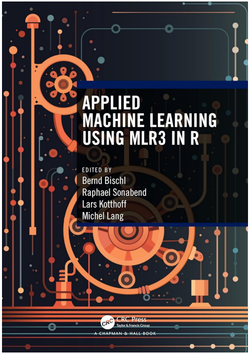
Machine Learning in R Using `mlr3`

- **Website:** `https://mlr-org.com/`
- **Github:** `https://github.com/mlr-org`
- **Book:** `https://mlr3book.mlr-org.com/`



April 16, 2024

Intro

R6

Data

Dictionaries

Learning Algorithms

Performance

Outro

SO YOU WANT TO DO ML IN R

- R gives you access to many machine learning methods

SO YOU WANT TO DO ML IN R

- R gives you access to many machine learning methods
- . . . but without a unified interface

SO YOU WANT TO DO ML IN R

- R gives you access to many machine learning methods
- . . . but without a unified interface
- things like performance evaluation are cumbersome

SO YOU WANT TO DO ML IN R

- R gives you access to many machine learning methods
- . . . but without a unified interface
- things like performance evaluation are cumbersome

Example:

```
# Specify what we want to model in a formula: target ~ features
svm_model = e1071::svm(Species ~ ., data = iris)
```

SO YOU WANT TO DO ML IN R

- R gives you access to many machine learning methods
- ...but without a unified interface
- things like performance evaluation are cumbersome

Example:

```
# Specify what we want to model in a formula: target ~ features
svm_model = e1071::svm(Species ~ ., data = iris)
```

vs.

```
# Pass the features as a matrix and the target as a vector
xgb_model = xgboost::xgboost(data = as.matrix(iris[1:4]),
  label = iris$Species, nrounds = 10)
```

SO YOU WANT TO DO ML IN R

```r
library("mlr3")
```

Ingredients:

- Data / Task
- Learning Algorithms
- Performance Evaluation
- Performance Comparison

Intro

R6

Data

Dictionaries

Learning Algorithms

Performance

Outro

R6 – ALL YOU NEED TO KNOW

`mlr3` uses the *R6* class system which facilitates OOP by allowing the creation of custom objects with methods and properties (it may look unusual if you see it the first time).

- *Objects* are created using `<Class>$new()`.

```
task = TaskClassif$new(id = "iris", backend = iris, target = "Species")
```

- Alternatively, the function `as_task_classif` can be used (or `as_task_regr` to construct a `TaskRegr` object for regression tasks). By default, the name of the object passed to `x` is used as `id`:

```
task = as_task_classif(x = iris, target = "Species")
```

R6 – ALL YOU NEED TO KNOW

`mlr3` uses the *R6* class system which facilitates OOP by allowing the creation of custom objects with methods and properties (it may look unusual if you see it the first time).

- *Objects* are created using `<Class>$new()`.

```
task = TaskClassif$new(id = "iris", backend = iris, target = "Species")
```

- Alternatively, the function `as_task_classif` can be used (or `as_task_regr` to construct a `TaskRegr` object for regression tasks). By default, the name of the object passed to `x` is used as `id`:

```
task = as_task_classif(x = iris, target = "Species")
```

- Objects have *fields* that contain information about the object.

```
task$nrow
#> [1] 150
```

R6 – ALL YOU NEED TO KNOW

`mlr3` uses the *R6* class system which facilitates OOP by allowing the creation of custom objects with methods and properties (it may look unusual if you see it the first time).

- *Objects* are created using `<Class>$new()`.

```
task = TaskClassif$new(id = "iris", backend = iris, target = "Species")
```

- Alternatively, the function `as_task_classif` can be used (or `as_task_regr` to construct a `TaskRegr` object for regression tasks). By default, the name of the object passed to `x` is used as `id`:

```
task = as_task_classif(x = iris, target = "Species")
```

- Objects have *fields* that contain information about the object.

```
task$nrow
#> [1] 150
```

- Objects have *methods* that are called like functions:

```
task$filter(rows = 1:10)
```

R6 – ALL YOU NEED TO KNOW

`mlr3` uses the *R6* class system which facilitates OOP by allowing the creation of custom objects with methods and properties (it may look unusual if you see it the first time).

