Lista 03 - Aprendizado de Máquinas

23 junho, 2023

Exercício 01

Neste exercício usaremos o banco `mtcars` que já vem instalando no `R`. Para carregar este banco, basta fazer o seguinte:

```
library(tidymodels)
df <- as_tibble(mtcars)
```

Faça a regressão linear usando o preditor `hp` para a resposta `mpg`. Neste exercício iremos usar o comando `lm` para fazer a regressão, esta função já vem definida na linguagem `base R`. Chame a função `summary` para ver o resultado da regressão. Comente resultados sobre a qualidade da regressão e a importância do preditor. Como ilustrado abaixo:

```
lin.model <- lm(mpg ~ hp, data = df)
summary(lin.model)
```

Faça o gráfico de dispersão de pontos e trace a reta usando o comando `geom_abline` da biblioteca `ggplot2` (note que essa função recebe argumentos `intercept` e `slope`).

Agora use todos os preditores do banco `mtcars` para predizer a resposta mpg. Use o comando `summary` e discuta os resultados. A variável hp possui a mesma importância do que no item anterior? Discuta o resultado em termos de colinearidade. Você pode calcular o fator de *variance inflation* usando o comando `vif` do pacote `car`, como no exemplo abaixo.

```
library(car)
lin.model <- lm(mpg ~ ., data = df)
summary(lin.model)
vif(lin.model)
```

Exercício 02

Neste exercício usaremos um banco de dados da FIFA publicado em 2018 sobre jogadores de futebol. Para carregar este banco, chame a seguinte função:

```
file_url = "https://drive.google.com/uc?export=download&id=1jiWcGsl_tbqK5F0ryUTq48kcDTKWTTuk"
df_orign <- read.csv(file_url) %>% as_tibble
```

Para simplificar o exercício, vamos escolher algumas variáveis para usar e limpar um pouco os dados, como a seguir:

```
library(stringr) # biblioteca de strings (para a função str_extract)
df <- df_orign %>%
    select(Age, Overall, Potential, Wage, Special,
           Acceleration, Aggression, Agility, Balance, Ball.control,
           Composure, Crossing, Curve, Dribbling, Finishing, Positioning,
           Stamina, Interceptions, Strength, Vision, Volleys, Jumping, Penalties,
           Shot.power, Sprint.speed, Heading.accuracy, Long.passing, Short.passing
           ) %>%
    # Extraindo somente a parte numérica
```

```
    mutate( Wage = as.integer(str_extract(Wage,"[0-9]+")) ) %>%
    # Converte todas colunas texto para inteiro
    mutate_if(is.character,as.integer) %>%
    # Remove entradas com dados faltantes NA
    na.omit()
```

Faça uma regressão linear com esses dados e identifique os principais preditores baseados nos $p-$valores. Chame o comando `vif` para calcular o fator de *variance inflation* discuta o resultado em termos de colinearidade e as implicações para esta análise usando $p-$valores.

Exercício 03

Ao invés de usar $p-$valores para identificar os preditores com maior relevância, vamos usar o método *Best Subset Selection* (BSS). Este método está implementando no pacote `leaps`. Para exercutar o método basta chamar `regsubsets` com a fórmula para a regressão, o nível desejado e o parâmetro `method` como `"exhaustive"`. Para ver as variáveis escolhidas para cada nível (com números de preditores diferentes) basta chamar a função `tidy` como ilustrado abaixo.

```
library(leaps)

# O argumento nvmax é o maior número de preditores para ser considerado.
# Vamos usar o maior nível, que seria nvmax = `nrow(df)-1`
# ATENCAO: Caso isso seja muito pesado computacionalmente para
#  o computador que você está utilizando, reduza o nível para
#  algo menor, como nvmax = 12 ou 8.
regfit.full = regsubsets(Wage ~ ., df, method = "exhaustive", nvmax=nrow(df)-1)
```

