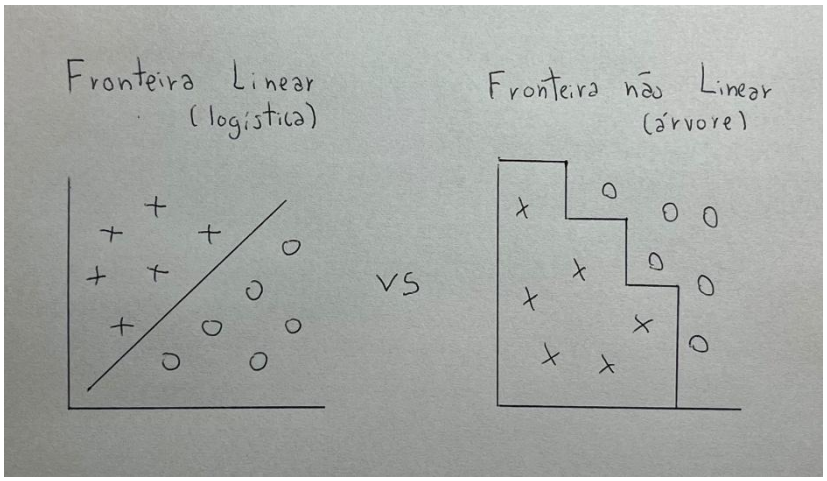
APS - Modelagem Preditiva - Grupo 10

Gustavo Colombi Ribolla

Q1- Regressão Logística x Árvore de Classificação (AUC em teste)

Diferenciação

A Regressão Logística cria uma fronteira linear para separar as classes, sendo mais simples e interpretável. Já a Árvore de Classificação faz divisões não lineares nos dados, capturando relações mais complexas entre variáveis. Assim, a logística é mais estável, enquanto a árvore é mais flexível.

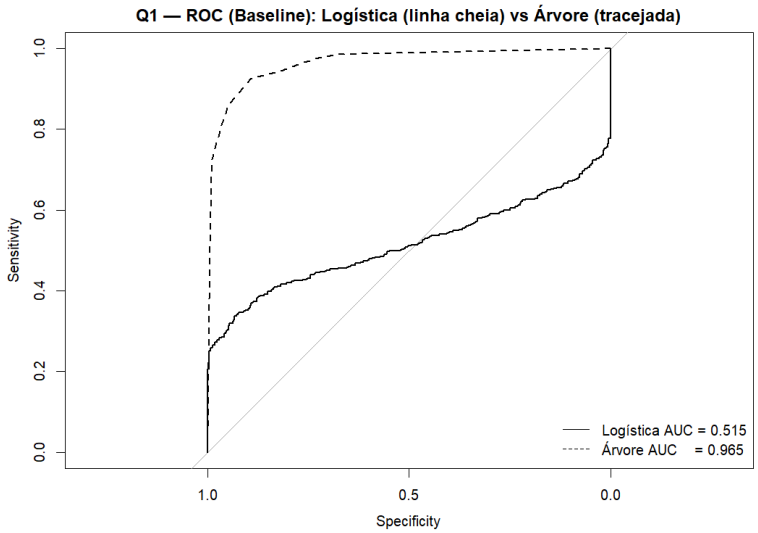


Objetivo, dados e modelagem

Treinei uma regressão logística e uma árvore de classificação com Q1_training.csv (x1, x2, y) e avaliei em Q1_test.csv usando AUC da curva ROC como métrica. A variável y foi tratada como fator {0,1} com classe positiva "1"

A regressão logística foi feita gerando as probabilidades de cada ponto pertencer à classe 1. A árvore de classificação foi criada com tree(y ~ x1 + x2), que também fornece probabilidades de classificação. Depois, usei o conjunto de teste para traçar a curva ROC e calcular a AUC, comparando o desempenho dos dois modelos.