- *Objects* are created using `<Class>$new()`.

```
task = TaskClassif$new(id = "iris", backend = iris, target = "Species")
```

- Alternatively, the function `as_task_classif` can be used (or `as_task_regr` to construct a `TaskRegr` object for regression tasks). By default, the name of the object passed to `x` is used as `id`:

```
task = as_task_classif(x = iris, target = "Species")
```

- Objects have *fields* that contain information about the object.

```
task$nrow
#> [1] 150
```

- Objects have *methods* that are called like functions:

```
task$filter(rows = 1:10)
```

- Methods may change ("mutate") the object (reference semantics)!

```
task$nrow
#> [1] 10
```

R6 AND ACTIVE BINDINGS

Some fields of R6-objects may be *“Active Bindings”*. Internally they are realized as functions that are called whenever the value is set or retrieved.

- Active bindings for read-only fields

```
task$nrow = 11
#> Error in assert_ro_binding(rhs):  Field/Binding is
read-only
```

R6 AND ACTIVE BINDINGS

Some fields of R6-objects may be *"Active Bindings"*. Internally they are realized as functions that are called whenever the value is set or retrieved.

- Active bindings for read-only fields

```
task$nrow = 11
#> Error in assert_ro_binding(rhs):  Field/Binding is
read-only
```

- Active bindings for argument checking

```
task$properties = NULL
#> Error in assert_set(rhs, .var.name = "properties"):
Assertion on 'properties' failed:  Must be of type
'character', not 'NULL'.
task$properties = c("property1", "property2")  # works
```

MLR3 PHILOSOPHY

- Overcome limitations of S3 with the help of **R6**
  - Truly object-oriented: data and methods live in the same object
  - Make use of inheritance
  - Reference semantics

MLR3 PHILOSOPHY

- Overcome limitations of S3 with the help of **R6**
  - Truly object-oriented: data and methods live in the same object
  - Make use of inheritance
  - Reference semantics
- Embrace **data.table**, both for arguments and internally
  - Fast operations for tabular data
  - List columns to arrange complex objects in tabular structure

MLR3 PHILOSOPHY

- Overcome limitations of S3 with the help of **R6**
  - Truly object-oriented: data and methods live in the same object
  - Make use of inheritance
  - Reference semantics
- Embrace **data.table**, both for arguments and internally
  - Fast operations for tabular data
  - List columns to arrange complex objects in tabular structure
- Be **light on dependencies**:
  - `R6`, `data.table`, `lgr`, `uuid`, `mlbench`, `digest`
  - Plus some of our own packages (`backports`, `checkmate`, ...)

Intro

R6

Data

Dictionaries

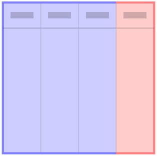
Learning Algorithms

Performance

Outro

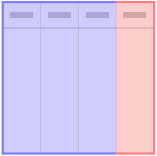
DATA

- Tabular data



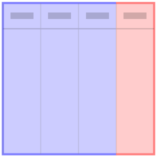
DATA

- Tabular data
- Features



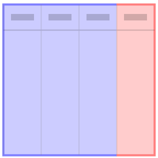
DATA

- Tabular data
- Features
- Target / outcome to predict



DATA

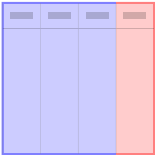
- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression



DATA

- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression

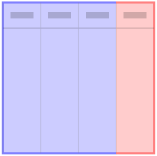
  ⇒ target determines the machine learning “Task”



DATA

- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression

  ⇒ target determines the machine learning "Task"

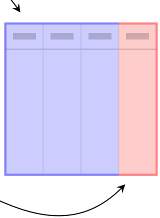


```
print(iris)  # included in R

#>   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
#> ...
```

DATA

- Tabular data
- Features
- Target / outcome to predict
    - discrete for classification
    - continuous for regression

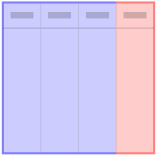
  ⇒ target determines the machine learning "Task"



```r
print(iris)  # included 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
#> ...
```

DATA

- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression
  - ⇒ target determines the machine learning "Task"

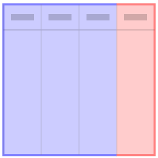


```r
print(iris)  # included in R

#>   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
#> ...
```

```r
task = TaskClassif$new("iris", iris, "Species")
task = as_task_classif(x = iris, target = "Species", id = "iris")
```

DATA

- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression
  - ⇒ target determines the machine learning "Task"



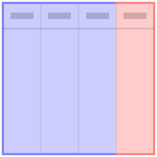
```
print(iris)  # included in R

#>   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
#> ...
```

Task ID

```
task = TaskClassif$new("iris", iris, "Species")
task = as_task_classif(x = iris, target = "Species", id = "iris")
```

