



AULÃO 2 • CAPÍTULO 2 DA IMERSÃO

Dados tabulares

Uma tabela comum: uma linha por caso, uma coluna por informação. Vocês já resolveram isso com floresta aleatória. Agora é com rede neural.



COMO LER ESTE DOCUMENTO

Este PDF é o registro do Aulão 2. Cada página mostra uma tela do material interativo já aberta, num estado escolhido, e a faixa de baixo explica o que está acontecendo ali. Os laboratórios aparecem em vários estados, para vocês compararem o que muda quando um controle se mexe — é justamente o que não cabe numa imagem só. O material vivo, com todos os laboratórios para mexer, está no endereço da última página.

Os quatro blocos do desafio

A trilha é a mesma da Fase 1. Cliquem num bloco para ver o que mudou.

01 · EXPLORAR

02 · CONFIGURAR

03 · MODELAR

04 · AVALIAR

JÁ ERA ASSIM NA FASE 1

Abrir a tabela, olhar as colunas, contar quantos casos tem cada classe.

O QUE É NOVO NA FASE 2

Antes de treinar, vocês medem o **piso** (a nota de quem só chuta a classe mais comum) e o **sinal** de cada coluna. Também contam quantos valores diferentes cada categoria tem.

Sem esse último número, dois dos três modelos nem rodam.

Os quatro blocos do desafio tabular. O Bloco 1 é o único que não envolve treinar nada: é onde se descobre o piso, quais colunas carregam sinal e quantos valores tem cada categoria. Quem pula este bloco descobre no último dia que uma coluna estava vazando o rótulo.

Os quatro blocos do desafio

A trilha é a mesma da Fase 1. Cliquem num bloco para ver o que mudou.

01 · EXPLORAR

02 · CONFIGURAR

03 · MODELAR

04 · AVALIAR

JÁ ERA ASSIM NA FASE 1

Escolher o modelo. Trocar de modelo era trocar uma palavra no código.

O QUE É NOVO NA FASE 2

Agora **cada modelo quer a entrada de um jeito**: um quer categorias em colunas de 0 e 1, outro quer índices e as cardinalidades, o terceiro dispensa a régua. E cada um tem os seus próprios botões.

O preparo deixou de ser detalhe e virou metade da escolha.

O Bloco 2 é de decisão, não de código. Escolher o modelo, a régua e os hiperparâmetros é o que define o resultado — e cada escolha aqui tem consequência medida nas curvas do Bloco 4.

Os quatro blocos do desafio

A trilha é a mesma da Fase 1. Cliquem num bloco para ver o que mudou.

01 · EXPLORAR

02 · CONFIGURAR

03 · MODELAR

04 · AVALIAR

JÁ ERA ASSIM NA FASE 1

Chamar o treino e esperar o resultado aparecer.

O QUE É NOVO NA FASE 2

O treino virou um **laço de épocas e lotes**, com hiperparâmetros para ajustar. Dentro da célula B existe a **Zona Livre**: o que vocês acrescentam ali é o que separa o mediano do bom.

Quem apenas preenche as lacunas tem resultado correto e mediano.

O Bloco 3 é onde se escreve. São três células, sempre as mesmas, nas três trilhas e nos nove modelos do material. Aprender esta forma uma vez economiza as outras oito.

Os quatro blocos do desafio

A trilha é a mesma da Fase 1. Cliquem num bloco para ver o que mudou.

01 · EXPLORAR

02 · CONFIGURAR

03 · MODELAR

04 · AVALIAR

JÁ ERA ASSIM NA FASE 1

Um número no fim: a métrica da entrega.

O QUE É NOVO NA FASE 2

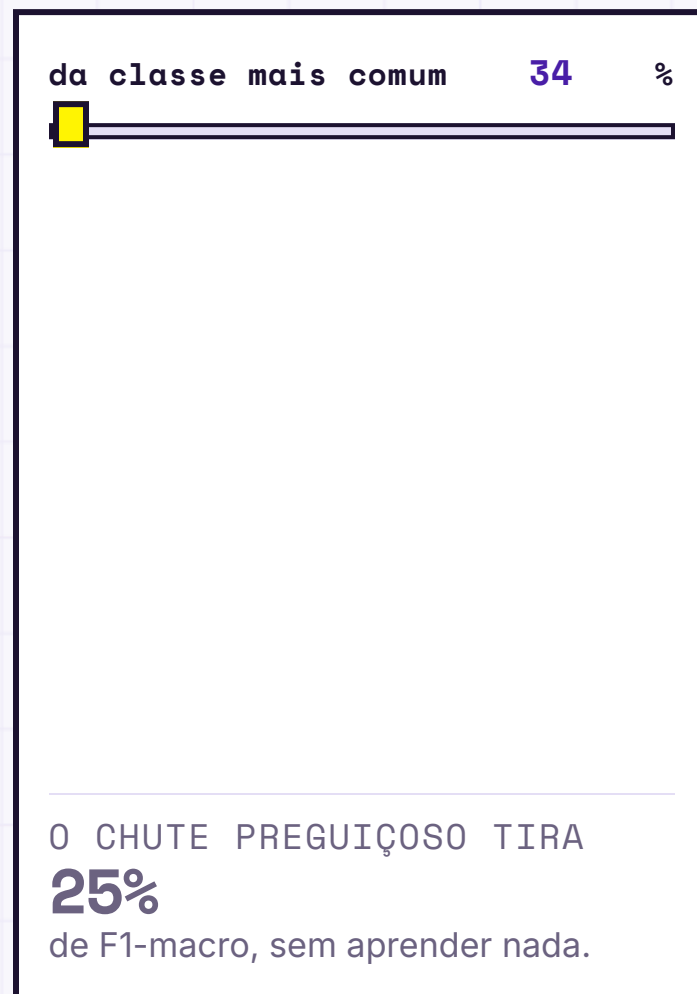
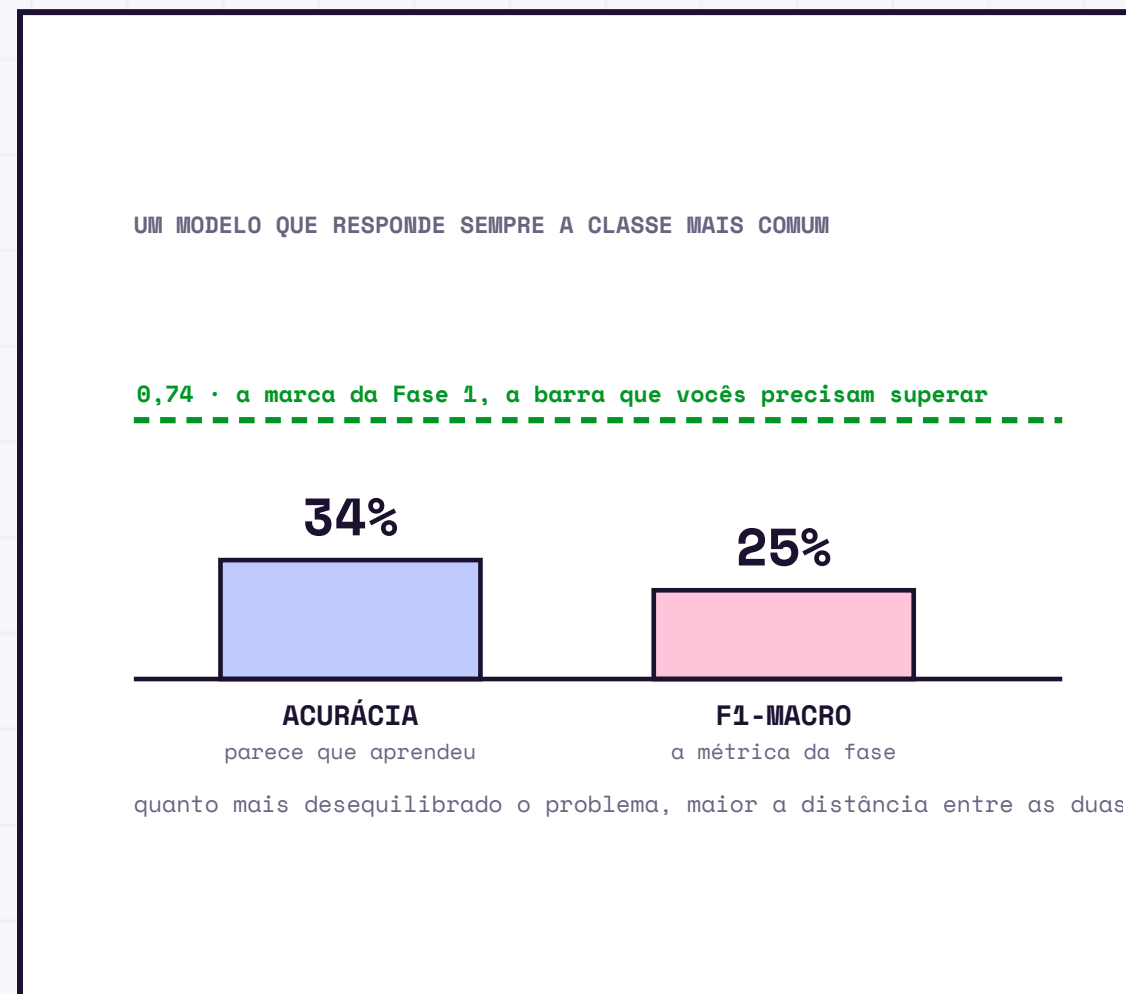
Agora são **duas curvas de perda** acompanhando o treino inteiro, uma **matriz de confusão** para ver onde o erro mora, e a conta de **custo contra ganho** entre os modelos.

A distância entre as duas curvas conta quase tudo.

O Bloco 4 não tem o que escrever: tem o que ler. É aqui que o formato das curvas conta o que aconteceu no treino, e é aqui que a equipe descobre em que gênero o modelo falha — que é a informação que o relatório precisa ter.

Antes de treinar, descubram o piso

Um modelo que não olha os dados e responde sempre a classe mais comum. Que nota ele tira?



A DISTÂNCIA ESTÁ ENORME

Com **34%** na classe maior, o chute preguiçoso tira **34%** de acurácia e só **25%** de F1-macro. Uma métrica premia a preguiça, a outra não — por isso o ranking usa o F1-macro.

Com as classes quase do mesmo tamanho, o chute preguiçoso — responder sempre a classe maior — rende pouco, e qualquer modelo supera isso com facilidade. Repare no F1-macro do chute: é esse número que vocês precisam anotar antes de treinar qualquer coisa.

Antes de treinar, descubram o piso

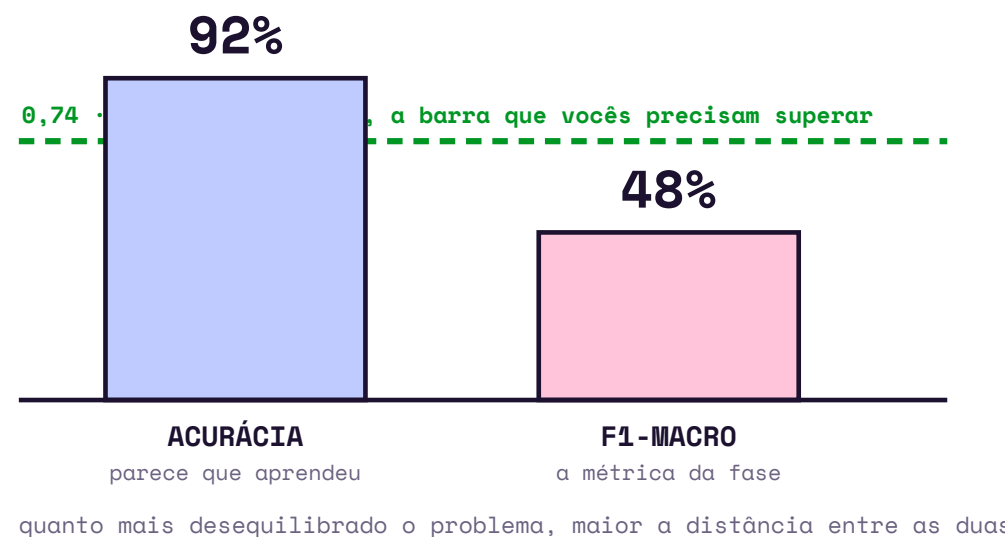
Um modelo que não olha os dados e responde sempre a classe mais comum. Que nota ele tira?

da classe mais comum 92 %



O CHUTE PREGUIÇOSO TIRA
48%
de F1-macro, sem aprender nada.

UM MODELO QUE RESPONDE SEMPRE A CLASSE MAIS COMUM



PROBLEMA EQUILIBRADO

Com **92%** na classe maior, o chute preguiçoso tira **92%** de acurácia e só **48%** de F1-macro. Uma métrica premia a preguiça, a outra não — por isso o ranking usa o F1-macro.

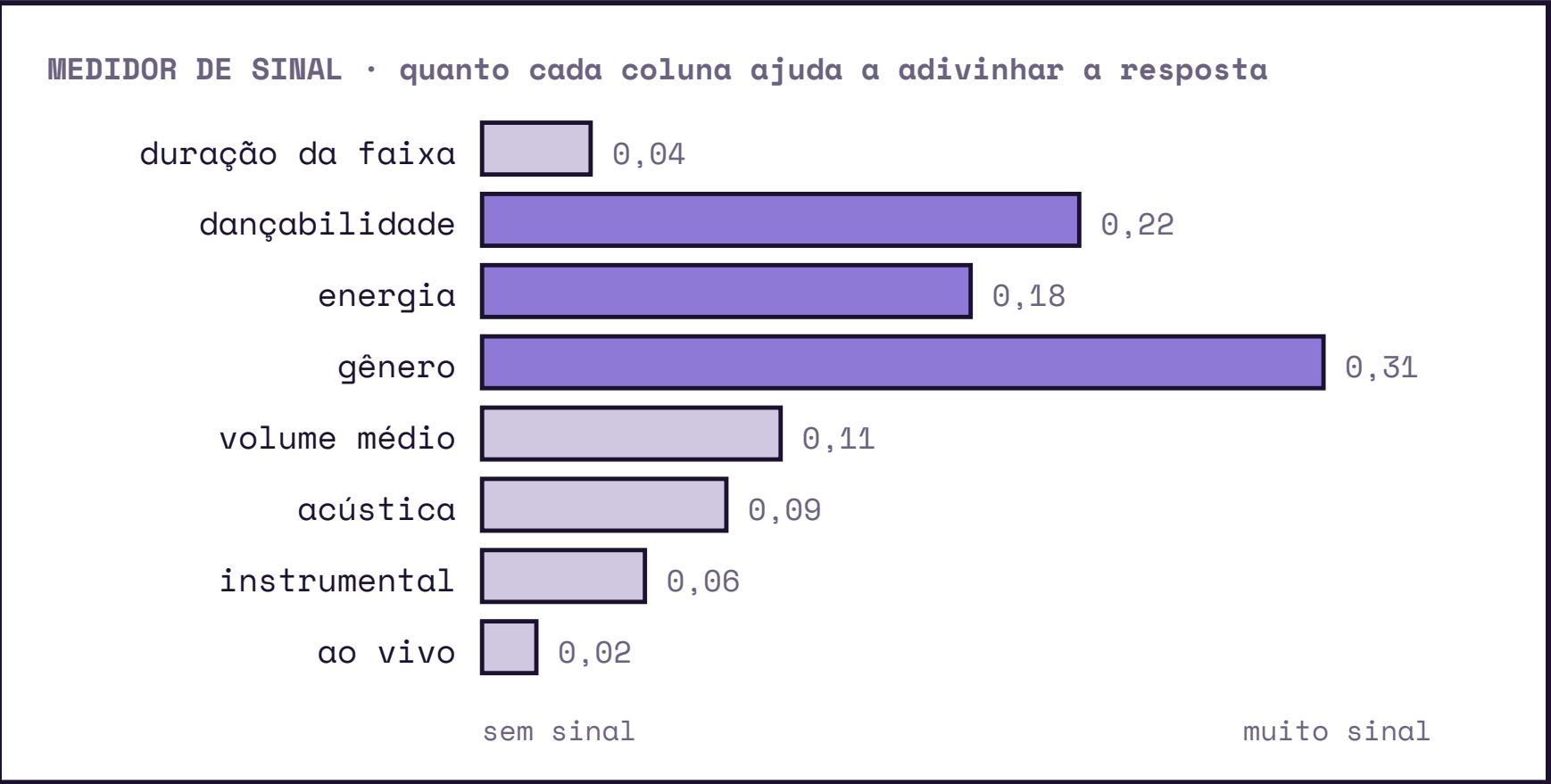
Agora com uma classe dominando. A acurácia do chute preguiçoso sobe junto com o desequilíbrio e chega a parecer um resultado — mas o F1-macro continua baixo, porque ele dá o mesmo peso à classe pequena. É exatamente por isso que a métrica da fase é o F1-macro, e não a acurácia: ela não deixa a classe pequena desaparecer na média.

Alguma coluna aqui ajuda?

Cada coluna ganha uma nota de 0 a 1: quanto ela, sozinha, diz sobre a resposta.

Problema com sinal

Problema sem sinal



ESTE PROBLEMA TEM ONDE SE APOIAR

Três colunas acima de 0,15 já dão material de trabalho ao modelo. Clique nas barras para ver cada uma.

