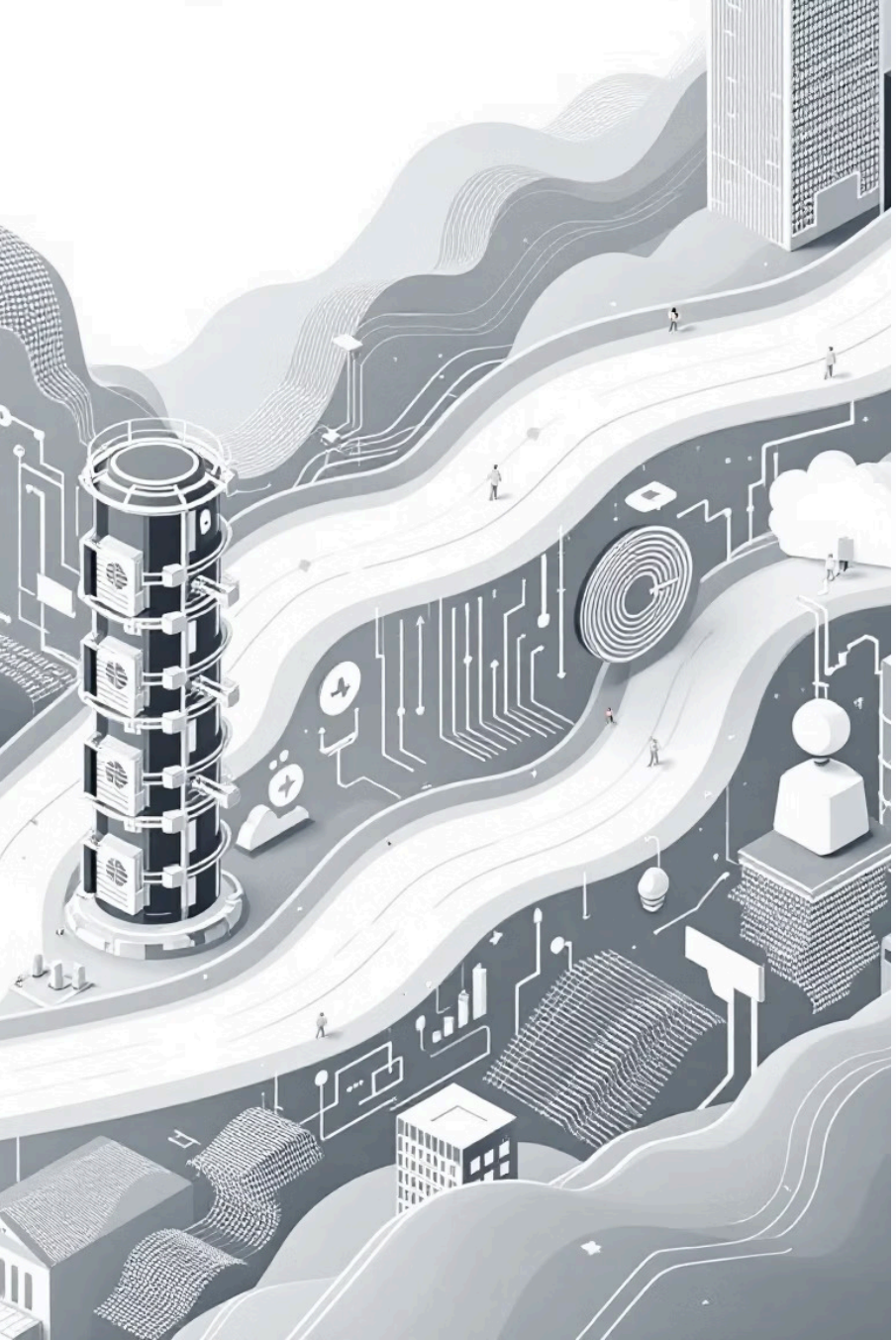
Previsão de Resultados Acadêmicos com Decision Trees

Pipeline completo de Machine Learning para prever se estudantes irão graduar, abandonar ou permanecer matriculados.

Aluno: João Victor Póvoa França



Objetivo do Projeto

Meta Principal

Prever AcademicOutcome: Graduate, Dropout ou Enrolled usando Decision Tree otimizada.

Entregas

Limpeza de dados, pipeline robusto, modelo otimizado e métricas completas (recall, ROC AUC, matriz de confusão).

Abordagem

Workflow estruturado com validação cruzada e busca de hiperparâmetros para máxima performance.

