

Capítulo 16 — Feature Engineering: Quando Combinar Variáveis Vale Mais Que Cada Uma Isolada

Dois tipos de combinação, com resultados opostos

Nem toda variável derivada agrega o mesmo tipo de valor. Testamos duas, de naturezas matemáticas diferentes, para mostrar essa diferença.

Primeiro exemplo, uma combinação linear, "Idade do imóvel na venda"

"Ano de venda" menos "Ano de construção" é uma combinação **linear** das duas variáveis de origem. Um modelo linear já consegue representar qualquer combinação linear das suas entradas, através dos próprios coeficientes, então adicionar essa variável não devia ajudar, e não ajudou (tabela abaixo),

"Regressão Linear" teve RMSE idêntico, com ou sem ela, 33.067, até a última casa decimal. Os algoritmos de árvore ganharam um pouco, entre 0,1% e 1,0%, porque não conseguem reconstruir essa mesma combinação de forma tão eficiente quanto uma coluna pronta.

Algoritmo	RMSE sem "Idade"	RMSE com "Idade"	Ganho
"Regressão Linear"	33.067	33.067	0,0%
"Árvore de Decisão"	41.545	41.279	-0,6%
"Random Forest"	26.457	26.421	-0,1%
"Gradient Boosting"	24.936	24.856	-0,3%
"XGBoost"	28.442	28.158	-1,0%

Segundo exemplo, uma interação, "Qualidade geral" multiplicada por "Área útil habitável"

Esta escolha de variáveis não veio de testar sistematicamente todas as combinações possíveis, veio de raciocínio de domínio, confirmado depois com número real.

Já sabíamos, desde o Capítulo 1, quais são as duas variáveis mais fortemente correlacionadas ao preço, isoladamente, "Qualidade geral", 0,80, e "Área útil habitável", 0,71, as duas mais fortes do dataset. Existe também intuição de negócio real por trás, um metro quadrado extra tende a valer mais numa casa de acabamento premium do que numa casa de padrão baixo, a dependência mútua entre as duas variáveis que uma interação multiplicativa capta.

O resultado, testado nos cinco algoritmos

Algoritmo	RMSE sem interação	RMSE com interação	Ganho
"Regressão Linear"	33.067	31.623	-4,4%
"Árvore de Decisão"	41.279	35.222	-14,7%
"Random Forest"	26.421	25.917	-1,9%
"Gradient Boosting"	24.856	24.175	-2,7%
"XGBoost"	28.158	27.373	-2,8%

Dessa vez, **todos os cinco algoritmos melhoraram**, e "Regressão Linear" teve um ganho real, 4,4%, nada de empate técnico.

Por que a diferença entre os dois exemplos é tão grande

Um modelo linear representa a previsão como uma soma ponderada das variáveis de entrada, $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots$.

Qualquer combinação **linear**, soma, subtração, multiplicação por constante, já está dentro do espaço que essa soma ponderada consegue capturar sozinha.

Um **produto entre duas variáveis**, $x_1 \times x_2$, é uma relação **não linear**, que nenhuma combinação de $\beta_1 x_1 + \beta_2 x_2$ consegue reproduzir, matematicamente são funções diferentes. Para o modelo linear representar esse efeito, a variável produto precisa vir pronta, como coluna própria.

A árvore de decisão isolada ganhou mais que todos os outros, 14,7%, porque aproximar um efeito multiplicativo usando só divisões binárias, sobre as duas variáveis separadas, exige muitas divisões sucessivas, uma aproximação cara e grosseira. Ter o produto já calculado permite à árvore capturar esse efeito com uma única divisão eficiente.

Testando se essa foi, de fato, a melhor escolha possível

A escolha de "Qualidade × Área útil" foi guiada por intuição, não por busca exaustiva. Vale testar se ela realmente foi a melhor candidata, comparando contra todas as outras interações possíveis entre as seis variáveis numéricas mais correlacionadas ao preço, quinze pares ao todo, medindo o ganho tanto em "Regressão Linear" quanto em "Gradient Boosting", como representantes dos dois estilos de algoritmo.

Interação	Ganho, "Regressão Linear"	Ganho, "Gradient Boosting"
"Qualidade" × "Vagas de garagem"	+1.890	+306
"Vagas de garagem" × "Área do porão"	+1.528	-268
"Área útil" × "Vagas de garagem"	+1.515	-19
"Qualidade" × "Área da garagem"	+1.494	-569

Interação	Ganho, "Regressão Linear"	Ganho, "Gradient Boosting"
"Qualidade" × "Área útil" (nossa escolha)	+1.444	+681
"Área da garagem" × "Área do porão"	-770	+881 (melhor de todas)

Note que não existe uma interação que seja, ao mesmo tempo, a melhor para os dois algoritmos. "Qualidade × Vagas de garagem" é a melhor opção isolada para "Regressão Linear", mas ajuda pouco "Gradient Boosting". "Área da garagem × Área do porão" é a melhor opção isolada para "Gradient Boosting", mas **piora** "Regressão Linear", um dos piores resultados de toda a busca para esse algoritmo.

