

stove - regression

2022.08.24.

Table of contents

1	Introduction	1
2	Import sample data	2
3	Data Setup Tab	2
4	Modeling Tab	3
4.1	Linear Regression	4
4.2	K Nearest Neighbor	5
4.3	Decision Tree	6
4.4	Random Forest	7
4.5	XGBoost	8
4.6	lightGBM	10
4.7	MLP	11
4.8	Modeling without hyperparameter	13
5	Sources for report	13
5.1	Regression plot (actual vs predicted)	13
5.2	Evaluation metrics	15

1 Introduction

- 1) 본 문서는 stove 패키지를 Shiny app에서 사용하는 것을 상정해 작성했습니다.
- 2) 본 문서의 케이스 스타일은 Camel case와 Snake case가 혼용되어 있습니다.
 - Camel case : stove의 함수명 및 파라미터명

- Snake case: 유저로부터 받는 입력, Shiny app의 server에서 사용(될 것이라고 예상)하는 Object명, snake case로 작성된 dependencies의 함수명 등

2 Import sample data

1) 전처리가 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)

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

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 전처리 정보를 담은 recipe를 생성합니다

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

4 Modeling Tab

User Input	description
algo	ML 알고리즘 선택
engine	engine 선택
mode	mode 선택
metric	Best performance에 대한 평가지표 선택
v	Cross validation시 train set을 몇 번 분할할 것인지 입력
...	각 모델의 hyperparameter의 최소/최대값(Min, Max), 몇 단계로 나눌지(Levels)

모델 object를 저장할 빈 리스트를 생성합니다.

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

4.1 Linear Regression

```
# User input

mode <- "regression"
algo <- "linearRegression"
engine <- "glmnet" # glmnet (default), glm, stan

penalty_range_min <- 0.001
penalty_range_max <- 1.0
penalty_range_levels <- 5
mixture_range_min <- 0.0
mixture_range_max <- 1.0
mixture_range_levels <- 5

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::linearRegression(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  penaltyRangeMin = penalty_range_min,
  penaltyRangeMax = penalty_range_max,
  penaltyRangeLevels = penalty_range_levels,
  mixtureRangeMin = mixture_range_min,
  mixtureRangeMax = mixture_range_max,
  mixtureRangeLevels = mixture_range_levels,
  metric = metric
)
```

```
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel
```

4.2 K Nearest Neighbor

```
# User input

mode <- "regression"
algo <- "KNN"
engine <- "kknk" # kknk (default)

neighbors_range_min <- 1
neighbors_range_max <- 10
neighbors_range_levels <- 10

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::KNN(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  neighborsRangeMin = neighbors_range_min,
  neighborsRangeMax = neighbors_range_max,
  neighborsRangeLevels = neighbors_range_levels,
  metric = metric
)

# Add the model to models_list

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

4.3 Decision Tree

```
# User input

mode <- "regression"
algo <- "decisionTree"
engine <- "rpart" # rpart (default), partykit

tree_depth_range_min <- 1
tree_depth_range_max <- 15
tree_depth_range_levels <- 3
min_n_range_min <- 2
min_n_range_max <- 40
min_n_range_levels <- 3
cost_complexity_range_min <- -2.0
cost_complexity_range_max <- -1.0
cost_complexity_range_levels <- 2

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::decisionTree(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  treeDepthRangeMin = tree_depth_range_min,
  treeDepthRangeMax = tree_depth_range_max,
  treeDepthRangeLevels = tree_depth_range_levels,
  minNRangeMin = min_n_range_min,
  minNRangeMax = min_n_range_max,
  minNRangeLevels = min_n_range_levels,
  costComplexityRangeMin = cost_complexity_range_min,
  costComplexityRangeMax = cost_complexity_range_max,
```

```

    costComplexityRangeLevels = cost_complexity_range_levels,
    metric = metric
)

# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel

```

4.4 Random Forest

```

# User input

mode <- "regression"
algo <- "randomForest"
engine <- "ranger" # ranger (default), randomForest, partykit

mtry_range_min <- 1
mtry_range_max <- 20
mtry_range_levels <- 3
trees_range_min <- 100
trees_range_max <- 1000
trees_range_levels <- 3
min_n_range_min <- 2
min_n_range_max <- 40
min_n_range_levels <- 3

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::randomForest(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,

```

```

mtryRangeMin = mtry_range_min,
mtryRangeMax = mtry_range_max,
mtryRangeLevels = mtry_range_levels,
treesRangeMin = trees_range_min,
treesRangeMax = trees_range_max,
treesRangeLevels = trees_range_levels,
minNRangeMin = min_n_range_min,
minNRangeMax = min_n_range_max,
minNRangeLevels = min_n_range_levels,
metric = metric
)

# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel

```

4.5 XGBoost

```

# User input

mode <- "regression"
algo <- "XGBoost"
engine <- "xgboost" # xgboost

tree_depth_range_min <- 5
tree_depth_range_max <- 15
tree_depth_range_levels <- 3
trees_range_min <- 8
trees_range_max <- 32
trees_range_levels <- 3
learn_rate_range_min <- -2.0
learn_rate_range_max <- -1.0
learn_rate_range_levels <- 2
mtry_range_min <- 0.0
mtry_range_max <- 1.0
mtry_range_levels <- 3
min_n_range_min <- 2
min_n_range_max <- 40
min_n_range_levels <- 3
loss_reduction_range_min <- -1.0
loss_reduction_range_max <- 1.0

```