DATA

- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression
  - ⇒ target determines the machine learning "Task"



```
print(iris)  # included 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
#> ...
```

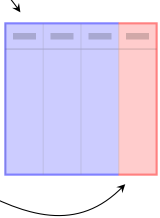
Task ID data

```
task = TaskClassif$new("iris", iris, "Species")
task = as_task_classif(x = iris, target = "Species", id = "iris")
```

DATA

- Tabular data
- Features
- Target / outcome to predict
  - discrete for classification
  - continuous for regression

  ⇒ target determines the machine learning "Task"



```r
print(iris)  # included 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
#> ...
```

Task ID   data   target name

```r
task = TaskClassif$new("iris", iris, "Species")
task = as_task_classif(x = iris, target = "Species", id = "iris")
```

DATA

```r
task = as_task_classif(x = iris, target = "Species")
```

```r
print(task)

# <TaskClassif:iris> (150 x 5)
# * Target: Species
# * Properties: multiclass
# * Features (4):
#   - dbl (4): Petal.Length, Petal.Width, Sepal.Length,
#     Sepal.Width
```

```r
task$ncol
task$nrow
task$feature_names
task$target_names
```

```r
task$head(n = )
task$truth(row_ids = )
task$data(rows = ,
          cols = )
```

```r
task$select(cols = )
task$filter(rows = )
task$cbind(data = )
task$rbind(data = )
```

Intro

R6

Data

Dictionaries

Learning Algorithms

Performance

Outro

DICTIONARIES

- `mlr3` uses `R6` classes to create dictionaries that store key-value pairs, i.e., associate keys (unique identifiers) with values (R6 objects).
- Dictionaries are easily extendable and allow adding and removing key-value pairs, e.g., add-on packages such as `mlr3learners` populate dictionaries with additional key-value pairs.

DICTIONARIES

- `mlr3` uses `R6` classes to create dictionaries that store key-value pairs, i.e., associate keys (unique identifiers) with values (R6 objects).
- Dictionaries are easily extendable and allow adding and removing key-value pairs, e.g., add-on packages such as `mlr3learners` populate dictionaries with additional key-value pairs.
- `mlr3` offers *Short Form* functions to get objects from a `Dictionary`:

Object	Dictionary	Short Form
`Task`	`mlr_tasks`	`tsk()`
`Learner`	`mlr_learners`	`lrn()`
`Measure`	`mlr_measures`	`msr()`
`Resampling`	`mlr_resamplings`	`rsmp()`

DICTIONARIES

```
# list items
tsk()

#> <DictionaryTask> with 11 stored values
#> Keys: boston_housing, breast_cancer, german_credit, iris,
#>   mtcars, penguins, pima, sonar, spam, wine, zoo

# retrieve object
tsk("iris")

#> <TaskClassif:iris> (150 x 5): Iris Flowers
#> * Target: Species
#> * Properties: multiclass
#> * Features (4):
#>   - dbl (4): Petal.Length, Petal.Width, Sepal.Length,
#>     Sepal.Width
```

SHORT FORMS AND DICTIONARIES

`as.data.table(<DICTIONARY>)` creates a `data.table` with metadata about objects in dictionaries:

```
as.data.table(mlr_learners)[1:5, c("key", "packages", "predict_types")]

# Key: <key>
#                    key    packages predict_types
#                 <char>      <list>        <list>
# 1:       classif.debug        mlr3 response,prob
# 2: classif.featureless        mlr3 response,prob
# 3:       classif.rpart mlr3,rpart response,prob
# 4:          regr.debug        mlr3   response,se
# 5:    regr.featureless mlr3,stats   response,se

library(mlr3learners) # mlr_learners dictionary gets populated
as.data.table(mlr_learners)[1:5, c("key", "packages", "predict_types")]

# Key: <key>
#                    key                 packages predict_types
#                 <char>                   <list>        <list>
# 1:   classif.cv_glmnet mlr3,mlr3learners,glmnet response,prob
# 2:       classif.debug                     mlr3 response,prob
# 3: classif.featureless                     mlr3 response,prob
# 4:      classif.glmnet mlr3,mlr3learners,glmnet response,prob
# 5:        classif.kknn   mlr3,mlr3learners,kknn response,prob
```

