funcml

`funcml` provides a machine learning framework for R. The package also includes native interpretability helpers for permutation importance, partial dependence, ICE, ALE, local surrogate explanations, SHAP, interaction strength, greedy breakdown profiles, and global surrogate models.

It also exposes native ensemble learners through `model = "stacking"` and `model = "superlearner"`.

The package is intentionally opinionated. It is not designed to compete feature for feature with broader frameworks such as `tidymodels`, `mlr3`, or `caret`. Instead, `funcml` focuses on a compact and explicit framework for users who want to fit, compare, tune, interpret, and estimate models through one coherent interface, with formula syntax as the main model-specification surface.

This design comes with a deliberate tradeoff:

- preprocessing is expected to happen outside `funcml`, so the data passed to `fit()` is the exact data being modeled
- the package favors explicit inputs and direct model specification over hidden preprocessing state inside training pipelines
- cross-validation is still the default, but the package also supports holdout splits, grouped CV, and time-aware rolling evaluation through the same resampling interface

Users who need broader preprocessing orchestration or specialized resampling designs should use a more complete framework.

Learner coverage currently includes:

- regression and classification: `glm`, `rpart`, `glmnet`, `ranger`, `nnet`, `e1071_svm`, `randomForest`, `gbm`, `kknn`, `ctree`, `cforest`, `lightgbm`, `xgboost`, `stacking`, `superlearner`
- regression and binary classification: `earth`, `gam`, `bart`
- classification only: `C50`, `naivebayes`, `fda`, `lda`, `qda`
- binary classification only: `adaboost`
- regression only: `pls`

```
fit_obj <- fit(mpg ~ wt + hp, data = mtcars, model = "ranger")

permute_obj <- interpret(fit_obj, mtcars, method = "permute", nsim = 5)
pdp_obj <- interpret(fit_obj, mtcars, method = "pdp", features = "wt")
ale_obj <- interpret(fit_obj, mtcars, method = "ale", features = "wt")
local_obj <- interpret(fit_obj, mtcars, method = "local_model", newdata = mtcars[1, , drop = FALSE], k =
shap_obj <- interpret(fit_obj, mtcars, method = "shap", newdata = mtcars[1, , drop = FALSE], nsim = 20)
profile_obj <- interpret(fit_obj, mtcars, method = "profile", newdata = mtcars[1, , drop = FALSE])
surrogate_obj <- interpret(fit_obj, mtcars, method = "surrogate")
```

```
eval_obj <- evaluate(mpg ~ wt + hp, data = mtcars, model = "glm", resampling = cv(5))
eval_obj
#> <funcml_eval> model: glm | task: regression
#>   metric      mean         sd n  std_error conf_level   conf_low  conf_high
#> 1   rmse 2.8383657 1.02012964 5 0.45621584       0.95 1.57170749  4.1050240
```

```
#> 2    mae 2.2469779 0.74220889 5 0.33192591       0.95 1.32540389  3.1685520
#> 3    mse 8.8888516 6.32085971 5 2.82677440       0.95 1.04046765 16.7372355
#> 4  medae 1.4926950 0.43102307 5 0.19275938       0.95 0.95750916  2.0278808
#> 5   mape 0.1167652 0.03074759 5 0.01375074       0.95 0.07858703  0.1549434
#> 6    rsq 0.5002385 0.29363137 5 0.13131594       0.95 0.13564696  0.8648300
```

```
tune_grid <- expand.grid(intercept = c(TRUE, FALSE))

tune_obj <- tune(mpg ~ wt + hp, data = mtcars, model = "glm", grid = tune_grid, search = "random", n_eva

tune_obj
#> <funcml_tune> metric=rmse direction=min search=random
#> Best:
#>   intercept     mean        sd n std_error conf_level  conf_low conf_high
#> 1      TRUE 2.772583 0.9774437 3 0.5643274       0.95 0.3444783  5.200688
```

