{400 \times \frac{1}{4} + 100 \times 0}{400 + 100}$$

$$= \frac{100}{500}$$

$$= 0.2$$

In the first split the R_2 region separate more negetive class than the first split. still our loss stays the same. So Classification Loss isn't sensivitive enough to split.

2. Cross entropy Loss:

To make loss more sensitive cross entropy loss is introduce.

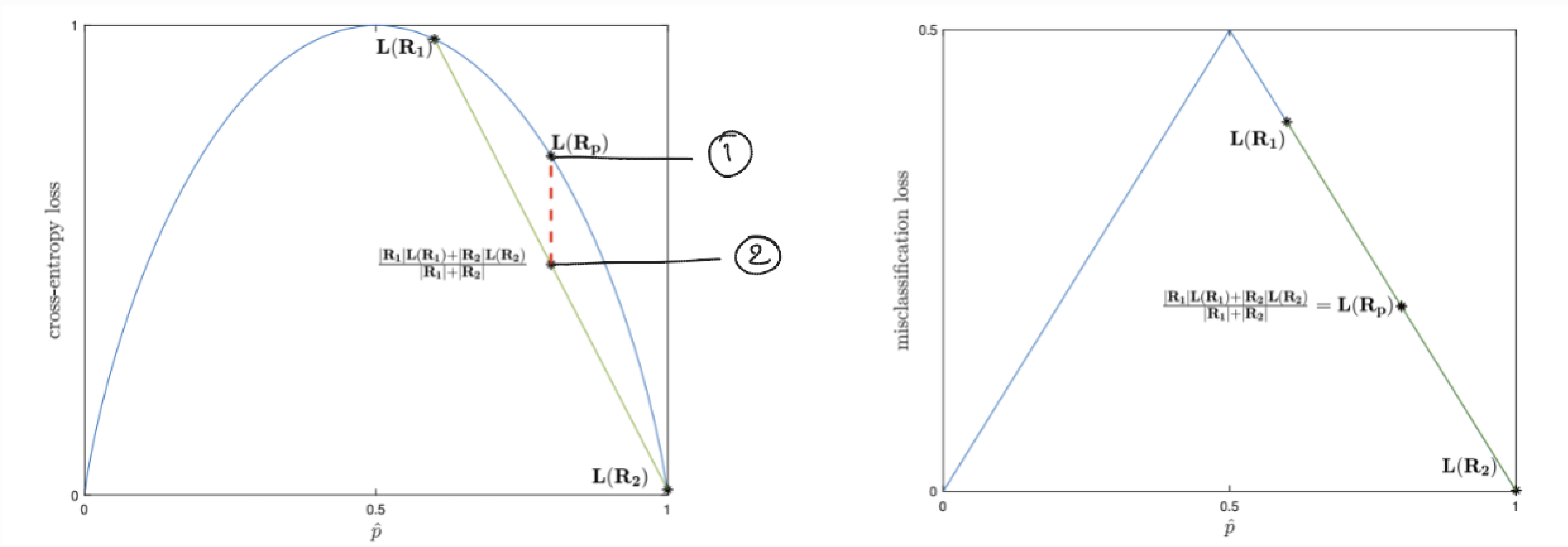
$$L_{cross}(R) = -\sum_c \hat{P}_c \log_2 \hat{P}_c$$

To understand the difference between the two loss let's see the picture bellow:

Cross-entropy loss gives a concave line. so there will always be a reduction in parent loss. point 1 shows the parent loss before the split and point

2 shows the loss after the split.

$$1 - 2 = \text{decrease in loss.}$$



in the misclassification class point 1 & 2 merge together so there is no decrease in loss.

3. Gini :

Gini is the most common measure of impurity.

$$L_{Gini} = \sum_c \hat{P}_c (1 - \hat{P}_c)$$

4. Mean Squared Loss :-

For the regression problem squared loss is used.

$$L_{squared} = \frac{\sum_{i \in R} (y_i - \hat{y})^2}{|R|}$$

where,

$$\hat{y} = \frac{\sum_{i \in R} y_i}{|R|}$$

5. Mean absolute Loss:-

$$L_{absolute} = \frac{\sum_{i \in R} |y_i - \text{median}(y)|}{|R|}$$

Regularization:-

Decision tree is very prune to overfitting. If we don't regularize it, each of the training example can be a separate region in the worst case. The ways of regularizing DTs are:

- Minimum leaf size : Do not split R if $|R| <$ threshold.
- Maximum Depth : Don't split if $\text{Depth}_{tree} >$ threshold
- Maximum Number of leaf : Don't split if $|\text{leaf node}| >$ threshold.
- Pruning : Grow the tree fully without using any max depth. Then remove some leaf node during validation set to reduce validation error.

Runtime:-

the training runtime of the tree is

$$O(nfd)$$

$n \Rightarrow$ # traing example.

$f \Rightarrow$ # input feature.

$d \Rightarrow$ depth of the tree

During testing runtime is

$O(d)$

if the tree is balanced :-

$O(\log n)$