Laboratório 4 – Predição e Decisão

ME905

Instruções Gerais

- Baixe os arquivos `produtos_treino.csv`, `produtos_valida.csv` e `produtos_teste.csv` disponíveis em:
  https://drive.google.com/drive/folders/1wLey8vhtYhYPEnI2lqL3FOEhSSJisb_k?usp=sharing
- Utilize qualquer pacote ou função que achar adequado para ajuste dos modelos e validação cruzada, desde que os métodos tenham sido discutidos em aula.
- Além do código, inclua explicações em texto sempre que considerar necessário. Justifique suas escolhas de métodos, parâmetros e comente os resultados obtidos.
- Seu relatório deve ser claro, organizado e bem documentado.

---

1. Leitura dos Dados

Carregue o conjunto de treino (`produtos_treino.csv`), que contém um `data.frame` com:

- **20 colunas (`x1` a `x20`)**: covariáveis descritivas dos produtos (cada linha corresponde a uma unidade);
- **Coluna `y`**: variável resposta, indicando se o produto está com defeito (`y = 1`) ou funcionando (`y = 0`);
- **Coluna `cost`**: custo de compra de cada produto (para os exercícios a partir do 4).

```r
library(readr)
treino <- read_csv('produtos_treino.csv')
treino$y <- factor(treino$y)
```

---

2. Ajuste de Modelos

Utilizando os métodos estudados ao longo do semestre, ajuste um modelo para estimar a probabilidade de que um produto esteja com defeito (`y = 1`), com base nas variáveis `x1` a `x20`.

Orientações:

- Utilize apenas os dados de treino (`produtos_treino.csv`) nesta etapa.
- Caso julgue necessário, use validação cruzada para avaliar o desempenho dos modelos.
- Apresente as diferentes abordagens testadas (métodos, escolha de hiperparâmetros, etc).
- Comente as métricas de desempenho obtidas (por exemplo: acurácia), destacando a escolha do modelo final.

```r
set.seed(123)

# hold out 80/20
i_treino <- sample(1:1000, size = 800, replace = F)
```

```r
sample_treino <- treino[i_treino,]
sample_teste <- treino[-i_treino,]

# ajustando modelos
fit1 <- neuralnet(y ~ . - cost, data = sample_treino, hidden = c(20, 15, 20),
                  linear.output = FALSE, act.fct = 'logistic', lifesign = 'full')
```

```
## hidden: 20, 15, 20    thresh: 0.01    rep: 1/1    steps:     114 error: 5.01719  time: 0.5 secs
```

```r
pred1 <- apply(predict(fit1, newdata = sample_teste), MARGIN = 1, which.max) - 1
tab1 <- table('modelo' = pred1, 'dados' = sample_teste$y)
acuracia1 <- (tab1[1] + tab1[4]) / sum(tab1)
tab1  # matriz de confusão
```

```
##       dados
## modelo   0   1
##      0 169  15
##      1   6  10
```

```r
fit2 <- randomForest(y ~ . - cost, data = sample_treino, ntree = 1000)
pred2 <- predict(fit2, newdata = sample_teste)
tab2 <- table('modelo' = pred2, 'dados' = sample_teste$y)
acuracia2 <- (tab2[1] + tab2[4]) / sum(tab2)
tab2  # matriz de confusão
```

```
##       dados
## modelo   0   1
##      0 173  19
##      1   2   6
```

```r
# modelo escolhido
fit <- neuralnet(y ~ . - cost, data = treino, hidden = c(20, 15, 20),
                 linear.output = FALSE, act.fct = 'logistic', lifesign = 'full')
```

```
## hidden: 20, 15, 20    thresh: 0.01    rep: 1/1    steps:     129 error: 2.00661  time: 0.8 secs
```

Foram feitas diversas tentativas manuais para a escolha da quantidade de neurons nas camadas ocultas da rede neural, sendo o melhor valor encontrado `hidden = c(20, 15, 20)`. Além disso, durante os testes, notou-se que utilziar a estratégia de afunilar e crescer de novo os números de neurons, isto é, colocar um número alto na primeira camada de neurons, e depois diminuir na segunda, permitia em geral uma convergência mais rápida e uma estabilização no erro, neste caso. É válido ressaltar que pode ter sido uma mera coincidência, mas vale a justificativa do gradiente estabilizar mais na hora do cálculo do erro. Por sua vez, o cenário em que havia poucos neurons na primeira camada para muitos neurons na segunda era muito demorado, instável e muitas vezes não convergia.

