10 Elastic Net GLM

Fabian Blasch

07/29/2022

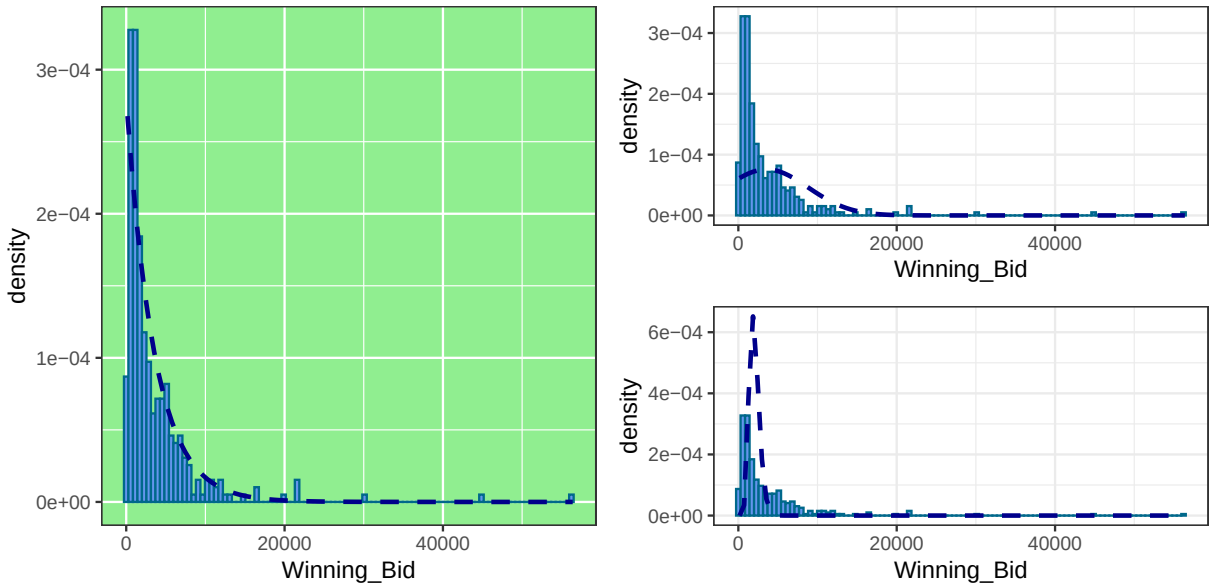
1 Load Data

```
# knitr options
knitr::opts_chunk$set(echo = TRUE,
                      fig.pos = "center",
                      fig.width = 8,
                      fig.height = 4,
                      fig.pos = "H")

# source AUX
source("./../Misc/Auxilliary.R")
source("./../Misc/model_eval.R")

# packages
get.package(c("lubridate", "glmnet", "glmnetUtils", "tidyverse", "patchwork",
              "selectiveInference"))

# load data
dat_aucs_eng <- readRDS("./../../Data/Bid Tab RDS/Aucs_df_feateng_split.RDS")

# data transformations
lapply(dat_aucs_eng, \(df){

  within(df, {

    Contract_ID <- NULL
    MLOT <- NULL
    EW_Diff <- NULL
    Winning_Bid <- Winning_Bid / 1e3
    Eng_Est <- Eng_Est / 1e3

  })

}) |> setNames(c("Train", "Test")) -> dat_aucs_mod
```

2 Auctions

2.1 GLM Link

```
# check fit
print(Dens_hist_plot(dat_aucs_mod[["Train"]], "Winning_Bid", dist = "exponential", distFUN = dexp, bins
Dens_hist_plot(dat_aucs_mod[["Train"]], "Winning_Bid", dist = "normal", distFUN = dnorm, bins = 100) /
Dens_hist_plot(dat_aucs_mod[["Train"]], "Winning_Bid", dist = "gamma", distFUN = dgamma, lower_b = 0.005
```