Intro

R6

Data

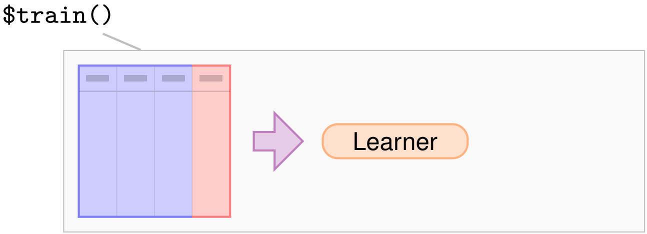
Dictionaries

Learning Algorithms

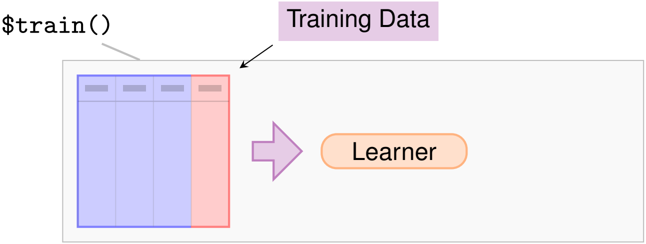
Performance

Outro

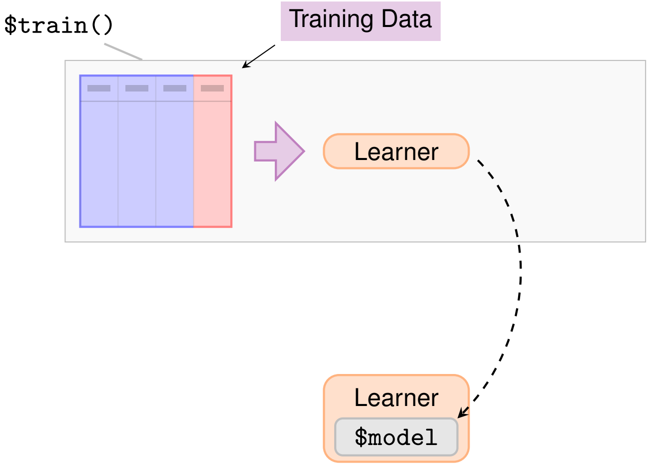
LEARNING ALGORITHMS



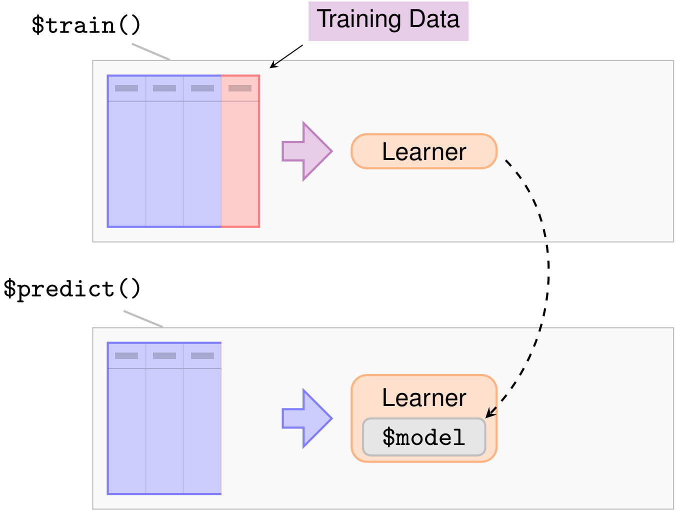
LEARNING ALGORITHMS



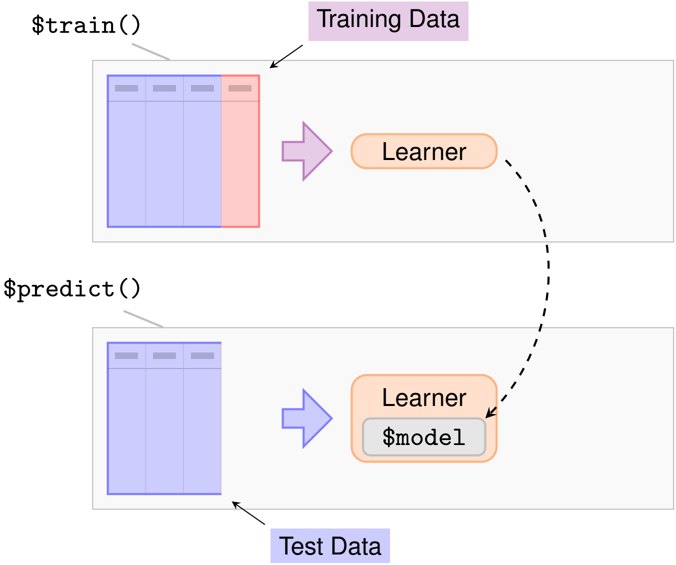
LEARNING ALGORITHMS



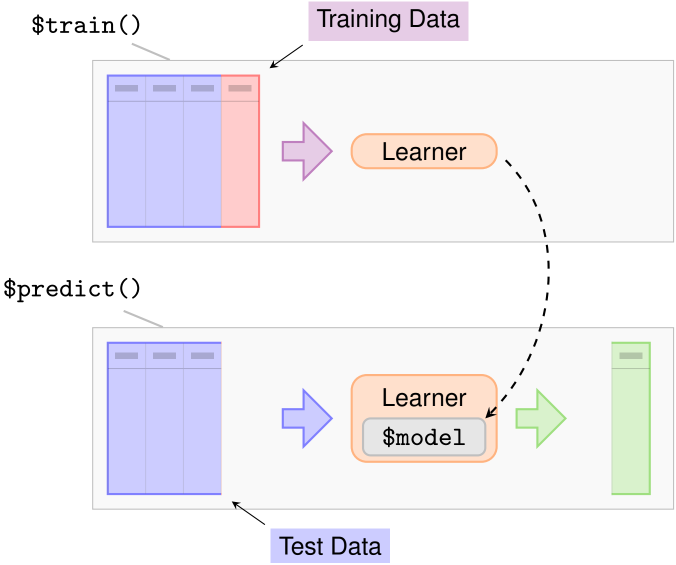
LEARNING ALGORITHMS



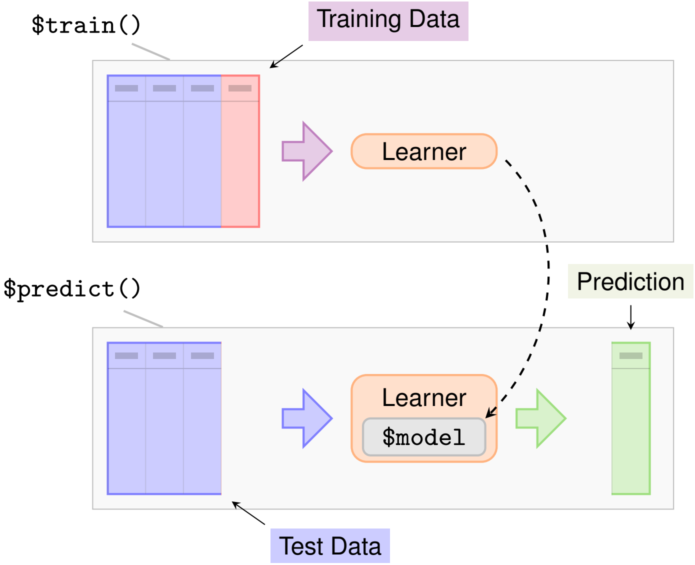
LEARNING ALGORITHMS



LEARNING ALGORITHMS



LEARNING ALGORITHMS



LEARNING ALGORITHMS

- Get a `Learner` provided by `mlr`

```
learner = lrn("classif.rpart")
```

LEARNING ALGORITHMS

- Get a `Learner` provided by `mlr`

```
learner = lrn("classif.rpart")
```

- Train the `Learner`

```
learner$train(task)
```

LEARNING ALGORITHMS

- Get a `Learner` provided by `mlr`

```
learner = lrn("classif.rpart")
```

- Train the `Learner`

```
learner$train(task)
```

- The `$model` is the `rpart` model: a decision tree

```
print(learner$model)

#> n= 150
#>
#> node), split, n, loss, yval, (yprob)
#>       * denotes terminal node
#>
#> 1) root 150 100 setosa (0.333 0.333 0.333)
#>   2) Petal.Length< 2.4 50   0 setosa (1.000 0.000 0.000) *
#>   3) Petal.Length>=2.4 100  50 versicolor (0.000 0.500 0.500)
#>     6) Petal.Width< 1.8 54   5 versicolor (0.000 0.907 0.093) *
#>     7) Petal.Width>=1.8 46   1 virginica (0.000 0.022 0.978) *
```

