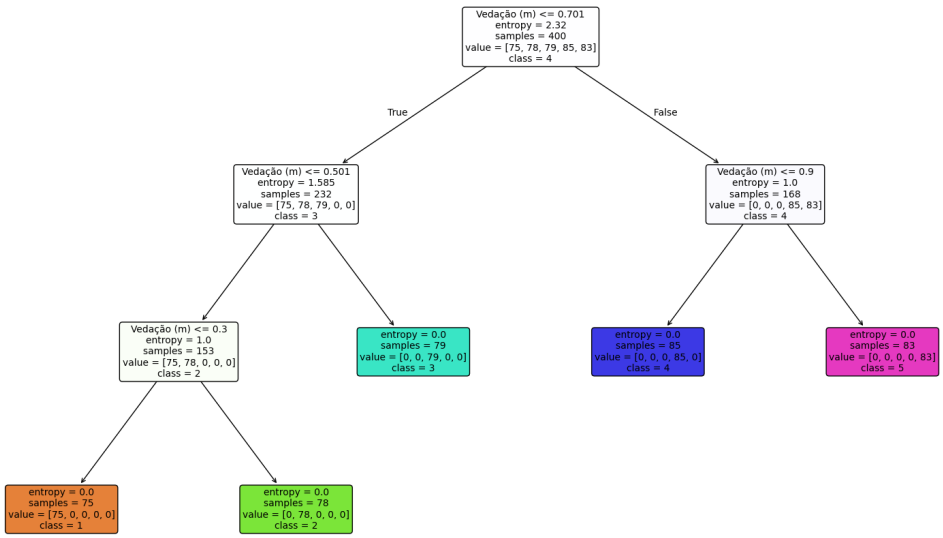
Segundo Trabalho Prático de IA - Árvore de Decisão

Aluno: Pedro Henrique Vilaça Valverde

Disciplina: Inteligência Artificial

Exercício 1: Classificação de Rolamentos

a) Árvore de Decisão Gerada:



b) Regras SE-ENTÃO Extraídas:

```
|--- Vedação (m) <= 0.70
|   |--- Vedação (m) <= 0.50
|   |   |--- Vedação (m) <= 0.30
|   |   |   |--- class: 1
|   |   |--- Vedação (m) >  0.30
|   |   |   |--- class: 2
|   |--- Vedação (m) >  0.50
|   |   |--- class: 3
|--- Vedação (m) >  0.70
|   |--- Vedação (m) <= 0.90
|   |   |--- class: 4
|   |--- Vedação (m) >  0.90
|   |   |--- class: 5
```

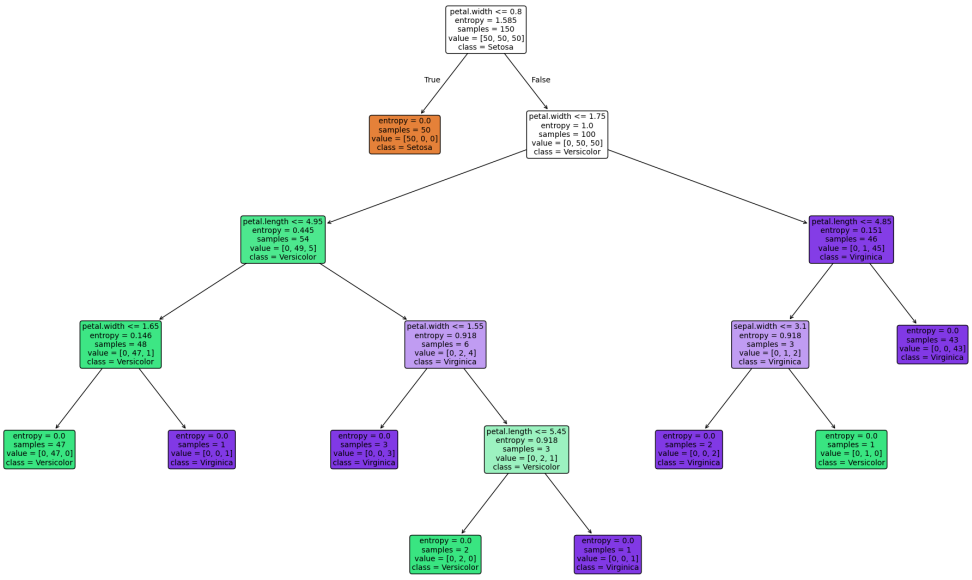
c) e d) Classificações de Novos Casos:

```
Acurácia geral no conjunto de teste: 0.00%
```

