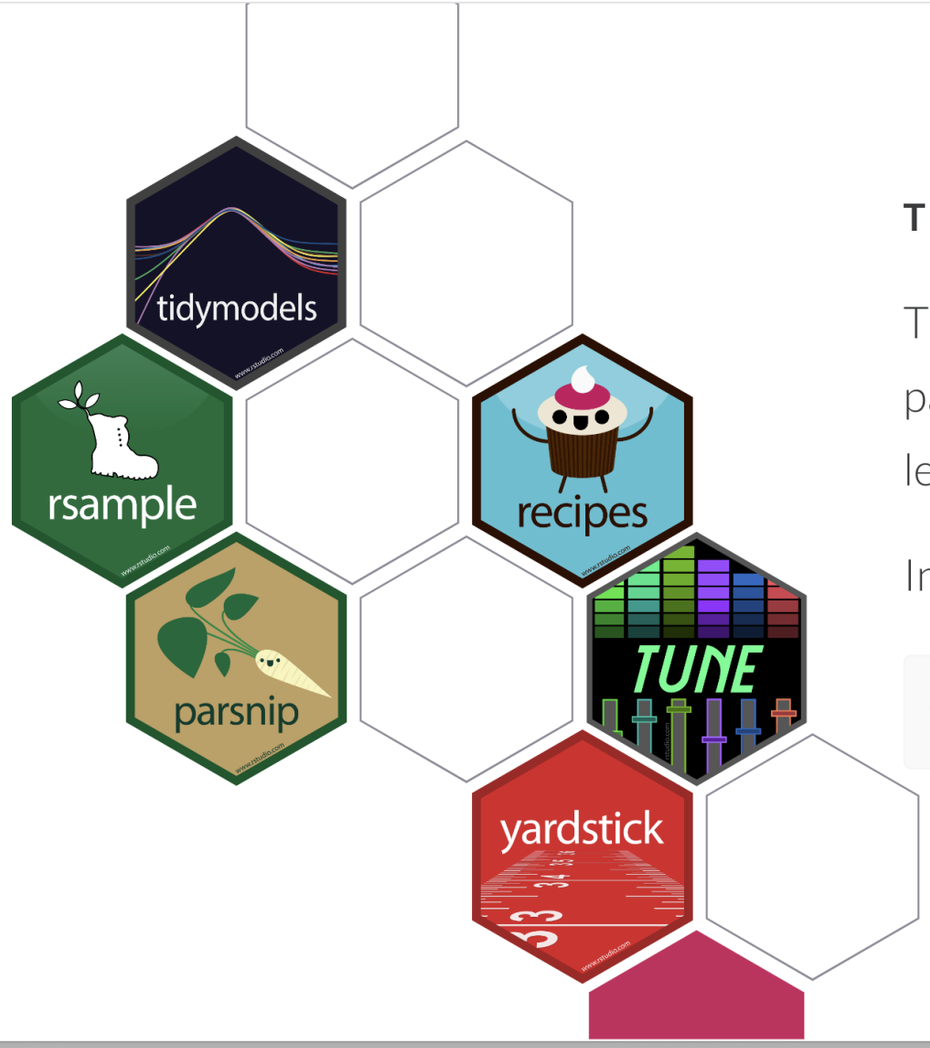
tidy models

Freddy Hernandez

tidymodels

tidymodels.org

tidymodels PACKAGES GET STARTED LEARN HELP CONTRIBUTE ABOUT FIND



TIDYMODELS

The tidymodels framework is a collection of packages for modeling and machine learning using tidyverse principles.

Install tidymodels with:

```
install.packages("tidymodels")
```

¿Qué es tidy models?


```
library(tidymodels)
#> ── Attaching packages ──────────────────────────────── tidymodels 1.1.1 ──
#> ✔ broom        1.0.5     ✔ rsample      1.2.0
#> ✔ dials        1.2.0     ✔ tibble       3.2.1
#> ✔ dplyr        1.1.2     ✔ tidyr        1.3.0
#> ✔ infer        1.0.5     ✔ tune         1.1.2
#> ✔ modeldata    1.2.0     ✔ workflows    1.1.3
#> ✔ parsnip      1.1.1     ✔ workflowsets 1.0.1
#> ✔ purrr        1.0.2     ✔ yardstick    1.2.0
#> ✔ recipes      1.0.8
#> ── Conflicts ─────────────────────────────────── tidymodels_conflicts() ──
#> ✖ purrr::discard() masks scales::discard()
#> ✖ dplyr::filter()  masks stats::filter()
#> ✖ dplyr::lag()     masks stats::lag()
#> ✖ recipes::step()  masks stats::step()
#> • Use tidymodels_prefer() to resolve common conflicts.
```

