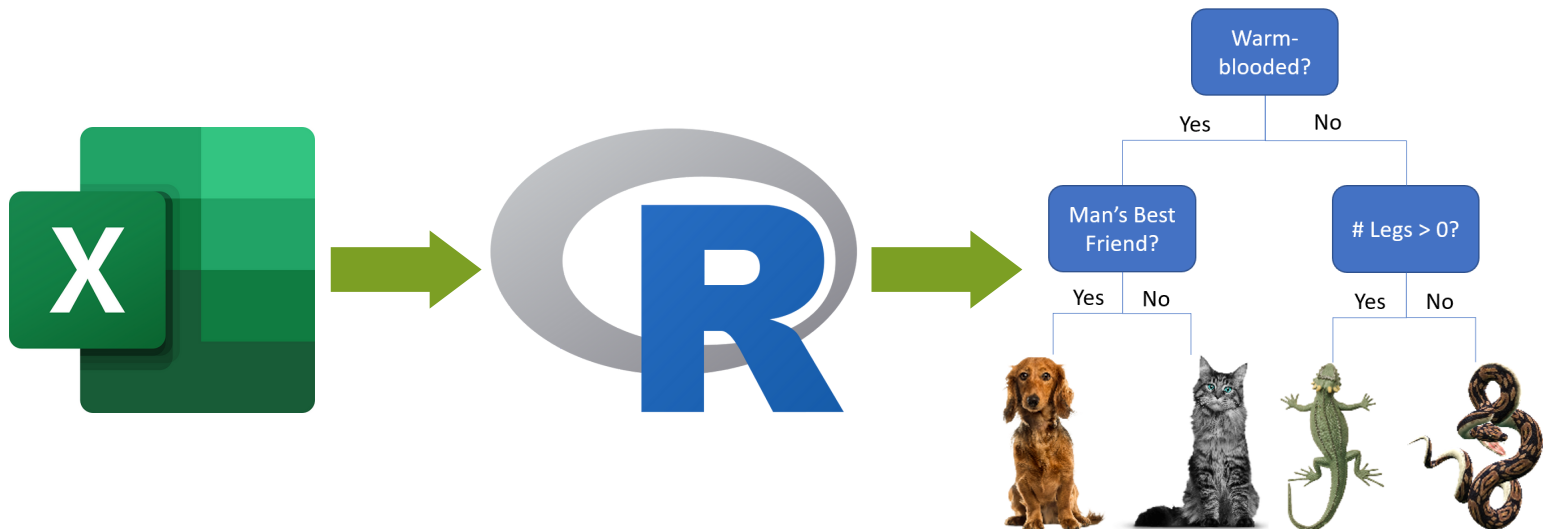


FROM EXCEL TO MACHINE LEARNING ONLINE



**YOUR EXCEL SKILLS WILL UNLOCK
ADVANCED ANALYTICS**

Be a Part of the Data-Driven Future

If you have Microsoft Excel skills I can teach you how to use R programming and machine learning to analyze data. Doesn't matter if you've never coded before. Doesn't matter if you've never took calculus at university.

I know. It's a bold statement.

First, don't take it from me. Here's some feedback from one of my students:

“First, I’d like to thank Dave for the great effort he put on this course ([Introduction to Machine Learning With R](#)). For an introductory course, I found it very detailed and well explained. The methods he uses on teaching are just awesome and that makes the course way too easy to understand.”

Dave is also very responsive and that just adds to the awesomeness. I honestly can't wait to take more courses with him. Also, the things you learn in this course are highly applicable in real life, and he makes the math to be so easy. **If you are a beginner in this field, then this is the right place.** If you are an experienced data scientist trying to do a refresher course, this is the right place too.

Keep up the good work Dave, I'm inspired. Thank you.“ – Joseph Kamau

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

If you want to have more impact at work using R, data analysis, and machine learning, I invite you to continue reading more about my self-paced online course bundle.

BTW - I offer generous discounts to students and teams/organizations. Drop me an email to learn more or if you have any questions: dave@daveondata.com

Document Goals

This document is designed to help you ascertain whether the Dave on Data self-paced "From Excel to Machine Learning" online course bundle is right for you.

The following are the goals of this document:

1. To demonstrate that any professional with Excel skills can learn R programming.
2. To demonstrate that any professional can acquire machine learning skills.
3. To provide an outline of the online course bundle curriculum.

Per the above, the document is structured as follows:

- Part 1 illustrates how Excel knowledge maps directly to R programming using material from the first course in the bundle - "R Programming Made Easy."
- Part 2 illustrates how approachable machine learning is to any professional using material from the second course in the bundle - "Introduction to Machine Learning With R."
- Part 3 provides the complete outline of topics for the online course bundle.

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

If you're interested in learning more, you can check out free lesson video previews via the link at the bottom of this page.

Additionally, if the online course bundle is right for you, use the coupon **LINKEDIN** at checkout to save an additional 20% off the purchase price.

Table of Contents

Part1 - Why R Programming Is Easy

Excel Users Write Code.....7

Excel Code

R Code

Excel Skills Translate Directly to R

Excel and R Are Like Icebergs.....10

It's All About the Tables.....11

Tables Are Key

Tables in Excel and R

Cells of Data in Excel and R

Columns of Data in Excel and R

Throw in Some Functions.....18

Using Functions in Excel and R

Common Functions

Awesome Data Visualizations.....21

Excel Data Visualizations

R Data Visualizations

Table of Contents

Part 2 - Machine Learning for Any Professional

Classification Tree Intuition.....23

Trees Are Rules

Let's Build a Tree!

Overfitting Intuition.....31

The Bugbear of Machine Learning

The Model Is Good - Or Is It?

What Happened?

Model Tuning Intuition

Gini Impurity.....36

Impurity Intuition

Gini Impurity

Gini Impurity Example

Table of Contents

Part 3 - Online Course Bundle Topic Outline

R Programming Made Easy.....39

Introduction to Machine Learning With R.....42

Wrap Up

You've Got This if You Want It.....45

About the Author.....46

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!

fx =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					

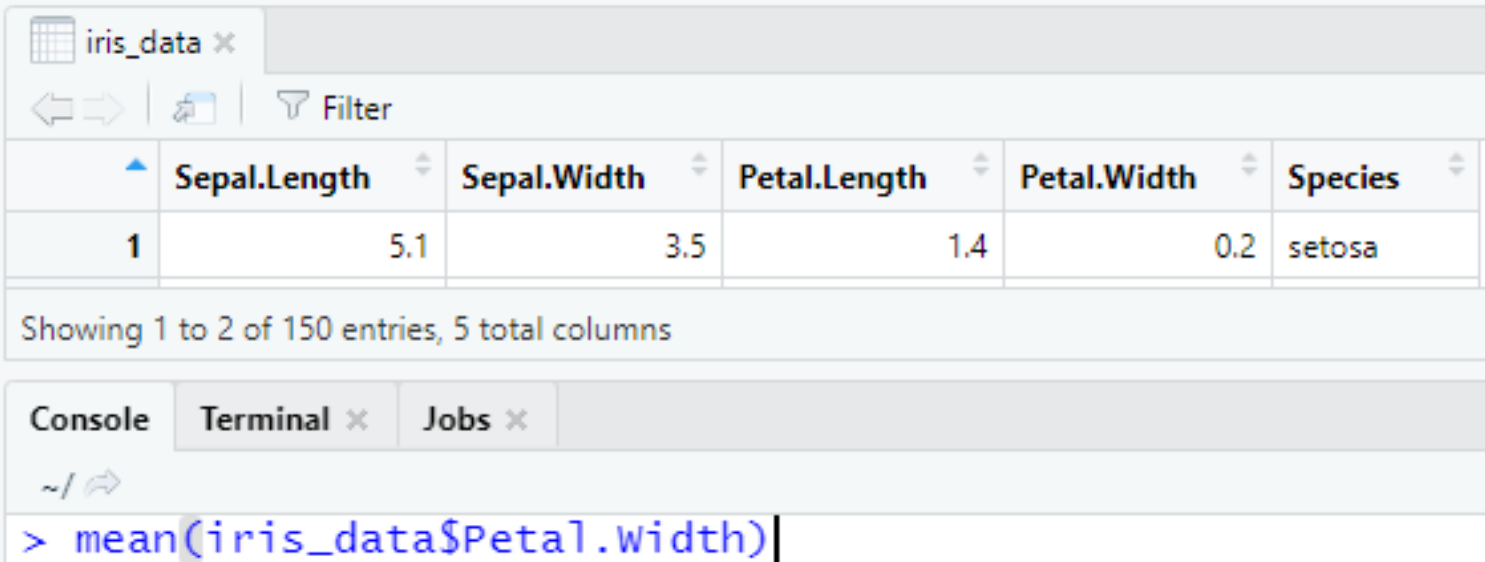
R Code

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.



