

Decision Tree & Random Forest

Mehmet Güray Güler

END3971

Artificial Intelligence and Expert Systems

Decision Tree

Supervised Learning

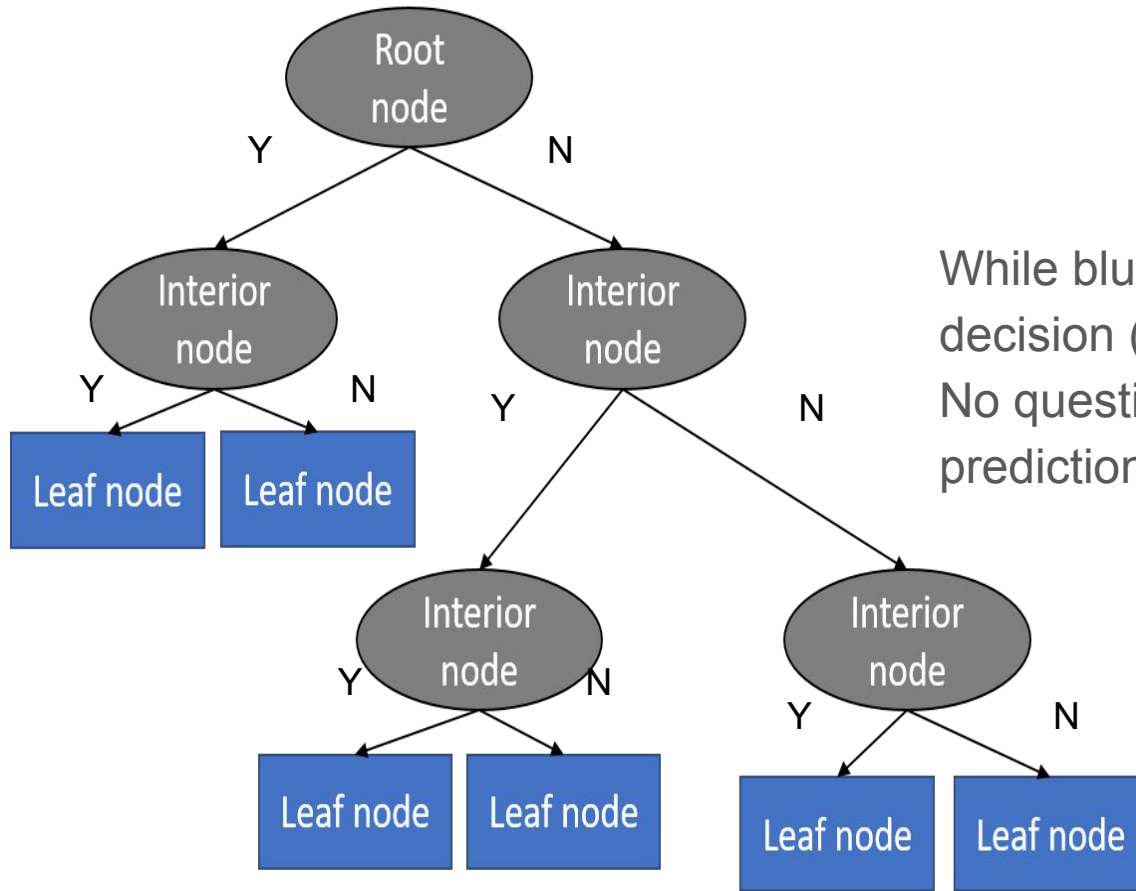
Can be used for both **Classification** and **Regression**

Also known as **CART** \Rightarrow **C**lassification **A**nd **R**egression **T**rees

As the name goes, it uses a tree-like model of decisions.

Predictions (classification or regression) are made based on **yes** or **no** answers.

Gives better results in classification than regression. Therefore, the classification trees are more used.



While blue **rectangles** are the final decision (prediction), **oval shapes** are Yes-No questions (i.e. alternatives) for making predictions.

Pros of Decision Tree

- ❑ Can be used both numerical and categorical data.
- ❑ Can also handle multi-output problems.
- ❑ Requires relatively little effort from user for data preparation
- ❑ Able to solve non-linear relationships between parameters
- ❑ Allows for partitioning data in a much deeper level, not as easily achieved with other decision-making classifiers such as logistic regression or support of vector machines.
- ❑ Does not require normalization of data.
- ❑ Missing values in the data also does NOT affect the process of building decision tree to any considerable extent.

Let's consider a very basic example that uses data set for predicting whether a customer will buy computer or not.

We have some information about customers:

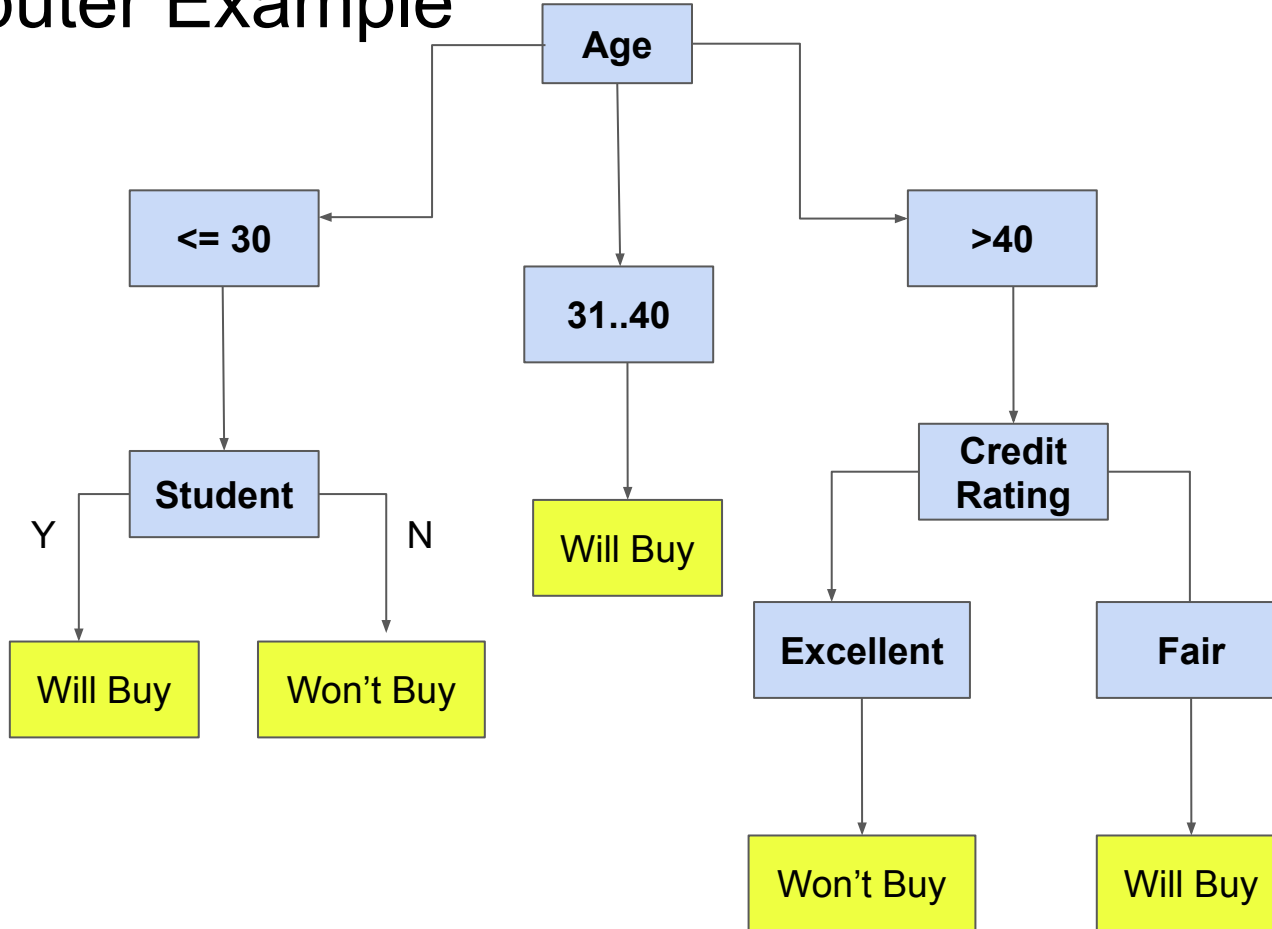
- age
- income
- student (or not)
- credit rating (fair or excellent)