Previsões para cada instância de teste:
  - Previsão para [1.693800e+01 5.813460e+02 7.797450e+03 8.970000e-01 2.970000e-01
 8.274900e+04 2.903100e+04 2.004436e-01 2.000000e+00]: Classe '1' (Valor real: '???')
  - Previsão para [1.808300e+01 3.012720e+02 9.500780e+03 7.400000e-02 5.473000e-01
 9.415400e+04 9.518000e+04 4.005661e-01 1.000000e+00]: Classe '2' (Valor real: '???')
  - Previsão para [4.526070e+01 4.966250e+02 1.300090e+03 4.360000e-01 5.219000e-01
 8.399600e+04 6.532000e+03 6.006665e-01 2.000000e+00]: Classe '3' (Valor real: '???')
  - Previsão para [9.090080e+01 1.319410e+02 5.343580e+03 5.680000e-01 3.400000e-03
 8.336200e+04 7.948100e+04 8.004689e-01 9.000000e+00]: Classe '4' (Valor real: '???')
  - Previsão para [4.546410e+01 4.492980e+02 7.994230e+03 1.410000e-01 8.527000e-01
 2.804900e+04 6.575300e+04 1.000814e+00 0.000000e+00]: Classe '5' (Valor real: '???')
  - Previsão para [8.843480e+01 9.319540e+02 7.964180e+03 7.770000e-01 3.866000e-01
 9.672100e+04 9.832200e+04 2.004447e-01 1.000000e+00]: Classe '1' (Valor real: '???')
  - Previsão para [9.337810e+01 1.264430e+02 4.547610e+03 4.570000e-01 5.148000e-01
 4.525300e+04 5.748300e+04 4.008766e-01 6.000000e+00]: Classe '2' (Valor real: '???')
  - Previsão para [7.003270e+01 4.987700e+02 8.257540e+03 6.110000e-01 9.500000e-02
 7.439000e+03 9.618300e+04 6.002808e-01 2.000000e+00]: Classe '3' (Valor real: '???')
  - Previsão para [3.142920e+01 9.780700e+01 5.739180e+03 8.900000e-01 1.719000e-01
 7.806700e+04 5.519500e+04 8.000908e-01 5.000000e+00]: Classe '4' (Valor real: '???')
  - Previsão para [7.6866800e+01 3.2681900e+02 1.6656000e+03 3.7000000e-01 7.6620000e-01
  8.4571000e+04 1.1130000e+03 1.0002803e+00 3.0000000e+00]: Classe '5' (Valor real:
'???')
  - Previsão para [7.692780e+01 1.711340e+02 3.435060e+03 7.470000e-01 6.640000e-01
 3.616100e+04 4.704000e+04 2.000531e-01 9.000000e+00]: Classe '1' (Valor real: '???')
  - Previsão para [7.633420e+01 4.596500e+02 7.448520e+03 3.500000e-02 2.199000e-01
 8.281000e+03 7.351100e+04 4.004123e-01 0.000000e+00]: Classe '2' (Valor real: '???')
  - Previsão para [5.105440e+01 6.678860e+02 2.877410e+03 9.280000e-01 2.206000e-01
 2.242000e+03 3.513400e+04 6.009187e-01 3.000000e+00]: Classe '3' (Valor real: '???')
  - Previsão para [8.786720e+01 3.471330e+02 3.675300e+02 8.920000e-01 4.440000e-02
 9.628800e+04 3.627500e+04 8.000185e-01 5.000000e+00]: Classe '4' (Valor real: '???')
  - Previsão para [5.7646000e+00 1.0338700e+02 1.5955000e+03 8.5000000e-01 3.1930000e-01
  9.6002000e+04 7.8858000e+04 1.0008244e+00 2.0000000e+00]: Classe '5' (Valor real:
'???')