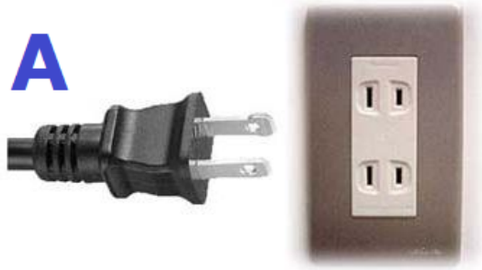
Max Kuhn – Posit

¿Qué debemos conocer para este tutorial?

- Haber usado el operador %>% de magrittr o el operador |> de R base.
- Estar familiarizado/a con las funciones de dplyr, tidyr y ggplot2.
- Tener conceptos estadísticos básicos.
- **No** necesitas tener familiaridad intermedia o experta con modelado o aprendizaje automático.

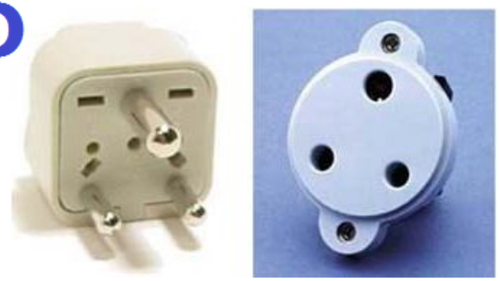
¿Para qué sirve tidy models?

A



C

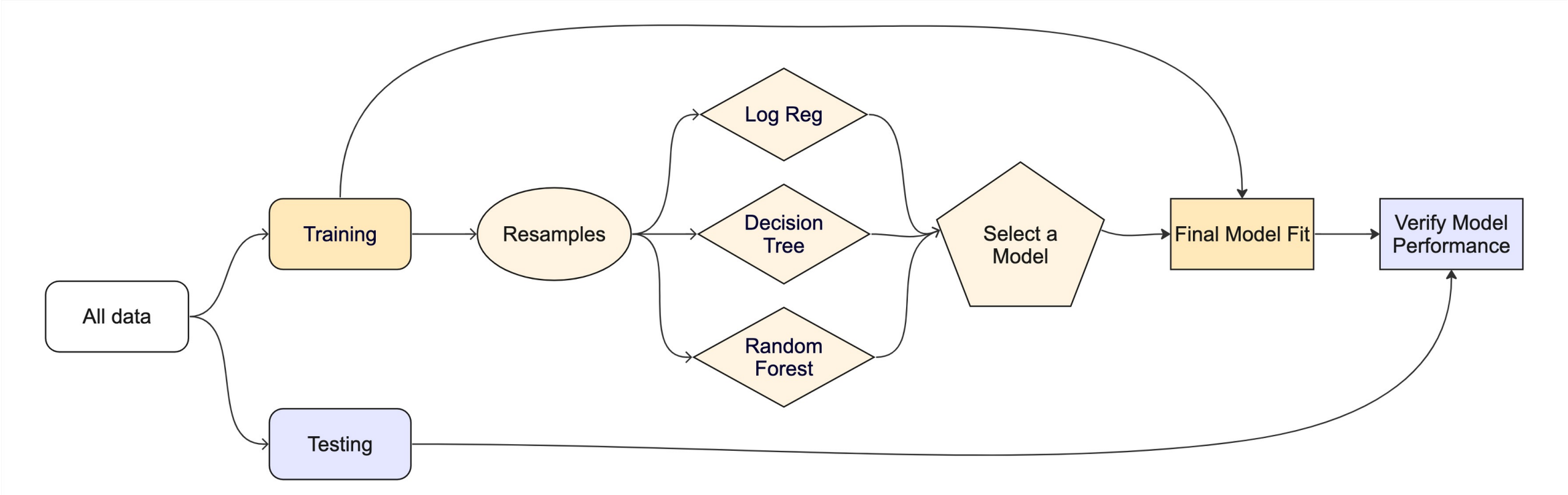
D



G

Adaptador

El panorama completo al modelar



Datos sobre viajes en taxi en Chicago

- La ciudad de Chicago publica datos anonimizados a nivel de viaje sobre los trayectos en taxi en la ciudad.
- Hemos extraído una muestra de 10000 viajes que tuvieron lugar a principios de 2022.
- Escribe `?taxi` para obtener más información sobre este conjunto de datos, incluyendo referencias.

