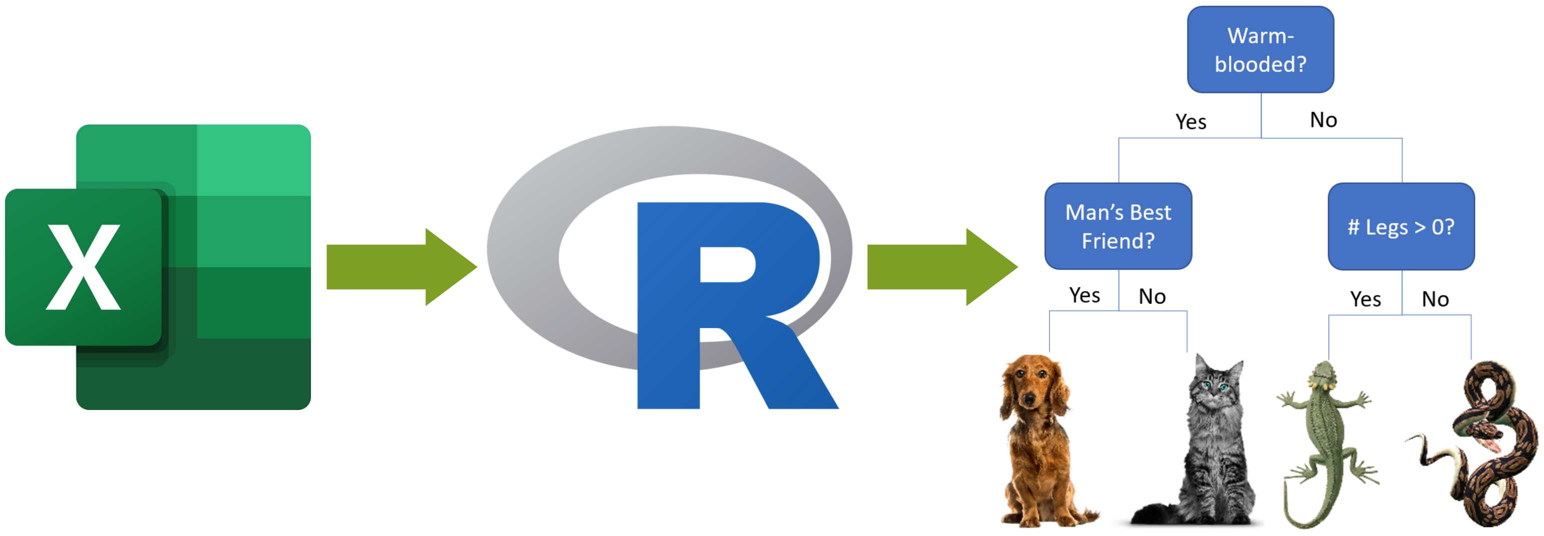
FROM EXCEL TO MACHINE LEARNING BOOTCAMP



YOUR TEAM'S EXCEL SKILLS WILL UNLOCK ADVANCED ANALYTICS

Dave ON DATA

Introduction

I am a hands-on analytics professional and experienced leader. I mention this only to provide credibility for what I am about to say next.

Over the years I have performed analytics on behalf of teams in Marketing, Product Management, Finance, Customer Service, and Supply Chain.

In the majority of cases, these analyses could be performed by the business subject matter experts (SMES). The techniques I use are not that hard to learn. This includes business SMEs acquiring machine learning skills to analyze their data.

I know. It sounds a bit crazy.

My philosophy is to let my content do the talking - which is why this document exists.

This document will demonstrate the following:

1. Any business team with Excel skills can learn to analyze data with R programming.
2. Any business team with R skills can acquire valuable machine learning skills.

Analyzing data with machine learning is applicable to all aspects of business:

[Human Resources] - What factors are highly predictive of bad attrition?
[Product Management] - What feature usage(s) are highly predictive of a sticky SaaS customer?
[Customer Service] - Are there factors related to churn that customer service can address?
[Marketing] - What are the customer journey behaviors that predict conversion?
[Supply Chain] - What are the factors/behaviors that predict customer returns?

If your team wants to have more business impact using data, I invite you to keep reading.

-Dave

Document Goals

This document is structured to accomplish the following goals:

1. To demonstrate that any business team with Excel skills can learn R programming.
2. To demonstrate that any business team can acquire machine learning skills.
3. To provide an outline of the 3-day bootcamp curriculum

Per the above, the document is structured as follows:

- Part 1 illustrates how Excel knowledge maps directly to R programming.
- Part 2 illustrates how approachable machine learning is to any business team.
- Part 3 provides the outline of topics for the 3-day bootcamp.

A secondary benefit of this document is that you will acquire some actual machine learning knowledge as the result of your reading.

Table of Contents

Table of Contents

Table of Contents

Excel Users Write Code

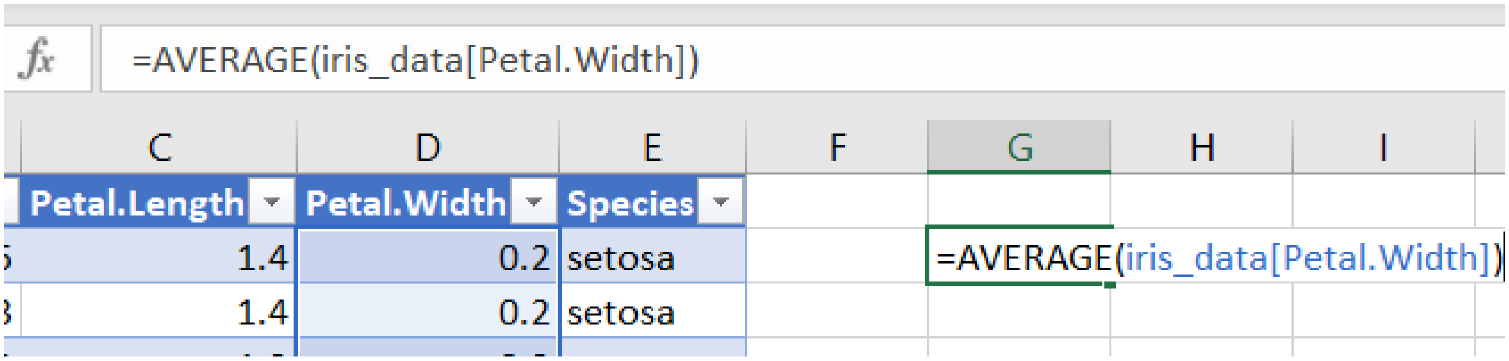
Excel Code

While most Excel users don't think of it this way, they spend a lot of time writing and debugging code in Excel. In fact, Microsoft Excel is by far and away the world's most popular programming environment.

Take the image below as an example. The user is using Excel's *AVERAGE* function to calculate the average of a column (i.e., *Petal.Width*) of a table (i.e., *iris_data*) in a worksheet.

Once the user hits the <*enter*> key, Excel attempts to interpret the instructions in the cell and perform the desired operation. If Excel doesn't understand what the user typed, it reports an error.

That's coding!



```
=AVERAGE(iris_data[Petal.Width])
```

C	D	E	F	G	H	I
Petal.Length	Petal.Width	Species				
1.4	0.2	setosa		=AVERAGE(iris_data[Petal.Width])		
1.4	0.2	setosa				

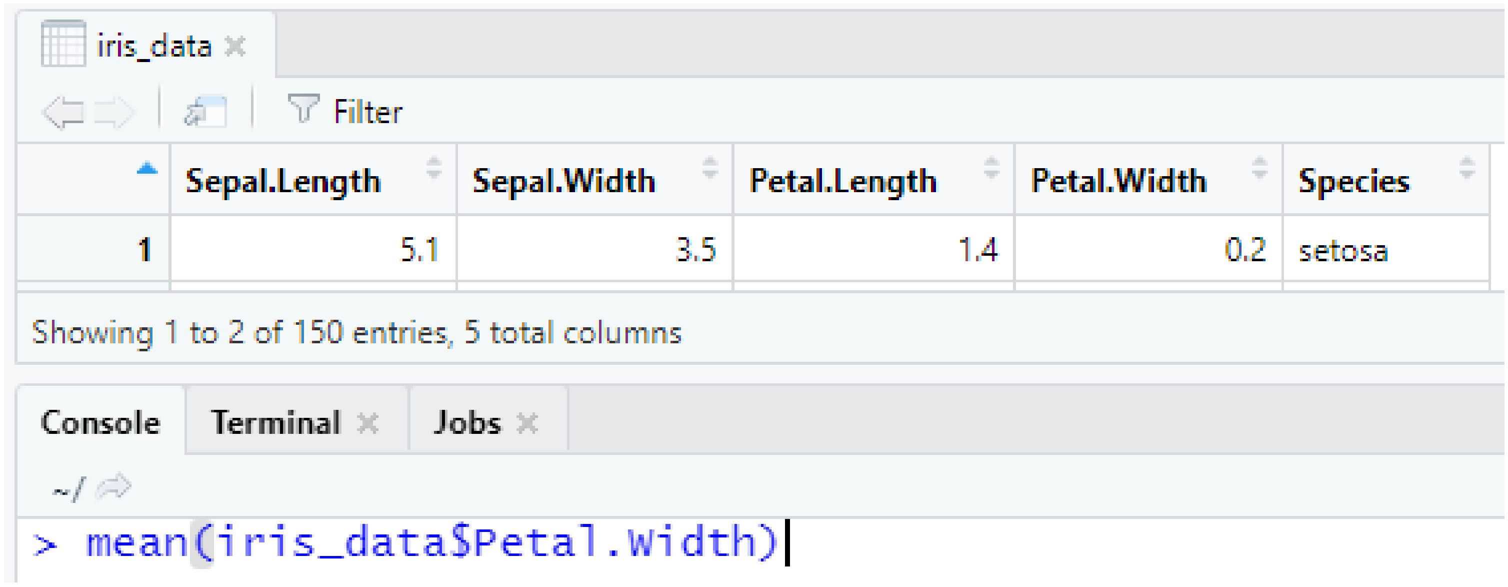
R Code

While it isn't the only way to code in Excel, calling Excel functions as depicted on the previous page is by far the most common. When using Excel in this way, Excel is operating as a code *interpreter*.

Using R as an interpreter is very common. The R user types some code and hits the <*enter*> key. R then tries to interpret the code, throwing an error if doesn't understand what was typed by the user.

In this way Excel and R are very similar, but it doesn't stop there. Even the code is very similar!

The image below depicts the same scenario as on the previous page, but using R instead. As depicted, the user is calculating the average (*mean* is just another name for the average) of the *Petal.Width* column of the *iris_data* table.



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa

Showing 1 to 2 of 150 entries, 5 total columns

```
> mean(iris_data$Petal.Width)
```

Excel Skills Directly Translate to R

While this example might seem simple, it demonstrates why R is the fastest, easiest way for ANY team to unlock advanced analytics.

As you will see through the rest of this document, Excel is a powerful analytical tool with many concepts and skills that need to be mastered to use Excel effectively.

This knowledge is directly applicable to R.

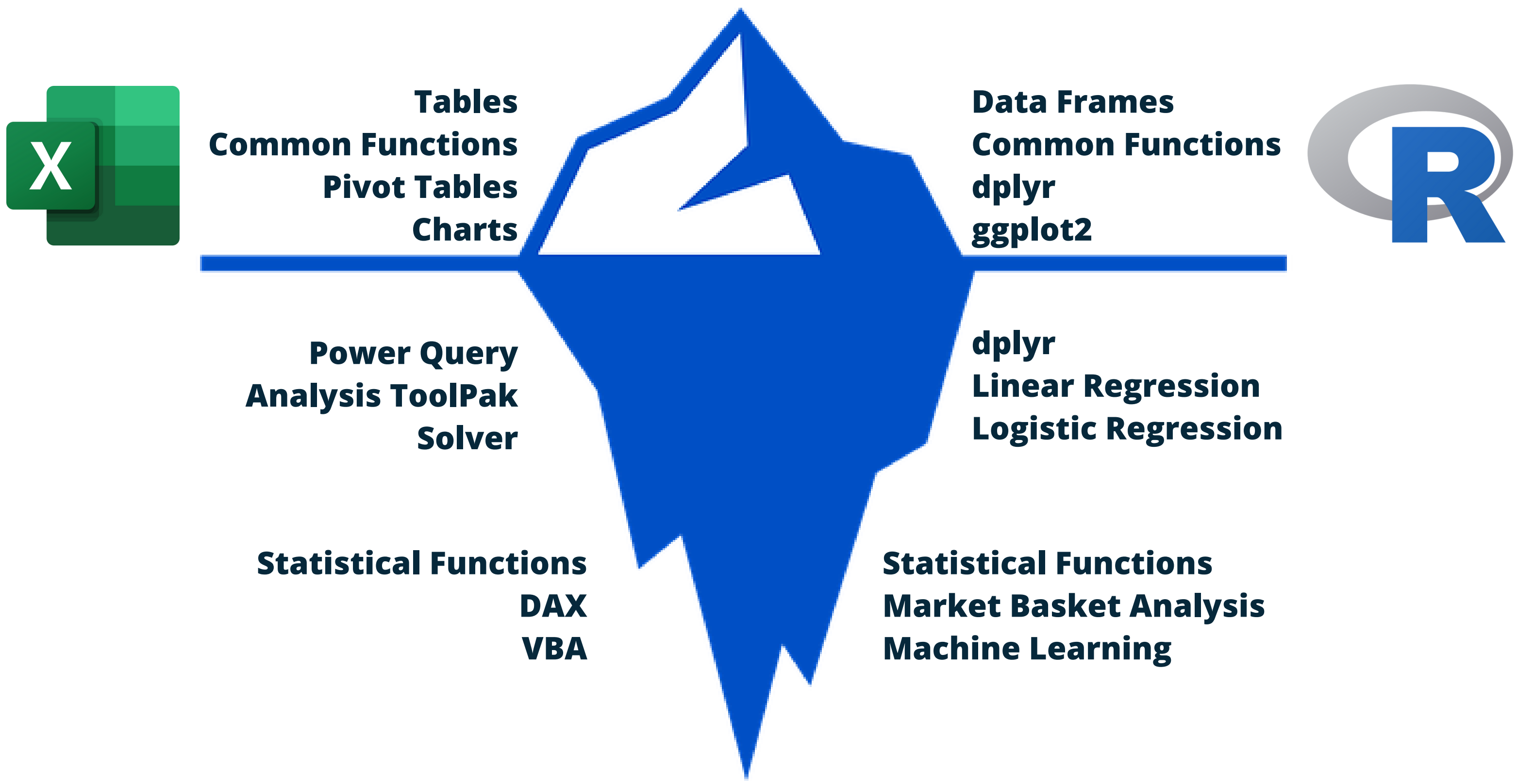
This knowledge makes the learning process an exercise in mapping Excel skills to R.

