An Introduction to Predictive Modeling in R

Ryan Benz • OCRUG Hackathon 2019 Tutorial

May 18, 2019

Build Something Useful!

- Predictive modeling: the process of combining data and algorithms in order to build *useful* models

- In contrast with explicitly programming rules, predictive modeling algorithms attempt to *learn patterns from the data itself*

- Predictive modeling has deep mathematical foundations, but in the end, it’s extremely practical

Predictive Modeling is Everywhere

- Is this email message spam?
- Will this person default on their loan?
- Which other products might this person also buy?
- Is that a cat?
- Which group of people should I target for my ad campaign
- Is this person sick or health?

Lots of Contexts, Lots of Terms

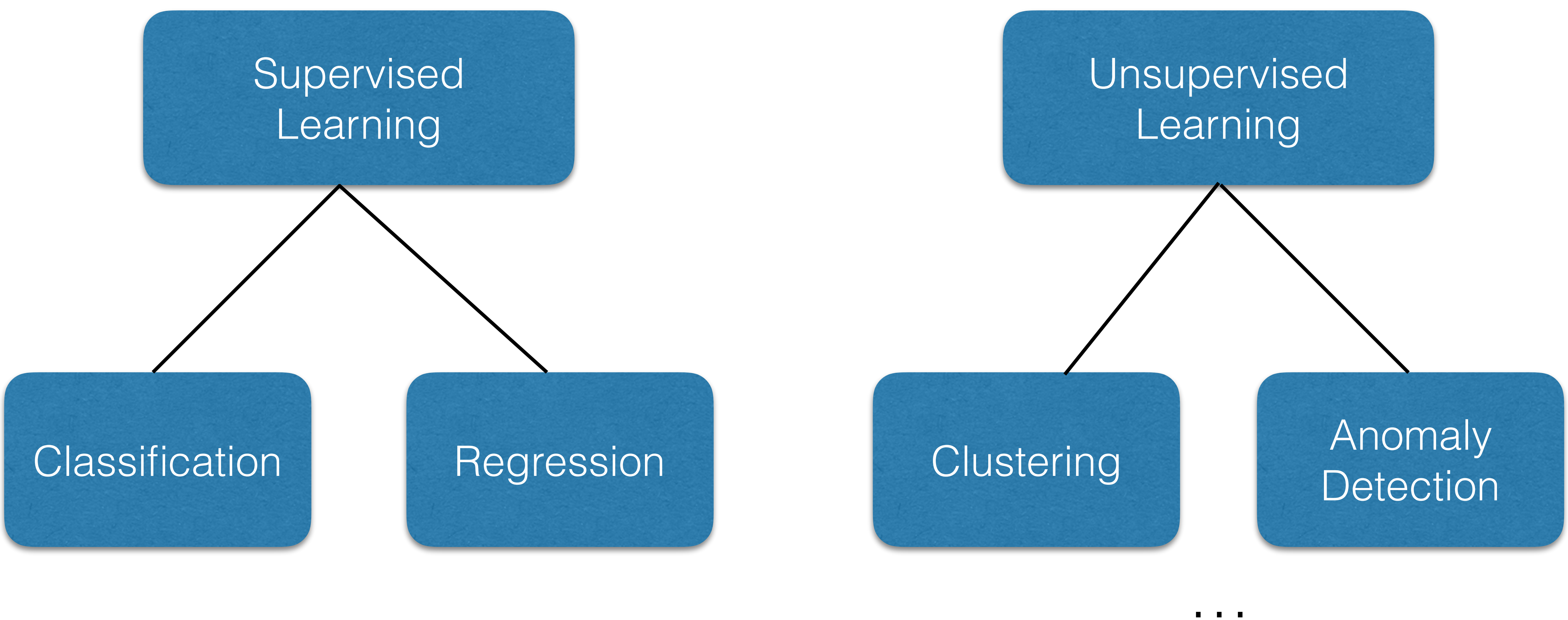
- People have been predictively modeling for a long time, and in lots of different fields
- Therefore, lots of different terms used for similar things