And we need to create a model that will be able to predict whether the customer will buy a computer or not. Suppose we have a **training and test** set for creating proper model.

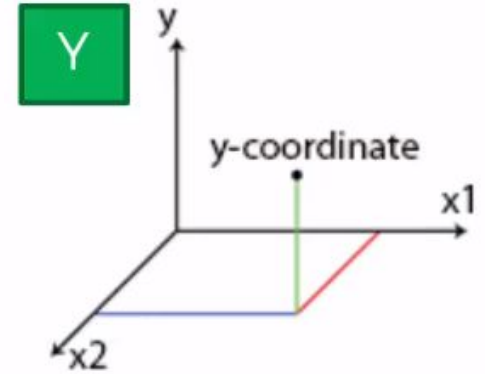
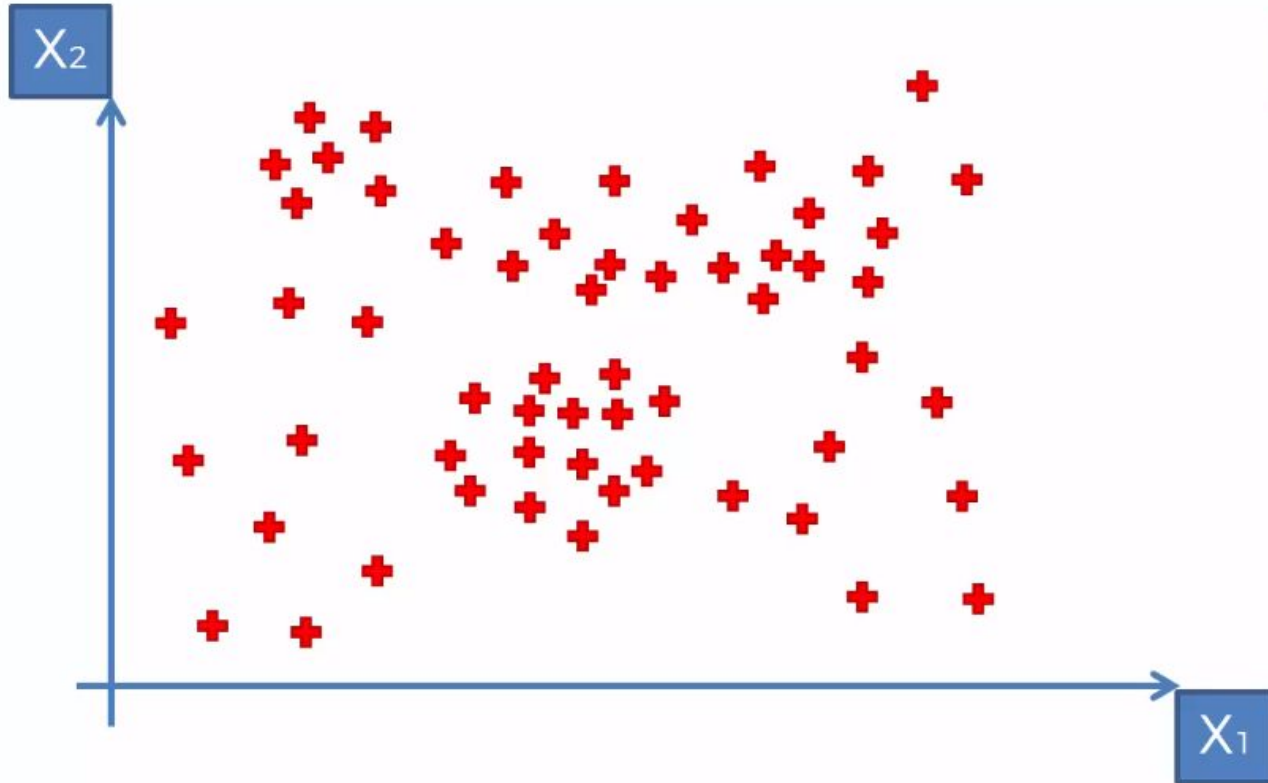
Training Set for Buying Computer Example