Datos sobre viajes en taxi en Chicago

- `n = 10000`
- La variable de interés es `tip`, con niveles `"yes"` y `"no"`
- **nominales**: company, local, dow, month.
- **numericas**: distance, hour.

Los datos

```
library(tidymodels)

taxi
#> # A tibble: 10,000 × 7
#>    tip   distance company                          local dow   month  hour
#>    <fct>    <dbl> <fct>                            <fct> <fct> <fct> <int>
#>  1 yes      17.2  Chicago Independents             no    Thu   Feb      16
#>  2 yes       0.88 City Service                     yes   Thu   Mar       8
#>  3 yes      18.1  other                            no    Mon   Feb      18
#>  4 yes      20.7  Chicago Independents             no    Mon   Apr       8
#>  5 yes      12.2  Chicago Independents             no    Sun   Mar      21
#>  6 yes       0.94 Sun Taxi                         yes   Sat   Apr      23
#>  7 yes      17.5  Flash Cab                        no    Fri   Mar      12
#>  8 yes      17.7  other                            no    Sun   Jan       6
#>  9 yes       1.85 Taxicab Insurance Agency Llc     no    Fri   Apr      12
#> 10 yes       1.47 City Service                     no    Tue   Mar      14
#> # ℹ 9,990 more rows
```

Su turno

Obtenga los registros 320, 558, 784 y 5600 de `train`. El resultado debe ser este:

```
#> # A tibble: 4 × 7
#>   tip   distance company                       local dow   month  hour
#>   <fct>    <dbl> <fct>                         <fct> <fct> <fct> <int>
#> 1 yes       17.3 other                         no    Fri   Jan      16
#> 2 no        18.1 Flash Cab                     no    Mon   Jan      20
#> 3 yes       17.3 Flash Cab                     no    Tue   Mar       9
#> 4 no         0   Taxicab Insurance Agency Llc yes   Wed   Feb      11
```

02:00

Explorando la variable respuesta

Vamos a construir una tabla de frecuencia relativa para la variable respuesta.

```
taxi |>
  select(tip) |>
  table() |>
  prop.table()
#> tip
#>    yes     no
#> 0.9209 0.0791
```

División de los datos

En machine learning se deben dividir los datos en **al menos** dos grupos:

- **training set**: se usa para estimar los parámetros del modelo (entrenamiento).
- **test set**: se usa para medir el desempeño del modelo.

No use 🚫 el test set en el entrenamiento.

División de los datos

Funciones para dividir los datos

- `initial_split(data, prop = 3/4, strata = NULL, ...)`: sirve para crear la partición.
- `training()`: sirve para extraer training de la partición.
- `testing()`: sirve para crear extraer test de la partición.

Creando la partición

```
set.seed(123)
taxi_split <- initial_split(data=taxi)

taxi_split
#> <Training/Testing/Total>
#> <7500/2500/10000>
```

Obteniendo los dos conjuntos

```
taxi_train <- taxi_split |> training()

taxi_test  <- taxi_split |> testing()
```

El conjunto de train

```
taxi_train
#> # A tibble: 7,500 × 7
#>    tip   distance company                    local dow   month  hour
#>    <fct>    <dbl> <fct>                      <fct> <fct> <fct> <int>
#>  1 yes        0.7  Taxi Affiliation Services yes   Tue   Mar      18
#>  2 yes        0.99 Sun Taxi                  yes   Tue   Jan       8
#>  3 yes        1.78 other                     no    Sat   Mar      22
#>  4 yes        0    Taxi Affiliation Services yes   Wed   Apr      15
#>  5 yes        0    Taxi Affiliation Services no    Sun   Jan      21
#>  6 yes        2.3  other                     no    Sat   Apr      21
#>  7 yes        6.35 Sun Taxi                  no    Wed   Mar      16
#>  8 yes        2.79 other                     no    Sun   Feb      14
#>  9 yes       16.6  other                     no    Sun   Apr      18
#> 10 yes        0.02 Chicago Independents      yes   Sun   Apr      15
#> # ℹ 7,490 more rows
```

