Decision Trees

So far, our models for Classification have attempted to separate classes via *linear* boundaries.

In this module, we will explore a model that facilitates *non-linear* boundaries.

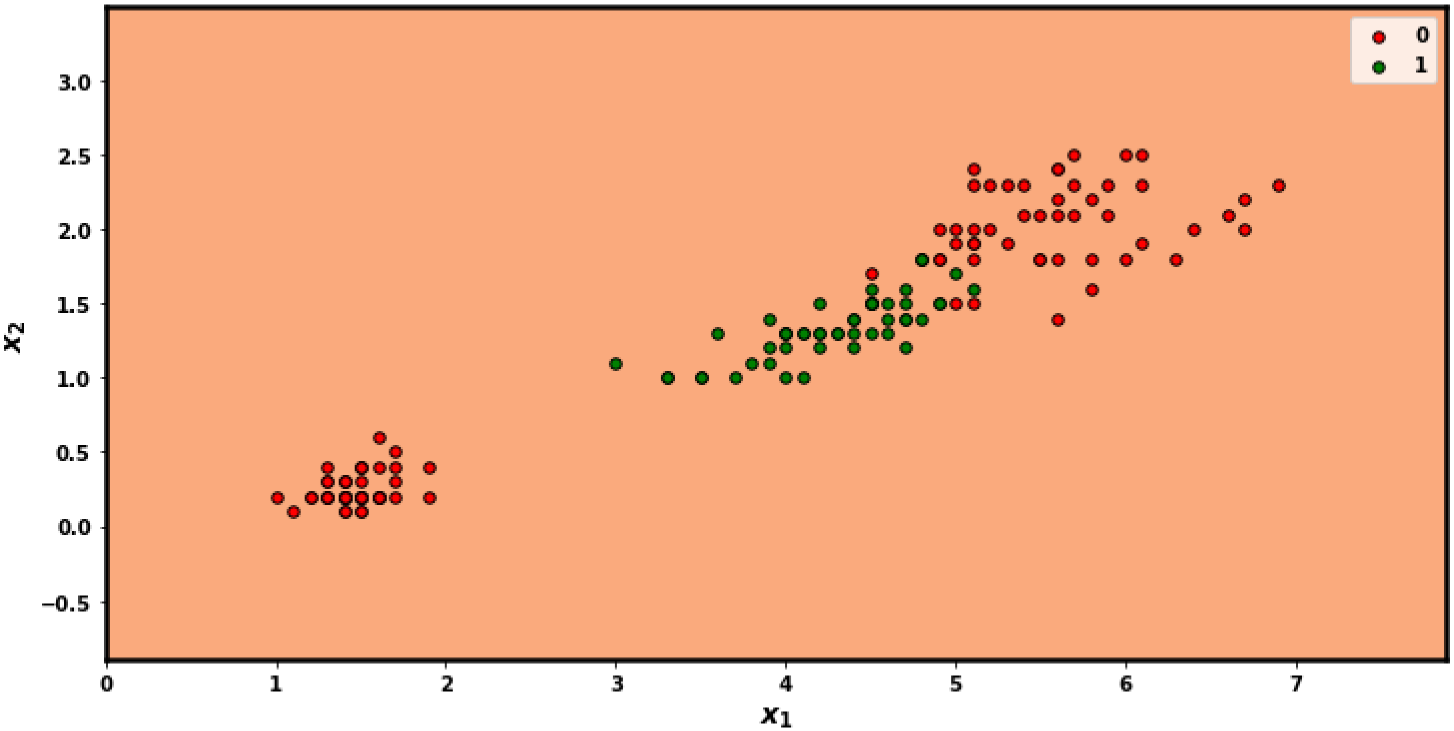
Non-linear boundaries are very powerful, as we will come to see in Deep Learning.

Let's illustrate with an example.

Consider the following dataset for a binary classification task (classes depicted as Red and Green).

