Laboratório 1 - Validação Cruzada e Seleção de Variáveis

ME905

Instruções

- Esta atividade contém duas partes com o mesmo peso na avaliação.
- Junto do código em cada item, descreva em forma de texto o método sendo utilizado e discuta os resultados. Apresente equações se achar necessário.
- Apresente respostas completas e apresente gráficos se achar necessário.
- A menos quando especificado, evite utilizar funções "prontas" para tarefas que podem ser feitas utilizando a sintaxe básica do R. Por exemplo, a separação dos bancos de dados em treino e teste deve ser implementada sem funções de pacotes.

Parte 1 - Seleção de Variáveis

O conjunto de dados `l1d1.csv` contém informações de 300 variáveis (`x001` a `x300`) e uma variável resposta `y` (contínua) para 5000 observações. O objetivo dessa parte é apresentar um modelo de regressão linear (perda quadrática) com algum subconjunto das 300 variáveis disponíveis.

(1) Obtenha um conjunto de variáveis com efeito significativo na resposta com base em testes de hipóteses. Faça os ajustes necessários (escolha um dos métodos) para controlar o número de variáveis selecionadas.

Utilizando a correção de Bonferroni, temos que $\alpha^* = \frac{\alpha}{300}$:

```
# configuração global
set.seed(282829)

# leitura dos dados
db <- read_csv("l1d1.csv")
```

```
## Rows: 5000 Columns: 301
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (301): y, x001, x002, x003, x004, x005, x006, x007, x008, x009, x010, x0...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
model <- lm(y ~., data = db)

ajuste <- summary(model)
ajuste$coefficients[ajuste$coefficients[,4] < 0.05/300, 4]
```

```
##         x030         x057         x059         x064         x103         x113
## 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##         x118         x127         x208         x212         x223         x226
## 0.000000e+00 0.000000e+00 5.223372e-92 0.000000e+00 0.000000e+00 0.000000e+00
```

```
##         x259         x271         x295
## 0.000000e+00 0.000000e+00 0.000000e+00
```

```
model_reajustado <- lm(y ~ ., data = db[c(1, which(ajuste$coefficients[,4] < 0.05/300))])
summary(model_reajustado)
```

```
##
## Call:
## lm(formula = y ~ ., data = db[c(1, which(ajuste$coefficients[,
##     4] < 0.05/300))])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3520 -0.6780 -0.0125  0.6507  3.7602
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.112723   0.281760     0.4    0.689
## x030        -1.298738   0.005698  -227.9   <2e-16 ***
## x057         7.415723   0.005251  1412.3   <2e-16 ***
## x059        -2.983097   0.009441  -316.0   <2e-16 ***
## x064         9.884747   0.006187  1597.8   <2e-16 ***
## x103         6.811676   0.021596   315.4   <2e-16 ***
## x113         7.198481   0.007182  1002.3   <2e-16 ***
## x118         5.695612   0.004848  1174.9   <2e-16 ***
## x127         5.229726   0.013846   377.7   <2e-16 ***
## x208         1.172738   0.007298   160.7   <2e-16 ***
## x212         7.673801   0.004875  1574.2   <2e-16 ***
## x223         5.785250   0.019126   302.5   <2e-16 ***
## x226         5.895854   0.004916  1199.4   <2e-16 ***
## x259        -2.315402   0.018251  -126.9   <2e-16 ***
## x271         7.746528   0.021080   367.5   <2e-16 ***
## x295         2.255074   0.011928   189.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9939 on 4984 degrees of freedom
## Multiple R-squared:  0.9996, Adjusted R-squared:  0.9996
## F-statistic: 9.313e+05 on 15 and 4984 DF,  p-value: < 2.2e-16
```

(2) Implemente o método *Forward Stepwise Regression* e obtenha o conjunto de variáveis que minimiza o erro de predição sob perda quadrática para um conjunto de dados de teste.