Excel and R Are Like Icebergs

While cliche, the iceberg metaphor is a powerful way to conceptualize the extent Excel knowledge maps to R.

As depicted below, Excel features above the water line (e.g., Pivot Tables) only scratch the surface of Excel's capabilities. However, these feature represent the bulk of Excel's use in practice.

Another similarity between Excel and R is the "choose your own adventure" aspect of the technologies. Just as many Excel users never learn Power Query, not every R user needs to learn statistical analysis to be effective in their work.



It's All About the Tables

Tables Are Key

Using Microsoft Excel is all about working with tables of data. You filter tables, you sort tables, you pivot tables, etc. You can think of *tables* as one of the fundamental *objects* in Excel that you create and manipulate to conduct analyses.

Excel tables can also be thought of as *container objects*. Tables contain *rows*, *columns*, *cells*, *data formats*, etc.

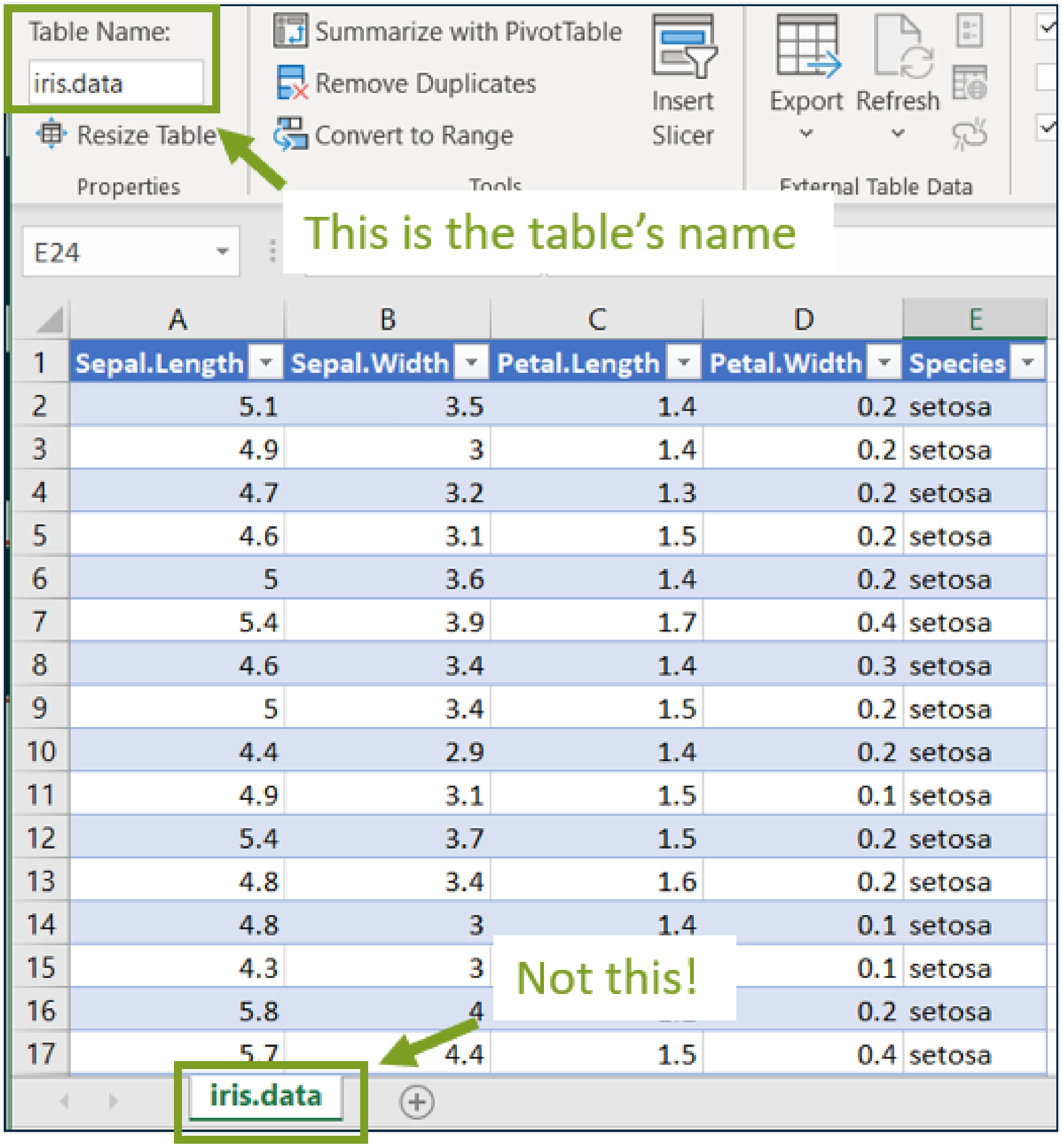
You probably can see where I'm going with this already.

When analyzing data with R, it's all about the tables - just like Excel.

Once again, your Excel knowledge directly translates to R.

Tables in Excel and R

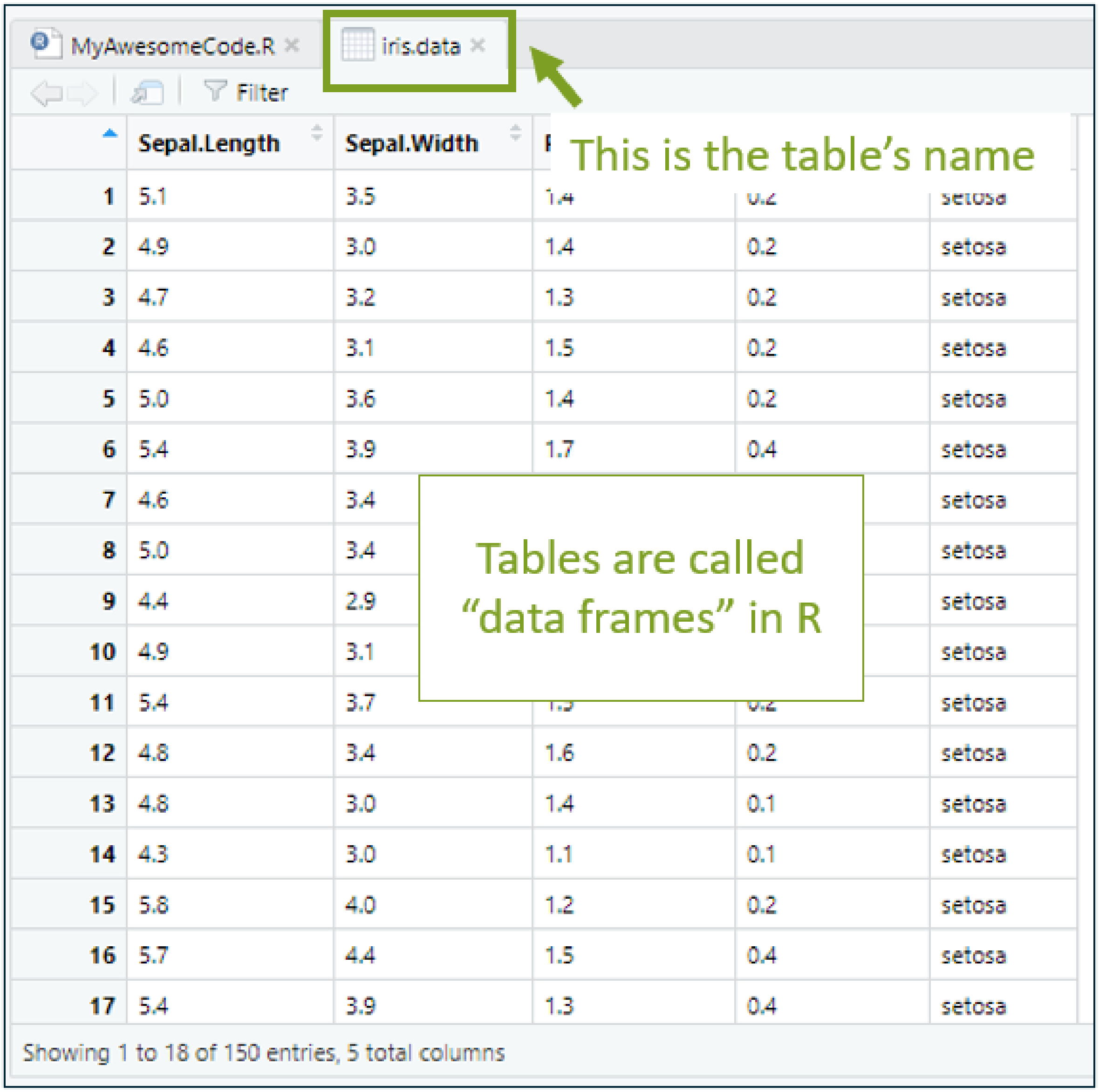
The image below demonstrates how Excel *tables* are *objects*. For example, every table in Excel has a name - whether you explicitly name a table or not. Table names allow you to directly access/manipulate tables using Excel code.



	A	B	C	D	E
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa
12	5.4	3.7	1.5	0.2	setosa
13	4.8	3.4	1.6	0.2	setosa
14	4.8	3	1.4	0.1	setosa
15	4.3	3		0.1	setosa
16	5.8	4		0.2	setosa
17	5.7	4.4	1.5	0.4	setosa

Things work in R exactly the same way. Tables of data in R (known as "*data frames*") have names just like Excel tables so that you can write R code to access/manipulate tables of data.

[Excel Code] *=AVERAGE(iris.data[Petal.Width])*
[R Code] *mean(iris.data$Petal.Width)*

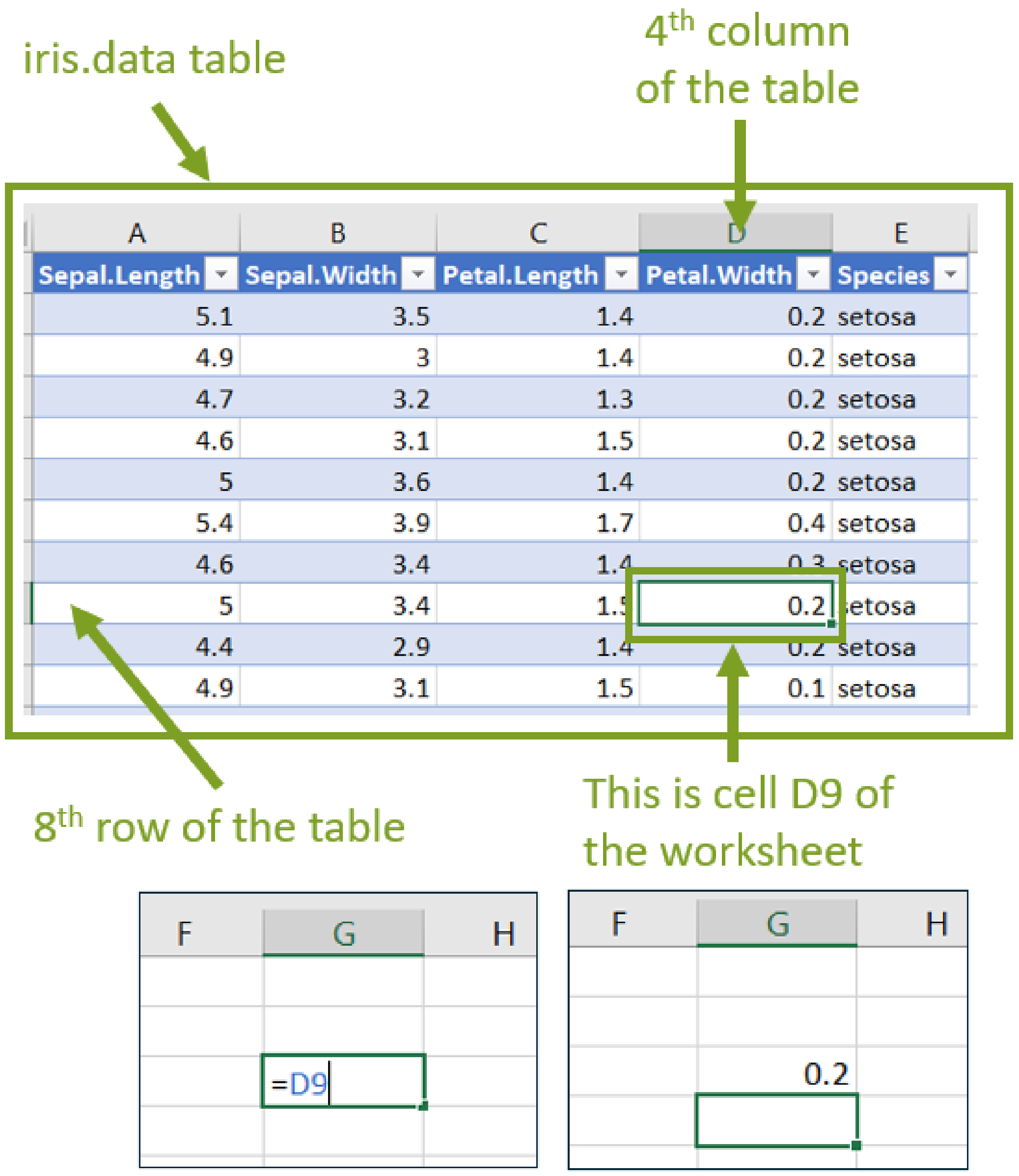


	Sepal.Length	Sepal.Width	P		
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	[illegible]	[illegible]	setosa
8	5.0	3.4	[illegible]	[illegible]	setosa
9	4.4	2.9	[illegible]	[illegible]	setosa
10	4.9	3.1	[illegible]	[illegible]	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa

Showing 1 to 18 of 150 entries, 5 total columns

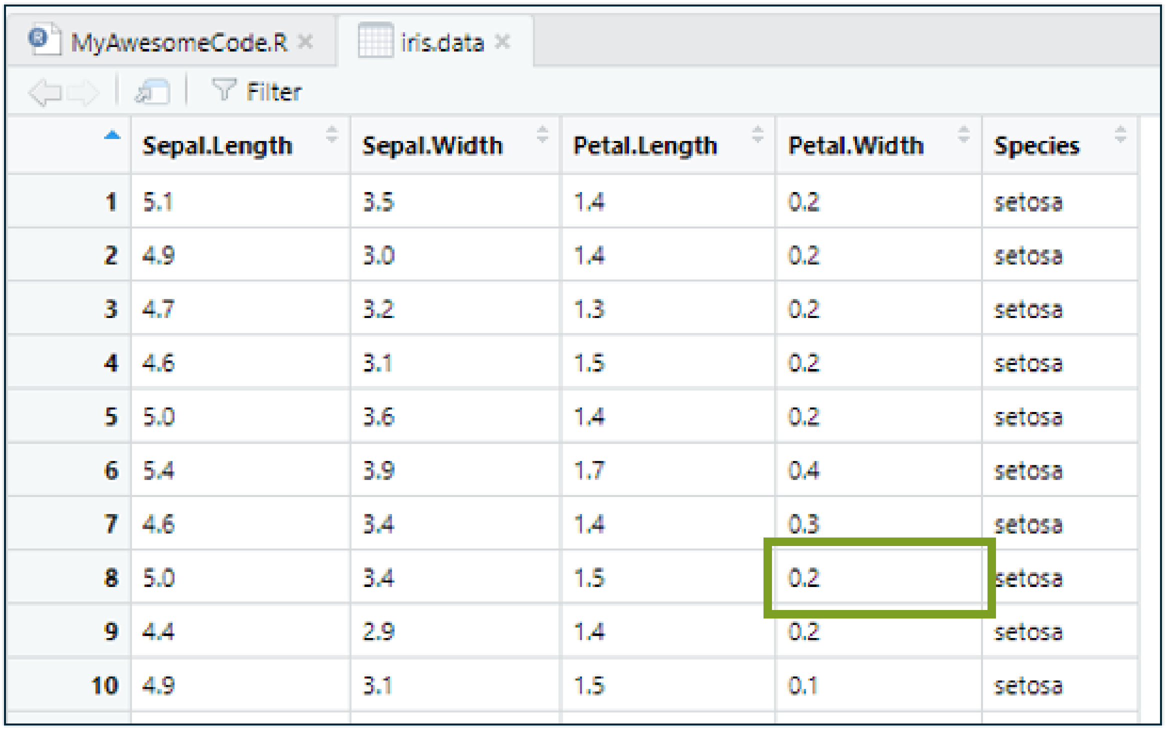
Cells of Data in Excel and R

Working with *cells* of data is very common in Excel. It is useful to think of *cells* as *objects* contained within *tables* - as depicted below. Once again, you use Excel code to access *cells*.

