KNN

Jan Alexander Jensen

2020/4/26

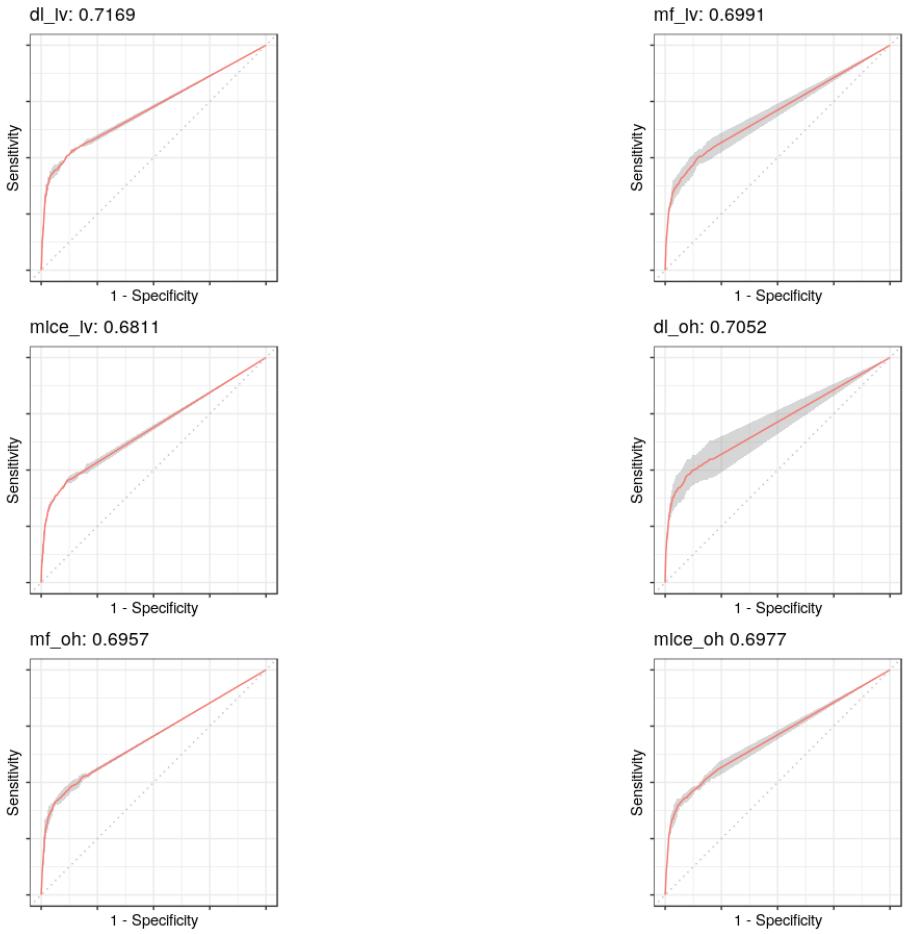
KNN

KNN (k-nearest neighbors) is a method used for classification. It takes k neighbors and based on how the neighbors are classified, and we will assign the new input to the most popular category in the k neighbors. Here the number of k and the method of calculating the distances between data point are crucial.

```
library(mlr3)
library(tidyverse)
library(ggplot2)
library(mlr3learners)
library(data.table)
library(mlr3viz)
library(mlr3tuning)
library(mlr3pipelines)
library(paradox)
library(skimr)
library(smotefamily)
library(gridExtra)

# suppress package making warning by start up in train
# Warning: "package 'kknn' was built under R version 3.6.3"
suppressPackageStartupMessages(library(kknn))

# read data with different encoding

dl_iv_data <- read.csv2("credit_card_prediction/iv_data/dl_iv_data.csv") %>% mutate(y = as.factor(y))
mf_iv_data <- read.csv2("credit_card_prediction/iv_data/mf_iv_data.csv") %>% mutate(y = as.factor(y))
mice_iv_data <- read.csv2("credit_card_prediction/iv_data/mice_iv_data.csv") %>% mutate(y = as.factor(y)

dl_oh_data <- read.csv("credit_card_prediction/oh_data/dl_oh_data.csv") %>% mutate(y = as.factor(y))
mf_oh_data <- read.csv("credit_card_prediction/oh_data/mf_oh_data.csv") %>% mutate(y = as.factor(y))
mice_oh_data <- read.csv("credit_card_prediction/oh_data/mice_oh_data.csv") %>% mutate(y = as.factor(y)


# load data directly into tasks for further training
tasks <- list(
  TaskClassif$new("dl_iv", backend = dl_iv_data, target = "y"),
  TaskClassif$new("mf_iv", backend = mf_iv_data, target = "y"),
  TaskClassif$new("mice_iv", backend = mice_iv_data, target = "y"),
  TaskClassif$new("dl_oh", backend = dl_oh_data, target = "y"),
  TaskClassif$new("mf_oh", backend = mf_oh_data, target = "y"),
  TaskClassif$new("mice_oh", backend = mice_oh_data, target = "y")
```

```
)

# remove raw data to save memory
rm(dl_iv_data, mf_iv_data, mice_iv_data, dl_oh_data, mf_oh_data, mice_oh_data)

# knn learner
knn_learner <- lrn("classif.kknn", predict_type = "prob")

# setting the tunning for parameters, and terminator
knn_param_set <- ParamSet$new(params = list(ParamInt$new("k", lower = 20, upper = 20)))
terms <- term("none")

# creat autotuner, using the inner sampling and tuning parameter with random search
inner_rsmp <- rsmp("cv", folds = 5L)
knn_auto <- AutoTuner$new(learner = knn_learner, resampling = inner_rsmp,
                          measures = msr("classif.auc"), tune_ps = knn_param_set,
