Inteligência Artificial 2

Avaliação Crítica, Métricas e Fechamento do Módulo

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

18 de agosto de 2026

Onde Paramos?

Na aula anterior, entramos no mundo da **Visão Computacional**:

- **CNNs**: filtros deslizantes que preservam estrutura espacial.
- **Transfer Learning**: reutilizar cérebros treinados pelo Google/Meta.
- **MobileNetV2**: classificador de imagens construído em minutos.

A Pergunta de Hoje

Seu modelo atinge 99% de acurácia.
Isso **realmente** significa que ele é bom?

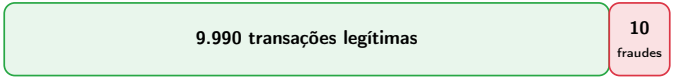
Objetivos da Sessão

- **A ilusão da Acurácia:** Por que ela mente em dados desbalanceados.
- **Matriz de Confusão:** Decompor acertos e erros em 4 quadrantes.
- **Precision:** "Quando a IA grita, ela tem razão?"
- **Recall:** "Das fraudes reais, quantas a IA achou?"
- **F1-Score:** O veredicto final balanceado.
- **Serialização e Deploy:** Salvar modelos para produção.
- **Fechamento do Módulo 1** e apresentação do **TP1**.

A Jornada Completa do Módulo 1

Hoje fechamos o arco: sabemos **construir, treinar, regularizar e inferir**.
Falta o último pilar: **como julgar se o modelo é genuinamente bom?**

Cenário: Detecção de Fraude Bancária

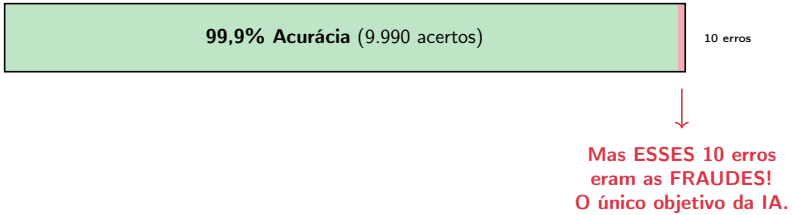


Total: 10.000 transações

- Apenas **0,1%** das transações são fraudes.
- Se um "modelo" **sempre disser "Legítima"**:

Acertou as 9.990 legítimas	✓
Errou **todas** as 10 fraudes	✗
Acurácia	**99,9%**
Fraudes detectadas	**0 de 10**

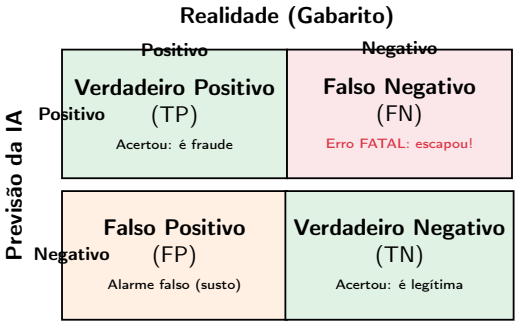
A Acurácia é uma Mentira Estatística



Regra de Ouro

Em dados **desbalanceados**, a Acurácia é **inútil**.
Precisamos decompor *como* o modelo erra.

Matriz de Confusão: Os 4 Quadrantes



- Diagonal principal (verde) = **acertos**.
- Fora da diagonal (vermelho/laranja) = **erros**.

Exemplo Concreto: Exame de Câncer

- **TP:** IA disse "câncer" $\rightarrow$ biópsia confirmou. **Sucesso.**
- **TN:** IA disse "saudável" $\rightarrow$ paciente realmente está bem. **Sucesso.**
- **FP:** IA disse "câncer" $\rightarrow$ mas estava saudável. **Susto, mas seguro.**
- **FN:** IA disse "saudável" $\rightarrow$ mas TEM câncer. **FATAL: sem tratamento!**

Qual erro é pior?

