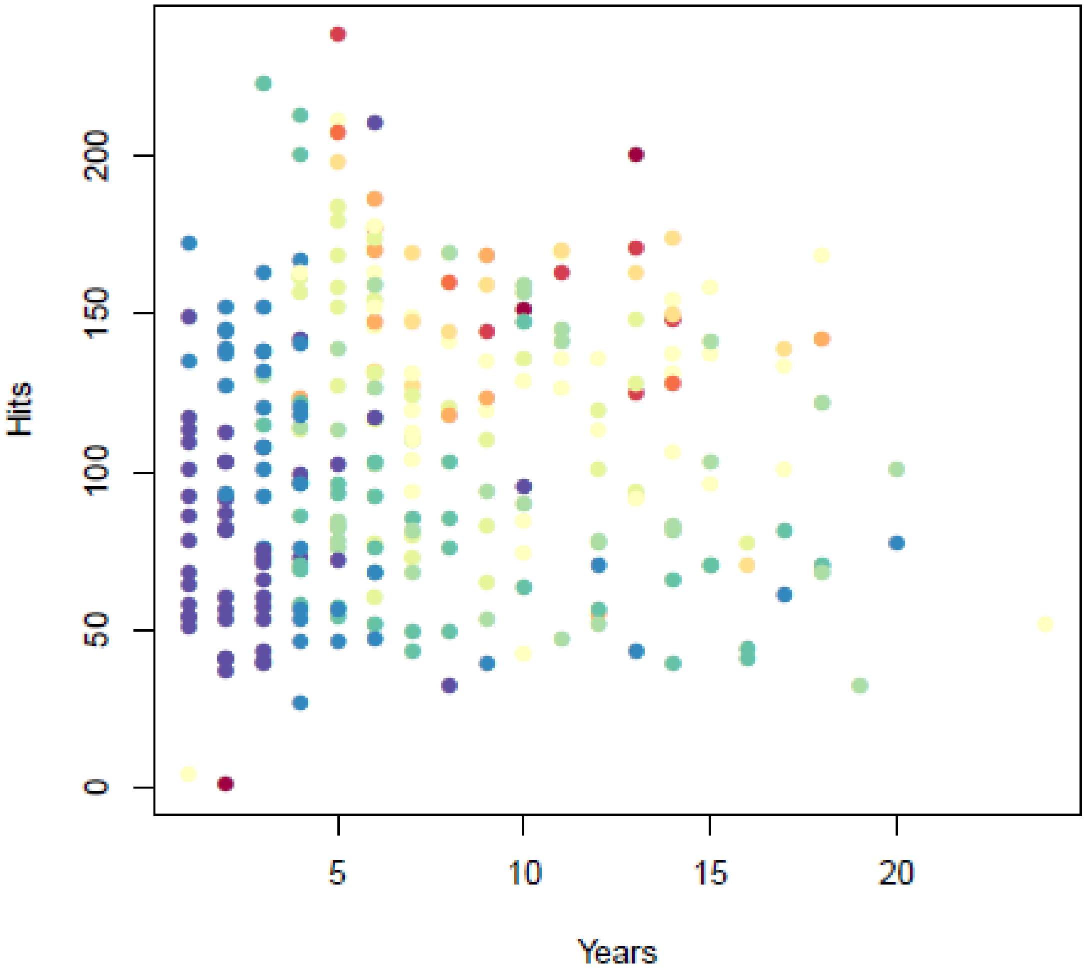
Decision Tree

Predictive Modeling II
Auburn University
Pei Xu

Example: Baseball Players' Salaries

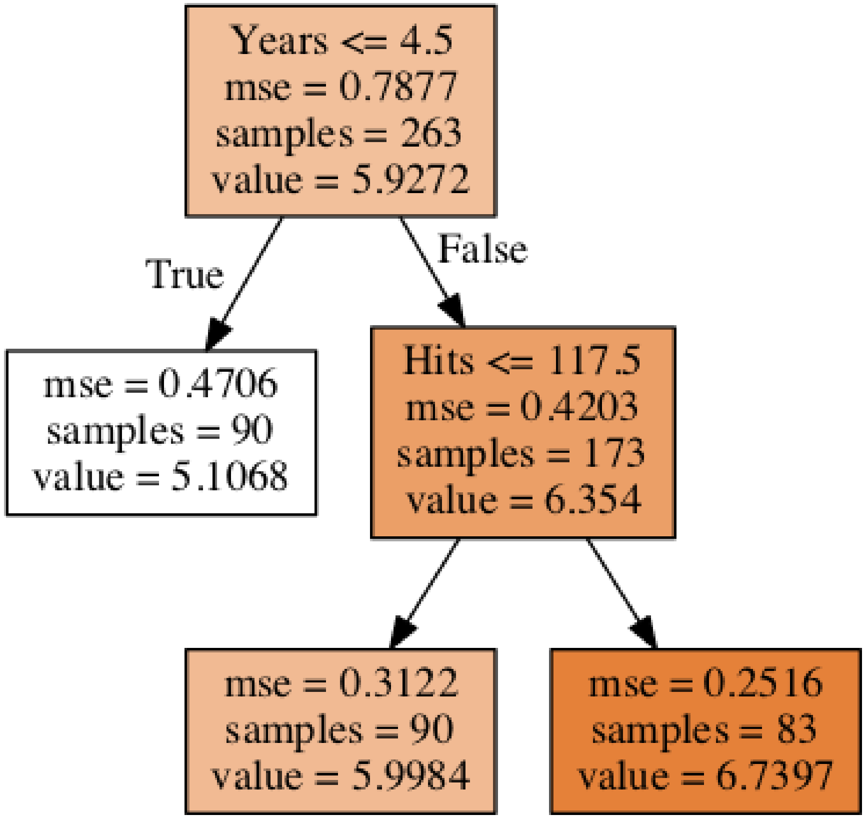
Salary is color-coded from low (blue, green) to high (yellow,red)



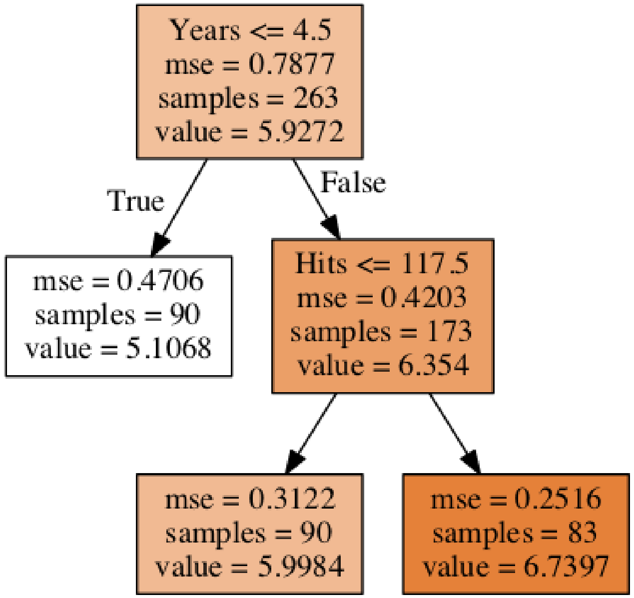
Example: Baseball Players' Salaries

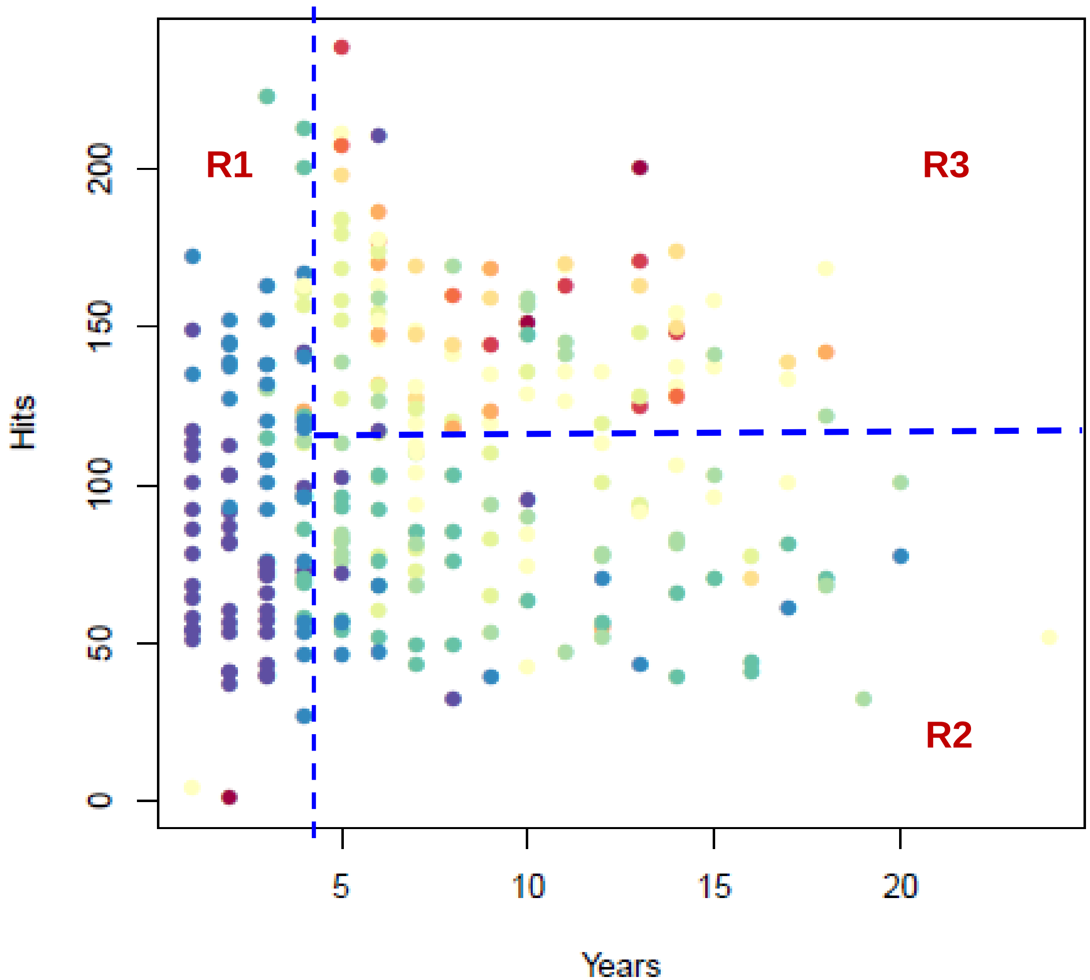
- The predicted Salary is the number in each leaf node. It is the <u>mean</u> of the response for the observations that fall there
- Note that Salary is measured in 1000s, and log-transformed
- The predicted salary for a player who played in the league for more than 4.5 years and had less than 117.5 hits last year is

$$\$1000 \times e^{6.00} = \$402,834$$



Another way of visualizing the decision tree...





Prediction using a Decision Tree

What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

For region R_j, the best prediction is simply <u>the average of all the responses</u> from our training data that fell in region R_j.

A Simple Decision Task – Fruit Classification

Tree-based Model

- A machine learning structure which is composed of a sequence of decisions to predict on an input vector of variables X=(X1 ,X2 , …, Xp)
- Tree-based methods involve *stratifying* or *segmenting* the predictor space into a number of simple regions.
- The regions are defined using a number of *splitting rules*.
- Since the set of splitting rules used to segment the predictor space can be summarized in a *tree diagram*, these approaches are known as decision-tree methods.

The Basic …

- To build a decision tree, you need a sample of data with an observable "target" (outcome or predictor) variable.
- In general, you have a "*training sample* with known values of the *target*. The training sample is used to build the new tree model.
- The model is then applied to future data for which the target has not been observed.
- Decision trees can be applied to both *regression* and *classification* problems.
  - Classification trees are used when the target is categorical.
  - Regression trees are used when the target is quantitative.

Overview: Steps to Creating a Decision Tree

1. Define a precise criterion: for selecting the variable and separation condition.
   - When the best separation has been found, the process is repeated on each node to increase the discrimination. This continues until…
2. There is a reason to stop.
   - The separation of individuals cannot be repeated further.
3. Pruning to find a parsimonious tree.

Step 1: Separation Criterion

- CHAID (Chi Square Automatic Interaction Detection)
  - For each independent variable, the group is split and combined with the target variable in a 2 X 2 contingency table.
  - From this table, a *chi-square test* of independence is calculated. A small p-value indicates significant differences or separation in Target.
  - *logworth = -ln(p-value),* where *p-value* is the p-value from the chi-square test for that variable.

Step 1: Separation Criterion

- CHAID (Chi Square Automatic Interaction Detection)
  - In fact, since the p-value for the tests are so small, a function of the p-value, the *logworth*, is used for determining the variable that gives maximum separation.
  - The *logworth* is computed for all the independent variables in the data set and the one with the largest logworth is selected for the first split.
  - CHAID is the simplest algorithm for splitting trees.

Step 2: Stopping

- Stopping Occurs When…
  - Depth of tree has reached a fixed limit, or,
  - Number of leaves has reached a fixed maximum, or,
  - A minimum number is contained in each node, or,
  - Further division of a node creates a child with too few observations, or,
  - Quality of the tree is adequate, or,
  - Quality of tree is no longer increasing significantly.

Step 3: Pruning

- As a general rule, there should be at least 20 to 30 individuals per node.
- Branches that lead to leaves with too few observations should be pruned.
- A good algorithm creates a tree of maximum size, then prunes according to a validation sample.

Pre-Pruning (Early Stopping Rule)

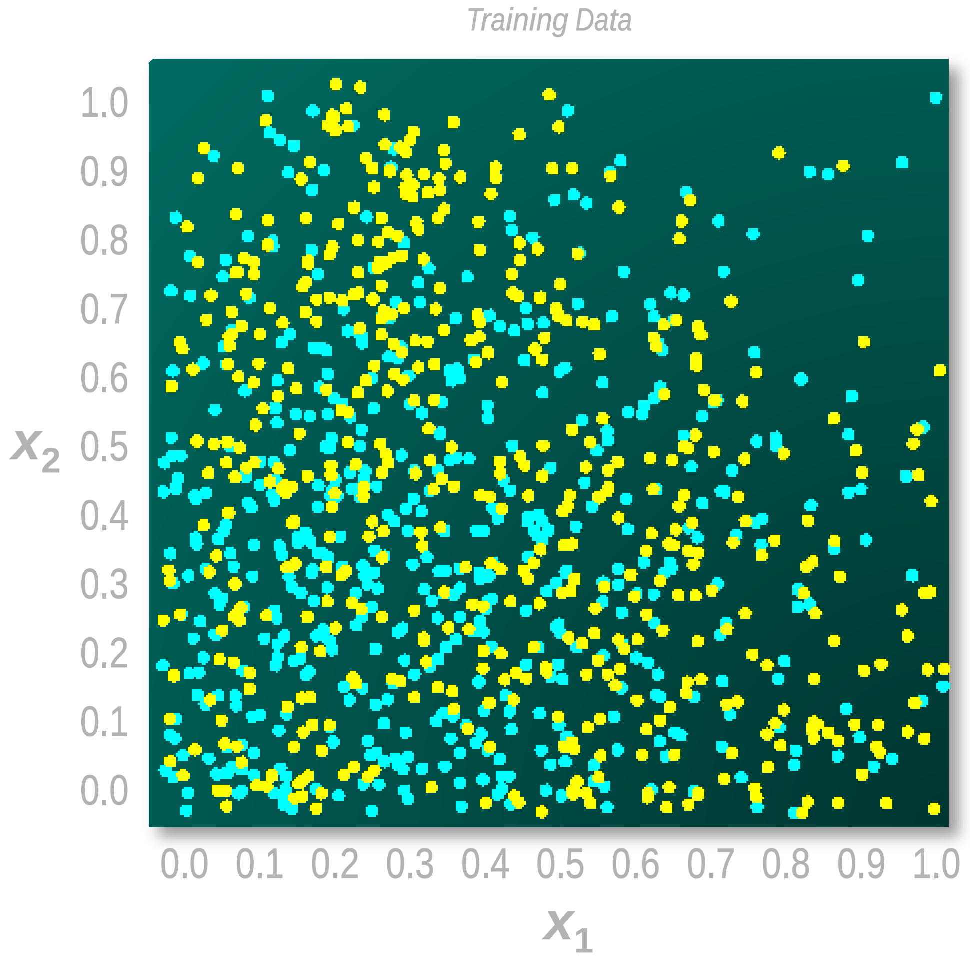
- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
  - Stop if estimated generalization error falls below certain threshold