The Subject	**The Data**	**The Outcomes**
Predictive modeling	Features	Classes
Predictive analytics	Predictors	Labels
Machine learning	(Independent) Variables	Dependent Variables
Data mining	Measures	Responses
Statistics	Attributes	Targets

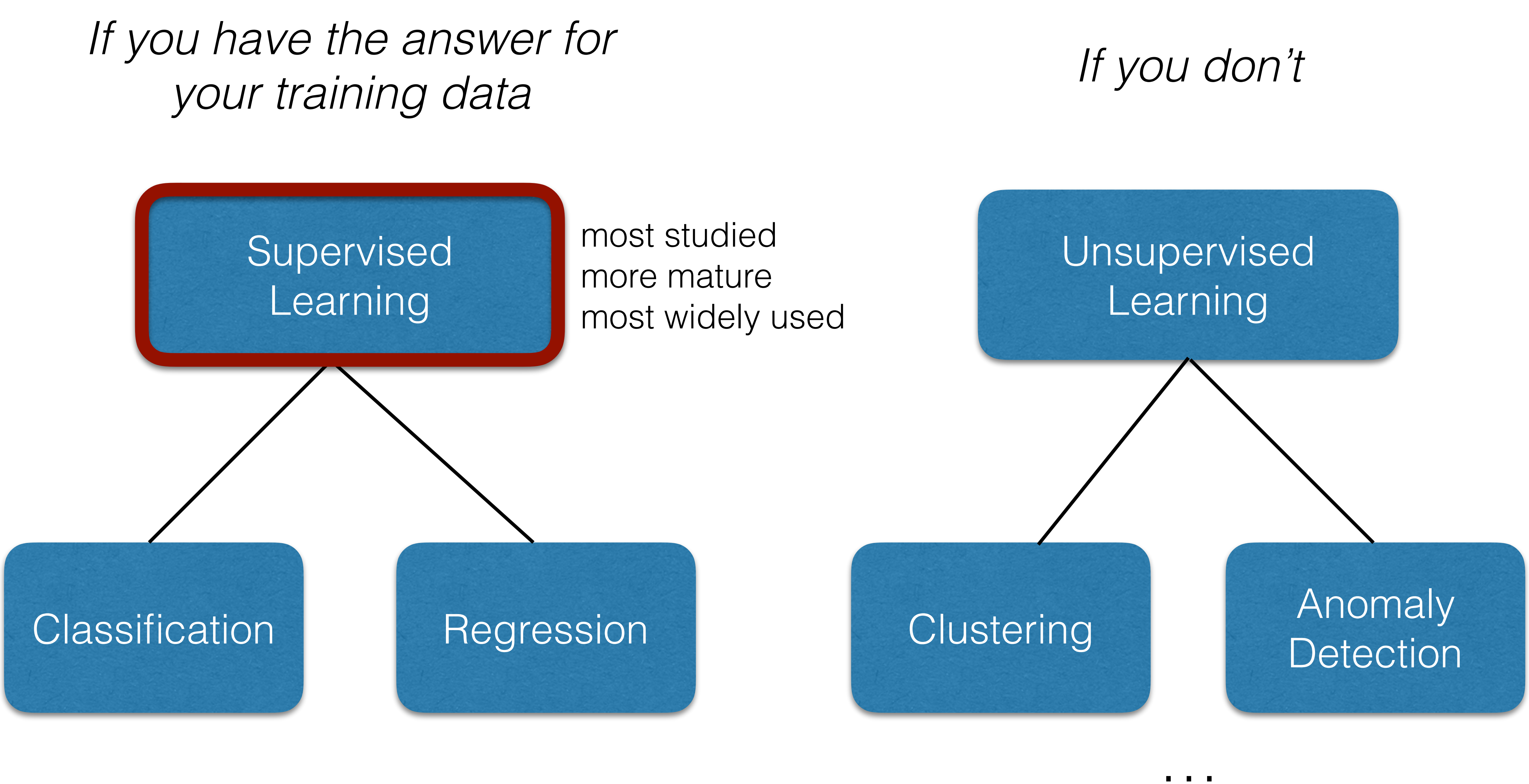
Two Main Branches of Machine Learning

If you have the answer for your training data

If you don't



Two Main Branches of Machine Learning



The *highly simplified* Model Building Process

Start with a question

Which stocks should I buy?
(How can I determine good & bad stocks?)