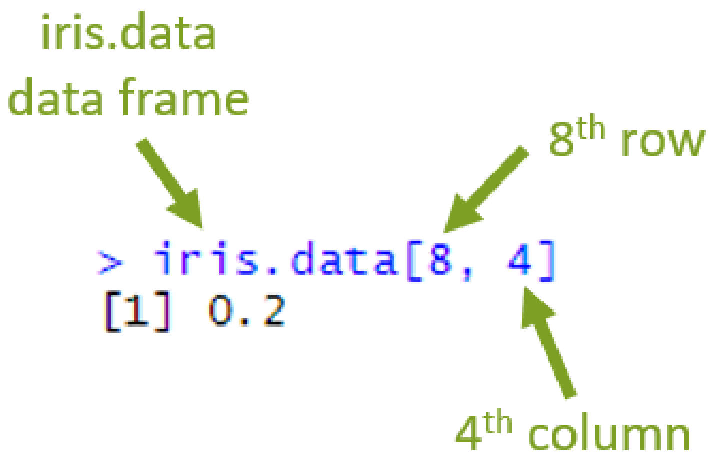


Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Although the R code to access/manipulate *cells* of data is slightly different, conceptually it matches what Excel users do all the time.



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

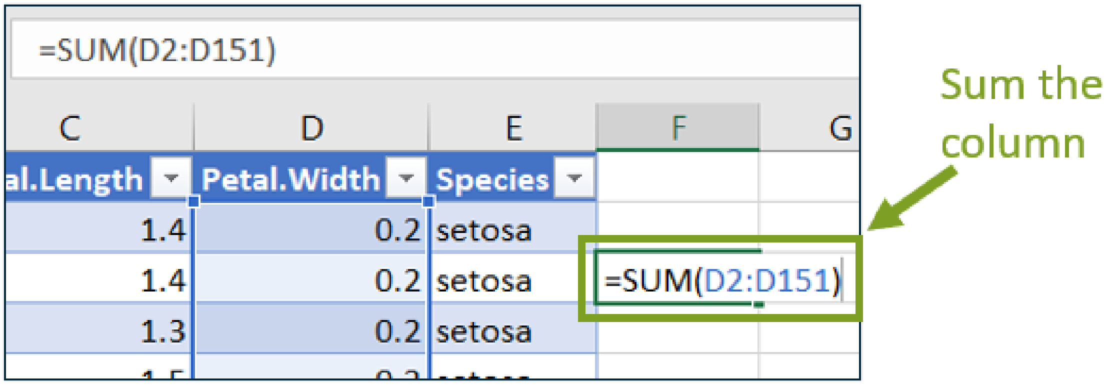


```
> iris.data[8, 4]
[1] 0.2
```

Columns of Data in Excel and R

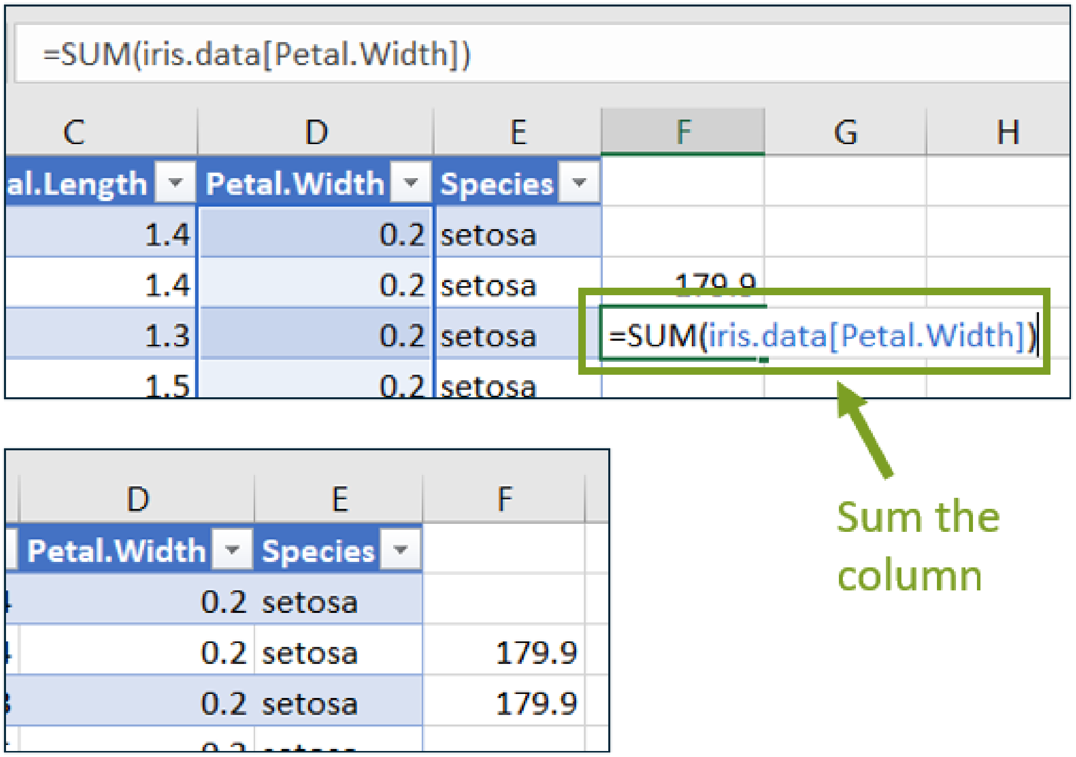
Excel code supports different ways of accessing *columns* of data within *tables*. Two examples:

- Providing a worksheet-based *cell range*
- Providing *object* names



```
=SUM(D2:D151)
```

C	D	E	F	G
al.Length	Petal.Width	Species		
1.4	0.2	setosa		
1.4	0.2	setosa	=SUM(D2:D151)	
1.3	0.2	setosa		



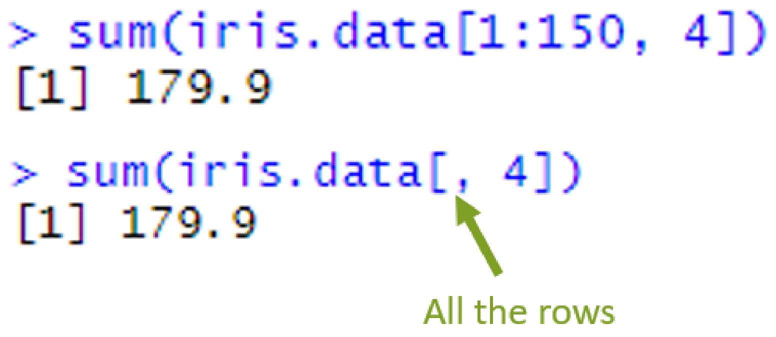
```
=SUM(iris.data[Petal.Width])
```

C	D	E	F	G	H
al.Length	Petal.Width	Species			
1.4	0.2	setosa			
1.4	0.2	setosa	179.9		
1.3	0.2	setosa	=SUM(iris.data[Petal.Width])		
1.5	0.2	setosa			

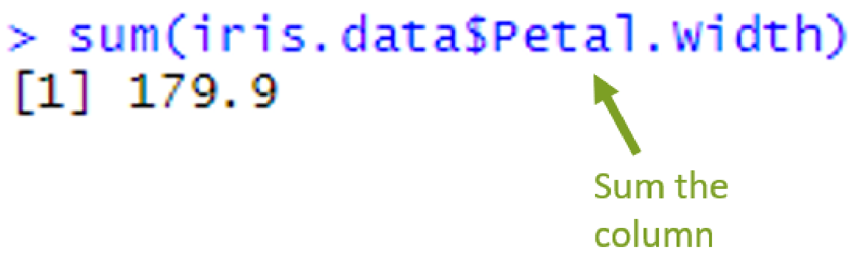
D	E	F
Petal.Width	Species	
0.2	setosa	
0.2	setosa	179.9
0.2	setosa	179.9

Once again, Excel knowledge maps directly to R code as illustrated below.

Notice how similar the actual R code is to Excel when using *object* names.



```
> sum(iris.data[1:150, 4])
[1] 179.9
> sum(iris.data[, 4])
[1] 179.9
```



```
> sum(iris.data$Petal.Width)
[1] 179.9
```

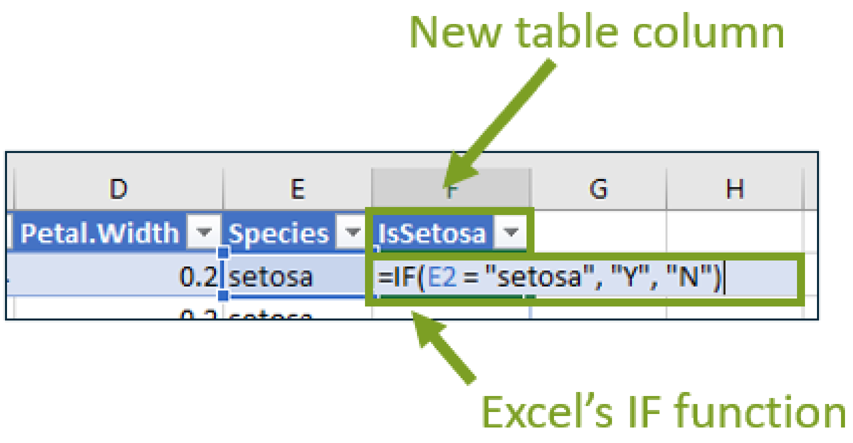
Throw in Some Functions

Using Functions in Excel and R

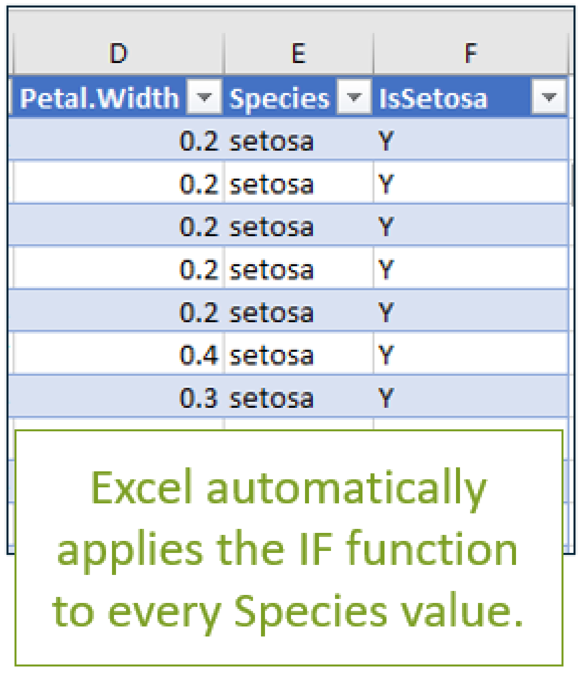
The bulk of code Excel users write call *functions*. Often, these *function* calls are nested and can be difficult to debug (again, that's coding!).

A common Excel scenario is depicted below - creating a new column of data (*IsSetosa*) by invoking a function on another column of data (Species) within the same table (*iris.data*).

In this utterly contrived example, Excel's mighty IF function (a commonly nested Excel function) is being used.



D	E	F	G	H
Petal.Width	Species	IsSetosa		
0.2	setosa	=IF(E2 = "setosa", "Y", "N")		


Excel automatically applies the IF function to every Species value.