Resultados

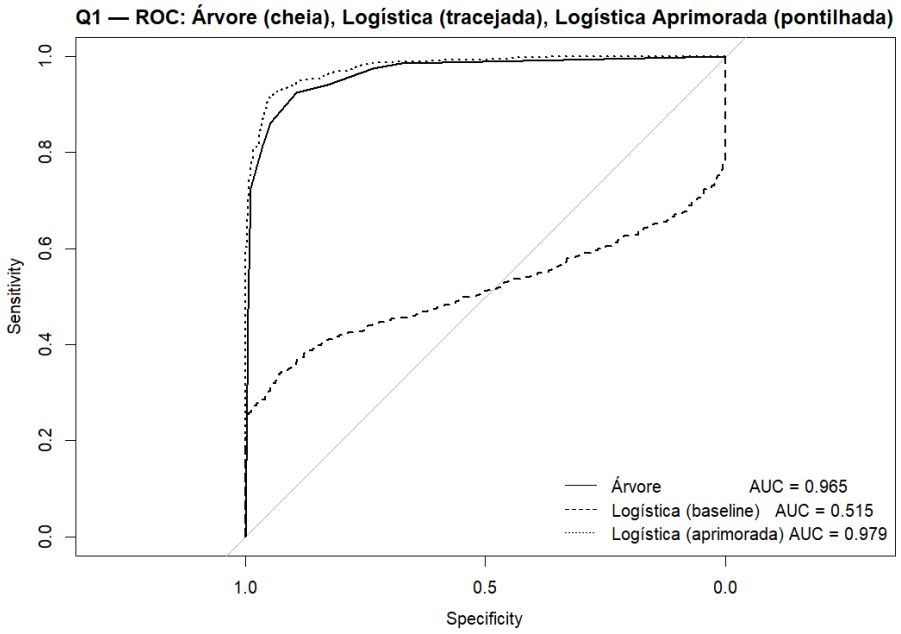


Interpretação

A logística pressupõe fronteira linear nos log-odds e, neste problema, essa suposição não descreve bem a geometria da separação entre classes. A árvore de classificação, por combinar cortes por eixos e formar regiões não lineares, capturou melhor o padrão de x1 e x2, resultando em AUC muito superior.

Modificação da logística

Para dar flexibilidade ao logit, incluí termos quadráticos em x1 e x2 e a interação x1:x2. O resultado foi um salto de desempenho: a logística aprimorada atingiu AUC = 0,979, superando tanto a árvore (AUC = 0,965) quanto a logística básica (AUC = 0,515). Na Figura abaixo, a curva pontilhada (logística aprimorada) fica acima das demais em praticamente todo o intervalo de especificidade, indicando melhor ranqueamento das probabilidades:



Conclusão

O problema apresenta separação não linear. A árvore de classificação já capturava bem essa estrutura, mas a logística com termos quadráticos e interação conseguiu igualar e até superar a árvore neste teste, mantendo um modelo paramétrico simples. Como cautela, esse acréscimo de termos pode aumentar a variância em outros cenários; se fosse implantação real, eu validaria com cross-validation/regularização. Para o caso, a evidência empírica mostra que modificar a regressão logística foi eficaz para melhorar a AUC.

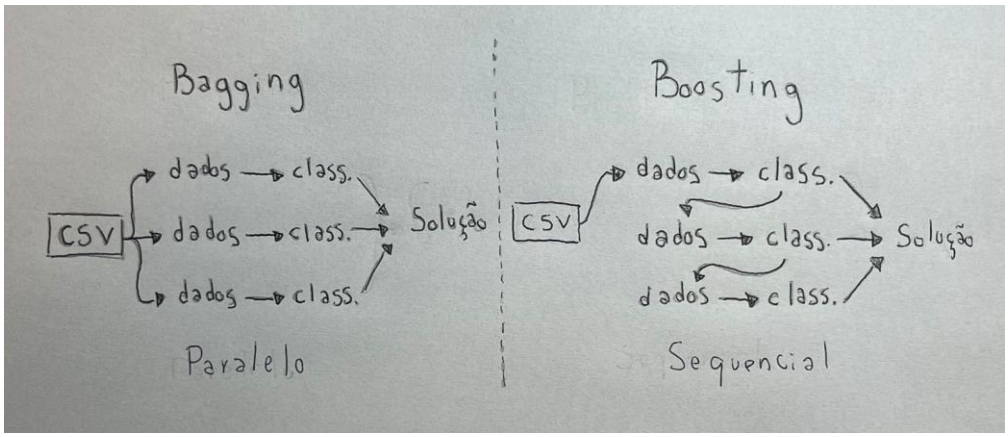
Q2- Bagging de árvores de regressão

Ideia do método (bagging)

Bagging ("bootstrap aggregating") reduz a variância de árvores de regressão: treinei várias árvores independentes em amostras levemente diferentes do treino e promediamos as previsões. O viés fica parecido ao de uma árvore, mas a variância cai, melhorando o erro de generalização.

