R `mlr3` w/ ChatGPT

feat. `mlr3`

Yeonhoon Jang

Contents

- Introduction
- Syntax
- Basic modeling
- Resampling
- Benchmarking

Introduction

Who am I?

- Graduate School of Public Health, SNU (2019.03 ~ 2021.02)
- Seoul National University Bundang Hospital (2021.06 ~)
- Data (NHIS, MIMIC-IV, Registry data, KNHNAES …)
- Comento mentor (R for healthcare) (2022.07 ~)

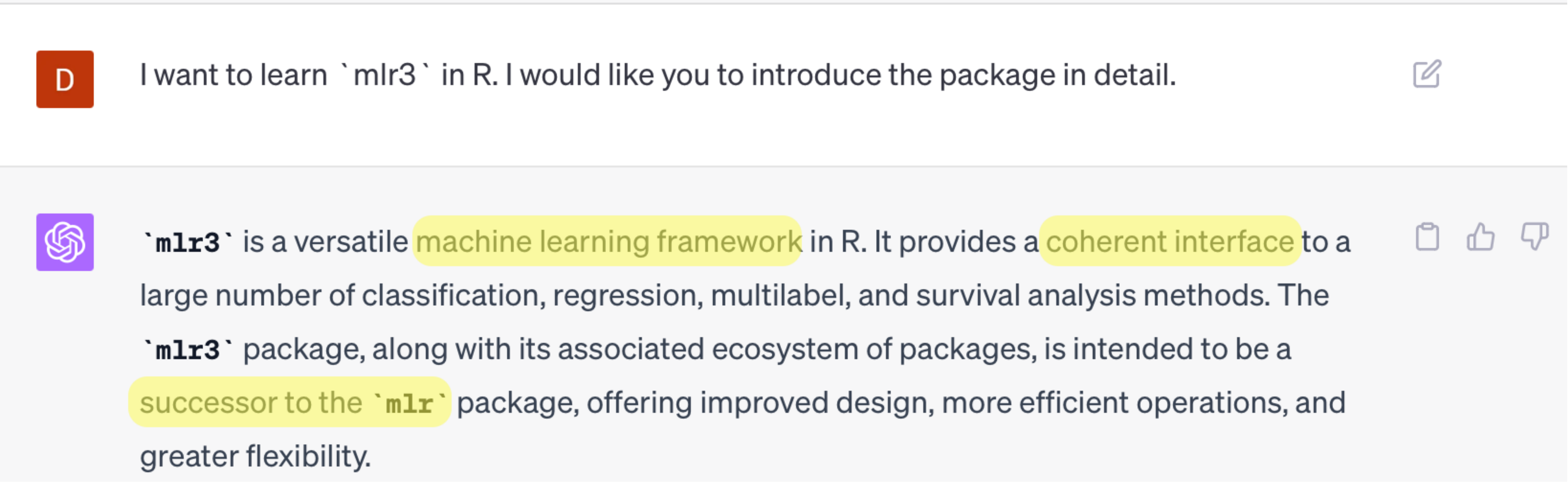
ML framework in R

tidymodels

mlr

What is `mlr3`?

`mlr3`: Machine **L**earning in **R 3**

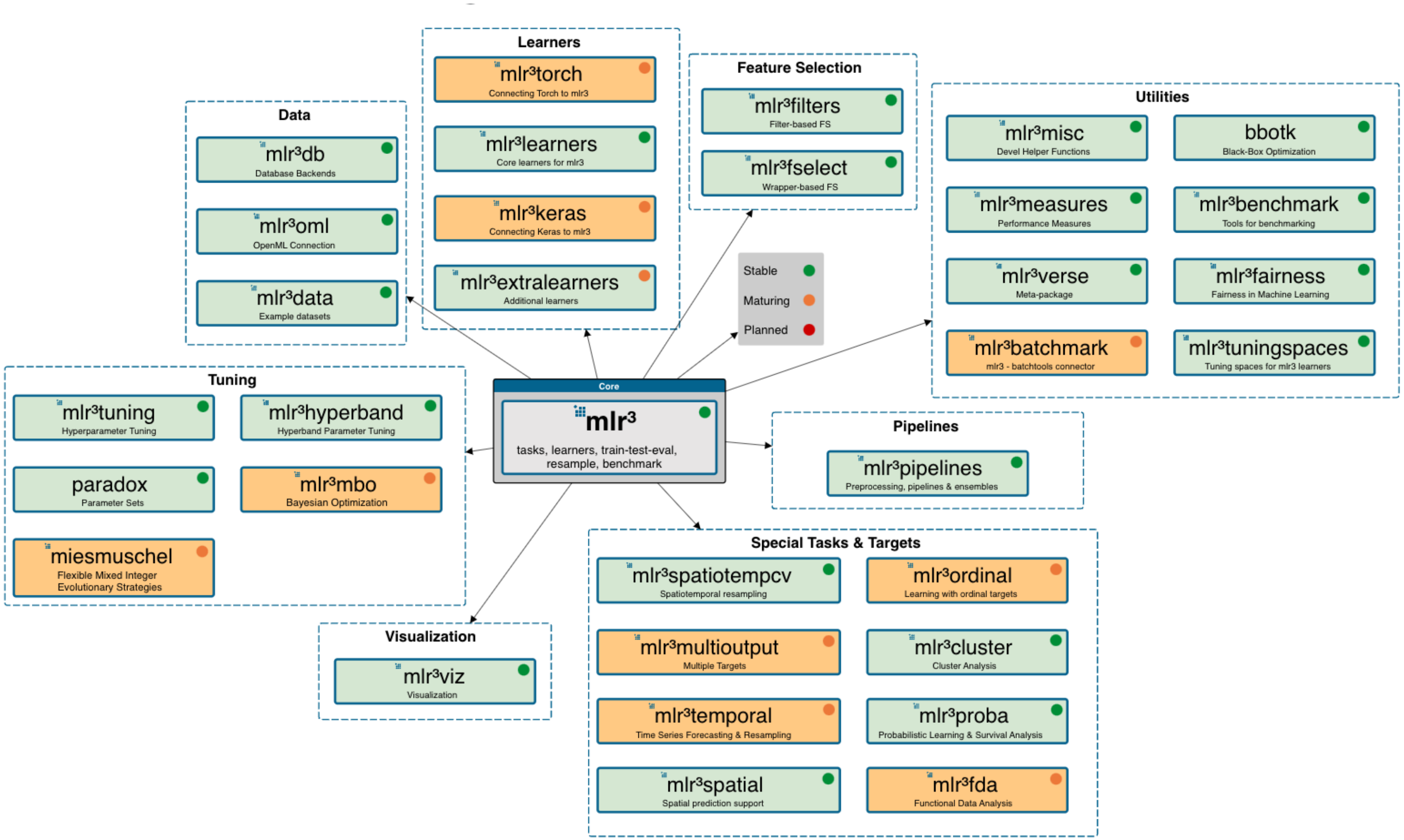


D

I want to learn `mlr3` in R. I would like you to introduce the package in detail.

`mlr3` is a versatile machine learning framework in R. It provides a coherent interface to a large number of classification, regression, multilabel, and survival analysis methods. The **`mlr3`** package, along with its associated ecosystem of packages, is intended to be a successor to the **`mlr`** package, offering improved design, more efficient operations, and greater flexibility.

`mlr3` & `mlr3verse`



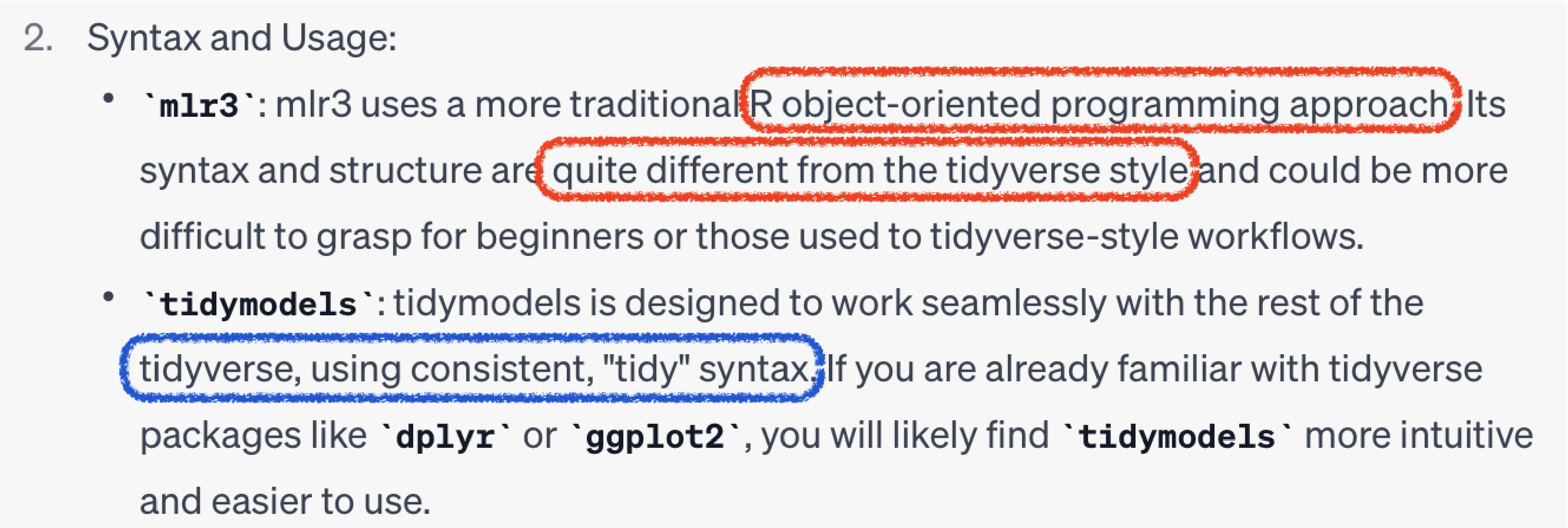
source: https://mlr3.mlr-org.com

Why `mlr3`?