                          terminator = terms, tuner = tnr("grid_search"))

outer_rsmp <- rsmp("cv", folds = 3L)
design = benchmark_grid(
  tasks = tasks,
  learners = knn_auto,
  resamplings = outer_rsmp
)

# set seed before traing, then run the benchmark
# save the results afterwards
set.seed(2020)
knn_bmr <- benchmark(design, store_models = TRUE)

# autoplot auc for all tasks (merged in one plot)
multiplot_roc <- function(models, type="roc"){
  plots <- list()
  thm <- theme(axis.text.x = element_blank(), axis.text.y = element_blank())
  model <- models$clone()$filter(task_id = "dl_iv")
  auc <- round(model$aggregate(msr("classif.auc"))[[7]], 4)
  plots[[1]] <- autoplot(model, type = type) + ggtitle(paste("dl_iv:", auc)) + thm

  model <- models$clone()$filter(task_id = "mf_iv")
  auc <- round(model$aggregate(msr("classif.auc"))[[7]], 4)
  plots[[2]] <- autoplot(model, type = type) + ggtitle(paste("mf_iv:", auc)) + thm

  model <- models$clone()$filter(task_id = "mice_iv")
  auc <- round(model$aggregate(msr("classif.auc"))[[7]], 4)
  plots[[3]] <- autoplot(model, type = type) + ggtitle(paste("mice_iv:", auc)) + thm

  model <- models$clone()$filter(task_id = "dl_oh")
  auc <- round(model$aggregate(msr("classif.auc"))[[7]], 4)
  plots[[4]] <- autoplot(model, type = type) + ggtitle(paste("dl_oh:", auc)) + thm

  model <- models$clone()$filter(task_id = "mf_oh")
  auc <- round(model$aggregate(msr("classif.auc"))[[7]], 4)
  plots[[5]] <- autoplot(model, type = type) + ggtitle(paste("mf_oh:", auc)) + thm
```

```
  model <- models$clone()$filter(task_id = "mice_oh")
  auc <- round(model$aggregate(msr("classif.auc"))[[7]], 4)
  plots[[6]] <- autoplot(model, type = type) + ggtitle(paste("mice_oh", auc)) + thm
  do.call("grid.arrange", plots)
}

multiplot_roc(knn_bmr)
```



KNN only with dl_vi

KNN performs with no significant difference between different encoding and missing data handling methods. Since we used a binary variable to indicate whether a category is present or not, the max distance can only be 1 or 0. Moreover, other numeric variables have a more significant distance, meaning that they have a more substantial impact on the distance than the categorical data without having a significant correlation with our target variable. To get better results, it would be necessary to either use other ways to handle categorical data better for distance calculation or using other training methods to perform classification instead of KNN.