The screenshot shows the RStudio interface. At the top, there's a tab labeled 'iris_data'. Below it, a table with 5 columns is visible: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. The first row of data is shown with values 5.1, 3.5, 1.4, 0.2, and 'setosa'. Below the table, it says 'Showing 1 to 2 of 150 entries, 5 total columns'. At the bottom, the 'Console' tab is active, showing the command `> mean(iris_data$Petal.Width)` being entered.

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

```
> 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.

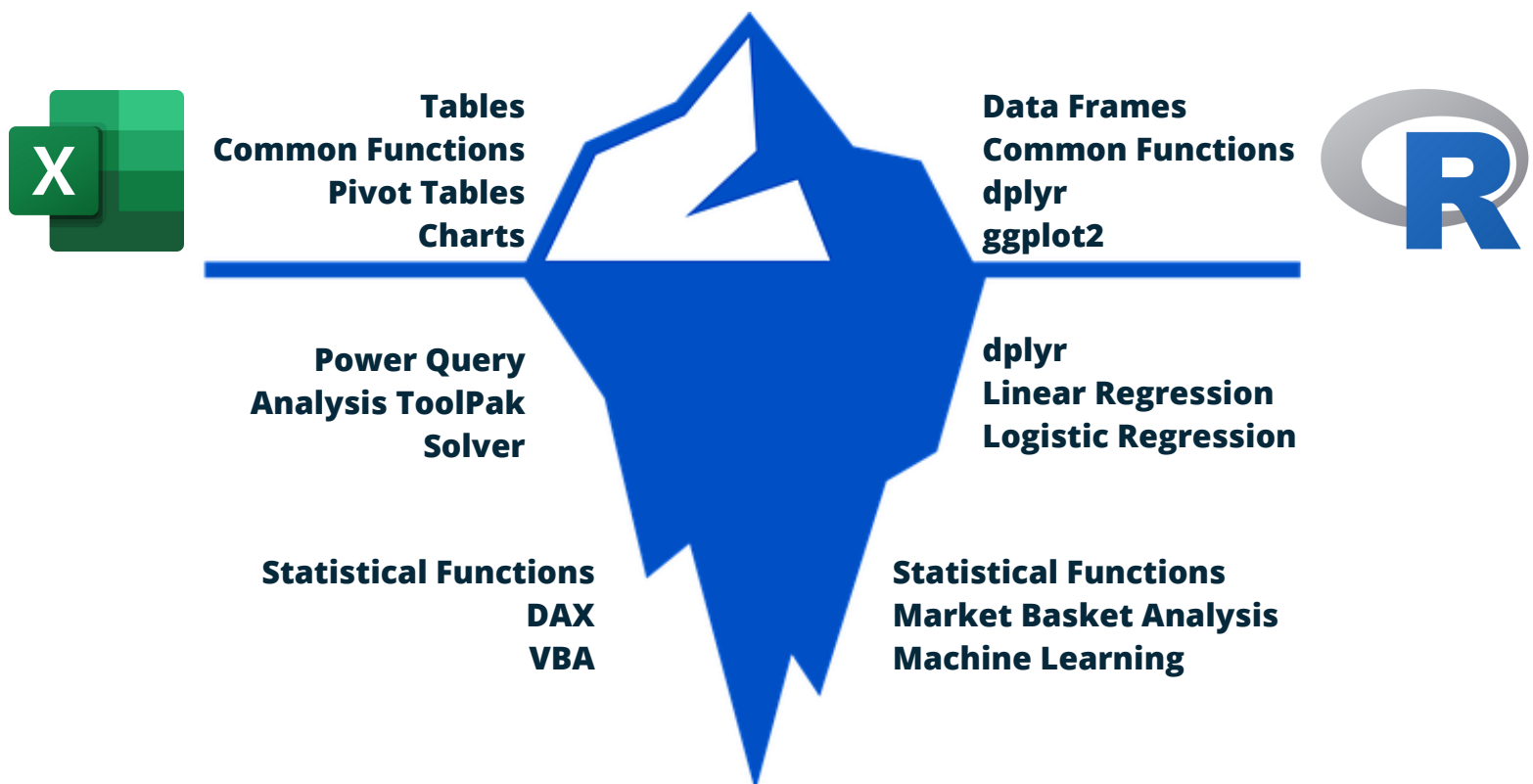
This knowledge makes the learning process an exercise in mapping Excel skills to R.

Excel and R Are Like Icebergs

While cliché, 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 features 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.

The screenshot shows an Excel table with the following data:

	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

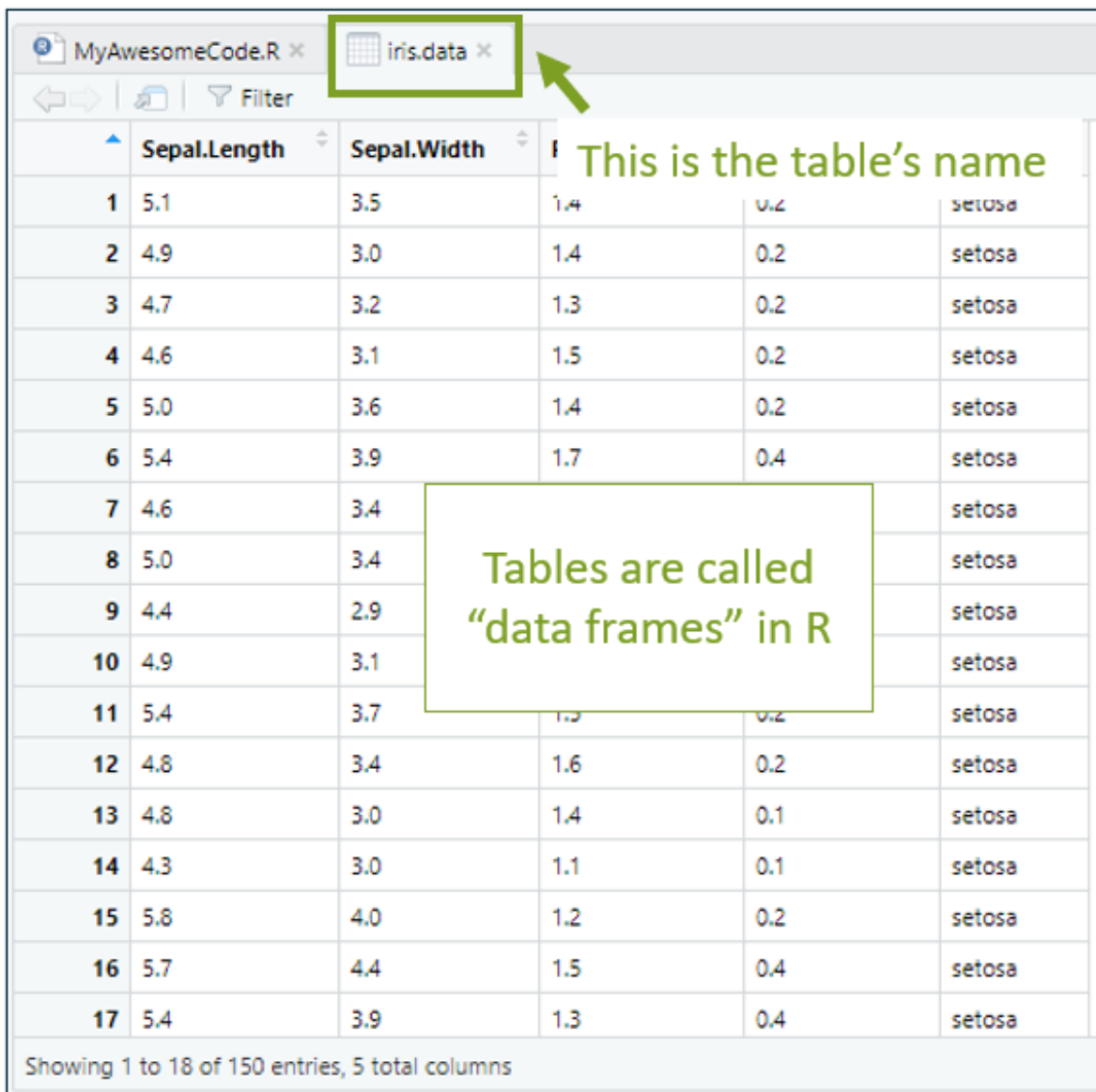
Annotations in the image:

- A green box highlights the "Table Name:" field in the ribbon, which contains the text "iris.data".
- A green arrow points from the text "This is the table's name" to the "Table Name:" field.
- A green box highlights the "iris.data" label in the bottom status bar.
- A green arrow points from the text "Not this!" to the value "4" in cell B16.

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)`



This is the table's name

Tables are called "data frames" in R

	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			setosa
8	5.0	3.4			setosa
9	4.4	2.9			setosa
10	4.9	3.1			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*.

iris.data table

4th column of the table

A	B	C	D	E
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

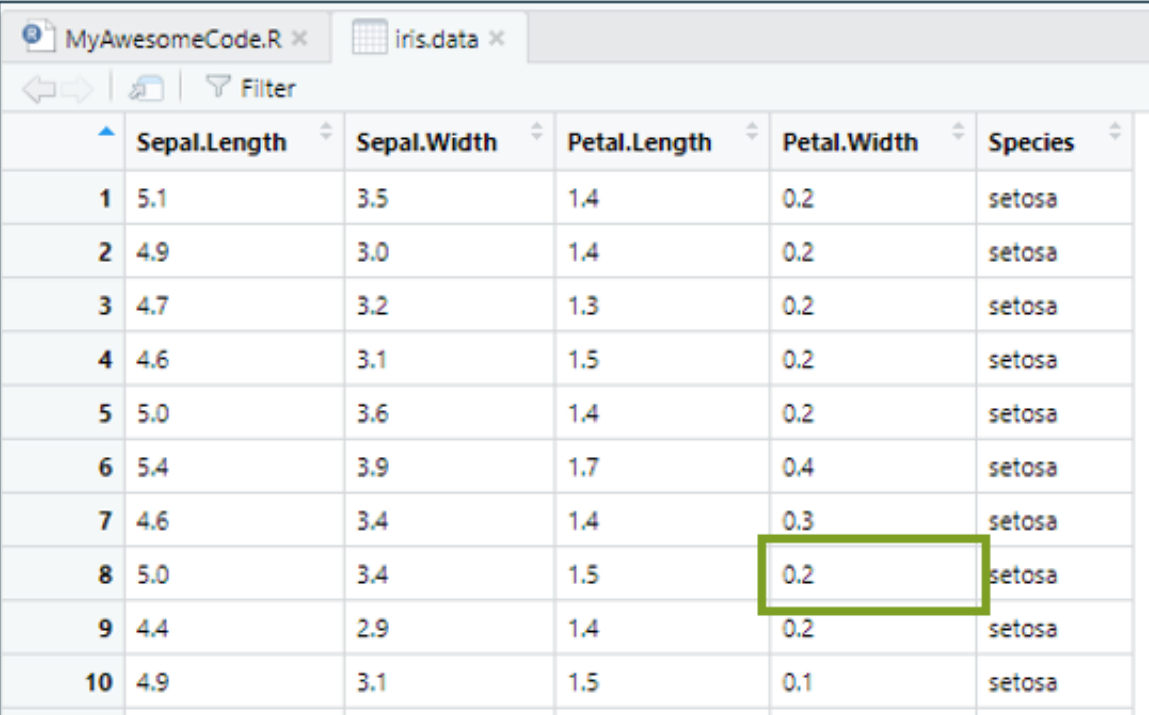
8th row of the table

This is cell D9 of the worksheet

F	G	H
	=D9	

F	G	H
	0.2	

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
data frame

8th row

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

4th column

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
Petal.Length	Petal.Width	Species		
1.4	0.2	setosa		
1.4	0.2	setosa		
1.3	0.2	setosa		
1.5	0.2	setosa		

`=SUM(D2:D151)`

Sum the column

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

C	D	E	F	G	H
Petal.Length	Petal.Width	Species			
1.4	0.2	setosa			
1.4	0.2	setosa			
1.3	0.2	setosa			
1.5	0.2	setosa			

179.9

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

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

Sum the column

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
```



