Script 5 — `5_Modelacao.R` (Modeling no CRISP-DM)

Objetivo do capítulo (CRISP-DM: Modeling)

1. **Seleção de modelos e hiperparâmetros via Cross-Validation apenas no treino**
2. Escolha do melhor modelo por **RMSE médio em Cross-Validation (treino)**
3. **Refit final** do melhor modelo em **train + val** (mais dados para treinar)
4. **Avaliação final 1x no teste** (holdout) — o teste só é usado uma vez, no fim
5. Gerava **artefactos de interpretação** (coeficientes, importâncias, plots)

---

1) Função principal `run_cap5(...)`

Inputs (vindos do Cap. 4)

- `train, val, test`: datasets **sem escala** (adequados para árvore / RF / GBM)
- `train_s, val_s, test_s`: datasets **escalados** (adequados para lm / glmnet)
- `out_dir`: pasta de outputs do Cap. 5
- `seed`: reprodutibilidade
- `k_folds` (default 7): nº de folds na Cross-Validation do treino

Outputs críticos (consumidos pelo Cap. 6)

- `metricas_cv.csv` — tabela comparativa de modelos (Cross-Validation treino)
- `best_model_by_cv.txt` — registo do melhor modelo por Cross-Validation
- `metricas_teste_final.csv` — métricas finais no teste (com e sem clipping)
- `previsoes_teste_final.csv` — y_true, y_pred, y_pred_clipped no teste
- `cap5_resumo_final.txt` — resumo textual do capítulo e lista de artefactos

---

2) Preparação e Helpers

2.1 Targets e folds estratificados

O script define:

- `y_tr`: target do treino (sem escala)
- `y_trs`: target do treino escalado (mesmos valores)
- `y_te`: target do teste

E cria folds:

```
folds <- make_folds_stratified(y_tr, k = k_folds, seed = seed)
```

2.2 Funções auxiliares internas

- `save_base_png(...)`: wrapper para gráficos base (rpart/randomForest/gbm), garantindo `dev.off()`.
- `save_preds_generic(...)`: salva previsões simples (apesar de, no fim, o script escrever `previsoes_teste_final.csv` com clipping também).

---

3) Estrutura de avaliação: tabela `res_cv`

O objeto `res_cv` acumula, para cada modelo:

- `RMSE`, `MAE`, `R2`, `PCT_0_5`

Todas calculadas por:

```
cv_evaluate(folds, y, predict_fun)
```

---

4) Modelos treinados e avaliados em Cross-Validation (só treino)

5.0 Baseline (média do treino)

Objetivo: referência mínima (benchmark). Em cada fold, prevê a média do y no sub-treino daquele fold.

5.1 Regressão Linear (lm) — em `train_s`

Porquê usar dados escalados? Embora `lm` não exija escalamento, manter uma versão escalada alinha com glmnet e deixa o pipeline consistente.

Em cada fold:

- ajusta `lm(score_review ~ .)`
- prevê no sub-val

Artefactos (se lm for o melhor):

- `lm_summary.txt` (resumo/coef/p-values)
- `lm_coeficientes.csv` (coef + SE + t + p)
- `lm_diagnosticos.png` (4 gráficos padrão: resíduos, QQ, leverage, etc.)

5.2 Ridge e 5.3 Lasso (glmnet) — nested Cross-Validation

nested Cross-Validation dentro de cada fold.

Em cada fold externo:

1. cria `x_tr, y_trf`
2. corre `cv.glmnet(...)` **só no sub-treino**
3. escolhe `lambda.1se`
4. prevê no sub-val

Opções:

- Ridge: `alpha = 0`
- Lasso: `alpha = 1`
- `standardize = FALSE` porque já estás a usar dados escalados (`train_s`).