Post-pruning

- Grow decision tree to its entirety
- Subtree replacement
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node
  - Class label of leaf node is determined from majority class of instances in the sub-tree
- Subtree raising
  - Replace subtree with most frequently used branch

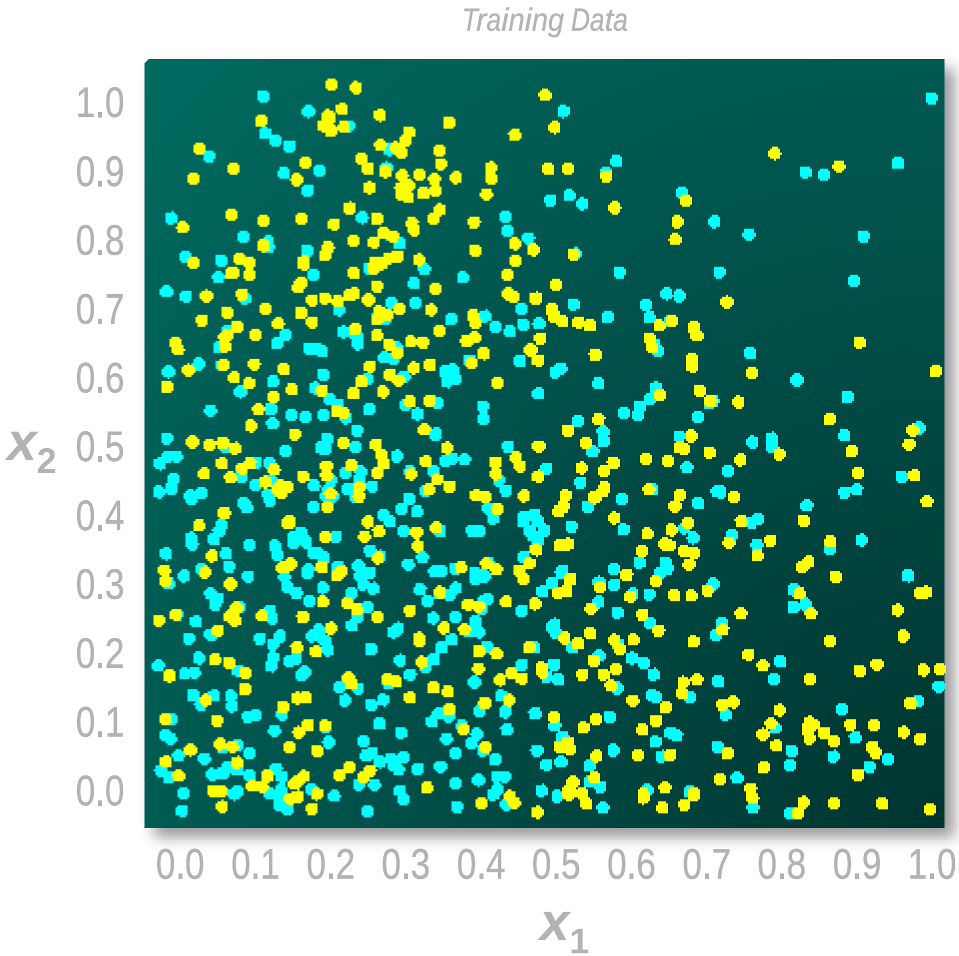
Simple Prediction Illustration

Predict dot color for each x_1 and x_2.

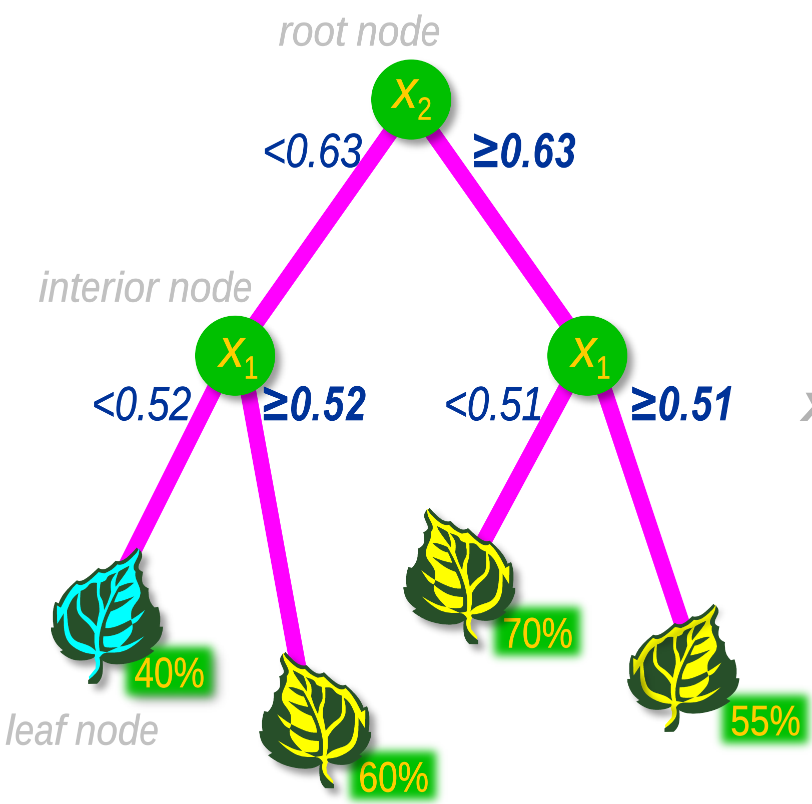


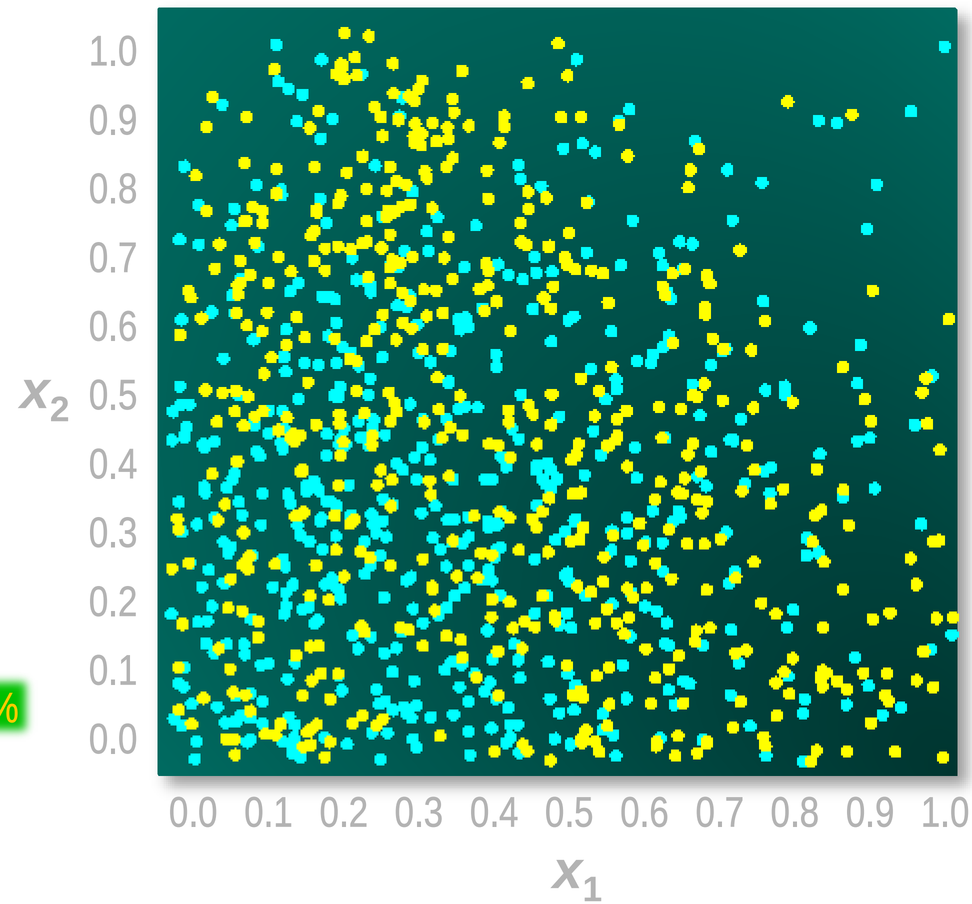
Simple Prediction Illustration

Predict dot color for each x_1 and x_2.