All the rows

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



Sum the
column

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.

New table column

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

Excel's IF function

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

Excel automatically applies the IF function to every Species value.

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.

```
> iris.data$IsSetosa
```

The table Table column
"access the table"

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

store data into
R's ifelse function
Apply ifelse function to every Species value

R is smart.
"IsSetosa" doesn't exist, so R adds it.

```
> 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.

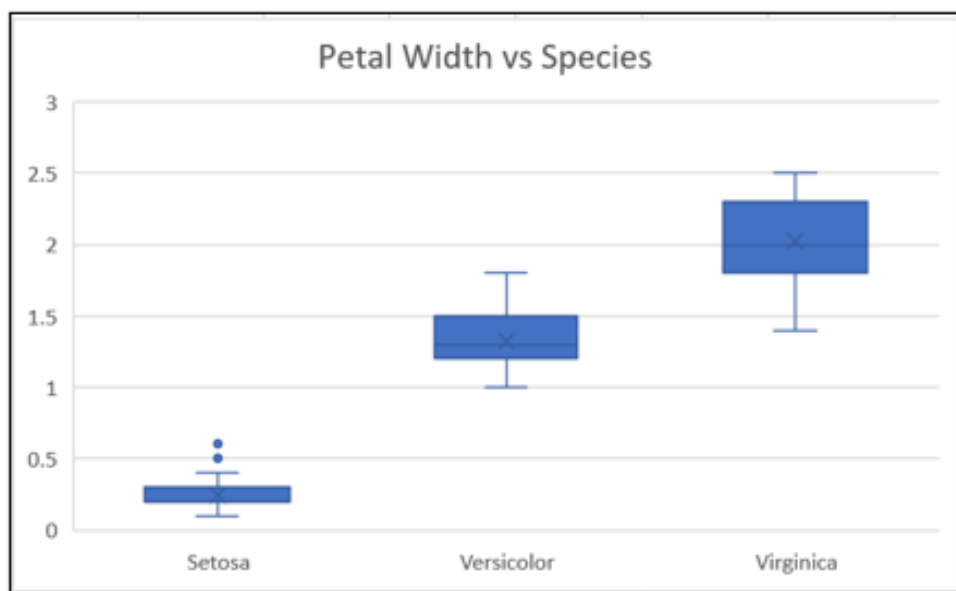
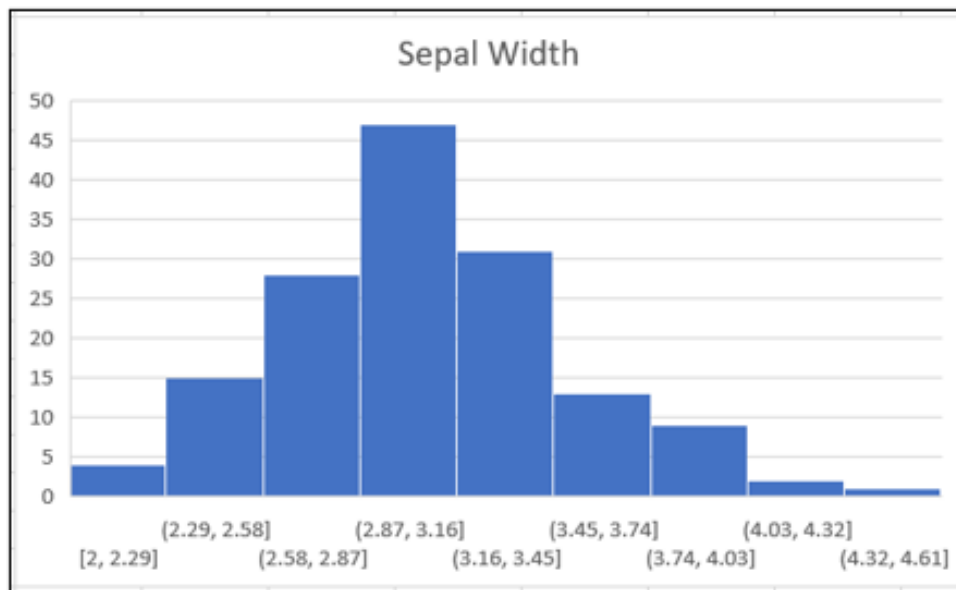
E	F	G	H
Species ▾			
setosa			
setosa		MIN(iris.data[Petal.Width])	0.1
setosa		AVERAGE(iris.data[Petal.Width])	1.199333
setosa		MEDIAN(iris.data[Petal.Width])	1.3
setosa		MAX(iris.data[Petal.Width])	2.5
setosa		PERCENTILE.EXC(iris.data[Petal.Width], 0.25)	0.3
setosa		STDEV.S(iris.data[Petal.Width])	0.762238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