- National Health Insurance System Data (NHIS-HEALS, NHIS-NSC)
- `dplyr` → `data.table`
- `Python` : `scikit-learn` = `R` : `??`
- `mlr3`: `data.table` based package

Syntax

`mlr3` vs `tidymodels`



2. Syntax and Usage:
   - **`mlr3`**: mlr3 uses a more traditional R object-oriented programming approach. Its syntax and structure are quite different from the tidyverse style and could be more difficult to grasp for beginners or those used to tidyverse-style workflows.
   - **`tidymodels`**: tidymodels is designed to work seamlessly with the rest of the tidyverse, using consistent, "tidy" syntax. If you are already familiar with tidyverse packages like **`dplyr`** or **`ggplot2`**, you will likely find **`tidymodels`** more intuitive and easier to use.

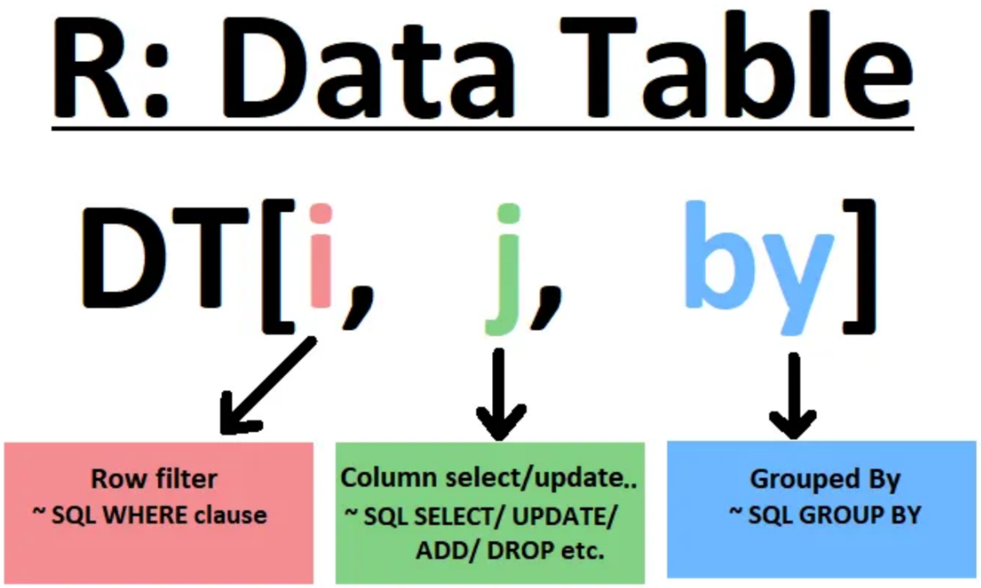
Core 1. `R6`

Object Oriented Programming (OOP)

- Objects: `foo = bar$new()`
- Methods: `$new()`
- Fields: `$baz`

```r
task = TaskClassif$new("xxx") # Objects
task$new()   # Methods
task$feature_names # Fields
```

Core 2. `data.table`



```r
DT[i >= 10] # filter rows
DT[, .(X,Y)] # select columns
DT[, mean(X), by=Y] # aggregate by group
```

Utils 1. Dictionary

```r
# Getting a specific object with `$get(key)`
mlr_learners$get("regr.rpart")
```

```
<LearnerRegrRpart:regr.rpart>: Regression Tree
* Model: -
* Parameters: xval=0
* Packages: mlr3, rpart
* Predict Types:  [response]
* Feature Types: logical, integer, numeric, factor, ordered
* Properties: importance, missings, selected_features, weights
```

Utils 1. Dictionary

```r
# Searching objects with $keys()
mlr_measures$keys() |> head()
```

```
[1] "aic"            "bic"            "classif.acc"    "classif.auc"
[5] "classif.bacc"   "classif.bbrier"
```

Utils 1. Dictionary

```r
# OR with `as.data.table()`
as.data.table(mlr_learners) |> head()
```

key	label	task_type	feature_types	packages	properties	predict_types
classif.cv_glmnet	NA	classif	logical, integer, numeric	mlr3, mlr3learners, glmnet	multiclass, selected_features, twoclass, weights	response, prob
classif.debug	Debug Learner for Classification	classif	logical, integer, numeric, character, factor, ordered	mlr3	hotstart_forward, missings, multiclass, twoclass	response, prob
classif.featureless	Featureless Classification Learner	classif	logical, integer, numeric, character, factor, ordered, POSIXct	mlr3	featureless, importance, missings, multiclass, selected_features, twoclass	response, prob
classif.glmnet	NA	classif	logical, integer, numeric	mlr3, mlr3learners, glmnet	multiclass, twoclass, weights	response, prob
classif.kknn	NA	classif	logical, integer, numeric, factor, ordered	mlr3, mlr3learners, kknn	multiclass, twoclass	response, prob
classif.lda	NA	classif	logical, integer, numeric, factor,	mlr3, mlr3learners,	multiclass, twoclass, weights	response, prob

key	label	task_type	feature_types	packages	properties	predict_types
			ordered	MASS		

Utils 2. Sugar functions

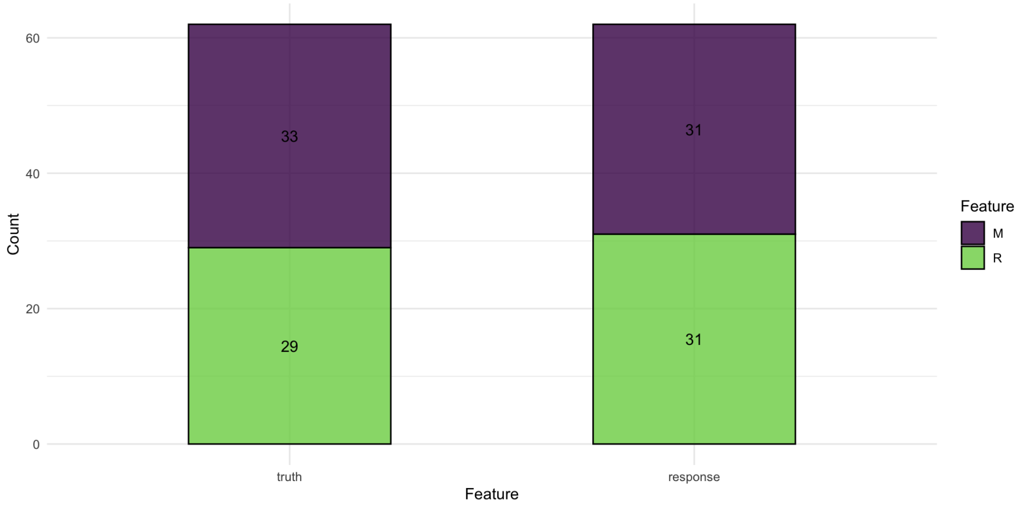
- `R6` class → `S3` type functions

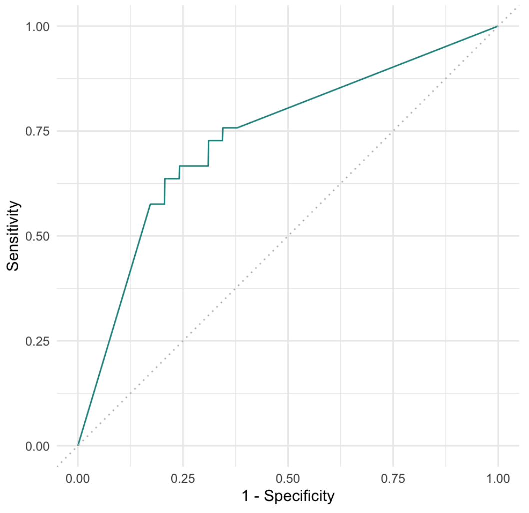
```r
# No sugar
LearnerClassifLogReg$new()

# Sugar
lrn("classif.log_reg")
```