2.2 CV Function

```
CV_disglmnet <- \(formula, data, families, type_measure = "mse", nfolds = 5,
                  alpha = c(0.01, seq(0, 1, 0.25),0.99), nlambda = 250,
                  ncore = NULL, seed = 33){

  # set up parallel compute cluster
  if(is.null(ncore)){

    # set amount of cores to the max available and leave one out
    ncore <- parallel::detectCores() - 1

    # we parallelize over folds - the maximum number of occupied cores should thus be
    ncore <- min(ncore, length(families))

  } else {

    # find min of ncore and folds
    ncore <- min(ncore, length(families))
```

```
  }

  # set up cluster
  clust <- parallel::makeCluster(ncore, outfile = "")

  # print cores that will be occupied
  cat(paste0(length(clust), " cores will be occupied by this process!"))

  # set cluster enviroment to function enviroment
  parallel::clusterExport(cl = clust,
                          varlist = c("data"),
                          envir = environment())


  # loop over families
  parallel::parLapply(clust, families, \(fam){

    # seed
    set.seed(seed)

    # cross validation
    glmnetUtils::cva.glmnet(formula, data = data,
                            family = fam,
                            type.measure = type_measure, nfolds = nfolds,
                            alpha = alpha,
                            nlambda = nlambda)

  }) |> setNames(names(families)) -> tmp

  # release cores
  on.exit(parallel::stopCluster(clust), add = TRUE)

  # return
  return(tmp)
}
```

```
# find best model cv
tmp <- readRDS("./../../Data/Models/Glmnets/Raw/cvfiteng_norm.RDS")

glmnetUtils::cva.glmnet(Winning_Bid ~., data = dat_aucs_mod[["Train"]],
                        family = "gaussian",
                        type.measure = "deviance", nfolds = 10,
                        alpha = seq(0., 1, 0.05),
                        lambda = seq(5, 500, 5)) -> tmp
# to lst
tmp <- list(tmp)

# best
lapply(tmp, \(lst){

  # obtain model list from CV results
  do.call(rbind, Map(function(x, y){
```

```
    cbind("Per" = sqrt(x$cvm),
          "Lambda" = x$lambda,
          "Alpha" = rep(y, length(x$lambda)))

  }, lst$modlist, lst$alpha)) -> mod_lst

  # best model
  mod_lst[which.min(mod_lst[, "Per"]), ]

}) -> modlist

# best model
modlist <- do.call(rbind, modlist)
best_mod <- modlist[which.min(modlist[, "Per"]), ]

# fit model using best parameters
fit_fin <- glmnetUtils::glmnet(Winning_Bid ~., data = dat_aucs_mod[["Test"]],
                               family = gaussian(link = "identity"),
                               alpha = best_mod["Alpha"],
                               lambda = best_mod["Lambda"])

# predict
pred_vals <- predict(fit_fin, dat_aucs_mod[["Test"]])
```

2.3 Best Model

```
# load Cv results
cv_res <- readRDS("./../../Data/Models/Glmnets/Raw/CV_elastic_net.RDS")

# obtain best model
lapply(cv_res, \(lst){

  # obtain model list from CV results
  do.call(rbind, Map(function(x, y){

    cbind("Per" = sqrt(x$cvm),
          "Lambda" = x$lambda,
          "Alpha" = rep(y, length(x$lambda)))

  }, lst$modlist, lst$alpha)) -> mod_lst

  # best model
  mod_lst[which.min(mod_lst[, "Per"]), ]

}) -> modlist

# best model
modlist <- do.call(rbind, modlist)
best_mod <- modlist[which.min(modlist[, "Per"]), ]

# fit model using best parameters
```

```r
fit_fin <- glmnetUtils::glmnet(Winning_Bid ~., data = dat_aucs_mod[["Train"]],
                               family = "gaussian",
                               alpha = best_mod["Alpha"],
                               lambda = best_mod["Lambda"])

# predict
pred_vals <- predict(fit_fin, dat_aucs_mod[["Test"]])
```

