Introduction to Machine Learning. Lec.6 Decision Trees

Aidos Sarsembayev, IITU, 2018

CART

- CART – is a classification and regression trees

CART

- CART – is a classification and regression trees

Classification Regression

CART

- CART – is a classification and regression trees

Classification

Regression

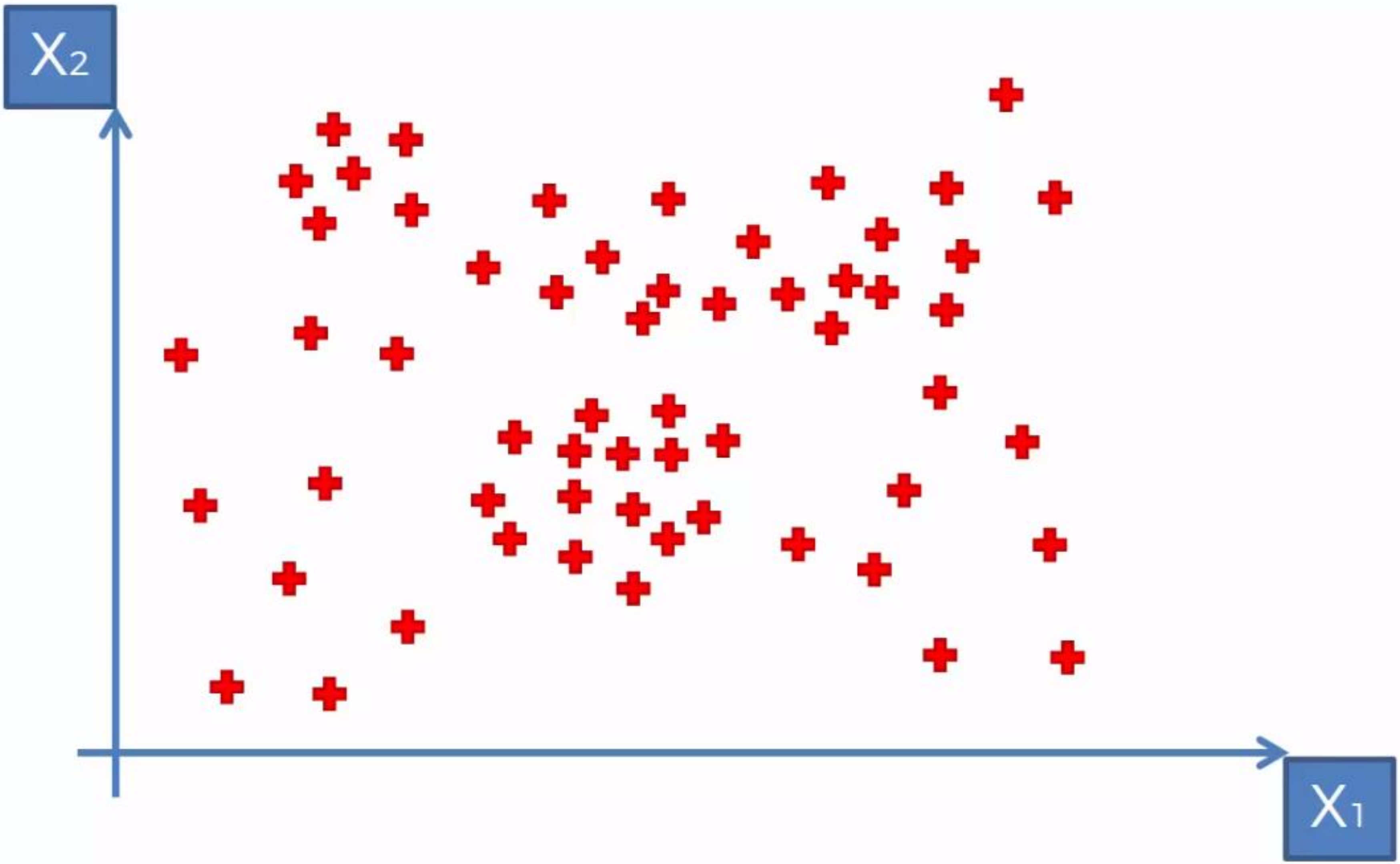
CART

- CART – is a classification and regression trees

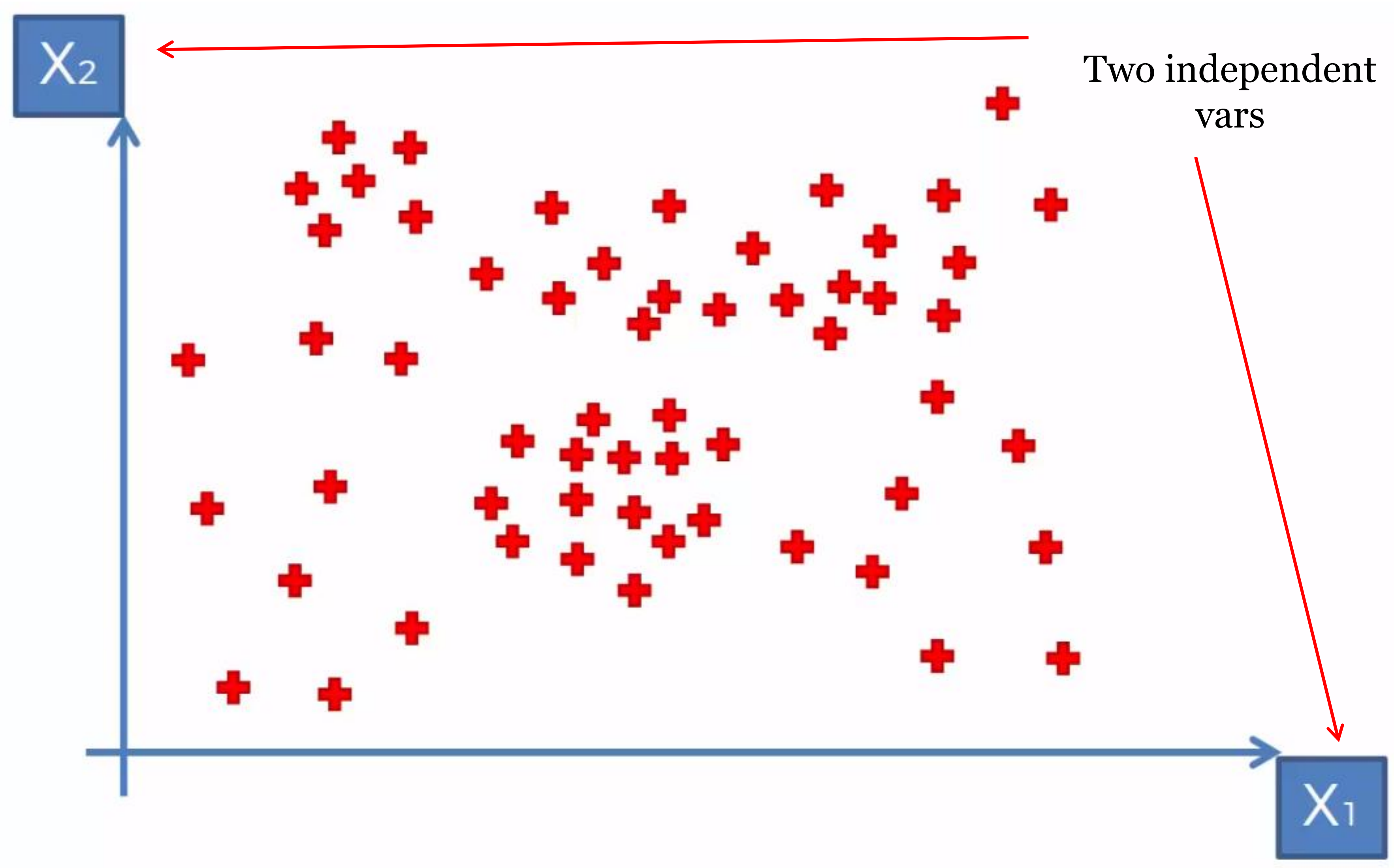
Classification

Regression

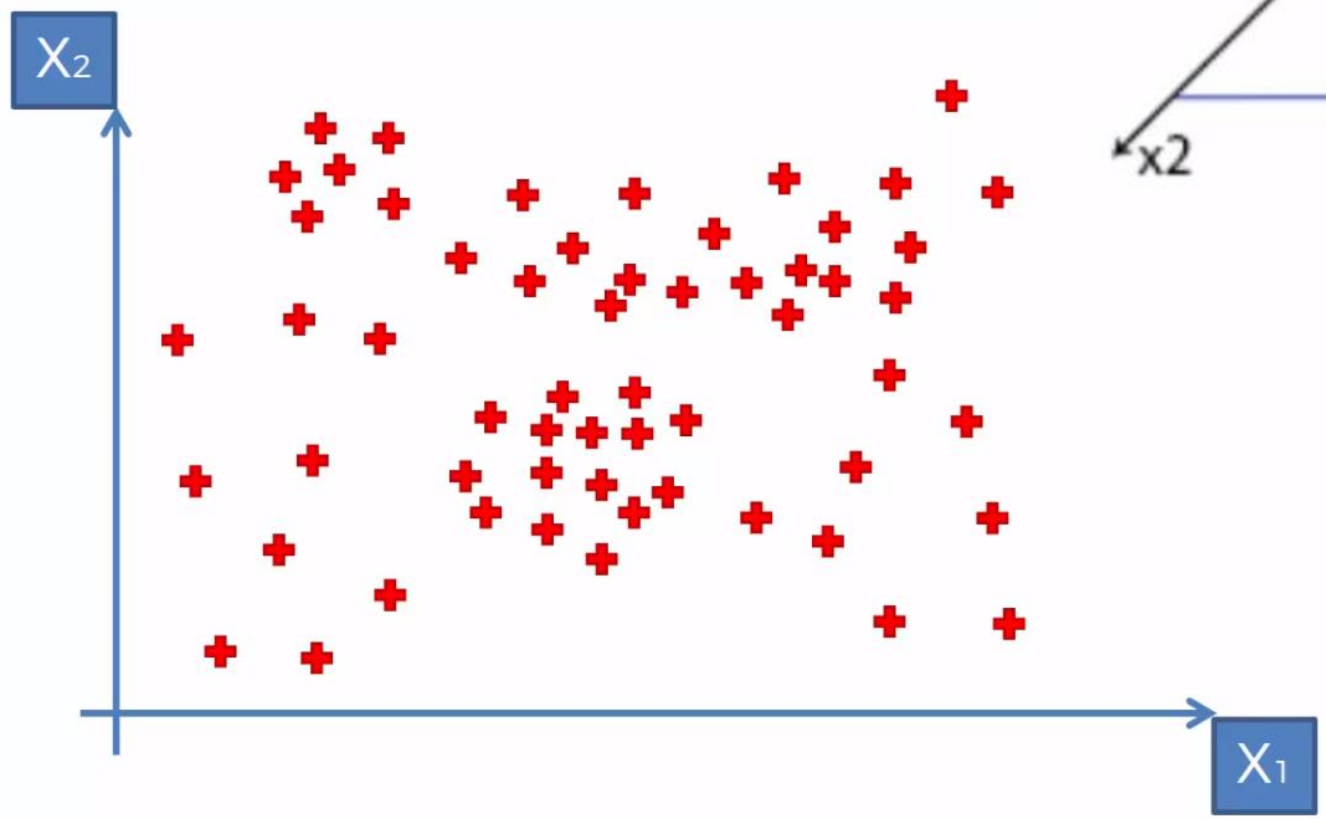
It’s a bit complex to understand