Utils 3. `mlr3viz`

- `autoplot()` visualization

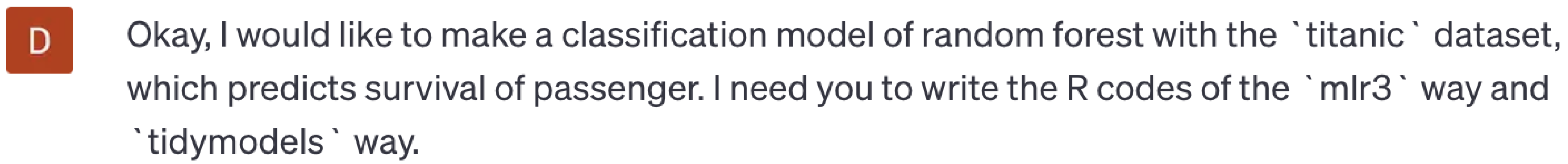
```
autoplot(pred)
autoplot(pred, type="roc")
```



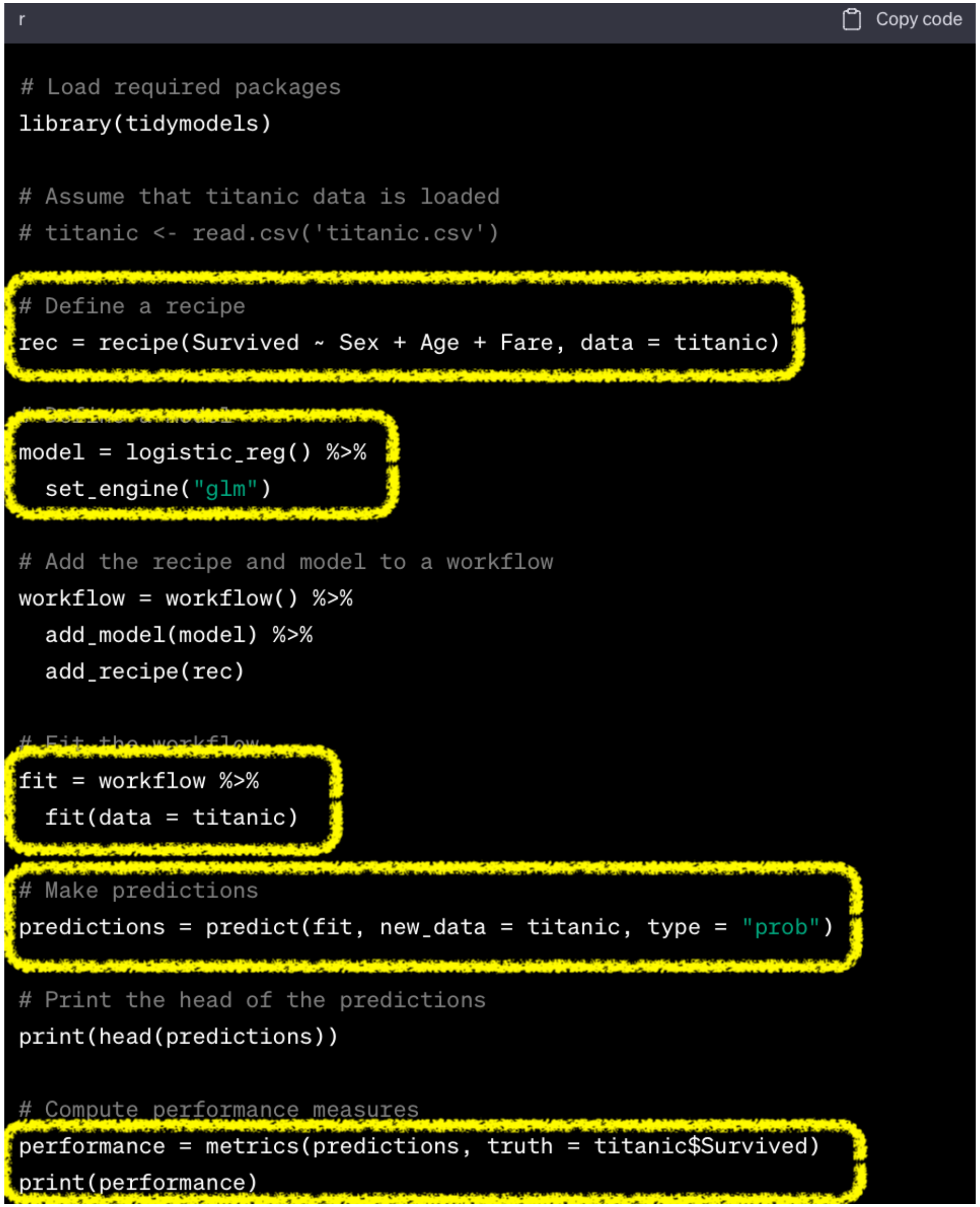


Basic modeling

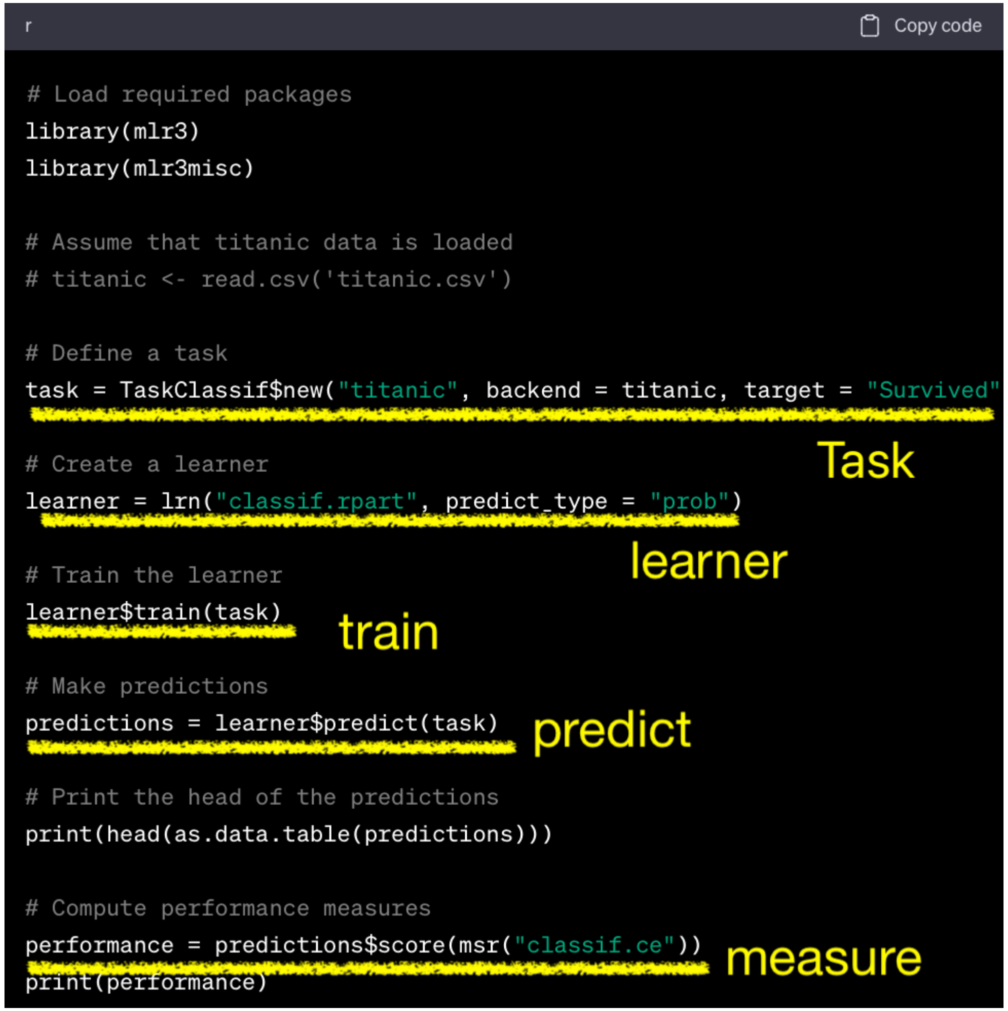
Ask ChatGPT!