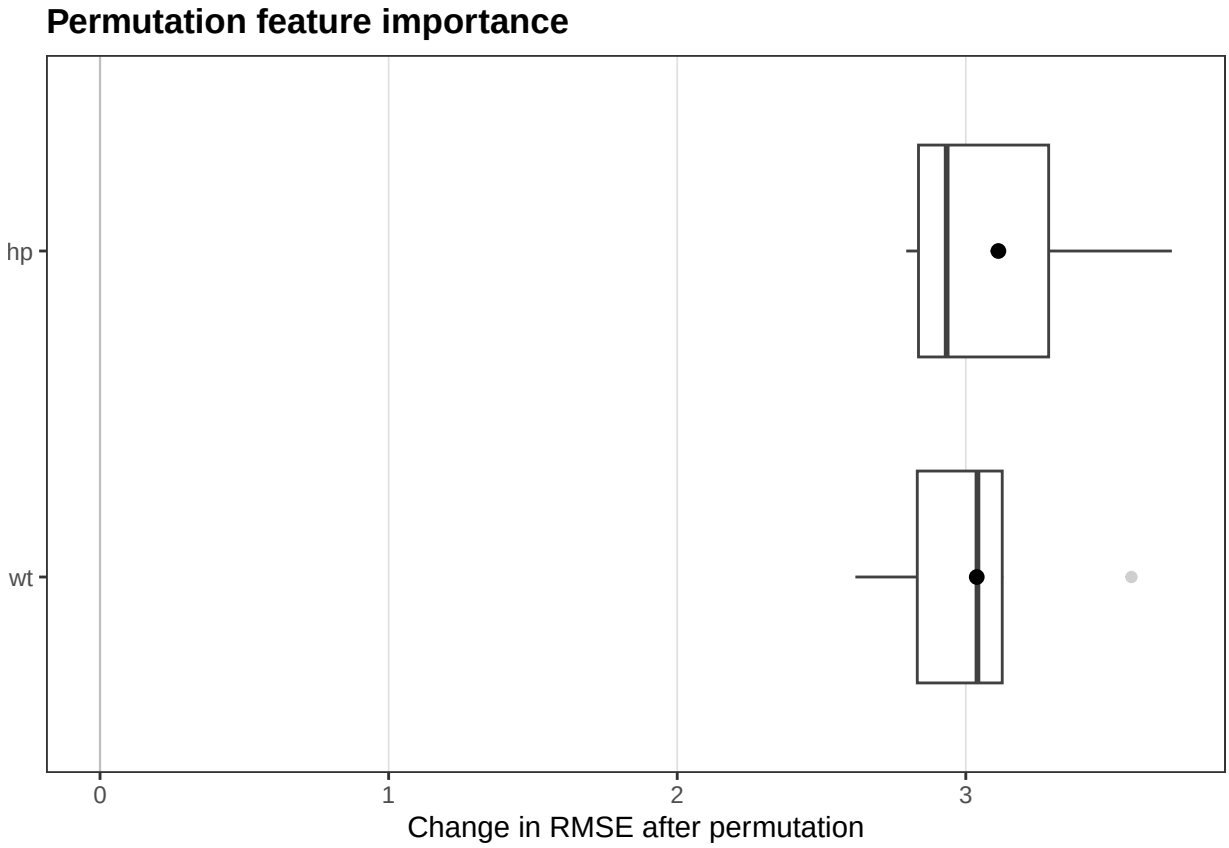
Nested CV is available through `outer_resampling`. The inner `resampling` argument still selects the best configuration, and the outer resampling loop provides an unbiased estimate of tuned model-selection performance.

```
nested_tune_obj <- tune(
  mpg ~ wt + hp,
  data = mtcars,
  model = "glm",
  grid = tune_grid,
  resampling = cv(v = 3, seed = 1),
  outer_resampling = cv(v = 4, seed = 2),
  metric = "rmse",
  seed = 1
)
nested_tune_obj
#> <funcml_tune> metric=rmse direction=min search=grid
#> Best:
#>   intercept     mean        sd n std_error conf_level  conf_low conf_high
#> 1      TRUE 2.772583 0.9774437 3 0.5643274       0.95 0.3444783  5.200688
#> Nested resampling:
#>   metric     mean      sd n std_error conf_level conf_low conf_high
#> 1   rmse 2.886158 1.08508 4 0.5425399       0.95 1.159553  4.612762
```

