stove - classification

2022.12.05.

Table of contents

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)

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 시간이 너무 길어질 수 있습니다. 빠른 결과를 얻고싶다면 1,000개 이하로 sampling하는 것을 추천합니다.

```
# 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) %>%
  mutate(TG = ifelse(TG < 150, 0, 1)) %>%
```

```
  mutate_at(vars(TG), factor) %>%
  group_by(TG) %>%
  sample_n(500) # TG(0):TG(1) = 500:500
```

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,
                        imputation = T,
                        normalization = T,
                        seed = seed
                        )
```

4 Modeling Tab

User Input	description
mode	mode 선택(분류/회귀)
algo	사용자정의 알고리즘명
engine	알고리즘 구현 engine 선택
v	교차검증시 훈련셋을 몇 번 분할할 것인지 입력
metric	Best performance에 대한 평가지표 선택
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
seed	결과 재현을 위한 시드값 설정

모델 object를 저장할 빈 리스트를 생성합니다.

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

4.1 Logistic Regression

```
# User input

mode <- "classification"
algo <- "logisticRegression" # Custom name
engine <- "glmnet" # glmnet (default)
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling

finalized <- stove::logisticRegression(
  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$finalFittedModel
```

4.2 K Nearest Neighbor

```
# User input

mode <- "classification"
algo <- "KNN"
engine <- "kknn" # kknn (defualt)
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
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$finalFittedModel
```

4.3 Naive Bayes

```
# User input

mode <- "classification"
algo <- "naiveBayes"
engine <- "naivebayes" # klaR (default), naivebayes
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling

finalized <- stove::naiveBayes(
  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$finalFittedModel
```

4.4 Decision Tree

```
# User input

mode <- "classification"
algo <- "decisionTree"
engine <- "partykit" # rpart (default), C5.0, partykit
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
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$finalFittedModel
```

4.5 Random Forest

```
# User input

mode <- "classification"
algo <- "randomForest"
```

```r
engine <- "randomForest" # ranger (default), randomForest, partykit(?)
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
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$finalFittedModel
```

4.6 XGBoost

```r
# User input

mode <- "classification"
algo <- "XGBoost"
engine <- "xgboost" # xgboost
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
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$finalFittedModel
```

4.7 lightGBM

```r
# User input

mode <- "classification"
algo <- "lightGBM"
engine <- "lightgbm" # lightgbm
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
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$finalFittedModel
```

4.8 MLP

```
# User input

mode <- "classification"
algo <- "MLP"
engine <- "nnet" # nnet
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
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$finalFittedModel
```

4.9 SVM - Linear kernel

```
# User input

mode <- "classification"
algo <- "SVM_linear"
engine <- "kernlab"
v <- 2
metric <- "roc_auc" # roc_auc (default), accuracy
gridNum <- 5
iter <- 10
seed <- 1234

# Modeling

finalized <- stove::SVMLinear(
  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
)
```