Okay, I would like to make a classification model of random forest with the `titanic` dataset, which predicts survival of passenger. I need you to write the R codes of the `mlr3` way and `tidymodels` way.



```r
# Load required packages
library(tidymodels)

# Assume that titanic data is loaded
# titanic <- read.csv('titanic.csv')

# Define a recipe
rec = recipe(Survived ~ Sex + Age + Fare, data = titanic)

# Define a model
model = logistic_reg() %>%
  set_engine("glm")

# Add the recipe and model to a workflow
workflow = workflow() %>%
  add_model(model) %>%
  add_recipe(rec)

# Fit the workflow
fit = workflow %>%
  fit(data = titanic)

# Make predictions
predictions = predict(fit, new_data = titanic, type = "prob")

# Print the head of the predictions
print(head(predictions))

# Compute performance measures
performance = metrics(predictions, truth = titanic$Survived)
print(performance)
```



```r
# Load required packages
library(mlr3)
library(mlr3misc)

# Assume that titanic data is loaded
# titanic <- read.csv('titanic.csv')

# Define a task
task = TaskClassif$new("titanic", backend = titanic, target = "Survived"

# Create a learner
learner = lrn("classif.rpart", predict_type = "prob")

# Train the learner
learner$train(task)

# Make predictions
predictions = learner$predict(task)

# Print the head of the predictions
print(head(as.data.table(predictions)))

# Compute performance measures
performance = predictions$score(msr("classif.ce"))
print(performance)
```

1. Tasks

- Objects with data and metadata
- Default datasets
- Dictionary: `mlr_tasks`
- Sugar function: `tsk()`

```r
# R6 methods
# mlr_tasks$get("titanic")

# Sugar function
tsk("titanic")
```

```
<TaskClassif:titanic> (1309 x 11): Titanic
* Target: survived
* Properties: twoclass
* Features (10):
  - chr (3): cabin, name, ticket
  - dbl (2): age, fare
  - fct (2): embarked, sex
  - int (2): parch, sib_sp
  - ord (1): pclass
```

1. Tasks

Or External data as `task`

- `as_task_regr()` : regression
- `as_task_classif()`: classification
- `as_task_clust()`: clustering

```r
task_mtcars = as_task_regr(mtcars,
                            target = "mpg")
task_mtcars
```

```
<TaskRegr:mtcars> (32 x 11)
* Target: mpg
* Properties: -
* Features (10):
  - dbl (10): am, carb, cyl, disp, drat, gear, hp, qsec, vs, wt
```

2. Learners

- ML algorithms
- Dictionary: `mlr_learners`
- Sugar function: `lrn()`
- regression (`regr.~`), classification(`classif.~`), and clustering (`clust.~`)
- `library(mlr3learners)`

Extra learners

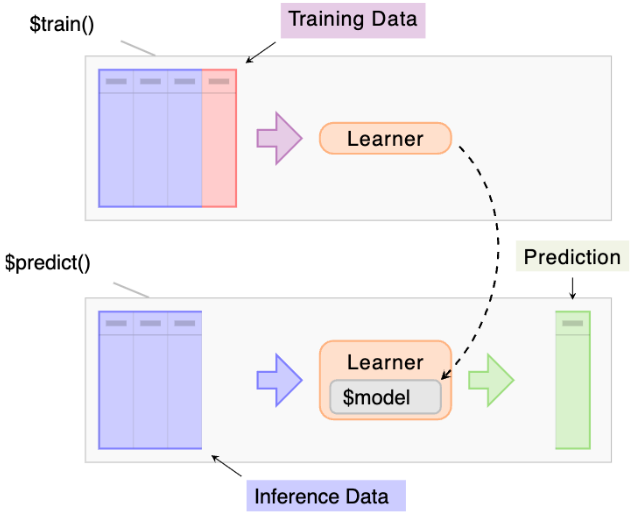
- only for `github` not CRAN
- e.g., `lightGBM`

```r
# remotes::install_github("mlr-org/mlr3extralearners@*release")
library(mlr3extralearners)
```

2. Learners

- `$train()`, `$predict()`

```
task = tsk("german_credit")
learner_dt = lrn("classif.rpart", predict_type="prob")
split = partition(task, ratio=.7)
learner_dt$train(task, row_ids = split$train)
prediction = learner_dt$predict(task, row_ids = split$test)
```



source: mlr3books

2. Learners

confusion matrix

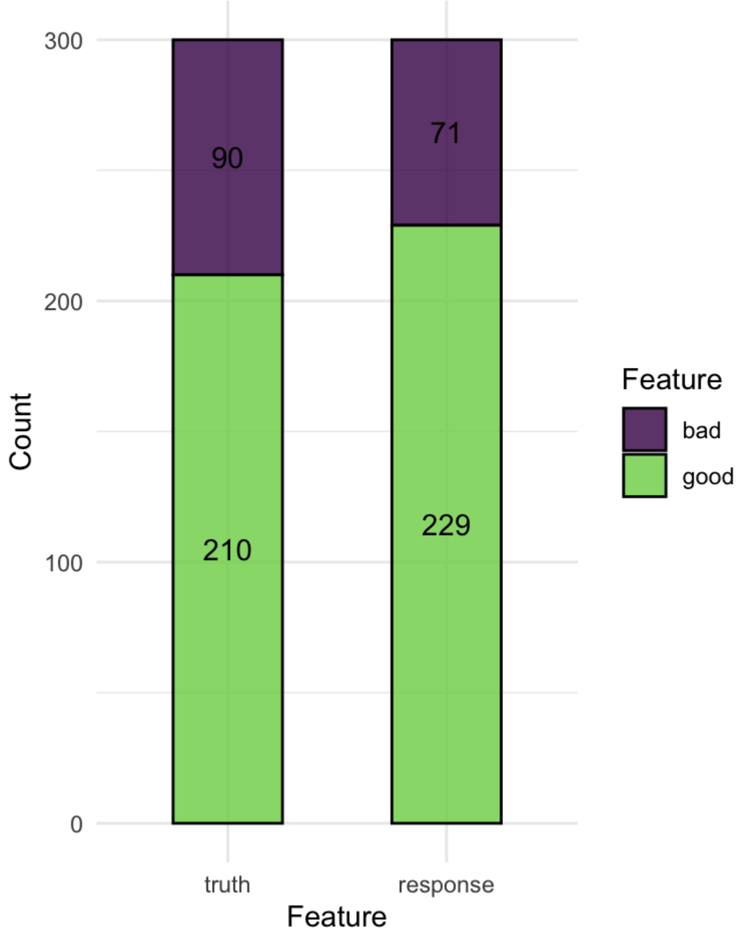
```
prediction$confusion
```

```
        truth
response good bad
    good  184  45
    bad    26  45
```

2. Learners

Or with `mlr3viz`

```r
autoplot(prediction)
```


300
200
100
0
Count
90
210
71
229
truth
response
Feature
Feature
bad
good

Hyperparameter

```r
# with learner
learner = lrn("classif.rpart", maxdepth = 1)

# Or
learner$param_set$set_values(xval = 2, maxdepth=3, cp=.5)

learner$param_set$values
```

```
$xval
[1] 2

$maxdepth
[1] 3

$cp
[1] 0.5
```

Hyperparameter

Setting hyperparameters

- `$param_set` of learners
- setting class, lower, upper

```r
as.data.table(learner$param_set) |> head()
```

id	class	lower	upper	levels	nlevels	is_bounded	special_vals	default	storage_type	tags
cp	ParamDbl	0	1	NULL	Inf	TRUE	NULL	0.01	numeric	train
keep_model	ParamLgl	NA	NA	TRUE, FALSE	2	TRUE	NULL	FALSE	logical	train
maxcompete	ParamInt	0	Inf	NULL	Inf	FALSE	NULL	4	integer	train
maxdepth	ParamInt	1	30	NULL	30	TRUE	NULL	30	integer	train
maxsurrogate	ParamInt	0	Inf	NULL	Inf	FALSE	NULL	5	integer	train
minbucket	ParamInt	1	Inf	NULL	Inf	FALSE	NULL	<environment: 0x147c590c8>	integer	train

3. Measures

- Evaluation of performances
- Dictionary: `mlr_measures`
- Sugar function: `msr()`, `msrs()`
- `classif.~`, `regr.~`
- `$score()`

```
as.data.table(mlr_measures) |> head()
```

key	label	task_type	packages	predict_type	task_properties
aic	Akaike Information Criterion	NA	mlr3	NA	
bic	Bayesian Information Criterion	NA	mlr3	NA	
classif.acc	Classification Accuracy	classif	mlr3 , mlr3measures	response	
classif.auc	Area Under the ROC Curve	classif	mlr3 , mlr3measures	prob	twoclass
classif.bacc	Balanced Accuracy	classif	mlr3 , mlr3measures	response	
classif.bbrier	Binary Brier Score	classif	mlr3 , mlr3measures	prob	twoclass

3. Measures

`msr()`: a single performance

```r
measure = msr("classif.acc")
prediction$score(measure)
```

```
classif.acc 
  0.7633333 
```