Artefactos (se ridge/lasso for o melhor):

- `*_lambdas.txt` (lambda.min e lambda.1se)
- `*_cv_plot.png` (curva de Cross-Validation)
- `*_coeficientes_lambda1se.csv` (coeficientes finais)

5.4 Árvore de regressão (rpart) — grid cp × maxdepth via Cross-Validation

Cria uma grelha:

- `cp ∈ {0.001, 0.005, 0.01, 0.02, 0.05}`
- `maxdepth ∈ {2,3,4,5,6}`

Para cada combinação, calcula métricas por Cross-Validation, guardando em:

- `tree_cv_grid.csv`

Escolhe a melhor por RMSE:

```
best_tree <- tree_cv_rows |> arrange(RMSE) |> slice(1)
```

Artefactos (se árvore for o melhor):

- `tree_best_params.txt`
- `tree_cptable.csv` (matriz cp/erro para poda)
- `tree_plotcp.png` (seleção cp / poda — matéria)
- `arvore_rpart.png` (visualização da árvore)
- `tree_importancia.csv` (variable.importance do rpart)

5.5 Random Forest — tuning de `mtry` via Cross-Validation (ntree fixo)

Define:

- `p = ncol(train) - 1` (nº preditores)
- cria grelha `mtry_grid` com valores típicos (inclui sqrt(p), p/3, etc.)

Para cada `mtry`, faz Cross-Validation e guarda:

- `rf_cv_grid.csv` Escolhe melhor por RMSE:
- `best_rf`

Nota de reprodutibilidade: Dentro do loop usa `set.seed(seed + mtryv)` para estabilizar resultados por configuração.

Artefactos (se RF for o melhor):

- `rf_best_params.txt`
- `rf_importancia.csv` (importance())
- `rf_varImpPlot.png` (plot da importância — matéria)
- `rf_oob_error.png` (erro OOB vs nº árvores — matéria)
- `modelo_random_forest.rds`

5.6 GBM — grid simples via Cross-Validation

Grid:

- `interaction.depth ∈ {1,2,3}`
- `shrinkage ∈ {0.05, 0.1, 0.2}`
- `n.trees ∈ {500, 1000, 2000}`

Para cada config:

- ajusta `gbm(..., distribution="gaussian")`
- prevê usando `n.trees = nts`
- avalia por Cross-Validation Guarda:
- `gbm_cv_grid.csv` Escolhe melhor por RMSE:
- `best_gbm`

Artefactos (se GBM for o melhor):

- `gbm_best_params.txt`
- `gbm_rel_influence.csv` + `gbm_rel_influence.png`
- `gbm_pdp_<var>.png` para top 3 variáveis
- `modelo_gbm.rds`

---

5) Escolha do melhor modelo (sem usar teste)

Após preencher `res_cv`, o script:

1. ordena por RMSE:

```
res_cv_sorted <- res_cv |> arrange(RMSE)
```

2. guarda:
    - `metricas_cv.csv`
3. define:
    - `best_model_id` e `best_model_desc`
4. escreve:
    - `best_model_by_cv.txt`

regra central: melhor modelo = menor RMSE médio na Cross-Validation do treino.

---

6) Ajuste final e avaliação no teste (holdout 1x)

6.1 Refit em train + val

Cria:

- `train_tv = bind_rows(train, val)`
- `train_tv_s = bind_rows(train_s, val_s)`

6.2 Treino final e previsões no teste

O script tem um bloco por modelo (`if (best_model_id == ...)`) que:

- ajusta o modelo final em `train_tv` ou `train_tv_s`
- produz `final_pred_test` no teste
- guarda o modelo em `.rds` (quando aplicável)
- cria artefactos interpretativos específicos

---

7) Outputs finais de teste (com e sem clipping)

7.1 `previsoes_teste_final.csv`

Cria:

- `y_true`
- `y_pred`
- `y_pred_clipped = clamp(y_pred, 1, 5)`

Porque decidi criar uma variável associado a "clipping"? Porque o **'y'** real só existe entre 1 e 5 e decidi evitar possiveis extrapolações, e o clipping permite comparar uma variante “realista”.

7.2 `metricas_teste_final.csv`

Calcula métricas duas vezes:

1. previsões cruas
2. previsões com clipping [1,5]

Métricas:

- RMSE, MAE, R^2
- `PCT_0_5` (within ±0.5)

---

8) Resumo final do capítulo e reprodutibilidade

`cap5_resumo_final.txt` inclui:

- seed e nº folds
- melhor modelo por Cross-Validation
- caminhos dos ficheiros chave
- lista de artefactos de interpretação por família de modelos