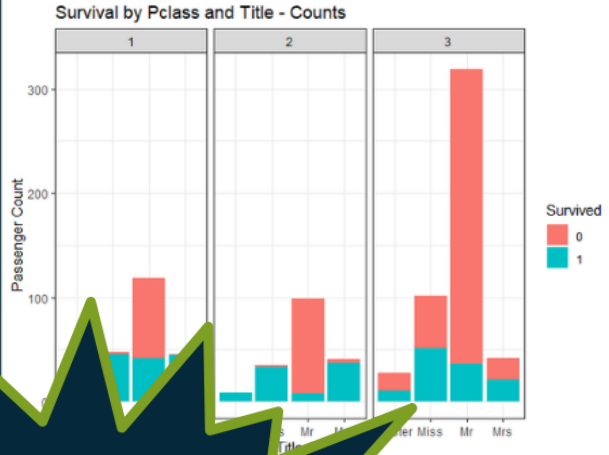
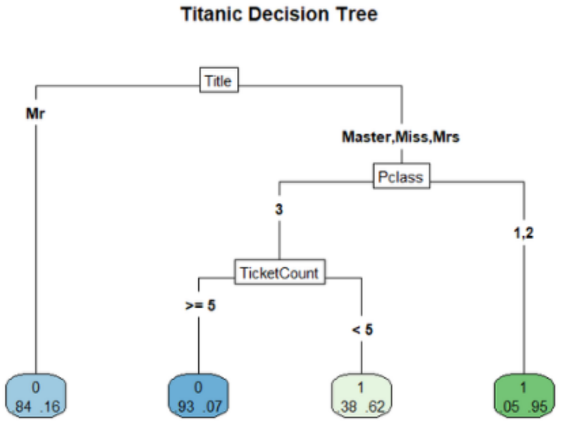
Dave
ON DATA

Introduction to Machine Learning With R

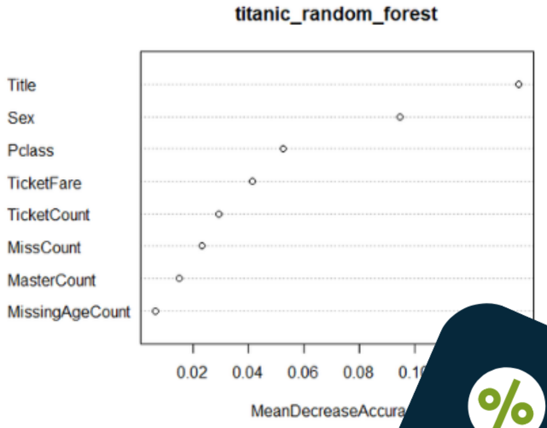
A Course Designed for ANY Professional



Survival by Pclass and Title - Counts



Titanic Decision Tree



titanic_random_forest

40% OFF!

https://bit.ly/DaveOnDataBlackFriday

%

BLACK FRIDAY SALE

ENDS NOV 29TH

Want to have more impact at work with data?

Need help applying data analytics to *your* business problem?

COMING JAN 2022

For students of my online and live courses, I will be offering personalized 1-on-1 coaching services.

- R programming coaching
- R & Excel data analysis coaching
- Machine learning coaching

CART
Classification Trees

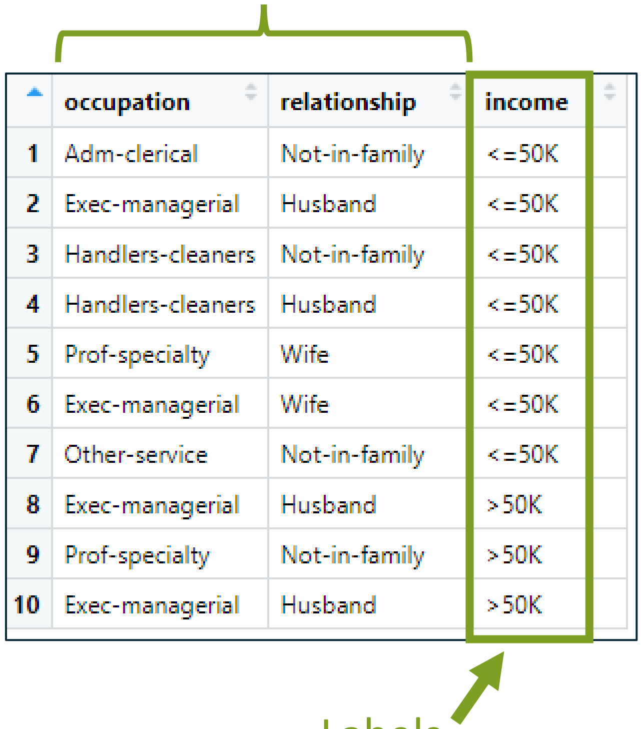
Classification Tree Intuition

Trees Are Rules

Classification trees embody a series of rules to assign/predict *labels*.

Take the following sample of the *Adult Census* data…

Feature/Variables



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

Labels

Using the data to the left, consider the following “rules”:

IF occupation **IS IN (**“Adm-clerical”**,** “Handlers-cleaners”**,** “Other-service”**)**
THEN income **=** “<=50K”

ELSE IF relationship **=** “Wife” **THEN** income **=** “<=50K”

ELSE income **=** “>50K”

How does the classification tree algorithm arrive at these rules?

Classification Trees Minimize "Impurity"

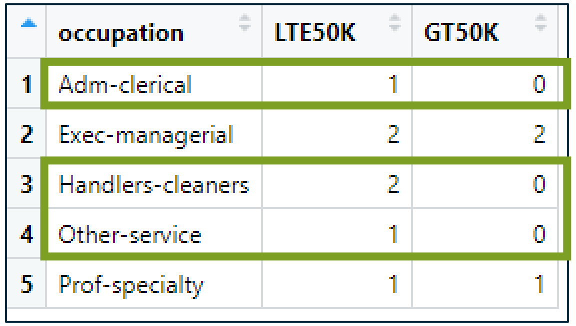
Machine learning algorithms work by trying to achieve an *objective*.

In the case of classification trees, the objective is to *minimize impurity*.

Later we will learn about the math of impurity, right now it's intuition time…

Feature Values

Label Counts



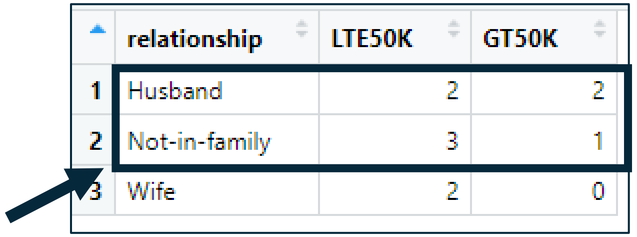
	occupation	LTE50K	GT50K
1	Adm-clerical	1	0
2	Exec-managerial	2	2
3	Handlers-cleaners	2	0
4	Other-service	1	0
5	Prof-specialty	1	1

These are "pure"

Feature Values

Label Counts

These are "impure"



	relationship	LTE50K	GT50K
1	Husband	2	2
2	Not-in-family	3	1
3	Wife	2	0

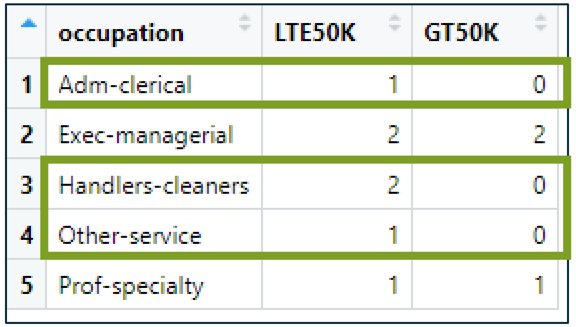
The classification tree algorithm iteratively uses the features to split the data into the largest collections of "purest" labels possible.

Let’s Build a Tree!

In this contrived example we have 10 *observations* and 2 *features*.

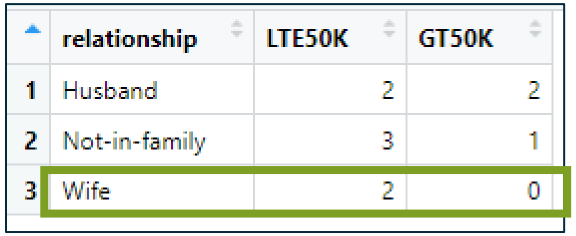
The classification tree algorithm *loves* a single feature with lots of observations and only a single *label*.

This obsessive love is known as being *greedy*…



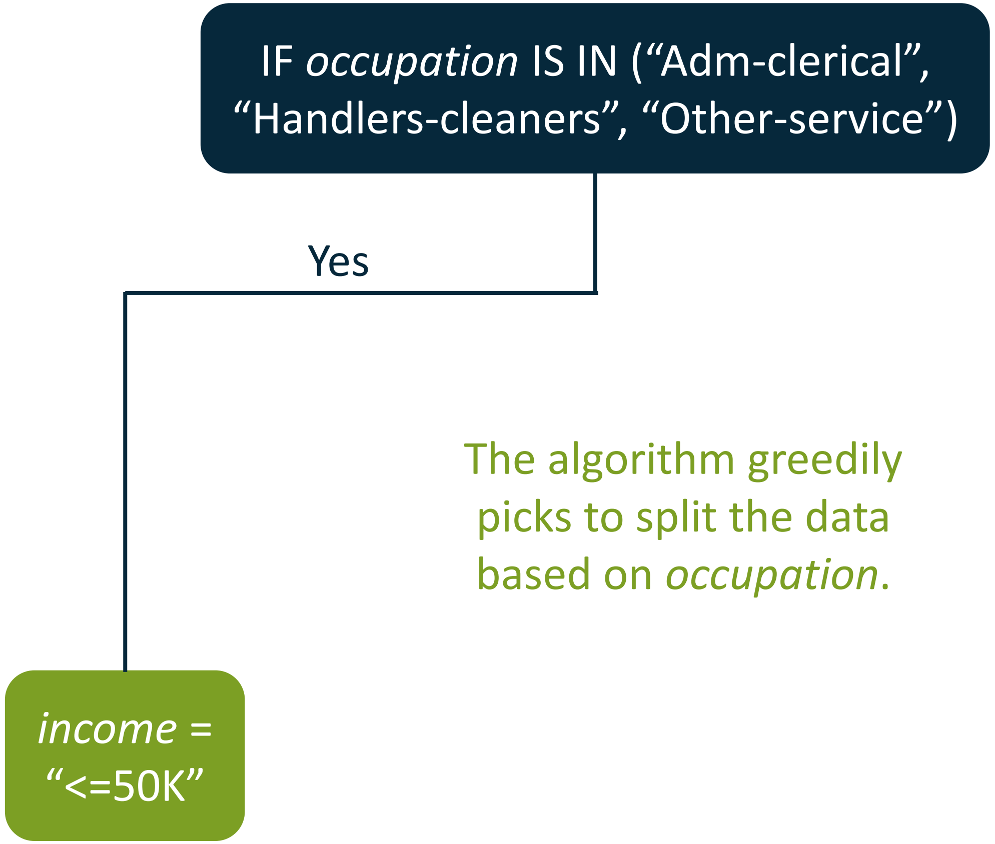
	occupation	LTE50K	GT50K
1	Adm-clerical	1	0
2	Exec-managerial	2	2
3	Handlers-cleaners	2	0
4	Other-service	1	0
5	Prof-specialty	1	1

With the *occupation* feature we get 4 observations all with the label “<=50K”



	relationship	LTE50K	GT50K
1	Husband	2	2
2	Not-in-family	3	1
3	Wife	2	0

With the *relationship* feature we get 2 observations all with the label “<=50K”



IF *occupation* IS IN (“Adm-clerical”, “Handlers-cleaners”, “Other-service”)

Yes

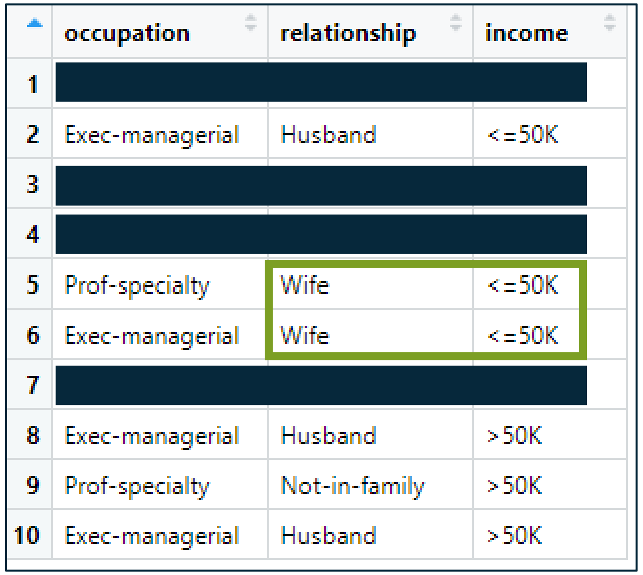
income = “<=50K”

The algorithm greedily picks to split the data based on *occupation*.

