PS5 Solutions

Teaching Staff

2023-03-01

A helper function for regression output because `broom::tidy()` is not allowed

```
regTable = function(model){
  m = summary(model)
  tidydf = data.frame(terms = names(m[["coefficients"]][,1]),
                      estimates = round(m[["coefficients"]][,1],4),
                      std.error = m[["coefficients"]][,2],
                      statistic = m[["coefficients"]][,3],
                      p.value = m[["coefficients"]][,4],
                      row.names = NULL)

  ## Round the numeric columns to 4 digits
  tidydf[, 2:ncol(tidydf)] = apply(tidydf[, 2:ncol(tidydf)],
                                   2,
                                   function(x) round(x, 4)
                                   )
  return(tidydf)
}
```

Question 1

```
load("../data/CreditClaim.RData")
x = credit_claim$x
y = credit_claim$y
```

What proportion of the documents are claiming credit?

```
mean(y)
```

```
## [1] 0.2584693
```

2

```
dim(x)
```

```
## [1]  797 7587
```

There are roughly 10 times the number of predictors than the number of observations. The model matrix will thus be rank deficient and the resulting equation has an infinite number of solutions.

Practically speaking, in an OLS regression, R will arbitrarily drop a lot of character string predictors in order to deal with this problem. In the logistic regression, R will hang for ages before doing something similar.

3

```
common_words = colMeans(x)
cols = names(common_words[order(common_words, decreasing = T)][1:20])
cols
```

```
##  [1] "congress"    "million"     "energy"      "funding"     "legislation"
##  [6] "release"     "american"    "byline"      "dateline"    "national"
## [11] "fed"         "contact"     "support"     "care"        "tax"
## [16] "help"        "government"  "security"    "people"      "president"
```

4

```
top20 = x[,cols]
m = glm(y ~ top20, family = "binomial")
```

terms	estimates	std.error	statistic	p.value
(Intercept)	-1.6320	0.3447	-4.7350	0.0000
top20congress	-0.2898	0.1001	-2.8944	0.0038
top20million	0.3185	0.0782	4.0752	0.0000
top20energy	-0.0097	0.0410	-0.2369	0.8127
top20funding	0.4883	0.0714	6.8341	0.0000
top20legislation	-0.1587	0.0825	-1.9230	0.0545
top20release	0.6241	0.2149	2.9037	0.0037
top20american	-0.0913	0.0751	-1.2159	0.2240
top20byline	0.7538	0.5111	1.4749	0.1402
top20national	0.0671	0.0690	0.9715	0.3313
top20fed	-0.8841	0.4715	-1.8751	0.0608
top20contact	0.0961	0.1888	0.5087	0.6110
top20support	-0.0080	0.0950	-0.0844	0.9328
top20care	-0.2083	0.0788	-2.6440	0.0082
top20tax	0.1177	0.0515	2.2873	0.0222
top20help	0.4535	0.0934	4.8560	0.0000
top20government	-0.3914	0.1247	-3.1377	0.0017
top20security	-0.0957	0.0748	-1.2787	0.2010
top20people	-0.2972	0.1343	-2.2128	0.0269
top20president	-0.5484	0.1607	-3.4128	0.0006

4 Bonus

We have at least two columns that are linear combinations of each other. R consequently drops one at "random" to fit the model.

In R, the dropped arguments tend to follow each other. In this case byline and dateline have the same value for every observation. This is unsurprising because dateline likely indicates when the release came out and byline indicates who wrote it, which is likely to be in every one of these documents.

```
mean(top20[,c("byline")] == top20[,c("dateline")])
```

```
## [1] 1
```

5-6

```
insamp_error = mean(m$model$y != as.numeric(m$fitted.values > .5))
insamp_error
```

```
## [1] 0.1794228
```

7

```
## It will be helpful to set the warning=F argument

predictions = vector(mode = "logical",
                     length = length(y))
loo.dat = as.data.frame(cbind(y, top20))
for(i in 1:length(y)){
  tmp = loo.dat[-i,]
  m = glm(y ~ ., data = tmp,
          family = "binomial")
  predictions[i] = predict(m,
                           newdata = loo.dat[i,],
                           type = "response")
}
yhat.cv = as.numeric(predictions > 0.5)
```

8

```
mean(y != yhat.cv)
```

```
## [1] 0.1882058
```

9

The classification error is better compared to the out of sample fit, which is unsurprising because the in sample fit uses all observations. It also suggests that the in-sample fit is in some sense overfitting the data.

