Prova IA (AB1)

Prof. Evandro Costa

Alunos: Carllos Eduardo

João Tavares

Rick Santos

Wagner Anthony

1) Construa 3 versões de bases de conhecimento com regras do tipo SE...Então... a partir da base de dados seguinte (**ampliada com mais 6 exemplos** E15, E16, E17, E18, E19, E20, **totalizando 20**), usando os 3 algoritmos C4.5, CART e PRISM (ou JRipper). Apresente a descrição processo de construção com as etapas de construção da árvore, nos casos do C4.5 e CART, etapas de construção das regras, no caso do PRISM. Discuta os resultados, fazendo comparações. Para as construções com C4.5, CART e PRISM (ou JRipper), use implementações deles disponíveis em alguma biblioteca de "Machine Learning" ou se preferir, implemente os algoritmos e mostre o processo de construção e o resultado.

Indução de Árvore de Decisão

- **Ex.: Conjunto de Exemplos de Avaliação de Risco de crédito:**
- **4 atributos e 3 classes** (Baseado no livro do Luger: IA)

	Historia de Crédito	Dívida	Garantia	Renda	Risco
E1	Ruim	Alta	Nenhuma	$0 a $15k	Alto
E2	Desconhecida	Alta	Nenhuma	$15 a $35k	Alto
E3	Desconhecida	Baixa	Nenhuma	$15 a $35k	Moderado
E4	Desconhecida	Baixa	Nenhuma	$0 a $15k	Alto
E5	Desconhecida	Baixa	Nenhuma	Acima de $35k	Baixo
E6	Desconhecida	Baixa	Adequada	Acima de $35k	Baixo
E7	Ruim	Baixa	Nenhuma	$0 a $15k	Alto
E8	Ruim	Baixa	Adequada	Acima de $35k	Moderado
E9	Boa	Baixa	Nenhuma	Acima de $35k	Baixo
E10	Boa	Alta	Adequada	Acima de $35k	Baixo
E11	Boa	Alta	Nenhuma	$0 a $15k	Alto
E12	Boa	Alta	Nenhuma	$15 a $35k	Moderado
E13	Boa	Alta	Nenhuma	Acima de $35k	baixo
E14	Ruim	Alta	Nenhuma	$15 a $35k	Alto

2) Com base na arquitetura conceitual de um agente baseado em conhecimento, mostrada nas aulas e a seguir, implemente um sistema de diagnóstico médico. Considere a base de conhecimento hipotética seguinte, ampliando-a com acréscimo de mais 5 regras (hipotéticas) para outros diagnósticos, por exemplo,

de resfriado, H3N2, dengue, covid19. A referida base é a seguinte, adaptado do Livro Inteligência Artificial, do Ben Coppin), ilustrando uma possível demanda por encadeamento misto.

R1: SE Dor de cabeça = Sim ENTÃO Receitar analgésico

R2: SE Dor de cabeça = Sim E Garganta inflamada = Sim E Tosse = Sim

ENTÃO Diagnóstico = Gripe

R3: SE Cansaço = Sim E Dor de cabeça = Sim

ENTÃO Diagnóstico = Mononucleose infecciosa

R4: SE Cansaço = Sim E Garganta inflamada = Sim

ENTÃO Diagnóstico = Amigdalite

R5: SE Cansaço = Sim ENTÃO Diagnóstico = Estresse

Neste caso, considere uma memória de trabalho com um valor inicial Dor de Cabeça = Sim, supondo uma situação com propósito de diagnóstico na qual um determinado usuário informa ao sistema que está com dor de cabeça. Assim, a máquina de inferência pode iniciar com encadeamento para frente para derivar novos fatos, após isso pode fazer perguntas para prosseguir sua análise usando encadeamento para trás.

prints das resoluções abaixo.

o código também se encontra no github classroom juntamente com os códigos da lista anterior.