3. Measures

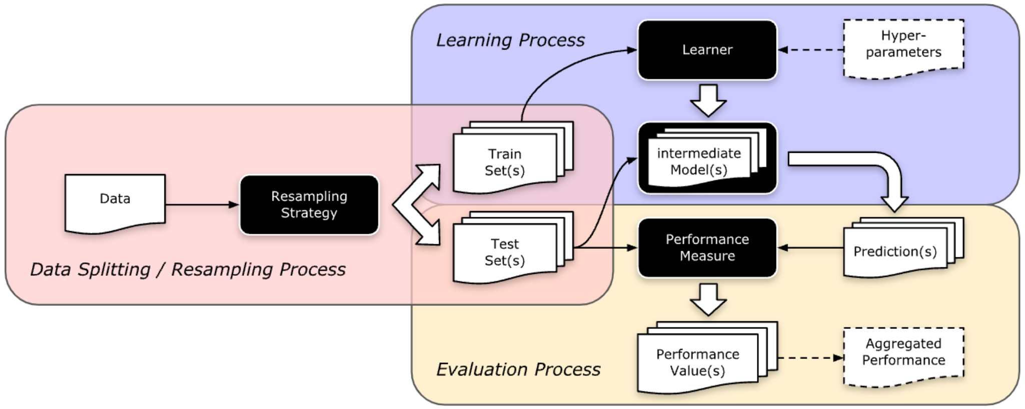
`msrs()`: multiple performances

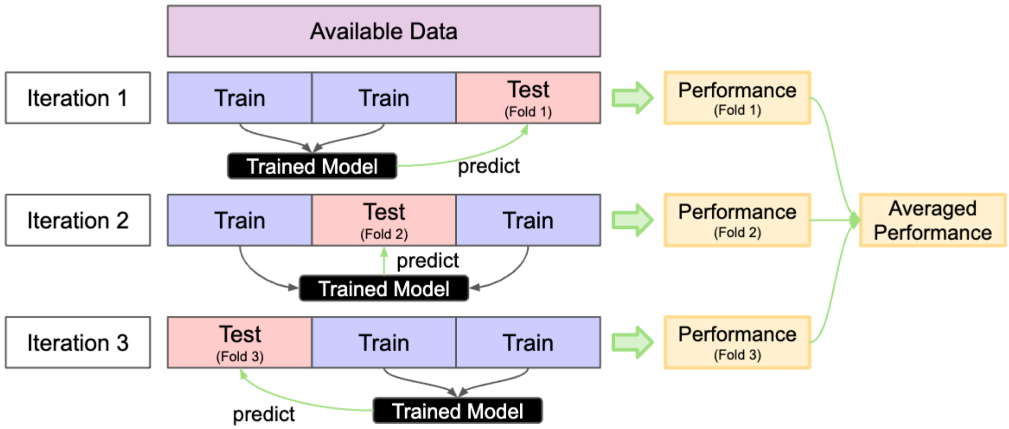
```r
# Multiple measurements
measures = msrs(c("classif.acc","classif.ppv","classif.npv","classif.auc"))
prediction$score(measures)
```

```
classif.acc classif.ppv classif.npv classif.auc 
  0.7633333   0.8034934   0.6338028   0.7558730 
```

Resampling

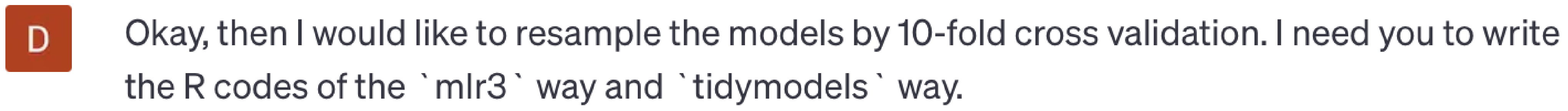
Resampling



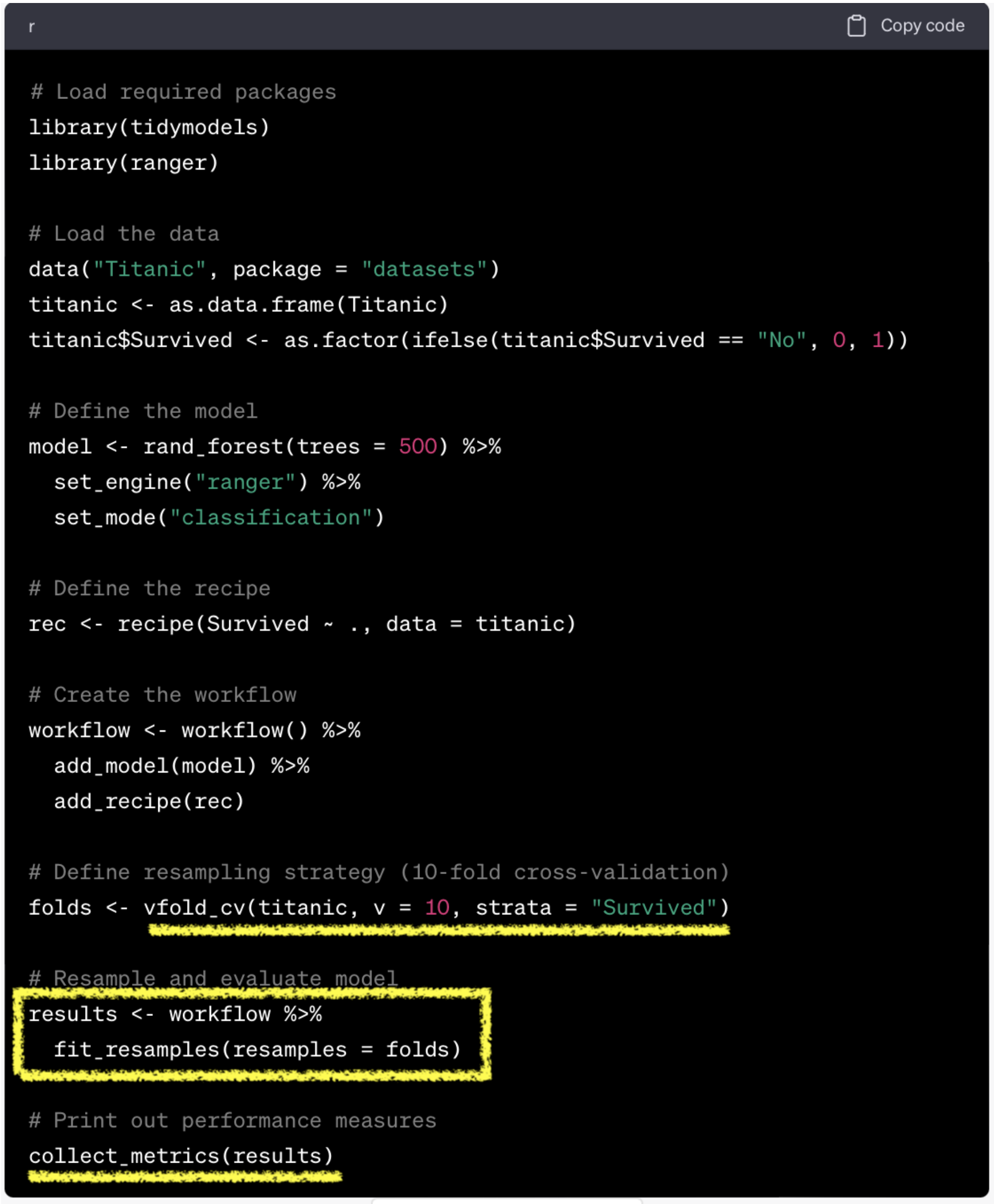


- Split available data into multiple training and test sets
- **Reliable** performance
- Prevent **Overfitting**

