ELDER RESEARCH
DATA SCIENCE · AI · MACHINE LEARNING

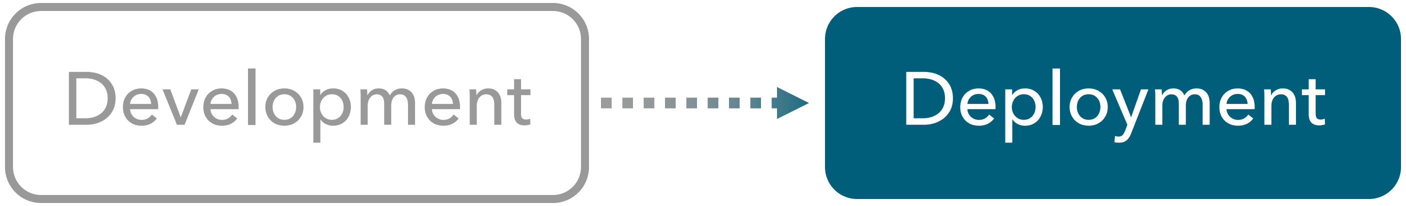
Building Governable ML Models with R

elderresearch.github.io/posit-conf-2025

Tom Shafer
Principal Data Scientist

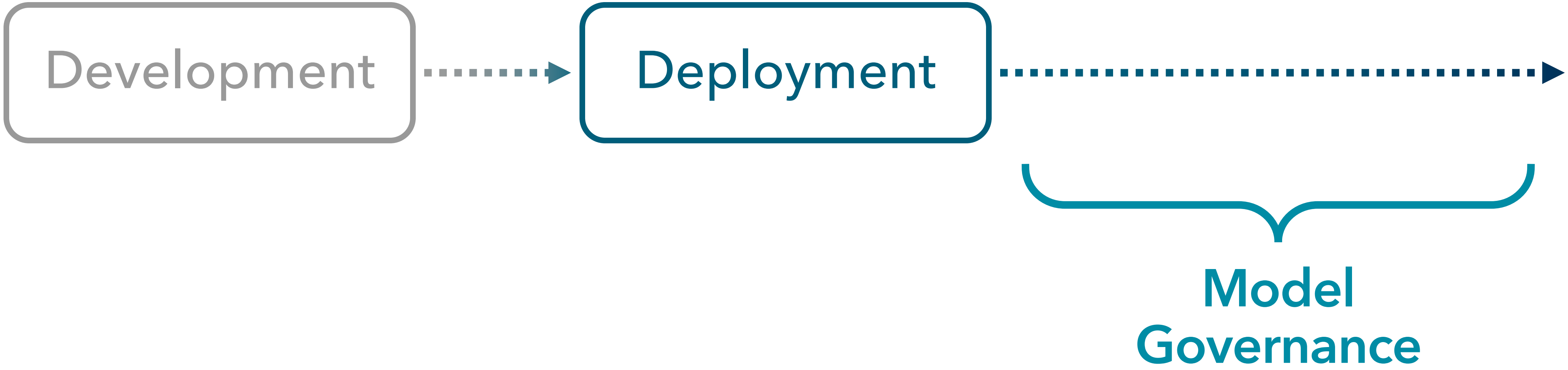
posit::conf
Sep 17, 2025

🎉 🎉 🎉



Development

Deployment

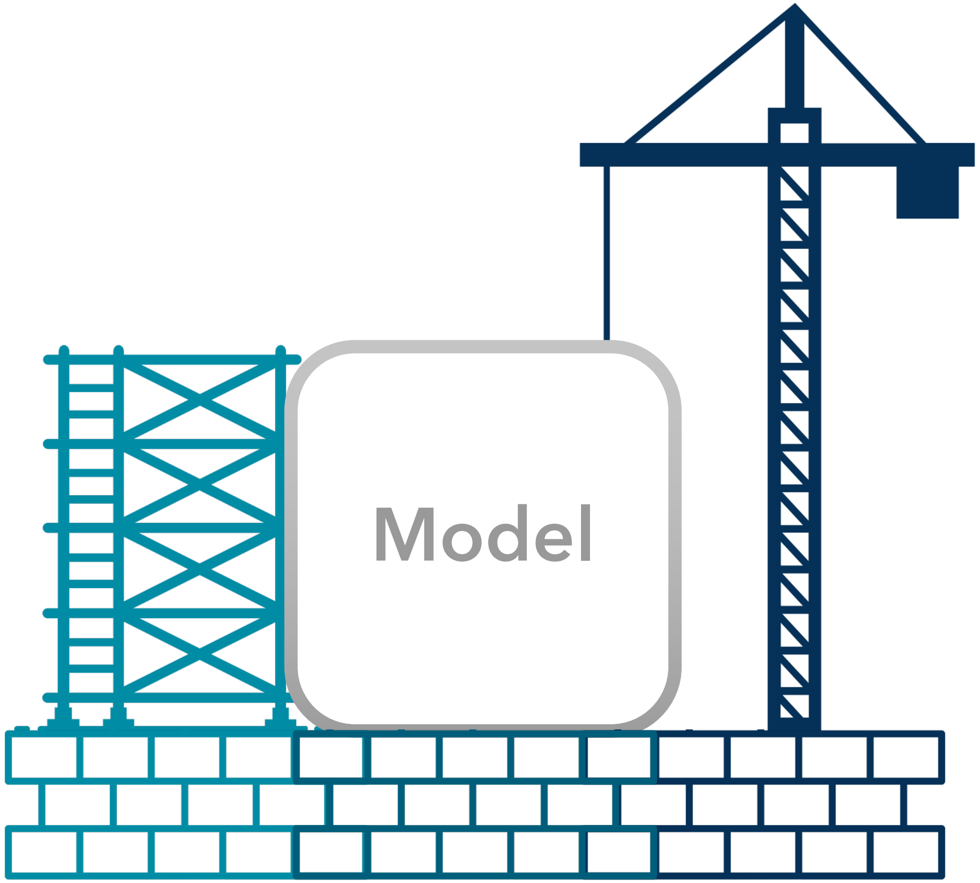