HYPERPARAMETERS

- Learners have *hyperparameters* that control their behavior

```
as.data.table(learner$param_set)[, 1:6]

#>                 id    class lower upper      levels nlevels
#>             <char>   <char> <num> <num>      <list>   <num>
#>  1:             cp ParamDbl     0     1                 Inf
#>  2:     keep_model ParamLgl    NA    NA  TRUE,FALSE       2
#>  3:     maxcompete ParamInt     0   Inf                 Inf
#>  4:       maxdepth ParamInt     1    30                  30
#>  5:   maxsurrogate ParamInt     0   Inf                 Inf
#>  6:      minbucket ParamInt     1   Inf                 Inf
#>  7:       minsplit ParamInt     1   Inf                 Inf
#>  8: surrogatestyle ParamInt     0     1                   2
#>  9:   usesurrogate ParamInt     0     2                   3
#> 10:           xval ParamInt     0   Inf                 Inf
```

HYPERPARAMETERS

- Changing hyperparameters after the creation of a `Learner` object:

```r
learner$param_set$values = list(maxdepth = 1, xval = 0)
```

- Changing hyperparameters when the `Learner` object is created:

```r
learner = lrn("classif.rpart", maxdepth = 1, xval = 0)
```

HYPERPARAMETERS

- Changing hyperparameters after the creation of a `Learner` object:

```
learner$param_set$values = list(maxdepth = 1, xval = 0)
```

- Changing hyperparameters when the `Learner` object is created:

```
learner = lrn("classif.rpart", maxdepth = 1, xval = 0)
```

- The `Learner` behavior changes and gives a smaller decision tree:

```
learner$train(task)
print(learner$model)
#> n= 150
#>
#> node), split, n, loss, yval, (yprob)
#>       * denotes terminal node
#>
#> 1) root 150 100 setosa (0.33 0.33 0.33)
#>   2) Petal.Length< 2.4 50   0 setosa (1.00 0.00 0.00) *
#>   3) Petal.Length>=2.4 100  50 versicolor (0.00 0.50 0.50) *
```

PREDICTION

- Let's make a prediction for some new data, e.g.:

```
new_data

#   Sepal.Length Sepal.Width Petal.Length Petal.Width
# 1            4           3            2           1
# 2            2           2            3           2
```

PREDICTION

- Let's make a prediction for some new data, e.g.:

```
new_data

#   Sepal.Length Sepal.Width Petal.Length Petal.Width
# 1            4           3            2           1
# 2            2           2            3           2
```

- To do so, we call the `$predict_newdata()` method using the new data:

```
prediction = learner$predict_newdata(new_data)
```

PREDICTION

- Let's make a prediction for some new data, e.g.:

```
new_data

#   Sepal.Length Sepal.Width Petal.Length Petal.Width
# 1            4           3            2           1
# 2            2           2            3           2
```

- To do so, we call the `$predict_newdata()` method using the new data:

```
prediction = learner$predict_newdata(new_data)
```

- We get a `Prediction` object:

```
prediction

#> <PredictionClassif> for 2 observations:
#>  row_ids truth   response
#>        1  <NA>     setosa
#>        2  <NA> versicolor
```

PREDICTION

- Let's make a prediction for some new data, e.g.:

```
new_data

#   Sepal.Length Sepal.Width Petal.Length Petal.Width
# 1            4           3            2           1
# 2            2           2            3           2
```

- To do so, we call the `$predict_newdata()` method using the new data:

```
prediction = learner$predict_newdata(new_data)
```

- We get a `Prediction` object:

```
prediction

#> <PredictionClassif> for 2 observations:
#>  row_ids truth   response
#>        1  <NA>     setosa
#>        2  <NA> versicolor
```

PREDICTION

- Let's make a prediction for some new data, e.g.:

```
new_data

#   Sepal.Length Sepal.Width Petal.Length Petal.Width
# 1            4           3            2           1
# 2            2           2            3           2
```

- To do so, we call the `$predict_newdata()` method using the new data:

```
prediction = learner$predict_newdata(new_data)
```

- We get a `Prediction` object:

```
prediction

#> <PredictionClassif> for 2 observations:
#>  row_ids truth   response
#>        1  <NA>     setosa
#>        2  <NA> versicolor
```

PREDICTION

- We can make the `Learner` predict *probabilities* when we set `predict_type`:

```
learner$predict_type = "prob"
learner$predict_newdata(new_data)
```

```
# <PredictionClassif> for 2 observations:
#  row_ids truth  response prob.setosa prob.versicolor
#        1  <NA>    setosa           1             0.0
#        2  <NA> virginica           0             0.5
#  prob.virginica
#             0.0
#             0.5
```

PREDICTION

What exactly is a `Prediction` object?

- Contains predictions and offers useful access fields / methods

PREDICTION

What exactly is a `Prediction` object?

