Script 6 — `6_Validacao_Avaliacao.R`

Objetivo do capítulo (CRISP-DM: Evaluation)

- separação correta entre seleção (Cross-Validation no treino) e avaliação (teste holdout)
- produção de evidência visual e tabular para o relatório
- robustez e reprodutibilidade (checa ficheiros essenciais, tolera opcionais)

---

1) Função principal: `run_cap6(out_cap5, out_dir, tol = 0.5)`

Inputs

- `out_cap5`: diretório onde o Cap. 5 guardou resultados
- `out_dir`: diretório onde o Cap. 6 vai guardar outputs
- `tol`: tolerância para a métrica "within ±tol" (default 0.5)

Output (invisível)

Devolve uma lista com:

- `metrics_cv` (ordenadas por RMSE)
- `metrics_test` (as métricas finais do Cap. 5)
- `preds_diag` (previsões com resíduos e flags de acerto)
- `top_vars` (importâncias/coeficientes agregados, se existirem)

---

2) Leitura e validação de ficheiros essenciais do Cap. 5 (obrigatórios)

O script define os caminhos para 4 ficheiros **obrigatórios**:

- `metricas_cv.csv`
- `best_model_by_cv.txt`
- `previsoes_teste_final.csv`
- `metricas_teste_final.csv`

Depois valida:

```
missing_must <- must_exist[!file.exists(must_exist)]
if (length(missing_must) > 0) stop(...)
```

Leitura efetiva

- `metrics_cv`, `metrics_test`, `preds_test` são lidos com `read.csv`.

Validação mínima das previsões

O Cap. 6 garante que `preds_test` tem:

- `y_true`
- `y_pred`

e, se não tiver `y_pred_clipped`, cria:

```
preds_test$y_pred_clipped <- clip_1_5(preds_test$y_pred)
```

---

3) Ranking e outputs de comparação (baseado em Cross-Validation do treino)

3.1 Ordenar métricas de Cross-Validation

```
metrics_cv_ord <- metrics_cv |> arrange(RMSE)
```

Guarda:

- `cap6_metricas_cv_ordenadas.csv`

3.2 Criar ranking explícito

Cria um ranking por RMSE com coluna `rank_rmse` e subset de colunas relevantes.

Guarda:

- `cap6_ranking_modelos_cv.csv`

3.3 Gráfico RMSE por modelo (barplot)

Cria:

- `cap6_fig_rmse_cv_por_modelo.png`

Detalhes:

- usa `barplot` com `las=2` (labels verticais, melhor legibilidade)
- desenha uma linha horizontal na melhor RMSE (mínimo) para destacar o vencedor

---

4) Diagnósticos no teste (modelo final)

Esta secção é o "core" da avaliação final.

4.1 Criar dataframe com diagnósticos

Chama a função do `Utils`:

```
preds_diag <- add_pred_diagnostics(...)
```

Isto adiciona ao dataset de previsões:

- `residuo = y_true - y_pred`
- `residuo_abs`
- `residuo_clip` (se necessário)
- `acc_0_5` (|erro| ≤ tol)
- `acc_0_5_clip`

Guarda:

- `cap6_predicoes_com_diagnosticos.csv`

4.2 Plots diagnósticos (4 gráficos clássicos)

Define `save_base_png` (tal como no Cap. 5) para garantir `dev.off()`.

Gera:

1. **Observado vs Previsto**

- `cap6_obs_vs_prev.png`
- inclui linha y=x (ideal = perfeito)

Interpretação típica:

- dispersão grande → erro elevado
- pontos sistematicamente acima/abaixo da linha → viés

1. **Resíduos vs Previsto**

- `cap6_residuos_vs_prev.png`
- linha horizontal em 0

Interpretação:

- "nuvem" sem padrão é bom
- padrão em funil → heterocedasticidade
- curvatura → não-linearidades não capturadas

1. **Histograma dos resíduos**

- `cap6_hist_residuos.png` Interpretação:
- centrado em 0 é bom

- assimetria indica viés

1. **Histograma do erro absoluto**

- `cap6_hist_erro_abs.png` Interpretação:
- distribuição do |erro|
- ajuda a explicar a métrica `PCT_0_5`

---

5) Interpretação automática: ler artefactos opcionais do Cap. 5 (sem falhar)

5.1 Lista de ficheiros opcionais

- Lasso: `lasso_coeficientes_lambda1se.csv`
- Ridge: `ridge_coeficientes_lambda1se.csv`
- Árvore: `tree_importancia.csv`
- RF: `rf_importancia.csv`
- GBM: `gbm_rel_influence.csv`

5.2 Extração de "top 15 variáveis" por modelo

Lasso / Ridge

- lê o csv
- assume 1ª coluna = termo, 2ª coluna = coeficiente (é como o glmnet costuma sair quando convertido e escrito)
- usa `abs(coef)` como "score"
- remove `(Intercept)`
- guarda top 15

Árvore (rpart)

- espera colunas `variavel` e `importancia`
- renomeia para esquema comum: `modelo`, `variavel`, `score`

Random Forest

- o ficheiro pode ter várias colunas numéricas de importância
- assume que a 1ª coluna contém o nome da variável
- escolhe a "primeira coluna numérica" encontrada como score
- guarda top 15

GBM

- espera colunas `var` e `rel.inf`
- usa `rel.inf` como score
- guarda top 15

5.3 Outputs desta secção

Se encontrar alguma coisa (`nrow(top_rows) > 0`), guarda:

- `cap6_top_importancias_por_modelo.csv` (tabela longa: modelo/variável/score)
- `cap6_top_variaveis_resumo.txt` (resumo em texto, organizado por modelo)

---

6) Discussão automática (template para o relatório)

O script lê o ficheiro do melhor modelo:

```
best_txt <- readLines(f_best)
```

E compõe um texto com:

- resultado da seleção por Cross-Validation
- métricas no teste (capturadas via `capture.output(print(metrics_test))`)
- paths para ficheiros e gráficos
- nota metodológica:

> "se RMSE no teste >> RMSE Cross-Validation, discutir overfitting/instabilidade."

Guarda:

- `cap6_discussao_modelos.txt`

---

7) Execução do capítulo

```
run_cap6(out_cap5, out_cap6, tol = 0.5)
```