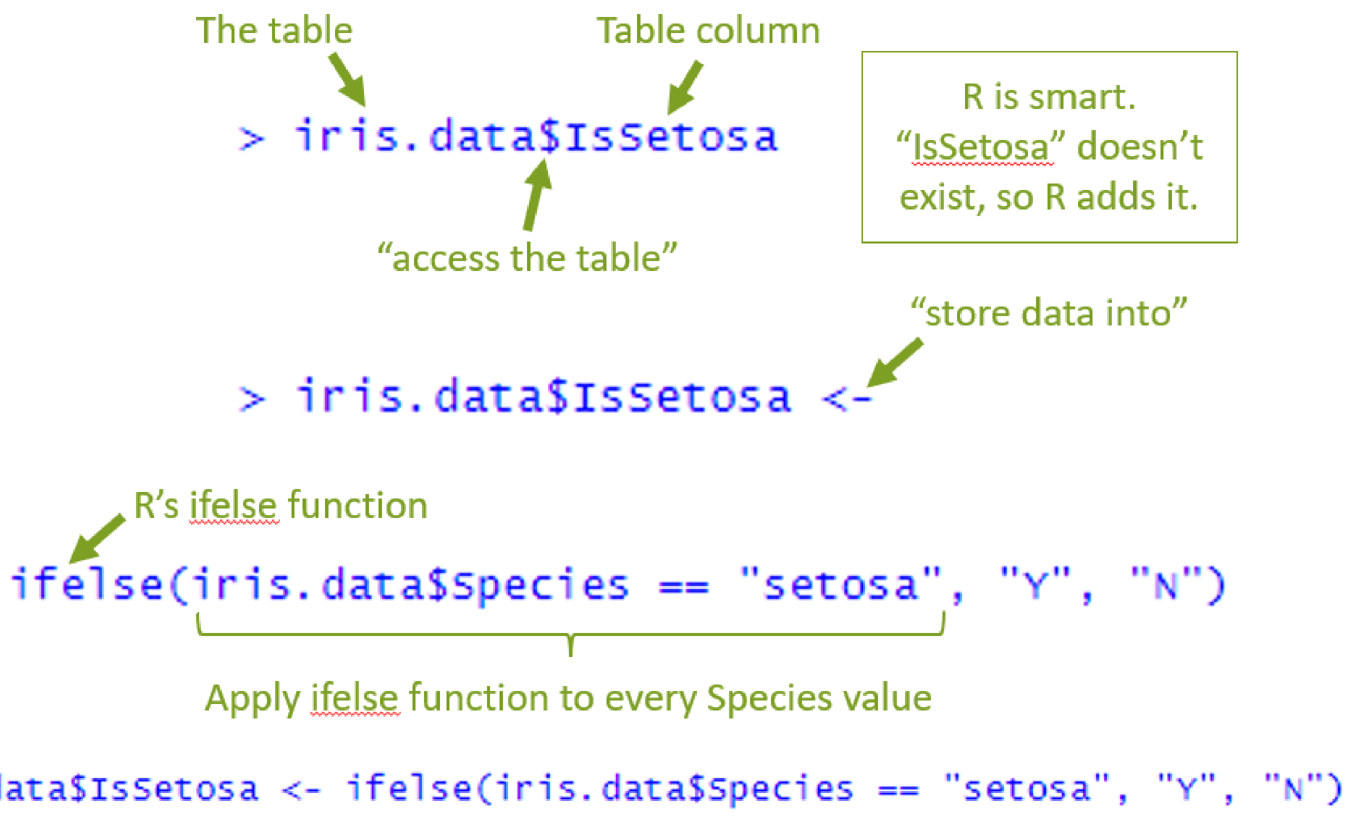
D	E	F
Petal.Width	Species	IsSetosa
0.2	setosa	Y
0.2	setosa	Y
0.2	setosa	Y
0.2	setosa	Y
0.2	setosa	Y
0.4	setosa	Y
0.3	setosa	Y

The contrived example from the previous page is repeated using R code.

A *IsSetosa* column is being added to the *iris.data* table (or *data frame*) and populated with new data derived from the existing *Species* column.

First, notice how the workflow is exactly the same as in Excel - only everything is done in code with R.

Second, notice how similar the R *ifelse* function call is to Excel code.

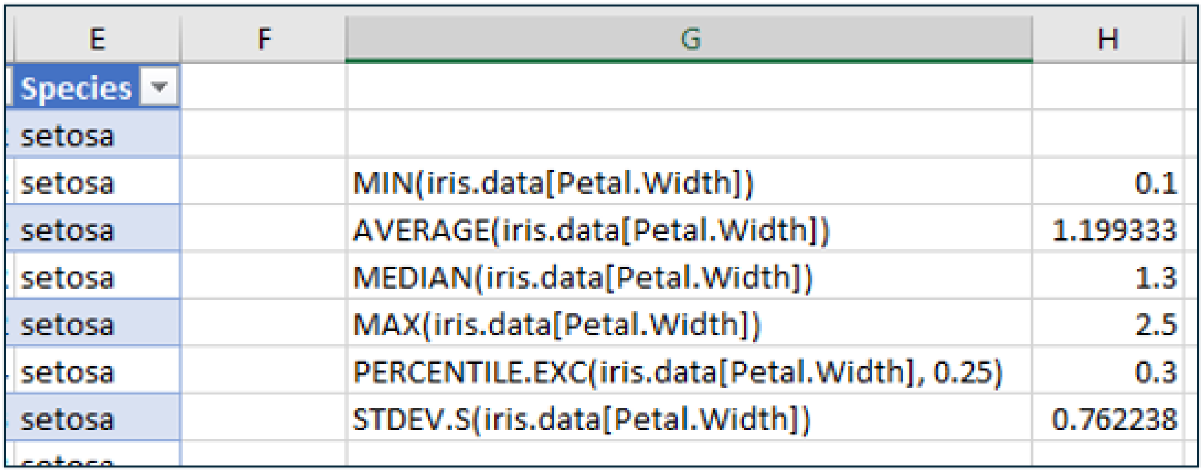

The table
Table column
"access the table"
R is smart. "IsSetosa" doesn't exist, so R adds it.
"store data into"
R's ifelse function
Apply ifelse function to every Species value

```
> iris.data$IsSetosa

> iris.data$IsSetosa <-

ifelse(iris.data$Species == "setosa", "Y", "N")

> iris.data$IsSetosa <- ifelse(iris.data$Species == "setosa", "Y", "N")
```

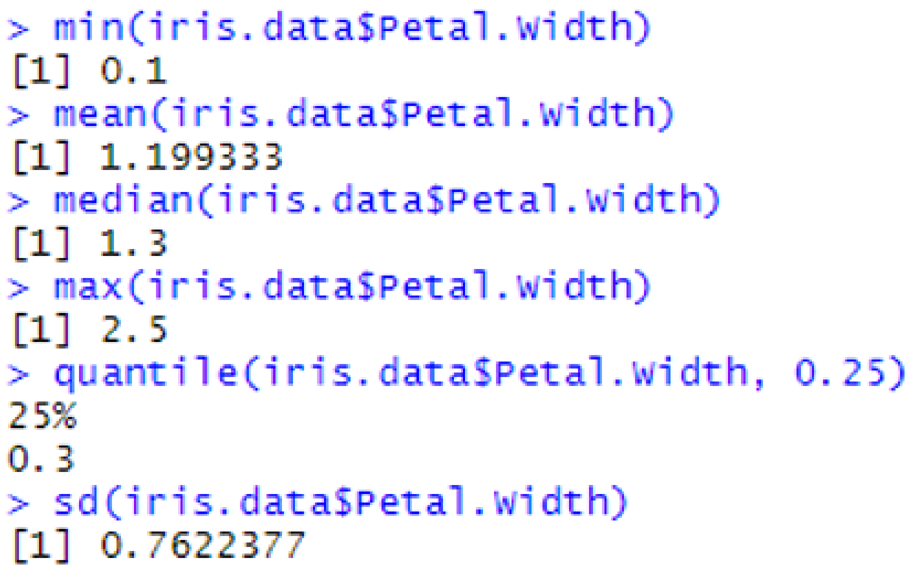
Common Functions

Like Excel, R comes out of the box with many, many functions to work with columns of data.

Many of the R functions share the same name with the corresponding Excel function. In other cases, mapping your Excel knowledge to R is straightforward, as depicted below.



E	F	G	H
Species			
setosa			
setosa		MIN(iris.data[Petal.Width])	0.1
setosa		AVERAGE(iris.data[Petal.Width])	1.199333
setosa		MEDIAN(iris.data[Petal.Width])	1.3
setosa		MAX(iris.data[Petal.Width])	2.5
setosa		PERCENTILE.EXC(iris.data[Petal.Width], 0.25)	0.3
setosa		STDEV.S(iris.data[Petal.Width])	0.762238

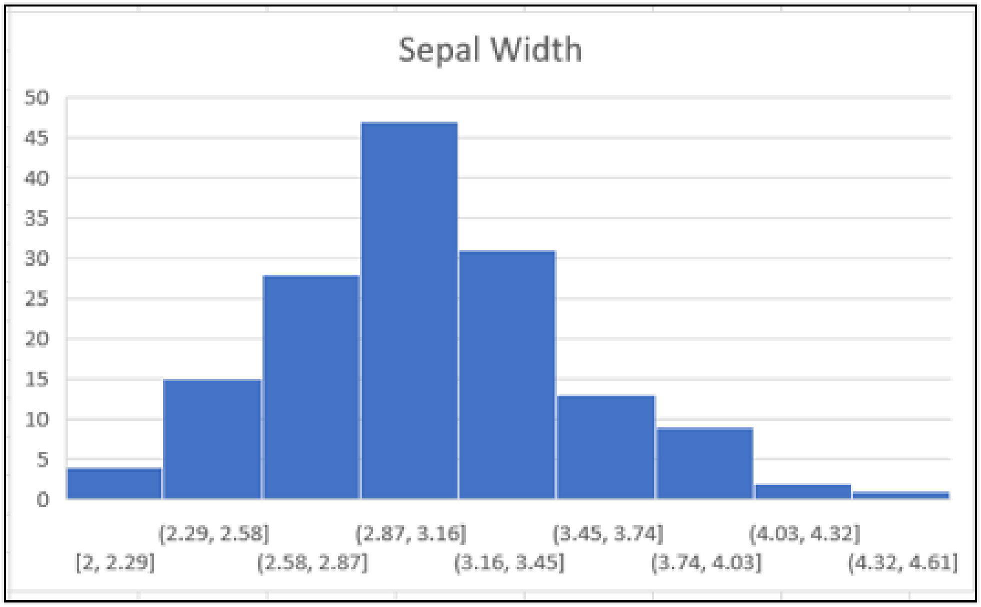


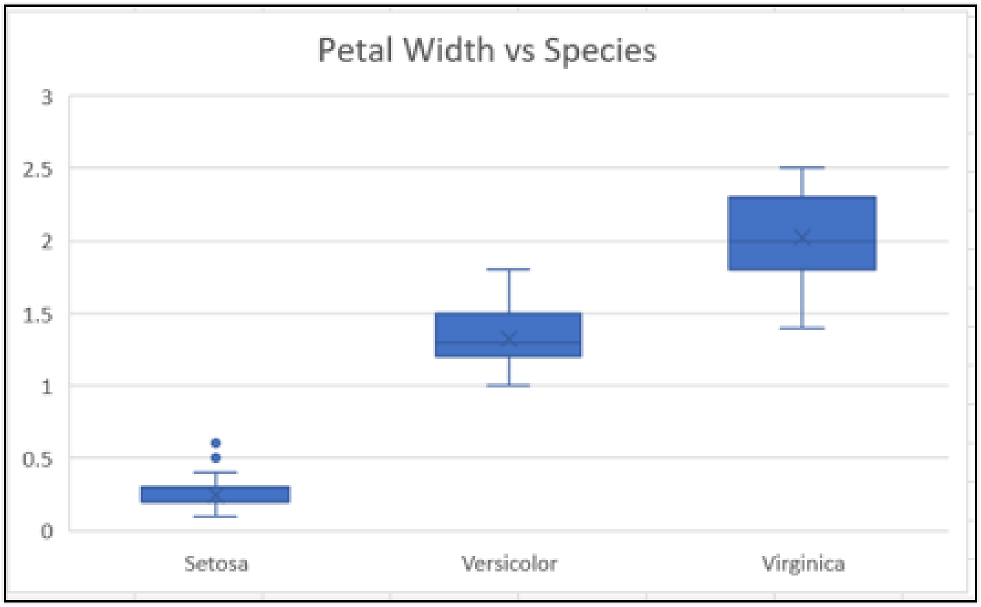
```
> min(iris.data$Petal.Width)
[1] 0.1
> mean(iris.data$Petal.Width)
[1] 1.199333
> median(iris.data$Petal.Width)
[1] 1.3
> max(iris.data$Petal.Width)
[1] 2.5
> quantile(iris.data$Petal.Width, 0.25)
25%
0.3
> sd(iris.data$Petal.Width)
[1] 0.7622377
```

Awesome Data Visualizations

Excel Data Visualizations

Excel is a great data visualization tool, supporting many ways to analyze data visually.



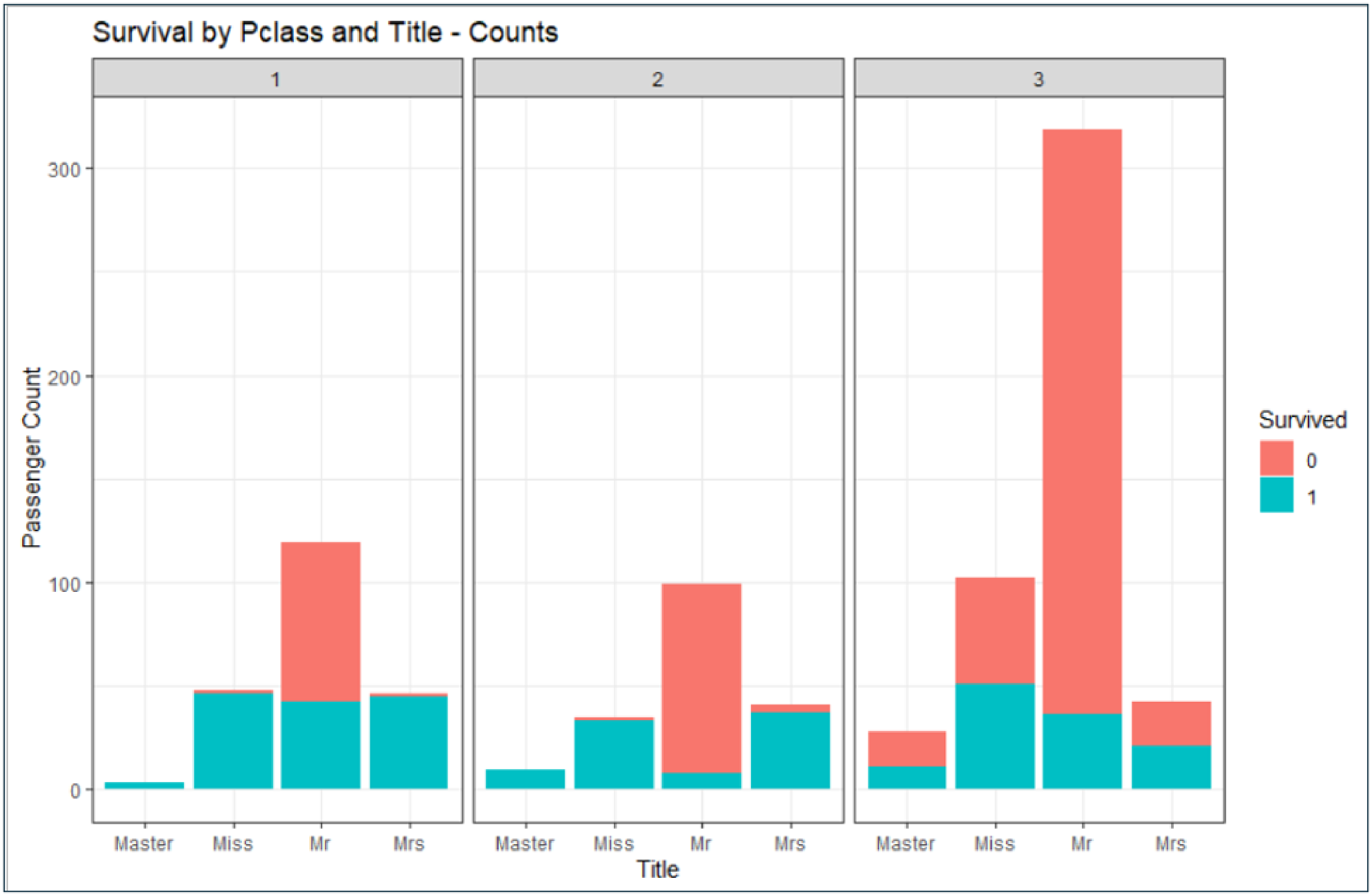


R Data Visualizations

R is renowned for it's ability to create print-quality data visualizations.