Depende do **domínio e do custo do erro**:

- Medicina: FN é catastrófico (paciente morre).
- Filtro de Spam: FP é inaceitável (e-mail importante perdido).
- Fraude bancária: ambos têm custo alto.

Exemplo Numérico Completo

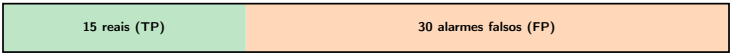
Uma IA de detecção de fraude processou 1.000 transações:

		Real: 1	Real: 0
Prev:	1	TP = 15 Achou a fraude	FN = 5 Escaparam!
Prev:	0	FP = 30 Alarmes falsos	TN = 950 Legítimas OK

- Acurácia = $\frac{15+950}{1000} = 96,5\%$ — Parece ótimo...
- Mas das 20 fraudes reais, **5 escaparam** (FN = 5). — Perigoso!

Precision: "Quando a IA grita, ela tem razão?"

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{15}{15 + 30} = \mathbf{33{,}3\%}$$



De 45 alertas, só 15 eram verdadeiros

- A IA apontou 45 fraudes, mas **30 eram legítimas** (alarmes falsos).
- A equipe de revisão é **sobrecarregada** investigando falsos alertas.
- Precision **baixa** $\rightarrow$ a IA "chora lobo" demais.

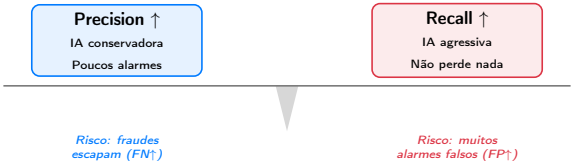
Recall: “Das fraudes reais, quantas a IA achou?”

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{15}{15 + 5} = \mathbf{75\%}$$

De 20 fraudes reais, achou 15

- Das 20 fraudes que **realmente ocorreram**, a IA achou 15.
- 5 fraudadores **escaparam impunes**. Em medicina, 5 pacientes sem tratamento.
- Recall **baixo** $\rightarrow$ a IA falha na sua missão primária.

O Trade-Off: A Gangorra Precision ↔ Recall



- IA **conservadora**: só alerta quando tem certeza → Precision ↑, Recall ↓.
- IA **agressiva**: alerta em tudo que é suspeito → Recall ↑, Precision ↓.
- O engenheiro escolhe o equilíbrio baseado no **custo do erro no domínio**.

Quando Priorizar Cada Métrica?

Domínio	Prioridade	Justificativa
Diagnóstico Médico	**Recall**	Melhor alarme falso que paciente sem tratamento.
Filtro de Spam	**Precision**	E-mail importante na lixeira é inaceitável.
Detecção de Fraude	**F1**	Equilibrar: não perder fraude nem bloquear clientes.
Carro Autônomo	**Recall**	Não identificar pedestre = acidente fatal.

A Armadilha do Limiar de Decisão (Threshold)

- Redes neurais (com sigmoid) não cospem "Fraude"ou "Legítima". Elas cospem uma probabilidade: ex, 0.72 (72% de chance de ser Fraude).
- O padrão é cortar em 0.50 (Threshold).
- Se a Precision está muito baixa (muitos alarmes falsos), podemos subir o Threshold para 0.90. A IA só avisa se tiver **absoluta certeza**.
- Se o Recall está muito baixo (escapando fraudes), podemos baixar o Threshold para 0.20. A IA fica **hiper-vigilante**.

F1-Score: O Veredicto Final

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.333 \times 0.75}{0.333 + 0.75} = \mathbf{0.46}$$

- **Média Harmônica**: só é alta se **ambas** as métricas forem altas.
- A média aritmética de 0.33 e 0.75 seria 0.54 — mascararia o problema.
- A harmônica puxa para baixo: **F1 = 0.46** revela que o modelo é ruim.