```

loss_reduction_range_levels <- 3
sample_size_range_min <- 0.0
sample_size_range_max <- 1.0
sample_size_range_levels <- 3
stop_iter <- 30

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::xgBoost(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  treeDepthRangeMin = tree_depth_range_min,
  treeDepthRangeMax = tree_depth_range_max,
  treeDepthRangeLevels = tree_depth_range_levels,
  treesRangeMin = trees_range_min,
  treesRangeMax = trees_range_max,
  treesRangeLevels = trees_range_levels,
  learnRateRangeMin = learn_rate_range_min,
  learnRateRangeMax = learn_rate_range_max,
  learnRateRangeLevels = learn_rate_range_levels,
  mtryRangeMin = mtry_range_min,
  mtryRangeMax = mtry_range_max,
  mtryRangeLevels = mtry_range_levels,
  minNRangeMin = min_n_range_min,
  minNRangeMax = min_n_range_max,
  minNRangeLevels = min_n_range_levels,
  lossReductionRangeMin = loss_reduction_range_min,
  lossReductionRangeMax = loss_reduction_range_max,
  lossReductionRangeLevels = loss_reduction_range_levels,
  sampleSizeRangeMin = sample_size_range_min,
  sampleSizeRangeMax = sample_size_range_max,
  sampleSizeRangeLevels = sample_size_range_levels,

```

```

    stopIter = stop_iter,
    metric = metric
)

# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel

```

4.6 lightGBM

```

# User input

mode <- "regression"
algo <- "lightGBM"
engine <- "lightgbm" # lightgbm

tree_depth_range_min <- 5
tree_depth_range_max <- 15
tree_depth_range_levels <- 3
trees_range_min <- 10
trees_range_max <- 100
trees_range_levels <- 2
learn_rate_range_min <- -2.0
learn_rate_range_max <- -1.0
learn_rate_range_levels <- 2
mtry_range_min <- 1
mtry_range_max <- 20
mtry_range_levels <- 3
min_n_range_min <- 2
min_n_range_max <- 40
min_n_range_levels <- 3
loss_reduction_range_min <- -1.0
loss_reduction_range_max <- 1.0
loss_reduction_range_levels <- 3

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

```

```

finalized <- stove::lightGbm(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  treeDepthRangeMin = tree_depth_range_min,
  treeDepthRangeMax = tree_depth_range_max,
  treeDepthRangeLevels = tree_depth_range_levels,
  treesRangeMin = trees_range_min,
  treesRangeMax = trees_range_max,
  treesRangeLevels = trees_range_levels,
  learnRateRangeMin = learn_rate_range_min,
  learnRateRangeMax = learn_rate_range_max,
  learnRateRangeLevels = learn_rate_range_levels,
  mtryRangeMin = mtry_range_min,
  mtryRangeMax = mtry_range_max,
  mtryRangeLevels = mtry_range_levels,
  minNRangeMin = min_n_range_min,
  minNRangeMax = min_n_range_max,
  minNRangeLevels = min_n_range_levels,
  lossReductionRangeMin = loss_reduction_range_min,
  lossReductionRangeMax = loss_reduction_range_max,
  lossReductionRangeLevels = loss_reduction_range_levels,
  metric = metric
)

# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel

```

4.7 MLP

```

# User input

mode <- "regression"
algo <- "MLP"
engine <- "nnet" # nnet

```

```

hidden_units_range_min <- 1
hidden_units_range_max <- 10
hidden_units_range_levels <- 3
penalty_range_min <- 0.001
penalty_range_max <- 1.0
penalty_range_levels <- 3
epochs_range_min <- 10
epochs_range_max <- 100
epochs_range_levels <- 2

v <- 2

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::MLP(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  v = v,
  hiddenUnitsRangeMin = hidden_units_range_min,
  hiddenUnitsRangeMax = hidden_units_range_max,
  hiddenUnitsRangeLevels = hidden_units_range_levels,
  penaltyRangeMin = penalty_range_min,
  penaltyRangeMax = penalty_range_max,
  penaltyRangeLevels = penalty_range_levels,
  epochsRangeMin = epochs_range_min,
  epochsRangeMax = epochs_range_max,
  epochsRangeLevels = epochs_range_levels,
  metric = metric
)

# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel

```

4.8 Modeling without hyperparameter

함수 내에 기본값을 선언해 뒀기때문에, 유저로부터 입력을 받지 않아도 모델링이 가능합니다. 아래처럼 hyperparameter 관련 파라미터, v를 따로 입력받지 않아도 됩니다.