Ademais, ressalta-se que primeira camada é crucial, pois ela determina a quantidade de combinações iniciais. Um exemplo (informal) é dado por: considerando `hidden = c(2, 100)` o modelo é limitado a apenas duas combinações das preditoras mas com uma certa flexibilidade, enquato `hidden = c(100, 2)` limita o modelo à 100 combinações das preditoras mas com menos flexibilidade que a anterior.

Além disso, o parâmetro `threshold` (menor valor absoluto da derivada parcial) foi fundamental para a rapidez da convergência. No sentido de que se ele for um valor pequeno, isso significa que o próximo passo é andar pouco, logo, não "compensa" andar.

Por fim, vale vale ressaltar que foi ajustado, de forma comparativa, uma floresta aleatória. Coincidentemente, ambos métodos apresentaram 89.5% de acurácia. Entretanto, pela matriz de confusão dos modelos, foi possível ver que a rede neural teve uma maior sensitividade (predizer que está quebrado dado que está quebrado) do que a floresta aleatória. Então, adotando uma postura mais "conservadora", optamos por ficar com o modelo da rede neural visando diminuiras perdas (comprar um produto quebrado). Vale comentar tambem que este modelo cometeu menos o erro de classificar como não quebrado quando estava quebrado (15 da rede contra 19 da floresta), outro quesito interessate para evitar perda.

---

3. Predição e Avaliação

Utilize o modelo final para gerar predições no conjunto de validação (`produtos_valida.csv`).

Tarefas:

- Calcule as probabilidades previstas de `y = 1` para os produtos da validação.
- Escolha um critério de classificação para converter as probabilidades em predições binárias.
- Construa a matriz de confusão e comente os resultados.

```
valida <- read_csv('produtos_valida.csv')
pred <- apply(predict(fit, newdata = valida), MARGIN = 1, which.max) - 1
(tab <- table('modelo' = pred, 'verdadeiro' = valida$y))
```

```
##       verdadeiro
## modelo   0   1
##      0 411  25
##      1  41  23
```

Pela matriz de confusão, erramos 66 produtos, o que gera uma acurácia de 86.8%.

---

4. Decisão Baseada em Custos

Suponha que:

- Cada produto pode ser comprado pelo valor especificado na variável `cost`;
- Produtos funcionando (`y = 0`) podem ser revendidos por **110**;
- Não é possível saber previamente se um produto está quebrado.

Tarefa:

- Determine uma regra de decisão, com base nas probabilidades preditas, que maximize o lucro esperado.
- Sua regra deve indicar, para cada produto, se ele deve ser comprado ou não.

Para modelar o problema, considere:

- ϕ: Preço de revenda do i-ésimo produto (110);
- ψ_i: Preço de compra do i-ésimo produto (variável *cost*);
- $\hat{y}$: Probabilidade do produto estar quebrado;
- Decisão: com base na probabilidade estimada do produto estar quebrado, vamos determinar se devemos comprá-lo ($d = 1$) ou não ($d = 0$).

Matriz de "Perda":

		y	
		0	1
d	0	0	0
	1	$-(\phi - \psi_i)$	ψ_i

A regra de decisão é dada pelo valor esperado da Tabela 1, ou seja, $-(\phi - \psi_i)(1 - \hat{y}) + \hat{y}\psi_i$, além disso, queremos que este valor seja menor que zero para máximizar o ganho. Logo, $\hat{y} < \dfrac{\phi - \psi_i}{\phi} = a$.

Intuição: para minimizar o custo esperado da decisão, quanto mais caro for o produto, menor o limiar a para a compra. Em outras palavras, mesmo que o produto barato tenha uma probabilidade estimada maior de quebra, ele vale mais a pena do que um caro com probabilidade baixa devido ao preço de revenda fixo ϕ. Note que se vendermos um produto que custou 11 reais, precisaríamos que outros 9 produtos com o mesmo custo quebrassem para que não exista lucro. Ao mesmo tempo, se o produto na verdade custa 100, se apenas um quebrar já estaremos no prejuízo.

---

5. Validação da Regra de Decisão

Aplique sua regra de decisão ao conjunto de validação (`produtos_valida.csv`).