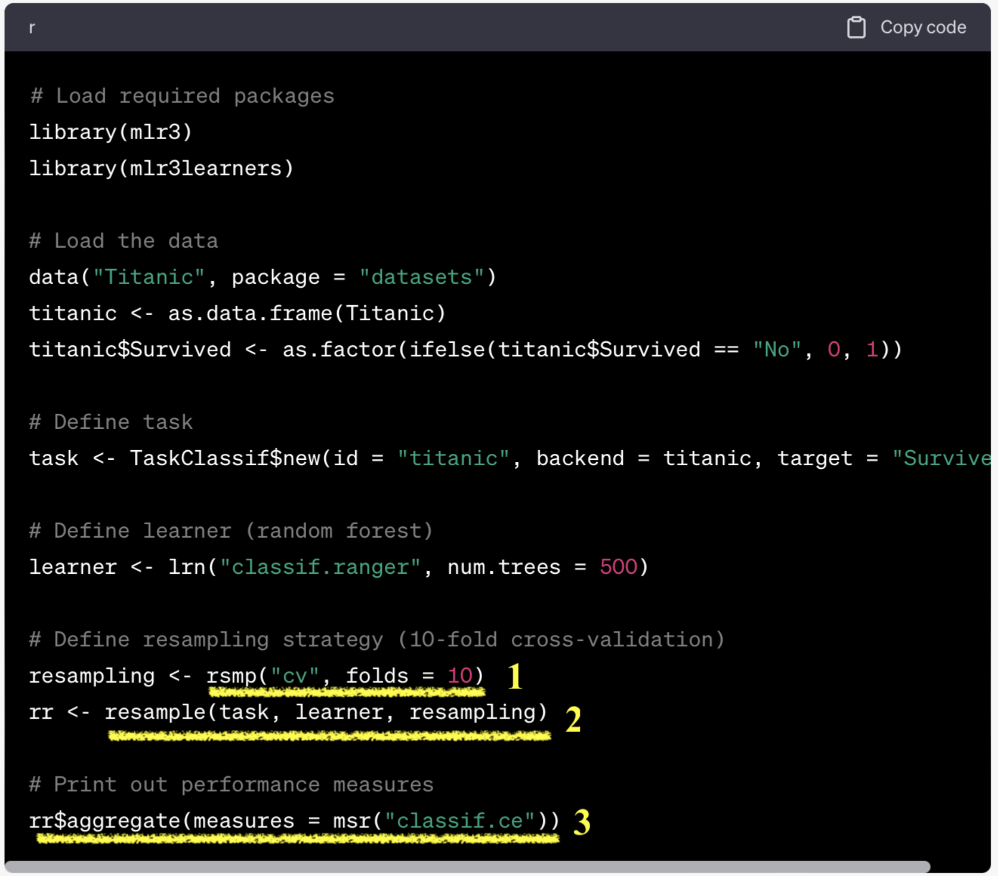
`tidymodels` vs `mlr3`



Okay, then I would like to resample the models by 10-fold cross validation. I need you to write the R codes of the `mlr3` way and `tidymodels` way.



```r
# Load required packages
library(tidymodels)
library(ranger)

# Load the data
data("Titanic", package = "datasets")
titanic <- as.data.frame(Titanic)
titanic$Survived <- as.factor(ifelse(titanic$Survived == "No", 0, 1))

# Define the model
model <- rand_forest(trees = 500) %>%
  set_engine("ranger") %>%
  set_mode("classification")

# Define the recipe
rec <- recipe(Survived ~ ., data = titanic)

# Create the workflow
workflow <- workflow() %>%
  add_model(model) %>%
  add_recipe(rec)

# Define resampling strategy (10-fold cross-validation)
folds <- vfold_cv(titanic, v = 10, strata = "Survived")

# Resample and evaluate model
results <- workflow %>%
  fit_resamples(resamples = folds)

# Print out performance measures
collect_metrics(results)
```



```r
# Load required packages
library(mlr3)
library(mlr3learners)

# Load the data
data("Titanic", package = "datasets")
titanic <- as.data.frame(Titanic)
titanic$Survived <- as.factor(ifelse(titanic$Survived == "No", 0, 1))

# Define task
task <- TaskClassif$new(id = "titanic", backend = titanic, target = "Survive

# Define learner (random forest)
learner <- lrn("classif.ranger", num.trees = 500)

# Define resampling strategy (10-fold cross-validation)
resampling <- rsmp("cv", folds = 10)
rr <- resample(task, learner, resampling)

# Print out performance measures
rr$aggregate(measures = msr("classif.ce"))
```

Resampling

- Dictionary: `mlr_resamplings`
- Sugar function: `rsmp()`

```r
as.data.table(mlr_resamplings)
```

key	label	params	iters
bootstrap	Bootstrap	ratio , repeats	30
custom	Custom Splits		NA
custom_cv	Custom Split Cross-Validation		NA
cv	Cross-Validation	folds	10
holdout	Holdout	ratio	1
insample	Insample Resampling		1
loo	Leave-One-Out		NA
repeated_cv	Repeated Cross-Validation	folds , repeats	100
subsampling	Subsampling	ratio , repeats	30

Resampling

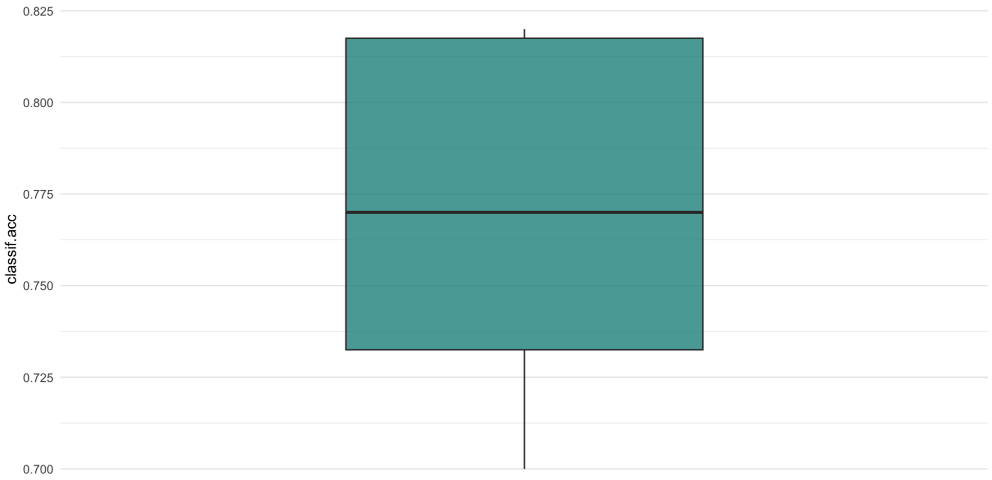
- `resample()`: initiate resampling
- `$aggregate()`: aggregate resampling performance

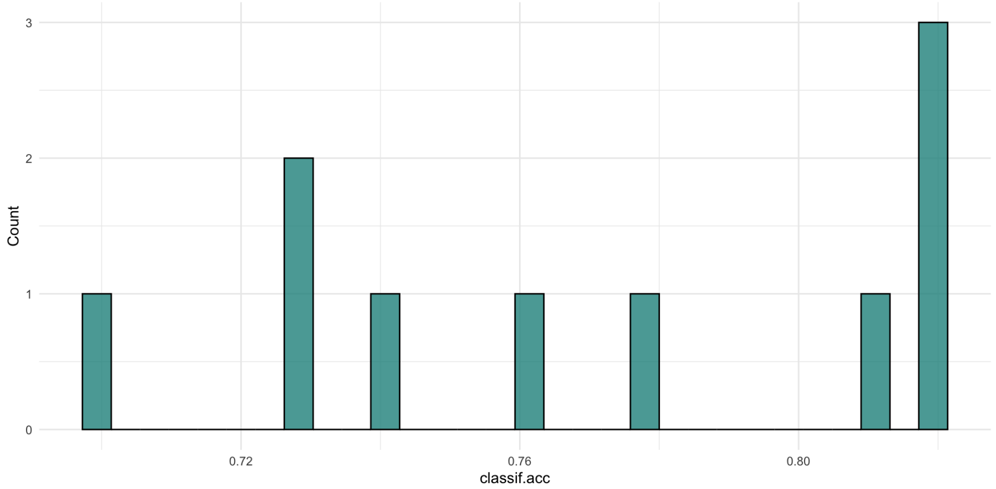
```r
task = tsk("german_credit")
learner = lrn("classif.ranger", predict_type="prob")
resample = rsmp("cv", folds=10)
rr  = resample(task, learner, resample, store_model=T)
measures = msrs(c("classif.acc","classif.ppv","classif.npv","classif.auc"))
rr$aggregate(measures)
```

```
classif.acc classif.ppv classif.npv classif.auc 
  0.7710000   0.7890524   0.6956910   0.7979774 
```

Resampling

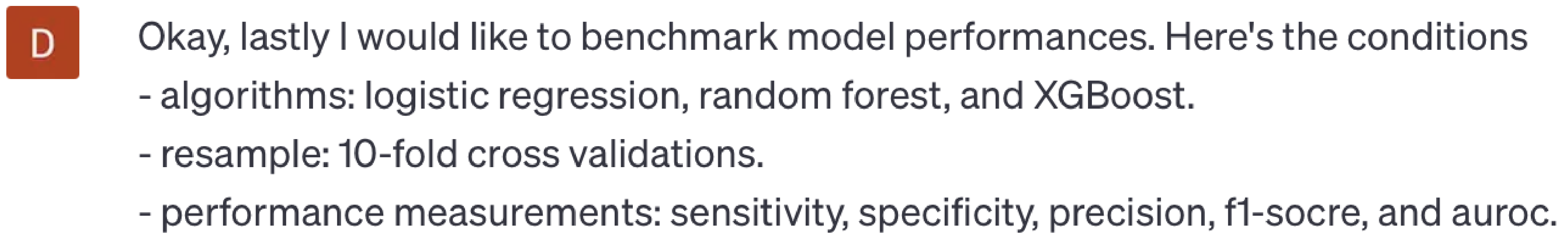
```r
autoplot(rr, type="boxplot", measure = msr("classif.acc"))
autoplot(rr, type="histogram", measure = msr("classif.acc"))
```



