Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Disciplina: Inteligência Artificial
Atividade: Lista 5 - IA

Prof.: Cristiane Neri Nobre
Nome: Davi Cândido de Almeida _857859

Objetivo

Implementar, comparar e justificar as escolhas de projeto de três algoritmos de árvore de decisão:
• ID3 (ganho de informação; atributos categóricos),
• C4.5 (razão de ganho; lida com contínuos),
• CART (índice de Gini; divisões binárias).

Códigos:

Implementação:
GitHub: https://github.com/DaviKandido/IA-Inteligencia-Artificial/tree/main/Listas/Lista%204
Colab: https://colab.research.google.com/drive/1JkK88RME31zvVKGeZvypEsSHsxHVzqMV?usp=sharing
Tratamento da base de dados do Titanic:
Colab: https://colab.research.google.com/drive/1JkK88RME31zvVKGeZvypEsSHsxHVzqMV?usp=sharing

Decisões/Discussão do algoritmo:

Algoritmo em Alto Nível

ArvoreDecesion(dataSet=dataSet, classe='Conclusao', criterio='entropia'):
1) Verifica se hora de parada (se todos os exemplos da classe forem iguais, não há dados para classificação)
2) Escolhe o "melhor" atributo
  2.1) Calcula a entropia da classe
  2.2) Calcula a entropia dos atributos
  2.3) Seleciona o atributo com maior ganho de informação
3) Cria uma aresta por valor do atributo
4) Cria um nó (filho) por valor
  4.1) Separa o subconjunto para sub valor do atributo filho
5) Chama a função recursivamente

Primeiramente se início a produção do algoritmo definindo as estruturas necessárias para a construção das árvores, estruturas estas que seriam comuns tanto para o ID3, C4.5 e CART:

Criou-se as estruturas/Classes de No, Árvore e funções úteis como enums e de cálculo de Entropia e Gini:

A estrutura do No:

```
class No:
  def __init__
     self.atributo              # usado quando for nó intermédio (Atributo a ser testado)
     self.valor                 # usado quando for nó intermédio
     self.grauConfusao          # Valor da Entropia ou Geni
     self.value = list(value)   # Quantidade de Exemplos em cada classe
     self.sample = sample       # Quantidade de Exemplos por no
     self.filhos = list(filhos) # usado quando for nó intermédio
     self.classe = classe       # usado quando for nó folha (Classe final)
  def eh_folha(self):
     return self.classe is not None

  def eh_intermediario(self):
     return not self.eh_folha()
```

A estrutura do Árvore:

```
class Árvore:
    def __init__:
          self.raiz = raiz          # No raiz da árvore
          self.regras = regras      # Conjunto de regras obtidas na árvore

           def imprimir(self, no=None, nivel=0): # Percorre a árvore imprimindo-a
```

Posteriormente se início a produção dos métodos de cálculo de entropia e gini:

```
def entropia(labels):
      valores, contagens = np.unique(labels, return_counts=True)
      probs = contagens / contagens.sum()
      return -np.sum(probs * np.log2(probs))

def gini(labels):
       valores, contagens = np.unique(labels, return_counts=True)
       probs = contagens / contagens.sum()
       return 1 - np.sum(probs**2)
```

Posteriormente a produção da estrutura básica, e dos métodos de apoio ao algoritmo se início a produção da lógica interna do mesmo, no qual será explicado em alto pelos tópicos a seguir:

O algoritmo se inicia pelo método Árvore Decisión(), que recebe como parâmetro, a base de dados, a classe a se classificar e o algoritmo escolhido (ID3, C4.5 ou CART)

```
ÁrvoreDecision()
        árvore = Árvore ()      # Inicializa-se a árvore
        árvore.raiz = No()      # Inicializa-se o nó raiz da árvore
        árvore.raiz = contruir_Arvore()
```

Chama o método recursivo de construção da árvore passando como parâmetro os mesmos atribuídos ao Árvore Decision() mais o nó inicial (raiz)