There is no linear boundary to completely separate the classes.

In [5]:
```
fig, ax = plt.subplots(figsize=(12,6))
_= bh.make_boundary(X_2c, y_2c, depth=1, ax=ax)
```



Aside

Perhaps a transformation that added an $\mathbf{x}_3$ feature

- That was high for examples with $(\mathbf{x}_1, \mathbf{x}_2)$ close to $(4, 1.25)$ and low for examples far away
- e.g., RBF
- Would induce linear separability ?

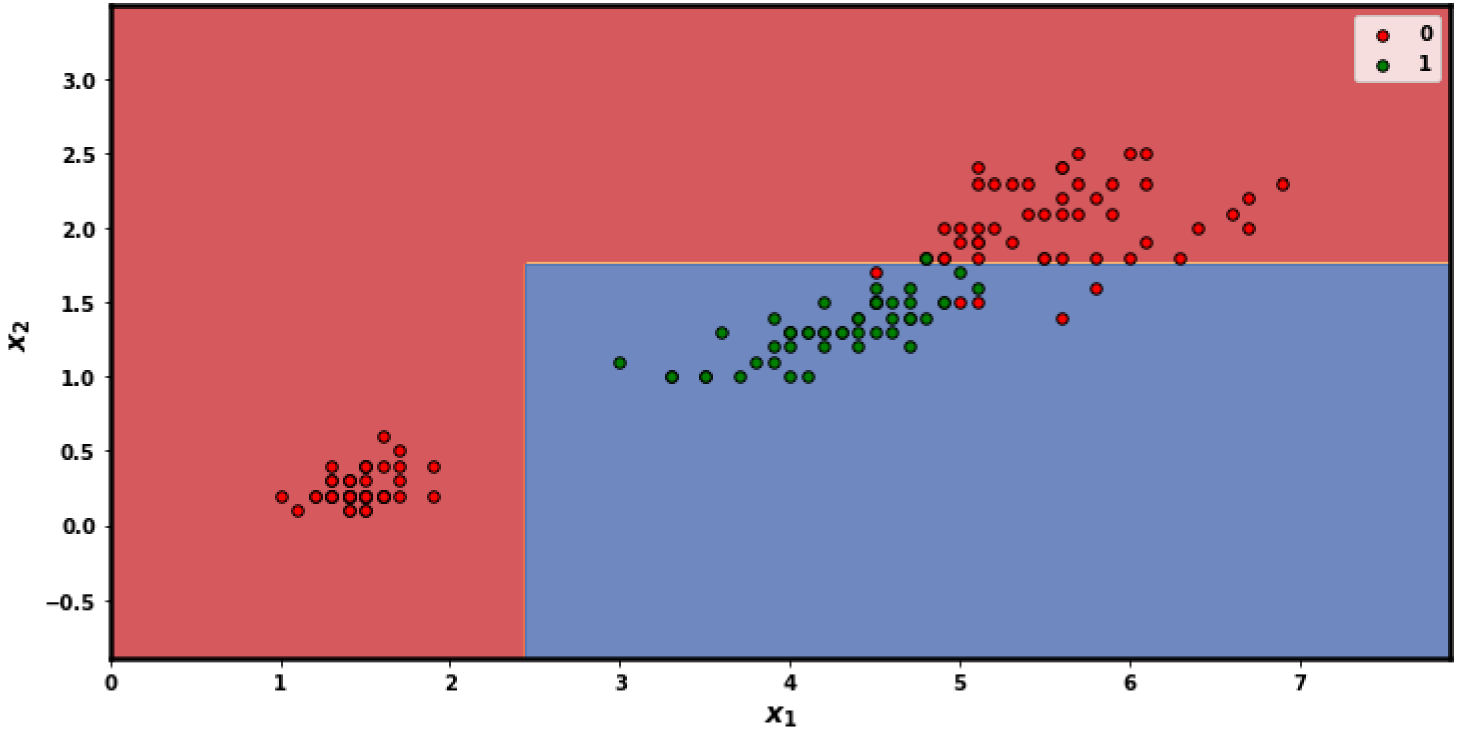
In this module:

- We will explore models with non-linear boundaries
- Rather than transformations followed by linear models

But allowing a simple non-linear boundary does a pretty good job.