Development

Deployment

Model Governance

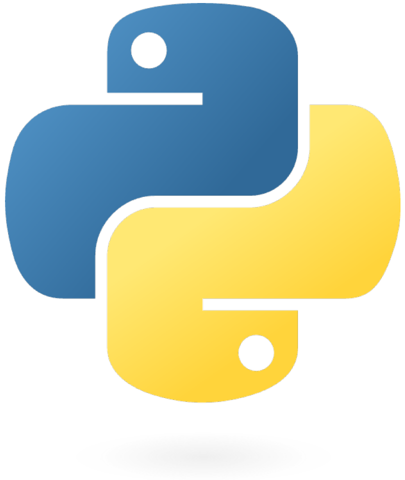
Model


Model

What can we do now
to make maintenance easier later?

What can we do now to make maintenance and development easier later?

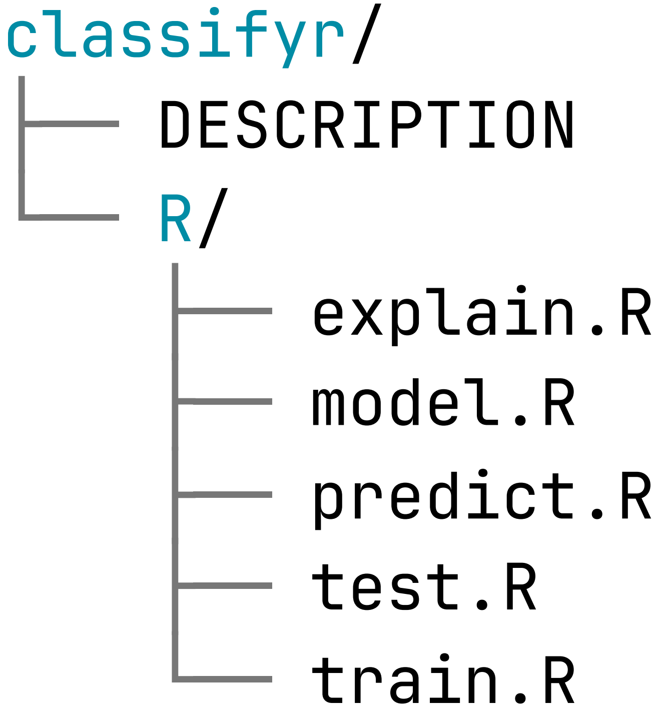
1. Packaging
2. Documentation
3. Testing
4. Legibility



