

Supplementary R Script for Data Processing and Model Comparison

Your Name *

December 7, 2024

1 Introduction

This supplementary document contains the R script used for comparing multiple machine learning algorithms using parallel computation. The script includes libraries for parallel computing, data preprocessing, and model evaluation.

2 R Script

The following R script demonstrates how to load libraries, set up parallel computation, read data, preprocess it, and apply various machine learning algorithms:

```
1 #1.
2 library(tidymodels)
3 library(bonsai)
4 setwd("D:/R work")
5 library(doParallel)
6 cl <- makePSOCKcluster(12)
7 registerDoParallel(cl)
8
9
10
11
12 #2.
13 rawdata <- readr::read_csv(file.choose())
14 colnames(rawdata)
15 for(i in c(2:9)){ rawdata[[i]] <- factor(rawdata[[i]])}
16 skimr::skim(rawdata)
17
18
19
20 #3.
21 yourpositivelevel <- "Yes"
22 yournegativelevel <- "No"
```

*Your Institution, Your Department, Your Country. Email: your.email@example.com.

```

23
24 levels(rawdata$Survival.months)
25 table(rawdata$Survival.months)
26 rawdata$Survival.months <- factor(
27   rawdata$Survival.months,
28   levels = c(yournegativelevel, yourpositivelevel)
29 )
30 levels(rawdata$Survival.months)
31 table(rawdata$Survival.months)
32
33
34
35 #4.
36 set.seed(2023)
37 datasplit <- initial_split(rawdata, prop = 0.7, strata =
   Survival.months)
38 traindata <- training(datasplit)
39 testdata <- testing(datasplit)
40
41
42
43 #5.
44 set.seed(2023)
45 folds <- vfold_cv(traindata, v =10, strata = Survival.months)
46 folds
47
48
49
50 new_dummy_names <- function (var, lvl, ordinal = FALSE) {
51   args <- vctrs::vec_recycle_common(var, lvl)
52   var <- args[[1]]
53   lvl <- args[[2]]
54   nms <- paste(var, lvl, sep = "_")
55   nms
56 }
57
58
59
60 #6.
61
62 datarecipe_dt <- recipe(formula = Survival.months ~ PSA.level
   + Age + M + Radiation + Chemotherapy + Marital.status +
   Income + Surg, traindata)
63
64
65 datarecipe_rf <- recipe(formula = Survival.months ~ PSA.level
   + Age + M + Radiation + Chemotherapy + Marital.status +
   Income + Surg, traindata)

```

```

66
67
68 datarecipe_xgboost <- recipe(Survival.months ~ PSA.level +
  Age + M + Radiation + Chemotherapy + Marital.status +
  Income + Surg, traindata) %>%
69   step_dummy(all_nominal_predictors(), naming = new_dummy_
    names)
70
71 datarecipe_lightgbm <- recipe(Survival.months ~ PSA.level +
  Age + M + Radiation + Chemotherapy + Marital.status +
  Income + Surg, traindata) %>%
72   step_dummy(all_nominal_predictors(), naming = new_dummy_
    names)
73
74
75 datarecipe_enet <- recipe(formula = Survival.months ~ PSA.
  level + Age + M + Radiation + Chemotherapy + Marital.
  status + Income + Surg, traindata) %>%
76   step_dummy(all_nominal_predictors(), naming = new_dummy_
    names) %>%
77   step_normalize(all_predictors())
78
79
80 datarecipe_svm <- recipe(formula = Survival.months ~ PSA.
  level + Age + M + Radiation + Chemotherapy + Marital.
  status + Income + Surg, traindata) %>%
81   step_dummy(all_nominal_predictors(), naming = new_dummy_
    names) %>%
82   step_normalize(all_predictors())
83
84
85 datarecipe_mlp <- recipe(formula = Survival.months ~ PSA.
  level + Age + M + Radiation + Chemotherapy + Marital.
  status + Income + Surg, traindata) %>%
86   step_dummy(all_nominal_predictors(), naming = new_dummy_
    names) %>%
87   step_range(all_predictors())
88
89
90 datarecipe_knn <- recipe(formula = Survival.months ~ PSA.
  level + Age + M + Radiation + Chemotherapy + Marital.
  status + Income + Surg, traindata) %>%
91   step_dummy(all_nominal_predictors(), naming = new_dummy_
    names) %>%
92   step_normalize(all_predictors())
93
94
95 datarecipe_logistic <- recipe(formula = Survival.months ~ PSA

```

```

    .level + Age + M + Radiation + Chemotherapy + Marital.
    status + Income + Surg, traindata)
96
97
98
99 #7.
100 model_dt <- decision_tree(
101   mode = "classification",
102   engine = "rpart",
103   tree_depth = tune(),
104   min_n = tune(),
105   cost_complexity = tune()
106 ) %>%
107   set_args(model=TRUE)
108 model_dt
109
110
111 model_rf <- rand_forest(
112   mode = "classification",
113   engine = "randomForest", # ranger
114   mtry = tune(),
115   trees = tune(),
116   min_n = tune()
117 ) %>%
118   set_args(importance = T)
119 model_rf
120
121
122
123 model_xgboost <- boost_tree(
124   mode = "classification",
125   engine = "xgboost",
126   mtry = tune(),
127   trees = 1000,
128   min_n = tune(),
129   tree_depth = tune(),
130   learn_rate = tune(),
131   loss_reduction = tune(),
132   sample_size = tune(),
133   stop_iter = 25
134 ) %>%
135   set_args(validation = 0.2,
136             event_level = "second")
137 model_xgboost
138
139
140
141

```

```

142
143 model_lightgbm <- boost_tree(
144   mode = "classification",
145   engine = "lightgbm",
146   tree_depth = tune(),
147   trees = tune(),
148   learn_rate = tune(),
149   mtry = tune(),
150   min_n = tune(),
151   loss_reduction = tune()
152 )
153 model_lightgbm
154
155
156
157
158 model_enet <- logistic_reg(
159   mode = "classification",
160   engine = "glmnet",
161   mixture = tune(),
162   penalty = tune()
163 )
164 model_enet
165
166
167
168
169 model_mlp <- mlp(
170   mode = "classification",
171   engine = "nnet",
172   hidden_units = tune(),
173   penalty = tune(),
174   epochs = tune()
175 )
176 model_mlp
177
178
179
180 model_svm <- svm_rbf(
181   mode = "classification",
182   engine = "kernlab",
183   cost = tune(),
184   rbf_sigma = tune()
185 )
186 model_svm
187
188
189

```

```

190
191 model_knn <- nearest_neighbor(
192   mode = "classification",
193   engine = "kkn",
194   neighbors = tune(),
195   weight_func = tune(),
196   dist_power = 2
197 )
198 model_knn
199
200
201
202 model_logistic <- logistic_reg(
203   mode = "classification",
204   engine = "glm"
205 )
206 model_logistic
207
208
209
210
211
212
213 #8.
214 wk_dt <-
215   workflow() %>%
216   add_recipe(datarecipe_dt) %>%
217   add_model(model_dt)
218 wk_dt
219
220
221 wk_rf <-
222   workflow() %>%
223   add_recipe(datarecipe_rf) %>%
224   add_model(model_rf)
225 wk_rf
226
227
228 wk_xgboost <-
229   workflow() %>%
230   add_recipe(datarecipe_xgboost) %>%
231   add_model(model_xgboost)
232 wk_xgboost
233
234 wk_lightgbm <-
235   workflow() %>%
236   add_recipe(datarecipe_lightgbm) %>%
237   add_model(model_lightgbm)

```

```

238 wk_lightgbm
239
240 wk_enet <-
241   workflow() %>%
242   add_recipe(datarecipe_enet) %>%
243   add_model(model_enet)
244 wk_enet
245
246 wk_mlp <-
247   workflow() %>%
248   add_recipe(datarecipe_mlp) %>%
249   add_model(model_mlp)
250 wk_mlp
251
252
253 wk_svm <-
254   workflow() %>%
255   add_recipe(datarecipe_svm) %>%
256   add_model(model_svm)
257 wk_svm
258
259 wk_knn <-
260   workflow() %>%
261   add_recipe(datarecipe_knn) %>%
262   add_model(model_knn)
263 wk_knn
264
265 wk_logistic <-
266   workflow() %>%
267   add_recipe(datarecipe_logistic) %>%
268   add_model(model_logistic)
269 wk_logistic
270
271
272
273
274
275
276
277 #9.
278 set.seed(2023)
279 hpgrid_dt <- parameters(
280   tree_depth(range = c(3, 7)),
281   min_n(range = c(5, 10)),
282   cost_complexity(range = c(-6, -3))
283 ) %>%
284   grid_random(size = 20)
285 hpgrid_dt