Responda às perguntas:

- Quantos produtos seriam comprados? Quantos não seriam?
- Qual seria o lucro (ou prejuízo) total ao final do processo de compra e revenda?
- Comente os resultados, avaliando se a decisão parece vantajosa.

```
val_venda <- 110
valida <- read_csv('produtos_valida.csv')
```

```
## Rows: 500 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (22): x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, ...
##
## 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.
```

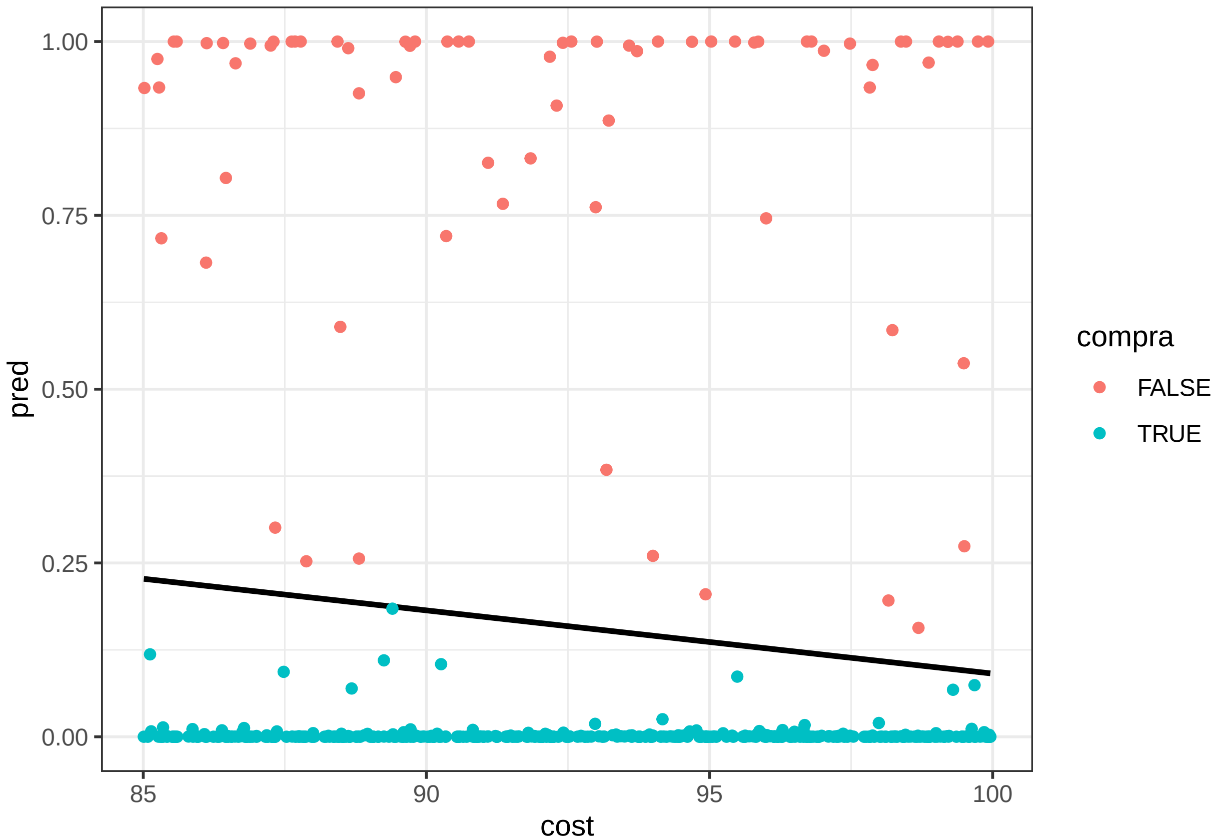
```
valida$y <- factor(valida$y)
pred_prob <- predict(fit, newdata = valida)  # prob predita
compra <- as.vector(pred_prob[,2] < (val_venda - valida$cost)/val_venda)  # indica se compra

despesa <- sum(valida$cost[compra])
venda_total <- sum(compra & (valida$y == 0)) * val_venda
lucro <- venda_total - despesa
```

Logo, compramos 427 produtos de 500, ou seja 73 não foram comprados. Além disso, obtivemos um lucro de 4632.25.