Imports Essenciais

Bibliotecas Core

```
import pandas as pd
import numpy as np
from sklearn.model_selection import (
    train_test_split,
    GridSearchCV
)
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
```

Pré-processamento e Modelo

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import (
    StandardScaler,
    OrdinalEncoder
)
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    roc_auc_score
)
```

Por quê: Colecionamos apenas os módulos necessários para o fluxo completo — pré-processamento, modelagem, métricas e visualização.

Carregamento Robusto do Dataset

```
try:
    from scipy.io import arff
    data, meta = arff.loadarff('/mnt/data/dataset_')
    df = pd.DataFrame(data)
    # Converter bytes -> str quando necessário
    for c in df.select_dtypes([object]).columns:
        df[c] = df[c].apply(
            lambda v: v.decode('utf-8')
            if isinstance(v, bytes) else v
        )
except Exception:
    # Fallback: parsear manualmente após @DATA
    import csv, io, re
    text = open('/mnt/data/dataset_',
                encoding='utf-8').read()
    attrs = re.findall(r"@ATTRIBUTE\s+([^\s]+)",
                       text, flags=re.IGNORECASE)
    data_part = text.split('@DATA',1)[1].strip()
    rows = list(csv.reader(
        io.StringIO(data_part),
        delimiter=',', quotechar="'"
    ))
    df = pd.DataFrame(rows, columns=attrs)
```

Estratégia: Garantir carregamento mesmo com formato ARFF não-padrão, usando fallback manual se necessário.

Análise Exploratória Rápida

3

Classes Alvo

Graduate, Dropout e Enrolled — distribuição verificada para balanceamento.

0

Top Nulos

Identificar colunas com mais valores ausentes para estratégia de imputação.

```
print("Dimensão:", df.shape)
print("Alvo (AcademicOutcome) - contagem:")
print(df['AcademicOutcome'].value_counts())
print("\nNulos por coluna (top 10):")
print(df.isna().sum().sort_values(
    ascending=False
).head(10))
```

Por quê: Estes números orientam decisões de limpeza, balanceamento e justificam escolhas metodológicas na apresentação.

Dataset Limpo...

Limpeza e Preparação do Target

Remover Duplicatas

Eliminar registros duplicados que distorcem validação.

Filtrar Target

Manter apenas instâncias com AcademicOutcome válido.

Converter Tipo

Garantir target como string para classificação.

```
df = df.drop_duplicates().reset_index(drop=True)
df = df[df['AcademicOutcome'].notna()].copy()
df['AcademicOutcome'] = df['AcademicOutcome'].astype(str)
```

Importante: Manter imputação dentro do pipeline evita vazamento de dados entre treino e teste.

Identificação de Tipos de Features

Features Numéricas

```
num_cols = df.select_dtypes(
    include=['int64','float64']
).columns.tolist()