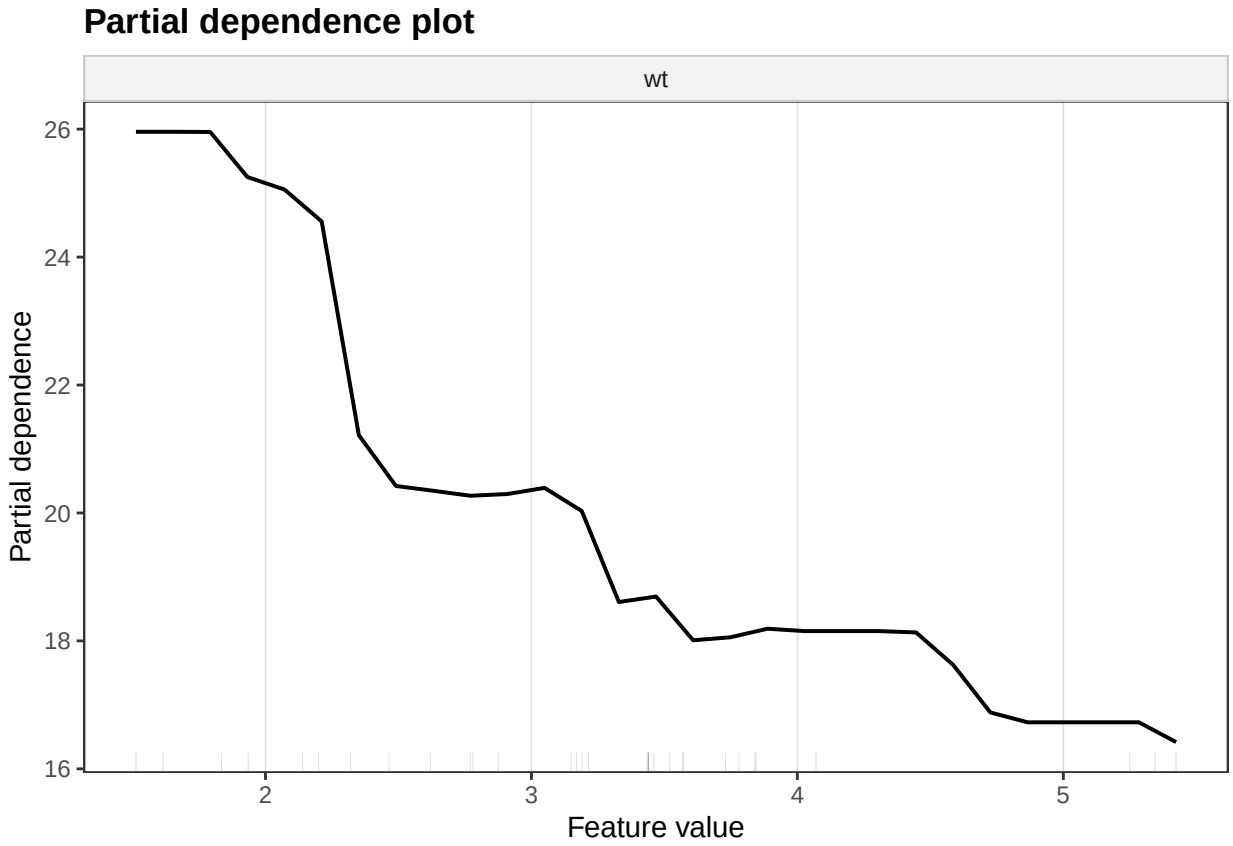
```
interaction_obj <- interpret(fit_obj, mtcars, method = "interaction")
plot(interaction_obj)
```

```
learners()
#>  [1] "glm"          "rpart"        "glmnet"       "ranger"       "nnet"
#>  [6] "e1071_svm"    "randomForest" "gbm"          "C50"          "kknn"
#> [11] "earth"        "gam"          "naivebayes"   "fda"          "adaboost"
#> [16] "pls"          "ctree"        "cforest"      "lda"          "qda"
#> [21] "lightgbm"     "bart"         "xgboost"      "stacking"     "superlearner"
```