Bootstrap (como entra no bagging)

Para cada árvore, sorteei uma amostra bootstrap do treino (mesmo tamanho, com reposição). Cada árvore vê dados um pouco distintos → modelos menos correlacionados. A previsão final do comitê é a média das previsões das árvores. A diferenciação é possível de ser vista no diagrama abaixo:



Implementação (passo a passo)

1-Split 50/50 em treino e teste. 2-Para (b=1, ... ,B): sortear bootstrap do treino, treinar uma árvore de regressão (library(tree)), prever no teste e guardar. 3-A previsão final é a média das (B) previsões por linha. 4-Métrica: RMSE no teste. Variação testada: subespaço aleatório por árvore (antes de treinar, escolher m preditores aleatórios; aqui, (m=3)).

Protocolo no *California housing*

Usando california.csv; ocean_proximity como fator e, se necessário, mediana do treino para NAs em total_bedrooms; gerei três modelos: (A) Árvore única (baseline). (B) Bagging com B=200 (amostras) e todas as variáveis. (C) Bagging + subespaço com (m=3) preditores por árvore (mesmo B). Avaliação sempre no mesmo teste (50% dos dados).

Resultados e leitura

Através dos resultados, pude perceber que houve ganho claro do bagging (~4% ↓ no RMSE), confirmando redução de variância. O subespaço piorou porque, ao limitar preditores por árvore, muitas árvores foram treinadas sem variáveis muito informativas (ex.: median_income), aumentando o viés; como evidenciado abaixo:

```
--- RMSE no conjunto de TESTE ---
Árvore única .............: 75958.73
Bagging ..................: 72732.51
Bagging + Subespaço (m=3): 82523.39
```

Conclusão

Neste conjunto, bagging puro (todas as variáveis em cada árvore) é a opção simples e eficaz com tree. O subespaço só tende a ajudar quando conseguimos descorrelacionar as árvores sem excluir sistematicamente preditores críticos, o que costuma exigir a estratégia de sorteio a cada divisão (random forests clássicos), não disponível aqui.

Q3 - Aleatorização dos splits em Random Forests

Definição: aleatorização dos splits

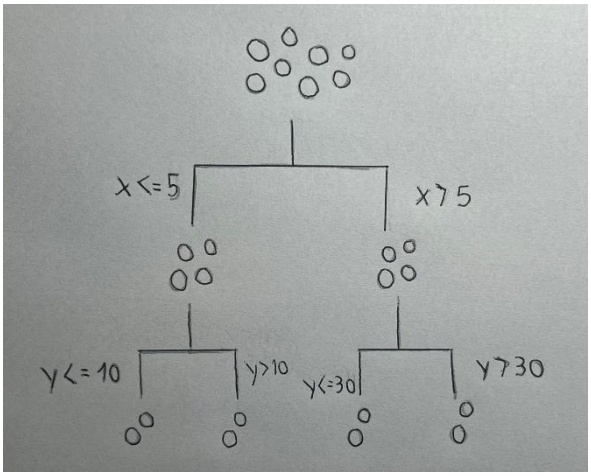
Em Random Forests, cada árvore é treinada em uma amostra bootstrap do treino e, em cada nó, apenas um subconjunto aleatório de variáveis (mtry) é testado para o corte; escolhe-se o melhor split dentro desse subconjunto. No fim, a predição é a média das árvores. Esse sorteio por nó é o elemento central que diferencia RF do bagging.

Intuição do ganho

A intuição do ganho é reduzir a correlação entre árvores. Se todas usam sempre os mesmos preditores "fortes", o comitê média modelos muito parecidos e a variância cai pouco. Limitando a busca a mtry variáveis por nó, as árvores exploram caminhos distintos; a variância do ensemble diminui mais, com leve aumento de viés. O parâmetro mtry controla o trade-off: mtry = p reproduz o bagging; mtry menor aumenta diversidade e, em geral, melhora o erro até certo ponto.