Collect relevant data

Amass historical stock/company data
WITH good/bad calls

Train a model

Model → *computery stuff* → Trained Model

Input new data → Trained Model → Good Stocks / Bad Stocks

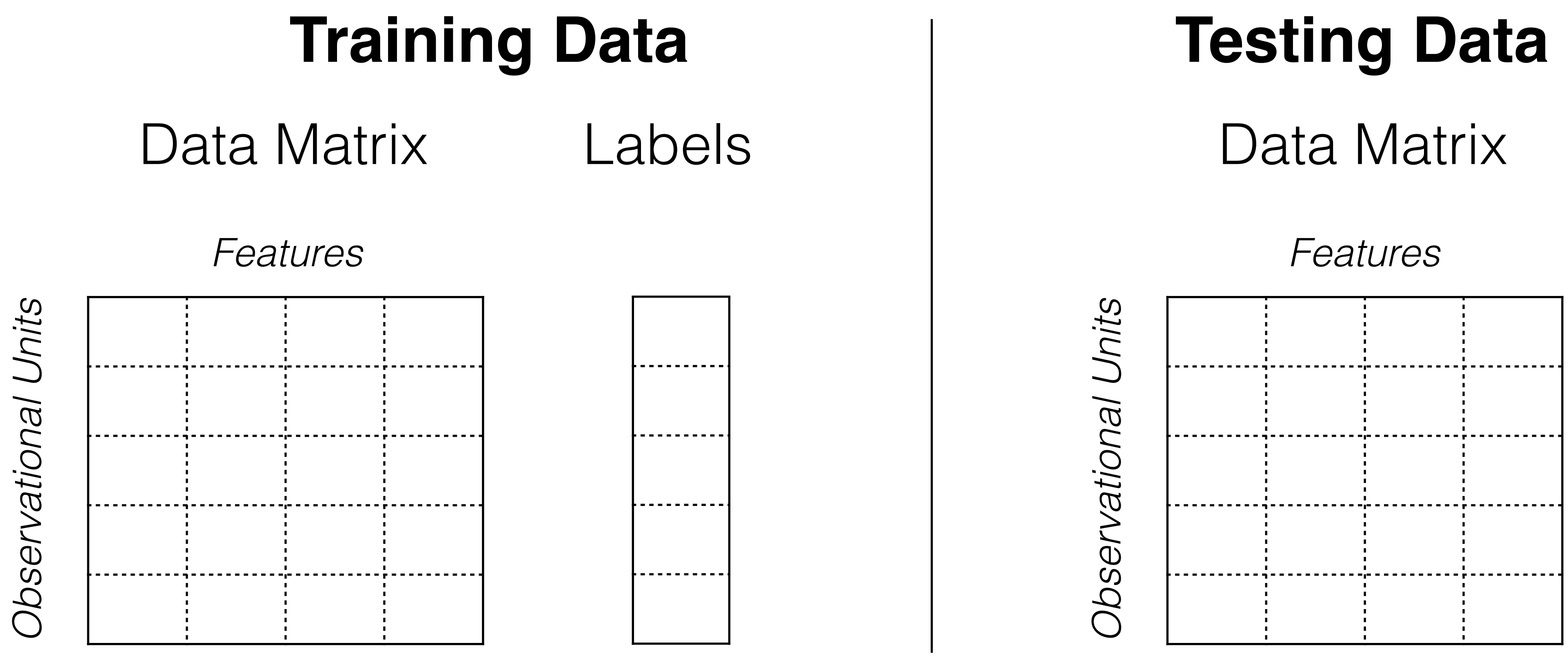
Good Stocks → Invest → Get RICH!!!