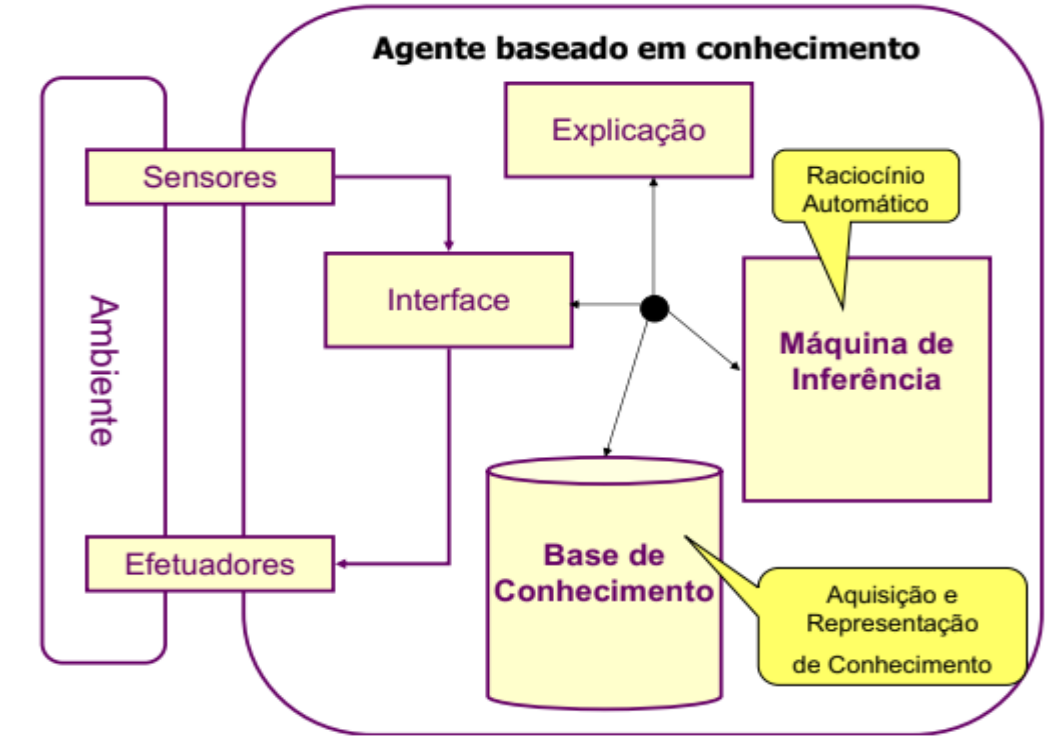
tips2/resolucoes-Ia: lista-1-JT4v4res created by GitHub Classroom

```python
dor_de_cabeca = False
febre = False

# Resfriado e h3n2
espirro = False

# Resfriado
coriza = False

# H1N1 e Dengue
dor_articulacao = False

# H1N1
olhos_irritados = False

# h3n2 e covid
dor_na_garganta = False

# Covid19
falta_de_ar = False

# Dengue
dor_atras_olhos = False

diagnostico = False


def factual_base():
    global dor_de_cabeca

    dor_de_cabeca = True
```

```python
def diagnostico_print(diagnostico):
    print(f'Diagnóstico é {diagnostico}')


if __name__ == '__main__':
    factual_base()

    print('-----------------------')
    print('QUAIS SINTOMAS VOCÊ TEM')
    print('dor_de_cabeca, febre')
    print('espirro, coriza')
    print('dor_articulacao')
    print('olhos_irritados, diarreia')
    print('dor_na_garganta, falta_de_ar')
    print('dor_atras_olhos')
    print('-----------------------')

    sintomas = input('Descreva sintomas: ')
    print(sintomas)
```