What’s Next?

The algorithm has used 40% of the data in the first split…

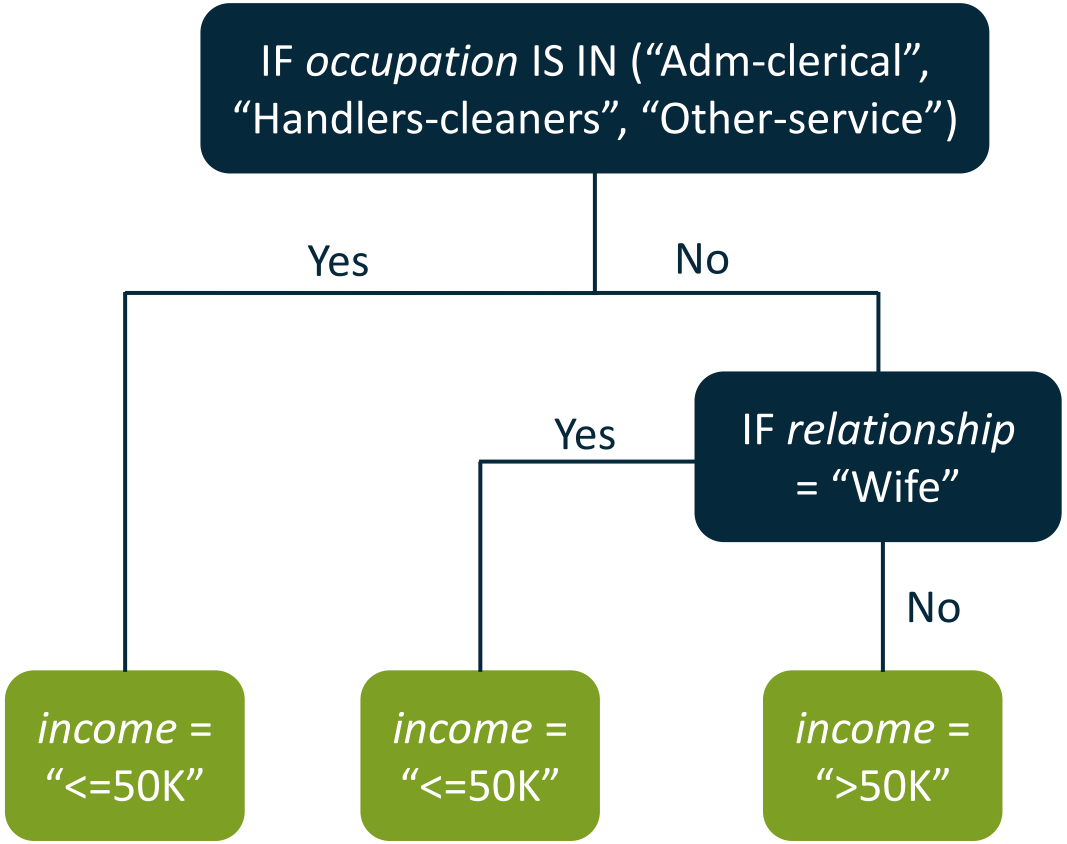
What’s next?



	occupation	relationship	income
1	[redacted]	[redacted]	[redacted]
2	Exec-managerial	Husband	<=50K
3	[redacted]	[redacted]	[redacted]
4	[redacted]	[redacted]	[redacted]
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	[redacted]	[redacted]	[redacted]
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

We know this split is pure!

Of the data remaining, the “impurity math” we’ll learn later tells us there are no splits left.



IF *occupation* IS IN (“Adm-clerical”, “Handlers-cleaners”, “Other-service”)

IF *relationship* = “Wife”

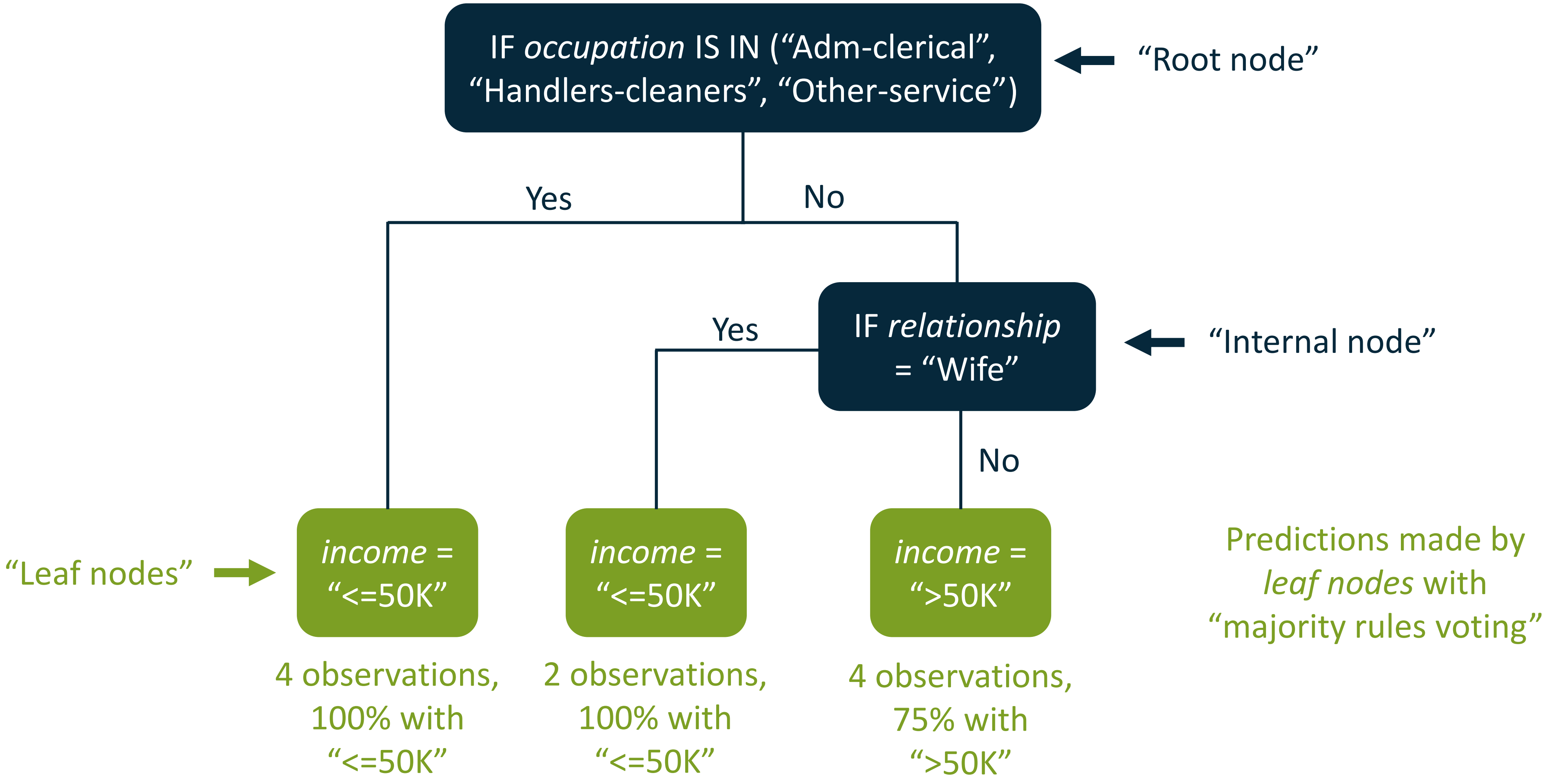
income = “<=50K”

income = “<=50K”

income = “>50K”

Getting Meta

Some things we need to know about trees…



IF *occupation* IS IN (“Adm-clerical”, “Handlers-cleaners”, “Other-service”)

“Root node”

Yes

No

IF *relationship* = “Wife”

“Internal node”

Yes

No

“Leaf nodes”

income = “<=50K”

4 observations, 100% with “<=50K”

income = “<=50K”

2 observations, 100% with “<=50K”

income = “>50K”

4 observations, 75% with “>50K”

Predictions made by *leaf nodes* with “majority rules voting”

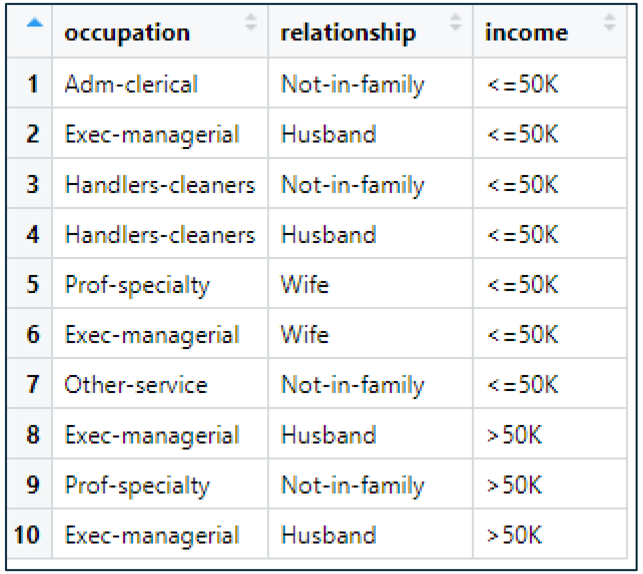
Overfitting Intuition

The Bugbear of Machine Learning

As a professional applying machine learning to your data, you must be paranoid about *overfitting*.

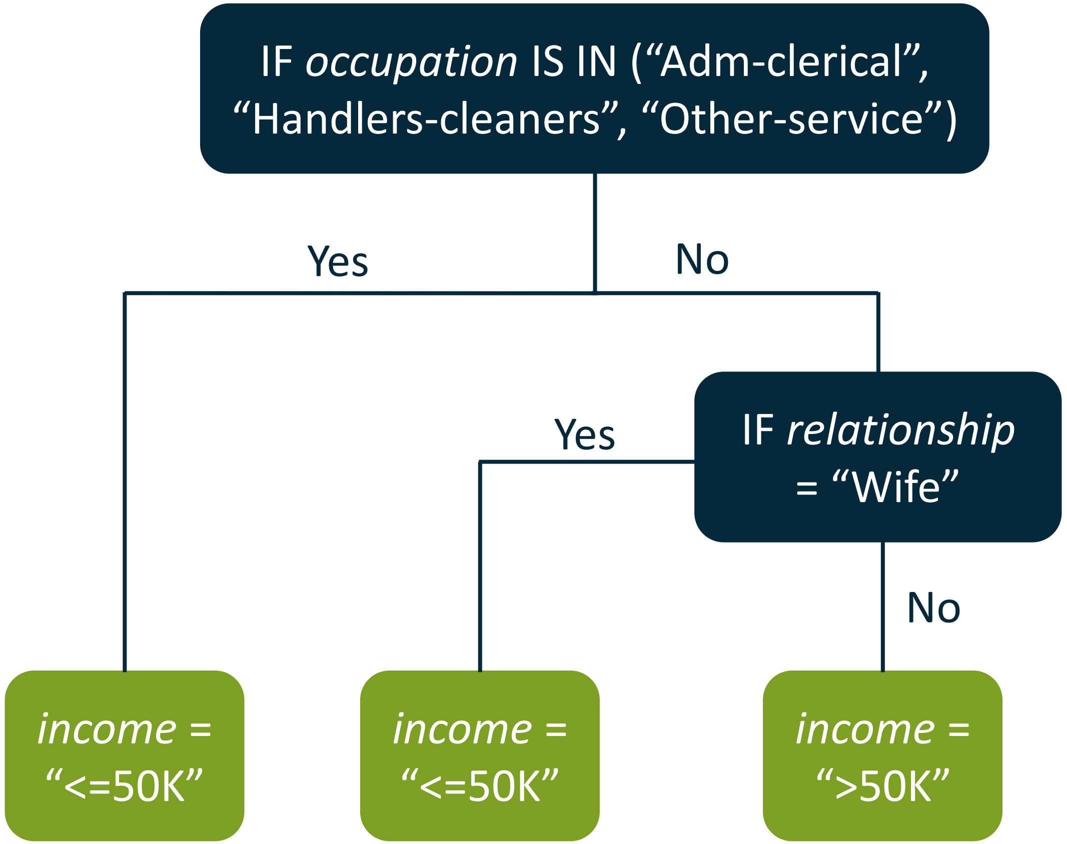
Simply put, overfitting is where your model's predictions are much less "accurate" on new data.