Exemplo intuitivo

Se median_income domina a explicação do preço, o bagging corta quase sempre nessa variável; com mtry por nó, alguns nós não a terão disponível e a árvore passa a usar localização, idade, etc., gerando estruturas menos correlacionadas.



Conclusão

Não é correto dizer que Breiman “apenas” usou o subespaço aleatório de Ho (1998). Ho propôs escolher um subconjunto de variáveis por árvore; Breiman combinou bootstrap por árvore com sorteio de variáveis a cada split, formalizou o uso do erro out-of-bag e popularizou medidas de importância. A definição moderna de Random Forest resulta dessa combinação e explica seu desempenho típico.

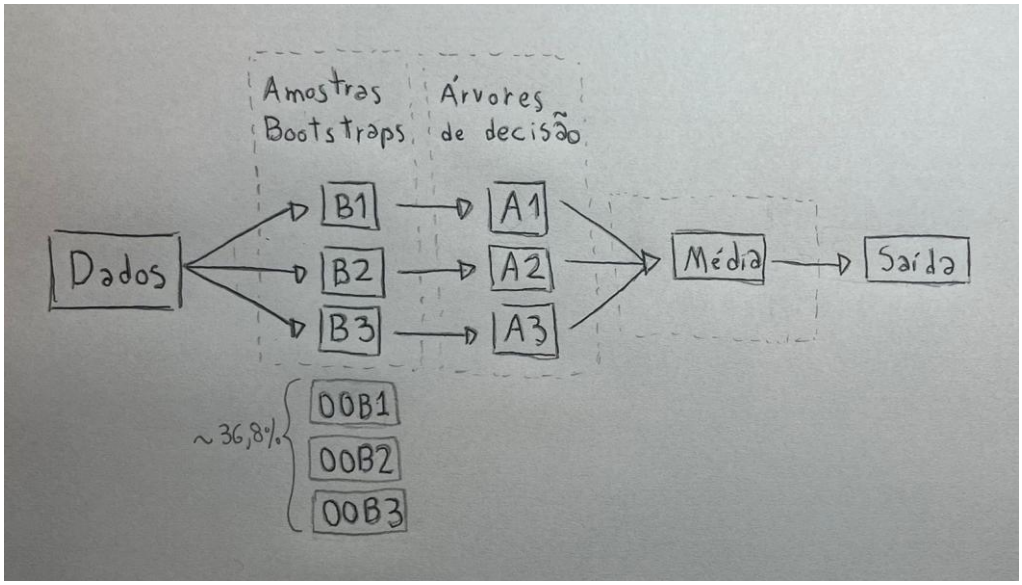
Q4 - Erro Out-of-Bag (OOB) vs Erro de Teste (Random Forests)

Definição e cálculo do OOB

Em uma Random Forest, cada árvore é treinada com uma amostra aleatória dos dados (*bootstrap*). Como esse processo é feito com reposição, cerca de 36,8% das observações ficam de fora, esses dados são chamados de out-of-bag (OOB).

Essas observações que ficaram fora são usadas para testar a árvore correspondente. Assim, para cada linha do conjunto de treino, podemos calcular uma previsão média usando apenas as árvores nas quais ela não foi usada no treino.

Depois, comparamos essas previsões com os valores reais para medir o erro OOB, que representa o quanto o modelo erra em dados que ele não viu. Quando aumentamos o número de árvores da floresta, o erro OOB tende a se estabilizar e a ficar muito próximo do erro de teste real, servindo como uma boa estimativa da capacidade de generalização do modelo.