```

```

286 log10(hpgrid_dt$cost_complexity)
287
288
289
290
291 set.seed(2023)
292 hpgrid_rf <- parameters(
293   mtry(range = c(2, 10)),
294   trees(range = c(200, 500)),
295   min_n(range = c(20, 50))
296 ) %>%
297   grid_random(size = 20) #
298 hpgrid_rf
299
300
301
302
303
304 set.seed(2023)
305 hpgrid_xgboost <- parameters(
306   mtry(range = c(2, 8)),
307   min_n(range = c(5, 20)),
308   tree_depth(range = c(1, 3)),
309   learn_rate(range = c(0.01, 0.3)),
310   loss_reduction(range = c(-3, 0)),
311   sample_prop(range = c(0.8, 1))
312 ) %>%
313   grid_random(size = 20) #
314 hpgrid_xgboost
315
316
317
318 set.seed(2023)
319 hpgrid_lightgbm <- parameters(
320   tree_depth(range = c(1, 3)),
321   trees(range = c(100, 500)),
322   learn_rate(range = c(-3, -1)),
323   mtry(range = c(2, 8)),
324   min_n(range = c(5, 10)),
325   loss_reduction(range = c(-3, 0))
326 ) %>%
327   grid_random(size = 20) #
328 hpgrid_lightgbm
329
330
331
332
333 set.seed(2023)

```

```

334 hpgrid_enet <- parameters(
335   mixture(),
336   penalty(range = c(-5, 0))
337 ) %>%
338   grid_regular(levels = c(5, 20))
339 hpgrid_enet
340
341
342
343
344
345 set.seed(2023)
346 hpgrid_mlp <- parameters(
347   hidden_units(range = c(15, 24)),
348   penalty(range = c(-3, 0)),
349   epochs(range = c(50, 150))
350 ) %>%
351   grid_regular(levels = 3) #
352 hpgrid_mlp
353
354
355
356
357
358
359
360 set.seed(2023)
361 hpgrid_svm <- parameters(
362   cost(range = c(-5, 5)),
363   rbf_sigma(range = c(-4, -1))
364 ) %>%
365   grid_random(size = 20) #
366 hpgrid_svm
367
368
369
370
371
372 set.seed(2023)
373 hpgrid_knn <- parameters(
374   neighbors(range = c(3, 11)),
375   weight_func()
376 ) %>%
377   grid_random(size = 20) #
378 hpgrid_knn
379
380
381

```

```

382
383
384
385
386 set.seed(2023)
387 tune_dt <- wk_dt %>%
388   tune_grid(
389     resamples = folds,
390     grid = hpgrid_dt,
391     metrics = metric_set(yardstick::accuracy,
392                           yardstick::roc_auc,
393                           yardstick::pr_auc),
394     control = control_grid(save_pred = T,
395                             verbose = T,
396                             event_level = "second",
397                             parallel_over = "everything",
398                             save_workflow = T)
399   )
400
401
402 set.seed(2023)
403 tune_rf <- wk_rf %>%
404   tune_grid(
405     resamples = folds,
406     grid = hpgrid_rf,
407     metrics = metric_set(yardstick::accuracy,
408                           yardstick::roc_auc,
409                           yardstick::pr_auc),
410     control = control_grid(save_pred = T,
411                             verbose = T,
412                             event_level = "second",
413                             parallel_over = "everything",
414                             save_workflow = T)
415   )
416
417
418 set.seed(2023)
419 tune_xgboost <- wk_xgboost %>%
420   tune_grid(
421     resamples = folds,
422     grid = hpgrid_xgboost,
423     metrics = metric_set(yardstick::accuracy,
424                           yardstick::roc_auc,
425                           yardstick::pr_auc),
426     control = control_grid(save_pred = T,
427                             verbose = T,
428                             event_level = "second",
429                             parallel_over = "everything",

```

```

430                                     save_workflow = T)
431   )
432
433
434 set.seed(2023)
435 tune_lightgbm <- wk_lightgbm %>%
436   tune_grid(
437     resamples = folds,
438     grid = hpgrid_lightgbm,
439     metrics = metric_set(yardstick::accuracy,
440                           yardstick::roc_auc,
441                           yardstick::pr_auc),
442     control = control_grid(save_pred = T,
443                             verbose = T,
444                             event_level = "second",
445                             parallel_over = "everything",
446                             save_workflow = T)
447   )
448
449
450 set.seed(2023)
451 tune_enet <- wk_enet %>%
452   tune_grid(
453     resamples = folds,
454     grid = hpgrid_enet,
455     metrics = metric_set(yardstick::accuracy,
456                           yardstick::roc_auc,
457                           yardstick::pr_auc),
458     control = control_grid(save_pred = T,
459                             verbose = T,
460                             event_level = "second",
461                             parallel_over = "everything",
462                             save_workflow = T)
463   )
464
465 set.seed(2023)
466 tune_mlp <- wk_mlp %>%
467   tune_grid(
468     resamples = folds,
469     grid = hpgrid_mlp,
470     metrics = metric_set(yardstick::accuracy,
471                           yardstick::roc_auc,
472                           yardstick::pr_auc),
473     control = control_grid(save_pred = T,
474                             verbose = T,
475                             event_level = "second",
476                             parallel_over = "everything",
477                             save_workflow = T)

```

```

478 )
479
480
481 set.seed(2023)
482 tune_svm <- wk_svm %>%
483   tune_grid(
484     resamples = folds,
485     grid = hpgrid_svm,
486     metrics = metric_set(yardstick::accuracy,
487                           yardstick::roc_auc,
488                           yardstick::pr_auc),
489     control = control_grid(save_pred = T,
490                             verbose = T,
491                             event_level = "second",
492                             parallel_over = "everything",
493                             save_workflow = T)
494   )
495
496
497 set.seed(2023)
498 tune_knn <- wk_knn %>%
499   tune_grid(
500     resamples = folds,
501     grid = hpgrid_knn,
502     metrics = metric_set(yardstick::accuracy,
503                           yardstick::roc_auc,
504                           yardstick::pr_auc),
505     control = control_grid(save_pred = T,
506                             verbose = T,
507                             event_level = "second",
508                             parallel_over = "everything",
509                             save_workflow = T)
510   )
511
512
513 #11.
514
515 eval_tune_dt <- tune_dt %>%
516   collect_metrics()
517 eval_tune_dt
518 hpbest_dt <- tune_dt %>%
519   select_by_one_std_err(metric = "roc_auc", desc(cost_
520     complexity))
521 hpbest_dt
522
523 eval_tune_rf <- tune_rf %>%
524   collect_metrics()

```

```

525 eval_tune_rf
526 hpbest_rf <- tune_rf %>%
527   select_by_one_std_err(metric = "roc_auc", desc(min_n))
528 hpbest_rf
529
530
531 eval_tune_xgboost <- tune_xgboost %>%
532   collect_metrics()
533 eval_tune_xgboost
534 hpbest_xgboost <- tune_xgboost %>%
535   select_by_one_std_err(metric = "roc_auc", desc(min_n))
536 hpbest_xgboost
537
538
539 eval_tune_lightgbm <- tune_lightgbm %>%
540   collect_metrics()
541 eval_tune_lightgbm
542 hpbest_lightgbm <- tune_lightgbm %>%
543   select_by_one_std_err(metric = "roc_auc", desc(min_n))
544 hpbest_lightgbm
545
546
547 eval_tune_enet <- tune_enet %>%
548   collect_metrics()
549 eval_tune_enet
550 hpbest_enet <- tune_enet %>%
551   select_by_one_std_err(metric = "roc_auc", desc(penalty))
552 hpbest_enet
553
554
555 eval_tune_mlp <- tune_mlp %>%
556   collect_metrics()
557 eval_tune_mlp
558 hpbest_mlp <- tune_mlp %>%
559   select_by_one_std_err(metric = "roc_auc", desc(penalty))
560 hpbest_mlp
561
562
563 eval_tune_svm <- tune_svm %>%
564   collect_metrics()
565 eval_tune_svm
566 hpbest_svm <- tune_svm %>%
567   select_best(metric = "roc_auc")
568 hpbest_svm
569
570
571 eval_tune_knn <- tune_knn %>%
572   collect_metrics()

```

```

573 eval_tune_knn
574 hpbest_knn <- tune_knn %>%
575   select_by_one_std_err(metric = "roc_auc", desc(neighbors))
576 hpbest_knn
577
578
579
580 #12.
581 #
582
582 set.seed(2023)
583 final_dt <- wk_dt %>%
584   finalize_workflow(hpbest_dt) %>%
585   fit(traindata)
586 final_dt
587
588
589 set.seed(2023)
590 final_rf <- wk_rf %>%
591   finalize_workflow(hpbest_rf) %>%
592   fit(traindata)
593 final_rf
594
595
596 set.seed(2023)
597 final_xgboost <- wk_xgboost %>%
598   finalize_workflow(hpbest_xgboost) %>%
599   fit(traindata)
600 final_xgboost
601
602
603 set.seed(2023)
604 final_lightgbm <- wk_lightgbm %>%
605   finalize_workflow(hpbest_lightgbm) %>%
606   fit(traindata)
607 final_lightgbm
608
609 set.seed(2023)
610 final_enet <- wk_enet %>%
611   finalize_workflow(hpbest_enet) %>%
612   fit(traindata)
613 final_enet
614
615 set.seed(2023)
616 final_mlp <- wk_mlp %>%
617   finalize_workflow(hpbest_mlp) %>%
618   fit(traindata)