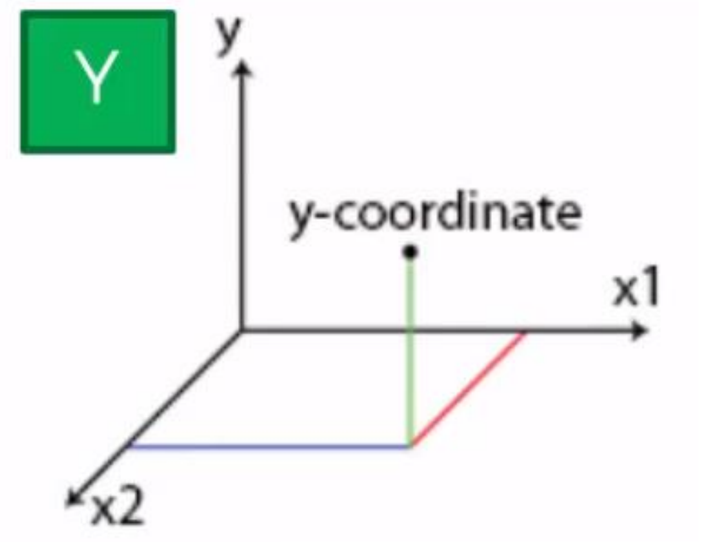


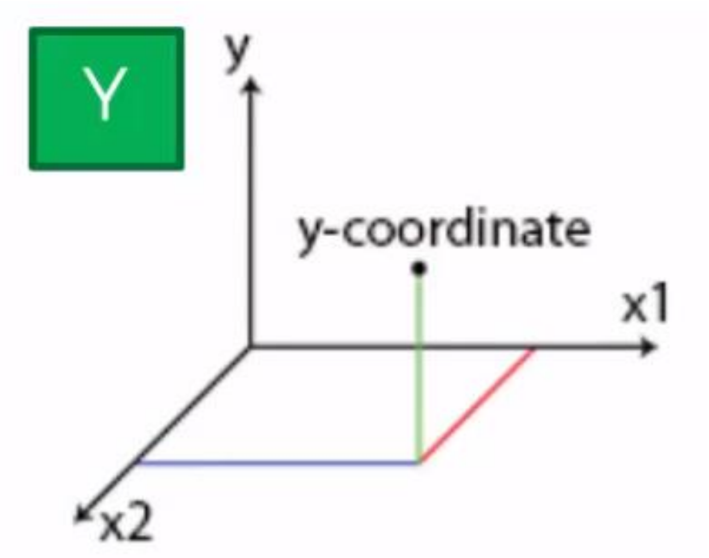
X_2
X_1

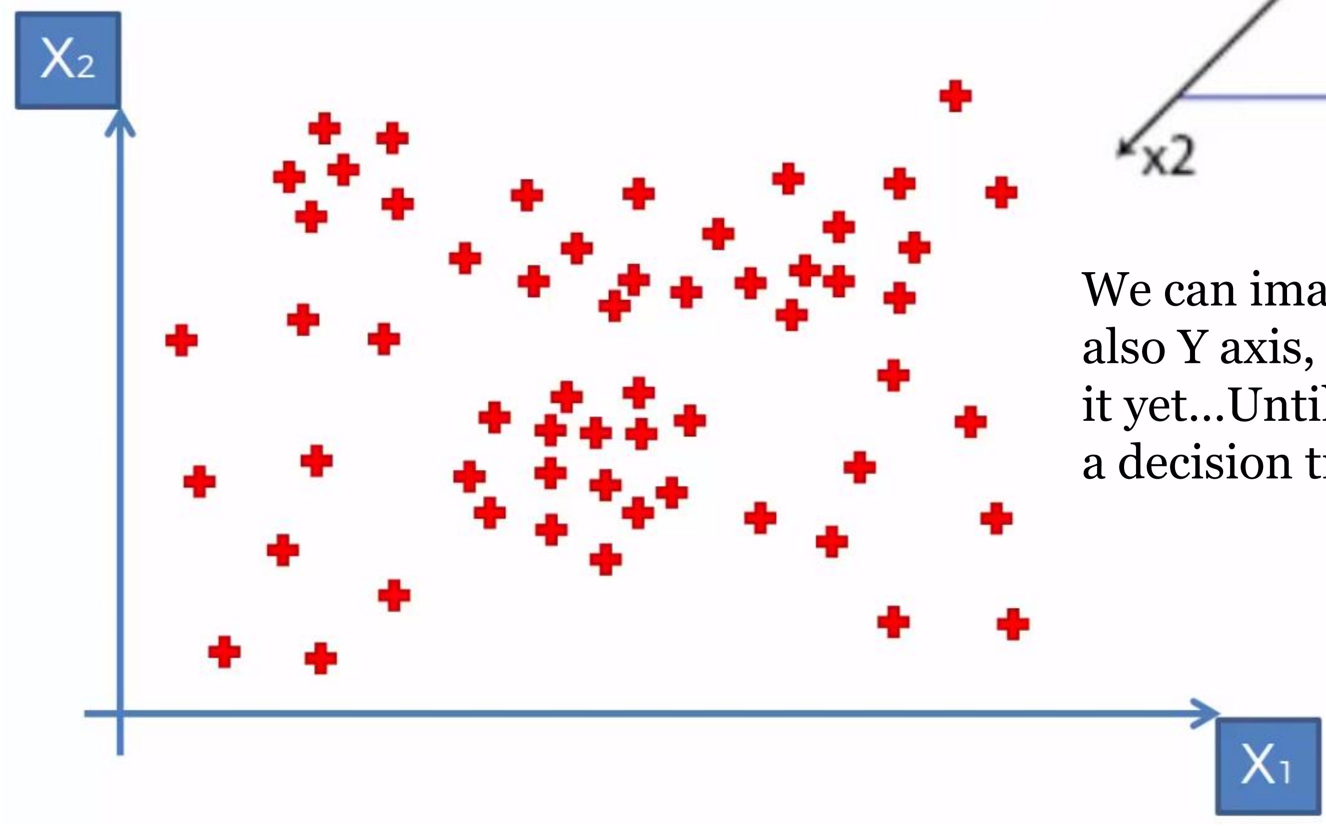


Two independent vars

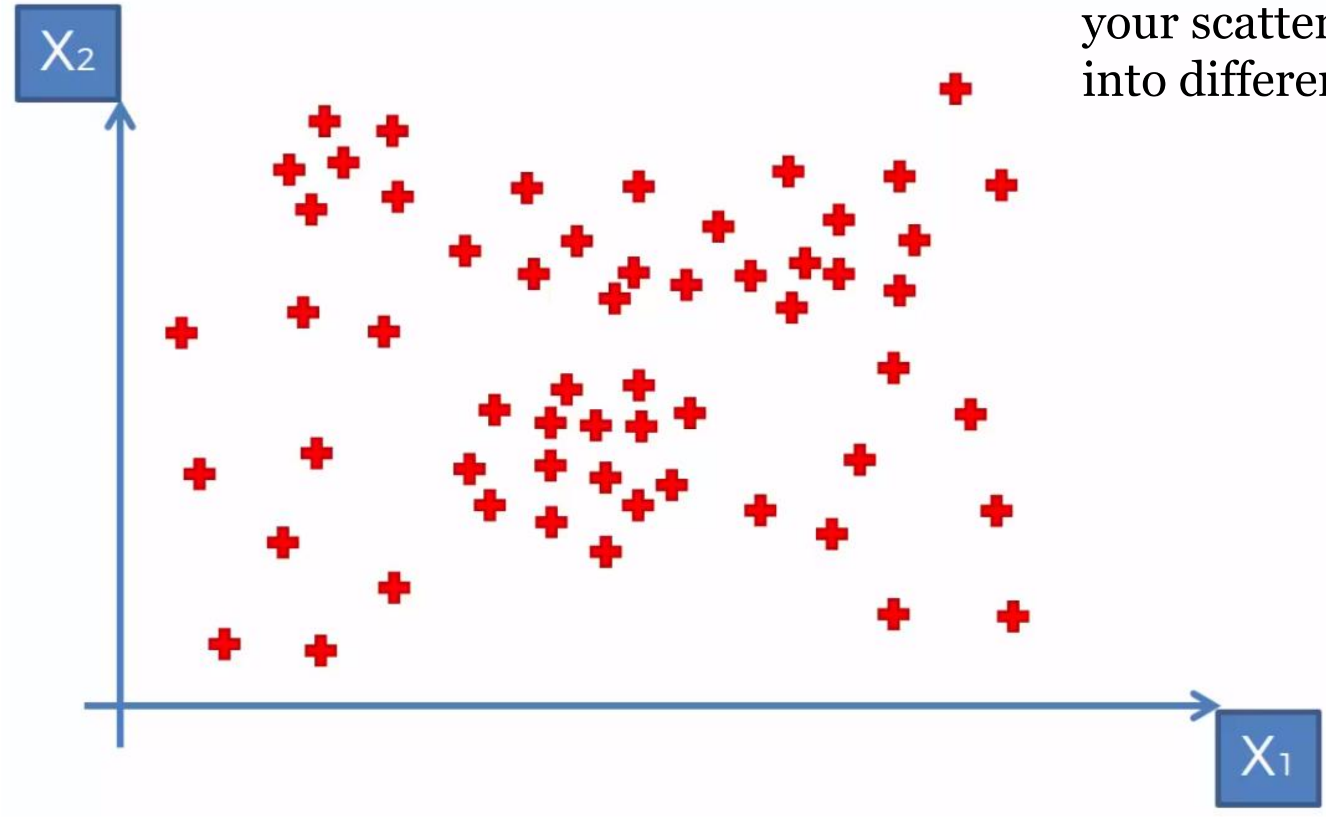

X2
X1


Y
y
y-coordinate
x1
x2

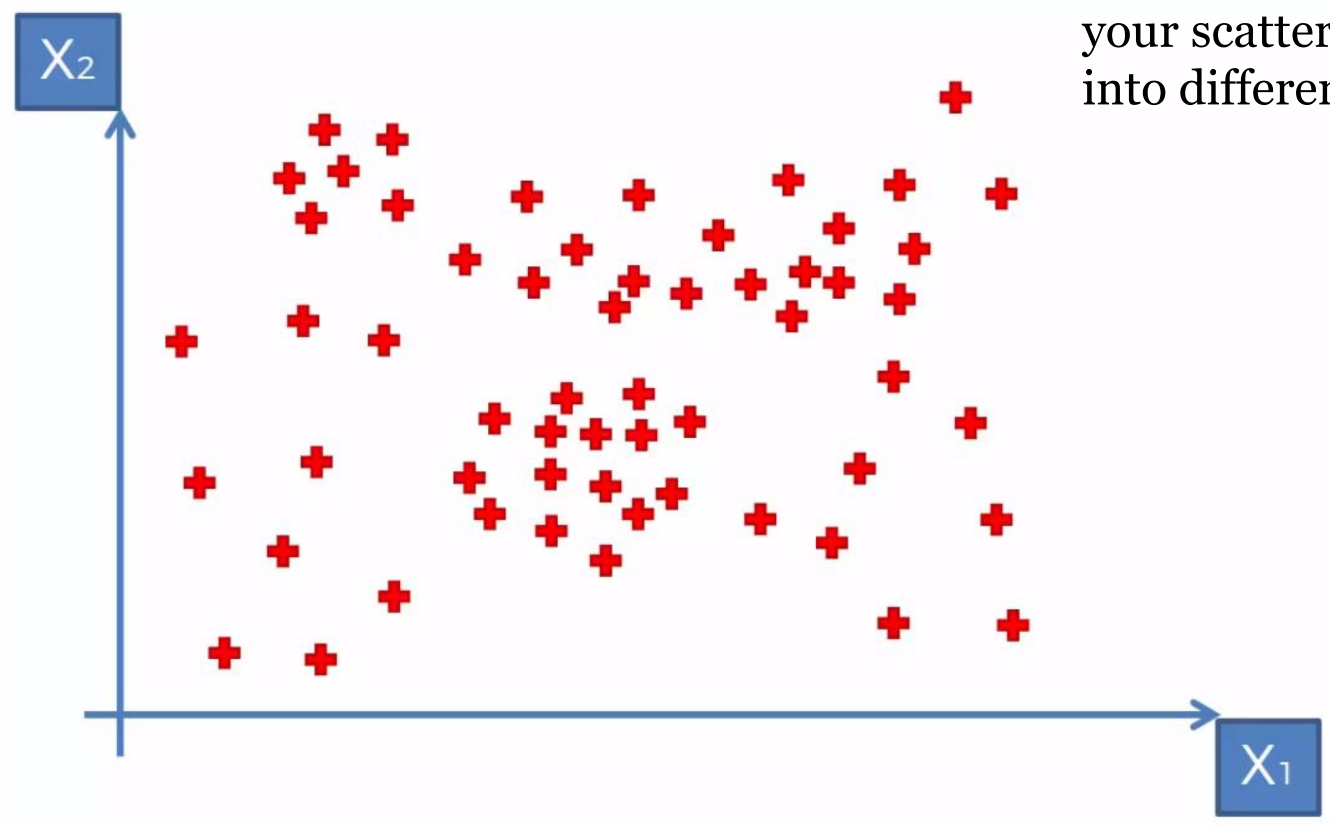




We can imagine that there is also Y axis, but we don’t need it yet…Until we build a decision tree

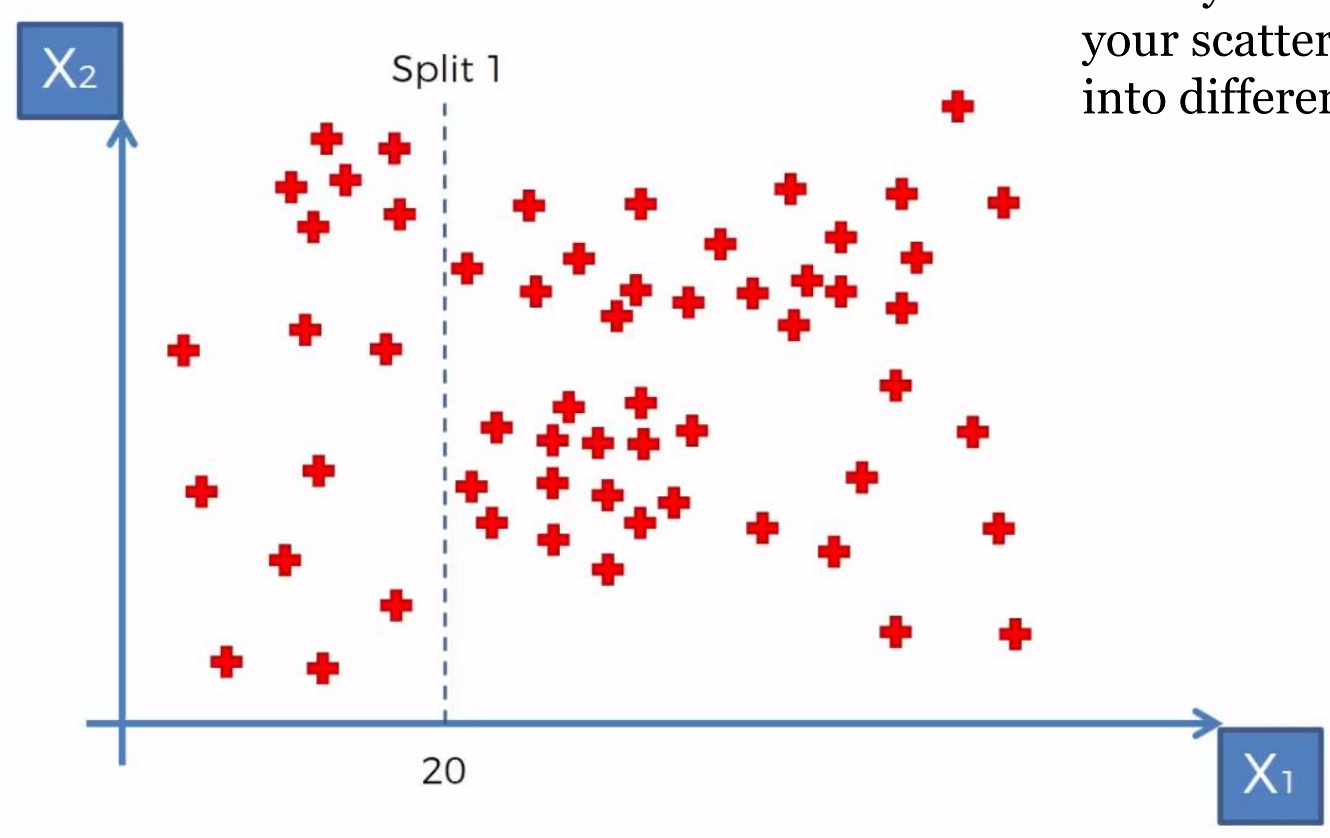


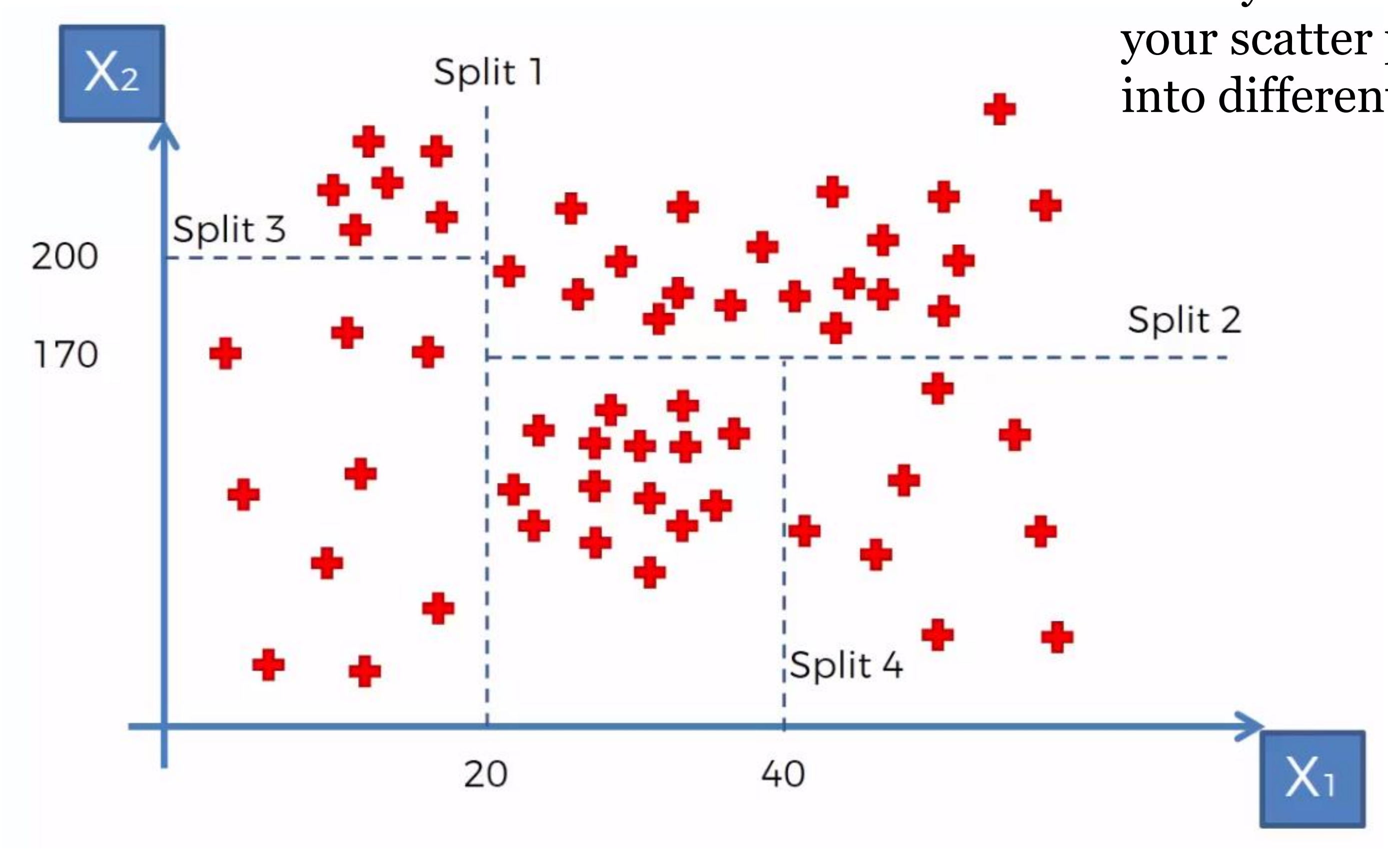
Once you run your DT algorithm your scatter plot will be divided into different parts (splits)