Cada barra é uma coluna, e o tamanho é o quanto ela separa as classes sozinha. Numa tabela honesta nenhuma coluna resolve o problema por conta própria: o sinal está espalhado, e é a combinação que classifica. É esse formato que vocês querem ver.

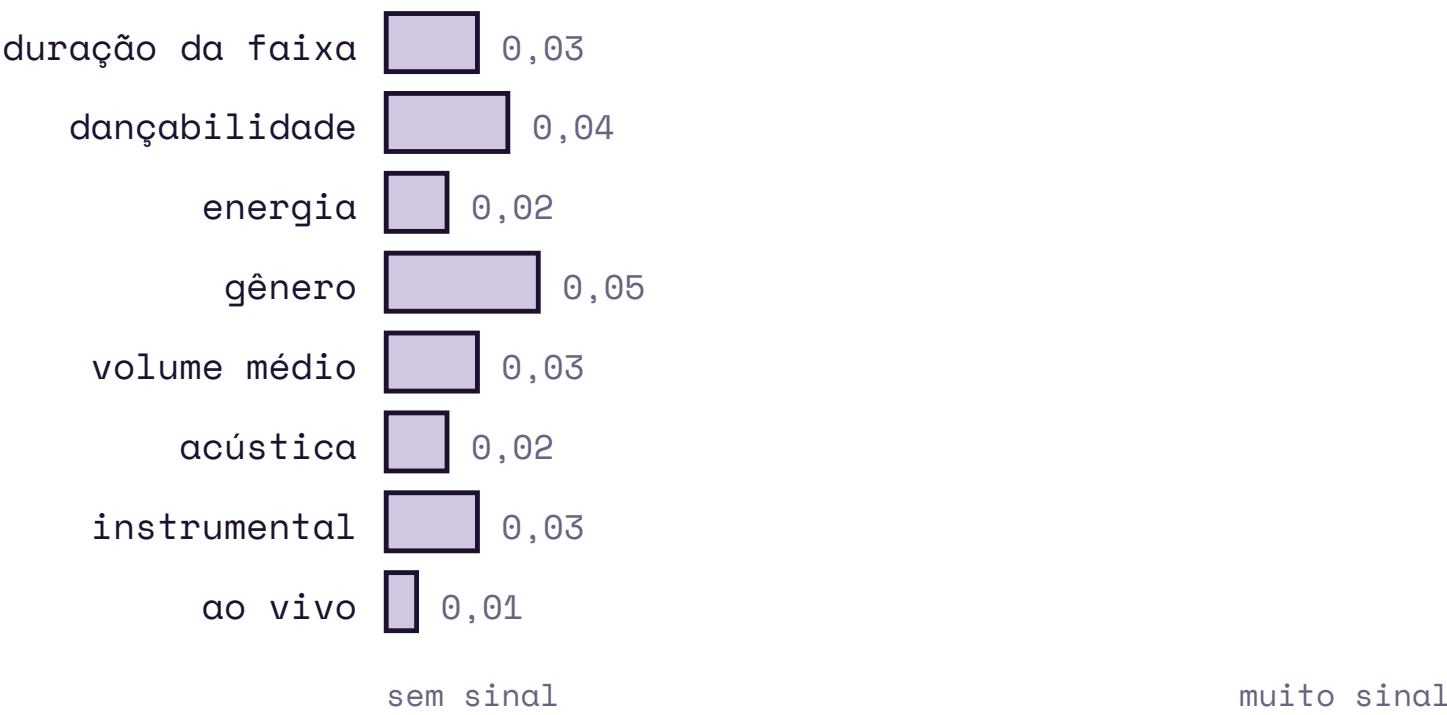
Alguma coluna aqui ajuda?

Cada coluna ganha uma nota de 0 a 1: quanto ela, sozinha, diz sobre a resposta.

Problema com sinal

Problema sem sinal

MEDIDOR DE SINAL · quanto cada coluna ajuda a adivinhar a resposta



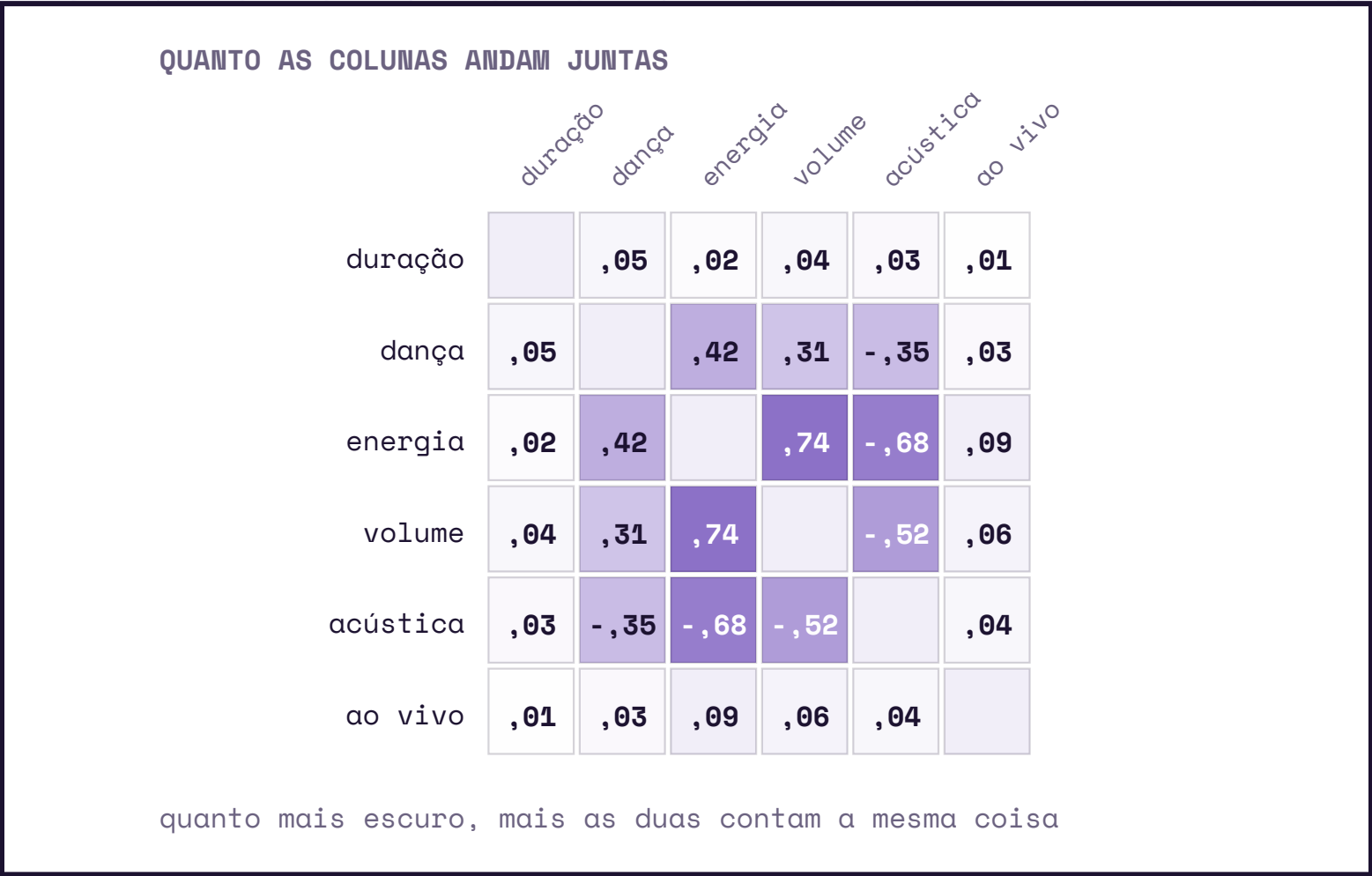
NENHUMA COLUNA PASSA DE 0,05

Num caso assim, nenhum modelo vai longe, por mais caro que seja. Descobrir isso no primeiro dia poupa a semana — e vira um parágrafo honesto no relatório.

Aqui uma coluna sozinha explica quase tudo. Isso quase nunca é sorte: é vazamento. Alguma informação que só existe depois de o rótulo ser conhecido entrou na tabela. Encontrar isso no primeiro dia poupa a semana; encontrar no último dia significa refazer o trabalho.

Duas colunas que contam a mesma coisa

Quanto mais escuro, mais as duas andam juntas. Informação repetida não soma.



O QUE ESTE MAPA MOSTRA

Colunas muito parecidas entre si são informação repetida: não somam nada e deixam o modelo mais lento. Cliquem num quadrado escuro.

Duas colunas que medem a mesma coisa por caminhos diferentes. Elas não atrapalham o modelo, mas atrapalham a leitura de importância do Bloco 4: o peso que deveria ser de uma se divide entre as duas, e nenhuma parece importante. Vale saber quais são antes de interpretar qualquer gráfico de importância.

Três números para anotar

Os três reaparecem no bloco seguinte.

Mostrar o primeiro (3)

01

0 piso

O F1-macro do chute preguiçoso. É a nota que o modelo de vocês precisa superar para ter aprendido alguma coisa.

02

As três colunas de maior sinal

São elas que o modelo vai usar. Também são as primeiras a olhar quando o resultado vier estranho.

03

Quantos valores tem cada categoria

Este é obrigatório. Sem esse número, dois dos três modelos nem rodam.

Descobrir no primeiro dia que nenhuma coluna ajuda poupa a semana inteira — e vira um parágrafo honesto no relatório.

OS TRÊS NÚMEROS PARA ANOTAR

Os três números que precisam sair do Bloco 1 escritos em algum lugar. O PISO é a nota do chute preguiçoso, e é contra ela que se mede se o modelo aprendeu alguma coisa — sem ela, um F1 de 0,30 pode ser bom resultado ou desastre, e não há como saber qual. AS TRÊS COLUNAS DE MAIOR SINAL são as primeiras a conferir quando o resultado vier estranho. E QUANTOS VALORES TEM CADA CATEGORIA é obrigatório: sem esse número, dois dos três modelos não rodam, porque a rede com embeddings precisa do tamanho de cada tabela de consulta e o TabNet precisa de duas listas — essa e a das posições. No caderno do desafio, nada disso é calculado para vocês.

Os três modelos desta trilha

Todos descritos pelos mesmos seis campos. Cliquem em um deles.

Rede densa

Rede densa com embeddings

TabNet

REDE DENSA · ENTRADA

Camadas de neurônios empilhadas, sem nenhuma ideia além disso.

ESCOLHAM SE

querem começar rápido e entender cada linha do modelo. Treina em minutos.

O QUE ESPERAR

Rápida, simples e mais competitiva do que parece. Não explica nada sobre as decisões.

JÁ VEM PRONTO

O otimizador, a parada antecipada e o laço de treino.

A PEGADINHA

Sem a régua nas colunas numéricas, o modelo trava. É a única das três que não perdoa isso.

A mais simples que ainda é rede neural. Tudo entra como número, e as categorias viram colunas de 0 e 1. É o ponto de partida honesto: se um modelo mais complicado não superar este, ele não se justifica.

Os três modelos desta trilha

Todos descritos pelos mesmos seis campos. Cliquem em um deles.

Rede densa

Rede densa com embeddings

TabNet

REDE DENSA COM EMBEDDINGS · INTERMEDIÁRIA

A mesma rede, mas cada categoria vira um punhado de números que o treino ajusta.

ESCOLHAM SE

a tabela tem colunas de categoria com muitos valores diferentes, como gênero musical.

O QUE ESPERAR

Costuma ganhar da rede densa quando há categorias grandes. Mesmo código, mesma velocidade.

JÁ VEM PRONTO

Tudo o que já vinha na rede densa, mais a construção dos embeddings.

A PEGADINHA

Ela precisa saber quantos valores diferentes cada coluna de categoria tem. Esse número vem do Bloco 1 — se estiver errado, quebra na hora de prever.

OS TRÊS MODELOS · COM EMBEDDINGS

Mesma rede, com uma diferença: cada valor de categoria ganha um vetor de números que a própria rede aprende durante o treino. Dois valores que funcionam de forma parecida acabam com vetores parecidos, e ninguém precisou ensinar isso. Brilha quando há categoria de cardinalidade alta.

Os três modelos desta trilha

Todos descritos pelos mesmos seis campos. Cliquem em um deles.

Rede densa

Rede densa com embeddings

TabNet

TABNET · AVANÇADA

Decide em vários passos e, em cada um, escolhe quais colunas vai olhar.

ESCOLHAM SE

o relatório de vocês vai se apoiar em explicar **por que** o modelo decidiu.

O QUE ESPERAR

É o único que mostra quais colunas usou em cada previsão. Tem mais botões para ajustar.

JÁ VEM PRONTO

A normalização de entrada e o laço de treino, que aqui é parecido com o da Fase 1.

A PEGADINHA

O tamanho do lote virtual precisa dividir exatamente o tamanho do lote. Se não dividir, erro na hora de rodar.

Arquitetura feita para tabela. Em vez de olhar todas as colunas de uma vez, decide a cada passo quais olhar — e por isso consegue dizer, no fim, quais usou. É o único dos três que entrega essa informação de graça.

A rede por dentro

A mesma rede densa do Aulão 1, agora com as colunas da tabela na entrada.

camadas escondidas1

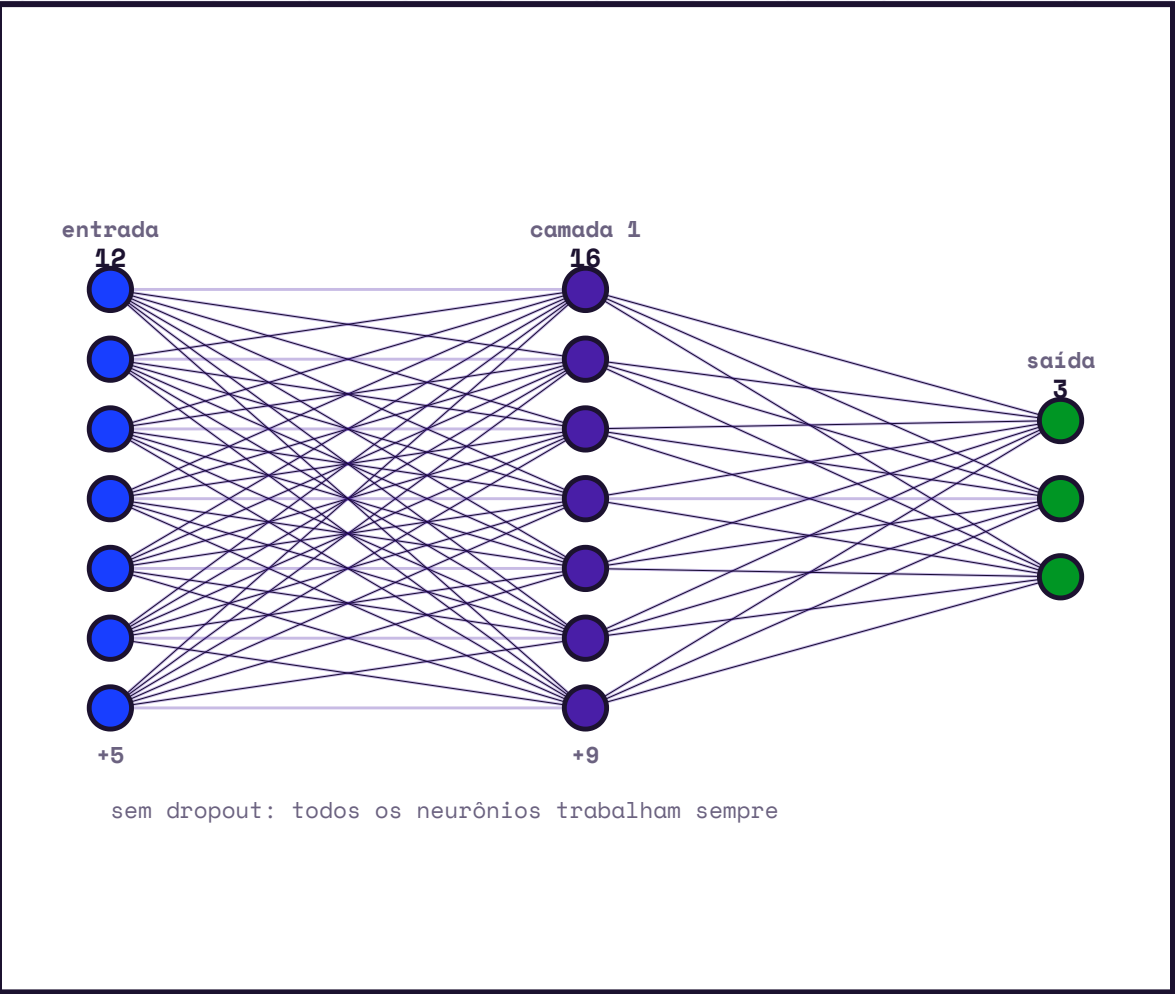
neurônios por camada16

dropout0,00

PESOS PARA AJUSTAR

259

12 colunas na entrada · 1 camada de 16 · 3 classes na saída



REDE PEQUENA DEMAIS

Com tão poucos pesos a rede não consegue representar as combinações entre as colunas. As duas curvas param altas e juntas: é subajuste.

