Lista 2

Inteligência Artificial

Nome: Vitor de Meira Gomes
Matrícula: 800643

Questão 1:

1.1) Uma árvore de decisão é gerada de forma recursiva: em cada passo o algoritmo avalia todos os atributos disponíveis e escolhe aquele que melhor separa as classes, ou seja, o que mais reduz a incerteza/impureza do conjunto (medida por critérios como entropia, ganho de informação ou índice de Gini). Esse atributo é colocado como nó interno, e para cada um de seus valores são criados ramos que subdividem o conjunto de exemplos. O processo se repete até que todos os exemplos em um nó pertençam à mesma classe ou não haja mais atributos para dividir.

O atributo na raiz da árvore é o mais informativo de todos: ele é aquele que, considerando o conjunto inteiro de treino, oferece a maior capacidade de separar os exemplos em subconjuntos mais homogêneos, reduzindo ao máximo a incerteza inicial. Por isso, é a “pergunta” mais relevante do problema e é testada primeiro.

1.2) Com uma árvore de decisão gerada a partir de uma base de dados é possível classificar novos exemplos ou estimar valores numéricos (dependendo se for usada para classificação ou regressão). A própria estrutura da árvore aplica perguntas sobre os atributos até chegar a uma folha, que representa a saída prevista. Além disso, cada caminho da raiz até uma folha pode ser interpretado como uma regra do tipo “se... então...”, o que facilita a explicação das decisões e a documentação do modelo. Isso permite identificar de forma transparente quais atributos tiveram maior impacto no resultado, servindo também como uma forma de seleção de variáveis.

A árvore ainda pode fornecer estimativas de probabilidade para cada classe, apoiar a análise exploratória de dados, revelar padrões ocultos e auxiliar na tomada de decisão em diferentes domínios, como diagnóstico médico, detecção de fraudes, previsão de comportamento de clientes e classificação de risco.

1.3)

Vantagens:

- Flexibilidade: não exigem suposições sobre a distribuição dos dados, funcionando como métodos não paramétricos que lidam bem com diferentes tipos de problemas.
- Seleção automática de atributos: o processo de construção já prioriza os atributos mais informativos, reduzindo a influência de variáveis irrelevantes ou redundantes.
- Interpretabilidade: cada caminho da raiz até uma folha pode ser expresso como uma regra “se... então...”, o que facilita explicar e justificar decisões para humanos.
- Eficiência: o treinamento costuma ser rápido, pois a estratégia gulosa de divisão e conquista cresce de forma quase linear em relação ao número de exemplos.

Desvantagens:

- Valores ausentes: exigem tratamento específico; caso contrário, a qualidade da árvore diminui.
- Atributos contínuos: demandam ordenação e escolha de pontos de corte, o que pode ser custoso em bases grandes.
- Instabilidade: pequenas variações nos dados de treino podem gerar árvores finais bastante diferentes.
- Overfitting: árvores muito profundas tendem a se ajustar ao ruído, tornando as inferências próximas às folhas menos confiáveis do que as próximas à raiz.

1.4) A qualidade de uma árvore de decisão é avaliada a partir do tipo de problema a ser resolvido e do objetivo do modelo. Em tarefas de classificação, utiliza-se principalmente a matriz de confusão, que compara as previsões do modelo com os resultados reais e permite calcular métricas como acurácia, precisão, recall e F1-score. Essas medidas indicam o quanto a árvore é capaz de distinguir corretamente as classes. Já em problemas de regressão, são aplicadas métricas como o erro quadrático médio (MSE) e o erro absoluto médio (MAE), que quantificam a diferença entre os valores previstos e os valores reais. Além disso, é comum recorrer à validação cruzada ou à comparação entre erros de treino e teste para avaliar a capacidade de generalização da árvore e detectar possíveis casos de overfitting. Durante o processo de construção, medidas internas como entropia, ganho de informação, razão de ganho e índice de Gini também são utilizadas para indicar a qualidade de cada divisão gerada.

1.5) As regras de uma árvore de decisão são extraídas a partir dos caminhos que ligam a raiz até cada folha. Cada percurso pode ser interpretado como uma sequência de condições que assume a forma de uma regra “se... então...”, em que os testes feitos nos nós internos representam os atributos avaliados e a folha indica a decisão ou classe final. Dessa forma, todo o modelo pode ser convertido em um conjunto de regras lógicas que explicam de forma clara como as previsões são realizadas, facilitando a interpretação e a justificativa das decisões do algoritmo.