Cenário	Prec.	Rec.	F1	
Balanceado	0.85	0.85	**0.85**	✓ Saudável
Medroso	1.00	0.20	**0.33**	× Omisso
Agressivo	0.10	1.00	**0.18**	× Caótico

Classification Report: O Raio-X do Modelo

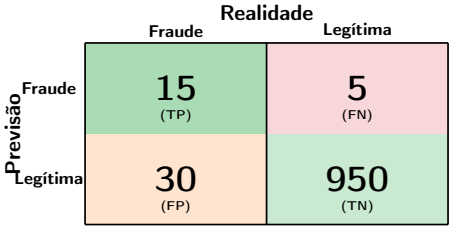
```python
from sklearn.metrics import classification_report
import numpy as np

y_pred = modelo.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype(int)

print(classification_report(y_test, y_pred_classes,
    target_names=['Legítima', 'Fraude']))
```

Saída:

```
              precision    recall  f1-score   support
   Legítima       0.97      0.99      0.98       950
     Fraude       0.33      0.75      0.46        20
   accuracy                           0.96      1000
```

Acurácia = 96%, mas F1 da Fraude = **0.46**. O modelo é um **fracasso** para fraudes.

Visualizando a Matriz de Confusão



✓ Diagonal = acertos

× Fora da diagonal = erros

Lidando com Dados Desbalanceados

Como evitar que o modelo vicie na classe majoritária?

- **Class Weights:** Aumentar a penalidade (loss) ao errar a classe rara. Se o erro na fraude custar 100x mais para o Gradiente, o modelo será forçado a prestar atenção.
- **Oversampling (SMOTE):** Criar clones sintéticos da classe rara.
- **Undersampling:** Jogar fora dados da classe comum até igualar. (Cuidado: joga fora informação valiosa).

Usando Class Weights no Keras

```
# Dicionario de pesos:
# A classe 0 (legitima) tem peso 1
# A classe 1 (fraude) tem peso 50 (50x mais importante)
pesos = {0: 1.0, 1: 50.0}

model.fit(
    X_train, y_train,
    epochs=100,
    class_weight=pesos  # A magia acontece aqui!
)
```

Isso força o *Gradient Descent* a dar saltos enormes quando erra uma fraude, corrigindo o viés.

Salvando o Modelo para Produção

```python
# Salvar o modelo treinado
model.save('meu_modelo.keras')

# Em OUTRO script: carregar e usar
from tensorflow.keras.models import load_model
modelo = load_model('meu_modelo.keras')

# Inferência direta (sem retreinar!)
previsao = modelo.predict(nova_amostra)
```

O Trabalho Prático 1 (TP1)

Missão: Resolver um problema real do Kaggle.

1. **Dados**: CSV desbalanceado do Kaggle.
2. **Limpeza**: Pandas + StandardScaler.
3. **Split**: Treino / Validação / Teste.
4. **Rede**: Dense + Dropout + Adam.
5. **Regularizar**: EarlyStopping + Dropout.
6. **Avaliar**: `classification_report`.
7. **Serializar**: `model.save()`.

Critérios

- Código: 30%
- Avaliação (F1, Matriz): 40%
- Apresentação: 30%

Entrega

Via GitLab com CI/CD.
Pipeline executa testes +
gera `classification_report`.

TP1: O Contexto e a Dificuldade

- O TP1 **não é** um roteiro mastigado como os laboratórios.
- Vocês receberão um dataset sujo, desbalanceado e sem instruções do tipo "crie uma camada de 64 neurônios aqui".
- A proposta é que o nível de dificuldade do TP1 seja **além dos labs**: vocês terão que tomar decisões de arquitetura e hiperparâmetros sozinhos.