Poucos neurônios por camada. O diagrama mostra o que a rede tem para trabalhar, e o painel mostra quantos números ela vai ajustar. Rede pequena demais não consegue aprender nem os exemplos que viu: é o subajuste, e no Bloco 4 ele aparece como duas curvas que param altas e juntas.

A rede por dentro

A mesma rede densa do Aulão 1, agora com as colunas da tabela na entrada.

camadas escondidas4

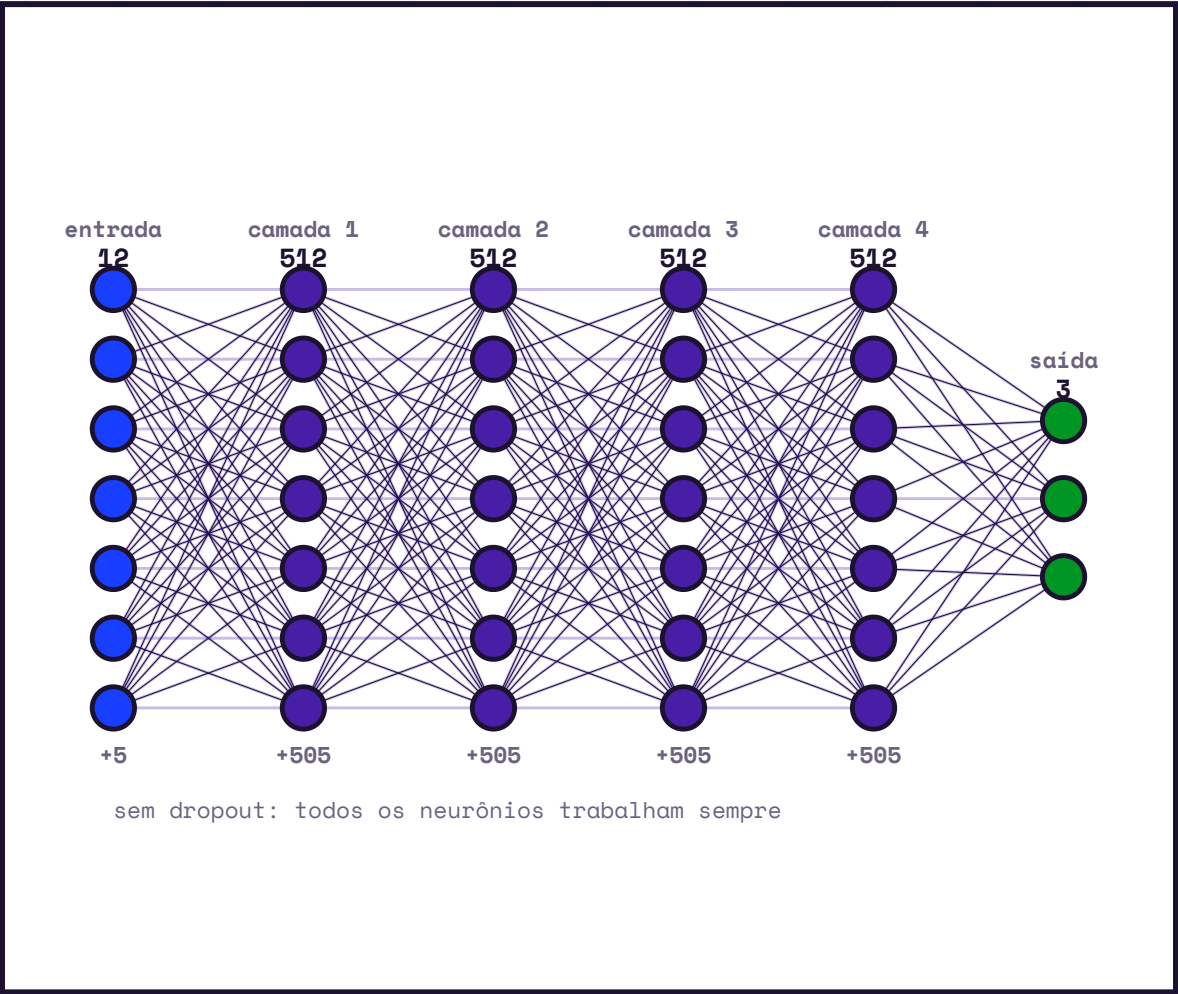
neurônios por camada512

dropout0,00

PESOS PARA AJUSTAR

796.163

12 colunas na entrada · 4 camadas de 512 · 3 classes na saída



MUITA REDE, POUCO FREIO

796.163 pesos para ajustar com uma tabela de milhares de linhas. Sem dropout, essa rede decora o treino: a curva de validação vai subir enquanto a de treino desce.

Mesma tela, agora com mais camadas e mais neurônios. Repare no salto no número de parâmetros: ele cresce muito mais rápido que a largura, porque cada camada se liga inteira à seguinte. Capacidade sobrando não é vantagem por si — é convite ao sobreajuste, e precisa de freio.

A rede por dentro

A mesma rede densa do Aulão 1, agora com as colunas da tabela na entrada.

camadas escondidas 4

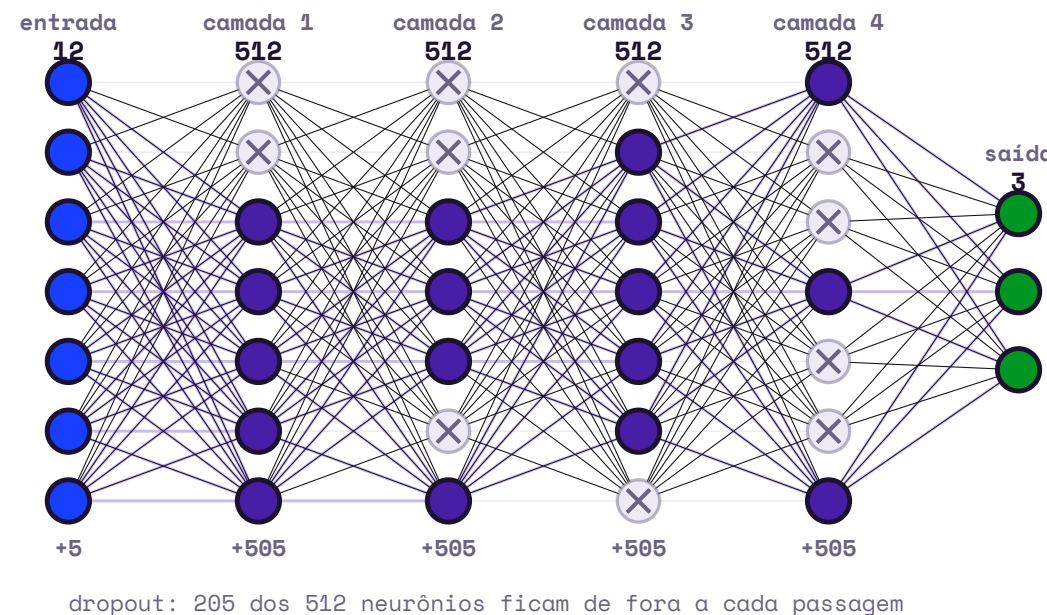
neurônios por camada 512

dropout 0,40

PESOS PARA AJUSTAR

796.163

12 colunas na entrada · 4 camadas de 512 · 3 classes na saída



PROPORÇÃO RAZOÁVEL

796.163 pesos com dropout de 0,40: a cada passagem do treino 205 neurônios de cada camada ficam de fora, e a rede aprende a não depender de nenhum deles.

A mesma rede larga, agora com dropout. Os neurônios marcados foram desligados ao acaso nesta passagem, e as ligações que saíam deles ficaram apagadas. A cada passagem o sorteio muda, e isso obriga a rede a espalhar o que aprendeu em vez de depender de um neurônio só. Acontece apenas no treino: na validação todos voltam a ligar.

Os hiperparâmetros e as curvas

Movam um botão e vejam o gráfico mudar antes de rodar qualquer treino.

Rede densa

Rede densa com embeddings

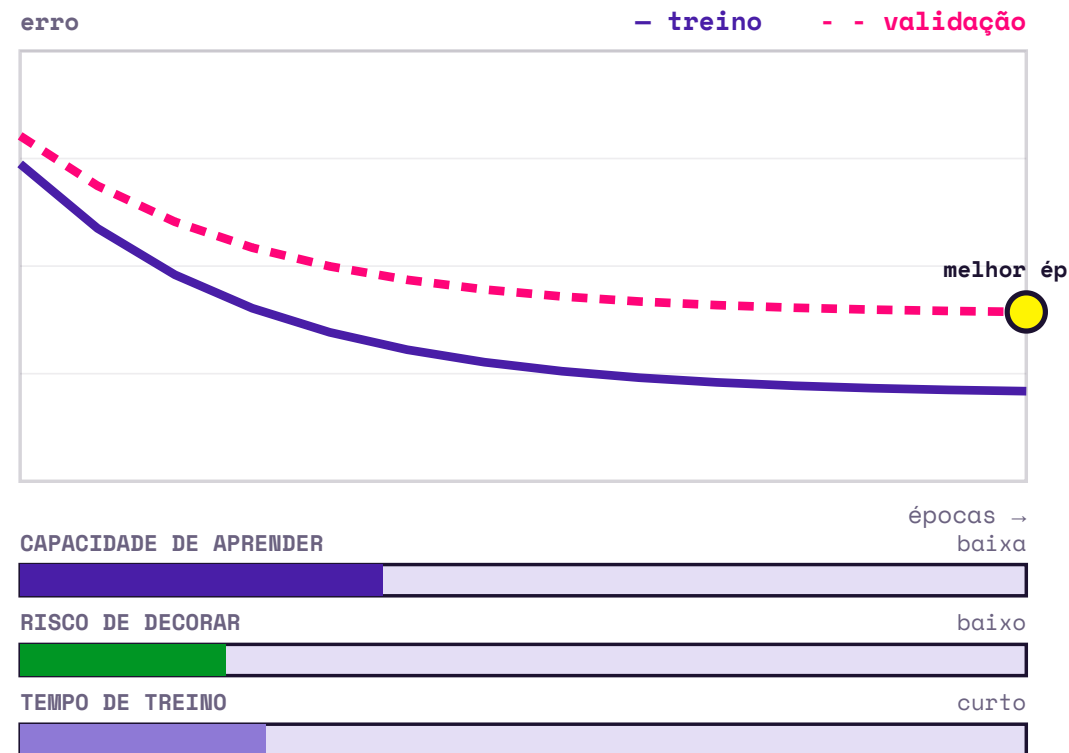
TabNet

largura 128 neurônios

dropout 0,30

taxa de aprendizado 1e-3

taxa de aprendizado. O tamanho do passo. O recomendado é 1e-3: a rede começa do zero e precisa andar.



MELHOR ÉPOCA DA VALIDAÇÃO

14 de 14

é o modelo que o ponto de verificação guarda · perda 0,43

TREINO SAUDÁVEL

As duas descem e a distância entre elas se estabiliza. É esta a forma que vocês querem ver no caderno.

Rede densa com a taxa de aprendizado em 1e-3, que é a recomendada para uma rede treinada do zero. As duas curvas descem juntas e a distância entre elas se estabiliza: é esta a forma que vocês querem ver no caderno do desafio.

Os hiperparâmetros e as curvas

Movam um botão e vejam o gráfico mudar antes de rodar qualquer treino.

Rede densa

Rede densa com embeddings

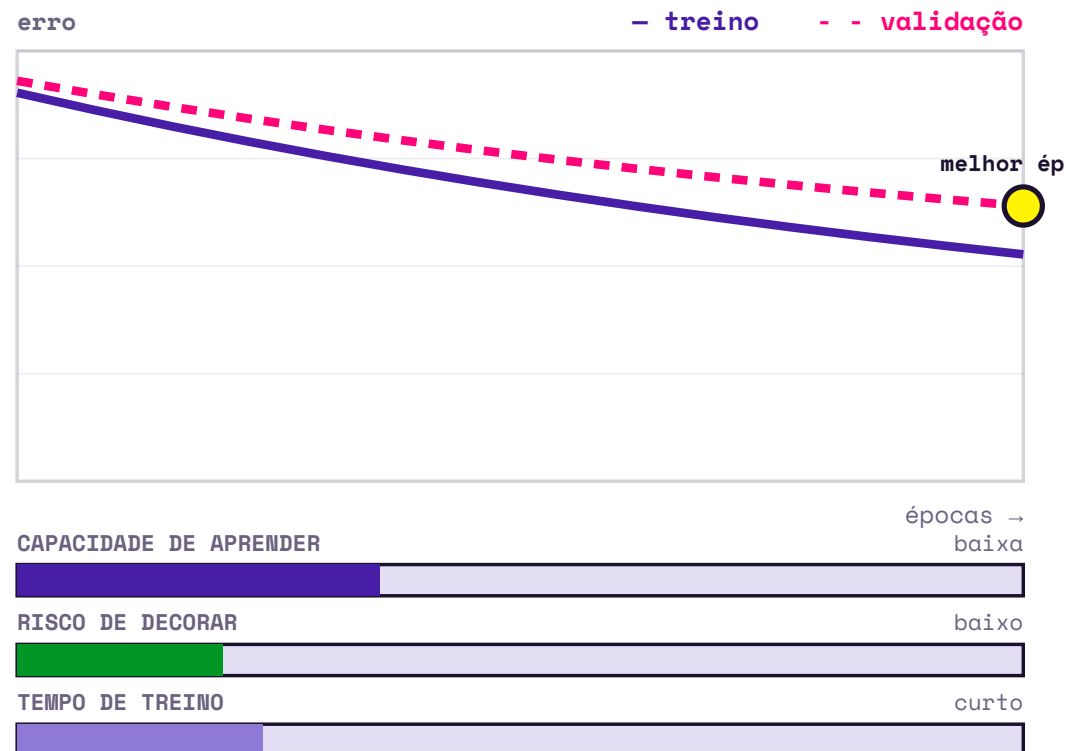
TabNet

largura 128 neurônios

dropout 0,30

taxa de aprendizado 1e-4

taxa de aprendizado. O tamanho do passo. O recomendado é 1e-3: a rede começa do zero e precisa andar.



MELHOR ÉPOCA DA VALIDAÇÃO

14 de 14

é o modelo que o ponto de verificação guarda · perda 0,70

TAXA BAIXA DEMAIS

As duas curvas descem, mas devagar: o passo é tão pequeno que o treino acaba antes de chegar ao fundo do vale. Não é falta de capacidade — é falta de passo. Multipliquem a taxa por dez.

Mesma rede, taxa dividida por dez: 1e-4. Compare com a página anterior. As curvas continuam lisas e descendo, sem serrilha nenhuma — só descem devagar demais, e o treino acaba antes de chegar ao fundo. O sintoma da taxa baixa é lentidão, não instabilidade. Em números: a perda final fica bem acima, e a melhor época é a última, sinal de que ainda havia ganho disponível.

Os hiperparâmetros e as curvas

Movam um botão e vejam o gráfico mudar antes de rodar qualquer treino.

Rede densa

Rede densa com embeddings

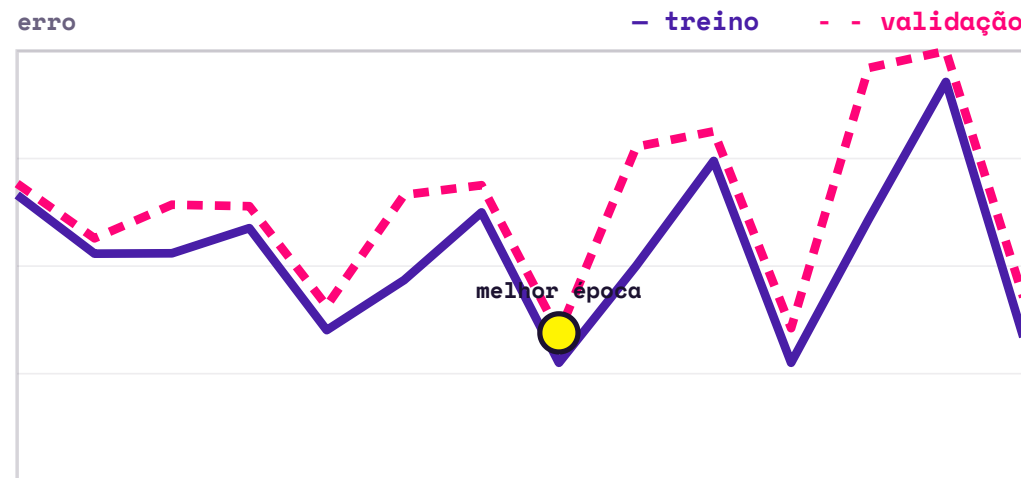
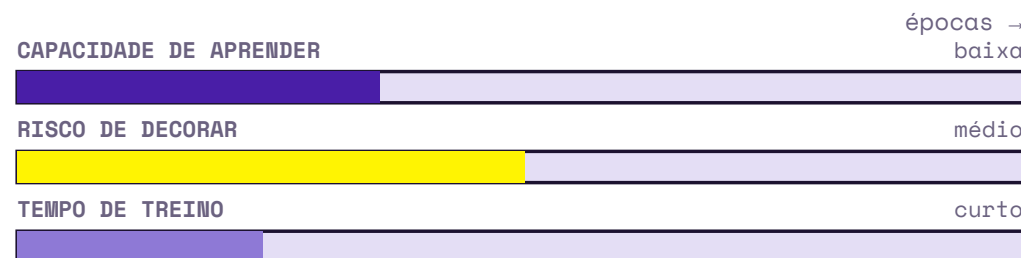
TabNet

largura 128 neurônios

dropout 0,30

taxa de aprendizado 1e-2

taxa de aprendizado. O tamanho do passo. O recomendado é 1e-3: a rede começa do zero e precisa andar.