```

```

619 final_mlp
620
621 set.seed(2023)
622 final_svm <- wk_svm %>%
623   finalize_workflow(hpbest_svm) %>%
624   fit(traindata)
625 final_svm
626
627 set.seed(2023)
628 final_knn <- wk_knn %>%
629   finalize_workflow(hpbest_knn) %>%
630   fit(traindata)
631 final_knn
632
633 final_logistic <- wk_logistic %>%
634   fit(traindata)
635 final_logistic
636
637
638
639
640
641 #13
642
643 evalcv_dt <- bestcv4cls2(
644   wkflow = wk_dt,
645   tunerresult = tune_dt,
646   hpbest = hpbest_dt,
647   yname = "Survival.months",
648   modelname = "DT",
649   v = 10,
650   positivelevel = yourpositivelevel
651 )
652 evalcv_dt$cvroc
653 evalcv_dt$cvpr
654 evalcv_dt$evalcv
655
656
657 evalcv_rf <- bestcv4cls2(
658   wkflow = wk_rf,
659   tunerresult = tune_rf,
660   hpbest = hpbest_rf,
661   yname = "Survival.months",
662   modelname = "rf",
663   v = 10,
664   positivelevel = yourpositivelevel
665 )
666 evalcv_rf$cvroc

```

```

667 evalcv_rf$cvpr
668 evalcv_rf$evalcv
669
670
671 evalcv_xgboost <- bestcv4cls2(
672   wkflow = wk_xgboost,
673   tunerresult = tune_xgboost,
674   hpbest = hpbest_xgboost,
675   yname = "Survival.months",
676   modelname = "xgboost",
677   v = 10,
678   positivelevel = yourpositivelevel
679 )
680 evalcv_xgboost$cvroc
681 evalcv_xgboost$cvpr
682 evalcv_xgboost$evalcv
683
684
685 evalcv_lightgbm <- bestcv4cls2(
686   wkflow = wk_lightgbm,
687   tunerresult = tune_lightgbm,
688   hpbest = hpbest_lightgbm,
689   yname = "Survival.months",
690   modelname = "lightgbm",
691   v = 10,
692   positivelevel = yourpositivelevel
693 )
694 evalcv_lightgbm$cvroc
695 evalcv_lightgbm$cvpr
696 evalcv_lightgbm$evalcv
697
698
699
700 evalcv_enet <- bestcv4cls2(
701   wkflow = wk_enet,
702   tunerresult = tune_enet,
703   hpbest = hpbest_enet,
704   yname = "Survival.months",
705   modelname = "enet",
706   v = 10,
707   positivelevel = yourpositivelevel
708 )
709 evalcv_enet$cvroc
710 evalcv_enet$cvpr
711 evalcv_enet$evalcv
712
713
714

```

```

715 evalcv_mlp <- bestcv4cls2(
716   wkflow = wk_mlp,
717   tunerresult = tune_mlp,
718   hpbest = hpbest_mlp,
719   yname = "Survival.months",
720   modelname = "mlp",
721   v = 10,
722   positivelevel = yourpositivelevel
723 )
724 evalcv_mlp$cvroc
725 evalcv_mlp$cvpr
726 evalcv_mlp$evalcv
727
728
729 evalcv_svm <- bestcv4cls2(
730   wkflow = wk_svm,
731   tunerresult = tune_svm,
732   hpbest = hpbest_svm,
733   yname = "Survival.months",
734   modelname = "svm",
735   v = 10,
736   positivelevel = yourpositivelevel
737 )
738 evalcv_svm$cvroc
739 evalcv_svm$cvpr
740 evalcv_svm$evalcv
741
742
743
744 evalcv_knn <- bestcv4cls2(
745   wkflow = wk_knn,
746   tunerresult = tune_knn,
747   hpbest = hpbest_knn,
748   yname = "Survival.months",
749   modelname = "knn",
750   v = 10,
751   positivelevel = yourpositivelevel
752 )
753 evalcv_knn$cvroc
754 evalcv_knn$cvpr
755 evalcv_knn$evalcv
756
757
758 cv_logistic <-
759   wk_logistic %>%
760   fit_resamples(
761     folds,
762     metrics = metric_set(yardstick::accuracy,

```

```

763         yardstick::roc_auc,
764         yardstick::pr_auc),
765     control = control_resamples(save_pred = T,
766                                 verbose = T,
767                                 event_level = "second",
768                                 parallel_over = "everything",
769                                 save_workflow = T)
770 )
771 cv_logistic
772
773
774 evalcv_logistic <- list()
775
776 metrictemp <- metric_set(yardstick::roc_auc, yardstick::pr_
777   auc)
778 evalcv_logistic$evalcv <-
779   collect_predictions(cv_logistic) %>%
780   group_by(id) %>%
781   metrictemp(Survival.months, .pred_Yes, event_level = "
782     second") %>%
783   group_by(.metric) %>%
784   mutate(model = "logistic",
785           mean = mean(.estimate),
786           sd = sd(.estimate)/sqrt(length(folds$splits)))
787 evalcv_logistic$evalcv
788
789 evalcv_logistic$cvroc <-
790   collect_predictions(cv_logistic) %>%
791   group_by(id) %>%
792   roc_curve(Survival.months, .pred_Yes, event_level = "second
793     ") %>%
794   ungroup() %>%
795   left_join(evalcv_logistic$evalcv %>% filter(.metric == "roc
796     _auc"),
797             by = "id") %>%
798   mutate(idAUC = paste(id, " ROCAUC:", round(.estimate, 4)),
799           idAUC = forcats::as_factor(idAUC)) %>%
800   ggplot(aes(x = 1-specificity, y = sensitivity, color =
801     idAUC)) +
802   geom_path(linewidth = 1) +
803   geom_abline(linetype = "dashed") +
804   scale_x_continuous(expand = c(0,0)) +
805   scale_y_continuous(expand = c(0,0)) +
806   labs(color = "") +
807   theme_bw() +
808   theme(legend.position = c(1,0),

```

```

806         legend.justification = c(1,0),
807         legend.background = element_blank(),
808         legend.key = element_blank())
809 evalcv_logistic$cvroc
810
811 evalcv_logistic$cvpr <-
812   collect_predictions(cv_logistic) %>%
813   group_by(id) %>%
814   pr_curve(Survival.months, .pred_Yes, event_level = "second"
815     ) %>%
816   ungroup() %>%
817   left_join(evalcv_logistic$evalcv %>% filter(.metric == "pr_
818     auc"),
819     by = "id") %>%
820   mutate(idAUC = paste(id, " PRAUC:", round(.estimate, 4)),
821     idAUC = forcats::as_factor(idAUC)) %>%
822   ggplot(aes(x = recall, y = precision, color = idAUC)) +
823   geom_path(linewidth = 1) +
824   geom_abline(linetype = "dashed", intercept = 1, slope = -1)
825   +
826   scale_x_continuous(expand = c(0,0)) +
827   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
828   labs(color = "") +
829   theme_bw() +
830   theme(legend.position = c(0,0),
831     legend.justification = c(0,0),
832     legend.background = element_blank(),
833     legend.key = element_blank())
834 evalcv_logistic$cvpr
835
836
837
838
839
840
841
842 #14.
843 predtrain_dt <- valuateBinaryClassificationModel(
844   model = final_dt,
845   dataset = traindata,
846   yname = "Survival.months",
847   modelname = "DT",
848   datasetname = "traindata",
849   cutoff = "yueden",
850   positivelevel = yourpositivelevel,

