20-模型集成

2023-04-14

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 1.1.0 ──

## ✔ broom 1.0.4 ✔ recipes 1.0.6  
## ✔ dials 1.2.0 ✔ rsample 1.1.1  
## ✔ dplyr 1.1.2 ✔ tibble 3.2.1  
## ✔ ggplot2 3.4.2 ✔ tidyr 1.3.0  
## ✔ infer 1.0.4 ✔ tune 1.1.1  
## ✔ modeldata 1.1.0 ✔ workflows 1.1.3  
## ✔ parsnip 1.1.0 ✔ workflowsets 1.0.1  
## ✔ purrr 1.0.1 ✔ yardstick 1.2.0

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ purrr::discard() masks scales::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ recipes::step() masks stats::step()  
## • Dig deeper into tidy modeling with R at https://www.tmwr.org

library(stacks)  
tidymodels\_prefer()

library(finetune)  
  
library(rules)  
library(baguette)

concrete <-   
 concrete %>%   
 group\_by(across(-compressive\_strength)) %>%   
 summarize(compressive\_strength = mean(compressive\_strength),  
 .groups = "drop")  
nrow(concrete)

## [1] 992

#> [1] 992  
 set.seed(1501)  
concrete\_split <- initial\_split(concrete, strata = compressive\_strength)  
concrete\_train <- training(concrete\_split)  
concrete\_test <- testing(concrete\_split)  
  
set.seed(1502)  
concrete\_folds <-   
 vfold\_cv(concrete\_train, strata = compressive\_strength)

normalized\_rec <-   
 recipe(compressive\_strength ~ ., data = concrete\_train) %>%   
 step\_normalize(all\_predictors())   
  
poly\_recipe <-   
 normalized\_rec %>%   
 step\_poly(all\_predictors()) %>%   
 step\_interact(~ all\_predictors():all\_predictors())  
  
linear\_reg\_spec <-   
 linear\_reg(penalty = tune(), mixture = tune()) %>%   
 set\_engine("glmnet")  
  
nnet\_spec <-   
 mlp(hidden\_units = tune(), penalty = tune(), epochs = tune()) %>%   
 set\_engine("nnet", MaxNWts = 2600) %>%   
 set\_mode("regression")  
  
mars\_spec <-   
 mars(prod\_degree = tune()) %>% #<- use GCV to choose terms  
 set\_engine("earth") %>%   
 set\_mode("regression")  
  
svm\_r\_spec <-   
 svm\_rbf(cost = tune(), rbf\_sigma = tune()) %>%   
 set\_engine("kernlab") %>%   
 set\_mode("regression")  
  
svm\_p\_spec <-   
 svm\_poly(cost = tune(), degree = tune()) %>%   
 set\_engine("kernlab") %>%   
 set\_mode("regression")  
  
knn\_spec <-   
 nearest\_neighbor(neighbors = tune(), dist\_power = tune(), weight\_func = tune()) %>%   
 set\_engine("kknn") %>%   
 set\_mode("regression")  
  
cart\_spec <-   
 decision\_tree(cost\_complexity = tune(), min\_n = tune()) %>%   
 set\_engine("rpart") %>%   
 set\_mode("regression")  
  
bag\_cart\_spec <-   
 bag\_tree() %>%   
 set\_engine("rpart", times = 50L) %>%   
 set\_mode("regression")  
  
rf\_spec <-   
 rand\_forest(mtry = tune(), min\_n = tune(), trees = 1000) %>%   
 set\_engine("ranger") %>%   
 set\_mode("regression")  
  
xgb\_spec <-   
 boost\_tree(tree\_depth = tune(), learn\_rate = tune(), loss\_reduction = tune(),   
 min\_n = tune(), sample\_size = tune(), trees = tune()) %>%   
 set\_engine("xgboost") %>%   
 set\_mode("regression")  
  
cubist\_spec <-   
 cubist\_rules(committees = tune(), neighbors = tune()) %>%   
 set\_engine("Cubist")   
  
nnet\_param <-   
 nnet\_spec %>%   
 extract\_parameter\_set\_dials() %>%   
 update(hidden\_units = hidden\_units(c(1, 27)))  
  
model\_vars <-   
 workflow\_variables(outcomes = compressive\_strength,   
 predictors = everything())  
  
no\_pre\_proc <-   
 workflow\_set(  
 preproc = list(simple = model\_vars),   
 models = list(MARS = mars\_spec, CART = cart\_spec, CART\_bagged = bag\_cart\_spec,  
 RF = rf\_spec, boosting = xgb\_spec, Cubist = cubist\_spec)  
 )  
no\_pre\_proc