No método construir_arvore() é onde boa parte da lógica da construção, divisão e seleção dos atributos a cada nó acontece, como explicado pelo algoritmo em alto nível a seguir:

```
construir_arvore()
        no.sample
        no.value        # Preenche estatísticas do nó atual

        if algoritimo == algoritimo.CART:       # Calcula o garu de confusão do no atual
          no.grauConfusao = gini(dataSet[classe])
         else:
          no.grauConfusao = entropia(dataSet[classe])

        if hora_de_parada()   # Verifica se é o momento de parada do algoritimo
                Parar

        melhor_atributo = escolher_melhor_atributo(dataSet, classe, algoritimo)

        # Método responsável por escolher o melhor atributo para o nó atual, com base no
algoritmo selecionado

        for valor in dataSet[melhor_atributo].unique():
                subset = separaSubsete      # Divide o DataBase para cada valor unico
                filho = construir_arvore()
                no.filhos.append(filho)

        # Chama recursivamente a construção da árvore para cada valor único do melhor
atributo do nó, o método recursivo garante que todas as possibilidades serão atendidas

        return no       # Retorna o nó com suas devidas classificações
```

Também vale ressaltar que o algoritmo de escolha do melhor atributo segue a lógica desenvolvida em sala, com pequenas adaptações para comportar as peculiaridades envolvidas nos algoritmos do C4.5 e CART como o uso do gini em vez da entropia para o cart e lidar com atributos contínuos, e com pesos para valores com alta cardinalidade para o C4.5

```
escolher_melhor_atributo()
    atributos = lista de colunas exceto a classe
    base_score = entropia(classe) ou gini(classe)   # depende do algoritmo escolhido

    para cada atributo em atributos:
        calcular score_atributo = média ponderada da impureza dos subconjuntos
        se algoritimo == C4.5:
            normalizar pelo split_info

        ganho = base_score - score_atributo
        atualizar melhor_atributo se ganho for maior

    return melhor_atributo
```

Foram produzidos testes de classificação com a base de dados dos restaurantes, haja vista que já se possuíam os resultados calculados facilitando assim os teste:

Veja abaixo a classificação/resultados de cada algoritmo:

---------- Árvore ID3: ----------

```
- Atributo: Cliente (impureza: 1.0000, amostra: 12)
  Valor da aresta: Alguns
    - Classe: Sim (amostra: 4)
  Valor da aresta: Cheio
    - Atributo: fome (impureza: 0.9183, amostra: 6)
      Valor da aresta: Sim
        - Atributo: Tipo (impureza: 1.0000, amostra: 4)
          Valor da aresta: Tailandes
            - Atributo: SexSab (impureza: 1.0000, amostra: 2)
              Valor da aresta: Nao
                - Classe: Nao (amostra: 1)
              Valor da aresta: Sim
                - Classe: Sim (amostra: 1)
          Valor da aresta: Italiano
            - Classe: Nao (amostra: 1)
          Valor da aresta: Hamburger
            - Classe: Sim (amostra: 1)
      Valor da aresta: Nao
        - Classe: Nao (amostra: 2)
  Valor da aresta: Nenhum
    - Classe: Nao (amostra: 2)
```

---------- Árvore C4 .5: ----------

- Atributo: Cliente (impureza: 1.0000, amostra: 12)
  Valor da aresta: Alguns
    - Classe: Sim (amostra: 4)
  Valor da aresta: Cheio
    - Atributo: fome (impureza: 0.9183, amostra: 6)
      Valor da aresta: Sim
        - Atributo: SexSab (impureza: 1.0000, amostra: 4)
          Valor da aresta: Nao
            - Classe: Nao (amostra: 1)
          Valor da aresta: Sim
            - Atributo: Preco (impureza: 0.9183, amostra: 3)
              Valor da aresta: R
                - Classe: Sim (amostra: 2)
              Valor da aresta: RRR
                - Classe: Nao (amostra: 1)
      Valor da aresta: Nao
        - Classe: Nao (amostra: 2)
  Valor da aresta: Nenhum
    - Classe: Nao (amostra: 2)

---------- Árvore CART: ----------

- Atributo: Cliente (impureza: 0.5000, amostra: 12)
  Valor da aresta: Alguns
    - Classe: Sim (amostra: 4)
  Valor da aresta: Cheio
    - Atributo: fome (impureza: 0.4444, amostra: 6)
      Valor da aresta: Sim
        - Atributo: Tipo (impureza: 0.5000, amostra: 4)
          Valor da aresta: Tailandes
            - Atributo: SexSab (impureza: 0.5000, amostra: 2)
              Valor da aresta: Nao
                - Classe: Nao (amostra: 1)
              Valor da aresta: Sim
                - Classe: Sim (amostra: 1)
          Valor da aresta: Italiano
            - Classe: Nao (amostra: 1)
          Valor da aresta: Hamburger
            - Classe: Sim (amostra: 1)
      Valor da aresta: Nao
        - Classe: Nao (amostra: 2)
  Valor da aresta: Nenhum
    - Classe: Nao (amostra: 2)