"Qualidade × Área útil", a escolha original deste capítulo, não venceu em nenhuma das duas categorias, isoladamente. Mas foi a única interação, entre as cinco melhores por "Regressão Linear", que também trouxe ganho positivo para "Gradient Boosting". Não era a melhor possível para nenhum modelo específico, mas era a escolha mais equilibrada, entre as testadas, para os dois ao mesmo tempo.

Combinar variáveis vale mais que cada uma isolada, depende do tipo de combinação, linear não ajuda modelo linear, não linear ajuda todos, em graus diferentes. E depende de para qual algoritmo você está otimizando, a melhor interação para "Regressão Linear" não é a melhor para "Gradient Boosting", e escolher uma interação "equilibrada", que sirva bem a vários algoritmos ao mesmo tempo, tem um custo reconhecido, deixar ganho adicional na mesa, para cada algoritmo individualmente, em troca de uma comparação mais simples entre eles. Essa mesma tensão, "o que é ótimo depende de qual algoritmo você está usando", já apareceu nos Capítulos 8, 9 e 13, sobre "Random Forest" e "XGBoost", e volta a aparecer aqui, confirmando que é um padrão estrutural deste tipo de problema, não uma coincidência isolada de um capítulo específico.

O que este capítulo ensina para o Cientista de Dados

1. Combinação linear NÃO ajuda modelos lineares — mas ajuda árvores um pouquinho

A Regressão Linear já consegue representar qualquer combinação linear de suas entradas através dos próprios coeficientes ($\beta_1 x_1 + \beta_2 x_2$). Adicionar a coluna derivada é matematicamente redundante — o modelo simplesmente reajusta os pesos. Já as árvores não fazem combinações lineares naturalmente; elas precisam de múltiplas divisões para aproximar uma subtração, então a coluna pronta dá um pequeno ganho. O Cientista de Dados deve saber que **feature engineering não é uma atividade "de uma receita só"** — o ganho depende de como o algoritmo interno representa a informação.

2. Interações multiplicativas (produtos) ajudam TODOS os algoritmos — e muito

O segundo teste, "Qualidade geral $\times$ Área útil habitável", é uma **interação não linear**. O resultado é completamente diferente:

Um modelo linear nunca consegue representar o produto $x_1 \times x_2$ através da soma $\beta_1 x_1 + \beta_2 x_2$ — são funções matematicamente diferentes. Para capturar que "um metro quadrado extra vale mais numa casa premium do que numa casa básica", o modelo *precisa* do termo de interação pronto. As árvores, especialmente a Árvore de Decisão isolada, também se beneficiam imensamente, porque aproximar uma superfície multiplicativa com divisões binárias sucessivas é ineficiente — ter o produto já calculado permite capturar o efeito com uma única divisão. **A mensagem central é:** vale a pena investir tempo criando interações de negócio com intuição por trás, especialmente para modelos lineares ou árvores rasas.

3. A "melhor" feature derivada é específica do algoritmo — não existe uma campeã universal

O Cientista de Dados precisa entender que **otimização de features é acoplada à escolha do modelo**. O que é "ótimo" para uma Regressão Linear pode ser "ruim" para um Gradient Boosting (piorar o erro em centenas de dólares). Portanto, não adianta criar uma feature "genérica" achando que ela vai ajudar todos os modelos igualmente — o ganho precisa ser validado empiricamente para cada algoritmo candidato.

4. valor do compromisso ("equilíbrio") vs. a busca pelo ótimo individual

No mundo real, você raramente vai otimizar um único modelo no vácuo. Muitas vezes, você precisa comparar múltiplos algoritmos para decidir qual colocar em produção. Nesse cenário, **escolher uma feature que funciona "bem o bastante" para todos pode ser mais estratégico do que buscar a feature "perfeita" para um modelo específico** — porque a primeira permite uma comparação limpa e generalizável, enquanto a segunda pode enviesar a escolha do algoritmo. O Cientista de Dados precisa saber quando está **otimizando para um modelo** e quando está **criando uma base de comparação neutra** — e comunicar essa distinção claramente.

5. Conclusão final para o Cientista de Dados:

"Feature Engineering não é sobre 'criar o máximo de colunas possíveis' — é sobre criar a coluna certa, do tipo certo (linear vs. interação), validada para o algoritmo certo. Uma interação que é ouro para a Regressão Linear pode ser veneno para o Gradient Boosting; e a melhor feature não é a que funciona no vácuo, mas a que se encaixa no seu objetivo — seja ele derrubar o erro de um modelo específico ou manter uma comparação justa entre competidores."