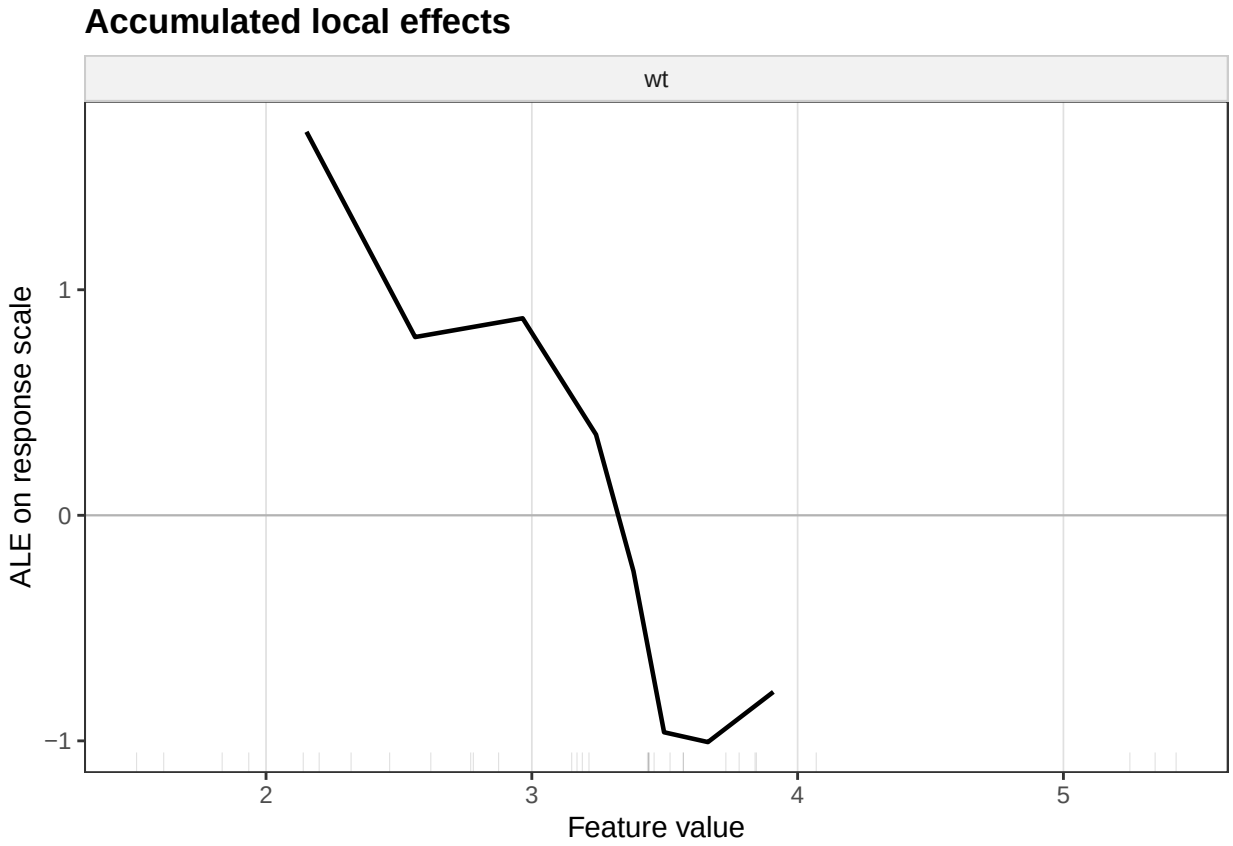
```
plot(permute_obj)
```


Permutation feature importance
hp
wt
0
1
2
3
Change in RMSE after permutation