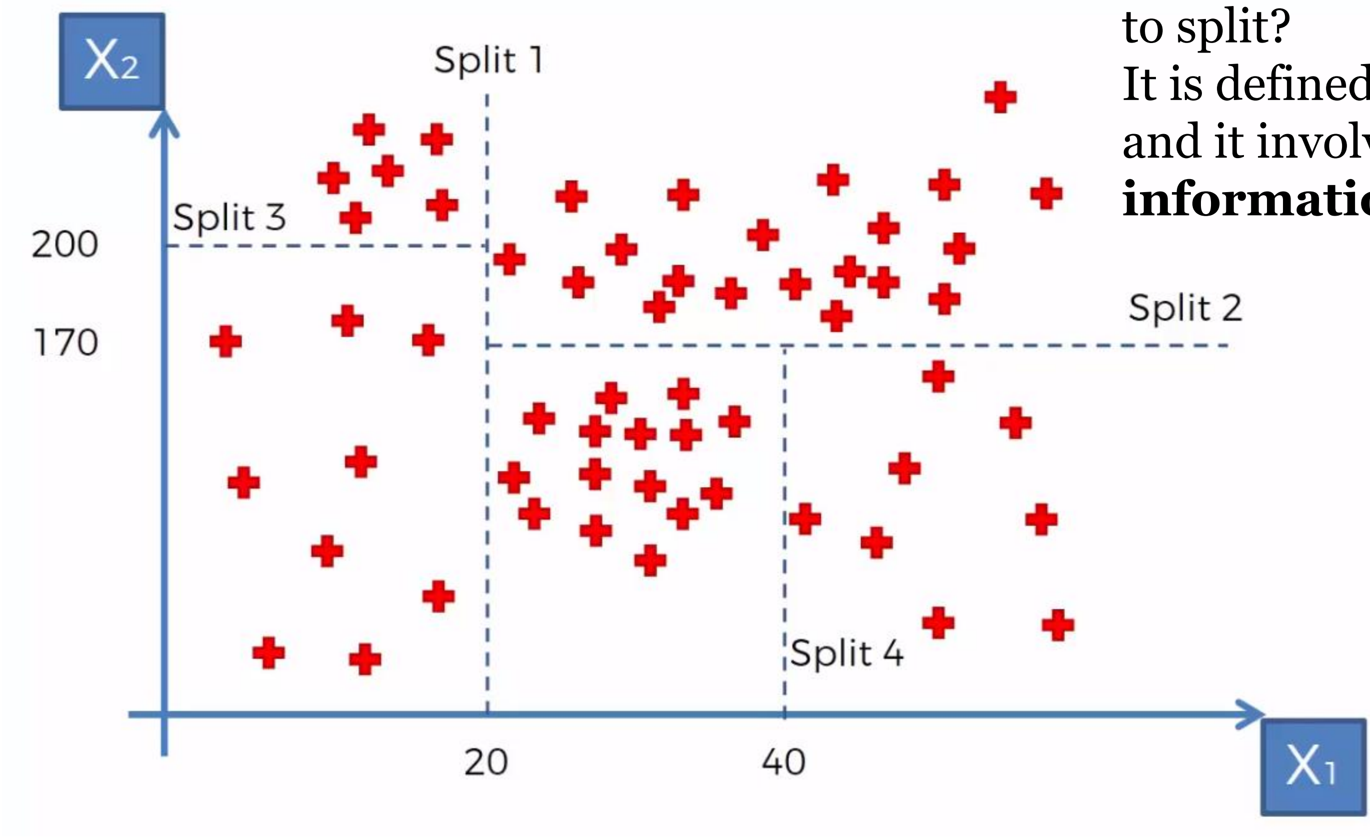
Once you run your DT algorithm your scatter plot will be divided into different parts (splits)

Once you run your DT algorithm your scatter plot will be divided into different parts (splits)





Once you run your DT algorithm your scatter plot will be divided into different parts (splits)



How do we choose how or where to split?
It is defined by the algorithm and it involves – **information entropy**

Information entropy

- is the average rate at which information is produced by a stochastic (random) source of data.

Information entropy

- is the average rate at which information is produced by a stochastic (random) source of data.
- Generally, *entropy* refers to disorder or uncertainty
- The measure of information entropy associated with each possible data value is the negative logarithm of the probability mass function for the value.

Information entropy

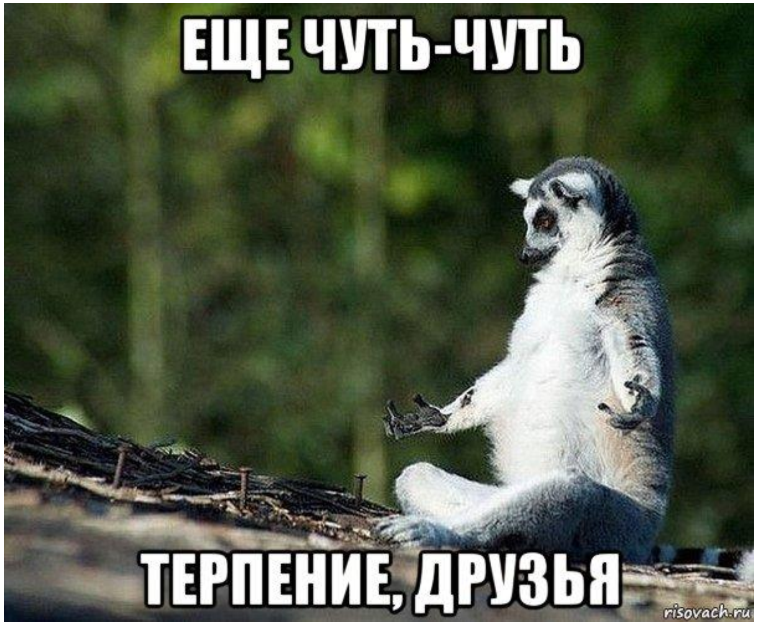
- is the average rate at which information is produced by a stochastic (random) source of data.
- The measure of information entropy associated with each possible data value is the negative logarithm of the probability mass function for the value.

$$H = -\sum_{i=1}^{n} p(x_i)\, log_n p(x_i)$$

Information entropy

$$H = -\sum_{i=1}^{n} p(x_i)\, log_n p(x_i)$$

Information entropy



$$H = -\sum_{i=1}^{n} p(x_i)\, log_n p(x_i)$$

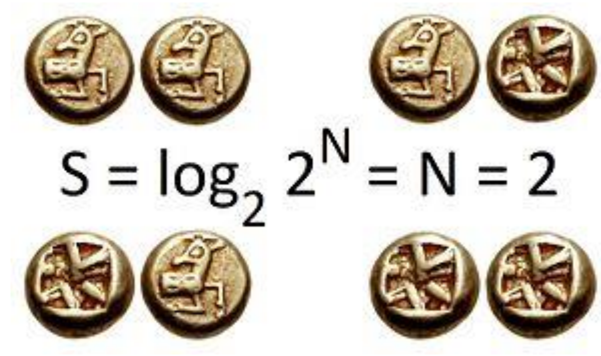
Information entropy

- When the data source has a lower-probability value (i.e., when a low-probability event occurs), the event carries more "information" ("surprisal") than when the source data has a higher-probability value.

Information entropy

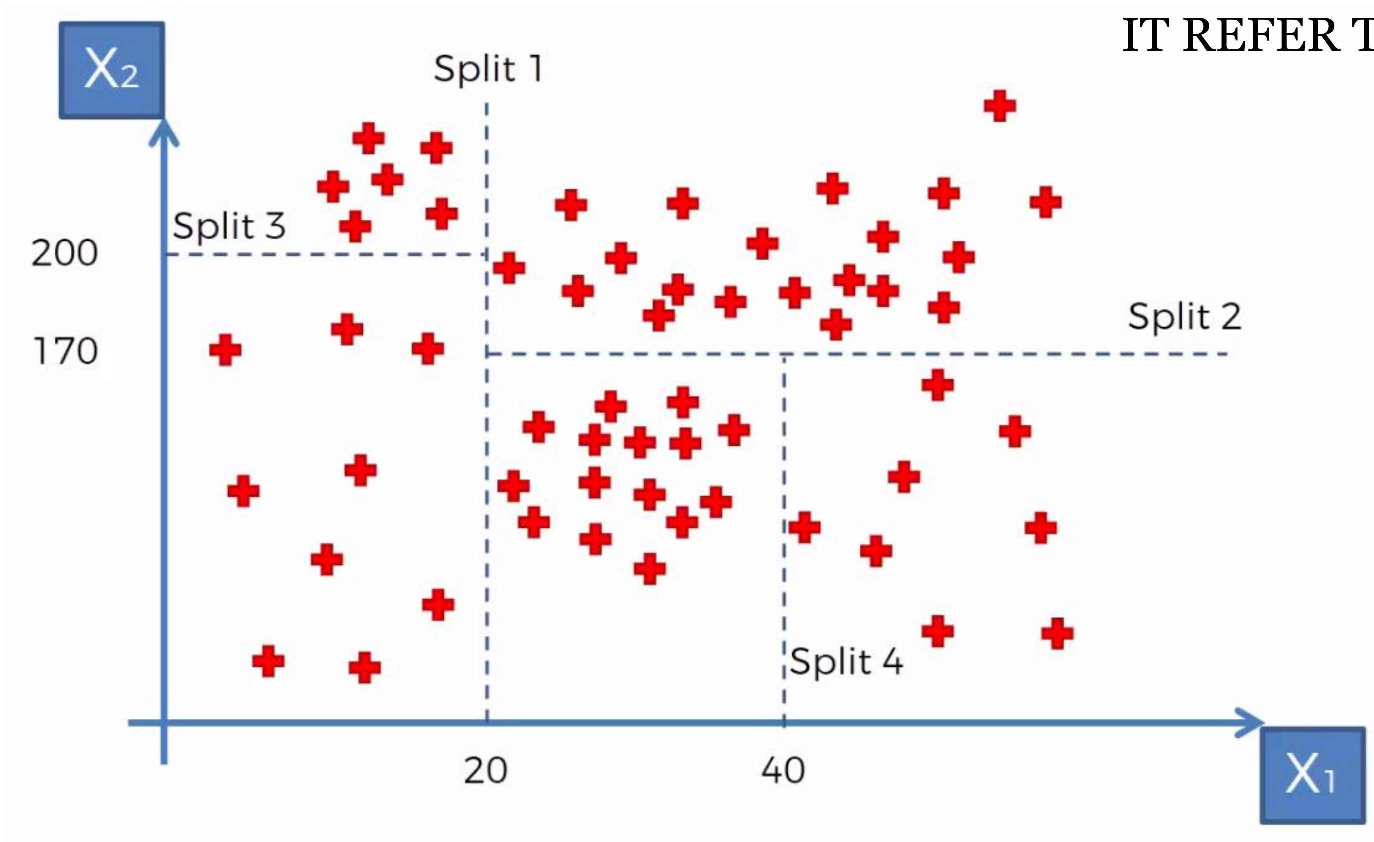
- When the data source has a lower-probability value (i.e., when a low-probability event occurs), the event carries more "information" ("surprisal") than when the source data has a higher-probability value.
- The amount of information conveyed by each event defined in this way becomes a random variable whose expected value is the **information entropy**.

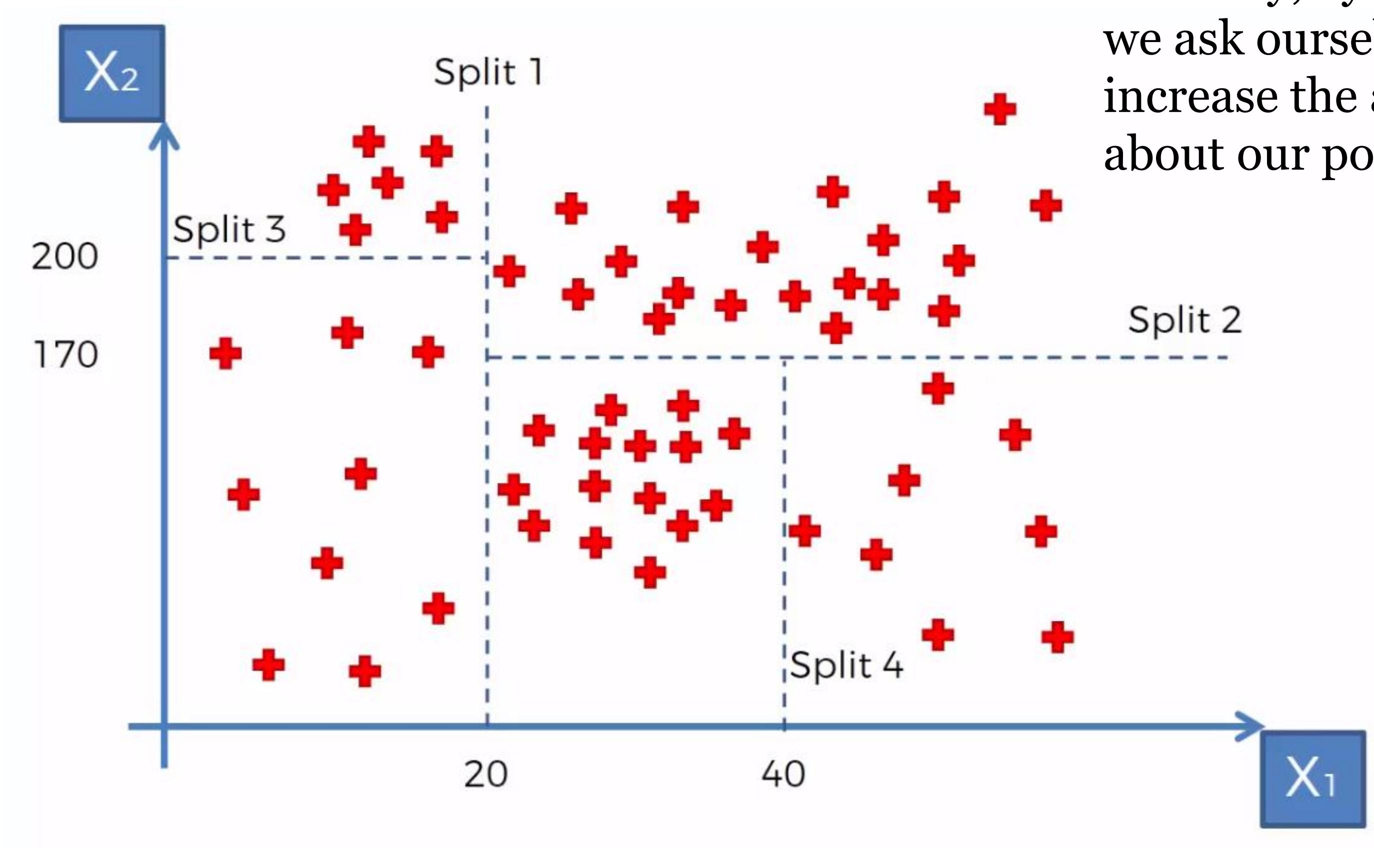
Information entropy



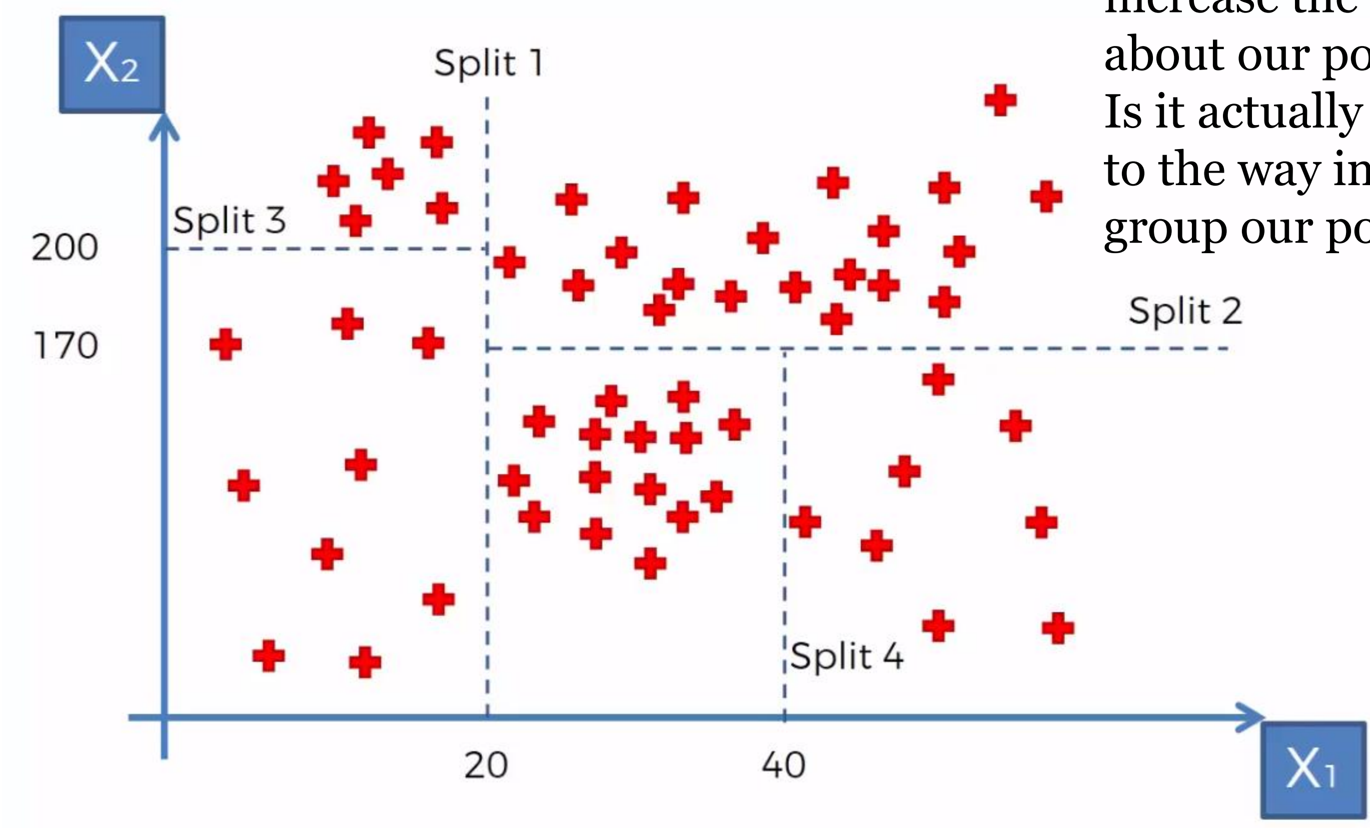
- Two bits of entropy: In the case of two fair coin tosses, the information entropy in bits is the base-2 logarithm of the number of possible outcomes; with two coins there are four possible outcomes, and two bits of entropy. Generally, information entropy is the average amount of information conveyed by an event, when considering all possible outcomes.

