PS10 Finley

Caden Finley

April 2023

1 Workflow

library(tidyverse) library(tidymodels) library(magrittr) library(modelsummary) library(rpart) library(e1071) library(kknn) library(nnet) library(kernlab) library(readr)

set.seed(100)

income ¡- read$_csv(”http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data”, col_names = FALSE)names(income) < -c(”age”, ”workclass”, ”fnlwgt”, ”edu$

Clean up the data Drop unnecessary columns income Make sure continuous variables are formatted as numeric income Make sure discrete variables are formatted as factors income Combine levels of factor variables that currently have too many levels income Advanced = c(”Masters”,”Doctorate”,”Prof-school”), Bachelors = c(”Bachelors”), SomeCollege = c(”Some-college”,”Assoc-acdm”,”Assoc-voc”), HSgrad = c(”HS-grad”,”12th”), HSdrop = c(”11th”,”9th”,”7th-8th”,”1st-4th”,”10th”,”5th-6th”,”Preschool”)), marital.status = fct$_collapse(marital.status, Married = c(”Married-civ-spouse”, ”Married-spouse-absent”, ”Married-AF-spouse”), Divorced = c(”Divorced”, ”Separated”), Widowed = c(”Widowed”), NeverMarried = c(”Never-married”)), race = fct_collapse(race, White = c(”White”), Black = c(”Black”), Asian = c(”Asian-Pac-Islander”), Other = c(”Other”, ”Amer-Indian-Eskimo”)), workclass = fct_collapse(workclass, Private = c(”Private”), SelfEmp = c(”Self-emp-not-inc”, ”Self-emp-inc”), Gov = c(”Federal-gov”, ”Local-gov”, ”State-gov”), Other = c(”Without-pay”, ”Never-worked”, ”?”)), occupation = fct_collapse(occupation, BlueCollar = c(”?”, ”Craft-repair”, ”Farming-fishing”, ”Handlers-cleaners”, ”Machine-op-inspct”, ”Transport-moving”), WhiteCollar = c(”Adm-clerical”, ”Exec-managerial”, ”Prof-specialty”, ”Sales”, ”Tech-support”), Services = c(”Armed-Forces”, ”Other-service”, ”Priv-house-serv”, ”Protective-serv”)))$

tidymodels time! income$_split < -initial_split(income, prop = 0.8)income_train < -training(income_split)income_test < -testing(income_split)$

logistic regression print(’Starting LOGIT’) set up the task and the engine tune$_logit_spec < -logistic_reg(penalty = tune(), tuningparametermixture = 11 = lasso, 0 = ridge)set_engine(”glmnet”)set_mode(”classification”)$

define a grid over which to try different values of the regularization parameter lambda lambda$_grid < -grid_regular(penalty(), levels = 50)$

3-fold cross-validation rec$_folds < -vfold_cv(income_train, v = 3)$

Workflow rec$_{w}f < -workflow()add_{m}odel(tune_{l}ogit_{s}pec)add_{f}ormula(high.earner\ education + marital.status + race + workclass + occupation + relationship + sex + age + capital.gain + capital.loss + hours)$

Tuning results rec$_{r}es < -rec_{w}ftune_{g}rid(resamples = rec_{f}olds, grid = lambda_{g}rid)$

what is the best value of lambda? top$_{a}cc < -show_{b}est(rec_{r}es, metric = ''accuracy'')best_{a}cc < -select_{b}est(rec_{r}es, metric = ''accuracy'')final_{l}ogit_{l}asso < -finalize_{w}orkflow(rec_{w}f, best_{a}cc)$

print('*********** LOGISTIC REGRESSION **************') logit$_{t}est < -last_{f}it(final_{l}ogit_{l}asso, income_{s}plit)collect_{m}etrics()$

logit$_{t}esttop_{a}cc$

combine results into a nice tibble (for later use) logit$_{a}ns < -top_{a}cclogit_{a}nsmutate(alg = ''logit'')$

tree model print('Starting TREE') set up the task and the engine tune$_{t}ree_{s}pec < -decision_{t}ree(min_{n} = tune(), tuningparametertree_{d}epth = tune(), tuningparametercost_{c}omplexity = tune(), tuningparameter)set_{e}ngine(''rpart'')set_{m}ode(''classification'')$

define a set over which to try different values of tuning parameters tree$_{g}rid < -grid_{l}atin_{h}ypercube(min_{n}(), tree_{d}epth(), cost_{c}omplexity(), size = 20)$

3-fold cross-validation tree$_{f}olds < -vfold_{c}v(income_{t}rain, v = 3)$

Workflow tree$_{w}f < -workflow()add_{m}odel(tune_{t}ree_{s}pec)add_{f}ormula(high.earner\ education + marital.status + race + workclass + occupation + relationship + sex + age + capital.gain + capital.loss + hours)$