Packaging

Packages provide structure

Literally – a file structure



```
classifyr/
├── DESCRIPTION
└── R/
    ├── explain.R
    ├── model.R
    ├── predict.R
    ├── test.R
    └── train.R
```

Packages provide structure

Structured dependencies

```
Imports:
    ranger,
    withr
Suggests:
    testthat (>= 3.0.0)
```

Packages provide structure

Structured versioning

```
Package: classifyr
Title: Example Classification Model
Version: 0.1.0
```

Packages support useful automations

Tie together good practices using devtools

```
# Generate package docs
devtools::document()
```

Packages support useful automations

Tie together good practices using devtools

```
# Run all package tests
devtools::test()
```

Packages support useful automations

Automate package checks when code changes

```r
# Check all functionality
devtools::check()
```

This builds on typical R patterns

```
library(classifyr)
train_classifyr(...)
```

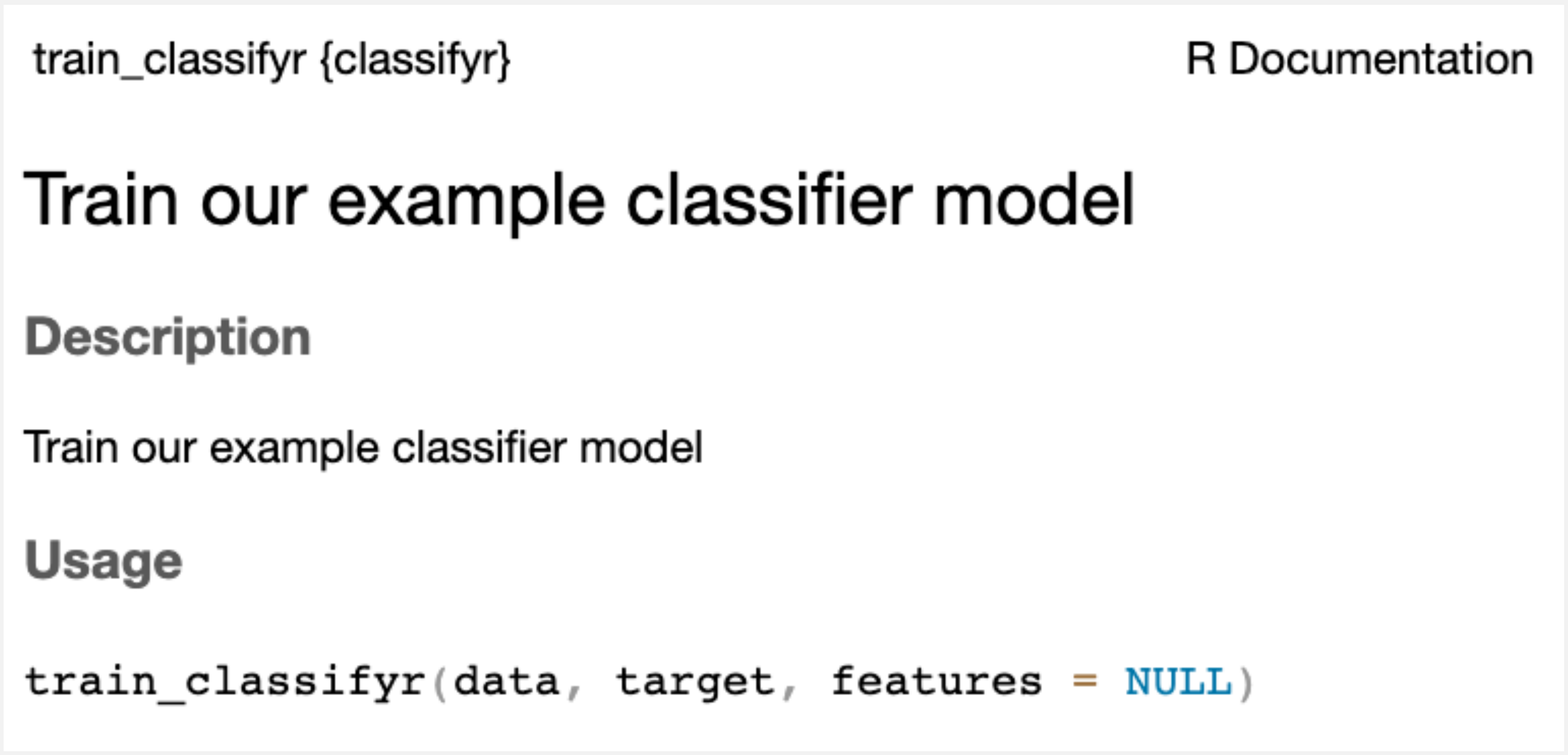
Documentation

Documentation matters…both now and later

- Documentation helps **future** you
- Documentation helps **current** you!