We will be covering overfitting in depth in the next section, for now it's intuition time...



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

The combination of *data*, *algorithm*, and *training regimen* produced the following model...



IF *occupation* IS IN ("Adm-clerical", "Handlers-cleaners", "Other-service")

Yes

No

IF *relationship* = "Wife"

Yes

No

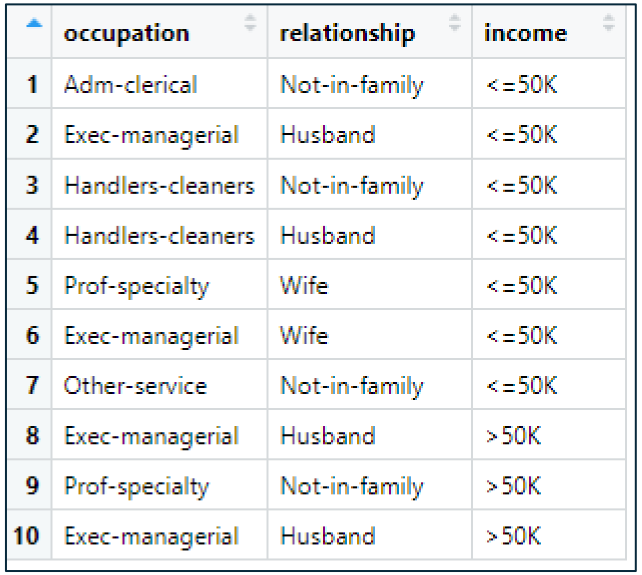
income = "<=50K"

income = "<=50K"

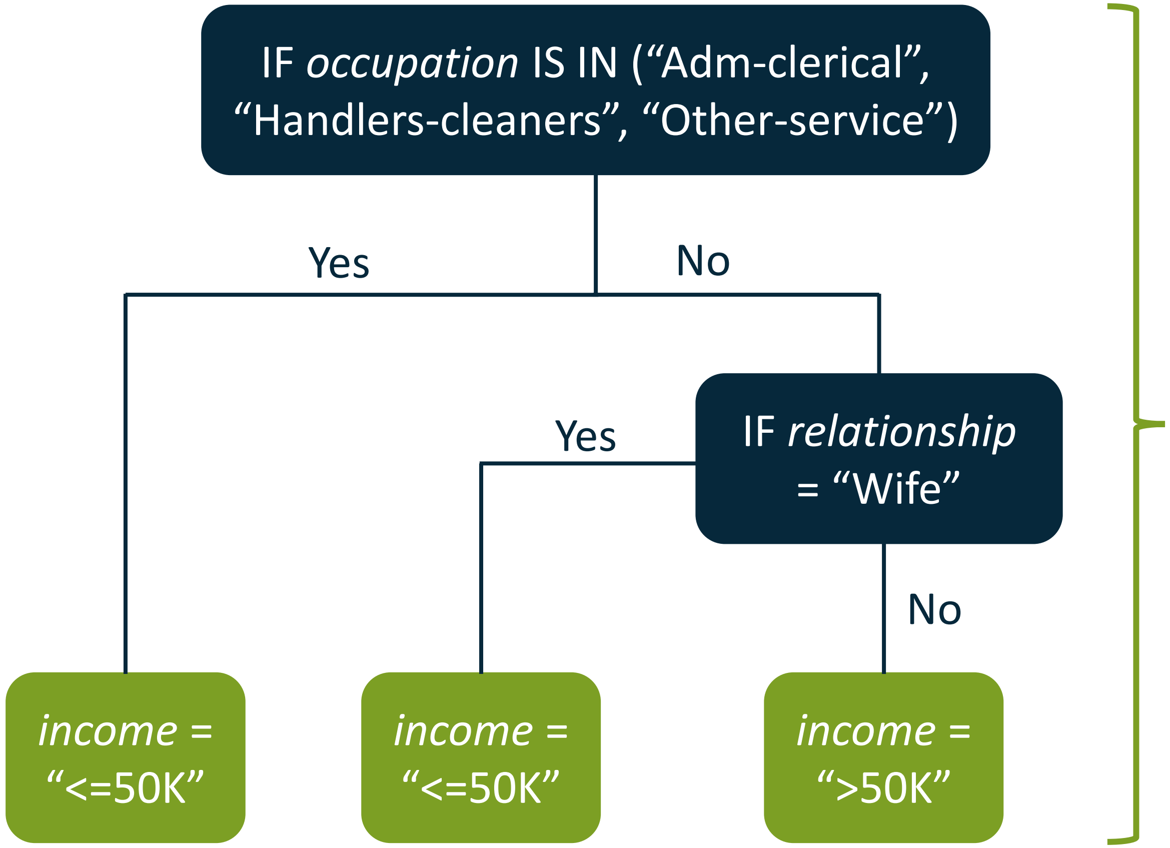
income = ">50K"

The Model Is Good!

Considering the data used to *train* the model, things look awesome!



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K



IF *occupation* IS IN ("Adm-clerical", "Handlers-cleaners", "Other-service")

Yes

No

Yes

IF *relationship* = "Wife"

No

income = "<=50K"

income = "<=50K"

income = ">50K"

4 observations, 100% with "<=50K"

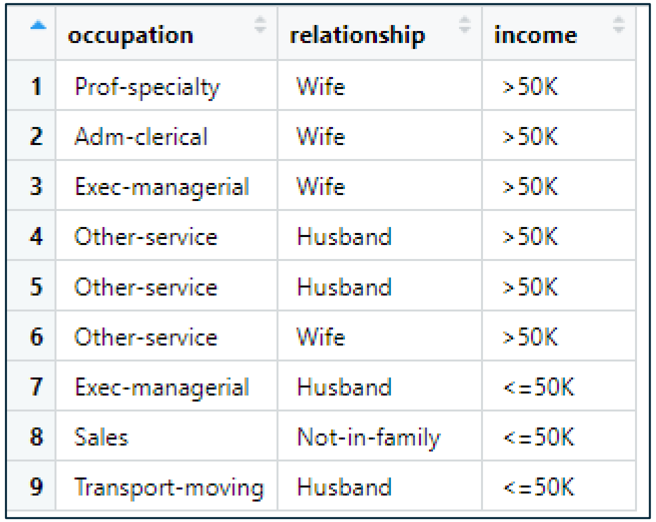
2 observations, 100% with "<=50K"

4 observations, 75% with ">50K"

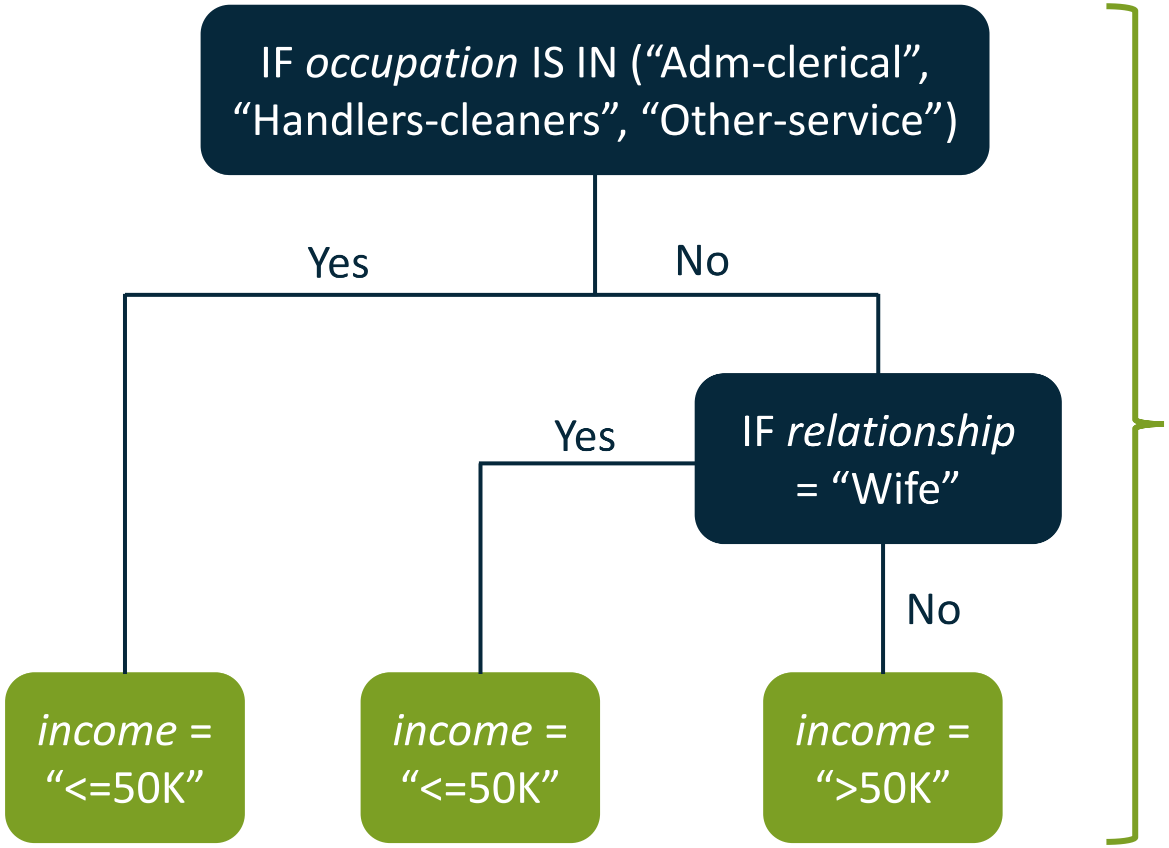
This model gets 9 out of 10 (i.e., 90%) of labels correct!

Or Is It?

Consider the following data that wasn't used in training the model…



	occupation	relationship	income
1	Prof-specialty	Wife	>50K
2	Adm-clerical	Wife	>50K
3	Exec-managerial	Wife	>50K
4	Other-service	Husband	>50K
5	Other-service	Husband	>50K
6	Other-service	Wife	>50K
7	Exec-managerial	Husband	<=50K
8	Sales	Not-in-family	<=50K
9	Transport-moving	Husband	<=50K


IF *occupation* IS IN ("Adm-clerical", "Handlers-cleaners", "Other-service")
Yes
No
IF *relationship* = "Wife"
Yes
No
income = "<=50K"
income = "<=50K"
income = ">50K"

4 observations, 100% with "<=50K"

2 observations, 100% with "<=50K"

4 observations, 75% with ">50K"

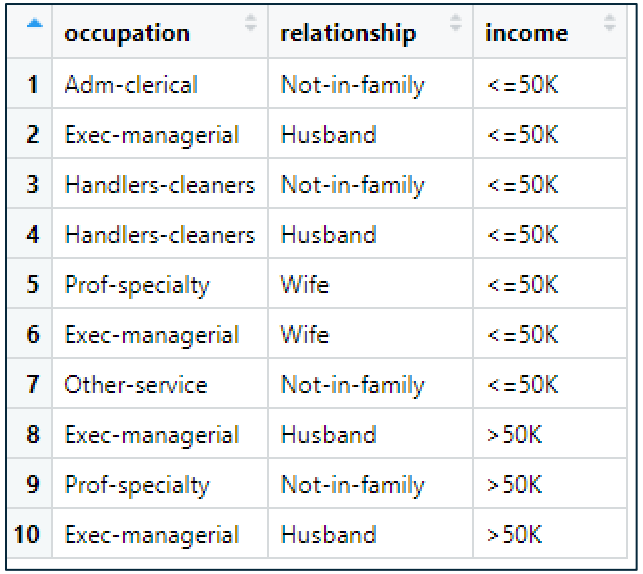
This model gets 9 out of 9 (i.e., 100%) of labels wrong!

This is the essence of overfitting

What Happened?