ALL RIGHT, BUT HOW DOES IT REFER TO THIS FIGURE??

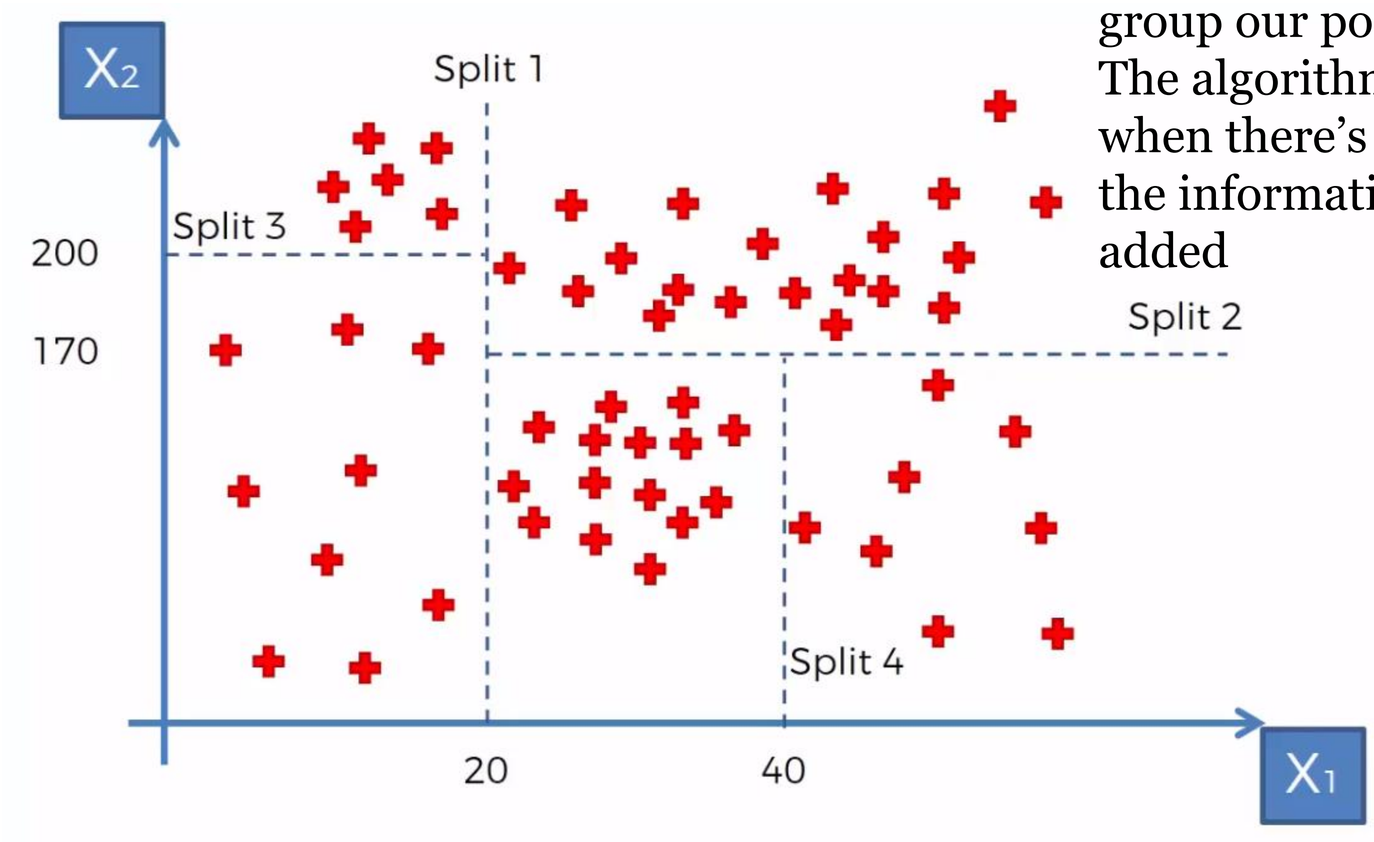




Basically, by performing a split we ask ourselves: does the split increase the amount of information about our points?

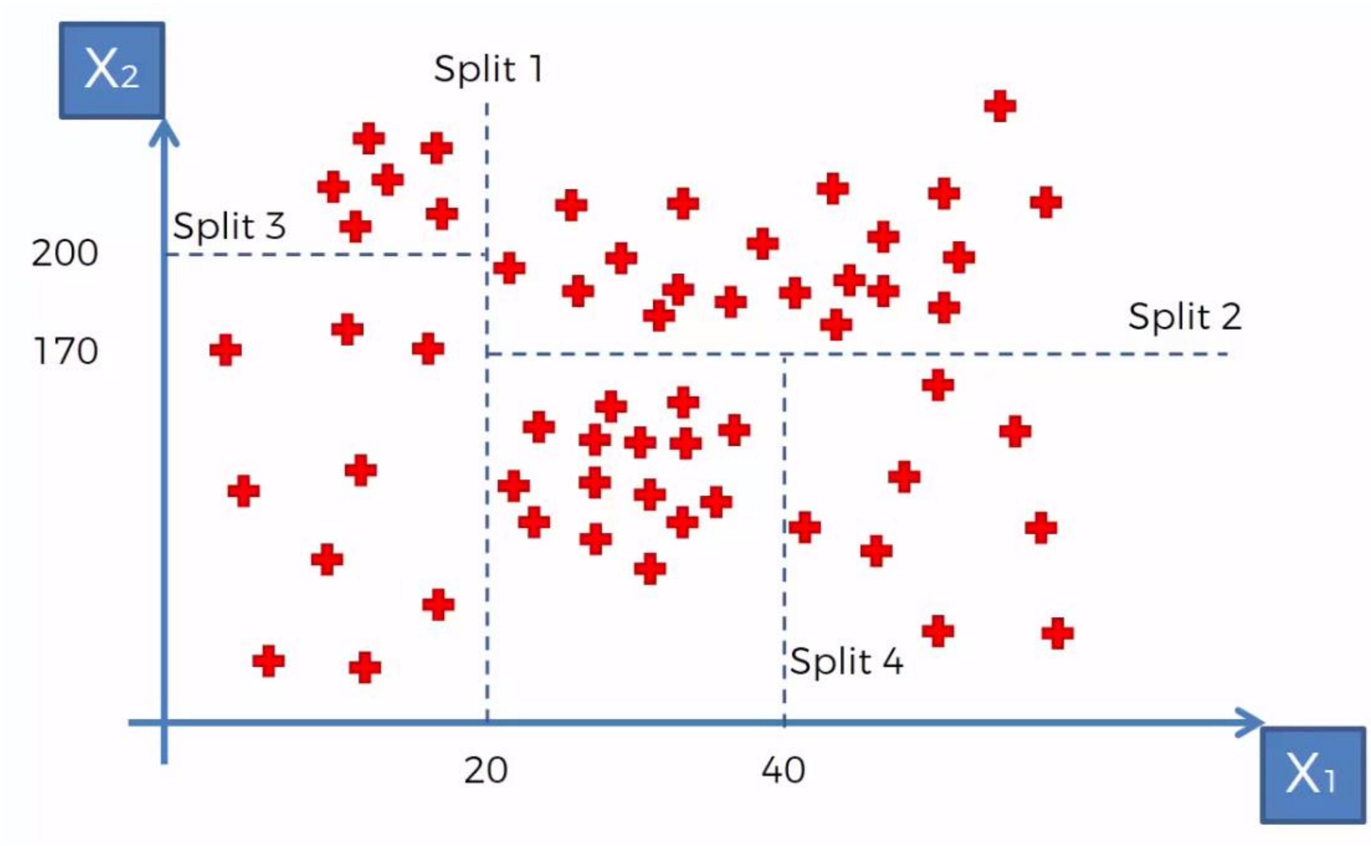


Basically, by performing a split we ask ourselves: does the split increase the amount of information about our points?
Is it actually adding some value to the way in which we want to group our points?



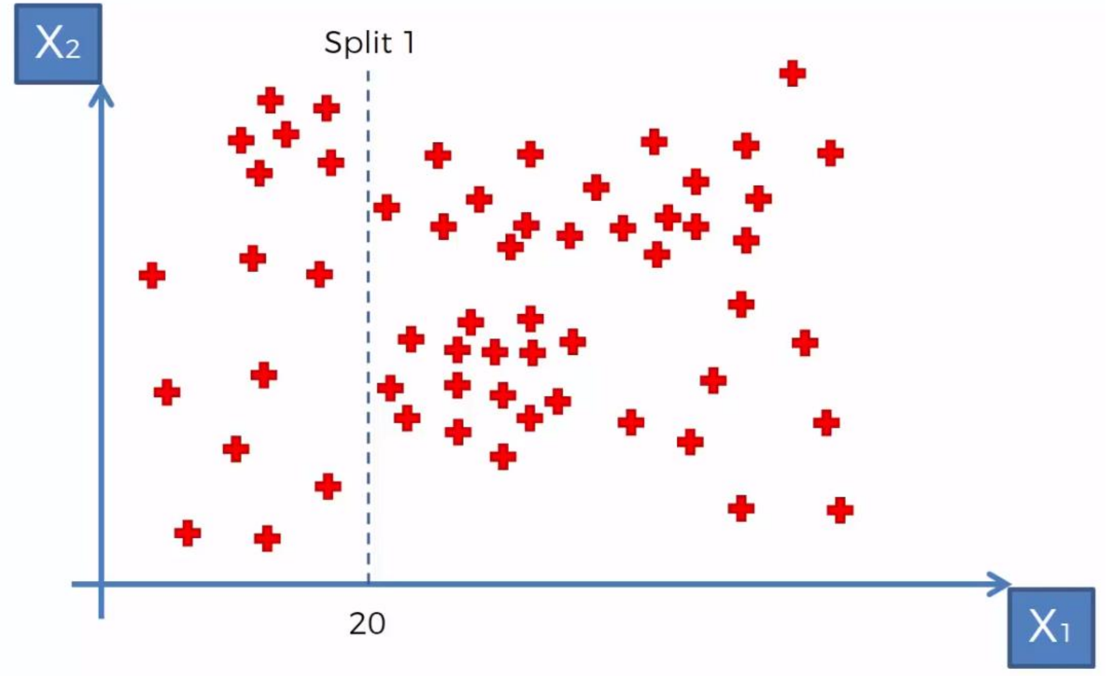
Basically, by performing a split we ask ourselves: does the split increase the amount of information about our points?
Is it actually adding some value to the way in which we want to group our points?
The algorithm knows when to stop, when there's certain minimum for the information that needs to be added

The good news:
this so much refers to the information theory, while this is the ML class.
We will not dive into the process of splitting the dataset into leaves.
The algorithm will take care of it for us