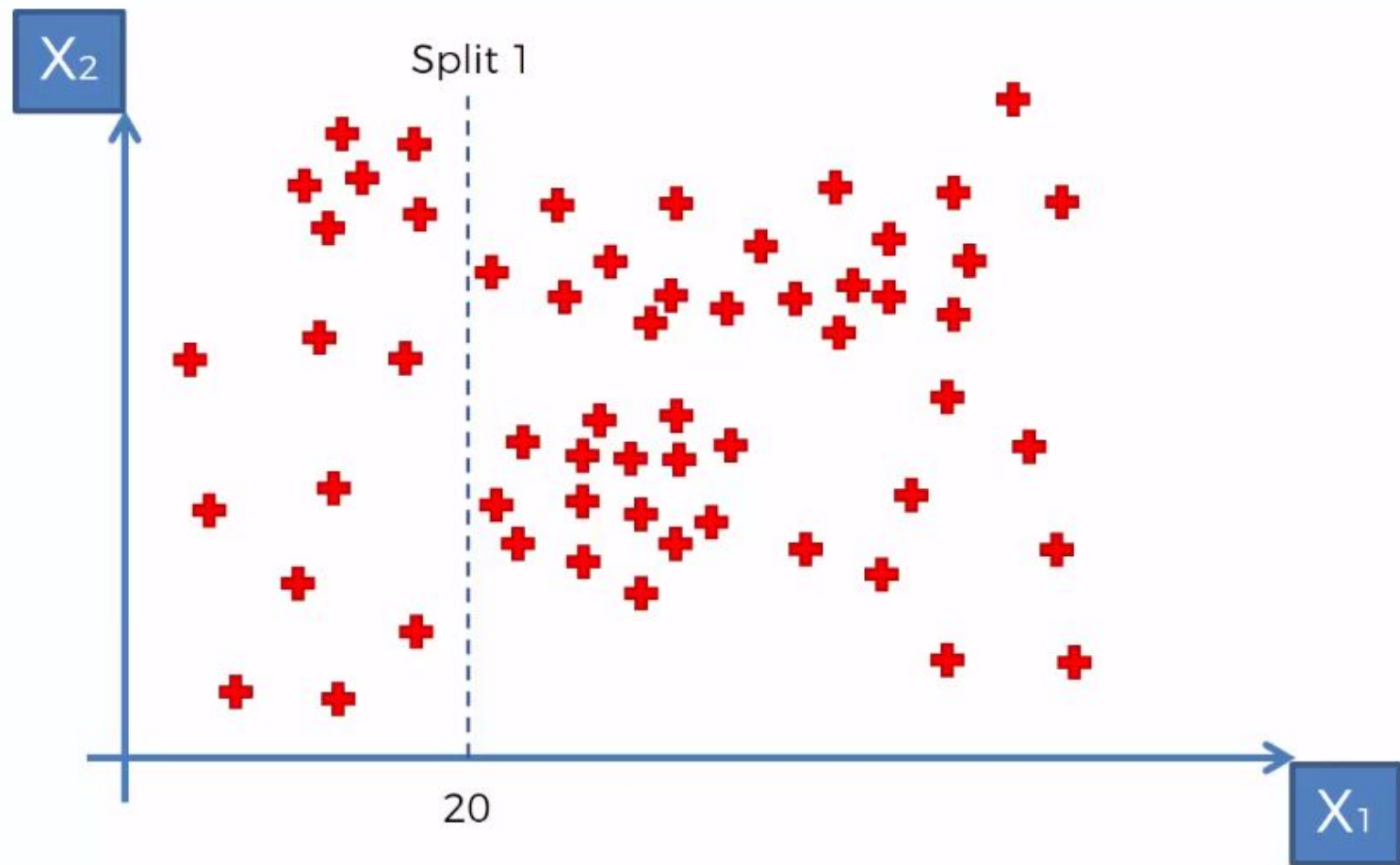
age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