El conjunto de test

```
taxi_test
#> # A tibble: 2,500 × 7
#>    tip   distance company                          local dow   month  hour
#>    <fct>    <dbl> <fct>                            <fct> <fct> <fct> <int>
#>  1 yes       0.94 Sun Taxi                         yes   Sat   Apr      23
#>  2 yes       1    Taxi Affiliation Services        no    Mon   Feb      18
#>  3 yes       1.91 Flash Cab                        no    Wed   Apr      15
#>  4 yes       1.1  Chicago Independents             no    Sat   Mar      10
#>  5 yes      11.7  other                            no    Wed   Mar      18
#>  6 yes      17.8  City Service                     no    Mon   Mar       9
#>  7 yes       0.53 Taxicab Insurance Agency Llc     yes   Wed   Apr       8
#>  8 yes       1.14 City Service                     no    Wed   Mar      14
#>  9 yes       1.77 other                            no    Thu   Apr      15
#> 10 yes      18.6  Flash Cab                        no    Thu   Apr      12
#> # ℹ 2,490 more rows
```

Su turno

- Divida los datos en 80% para train y 20% para test.
- Coloque dentro de `initial_split()` el argumento `strata=tip` para que haga la división teniendo en cuenta la variable `tip`.
- Use la semilla 2023 dentro de `set.seed()`.
- Llame a los conjuntos `taxi_train` y `taxi_test`.
- Compare su `taxi_train` con el que aparece en la siguiente diapositiva.

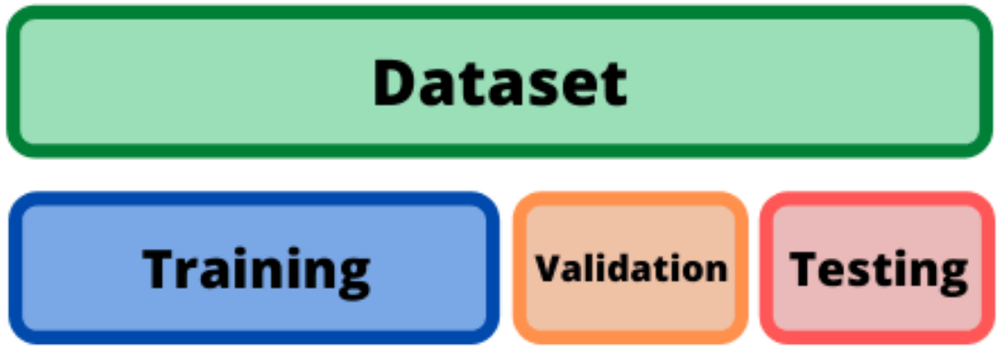
05:00

El nuevo `taxi_train`

```
#> # A tibble: 8,000 × 7
#>    tip   distance company                       local dow   month  hour
#>    <fct>    <dbl> <fct>                         <fct> <fct> <fct> <int>
#>  1 yes       17.2 Chicago Independents          no    Thu   Feb      16
#>  2 yes        0.88 City Service                 yes   Thu   Mar       8
#>  3 yes       18.1 other                         no    Mon   Feb      18
#>  4 yes       12.2 Chicago Independents          no    Sun   Mar      21
#>  5 yes        0.94 Sun Taxi                     yes   Sat   Apr      23
#>  6 yes       17.5 Flash Cab                     no    Fri   Mar      12
#>  7 yes       17.7 other                         no    Sun   Jan       6
#>  8 yes        1.85 Taxicab Insurance Agency Llc no    Fri   Apr      12
#>  9 yes        0.53 Sun Taxi                     no    Tue   Mar      18
#> 10 yes        6.65 Taxicab Insurance Agency Llc no    Sun   Apr      11
#> # ℹ 7,990 more rows
```

Funciones para dividir los datos

- `initial_validation_split(data, prop = c(0.6, 0.2), strata = NULL, ...)`: sirve para crear la partición.
- `training()`: sirve para extraer training de la partición.
- `testing()`: sirve para crear extraer test de la partición.
- `validation()`: sirve para crear extraer validation de la partición.