```

```

851   negativelevel = yournegativelevel
852 )
853 predtrain_dt$prediction
854 predtrain_dt$rocrestult
855 predtrain_dt$rocplot
856 predtrain_dt$prresult
857 predtrain_dt$prplot
858 predtrain_dt$cmresult
859 predtrain_dt$cmplot
860 predtrain_dt$metrics
861 predtrain_dt$diycutoff
862 predtrain_dt$ksplot
863 predtrain_dt$dcaplot
864
865 pROC::auc(predtrain_dt$proc)
866 pROC::ci.auc(predtrain_dt$proc)
867
868
869 predtest_dt <- valuateBinaryClassificationModel(
870   model = final_dt,
871   dataset = testdata,
872   yname = "Survival.months",
873   modelname = "DT",
874   datasetname = "testdata",
875   cutoff = predtrain_dt$diycutoff,
876   positivelevel = yourpositivelevel,
877   negativelevel = yournegativelevel
878 )
879 predtest_dt$prediction
880 predtest_dt$rocrestult
881 predtest_dt$rocplot
882 predtest_dt$prresult
883 predtest_dt$prplot
884 predtest_dt$cmresult
885 predtest_dt$cmplot
886 predtest_dt$metrics
887 predtest_dt$diycutoff
888 predtest_dt$ksplot
889 predtest_dt$dcaplot
890
891 pROC::auc(predtest_dt$proc)
892 pROC::ci.auc(predtest_dt$proc)
893
894
895 predtrain_dt$rocrestult %>%
896   bind_rows(predtest_dt$rocrestult) %>%
897   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
898          dataAUC = forcats::as_factor(dataAUC)) %>%

```

```

899   ggplot(aes(x = 1-specificity,
900             y = sensitivity,
901             color = dataAUC)) +
902   geom_path(linewidth = 1) +
903   geom_abline(linetype = "dashed") +
904   scale_x_continuous(expand = c(0,0)) +
905   scale_y_continuous(expand = c(0,0)) +
906   labs(color = "") +
907   theme_bw() +
908   theme(legend.position = c(1,0),
909         legend.justification = c(1,0),
910         legend.background = element_blank(),
911         legend.key = element_blank())
912
913 predtrain_dt$prresult %>%
914   bind_rows(predtest_dt$prresult) %>%
915   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
916          dataAUC = forcats::as_factor(dataAUC)) %>%
917   ggplot(aes(x = recall,
918             y = precision,
919             color = dataAUC)) +
920   geom_path(linewidth = 1) +
921   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
922   +
923   scale_x_continuous(expand = c(0,0)) +
924   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
925   labs(color = "") +
926   theme_bw() +
927   theme(legend.position = c(0,0),
928         legend.justification = c(0,0),
929         legend.background = element_blank(),
930         legend.key = element_blank())
931
932
933
934
935 predtrain_rf <- valuateBinaryClassificationModel(
936   model = final_rf,
937   dataset = traindata,
938   yname = "Survival.months",
939   modelname = "rf",
940   datasetname = "traindata",
941   cutoff = "yueden",
942   positivelevel = yourpositivelevel,
943   negativelevel = yournegativelevel
944 )
945 predtrain_rf$prediction

```

```

946 predtrain_rf$rocrestult
947 predtrain_rf$rocplot
948 predtrain_rf$prresult
949 predtrain_rf$prplot
950 predtrain_rf$cmresult
951 predtrain_rf$cmplot
952 predtrain_rf$metrics
953 predtrain_rf$diycutoff
954 predtrain_rf$ksplot
955 predtrain_rf$dcaplot
956
957 pROC::auc(predtrain_rf$proc)
958 pROC::ci.auc(predtrain_rf$proc)
959
960
961 predtest_rf <- valuateBinaryClassificationModel(
962   model = final_rf,
963   dataset = testdata,
964   yname = "Survival.months",
965   modelname = "rf",
966   datasetname = "testdata",
967   cutoff = predtrain_rf$diycutoff,
968   positivelevel = yourpositivelevel,
969   negativelevel = yournegativelevel
970 )
971 predtest_rf$prediction
972 predtest_rf$rocrestult
973 predtest_rf$rocplot
974 predtest_rf$prresult
975 predtest_rf$prplot
976 predtest_rf$cmresult
977 predtest_rf$cmplot
978 predtest_rf$metrics
979 predtest_rf$diycutoff
980 predtest_rf$ksplot
981 predtest_rf$dcaplot
982
983 pROC::auc(predtest_rf$proc)
984 pROC::ci.auc(predtest_rf$proc)
985
986
987
988 predtrain_rf$rocrestult %>%
989   bind_rows(predtest_rf$rocrestult) %>%
990   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
991          dataAUC = forcats::as_factor(dataAUC)) %>%
992   ggplot(aes(x = 1-specificity,
993             y = sensitivity,

```

```

994         color = dataAUC)) +
995 geom_path(linewidth = 1) +
996 geom_abline(linetype = "dashed") +
997 scale_x_continuous(expand = c(0,0)) +
998 scale_y_continuous(expand = c(0,0)) +
999 labs(color = "") +
1000 theme_bw() +
1001 theme(legend.position = c(1,0),
1002       legend.justification = c(1,0),
1003       legend.background = element_blank(),
1004       legend.key = element_blank())
1005
1006 predtrain_rf$prresult %>%
1007   bind_rows(predtest_rf$prresult) %>%
1008   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1009          dataAUC = forcats::as_factor(dataAUC)) %>%
1010   ggplot(aes(x = recall,
1011             y = precision,
1012             color = dataAUC)) +
1013   geom_path(linewidth = 1) +
1014   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1015   +
1016   scale_x_continuous(expand = c(0,0)) +
1017   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1018   labs(color = "") +
1019   theme_bw() +
1020   theme(legend.position = c(0,0),
1021         legend.justification = c(0,0),
1022         legend.background = element_blank(),
1023         legend.key = element_blank())
1024
1025
1026
1027
1028 predtrain_xgboost <- valuateBinaryClassificationModel(
1029   model = final_xgboost,
1030   dataset = traindata,
1031   yname = "Survival.months",
1032   modelname = "xgboost",
1033   datasetname = "traindata",
1034   cutoff = "yueden",
1035   positivelevel = yourpositivelevel,
1036   negativelevel = yournegativelevel
1037 )
1038 predtrain_xgboost$prediction
1039 predtrain_xgboost$rocrestult
1040 predtrain_xgboost$rocplot

```

```

1041 predtrain_xgboost$prresult
1042 predtrain_xgboost$prplot
1043 predtrain_xgboost$cmresult
1044 predtrain_xgboost$cmplot
1045 predtrain_xgboost$metrics
1046 predtrain_xgboost$diycutoff
1047 predtrain_xgboost$ksplot
1048 predtrain_xgboost$dcaplot
1049
1050 pROC::auc(predtrain_xgboost$proc)
1051 pROC::ci.auc(predtrain_xgboost$proc)
1052
1053 #
1054 predtest_xgboost <- valuateBinaryClassificationModel(
1055   model = final_xgboost,
1056   dataset = testdata,
1057   yname = "Survival.months",
1058   modelname = "xgboost",
1059   datasetname = "testdata",
1060   cutoff = predtrain_xgboost$diycutoff,
1061   positivelevel = yourpositivelevel,
1062   negativelevel = yournegativelevel
1063 )
1064 predtest_xgboost$prediction
1065 predtest_xgboost$rocresult
1066 predtest_xgboost$rocplot
1067 predtest_xgboost$prresult
1068 predtest_xgboost$prplot
1069 predtest_xgboost$cmresult
1070 predtest_xgboost$cmplot
1071 predtest_xgboost$metrics
1072 predtest_xgboost$diycutoff
1073 predtest_xgboost$ksplot
1074 predtest_xgboost$dcaplot
1075
1076 pROC::auc(predtest_xgboost$proc)
1077 pROC::ci.auc(predtest_xgboost$proc)
1078
1079
1080 #
1081 predtrain_xgboost$rocresult %>%
1082   bind_rows(predtest_xgboost$rocresult) %>%
1083   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1084          dataAUC = forcats::as_factor(dataAUC)) %>%
1085   ggplot(aes(x = 1-specificity,
1086              y = sensitivity,
1087              color = dataAUC)) +
1088   geom_path(linewidth = 1) +

```

```

1089 geom_abline(linetype = "dashed") +
1090 scale_x_continuous(expand = c(0,0)) +
1091 scale_y_continuous(expand = c(0,0)) +
1092 labs(color = "") +
1093 theme_bw() +
1094 theme(legend.position = c(1,0),
1095       legend.justification = c(1,0),
1096       legend.background = element_blank(),
1097       legend.key = element_blank())
1098
1099 predtrain_xgboost$prresult %>%
1100   bind_rows(predtest_xgboost$prresult) %>%
1101   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1102          dataAUC = forcats::as_factor(dataAUC)) %>%
1103   ggplot(aes(x = recall,
1104             y = precision,
1105             color = dataAUC)) +
1106   geom_path(linewidth = 1) +
1107   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1108   +
1109   scale_x_continuous(expand = c(0,0)) +
1110   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1111   labs(color = "") +
1112   theme_bw() +
1113   theme(legend.position = c(0,0),
1114         legend.justification = c(0,0),
1115         legend.background = element_blank(),
1116         legend.key = element_blank())
1117
1118
1119
1120 predtrain_lightgbm <- valuateBinaryClassificationModel(
1121   model = final_lightgbm,
1122   dataset = traindata,
1123   yname = "Survival.months",
1124   modelname = "lightgbm",
1125   datasetname = "traindata",
1126   cutoff = "yueden",
1127   positivelevel = yourpositivelevel,
1128   negativelevel = yournegativelevel
1129 )
1130 predtrain_lightgbm$prediction
1131 predtrain_lightgbm$roccresult
1132 predtrain_lightgbm$rocplot
1133 predtrain_lightgbm$prresult
1134 predtrain_lightgbm$prplot
1135 predtrain_lightgbm$cmresult