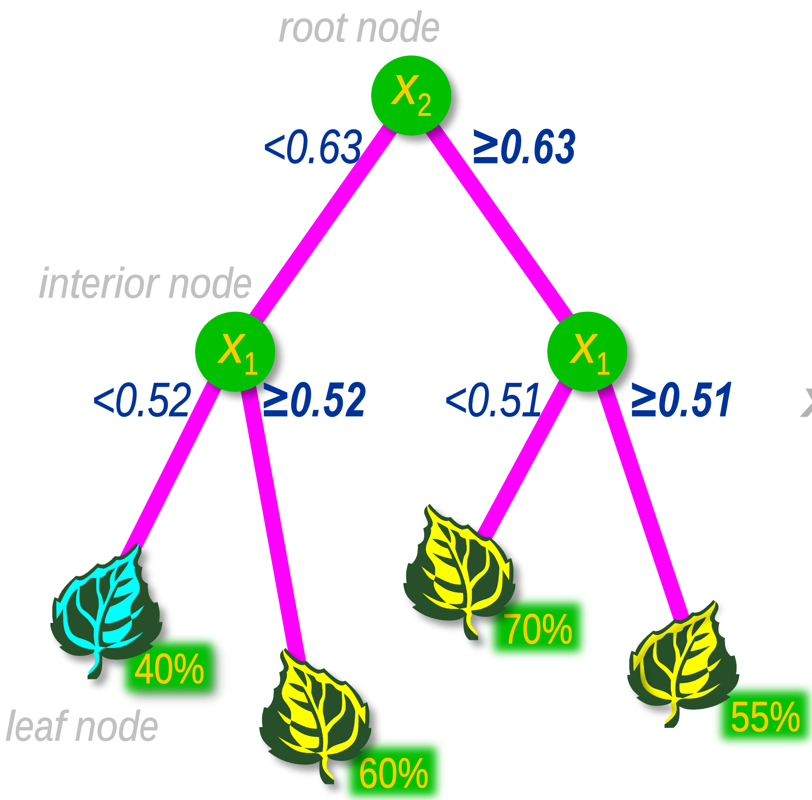


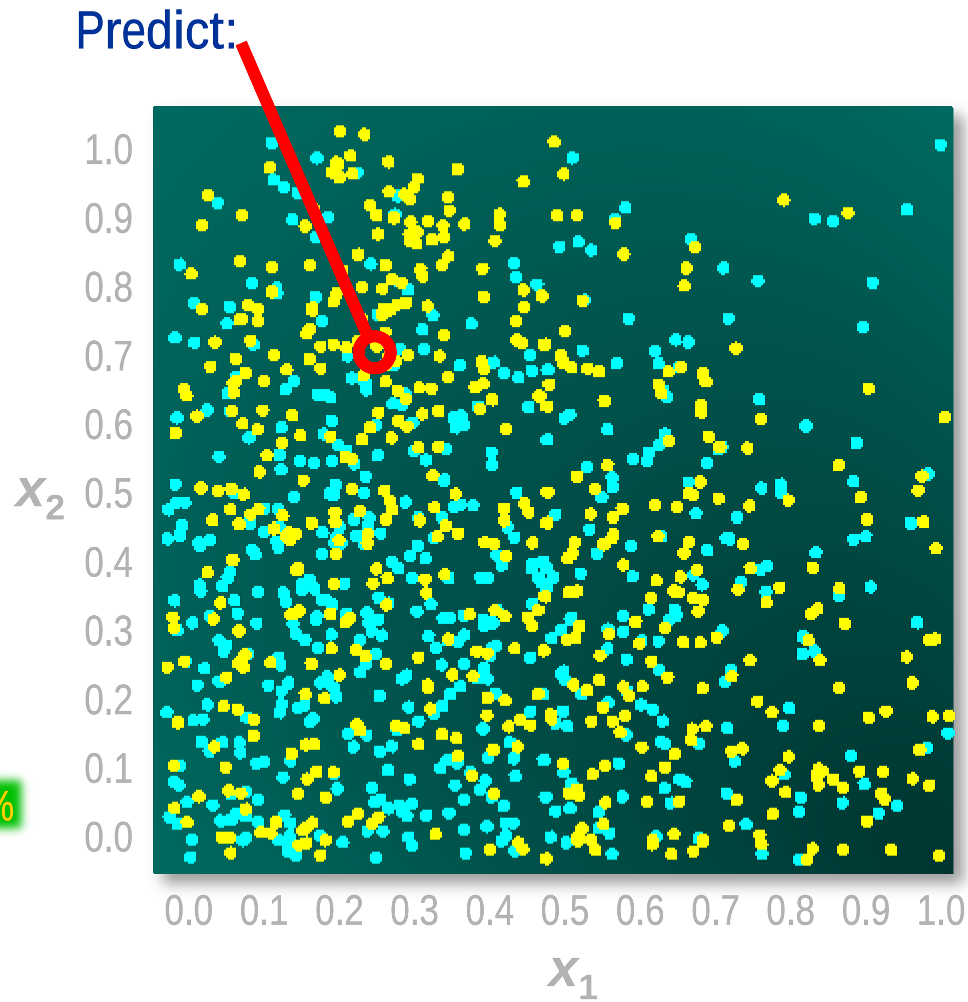
Decision Tree Prediction Rules



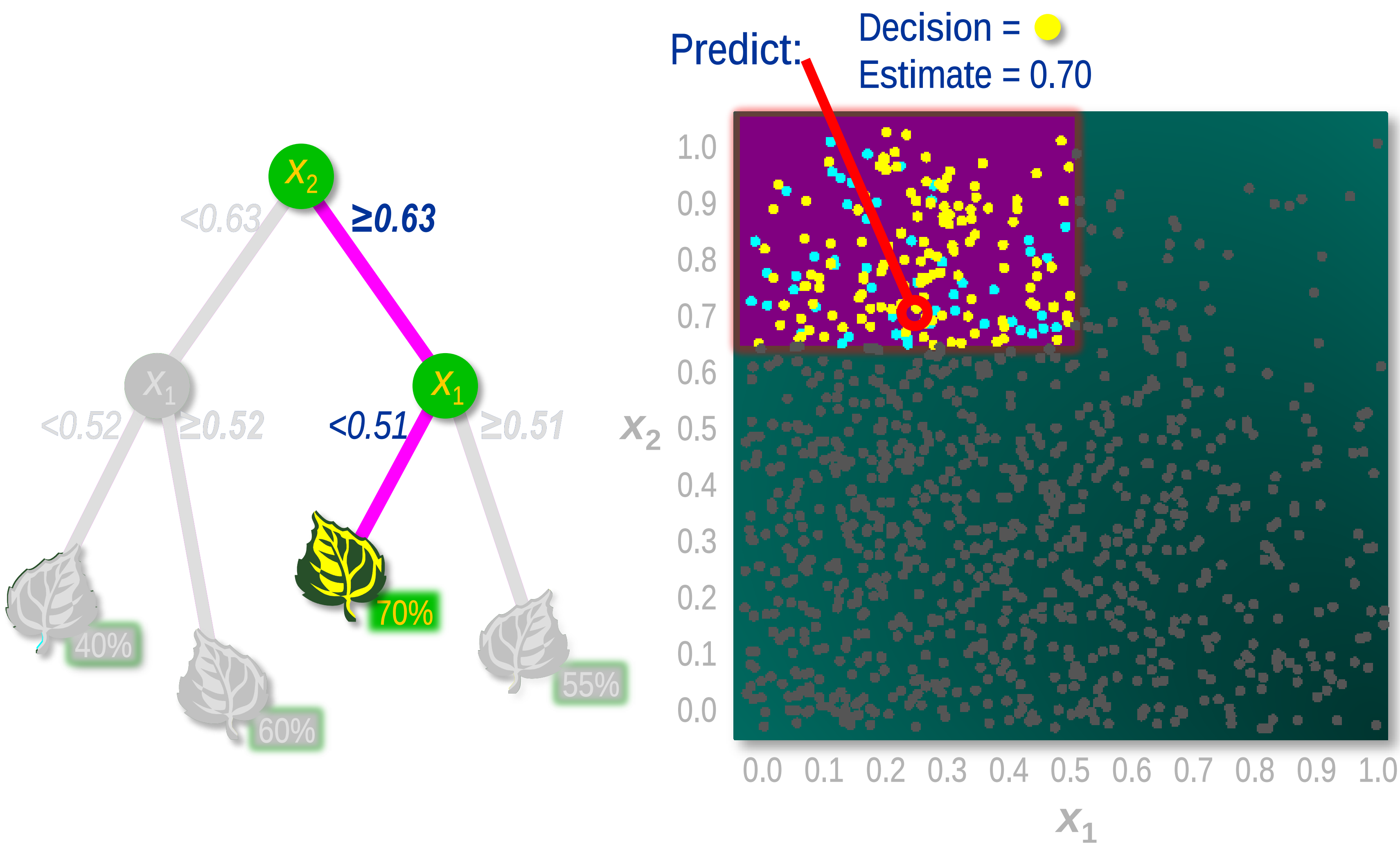


Decision Tree Prediction Rules

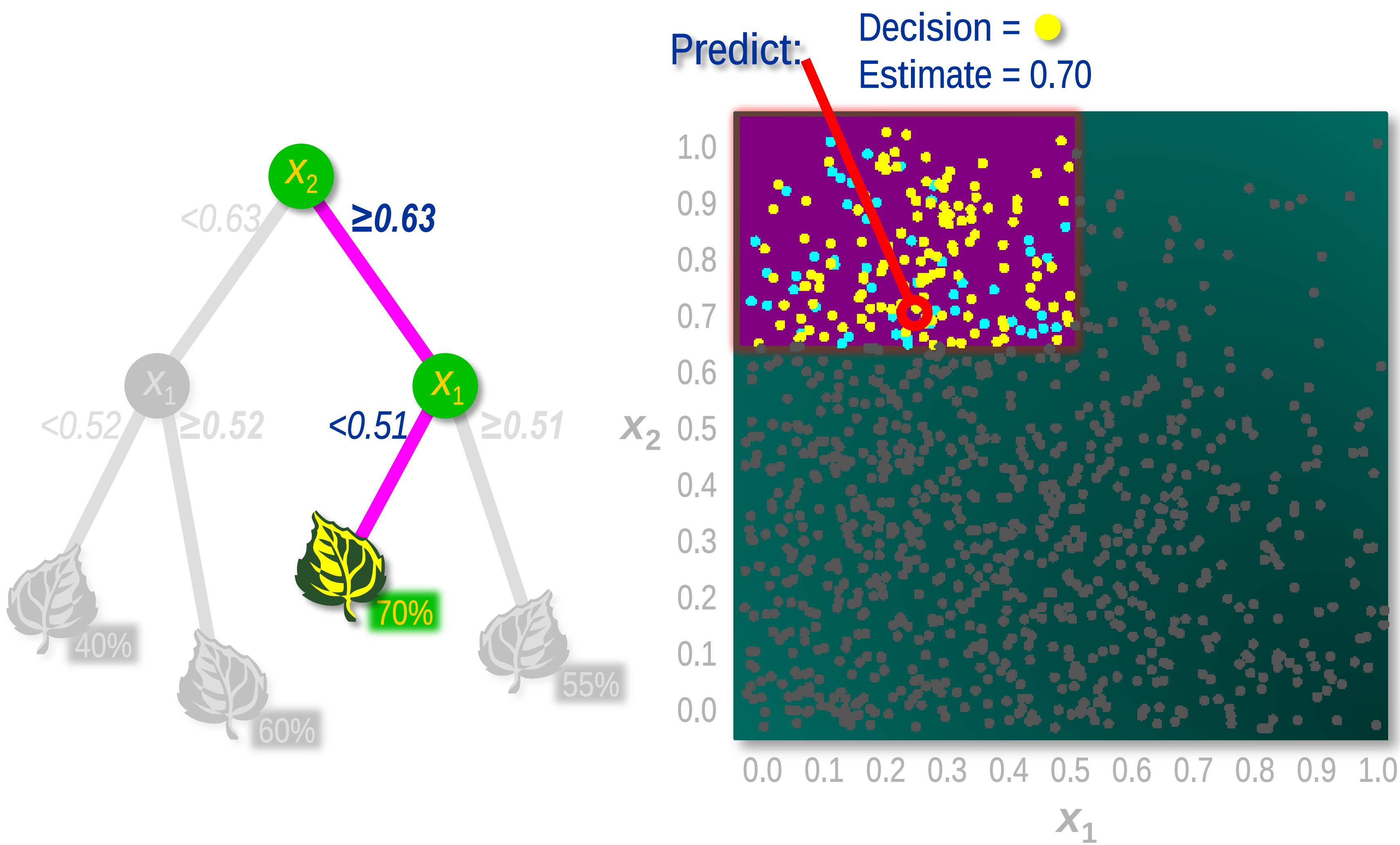




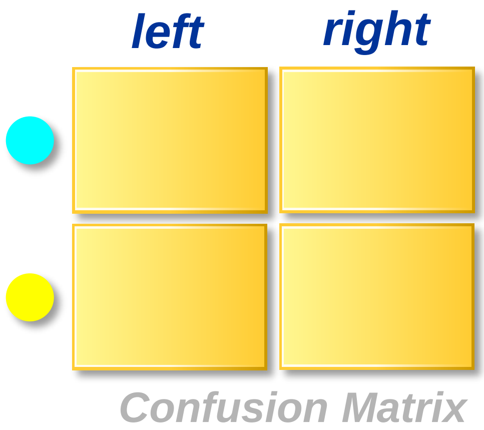
Decision Tree Prediction Rules

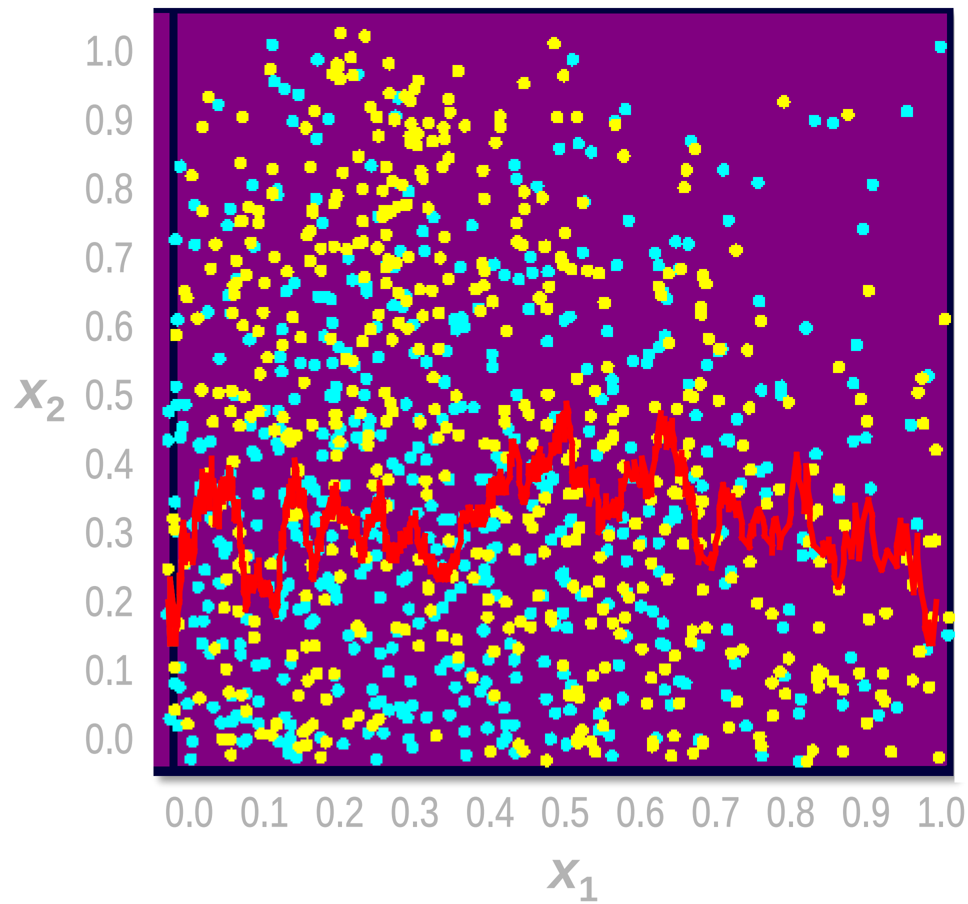


Decision Tree Prediction Rules



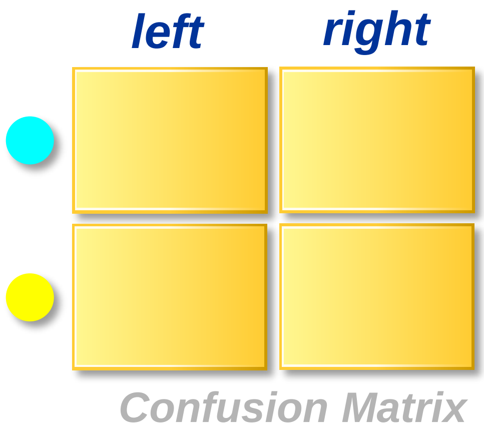
Decision Tree Split Search

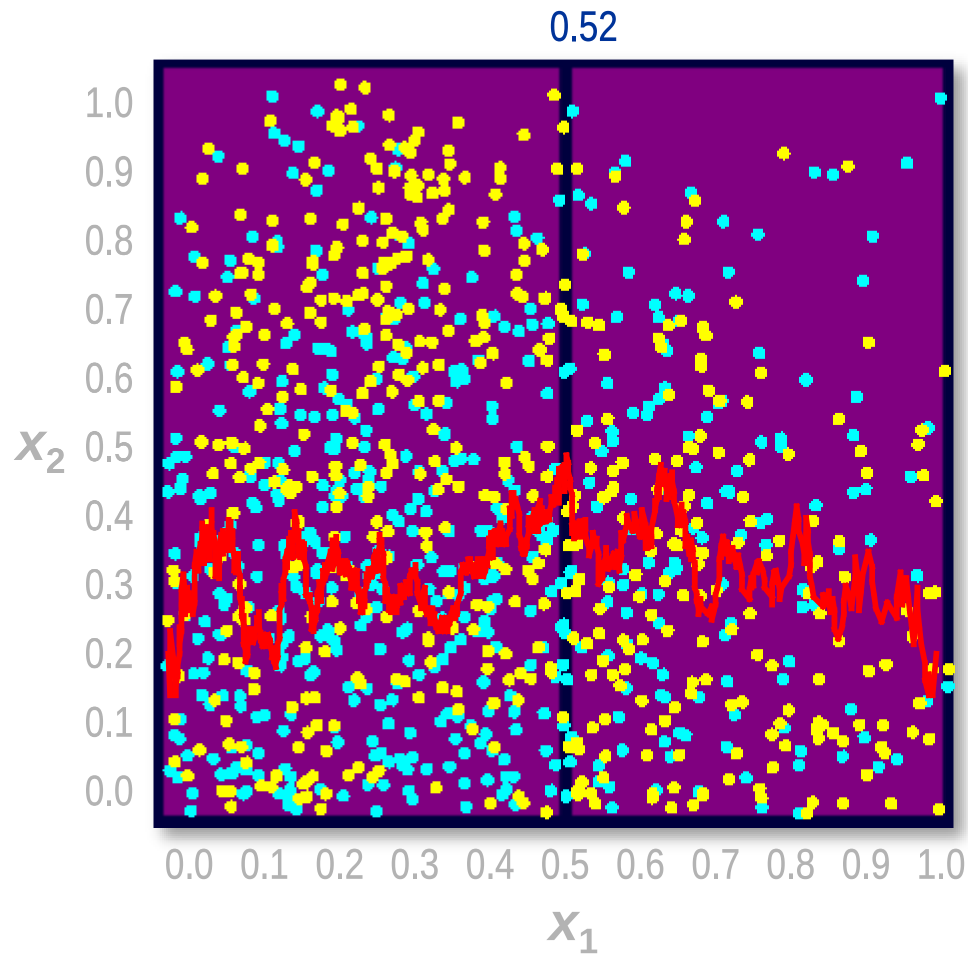




Calculate the *logworth* of every partition on input x_1.

Decision Tree Split Search



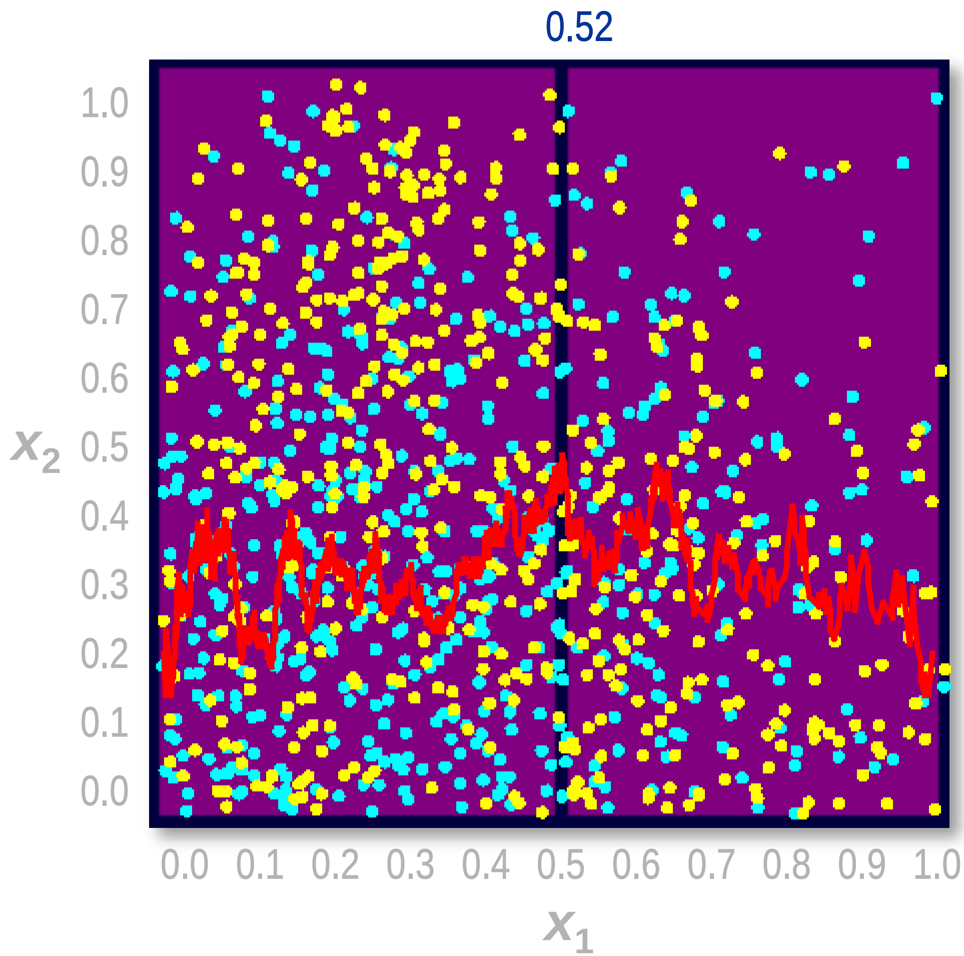


Calculate the *logworth* of every partition on input x_1.