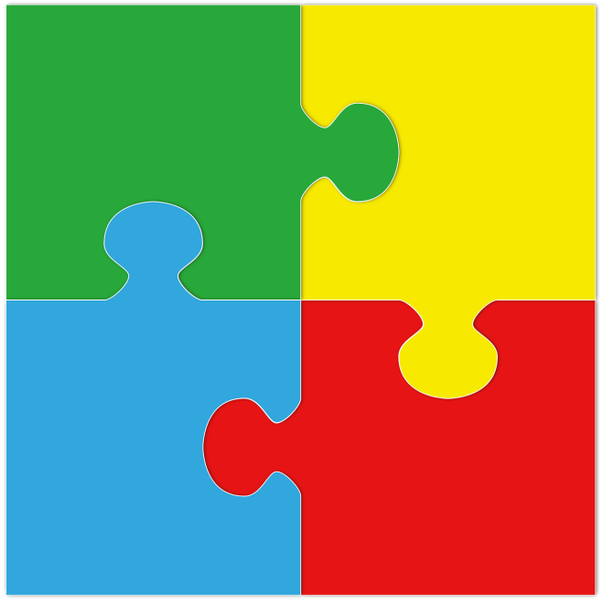


Modelos a usar en este tutorial

- Árbol de clasificación.
- Regresión logística.

Partes de un tidymodel

1. Especificación del modelo.
2. La receta.
3. El workflow.
4. El entrenamiento.



Elementos para especificar el modelo

- Elegir el modelo (model).
- Elegir el motor (engine).
- Definir el modo (regresión o clasificación).

Especificando el árbol

- Elegir el modelo (model).
- Elegir el motor (engine).
- Definir el modo (regresión o clasificación).

```
tree_spec <- decision_tree() |>
  set_args(cost_complexity = 0.0001) |>
  set_engine("rpart") |>
  set_mode("classification")

tree_spec
#> Decision Tree Model Specification (classification)
#>
#> Main Arguments:
#>   cost_complexity = 1e-04
#>
#> Computational engine: rpart
```

Especificando la regresión logística

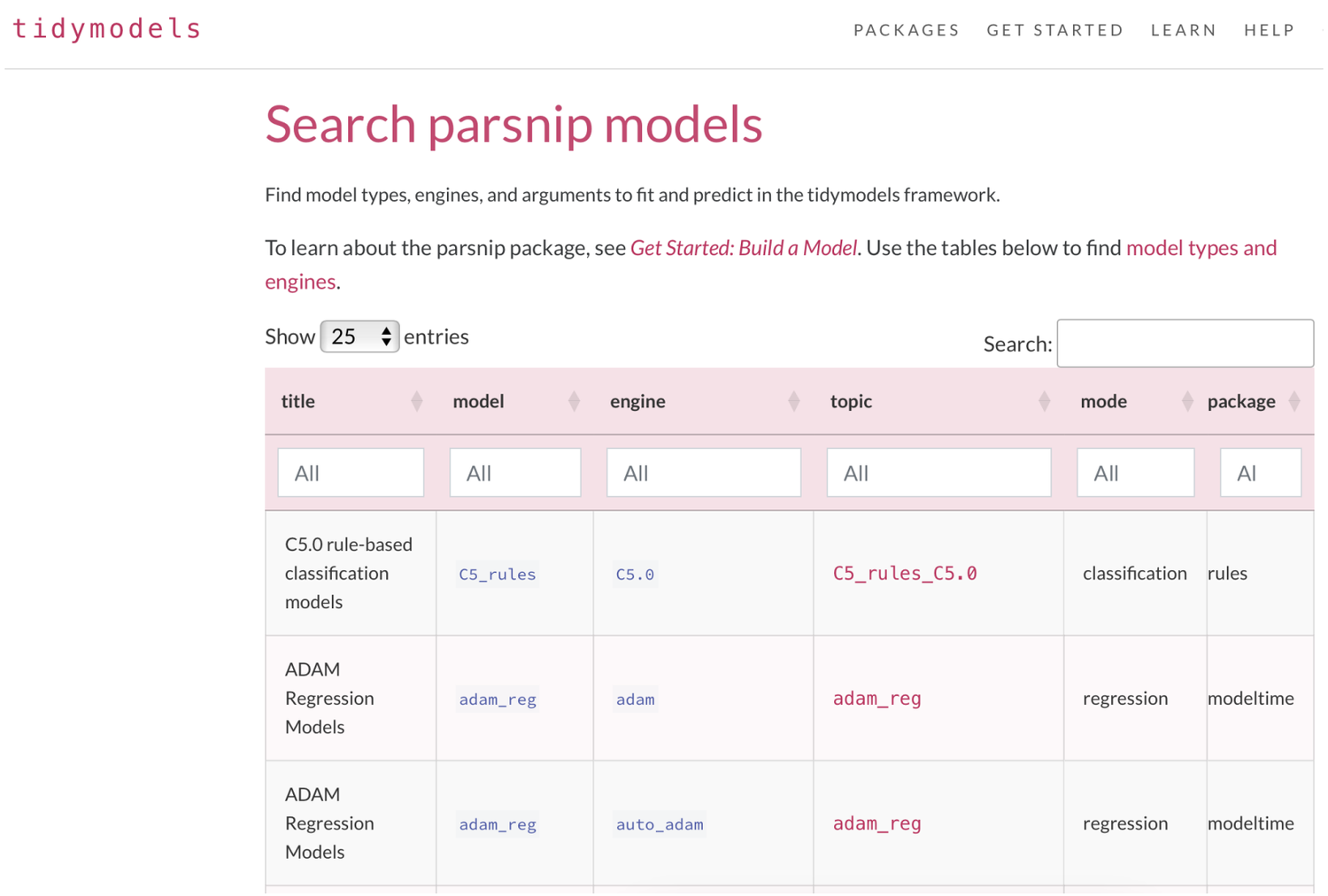
- Elegir el modelo (model).
- Elegir el motor (engine).
- Definir el modo (regresión o clasificación).

```
lr_spec <- logistic_reg() |>
  set_args(family = binomial(link="logit")) |>
  set_engine("glm") |>
  set_mode("classification")

lr_spec
#> Logistic Regression Model Specification (classification)
#>
#> Computational engine: glm
```