O método forward stepwise regression baseia-se em buscar o melhor subconjunto de variáveis (de acordo com alguma métrica, como EQM) de maneira recursiva (ou gulosa), sendo assim calculamos o melhor modelo com um parâmetro, depois o melhor modelo com dois parâmetros dado o melhor modelo com um parâmetro, e assim por diante. A separação do conjunto de dados entre treino e teste e a avaliação do erro de predição auxilia na escolha da melhor quantidade de variáveis dentre os melhores subconjuntos.

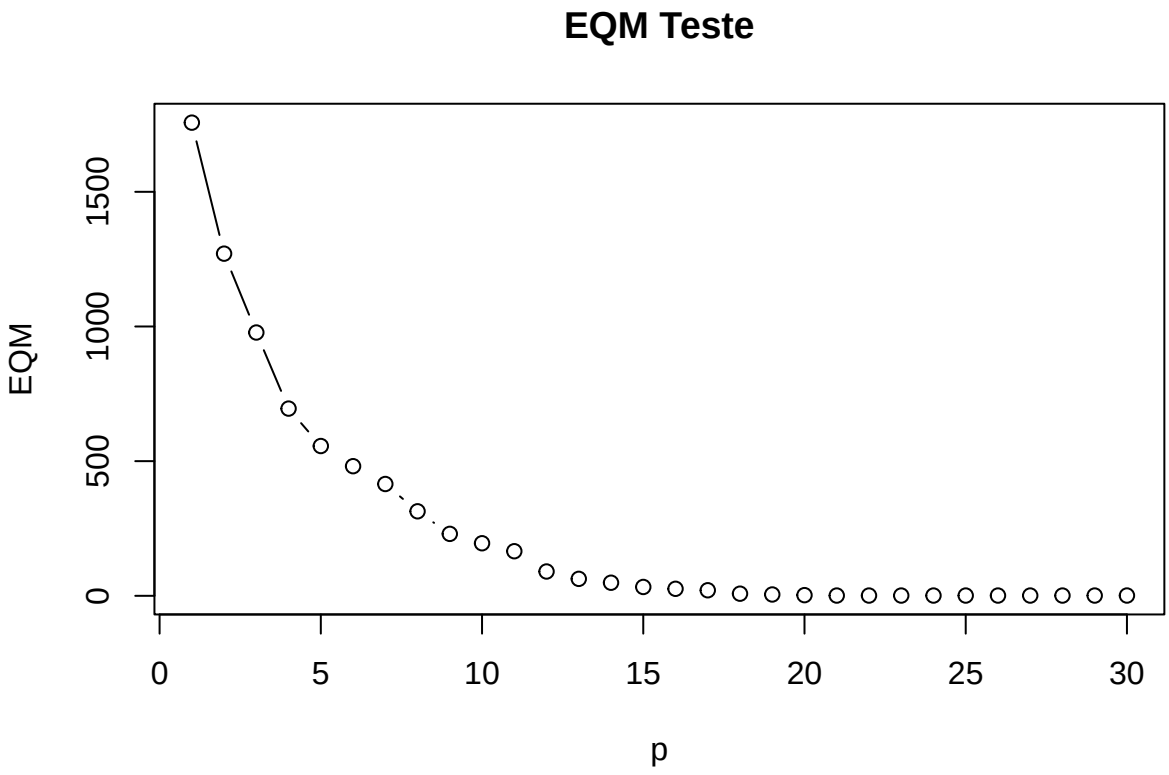
```
# data_treino - banco de dados para treino
# data_teste - banco de dados para teste
# p_max - quantidade de preditoras máxima no último modelo ajustado

fsr <- function(data_treino, data_teste, p_max = 30) {
  # objetos
  pred_selecionado <- character(p_max)   # vetor com o nome dos pred selecionados
```

```r
  EQM_teste <- numeric(p_max)             # salvar EQM minimo por iteração
  EQM_treino_min <- numeric(p_max)        # salva o EQM treino

  names_pred_all <- names(data_treino[-1]) # nome de todas as preditoras

  for (p in 1:p_max) {
    names_pred <- setdiff(names_pred_all, pred_selecionado)  # preditores não selecionados
    EQM_treino <- vector(mode='numeric', length(names_pred))
    j <- 1

    for (i in names_pred) {  # iterando nos preditores não selecionados
      formula <- formula(paste('y ~', paste(c(pred_selecionado[1:p], i),
                                            collapse = ' + ')))
      EQM_treino[j] <- mean(lm(formula, data_treino)$residuals^2)
      j <- j + 1
    }

    EQM_treino_min[p] <- min(EQM_treino)
    pred_selecionado[p] <- names_pred[which.min(EQM_treino)]
    best_fit <- lm(y ~ ., data_treino[c('y', pred_selecionado[1:p])])
    EQM_teste[p] <- sum((data_teste$y - predict(best_fit, newdata =
                                                data_teste))^2)/nrow(data_teste)
  }

  result <- data.frame(p = 1:p_max, EQM_treino = EQM_treino_min,
                       EQM_teste = EQM_teste, pred = pred_selecionado)
  return(list('preditores' = pred_selecionado, 'result' = result))
}

shuffle <- sample(rep(c(1,2), times = c(4000, 1000)))

obj <- fsr(db[shuffle == 1,], db[shuffle == 2,])

knitr::kable(obj$result)
```