Documentation supports other nice tools

Help is integrated into IDEs



train_classifyr {classifyr}   R Documentation

Train our example classifier model

Description

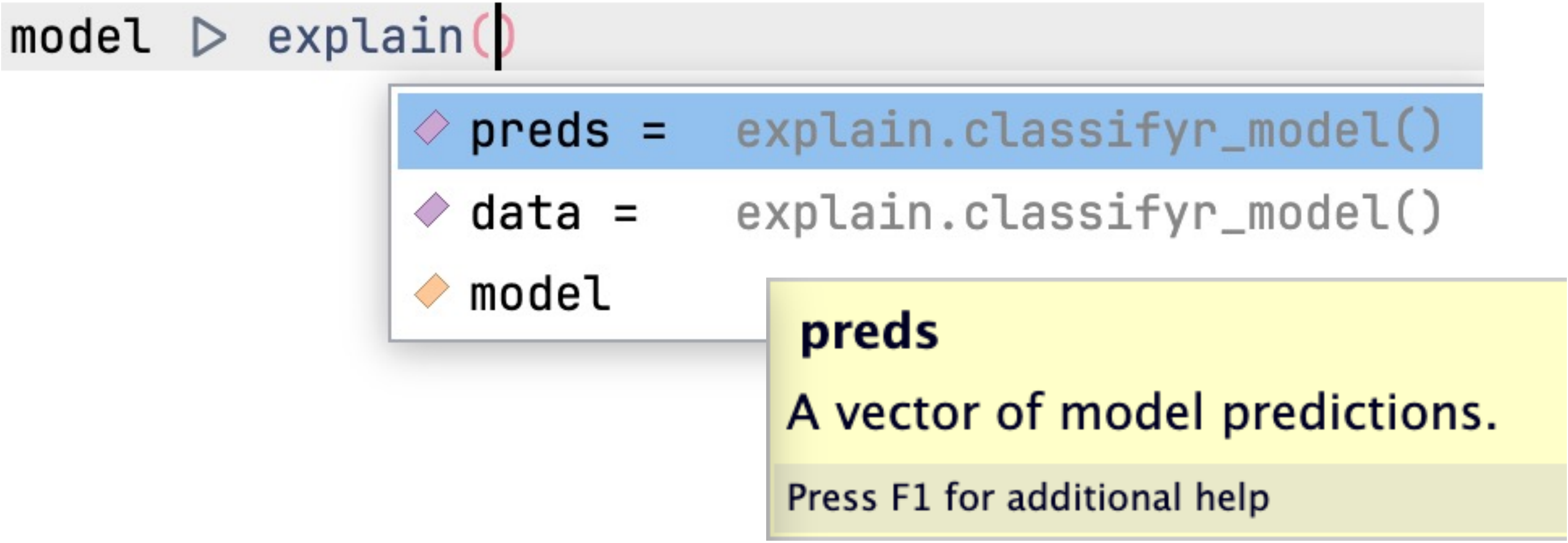
Train our example classifier model

Usage

```
train_classifyr(data, target, features = NULL)
```

Documentation supports other nice tools

IDEs and language models provide completions



```
model ▷ explain()
```

```
◇ preds =    explain.classifyr_model()
◇ data =     explain.classifyr_model()
◇ model
```

preds

A vector of model predictions.

Press F1 for additional help

Testing

Testing provides a safety harness

If we make a mistake, tests can catch us

- Does my code work **right now**?
- Does my code **still** work later?

  Have dependencies broken something?

  Did I break my code?

Start small and grow

Don’t boil the ocean

1. Identify and test key functionality
2. Add tests for bug fixes and new work

Get started with snapshots

testthat + snapshots: Store known-good outputs

```
model <- train_classifyr(
  data = iris,
  target = "Species",
  features = NULL
)
```

Input

Output

Snapshot

Get started with snapshots

testthat + snapshots: Store known-good outputs

```r
model <- train_classifyr(
  data = iris,
  target = "Species",
  features = NULL
)
```

Input

Output

Snapshot

Get started with snapshots

testthat + snapshots: Store known-good outputs

```
expect_snapshot(model)
```

Input

Output

Snapshot

Get started with snapshots

testthat + snapshots: Store known-good outputs

```
Code
  model
Message
  Training classifyr model
Output
  $preprocessor
  [1] 682402 ...
```

Input

Output

Snapshot

Get started with snapshots