```

```

1136 predtrain_lightgbm$cmplot
1137 predtrain_lightgbm$metrics
1138 predtrain_lightgbm$diycutoff
1139 predtrain_lightgbm$ksplot
1140 predtrain_lightgbm$dcaplot
1141
1142 pROC::auc(predtrain_lightgbm$proc)
1143 pROC::ci.auc(predtrain_lightgbm$proc)
1144
1145 #
1146 predtest_lightgbm <- valuateBinaryClassificationModel(
1147   model = final_lightgbm,
1148   dataset = testdata,
1149   yname = "Survival.months",
1150   modelname = "lightgbm",
1151   datasetname = "testdata",
1152   cutoff = predtrain_lightgbm$diycutoff,
1153   positivelevel = yourpositivelevel,
1154   negativelevel = yournegativelevel
1155 )
1156 predtest_lightgbm$prediction
1157 predtest_lightgbm$rocresult
1158 predtest_lightgbm$rocplot
1159 predtest_lightgbm$prresult
1160 predtest_lightgbm$prplot
1161 predtest_lightgbm$cmresult
1162 predtest_lightgbm$cmplot
1163 predtest_lightgbm$metrics
1164 predtest_lightgbm$diycutoff
1165 predtest_lightgbm$ksplot
1166 predtest_lightgbm$dcaplot
1167
1168 pROC::auc(predtest_lightgbm$proc)
1169 pROC::ci.auc(predtest_lightgbm$proc)
1170
1171 #
1172 #
1173 predtrain_lightgbm$rocresult %>%
1174   bind_rows(predtest_lightgbm$rocresult) %>%
1175   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1176          dataAUC = forcats::as_factor(dataAUC)) %>%
1177   ggplot(aes(x = 1-specificity,
1178              y = sensitivity,
1179              color = dataAUC)) +
1180   geom_path(linewidth = 1) +
1181   geom_abline(linetype = "dashed") +
1182   scale_x_continuous(expand = c(0,0)) +
1183   scale_y_continuous(expand = c(0,0)) +

```

```

1184 labs(color = "") +
1185 theme_bw() +
1186 theme(legend.position = c(1,0),
1187       legend.justification = c(1,0),
1188       legend.background = element_blank(),
1189       legend.key = element_blank())
1190
1191 predtrain_lightgbm$prresult %>%
1192   bind_rows(predtest_lightgbm$prresult) %>%
1193   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1194          dataAUC = forcats::as_factor(dataAUC)) %>%
1195   ggplot(aes(x = recall,
1196             y = precision,
1197             color = dataAUC)) +
1198   geom_path(linewidth = 1) +
1199   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1200   +
1201   scale_x_continuous(expand = c(0,0)) +
1202   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1203   labs(color = "") +
1204   theme_bw() +
1205   theme(legend.position = c(0,0),
1206         legend.justification = c(0,0),
1207         legend.background = element_blank(),
1208         legend.key = element_blank())
1209
1210
1211
1212
1213 predtrain_enet <- valuateBinaryClassificationModel(
1214   model = final_enet,
1215   dataset = traindata,
1216   yname = "Survival.months",
1217   modelname = "enet",
1218   datasetname = "traindata",
1219   cutoff = "yueden",
1220   positivelevel = yourpositivelevel,
1221   negativelevel = yournegativelevel
1222 )
1223 predtrain_enet$prediction
1224 predtrain_enet$rocrestult
1225 predtrain_enet$rocplot
1226 predtrain_enet$prresult
1227 predtrain_enet$prplot
1228 predtrain_enet$cmresult
1229 predtrain_enet$cmplot
1230 predtrain_enet$metrics

```

```

1231 predtrain_enet$diycutoff
1232 predtrain_enet$ksplot
1233 predtrain_enet$dcaplot
1234
1235 pROC::auc(predtrain_enet$proc)
1236 pROC::ci.auc(predtrain_enet$proc)
1237 #
1238   dataset = testdata,
1239   yname = "Survival.months",
1240   modelname = "enet",
1241   datasetname = "testdata",
1242   cutoff = predtrain_enet$diycutoff,
1243   positivelevel = yourpositivelevel,
1244   negativelevel = yournegativelevel
1245 )
1246 predtest_enet$prediction
1247 predtest_enet$rocrestult
1248 predtest_enet$rocplot
1249 predtest_enet$prresult
1250 predtest_enet$prplot
1251 predtest_enet$cmresult
1252 predtest_enet$cmplot
1253 predtest_enet$metrics
1254 predtest_enet$diycutoff
1255 predtest_enet$ksplot
1256 predtest_enet$dcaplot
1257
1258 pROC::auc(predtest_enet$proc)
1259 pROC::ci.auc(predtest_enet$proc)
1260
1261
1262 #
1263 predtrain_enet$rocrestult %>%
1264   bind_rows(predtest_enet$rocrestult) %>%
1265   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1266          dataAUC = forcats::as_factor(dataAUC)) %>%
1267   ggplot(aes(x = 1-specificity,
1268             y = sensitivity,
1269             color = dataAUC)) +
1270   geom_path(linewidth = 1) +
1271   geom_abline(linetype = "dashed") +
1272   scale_x_continuous(expand = c(0,0)) +
1273   scale_y_continuous(expand = c(0,0)) +
1274   labs(color = "") +
1275   theme_bw() +
1276   theme(legend.position = c(1,0),
1277         legend.justification = c(1,0),
1278         legend.background = element_blank(),

```

```

1279         legend.key = element_blank())
1280
1281 predtrain_enet$prresult %>%
1282   bind_rows(predtest_enet$prresult) %>%
1283   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1284          dataAUC = forcats::as_factor(dataAUC)) %>%
1285   ggplot(aes(x = recall,
1286              y = precision,
1287              color = dataAUC)) +
1288   geom_path(linewidth = 1) +
1289   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1290   +
1291   scale_x_continuous(expand = c(0,0)) +
1292   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1293   labs(color = "") +
1294   theme_bw() +
1295   theme(legend.position = c(0,0),
1296         legend.justification = c(0,0),
1297         legend.background = element_blank(),
1298         legend.key = element_blank())
1299
1300
1301
1302
1303 predtrain_mlp <- valuateBinaryClassificationModel(
1304   model = final_mlp,
1305   dataset = traindata,
1306   yname = "Survival.months",
1307   modelname = "mlp",
1308   datasetname = "traindata",
1309   cutoff = "yueden",
1310   positivelevel = yourpositivelevel,
1311   negativelevel = yournegativelevel
1312 )
1313 predtrain_mlp$prediction
1314 predtrain_mlp$rocresult
1315 predtrain_mlp$rocplot
1316 predtrain_mlp$prresult
1317 predtrain_mlp$prplot
1318 predtrain_mlp$cmresult
1319 predtrain_mlp$cmplot
1320 predtrain_mlp$metrics
1321 predtrain_mlp$diycutoff
1322 predtrain_mlp$ksplot
1323 predtrain_mlp$dcaplot
1324
1325 pROC::auc(predtrain_mlp$proc)

```

```

1326 pROC::ci.auc(predtrain_mlp$proc)
1327
1328 #
1329 predtest_mlp <- valuateBinaryClassificationModel(
1330   model = final_mlp,
1331   dataset = testdata,
1332   yname = "Survival.months",
1333   modelname = "mlp",
1334   datasetname = "testdata",
1335   cutoff = predtrain_mlp$diycutoff,
1336   positivelevel = yourpositivelevel,
1337   negativelevel = yournegativelevel
1338 )
1339 predtest_mlp$prediction
1340 predtest_mlp$rocresult
1341 predtest_mlp$rocplot
1342 predtest_mlp$prresult
1343 predtest_mlp$prplot
1344 predtest_mlp$cmresult
1345 predtest_mlp$cmplot
1346 predtest_mlp$metrics
1347 predtest_mlp$diycutoff
1348 predtest_mlp$ksplot
1349 predtest_mlp$dcaplot
1350
1351 pROC::auc(predtest_mlp$proc)
1352 pROC::ci.auc(predtest_mlp$proc)
1353
1354
1355 #
1356 predtrain_mlp$rocresult %>%
1357   bind_rows(predtest_mlp$rocresult) %>%
1358   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1359          dataAUC = forcats::as_factor(dataAUC)) %>%
1360   ggplot(aes(x = 1-specificity,
1361              y = sensitivity,
1362              color = dataAUC)) +
1363   geom_path(linewidth = 1) +
1364   geom_abline(linetype = "dashed") +
1365   scale_x_continuous(expand = c(0,0)) +
1366   scale_y_continuous(expand = c(0,0)) +
1367   labs(color = "") +
1368   theme_bw() +
1369   theme(legend.position = c(1,0),
1370         legend.justification = c(1,0),
1371         legend.background = element_blank(),
1372         legend.key = element_blank())
1373