Question 2

1

```
load("../data/CreditClaim.RData")
x = credit_claim$x
y = credit_claim$y
n.total = length(y)
prop.train = 0.7
set.seed(54321)
r = sample(1:n.total,
           round(prop.train*n.total),
           replace = FALSE)
x.train = x[r,]
x.test = x[-r,]
y.train = y[r]
y.test = y[-r]
```

2

```
set.seed(123)
cv.results = cv.glmnet(x=x.train,
                       y = y.train,
                       family = "binomial",
                       nfolds = 5,
                       alpha = 1)
```

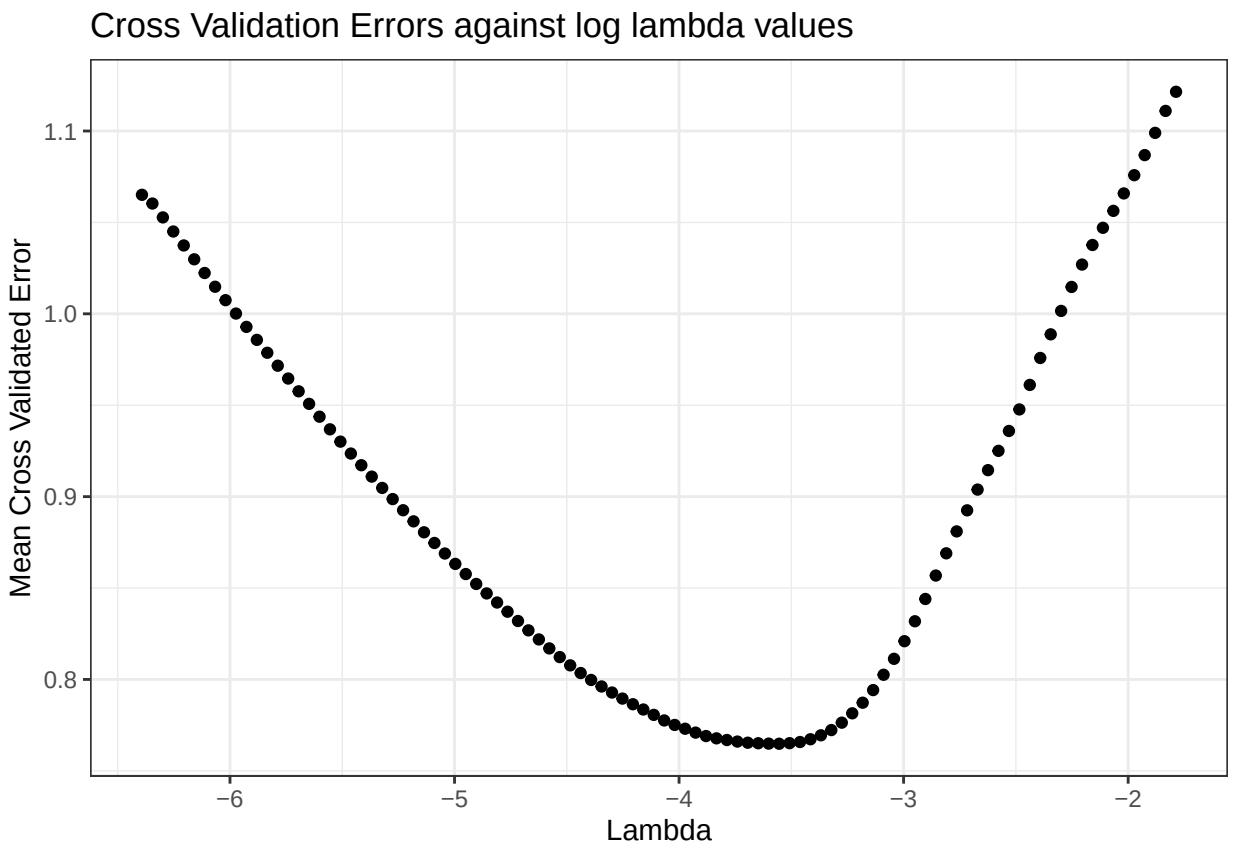
`cv.glmnet()` performs k-fold cross validation on a logistic regression learning model (family = "binomial"). The response variable y is identical to Question 1's outcome variable. There are 7587 predictors. `nfolds=5` indicates performing 5 cross validation folds. `alpha=1` indicates we're running a lasso regression.