MELHOR ÉPOCA DA VALIDAÇÃO

8 de 14

é o modelo que o ponto de verificação guarda · perda 0,49

TAXA ALTA DEMAIS

A curva serrilha e sobe: o passo está grande demais e o treino está explodindo. Dividam a taxa por dez.

Agora a taxa multiplicada por dez: 1e-2. O formato é completamente diferente do anterior, e esta é a lição do laboratório. A curva serrilha e sobe depois do mínimo: o passo é grande demais e o treino passa do ponto a cada atualização. Taxa baixa e taxa alta são dois problemas diferentes, com dois formatos diferentes — confundi-los faz a equipe mexer no botão na direção errada.

Os hiperparâmetros e as curvas

Movam um botão e vejam o gráfico mudar antes de rodar qualquer treino.

Rede densa

Rede densa com embeddings

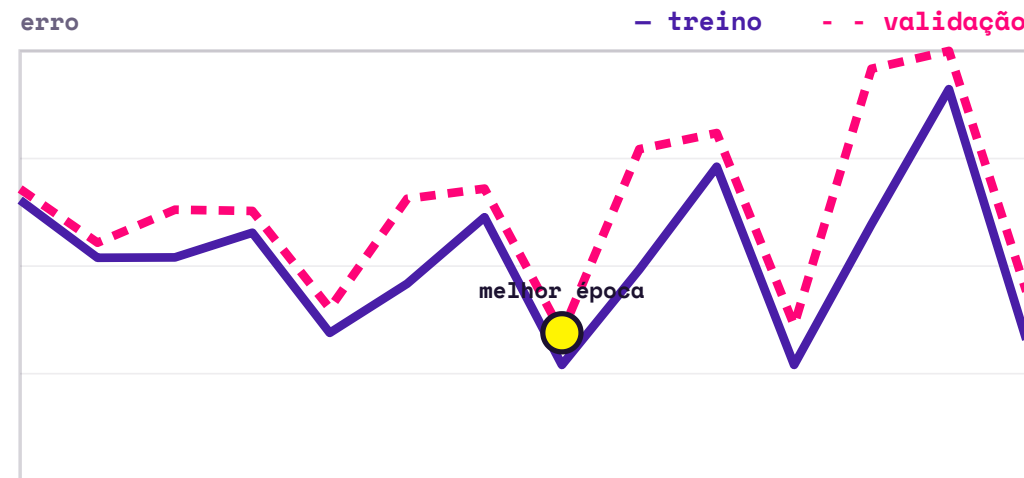
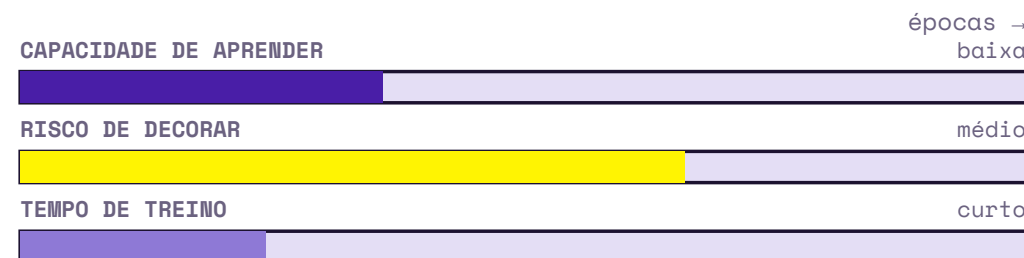
TabNet

largura 128 neurônios

dropout 0,00

taxa de aprendizado 1e-2

dropout. O freio do sobreajuste: desliga neurônios ao acaso durante o treino. Comecem em 0,30.



MELHOR ÉPOCA DA VALIDAÇÃO

8 de 14

é o modelo que o ponto de verificação guarda · perda 0,50

TAXA ALTA DEMAIS

A curva serrilha e sobe: o passo está grande demais e o treino está explodindo. Dividam a taxa por dez.

Taxa de volta ao recomendado, rede larga e dropout em zero. O treino continua caindo e a validação vira para cima: a rede passou a decorar os exemplos. É para este momento que existem a parada antecipada e o ponto de verificação, e é por isso que o modelo entregue é o da melhor época, não o da última.

Os hiperparâmetros e as curvas

Movam um botão e vejam o gráfico mudar antes de rodar qualquer treino.

Rede densa

Rede densa com embeddings

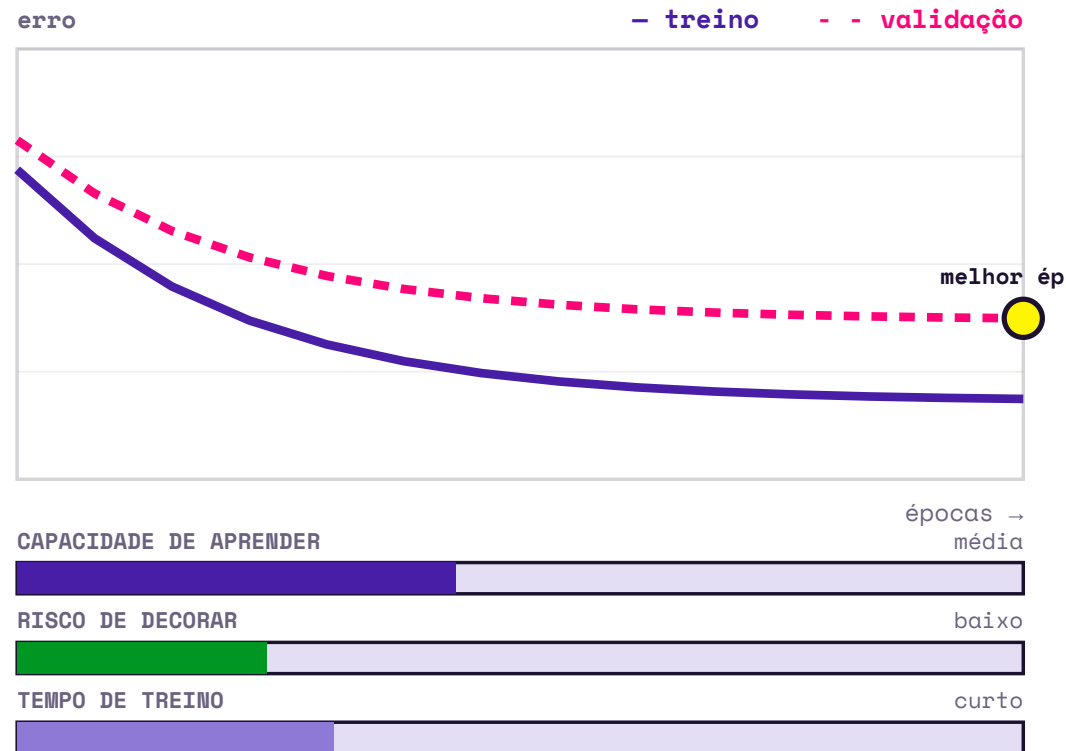
TabNet

tamanho do embedding 8 números

largura 128 neurônios

dropout 0,30

tamanho do embedding. Quantos números cada categoria ganha. Pequeno já resolve: 8 costuma bastar.



MELHOR ÉPOCA DA VALIDAÇÃO

14 de 14

é o modelo que o ponto de verificação guarda · perda 0,41

TREINO SAUDÁVEL

As duas descem e a distância entre elas se estabiliza. É esta a forma que vocês querem ver no caderno.

Modelo dois. Os controles mudam: aqui entra o tamanho do embedding, que é quantos números cada categoria ganha. Pequeno já resolve — oito costuma bastar, e dar 24 números a uma coluna de 2 valores só acrescenta parâmetro para decorar.

Os hiperparâmetros e as curvas

Movam um botão e vejam o gráfico mudar antes de rodar qualquer treino.

Rede densa

Rede densa com embeddings

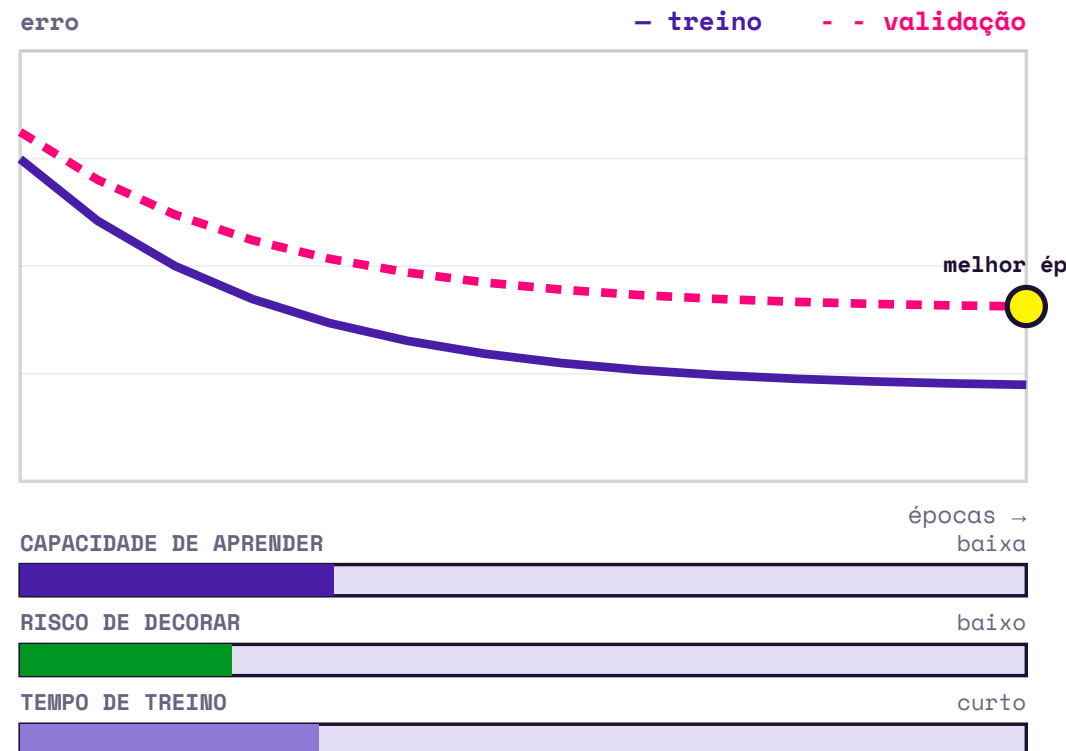
TabNet

n_steps 4 passos

n_d / n_a 16

taxa de aprendizado 2e-2

taxa de aprendizado. A mais alta das três: 2e-2. Este modelo foi feito para começar rápido e ir freando.



MELHOR ÉPOCA DA VALIDAÇÃO

14 de 14

é o modelo que o ponto de verificação guarda · perda 0,45

TREINO SAUDÁVEL

As duas descem e a distância entre elas se estabiliza. É esta a forma que vocês querem ver no caderno.

Modelo três, e repare na taxa: 2e-2, vinte vezes a da rede densa. O TabNet foi feito para começar rápido e ir freando, e usar nele a taxa da rede densa deixa o treino inacabado dentro do limite de épocas. Cada arquitetura tem a sua faixa, e copiar a do vizinho é um erro comum.

0 mesmo dado, três preparos

A maior diferença para a Fase 1: agora cada modelo quer a entrada de um jeito.

Rede densa**Rede densa com embeddings****TabNet**

célula A · preparo categorias: **colunas de 0 e 1** · régua: **obrigatória**

```
# categorias viram colunas de 0 e 1
X = pd.get_dummies(X, columns=categoricas)

# a régua é ajustada SÓ no treino
regua = StandardScaler().fit(X_tr)
X_tr = regua.transform(X_tr)
X_va = regua.transform(X_va)
```

POR QUE ASSIM

A rede densa soma tudo junto. Se uma coluna vai até 200 e outra até 1, a primeira domina sem ter mais importância — por isso a régua é obrigatória. E as categorias viram colunas de zero e um porque o número da categoria não tem ordem: gênero 7 não é maior que gênero 3.

O mesmo dado, três preparos diferentes — e esta é a única célula que muda de verdade entre os três modelos. Para a rede densa, cada categoria se abre em colunas de 0 e 1. O motivo é que o número de uma categoria não tem ordem: dizer que a tonalidade 11 está onze unidades acima da tonalidade 0 é falso, porque tonalidade é um ciclo.

0 mesmo dado, três preparos

A maior diferença para a Fase 1: agora cada modelo quer a entrada de um jeito.

Rede densa**Rede densa com embeddings****TabNet**

célula A · preparo categorias: **índices inteiros** · régua: **obrigatória**

```
# categorias viram números inteiros
X[categoricas] = X[categoricas].apply(codificar)

# o modelo precisa saber quantos valores tem cada uma
cardinalidades = [114, 2, 12, 2, 5]

regua = StandardScaler().fit(X_tr[numericas])
```

POR QUE ASSIM

Aqui as categorias viram índices e o modelo aprende sozinho um punhado de números para cada valor — o embedding. Por isso ele precisa saber, de antemão, quantos valores diferentes cada coluna tem. É o número que vocês anotaram no Bloco 1.

Para a rede com embeddings, a categoria vira um índice inteiro e o modelo precisa saber quantos valores existem — é o número anotado no Bloco 1. E a régua vale só para as colunas numéricas: padronizar um índice de categoria destrói justamente a informação que o embedding vai usar.

0 mesmo dado, três preparos

A maior diferença para a Fase 1: agora cada modelo quer a entrada de um jeito.

Rede densa

Rede densa com embeddings

TabNet

célula A · preparo categorias: **índices inteiros** · régua: **dispensável**

```
# categorias viram números inteiros
X[categoricas] = X[categoricas].apply(codificar)
idx_cat = [0, 3, 7, 8, 9]
dim_cat = [114, 2, 12, 2, 5]

# sem régua: o modelo normaliza por dentro
```

POR QUE ASSIM

O TabNet tem uma camada de normalização logo na entrada, então ele mesmo cuida da escala: uma etapa a menos para vocês. Em compensação, quer a posição e o tamanho de cada coluna de categoria, nas duas listas.

Para o TabNet, duas listas: onde cada coluna de categoria está, e quantos valores ela tem. Trocar uma pela outra é o engano mais comum desta trilha, e o erro não aparece como mensagem clara — aparece como índice fora da tabela, ou como um modelo que treina e não aprende.

Três células, sempre as mesmas

Esta forma se repete nas três trilhas e nos nove modelos. Cliquem em cada uma.

CÉLULA A

Preparo

Carrega os dados, separa treino e validação e transforma a entrada. É **a única célula que muda de verdade** entre os três modelos — foi o laboratório anterior.

CÉLULA B

Treino

Constrói o modelo, configura e treina. É onde ficam as lacunas de hiperparâmetro e a **Zona Livre**. O laço de treino já vem escrito.

CÉLULA C

Avaliação

Calcula as métricas, desenha a matriz de confusão e as curvas. Vem pronta, sem lacuna: ela não é para escrever, é **para ler**.

Aprendam esta forma uma vez. Ela vale para os nove modelos do material.

Esta forma se repete nas três trilhas e nos nove modelos do material de imersão. A célula A prepara, a B constrói e treina, a C avalia. A C vem pronta e sem lacuna de propósito: ela não é para escrever, é para ler. E o laço de treino, dentro da B, também já vem escrito — o que vocês decidem é como configurá-lo.

0 caminho que o código percorre

O laço vem escrito no caderno. Não é para escrever: é para entender a ordem.

```
# uma volta por época: o modelo vê o conjunto inteiro
for epoca in range(epocas):

    # uma volta por lote: cada lote gera UM ajuste dos pesos
    for lote, resposta in dados_de_treino:

        otimizador.zero_grad()
        saida = modelo(lote)
        erro = perda(saida, resposta)
        erro.backward()
        otimizador.step()
```

< volta

Linha 2 >

1 · APAGA O PASSO ANTERIOR

Zera os ajustes calculados na volta passada. Sem isso, eles se acumulam e o modelo anda na direção errada.

O caminho que o código percorre a cada lote, passo por passo. Esta sequência está pronta no caderno do desafio, mas entendê-la é o que permite ler a mensagem de erro quando algo quebra.

0 caminho que o código percorre

O laço vem escrito no caderno. Não é para escrever: é para entender a ordem.

```
# uma volta por época: o modelo vê o conjunto inteiro
for epoca in range(epocas):

    # uma volta por lote: cada lote gera UM ajuste dos pesos
    for lote, resposta in dados_de_treino:

        otimizador.zero_grad()
        saida = modelo(lote)
        erro = perda(saida, resposta)
        erro.backward()
        otimizador.step()
```

< volta

Linha 3 >

2 · PASSAGEM PARA A FRENTE

O lote atravessa a rede e sai o palpite. É a conta que vocês viram no Aulão 1, repetida camada por camada.

Segundo passo. Repare que a ordem não é negociável: cada etapa depende do resultado da anterior, e inverter duas delas produz um treino que roda e não aprende — sem dar erro nenhum.