Lista de todos los modelos

https://www.tidymodels.org/find/parsnip/



Creando la receta

El paquete recipes tiene muchas funciones que nos permite transformar las variables. Consultar https://recipes.tidymodels.org.

```
general_rec <- recipe(tip ~ ., data=taxi)
```

Creando el workflow

Para crear el workflow se usa la función `workflow()`.

```
tree_wf <- workflow() |>
  add_model(tree_spec) |>
  add_recipe(general_rec)

lr_wf <- workflow() |>
  add_model(lr_spec) |>
  add_recipe(general_rec)
```

Entrenando el modelo

Para entrenar el modelo se usa la función `fit()`.

```
# The tree
tree_fit <- tree_wf |>
  fit(data = taxi_train)

# The LR
lr_fit <- lr_wf |>
  fit(data = taxi_train)
```

Haciendo predicciones

Vamos a predecir la variable `tip` usando `taxi_train` y el árbol de clasificación por medio de la función `predict(object, new_data, type=c("numeric", "class", "prob", ...))`.

```
predict(object=tree_fit, new_data=taxi_train)
#> # A tibble: 8,000 × 1
#>    .pred_class
#>    <fct>
#>  1 yes
#>  2 yes
#>  3 yes
#>  4 yes
#>  5 yes
#>  6 yes
#>  7 yes
#>  8 yes
#>  9 yes
#> 10 yes
#> # ℹ 7,990 more rows
```

Agregando las predicciones a los datos

Es posible agregar los resultados del modelo a los datos.

```
augment(x=tree_fit, new_data = taxi_train)
#> # A tibble: 8,000 × 10
#>    tip   distance company local dow   month  hour .pred_class .pred_yes .pred_no
#>    <fct>    <dbl> <fct>   <fct> <fct> <fct> <int> <fct>           <dbl>    <dbl>
#>  1 yes      17.2  Chicag… no    Thu   Feb      16 yes             0.967   0.0333
#>  2 yes       0.88 City S… yes   Thu   Mar       8 yes             0.935   0.0646
#>  3 yes      18.1  other   no    Mon   Feb      18 yes             0.967   0.0333
#>  4 yes      12.2  Chicag… no    Sun   Mar      21 yes             0.949   0.0507
#>  5 yes       0.94 Sun Ta… yes   Sat   Apr      23 yes             0.821   0.179
#>  6 yes      17.5  Flash … no    Fri   Mar      12 yes             0.967   0.0333
#>  7 yes      17.7  other   no    Sun   Jan       6 yes             0.967   0.0333
#>  8 yes       1.85 Taxica… no    Fri   Apr      12 yes             0.938   0.0616
#>  9 yes       0.53 Sun Ta… no    Tue   Mar      18 yes             0.938   0.0616
#> 10 yes       6.65 Taxica… no    Sun   Apr      11 yes             0.931   0.0694
#> # ℹ 7,990 more rows
```

