

# Lista 4 - IA

Otávio Augusto de Assis Ferreira Monteiro - 851568

Link para a atividade com os códigos no github:

<https://github.com/otavioaugustoafm/Faculdade/tree/main/IA/Lista%204>

## Questão 1

### Principal

```
from __future__ import annotations
from typing import Tuple
import numpy as np
import pandas as pd

def train_test_split_stratified(y: np.ndarray, test_size: float = 0.2, seed: int = 42) -> Tuple:
    rng = np.random.default_rng(seed)
    idx = np.arange(len(y))
    test_idx = []
    for c in np.unique(y):
        class_idx = idx[y == c]
        rng.shuffle(class_idx)
        n_test = max(1, int(round(test_size * len(class_idx))))
        test_idx.extend(class_idx[:n_test])
    test_idx = np.array(sorted(test_idx))
    train_idx = np.array([i for i in idx if i not in set(test_idx)])
    return train_idx, test_idx

def discretize_equal_frequency(series: pd.Series, bins: int = 4, labels: bool = True) -> pd.Series:
    """Discretiza por quantis (-mesmo número de amostras por faixa)."""
    q = np.linspace(0, 1, bins + 1)
    edges = np.unique(series.quantile(q).values)
    edges[0] = -np.inf
    edges[-1] = np.inf
    cats = pd.cut(series, bins=edges, include_lowest=True)
    return cats.astype(str) if labels else cats

def discretize_equal_width(series: pd.Series, bins: int = 4, labels: bool = True) -> pd.Series:
    """Discretiza por largura fixa (intervalos iguais)."""
    cats = pd.cut(series, bins=bins, include_lowest=True)
    return cats.astype(str) if labels else cats

def impute_simple(df: pd.DataFrame) -> pd.DataFrame:
    """Imputa NaNs: numéricos -> mediana; categóricos/objeto -> moda."""
    out = df.copy()
    for col in out.columns:
        if out[col].dtype == object:
            if out[col].isna().any():
                out[col] = out[col].fillna(out[col].mode().iloc[0])
        else:
            if out[col].isna().any():
                out[col] = out[col].fillna(out[col].median())
    return out
```

## Metricas:

```
from __future__ import annotations
import numpy as np

def accuracy(y_true, y_pred) -> float:
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)
    return float((y_true == y_pred).mean())

def confusion_matrix(y_true, y_pred):
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)
    labels = sorted(list(set(y_true) | set(y_pred)))
    L = len(labels)
    lab2i = {lab: i for i, lab in enumerate(labels)}
    m = np.zeros((L, L), dtype=int)
    for t, p in zip(y_true, y_pred):
        m[lab2i[t], lab2i[p]] += 1
    return labels, m
```

## Questão 2

.1)

<https://github.com/otavioaugustoafm/Faculdade/blob/main/IA/Lista%204/Codigos/ID3.py>

.2)<http://github.com/otavioaugustoafm/Faculdade/blob/main/IA/Lista%204/Codigos/C45.py>

.3)<https://github.com/otavioaugustoafm/Faculdade/blob/main/IA/Lista%204/Codigos/CART.py>

## Questao 3

Considerando um split de 80/20 e max depth de 6

.3) ID3

Acuracia do treino: 0.8808

Acurácia do teste: 0.8034

Matriz de confusão do teste:

labels:

0, 1

||0|1|

|-|-|-|

|0|99|11|

|1|24|44|

## Árvore:

```
[Sex]
-> female:
[Pclass]
-> 1:
[Fare]
-> (14.454, 30.5]:
[Embarked]
-> C:
[Age]
-> (35.0, inf]:
[Parch]
-> 0:
  Folha: 0
-> S:
  Folha: 1
-> (30.5, inf]:
  Folha: 1
-> 2:
[Age]
-> (-inf, 22.0]:
  Folha: 1
-> (22.0, 28.0]:
[Parch]
-> 0:
  [Embarked]
  -> C:
    Folha: 1
  -> S:
    [Fare]
    -> (14.454, 30.5]:
      Folha: 1
    -> (30.5, inf]:
      Folha: 1
    -> (7.896, 14.454]:
      Folha: 1
-> 1:
  [SibSp]
  -> 0:
    Folha: 1
  -> 1:
    [Embarked]
    -> S:
      Folha: 0
  -> 2:
    Folha: 1
-> 2:
  Folha: 1
-> 3:
  Folha: 1
-> (28.0, 35.0]:
  Folha: 1
-> (35.0, inf]:
[Parch]
-> 0:
[Fare]
-> (14.454, 30.5]:
[Embarked]
-> S.
```

3.2) C45 Acurácia do treino: 0.8219

Acurácia do teste: 0.8202

Matriz de confusão do teste

labels: 0, 1

||0|1|

|-|-|-|

|0|103|7|

|1|25|43|

## Arvore:

```
[Sex]
-> female:
  [SibSp < 6]
  -> < :
    [Pclass < 2.5]
    -> < :
      [Fare < 28.8562]
      -> < :
        [Age < 53.5]
        -> < :
          [Parch < 1.5]
          -> < :
            Folha: 1
          -> >=:
            Folha: 1
        -> >=:
          [Embarked]
          -> S:
            Folha: 1
        -> >=:
          Folha: 1
      -> >=:
        [Fare < 32.8813]
        -> < :
          [Age < 1.5]
          -> < :
            Folha: 1
          -> >=:
            [Embarked]
            -> C:
              Folha: 1
            -> Q:
              Folha: 1
            -> S:
              Folha: 0
          -> >=:
            Folha: 0
        -> >=:
          Folha: 0
    -> >=:
      Folha: 0
-> male:
  [Age < 1.5]
  -> < :
    Folha: 1
  -> >=:
    [Fare < 387.665]
    -> < :
      [SibSp < 4.5]
      -> < :
        [Parch < 2.5]
        -> < :
          [Pclass < 1.5]
          -> < :
            Folha: 0
          -> >=:
            Folha: 0
        -> >=:
          Folha: 0
      -> >=:
        Folha: 0
```

3.3) CART Acurácia do treino: 0.8682

Acurácia do teste: 0.8258

Matriz de confusão do teste

: labels:

0, 1

||0|1|

|-|-|-|

|0|100|10|

|1|21|47|

Árvore:

```
[Sex ~ ['female']]
-> ~ :
[Pclass < 2.5]
-> < :
[Fare < 28.8562]
-> < :
[Fare < 28.2312]
-> < :
[Age < 53.5]
-> < :
[SibSp < 0.5]
-> < :
  Folha: 1
-> >=:
  Folha: 1
-> >=:
[Pclass < 1.5]
-> < :
  Folha: 1
-> >=:
  Folha: 0
-> >=:
  Folha: 0
-> >=:
  Folha: 1
-> >=:
[Fare < 20.6625]
-> < :
[Age < 7]
-> < :
  Folha: 1
-> >=:
[Fare < 8.0396]
-> < :
[Age < 29.25]
-> < :
  Folha: 1
-> >=:
  Folha: 0
-> >=:
[Fare < 15.8]
-> < :
  Folha: 0
-> >=:
  Folha: 1
-> >=:
[Parch < 0.5]
-> < :
  Folha: 1
-> >=:
[Age < 5.5]
-> < :
[Age < 3.5]
-> < :
  Folha: 0
-> >=:
```