print("Numéricas:",
      num_cols[:8],
      "... total:",
      len(num_cols))
```

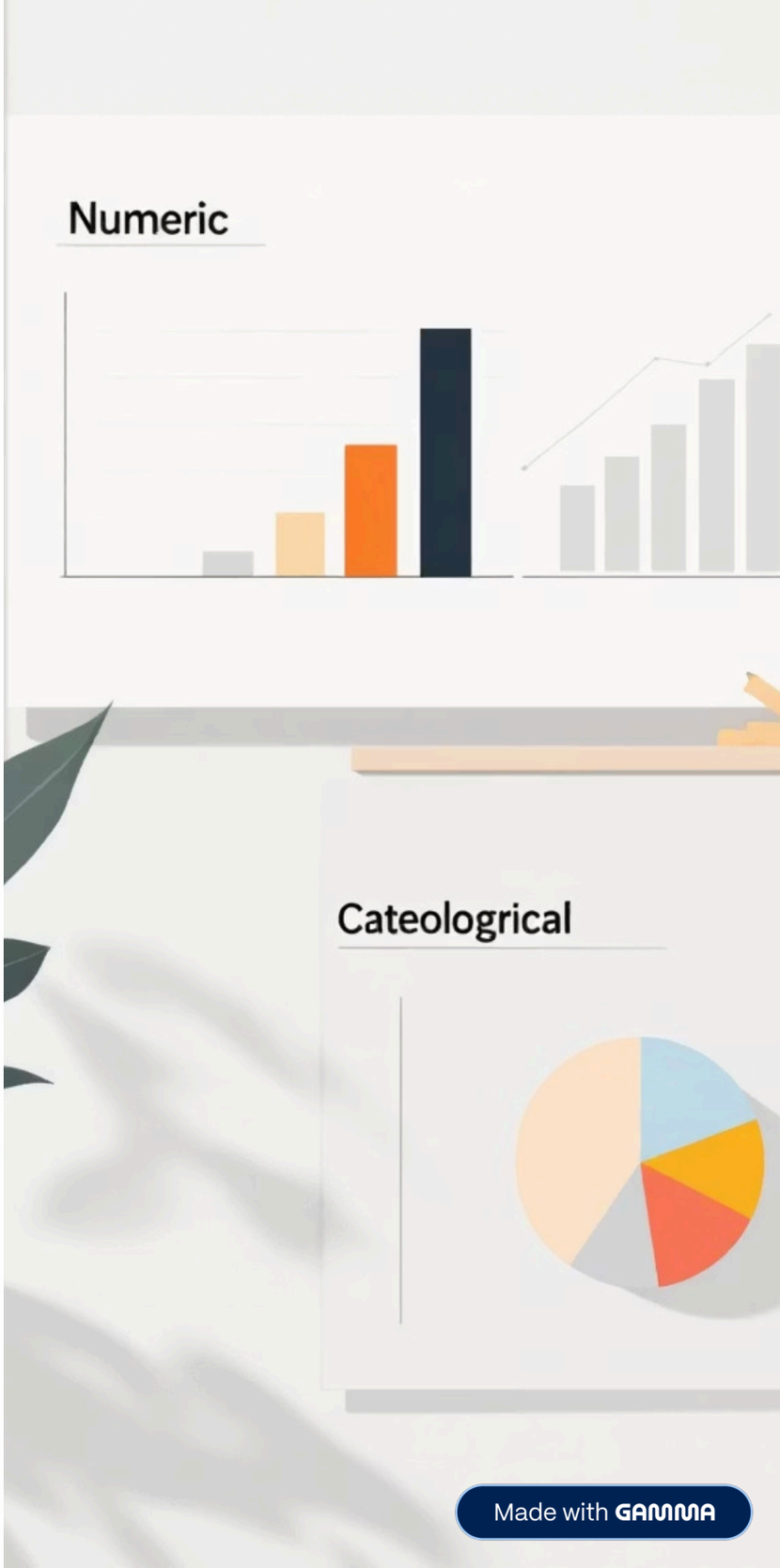
Idade, notas, créditos cursados, indicadores financeiros.

Features Categóricas

```
cat_cols = df.select_dtypes(
    include=['object']
).columns.tolist()

if 'AcademicOutcome' in
cat_cols:

cat_cols.remove('AcademicOu
tcome')

print("Categóricas:",
      cat_cols[:8],
      "... total:",
      len(cat_cols))
```

Curso, gênero, tipo de escola, status de bolsa.

Por quê: Define como cada coluna será tratada no ColumnTransformer e justifica escolha entre codificação ordinal vs one-hot.



Pipeline de Pré-processamento

01

Numéricas: Imputação + Escala

Mediana (robusta a outliers) + StandardScaler para normalização.

02

Categóricas: Imputação + Codificação

Valor constante 'missing' + OrdinalEncoder com tratamento de unknowns.

03

ColumnTransformer

Unifica pipelines numérico e categórico, descartando colunas não especificadas.

```
numeric_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='constant',
                              fill_value='missing')),
    ('ord', OrdinalEncoder(handle_unknown='use_encoded_value',
                           unknown_value=-1))
])

