1 model development

Variable selection using ElasticNet with stratified cross-validation for hyper-parameter tuning

Ing. Peter Tomko, M.A.

1/9/2020

Introduction

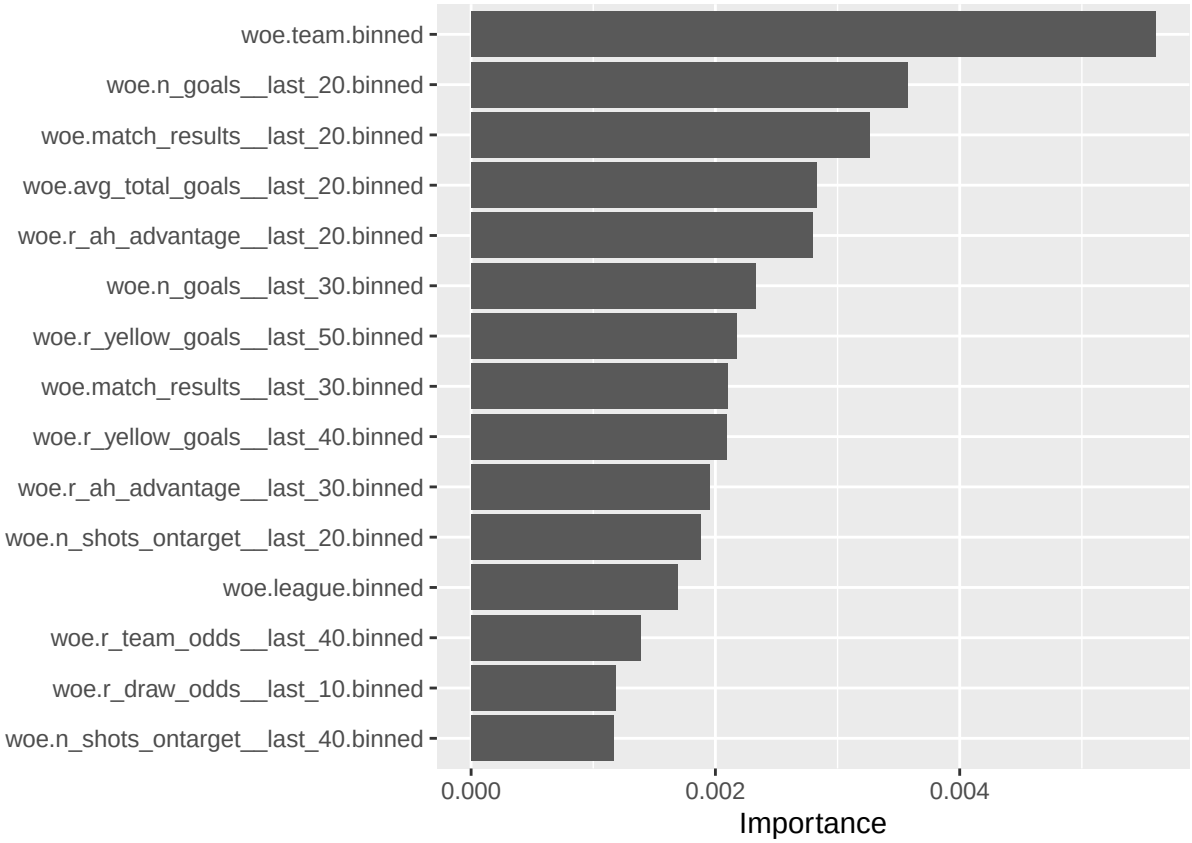
`woeBinning` helped to preliminary select potential variables. After applying elastic net, further elimination will help to narrow down potential variables to cca. 7-10 for final model.

Especially interesting, for this exercise, is to use standardized user interface provided in the package `tidymodels`.