```
Setting default kernel parameters
```

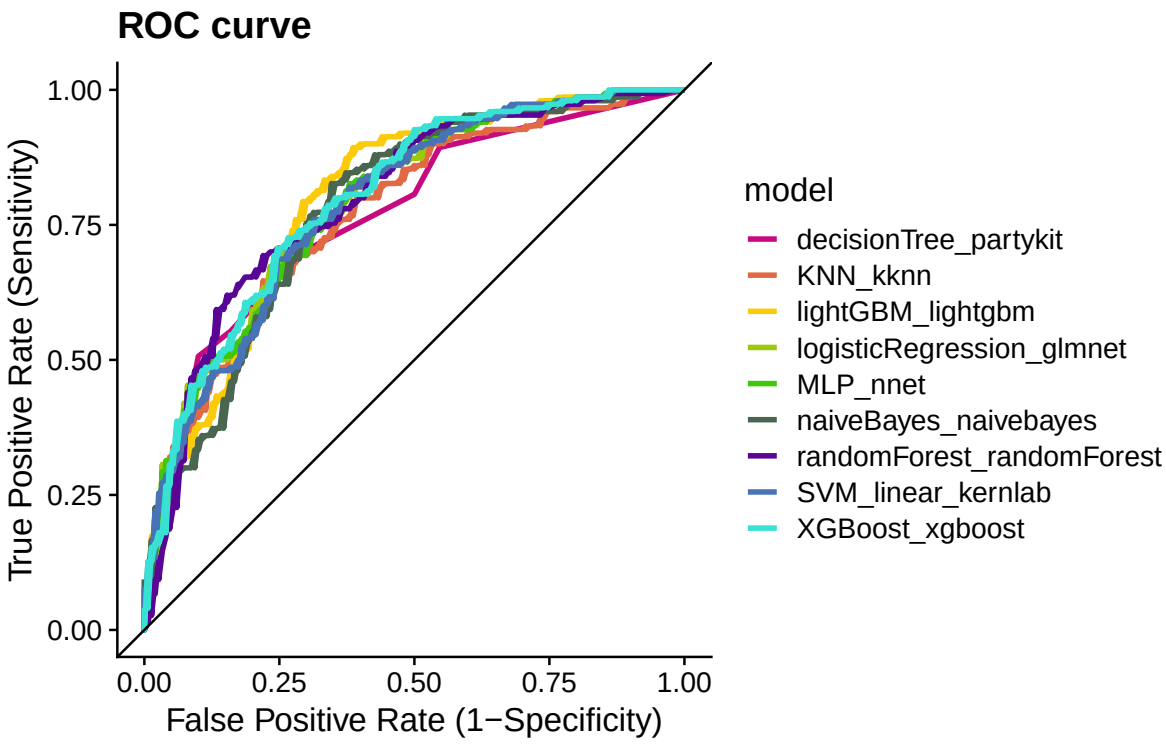
```
# Add the model to models_list
models_list[[paste0(algo, "_", engine)]] <- finalized$finalFittedModel
```

5 Sources for report

5.1 ROC Curve

유저가 선택한 모델의 ROC curve 출력