In [6]:

```
fig, ax = plt.subplots(figsize=(12,6))
bh  = decision_trees_helper.Boundary_Helper()
_= bh.make_boundary(X_2c, y_2c, depth=2, ax=ax)
```

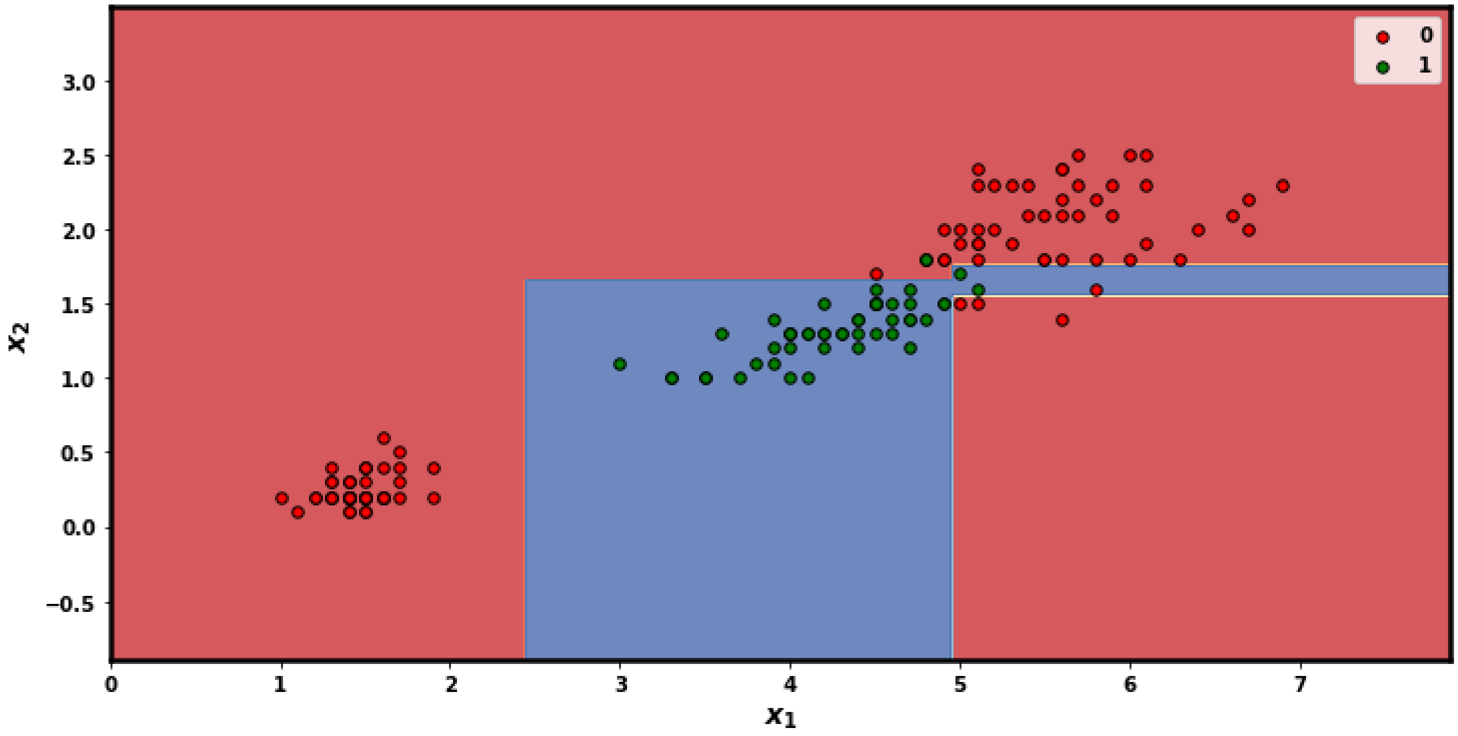


And an even more complex boundary almost completely separates the classes.