```
> 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.

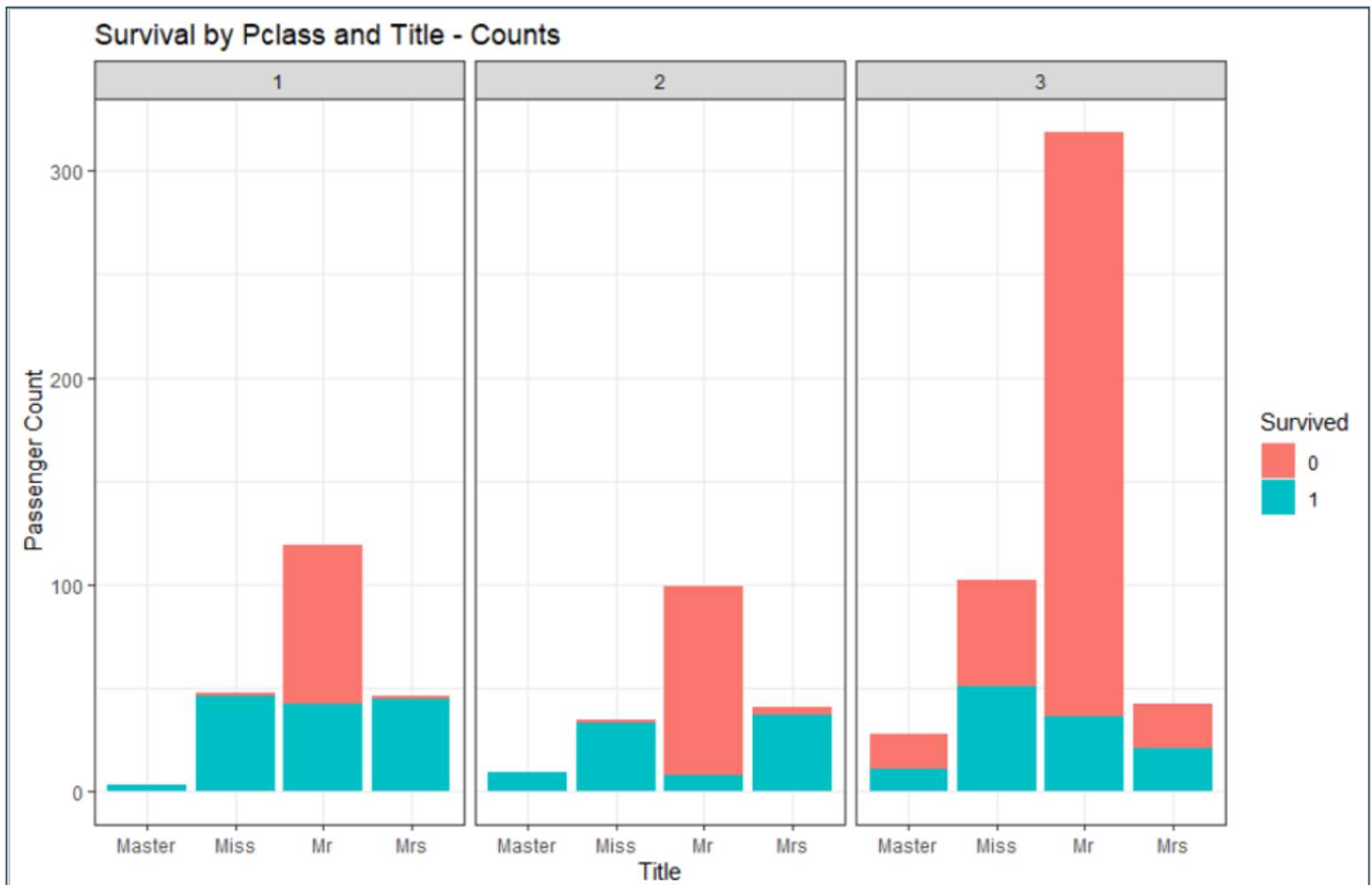


R Data Visualizations

R is renowned for its ability to create print-quality data visualizations.

R also easily produces data visualizations that are difficult, or not possible, to do without the Excel features.

Analyzing data visually is one of R's specialties.

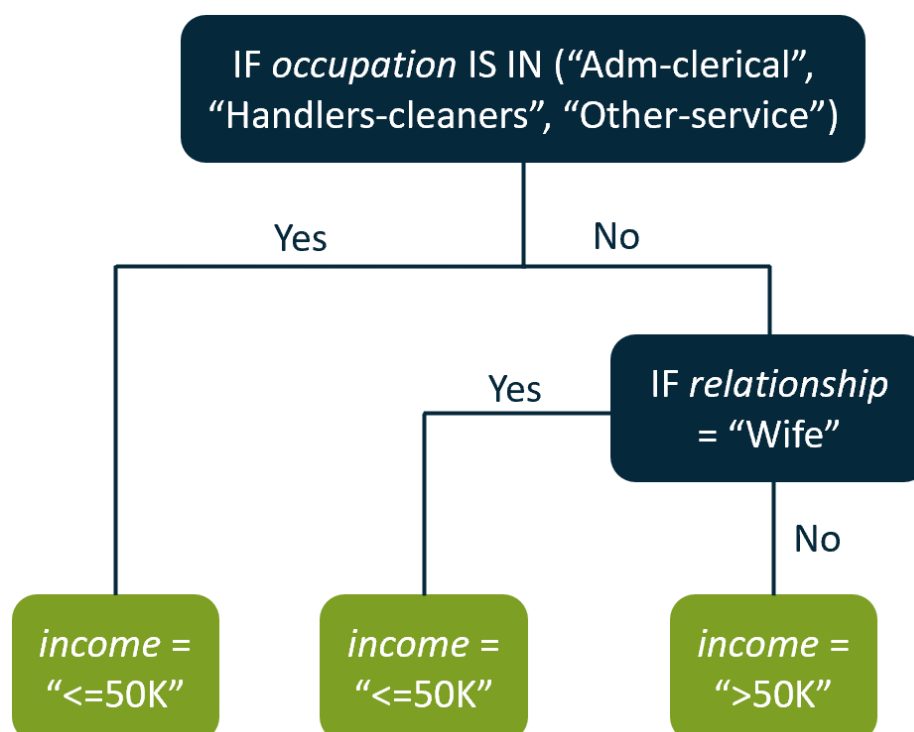


Classification Tree Intuition

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.