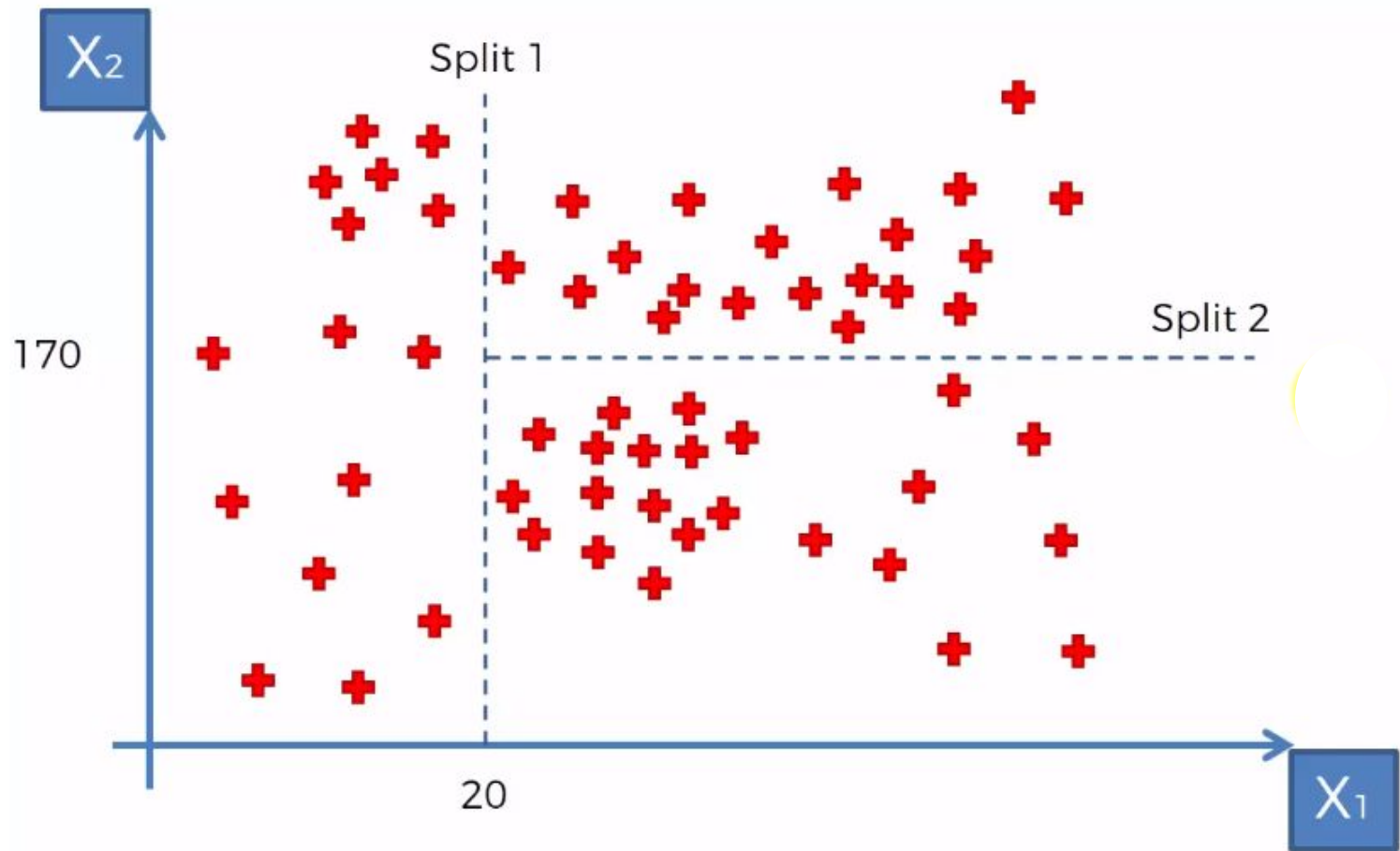
Computer Example

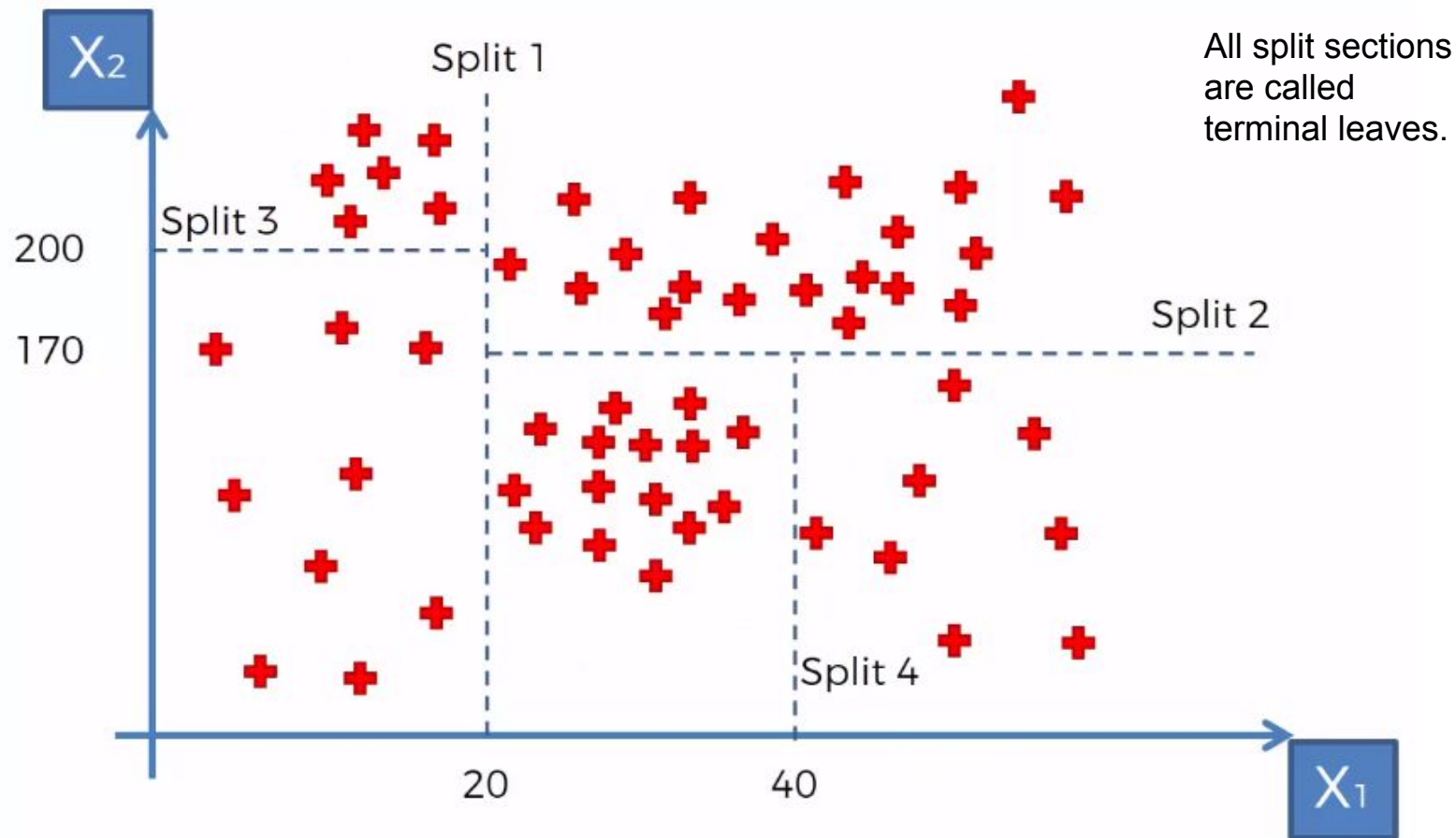


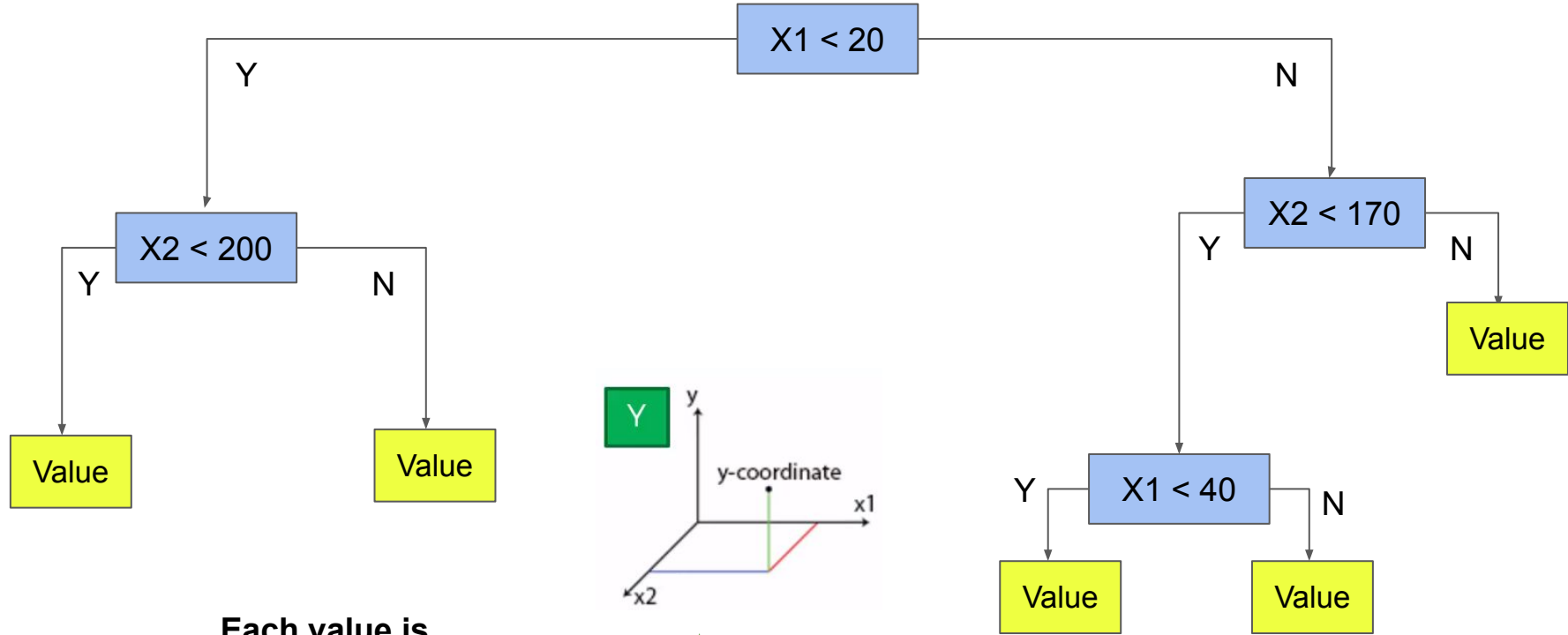