- There are still a few Green points in Red territory and vice-versa

In [7]:

```
fig, ax = plt.subplots(figsize=(12,6))
_= bh.make_boundary(X_2c, y_2c, depth=4, ax=ax)
```



Notice that the boundary lines *partition* the values in the range of each feature

- That is: they divide the features according to whether the value is above/below a threshold.

The model that we used creates boundaries via a series of questions, such as

- Is feature j less than $t_{\mathrm{n},j}$?

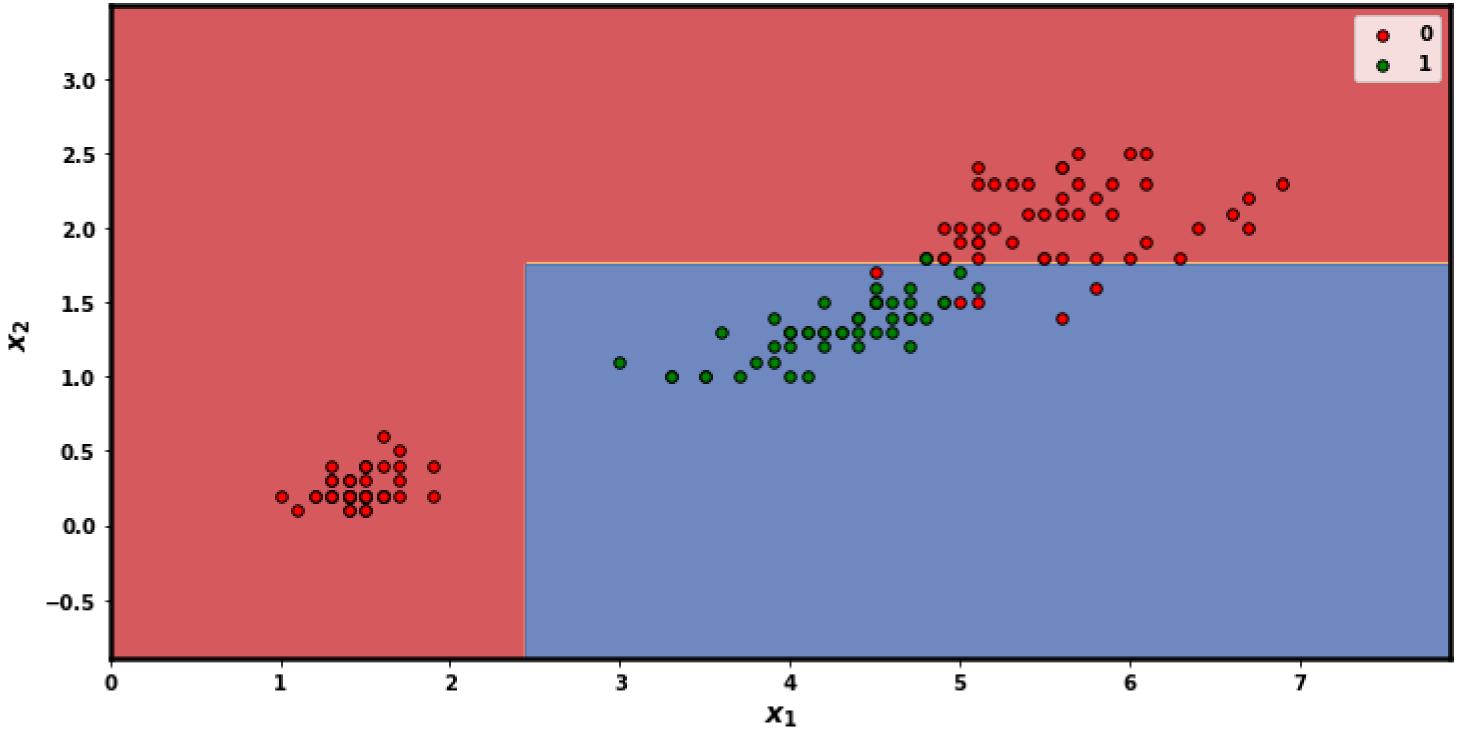
The answers partitions the examples into those with a Positive answer and those with a Negative answer.