```

```

1374 predtrain_mlp$prresult %>%
1375   bind_rows(predtest_mlp$prresult) %>%
1376   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1377          dataAUC = forcats::as_factor(dataAUC)) %>%
1378   ggplot(aes(x = recall,
1379              y = precision,
1380              color = dataAUC)) +
1381   geom_path(linewidth = 1) +
1382   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1383   +
1384   scale_x_continuous(expand = c(0,0)) +
1385   scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1386   labs(color = "") +
1387   theme_bw() +
1388   theme(legend.position = c(0,0),
1389         legend.justification = c(0,0),
1390         legend.background = element_blank(),
1391         legend.key = element_blank())
1392
1393
1394
1395 predtrain_svm <- valuateBinaryClassificationModel(
1396   model = final_svm,
1397   dataset = traindata,
1398   yname = "Survival.months",
1399   modelname = "svm",
1400   datasetname = "traindata",
1401   cutoff = "yueden",
1402   positivelevel = yourpositivelevel,
1403   negativelevel = yournegativelevel
1404 )
1405 predtrain_svm$prediction
1406 predtrain_svm$rocresult
1407 predtrain_svm$rocplot
1408 predtrain_svm$prresult
1409 predtrain_svm$prplot
1410 predtrain_svm$cmresult
1411 predtrain_svm$cmplot
1412 predtrain_svm$metrics
1413 predtrain_svm$diycutoff
1414 predtrain_svm$ksplot
1415 predtrain_svm$dcaplot
1416
1417 pROC::auc(predtrain_svm$proc)
1418 pROC::ci.auc(predtrain_svm$proc)
1419
1420 #

```

```

1421 predtest_svm <- valuateBinaryClassificationModel(
1422   model = final_svm,
1423   dataset = testdata,
1424   yname = "Survival.months",
1425   modelname = "svm",
1426   datasetname = "testdata",
1427   cutoff = predtrain_svm$diycutoff,
1428   positivelevel = yourpositivelevel,
1429   negativelevel = yournegativelevel
1430 )
1431 predtest_svm$prediction
1432 predtest_svm$rocresult
1433 predtest_svm$rocplot
1434 predtest_svm$prresult
1435 predtest_svm$prplot
1436 predtest_svm$cmresult
1437 predtest_svm$cmplot
1438 predtest_svm$metrics
1439 predtest_svm$diycutoff
1440 predtest_svm$ksplot
1441 predtest_svm$dcaplot
1442
1443 pROC::auc(predtest_svm$proc)
1444 pROC::ci.auc(predtest_svm$proc)
1445
1446
1447 #
1448 predtrain_svm$rocresult %>%
1449   bind_rows(predtest_svm$rocresult) %>%
1450   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1451          dataAUC = forcats::as_factor(dataAUC)) %>%
1452   ggplot(aes(x = 1-specificity,
1453             y = sensitivity,
1454             color = dataAUC)) +
1455   geom_path(linewidth = 1) +
1456   geom_abline(linetype = "dashed") +
1457   scale_x_continuous(expand = c(0,0)) +
1458   scale_y_continuous(expand = c(0,0)) +
1459   labs(color = "") +
1460   theme_bw() +
1461   theme(legend.position = c(1,0),
1462         legend.justification = c(1,0),
1463         legend.background = element_blank(),
1464         legend.key = element_blank())
1465
1466 predtrain_svm$prresult %>%
1467   bind_rows(predtest_svm$prresult) %>%
1468   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),

```

```

1469         dataAUC = forcats::as_factor(dataAUC)) %>%
1470     ggplot(aes(x = recall,
1471               y = precision,
1472               color = dataAUC)) +
1473     geom_path(linewidth = 1) +
1474     geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1475     +
1476     scale_x_continuous(expand = c(0,0)) +
1477     scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1478     labs(color = "") +
1479     theme_bw() +
1480     theme(legend.position = c(0,0),
1481           legend.justification = c(0,0),
1482           legend.background = element_blank(),
1483           legend.key = element_blank())
1484
1485
1486 predtrain_knn <- valuateBinaryClassificationModel(
1487   model = final_knn,
1488   dataset = traindata,
1489   yname = "Survival.months",
1490   modelname = "knn",
1491   datasetname = "traindata",
1492   cutoff = "yueden",
1493   positivelevel = yourpositivelevel,
1494   negativelevel = yournegativelevel
1495 )
1496 predtrain_knn$prediction
1497 predtrain_knn$rocresult
1498 predtrain_knn$rocplot
1499 predtrain_knn$prresult
1500 predtrain_knn$prplot
1501 predtrain_knn$cmresult
1502 predtrain_knn$cmplot
1503 predtrain_knn$metrics
1504 predtrain_knn$diycutoff
1505 predtrain_knn$ksplot
1506 predtrain_knn$dcaplot
1507
1508 pROC::auc(predtrain_knn$proc)
1509 pROC::ci.auc(predtrain_knn$proc)
1510
1511 # valuateBinaryClassificationModel(
1512   model = final_knn,
1513   dataset = testdata,
1514   yname = "Survival.months",
1515   modelname = "knn",

```

```

1516 datasetname = "testdata",
1517 cutoff = predtrain_knn$diycutoff,
1518 positivelevel = yourpositivelevel,
1519 negativelevel = yournegativelevel
1520 )
1521 predtest_knn$prediction
1522 predtest_knn$rocresult
1523 predtest_knn$rocplot
1524 predtest_knn$prresult
1525 predtest_knn$prplot
1526 predtest_knn$cmresult
1527 predtest_knn$cmplot
1528 predtest_knn$metrics
1529 predtest_knn$diycutoff
1530 predtest_knn$ksplot
1531 predtest_knn$dcaplot
1532
1533 pROC::auc(predtest_knn$proc)
1534 pROC::ci.auc(predtest_knn$proc)
1535
1536
1537 #
1538 predtrain_knn$rocresult %>%
1539   bind_rows(predtest_knn$rocresult) %>%
1540   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1541          dataAUC = forcats::as_factor(dataAUC)) %>%
1542   ggplot(aes(x = 1-specificity,
1543              y = sensitivity,
1544              color = dataAUC)) +
1545   geom_path(linewidth = 1) +
1546   geom_abline(linetype = "dashed") +
1547   scale_x_continuous(expand = c(0,0)) +
1548   scale_y_continuous(expand = c(0,0)) +
1549   labs(color = "") +
1550   theme_bw() +
1551   theme(legend.position = c(1,0),
1552         legend.justification = c(1,0),
1553         legend.background = element_blank(),
1554         legend.key = element_blank())
1555
1556 predtrain_knn$prresult %>%
1557   bind_rows(predtest_knn$prresult) %>%
1558   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1559          dataAUC = forcats::as_factor(dataAUC)) %>%
1560   ggplot(aes(x = recall,
1561              y = precision,
1562              color = dataAUC)) +
1563   geom_path(linewidth = 1) +

```

```

1564 geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1565 +
1566 scale_x_continuous(expand = c(0,0)) +
1567 scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1568 labs(color = "") +
1569 theme_bw() +
1570 theme(legend.position = c(0,0),
1571       legend.justification = c(0,0),
1572       legend.background = element_blank(),
1573       legend.key = element_blank())
1574
1575
1576 predtrain_logistic <- valuateBinaryClassificationModel(
1577   model = final_logistic,
1578   dataset = traindata,
1579   yname = "Survival.months",
1580   modelname = "Logistic",
1581   datasetname = "traindata",
1582   cutoff = "yueden",
1583   positivelevel = yourpositivelevel,
1584   negativelevel = yournegativelevel
1585 )
1586 predtrain_logistic$prediction
1587 predtrain_logistic$predprobplot
1588 predtrain_logistic$rocresult
1589 predtrain_logistic$rocplot
1590 predtrain_logistic$prresult
1591 predtrain_logistic$prplot
1592 predtrain_logistic$cmresult
1593 predtrain_logistic$cmplot
1594 predtrain_logistic$metrics
1595 predtrain_logistic$diycutoff
1596 predtrain_logistic$ksplot
1597 predtrain_logistic$dcaplot
1598
1599 pROC::auc(predtrain_logistic$proc)
1600 pROC::ci.auc(predtrain_logistic$proc)
1601
1602 #
1603 predtest_logistic <- valuateBinaryClassificationModel(
1604   model = final_logistic,
1605   dataset = testdata,
1606   yname = "Survival.months",
1607   modelname = "Logistic",
1608   datasetname = "testdata",
1609   cutoff = predtrain_logistic$diycutoff,
1610   positivelevel = yourpositivelevel,