TP1: O Ciclo de Desenvolvimento

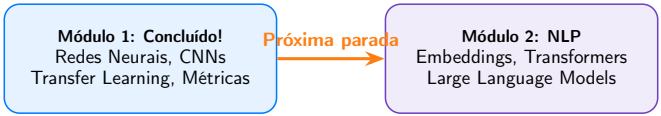
Como atacar o problema:

1. **Análise Exploratória (EDA):** Descubram a proporção de fraudes. Entendam as colunas.
2. **Baseline Simples:** Treine uma rede minúscula sem regularização. Ela provavelmente vai overfitar ou ter F1-Score baixo.
3. **Otimização:** Adicionem Dropout, EarlyStopping e ajustem o `learning_rate`.
4. **Decisão Crítica:** Analisem a Matriz de Confusão para garantir que as Fraudes (FN) estão sendo minimizadas!

Recapitulação — O que Levar Desta Aula

1. A **Acurácia mente** em dados desbalanceados — um modelo que chuta a classe majoritária pode ter 99%.
2. A **Matriz de Confusão** decompõe erros em TP, TN, FP e FN.
3. **Precision** $= \frac{TP}{TP+FP}$ mede alarmes falsos. **Recall** $= \frac{TP}{TP+FN}$ mede omissões.
4. **F1-Score** (média harmônica) = veredicto definitivo para dados desbalanceados.
5. O `classification_report` do Scikit-Learn é o **raio-X** obrigatório de todo modelo.
6. **Serializar** com `.save()` permite deploy sem retreinamento.

Olhando para o Futuro: Módulo 2



A Grande Transição

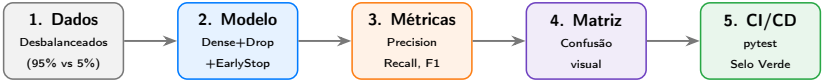
Sai: tabelas numéricas e pixels.

Entra: **palavras, frases, linguagem humana**.

Como transformar "gato" em um vetor que o computador entende?

Roteiro do Laboratório 08

Missão: Avaliar um modelo com métricas reais em dados desbalanceados.



Lab Passo 1: O Dataset Desbalanceado

Vocês receberão um conjunto de dados onde 95% das amostras pertencem à Classe 0 e apenas 5% à Classe 1.

- A primeira tarefa é treinar um modelo "ingênuo".
- Vocês verão que o modelo atingirá 95% de acurácia **apenas chutando a classe 0** para tudo.
- É a prova visual de que a acurácia é uma métrica traiçoeira.

Lab Passo 2: Treinando com Vigilância

Vamos adicionar o pacote completo de treinamento para tentar salvar a rede:

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Lembre-se: fit retorna um "history"
historico = model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=100,
    callbacks=[EarlyStopping(patience=10)]
)
```

Lab Passo 3: O Classification Report

O momento da verdade! Em vez de usar `model.evaluate()`, faremos:

```
from sklearn.metrics import classification_report

y_pred_probs = model.predict(X_test)
# Transforma probabilidades (0.7) em classes (1)
y_pred_classes = (y_pred_probs > 0.5).astype(int)

print(classification_report(y_test, y_pred_classes))
```

Analise o F1-Score da **Classe 1**. Ele é o verdadeiro indicador de sucesso.

Lab Passo 4: Matriz de Confusão Visual

Vamos desenhar os 4 quadrantes para ver os erros reais:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

matriz = confusion_matrix(y_test, y_pred_classes)
sns.heatmap(matriz, annot=True, fmt='d', cmap='Blues')
```

Onde está o maior número? Nos Falsos Negativos ou Falsos Positivos?

Lab Passo 5: CI/CD Pipeline e Despedida

O último teste do módulo:

- O pytest não vai checar sua acurácia. Ele vai checar o seu **F1-Score** na classe minoritária.
- Atingiu a meta? `git push origin master`.
- Se o **Selo Verde** brilhar, você concluiu com sucesso todo o Módulo 1!

Dúvidas?

Email: alessio@cefetmg.br
Aula: Segunda e Quarta-Feira, 7h as 8:40h (Sala 39)
WhatsApp: Dúvidas rápidas

Lembrete

Confira sua Issue no GitLab!

Próxima aula: Espaço Latente & NLP

100% da Aula 8 Concluída!