Decision Tree Split Search

	left	*right*
cyan	53%	42%
yellow	47%	58%

max
logworth(x_1)
0.95

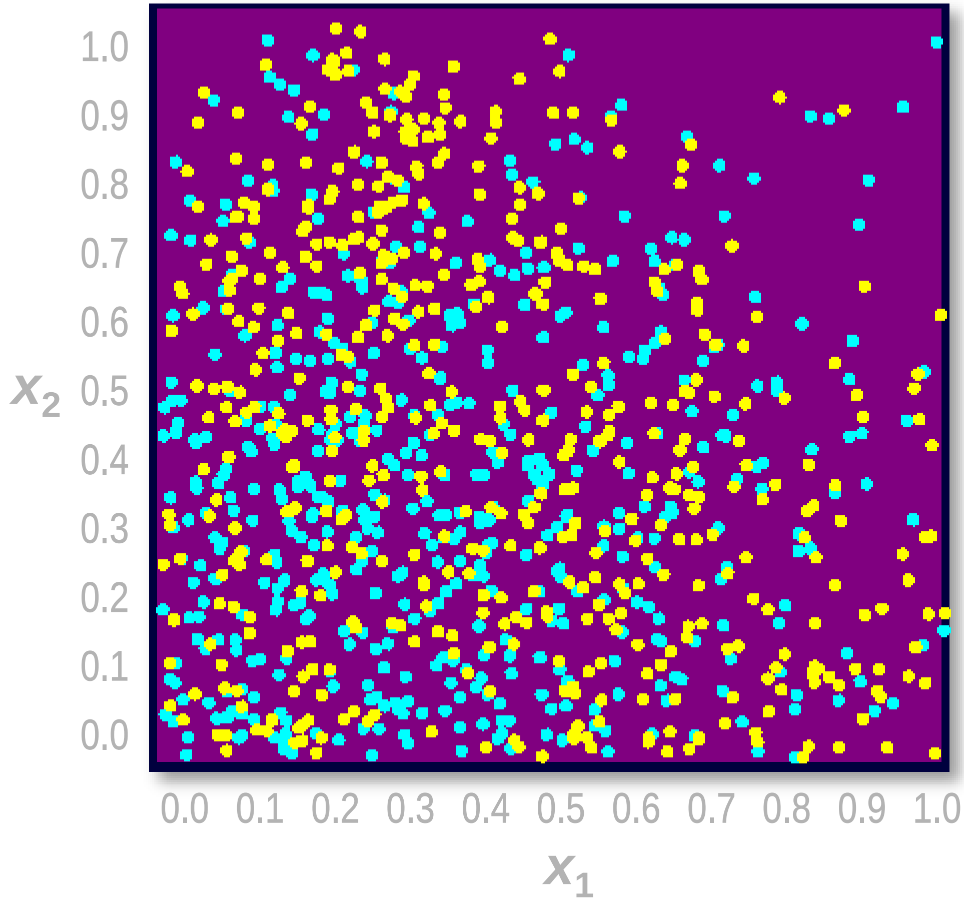


Select the partition with the maximum *logworth*.

Decision Tree Split Search

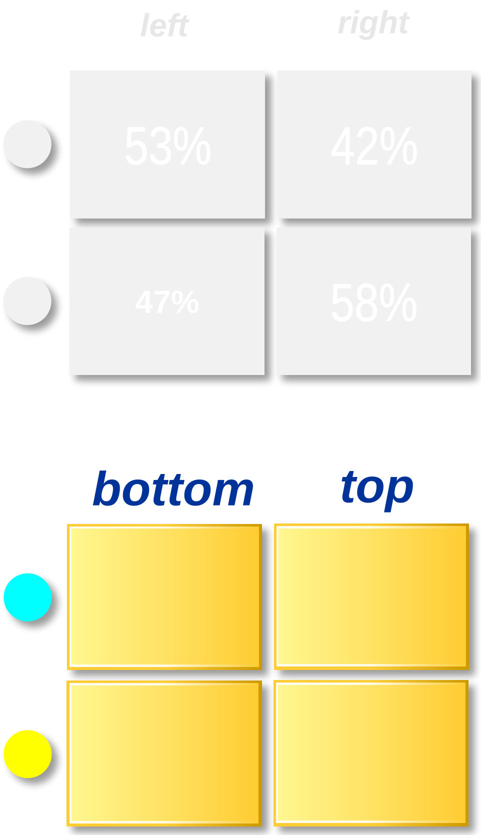
	left	right
	53%	42%
	47%	58%

max
logworth(x_1)
0.95

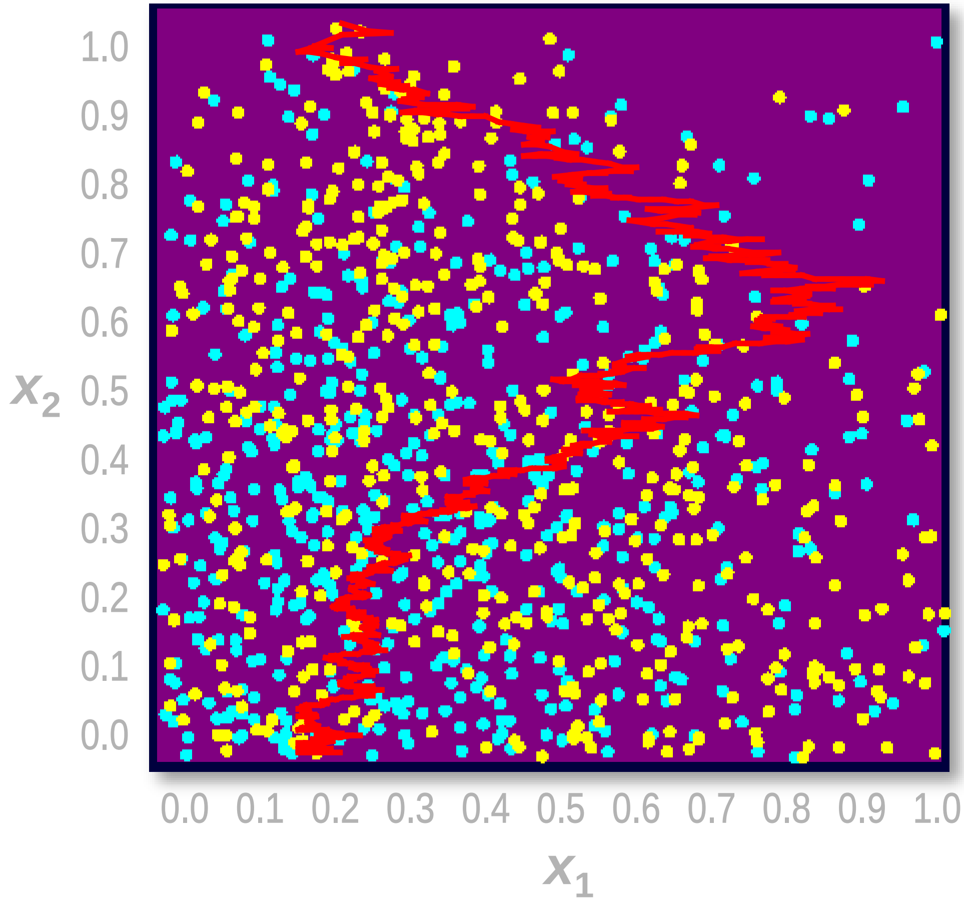


Repeat for input x_2.

Decision Tree Split Search



max
logworth(x_1)
0.95



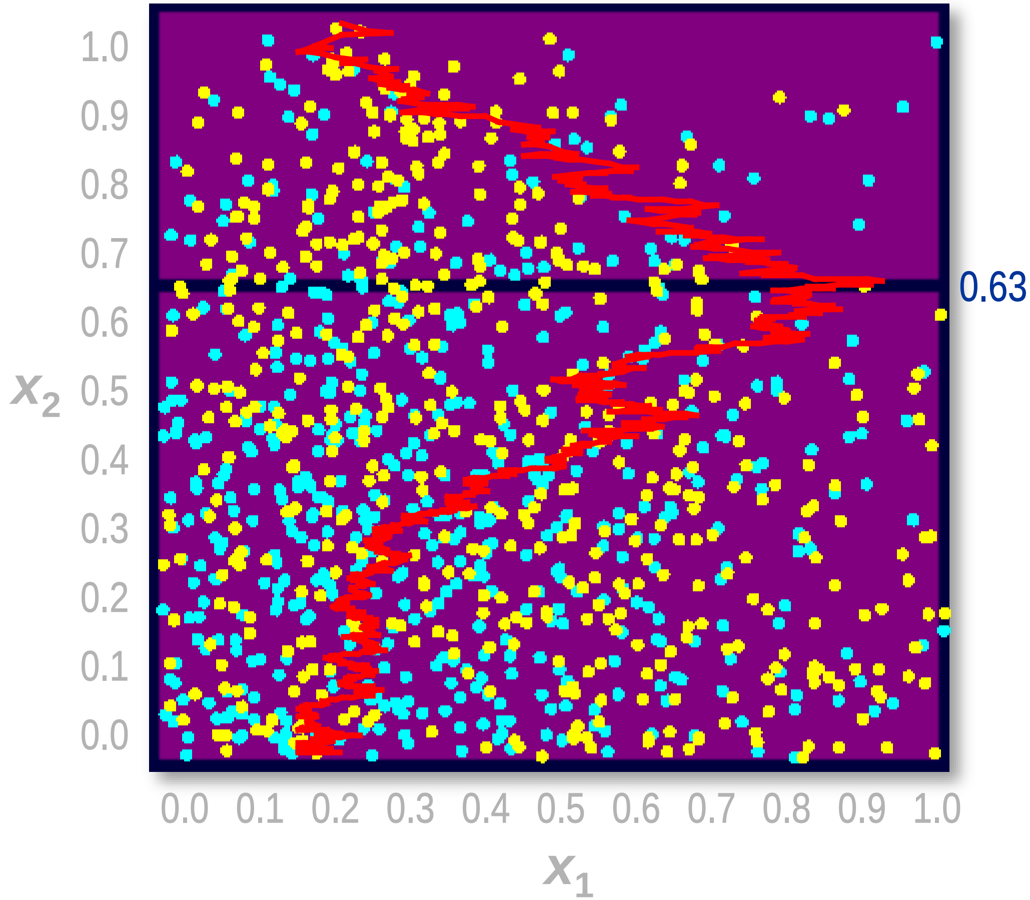
Decision Tree Split Search

	left	right
	53%	42%
	47%	58%

max
logworth(x_1)
0.95

	bottom	top
	54%	35%
	46%	65%

max
logworth(x_2)
4.92



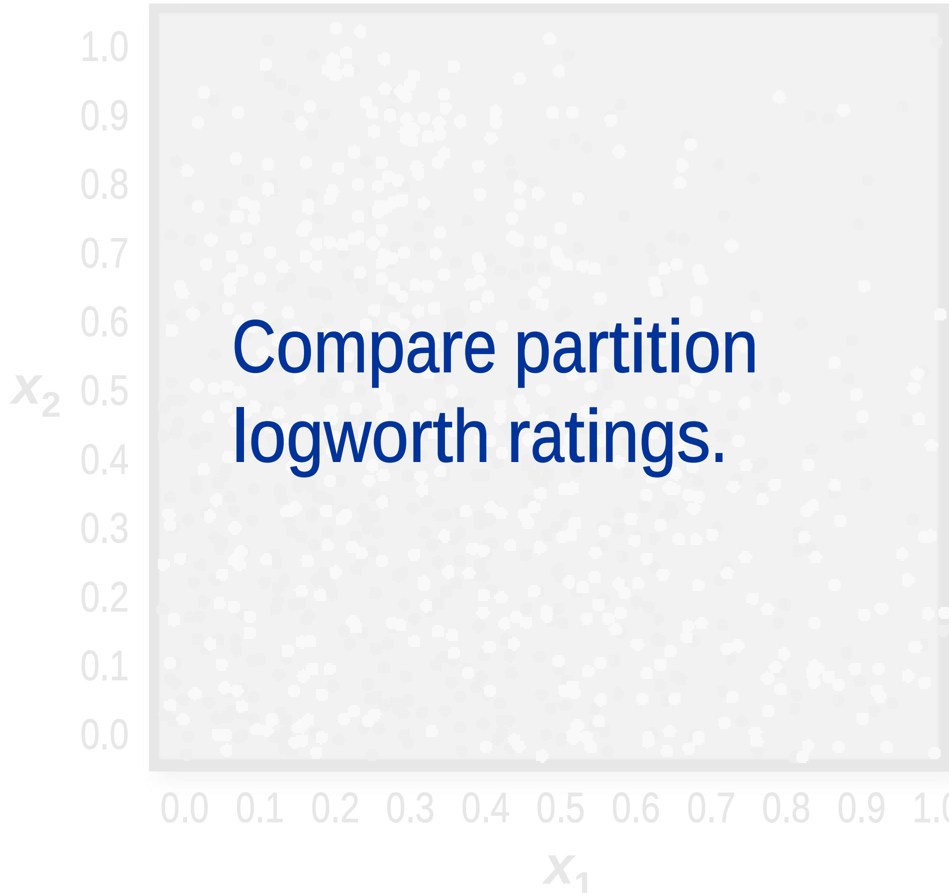
Decision Tree Split Search

	left	*right*
cyan	53%	42%
yellow	47%	58%

max logworth(x_1) 0.95

	bottom	*top*
cyan	54%	35%
yellow	46%	65%

max logworth(x_2) 4.92



Compare partition logworth ratings.