```

```

1611     negativelevel = yournegativelevel
1612 )
1613 predtest_logistic$prediction
1614 predtest_logistic$predprobplot
1615 predtest_logistic$rocresult
1616 predtest_logistic$rocplot
1617 predtest_logistic$prresult
1618 predtest_logistic$prplot
1619 predtest_logistic$cmresult
1620 predtest_logistic$cmplot
1621 predtest_logistic$metrics
1622 predtest_logistic$diycutoff
1623 predtest_logistic$ksplot
1624 predtest_logistic$dcaplot
1625
1626 pROC::auc(predtest_logistic$proc)
1627 pROC::ci.auc(predtest_logistic$proc)
1628 pROC::roc.test(predtrain_logistic$proc, predtest_logistic$
      proc)
1629
1630 #
1631 predtrain_logistic$rocresult %>%
1632   bind_rows(predtest_logistic$rocresult) %>%
1633   mutate(dataAUC = paste(data, " ROCAUC:", round(ROCAUC, 4)),
1634          dataAUC = forcats::as_factor(dataAUC)) %>%
1635   ggplot(aes(x = 1-specificity,
1636              y = sensitivity,
1637              color = dataAUC)) +
1638   geom_path(linewidth = 1) +
1639   geom_abline(linetype = "dashed") +
1640   scale_x_continuous(expand = c(0,0)) +
1641   scale_y_continuous(expand = c(0,0)) +
1642   labs(color = "") +
1643   theme_bw() +
1644   theme(legend.position = c(1,0),
1645         legend.justification = c(1,0),
1646         legend.background = element_blank(),
1647         legend.key = element_blank())
1648
1649 predtrain_logistic$prresult %>%
1650   bind_rows(predtest_logistic$prresult) %>%
1651   mutate(dataAUC = paste(data, " PRAUC:", round(PRAUC, 4)),
1652          dataAUC = forcats::as_factor(dataAUC)) %>%
1653   ggplot(aes(x = recall,
1654              y = precision,
1655              color = dataAUC)) +
1656   geom_path(linewidth = 1) +
1657   geom_abline(linetype = "dashed", slope = -1, intercept = 1)

```

```

+
1658 scale_x_continuous(expand = c(0,0)) +
1659 scale_y_continuous(expand = c(0,0), limits = c(0, 1)) +
1660 labs(color = "") +
1661 theme_bw() +
1662 theme(legend.position = c(0,0),
1663       legend.justification = c(0,0),
1664       legend.background = element_blank(),
1665       legend.key = element_blank())
1666 #
1667 predtrain_logistic$metrics %>%
1668   bind_rows(predtest_logistic$metrics) %>%
1669   dplyr::select(-.estimator) %>%
1670   pivot_wider(names_from = .metric, values_from = .estimate)
1671
1672
1673
1674
1675
1676
1677 #15
1678 save(datarecipe_dt,
1679       model_dt,
1680       wk_dt,
1681       hpgrid_dt,
1682       tune_dt,
1683       predtrain_dt,
1684       predtest_dt,
1685       evalcv_dt,
1686       file = ".\\MLresult\\MLdt.RData")
1687
1688
1689 save(datarecipe_rf,
1690       model_rf,
1691       wk_rf,
1692       hpgrid_rf,
1693       tune_rf,
1694       predtrain_rf,
1695       predtest_rf,
1696       evalcv_rf,
1697       file = ".\\MLresult\\MLrf.RData")
1698
1699
1700 save(datarecipe_xgboost,
1701       model_xgboost,
1702       wk_xgboost,
1703       hpgrid_xgboost,
1704       tune_xgboost,

```

```

1705     predtrain_xgboost,
1706     predtest_xgboost,
1707     evalcv_xgboost,
1708     file = ".\\MLresult\\MLxgboost.RData")
1709
1710
1711 save(datarecipe_enet,
1712      model_enet,
1713      wk_enet,
1714      hpgrid_enet,
1715      tune_enet,
1716      predtrain_enet,
1717      predtest_enet,
1718      evalcv_enet,
1719      file = ".\\MLresult\\MLenet.RData")
1720
1721
1722 save(datarecipe_svm,
1723      model_svm,
1724      wk_svm,
1725      hpgrid_svm,
1726      tune_svm,
1727      predtrain_svm,
1728      predtest_svm,
1729      evalcv_svm,
1730      file = ".\\MLresult\\MLsvm.RData")
1731
1732
1733 save(datarecipe_mlp,
1734      model_mlp,
1735      wk_mlp,
1736      hpgrid_mlp,
1737      tune_mlp,
1738      predtrain_mlp,
1739      predtest_mlp,
1740      evalcv_mlp,
1741      file = ".\\MLresult\\MLmlp.RData")
1742
1743
1744
1745 save(datarecipe_lightgbm,
1746      model_lightgbm,
1747      wk_lightgbm,
1748      hpgrid_lightgbm,
1749      tune_lightgbm,
1750      predtrain_lightgbm,
1751      predtest_lightgbm,
1752      evalcv_lightgbm,

```

```

1753     file = ".\\MLresult\\MLlightgbm.RData")
1754
1755
1756
1757 save(datarecipe_knn,
1758       model_knn,
1759       wk_knn,
1760       hpgrid_knn,
1761       tune_knn,
1762       predtrain_knn,
1763       predtest_knn,
1764       evalcv_knn,
1765       file = ".\\MLresult\\MLknn.RData")
1766
1767
1768 save(datarecipe_logistic,
1769       model_logistic,
1770       wk_logistic,
1771       cv_logistic,
1772       predtrain_logistic,
1773       predtest_logistic,
1774       evalcv_logistic,
1775       file = ".\\MLresult\\MLlogistic.RData")
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793 #16
1794 #
1795 evalfiles <- list.files(".\\MLresult\\", full.names = T)
1796 lapply(evalfiles, load, .GlobalEnv)
1797
1798 #
1799 nmodels <- 9
1800

```

```

1801 #
1802 cols4model <- c("#E41A1C", #
1803                 "#377EB8", #
1804                 "#4DAF4A", #
1805                 "#984EA3", #
1806                 "#FF7F00", #
1807                 "#FFFF33", #
1808                 "#A65628", #
1809                 "#F781BF", #
1810                 "#999999") #
1811
1812
1813 #
1814 predtest_dt$metrics
1815 eval <- bind_rows(
1816   lapply(list(predtest_logistic, predtest_dt, predtest_enet,
1817               predtest_knn, predtest_lightgbm, predtest_rf,
1818               predtest_xgboost, predtest_svm, predtest_mlp),
1819           "[[",
1820           "metrics")
1821 ) %>%
1822   mutate(model = forcats::as_factor(model))
1823 eval
1824
1825 #
1826 eval2 <- eval %>%
1827   dplyr::select(-.estimator) %>%
1828   pivot_wider(names_from = .metric, values_from = .estimate)
1829 eval2
1830
1831 #
1832 predtest <- bind_rows(
1833   lapply(list(predtest_logistic, predtest_dt, predtest_enet,
1834               predtest_knn, predtest_lightgbm, predtest_rf,
1835               predtest_xgboost, predtest_svm, predtest_mlp),
1836           "[[",
1837           "prediction")
1838 ) %>%
1839   mutate(model = forcats::as_factor(model))
1840 predtest
1841
1842 # ROC
1843 predtest %>%
1844   group_by(model) %>%
1845   roc_curve(.obs, .pred_Yes, event_level = "second") %>%
1846   left_join(eval2[, c("model", "roc_auc")]) %>%
1847   mutate(modelauc = paste0(model,
1848                             ", ROCAUC=", round(roc_auc, 4)),

