stove - regression

5/2/23

Table of contents

1 Introduction

1) 본 문서는 stove 패키지를 Shiny app에서 사용하는 것을 상정해 작성했습니다.
2) stove 패키지의 code 스타일은 OHDSI code style을 따랐습니다.

3) 본 문서에서 사용하는 global preprocessing / local preprocesing은 다음을 의미합니다. 다른 값이나 데이터의 분산 등을 사용하지 않는 전처리: 중복값 제거, 원-핫 인코딩, 피처 선택 등 -> global preprocessing 다른 값이나 데이터의 분산 등을 사용하는 전처리: Imputation, Scaling, Oversampling 등 -> local preprocessing

2 Import sample data

1) global preprocessing이 완료된 샘플데이터를 불러옵니다.

- NA가 없어야 함
- string value가 있는 열은 factor로 변환
- 한 열이 모두 같은 값으로 채워져 있을 경우 제외해야 함
- Date type column이 없어야 함
- Outcome 변수는 classification의 경우 factor, regression의 경우 numeric이어야 함 (clustering은 outcome변수를 사용하지 않음)

2) 본 문서에서 사용한 혈액검사 샘플데이터의 정보는 아래와 같습니다.

- SEX : 성별(남성:1, 여성:2)
- AGE_G : 연령(그룹)
- HGB : 혈색소
- TCHOL : 총콜레스테롤
- TG : 중성지방
- HDL : HDL 콜레스테롤
- ANE : 빈혈 진료여부(있음:1, 없음:0)
- IHD : 허혈심장질환 진료여부(있음:1, 없음:0)
- STK : 뇌혈관질환 진료여부(있음:1, 없음:0)

3) N수가 너무 크면 알고리즘에 따라 모델링 시간무 길어질 수 있습니다.

```
# remotes::install_github("statgarten/datatoys")
library(stove)
library(datatoys)
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.2.3
```

```
set.seed(1234)
cleaned_data <- datatoys::bloodTest
cleaned_data <- cleaned_data %>%
  mutate_at(vars(SEX, ANE, IHD, STK), factor) %>%
  sample_n(1000)
```

3 Data Setup Tab

User Input	description
target_var	목적 변수
train_set_ratio	전체 데이터 중 train set의 비율 (range: 0.0 – 1.0)

1) User input을 다음과 같이 받습니다.

- formula는 user가 target_var를 입력할 때 함께 생성되도록 함

```
target_var <- "TG"
train_set_ratio <- 0.7
seed <- 1234
formula <- paste0(target_var, " ~ .")
```

2) Train-test split 작업이 완료된 Object를 저장하고, Train set을 보여줍니다.

```
split_tmp <- stove::trainTestSplit(data = cleaned_data,
                                   target = target_var,
                                   prop = train_set_ratio,
                                   seed = seed
                                   )
data_train <- split_tmp[[1]] # train data
data_test <- split_tmp[[2]] # test data
data_split <- split_tmp[[3]] # whole data with split information
```

3) train set에 적용할 local preprocessing 정보를 담은 recipe를 생성합니다

```
rec <- stove::prepForCV(data = data_train,
                        formula = formula,
                        seed = seed
                        )
```

4 Modeling Tab

User Input	description
algo	사용자정의 알고리즘명

User Input	description
engine	알고리즘 구현 engine 선택
mode	mode 선택(분류/회귀)
trainingData	훈련데이터 셋
splitedData	분할정보가 담긴 전체 데이터 셋
formula	Target 변수와 Feature 변수를 정의한 formula
rec	교차 검증에서 각 fold에 적용할 local preprocessing 정보를 담은 recipe
v	교차검증시 훈련셋을 몇 번 분할할 것인지 입력
gridNum	각 하이퍼파라미터 별로 몇 개의 그리드를 할당해 베이지안 최적화를할지 설정 (ex. 모델의 하이퍼파라미터가 3개, gridNum이 5일 때, 하이퍼파라미터 최적화를 위한 그리드는 3*5=15개)
iter	베이지안 최적화 시 반복 횟수
metric	Best performance에 대한 평가지표 선택
seed	결과 재현을 위한 시드값 설정