Decision Tree for Regression



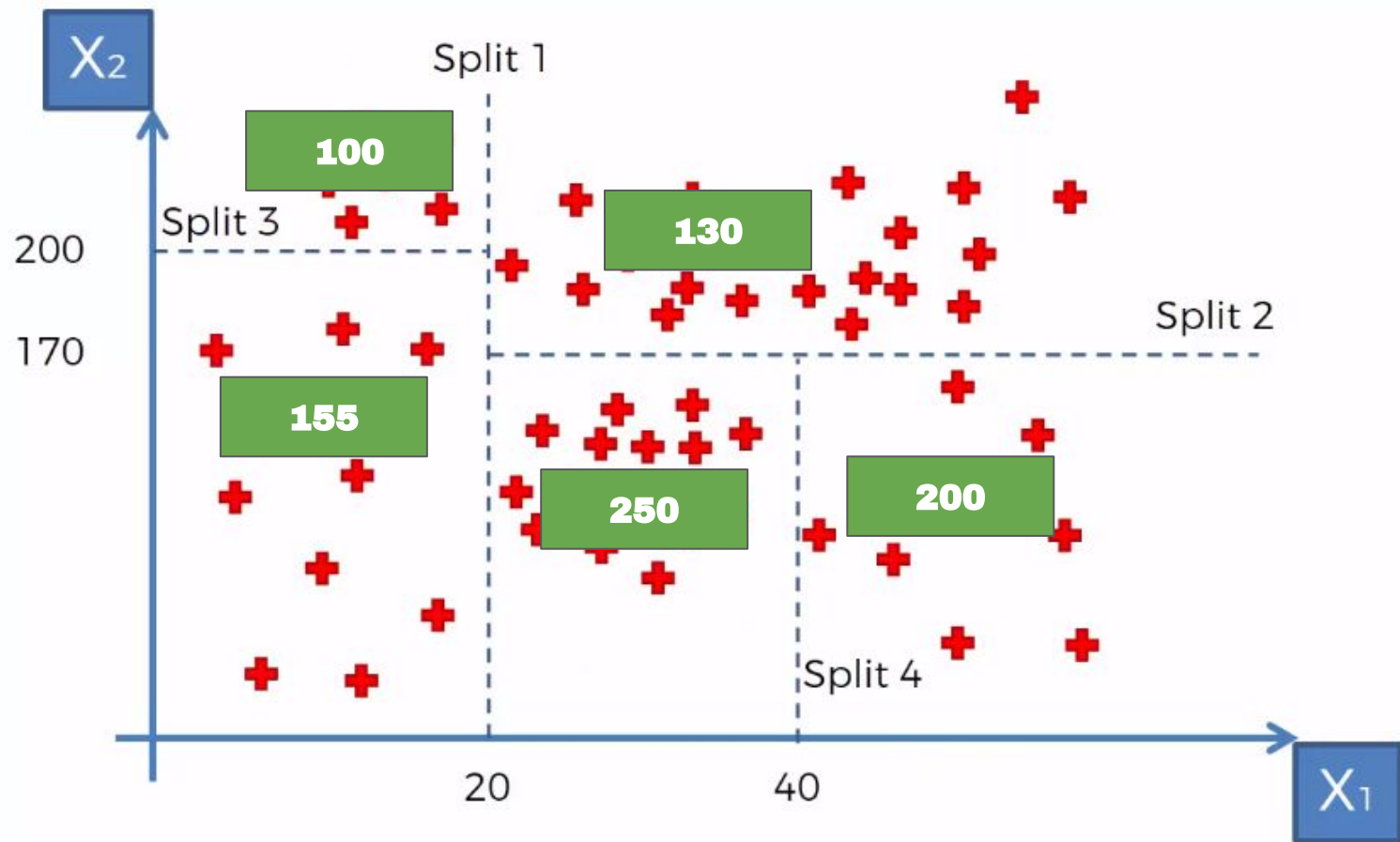


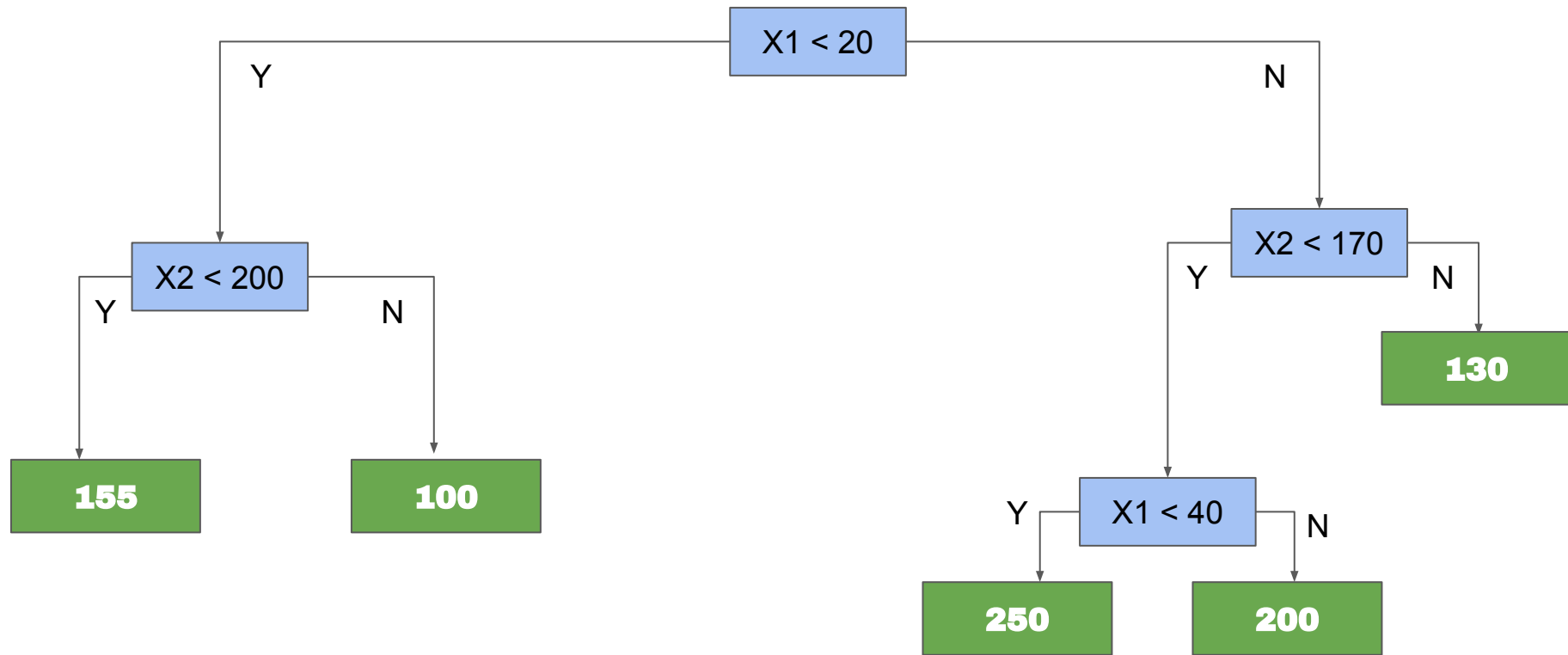