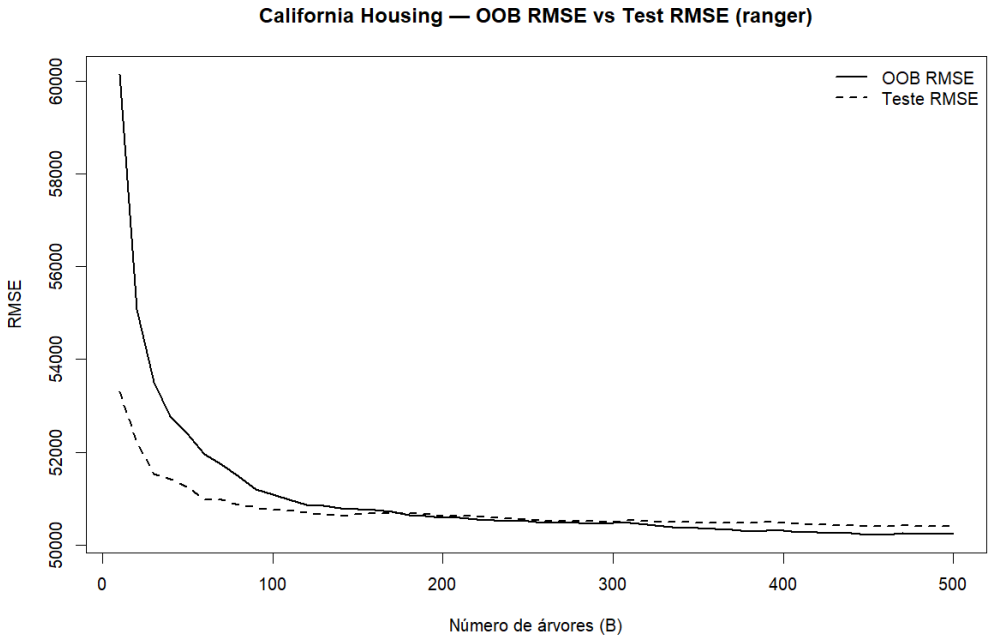
Protocolo do estudo (California housing)

Dividi o conjunto 50/50 em treino e teste, defini ocean_proximity como fator e removi eventuais linhas com NA (abordagem simples e reprodutível). Usando ranger, variei o nº de árvores B ∈ {10,...,500}, com mtry = p/3 fixo e semente 42. Para cada (B), coletei o OOB RMSE direto do modelo e o RMSE no teste por predição no hold-out.

Resultados e leitura

As curvas de OOB RMSE e Teste RMSE caem e se aproximam conforme aumento (B); após algumas centenas de árvores, ficam praticamente sobrepostas, indicando que o erro OOB é uma boa estimativa do erro de teste. Tipicamente o OOB aparece ligeiramente acima no início (mais ruidoso com poucas árvores) e decai próximo ao erro de teste quando a floresta é maior, mostrado abaixo:

```
Em B=500: OOB=50247.4 | Test=50411.7 | Diferença=164.3 (0.33%)
B=500 | OOB_RMSE=50247.37540 | Test_RMSE=50411.71064
```



Conclusão prática

O OOB permite avaliar e comparar modelos (e até ajustar hiperparâmetros) sem precisar de um conjunto de validação separado, economizando dados e tempo. No California housing, o experimento confirma a convergência OOB ≈ Teste com o aumento de (B), validando seu uso como proxy do erro de generalização.

Aplicação 1 - Churn (classificação)

Dados e protocolo

Utilizei o arquivo churn.csv, convertendo Exited para fator binário (No/Yes). Fiz um *split* 50/50 em treino e teste. Para métodos que exigem matriz numérica (k-NN e CatBoost), gerei *one-hot encoding* com model.matrix garantindo as mesmas colunas em treino e teste. No k-NN, padronizei os preditores usando média e desvio do treino.

Modelos e parametrização

Treinei cinco modelos: k-NN com a "regra de bolso" $k \approx \sqrt{n_{\text{treino}}}$ ajustado para ímpar; Regressão Logística com todas as variáveis; Árvore de Classificação (tree) sem poda explícita; Random Forest (ranger, probability=TRUE, 500 árvores, seed=42); CatBoost (perda Logloss, 300 iterações, profundidade 6, *learning rate* 0,1, random_seed=42).