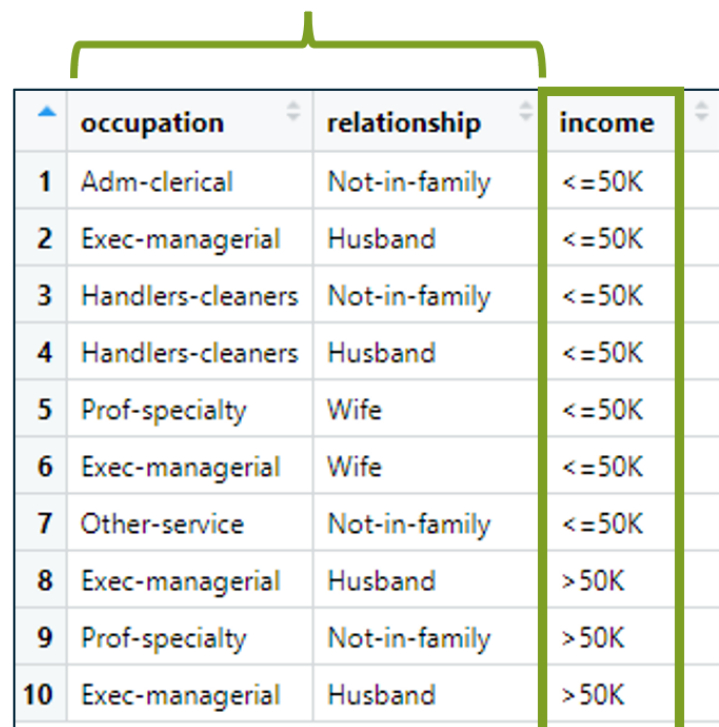
Trees Are Rules

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*.

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

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.

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**

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

These are “impure”

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

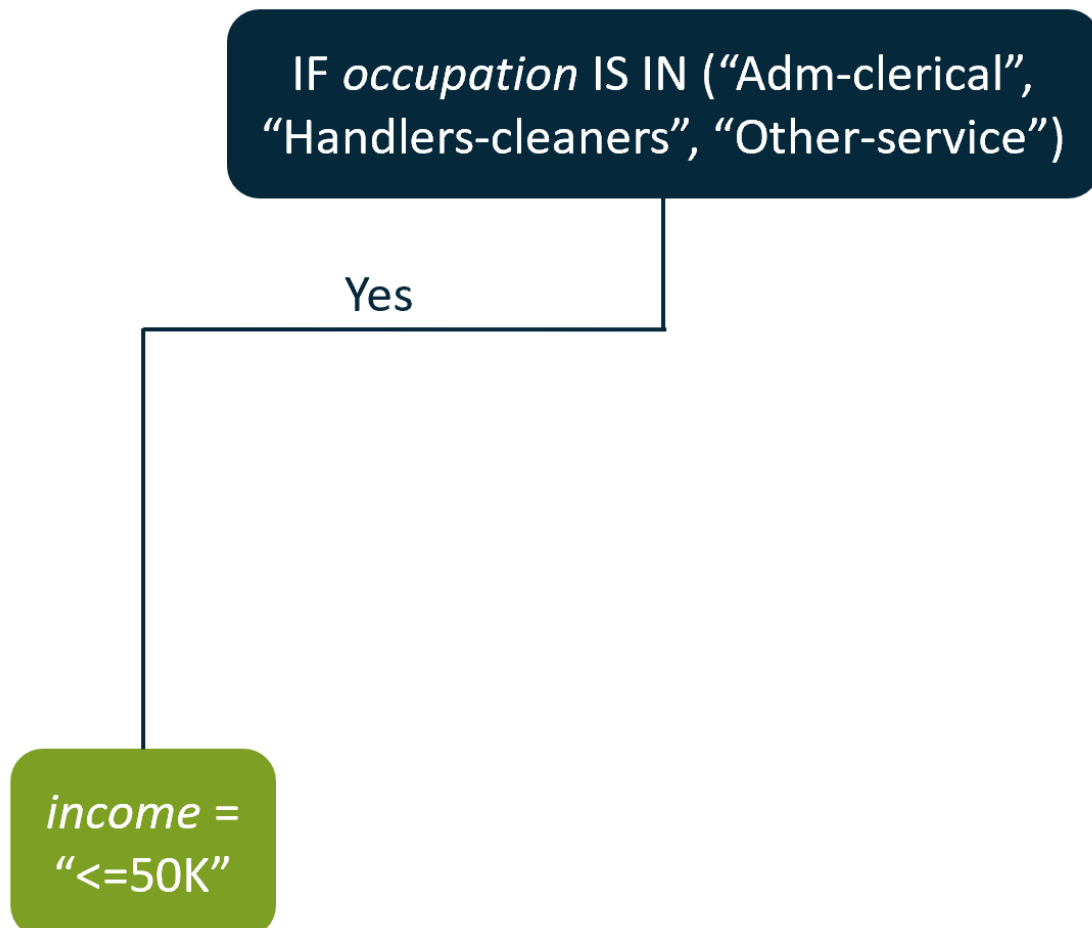
With the *relationship* feature we get 2 observations all with the label “≤50K”

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!



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.

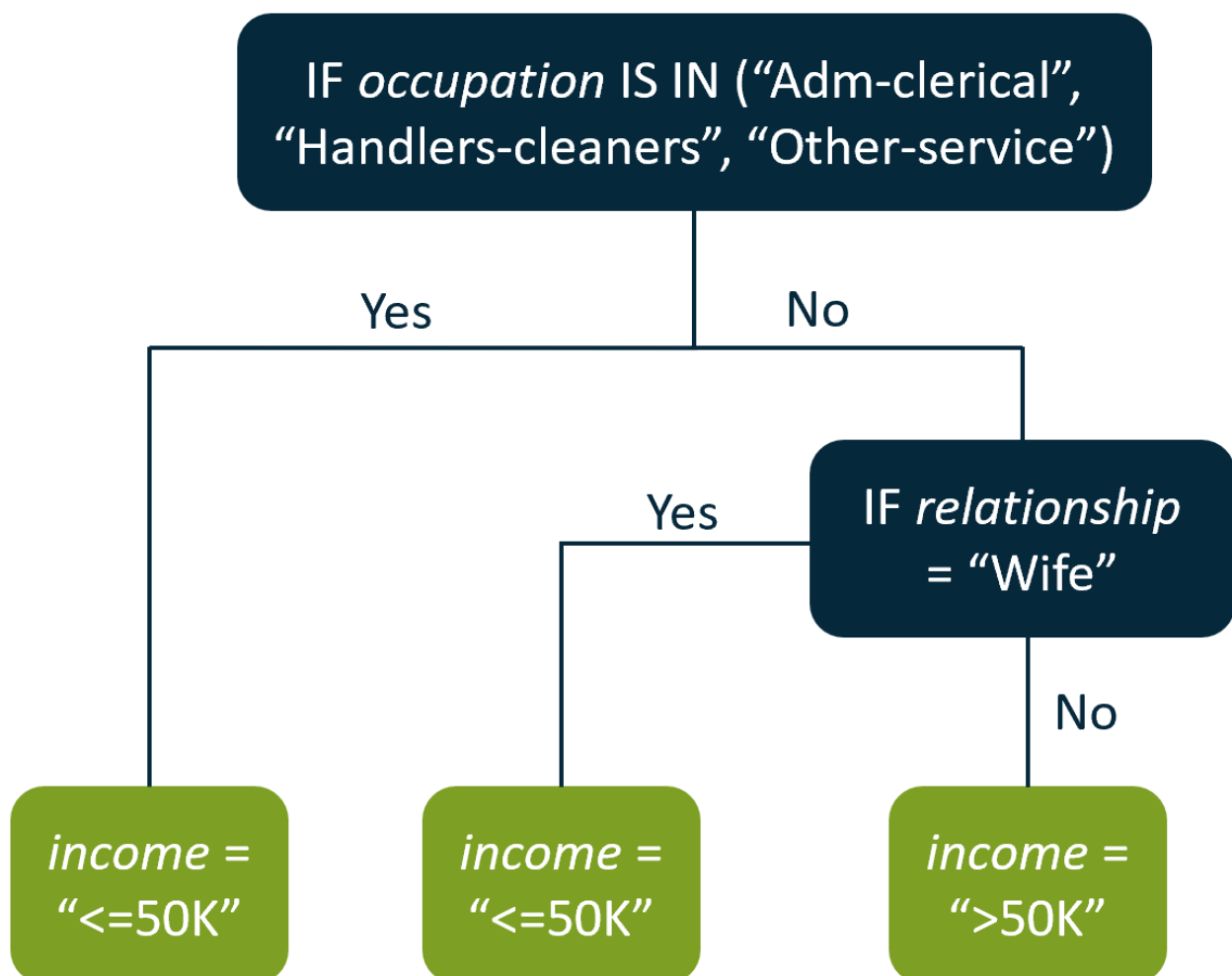
	occupation	relationship	income
1			
2	Exec-managerial	Husband	<=50K
3			
4			
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7			
8	Exec-managerial	Husband	>50K
9	Prof-specialty	Not-in-family	>50K
10	Exec-managerial	Husband	>50K