We can represent the series of questions as a tree, hence the model is called the *Decision Tree* model.

For example,the partition in the following picture

In [8]:

```
fig, ax = plt.subplots(figsize=(12,6))
bh2  = decision_trees_helper.Boundary_Helper()
_= bh2.make_boundary(X_2c, y_2c, depth=2, ax=ax)
```

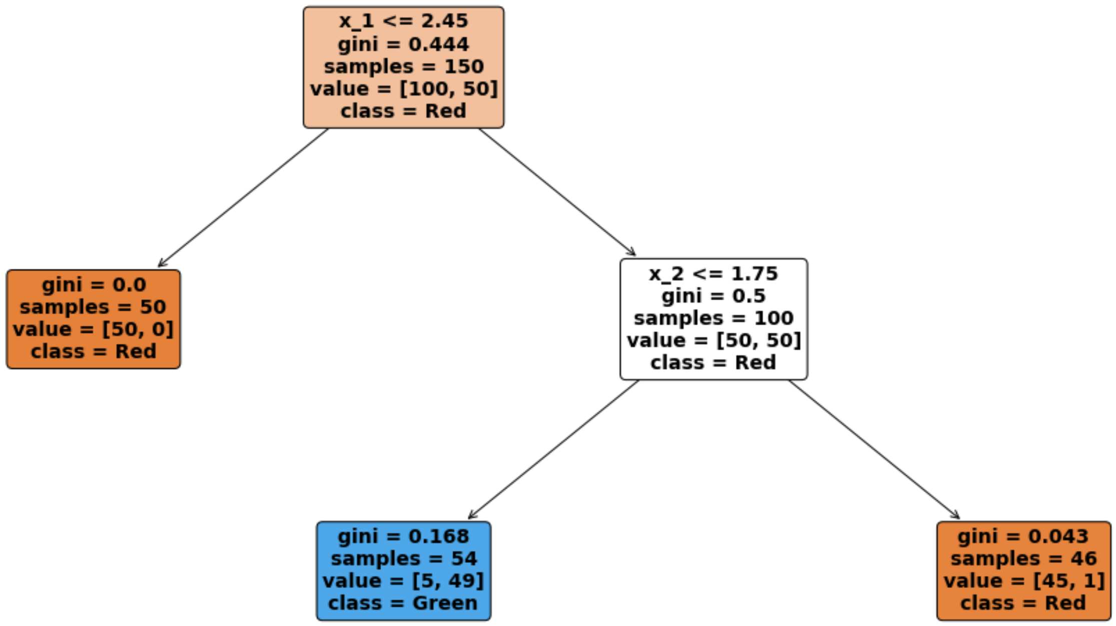


was created by the following tree of questions

In [10]:

```
fig
```

Out[10]:



We will subsequently explain the details of each part of the tree.