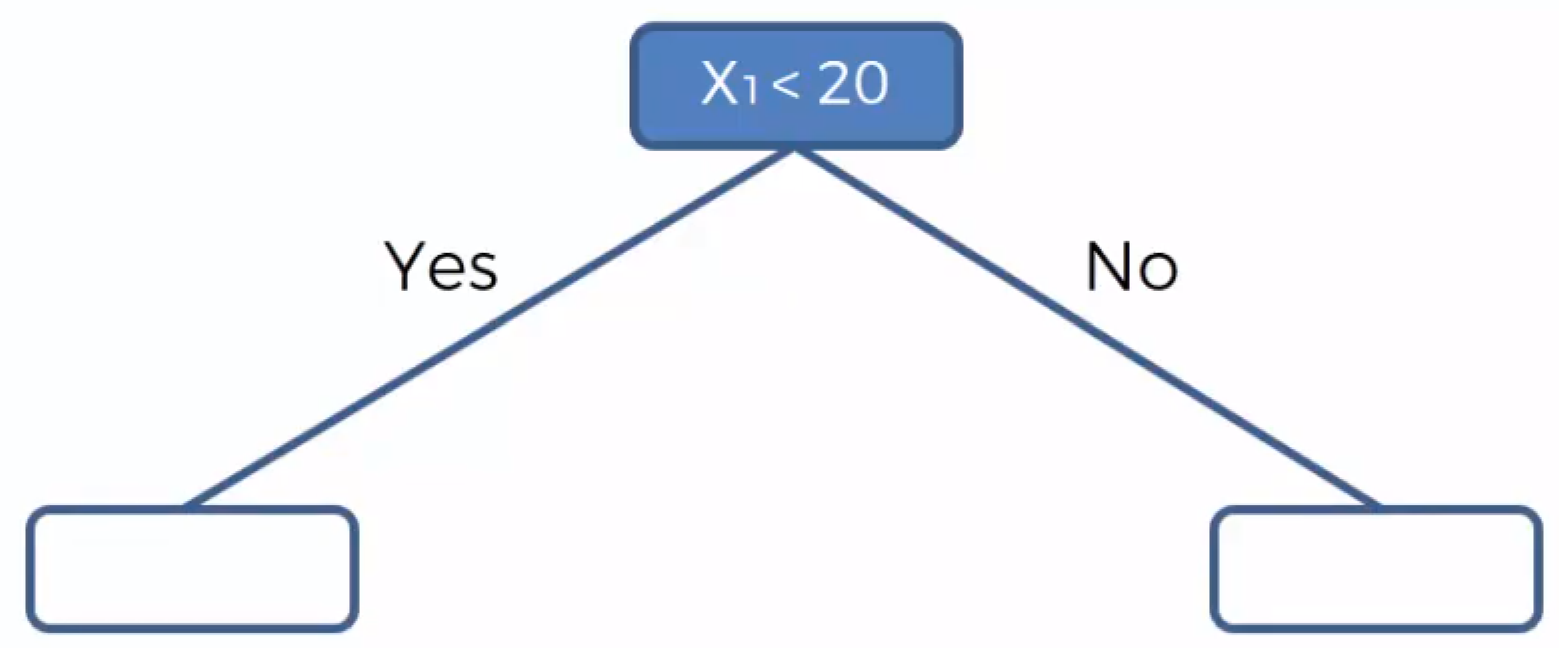
DT

- Now let's actually build the tree by doing the first split



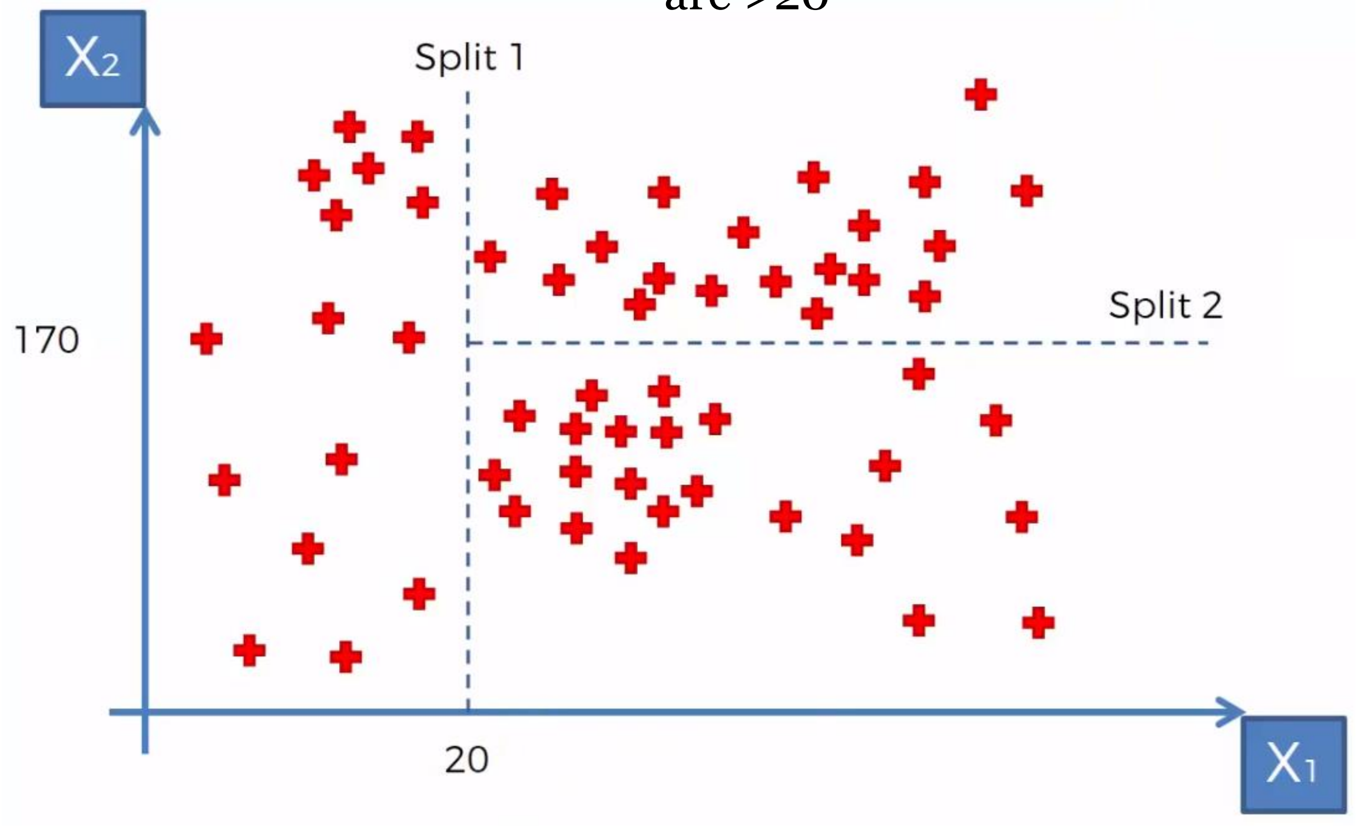
DT

DT

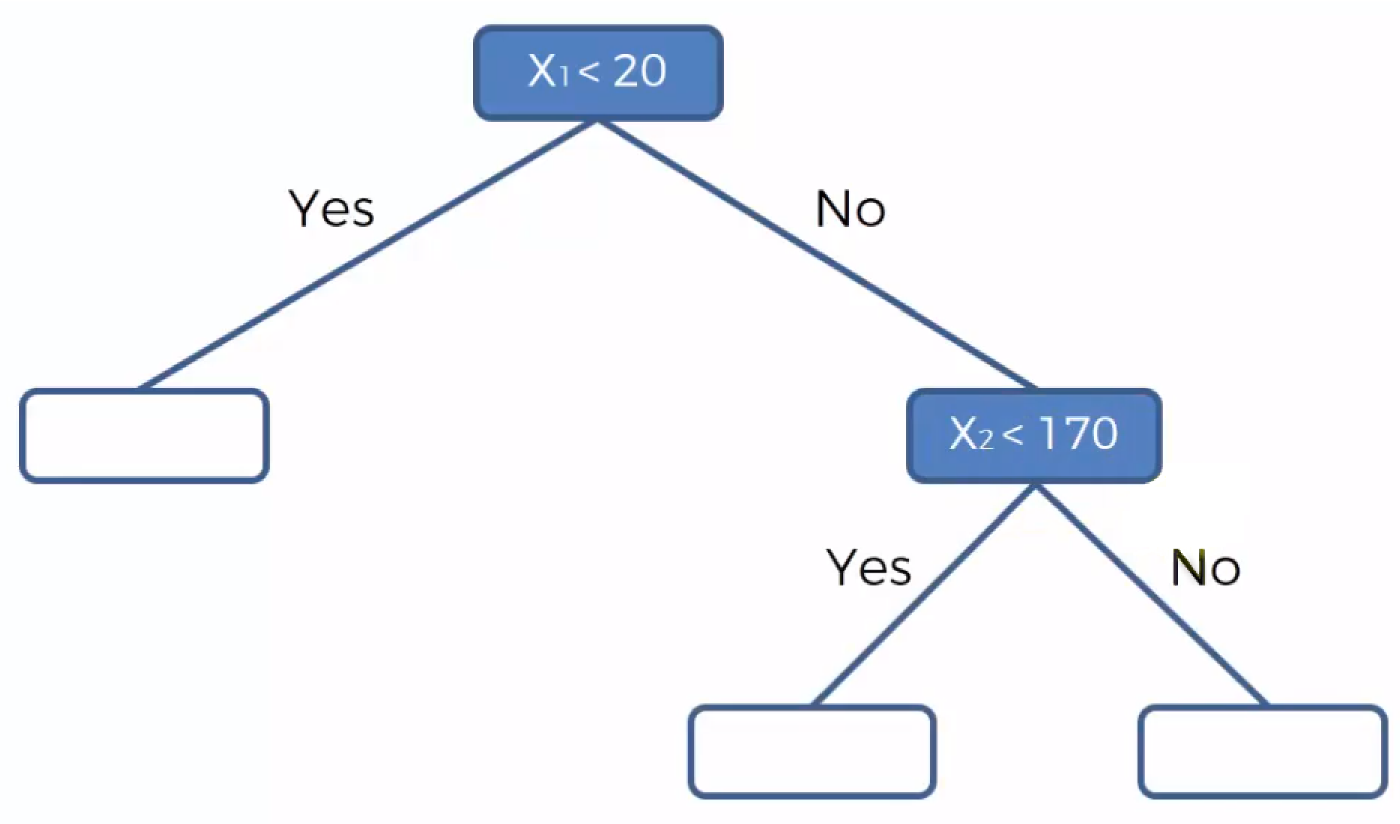


DT

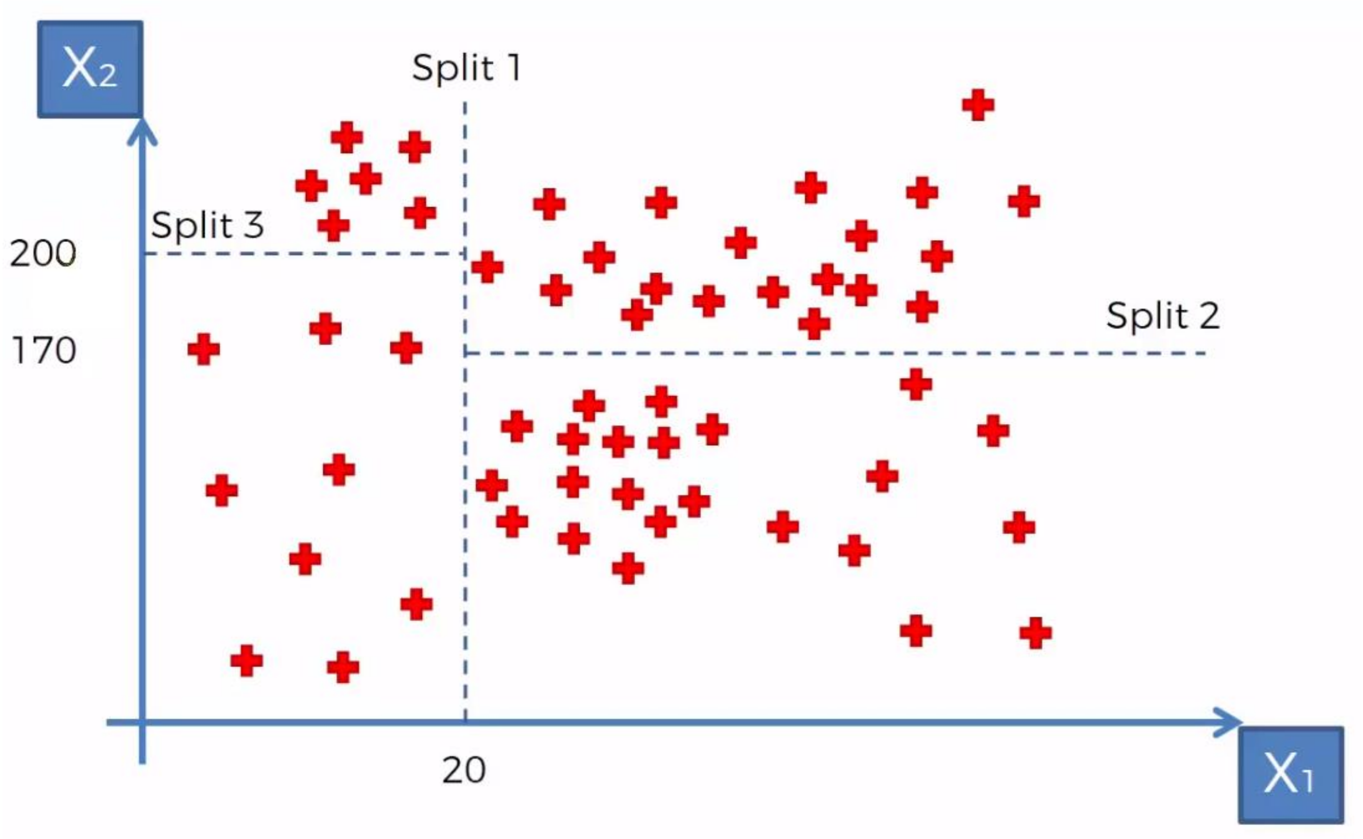
Next, happens the split at 170.
But it only happens for the values that are >20