We know this split is pure!

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.



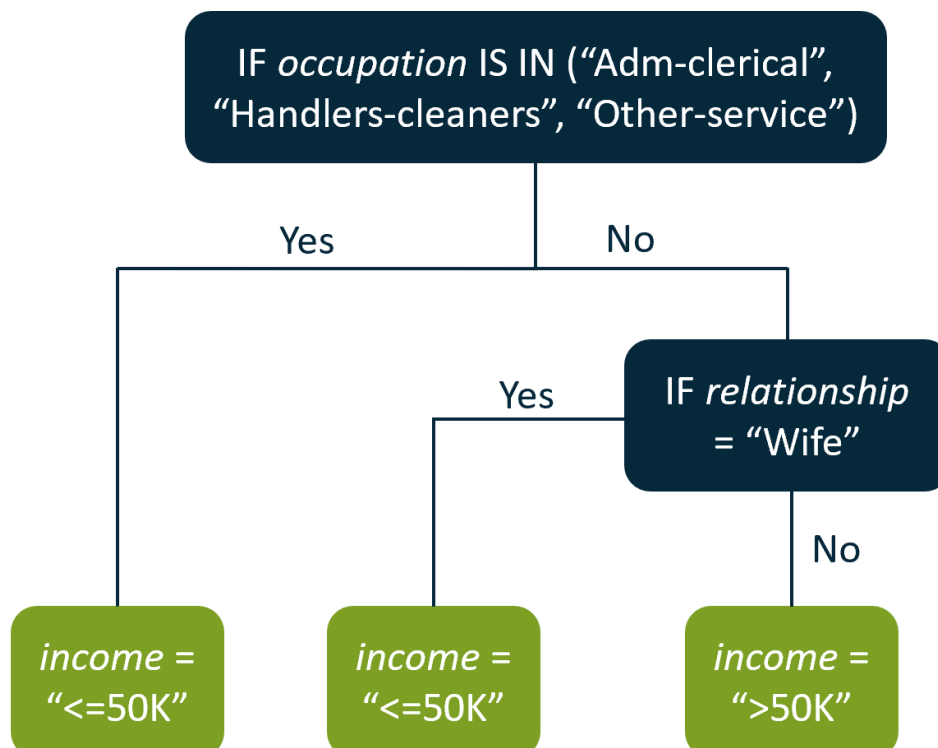
Overfitting Intuition

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 course bundle covers overfitting in depth, this document will focus on the intuition. Take the tree from the previous page as an example.

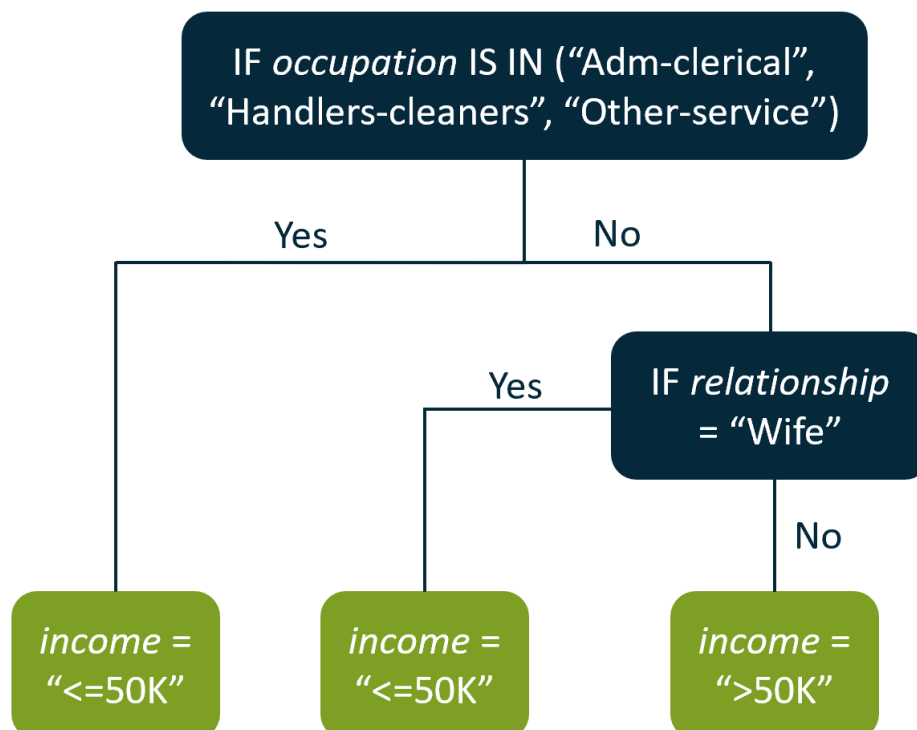


	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty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



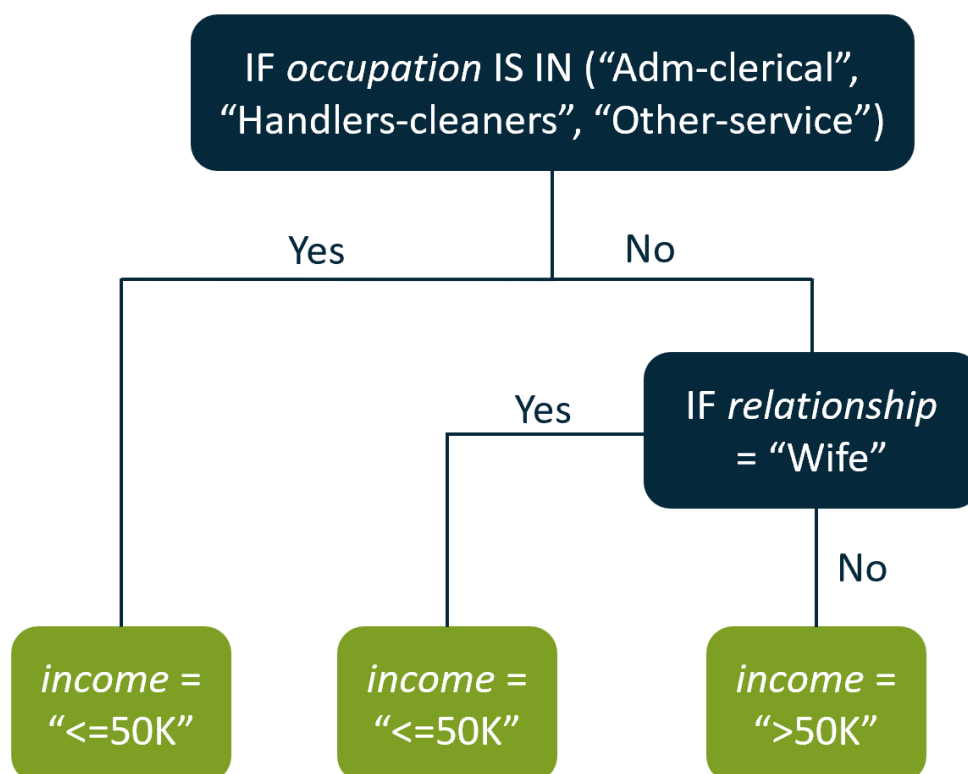
	occupation	relationship	income
1	Prof-specialty	Wife	>50K
2	Adm-clerical	Wife	>50K
3	Exec-managerial	Wife	>50K
4	Other-service	Husband	>50K
5	Other-service	Husband	>50K
6	Other-service	Wife	>50K
7	Exec-managerial	Husband	<=50K
8	Sales	Not-in-family	<=50K
9	Transport-moving	Husband	<=50K