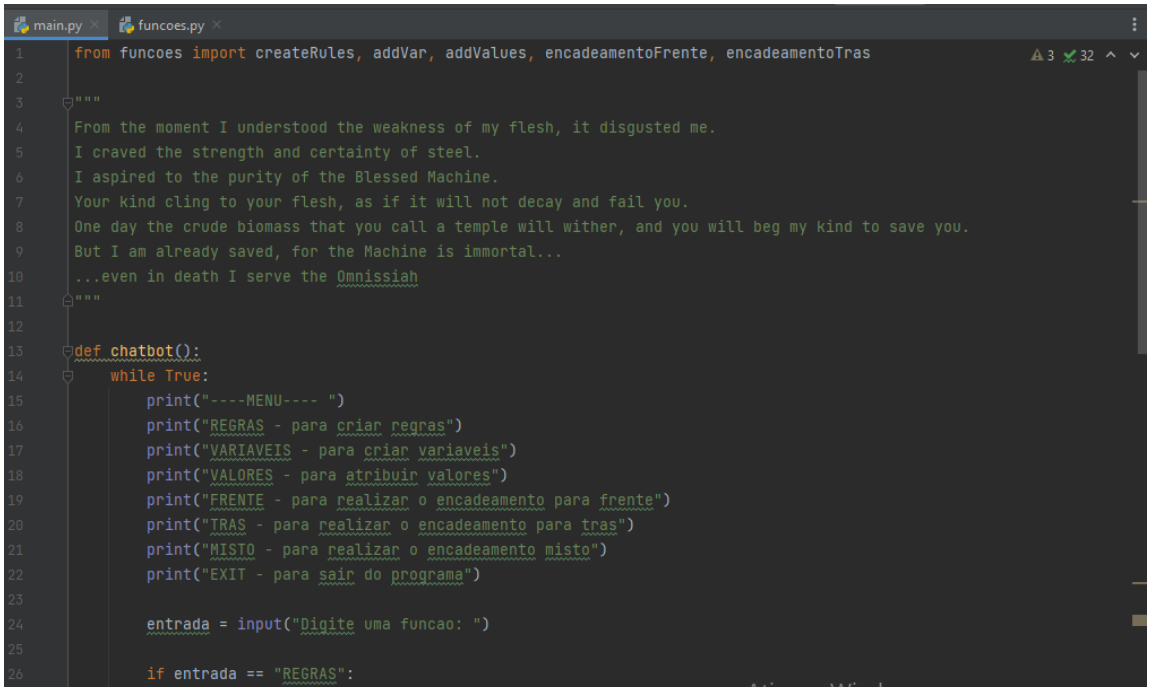
```
while True:
    if dor_de_cabeca and febre and dor_articulacao and olhos_irritados:
        diagnostico_print("H1N1")
        break

    if dor_de_cabeca and febre and dor_articulacao:
        diagnostico_print("Dengue")
        break

    if dor_de_cabeca and febre and espirro:
        diagnostico_print("H3N2")
        break

    if dor_de_cabeca and febre and falta_de_ar:
        diagnostico_print("COVID19")
        break

    if dor_de_cabeca and coriza and espirro:
        diagnostico_print("Resfriado")
        break

    if dor_de_cabeca:
        diagnostico_print("Dor de cabeça, tome um analgésico")
        break

    if "febre" in sintomas:
        febre = True

    if "espirro" in sintomas:
        espirro = True
```

```python
if "coriza" in sintomas:
    coriza = True

if "dor_articulacao" in sintomas:
    dor_articulacao = True

if "olhos_irritados" in sintomas:
    olhos_irritados = True

if "diarreia" in sintomas:
    diarreia = True

if "dor_na_garganta" in sintomas:
    dor_na_garganta = True

if "falta_de_ar" in sintomas:
    falta_de_ar = True

if "dor_atras_olhos" in sintomas:
    dor_atras_olhos = True
```



3) Elabore e implemente uma ferramenta para construção de sistema baseado em conhecimento, com base na arquitetura conceitual do agente baseado em conhecimento, mostrado na questão 2. Especificamente, esta ferramenta deveria ter os módulos e funcionalidade similar ao Expert Sinta, considerando editor de base de conhecimento, motor de inferência (só que aqui é encadeamento híbrido, para trás e para frente), explanação e diálogo em linguagem natural via chatbot. Para avaliar a ferramenta desenvolvida, considere a base de conhecimento a seguir e a da questão 2.