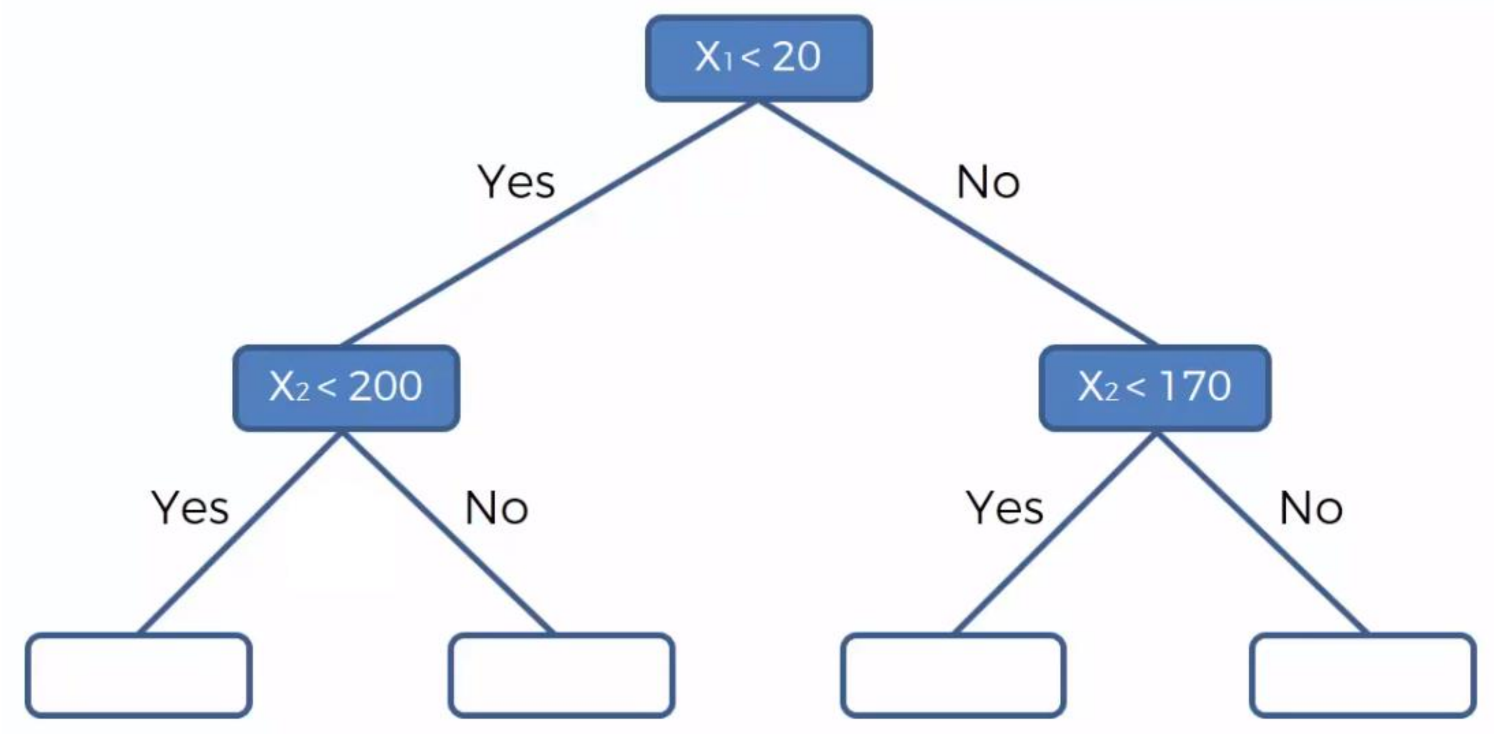
DT



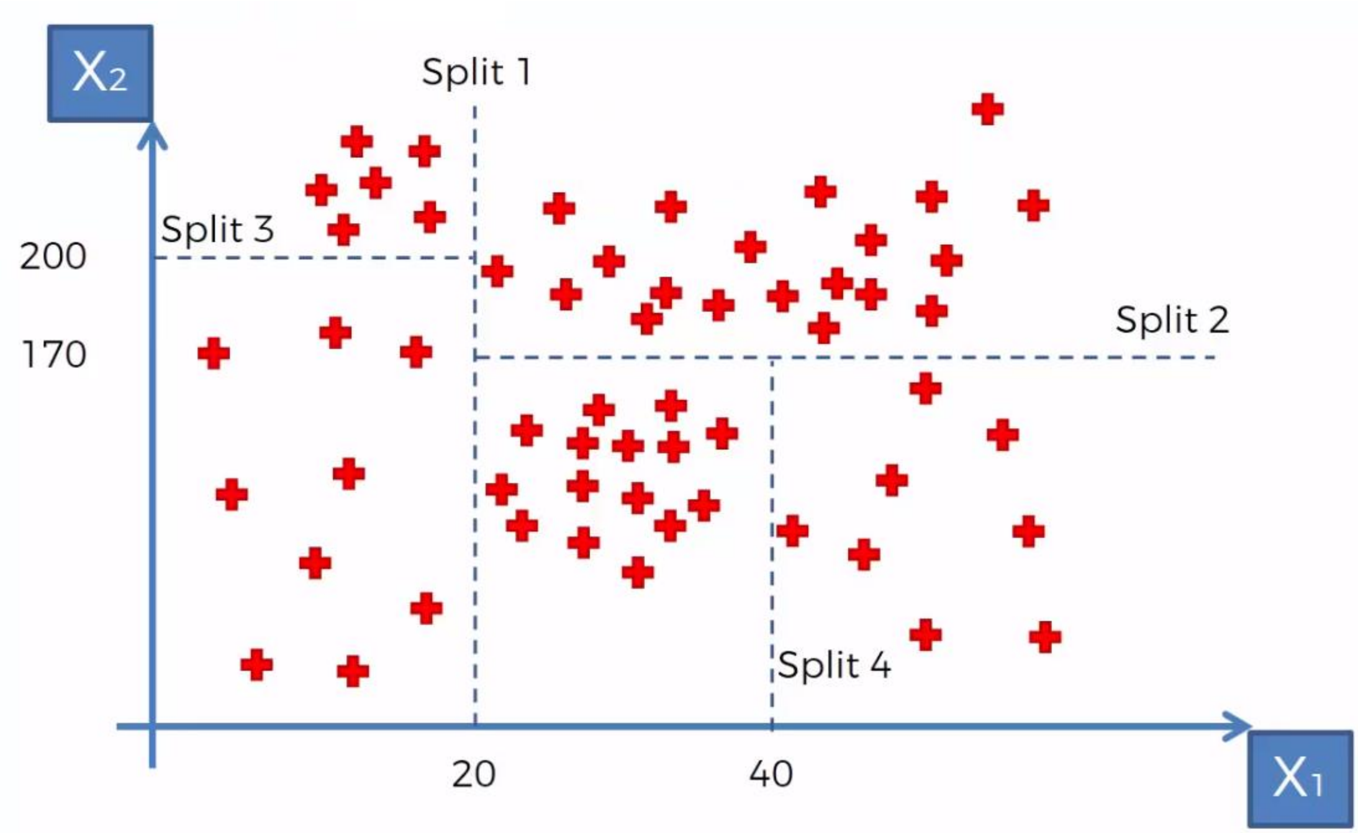
DT



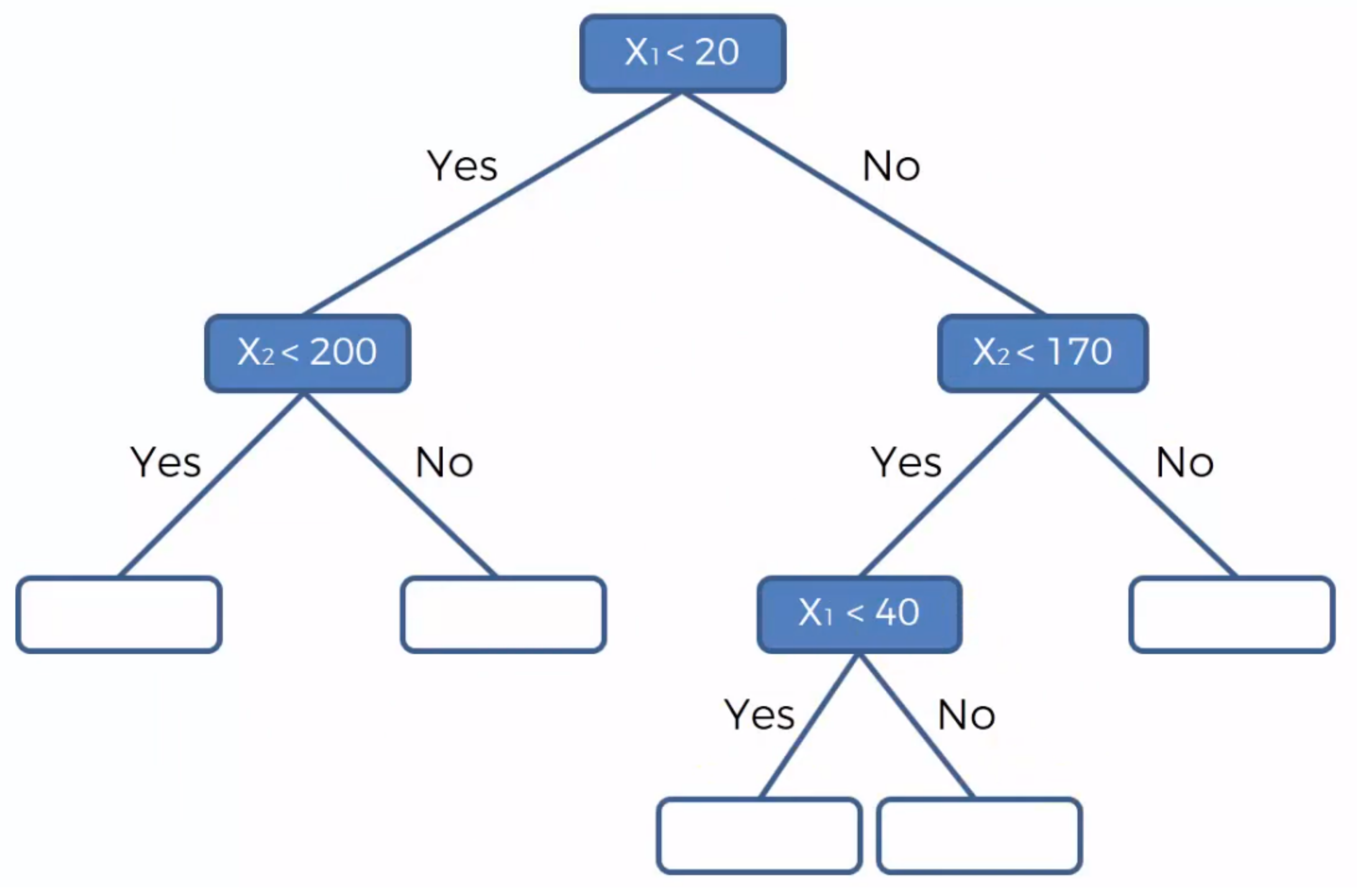
DT



DT

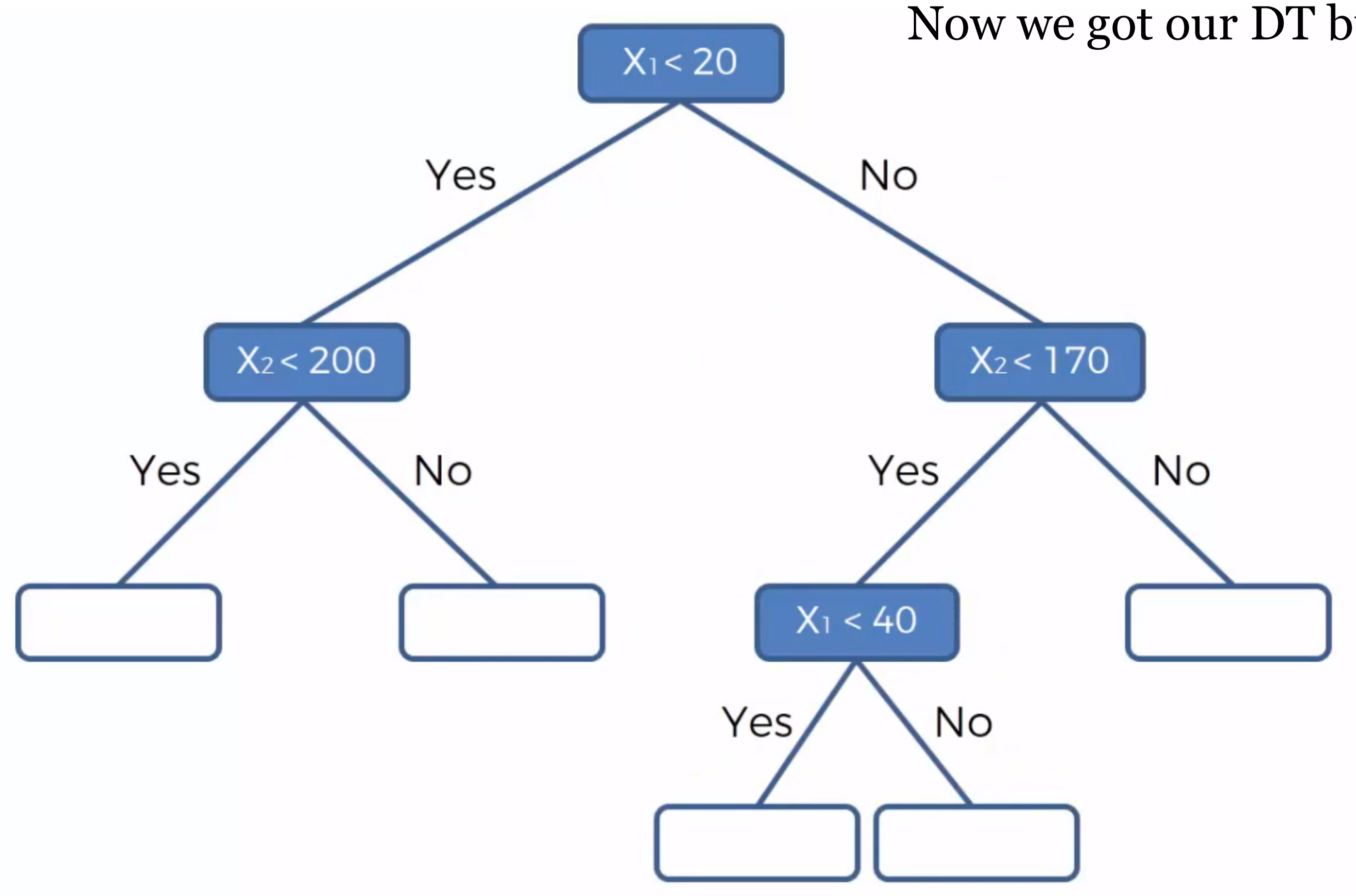


DT



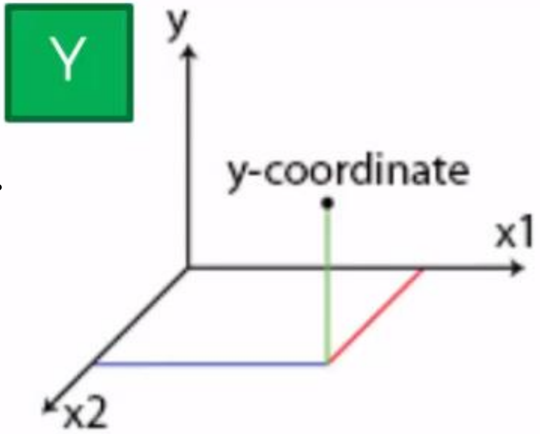
DT

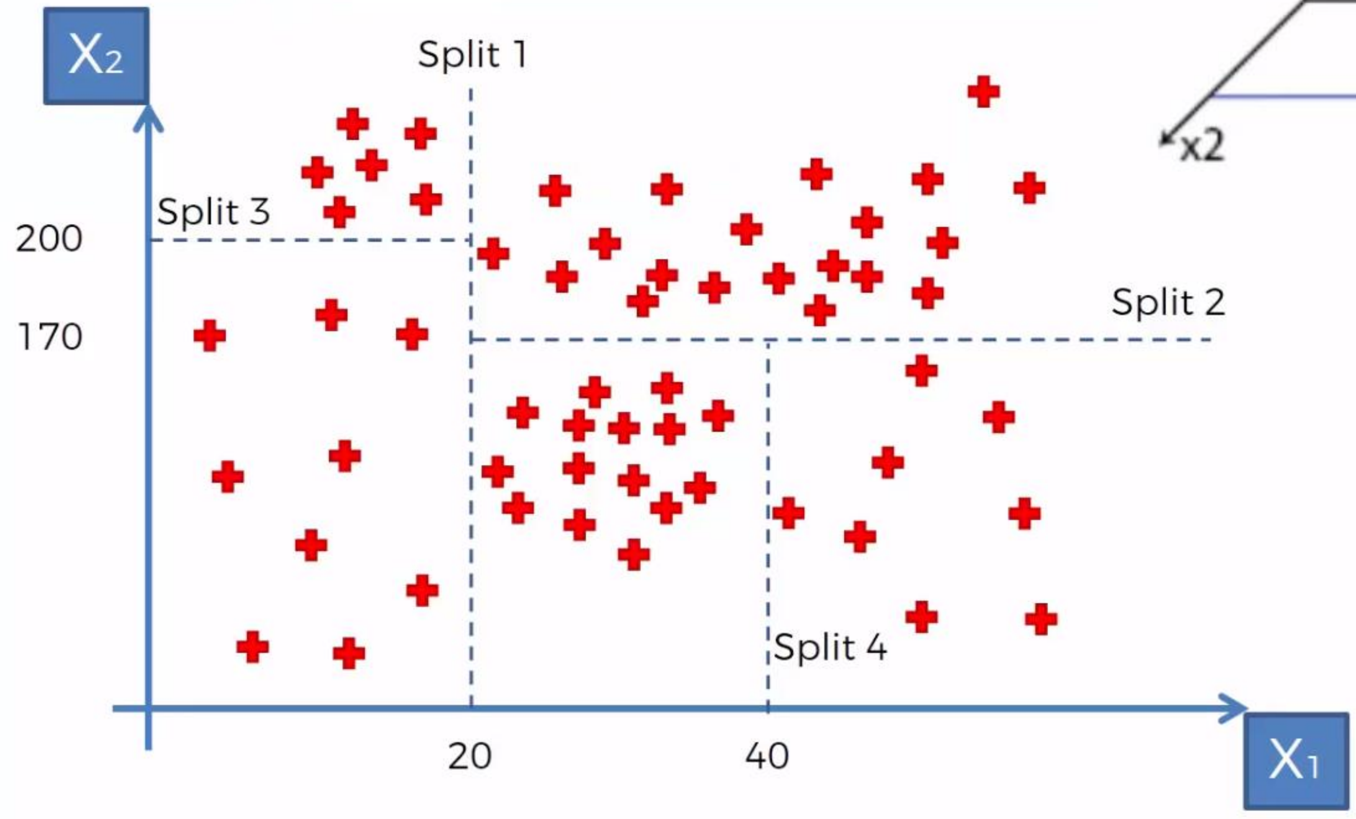
Now we got our DT built!