R also easily produces data visualizations that are difficult, or not possible, to do with out of the box Excel features.

Analyzing data visually is one of R's specialities.

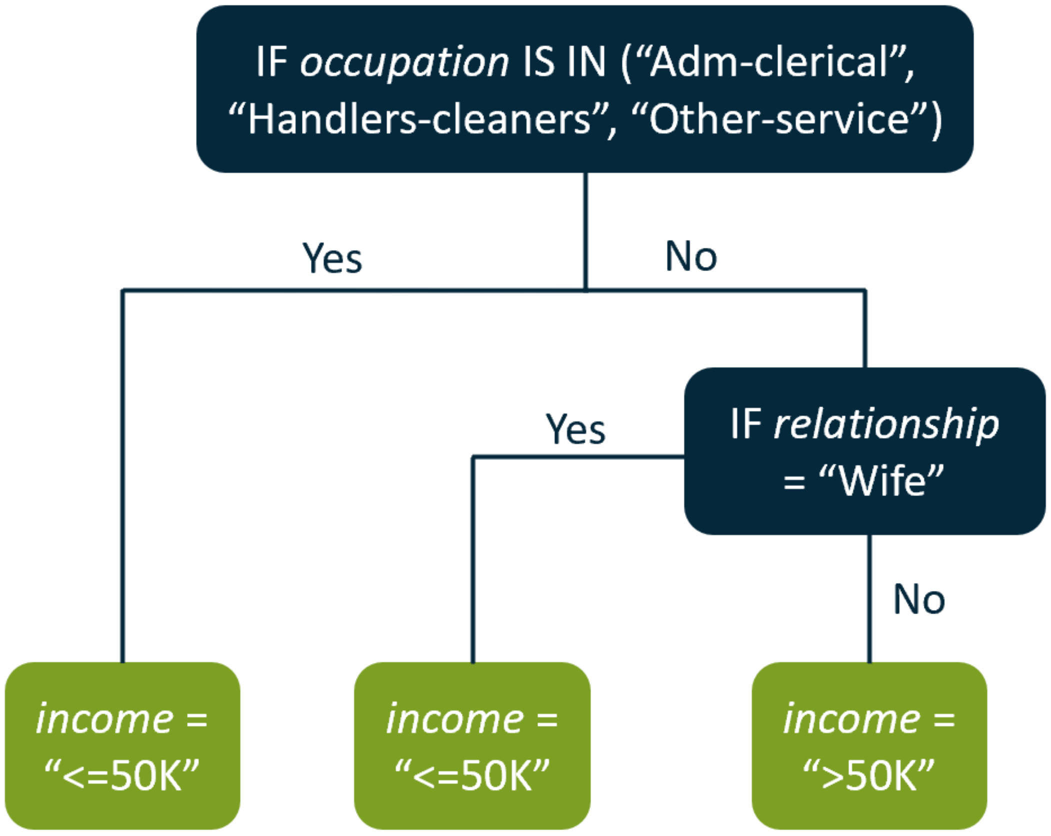


Classification Tree Intuition

In machine learning there are many techniques for building predictive models. The technical term for these techniques is the word *algorithm*. Think of an algorithm as a kind of recipe that the machine (i.e., the computer) follows to build the predictive model.

Turns out that one of the most useful machine learning algorithms for business data is also one of the simplest to learn - *decision trees*. Almost everyone is familiar with a decision tree. You start at the top of the tree and move down each decision in the tree until you reach the bottom.

A simple decision tree is depicted below.

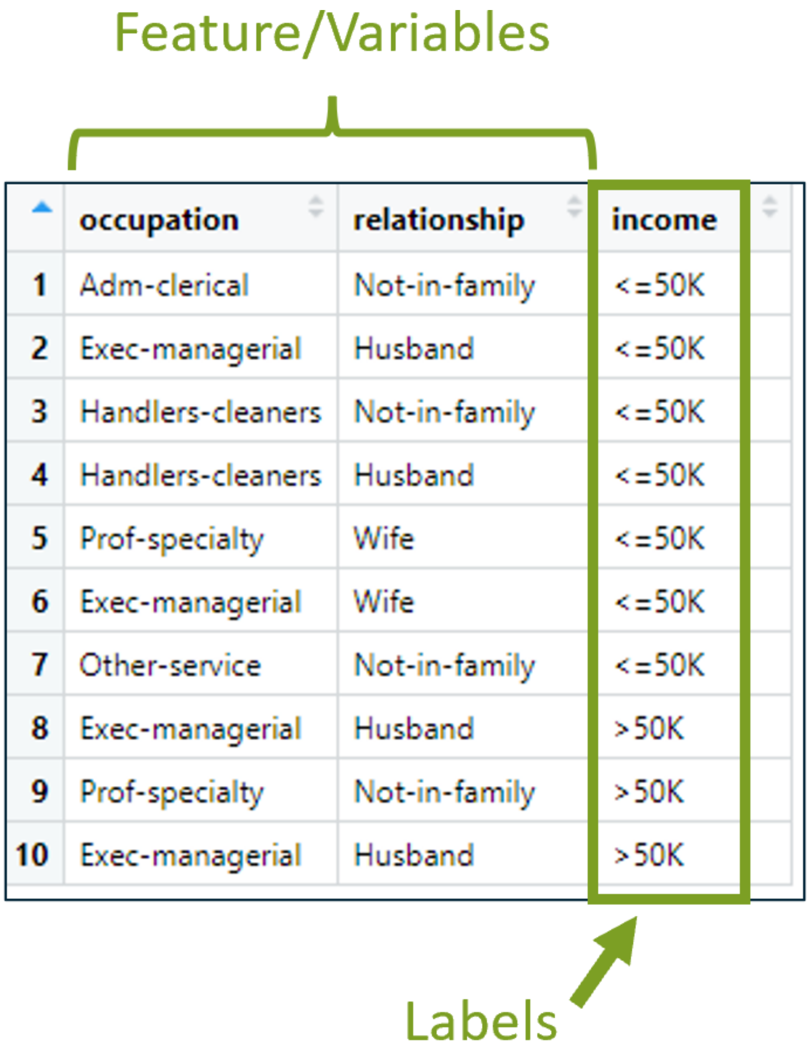

IF *occupation* IS IN (“Adm-clerical”, “Handlers-cleaners”, “Other-service”)
Yes
No
IF *relationship* = “Wife”
Yes
No
income = “<=50K”
income = “<=50K”
income = “>50K”

Trees Are Rules

Machine learning algorithms learn from data. In the case of decision trees, the algorithm learns rules from the data.

Some sample data that was used to create the tree on the previous page is depicted below. When we apply the decision tree algorithm to the data, the algorithm tries to learn the rules to accurately predict the *label* using the *features* in the data set.

When a decision tree is asked to learn how to predict labels, it is known as a *classification tree*.



	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

Most of us are familiar with decision trees being depicted in a graphical form as shown on page 23. However, you can also represent a decision tree using text.

The tree learned from the data listed on the previous page is represented below as text.

What is clear from this representation is that trees are collections of rules used to make predictions.

The textual rules below clearly show what the algorithm learned from the data in terms of predicting income levels (i.e., the *label*).

For example, using the rules below, a person that had an occupation of "Adm-clerical" is predicted to have an annual income of less than or equal to $50,000 USD.

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

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

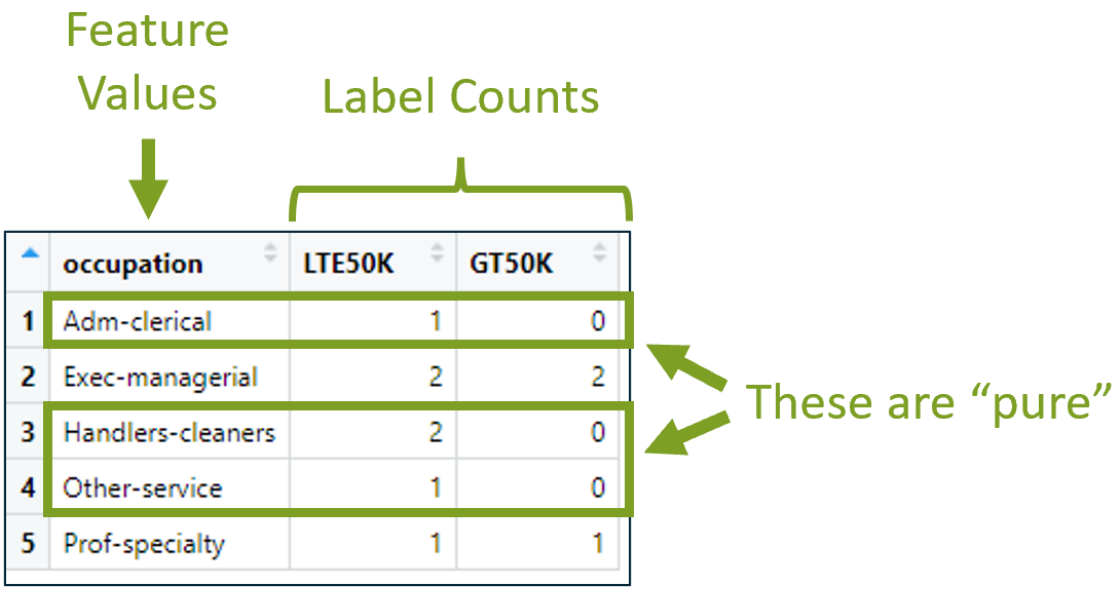
ELSE income **=** “>50K”

How does the classification tree algorithm arrive at these rules?

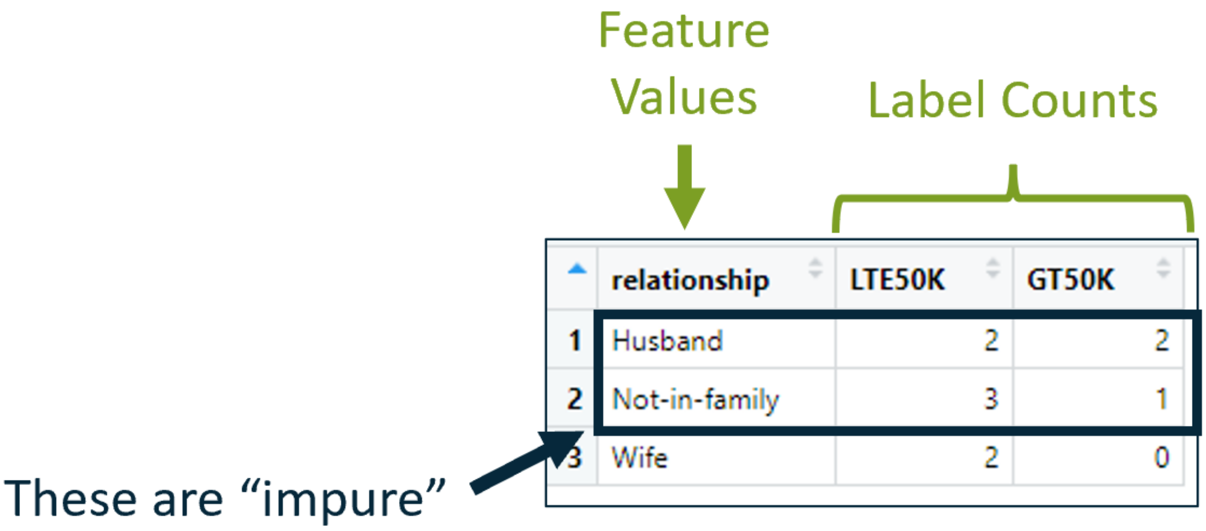
Machine learning algorithms work by pursuing some sort of *objective*. Think of the objective as how the algorithm determines if it has learned all that it can from the data.

In the case of classification trees, the objective is to *minimize impurity*. Later you will learn about the math of *impurity*, for now let's focus on the intuition.

A collection of data is *pure* when all the labels are the same. Reflexively, a collection of data is impure when the labels are not all the same.



	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



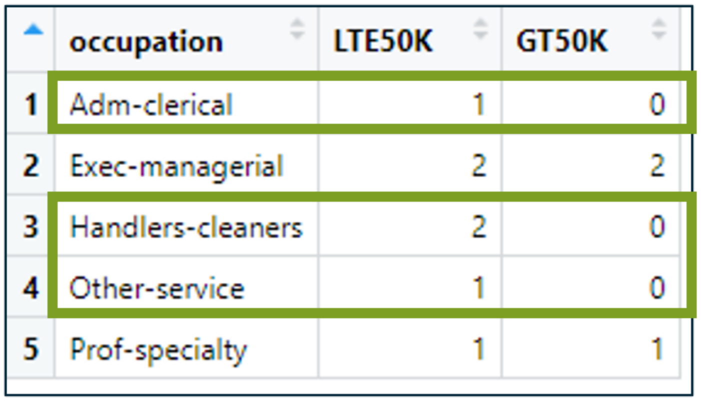
	relationship	LTE50K	GT50K
1	Husband	2	2
2	Not-in-family	3	1
3	Wife	2	0

Let's Build a Tree!

In this contrived example we have 10 *observations* (or rows) and 2 *features* of data.

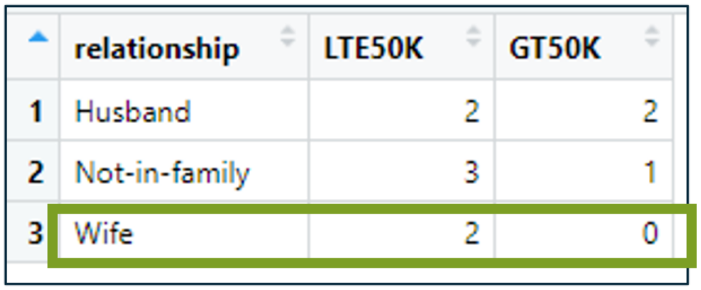
The classification tree algorithm **loves** a single feature with lots of observations and only a single label value.