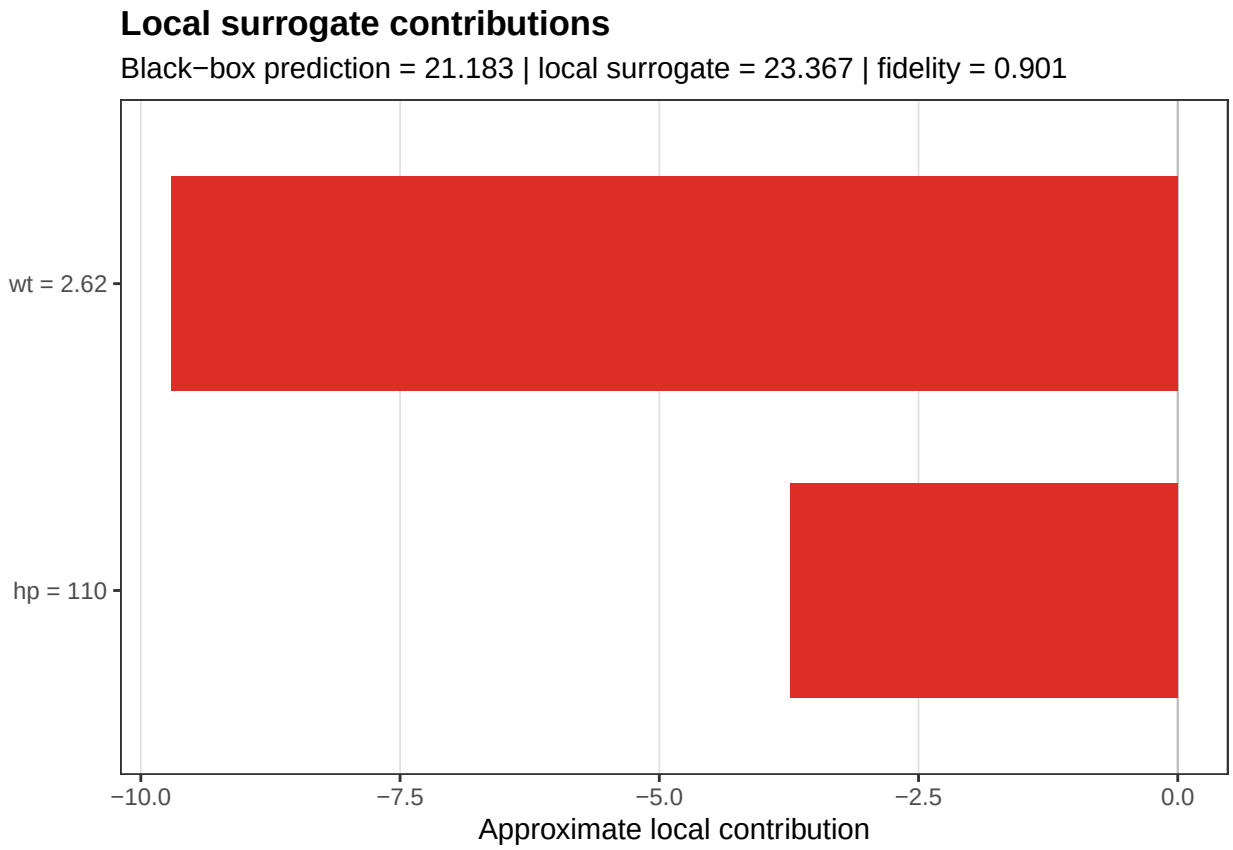
```
plot(pdp_obj)
```


Partial dependence plot
wt
26
24
22
20
18
16
Partial dependence
2
3
4
5
Feature value