p	EQM_treino	EQM_teste	pred
1	1697.9864323	1756.8271124	x236
2	1223.2045558	1270.4205592	x212
3	944.6443592	977.5856819	x226
4	690.5397761	695.0966014	x118
5	526.6260927	556.1578959	x087
6	441.3524729	481.2647133	x178
7	386.5778072	415.1240009	x061
8	307.3298423	313.7195942	x064
9	223.0065525	229.9564711	x131
10	188.9399728	195.1298924	x127
11	163.0179275	165.5145475	x057
12	89.3436809	90.2573583	x113
13	60.1299451	62.9018714	x103
14	45.5531663	48.5920253	x059
15	30.6974049	32.7963480	x030
16	23.7835683	26.0094263	x238
17	19.2843143	20.5878424	x271

p	EQM_treino	EQM_teste	pred
18	7.9563694	7.8506758	x223
19	5.1575844	4.9273169	x259
20	2.6112048	2.4374841	x295
21	0.9849345	0.9802246	x208
22	0.9830769	0.9826228	x181
23	0.9815091	0.9817856	x195
24	0.9803239	0.9818235	x280
25	0.9791900	0.9837741	x234
26	0.9777969	0.9853574	x124
27	0.9762756	0.9882611	x097
28	0.9751167	0.9901200	x145
29	0.9738052	0.9944822	x250
30	0.9724874	0.9978617	x170

```
plot(obj$result$p, obj$result$EQM_teste, xlab = 'p', ylab = 'EQM', type = 'b',
     main = 'EQM Teste')
```

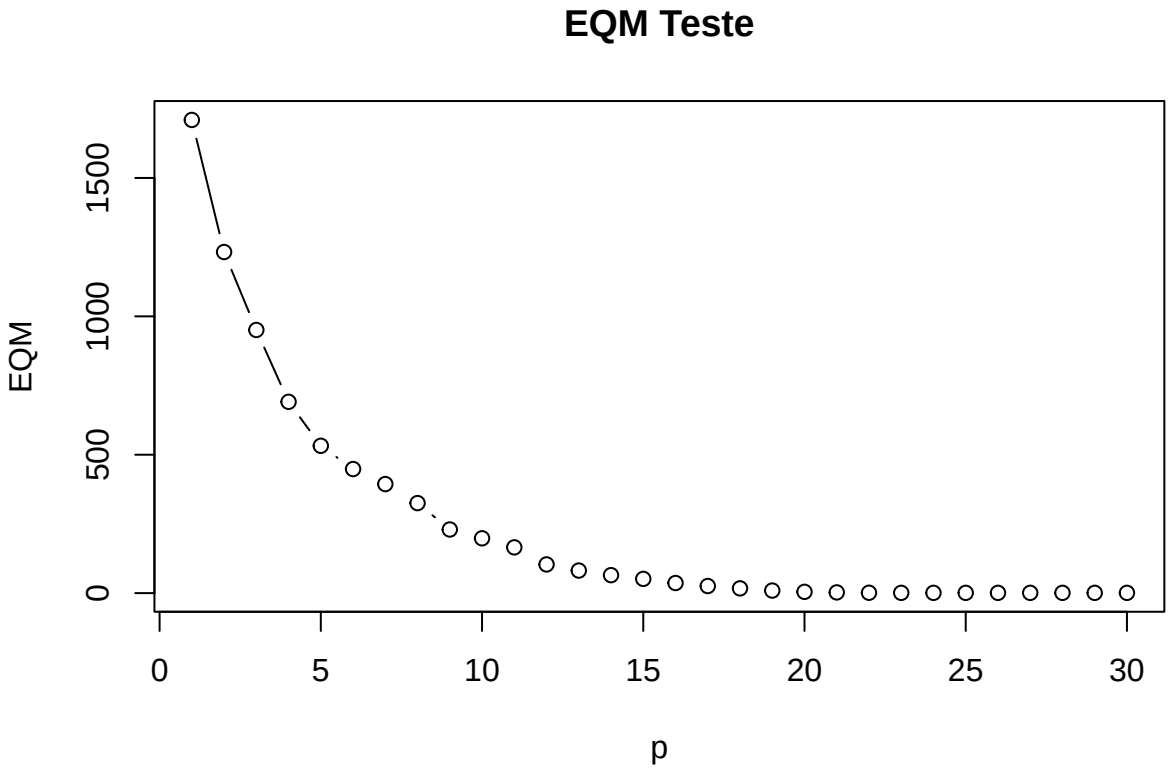


(3) Refaça o item 2 utilizando o método de validação cruzada k-fold, com k = 5.

Na validação cruzada o objetivo continua sendo operacionalizar a escolha do melhor número de variáveis dentre os melhores subconjuntos, tendo em vista predição; com a diferença que ao invés de confiar em apenas uma medida de um split, obtemos a média de 5 diferentes medidas de erro de predição vindas de 5 partições nos dados, por exemplo.

```
# data - banco de dados
# p_max - quantidade de preditoras máxima no último modelo ajustado
# k - quantidade de folds
```

```r
fsr_cv <- function(data, p_max = 30, k = 5) {
  resultado <- vector(mode = 'list', length = k)
  shuffle <- sample(1:k, size = nrow(data), replace = T)

  for (fold in 1:k) {
    resultado[[fold]] <- fsr(data_treino = data[shuffle != fold,],
                             data_teste = data[shuffle == fold,], p_max = p_max)
  }

  return(resultado)
}

a <- fsr_cv(db)

EQM <- data.frame(p = 1:30, EQM = rowMeans(sapply(a, \(x) x$result$EQM_treino)))

plot(EQM$p, EQM$EQM, xlab = 'p', ylab = 'EQM', main = 'EQM Teste', type = 'b')
```



(4) Com base nos métodos discutidos e nos resultados obtidos, qual subconjunto das 300 variáveis você diria que possuem um efeito não nulo na resposta y?

Analisando a tabela e gráficos anterioriores, determinou-se que a quantidade ideal de preditoras é 13. Então, buscaremos o modelo com 13 variáveis utilizando foward stepwise regression em todo o dataset.

```r
fsr_total <- function(data, p_max) {
  # objetos
  pred_selecionado <- character(p_max)    # vetor com o nome dos pred selecionados
  EQM_teste <- numeric(p_max)             # salvar EQM minimo por iteração
  names_pred_all <- names(data[-1])       # nome de todas as preditoras
```

```
  for (p in 1:p_max) {
    names_pred <- setdiff(names_pred_all, pred_selecionado) # preditores não selecionados
    EQM <- vector(mode='numeric', length(names_pred))
    j <- 1

    for (i in names_pred) {  # iterando nos preditores não selecionados
      formula <- formula(paste('y ~', paste(c(pred_selecionado[1:p], i),
                                            collapse = ' + ')))
      EQM[j] <- mean(lm(formula, data)$residuals^2)
      j <- j + 1
    }

    pred_selecionado[p] <- names_pred[which.min(EQM)]
    best_fit <- lm(y ~ ., data[c('y', pred_selecionado[1:p])])
  }

  result <- data.frame(p = 1:p_max, pred = pred_selecionado)
  return(pred_selecionado)
}

obj2 <- fsr_total(db, 13)
```