preprocessor = ColumnTransformer([
    ('num', numeric_pipeline, num_cols),
    ('cat', categorical_pipeline, cat_cols)
], remainder='drop')
```

Decisão estratégica: OrdinalEncoder evita explosão dimensional do one-hot. Decision Trees lidam perfeitamente com códigos inteiros.

Esse bloco monta um **pré-processamento automático** que:

1. transforma colunas numéricas (imputação + escala),
2. transforma colunas categóricas em números (imputação + `OrdinalEncoder`),
3. junta tudo num único array que segue para o classificador.

A vantagem: tudo é feito **dentro do pipeline** de modo que o ajuste (fit) usa **apenas os dados de treino** — evita vazamento de informação e facilita validação cruzada.

Split Estratificado e Pipeline do Modelo

Divisão dos Dados

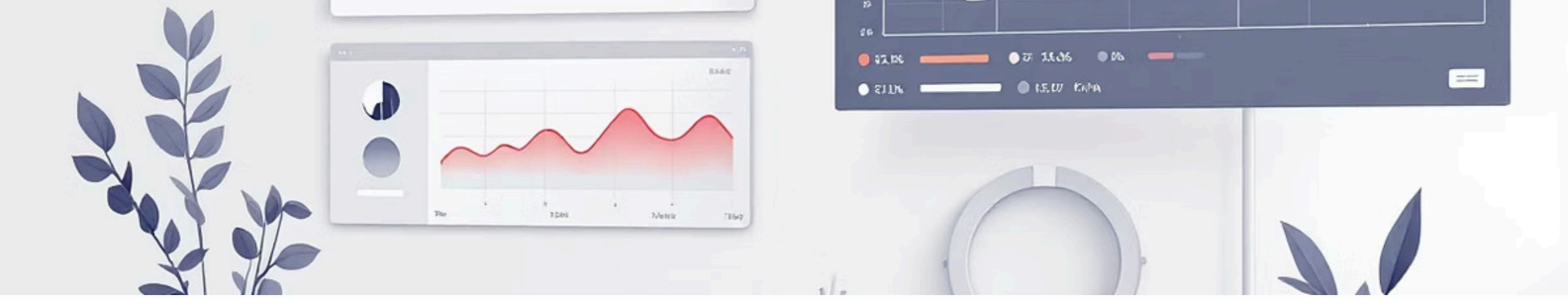
```
RANDOM_STATE = 42

X = df.drop(columns=['AcademicOutcome'])
y = df['AcademicOutcome']

X_train, X_test, y_train, y_test = \
    train_test_split(
        X, y,
        test_size=0.25,
        random_state=RANDOM_STATE,
        stratify=y
    )
```

Stratify: Mantém proporção das classes em treino e teste. Sendo 25% para 75%

Pipeline Completo

```
pipe = Pipeline([
    ('pre', preprocessor),
    ('clf', DecisionTreeClassifier(
        random_state=RANDOM_STATE
    ))
])
```

Vantagem: Pipeline unifica transformações e classificador, facilitando validação cruzada e evitando vazamento de dados.



Otimização de Hiperparâmetros

Grid de Busca

- max_depth: [5, 8]
- min_samples_leaf: [1, 5, 10]

Validação Cruzada

CV=3 para resposta rápida mantendo robustez estatística.

Métrica de Seleção

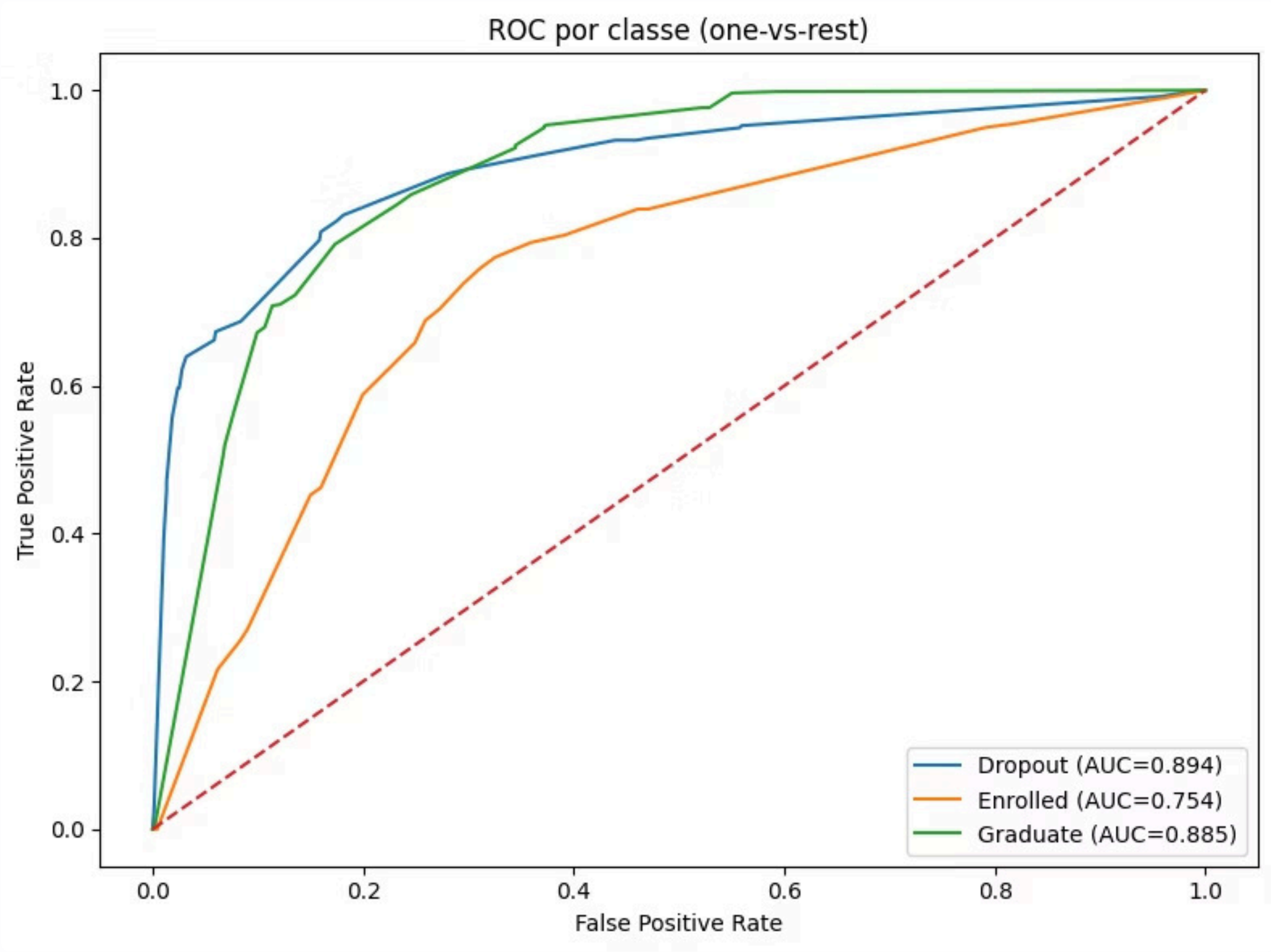
ROC AUC OVR para separabilidade global entre as três classes.