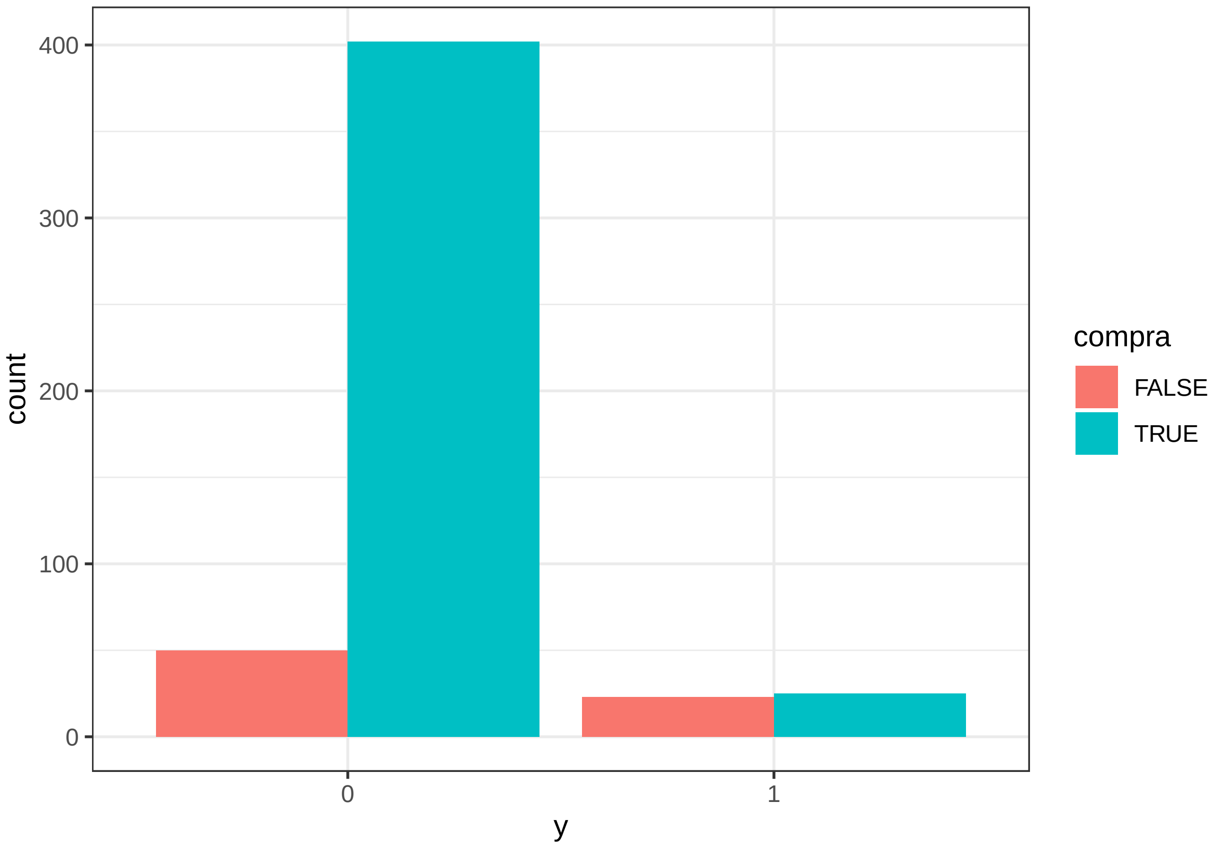
```
valida$compra <- compra
valida$pred <- pred_prob[,2]

valida |>
  ggplot(aes(x = cost, y = pred, col = compra)) +
  geom_function(fun = \(x) (val_venda - x)/val_venda, col = 'black', lwd = 1) +
  geom_point() +
  theme_bw()
```



A linha preta representa nossa regra de decisão, onde todos os itens que estão abaixo são comprados.

```
valida |>
  ggplot(aes(x = y, fill = compra)) +
  geom_bar(position = 'dodge') +
  theme_bw()
```



A regra parece vantajosa, afinal estamos ganhando dinheiro. O valor de ϕ ser fixo e maior do que qualquer custo de compra influencia nessa tomada de decisão.

---

6. Aplicação a Novos Dados

O arquivo `produtos_teste.csv` contém um novo conjunto de produtos, **sem a variável resposta `y`**.

Tarefa:

- Aplique o modelo treinado e sua regra de decisão a esse conjunto.
- Gere um arquivo `.csv` contendo apenas uma coluna chamada `d`, com as seguintes codificações:
  - `d = 1`: decisão de **comprar** o produto;
  - `d = 0`: decisão de **não comprar**.

Exemplo de saída:

d
1
0
0
1

```r
prod_test <- read_csv('produtos_teste.csv', show_col_types = F)
pred_prob <- predict(fit, newdata = prod_test)
```

```r
compra <- as.numeric(pred_prob[,2] < (val_venda - prod_test$cost)/val_venda)  # indica se compra
write_csv(data.frame(d = compra), 'l4-pred-J.csv')
```

---

7. Entrega

- Submeta no Moodle:
  1. Seu **relatório completo** (em PDF gerado a partir deste Rmd).
  2. O **arquivo `.csv` com as decisões** (`d`) para o conjunto de teste.