Building Models with R and caret

Modeling in R

- R has 100’s of modeling packages; if you know about it, there’s probably an R package for it
- Lots (most?) modeling packages follow a somewhat standard way to work with models
    - train a model: `model_func(training_matrix, training_labels, …)`
    - make predictions: `predict(model_obj, testing_matrix)`
- However, there are often subtle differences between packages so you have to be careful & read the documentation

Building Models



Model Training `model_func(training_matrix, training_labels, …)`

Model Predictions `predict(model_obj, testing_matrix)`

Some Examples

e1071

```
svm(train_mtrx, train_lbls, probability = TRUE, ...)
predict(model_obj, test_mtrx, probability = TRUE)
```

randomForest

```
randomForest(train_mtrx, train_lbls, ...)
predict(model_obj, test_mtrx, type = ”prob”)
```

stats

```
glm(formula, ...)
predict(model_obj, test_mtrx, type = ”response”)
```

Can you spot the differences?

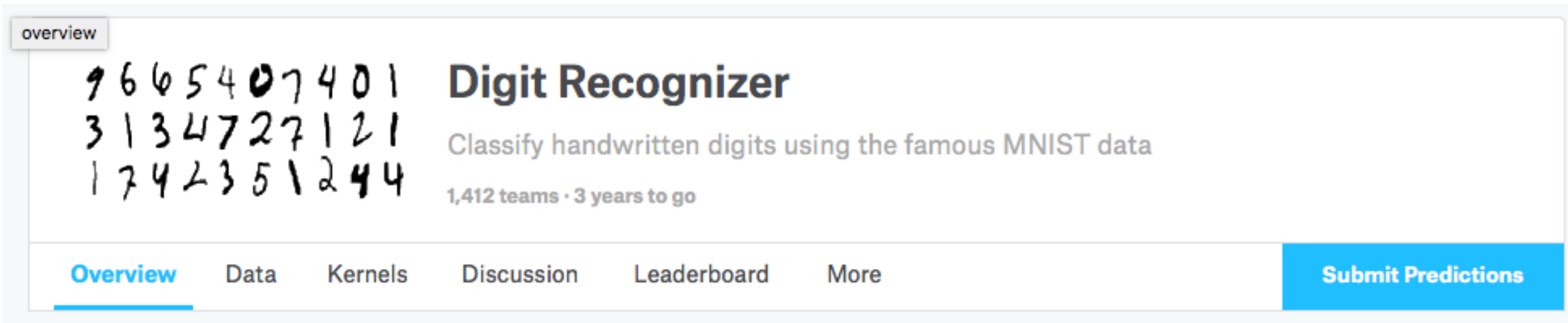
Tips on Working with Models

- For a centralized listing of many of the models in R, check out the model listing on the caret repo
http://topepo.github.io/caret/available-models.html

- Model training functions are typically named after the model (see previous slide)

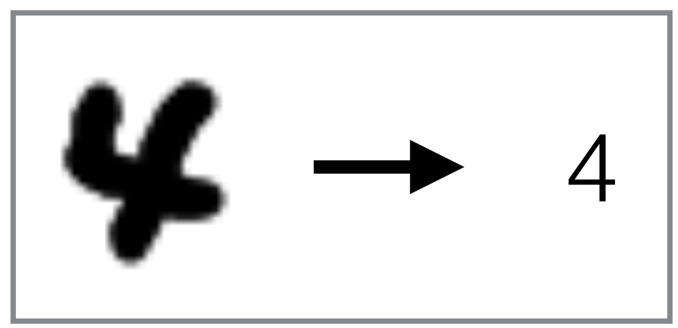
- Use the documentation to remind yourself of the function arguments and what they mean
- e.g. `?svm`, `?randomForest`, `?glm`
- for most predict functions use:
`?predict.svm,?predict.randomForest,?predict.glm`

A Real Example:
Kaggle Digit Classification Competition



Task

Given an image of a handwritten digit, determine which one it is



Training Data

A vector of length 785 for each example (digit)