모델 object를 저장할 빈 리스트를 생성합니다.

```
models_list <- list()
tuned_results_list <- list()
```

4.1 Linear Regression

```
# User input
mode <- "regression"
algo <- "linearRegression"
engine <- "glmnet" # glmnet (default)
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling
finalized <- stove::linearRegression(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
```

```
  formula = formula,
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

4.2 K Nearest Neighbor

```
# User input
mode <- "regression"
algo <- "KNN"
engine <- "kknn" # kknn (defualt)
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling
finalized <- stove::KNN(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
```

```
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

4.3 Decision Tree

```
# User input
mode <- "regression"
algo <- "decisionTree"
engine <- "rpart" # rpart (default), partykit
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling
finalized <- stove::decisionTree(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

4.4 Random Forest

```r
# User input
mode <- "regression"
algo <- "randomForest"
engine <- "ranger" # ranger (default), randomForest, partykit
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling
finalized <- stove::randomForest(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

4.5 XGBoost

```r
# User input
mode <- "regression"
algo <- "XGBoost"
engine <- "xgboost" # xgboost
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234
```

```r
# Modeling
finalized <- stove::xgBoost(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

4.6 lightGBM

```r
# User input
mode <- "regression"
algo <- "lightGBM"
engine <- "lightgbm" # lightgbm
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling
finalized <- stove::lightGbm(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
```

```
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

4.7 MLP

```
# User input
mode <- "regression"
algo <- "MLP"
engine <- "nnet" # nnet
v <- 2
metric <- "rmse" # rmse (default), rsq
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling
finalized <- stove::MLP(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  gridNum = gridNum,
  iter = iter,
  metric = metric,
  seed = seed
)
# Add the model to models_list
```

```
models_list[[paste0(algo, "_", engine)]] <- finalized$finalized$finalFittedModel
tuned_results_list[[paste0(algo, "_", engine)]] <- finalized$bayes_opt_result
```