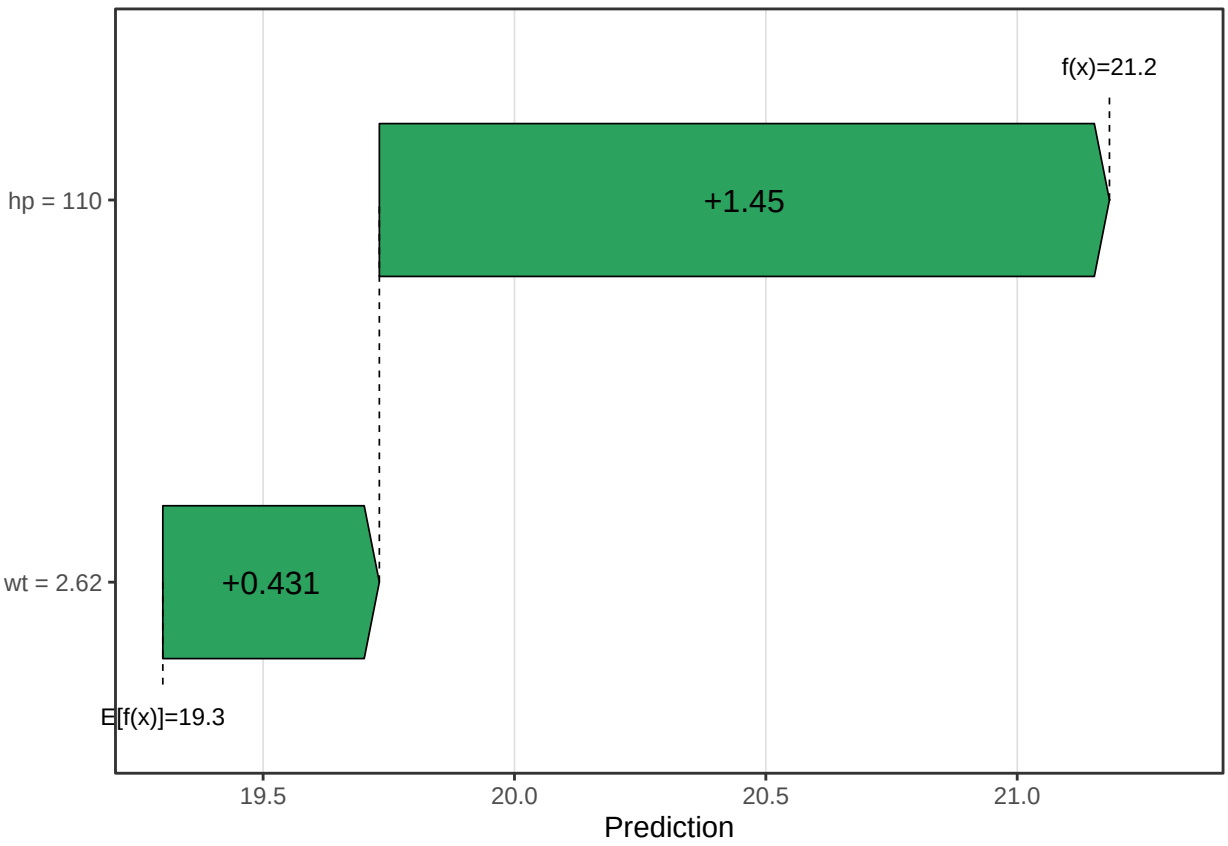
```
plot(ale_obj)
```



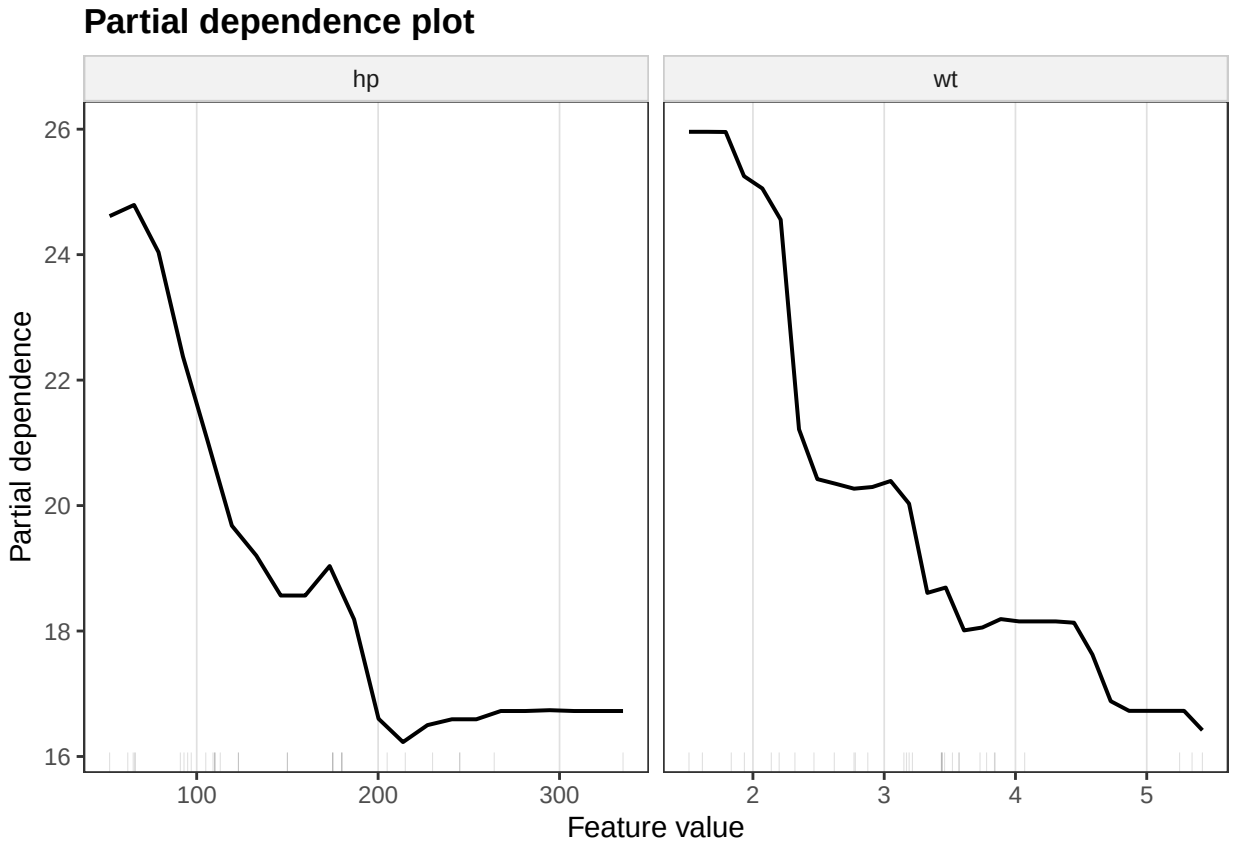

```
plot(local_obj)
```



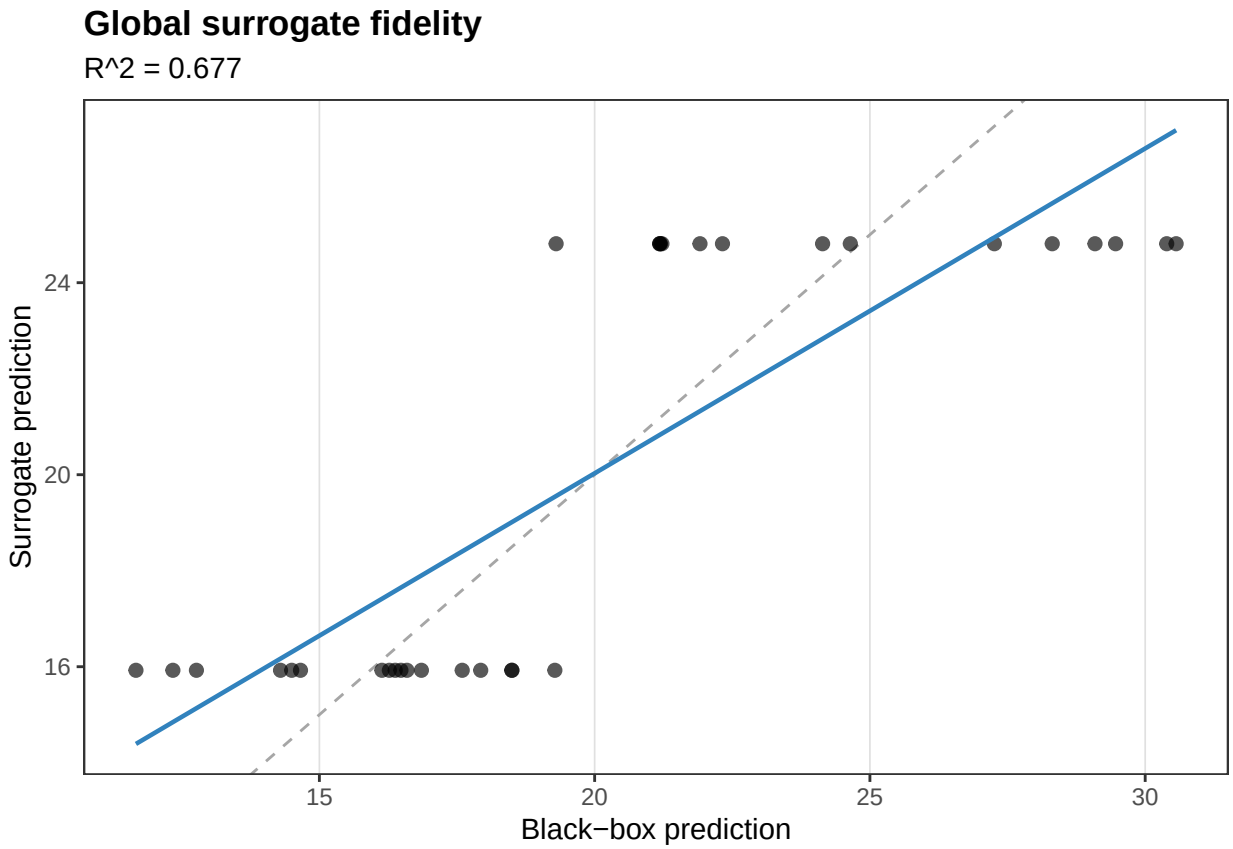

```
plot(shap_obj, kind = "waterfall")
```




```
plot(profile_obj)
```




```
plot(surrogate_obj)
```


Global surrogate fidelity
R^2 = 0.677
Surrogate prediction
24
20
16
15
20
25
30
Black−box prediction