Atividade: Árvores de Decisão em Python

Tiago Alves de Oliveira

Objetivo

Desenvolver uma árvore de decisão mais complexa com **10 perguntas** e explorar datasets do **Kaggle** e do **UCI Machine Learning Repository** para criar novas aplicações utilizando árvores de decisão.

Parte 1: Desafio - Árvore de Decisão com 10 Perguntas

Implemente uma árvore de decisão que ajude uma pessoa a decidir entre diferentes hobbies ou carreiras, com base nas respostas a 10 perguntas.

Exemplo de Estrutura de Perguntas

Aqui está uma sugestão de estrutura para as perguntas:

1. Você gosta de trabalhar em equipe? (Sim/Não)
2. Prefere atividades ao ar livre? (Sim/Não)
3. Gosta de lidar com números? (Sim/Não)
4. Prefere usar habilidades artísticas? (Sim/Não)
5. Se sente confortável trabalhando com tecnologia? (Sim/Não)
6. Gosta de resolver problemas complexos? (Sim/Não)
7. Prefere atividades que envolvam comunicação? (Sim/Não)
8. Se interessa por cuidar de outras pessoas? (Sim/Não)
9. Gosta de desafios físicos? (Sim/Não)
10. Prefere trabalhar em ambientes organizados? (Sim/Não)

Com base nas respostas, a árvore pode sugerir áreas como:

- Carreiras: Engenharia, Medicina, Artes, Tecnologia, Direito.
- Hobbies: Esportes, Música, Fotografia, Programação, Jardinagem.

Exemplo de Código Inicial

```
def arvore_decisao():
    equipe = input("Voc  gosta de trabalhar em equipe? (sim/n o): ").
        strip().lower()
    if equipe == "sim":
        ar_livre = input("Prefere atividades ao ar livre? (sim/n o): "
            ).strip().lower()
        if ar_livre == "sim":
            print("Sugest o: Explore atividades como esportes
                coletivos ou jardinagem.")
        else:
            tecnologia = input("Se sente confort vel trabalhando com
                tecnologia? (sim/n o): ").strip().lower()
            if tecnologia == "sim":
                print("Sugest o: Considere  reas  de tecnologia ou
                    programa  o.")
            else:
                print("Sugest o: Experimente hobbies criativos como
                    pintura ou m sica.")
    else:
        numeros = input("Gosta de lidar com n meros? (sim/n o): ").
            strip().lower()
        if numeros == "sim":
            print("Sugest o:  reas  como finan as ou engenharia podem
                ser interessantes.")
        else:
            print("Sugest o: Considere hobbies individuais como
                leitura ou fotografia.")

arvore_decisao()
```

Adapte esse exemplo para incluir mais perguntas e oferecer respostas personalizadas com base nas combinações de respostas.

Parte 2: Exploração de Datasets no Kaggle e UCI

Agora, vamos explorar dados reais para aplicar árvores de decisão em problemas práticos.

Instruções

1. Visite os seguintes sites:
   - **Kaggle**: `https://www.kaggle.com`
   - **UCI Machine Learning Repository**: `https://archive.ics.uci.edu/ml/index.php`
2. Pesquise datasets relacionados a temas de seu interesse, como:
   - Saúde (e.g., diagnósticos médicos)
   - Finanças (e.g., classificação de crédito)
   - Educação (e.g., desempenho acadêmico)

- Esportes (e.g., análise de desempenho de jogadores)

3. Faça o download de um dataset e carregue-o no Python utilizando bibliotecas como `pandas`.

Exemplo de Código para Explorar um Dataset

Aqui está um exemplo para carregar e usar um dataset do Kaggle ou UCI:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Carregar o dataset (substitua 'seu_dataset.csv' pelo arquivo baixado)
df = pd.read_csv("seu_dataset.csv")

# Inspecionar os dados
print(df.head())

# Pr -processar os dados (ajuste as colunas conforme necess rio)
X = df[['coluna1', 'coluna2', 'coluna3']]  # Insira as colunas de
    entrada
y = df['target']  # Insira a coluna alvo

# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.3, random_state=42)

# Criar e treinar a  rvore  de decis o
clf = DecisionTreeClassifier(max_depth=4, random_state=42)
clf.fit(X_train, y_train)

# Fazer previs es e avaliar
y_pred = clf.predict(X_test)
print(f"Acur cia: {accuracy_score(y_test, y_pred):.2f}")

# Visualizar a  rvore  de decis o
plt.figure(figsize=(16, 10))
plot_tree(clf, feature_names=X.columns, class_names=['Classe1', '
    Classe2'], filled=True)
plt.show()
```

Tarefa

1. Escolha um dataset no Kaggle ou UCI. 2. Crie uma árvore de decisão para resolver um problema baseado nesse dataset. 3. Documente:

- O dataset escolhido (nome e fonte).
- O problema que a árvore de decisão está resolvendo.
- A acurácia ou métrica de desempenho obtida.

Entrega

Entregue:

1. O código completo da sua árvore de decisão com 10 perguntas.
2. O código e relatório da análise do dataset escolhido no Kaggle ou UCI.

Boa sorte e divirta-se explorando árvores de decisão!