Exercício 2: Classificação de Flores Íris

a) Árvore de Decisão Gerada:



b) Regras SE-ENTÃO Extraídas:

```
|--- petal.width <= 0.80
|   |--- class: Setosa
|--- petal.width >  0.80
|   |--- petal.width <= 1.75
|   |   |--- petal.length <= 4.95
|   |   |   |--- petal.width <= 1.65
|   |   |   |   |--- class: Versicolor
|   |   |   |--- petal.width >  1.65
|   |   |   |   |--- class: Virginica
|   |   |--- petal.length >  4.95
|   |   |   |--- petal.width <= 1.55
|   |   |   |   |--- class: Virginica
|   |   |   |--- petal.width >  1.55
|   |   |   |   |--- petal.length <= 5.45
|   |   |   |   |   |--- class: Versicolor
|   |   |   |--- petal.length >  5.45
|   |   |   |   |--- class: Virginica
|   |--- petal.width >  1.75
|   |   |--- petal.length <= 4.85
|   |   |   |--- sepal.width <= 3.10
|   |   |   |   |--- class: Virginica
|   |   |   |--- sepal.width >  3.10
|   |   |   |   |--- class: Versicolor
|   |   |--- petal.length >  4.85
|   |   |   |--- class: Virginica
```

c) e d) Classificações de Novos Casos:

```
Previsão para [5.1, 3.5, 1.4, 0.2]: Espécie 'Setosa'
Previsão para [6.7, 3.0, 5.2, 2.3]: Espécie 'Virginica'
```