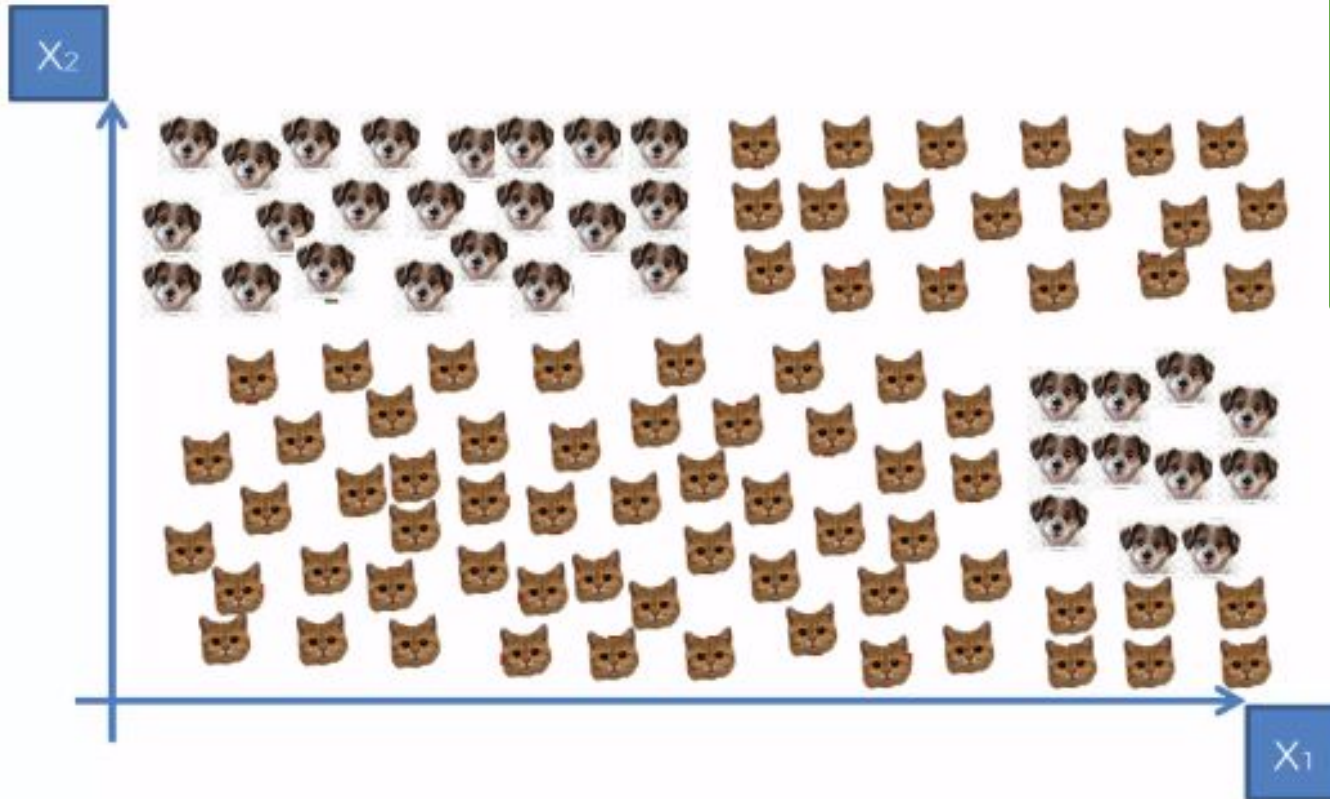


Each value is the average of (y values) the terminal leaf to which it belongs.