0 caminho que o código percorre

O laço vem escrito no caderno. Não é para escrever: é para entender a ordem.

```
# uma volta por época: o modelo vê o conjunto inteiro
for epoca in range(epocas):

    # uma volta por lote: cada lote gera UM ajuste dos pesos
    for lote, resposta in dados_de_treino:

        otimizador.zero_grad()
        saida = modelo(lote)
        erro = perda(saida, resposta)
        erro.backward()
        otimizador.step()
```

< volta

Linha 4 >

3 · O QUANTO ERROU

Compara o palpite com a resposta certa e devolve um número só: a perda daquele lote.

Terceiro passo. Pergunta que vale pensar antes de seguir: o que aconteceria se os gradientes do lote anterior não fossem zerados antes de calcular os novos? A resposta explica por que existe uma etapa que parece não fazer nada.

0 caminho que o código percorre

O laço vem escrito no caderno. Não é para escrever: é para entender a ordem.

```
# uma volta por época: o modelo vê o conjunto inteiro
for epoca in range(epocas):

    # uma volta por lote: cada lote gera UM ajuste dos pesos
    for lote, resposta in dados_de_treino:

        otimizador.zero_grad()
        saida = modelo(lote)
        erro = perda(saida, resposta)
        erro.backward()
        otimizador.step()
```

< volta

Linha 5 >

4 · PROPAGA O ERRO PARA TRÁS

O backpropagation descobre, para cada peso, em que direção mexer para o erro diminuir.

Quarto passo. É aqui que o erro se transforma em direção: para cada peso, de que lado mexer e o quanto.

0 caminho que o código percorre

O laço vem escrito no caderno. Não é para escrever: é para entender a ordem.

```
# uma volta por época: o modelo vê o conjunto inteiro
for epoca in range(epocas):

    # uma volta por lote: cada lote gera UM ajuste dos pesos
    for lote, resposta in dados_de_treino:

        otimizador.zero_grad()
        saida = modelo(lote)
        erro = perda(saida, resposta)
        erro.backward()
        otimizador.step()
```

< volta

✓ volta completa

5 · ANDA

Move cada peso naquela direção, no tamanho da taxa de aprendizado. Só aqui o modelo muda de verdade.

Último passo, e o único em que o modelo muda de verdade. Os quatro passos anteriores calculam; este aplica. Daqui o laço volta ao começo com o lote seguinte, e repete até a época acabar.

Quatro peças cuidam do treino por vocês

Elas já vêm no caderno. Liguem e desliguem para ver o que cada uma muda na curva.

✓ OTIMIZADOR

AdamW

✓ AGENDADOR

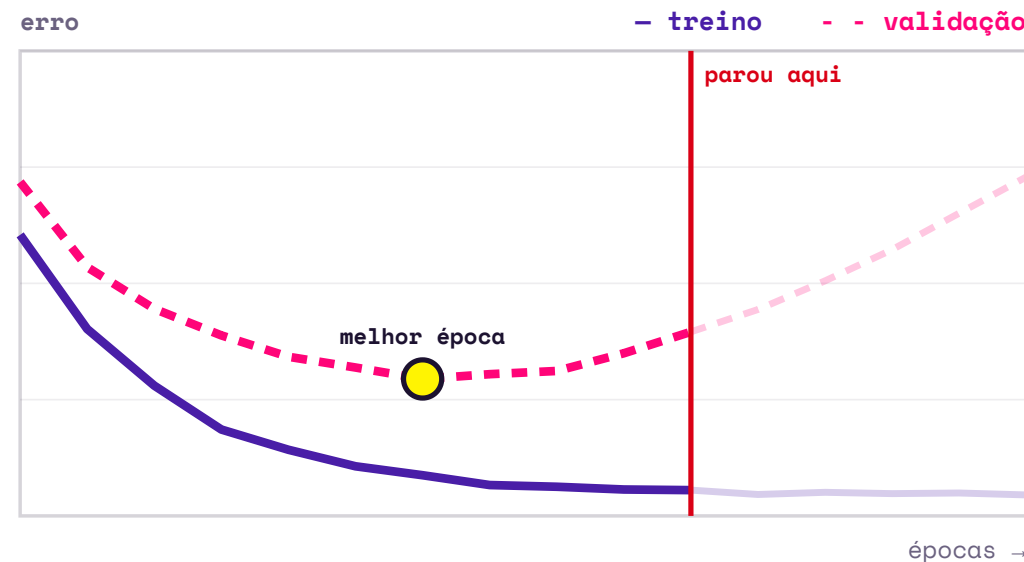
ligado

✓ PARADA ANTECIPADA

ligada

✓ PONTO DE VERIFICAÇÃO

ligado



parada antecipada ligada · parou na época 11 de 16

ponto de verificação ligado · entrega o modelo da melhor época

O MODELO QUE VOCÊS ENTREGAM

0,32

perda de validação da época 7, a melhor de todas

PONTO DE VERIFICAÇÃO · LIGADO

Ponto de verificação ligado.

Uma cópia do modelo é salva na melhor época, não na última. É ele que vocês entregam.

As quatro peças que cuidam do treino: otimizador, agendador, parada antecipada e ponto de verificação. Todas ligadas, a curva tem o formato que se quer e o modelo entregue é o da melhor época. As páginas seguintes desligam uma de cada vez.

Quatro peças cuidam do treino por vocês

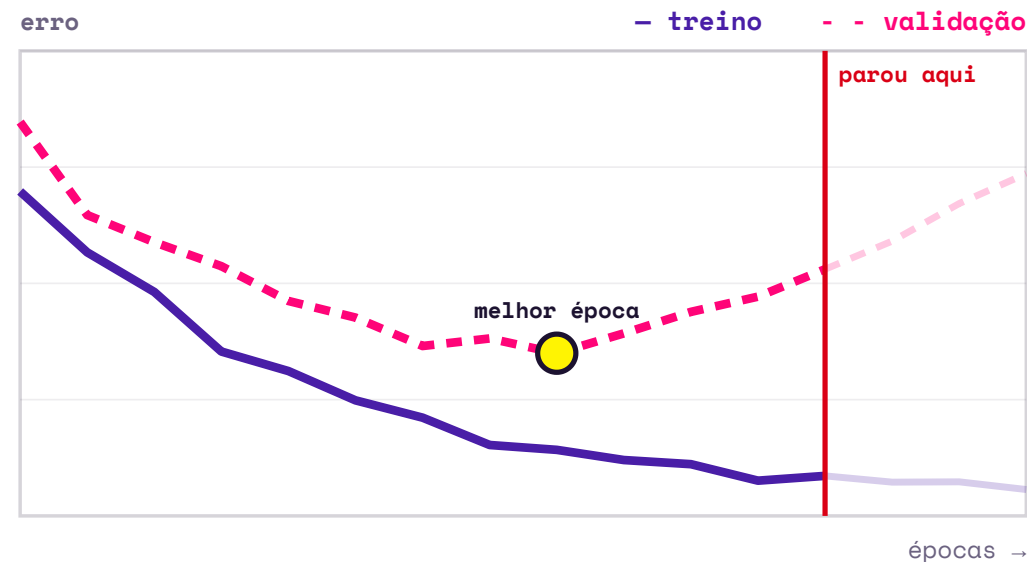
Elas já vêm no caderno. Liguem e desliguem para ver o que cada uma muda na curva.

× OTIMIZADOR
SGD simples

✓ AGENDADOR
ligado

✓ PARADA ANTECIPADA
ligada

✓ PONTO DE VERIFICAÇÃO
ligado



parada antecipada ligada · parou na época 13 de 16
ponto de verificação ligado · entrega o modelo da melhor época

O MODELO QUE VOCÊS ENTREGAM

0,38

perda de validação da época 9,
a melhor de todas

**OTIMIZADOR · SGD
SIMPLES**

SGD simples. Anda sempre no mesmo tamanho de passo, sem memória. A curva desce mais devagar e treme mais — dá para chegar no mesmo lugar, levando mais épocas.

Otimizador desligado. A curva desce muito mais devagar e com mais ruído: sem adaptação por peso, o mesmo passo serve bem para alguns pesos e mal para outros. É a peça que mais muda a velocidade do treino.

Quatro peças cuidam do treino por vocês

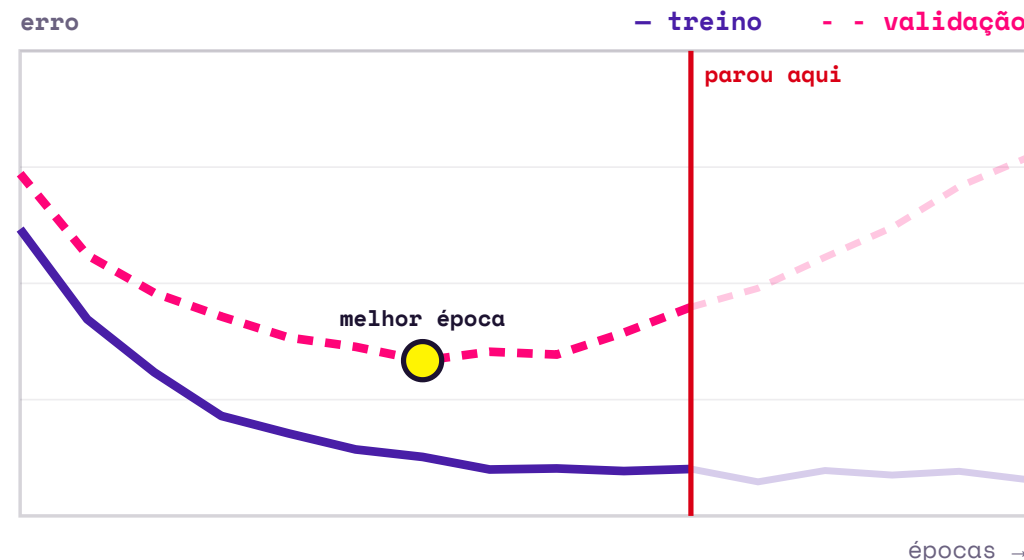
Elas já vêm no caderno. Liguem e desliguem para ver o que cada uma muda na curva.

✓ OTIMIZADOR
AdamW

× AGENDADOR
desligado

✓ PARADA ANTECIPADA
ligada

✓ PONTO DE VERIFICAÇÃO
ligado



parada antecipada ligada · parou na época 11 de 16
ponto de verificação ligado · entrega o modelo da melhor época

O MODELO QUE VOCÊS ENTREGAM

0,37

perda de validação da época 7, a melhor de todas

AGENDADOR · DESLIGADO

Agendador desligado. O passo continua do mesmo tamanho até o fim. Perto do fundo do vale, passos grandes fazem o modelo passar do ponto, e a curva fica tremida.

Agendador desligado. O passo fica do mesmo tamanho do início ao fim, e a curva estaciona mais alto: faltou o ajuste fino do final, quando a rede precisa de passos pequenos para assentar em vez de pular em volta do mínimo.

Quatro peças cuidam do treino por vocês

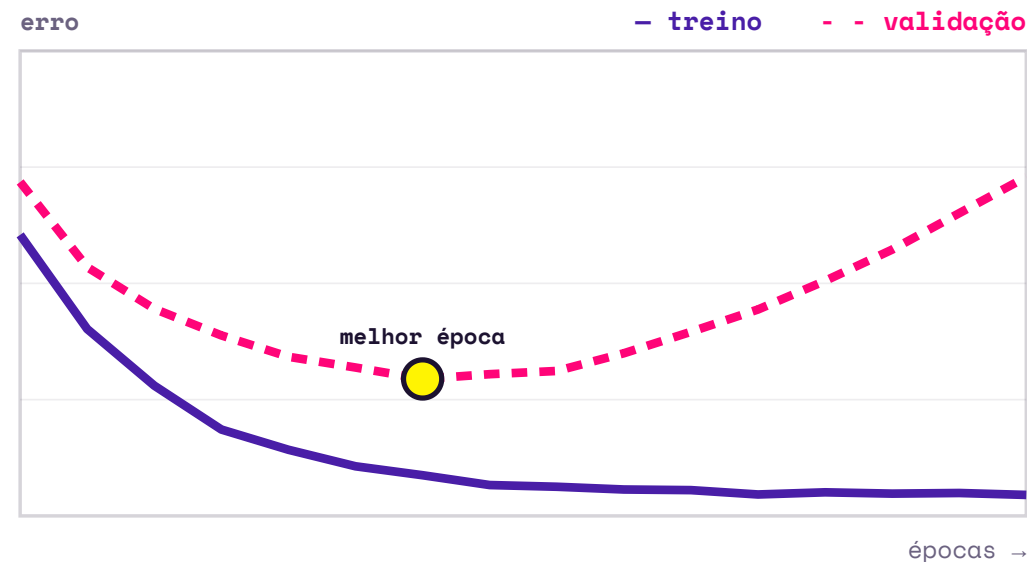
Elas já vêm no caderno. Liguem e desliguem para ver o que cada uma muda na curva.

✓ OTIMIZADOR
AdamW

✓ AGENDADOR
ligado

× PARADA ANTECIPADA
desligada

✓ PONTO DE VERIFICAÇÃO
ligado



sem parada antecipada · o treino vai até a época 16, mesmo piorando
ponto de verificação ligado · entrega o modelo da melhor época

O MODELO QUE VOCÊS ENTREGAM

0,32

perda de validação da época 7, a melhor de todas

PARADA ANTECIPADA · DESLIGADA

Parada antecipada desligada. O treino vai até a última época programada, mesmo depois que a validação começou a piorar. Gasta tempo e piora o resultado.

Parada antecipada desligada. O treino vai até a última época mesmo depois de a validação piorar, gastando tempo para ficar pior. Treinar mais não é treinar melhor, e este é o gráfico que mostra isso.

Quatro peças cuidam do treino por vocês

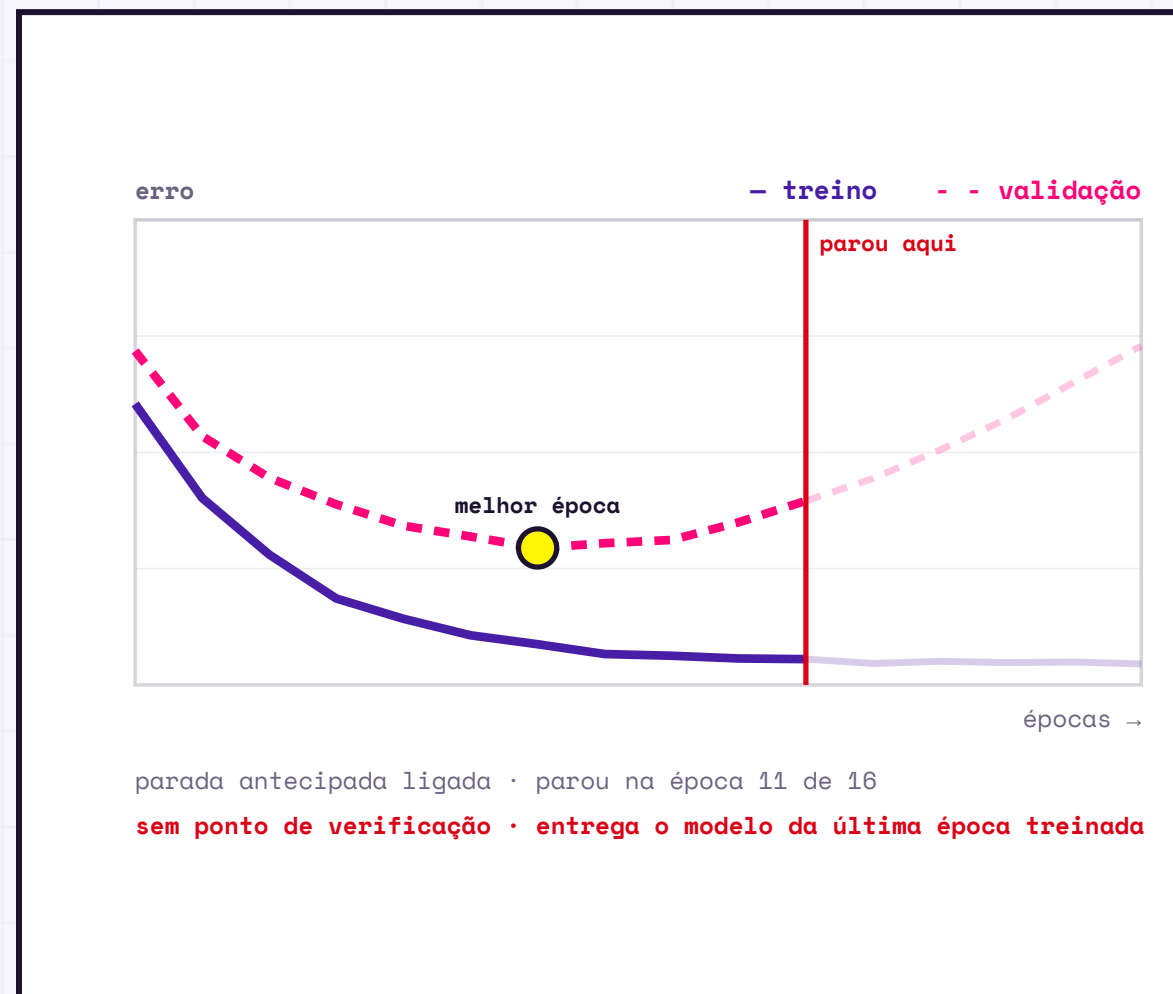
Elas já vêm no caderno. Liguem e desliguem para ver o que cada uma muda na curva.