Regra1: Se Rendimento = Alto então conceda_empréstimo = Sim. Regra2: Se Rendimento = Médio E É_bacharel_ou_superior = Sim E Tem_Emprego = Sim então conceda_empréstimo = Sim.
Regra 3: Se Rendimento = Médio E É_bacharel_ou_superior = Sim E Tem_Emprego = Não então continua_a_investigar = Sim.
Regra 4: Se Rendimento = Médio E É_bacharel_ou_superior = Não E Tem_Emprego = Não então conceda_empréstimo = Não.
Regra 5: Se Rendimento = Médio E É_bacharel_ou_superior = Não E Tem_Emprego = Sim então continua_a_investigar = Sim.
Regra 6: Se Rendimento = Baixo E Referências = Boas então continua_a_investigar = Sim.
Regra 7: Se Rendimento = Baixo E Referências = Más então conceda_empréstimo = Não.



main.py | funcoes.py

```
        if entrada == "REGRAS":
            createRules()

        if entrada == "VARIAVEIS":
            addVar()

        if entrada == "VALORES":
            addValues()

        if entrada == "FRENTE":
            encadeamentoFrente()

        if entrada == "TRAS":
            encadeamentoTras()

        if entrada == "MISTO":
            encadeamentoFrente()
            encadeamentoTras()

        if entrada == "EXIT":
            break

chatbot()
```

main.py | funcoes.py

```
data = {"variaveis": [], "valores": [], "se": [], "entao": [], "rules": [],
        "baseData":["referencias_sim=sim","e_bacharel_ou_superior_nao=nao"],
        "objetivo": ['rendimento_alto=alto']}

def searchObj(lista, obj):
    for x in lista:
        if x == obj:
            print("Encontramos o objetivo: " + str(obj))

def searchVar(lista, var):
    x = 0
    for n in lista:
        if n == var:
            return x
        x += 1
    return -1

def addVar():
    print("Digite as variaveis ou digite EXIT para sair...")

    while True:
        i = input("Digite uma variavel: ")

        if i != "EXIT":
            data["variaveis"].append(i)
        else:
```

main.py funcoes.py

```
def addVar():
    print("Digite as variaveis ou digite EXIT para sair...")

    while True:
        i = input("Digite uma variavel: ")

        if i != "EXIT":
            data["variaveis"].append(i)
        else:
            break

def addValues():
    print("Digite os valores para as variaveis ou digite EXIT para sair...")

    for x in range(len(data["variaveis"])):
        i = input("Digite um valor: ")

        if i != "EXIT":
            data["valores"].append(i)
        else:
            break

def selectOperators():
    i = input("Digite o operador AND ou ENTAO")
```

main.py funcoes.py

```
def addValues():
    print("Digite os valores para as variaveis ou digite EXIT para sair...")

    for x in range(len(data["variaveis"])):
        i = input("Digite um valor: ")

        if i != "EXIT":
            data["valores"].append(i)
        else:
            break

def selectOperators():
    i = input("Digite o operador AND ou ENTAO")

    if i == "AND":
        return "and"
    else:
        return "entao"

def rmvSame(lista):
    new_lista = []

    for i in lista:
        if i not in new_lista:
            i = i.replace(" ", "")
```

```python
def rmvSame(lista):
    new_lista = []

    for i in lista:
        if i not in new_lista:
            i = i.replace(" ", "")
            new_lista.append(i)

    return new_lista

def createRules():
    rule = ""

    while True:
        if rule[-6:-1].replace(" ", "") != 'entao':
            rEntao = True

        else:
            rEntao = False

        m = input("Digite a primeira variavel: ")
        mIndex = searchVar(data["variaveis"], m)
```

```python
    rule = ""

    while True:
        if rule[-6:-1].replace(" ", "") != 'entao':
            rEntao = True

        else:
            rEntao = False

        m = input("Digite a primeira variavel: ")
        mIndex = searchVar(data["variaveis"], m)

        if mIndex == -1:
            print("Digite uma variavel pertencente ao banco de dados e tente novamente...")
            return -1
        else:
            rule += f' {data["variaveis"][mIndex]} = {data["valores"][mIndex]} '

        if rEntao:
            op = selectOperators()

            data["se"].append(rule)

            rule += "/" + str(op) + "/"
```

[tips2/resolucoes-Ia: lista-1-JT4v4res created by GitHub Classroom]

Obs.: Os valores das questões, são: 1) e 2), cada uma vale 2 pontos; 3) vale 6 pontos.