```

```

1849     modelauc = forcats::as_factor(modelauc)) %>%
1850 ggplot(aes(x = 1-specificity, y = sensitivity, color =
      modelauc)) +
1851 geom_path(linewidth = 1) +
1852 geom_abline(linetype = "dashed") +
1853 scale_color_manual(values = cols4model) +
1854 scale_x_continuous(limits = c(0, 1), expand = c(0, 0)) +
1855 scale_y_continuous(limits = c(0, 1), expand = c(0, 0)) +
1856 labs(color = "", title = paste0("ROCs on testdata")) +
1857 theme_bw() +
1858 theme(legend.position = c(1,0),
1859       legend.justification = c(1,0),
1860       legend.background = element_blank(),
1861       legend.key = element_blank())
1862
1863 # PRAUC
1864 predtest %>%
1865   group_by(model) %>%
1866   pr_curve(.obs, .pred_Yes, event_level = "second") %>%
1867   left_join(eval2[, c("model", "pr_auc")]) %>%
1868   mutate(modelauc = paste0(model,
1869                             ", PRAUC=", round(pr_auc, 4)),
1870          modelauc = forcats::as_factor(modelauc)) %>%
1871   ggplot(aes(x = recall, y = precision, color = modelauc)) +
1872   geom_path(linewidth = 1) +
1873   geom_abline(linetype = "dashed", slope = -1, intercept = 1)
1874   +
1875   scale_color_manual(values = cols4model) +
1876   scale_x_continuous(limits = c(0, 1), expand = c(0, 0)) +
1877   scale_y_continuous(limits = c(0, 1), expand = c(0, 0)) +
1878   labs(color = "", title = paste0("PRs on testdata")) +
1879   theme_bw() +
1880   theme(legend.position = c(0,0),
1881         legend.justification = c(0,0),
1882         legend.background = element_blank(),
1883         legend.key = element_blank())
1884 #####
1885
1886 #
1887 predtest2 <- predtest %>%
1888   dplyr::select(-.pred_No) %>%
1889   mutate(id = rep(1:nrow(predtest_logistic$prediction),
1890                  length(unique(predtest$model)))) %>%
1891   pivot_wider(id_cols = c(id, .obs),
1892               names_from = model,
1893               values_from = .pred_Yes) %>%
1894   dplyr::select(id, .obs, sort(unique(predtest$model)))

```

```

1895 predtest2
1896
1897 #####
1898
1899
1900 #
1901 library(probably)
1902
1903 #
1904 predtest %>%
1905   cal_plot_breaks(.obs, .pred_Yes, event_level = "second",
1906     num_breaks = 5, .by = model) +
1907   scale_color_manual(values = cols4model) +
1908   theme_bw() +
1909   theme(legend.position = "none")
1910
1911 #
1912 predtest %>%
1913   cal_plot_windowed(.obs, .pred_Yes, event_level = "second",
1914     window_size = 0.5, .by = model) +
1915   scale_color_manual(values = cols4model) +
1916   theme_bw() +
1917   theme(legend.position = "none")
1918
1919 #
1920 bs <- predtest %>%
1921   group_by(model) %>%
1922   yardstick::brier_class(.obs, .pred_No) %>%
1923   mutate(meanpred = 0.8, meanobs = 0.25, text = paste0("BS: ",
1924     , round(.estimate, 3)))
1925
1926 #
1927 predtest %>%
1928   cal_plot_windowed(.obs, .pred_Yes, event_level = "second",
1929     window_size = 0.5, .by = model) +
1930   geom_text(bs, mapping = aes(x = meanpred, y = meanobs,
1931     label = text)) +
1932   scale_color_manual(values = cols4model) +
1933   theme_bw() +
1934   theme(legend.position = "none")
1935
1936 # DCA
1937 dca_obj <- dcurves::dca(as.formula(
1938   paste0(".obs ~ ",
1939     paste(colnames(predtest2)[3:ncol(predtest2)],
1940       collapse = " + "))

```

```

1938 ),
1939 data = predtest2,
1940 thresholds = seq(0, 1, by = 0.01)
1941 )
1942 plot(dca_obj, smooth = T, span = 0.5) +
1943   scale_color_manual(values = c("black", "grey", cols4model))
1944   +
1945   labs(title = "DCA on testdata")
1946 #####
1947
1948 #
1949 evalcv <- bind_rows(
1950   lapply(list(evalcv_logistic, evalcv_dt, evalcv_enet,
1951             evalcv_knn, evalcv_lightgbm, evalcv_rf,
1952             evalcv_xgboost, evalcv_svm, evalcv_mlp),
1953           "[",
1954           "evalcv")
1955 ) %>%
1956   mutate(
1957     model = forcats::as_factor(model),
1958     modelperf = paste0(model, "(", round(mean, 2), " ",
1959                        round(sd, 2), ")")
1960   )
1961 evalcv
1962
1963 # ROC
1964 evalcvroc_max <- evalcv %>%
1965   filter(.metric == "roc_auc") %>%
1966   group_by(id) %>%
1967   slice_max(.estimate)
1968 evalcvroc_min <- evalcv %>%
1969   filter(.metric == "roc_auc") %>%
1970   group_by(id) %>%
1971   slice_min(.estimate)
1972 evalcv %>%
1973   filter(.metric == "roc_auc") %>%
1974   ggplot(aes(x = id, y = .estimate,
1975             group = modelperf, color = modelperf)) +
1976   geom_point() +
1977   geom_line() +
1978   ggrepel::geom_text_repel(evalcvroc_max,
1979                             mapping = aes(label = model),
1980                             nudge_y = 0.01,
1981                             show.legend = F) +
1982   ggrepel::geom_text_repel(evalcvroc_min,
1983                             mapping = aes(label = model),
1984                             nudge_y = -0.01,

```

```

1985         show.legend = F) +
1986     scale_y_continuous(limits = c(0.5, 1)) +
1987     scale_color_manual(values = cols4model) +
1988     labs(x = "", y = "ROCAUC", color = "Model") +
1989     theme_bw()
1990
1991 # PR
1992 evalcvpr_max <- evalcv %>%
1993   filter(.metric == "pr_auc") %>%
1994   group_by(id) %>%
1995   slice_max(.estimate)
1996 evalcvpr_min <- evalcv %>%
1997   filter(.metric == "pr_auc") %>%
1998   group_by(id) %>%
1999   slice_min(.estimate)
2000 evalcv %>%
2001   filter(.metric == "pr_auc") %>%
2002   ggplot(aes(x = id, y = .estimate,
2003             group = modelperf, color = modelperf)) +
2004   geom_point() +
2005   geom_line() +
2006   ggrepel::geom_text_repel(evalcvpr_max,
2007                             mapping = aes(label = model),
2008                             nudge_y = 0.01,
2009                             show.legend = F) +
2010   ggrepel::geom_text_repel(evalcvpr_min,
2011                             mapping = aes(label = model),
2012                             nudge_y = -0.01,
2013                             show.legend = F) +
2014   scale_y_continuous(limits = c(0.5, 1)) +
2015   scale_color_manual(values = cols4model) +
2016   labs(x = "", y = "prAUC", color = "Model") +
2017   theme_bw()
2018
2019 #
2020 # ROC
2021 evalcv %>%
2022   filter(.metric == "roc_auc") %>%
2023   group_by(model) %>%
2024   sample_n(size = 1) %>%
2025   ungroup() %>%
2026   ggplot(aes(x = model, y = mean, color = model)) +
2027   geom_point(size = 2, show.legend = F) +
2028   # geom_line(group = 1) +
2029   geom_errorbar(aes(ymin = mean-sd,
2030                    ymax = mean+sd),
2031                 width = 0.1,
2032                 linewidth = 1.2,

```

```

2033         show.legend = F) +
2034     scale_y_continuous(limits = c(0.7, 1)) +
2035     scale_color_manual(values = cols4model) +
2036     labs(y = "cv roc_auc") +
2037     theme_bw()
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047 source("shap.R")
2048
2049
2050 colnames(traindata)
2051
2052 traindatax <- traindata %>%
2053     dplyr::select(-Survival.months)
2054
2055 colnames(traindatax)
2056
2057
2058 catvars <- colnames(traindatax)[sapply(traindatax, is.factor)
2059     ]
2060
2061 catvars
2062
2063
2064
2065 convars <- setdiff(colnames(traindatax), catvars)
2066
2067 convars
2068
2069
2070 shapresult <- generateShapVisualizations(
2071     finalmodel = final_xgboost,
2072     predfunc = function(model, newdata) {
2073         predict(model, newdata, type = "prob") %>%
2074         dplyr::select(ends_with(yourpositivelevel)) %>%
2075         pull()
2076     },
2077     datax = traindatax,
2078     datay = traindata$Survival.months,

```

```

2080 yname = "Survival.months",
2081 flname = catvars,
2082 lxname = convars
2083 )
2084
2085 # SHAP
2086 shapresult$shapvip
2087 #
2088 shapresult$shapvipplot
2089
2090 #
2091                                     shapvizSHAP
2092 shapley <- shapviz::shapviz(
2093   shapresult$shapley, # S H A P
2094   X = traindatax, #
2095   baseline = mean(predtrain_xgboost$prediction$.pred_Yes) # Yes
2096 )
2097 # 1force
2098 shapviz::sv_force(shapley, row_id = 1)
2099 # 1waterfall
2100 shapviz::sv_waterfall(shapley, row_id = 1)
2101
2102 # SHAP
2103 shapresult$shapplotd_facet
2104 shapresult$shapplotd_one
2105
2106 # SHAP
2107 shapresult$shapplotc_facet
2108 shapresult$shapplotc_one
2109 shapresult$shapplotc_one2

```

Listing 1: R Script for Model Comparison and Parallel Computation

3 Conclusion

This document provides an R script suitable for comparing multiple machine learning models with parallel computation. The code is ready to be used for model evaluation and preprocessing tasks.