✓ OTIMIZADOR
AdamW

✓ AGENDADOR
ligado

✓ PARADA ANTECIPADA
ligada

× PONTO DE VERIFICAÇÃO
desligado



O MODELO QUE VOCÊS ENTREGAM

0,44

perda de validação da época 11,
a última treinada — **0,11 pior** que
a melhor

**PONTO DE VERIFICAÇÃO ·
DESLIGADO**

Ponto de verificação desligado.
Vocês entregam o modelo da
última época treinada. Se a
validação já tinha piorado, o
trabalho das épocas boas foi
jogado fora.

Esta é a mais sutil das quatro. A parada antecipada interrompe o treino DEPOIS de a validação piorar por algumas épocas — e naquele instante os pesos na memória já são os piores. Sem o ponto de verificação, a parada detecta o sobreajuste e entrega o modelo sobreajustado mesmo assim. Repare qual época fica marcada como entregue.

Completem a célula A · preparo

É o código do caderno de vocês. Escolham juntos o que entra em cada espaço.

Rede densa

Rede densa com embeddings

TabNet

```
# == CÉLULA A · preparo ==  
# categoria vira coluna de 0 e 1: o número não tem ordem  
X = pd.get_dummies(X, columns=categoricas)  
  
# separa mantendo a proporção das classes  
X_tr, X_va, y_tr, y_va = train_test_split(X, y, test_size=0.2, stratify=y)  
  
# a régua aprende média e desvio SÓ no treino  
regua = StandardScaler().fit(X_tr)  
X_tr, X_va = regua.transform(X_tr), regua.transform(X_va)
```

3 de 3 certas · 0 em branco

recomeçar

LACUNA 3 DE 3 · JÁ RESPONDIDA

responder de novo

✓ CÉLULA COMPLETA

As 3 lacunas estão certas. Assim a célula roda — e sem o erro silencioso, que não dá mensagem.

A célula A da rede densa, com as lacunas já resolvidas. Duas decisões importam aqui: quais colunas se abrem em 0 e 1, e com qual conjunto a régua aprende. A segunda é a que mais machuca quando sai errada, porque não dá erro — dá uma nota alta que não vale nada.

Completem a célula A · preparo

É o código do caderno de vocês. Escolham juntos o que entra em cada espaço.

Rede densa

Rede densa com embeddings

TabNet

```
# == CÉLULA A · preparo ==  
# a categoria vira inteiro; o modelo aprende um vetor por valor  
X[categoricas] = X[categoricas].apply(codificar)  
  
# o modelo precisa saber quantos valores tem cada categoria  
# é o número que vocês anotaram no Bloco 1  
cardinalidades = [114, 2, 12, 2, 5]  
  
X_tr, X_va, y_tr, y_va = train_test_split(X, y, test_size=0.2, stratify=y)  
  
# a régua vale só para as colunas numéricas  
regua = StandardScaler().fit(X_tr[numéricas])
```

3 de 3 certas · 0 em branco

recomeçar

LACUNA 3 DE 3 · JÁ RESPONDIDA

responder de novo

✓ CÉLULA COMPLETA

As 3 lacunas estão certas. Assim a célula roda — e sem o erro silencioso, que não dá mensagem.

A mesma célula para a rede com embeddings. Aqui aparece a lista de cardinalidades — o terceiro número do Bloco 1 — e a régua passa a valer só para as colunas numéricas, porque as categóricas já são índices.

Completem a célula A · preparo

É o código do caderno de vocês. Escolham juntos o que entra em cada espaço.

Rede densa

Rede densa com embeddings

TabNet

```
# == CÉLULA A · preparo ==  
X[categoricas] = X[categoricas].apply(codificar)  
  
# onde cada coluna de categoria está na tabela  
idx_cat = [0, 3, 7, 8, 9]  
# e quantos valores diferentes cada uma delas tem  
dim_cat = [114, 2, 12, 2, 5]  
  
X_tr, X_va, y_tr, y_va = train_test_split(X, y, test_size=0.2, stratify=y)  
  
# aqui NÃO entra régua: o TabNet normaliza por dentro
```

3 de 3 certas · 0 em branco

recomeçar

LACUNA 3 DE 3 · JÁ RESPONDIDA

responder de novo

✓ CÉLULA COMPLETA

As 3 lacunas estão certas. Assim a célula roda — e sem o erro silencioso, que não dá mensagem.

E para o TabNet, as duas listas. Olhem as opções erradas que o laboratório oferece: elas são exatamente os enganos que acontecem de verdade. Trocar posições por cardinalidades é o primeiro deles.

A régua é ajustada depois da separação

Troquem a ordem e vejam o que acontece. Nenhuma mensagem de erro aparece nos dois casos.

✓ régua depois da separação

× régua antes da separação

```
# 1 · separa primeiro
X_tr, X_va = separar(X, y)

# 2 · a régua aprende SÓ com o treino
regua = StandardScaler().fit(X_tr)
X_tr = regua.transform(X_tr)
X_va = regua.transform(X_va)
```

NA VALIDAÇÃO

0,76

NA PLATAFORMA

0,75

A ORDEM CERTA

A validação continua sendo dado novo: a régua nunca olhou para ela. O número que aparece na validação é o que vocês podem esperar da plataforma.

A RÉGUA · O JEITO ERRADO

A régua ajustada antes da separação. O código roda, nenhuma mensagem aparece, e a nota de validação sobe. O problema é que a média e o desvio usados para normalizar já incluíam as faixas da validação: o modelo soube delas antes da hora, e a estimativa de desempenho virou ficção.

A régua é ajustada depois da separação

Troquem a ordem e vejam o que acontece. Nenhuma mensagem de erro aparece nos dois casos.

✓ régua depois da separação

× régua antes da separação

```
# 1 · a régua aprende com a tabela INTEIRA
regua = StandardScaler().fit(X)
X = regua.transform(X)

# 2 · só depois separa
X_tr, X_va = separar(X, y)
```

NA VALIDAÇÃO

0,81

NA PLATAFORMA

0,69

O VAZAMENTO

A régua levou para o treino uma informação que veio da validação — a média e a escala do conjunto todo. É vazamento, com outra roupa: a validação melhora e a plataforma piora. E nenhuma mensagem de erro aparece.

A régua ajustada depois da separação, e só com o treino. A nota cai — e essa nota menor é a verdadeira. Guardem o nome: vazamento. É o erro mais caro desta fase justamente porque ele não dá erro, e só aparece quando o modelo encontra dados de verdade.

Completem a célula B · treino

Aqui ficam os hiperparâmetros e as quatro peças que vocês acabaram de ver.

Rede densa

Rede densa com embeddings

TabNet

```
# == CÉLULA B · treino ==
# dropout é o freio do sobreajuste desta rede
modelo = RedeDensa(entradas=X_tr.shape[1], saidas=3, dropout=0.3)

# quem move os pesos depois que o erro apontou a direção
otimizador = AdamW(modelo.parameters(), lr=1e-3)

# o passo vai encurtando ao longo do treino
agendador = CosineAnnealingLR(otimizador, T_max=60)

# guarda a melhor época, não a última
treino = Config(epocas=60, parada_antecipada=10, ponto_de_verificacao=True)
```

5 de 5 certas · 0 em branco

recomeçar

LACUNA 5 DE 5 · JÁ RESPONDIDA

responder de novo

✓ CÉLULA COMPLETA

As 5 lacunas estão certas. Assim a célula roda — e sem o erro silencioso, que não dá mensagem.

A célula B da rede densa, resolvida. Aqui ficam os hiperparâmetros e as quatro peças que vocês acabaram de ligar e desligar. Repare que o laço de treino não aparece: ele já vem escrito, e o que se escreve é a configuração.

Completem a célula B · treino

Aqui ficam os hiperparâmetros e as quatro peças que vocês acabaram de ver.

Rede densa

Rede densa com embeddings

TabNet

```
# == CÉLULA B · treino ==  
# cada categoria vira um vetor deste tamanho  
modelo = RedeComEmbeddings(cardinalidades=cardinalidades,  
                             dim_embedding=8, dropout=0.3)  
  
otimizador = AdamW(modelo.parameters(), lr=1e-3)  
  
# para quando a validação parar de melhorar  
treino = Config(epocas=60, parada_antecipada=10, ponto_de_verificacao=True)
```

3 de 3 certas · 0 em branco

recomeçar

LACUNA 3 DE 3 · JÁ RESPONDIDA

responder de novo

✓ CÉLULA COMPLETA

As 3 lacunas estão certas. Assim a célula roda — e sem o erro silencioso, que não dá mensagem.

A mesma célula para os embeddings. Menos lacunas, porque parte da configuração é a mesma — e é esse o ganho de aprender a forma uma vez: o que muda entre modelos é pouco e fica evidente.

Completem a célula B · treino

Aqui ficam os hiperparâmetros e as quatro peças que vocês acabaram de ver.

Rede densa

Rede densa com embeddings

TabNet

```
# == CÉLULA B · treino ==  
# n_steps é quantas vezes ele decide quais colunas olhar  
modelo = TabNetClassifier(cat_idx=idx_cat, cat_dims=dim_cat, n_steps=5,  
                           optimizer_params=dict(lr=2e-2))  
  
# o lote virtual precisa dividir o lote EXATAMENTE  
modelo.fit(X_tr, y_tr, batch_size=1024, virtual_batch_size=128,  
           max_epochs=60, patience=10)
```

4 de 4 certas · 0 em branco

recomeçar

LACUNA 4 DE 4 · JÁ RESPONDIDA

responder de novo

✓ CÉLULA COMPLETA

As 4 lacunas estão certas. Assim a célula roda — e sem o erro silencioso, que não dá mensagem.

E a do TabNet. Atenção à última lacuna: o lote virtual precisa dividir o lote exatamente. Essa é uma exigência da biblioteca, não um conceito de aprendizado de máquina — mas quebra o treino na primeira época, e vale saber antes.

A Zona Livre da célula B

Quem só preenche as lacunas tem resultado correto e mediano. Acrescentem um ajuste por vez.

```
modelo = RedeDensa(  
  # == ZONA LIVRE ==  
  
  # ==  
  
)
```

< tirar

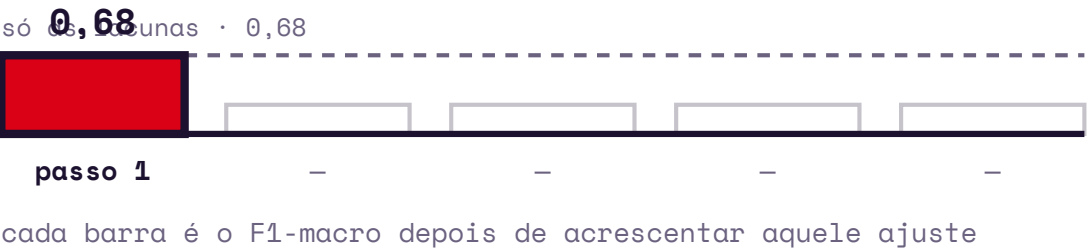
Acrescentar um ajuste >

F1-MACRO NA VALIDAÇÃO

0,68

Só as lacunas preenchidas

O QUE CADA AJUSTE ACRESCENTA



SÓ AS LACUNAS PREENCHIDAS

Tudo no valor padrão. Correto, e mediano — é o resultado que todo mundo alcança.

Zona Livre vazia: é o resultado de quem preencheu as lacunas e parou ali. Correto e mediano, e o número à direita mostra onde isso chega. As páginas seguintes acrescentam um ajuste por vez.

A Zona Livre da célula B

Quem só preenche as lacunas tem resultado correto e mediano. Acrescentem um ajuste por vez.

```
modelo = RedeDensa(
    # == ZONA LIVRE ==
    dropout=0.3,
    # =====
)
```

< tirar

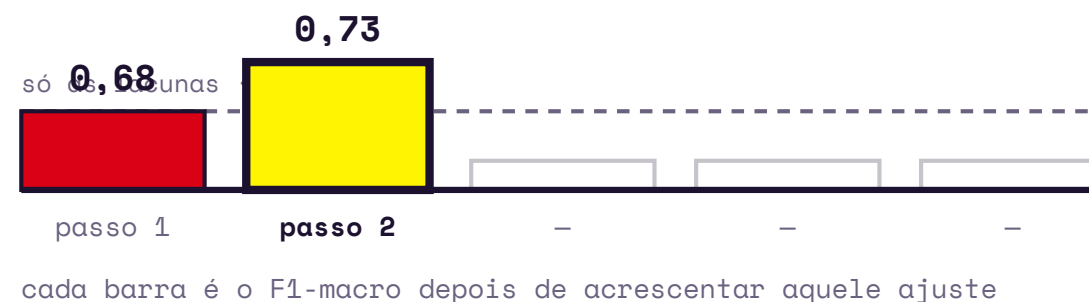
Acrescentar um ajuste >

F1-MACRO NA VALIDAÇÃO

0,73

+ dropout

O QUE CADA AJUSTE ACRESCENTA



+ DROPOUT

O freio do sobreajuste desta rede. Desliga neurônios ao acaso no treino, e o modelo para de decorar.

Primeiro ajuste acrescentado. Um só, e o efeito no F1 já é visível. O método importa tanto quanto o ajuste: mudar uma coisa por vez é o que permite saber a qual decisão o ganho pertence.

A Zona Livre da célula B

Quem só preenche as lacunas tem resultado correto e mediano. Acrescentem um ajuste por vez.

```
modelo = RedeDensa(  
    # == ZONA LIVRE ==  
    dropout=0.3,  
    largura=[256, 128],  
    # =====  
)
```

< tirar

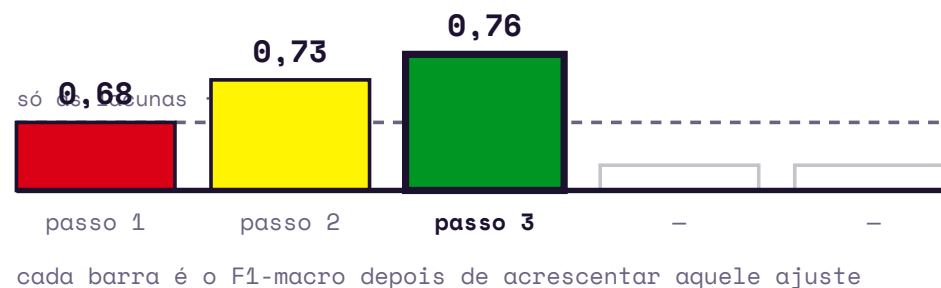
Acrescentar um ajuste >

F1-MACRO NA VALIDAÇÃO

0,76

+ camadas mais largas

O QUE CADA AJUSTE ACRESCENTA



+ CAMADAS MAIS LARGAS

Mais capacidade para aprender padrões. Só funciona porque o dropout está segurando o sobreajuste: os dois andam juntos.

Segundo ajuste: camadas mais largas. Repare que ele só funciona porque o dropout da página anterior está segurando o sobreajuste — capacidade e freio andam juntos, e acrescentar capacidade sem freio é o erro da última página deste laboratório.

A Zona Livre da célula B

Quem só preenche as lacunas tem resultado correto e mediano. Acrescentem um ajuste por vez.

```
modelo = RedeDensa(  
    # == ZONA LIVRE ==  
    dropout=0.3,  
    largura=[256, 128],  
    peso_por_classe="balanceado",  
    # =====  
)
```

< tirar

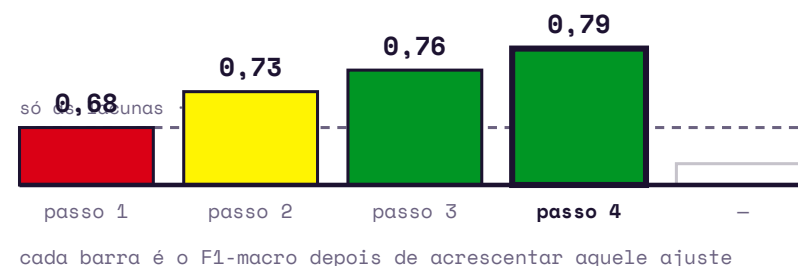
Acrescentar um ajuste >

F1-MACRO NA VALIDAÇÃO

0,79

+ peso por classe

O QUE CADA AJUSTE ACRESCENTA



+ PESO POR CLASSE

A classe menor deixa de ser sacrificada. Como a métrica é o F1-macro, isso costuma render bastante.

Terceiro ajuste: peso por classe. Como a métrica da fase é o F1-macro, deixar de sacrificar a classe menor costuma render bastante — e este é o melhor resultado do laboratório. Guardem este número para comparar com a próxima página.