Questão 2 nas próximas páginas

```python
import sys
import pandas as pd
import numpy as np
from typing import Dict, List, Tuple


def H(y: pd.Series) -> float:
    """Entropia de Shannon (bits)."""
    counts = y.value_counts(dropna=False).astype(float)
    p = counts / counts.sum()
    p = p[p > 0]
    return float(-(p * np.log2(p)).sum())

def IG(df: pd.DataFrame, attr: str, y_col: str, baseH: float = None) -> float:
    """Ganho de informação de attr sobre y_col (atributo nominal)."""
    if baseH is None:
        baseH = H(df[y_col])
    n = len(df)
    condH = 0.0
    for _, g in df.groupby(attr, dropna=False, observed=True):
        condH += (len(g) / n) * H(g[y_col])
    return baseH - condH

def rank_ig(df: pd.DataFrame, y_col: str, attrs: List[str]) -> List[Tuple[str, float]]:
    """Retorna [(attr, IG)] ordenado do maior para o menor."""
    base = H(df[y_col])
    pares = [(a, IG(df, a, y_col, base)) for a in attrs]
    pares.sort(key=lambda kv: kv[1], reverse=True)
    return pares

def read_any_sep(path: str) -> pd.DataFrame:
    """Lê CSV inferindo separador automaticamente (fallback = vírgula)."""
    try:
        return pd.read_csv(path, sep=None, engine="python")
    except Exception:
        return pd.read_csv(path)
```

```python
def as_categorical(df: pd.DataFrame, y_col: str) -> pd.DataFrame:
    """Converte preditores para category; mantém alvo como está."""
    for c in df.columns:
        if c != y_col:
            df[c] = df[c].astype("category")
    return df

def print_table(title: str, pares: List[Tuple[str, float]], k: int = None):
    print(f"\n{title}")
    print("  Atributo".ljust(16), "IG")
    print("  " + "-" * 24)
    for i, (a, v) in enumerate(pares):
        if k is not None and i >= k: break
        print(f"  {a.ljust(16)} {v:.6f}")


def raiz_e_segundo_nivel(df: pd.DataFrame, y_col: str) -> Dict:
    """Devolve raiz (maior IG) e, para cada valor do ramo da raiz, o melhor atributo seguinte."""
    attrs = [c for c in df.columns if c != y_col]
    base = H(df[y_col])
    ranking = rank_ig(df, y_col, attrs)
    raiz, ig_raiz = ranking[0]

    por_ramo = {}
    for valor in df[raiz].dropna().unique().tolist():
        sub = df[df[raiz] == valor].copy()
        candidatos = [c for c in attrs if c != raiz]
        if len(sub) == 0 or len(candidatos) == 0:
            por_ramo[valor] = []
            continue
        sub_rank = rank_ig(sub, y_col, candidatos)
        por_ramo[valor] = sub_rank
    return {"ranking_global": ranking, "raiz": (raiz, ig_raiz), "por_ramo": por_ramo}
```

```python
def main():
    if len(sys.argv) < 3:
        print("Uso: python ig_restaurante_v2.py <csv_path> <coluna_alvo>")
        sys.exit(1)

    csv_path, y_col = sys.argv[1], sys.argv[2]
    df = read_any_sep(csv_path)

    if y_col not in df.columns:
        raise ValueError(f'Coluna alvo "{y_col}" não encontrada. Colunas: {list(df.columns)}')

    df = as_categorical(df, y_col)
    print("Colunas:", list(df.columns))

    resultado = raiz_e_segundo_nivel(df, y_col)

    # ranking global
    print_table("Ganhos de informação (global)", resultado["ranking_global"])
    raiz, ig_raiz = resultado["raiz"]
    print(f"\n>>> Atributo raiz sugerido (ID3): {raiz}  (IG={ig_raiz:.6f})")

    # melhores do 2º nível por ramo
    print("\n=== 2º nível sugerido por ramo da raiz ===")
    for valor, rank in resultado["por_ramo"].items():
        print_table(f'Ramo {raiz} = "{valor}" (top 3)', rank, k=3)
        if len(rank) > 0:
            print(f'>>> Melhor próximo atributo neste ramo: {rank[0][0]}\n')

if __name__ == "__main__":
    main()
```

```
vitor@vitor-B360-AORUS-GAMING-3:~/Documentos/Repositories/Matérias/IA/Listas/Lista 2$ python3 teste.py restaurante.csv Conclusao
Colunas: ['Alternativo', 'Bar', 'SexSab', 'fome', 'Cliente', 'Preco', 'Chuva', 'Res', 'Tipo', 'Tempo', 'Conclusao']

Ganhos de informação (global)
  Atributo       IG
  ----------------------
  Cliente          0.540852
  Tempo            0.207519
  fome             0.195710
  Preco            0.195710
  SexSab           0.020721
  Chuva            0.020721
  Res              0.020721
  Alternativo      0.000000
  Bar              0.000000
  Tipo             0.000000

>>> Atributo raiz sugerido (ID3): Cliente  (IG=0.540852)

=== 2º nível sugerido por ramo da raiz ===

Ramo Cliente = "Alguns" (top 3)
  Atributo       IG
  ----------------------
  Alternativo      -0.000000
  Bar              -0.000000
  SexSab           -0.000000
>>> Melhor próximo atributo neste ramo: Alternativo


Ramo Cliente = "Cheio" (top 3)
  Atributo       IG
  ----------------------
  fome             0.251629
  Preco            0.251629
  Res              0.251629
>>> Melhor próximo atributo neste ramo: fome


Ramo Cliente = "Nenhum" (top 3)
  Atributo       IG
  ----------------------
  Alternativo      -0.000000
  Bar              -0.000000
  SexSab           -0.000000
>>> Melhor próximo atributo neste ramo: Alternativo
```

Questão 2:

2.1) O cálculo do ganho de informação para cada atributo deu os seguintes resultados:

- Cliente → 0.541
- Tempo → 0.208
- Fome → 0.196
- Preço → 0.196
- SexSab → 0.021
- Chuva → 0.021
- Reserva → 0.021
- Alternativo, Bar e Tipo → 0.000

O atributo com maior ganho de informação é Cliente (0.541), portanto ele será a raiz da árvore

2.2) Após escolher Cliente como raiz, tem-se os subconjuntos para cada valor desse atributo:

- Cliente = “Alguns” → ramo puro (todos exemplos são da mesma classe), portanto não precisa de outro atributo.
- Cliente = “Nenhum” → ramo puro também, ou seja, a classe já está determinada.
  Cliente = “Cheio” → este ramo ainda está misturado. Os atributos Fome, Preço e Reserva apresentam o maior ganho (≈0.252). Assim, qualquer um deles pode ser escolhido como próximo nó.

Portanto, no segundo nível da árvore, apenas o ramo Cliente = Cheio é expandido, e nele o melhor atributo é Fome (ou, de forma equivalente, Preço ou Reserva, pois todos têm o mesmo ganho).