- first entry is the label (a digit 0 - 9)
- the remaining 784 entries are each numbers 0 - 255 (black … white) representing a 28 x 28 gray-scale image of the digit

e.g.: 3,0,0,0,27,59,82,171,201,163,74,30,0,0…0,0,0

Testing Data

A vector of length 784 for each *new* example; NO LABELS

Submission

```
ImageId,Label
1,3
2,7
3,8
(27997 more lines)
```

https://www.kaggle.com/c/digit-recognizer

A Real Example:
Kaggle Digit Classification Competition

Code

Download Code

```r
# Creates a simple random forest benchmark

library(randomForest)
library(readr)

set.seed(0)

numTrain <- 10000
numTrees <- 25

train <- read_csv("../input/train.csv")
test <- read_csv("../input/test.csv")

rows <- sample(1:nrow(train), numTrain)
labels <- as.factor(train[rows,1])
train <- train[rows,-1]

rf <- randomForest(train, labels, xtest=test, ntree=numTrees)
predictions <- data.frame(ImageId=1:nrow(test), Label=levels(labels)[rf$test$predicted])
head(predictions)

write_csv(predictions, "rf_benchmark.csv")
```

show less

Most of the code is about data prep!

One line to build the model, one line to make the predictions

This model is 93.5% accurate

https://www.kaggle.com/c/digit-recognizer

The caret Package

- **C**lassification **A**nd **Re**gression **T**raining

- Provides a uniform interface for working with most of R’s modeling packages and a bunch of tools to streamline the modeling process

- *Pros*: takes care of the details for you, can help you avoid modeling mistakes

- *Cons*: can make modeling even more black-boxy, particularly for new users

Training, Tuning & Evaluating Models

- Training: the process of fitting a model based on supplied data
- Model method: the underlying algorithm used in the training process
- Model parameters: adjustable parameters associated with a given modeling method that affect how the model is trained and the model output
- Model tuning: the process of adjusting the model parameters to find the ones that give the “best” performance
- Resampling: a process where you split your data into partitions, typically ones for training your model, and ones for evaluating it