testthat + snapshots: Store known-good outputs

```r
expect_snapshot(model)
```

Input

Output

Snapshot

Get started with snapshots

testthat + snapshots: Store known-good outputs

```
==> devtools::test()

ℹ Testing classifyr
✔ | F W  S  OK | Context
✔ |          1 | explain
✔ |          1 | train
```

Input

Output

Snapshot

Legibility

The real world is complicated

- Complicated data
- Complicated models
- Complicated prediction logic

One approach: Lists and custom functions

```
train_model <- function(...) {
    ...
    list(prep = model_a, model = model_b)
}
```

One approach: Lists and custom functions

```r
model1 <- custom_fn1(model[[1]])
model2 <- custom_fn2(model[[2]])
custom_predict(model1, model2, ...)
```

But we already have a nicer way: S3!

We’re already used to this

```
print(object)

predict(object, data)
```

Adopt S3 in two easy steps

First, attach a class attribute

```
structure(
    list(prep = model_a, model = model_b),
    class = c("classifyr_model", "list")
)
```

Adopt S3 in two easy steps

Second, write methods

```
predict.classifyr_model <- function(x, ...) {
  # Really complex things can go in here!
  ...
}
```

Adopt S3 in two easy steps

Second, write methods

```r
predict.classifyr_model <- function(x, ...) {
  ...
}

preds <- predict(model, data)
```

We can define entirely new methods, too

Create the method

```
explain <- function(obj, ...) {
  UseMethod("explain")
}


explain.classifyr_model <- function(obj, ...) {
  ...
}
```

What can we do now to make maintenance and development easier later?

1. Packaging
2. Documentation
3. Testing
4. Legibility

ELDER RESEARCH

DATA SCIENCE · AI · MACHINE LEARNING

Companion website with examples and more details