5 Sources for report

5.1 Regression plot (actual vs predicted)

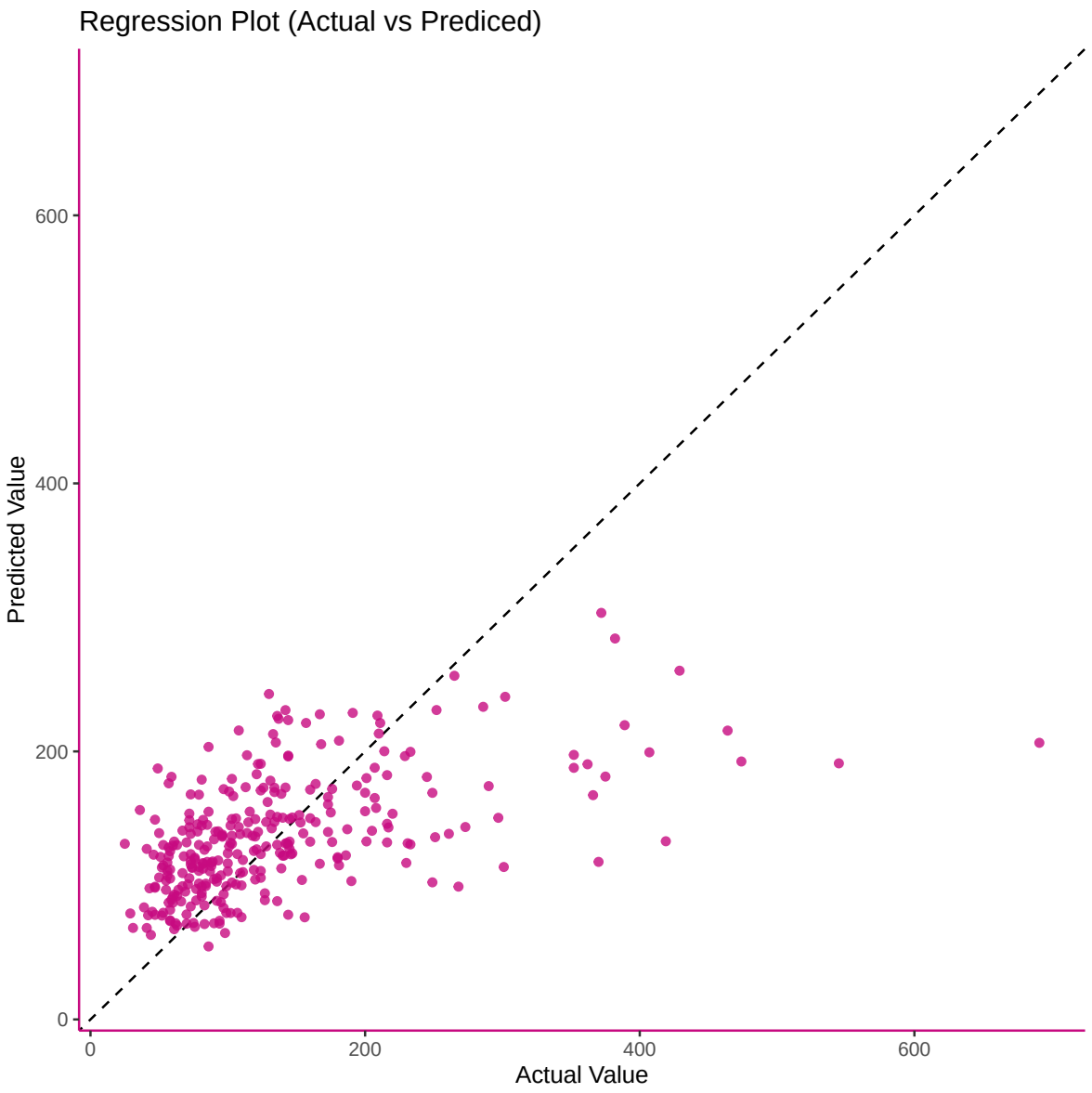
유저가 선택한 모델의 confusion matrix 출력 리스트 내 모델의 이름은 {algo}_{engine}의 형태로 저장되어 있음

```
# User input
names(models_list)
```

```
[1] "linearRegression_glmnet" "KNN_kknn"
[3] "decisionTree_rpart"      "randomForest_ranger"
[5] "XGBoost_xgboost"         "lightGBM_lightgbm"
[7] "MLP_nnet"
```

```
model_name <- "randomForest_ranger"
```

```
rp <- stove::regressionPlot(modelName = model_name,
                            modelsList = models_list,
                            targetVar = target_var)
rp
```