```
roc_curve <- stove::rocCurve(
  modelsList = models_list,
  targetVar = target_var
)
roc_curve
```



5.2 Confusion Matrix

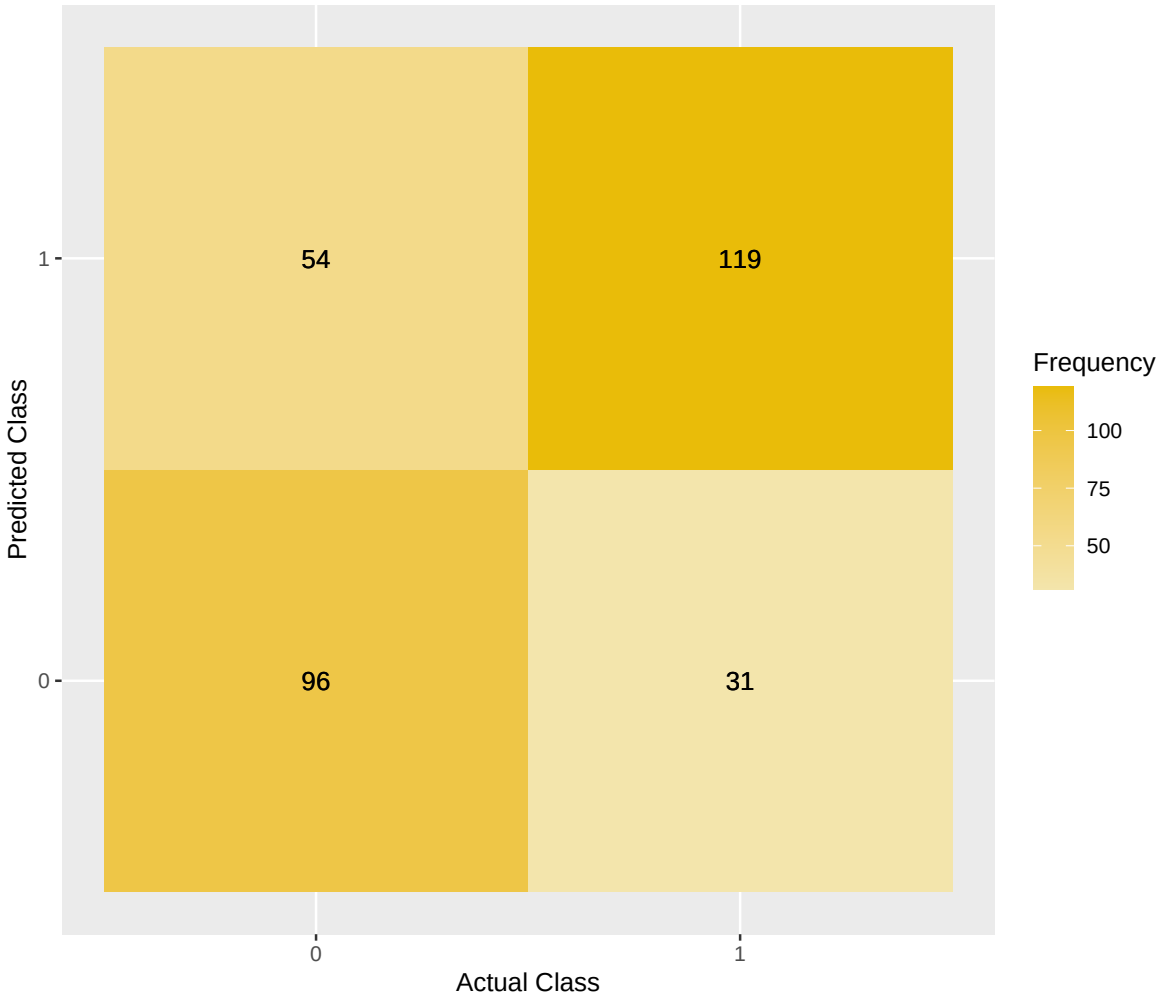
유저가 선택한 모델의 confusion matrix 출력 리스트 내 모델의 이름은 {algo}_{engine}의 형태로 저장되어 있음

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

```
[1] "logisticRegression_glmnet" "KNN_kknn"
[3] "naiveBayes_naivebayes"     "decisionTree_partykit"
[5] "randomForest_randomForest" "XGBoost_xgboost"
[7] "lightGBM_lightgbm"         "MLP_nnet"
[9] "SVM_linear_kernlab"
```

```
model_name <- "SVM_linear_kernlab"
```

```
cm <- stove::confusionMatrix(
  modelName = model_name,
  modelsList = models_list,
  targetVar = target_var
)
cm
```



5.3 Evaluation metrics

- 모델 성능 비교를 위한 표 출력
- options(yardstick.event_level = “second”)은 오름차순으로 factor의 level 설정하기 위한 옵션

```
options(yardstick.event_level = "second")
evalMet <- stove::evalMetricsC(models_list, target_var)
knitr::kable(evalMet)
```

	Accuracy	Recall	Specificity	Precision	F1-score	Kappa	MCC
logisticRegression_glmnet	0.710	0.667	0.753	0.730	0.697	0.420	0.422
KNN_kknn	0.693	0.647	0.740	0.713	0.678	0.387	0.388
naiveBayes_naivebayes	0.707	0.733	0.680	0.696	0.714	0.413	0.414
decisionTree_partykit	0.703	0.707	0.700	0.702	0.704	0.407	0.407
randomForest_randomForest	0.713	0.680	0.747	0.729	0.703	0.427	0.428
XGBoost_xgboost	0.727	0.713	0.740	0.733	0.723	0.453	0.453
lightGBM_lightgbm	0.747	0.700	0.793	0.772	0.734	0.493	0.495
MLP_nnet	0.710	0.673	0.747	0.727	0.699	0.420	0.421
SVM_linear_kernlab	0.717	0.640	0.793	0.756	0.693	0.433	0.439