- Contains predictions and offers useful access fields / methods

⇒ Use `as.data.table()` to extract data

```
as.data.table(prediction)
#>    row_ids  truth   response
#>      <int> <fctr>     <fctr>
#> 1:       1   <NA>     setosa
#> 2:       2   <NA> versicolor
```

PREDICTION

What exactly is a `Prediction` object?

- Contains predictions and offers useful access fields / methods

⇒ Use `as.data.table()` to extract data

```
as.data.table(prediction)
```

```
#>    row_ids  truth   response
#>      <int> <fctr>     <fctr>
#> 1:       1   <NA>     setosa
#> 2:       2   <NA> versicolor
```

⇒ Active bindings and functions that give further information: `$response`, `$truth`, ...

```
prediction$response
```

```
#> [1] setosa     versicolor
#> Levels: setosa versicolor virginica
```

Intro

R6

Data

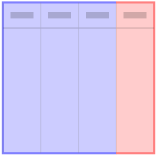
Dictionaries

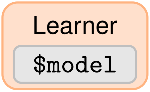
Learning Algorithms

Performance

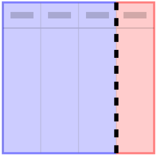
Outro

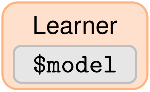
PERFORMANCE EVALUATION



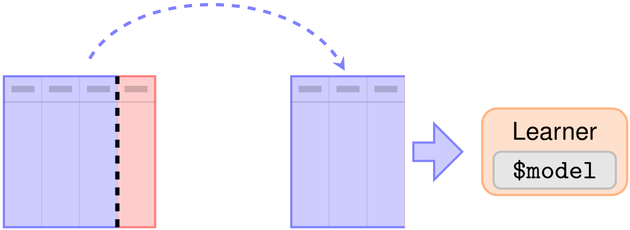


PERFORMANCE EVALUATION

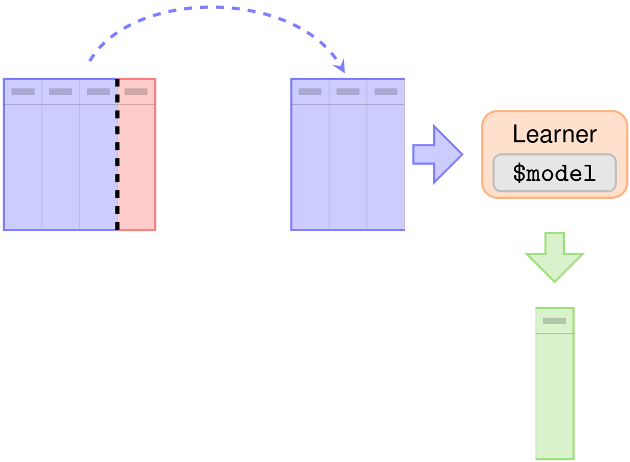




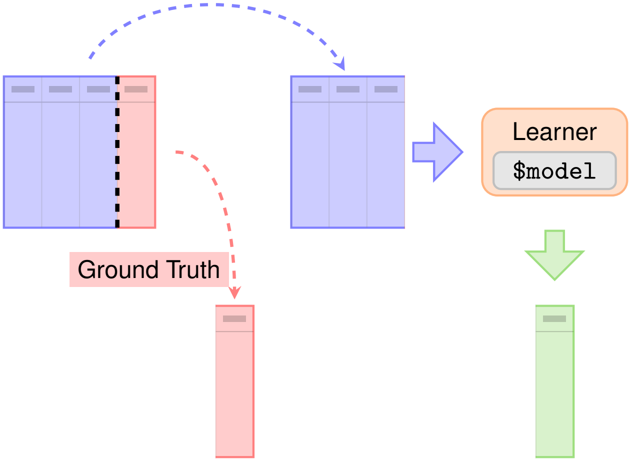
PERFORMANCE EVALUATION



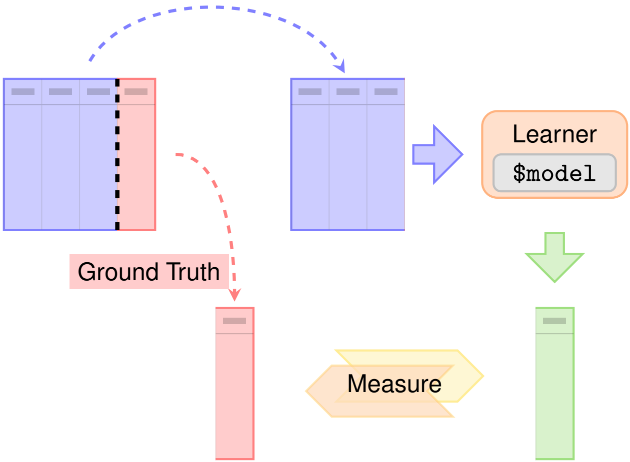
PERFORMANCE EVALUATION



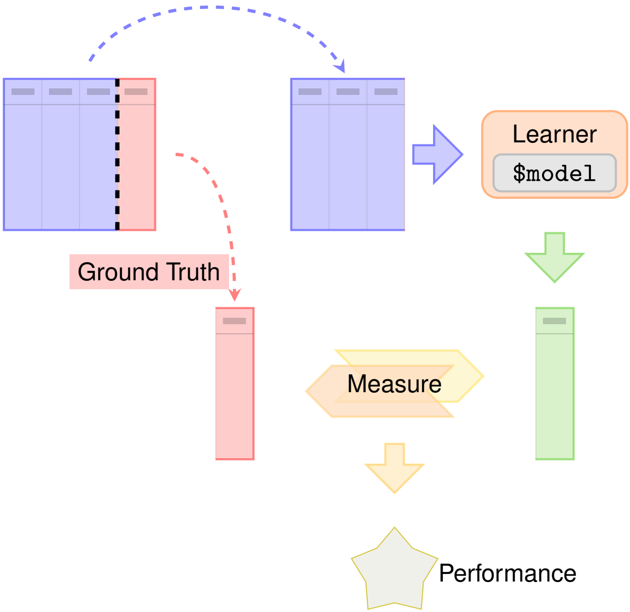
PERFORMANCE EVALUATION



PERFORMANCE EVALUATION



PERFORMANCE EVALUATION



PERFORMANCE EVALUATION

- Prediction ‘Task‘ with known data

```
known_truth_task$data()

#    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#     <fctr>        <num>       <num>        <num>       <num>
# 1:  setosa            2           1            4           3
# 2:  setosa            3           2            2           2
```

PERFORMANCE EVALUATION

- Prediction ‘Task‘ with known data

```
known_truth_task$data()
```

```
#    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#     <fctr>        <num>       <num>        <num>       <num>
# 1:  setosa            2           1            4           3
# 2:  setosa            3           2            2           2
```