```
# User input

mode <- "regression"
algo <- "linearAuto"
engine <- "glmnet" # glmnet (default), glm, stan

metric <- "rmse" # rmse (default), rsq

# Modeling

finalized <- stove::linearRegression(
  algo = algo,
  engine = engine,
  mode = mode,
  trainingData = data_train,
  splitedData = data_split,
  formula = formula,
  rec = rec,
  # v = v,
  # penaltyRangeMin = penalty_range_min,
  # penaltyRangeMax = penalty_range_max,
  # penaltyRangeLevels = penalty_range_levels,
  # mixtureRangeMin = mixture_range_min,
  # mixtureRangeMax = mixture_range_max,
  # mixtureRangeLevels = mixture_range_levels,
  metric = metric
)
```

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

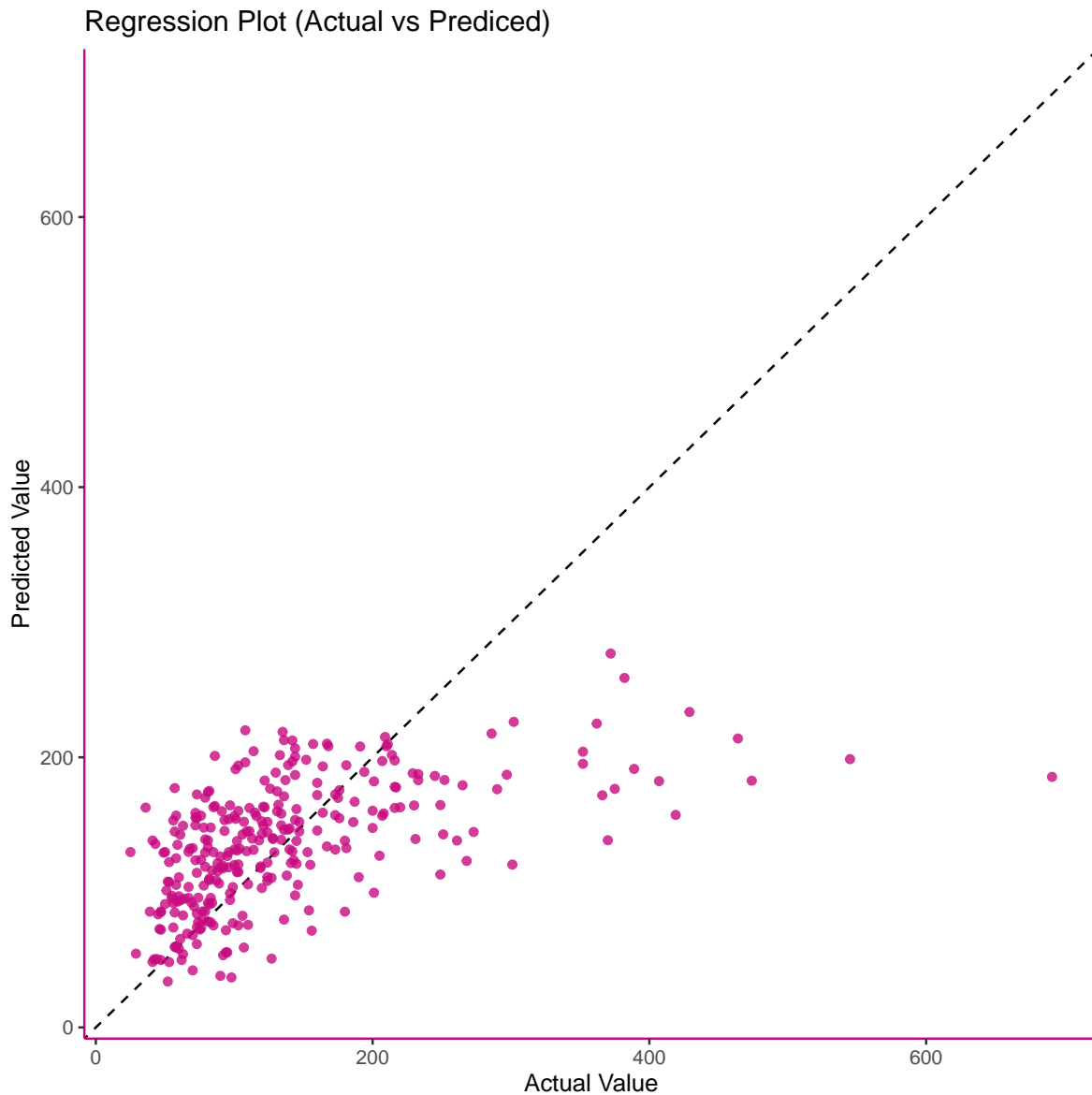
```
# User input
names(models_list)
```

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

```
model_name <- "linearRegression_glmnet"
```

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

Warning: Removed 3 rows containing missing values (geom_point).



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	75.896	0.308	51.424	0.568	1.204
KNN_kknn	80.305	0.232	53.693	0.593	1.138
decisionTree_rpart	79.611	0.246	56.078	0.619	1.147
randomForest_ranger	76.293	0.301	52.303	0.578	1.197
XGBoost_xgboost	75.166	0.323	49.546	0.547	1.215
lightGBM_lightgbm	77.573	0.284	52.465	0.579	1.178
MLP_nnet	75.577	0.317	50.189	0.554	1.209