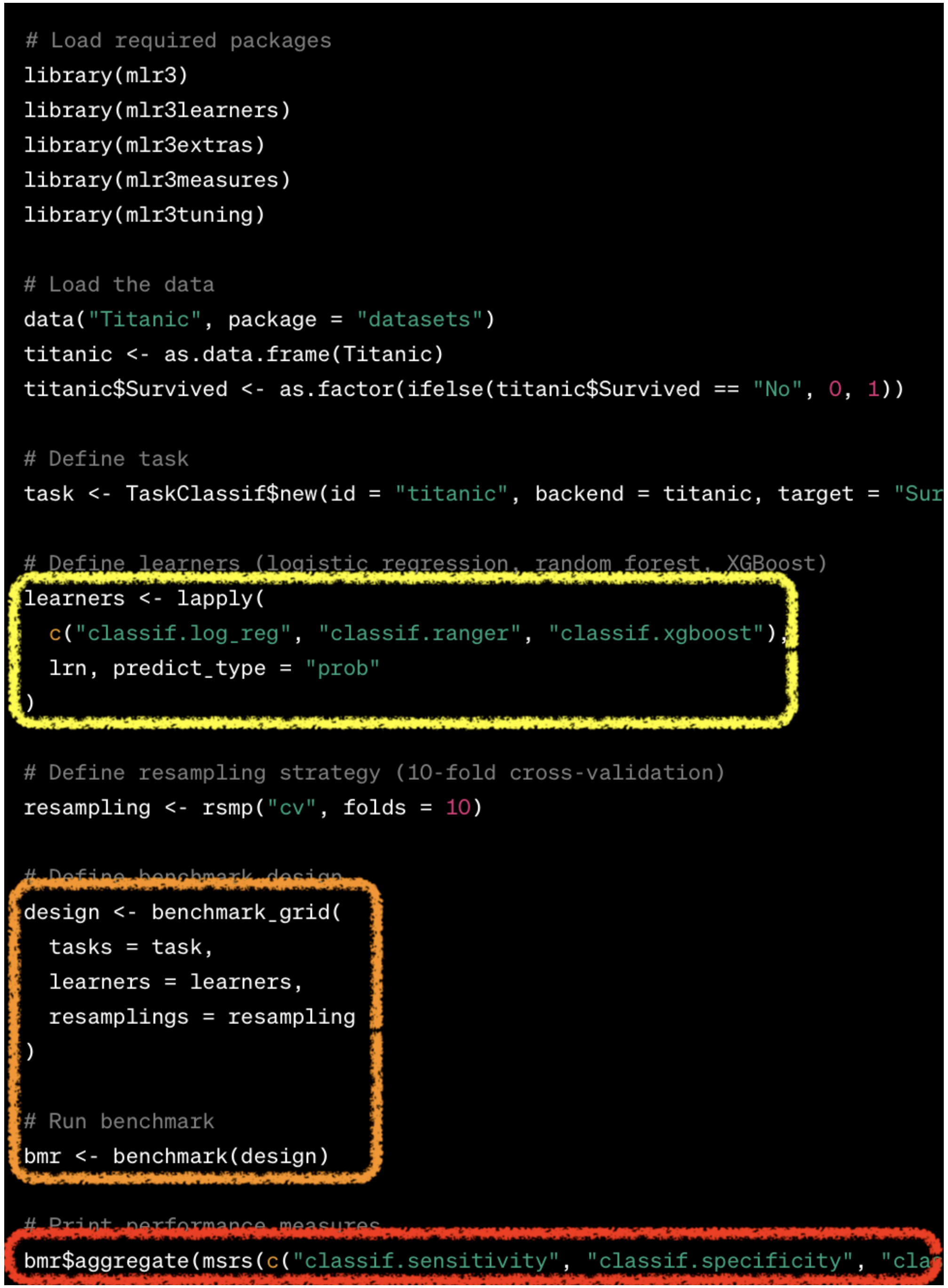


Benchmarking

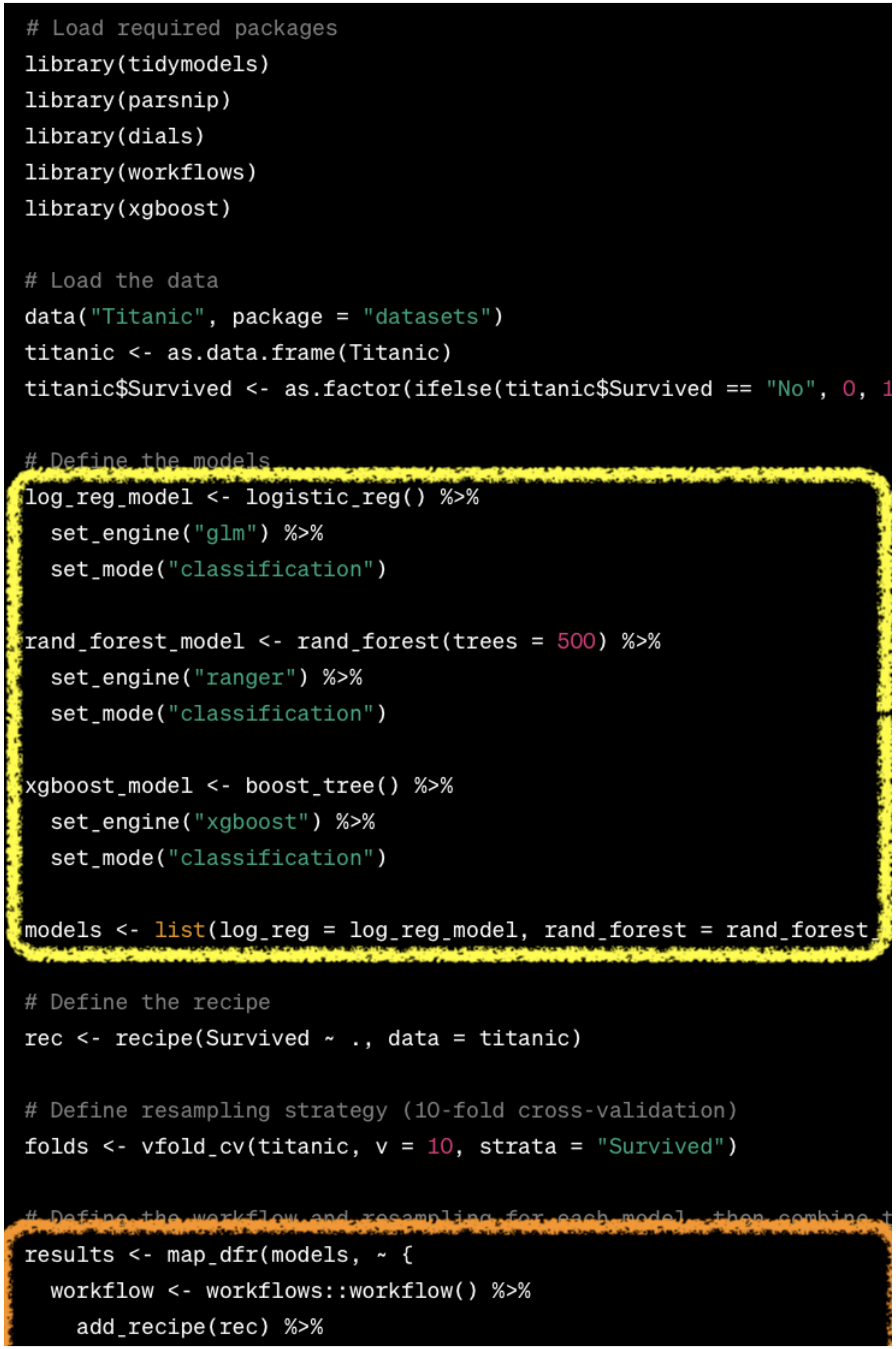
`tidymodels` vs `mlr3`



Okay, lastly I would like to benchmark model performances. Here's the conditions
- algorithms: logistic regression, random forest, and XGBoost.
- resample: 10-fold cross validations.
- performance measurements: sensitivity, specificity, precision, f1-socre, and auroc.



```
# Load required packages
library(mlr3)
library(mlr3learners)
library(mlr3extras)
library(mlr3measures)
library(mlr3tuning)

# Load the data
data("Titanic", package = "datasets")
titanic <- as.data.frame(Titanic)
titanic$Survived <- as.factor(ifelse(titanic$Survived == "No", 0, 1))

# Define task
task <- TaskClassif$new(id = "titanic", backend = titanic, target = "Sur

# Define learners (logistic regression, random forest, XGBoost)
learners <- lapply(
  c("classif.log_reg", "classif.ranger", "classif.xgboost"),
  lrn, predict_type = "prob"
)

# Define resampling strategy (10-fold cross-validation)
resampling <- rsmp("cv", folds = 10)

# Define benchmark design
design <- benchmark_grid(
  tasks = task,
  learners = learners,
  resamplings = resampling
)

# Run benchmark
bmr <- benchmark(design)

# Print performance measures
bmr$aggregate(msrs(c("classif.sensitivity", "classif.specificity", "cla
```



```
# Load required packages
library(tidymodels)
library(parsnip)
library(dials)
library(workflows)
library(xgboost)

# Load the data
data("Titanic", package = "datasets")
titanic <- as.data.frame(Titanic)
titanic$Survived <- as.factor(ifelse(titanic$Survived == "No", 0,

# Define the models
log_reg_model <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

rand_forest_model <- rand_forest(trees = 500) %>%
  set_engine("ranger") %>%
  set_mode("classification")

xgboost_model <- boost_tree() %>%
  set_engine("xgboost") %>%
  set_mode("classification")

models <- list(log_reg = log_reg_model, rand_forest = rand_forest

# Define the recipe
rec <- recipe(Survived ~ ., data = titanic)

# Define resampling strategy (10-fold cross-validation)
folds <- vfold_cv(titanic, v = 10, strata = "Survived")

# Define the workflow and resampling for each model, then combine
results <- map_dfr(models, ~ {
  workflow <- workflows::workflow() %>%
    add_recipe(rec) %>%
```

Benchmarking

- Comparison of multiple learners on a single task (or multiple tasks).
- `benchmark_grid()`: design a benchmarking

```r
tasks = tsks(c("german_credit", "sonar", "breast_cancer"))

learners = list(
  lrn("classif.log_reg", predict_type="prob", id="LR"),
  lrn("classif.rpart", predict_type="prob", id="DT"),
  lrn("classif.ranger", predict_type="prob", id="RF")
)

rsmp = rsmp("cv", folds=5)

design = benchmark_grid(
  tasks = tasks,
  learners = learners,
  resamplings = rsmp)
```