Tuning results tree$_{r}es < -tree_{w}ftune_{g}rid(resamples = tree_{f}olds, grid = tree_{g}rid)$

what are the best values of the tuning parameters? top$_{a}cc_{t}ree < -show_{b}est(tree_{r}es, metric = ''accuracy'')best_{a}cc_{t}ree < -select_{b}est(tree_{r}es, metric = ''accuracy'')final_{t}ree < -finalize_{w}orkflow(tree_{w}f, best_{a}cc_{t}ree)$

print('*********** DECISION TREE **************') tree$_{t}est < -last_{f}it(final_{t}ree, income_{s}plit)collect$

tree$_{t}esttop_{a}cc_{t}ree$

combine results into a nice tibble (for later use) tree$_{a}ns < -top_{a}cc_{t}reetree_{a}nsmutate(alg = ''tree'')$

k-nearest neighbors print('Starting KNN') set up the task and the engine tune$_{k}nn_{s}pec < -nearest_{n}eighbor(weight_{f}unc = tune(), tuningparameterneighbors = tune()tuningparameter)set_{e}ngine(''kknn'')set_{m}ode(''classification'')$

define a set over which to try different values of tuning parameters knn$_{g}rid < -grid_{l}atin_{h}ypercube(weight_{f}unc(), neighbors(), size = 20)$

3-fold cross-validation knn$_{f}olds < -vfold_{c}v(income_{t}rain, v = 3)$

Workflow knn$_{w}f < -workflow()add_{m}odel(tune_{k}nn_{s}pec)add_{f}ormula(high.earner\ education + marital.status + race + workclass + occupation + relationship + sex + age + capital.gain + capital.loss + hours)$

Tuning results knn$_{r}es < -knn_{w}ftune_{g}rid(resamples = knn_{f}olds, grid = knn_{g}rid)$

what are the best values of the tuning parameters? top$_{a}cc_{k}nn < -show_{b}est(knn_{r}es, metric = ''accuracy'')best_{a}cc_{k}nn < -select_{b}est(knn_{r}es, metric = ''accuracy'')final_{k}nn < -finalize_{w}orkflow(knn_{w}f, best_{a}cc_{k}nn)$

print('*********** K-NEAREST NEIGHBORS **************') knn$_test < -last_fit(final_knn, income_split)collect_metrics()$

knn$_ttesttop_acc_knn$

combine results into a nice tibble (for later use) knn$_ans < -top_acc_knn$

SVM

print('Starting SVM') set up the task and the engine tune$_svm_spec < -svm_poly(degree = tune(), tuningparameter : degreeofpolynomialkernelscale = tune(), tuningparameter : scalingfactorforkernelcost = tune(), tuningparameter : costofviolationepsilon = 0.1epsilonforepsilon-insensitivelossfunction)set_engine(''LiblineaR'')set_mode(''classification'')$

define a grid over which to try different values of tuning parameters svm$_grid < -grid_tune(degree = seq(1, 5, by = 1), trydegreesfrom1to5scale = seq(0.1, 1, by = 0.1), tryscalingfactorsfrom0.1to1cost = 10^seq(-3, 3, by = 1)trycostsfrom0.001to1000)$

Workflow svm$_wf < -workflow()add_model(tune_svm_spec)add_formula(high.earner\ education + marital.status + race + workclass + occupation + relationship + sex + age + capital.gain + capital.loss + hours)$

Tuning results svm$_res < -svm_wftune_grid(resamples = rec_folds, grid = svm_grid)$

what is the best combination of tuning parameters? top$_acc_svm < -show_best(svm_res, metric = ''accuracy'')best_acc_svm < -select_best(svm_res, metric = ''accuracy'')final_svm < -finalize_workflow(svm_wf, best_acc_svm)print('***********SUPPORTVECTORMACHINE(SVM)* *************')svm_test < -last_fit(final_svm, income_split)collect_metrics()$

svm$_ttesttop_acc_svm$

combine results into a nice tibble (for later use) svm$_ans < -top_acc_svmsvm_ansmutate(alg = ''svm'')$

combine answers all$_ans < -bind_rows(logit_ans, tree_ans, nnet_ans, knn_ans, svm_ans)datasummary_df(all_ans$

2 Explanation

I apologize for the messy code and lack of answers, I was not sure how to interpret some of what was required in the problem set. I had issues with the time constraint this time so I could not get it presented like I wanted to. Alongside that, getting this code to translate over in LaTex was giving me trouble, so again I apologize for the sloppy work.