3

100 were fit. The model fit `{r length(cv.results$lambda *5}` models. By default, the type measure is binomial deviance.

4

```
data.frame(x = log(cv.results$lambda), y = cv.results$cvm) |>
  ggplot(aes(x,y))+
  geom_point()+
  labs(title="Cross Validation Errors against log lambda values",
       x = "Lambda",
       y = "Mean Cross Validated Error")+
  theme_bw()
```



5

```
# if you wanted to do it by hand for some reason
# cv.results$lambda[which.min(cv.results$cvm)]
cv.results$lambda.min
```

```
## [1] 0.02859915
```

6

```
bestlambda = cv.results$lambda.min
m2 = glmnet(x = x.train, y = y.train,
            family = "binomial",
            lambda = bestlambda,
            alpha = 1)
summary(m2)
```

```
##            Length Class     Mode
## a0            1   -none-    numeric
## beta       7587   dgCMatrix S4
## df            1   -none-    numeric
## dim           2   -none-    numeric
## lambda        1   -none-    numeric
## dev.ratio     1   -none-    numeric
## nulldev       1   -none-    numeric
## npasses       1   -none-    numeric
## jerr          1   -none-    numeric
## offset        1   -none-    logical
## classnames    2   -none-    character
## call          6   -none-    call
## nobs          1   -none-    numeric
```

Now we find the non zero coefficients

```
lasso.coefs = predict(m2,
                      type = "coefficients",
                      s = bestlambda)

### it'll be 89 with the intercept
length(lasso.coefs[lasso.coefs != 0]) - 1
```

```
## <sparse>[ <logic> ]: .M.sub.i.logical() maybe inefficient
```

```
## [1] 88
```

```
### alternatively
sum(m2$beta != 0)
```

```
## [1] 88
```

Bonus

```
coef(m2)@Dimnames[[1]][which(coef(m2) !=0)]
```

```
##  [1] "(Intercept)"     "aeronautics"     "agricultures"    "alternatives"
##  [5] "announce"        "announced"       "announces"       "appropriations"
##  [9] "arsenal"         "base"            "brac"            "buster"
```

```
## [13] "byers"           "cazayoux"        "cents"           "chamber"
## [17] "childs"          "common"          "communitys"      "congress"
## [21] "contributions"   "doe"             "dot"             "ears"
## [25] "endorsed"        "equipment"       "exhaust"         "expand"
## [29] "faa"             "facility"        "fairpoint"       "firefighter"
## [33] "fostering"       "funding"         "gillies"         "grant"
## [37] "harder"          "homeport"        "linda"           "loophole"
## [41] "manufacture"     "maryland"        "milk"            "nation"
## [45] "neighborhoods"   "nfip"            "park"            "patty"
## [49] "political"       "pollution"       "portland"        "postcard"
## [53] "preserve"        "programming"     "project"         "prosecutors"
## [57] "questions"       "rain"            "reaffirms"       "reauthorization"
## [61] "reductions"      "regarding"       "regional"        "reinvestment"
## [65] "rent"            "residential"     "river"           "rmt"
## [69] "rutgers"         "schultz"         "secured"         "sediment"
## [73] "shoring"         "simultaneously"  "smart"           "spare"
## [77] "stand"           "station"         "stimulating"     "streets"
## [81] "tag"             "tested"          "treasure"        "tricare"
## [85] "understand"      "urban"           "venture"         "watersheds"
## [89] "weatherization"
```

7

```
mean(y.test != (as.numeric(predict(m2,
                                   s = bestlambda,
                                   type = "response",
                                   newx = x.test)) > 0.5))
```

```
## [1] 0.2175732
```

```
replicates = 200
results = vector(mode ="logical", length = replicates)
N = nrow(x.train)
l = cv.results$lambda.1se

set.seed(567)

for(i in 1:replicates){
  k = sample(N, N,replace = TRUE)
  x.tmp = x.train[k,]
  y.tmp = y.train[k]
  m = glmnet(x=x.train[k,], y = y.train[k],
             family = "binomial", lambda = l,
             alpha = 1)
  results[i] = predict(m, s = l, type = "response",
                       newx = x.test[1,,drop =FALSE])
}

quantile(results, probs = c(0.025, 0.975))
```

8

```
##      2.5%     97.5%
## 0.1105388 0.3930125
```