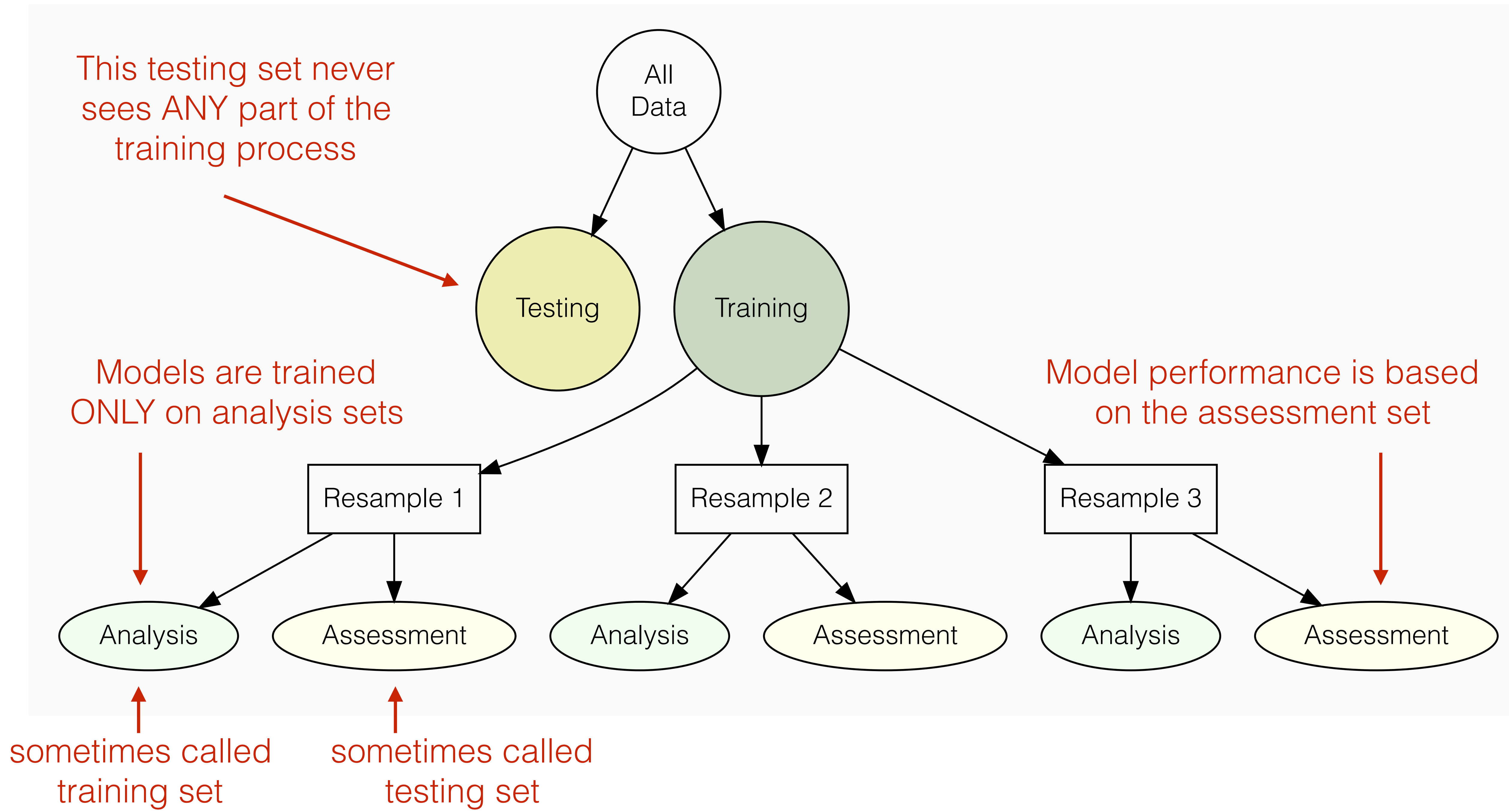
Resampling

- Lots of commonly used models have the flexibility to completely describe your training data

- Model performance on your training data is often over-optimistic, does represent how well the model will generalize to new data

- Resampling can be used to help address this problem, e.g.
  - cross-validation
  - random splits

Resampling



https://github.com/topepo/rstudio-conf-2019

A Model Training Workflow

Caret will take care of all of this

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

http://topepo.github.io/caret/model-training-and-tuning.html

Building and Assessing Models with caret

- caret can automatically choose *parameter sets* and optimize them within a *resampling* approach

- Step 1: define a `trainControl` object
  - resampling method
  - how to evaluate performance
  - other model specific options

- Step 2: perform model training workflow with `train`

- Step 3: review performance and select “best” model

Main Code

```r
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = twoClassSummary)
```

```r
model_fit <- train(Class ~ ., data = training,
                   method = "gbm",
                   trControl = fitControl,
                   verbose = FALSE,
                   tuneGrid = gbmGrid,
                   ## Specify which metric to optimize
                   metric = "ROC")
```

(live example)

`caret_example.R`

Some Thoughts About Building Predictive Models

- Ensuring your model is going to work on new, unseen data is really important
  - Is your training data representative of the new data?
  - Use resampling methods (e.g. cross validation) to estimate generalization performance

- Information “leakage” can ruin your model, is often subtle and not immediately evident; be careful

- Learning the mathematical/statistical details of various modeling algorithms and methods can be useful, though…

- It’s usually advantageous to spend time understanding the problem domain, finding relevant data

- Predictive modeling is very practical, and you get good at it through lots of practice

Resources

- THE Book by Kuhn & Johnson
  Applied Predictive Modeling
  http://appliedpredictivemodeling.com

- New Book by Kuhn & Johnson
  Feature Engineering and Selection: A Practical Approach for Predictive Models
  http://www.feat.engineering

- Other books
  - Elements of Statistical Learning (Hastie, et.al.)
  - Pattern Recognition and Machine Learning (Bishop)
  - Data Mining with R: Learning with Case Studies (Torgo)

Resources

- R Packages
  - 100's of modeling packages are available (e.g. `e1071`, `randomForest`, `glmnet`)
  - **`caret`**: addresses the entire modeling workflow
    http://topepo.github.io/caret/index.html
  - `tidymodels`, `parsnip`, etc…

- Where to Practice
  - Kaggle (www.kaggle.com)
  - Flowing Data (https://flowingdata.com/category/statistics/data-sources/)
  - UCI Machine Learning Repository
    (http://archive.ics.uci.edu/ml/index.php)