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.



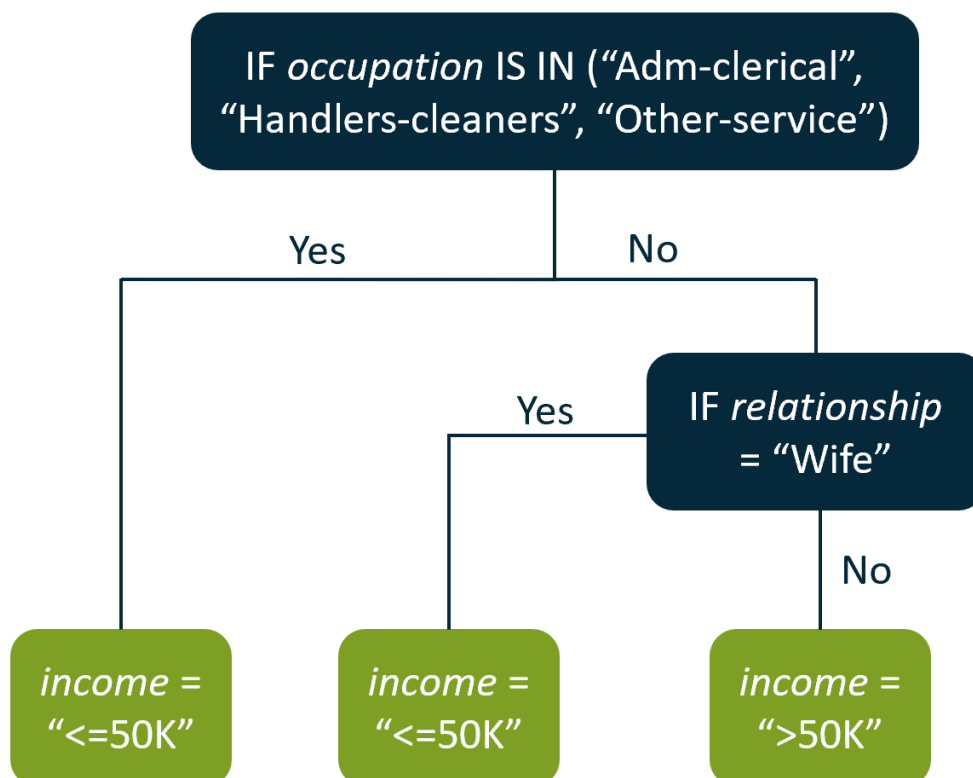
	occupation	relationship	income
1	Adm-clerical	Not-in-family	<=50K
2	Exec-managerial	Husband	<=50K
3	Handlers-cleaners	Not-in-family	<=50K
4	Handlers-cleaners	Husband	<=50K
5	Prof-specialty	Wife	<=50K
6	Exec-managerial	Wife	<=50K
7	Other-service	Not-in-family	<=50K
8	Exec-managerial	Husband	>50K
9	Prof-specialty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



Model Tuning Intuition

The course bundle 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

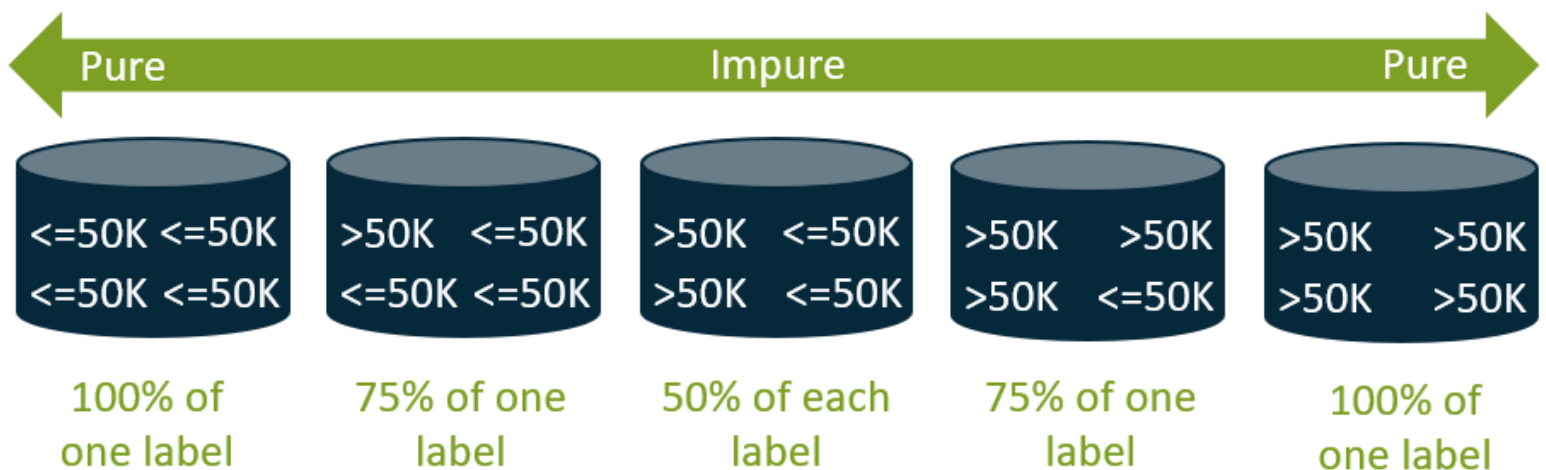
The course bundle has robust coverage of the mathematics of decision trees. Here's the good news:

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...

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



Gini Impurity

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

The course bundle uses the *Gini impurity* calculation.

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

Don't panic!

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

The diagram illustrates the Gini impurity formula with several annotations in green text and arrows:

- "Go through all the class labels"**: An arrow points to the summation symbol \sum in the formula.
- "Square the proportion"**: An arrow points to the squared term $[p(i|t)]^2$ in the formula.
- "Subtract the stuff to the right from 1"**: An arrow points to the minus sign $-$ in the formula.
- "Proportion of class labels"**: A bracket under the term $p(i|t)$ points to this annotation.