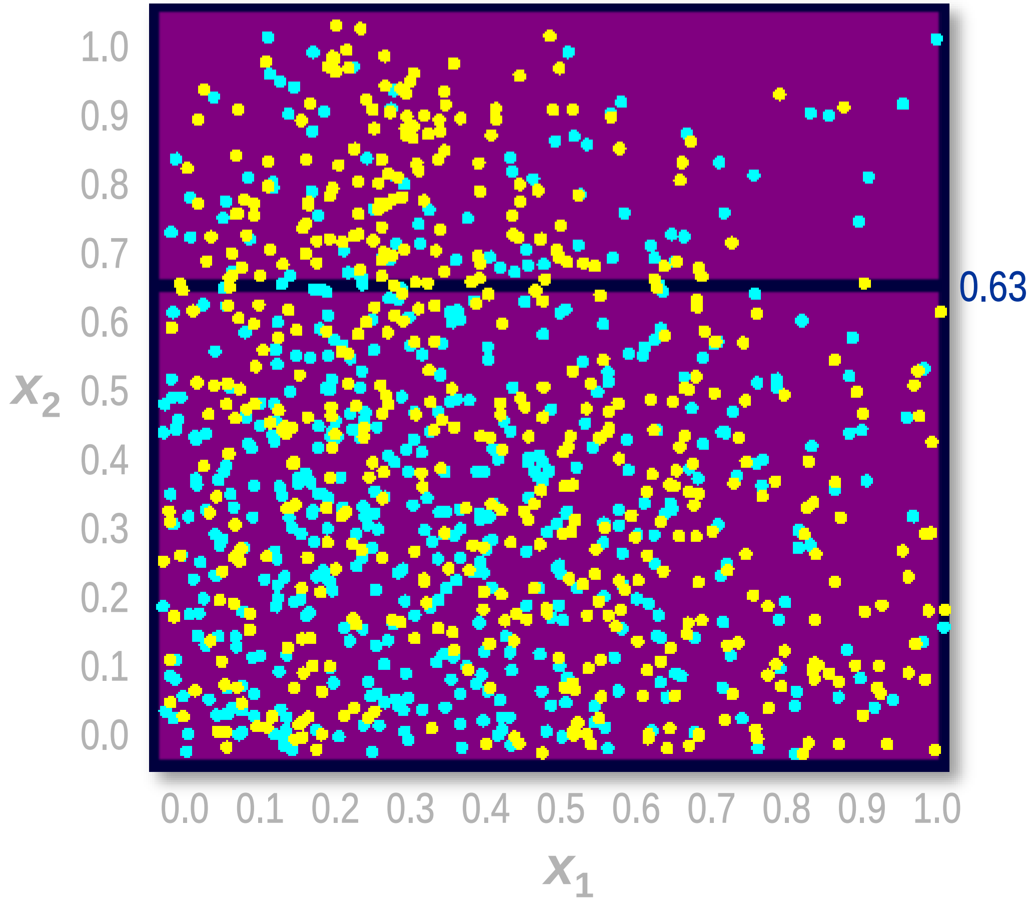
Decision Tree Split Search

	left	*right*
	53%	42%
	47%	58%

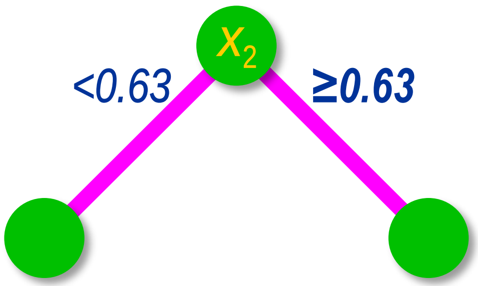
max logworth(x_1) 0.95

	bottom	***top***
cyan	54%	35%
yellow	46%	65%

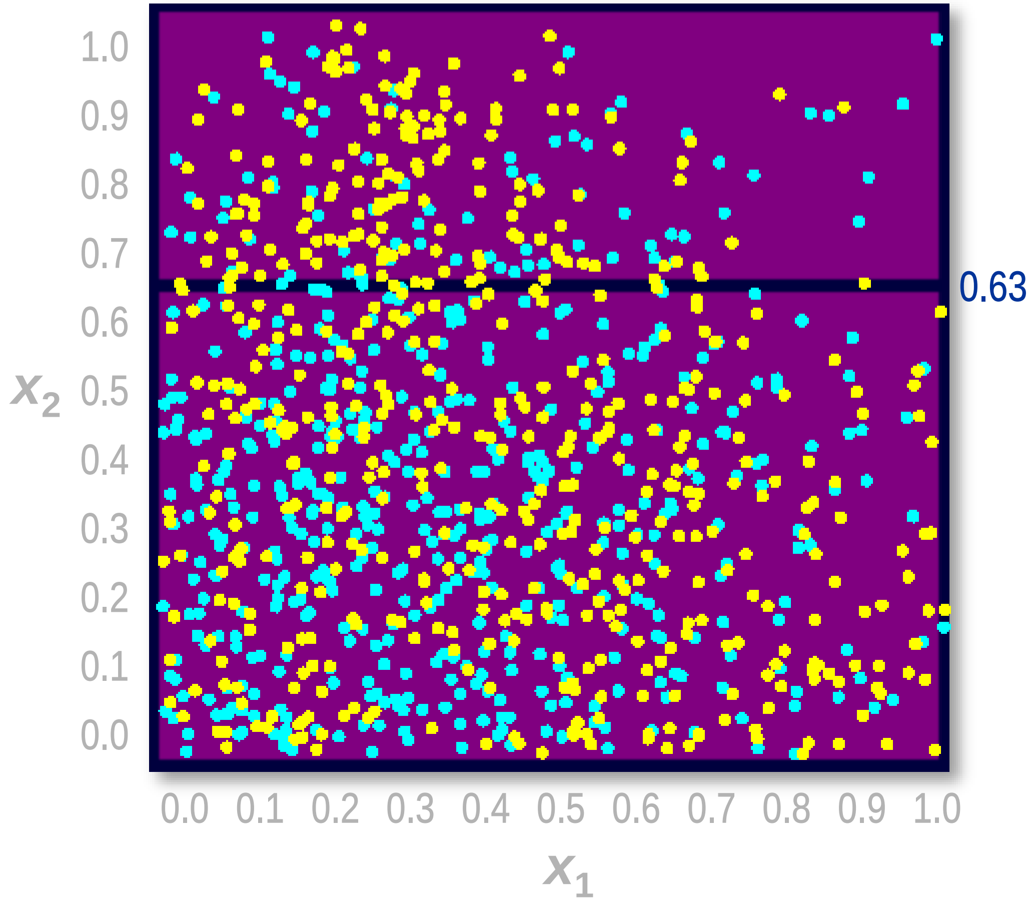
max logworth(x_2) 4.92



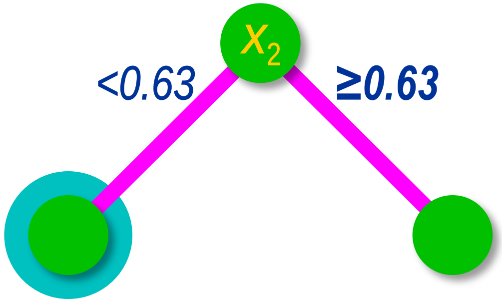
Decision Tree Split Search



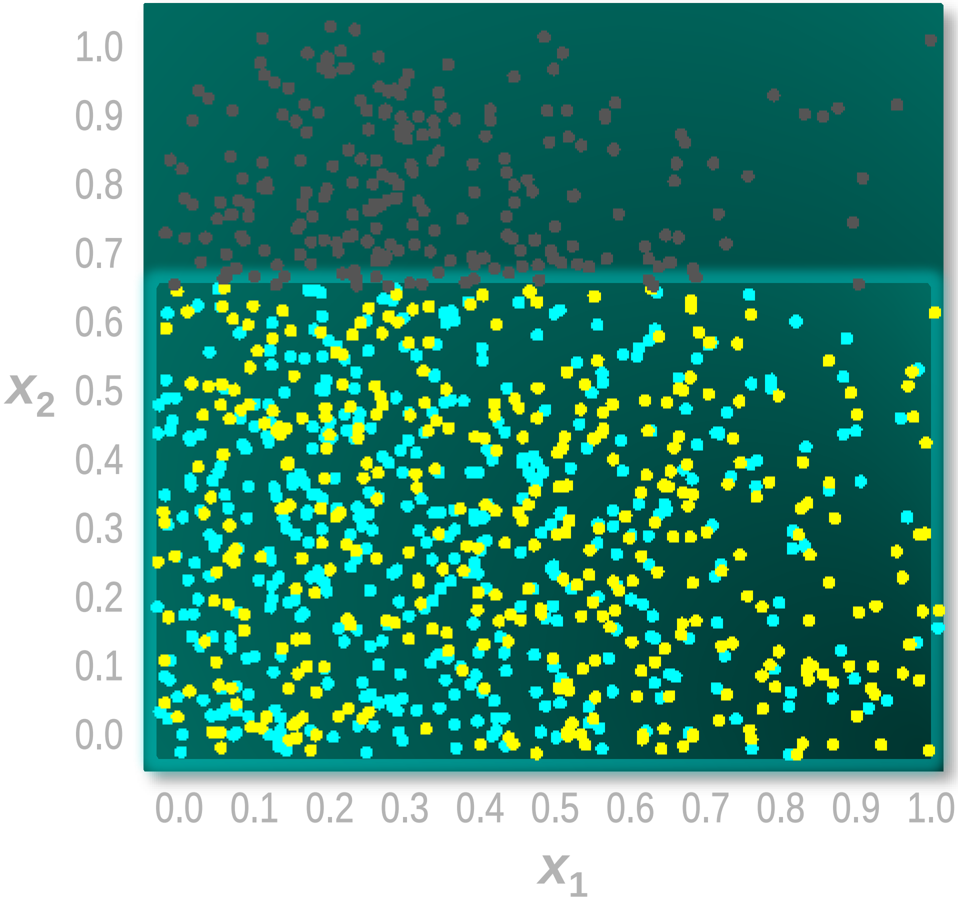
Create a partition rule from the best partition across all inputs.



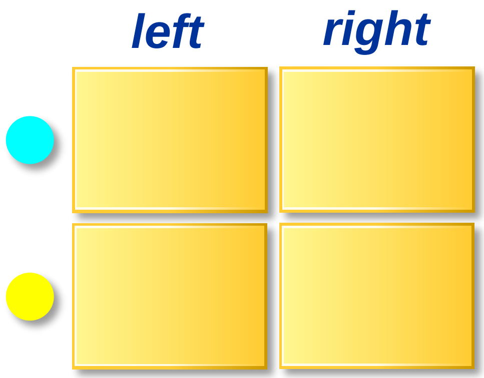
Decision Tree Split Search

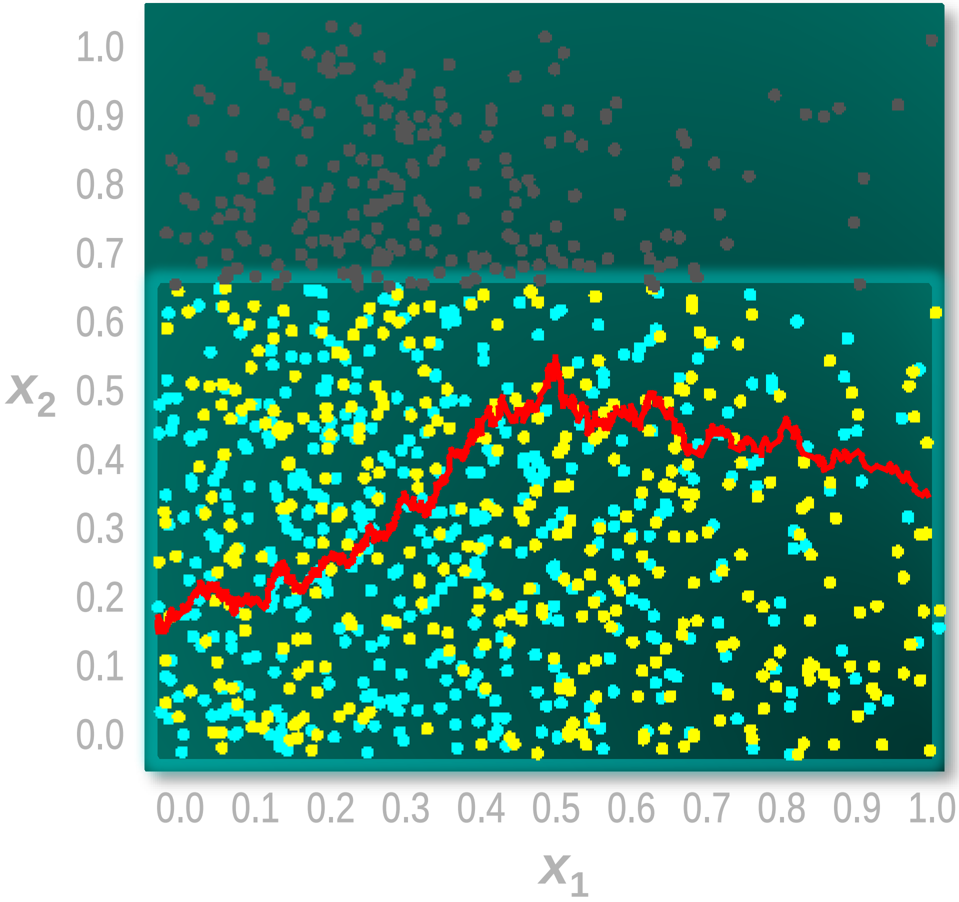


Repeat the process in each subset.