In this example, the overfitting can be blamed on the *training regimen*…

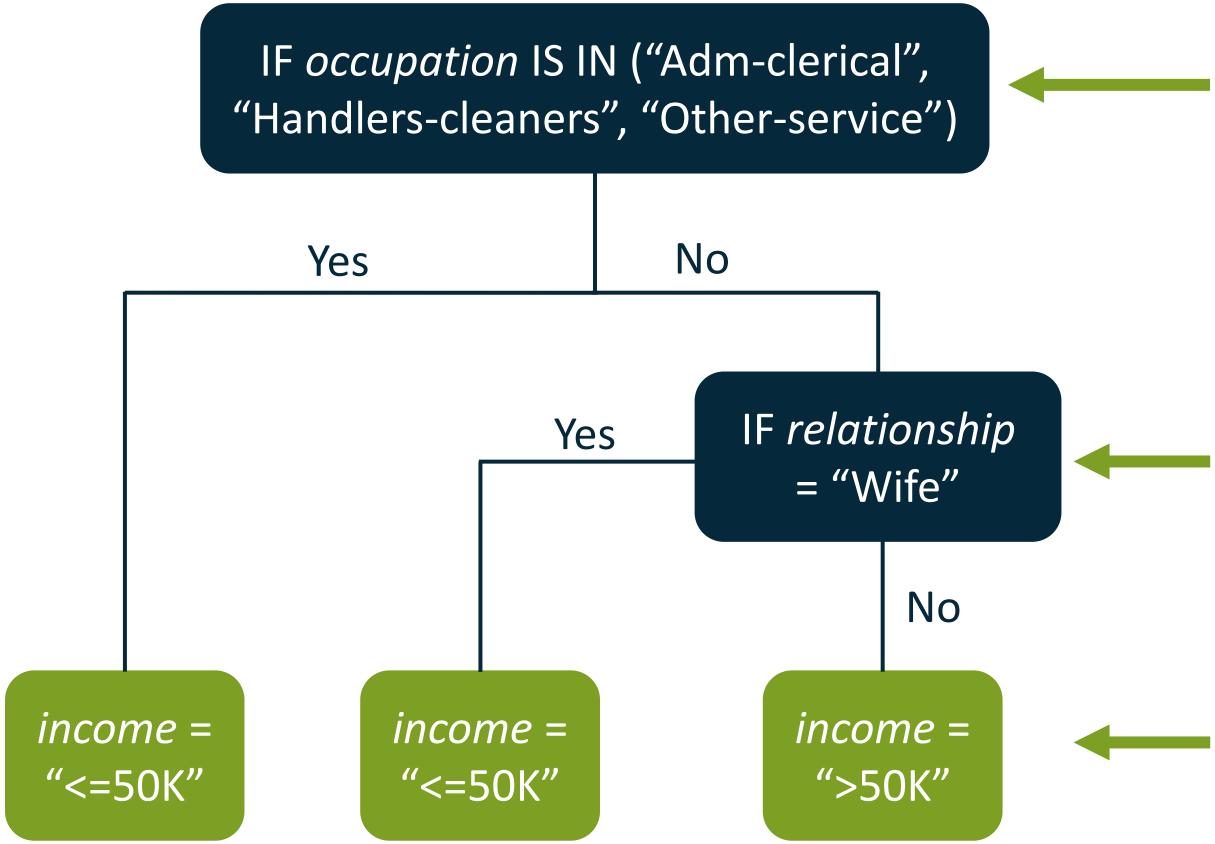
Given this small amount of data…



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

The greediness of decision trees makes them prone to overfit!

This model is far too specialized (or *complex*)!



IF *occupation* IS IN (“Adm-clerical”, “Handlers-cleaners”, “Other-service”)

IF *relationship* = “Wife”

income = “<=50K”

income = “<=50K”

income = “>50K”

These professions always have *incomes* of “<=50K”

Married women always have *incomes* of “<=50K”

All others always have *incomes* of “>50K”

Model Tuning Intuition

The goal of your *training regimen* is to *tune* your models to combat *overfitting*.

Conceptually, think of it like tuning your car for optimal performance.

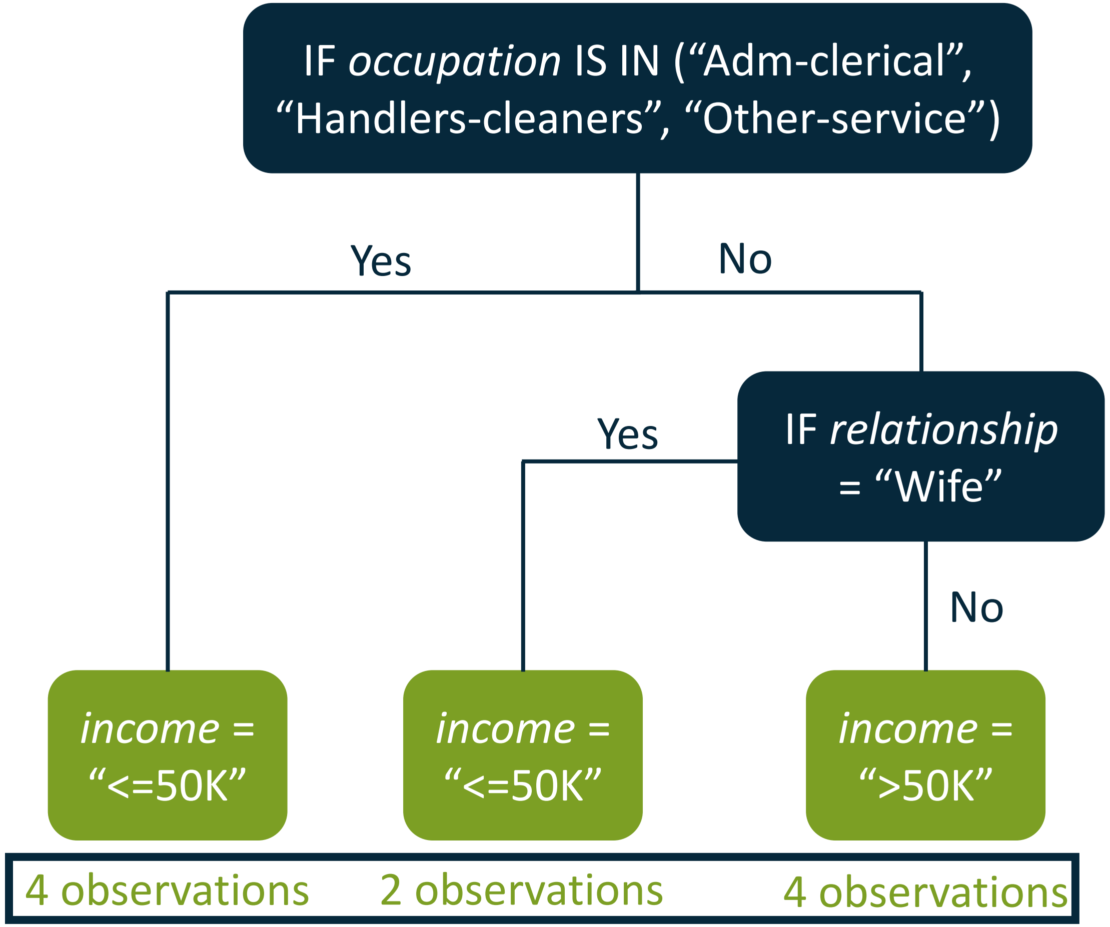
In terms of decisions trees…

One aspect of the decision tree "engine" we can tune is the minimum number of observations required for *leaf nodes*.

Small numbers of leaf node observations allow for more specialized (i.e., *complex*) trees

The minimum number of leaf node observations allowed is known as a *hyperparameter* and can be *tuned* to combat *overfitting*.

We will see later how *algorithms* combat overfitting.



Gini Impurity

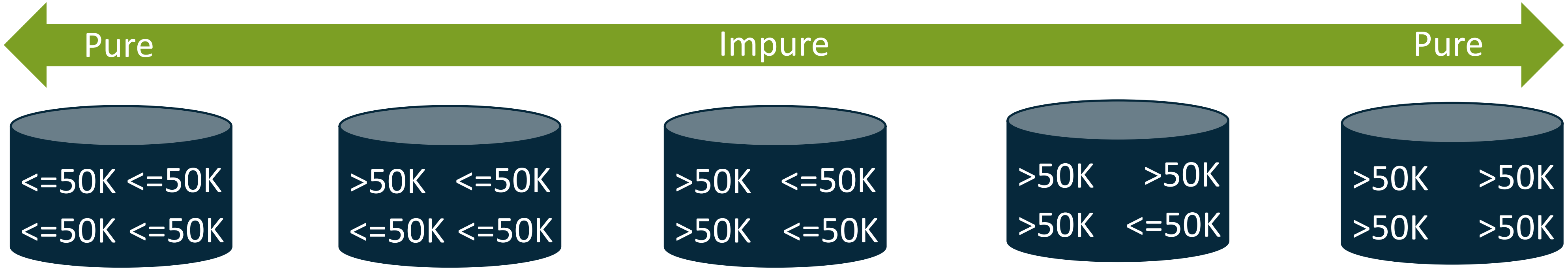
Impurity Intuition

We’ll start with the 2-label (i.e., the *binary*) scenario of thinking about *impurity*.

Using the *Adult Census* labels as an example, let’s think about “buckets” of labels.

We can think of impurity as a spectrum…

The classification tree algorithm needs a calculation that embodies this spectrum to allow it to evaluate each data split.



100% of one label

75% of one label

50% of each label

75% of one label

100% of one label

Gini Impurity

Turns out there are several calculations that manifest the intuition of the previous slide.

In this course we’ll use the *Gini impurity* calculation.

Gini impurity is widely used and is the default calculation used in the R packages used in this course.

Don’t panic! Here’s the equation…

“Go through all the class labels”

“Square the proportion”

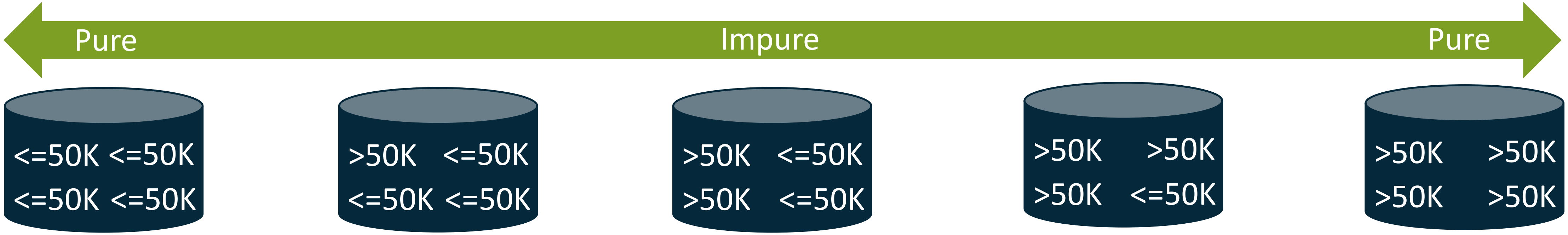
$$Gini(t) = 1 - \boxed{\sum_{i=1}^{c}} \underbrace{[p(i|t)]}_{\text{“Proportion of class labels”}}^{2}$$

“Subtract the stuff to the right from 1”

Gini Impurity Example

As we know, the classification tree algorithm's *objective* is to *minimize impurity*.

Now that we've got the math, let's revisit our buckets of labels…



4/4 == "<=50K"
0/4 == ">50K"

Gini = 0.375

2/4 == "<=50K"
2/4 == ">50K"

Gini = 0.375

0/4 == "<=50K"
4/4 == ">50K"

Gini = $1 – (4/4)^2 – (0/4)^2$

Gini = 1 – 1 – 0

Gini = 0

Gini = $1 – (2/4)^2 – (2/4)^2$

Gini = 1 – 0.25 – 0.25

Gini = 0.5

Gini = $1 – (0/4)^2 – (4/4)^2$

Gini = 1 – 0 – 1

Gini = 0

Gini Change

Minimizing Gini Impurity

OK, we know how to calculate *Gini impurity* and it isn’t difficult.

We also know that the classification decision tree algorithm’s *objective* is to minimize impurity.

The algorithm is going to *greedily* choose splits that produce *the lowest impurity for the most observations*.