## # A workflow set/tibble: 6 × 4  
## wflow\_id info option result   
## <chr> <list> <list> <list>   
## 1 simple\_MARS <tibble [1 × 4]> <opts[0]> <list [0]>  
## 2 simple\_CART <tibble [1 × 4]> <opts[0]> <list [0]>  
## 3 simple\_CART\_bagged <tibble [1 × 4]> <opts[0]> <list [0]>  
## 4 simple\_RF <tibble [1 × 4]> <opts[0]> <list [0]>  
## 5 simple\_boosting <tibble [1 × 4]> <opts[0]> <list [0]>  
## 6 simple\_Cubist <tibble [1 × 4]> <opts[0]> <list [0]>

normalized <-   
 workflow\_set(  
 preproc = list(normalized = normalized\_rec),   
 models = list(SVM\_radial = svm\_r\_spec, SVM\_poly = svm\_p\_spec,   
 KNN = knn\_spec, neural\_network = nnet\_spec)  
 )  
normalized

## # A workflow set/tibble: 4 × 4  
## wflow\_id info option result   
## <chr> <list> <list> <list>   
## 1 normalized\_SVM\_radial <tibble [1 × 4]> <opts[0]> <list [0]>  
## 2 normalized\_SVM\_poly <tibble [1 × 4]> <opts[0]> <list [0]>  
## 3 normalized\_KNN <tibble [1 × 4]> <opts[0]> <list [0]>  
## 4 normalized\_neural\_network <tibble [1 × 4]> <opts[0]> <list [0]>

with\_features <-   
 workflow\_set(  
 preproc = list(full\_quad = poly\_recipe),   
 models = list(linear\_reg = linear\_reg\_spec, KNN = knn\_spec)  
 )  
  
all\_workflows <-   
 bind\_rows(no\_pre\_proc, normalized, with\_features) %>%   
 # Make the workflow ID's a little more simple:   
 mutate(wflow\_id = gsub("(simple\_)|(normalized\_)", "", wflow\_id))  
all\_workflows

## # A workflow set/tibble: 12 × 4  
## wflow\_id info option result   
## <chr> <list> <list> <list>   
## 1 MARS <tibble [1 × 4]> <opts[0]> <list [0]>  
## 2 CART <tibble [1 × 4]> <opts[0]> <list [0]>  
## 3 CART\_bagged <tibble [1 × 4]> <opts[0]> <list [0]>  
## 4 RF <tibble [1 × 4]> <opts[0]> <list [0]>  
## 5 boosting <tibble [1 × 4]> <opts[0]> <list [0]>  
## 6 Cubist <tibble [1 × 4]> <opts[0]> <list [0]>  
## 7 SVM\_radial <tibble [1 × 4]> <opts[0]> <list [0]>  
## 8 SVM\_poly <tibble [1 × 4]> <opts[0]> <list [0]>  
## 9 KNN <tibble [1 × 4]> <opts[0]> <list [0]>  
## 10 neural\_network <tibble [1 × 4]> <opts[0]> <list [0]>  
## 11 full\_quad\_linear\_reg <tibble [1 × 4]> <opts[0]> <list [0]>  
## 12 full\_quad\_KNN <tibble [1 × 4]> <opts[0]> <list [0]>

library(parallel)  
cl.cores <- detectCores()  
makeCluster(getOption("cl.cores", 10))

## socket cluster with 10 nodes on host 'localhost'

#cl <- parallelly::makeClusterPSOCK(12, autoStop = TRUE)  
  
library(finetune)  
  
race\_ctrl <-  
 control\_race(  
 save\_pred = TRUE,  
 parallel\_over = "everything",  
 save\_workflow = TRUE  
 )  
  
race\_results <-  
 all\_workflows %>%  
 workflow\_map(  
 "tune\_race\_anova",  
 seed = 1503,  
 resamples = concrete\_folds,  
 grid = 10,  
 control = race\_ctrl  
 )

## ✖ The workflow requires packages that are not installed: 'kernlab'. Skipping this workflow.  
## ✖ The workflow requires packages that are not installed: 'kernlab'. Skipping this workflow.