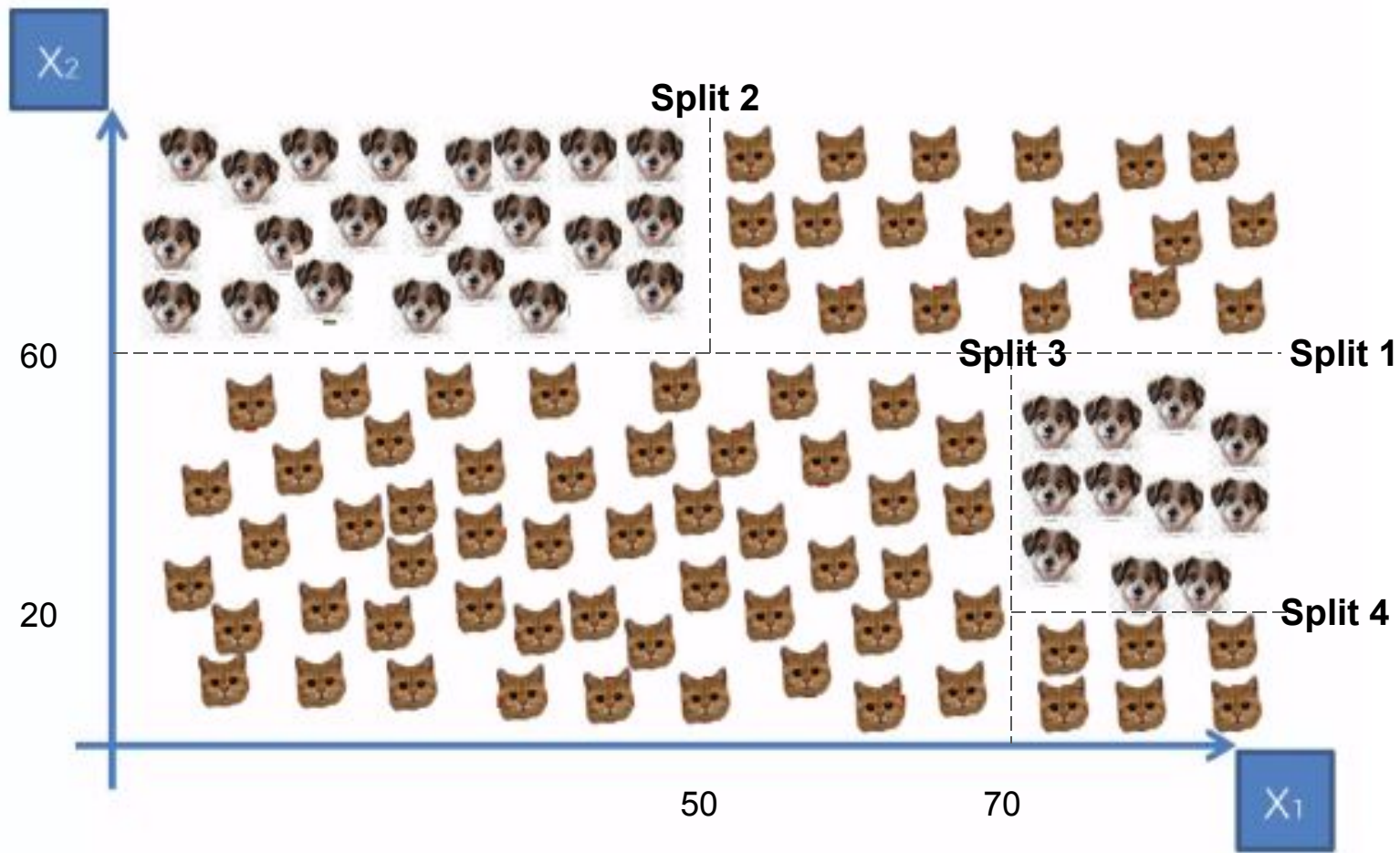


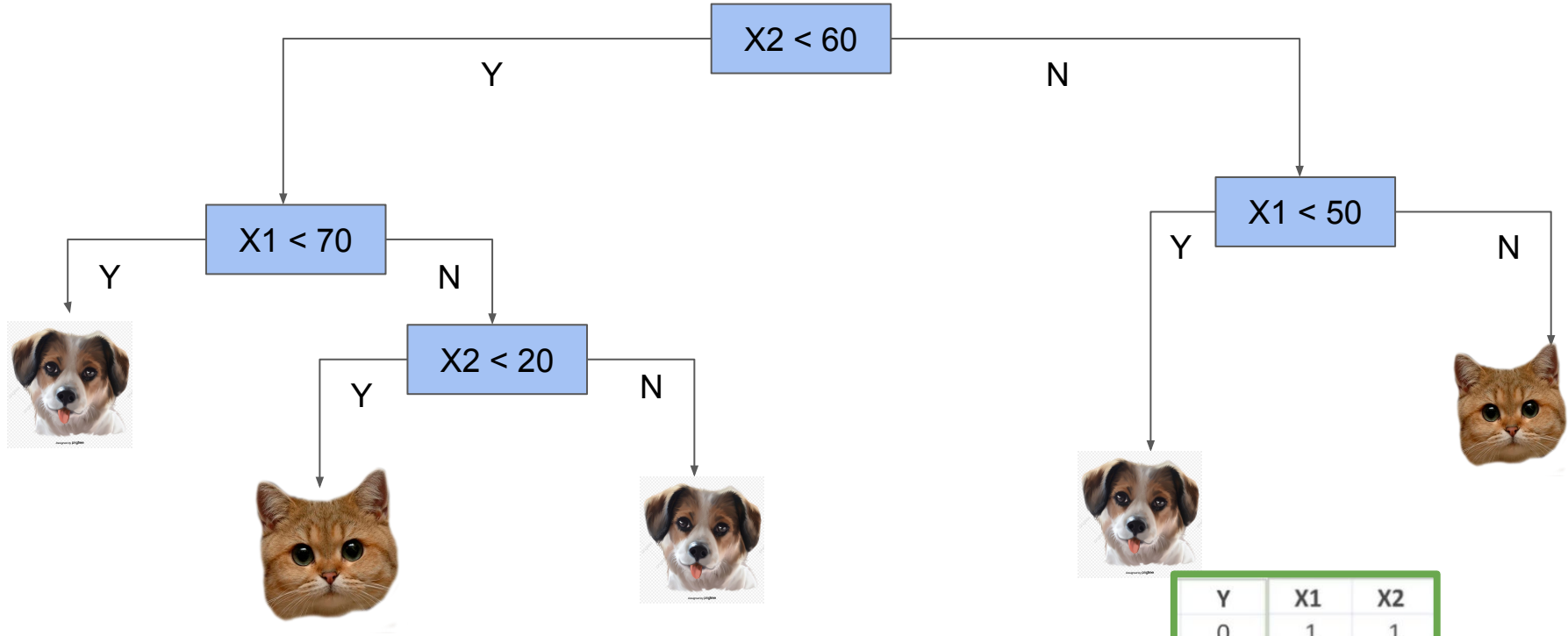


Decision Tree for Classification




Y	X1	X2
0	1	1
0	1	1
0	0	0
1	0	1
1	1	0
1	0	1





As a result, the value to be given (here, cat or dog) is the majority of values in the terminal leaf to which they belong.



Y	X1	X2
0	1	1
0	1	1
0	0	0
1	0	1
1	1	0
1	0	1

Cons of Decision Tree

- ❑ Inadequate for applying regression and predicting continuous values.
- ❑ If a decision tree is a fully grown, it may lose some generalization capability (i.e. **overfitting**).
- ❑ Information gain in a decision tree with categorical variables gives a biased response for attributes with greater number of categories.

Random Forest

Random Forest

Ensemble Learning: Helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. That is why ensemble methods placed first in many prestigious machine learning competitions, such as the Netflix Competition, KDD 2009, and Kaggle.

The random forest algorithm is an example of an ensemble learning (because it consists of multiple decision trees).