Don’t panic! Here’s the equation…

“Count of observations for the split”

“Go through both splits”

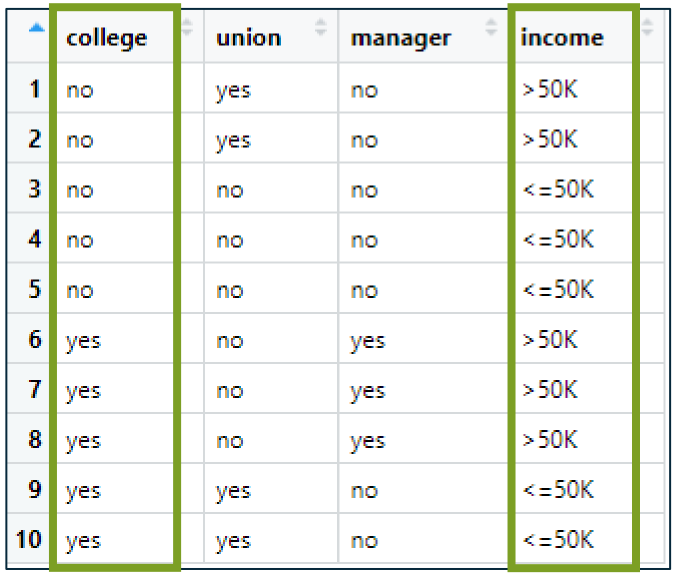
$$Gini\ Change = Gini(parent\ node) - \boxed{\sum_{j=1}^{2}} \frac{N(v_j)}{N} Gini(v_j)$$

“Gini impurity for split”

“Count of observations in parent node”

Training Our First Classification Tree

We now know what is needed to learn (or *train*) our first classification tree from data…



	college	union	manager	income
1	no	yes	no	>50K
2	no	yes	no	>50K
3	no	no	no	<=50K
4	no	no	no	<=50K
5	no	no	no	<=50K
6	yes	no	yes	>50K
7	yes	no	yes	>50K
8	yes	no	yes	>50K
9	yes	yes	no	<=50K
10	yes	yes	no	<=50K

Step #1 – What's our beginning Gini impurity?

$$\text{Gini Start} = 1 – (5/10)^2 – (5/10)^2$$

$$\text{Gini Start} = 1 – (0.25) – (0.25)$$

$$\text{Gini Start} = 0.5$$

Step #2 – What's Gini impurity for *college* based on splits?

$$\text{Gini “yes”} = 1 – (2/5)^2 – (3/5)^2$$

$$\text{Gini “yes”} = 1 – (0.16) – (0.36)$$

$$\text{Gini “yes”} = 0.48$$

$$\text{Gini “no”} = 1 – (3/5)^2 – (2/5)^2$$

$$\text{Gini “no”} = 1 – (0.36) – (0.16)$$

$$\text{Gini “no”} = 0.48$$

Step #3 – Weight the *college* splits by observation counts:

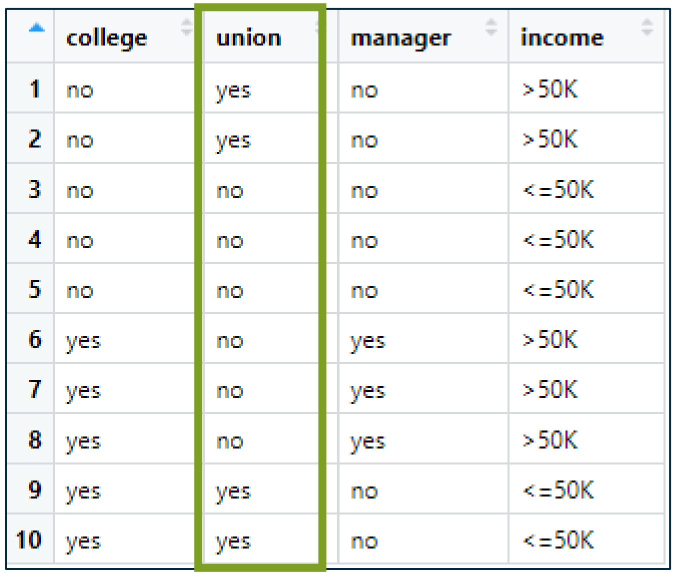
$$\text{Weighted Gini “yes”} = (5/10)(0.48) = 0.24$$

$$\text{Weighted Gini “no”} = (5/10)(0.48) = 0.24$$

Step #4 – Gini change with *college*:

$$\text{Gini Change} = (0.5) – (0.24) – (0.24) = 0.02$$

Training Our First Classification Tree



	college	union	manager	income
1	no	yes	no	>50K
2	no	yes	no	>50K
3	no	no	no	<=50K
4	no	no	no	<=50K
5	no	no	no	<=50K
6	yes	no	yes	>50K
7	yes	no	yes	>50K
8	yes	no	yes	>50K
9	yes	yes	no	<=50K
10	yes	yes	no	<=50K

Step #5 – What’s Gini impurity for *union* based on splits?

Gini “yes” = $1 - (2/4)^2 - (2/4)^2$

Gini “yes” = 1 – (0.25) – (0.25)

Gini “yes” = 0.5

Gini “no” = $1 - (3/6)^2 - (3/6)^2$

Gini “no” = 1 – (0.25) – (0.25)

Gini “no” = 0.5

Step #6 – Weight the *union* splits by observation counts:

Weighted Gini “yes” = (4/10)(0.5) = 0.2

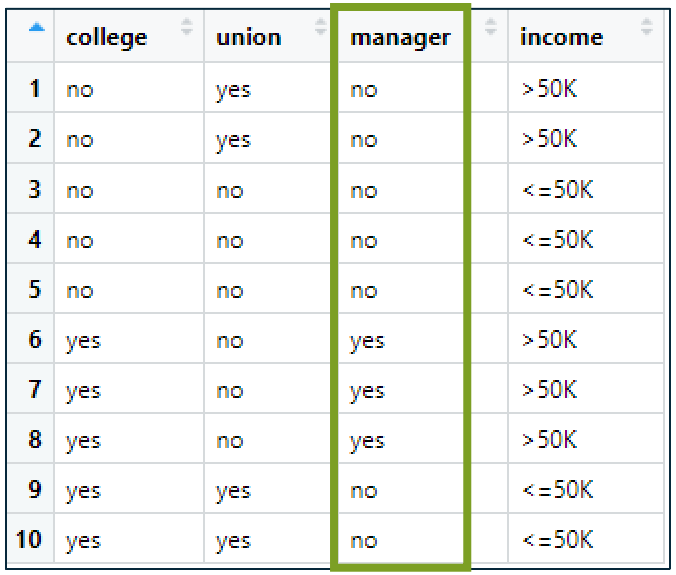
Weighted Gini “no” = (6/10)(0.5) = 0.3

Step #7 – Gini change with *union*:

Gini Change = (0.5) – (0.2) – (0.3) = 0.0

Using the *union* feature to split the data doesn’t help things!

Training Our First Classification Tree



	college	union	manager	income
1	no	yes	no	>50K
2	no	yes	no	>50K
3	no	no	no	<=50K
4	no	no	no	<=50K
5	no	no	no	<=50K
6	yes	no	yes	>50K
7	yes	no	yes	>50K
8	yes	no	yes	>50K
9	yes	yes	no	<=50K
10	yes	yes	no	<=50K

Step #8 – What’s Gini impurity for *manager* based on splits?

Gini “yes” = $1 - (0/3)^2 - (3/3)^2$

Gini “yes” = $1 - (0.0) - (1)$

Gini “yes” = 0.0

Gini “no” = $1 - (5/7)^2 - (2/7)^2$

Gini “no” = $1 - (0.51) - (0.08)$

Gini “no” = 0.41

Step #9 – Weight the *manager* splits by observation counts:

Weighted Gini “yes” = (3/10)(0.0) = 0.0

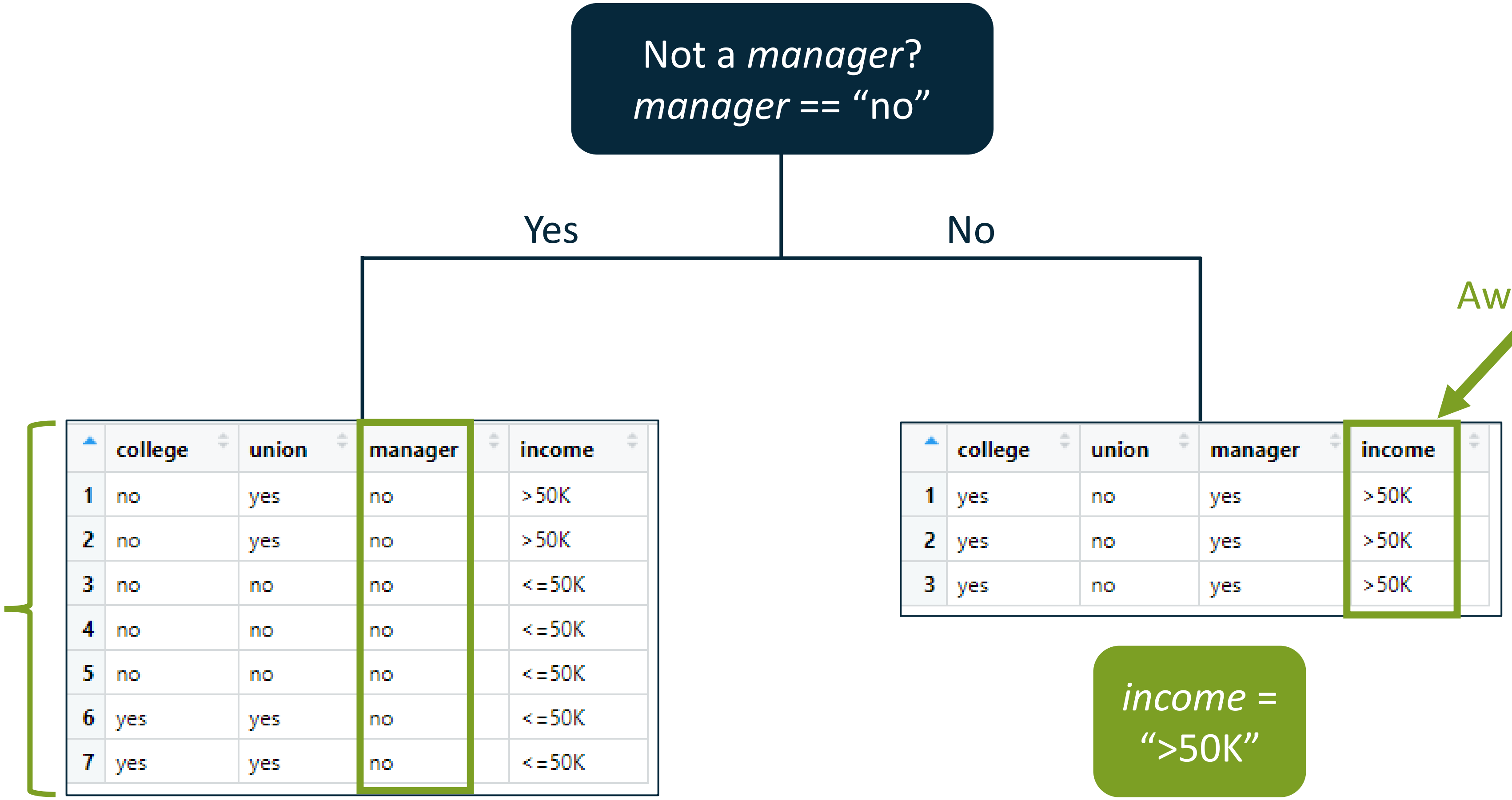
Weighted Gini “no” = (7/10)(0.41) = 0.29

Step #10 – Gini change with *manager*:

Gini Change = (0.5) – (0.0) – (0.29) = 0.21

Using the *manager* feature to split the data helps the most!

The Tree So Far



	college	union	manager	income
1	no	yes	no	>50K
2	no	yes	no	>50K
3	no	no	no	<=50K
4	no	no	no	<=50K
5	no	no	no	<=50K
6	yes	yes	no	<=50K
7	yes	yes	no	<=50K

	college	union	manager	income
1	yes	no	yes	>50K
2	yes	no	yes	>50K
3	yes	no	yes	>50K

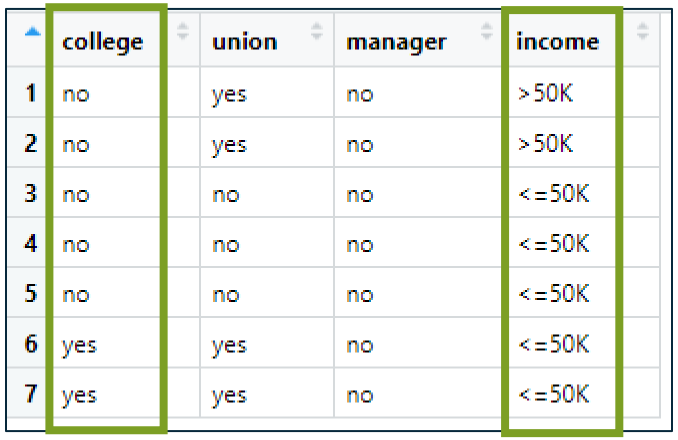
Got some work to do…

Awesome!

income = “>50K”

The *manager* feature is no longer useful

Training Our First Classification Tree



	college	union	manager	income
1	no	yes	no	>50K
2	no	yes	no	>50K
3	no	no	no	<=50K
4	no	no	no	<=50K
5	no	no	no	<=50K
6	yes	yes	no	<=50K
7	yes	yes	no	<=50K

Step #11 – Beginning Gini impurity:

Gini Start = 1 – $(5/7)^2$ – $(2/7)^2$

Gini Start = 1 – (0.51) – (0.08) = 0.41

Step #12 – What's Gini impurity for *college* based on splits?