For now

- Notice how some questions lead to follow-up questions
    - This is what creates the non-linear boundary
- Some questions have no follow-up
    - These "leaves" of the tree are labeled with the Class (i.e., the decision as to the example's class)

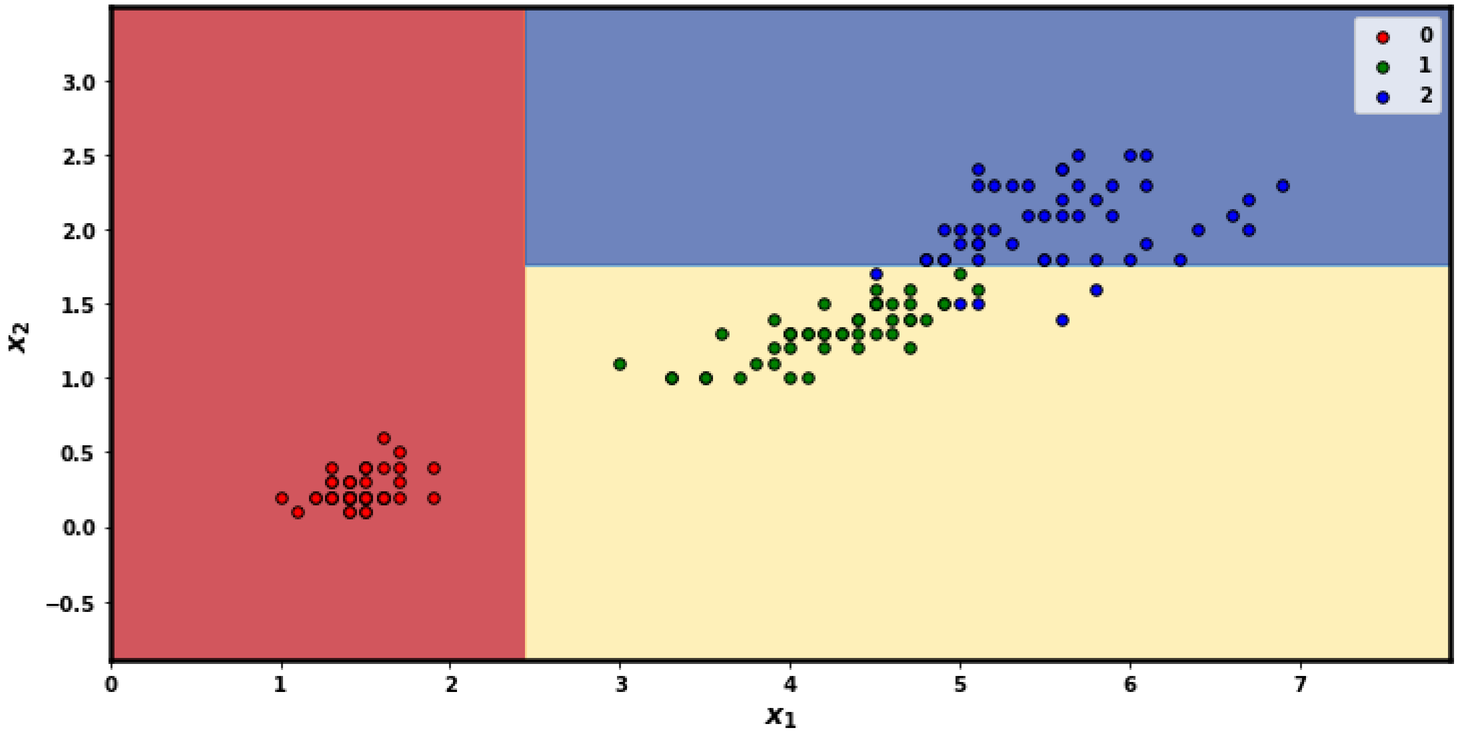
One advantage of Decision Trees over Classifiers based on linear boundaries is the inherent ability to deal with multinomial classification

- No need to create a "One versus All" binary classifier for each class

Here is a partition created by a Decision Tree on three classes

In [12]:

```
fig, ax = plt.subplots(figsize=(12,6))
_= bh.make_boundary(X_dt, y_dt, depth=2, ax=ax)
```