O commando `tidy` também é importante. Ele retorna uma `tibble` com as variáveis incluídas em cada nível. Para ficar mais fácil de visualizar essa tabela, fiz um pipe `%>%` do resultado para o comando `View`. As linhas desta tabela representa os níveis e um valor de `TRUE` representa que a variável é incluida naquele nível.

```
tidy(regfit.full) %>% View
```

Você pode encontrar o melhor modelo dentre os níveis escolhendo o que possui maior R^2 ajustado:

```
regfit.summary = tidy(regfit.full)
which.max(regfit.summary$adj.r.squared)
```

Mas para tornar o resultado mais interessante, faça um gráfico do R^2 ajustado a medida que o número de preditores varia usando a tabela gerada pelo comando `tidy`. Quantos preditores você escolheria? (Neste exercício, como não estamos usando o erro de validação, não se preocupe em calcular o *standard error*, use somente o bom senso)

Exercício 04

Vamos continuar usando o conjunto de dados da FIFA do exercício anterior, mas desta vez, vamos usar o *Forward Subset Selection* (FSS) e vamos procurar dentre todos os níveis possíveis (com preditores variando de 1 até 27 `=ncol(df)-1`).

```
regfit.forward = regsubsets(Wage ~., df, method = "forward", nvmax=ncol(df)-1)
regfit.summary = tidy(regfit.forward)
which.max(regfit.summary$adj.r.squared)
```

Como feito no exercício anterior, faça um gráfico do R^2 ajustado a medida que o número de preditores aumenta. Qual modelo você escolheria de acordo com esse critério do R^2 ajustado?

Quais são as **vantagens** e **desvantagens** do método FSS em relação ao método BSS?

Exercício 05

No exercício anterior, usamos o R^2 ajustado para escolher o melhor conjunto de preditores para o nosso modelo. Para termos uma medida melhor do erro, o ideal seria escolher o número de preditores usando validação cruzada.

Para isso, usaremos o comando `vfold_cv` para gerar a partição dos dados em *folds*. Escolhemos $v = 10$ abaixo para fazer um $10-$fold.

```
cv.split = vfold_cv(df,v=10)
```

Para terminar este exercício complete o seguinte código

```
## A matriz results vai ser usada para
##  guardar os resultados de cada fold.
##  Linhas representam folds e colunas
##  representam números diferentes de
##  preditores

results <- matrix(0,nrow=nrow(cv.split),ncol=ncol(df)-1)
for( i in 1:nrow(cv.split) )
{
    s = cv.split$splits[[i]]
    # Cria o conjunto de treinamento e teste
    #  a partir do fold s
    train = analysis(s)
    test  = assessment(s)

    # Rodar o método FSS para seleção dos
    #  preditores. Guarde o resultado
    #  na variável rss.fit

    rss.fit <- #<<INCLUIR SEU CODIGO>>

    rss.td = tidy(rss.fit)
    for( j in 1:nrow(rss.td) )
    {
        # Recebe os coeficientes da regressão
        #  para o melhor modelo do nível j.
        #  Ou seja, coefs são os betas chapéu.
        coefs   <- coef(rss.fit,id = j)

        # Indentifica os preditores do melhor
        #  modelo do nível j
        v.names <- names(coefs)

        # A função model.matrix gera a matriz N
        #  usada para fazer a regressão. Aqui,
        #  criamos a matriz para o conjunto de
        #  testes.
        test.mat<- model.matrix(Wage ~ ., data = test)

        # Fazemos a predição dos valores de acordo
        # com os coeficientes ajustados no conjunto
        # de treinamento
        pred    <- test.mat[,v.names] %*% coefs
```

```
        # Com os valores reais do conjunto de testes
        #  e os valores preditos pelo modelo da
        #  variável pred, calcule o MSS (mean
        #  squared error)

        MSS <- #<<SEU CODIGO AQUI>>

        results[i,j] = MSS
    }
}
plot.df <- tibble(rowid = 1:ncol(results),
                  MSS = colMeans(results))

ggplot(plot.df, aes(x = rowid, y = MSS)) +
    geom_line() +
    geom_point()

which.min(plot.df$MSS)
```