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

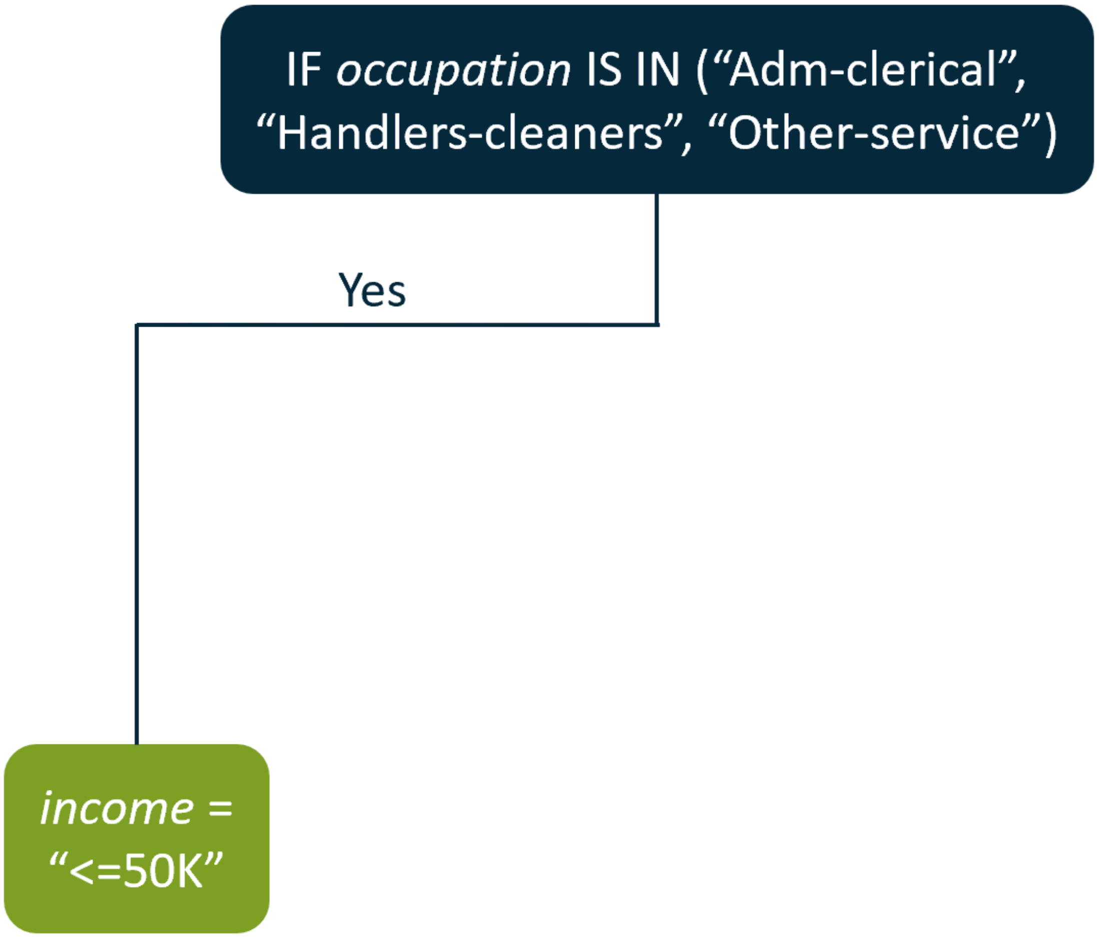
With the *relationship* feature we get 2 observations all with the label "<=50K"

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

This is the first conceptual step of the classification tree algorithm learning from the contrived data set.

Graphically, we can represent what the algorithm has learned so far as depicted below.

However, not all the data has been used in the algorithm's learning process, so we are not done yet!

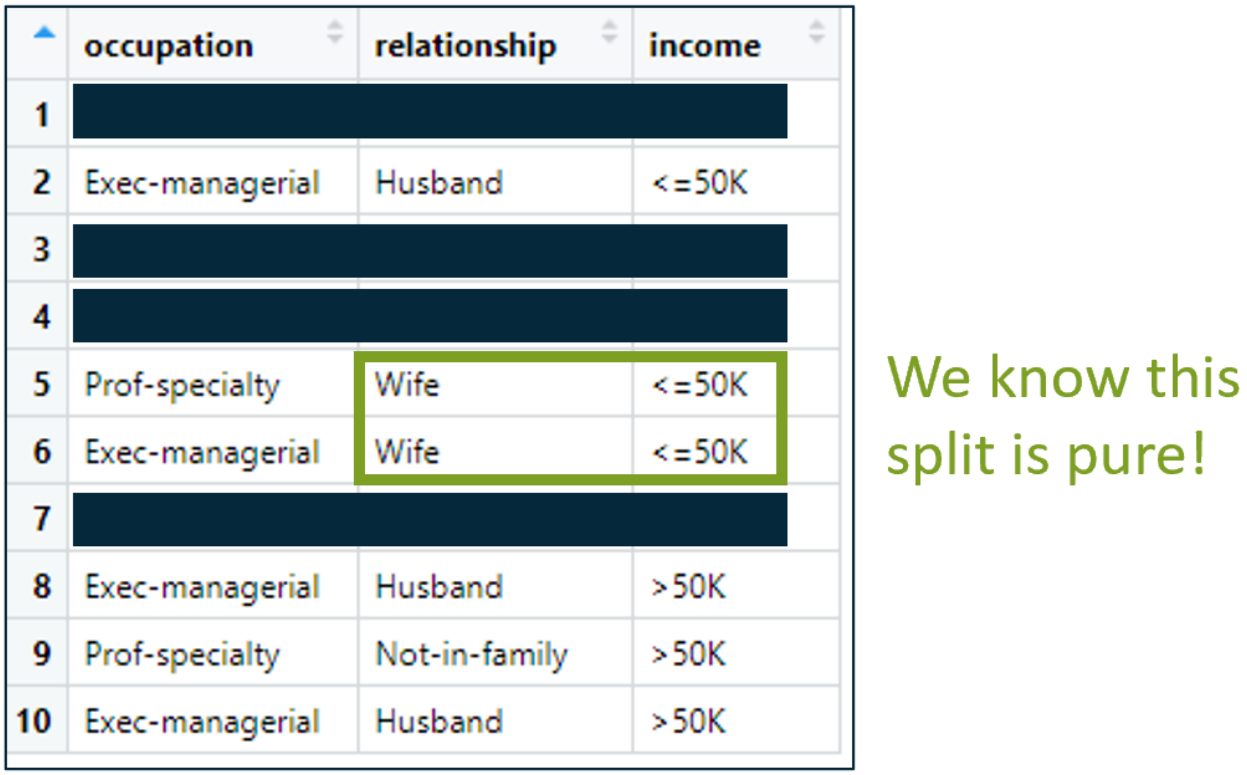

IF *occupation* IS IN ("Adm-clerical", "Handlers-cleaners", "Other-service")
Yes
income = "<=50K"

As depicted below, the tree so far has only used 40% of the data for learning.

The observations used so far have been hidden by bars to clearly show what data is left for the algorithm to use.

Of the data left for use by the algorithm, a pure split using the *relationship* feature is highlighted.

The algorithm chooses this feature next.

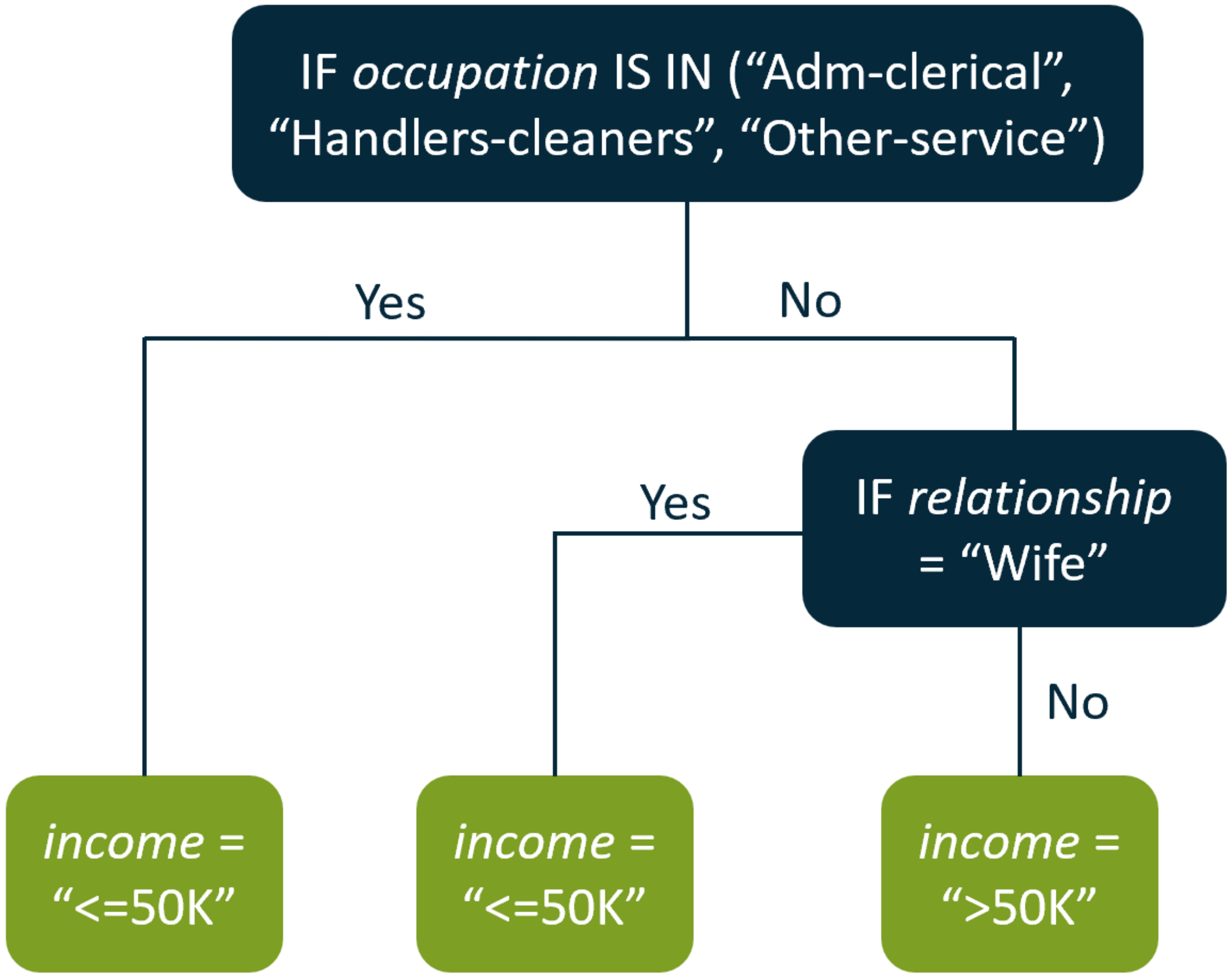


	occupation	relationship	income
1	[redacted]	[redacted]	[redacted]
2	Exec-managerial	Husband	<=50K
3	[redacted]	[redacted]	[redacted]
4	[redacted]	[redacted]	[redacted]
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	[redacted]	[redacted]	[redacted]
8	Exec-managerial	Husband	>50K
9	Prof-specialty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

The math of impurity tells the algorithm that after using the *relationship* feature value of *Wife*, there are no valid ways of splitting the data further.

As such, the algorithm completes the tree as depicted below.

Viola! You now have a conceptual understanding of how classification trees learn from data.


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

Overfitting Intuition

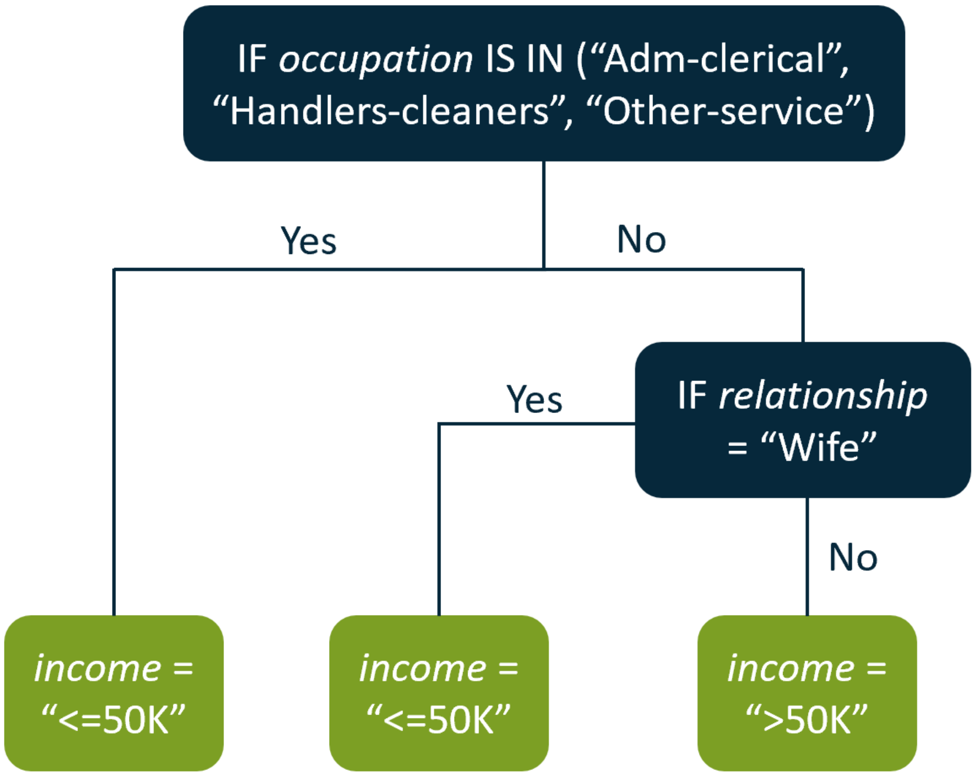
The Bugbear of Machine Learning

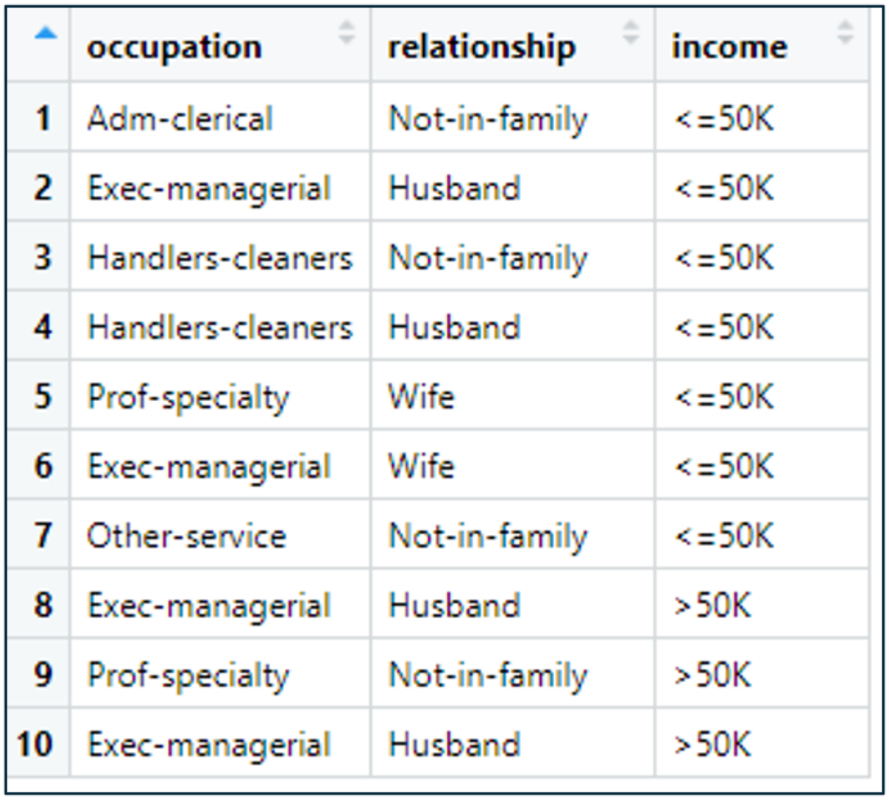
As a business professional applying machine learning to your data, you must be paranoid about *overfitting*.