Anexo: Código Fonte (resolver_problema.py)

```
# -*- coding: utf-8 -*-

import os
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_text
from sklearn.metrics import accuracy_score

# --- CONFIGURAÇÕES ---
RESULT_DIR = "./result"

# --- CRIAÇÃO DO DIRETÓRIO DE RESULTADOS ---
if not os.path.exists(RESULT_DIR):
    os.makedirs(RESULT_DIR)
print(f"Diretório '{RESULT_DIR}' pronto para receber os resultados.")


# --- FUNÇÃO PARA RESOLVER O EXERCÍCIO 1: ROLAMENTOS ---
def resolver_exercicio_1():
    """
    Resolve o exercício de classificação de rolamentos usando arquivos de treino e teste.
    """
    print("\n--- Iniciando Exercício 1: Classificação de Rolamentos ---")

    # 1. Carregar os dados de TREINAMENTO
    caminho_treino = os.path.join(
        "content", "Classificacao de Mancais de Rolamentos (Treinamento).csv"
    )
    try:
        dados_treino = pd.read_csv(caminho_treino)
    except FileNotFoundError:
        print(f"ERRO: Arquivo de treinamento '{caminho_treino}' não encontrado.")
        return

    # 2. Separar Features (X) e Target (Y) para o treinamento
    previsores_nomes = list(dados_treino.columns[:-1])
    X_treino = dados_treino.iloc[:, :-1]
    Y_treino = dados_treino.iloc[:, -1]

    # 3. Criar e Treinar o Classificador
    arvore = DecisionTreeClassifier(criterion="entropy")
    arvore.fit(X_treino, Y_treino)

    acuracia_treino = arvore.score(X_treino, Y_treino)
    print(f"Acurácia do modelo nos dados de treino: {acuracia_treino*100:.2f}%")

    # 4. Gerar e Salvar os Resultados Visuais e Regras

    # a) Construir a árvore (Salvar como imagem)
    fig, ax = plt.subplots(figsize=(20, 12))
    plot_tree(
        arvore,
        feature_names=previsores_nomes,
        class_names=[str(c) for c in arvore.classes_],
        filled=True,
        rounded=True,
        fontsize=10,
    )
```

```
    caminho_imagem = os.path.join(RESULT_DIR, "ex1_arvore.png")
    plt.savefig(caminho_imagem)
    plt.close(fig)
    print(f"Imagem da árvore salva em: {caminho_imagem}")

    # b) Indicar a regra SE-ENTÃO
    regras = export_text(arvore, feature_names=previsores_nomes)
    caminho_regras = os.path.join(RESULT_DIR, "ex1_regras.txt")
    with open(caminho_regras, "w", encoding="utf-8") as f:
        f.write(regras)
    print(f"Regras SE-ENTÃO salvas em: {caminho_regras}")

    # 5. Carregar dados de TESTE e fazer as classificações
    caminho_teste = os.path.join(
        "content", "Classificacao de Mancais de Rolamentos (Teste).csv"
    )
    try:
        dados_teste = pd.read_csv(caminho_teste)
        X_teste = dados_teste.iloc[:, :-1]
        Y_verdadeiro = dados_teste.iloc[:, -1]
    except FileNotFoundError:
        print(
            f"ERRO: Arquivo de teste '{caminho_teste}' não encontrado. As previsões não serão geradas."
        )
        return

    # Fazer previsões para todo o conjunto de teste
    Y_previsoes = arvore.predict(X_teste)

    # Calcular a acurácia no conjunto de teste
    acuracia_teste = accuracy_score(Y_verdadeiro, Y_previsoes)
    print(f"Acurácia do modelo nos dados de teste: {acuracia_teste*100:.2f}%")

    # Salvar os resultados das previsões em um arquivo
    previsoes_texto = []
    previsoes_texto.append(
        f"Acurácia geral no conjunto de teste: {acuracia_teste*100:.2f}%\n"
    )
    previsoes_texto.append("Previsões para cada instância de teste:")

    for i in range(len(dados_teste)):
        instancia = X_teste.iloc[i].values
        previsao = Y_previsoes[i]
                resultado = f"    - Previsão para {instancia}: Classe '{previsao}' (Valor real:
'{Y_verdadeiro.iloc[i]}')"
        previsoes_texto.append(resultado)

    caminho_previsoes = os.path.join(RESULT_DIR, "ex1_previsoes.txt")
    with open(caminho_previsoes, "w", encoding="utf-8") as f:
        f.write("\n".join(previsoes_texto))
    print(f"Previsões do conjunto de teste salvas em: {caminho_previsoes}")
    print("--- Exercício 1 Finalizado ---")


# --- FUNÇÃO PARA RESOLVER O EXERCÍCIO 2: IRIS DATASET ---
def resolver_exercicio_2():
    """
    Resolve o exercício de classificação do dataset Iris.
    """
    print("\n--- Iniciando Exercício 2: Classificação de Flores Íris ---")
```

```
    caminho_arquivo = os.path.join("content", "iris.csv")
    try:
        dados = pd.read_csv(caminho_arquivo)
    except FileNotFoundError:
        print(f"ERRO: Arquivo '{caminho_arquivo}' não encontrado.")
        return

    previsores_nomes = list(dados.columns[:-1])
    XX = dados.iloc[:, :-1]
    YY = dados.iloc[:, -1]

    arvore = DecisionTreeClassifier(criterion="entropy")
    arvore.fit(XX, YY)

    acuracia = arvore.score(XX, YY)
    print(f"Acurácia do modelo nos dados de treino: {acuracia*100:.2f}%")

    fig, ax = plt.subplots(figsize=(25, 15))
    plot_tree(
        arvore,
        feature_names=previsores_nomes,
        class_names=arvore.classes_,
        filled=True,
        rounded=True,
        fontsize=10,
    )
    caminho_imagem = os.path.join(RESULT_DIR, "ex2_arvore.png")
    plt.savefig(caminho_imagem)
    plt.close(fig)
    print(f"Imagem da árvore salva em: {caminho_imagem}")

    regras = export_text(arvore, feature_names=previsores_nomes)
    caminho_regras = os.path.join(RESULT_DIR, "ex2_regras.txt")
    with open(caminho_regras, "w", encoding="utf-8") as f:
        f.write(regras)
    print(f"Regras SE-ENTÃO salvas em: {caminho_regras}")

    previsoes_texto = []

    dados_c = [[5.1, 3.5, 1.4, 0.2]]
    previsao_c = arvore.predict(dados_c)
    resultado_c = f"Previsão para [5.1, 3.5, 1.4, 0.2]: Espécie '{previsao_c[0]}'"
    previsoes_texto.append(resultado_c)
    print(resultado_c)

    dados_d = [[6.7, 3.0, 5.2, 2.3]]
    previsao_d = arvore.predict(dados_d)
    resultado_d = f"Previsão para [6.7, 3.0, 5.2, 2.3]: Espécie '{previsao_d[0]}'"
    previsoes_texto.append(resultado_d)
    print(resultado_d)

    caminho_previsoes = os.path.join(RESULT_DIR, "ex2_previsoes.txt")
    with open(caminho_previsoes, "w", encoding="utf-8") as f:
        f.write("\n".join(previsoes_texto))
    print(f"Previsões salvas em: {caminho_previsoes}")
    print("--- Exercício 2 Finalizado ---")


# --- EXECUÇÃO PRINCIPAL ---
```

```
if __name__ == "__main__":
    resolver_exercicio_1()
    resolver_exercicio_2()
    print(
        "\nProcesso finalizado. Todos os arquivos de resultado foram salvos na pasta 'result/'."
    )
```