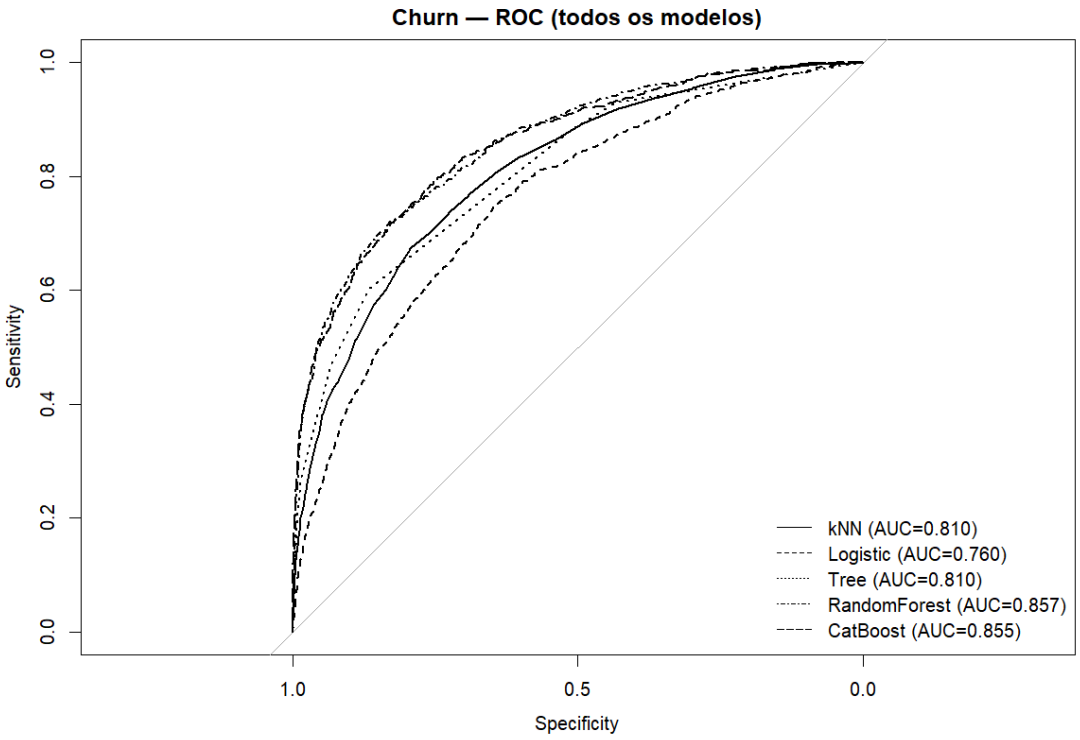
Métricas e comparação

Como pedido, comparei os métodos por acurácia no teste usando limiar 0,5 nas probabilidades. Para uma avaliação mais robusta, construí as curvas ROC de todos os modelos e comparei suas AUCs (ver Figura ROC).

Resultados (teste 50/50)

Modelo	Acurácia	AUC
k-NN	0,8190	0,8096
Logística	0,8104	0,7597
Árvore	0,8376	0,8096
Random Forest	0,8606	0,8571
CatBoost	0,8630	0,8552

A Random Forest e o CatBoost foram os melhores, muito próximos entre si (AUC ≈ 0,86). k-NN e Árvore ficaram em torno de 0,81 de AUC; a Logística ficou abaixo (0,76), sugerindo fronteiras não lineares relevantes no problema. Sendo possível de evidenciar também na curva ROC plotada:



Leitura dos resultados

O ganho dos ensembles de árvores (RF e CatBoost) é consistente com a literatura: capturam não linearidades e interações sem forte pré-processamento e reduzem variância via agregação/boosting. A Árvore única tem maior variância e por isso ficou atrás dos ensembles. O k-NN melhora em relação à Logística por explorar vizinhança no espaço padronizado, mas sem a mesma capacidade de modelar interações complexas. A Logística com limiar 0,5, apesar de simples e interpretável, é limitada para padrões mais curvos.

Conclusão

No *split* 50/50 desta base, Random Forest e CatBoost entregam a melhor performance (acurácia ≈ 0,86 e AUC ≈ 0,86), seguidos por Árvore e k-NN (AUC ≈ 0,81). A Logística fica atrás (AUC ≈ 0,76), indicando que fronteiras não lineares são importantes para prever churn aqui. Para uso prático, a escolha entre RF e CatBoost pode considerar tempo de treino, facilidade de *tuning* e interpretabilidade/recursos disponíveis.

Aplicação 2 - Preço de automóveis usados (regressão)

Dados e protocolo

Usei used_cars.csv, tendo price como variável-alvo. Fiz *split* 50/50 em treino e teste. Para evitar problemas de níveis em fatores, alinhei os níveis do teste aos do treino. Todas as avaliações foram feitas no teste.