Random Forest Regression

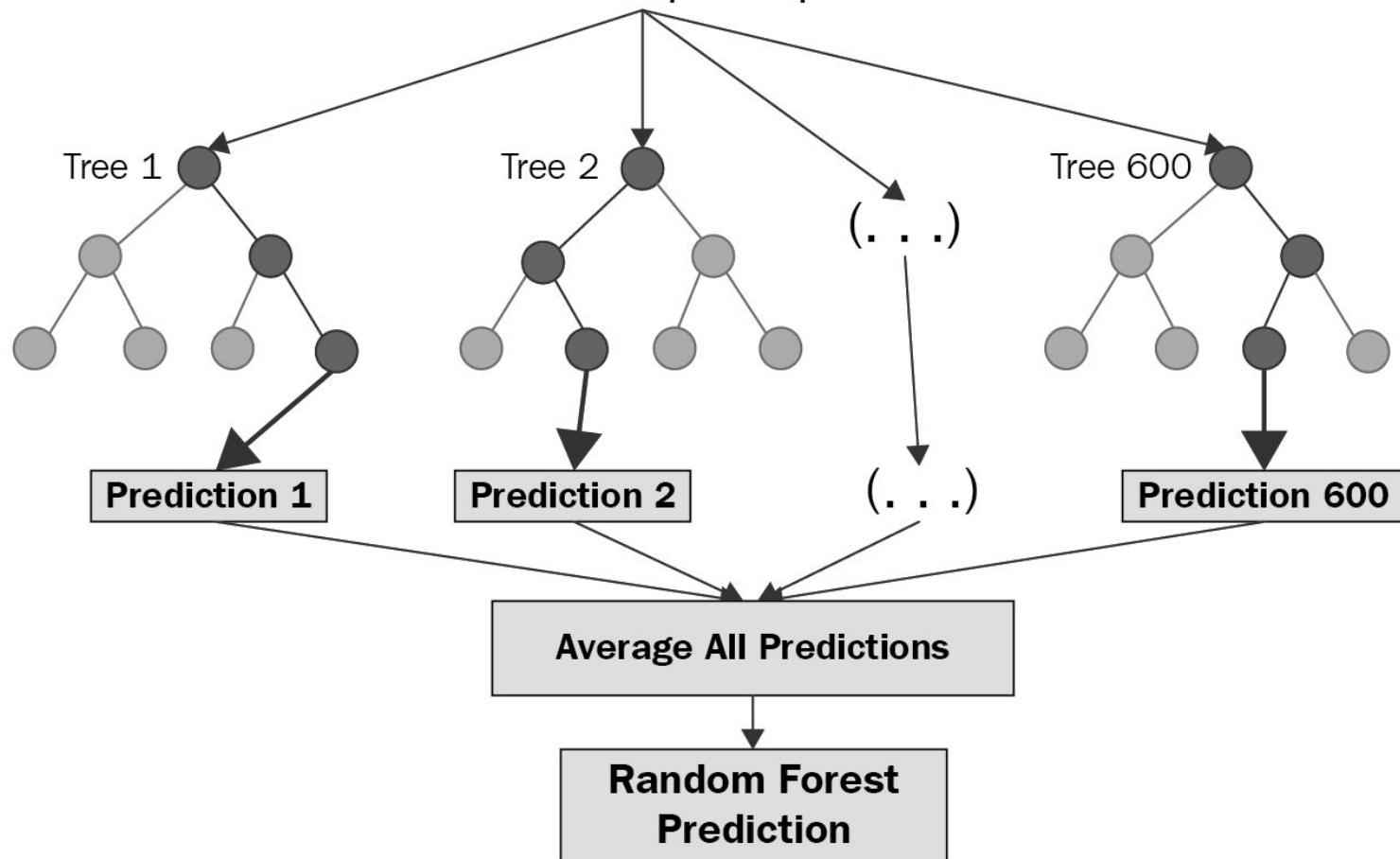
Step 1 : Pick at random K data points from the Training set.

Step 2: Build the Decision Tree associated to these K data points.

Step 3: Choose the number of N_{tree} of you want to build and repeat STEPS 1&2.

Step 4: For a new data point, make each one of your N_{tree} trees predict the value of Y for the data point in question, and assign the new data point the **average** across all of the predicted Y values.

Test Sample Input



Random Forest Classification

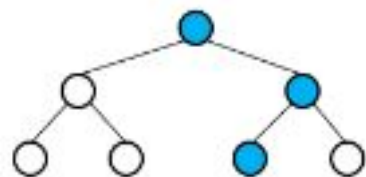
Step 1 : Pick at random K data points from the Training set.

Step 2: Build the Decision Tree associated to these K data points.

Step 3: Choose the number of N_{tree} of you want to build and repeat STEPS 1&2.

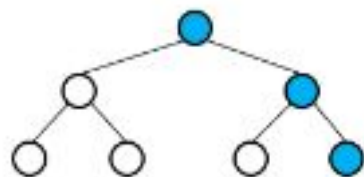
Step 4: For a new data point, make each one of your N_{tree} trees predict the category to which data point belongs, and assign the new data point to the **category that wins the majority vote.**

X dataset



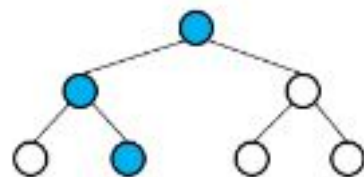
TREE #1

CLASS C



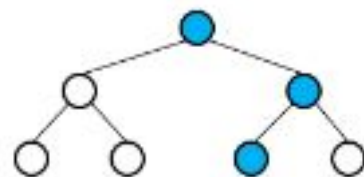
TREE #2

CLASS D



TREE #3

CLASS B



TREE #4

CLASS C

MAJORITY VOTING

FINAL CLASS

Pros and Cons of Random Forest

- ❑ To find any in the population, the greater the number of samples we use, the closer we get to the real value (remember from statistics).
- ❑ It has an effective method for estimating missing data and maintains accuracy when large proportion of the data are missing.
- ❑ It surely does a good job at classification but not as for regression problem as it does not gives precise continuous nature prediction. In case of regression, it doesn't predict beyond the range in the training data, and that they may over fit data sets that are particularly noisy.

An Example from Microsoft



Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

Abstract

We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. Our large and highly varied training dataset allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes.

The system runs at 200 frames per second on consumer hardware. Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton nearest neighbor matching.

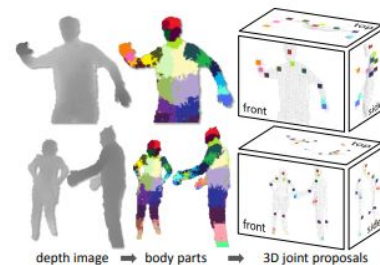


Figure 1. **Overview.** From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.

joints of interest. Reprojecting the inferred parts into world

Microsoft used Random Forest algorithm in Xbox to predict movements.

<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf>