Lista 1 - Mineração

Victor Alves Dogo Martins, RA: 744878 Ana Beatriz Alves Monteiro, RA: 727838
Larissa Torres, RA: 631914

03-07-2022

Item 1

Como orientado no enunciado deste item, foi feita a normalização da covariável 'PIB per capita' através da seguinte fórmula:

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

Computacionalmente, esse procedimento foi feito através do comando `mutate` abaixo, onde temos o comentário `# Normalizando a covariavel`. Além disso, também segue uma parcela do banco de dados após a normalização para fins de demonstração:

```
### Carregando Pacotes

library(tidyverse)
library(knitr)
library(kableExtra)

# Lendo dados

df <- readr::read_csv('worldDevelopmentIndicators.csv') |>
  select(-CountryName) |>
  rename(y=LifeExpectancy,
         x=GDPercapita) |>
  mutate(x = (x-min(x))/(max(x)-min(x))) # Normalizando a covariavel

# Mostrando dados

head(df) |>
  kable('latex',digits=4, align='cc',
        caption = 'Primeiras linhas do banco de dados após normalização') |>
  kable_styling(position="center",
                latex_options="HOLD_position")
```

Table 1: Primeiras linhas do banco de dados após normalização

y	x
60.5091	0.0042
51.4640	0.0505
77.3505	0.0362
69.9497	0.0710
76.9579	0.4000
76.0127	0.1093

Item 2

Para realizarmos esta etapa da lista, como explicitado no enunciado, utilizaremos o método de validação *leave-one-out cross validation*. Em outras palavras, o estimador do risco será dado por:

$$\hat{R}(g) = \frac{1}{n} \cdot \sum_{i=1}^{n} (Y_i - g_{-i}(X_i))^2$$

Onde g_{-i} é o modelo ajustado para todas as observações menos a i-ésima delas.

Além disso, para a construção das fórmulas de cada um dos modelos (com $p = 1, 2, \ldots, 30$), foi utilizado o comando `paste()` para que elas não fossem escritas à mão, como foi devidamente anotado no código abaixo:

```
# Criando lista com formulas de g(x) para cada valor de p

formulas <- list()

for (p in 1:30) {

  if(p==1){

    # Se p for igual a 1, a expressao mantem-se da forma abaixo

    formulas[[p]] <- "y~sin(2*pi*x)+cos(2*pi*x)"
  } else {

    # Para cada p maior do que 1, sua expressao g(x) sera dada pela expressao
    # do p anterior MAIS os seno e cosseno de 2*pi*x vezes o valor de p da
    # iteracao atual

    formulas[[p]] <- paste0(formulas[[p-1]],
                            "+sin(2*",p,
                            "*pi*x)+cos(2*",
                            p, "*pi*x)")
  }
}

# Definido tibble para guardarmos estimativas do risco das regressões de cada
# valor de p

tbl_result <- tibble(
  p=1:30,
```

```
  mse=as.double(1:30)
)

# Calculando erro quadrático médio para cada p via leave-one-out

for (p in 1:length(formulas)) {

  erros <- NULL

  model <- NULL

  for (ii in 1:nrow(df)) {

    # Para o p atual, ajusta-se uma regressao retirando cada obs.
    # ii e utilizando o restante do banco de dados para validarmos
    # o modelo com a obs. ii retirada

    model <- lm(formulas[[p]], data=df[-ii,])

    erros <- c(erros,
               (df[ii,1] - predict(model, df[ii,]))^2)
  }

  # Fazendo a media dos erros dos ajustes para cada uma das observacoes
  # retiradas

  tbl_result[p,2] <- mean(unlist(erros))

}

# Mostrando inicio dos erros

head(tbl_result) |>
  kable('latex',digits=4, align='cc',
        caption = 'Primeiras linhas dos riscos estimados para cada p',
        col.names=c('P', 'MSE')) |>
  kable_styling(position="center",
                latex_options="HOLD_position")
```

Table 2: Primeiras linhas dos riscos estimados para cada p

P	MSE
1	51.2822
2	46.1098
3	42.9456
4	44.4956
5	43.0823
6	54.7683

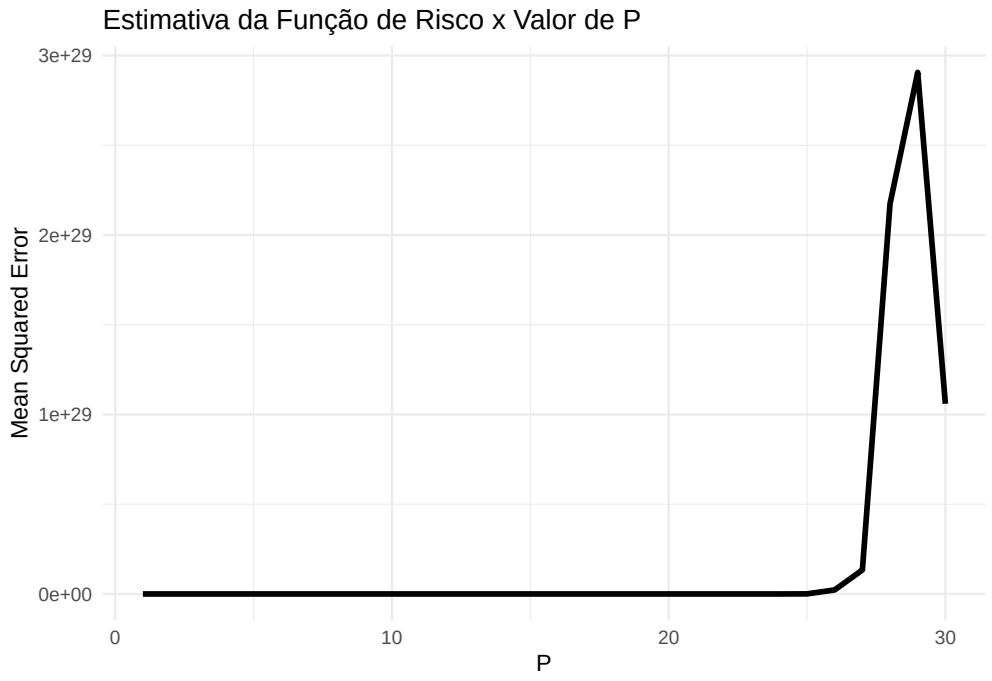
Como pode ser visto acima, o cálculo das estimativas dos riscos procedeu como esperado, com o modelo onde $p = 3$ aparentando ser o mais promissor, com pouca diferença do modelo onde $p = 5$. A seguir, iremos

visualizar graficamente as estimativas de forma a compreender melhor qual dos modelos ajustados foi o melhor.

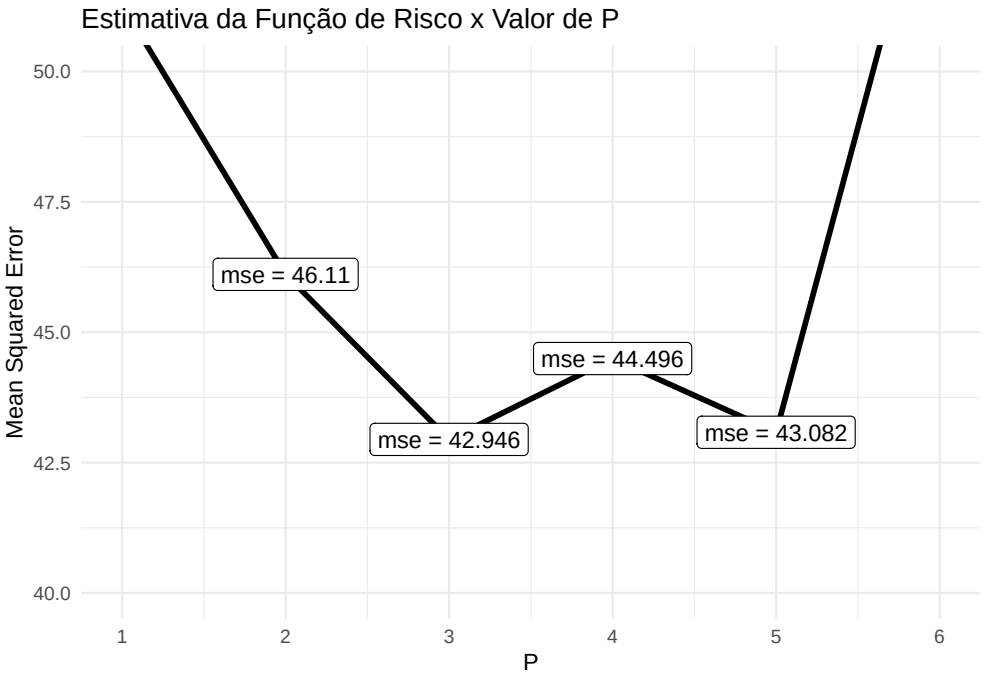
Item 3

A seguir, visualizaremos graficamente os riscos estimados para cada um dos modelos ajustados.

```
tbl_result |>
  ggplot()+
  aes(x=p,y=mse)+
  geom_line(size=1.25)+
  theme_minimal()+
  labs(x = 'P', y = "Mean Squared Error",
       title='Estimativa da Função de Risco x Valor de P')
```




```
tbl_result |>
  ggplot()+
  aes(x=p,y=mse)+
  geom_line(size=1.25)+
  geom_label(aes(label=paste0('mse = ', round(mse,3))))+
  coord_cartesian(ylim=c(40,50), xlim=c(1,6))+
  scale_x_continuous(breaks = 1:6)+
  theme_minimal()+
  labs(x = 'P', y = "Mean Squared Error",
       title='Estimativa da Função de Risco x Valor de P')
```



Através do gráfico do risco estimado vs p, é possível visualizarmos como o risco estimado se comporta em função do aumento de parâmetros da função. No primeiro gráfico, observa-se que o risco estimado cresce conforme o p aumenta: esse efeito é dado pelo aumento da variância que ocorre quanto cresce o número de parâmetros da função do modelo.

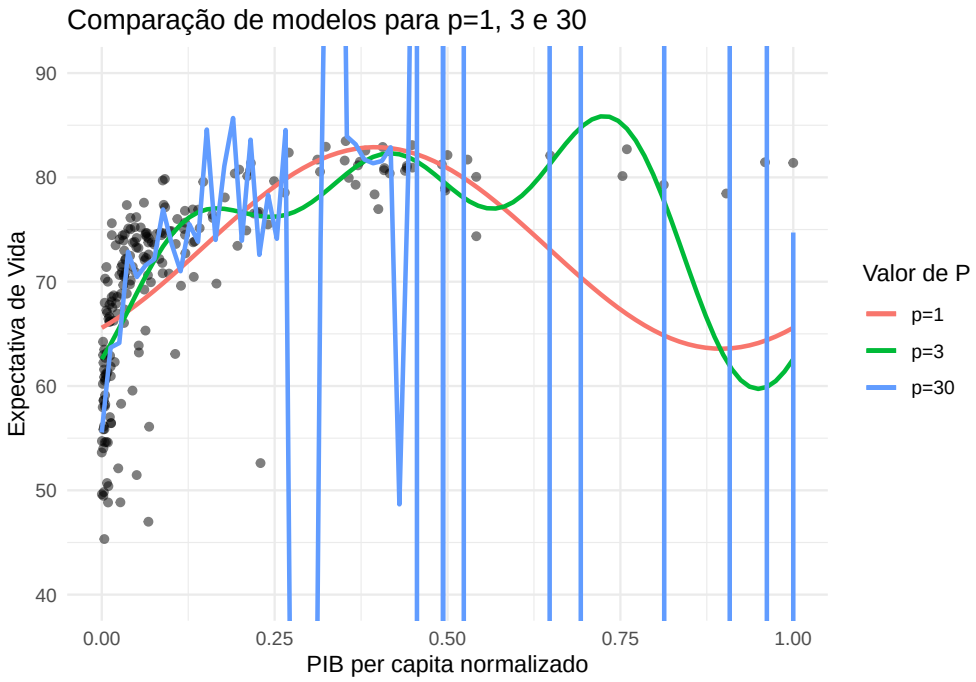
Já no segundo gráfico, temos o *zoom* do quadrante onde os valores de p são mais baixos, sendo intuitivo imaginar que o menor erro seria atingido no menor valor de p possível. No entanto, isso não acontece por conta do viés que modelos muito simples podem apresentar. A reta plotada no gráfico nos mostra exatamente isso, sendo possível observar que o ponto de mínimo risco estimado é atingido em $p = 3$, com o erro de 42.9.

Portanto, através do critério do menor risco estimado possível, temos que $p_esc = 3$.

Item 4

```
cores <- c('p=1'="#F8766D",
           'p=3'="#00BA38",
           'p=30'="#619BFF")

df |>
  ggplot()+
  aes(x=x,y=y)+
  geom_point(alpha=0.5)+
  geom_smooth(aes(color='p=1'),
              method = 'lm', se=FALSE, formula=formulas[[1]])+
  geom_smooth(aes(color='p=3'),
              method = 'lm', se=FALSE, formula=formulas[[3]])+
  geom_smooth(aes(color='p=30'),
              method = 'lm', se=FALSE, formula=formulas[[30]])+
  coord_cartesian(ylim=c(40,90))+
```

```
  theme_minimal()+
  labs(color='Valor de P',
       x='PIB per capita normalizado',
       y='Expectativa de Vida',
       title='Comparação de modelos para p=1, 3 e 30')
```



Item 5

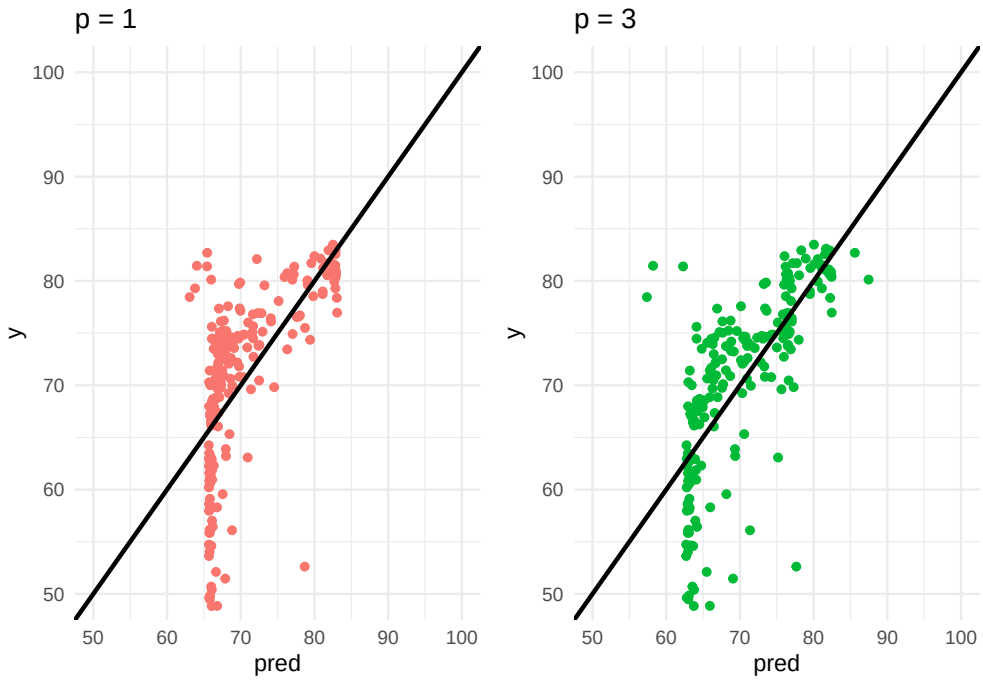
Agora, iremos calcular os valores preditos para os modelos onde $p = 1, 3, 30$ apenas. Com isso, utilizamos uma `tibble` onde uma coluna corresponde aos valores observados, outra aos valores preditos e uma terceira que corresponde ao modelo ao qual os valores observados e preditos correspondem.

Abaixo, temos os gráficos para todos os modelos separados e juntos:

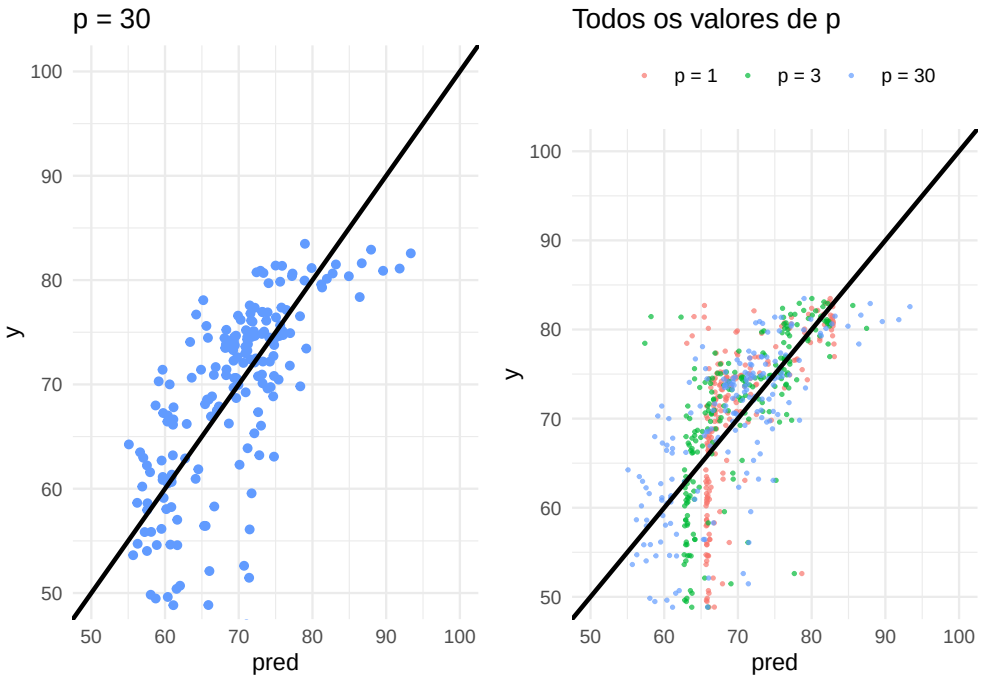
```
# Criando tabela para armazenar valores preditos

predict_new <- tibble(
  y = rep(df$y, 3),
  pred = rep(0, 3*nrow(df)),
  p_fator = c(rep('p = 1', nrow(df)),
              rep('p = 3', nrow(df)),
              rep('p = 30', nrow(df)))
)

model <- NULL

for (ii in 1:nrow(df)) {

  #p=1
```

```
  model <- lm(formulas[[1]], data=df[-ii,]) # Ajuste do modelo sem obs. ii

  predict_new[ii,2] <- predict(model, df[ii,]) # Predizendo obs ii

  #p=3

  model <- lm(formulas[[3]], data=df[-ii,]) # Ajuste do modelo sem obs. ii

  predict_new[ii+211,2] <- predict(model, df[ii,]) # Predizendo obs ii

  #p=30

  model <- lm(formulas[[30]], data=df[-ii,]) # Ajuste do modelo sem obs ii

  predict_new[ii+422,2] <- predict(model, df[ii,]) # Predizendo obs ii

}

library(gridExtra)

# grafico para p = 1

p1 <- predict_new[1:211,] |>
  ggplot()+
  aes(x=pred,y=y)+
  geom_point(color="#F8766D")+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('p = 1')

# grafico para p = 3

p3 <- predict_new[212:422,] |>
  ggplot()+
  aes(x=pred,y=y)+
  geom_point(color="#00BA38")+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('p = 3')

p30 <- predict_new[423:633,] |>
  ggplot()+
  aes(x=pred,y=y)+
  geom_point(color="#619BFF")+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('p = 30')
```

```
todos <- predict_new |>
  ggplot()+
  aes(x=pred,y=y, color=p_fator)+
  geom_point(size=0.5, alpha=0.7)+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  labs(color='',
       title='Todos os valores de p')+
  theme(legend.position = 'top')

# Mostrando p=1 e p=3

grid.arrange(p1, p3, ncol=2)
```




```
# Mostrando p=30 e todos os p

grid.arrange(p30, todos, ncol=2)
```



Dos três modelos ajustados, aquele que parece mais razoável é o que possui $p = 3$, estando mais próximo possível da linha diagonal em preto. No entanto, vale ressaltarmos que todos os modelos apresentaram desempenho distante do observado para valores preditos aproximadamente menores do que 65. O modelo com o pior desempenho é o que possui $p = 30$. A seguir, veremos as estimativas do risco para estes três modelos para compreendermos de que forma diferem:

```
# Mostrando MSE estimado para p = 1, p = 3 e p = 30

tbl_result[c(1,3,30),] |>
  kable('latex',digits=4, align='cc',
        caption = 'Risco estimado para p=1, p=3 e p=30',
        col.names=c('P', 'MSE'), format.args = list(scientific = FALSE)) |>
  kable_styling(position="center",
                latex_options="HOLD_position")
```

Table 3: Risco estimado para p=1, p=3 e p=30

P	MSE
1	51.2822
3	42.9456
30	1059644387937349016404222222882.0000

Podemos ver pela tabela acima que, quando $p = 30$, temos o modelo menos razoável. Por outro lado, ainda que a diferença seja relativamente menor, o modelo que possui $p = 3$ apresenta desempenho menor tanto pelo risco estimado como pelo gráfico de valores observados vs preditos apresentado anteriormente.

Item 6

As vantagens de usarmos validação *leave-one-out* ao invés de *data-splitting* acontecem quando possuímos um banco de dados razoavelmente pequeno, já que retirar uma parte da amostra para realizar a validação via *data-splitting* poderia gerar um prejuízo no ajuste do modelo, também conhecido como *overfitting*. Isto é, o modelo estaria extremamente ajustado aos dados e, por isso, não possuiria um poder alto de predição, já que foram utilizadas poucas observações para sua construção.

Por outro lado, quando possuímos uma quantidade de dados suficiente para um ajuste bom e também para a realização de uma validação via *data-splitting*, é possível evitarmos problemas já citados anteriormente, como o *overfitting*, podendo verificar se o nosso modelo consegue fazer boas previsões.