```
param_grid = {
    'clf__max_depth': [5, 8],
    'clf__min_samples_leaf': [1, 5, 10]
}

gs = GridSearchCV(
    pipe, param_grid,
    cv=3,
    scoring='roc_auc_ovr',
    n_jobs=1,
    verbose=1
)

gs.fit(X_train, y_train)

print('Melhor CV score:', gs.best_score_)
print('Melhor params:', gs.best_params_)
```

Explicação: Se `best_params_` escolher `max_depth=5` e `min_samples_leaf=10`, isso indica que **uma árvore mais simples generaliza melhor** no seu dataset.

- `best_score_` alto → bom poder de separação na validação; mas **compare com teste**:

Identificação das métricas de avaliação



Explicação

- Cada curva é a **ROC** (one-vs-rest) para uma classe: para cada threshold calcula-se **TPR** (sensibilidade / recall) vs **FPR** (falsos positivos).
- A linha tracejada diagonal representa um classificador aleatório (AUC = 0.5).
- **AUC** (área sob a curva) resume a separabilidade: 1.0 = perfeito, 0.5 = aleatório.

Valores observados

- **Dropout — AUC = 0.894** → ótima separabilidade; o modelo distingue bem Dropout vs outros.
- **Graduate — AUC = 0.885** → também muito boa separabilidade.
- **Enrolled — AUC = 0.754** → separabilidade moderada/baixa comparada às outras duas — é a classe mais difícil.

Métricas de Avaliação Detalhadas

1

Matriz de Confusão

- O que mostra: contagens reais vs preditas; linha = reais, coluna = predito
- Como ler: célula diagonal = acertos; células fora da diagonal = tipos específicos de erro
- Por que é importante: identifica qual classe é confundida com qual outra

2

Acurácia (Accuracy)

- Definição: proporção de previsões corretas = (TP total) / (total)
- Interpretação: ~72% das amostras de teste foram classificadas corretamente
- Limitação: pode esconder problemas em classes minoritárias

3

Precisão (Precision)

- Definição: entre as instâncias que o modelo rotulou como X, quantas realmente eram X?
- Enrolled precision ≈ 0.406 → poucas previsões "Enrolled" são realmente Enrolled
- Dropout precision ≈ 0.842, Graduate ≈ 0.777 → previsões confiáveis

4

Revocação/Sensibilidade (Recall)

- Definição: entre todas as instâncias reais de X, quantas o modelo conseguiu identificar?
- Enrolled recall ≈ 0.432 → detecta < 50% dos Enrolled reais
- Graduate recall ≈ 0.859 → maioria dos Graduate é corretamente capturada
- Dropout recall ≈ 0.673 → identificação moderada

5

F1-Score

- Definição: média harmônica entre precision e recall
- Enrolled F1 ≈ 0.419 (baixo) → classificador pouco equilibrado
- Graduate F1 ≈ 0.816, Dropout F1 ≈ 0.748 (bons/razoáveis)

Resumo por Classe