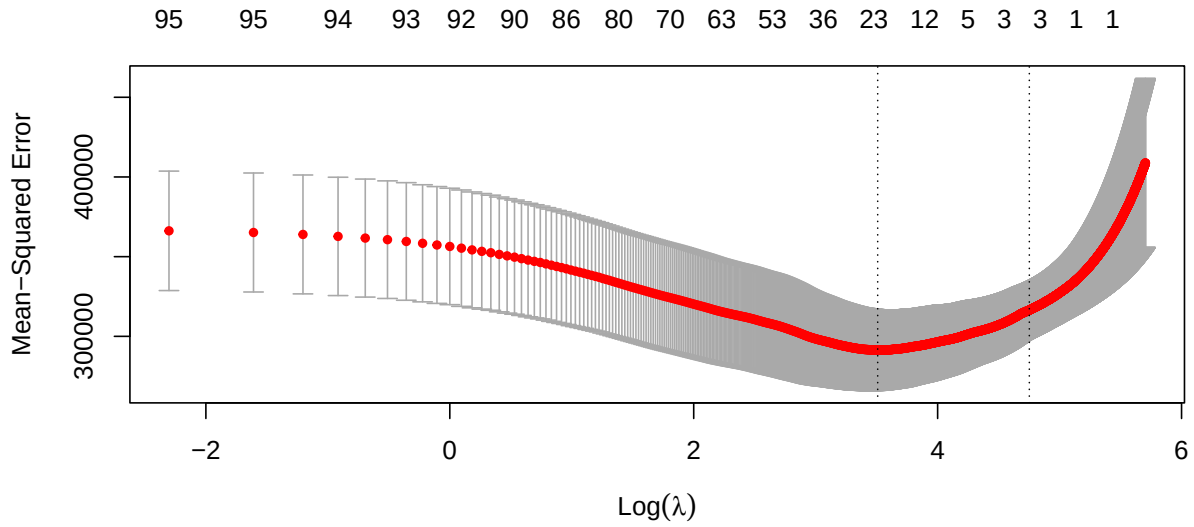
```r
# long dataset for boxplot
rbind(cbind(dat_aucs_mod[["Test"]][["Winning_Bid"]], "Act."),
      cbind(pred_vals, "Lasso Reg."),
      cbind(dat_aucs_mod[["Test"]][["Eng_Est"]], "Eng. Est."))|> as.data.frame()|>
      setNames(c("Award_Price", "Model")) -> dat_box


# change
within(dat_box,{
        Award_Price <- as.numeric(Award_Price)
        Model <- factor(Model, levels = c("Act.", "Lasso Reg.", "Eng. Est."))
        }) -> dat_box

# save data
# saveRDS(dat_box, "./../../Data/Misc Data/Figure_Data/Lasso_Performance_r2.RDS")
```

```r
# coefficients
coefft_auc <- coef(fit_fin, s = "lambda.min")
coef_ordered_auc <- coefft_auc[order(abs(coefft_auc[, 1]), decreasing = TRUE), ]
signum_auc <- coef_ordered_auc[-1] |> sign()
```

3 Selective Inference

```r
# rejoin data
dat <- do.call(rbind, dat_aucs_mod)

# split into label and model matirx
label <- dat[, "Winning_Bid"]
dat[, "Winning_Bid"] <- NULL

# model matrix
mod_mat <- model.matrix(~. + 0, data = dat)

# remove all vars with less then 10 entries
cs <- mod_mat |> colSums()

# mod mat sub
mod_mat <- mod_mat[, cs > 20]

# remove duplicate cols
duplicated.columns <- duplicated(t(mod_mat))
mod_mat <- mod_mat[, !duplicated.columns]
```

```r
# scale and standardize
mod_mat_sc <- scale(mod_mat, TRUE, TRUE)

# retune using the entire dataset
glmnet::cv.glmnet(x = mod_mat_sc, y = label,
                  standardize = FALSE, alpha = 1,
                  lambda = seq(0.1, 300, 0.1),
                  family = "gaussian", nfolds = 5) -> cv_res

# plot
plot(cv_res)
```




```r
# fit best model
glmnet::glmnet(x = mod_mat_sc, y = label, family = "gaussian",
               alpha = 1, lambda = cv_res$lambda.min,
               standardize = FALSE, tresh = 1e-25) -> fit_ent

# extract ceofs
beta <-  coef(fit_ent, x = mod_mat_sc, y = label,
              s = cv_res$lambda.min / nrow(dat), exact = TRUE,
              tresh = 1e-25)[-1]

# estimate sigma
sigma <- estimateSigma(x = mod_mat_sc, y = label, intercept = TRUE)

# selective inference
pselInf <- fixedLassoInf(x = mod_mat_sc, y = label, beta = beta,
                         lambda = cv_res$lambda.min)

# p vals
pvals <- pselInf$pv
names(pvals) <- colnames(mod_mat)
```

```r
pvals <- sort(pvals, decreasing = FALSE)

# return
knitr::kable(pvals[pvals < 0.1], col.names = "p-value")
```

	p-value
Eng_Est	0.0000000
Vend_005A	0.0086086
street	0.0253774
Vend_1275A	0.0600738
Letting_Month3	0.0649190
deck	0.0751292
Letting_Month8	0.0816437
sidewalk	0.0824628
Vend_757A	0.0923069