Com isso temos o conjunto de variáveis com efeito não nulo na resposta:

```
obj2
```

```
##  [1] "x236" "x212" "x226" "x118" "x087" "x178" "x061" "x064" "x131" "x127"
## [11] "x057" "x113" "x103"
```

Modelo final obtido:

```
lm(formula(paste('y ~', paste(obj2, collapse=' + '))), db) |> summary()
```

```
##
## Call:
## lm(formula = formula(paste("y ~", paste(obj2, collapse = " + "))),
##     data = db)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.6602  -5.2482  -0.0118   5.3080  28.0936
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -33.84989    2.14774 -15.761  < 2e-16 ***
## x236         -0.95628    0.03848 -24.850  < 2e-16 ***
## x212          7.60454    0.03822 198.961  < 2e-16 ***
## x226          6.02129    0.03841 156.784  < 2e-16 ***
## x118          5.69540    0.03804 149.720  < 2e-16 ***
## x087         -1.09506    0.10515 -10.415  < 2e-16 ***
## x178          1.56691    0.06668  23.500  < 2e-16 ***
## x061          1.37031    0.10380  13.202  < 2e-16 ***
## x064          9.00076    0.07064 127.423  < 2e-16 ***
## x131          0.44060    0.11364   3.877 0.000107 ***
## x127          7.07626    0.10997  64.348  < 2e-16 ***
## x057          5.84236    0.06362  91.825  < 2e-16 ***
```

```
## x113          6.88693    0.09236  74.568  < 2e-16 ***
## x103          8.11587    0.16663  48.705  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.795 on 4986 degrees of freedom
## Multiple R-squared:  0.9781, Adjusted R-squared:  0.978
## F-statistic: 1.709e+04 on 13 and 4986 DF,  p-value: < 2.2e-16
```

Parte 2 - LASSO

Para essa parte, utilize a função `glmnet` do pacote `glmnet` para realizar ajustes utilizando o método LASSO.

(1) Leia a documentação da função `glmnet` e a vignette disponível em https://glmnet.stanford.edu/articles/glmnet.html. Descreva os principais parâmetros da função que serão necessários para ajustar um modelo baseado em LASSO para um determinado conjunto de dados.

x: matriz de input de dimensão $n \times p$. Necessário que $p \geq 2$.

y: variável resposta. o parâmetro `family` pode ser usado para especificar se a variável é quantitativa, de contagem, entre outros.

alpha: faz uma "mistura" entre Lasso (`alpha = 1`) e Ridge (`alpha = 0`).

lamdda: parâmetro de penalização do LASSO. Quanto maior o seu valor, mais os coeficientes são encolhidos.

(2) Separe 10% do seu conjunto de dados como um conjunto de dados de teste. Com os 90% restante (conjunto de treino), ajuste uma regressão com LASSO considerando $\lambda = 2$. Calcule o Erro Quadrático Médio de predição para o conjunto de treino e o conjunto de teste. Quantas variáveis tiveram coeficientes não nulos para este ajuste?