Gini "yes" = 1 – $(2/2)^2$ – $(0/2)^2$

Gini "yes" = 1 – (1) – (0.0)

Gini "yes" = 0.0

Gini "no" = 1 – $(3/5)^2$ – $(2/5)^2$

Gini "no" = 1 – (0.36) – (0.16)

Gini "no" = 0.48

Step #13 – Weight the *college* splits by observation counts:

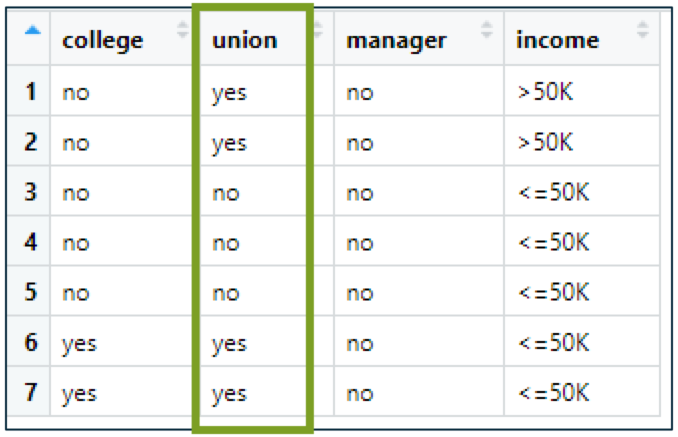
Weighted Gini "yes" = (2/7)(0.0) = 0.0

Weighted Gini "no" = (5/7)(0.48) = 0.34

Step #14 – Gini change with *college*:

Gini Change = (0.41) – (0.0) – (0.34) = 0.07

Training Our First Classification Tree



	college	union	manager	income
1	no	yes	no	>50K
2	no	yes	no	>50K
3	no	no	no	<=50K
4	no	no	no	<=50K
5	no	no	no	<=50K
6	yes	yes	no	<=50K
7	yes	yes	no	<=50K

Step #15 – What’s Gini impurity for *union* based on splits?

Gini “yes” = $1 – (2/4)^2 – (2/4)^2$

Gini “yes” = 1 – (0.25) – (0.25)

Gini “yes” = 0.5

Gini “no” = $1 – (3/3)^2 – (0/0)^2$

Gini “no” = 1 – (1) – (0)

Gini “no” = 0.0

Step #16 – Weight the *union* splits by observation counts:

Weighted Gini “yes” = (4/7)(0.5) = 0.29

Weighted Gini “no” = (3/7)(0.0) = 0.0

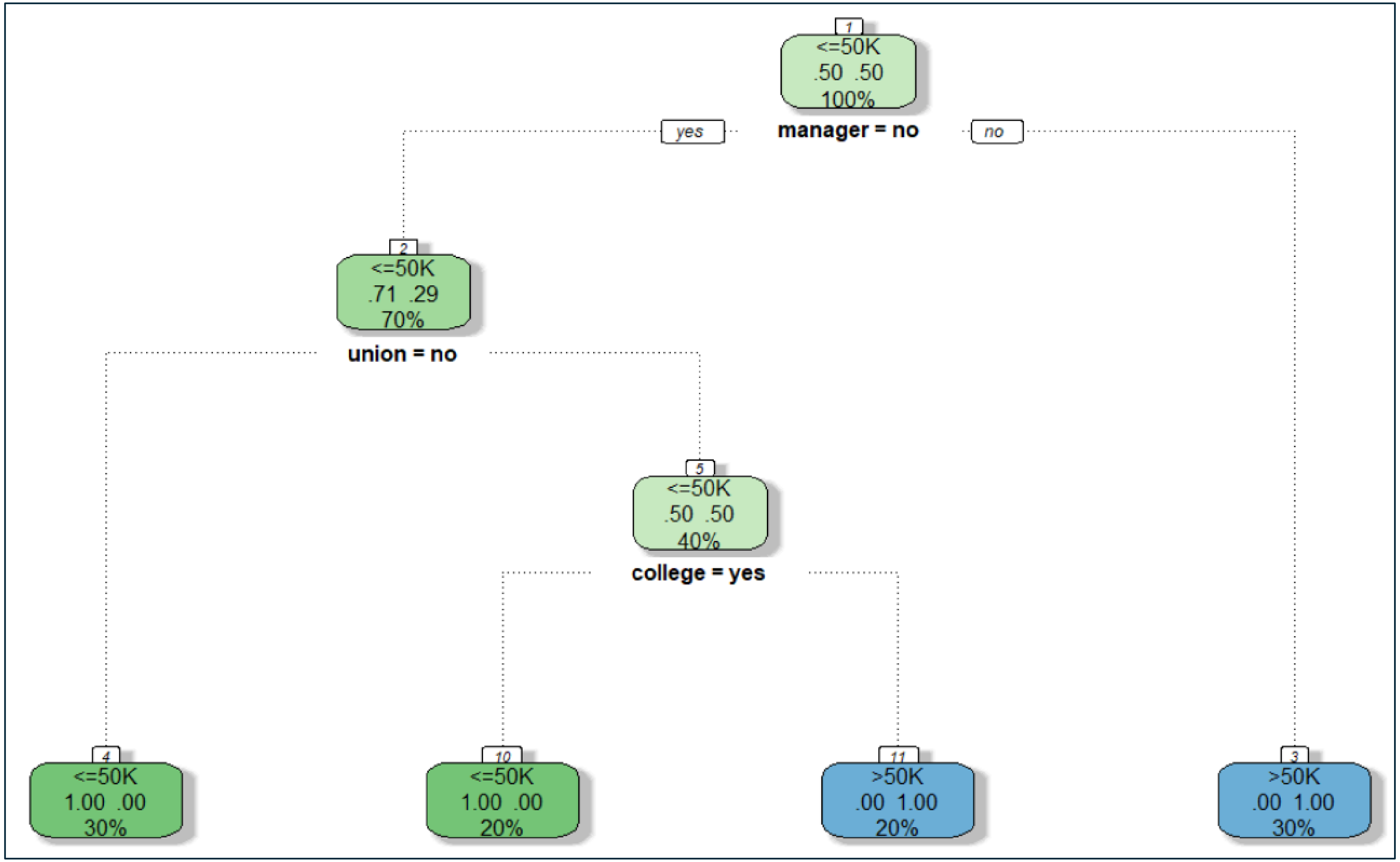
Step #17 – Gini change with *union*:

Gini Change = (0.41) – (0.29) – (0.0) = 0.12

Using the *union* feature to split the data helps the most!

The Real Deal

A classification tree trained in R using the contrived data of this example…



Many Categories Impurity

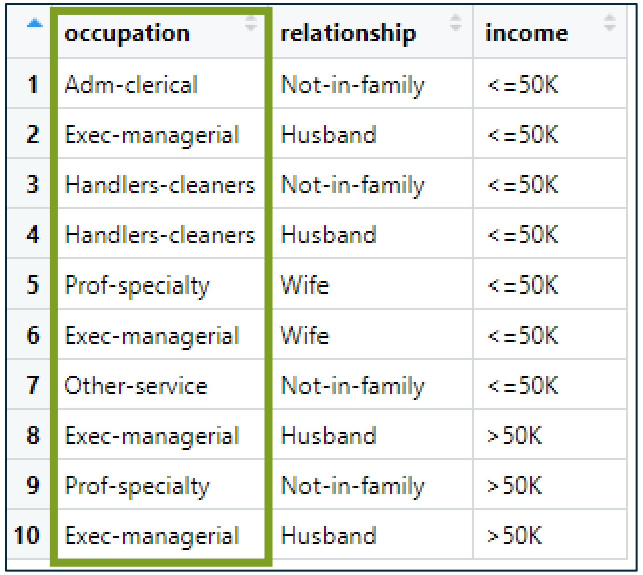
So Many Possibilities

As we know, the CART algorithm uses 2-way (i.e., *binary*) data splits.

So how does this map to a classification tree where a feature has many categories?

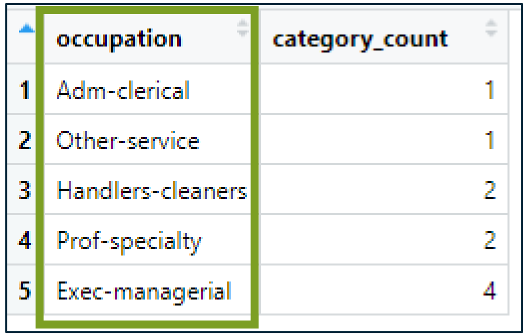
For example…

Pivoting the *occupation* feature…



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

5 categories…



	occupation	category_count
1	Adm-clerical	1
2	Other-service	1
3	Handlers-cleaners	2
4	Prof-specialty	2
5	Exec-managerial	4

So many possibilities!

IF *occupation* IS IN (“Adm-clerical”)

IF *occupation* IS IN (“Adm-clerical”, “Other-service”)

IF *occupation* IS IN (“Adm-clerical”, “Other-service”, “Handlers-cleaners”)

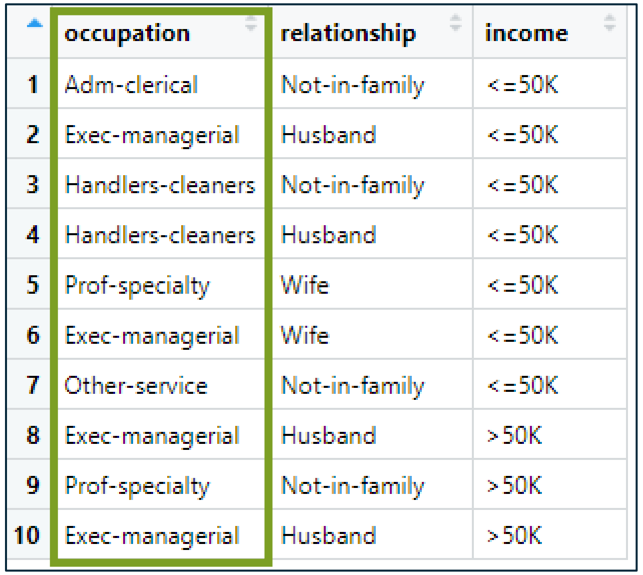
IF *occupation* IS IN (“Adm-clerical”, “Other-service”, “Handlers-cleaners”, “Prof-specialty”)

CART Is Smart

As the number of categories increases, the number of possible binary splits explodes.

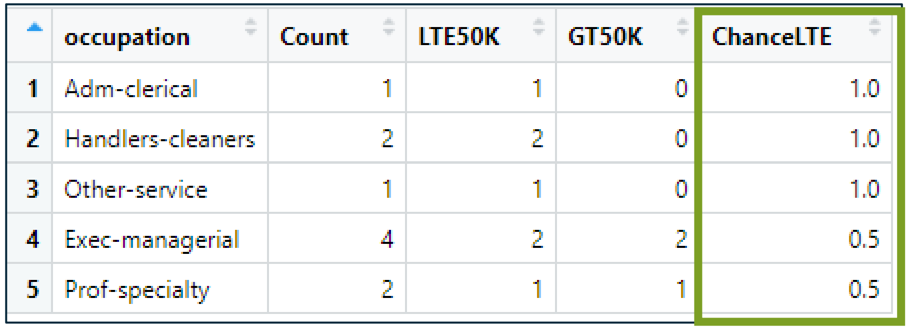
Luckily, CART is smart and uses an optimization…

Pivoting the *occupation* feature…



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

Sort the categories by the chance of income being "<=50K"…



	occupation	Count	LTE50K	GT50K	ChanceLTE
1	Adm-clerical	1	1	0	1.0
2	Handlers-cleaners	2	2	0	1.0
3	Other-service	1	1	0	1.0
4	Exec-managerial	4	2	2	0.5
5	Prof-specialty	2	1	1	0.5

Calculate the Gini change by adding each category in turn…

IF *occupation* IS IN ("Adm-clerical")

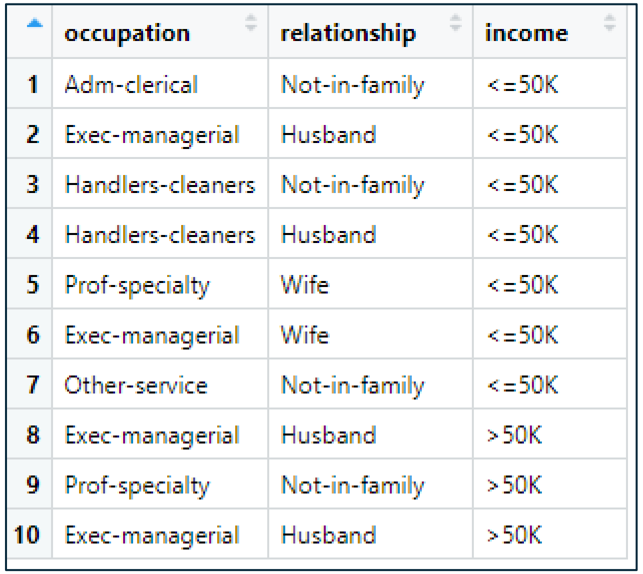
IF *occupation* IS IN ("Adm-clerical", "Handlers-cleaners")

IF *occupation* IS IN ("Adm-clerical", "Handlers-cleaners", "Other-service")

CART only needs to calculate the Gini change for a tiny subset of possible splits!