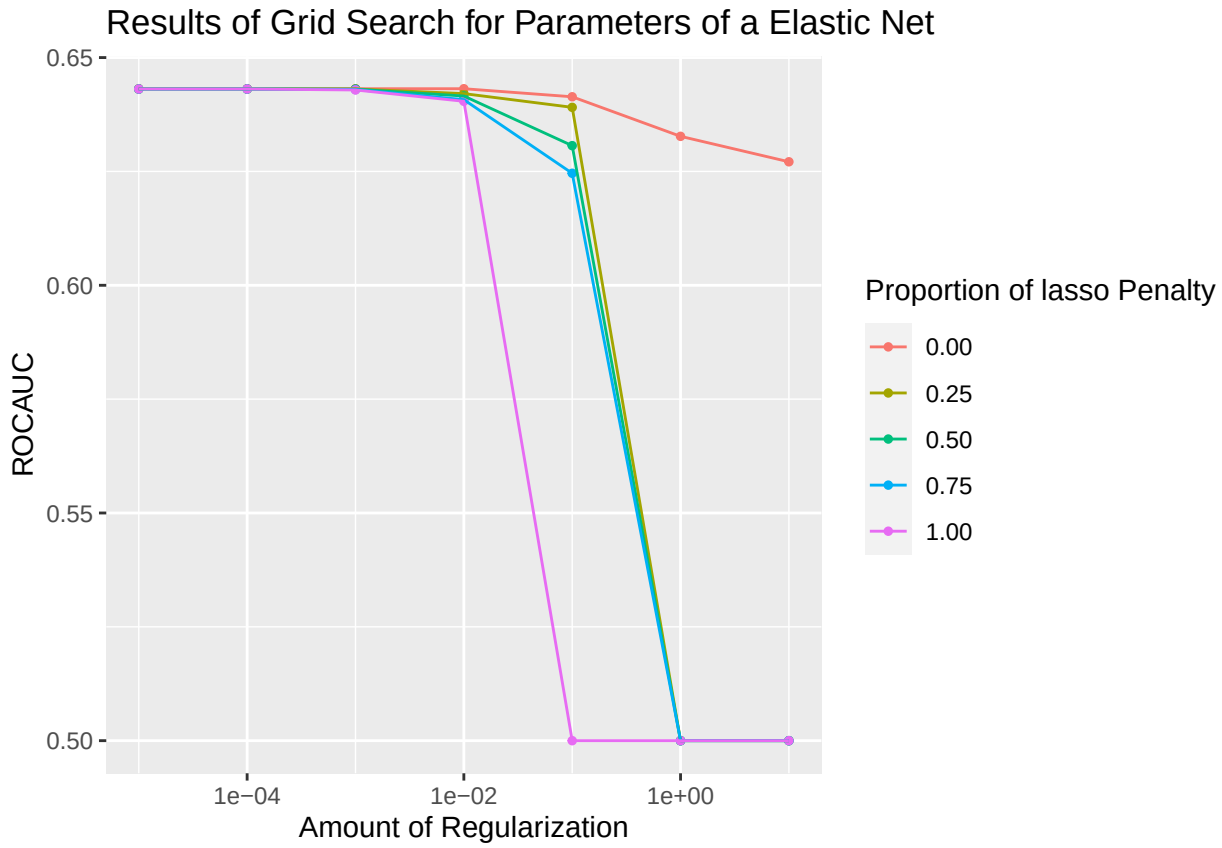
```
library(vip)
library(tidymodels)
library(stringr)
library(tidyverse)
library(recipes)

# ----- Specify Output Path -----
proj_path <- "C:/Users/Peter/Desktop/ds_projects/betting_data_science"
folder_path <- "6 betting/1 model development/preliminary_data"
file_path <- "1a variable selection - binning.RData"
out_file_path <- "1b elastic net modelling.RData"

input_path <- paste(proj_path, "/", folder_path, "/", file_path, sep = "")

# ----- Upload Image -----
load(input_path)

# - set output folder
output_path <- paste(proj_path, "/", folder_path, "/", out_file_path, sep = "")

rm(proj_path)
rm(folder_path)
rm(file_path)
rm(out_file_path)

# ----- Upsampled Data for Modelling -----
train_data_glm <-
  recipe(n_goals_cat ~ .,
         data = master_data %>%
           filter(data_type %in% "Train") %>%
           select(data_type, binned_data) %>%
           unnest(c(binned_data)) %>%
           as.data.frame() %>%
           select(-data_type, -is_home, -created_at, -match_id, -n_goals) %>%
```

```
          mutate_if(is.character, as.factor)) %>%
  themis::step_upsample(n_goals_cat) %>%
  prep() %>%
  juice() %>%
  as.data.frame()

# ----- Model Structure -----
ml_model_str <-

  # - parameters for tuning (similar to lambda and alpha from glmnet package)
  logistic_reg(penalty = tune(),
               mixture = tune()) %>%

  # - glmnet, i.e. elastic net engine
  set_engine("glmnet") %>%
  set_mode("classification")

# ----- Stratified Sampling -----
cv_splits <- vfold_cv(train_data_glm, strata = n_goals_cat)

# ----- Create Workflow -----
ml_workflow <-
  workflow() %>%
  add_model(ml_model_str) %>%
  add_formula(n_goals_cat ~ .)

# ----- Set Parameters -----
glmn_set <- parameters(penalty(range = c(-5, 1), trans = log10_trans()),
                       mixture(range = c(0, 1)))

# ----- Create Grid -----
glmn_grid <- grid_regular(glmn_set, levels = c(7, 5))
ctrl <- control_grid(save_pred = TRUE, verbose = TRUE)

# ----- Tune Model -----
glmn_tune <-
  tune_grid(ml_workflow,
            resamples = cv_splits,
            grid = glmn_grid,
            metrics = metric_set(roc_auc),
            control = ctrl)

# ----- Create Final Model -----
ml_model_fit <-
  ml_workflow %>%
  finalize_workflow(select_best(glmn_tune, metric = "roc_auc")) %>%
  fit(., data = train_data_glm)

# ----- Variable Importance -----
var_importance <-
  ml_model_fit %>%
  pull_workflow_fit() %>%
  vip(num_features = 15)
```

Variable Importance



Grid Search Results




```
rm(ctrl)
rm(cv_splits)
rm(glmn_grid)
rm(glmn_set)
rm(glmn_tune)

save.image(output_path)
```