Na regressão LASSO incluímos um termo de penalização de norma 1 na função de loss: $\lambda \sum_{j=1}^{p} |\beta_j|$. De maneira que os coeficientes dos parâmetros são encolhidos, e muitas vezes até zerados, auxiliando na predição e na seleção de variáveis. Para $\lambda = 2$ fixado temos:

```r
# separando dados
# 0.9 * nrow(db)  # 90% = 4500 linhas
# 0.1 * nrow(db)  # 10% =  500 linhas
#set.seed(123)
shuffle <- sample(rep(c(1,2), times = c(4500, 500)))
treino <- db[shuffle == 1,]
teste <- db[shuffle == 2,]

# ajustando modelo
modelasso <- glmnet(x = as.matrix(treino[,-1]), y = as.matrix(treino[,1]), alpha = 1, lambda = 2)

# predições
pred_treino <- predict(modelasso, newx = as.matrix(treino[,-1]), s = 2)
pred_teste <- predict(modelasso, newx = as.matrix(teste[,-1]), s = 2)

# EQM
eqm_treino <- mean((as.matrix(treino[,1]) - pred_treino)^2)
eqm_teste <- mean((as.matrix(teste[,1]) - pred_teste)^2)

# quantidade de variáveis com coeficientes não nulos
qnt_var_coef_nao__nulo <- sum(coef(modelasso) != 0)
```

O erro quadrático médio de predição para o conjunto de dados de treino foi 144.0578656, e para o conjunto de dados de teste foi 144.0578656. a quantidade de variáveis com coeficiente não nulo foi 22.

(3) Escolha (pelo menos) 10 valores de λ. Para cada valor de λ, refaça o item (2), com os mesmos conjuntos de treino/teste. Compare os erros de predição no teste para cada valor de λ. Qual valor de λ produziu o menor erro de predição?

```
lambda <- c(0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.75, 1, 1.5, 3)
pred_treino_grid <- matrix(0, nrow = nrow(treino), ncol = length(lambda))
pred_teste_grid <- matrix(0, nrow = nrow(teste), ncol = length(lambda))
eqm_treino_grid <- numeric(10)
eqm_teste_grid <- numeric(10)

for (i in 1:length(lambda)) {
  fit <-  glmnet(x = as.matrix(treino[,-1]), y = as.matrix(treino[,1]),
                 alpha = 1, lambda = lambda[i])
  pred_treino_grid[,i] <- predict(fit, newx = as.matrix(treino[,-1]), s = lambda[i])
  pred_teste_grid[,i] <- predict(fit, newx = as.matrix(teste[,-1]), s = lambda[i])

  eqm_treino_grid[i] <- mean((as.matrix(treino[,1]) - pred_treino_grid[,i])^2)
  eqm_teste_grid[i] <- mean((as.matrix(teste[,1]) - pred_teste_grid[,i])^2)
}

grid <- cbind(eqm_treino_grid, eqm_teste_grid, lambda)
min_eqm_lambda_treino <- lambda[which.min(eqm_treino_grid)]
min_eqm_lambda_teste <- lambda[which.min(eqm_teste_grid)]
```

O lambda que produziu o menor erro de predição foi o $\lambda = 0.01$ que minimiza o erro para o conjunto de teste (o mais importante para avaliar performance de predição), e coincidentemente também o de treino. o erro obtido foi de 1.0330297 no conjunto de teste.

(4) Considere os dados no arquivo `l1d1-val.csv`. Este conjunto de dados não possui o valor da variável resposta. Gere um arquivo chamada `l1-pred-[grupo].csv` contendo as predições para as 1000 observações disponíveis no arquivo, com base no ajuste que você considera mais apropriado para fazer predições. Substitua `[grupo]` pela letra associada ao grupo no Moodle.

```
db_val <- read_csv("l1d1-val.csv")
```

```
## Rows: 1000 Columns: 300
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (300): x001, x002, x003, x004, x005, x006, x007, x008, x009, x010, x011,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
best_modelasso <- glmnet(x = as.matrix(treino[,-1]), y = as.matrix(treino[,1]),
                         alpha = 1, lambda = 0.01)
pred_best_modelasso <- predict(best_modelasso, newx = as.matrix(db_val), s = 0.01)

write.csv(pred_best_modelasso, "l1-pred-J.csv")
```