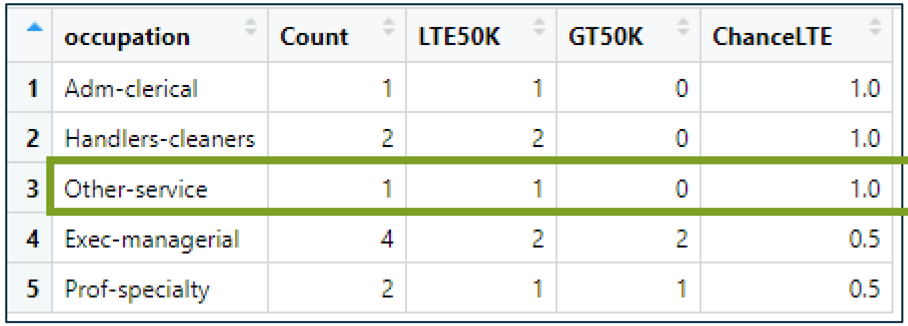
An Example

As an example, take the first split in a classification tree *trained* from the following data…



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

Gini Start = 0.42



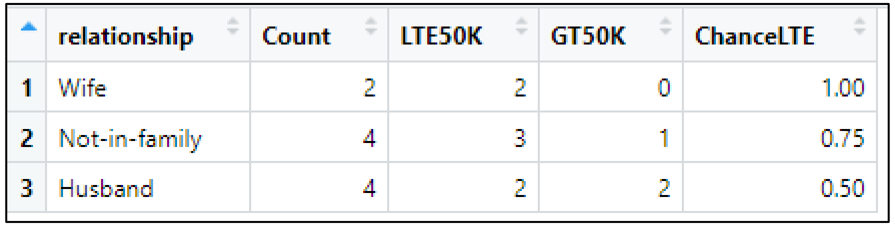
	occupation	Count	LTE50K	GT50K	ChanceLTE
1	Adm-clerical	1	1	0	1.0
2	Handlers-cleaners	2	2	0	1.0
3	Other-service	1	1	0	1.0
4	Exec-managerial	4	2	2	0.5
5	Prof-specialty	2	1	1	0.5

Gini Change = 0.02

Gini Change = 0.08

Gini Change = 0.12

Gini Change = 0.06



	relationship	Count	LTE50K	GT50K	ChanceLTE
1	Wife	2	2	0	1.00
2	Not-in-family	4	3	1	0.75
3	Husband	4	2	2	0.50

Gini Change = 0.05

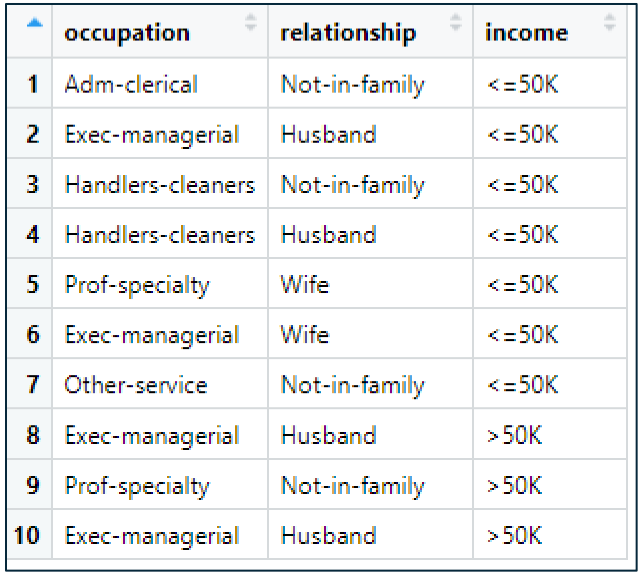
Gini Change = 0.05

The best first split for the tree:

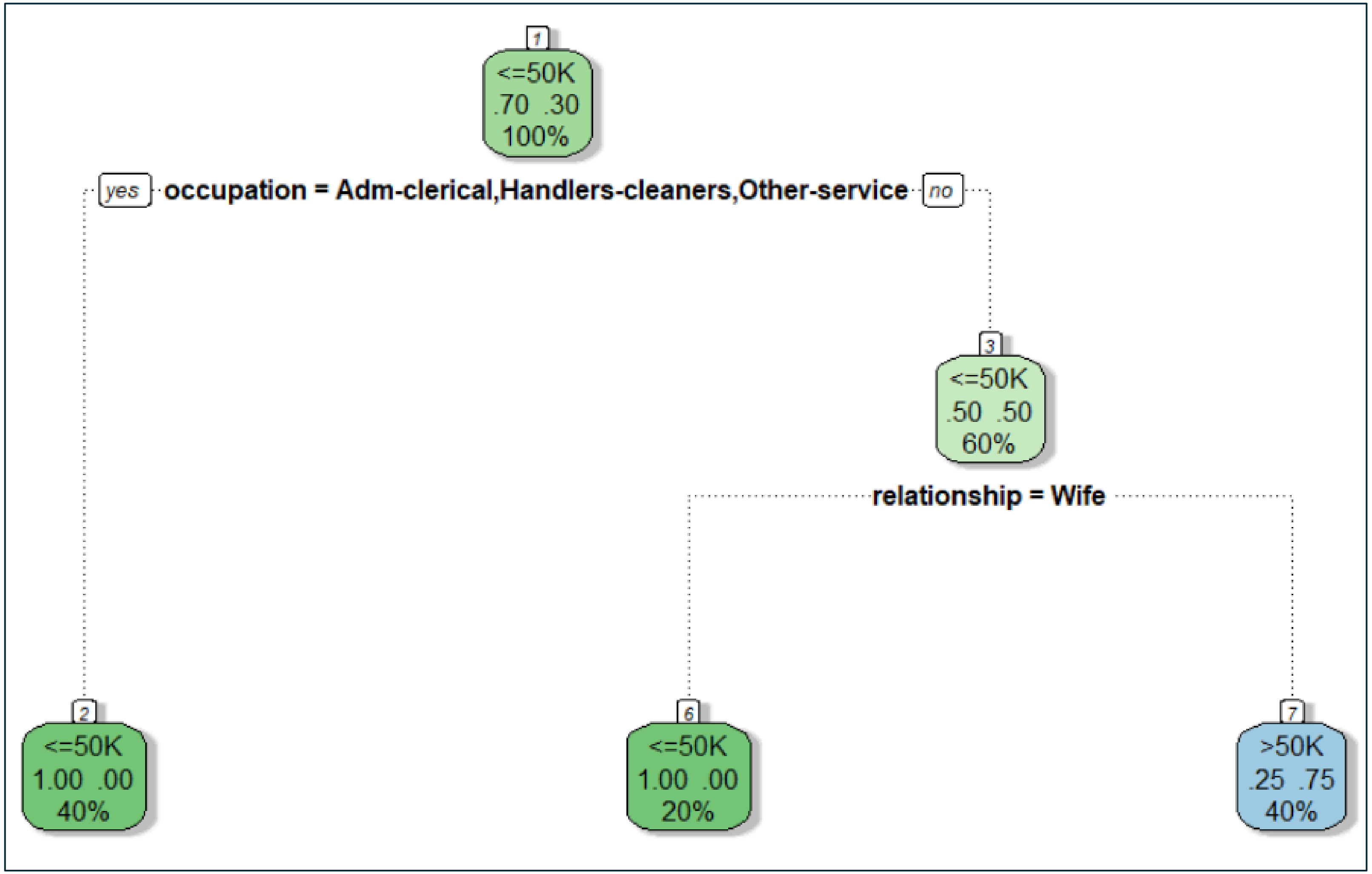
IF *occupation* IS IN (“Adm-clerical”, “Handlers-cleaners”, “Other-service”)

The Real Deal

Check out a classification tree trained in R…



	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K



Numeric Feature Impurity

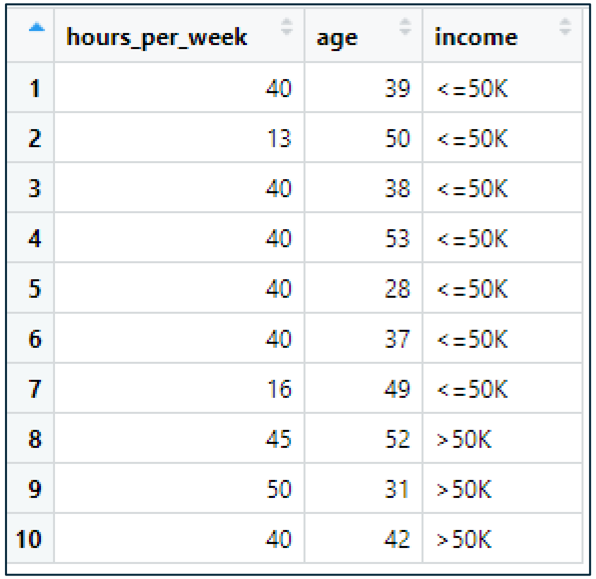
Again, So Many Possibilities

We learned in the last lesson how the CART algorithm finds the best split for features with many categories.

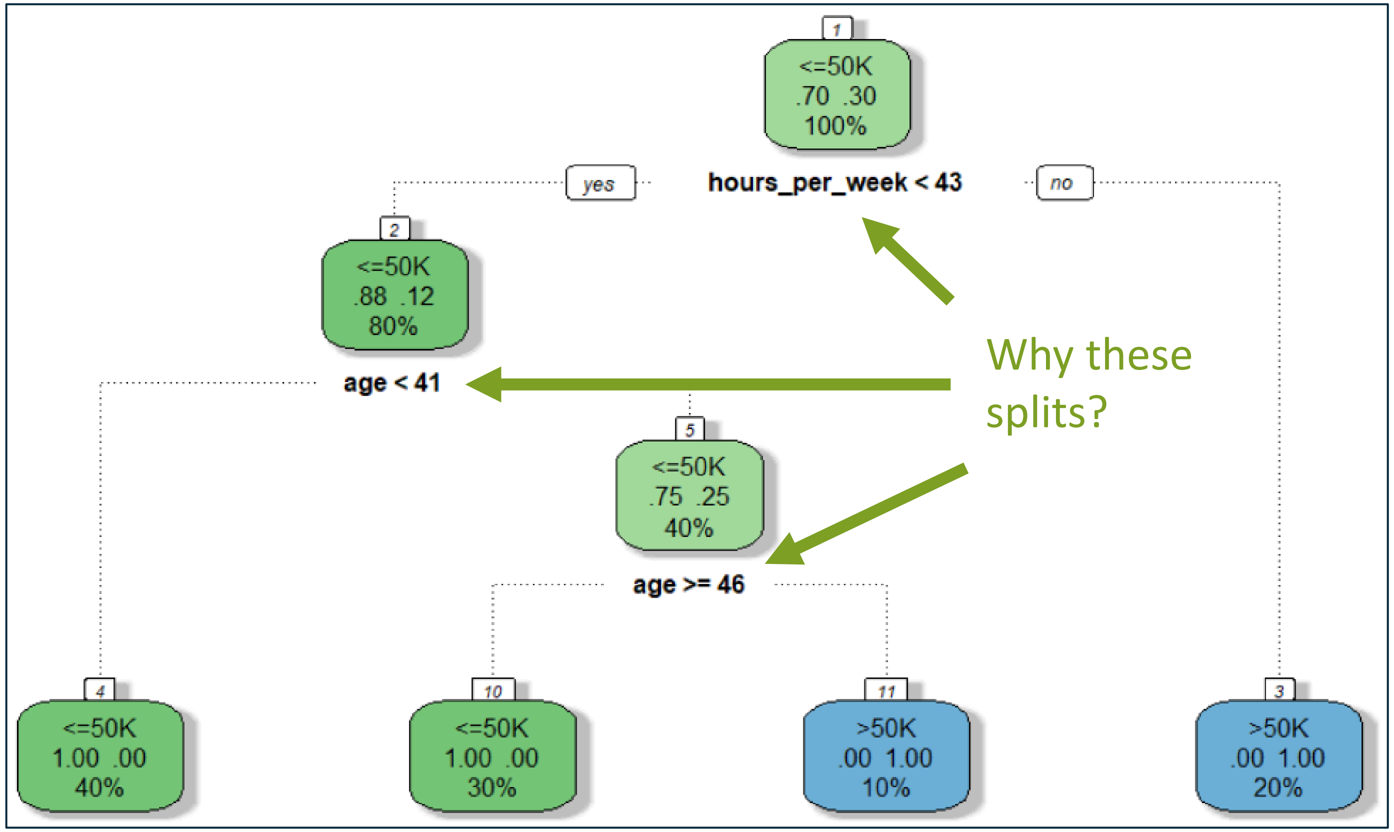
Think about numeric features. Numeric features often have many unique values.