Modelos e parametrização

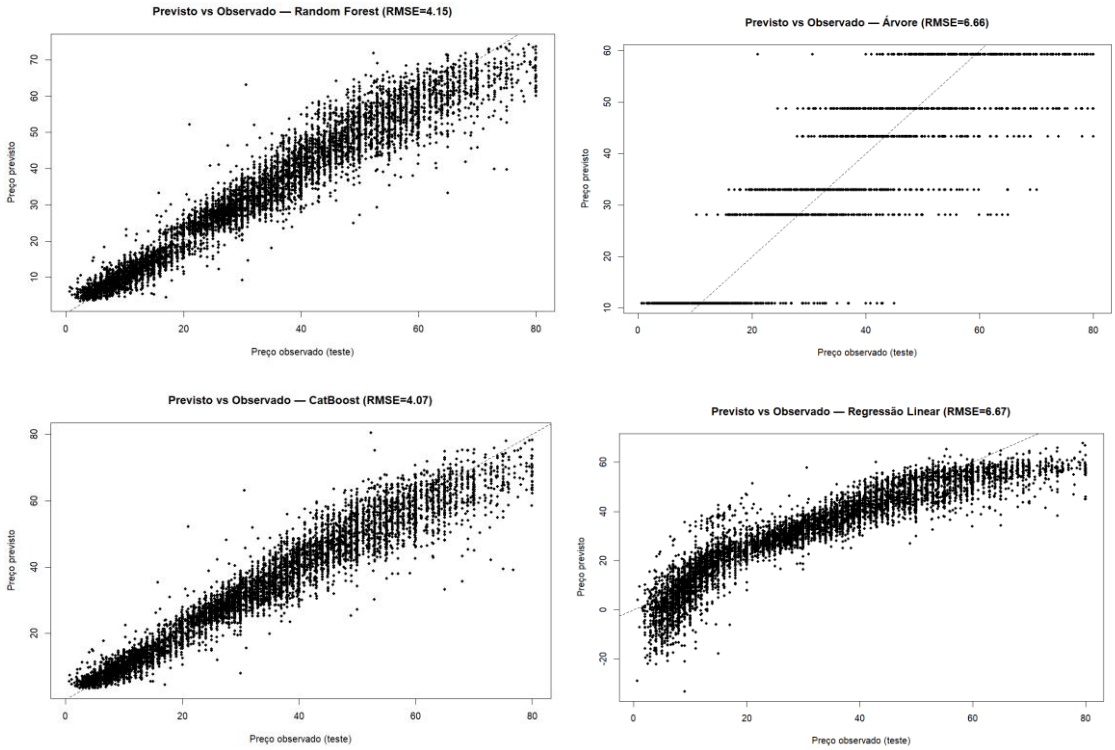
Treinei quatro modelos: Regressão Linear Múltipla com todas as variáveis; Árvore de Regressão (tree) sem poda explícita; Random Forest (ranger, 500 árvores, seed=42); e CatBoost com perda RMSE, 300 iterações, profundidade 6, *learning rate* 0,1, random_seed=42 (sem *tuning* adicional, como pede o enunciado).

Métrica e visualização

Comparei os métodos pela RMSE no teste. Para interpretar, gerei gráficos Previsto vs Observado. A árvore exibe degraus (previsões constantes por região), a regressão linear mostra tendência média com erro crescente nos extremos, e os ensembles (RF e CatBoost) aproximam melhor a diagonal ao longo de toda a faixa de preços.

Resultados (teste 50/50)

Modelo	RMSE
Linear	6,67
Árvore	6,66
Random Forest	4,15
CatBoost	4,07



Os dois ensembles ficaram claramente à frente, com CatBoost levemente melhor que Random Forest. Linear e Árvore ficaram bem atrás, sugerindo que a relação entre atributos e preço contém não linearidades e interações que os modelos simples não capturam bem.

Conclusão

Para esta base, CatBoost e Random Forest são as escolhas recomendadas (RMSE $\approx$ 4,1), fornecendo previsões mais próximas da linha ideal em todo o intervalo de preços. A Regressão Linear e a Árvore servem como bons *baselines*, mas perdem desempenho por restrições do formato (linearidade) ou por alta variância e previsões em degraus (árvore única).