A Zona Livre da célula B

Quem só preenche as lacunas tem resultado correto e mediano. Acrescentem um ajuste por vez.

```
modelo = RedeDensa(
    # == ZONA LIVRE ==
    dropout=0.0,
    largura=[1024, 1024, 1024],
    # =====
)
```

< tirar

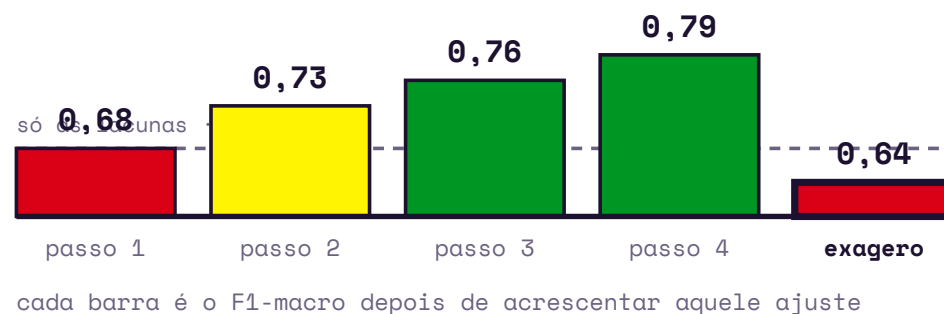
✓ fim da demonstração

F1-MACRO NA VALIDAÇÃO

0,64

exagero: rede enorme sem freio

O QUE CADA AJUSTE ACRESCENTA



EXAGERO: REDE ENORME SEM FREIO

Mais nem sempre é melhor. Rede gigante sem dropout decora o treino e desaba na validação.

E aqui está a razão de o laboratório não terminar no melhor número. Rede enorme, dropout em zero: o F1 cai para baixo do ponto de partida, pior do que não ter mexido em nada. Mais nem sempre é melhor, e a Zona Livre pune tanto quanto premia. O método que protege disso é o mesmo de sempre: um ajuste por vez, anotando o efeito — assim dá para voltar atrás no que piorou em vez de descobrir no fim que o conjunto regrediu.

Perda de treino e perda de validação

Na Fase 1 vocês tinham um número no fim. Agora vocês têm um filme.

época 4 de 12

PERDA DE TREINO

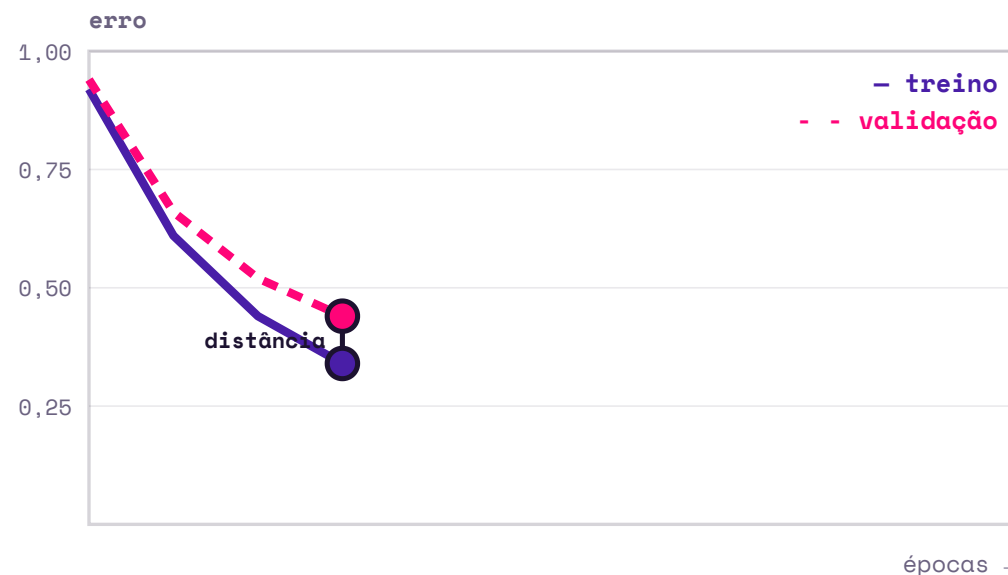
0,34

PERDA DE VALIDAÇÃO

0,44

A DISTÂNCIA ENTRE ELAS

0,10



TREINO

Erro nos exemplos que o modelo **usa para aprender**. Quase sempre cai: ele pode decorar.

VALIDAÇÃO

Erro em exemplos que ele **nunca usou**. É esta que importa.

AS DUAS ESTÃO CAINDO

Treino e validação descem juntas. É exatamente o que vocês querem ver: o modelo está aprendendo padrões que valem para casos novos.

As duas perdas nas primeiras épocas. Elas caem juntas e quase colam uma na outra: o modelo está aprendendo o que generaliza, e ainda não teve tempo de decorar nada.

Perda de treino e perda de validação

Na Fase 1 vocês tinham um número no fim. Agora vocês têm um filme.

época 7 de 12

PERDA DE TREINO

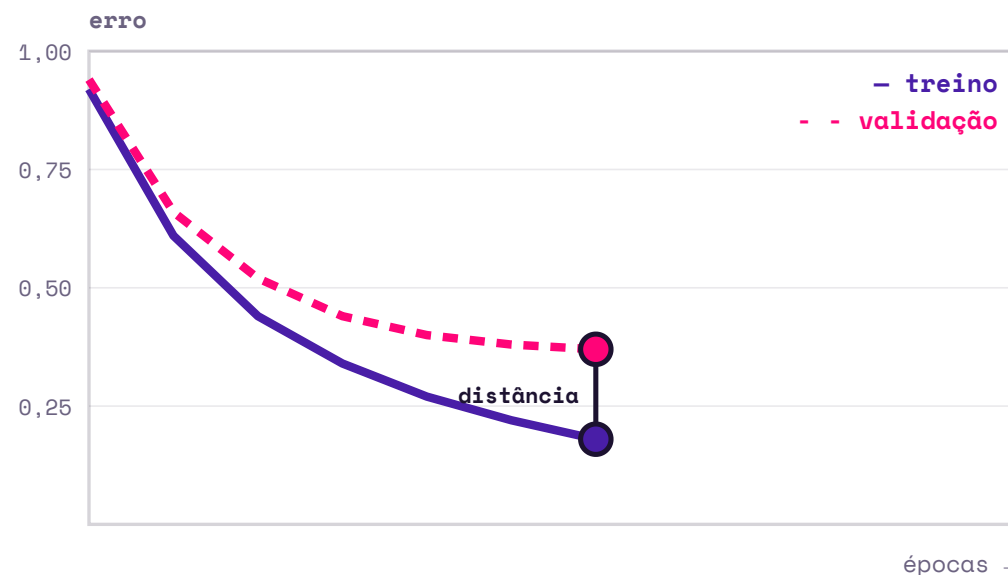
0,18

PERDA DE VALIDAÇÃO

0,37

A DISTÂNCIA ENTRE ELAS

0,19



TREINO

Erro nos exemplos que o modelo **usa para aprender**. Quase sempre cai: ele pode decorar.

VALIDAÇÃO

Erro em exemplos que ele **nunca usou**. É esta que importa.

COMEÇOU A DECORAR

Treino caindo e validação **subindo**. Da época 6 em diante o modelo está decorando os exemplos de treino. Sem parada antecipada, vocês entregariam este modelo.

A perda de validação chegou ao mínimo. Este é o modelo que o ponto de verificação guarda, e é este que deve ser entregue. Daqui em diante, treinar mais piora o resultado em dados novos.

Perda de treino e perda de validação

Na Fase 1 vocês tinham um número no fim. Agora vocês têm um filme.

época 12 de 12

PERDA DE TREINO

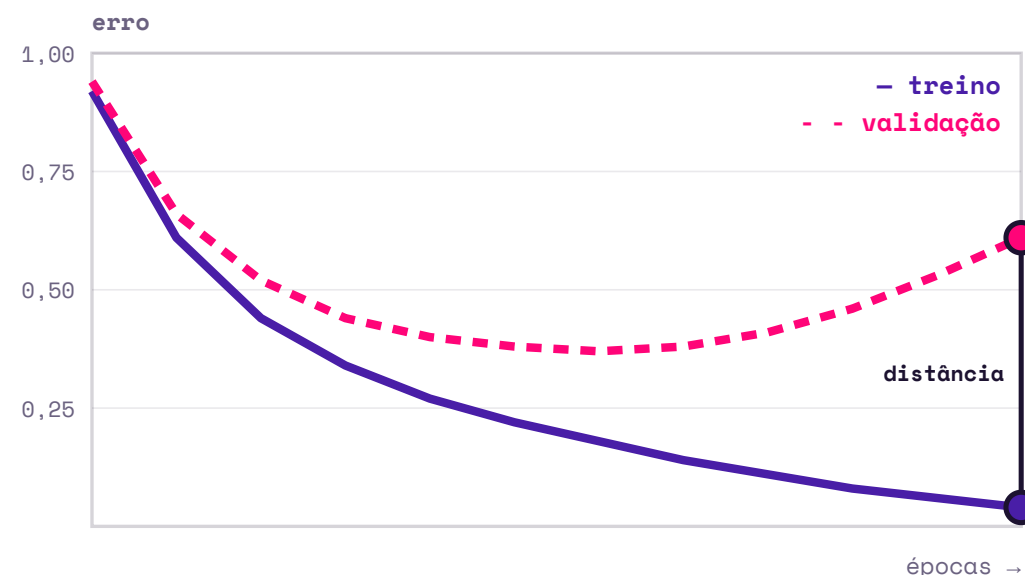
0,04

PERDA DE VALIDAÇÃO

0,61

A DISTÂNCIA ENTRE ELAS

0,57



TREINO

Erro nos exemplos que o modelo **usa para aprender**. Quase sempre cai: ele pode decorar.

VALIDAÇÃO

Erro em exemplos que ele **nunca usou**. É esta que importa.

COMEÇOU A DECORAR

Treino caindo e validação **subindo**. Da época 6 em diante o modelo está decorando os exemplos de treino. Sem parada antecipada, vocês entregariam este modelo.

Treino continuando além do mínimo. A curva de treino segue descendo — a rede está decorando cada vez melhor — e a de validação sobe. A distância entre as duas é a medida de quanto foi decoreba. Compare o número da distância com o da página anterior.

Quatro treinos, quatro diagnósticos

Linha cheia é o treino, pontilhada é a validação.

Treino saudável

Sobreajuste

Subajuste

Taxa alta demais



TREINO SAUDÁVEL

As duas descem e ficam juntas, com a distância estabilizando. É o que vocês querem ver.

O QUE FAZER

Nada. Guardem a melhor época e usem o tempo que sobrou para interpretar o resultado.

Primeiro dos quatro formatos. Aprender a reconhecer o formato é o que transforma o gráfico do caderno numa instrução sobre qual botão mexer — e em qual direção.

Quatro treinos, quatro diagnósticos

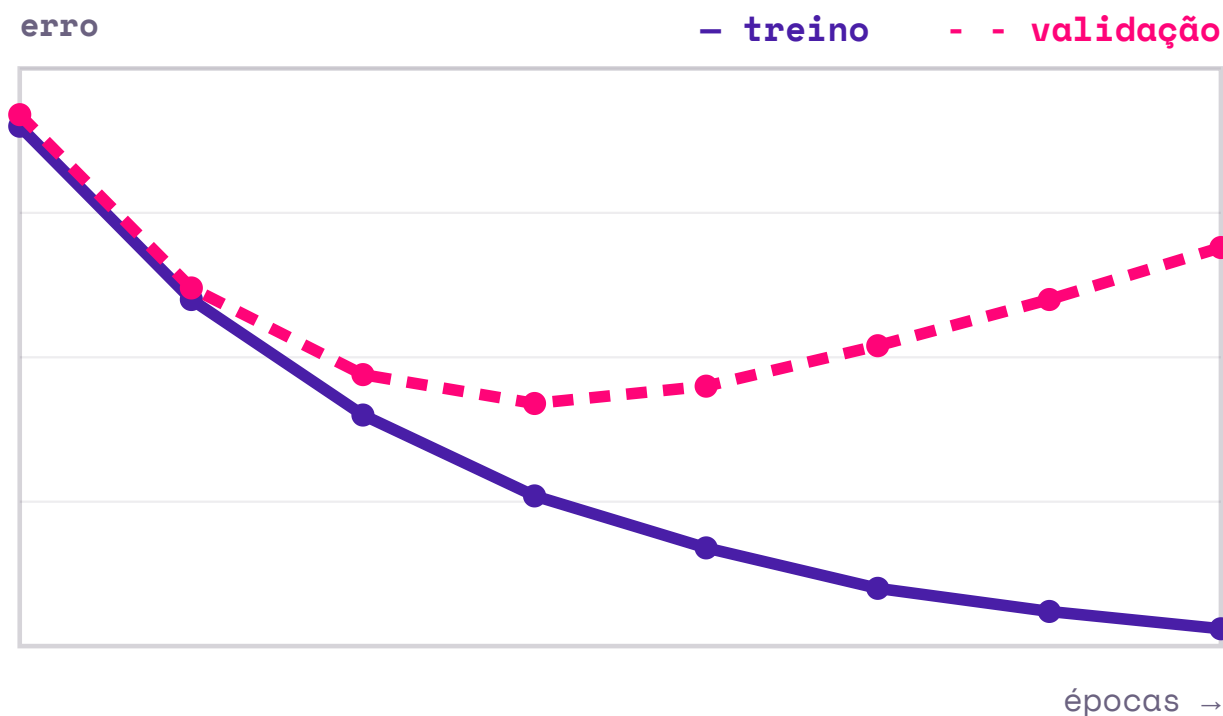
Linha cheia é o treino, pontilhada é a validação.

Treino saudável

Sobreajuste

Subajuste

Taxa alta demais



SOBREAJUSTE

A do treino despenca e a da validação sobe. Clássico: o modelo decorou os exemplos.

O QUE FAZER

Parada antecipada, mais dropout, rede menor — ou mais dados.

Segundo formato. Repare que a recomendação muda completamente em relação ao anterior, mesmo o gráfico sendo do mesmo par de curvas: é o formato que decide.

Quatro treinos, quatro diagnósticos

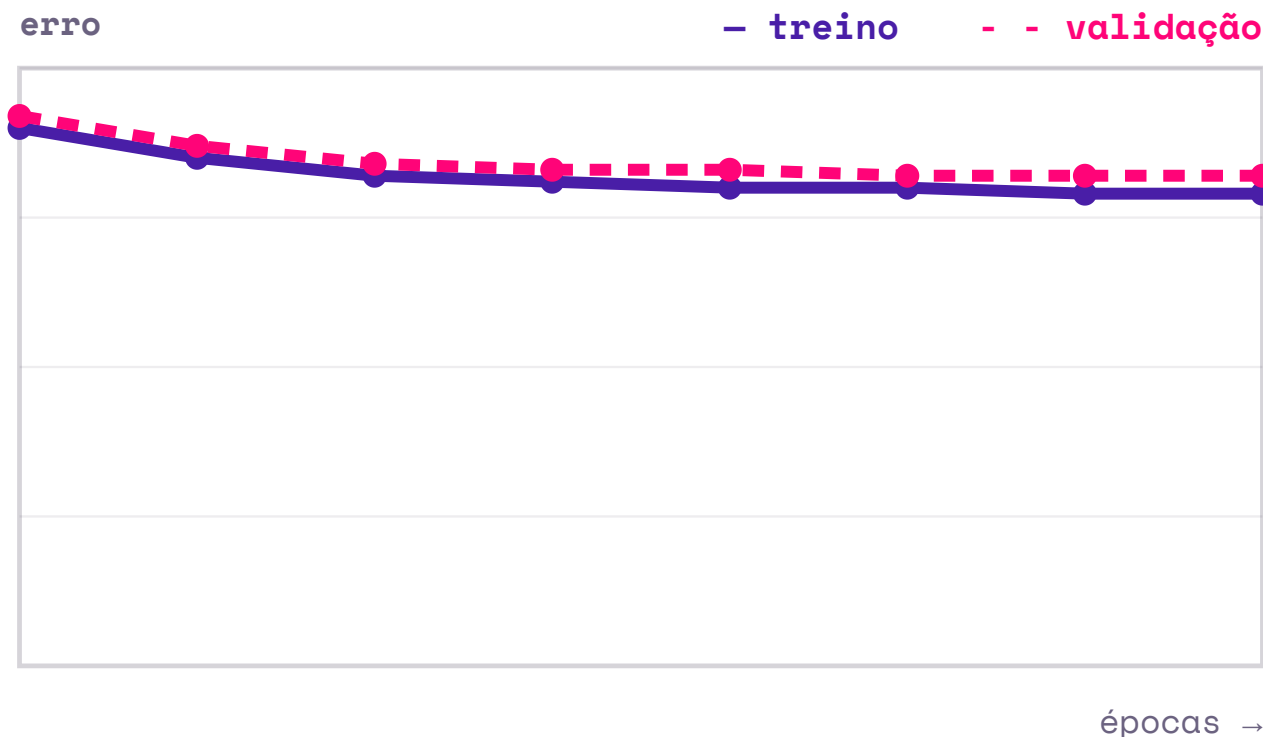
Linha cheia é o treino, pontilhada é a validação.

Treino saudável

Sobreajuste

Subajuste

Taxa alta demais



SUBAJUSTE

As duas param alto e juntas. O modelo é simples demais, ou o passo é pequeno demais para chegar longe.

O QUE FAZER

Aumentem a capacidade do modelo, aumentem a taxa, ou deem mais épocas. Confirmam também se a régua foi aplicada.

Terceiro formato. Este é o que mais se confunde com os outros, e vale olhar com calma qual detalhe o distingue.