Decision Tree Split Search

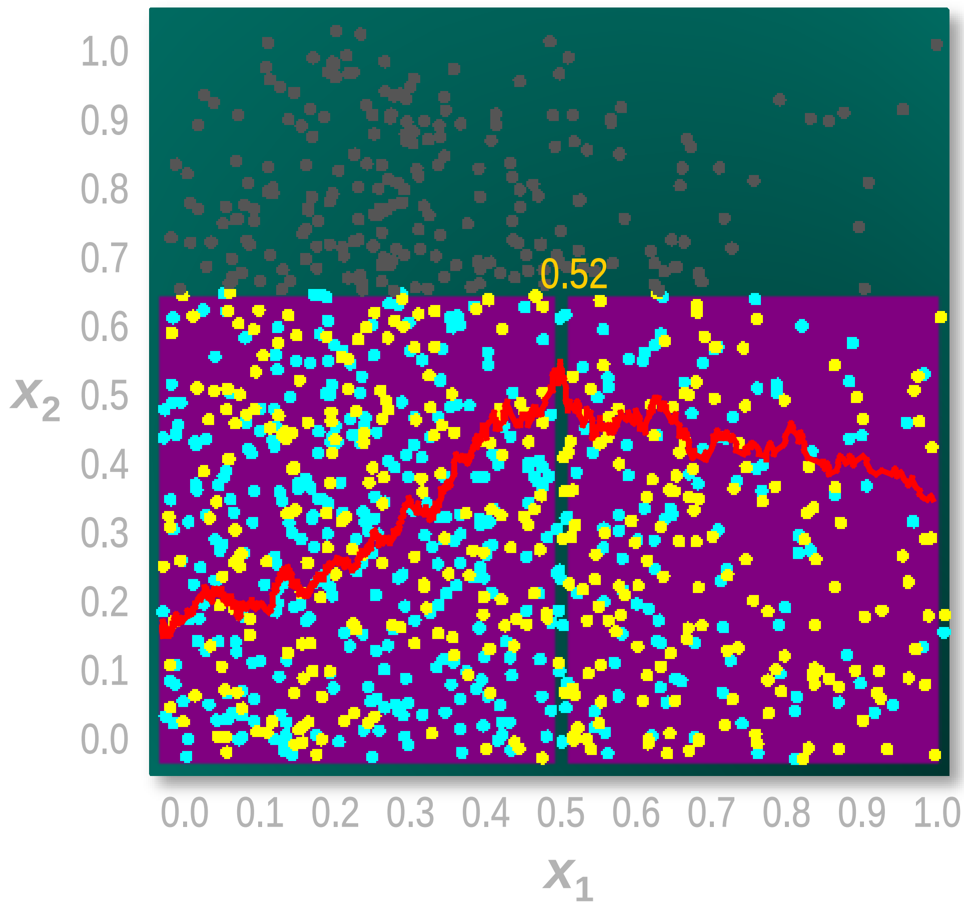




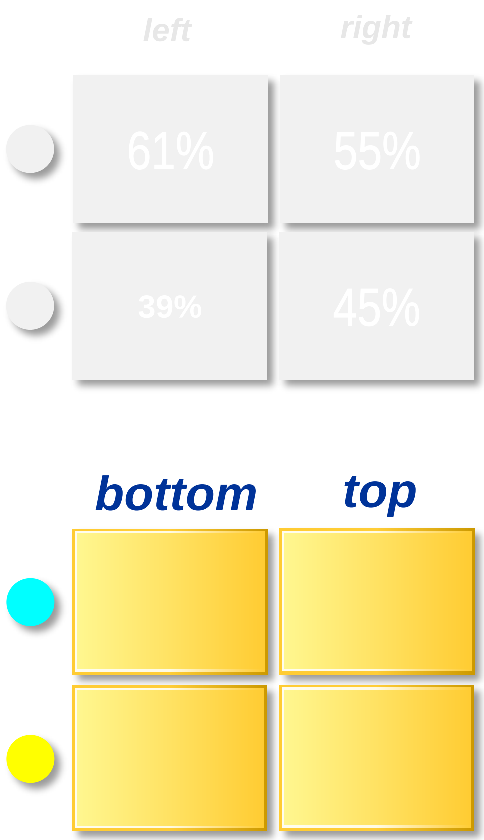
Decision Tree Split Search

	left	right
	61%	55%
	39%	45%

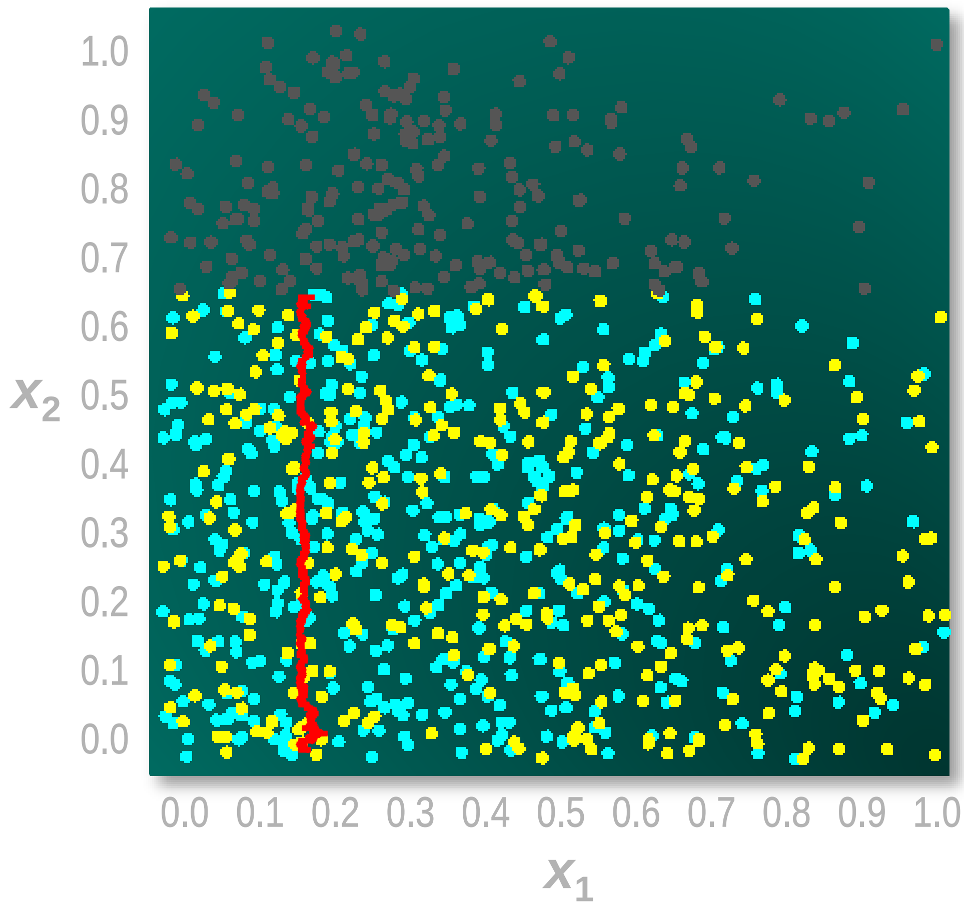
max
logworth(x_1)
5.72



Decision Tree Split Search



max
logworth(x_1)
5.72



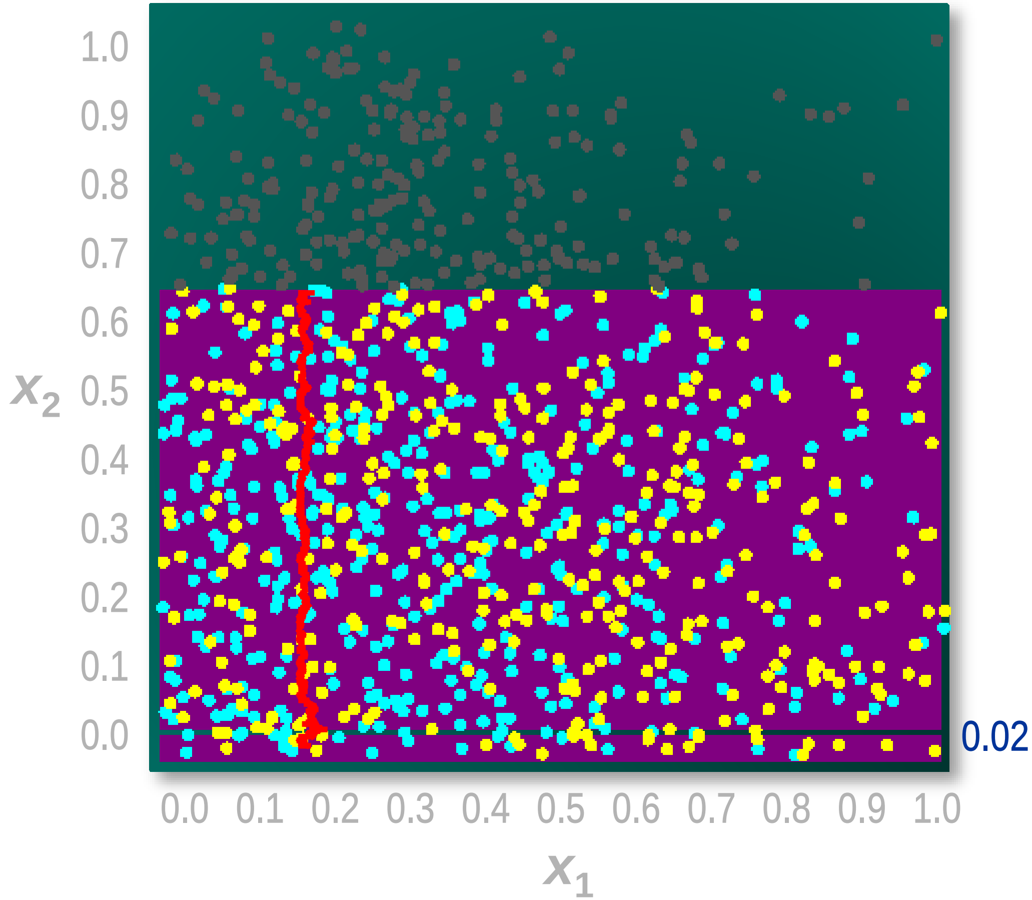
Decision Tree Split Search

	left	right
	61%	55%
	39%	45%

max
logworth(x_1)
5.72

	bottom	top
	38%	55%
	62%	45%

max
logworth(x_2)
-2.01



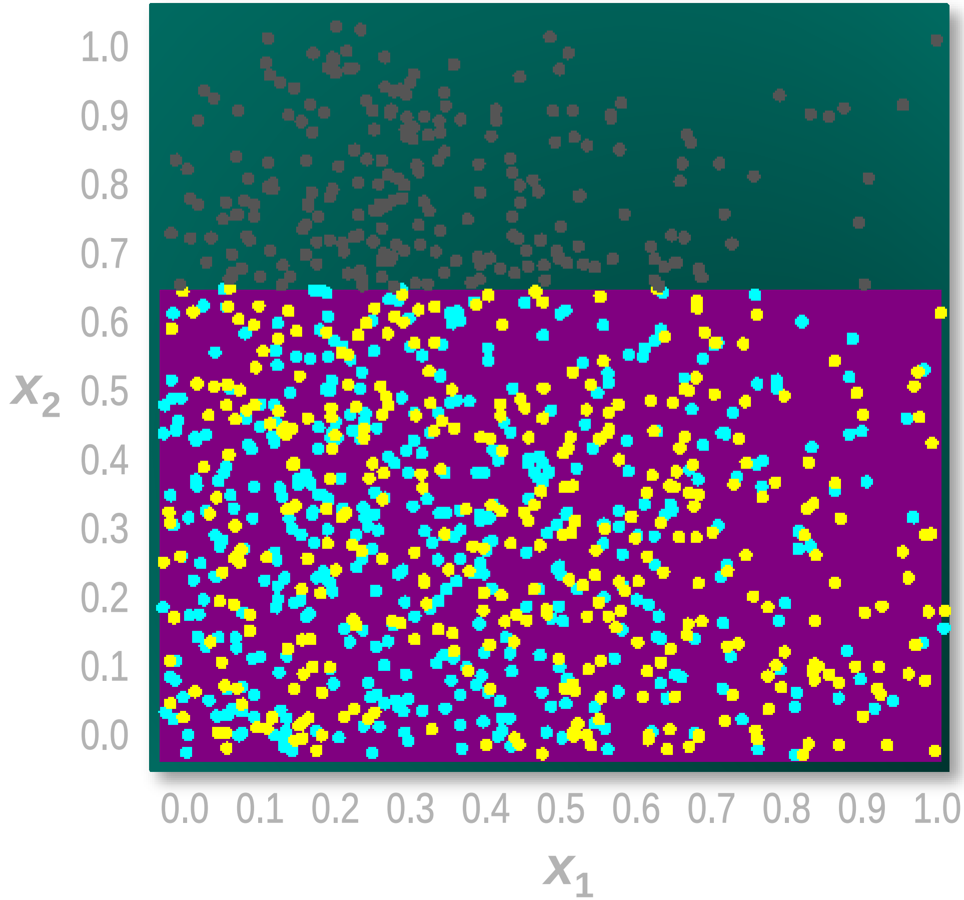
Decision Tree Split Search

	left	*right*
	61%	55%
	39%	45%

max logworth(x_1)
5.72

	bottom	*top*
	38%	55%
	62%	45%

max logworth(x_2)
-2.01



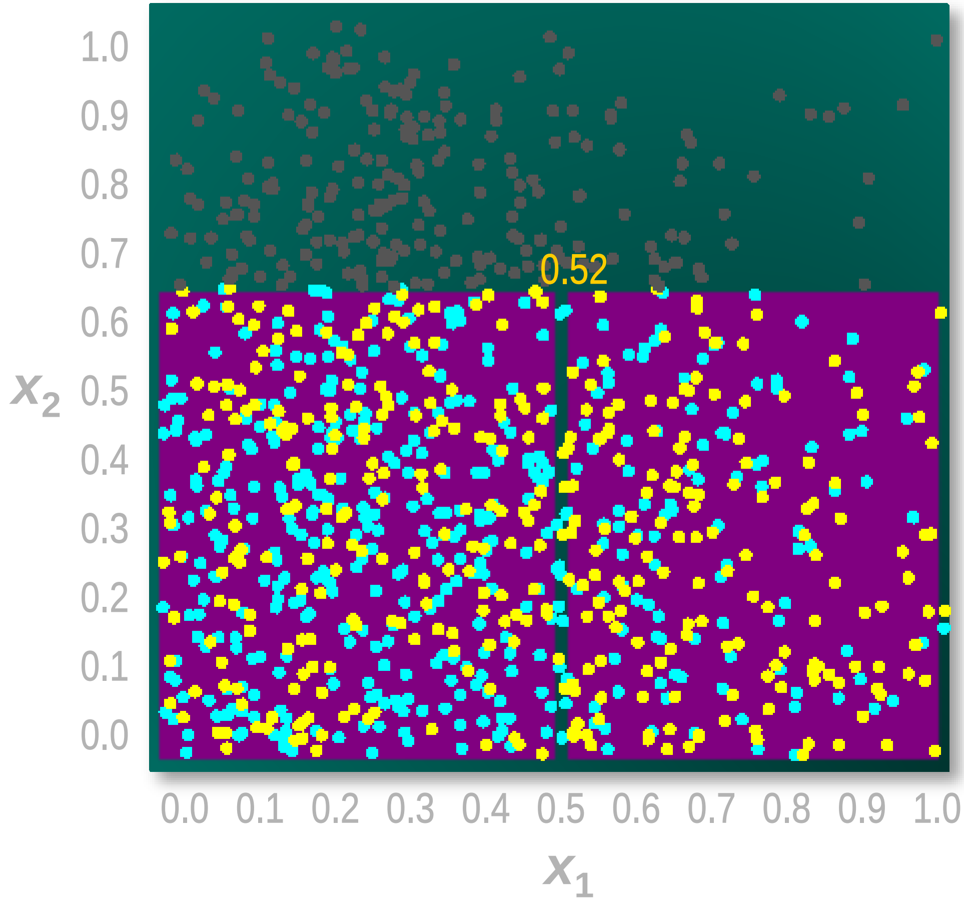
Decision Tree Split Search

	left	right
(cyan)	61%	55%
(yellow)	39%	45%

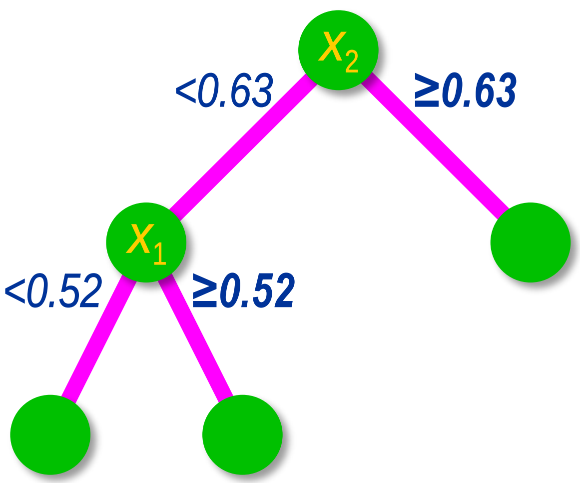
max
logworth(x_1)
5.72

	bottom	top
	38%	55%
	62%	45%

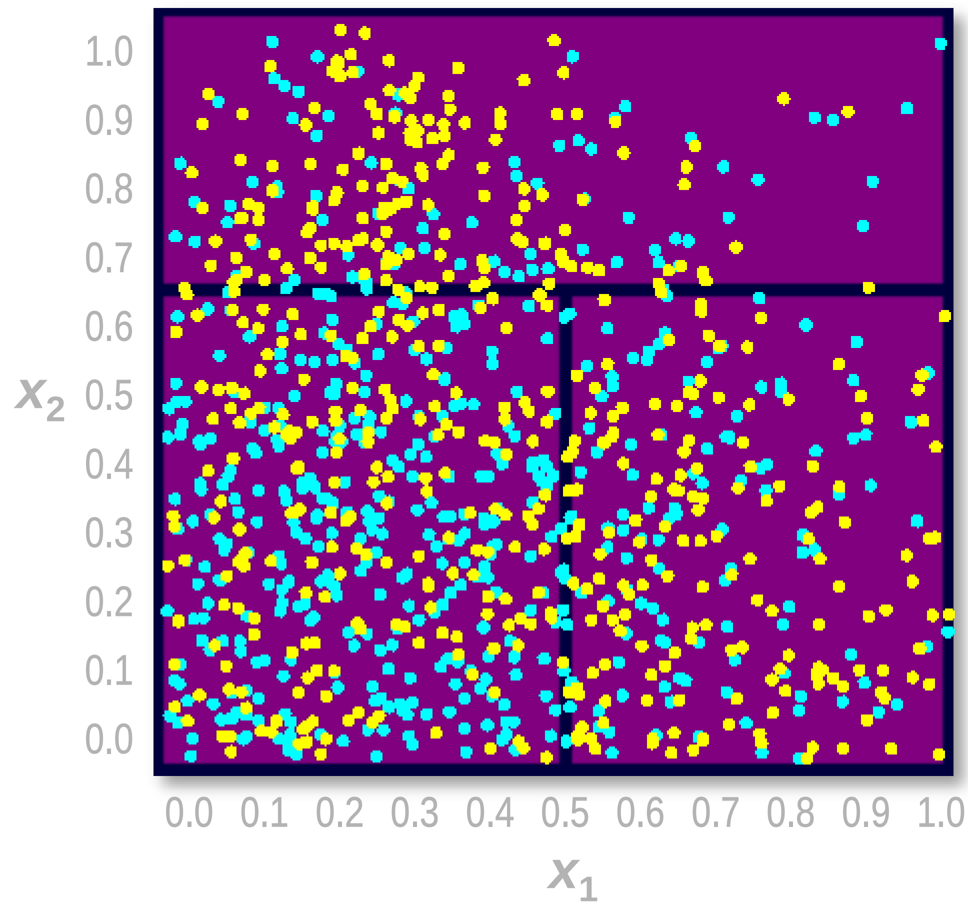
max
logworth(x_2)
-2.01



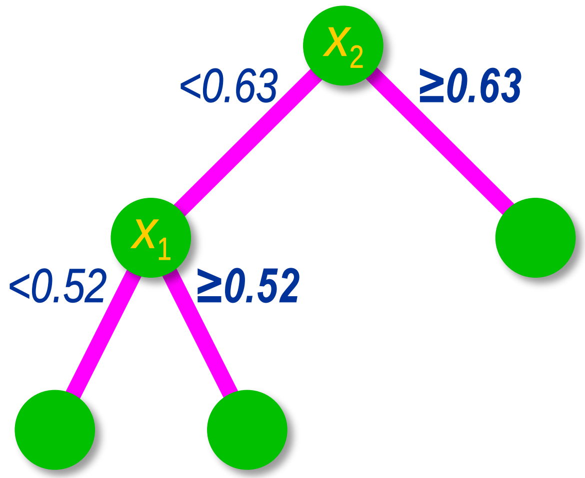
Decision Tree Split Search



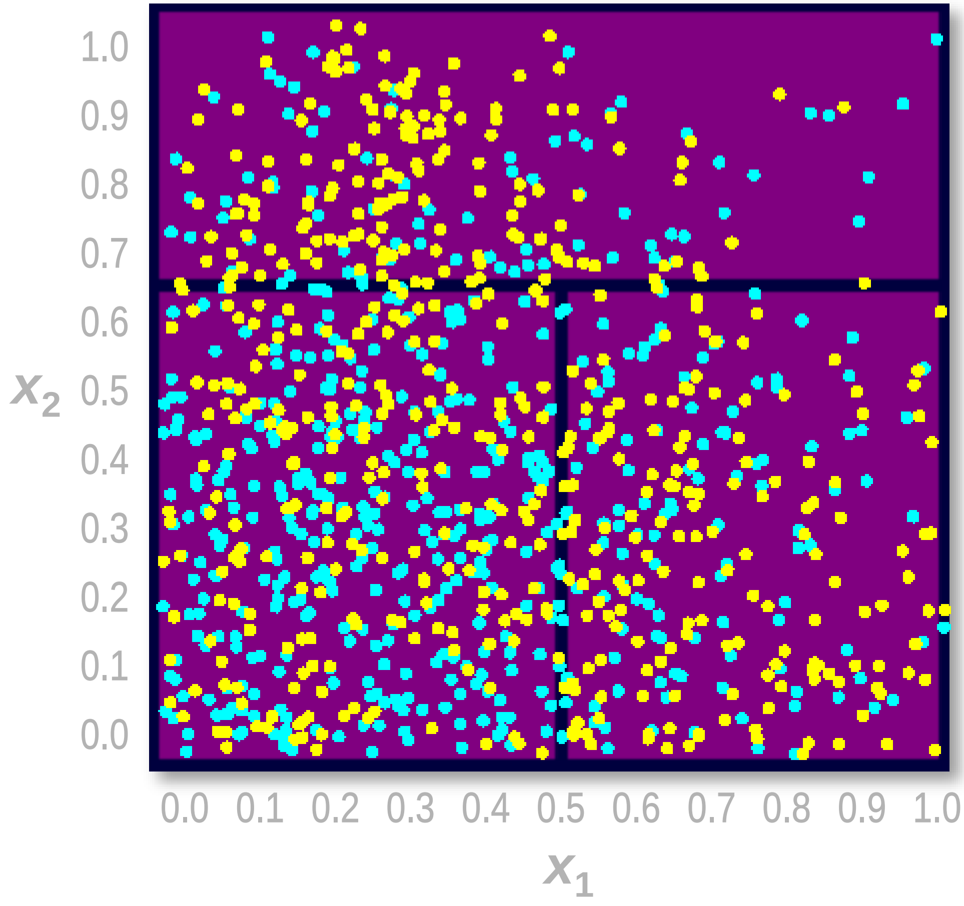
Create a second partition rule.