## ✖ The workflow requires packages that are not installed: 'kknn'. Skipping this workflow.  
## ✖ The workflow requires packages that are not installed: 'kknn'. Skipping this workflow.

concrete\_stack <-   
 stacks() %>%   
 add\_candidates(race\_results)

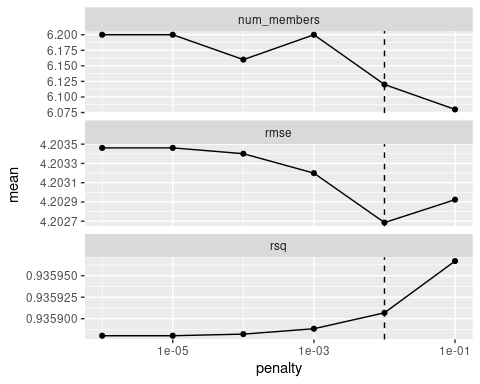
## Warning: ! Some elements of the supplied workflow set failed to evaluate with resamples.  
## ℹ The workflow with ID `SVM\_radial`, `SVM\_poly`, `KNN`, and `full\_quad\_KNN`  
## will be excluded from the data stack.

concrete\_stack

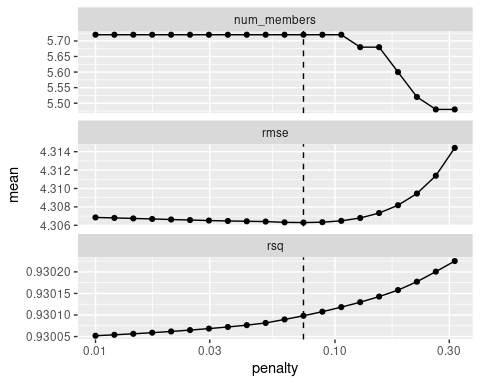
## # A data stack with 8 model definitions and 18 candidate members:  
## # MARS: 1 model configuration  
## # CART: 1 model configuration  
## # CART\_bagged: 1 model configuration  
## # RF: 1 model configuration  
## # boosting: 1 model configuration  
## # Cubist: 3 model configurations  
## # neural\_network: 5 model configurations  
## # full\_quad\_linear\_reg: 5 model configurations  
## # Outcome: compressive\_strength (numeric)

set.seed(2001)  
ens <- blend\_predictions(concrete\_stack)

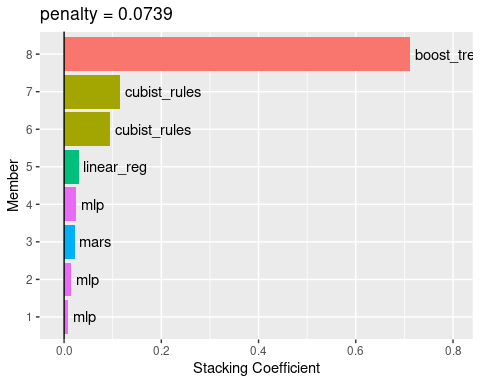
autoplot(ens)



set.seed(2002)  
ens <- blend\_predictions(concrete\_stack, penalty = 10^seq(-2, -0.5, length = 20))  
autoplot(ens)



autoplot(ens, "weights") +  
 geom\_text(aes(x = weight + 0.01, label = model), hjust = 0) +   
 theme(legend.position = "none") +  
 lims(x = c(-0.01, 0.8))



在测试集的评估指标

ens\_fit <- ens %>% fit\_members()  
reg\_metrics <- metric\_set(rmse, rsq)  
ens\_test\_pred <-  
 predict(ens\_fit , concrete\_test) %>%  
 bind\_cols(concrete\_test)  
  
ens\_test\_pred %>%  
 reg\_metrics(compressive\_strength, .pred)

## # A tibble: 2 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 3.29   
## 2 rsq standard 0.957