Quatro treinos, quatro diagnósticos

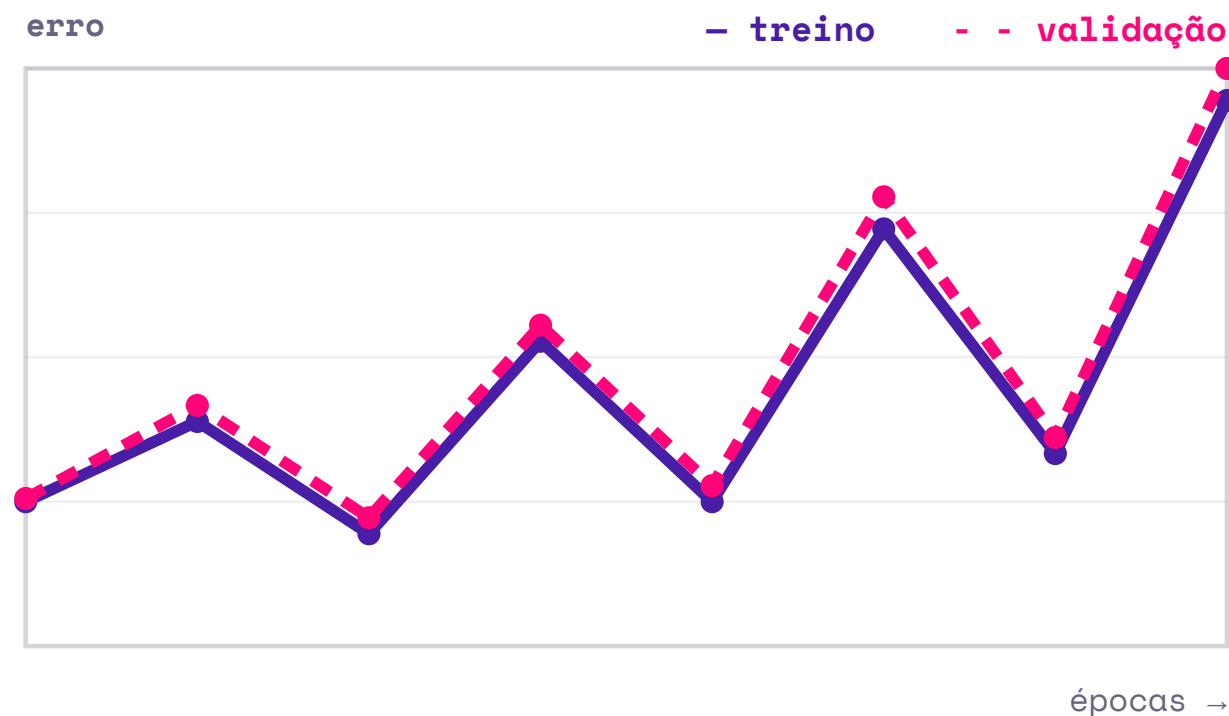
Linha cheia é o treino, pontilhada é a validação.

Treino saudável

Sobreajuste

Subajuste

Taxa alta demais



TAXA ALTA DEMAIS

Serrilhada e subindo. A taxa de aprendizado está grande demais e o treino está explodindo.

O QUE FAZER

Dividam a taxa por dez e rodem de novo. É sempre a primeira coisa a tentar.

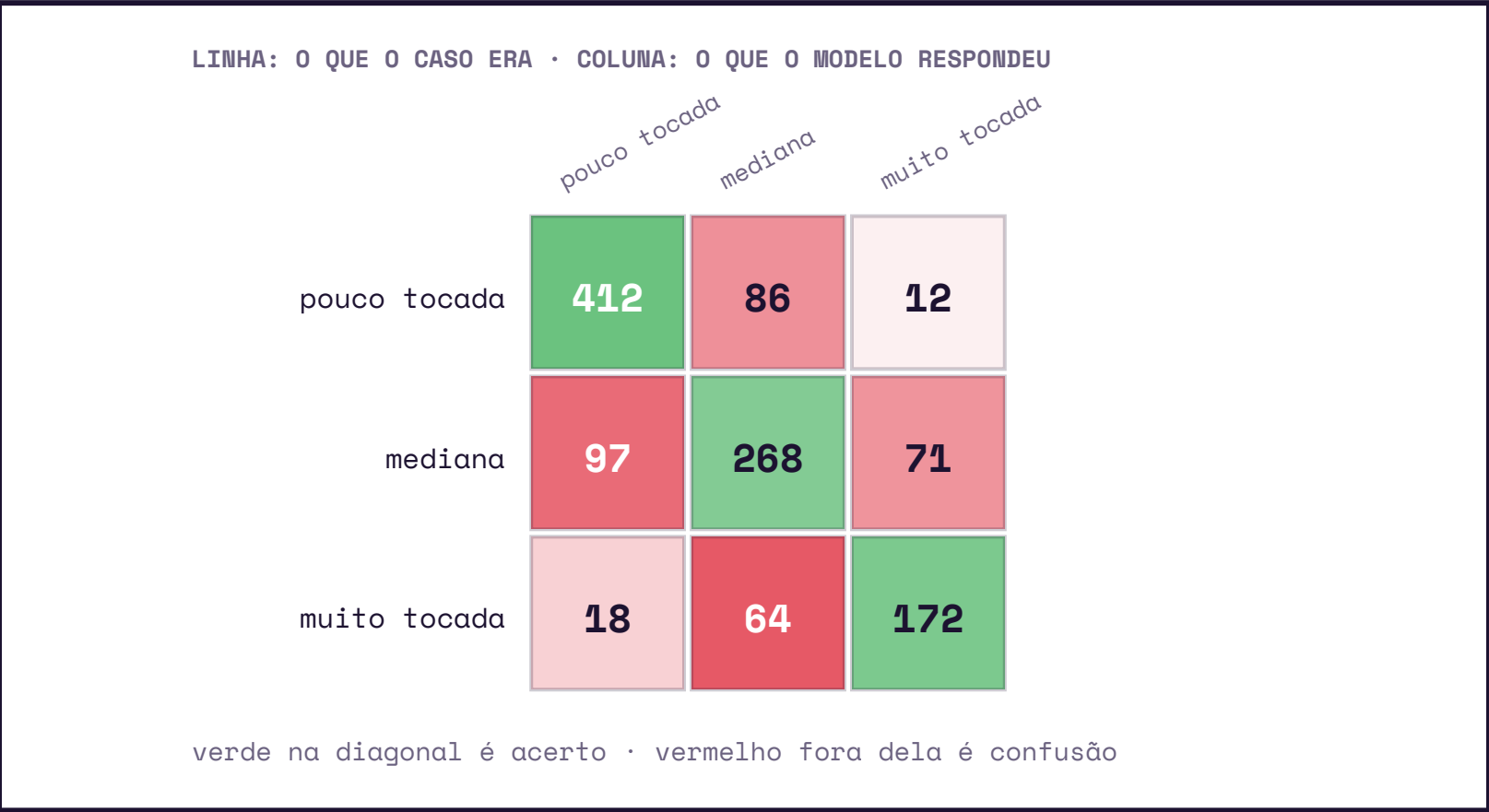
Quarto e último formato. Os quatro juntos são o vocabulário que o Bloco 4 do caderno usa para explicar o que aconteceu no treino de vocês. Decorar os nomes não serve de nada; reconhecer os formatos serve para tudo.

Onde o erro mora

A métrica diz quanto o modelo errou. A matriz diz com o que ele confundiu o quê.

contagem

proporção da linha



COMO LER ESTA TABELA

Cliquem numa casa vermelha para ver o que aquela confusão significa. E leiam sempre em **proporção**: 40 erros numa classe grande podem ser menos graves que 12 numa classe pequena.

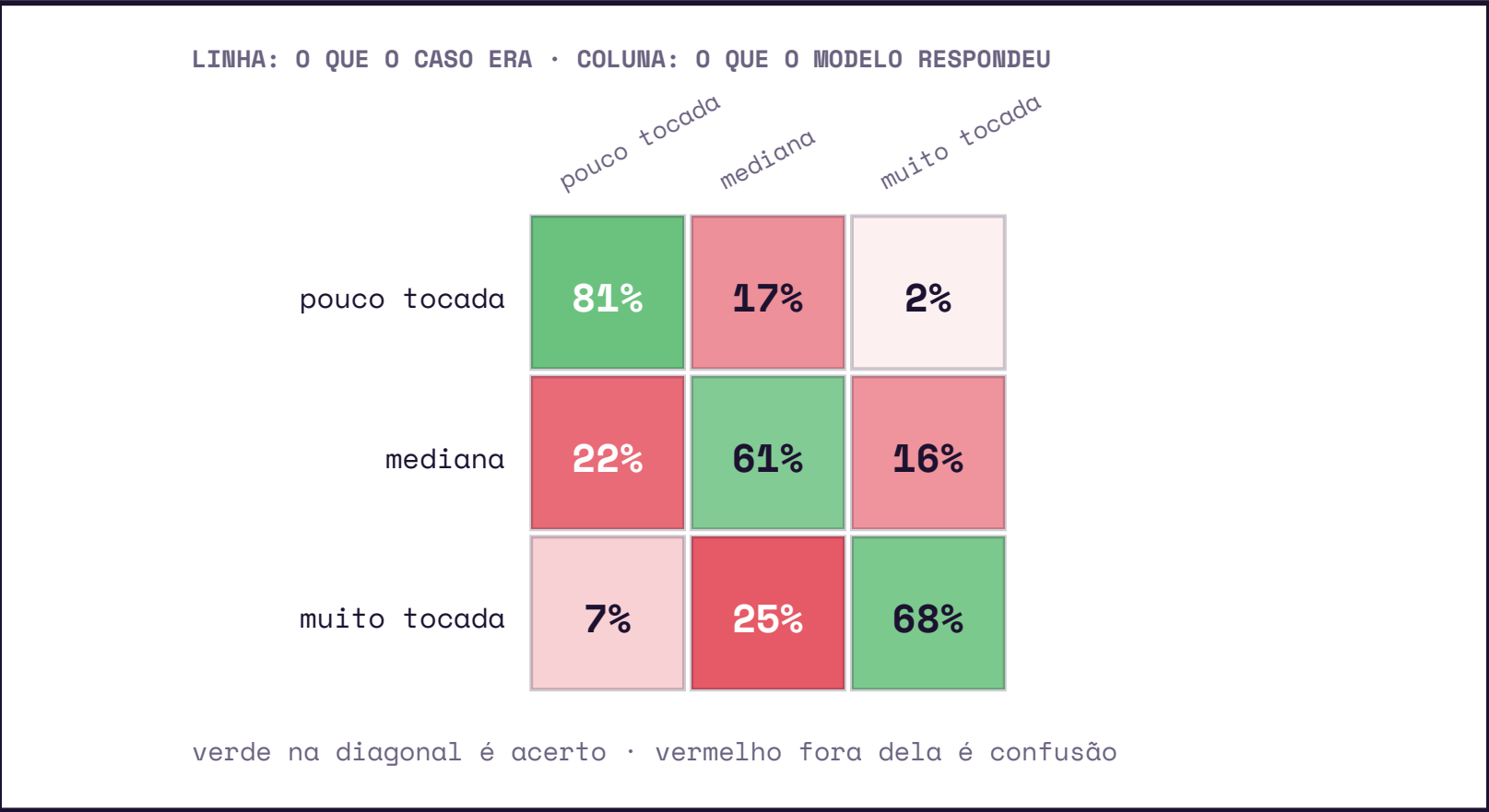
A matriz de confusão em números absolutos. A diagonal são os acertos, e o que está fora dela mostra a direção de cada erro. Em contagem, as classes grandes dominam a imagem e as pequenas quase desaparecem.

Onde o erro mora

A métrica diz quanto o modelo errou. A matriz diz com o que ele confundiu o quê.

contagem

proporção da linha



COMO LER ESTA TABELA

Cliquem numa casa vermelha para ver o que aquela confusão significa. E leiam sempre em **proporção**: 40 erros numa classe grande podem ser menos graves que 12 numa classe pequena.

A mesma matriz, agora com cada linha somando 100%. Isso iguala as classes e revela o que a contagem escondia: uma classe pequena pode estar sendo quase toda confundida com outra, e no gráfico anterior isso era um quadradinho claro que ninguém notaria. Nos dois casos vale perguntar se os pares confundidos fazem sentido para um ouvido humano — errar entre vizinhos é diferente de errar entre coisas distantes.

Custo contra ganho

A última pergunta do bloco não é qual acerta mais. É quanto cada ponto a mais custou.

Rede densa

Rede com embeddings

TabNet



REDE DENSA · 0,76 EM 2 MIN

Dois minutos de treino e já passa da marca da Fase 1. É a referência contra a qual as outras duas precisam se justificar.

A primeira das três decisões de custo contra ganho. Nem todo ganho de F1 vale o que custa, e essa conta não é técnica: depende do tempo que a equipe tem e do que mais poderia fazer com ele.

Custo contra ganho

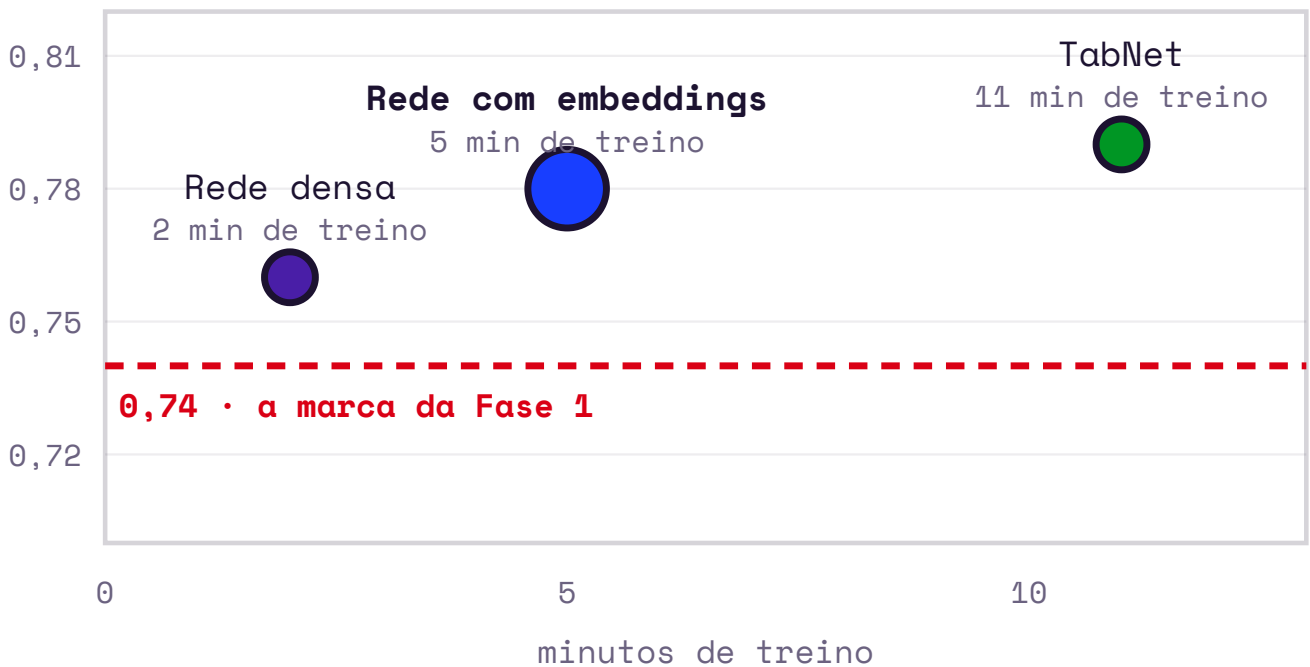
A última pergunta do bloco não é qual acerta mais. É quanto cada ponto a mais custou.

Rede densa

Rede com embeddings

TabNet

F1-macro



REDE COM EMBEDDINGS · 0,78 EM 5 MIN

Dois centésimos a mais por duas vezes e meia o tempo. Com categorias grandes, costuma valer — e a justificativa é fácil de escrever.

Segunda opção. Compare a relação entre as duas barras com a da página anterior: é essa relação, e não o tamanho do ganho, que decide o que fazer primeiro.

Custo contra ganho

A última pergunta do bloco não é qual acerta mais. É quanto cada ponto a mais custou.

Rede densa

Rede com embeddings

TabNet



TABNET · 0,79 EM 11 MIN

Um centésimo sobre a rede com embeddings, e cinco vezes o tempo da rede densa. Só vale se o relatório de vocês for usar a explicação que ele dá de quais colunas olhou.

Terceira opção. No desafio real essa conta aparece disfarçada de pergunta técnica — "vale usar o TabNet?" — e a resposta honesta quase sempre depende de quanto tempo sobrou, não de qual modelo é melhor no papel.

O QUE LEVAR DAQUI

Cinco frases e um endereço

Descubram o piso antes de treinar qualquer coisa.

Cada modelo quer a entrada de um jeito.

A Zona Livre é onde o resultado deixa de ser mediano.

A curva de validação é a que importa.

Sem ponto de verificação, entra na entrega a última época.

Agora abram o Capítulo 2 do material de imersão e refaçam os laboratórios com calma.

olimpiadadeia.ceia.digital

olimpiada@ceia.ufg.br



PARA LEVAR DAQUI

As cinco frases do aulão e o endereço do material vivo. Abram o material e refaçam os quinze laboratórios com calma, mexendo nos controles sem pressa: o que está neste PDF é um estado escolhido de cada um, e a maior parte do aprendizado está em comparar estados que vocês mesmos escolherem.