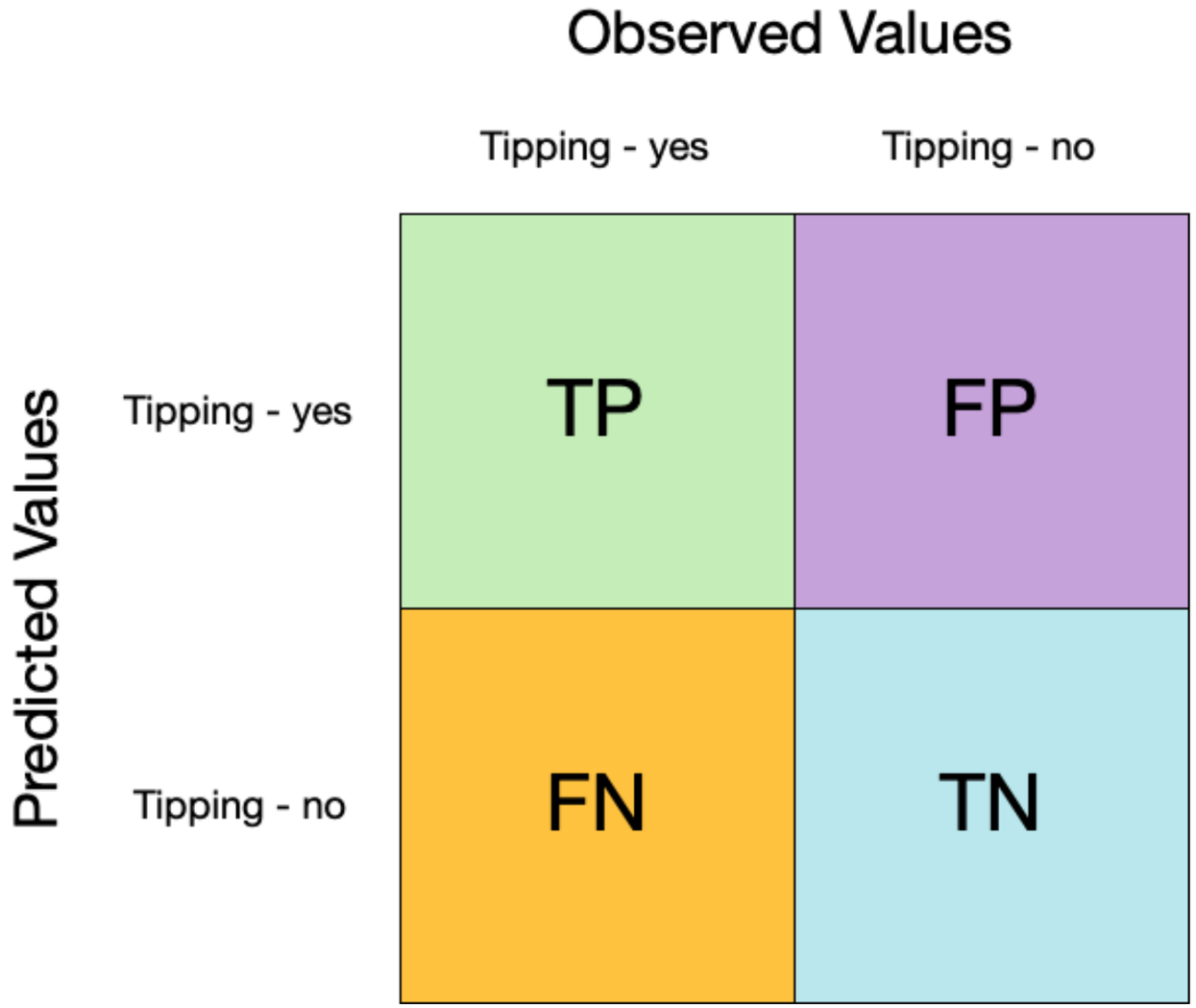
Su turno

Repita el procedimiento anterior pero usando `taxi_test`. El resultado debe ser el siguiente:

```
#> # A tibble: 2,000 × 10
#>    tip   distance company local dow   month  hour .pred_class .pred_yes .pred_no
#>    <fct>    <dbl> <fct>   <fct> <fct> <fct> <int> <fct>           <dbl>    <dbl>
#>  1 yes       20.7  Chicag… no    Mon   Apr       8 yes             0.967   0.0333
#>  2 yes        1.47 City S… no    Tue   Mar      14 yes             0.938   0.0616
#>  3 yes        1    Taxi A… no    Mon   Feb      18 yes             0.938   0.0616
#>  4 yes        1.91 Flash … no    Wed   Apr      15 yes             0.948   0.0519
#>  5 yes       17.2  City S… no    Mon   Apr       9 yes             0.967   0.0333
#>  6 yes       17.8  City S… no    Mon   Mar       9 yes             0.967   0.0333
#>  7 yes        0.53 Taxica… yes   Wed   Apr       8 yes             0.935   0.0646
#>  8 yes        1.77 other   no    Thu   Apr      15 yes             0.938   0.0616
#>  9 yes       18.6  Flash … no    Thu   Apr      12 yes             0.967   0.0333
#> 10 no         1.13 other   no    Sat   Feb      14 yes             0.938   0.0616
#> # ℹ 1,990 more rows
```