Take the following example from the *Adult Census* data…

Using this data, R trains the following tree…



	hours_per_week	age	income
1	40	39	<=50K
2	13	50	<=50K
3	40	38	<=50K
4	40	53	<=50K
5	40	28	<=50K
6	40	37	<=50K
7	16	49	<=50K
8	45	52	>50K
9	50	31	>50K
10	40	42	>50K



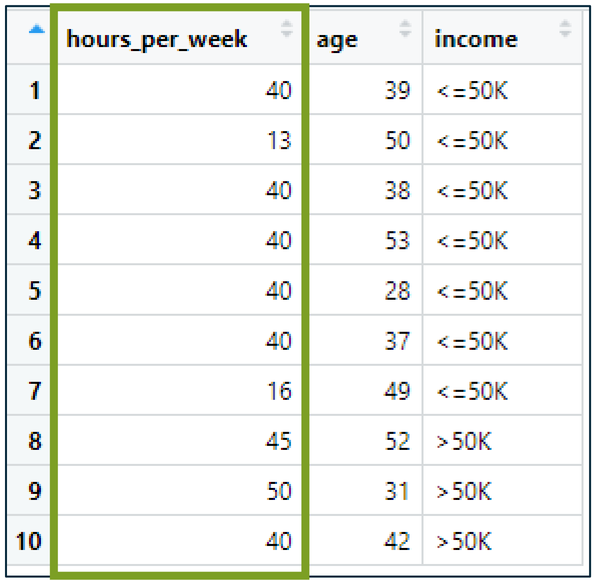
Why these splits?

Again, CART Is Smart

In real data sets, numeric features can have many unique values.

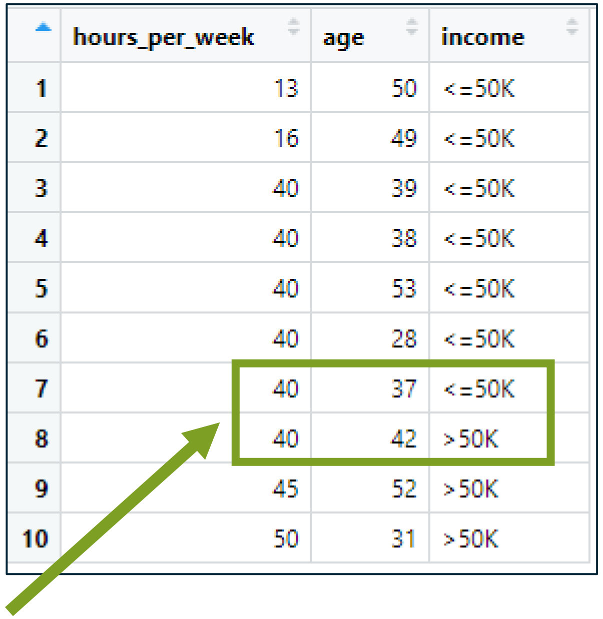
Once again, CART is smart and uses an optimization to find optimal splits…

Sort the *hours_per_week* feature…



	hours_per_week	age	income
1	40	39	<=50K
2	13	50	<=50K
3	40	38	<=50K
4	40	53	<=50K
5	40	28	<=50K
6	40	37	<=50K
7	16	49	<=50K
8	45	52	>50K
9	50	31	>50K
10	40	42	>50K

Find the observations where the label changes…

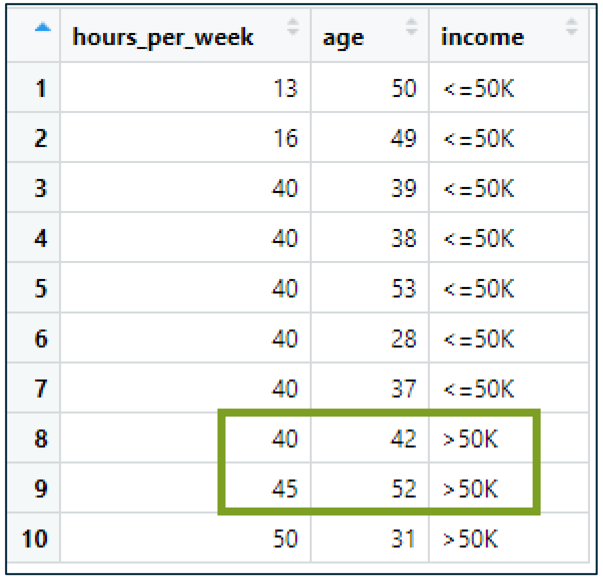


	hours_per_week	age	income
1	13	50	<=50K
2	16	49	<=50K
3	40	39	<=50K
4	40	38	<=50K
5	40	53	<=50K
6	40	28	<=50K
7	40	37	<=50K
8	40	42	>50K
9	45	52	>50K
10	50	31	>50K

Calculate Gini change by splitting between the two values

Doh! Values are the same!

Move split point down to where values change…



	hours_per_week	age	income
1	13	50	<=50K
2	16	49	<=50K
3	40	39	<=50K
4	40	38	<=50K
5	40	53	<=50K
6	40	28	<=50K
7	40	37	<=50K
8	40	42	>50K
9	45	52	>50K
10	50	31	>50K

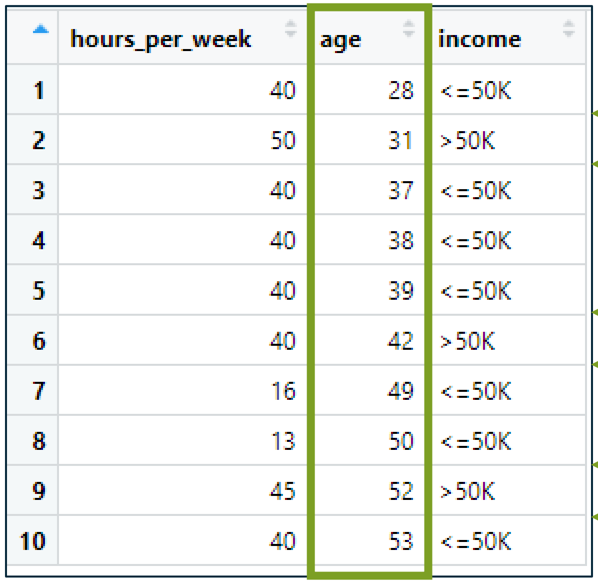
Split = (40 + 45) / 2 = 42.5 = 43

Gini Change = 0.245

Continuing…

Sort the *age* feature…

Possible splits…



	hours_per_week	age	income
1	40	28	<=50K
2	50	31	>50K
3	40	37	<=50K
4	40	38	<=50K
5	40	39	<=50K
6	40	42	>50K
7	16	49	<=50K
8	13	50	<=50K
9	45	52	>50K
10	40	53	<=50K

Gini Change = 0.02

Gini Change = 0.02

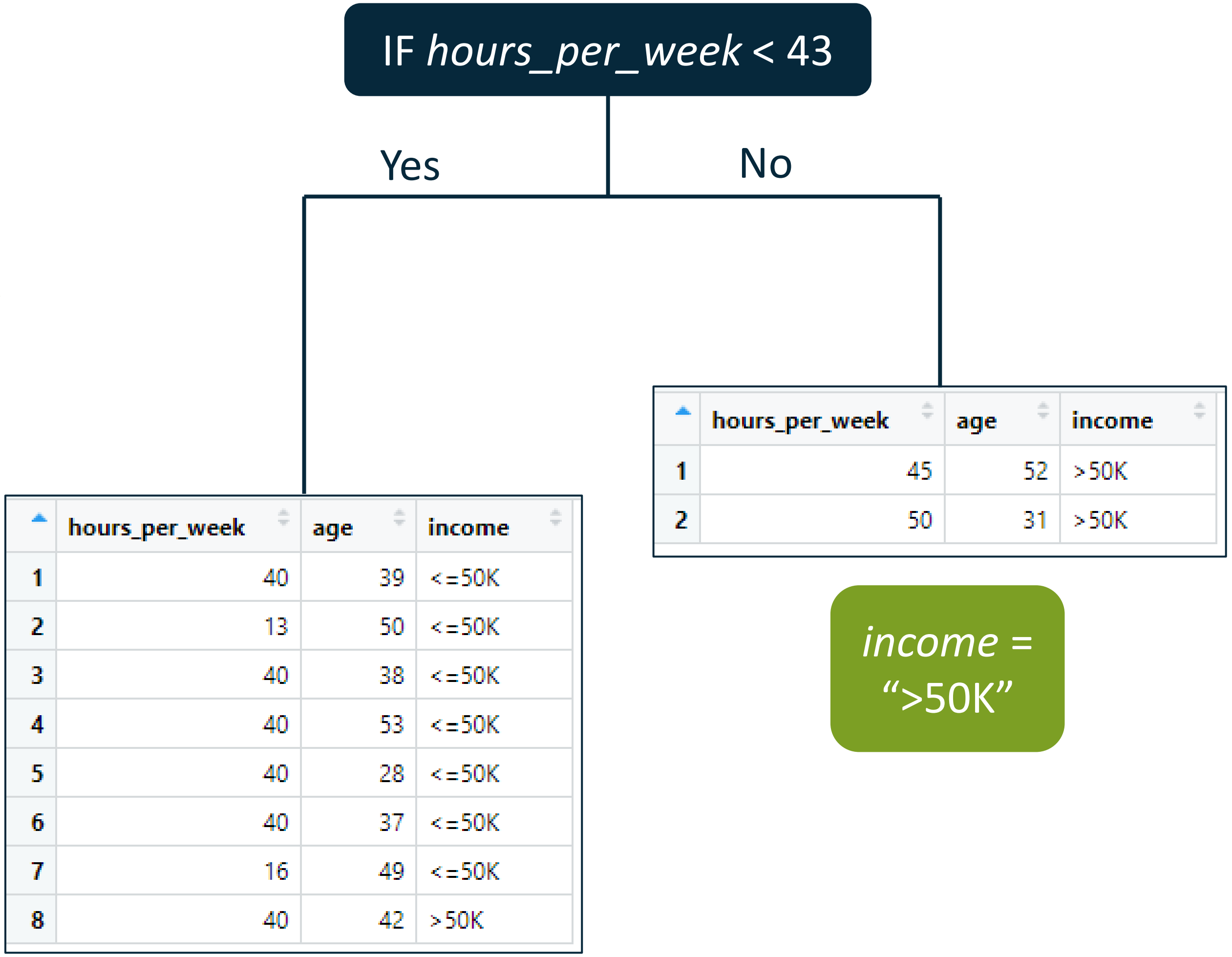
Gini Change = 0.02

Gini Change = 0.003

Gini Change = 0.02

Gini Change = 0.02

The *hours_per_week* split is optimal…



IF *hours_per_week* < 43

Yes

No

	hours_per_week	age	income
1	40	39	<=50K
2	13	50	<=50K
3	40	38	<=50K
4	40	53	<=50K
5	40	28	<=50K
6	40	37	<=50K
7	16	49	<=50K
8	40	42	>50K

	hours_per_week	age	income
1	45	52	>50K
2	50	31	>50K

income = “>50K”

Process continues with left side of tree using the *age* feature.