- Predict again

```
pred = learner$predict(known_truth_task)
pred
```

```
#> <PredictionClassif> for 2 observations:
#>  row_ids  truth  response
#>        1 setosa    setosa
#>        2 setosa virginica
```

PERFORMANCE EVALUATION

- Prediction ‘Task‘ with known data

```
known_truth_task$data()

#    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#     <fctr>        <num>       <num>        <num>       <num>
# 1:  setosa            2           1            4           3
# 2:  setosa            3           2            2           2
```

- Predict again

```
pred = learner$predict(known_truth_task)
pred

#> <PredictionClassif> for 2 observations:
#>  row_ids  truth  response
#>        1 setosa    setosa
#>        2 setosa virginica
```

- Score the prediction

```
pred$score(msr("classif.ce"))

#> classif.ce
#>        0.5
```

PERFORMANCE EVALUATION

- Prediction ‘Task‘ with known data

```
known_truth_task$data()
```

```
#     Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#      <fctr>        <num>       <num>        <num>       <num>
# 1:   setosa            2           1            4           3
# 2:   setosa            3           2            2           2
```

- Predict again

```
pred = learner$predict(known_truth_task)
pred
```

```
#> <PredictionClassif> for 2 observations:
#>  row_ids  truth  response
#>        1 setosa    setosa
#>        2 setosa virginica
```

- Score the prediction

```
pred$score(msr("classif.ce"))
```

```
#> classif.ce
#>        0.5
```

Intro

R6

Data

Dictionaries

Learning Algorithms

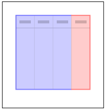
Performance

Outro

OVERVIEW

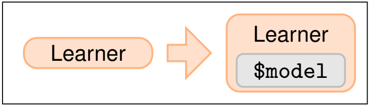
Ingredients:

Data



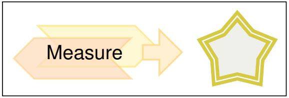
TaskClassif,
TaskRegr,
tsk()

Learning Algorithms



lrn() ⇒ Learner,
↪ Learner$train(),
↪ Learner$predict() ⇒ Prediction

Measure Performance



Prediction$score(),
msr() ⇒ Measure