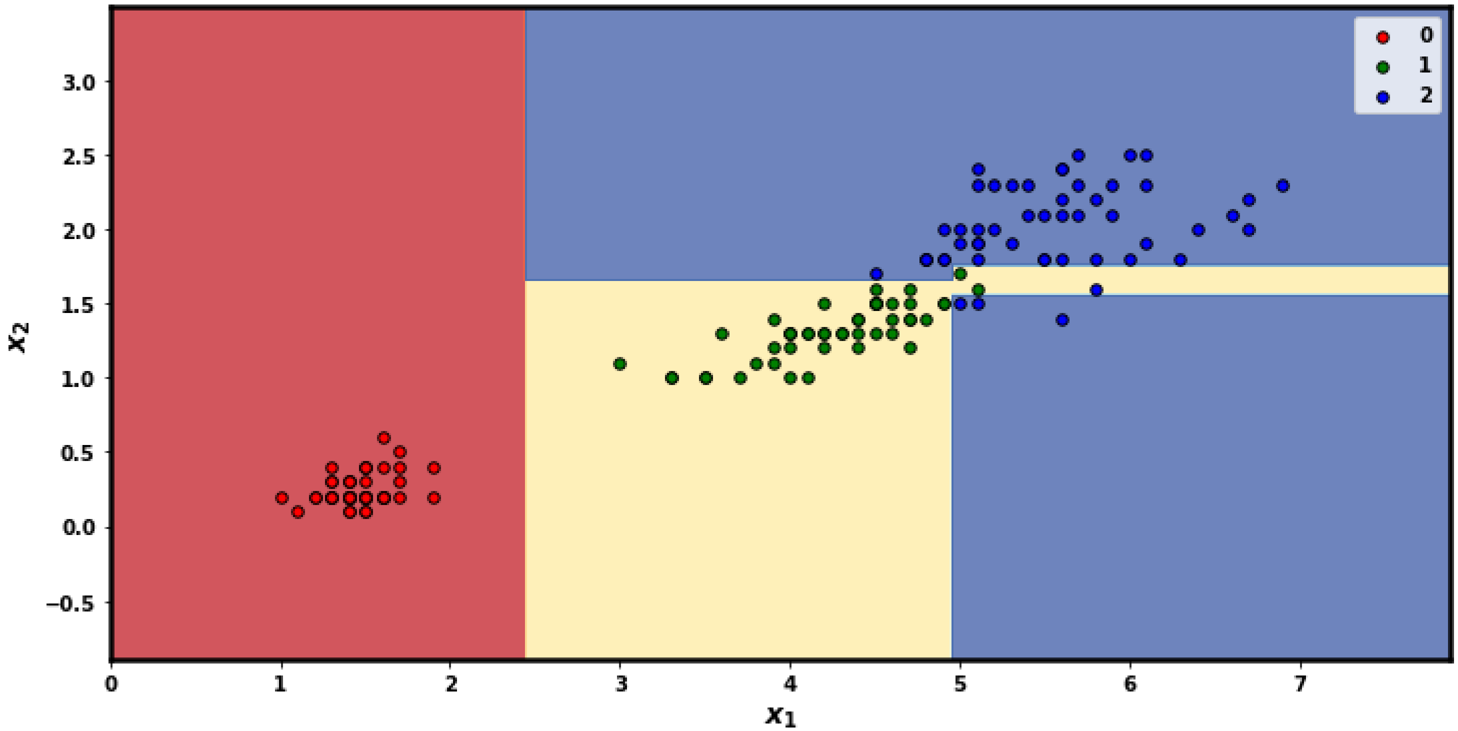
The ability to create complex boundaries comes with a potential risk

- Overfitting

With enough questions, we can exactly identify each example.

- Recall: the use of transformation that created cross-features that identified individual examples

In [13]:
```
fig, ax = plt.subplots(figsize=(12,6))
_= bh.make_boundary(X_dt, y_dt, depth=4, ax=ax)
```



In [14]:

```
print("Done")
```

```
Done
```