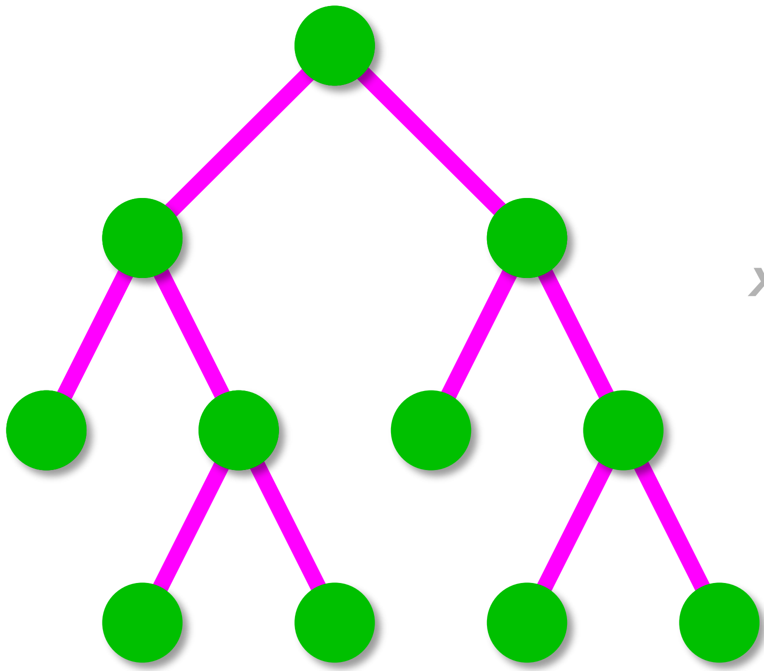
Decision Tree Split Search

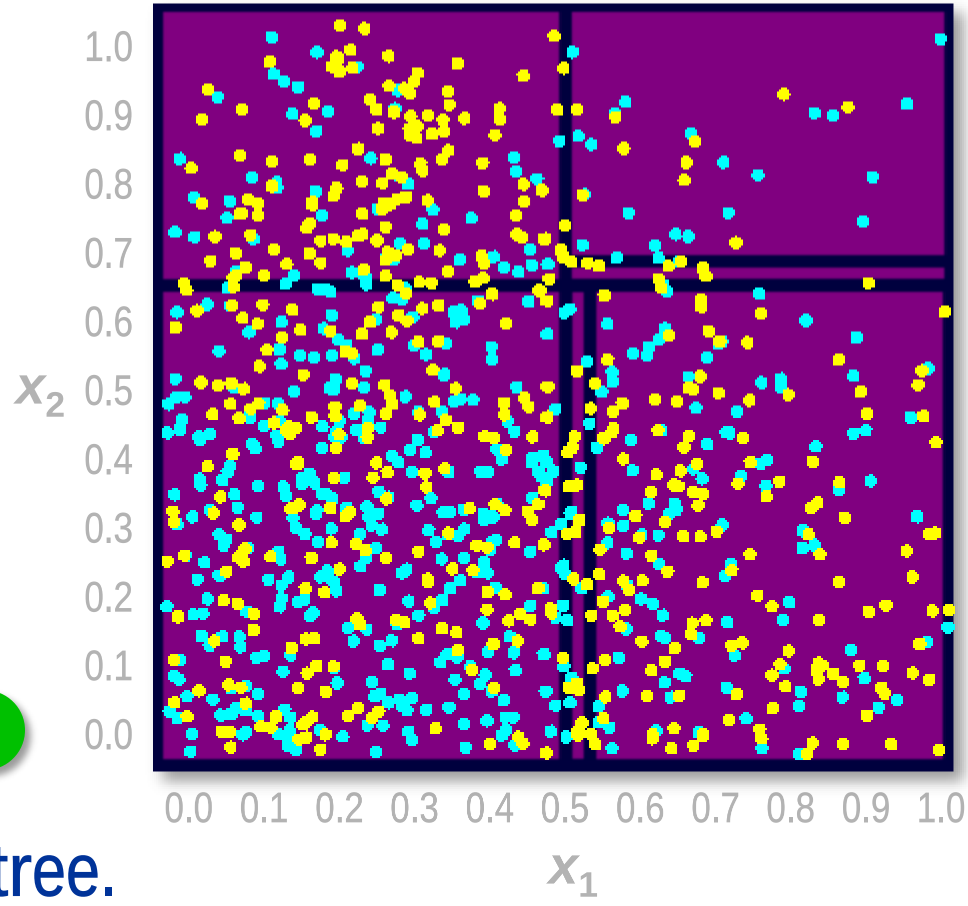


Create a second partition rule.



Decision Tree Split Search





Repeat to form a maximal tree.

Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (Hunt, 1966)
  - ChAID (Kass, 1980)
  - CART (Breiman, Friedman, Olshen, & Stone, 1984)
  - ID3, C4.5 (Quinlan, 1986, 1993)
  - SLIQ (Mehta, Agrawal, Rissanen,1996)
  - SPRINT (Shaffer, Agrawal, Mehta, 1996)

Growing a Classification Tree

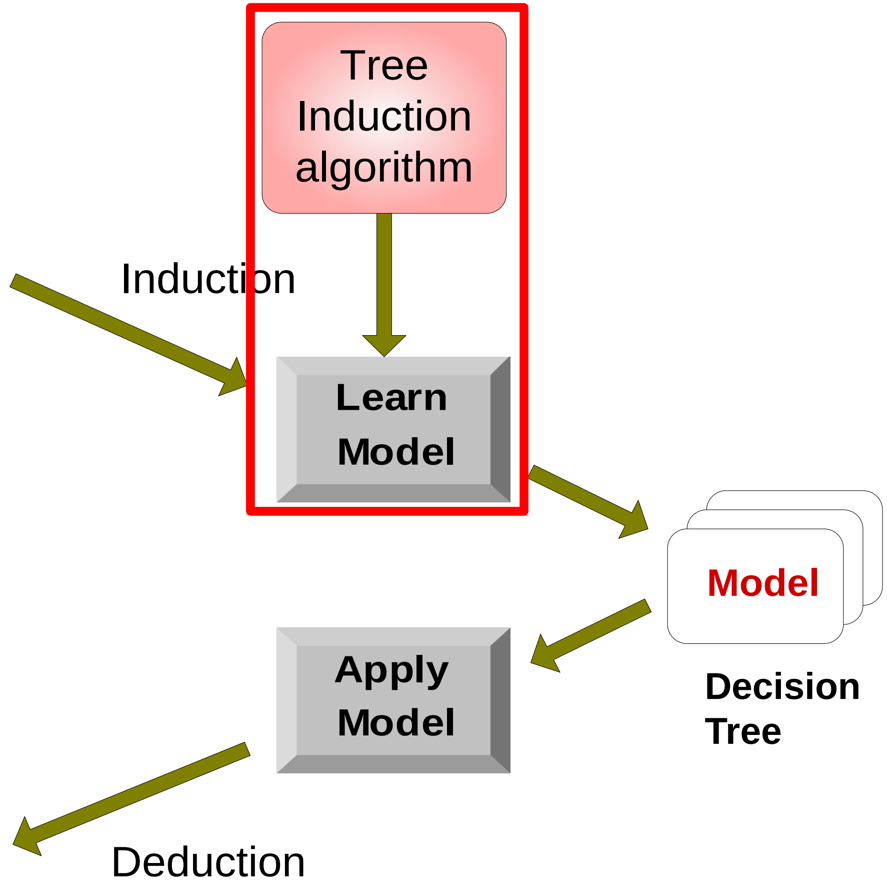
- A classification tree is very similar to a regression tree except that we try to make a prediction for a categorical rather than continuous Y.
- For each region (or node) we predict <u>the most common category</u> among the training data within that region.
- There are several possible different criteria to use such as the “gini index” and “logworth” but the easiest one to think about is to minimize the error rate.

Applying a Decision Tree Model

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



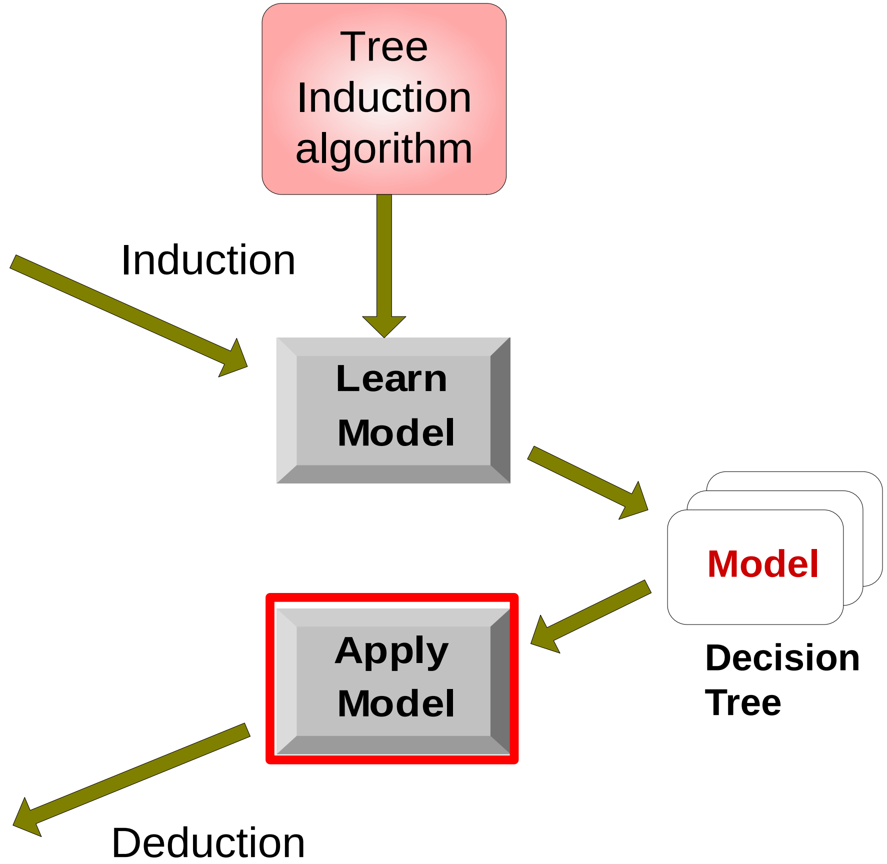
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



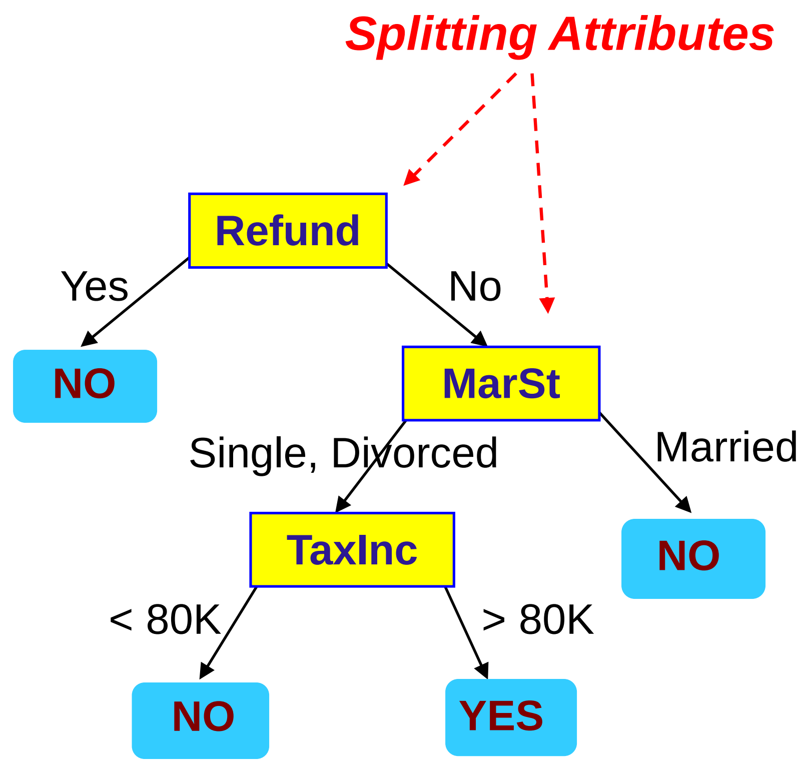
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Example of a Decision Tree – Tax Fraud Detection

categorical categorical continuous class

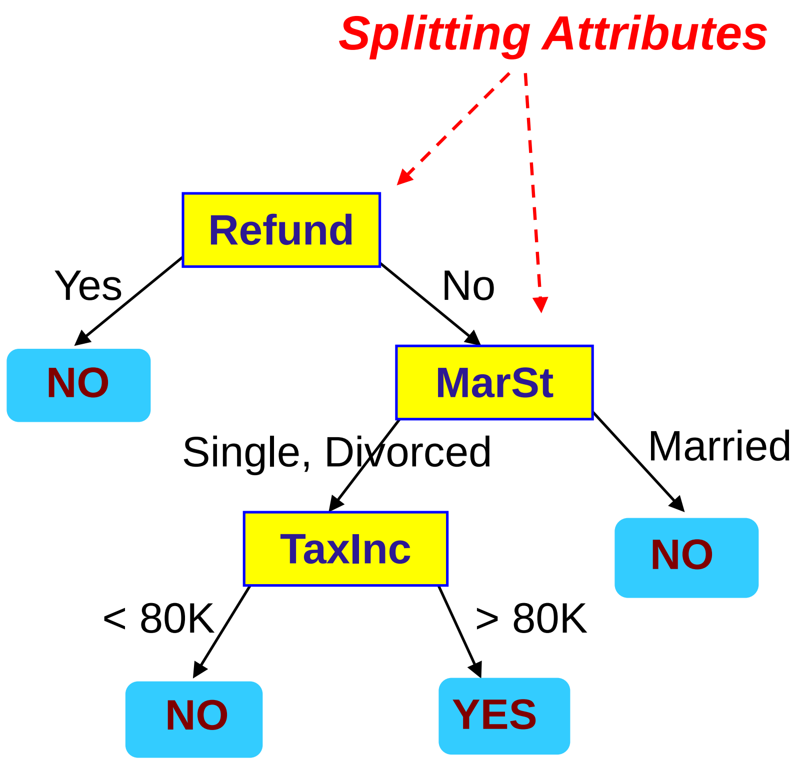
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Training Data

Model: Decision Tree

Trees as Sets of Rules



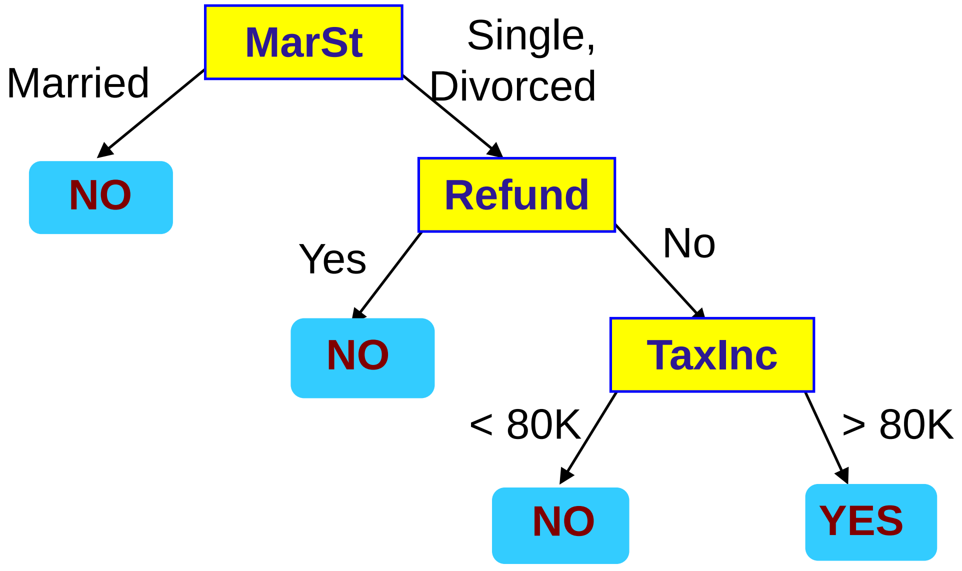
Model: Decision Tree

If a tax refund is requested, *then* the person is not cheating on Tax.