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

The full bootcamp covers overfitting in depth, this document will focus on the intuition.

Take the tree from the previous page as an example.


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

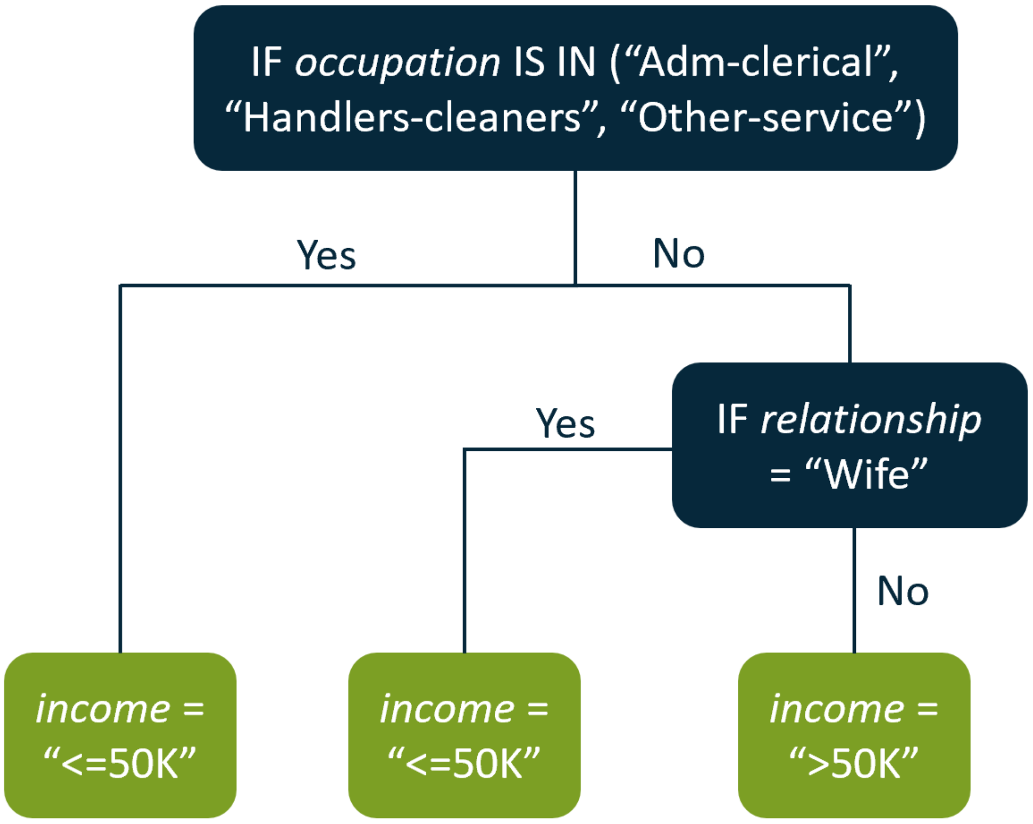


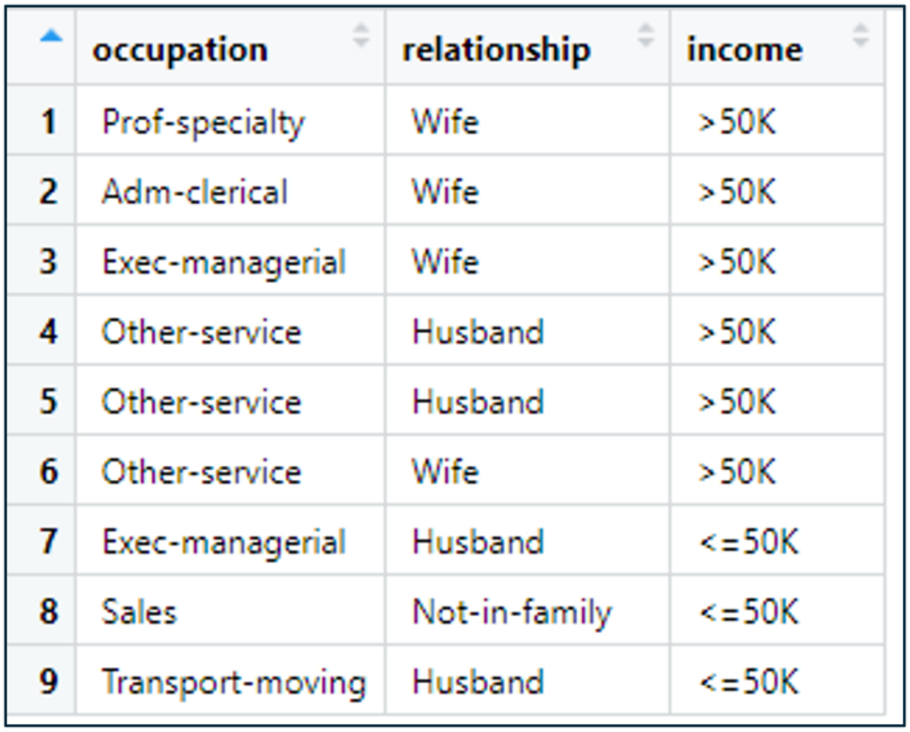
	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty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

The Model Is Good!

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

When we look at the original data and the tree below, we see that the model correctly predicts 9 out of 10 (i.e., 90%) of the labels correctly!


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



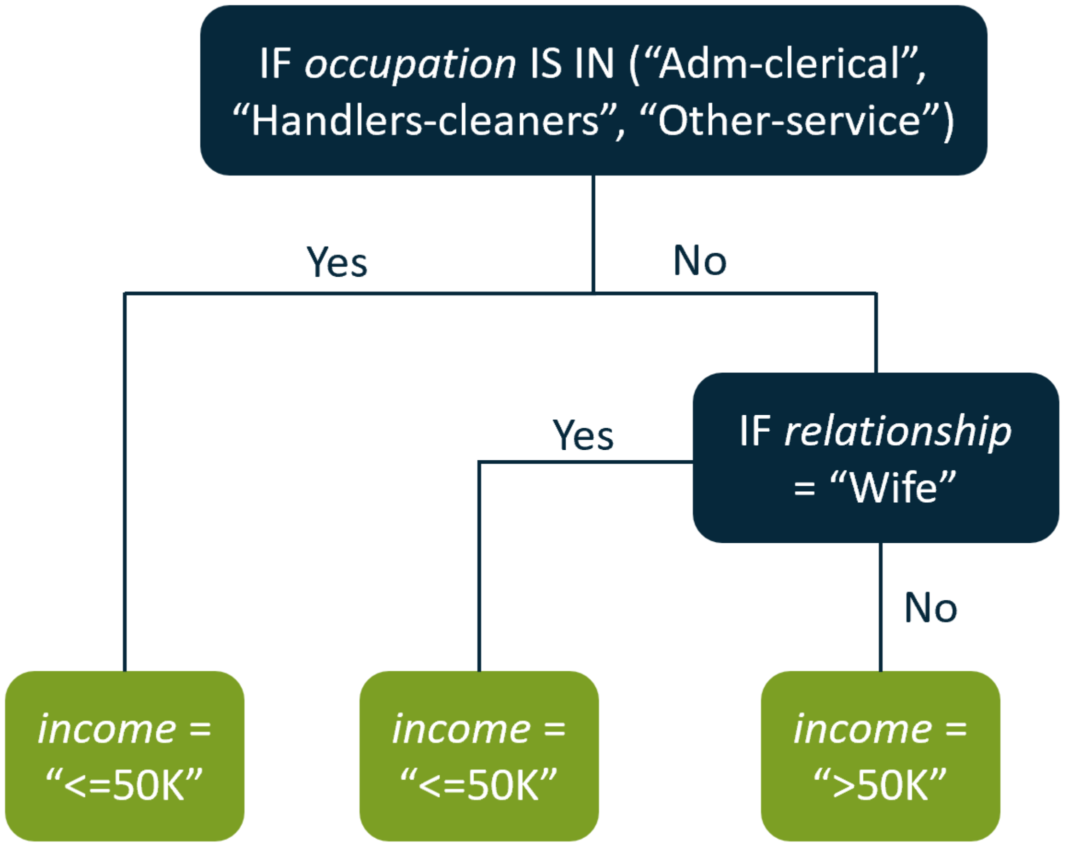
	occupation	relationship	income
1	Prof-specialty	Wife	>50K
2	Adm-clerical	Wife	>50K
3	Exec-managerial	Wife	>50K
4	Other-service	Husband	>50K
5	Other-service	Husband	>50K
6	Other-service	Wife	>50K
7	Exec-managerial	Husband	<=50K
8	Sales	Not-in-family	<=50K
9	Transport-moving	Husband	<=50K

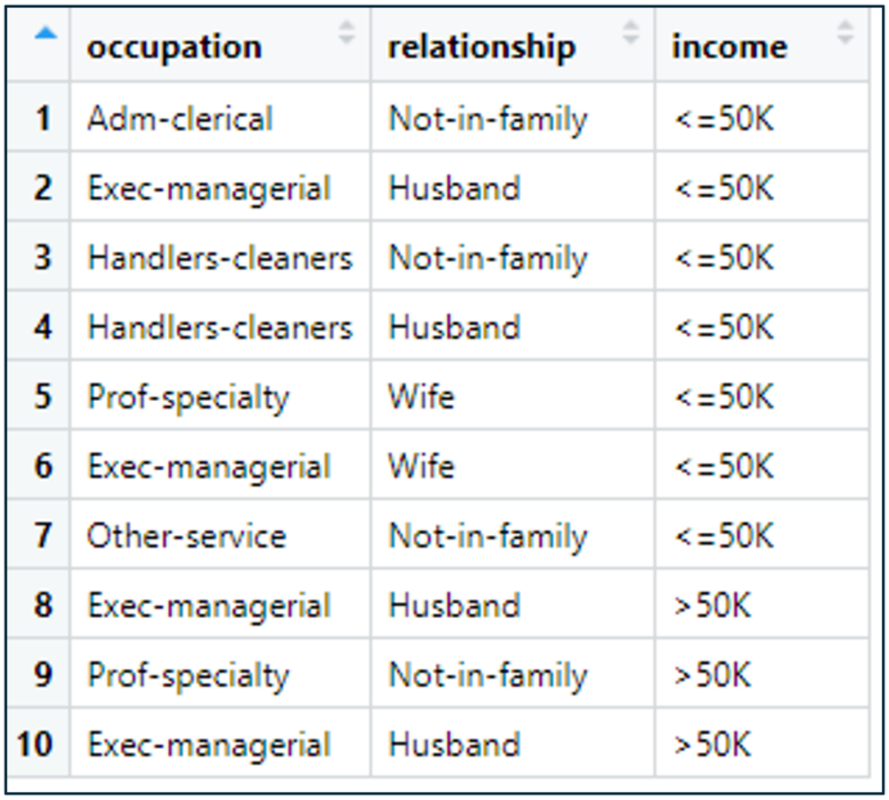
Or Is It?

Take the data to the left that was not used in training the model. This is "new data."

When the model below is used to make predictions for this data it gets 9 out of 9 (i.e., 100%) of the labels wrong!

This is the essence of overfitting.


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



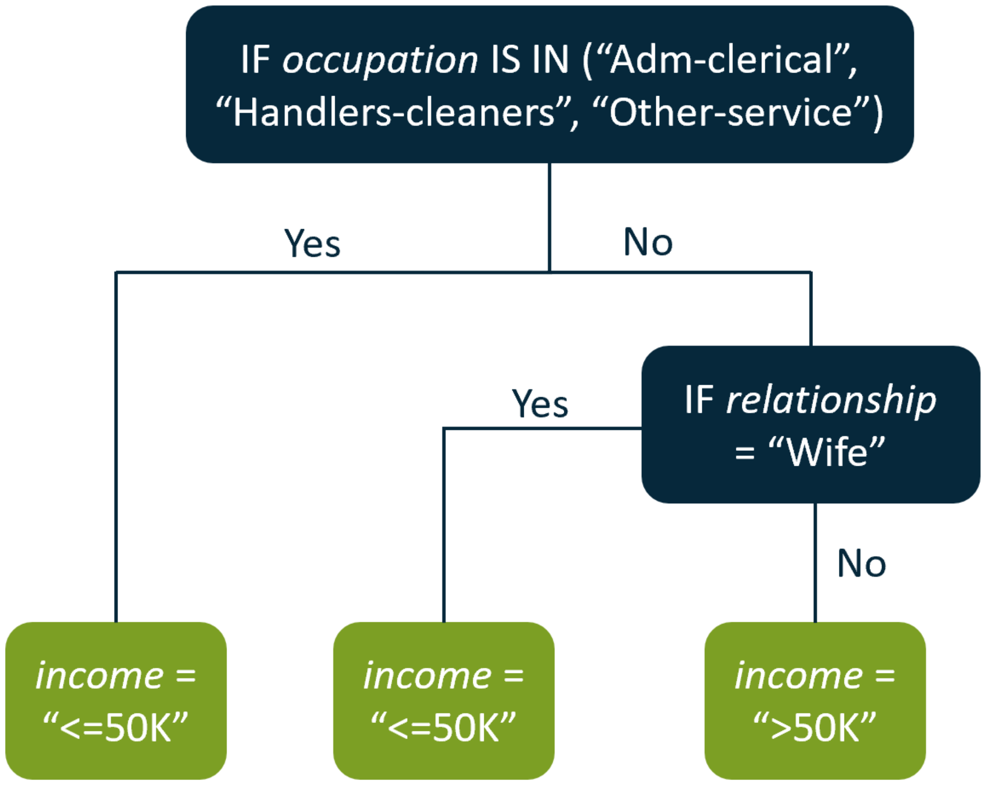
	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty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

What Happened?