5.2 Evaluation metrics

- 모델 성능 비교를 위한 표 출력

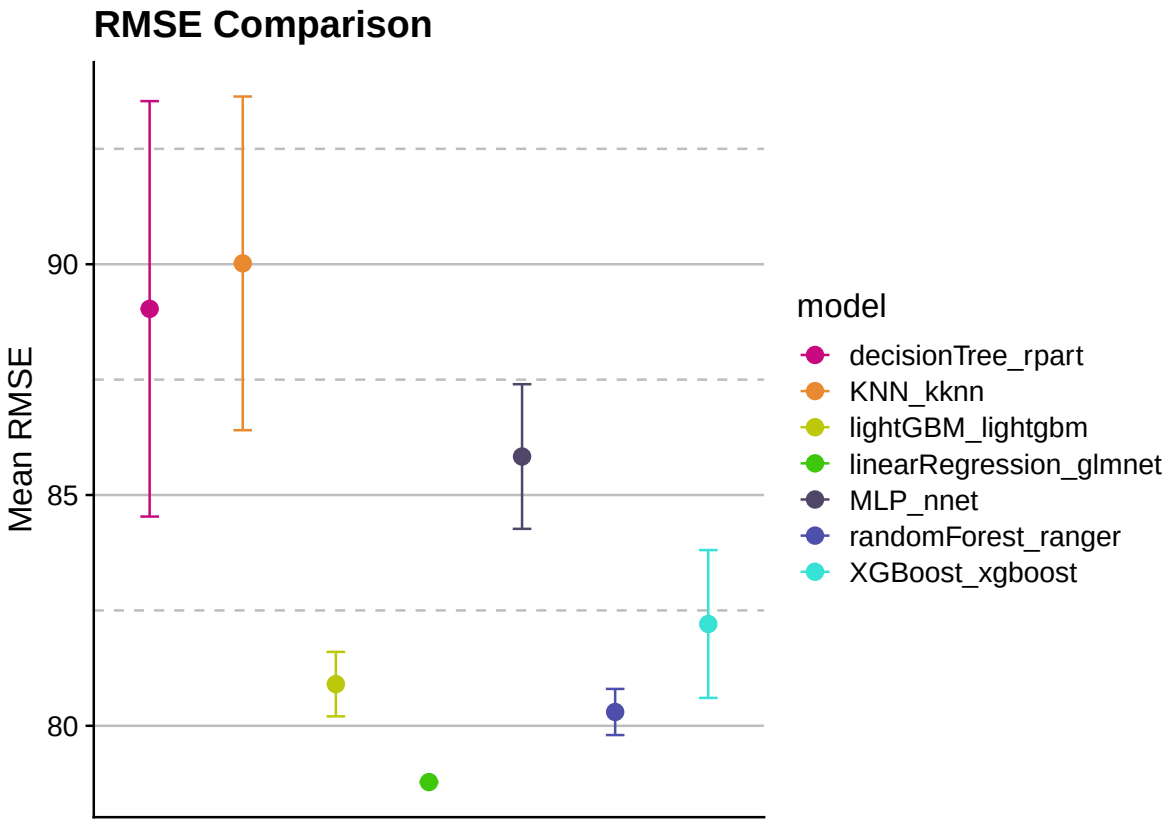
```
evalMet <- stove::evalMetricsR(models_list, target_var)
knitr::kable(evalMet)
```

	RMSE	RSQ	MAE	MASE	RPD
linearRegression_glmnet	75.879	0.309	51.428	0.568	1.204
KNN_kknn	78.338	0.263	52.434	0.579	1.166
decisionTree_rpart	79.390	0.255	55.598	0.614	1.151
randomForest_ranger	76.129	0.310	52.189	0.576	1.200
XGBoost_xgboost	73.989	0.344	50.180	0.554	1.235
lightGBM_lightgbm	76.819	0.297	51.723	0.571	1.189
MLP_nnet	91.223	0.005	62.893	0.695	1.001

5.3 RMSE plot

```
rmse_plot <- stove::plotRmseComparison(tunedResultsList = tuned_results_list,
                                       v = v,
                                       iter = iter)
rmse_plot
```

```
$rmse_plot
```




```
$rmse_summary
# A tibble: 7 x 5
  model                  mean_rmse rmse_se lower_bound upper_bound
  <chr>                      <dbl>   <dbl>       <dbl>       <dbl>
1 KNN_kknn                    90.0 1.84            86.4        93.6
2 MLP_nnet                    85.8 0.799           84.3        87.4
3 XGBoost_xgboost             82.2 0.817           80.6        83.8
4 decisionTree_rpart          89.0 2.30            84.5        93.5
5 lightGBM_lightgbm           80.9 0.356           80.2        81.6
6 linearRegression_glmnet     78.8 0.00844         78.8        78.8
7 randomForest_ranger         80.3 0.255           79.8        80.8

$model_name
[1] "linearRegression_glmnet" "KNN_kknn"
[3] "decisionTree_rpart"      "randomForest_ranger"
[5] "XGBoost_xgboost"         "lightGBM_lightgbm"
[7] "MLP_nnet"
```