…

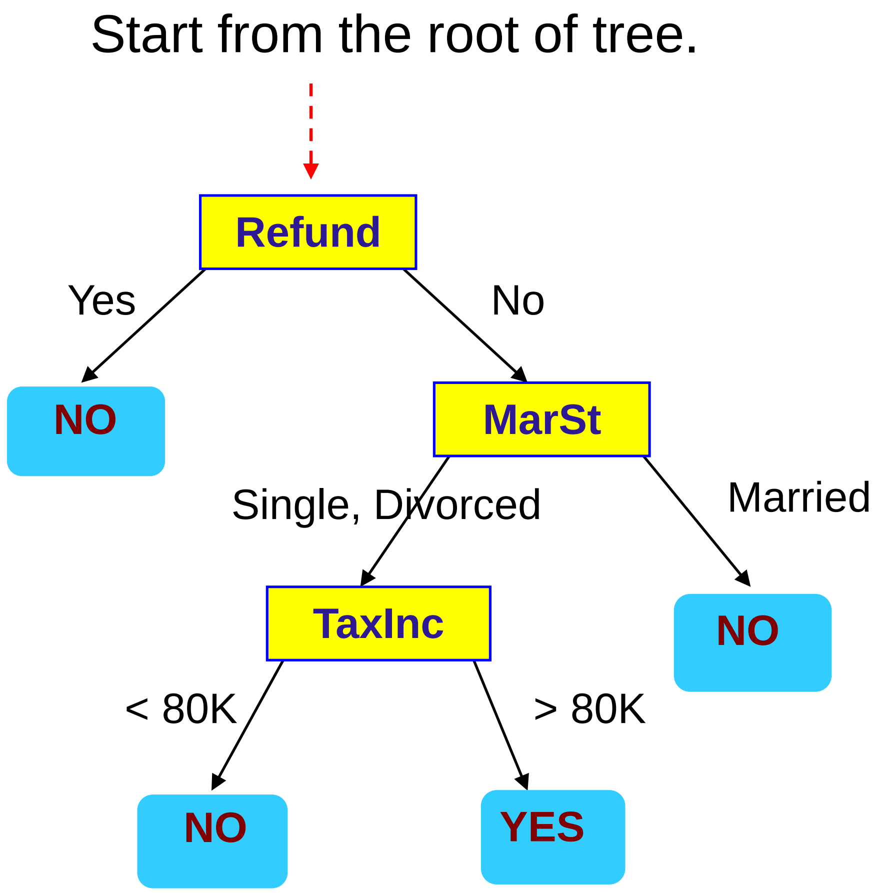
Example of a Decision Tree – Tax Fraud Detection

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

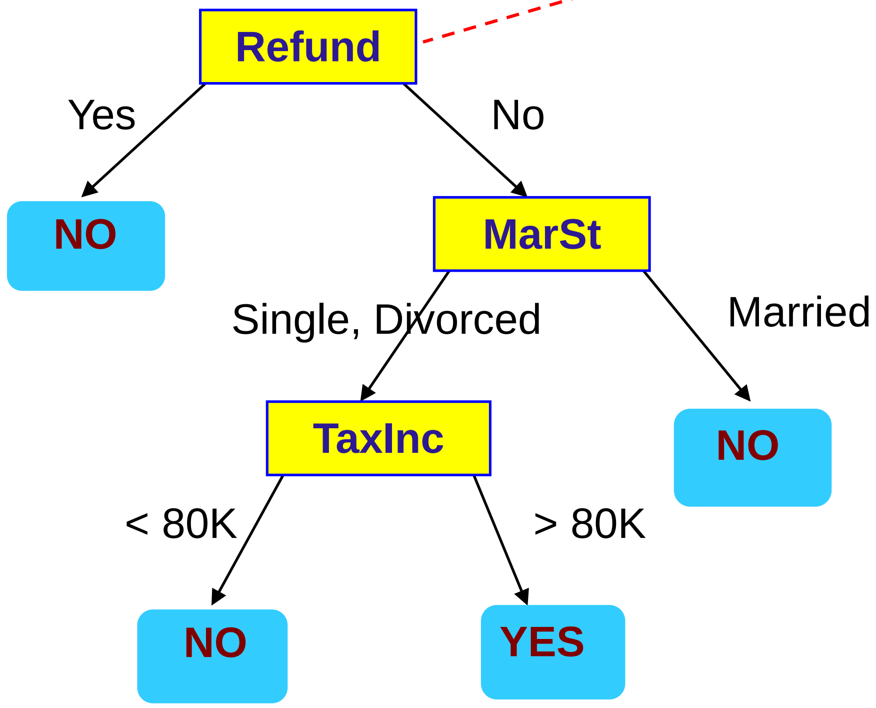
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

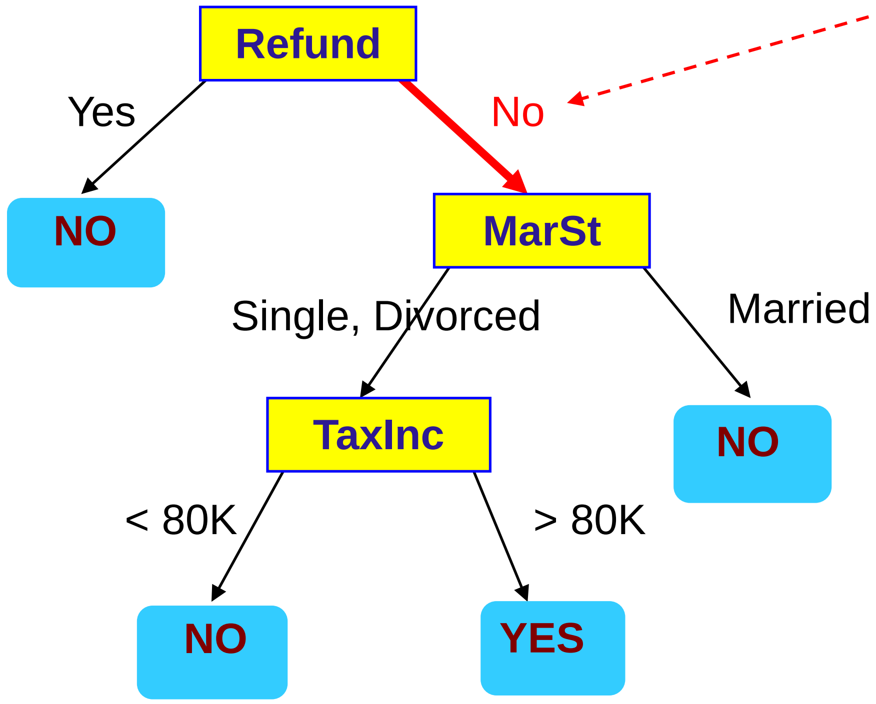
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

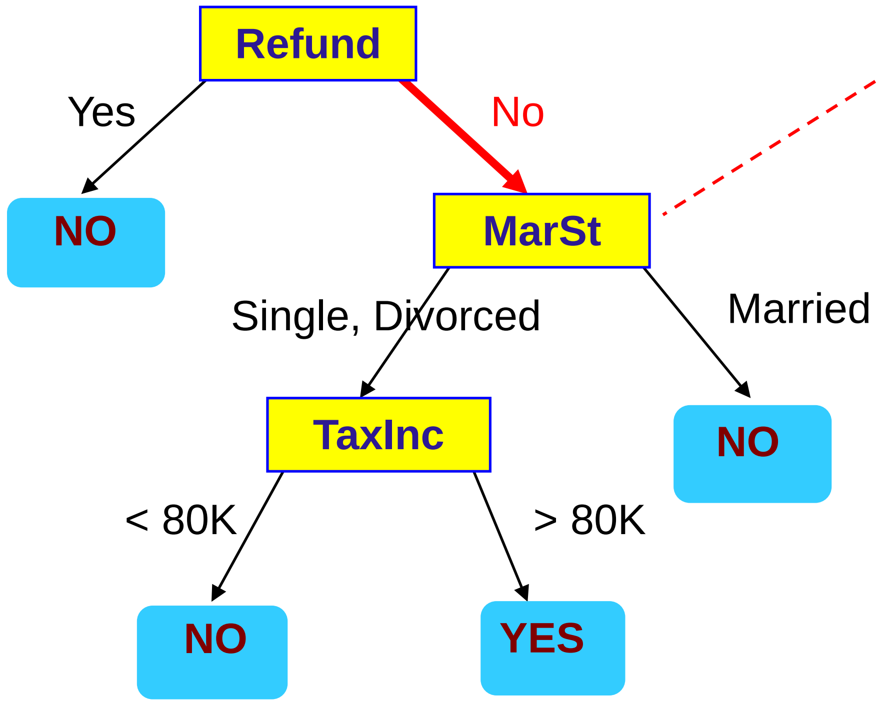
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



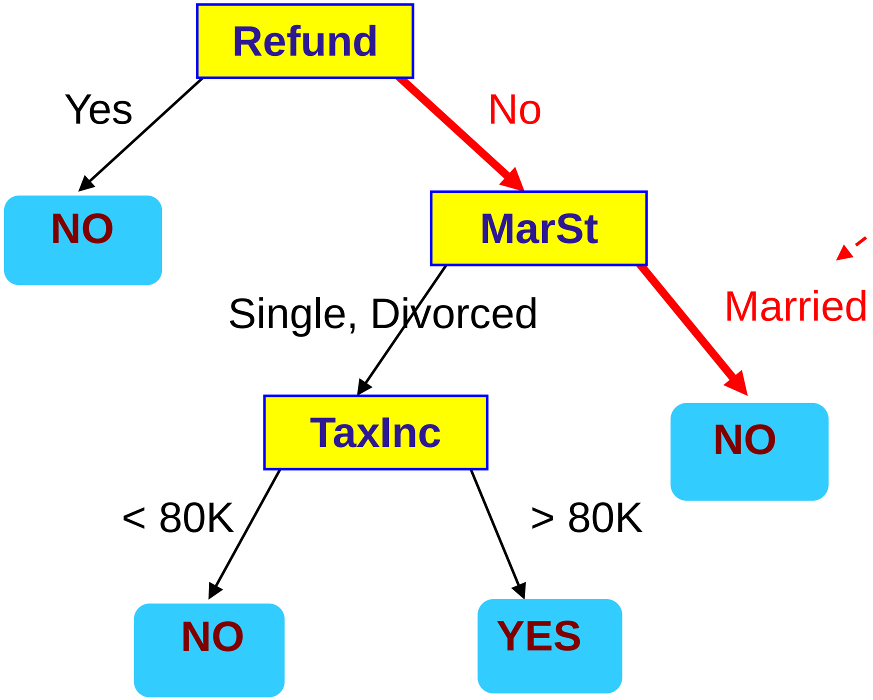
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

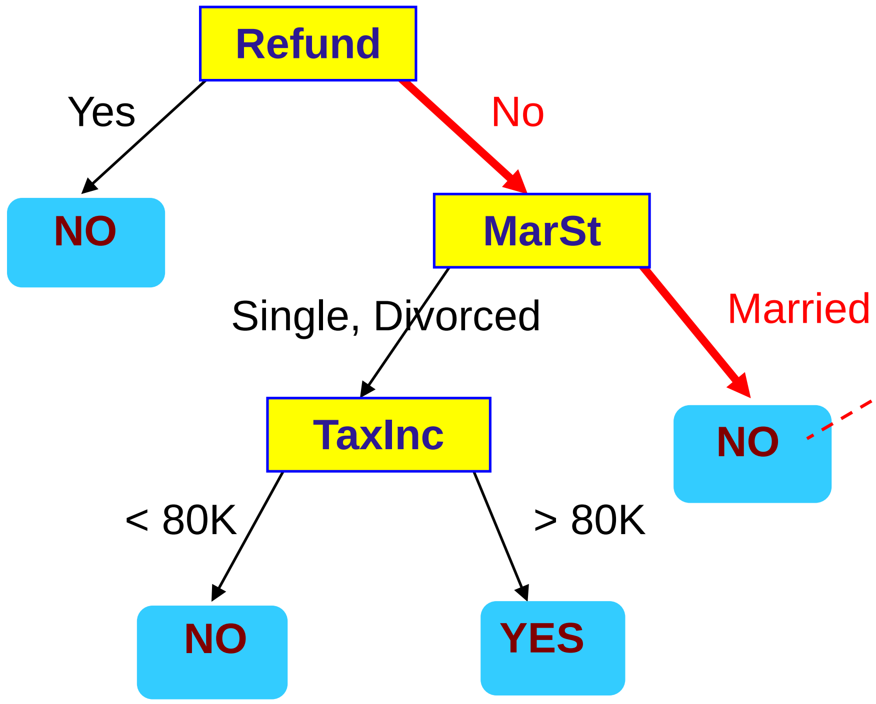
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

Trees vs. Linear models

Trees vs. Linear Models

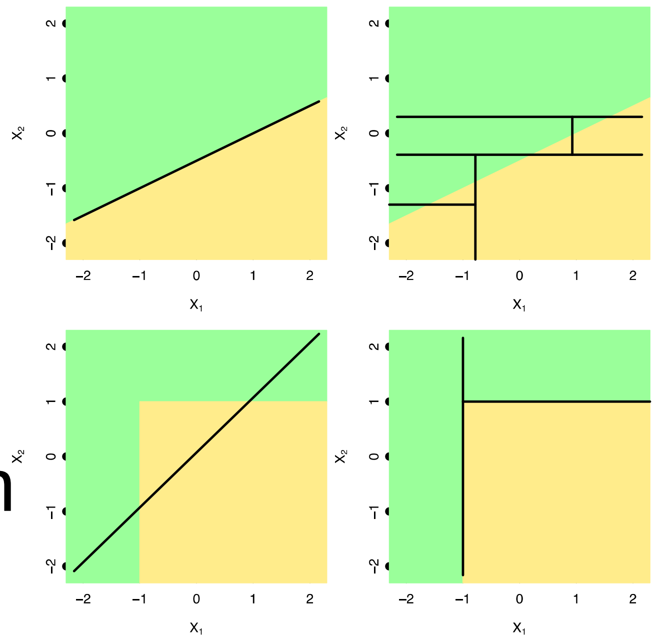
- In general, which model is better?
  - If the relationship between the predictors and response is linear, then classical linear models such as linear regression would outperform regression trees
  - On the other hand, if the relationship between the predictors is non-linear, then decision trees would outperform classical approaches

Trees vs. Linear Models

- Regression Models are global, and they do not do a good job of fitting data that has local characteristics.
- Decision tree models are local – it is fine for the relationship between variables to be quite different in different leaves.
- Decision tree segment data into boxes, while logistic regression/SVM partition data into classes by drawing lines
  - Global models are weak when there are several very different ways for record to become part of the target class

Trees vs. Linear Model: Classification Example

- Top row: the true decision boundary is linear
  - Left: linear model (good)
  - Right: decision tree
- Bottom row: the true decision boundary is non-linear
  - Left: linear model
  - Right: decision tree (good)



Pros and Cons of Decision Trees

- Pros:
  - Trees are very easy to explain to people (probably even easier than linear regression)
  - Trees can be plotted graphically, and are easily interpreted even by non-expert
  - They work fine on both classification and regression problems

- Cons:
  - Trees don’t have the same prediction accuracy as some of the more complicated approaches that we examine in this course

Reference

- Tan, Pang-Ning, Steinbach, Michael, Kumar, Vipin, Karpatne, Anuj. "*Introduction to Data Mining*", (Pearson, 2nd edition, 2018) [chapter 3]
- SAS Institute. Predictive Modeling Slides