01:00

Matriz de confusión



Creando la matriz de confusión

La matriz de confusión se obtiene con `conf_mat()` aplicada luego de `augment()`.

```
augment(x=tree_fit, new_data = taxi_train) %>%
  conf_mat(truth = tip, estimate = .pred_class)
#>           Truth
#> Prediction  yes   no
#>        yes 7341  536
#>        no    43   80
```

El paquete yardstick tiene funciones útiles que se pueden consultar en https://yardstick.tidymodels.org.

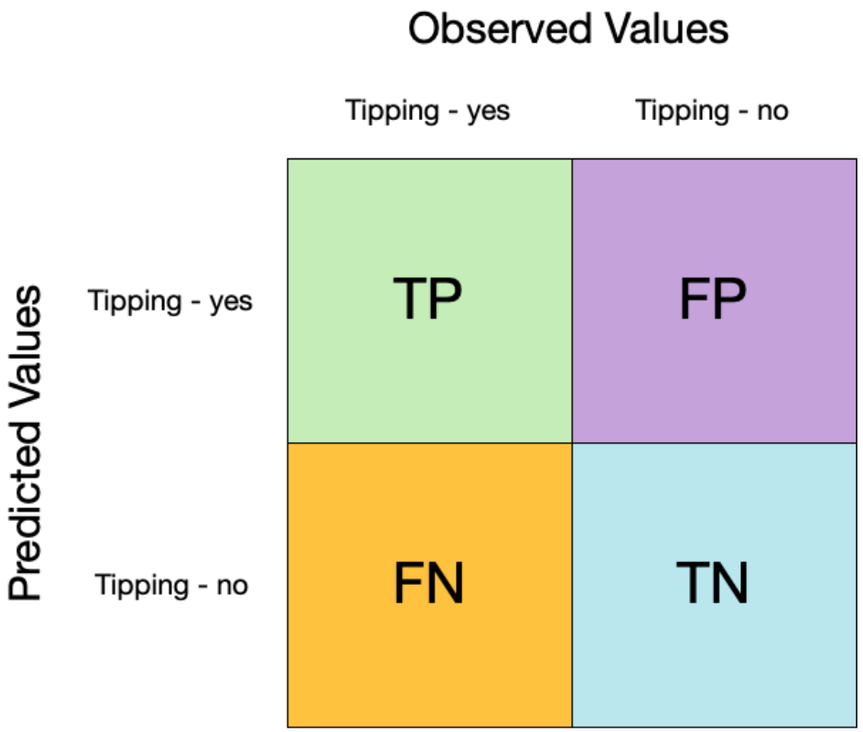
Su turno

Construya la matriz de confusión pero usando `taxi_test`. El resultado debe ser el siguiente:

```
#>            Truth
#> Prediction  yes   no
#>        yes 1807  165
#>        no    18   10
```

01:00

Precisión o accuracy



Accuracy

$$\frac{TP + TN}{TP + FP + FN + TN}$$

Calculando la precisión


```
augment(x=tree_fit, new_data = taxi_train) %>%
  accuracy(truth = tip, estimate = .pred_class)
#> # A tibble: 1 × 3
#>   .metric  .estimator .estimate
#>   <chr>    <chr>          <dbl>
#> 1 accuracy binary         0.928
```

Su turno

Calcule la precisión pero usando `taxi_test`. El resultado debe ser el siguiente:

```
#> # A tibble: 1 × 3
#>   .metric  .estimator .estimate
#>   <chr>    <chr>          <dbl>
#> 1 accuracy binary         0.908
```

01:00

Recursos útiles en la web

Les recomiendo consultar el siguiente material para conocer más sobre tidymodels.

- Curso completo: https://workshops.tidymodels.org
- La organización: https://www.tidymodels.org
- Libro: Tidy Modeling with R https://www.tmwr.org


GRACIAS!