PD2

Dominik Rafacz

28.04.2020

Preprocessing

First, I prepared data analogously like in the first homework with one difference – I decided to keep `referral_source` column. However, I used one-hot-encoding on it.

Basic model

Then I trained basic xgboost model.

```
task_basic <- TaskClassif$new("sick", dat[indices_train, ], "sick", "sick")

set.seed(1998)
learner_xgboost <- lrn("classif.xgboost")
learner_xgboost$predict_type = "prob"

resampling_outer <- rsmp("cv", folds = 5)
measures <- list(msr("classif.auc"), msr("classif.auprc"))
```

```
result <- resample(task = task_basic,
                   learner = learner_xgboost,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9557524	0.875612

As we can see, it achieved relatively good result in terms of both AUC and AUPRC. I decided to train another xgboost model, this time setting up some hyperparameters values suggested for imbalanced datasets.

```
set.seed(1998)
learner_xgboost_suggested <- lrn("classif.xgboost",
                                 min_child_weight = 1,
                                 scale_pos_weight = 1,
                                 max_delta_step = 5,
                                 gamma = 0.1)
learner_xgboost_suggested$predict_type = "prob"
```

```
result <- resample(task = task_basic,
                   learner = learner_xgboost_suggested,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9557524	0.875612

In this case there was no gain in performance.

Data transformation

Next, I applied some data transformation – summarized information on how many measurements were made.

```
dat_transformed <- dat %>%
  mutate(measurements = T4U_FTI_measured + TT4_measured + T3_measured + TSH_measured) %>%
  select(-ends_with("measured"))

task_transformed <- TaskClassif$new("sick", dat_transformed[indices_train, ], "sick", "sick")
```

```
result <- resample(task = task_transformed,
                   learner = learner_xgboost,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9677598	0.868842

Despite the fact that AUPRC dropped in value insignificantly, AUC rose by above one percent point, so I decided to keep this transformation.

Data imputation

The next step was imputing missing values. As previously – I tried imputation by histogram and using MICE algorithm.

```
po_imp_hist <- po("imputehist")
task_imp_hist <- po_imp_hist$train(list(task_transformed))[[1]]
```

```
result <- resample(task = task_imp_hist,
                   learner = learner_xgboost,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9604067	0.866983

```
task_imp_mice <- TaskClassif$new("sick",
                        cbind(
                          complete(
                            mice(
                              dat_transformed[indices_train, -21])),
                          dat_transformed[indices_train, "sick"]),
                        "sick", "sick")

set.seed(1998)
```

```
result <- resample(task = task_imp_mice,
                   learner = learner_xgboost,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9581645	0.8841467

In the first case neither one of measures gained on value, but in the second case there was improvement in AUPRC, but drop in AUC. Not sure if to keep this step I tried using another learners.

```
learner_bayes <- lrn("classif.naive_bayes")
learner_bayes$predict_type = "prob"
```

```
result <- resample(task = task_imp_hist,
                   learner = learner_bayes,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.8991092	0.6037792

Naive Bayes performed very poorly as long as AUPRC is concerned so I moved on to ranger.

```
learner_ranger <- lrn("classif.ranger")
learner_ranger$predict_type = "prob"
```

```
result <- resample(task = task_imp_mice,
                   learner = learner_ranger,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9931347	0.9069761

```
result <- resample(task = task_imp_hist,
                   learner = learner_ranger,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9937949	0.9121587

It turned out that ranger performance is way better than xgboost. I kept using it with histogram imputation.

Oversampling

The last step in exploring the ocean of possibilities of model improvement was in my case using oversampling. As in the previous homework, I used SMOTE algorithm.

```r
dat_numerized <- dat_transformed %>%
  mutate_all(as.numeric) %>%
  mutate(sick = dat_transformed$sick)

task_numerized <- TaskClassif$new("sick", dat_numerized[indices_train, ], "sick", "sick")

task_numerized <- po_imp_hist$train(list(task_numerized))[[1]]

po_smote <- po("smote", dup_size = 2)   # create twice as much positive class observations
task_numerized <- po_smote$train(list(task_numerized))[[1]]
```

```r
result <- resample(task = task_numerized,
                   learner = learner_ranger,
                   resampling = resampling_outer)
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9984646	0.9850904

AUPRC with artificial data generation jumped almost to 100%. However, having the experience form the previous homework in mind, I was very cautious with the optimism, because of the fact that up until nowe I was measuring performance using crossvalidation and in this case, when generating artificial data, it can be not very relevant result.

Final results

Finally, I used ranger trained on data imputed with histogram with some minor data transformation. I checked two configurations on the test data – with and without data imputation.

```r
enr_test_size <- nrow(task_numerized$data())

# append test set
task_numerized$rbind(dat_numerized[indices_test, ])

task_numerized <- po("imputehist")$train(list(task_numerized))[[1]]

learner_ranger$train(task_numerized,
                     row_ids = 1:enr_test_size)
```

```r
prediction <- learner_ranger$predict(
  task_numerized,
  row_ids = (enr_test_size + 1):nrow(task_numerized$data()))
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9910872	0.8652895

```r
# append test set
task_imp_hist$rbind(dat_numerized[indices_test, ])

task_imp_hist <- po("imputehist")$train(list(task_imp_hist))[[1]]
```

```r
learner_ranger$train(task_imp_hist,
                     row_ids = 1:length(indices_train))
```

```r
prediction <- learner_ranger$predict(
  task_imp_hist,
  row_ids = (length(indices_train) + 1):nrow(task_imp_hist$data()))
result$aggregate(measures)
```

classif.auc	classif.auprc
0.9930679	0.8963976

Accordingly to my expectations, model without data imputation performed better. My final result is AUPRC = 0.8963976.

Comparison to whitebox

As we can see, the result is significantly higher. Using only whitebox model (rpart) I was able to achieve AUPRC = 0.7687503