Given the relatively small amount of data to the left, the tree is far too specialized.

For example, the model assumes that any person with a *relationship* of *Wife* makes under $50,000 USD.

While that doesn't make sense to us humans, it makes perfect sense to the greedy algorithm based on the data and how the model was trained.


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

Model Tuning Intuition

The bootcamp has robust coverage of model tuning. For the purposes of this document, the focus will be on the intuition behind tuning machine learning models.

Conceptually, think of model tuning like tuning your automobile for optimal performance.

In terms of decision trees...

The decision tree "engine" has a number of settings that you can control as the machine learning practitioner

These settings are called *hyperparameters* and are typically changed as needed for optimal decision tree performance - just like a mechanic can change the engine settings in your automobile for optimal performance.

In the case of decision trees, tuning manifests as controlling how specialized (i.e., how *complex*) a tree can become from the data used to train it.

In the case of the tree on the previous page, you would tune the hyperparameters to make a less complex tree to pursue better predictions on new data.

Gini Impurity

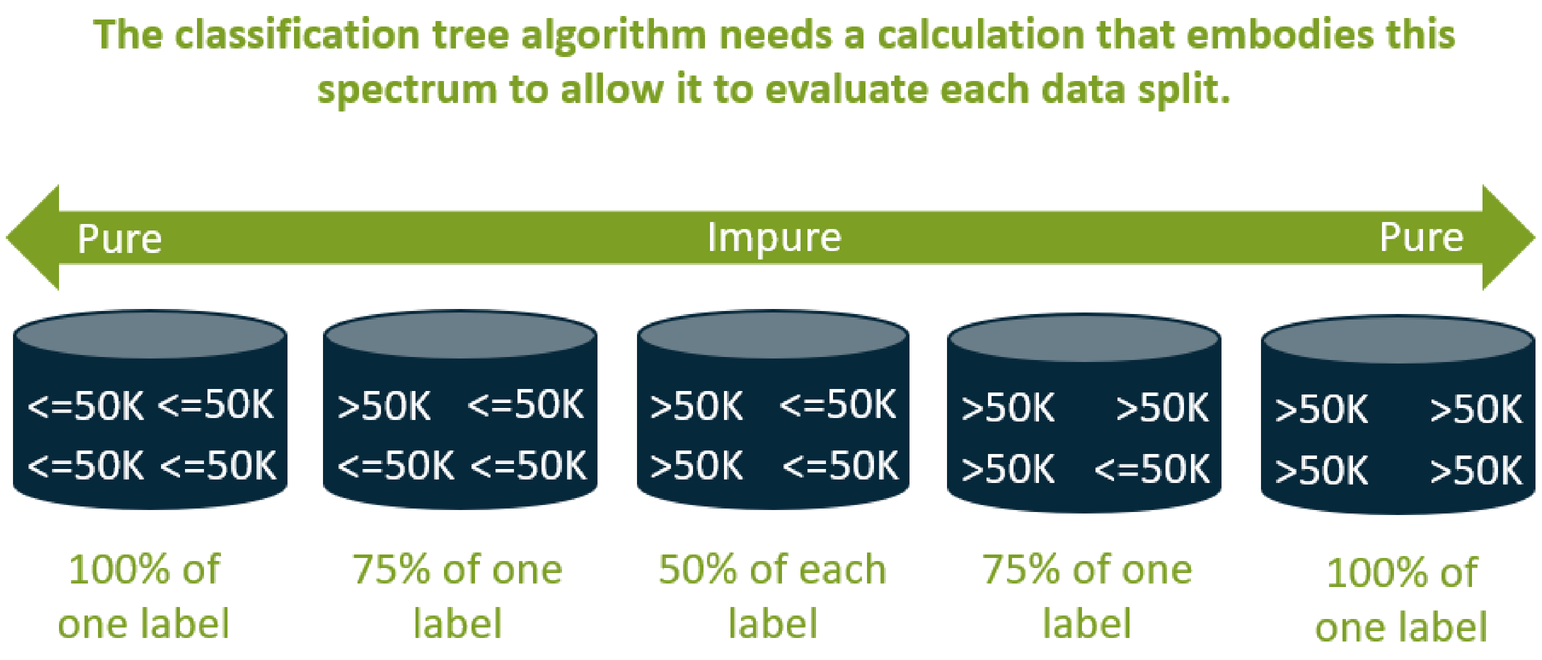
Impurity Intuition

The bootcamp has robust coverage of the mathematics of decision trees. Here's the good news:

1. The mathematics are simple and accessible to any professional.
2. Thorough understanding mathematics is useful, but not required.

This document will focus on the intuition of impurity math, starting with a 2-label (i.e., *binary*) case.

Using our example, we can think about "buckets" of labels and impurity as a spectrum...



Gini Impurity

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

The bootcamp uses the *Gini impurity* calculation.

Gini impurity is widely used and is the default calculation used by the R packages taught in the bootcamp.

Don't panic!

While the math may appear intimidating, when you think about it in English, it is really quite easy...

"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"

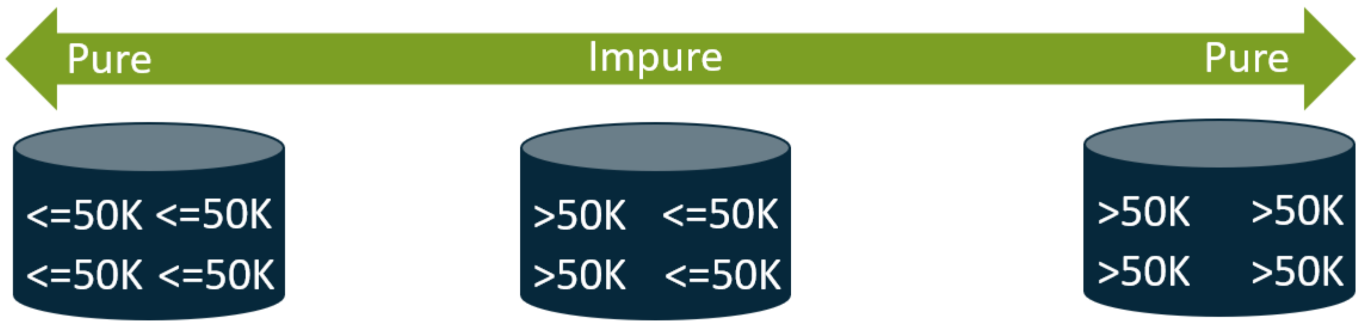
Gini Impurity Example

As we know, the classification tree algorithm loves to split up the data into groups where the labels are all the same (i.e., *pure*).

In other words, the classification tree algorithm's *objective* is to *minimize impurity*.

Using the math from the previous page we can now do some calculations and see how pure buckets of labels have the smallest Gini impurity scores.

Notice that the worst case scenario is a 50/50 split of labels. This makes intuitive sense when you have 2 labels - it is essentially a coin flip.



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

$\text{Gini} = 1 - (4/4)^2 - (0/4)^2$

$\text{Gini} = 1 - 1 - 0$

$\text{Gini} = 0$

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

$\text{Gini} = 1 - (2/4)^2 - (2/4)^2$

$\text{Gini} = 1 - 0.25 - 0.25$

$\text{Gini} = 0.5$

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

$\text{Gini} = 1 - (0/4)^2 - (4/4)^2$

$\text{Gini} = 1 - 0 - 1$

$\text{Gini} = 0$

Day 1 - R Programming Made Easy

Module 1 - Welcome

- Bootcamp Expectations
- The Bootcamp Data
    - Iris Data Set
    - Adult Census Income Data Set
    - Titanic Data Set

Module 2 - R and RStudio

- The History of R
- Similarities Between Excel & RStudio
    - Excel and R as Data Analysis Ecosystems
    - Excel Users Write Code
    - Excel Code Is Like R Code
    - Extending the Excel & R Ecosystems

Module 3 - Data Analysis Is All About Objects

- Objects in Excel & R
- It All Starts With Tables
    - Tables in Excel & R
    - Cells of Data
    - Columns of Data
- Vectors of Data
    - Excel Cell Ranges
    - R Vectors
- Data Types
    - Excel Data Formats
    - R Data Types
- Math With Vectors
    - Excel-Style Vector Math
    - R Vector Math

Module 4 - Filtering R Tables

- Logical Filtering
  - Excel Logical Filtering
  - R Logical Filtering
  - Filtering with %in%
- R Filtering
  - Filtering With Numbers & Names
  - Filtering With Direct Logic
- Hands-on Lab #1

Module 5 - Basic Functions

- Missing Data
- Common Stats Functions
  - Mapping Excel Functions to R Functions
- The R summary Function
  - Summary Stats in Excel
- The data.frame Function
- The cbind & rbind Functions
- The aggregate Function
  - Excel Data Aggregation
  - R Data Aggregation
- Hands-on Lab #2

Module 6 - Pivoting Data

- The Mighty dplyr
- Another Kind of Table - tibbles
- Making Your Own Data
  - Making Data in Excel & R
  - Making Vectors of Data
- Selecting & Filtering Data
- Grouping & Summarizing Data
  - Pivoting Data in Excel & dplyr

Module 7 - Multiple Tables of Data

- Joining Data
  - Joining Data in Excel
  - Joining Data in R
- Sorting Data
- Hands-on Lab #3

Module 8 - Data Visualization

- Introducing ggplot2
- Box Plots
- Histograms
- Bar Charts
- Scatter Plots
- Hands-on Lab #4

Days 2 & 3 - Introduction to Machine Learning With R

Module 9 - Supervised Learning

- You Are the Teacher
  - What is Machine Learning?
  - Supervised Learning
  - Classification vs Regression
- Why Decision Trees?
  - Ease of Use
  - Optimal for Many Business Problems

Module 10 - Classification Trees

- Basic Intuition
  - Trees Are Rules
  - Sample Decision Tree
- Overfitting Intuition
  - The Bugbear of Machine Learning
  - The Model Is Good! Or Is It?

Module 11 - Classification Tree Math

- Gini Impurity
- Gini Change
- Many Categories Impurity
- Numeric Feature Impurity

Module 12 - Using Classification Trees

- Classification Trees, tidymodels Style
  - The tidymodels Universe
  - Recipes
  - Model Specifications
  - Workflows
  - Model Fitting
- Hands-on Lab #5

Module 13 - Introducing the Bias-Variance Tradeoff

- Under/Overfitting
  - The Goldilocks Zone
  - Controlling Complexity
- The Bias-Variance Tradeoff
  - Intuitive Example
  - Model Example

Module 14 - Model Tuning

- Supervising the Data
  - Splitting the Data
  - Cross-Validation
- Model Tuning Intuition
  - Making an Intuitive Example Real
  - Estimating Generalization Error
  - What About the Test Set?
- Pruning Classification Trees
  - Pruning Intuition
  - Pre/Post-Pruning

Module 15 - Model Tuning With tidymodels

- Measuring Model Accuracy
    - Accuracy
    - Confusion Matrices
    - Sensitivity
    - Specificity
- Model Tuning with Tidymodels
    - Setting up Cross-Validation
    - Cross-Validation Results
    - Tuning the Tree
    - Tuning the Results
- Hands-on Lab #6

Module 16 - Feature Engineering

- Intuition
    - What is Feature Engineering?
    - Extracting Features
    - Row vs Column Features
- Data Leakage
    - What Is It?
    - An Example
    - tidymodels to the Rescue
- Engineering Features for Decision Trees
    - Visualizing Decision Boundaries
    - Concepts to Remember
- Missing Data
    - Why Is Data Missing
    - Dealing With Missing Data
    - Imputation With tidymodels
- Hands-on Lab #7

Module 17 - Regression Trees

- The Basics
  - Regression Trees Minimize SSE
  - Calculating SSE
- Numeric Feature SSE
- Many Categories SSE
- Regression Trees With tidymodels
  - Measuring "Accuracy"
  - Model Specification
  - Regression Trees in Practice

Module 18 - The Mighty Random Forest

- Bad, Tree! Bad!
  - Decision Tree Variance
  - High Variance Leads to Overfitting
  - Real-World Decision Trees
- Ensembles
  - Wisdom of the Crowd
  - Manufacturing Independence
- Bagging
  - Randomizing Rows
  - Bagging in Action
  - The Power of Bagging
- Feature Randomization
  - Intuition
  - Randomizing Columns
  - Feature Randomization in Action

Module 19 - Using the Random Forest

- Tuning Random Forests
    - The Bias-Variance Tradeoff
    - Random Forest Hyperparameters
- Feature Importance
    - Out of Bag (OOB) Data
    - Permutation Importance
- Random Forests With tidymodels
- Hands-on Lab #8

Module 20 - Bootcamp Wrap-Up

- Want to Kaggle?
- Additional Resources

Ready to Empower Your Team?

As this document clearly demonstrates, if your team has Excel skills I can teach them R programming. R programming opens up the universe of advanced analytics of which machine learning is one example.

My speciality is customizing analytics training for your team to ensure your investment generates business impact.

This customization takes on two primary forms:

1. Delivering a curriculum optimized for your team, either on-site or virtually.
2. Providing ongoing virtual coaching services to cement skills with your team.

Here are some common curriculum customizations I delivered to clients:

- Delivering days 2 & 3 of the bootcamp to teams that already know R.
- Adding 2 days of training on linear and logistic regression with R.
- Swapping out machine learning for 2 days of linear/logistic regression with R.

Many of my clients augment bootcamp training with 1-on-1 coaching services for their team. By coaching your team on how to apply the skills learned to your specific business problems, your return on investment is ensured.

BTW - For my coaching clients I offer a guarantee. I will continue to coach your team free of charge until they are successful with the learned skills.

Interested in learning more?
Email me: dave@daveondata.com

About the Author

My name is Dave Langer and I am the founder of Dave on Data.

I'm a hands-on analytics professional, having used my skills with Excel, SQL, and R to craft insights, advise leaders, and shape company strategy.

I'm also a skilled educator, having trained 100s of working professionals in a live classroom setting and 1000s more via my online courses and tutorials.

In the past, I've held analytics leaderships roles at Schedulicity, Data Science Dojo, and Microsoft.

I am proud to partner with TDWI to deliver the best analytics training to ANY team.

Drop me an email if you have any questions: dave@daveondata.com