DT

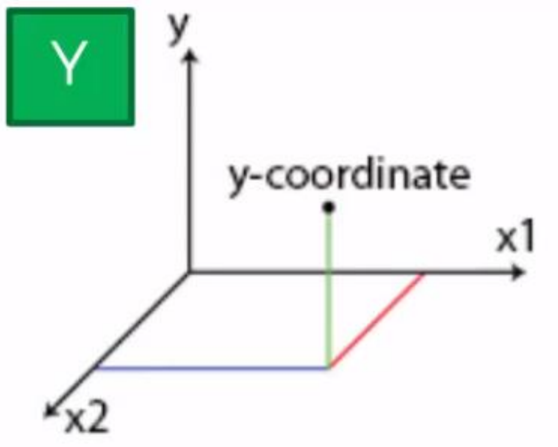
Now we got our DT built!
It's time to consider our third dimension.
Labels

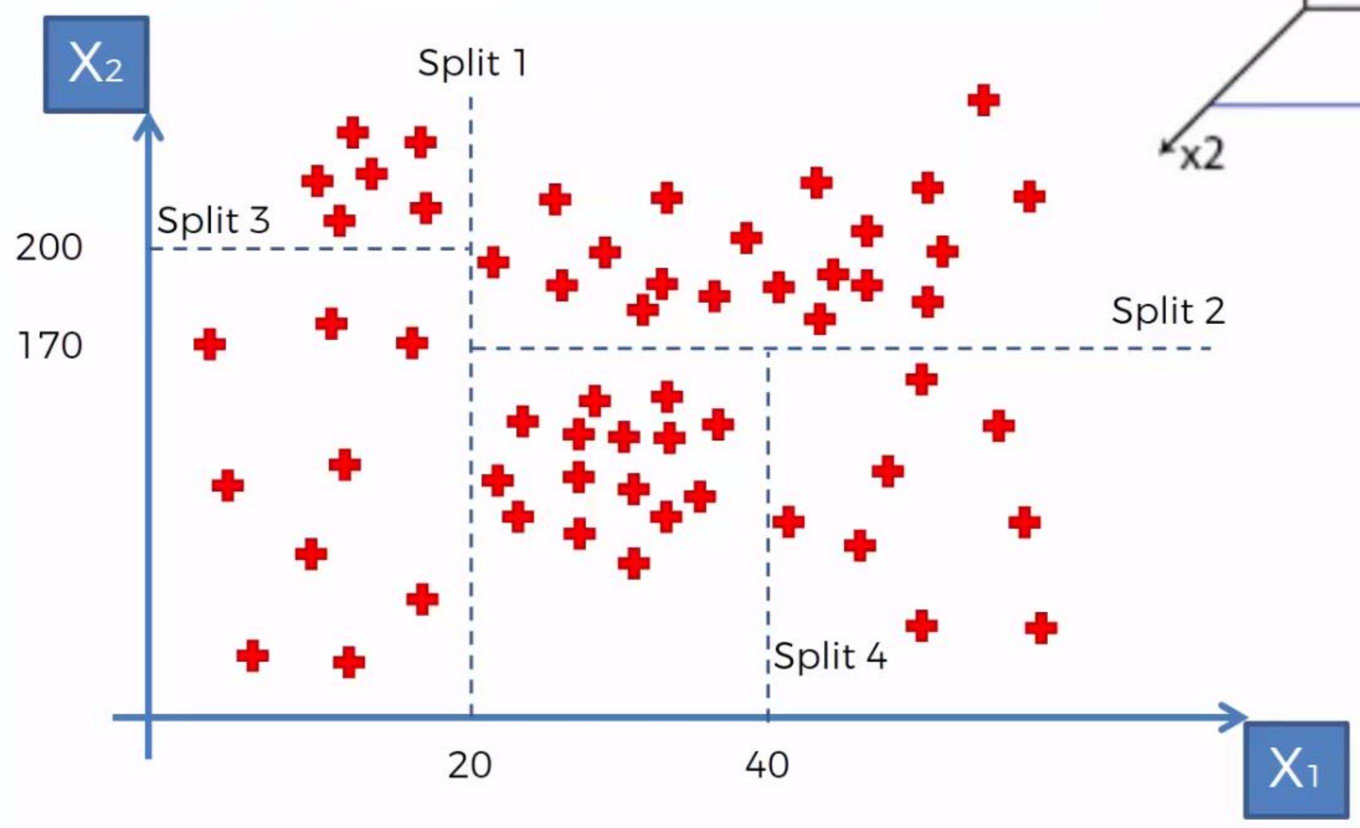




DT

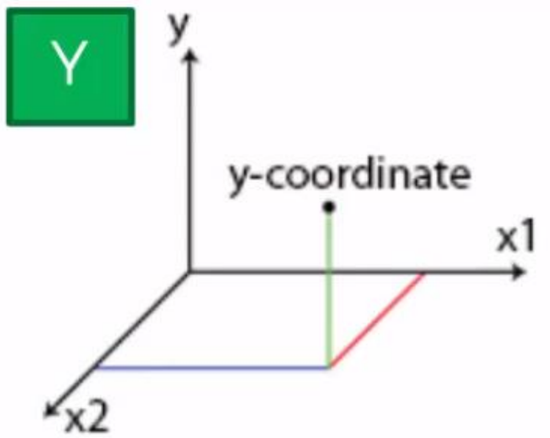
How do we actually do predictions??

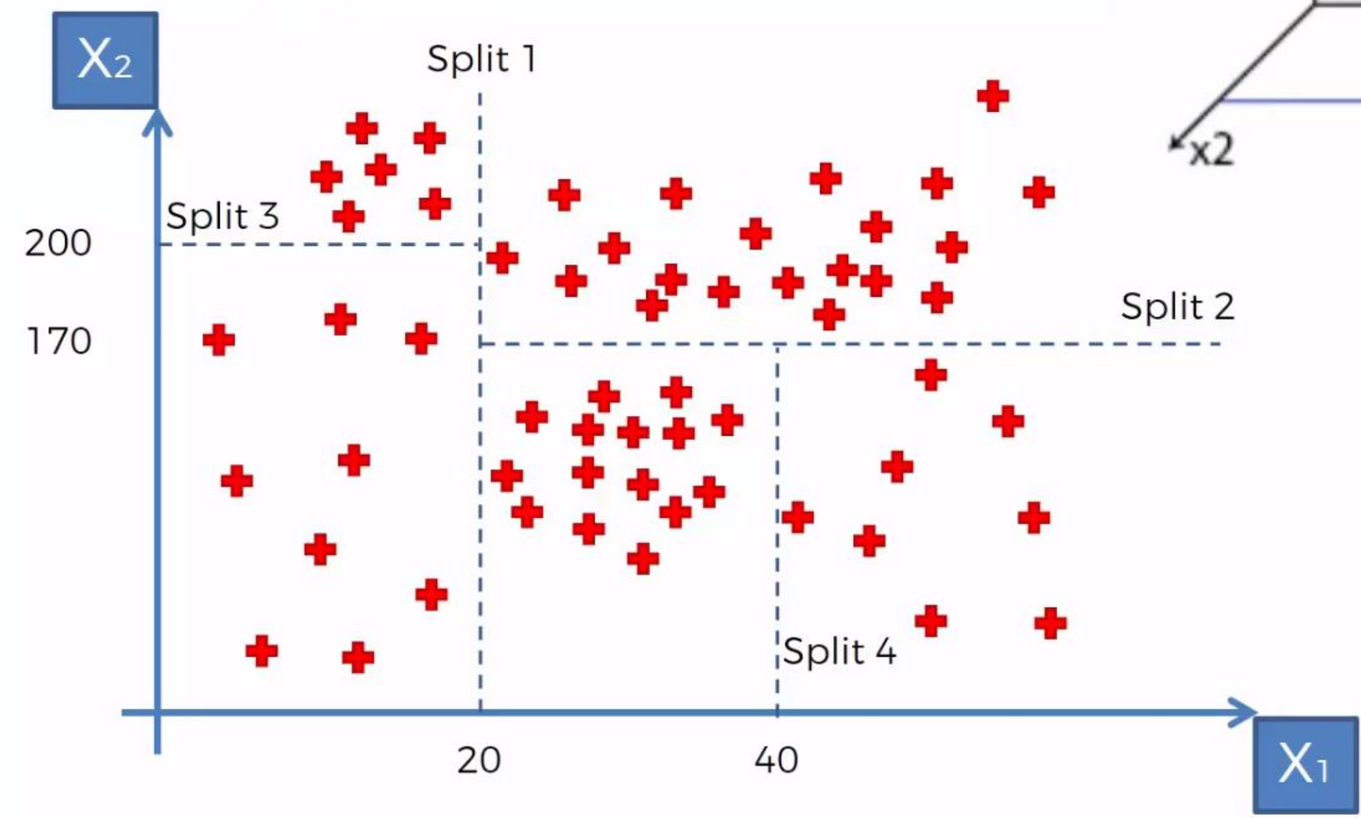




DT

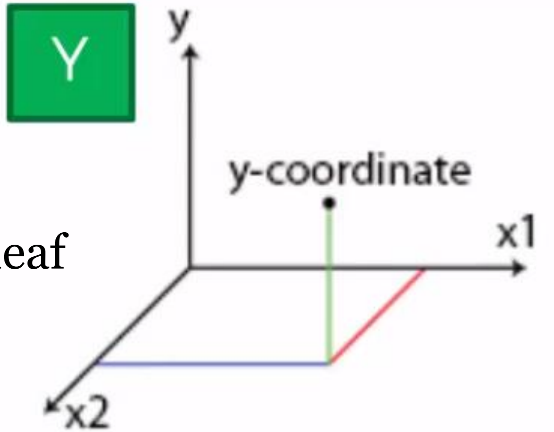
How do we actually do predictions??
Let's say we got a new data point with:
X1 = 30 and X2 = 50

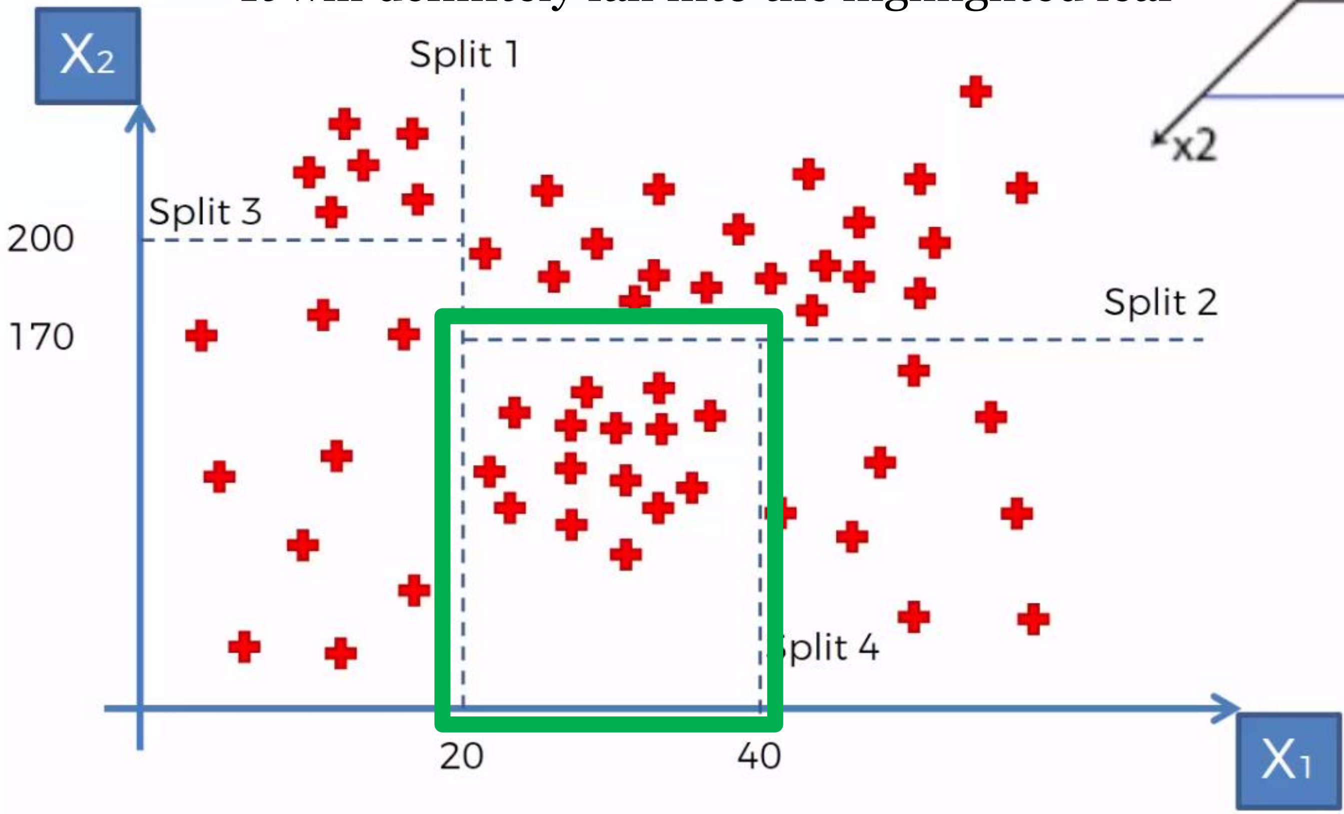




DT

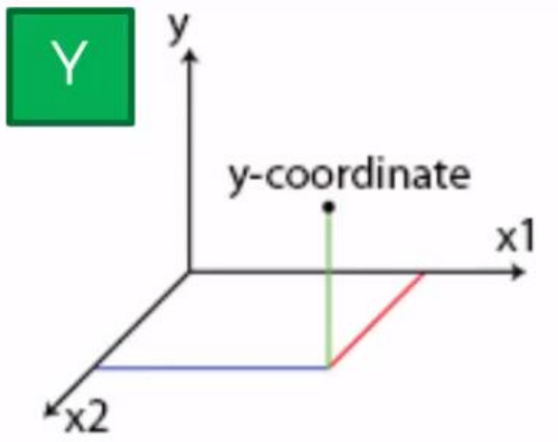
How do we actually do predictions??
Let's say we got a new data point with:
X1 = 30 and X2 = 50
It will definitely fall into the highlighted leaf

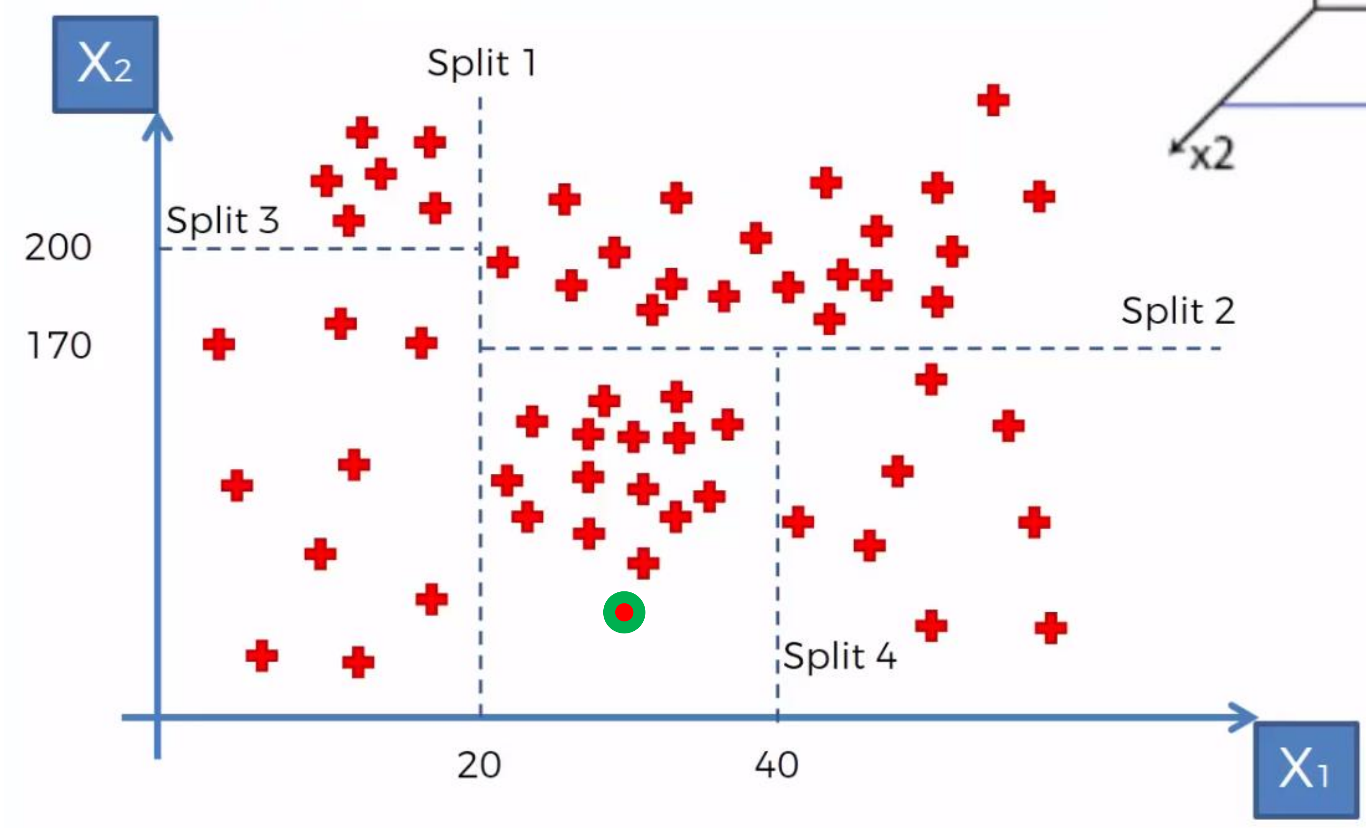




DT

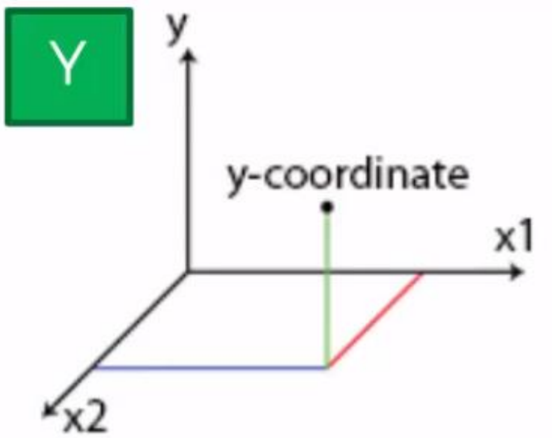
Can we predict the Y of new data point?