Benchmarking

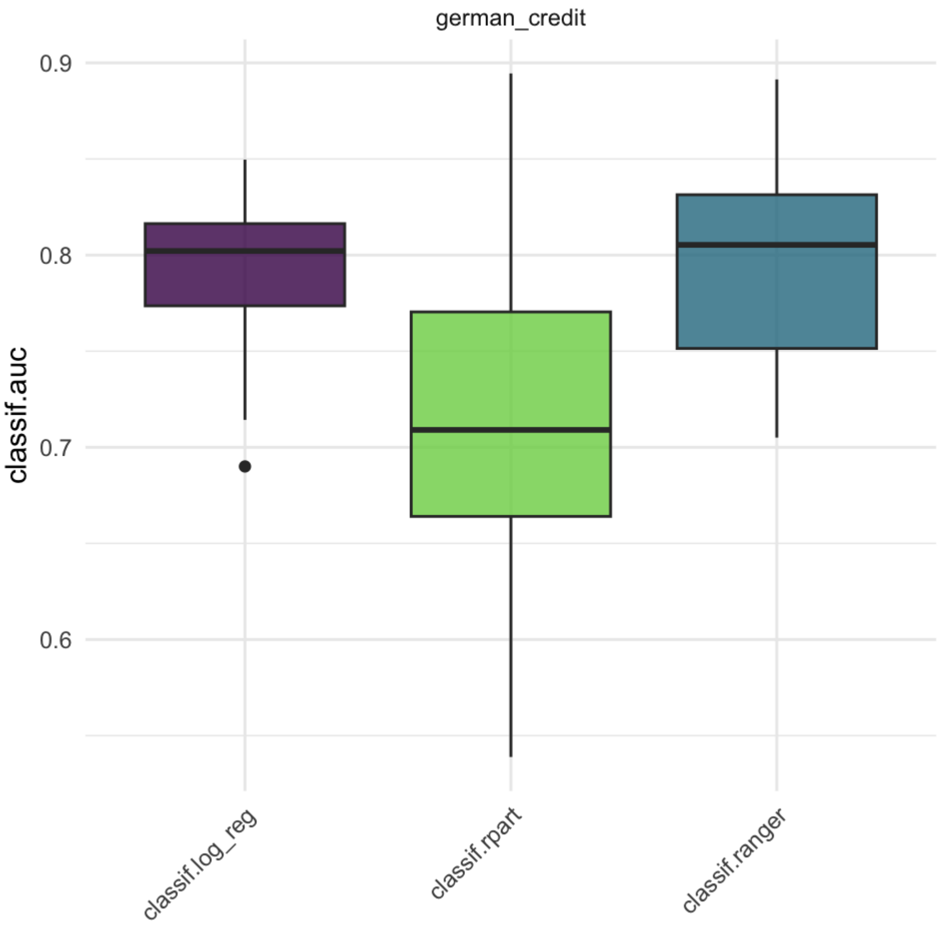
- `benchmark()`: execute benchmarking

```r
bmr = benchmark(design)
measures = msrs(c("classif.acc","classif.ppv", "classif.npv", "classif.auc"))
as.data.table(bmr$aggregate(measures))[,-c("nr","resample_result","resampling_id","iters")] |> DT()
```

task_id	learner_id	classif.acc	classif.ppv	classif.npv	classif.auc
german_credit	LR	0.7540000	0.7959935	0.6128794	0.7682786
german_credit	DT	0.7220000	0.7720000	0.5715187	0.7009023
german_credit	RF	0.7670000	0.7866093	0.6820459	0.7916496
sonar	LR	0.7027875	0.7229497	0.6805154	0.7122449
sonar	DT	0.7262485	0.7250771	0.7382659	0.7524838
sonar	RF	0.8174216	0.8101012	0.8425397	0.9232502
breast_cancer	LR	0.9252791	0.9361270	0.9195608	0.9418515
breast_cancer	DT	0.9502362	0.9167371	0.9675106	0.9543396
breast_cancer	RF	0.9751181	0.9549859	0.9860113	0.9938067

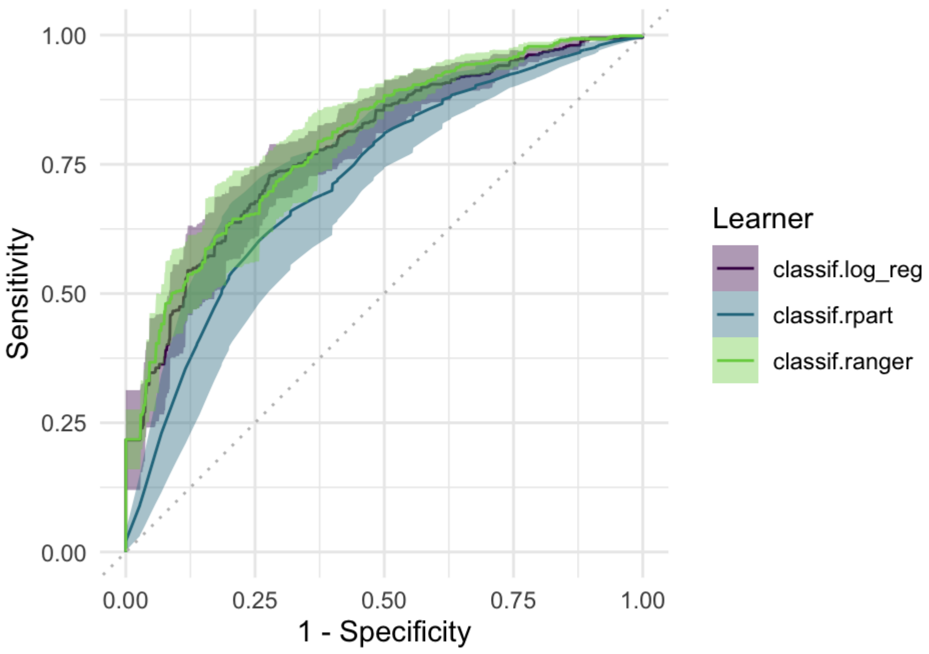
Benchmarking result

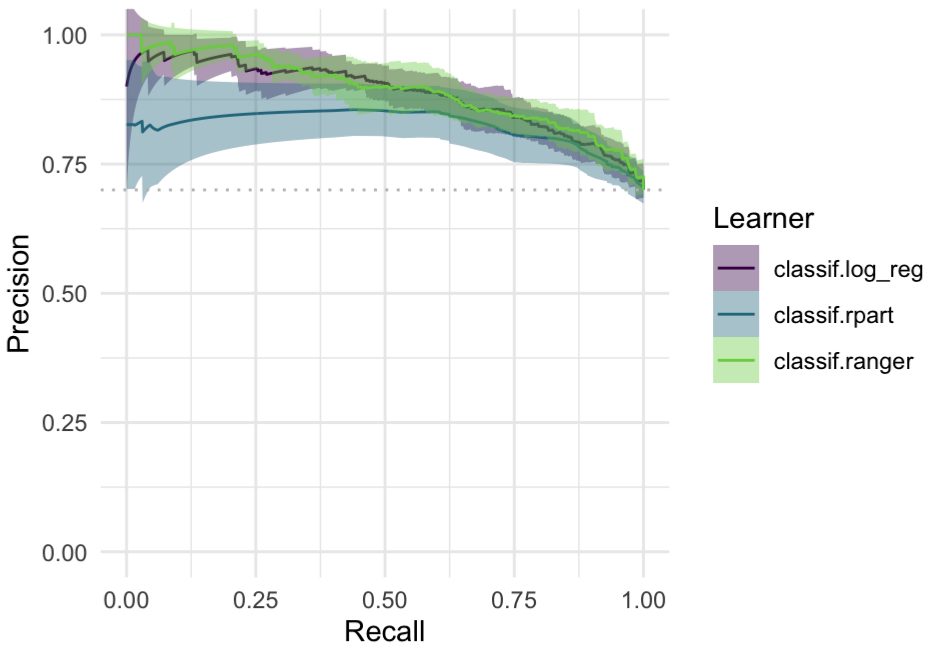
```r
task = tsk("german_credit")
learners = list(
  lrn("classif.log_reg", predict_type="prob"),
  lrn("classif.rpart", predict_type="prob"),
  lrn("classif.ranger", predict_type="prob")
)
cv10 = rsmp("cv", folds=10)
design = benchmark_grid(
  task = task,
  learners = learners,
  resamplings = cv10)
bmr = benchmark(design)
autoplot(bmr, measure = msr("classif.auc"))
```



Benchmarking result

```r
autoplot(bmr, type = "roc")
autoplot(bmr, type = "prc")
```


Sensitivity
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1 - Specificity
Learner
classif.log_reg
classif.rpart
classif.ranger


Precision
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
Recall
Learner
classif.log_reg
classif.rpart
classif.ranger

More about `mlr3`

- Hyperparameter optimization
- Feature selection
- ML pipelines

Summary

`mlr3`

- `R6`, `data.table` based ML framework
- Sugar function + Dictionary
- Task, Learner, Measure
- Resampling
- Benchmarking
- Still in development (ver 0.16.0)
- A great textbook: **mlr3book**

Thank you for listening!