The formula itself is:

$$Gini(t) = 1 - \sum_{i=1}^c [p(i|t)]^2$$

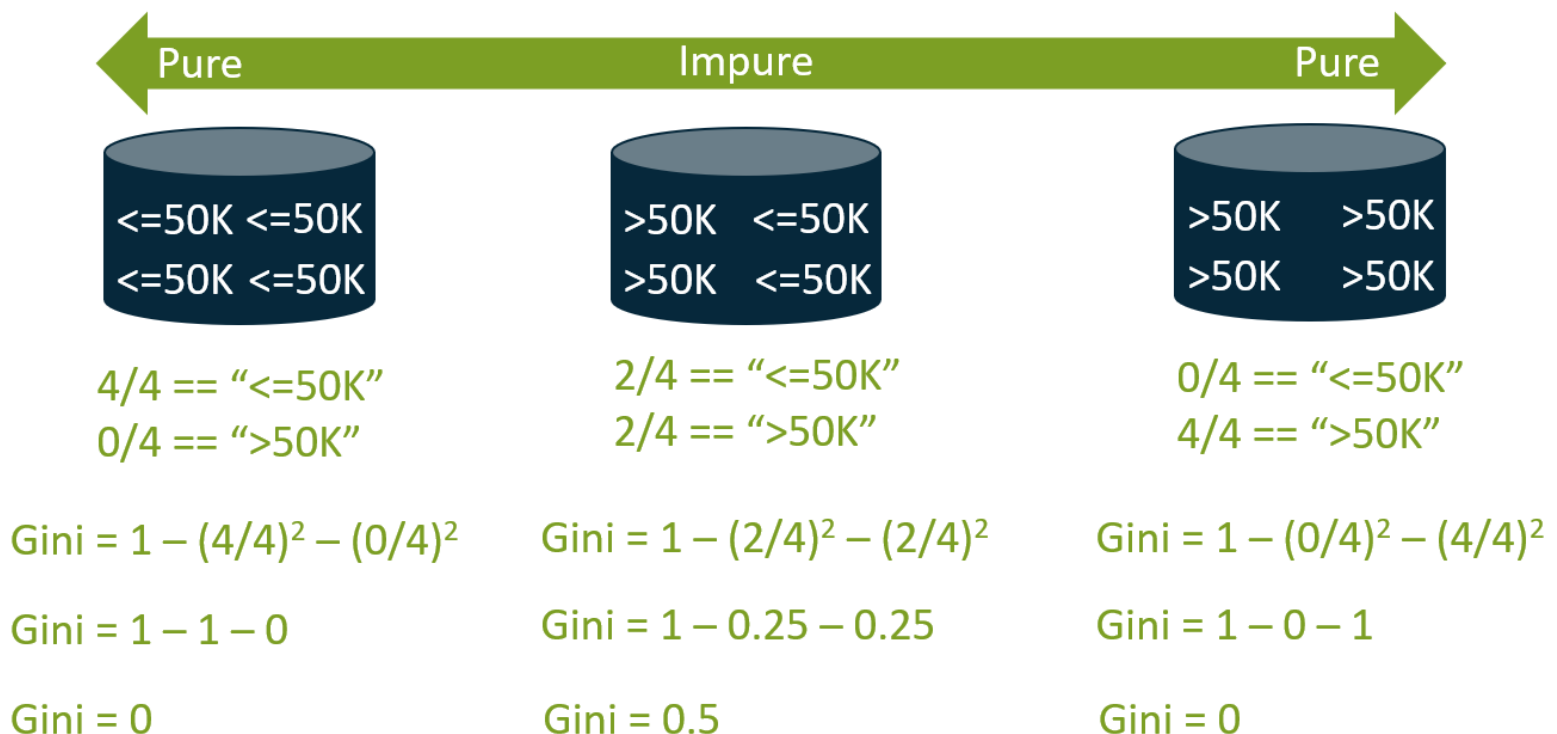
Gini Impurity Example

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.



R Programming Made Easy

Section 1 - Course Introduction

- Welcome
- The Data

Section 2 - What Is R and RStudio?

- A Brief History of R
- Excel and RStudio
- Windows Installation
- Mac Installation
- Exercise #1 - R and RStudio Basics

Section 3 - It's All About the Objects

- A World of Objects
- First There Were Tables
- Exercise #2 - Basic Tables
- Then There Were Vectors
- Vectors Have Types
- Math With Vectors
- Exercise #3 - Fun With Vectors

Section 4 - Data Frame Filtering

- Filtering With Logic
- Filtering the R Way
- Exercise #4 - More Tables

Hands-on Lab #1 - Titanic First Look

- Lab Instructions
- Lab Walkthrough

Section 5 - Throw in Some Functions

- Missing Data
- Common Stats Functions
- The summary Function
- The data.frame Function
- The cbind & rbind Functions
- The aggregate Function
- Exercise #5 - Some Functions

Hands-on Lab #2 - Working Titanic Data

- Lab Instructions
- Lab Walkthrough

Section 6 - Pivoting Data With dplyr

- Introducing dplyr
- Tibbles
- Mutating Data
- Selecting and Filtering Data
- Exercise #6 - Basic dplyr
- Grouping and Summarizing Data
- Joining Data
- Arranging Data
- Exercise #7 - More dplyr

Hands-on Lab #3 - Pivoting Titanic Data

- Lab Instructions
- Lab Walkthrough

Section 7 - Visualizing Data With ggplot2

- Introducing ggplot2
- Boxplots
- Histograms
- Bar Charts
- Scatter Plots
- Exercise #8 - ggplot2

Hands-on Lab #4 - Visualizing Titanic Data

- Lab Instructions
- Lab Walkthrough

Section 8 - Course Wrap-Up

- When You Get Stuck
- Continue Your Learning

Introduction to Machine Learning With R

Section 1 - Welcome

- Welcome to the Course!

Section 2 - Supervised Learning

- Data Analyst, Teacher
- Why Trees?

Section 3 - The Data Sets

- The Data
- Exploratory Data Analysis (EDA)

Hands-on Lab #1 - Titanic EDA

- Lab Instructions
- Lab Walkthrough

Section 4 - Classification Trees

- Classification Tree Intuition
- Overfitting Intuition
- Gini Impurity
- Gini Change
- Many Categories Impurity
- Numeric Feature Impurity
- Classification Trees With Tidymodels

Hands-on Lab #2 - Titanic Classification Tree

- Lab Instructions
- Lab Walkthrough

Section 5 - Awesome Classification Trees

- Under/Over Fitting
- The Bias-Variance Tradeoff
- Supervising the Data
- Model Tuning Intuition
- Classification Tree Pruning
- Measuring Awesomeness
- Model Tuning With Tidymodels

Hands-on Lab #3 - Titanic Tree Pruning

- Lab Instructions
- Lab Walkthrough

Section 6 - Feature Engineering

- Feature Engineering Intuition
- Data Leakage
- Decision Tree Feature Engineering
- Missing Data

Hands-on Lab #4 - Titanic Feature Engineering

- Lab Instructions
- Lab Walkthrough

Section 7 - Regression Trees

- Regression Tree Basics
- Numeric Feature SSE
- Many Categories SSE
- Regression Trees With Tidymodels

Section 8 - The Mighty Random Forest

- Bad, Tree! Bad!
- Ensembles
- Bagging
- Feature Randomization
- Tuning Random Forests
- Feature Importance
- Random Forests With Tidymodels

Hands-on Lab #5 - Titanic Random Forest

- Lab Instructions
- Lab Walkthrough

Section 9 - Course Wrap-Up

- Want to Kaggle?
- Additional Resources

You've Got This if You Want It

If you've read this document, I hope I've convinced you that ANY professional can acquire valuable machine learning skills.

Over the years I've helped 1000s of professionals gain skills in R and machine learning. So you can believe me when I say this - you've got this if you want it.

I have a passion for teaching and I want all of my students to be successful in applying what they learn from my live and online classes in their daily work.

Please know that I'm here to help.

I actively encourage my students to ask any and all questions via the online course platform. I obsessively monitor inbound student questions and answer them promptly - as Joseph Kamau mentioned in his testimonial at the beginning of this document.

Many of my students also find it beneficial to augment courses with dedicated 1-on-1 coaching. I help my coaching clients to successfully apply what they have learned in their work - their business problems and their data.

If you're interested in learning more about coaching, you can find more information using the link below.

If you decide that my "From Excel to Machine Learning" self-paced online course bundle is right for you, be sure to use coupon **LINKEDIN** at checkout to save an additional 20% off the purchase price.

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