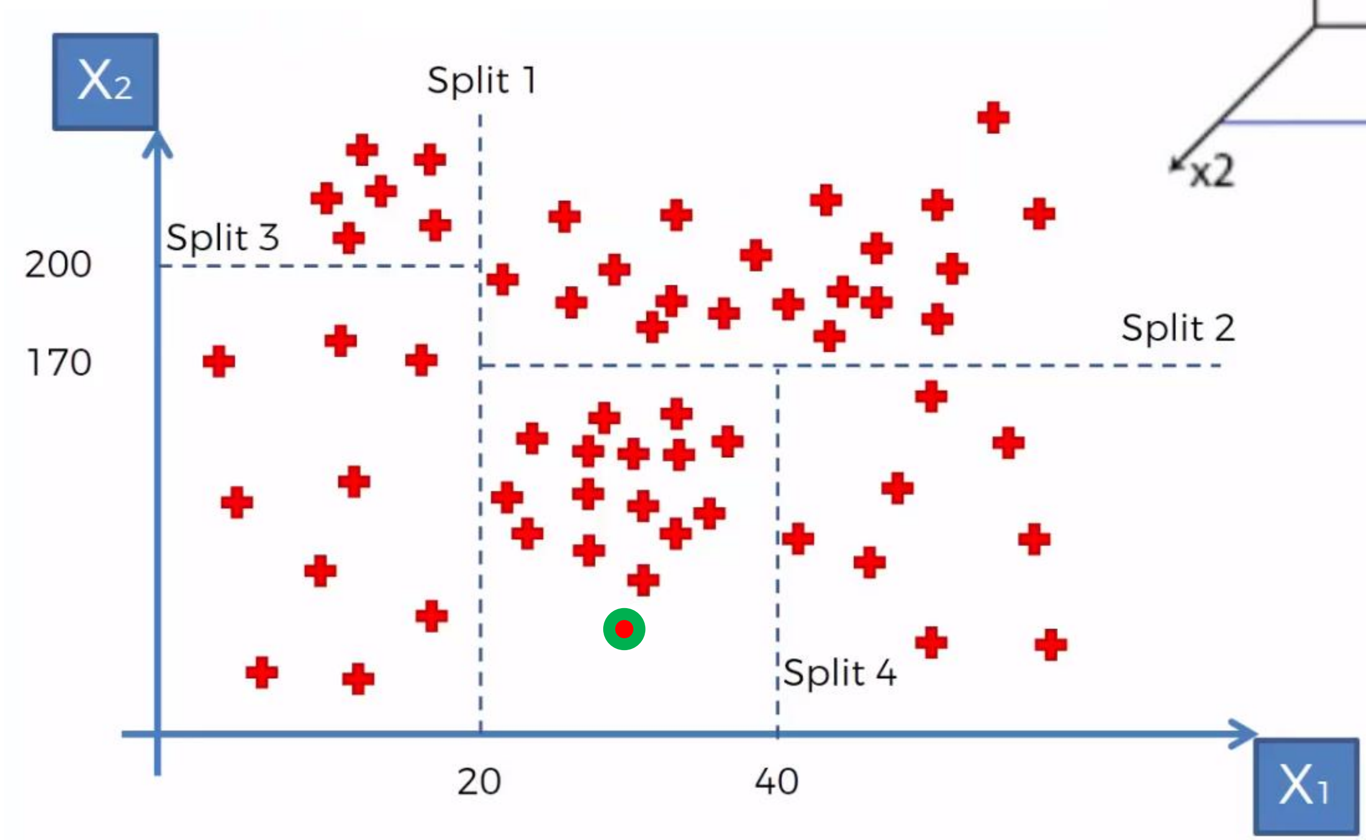




DT

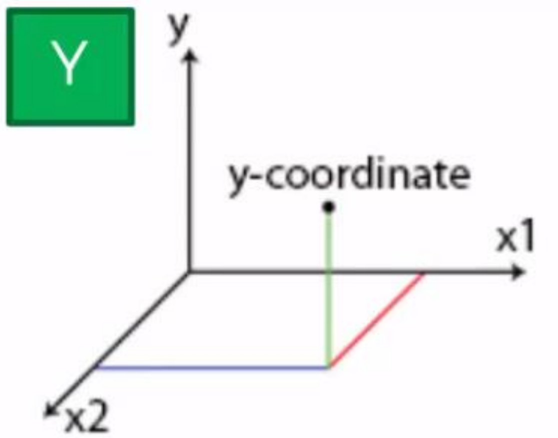
What we have to do is to get the **average** of each leaf's outputs (Y)

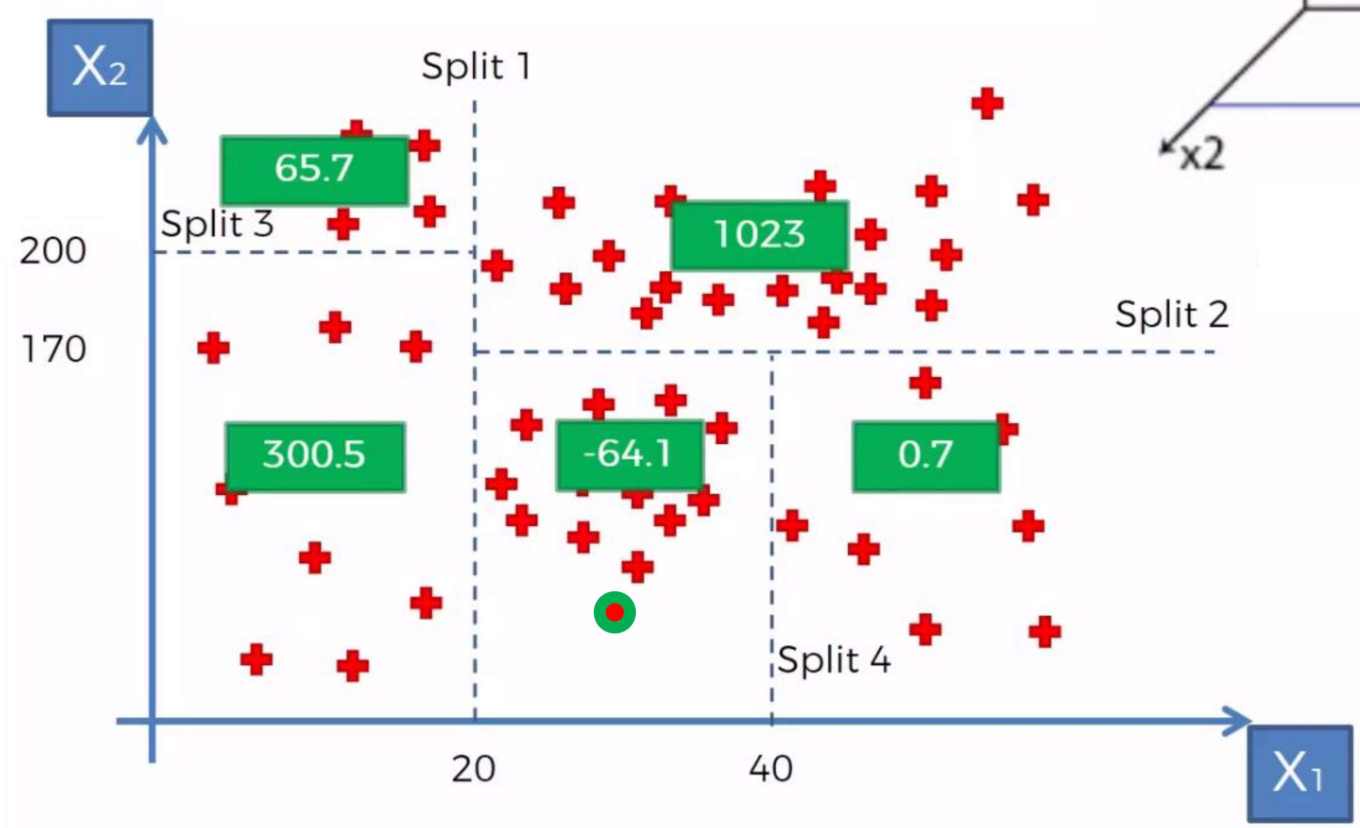




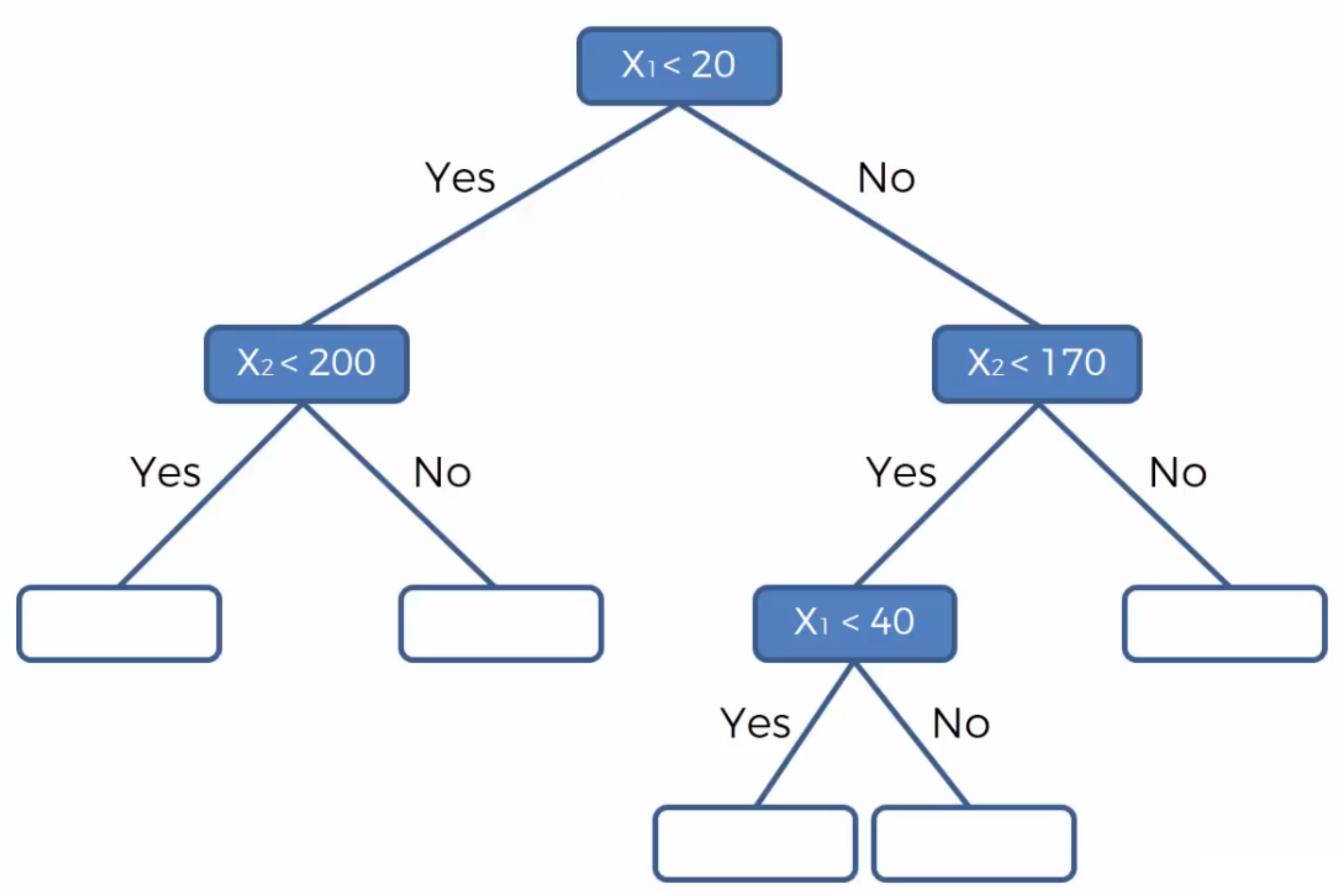
DT

What we have to do is to get the **average** of each leaf's outputs (Y)

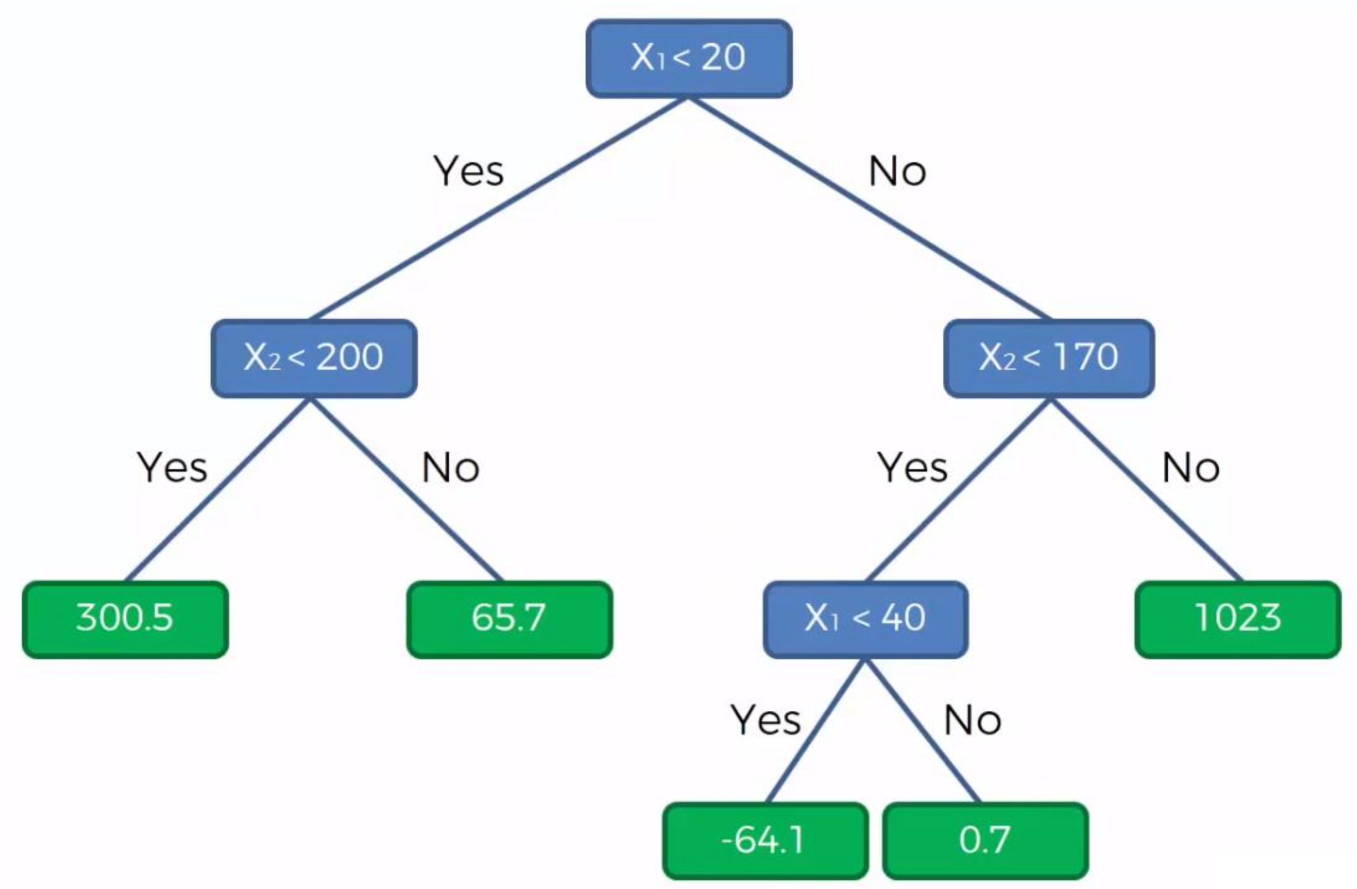




DT



DT


X1 < 20
Yes
No
X2 < 200
X2 < 170
Yes
No
Yes
No
300.5
65.7
X1 < 40
1023
Yes
No
-64.1
0.7