Definir confiança

confiança = 0.015

BD = carregarBaseDeDados()

**def** carregarBaseDeDados():  
 print(**"Leitura da base de dados"**)  
 **with** open(**'bd.csv'**,**'r'**) **as** arquivo:  
 dados = []  
 cabecalho = next(arquivo)

**for** linha **in** arquivo: itens = linha.split(**','**)  
 itens = itens[1:]  
 linhatemporaria = []  
 **for** item **in** itens:  
 linhatemporaria.append(item.rstrip())  
 dados.append(linhatemporaria)  
 arquivo.close()  
 **return** dados

APOS lida a BD temos que PEGAR TODOS os padrões já encontrados e gerar as regras com eles,

Portanto devemos ler o arquivo de padrões gerados:

Lembrando que há um divisor entre o padrão e o valor de suporte é o operador (:)

**def** carregaPadroes():  
 listaDePadroes = []  
 listaDeSuportes = []  
 print(**"Leitura dos padrões identificados"**)  
 **with** open(**'padroesEncontrados.csv'**,**'r'**) **as** conjuntoDePadroes:  
 **for** linha **in** conjuntoDePadroes:  
 valores = linha.split(**':'**)  
 padrao = valores[0].split(**','**)  
 suporte = valores[1]  
 listaDePadroes.append(padrao)  
 listaDeSuportes.append(suporte)  
  
  
 **return** listaDePadroes, listaDeSuportes

PARA CADA PADRÃO

GERAR REGRAS

**def** geraRegras(padrao,BD,confianca,suportePadrao):  
 print(**'combinando itens do padrao %s'** %padrao)  
 tamPadrao = len(padrao)

**for** i **in** range(tamPadrao):  
 antecedente = padrao[:i+1]  
 consequente = padrao[i+1:]  
  
 confRegra = calculaConfianca(antecedente,consequente,BD,suportePadrao)  
 print(confRegra)

**if** confRegra >= confianca:  
 regraTraduzida = traduzRegra(antecedente,consequente)  
 gravaNoArquivo(**'regrasEncontradas-Traduzidas.csv'**,str(regraTraduzida), confRegra)  
  
 regraLimpa = limpaRegras(antecedente,consequente)  
 gravaNoArquivo(**'regrasEncontradas.csv'**,str(regraLimpa), confRegra)

função para limpar as regras dos colchetes e salvar em um format mais bonito

**def** limpaRegras(antecedente,consequente):  
 regraLimpa = **','**.join(antecedente)  
 regraLimpa += **"-->"** regraLimpa += **','**.join(consequente)  
 **return** regraLimpa

Que tal imprimir as regras de uma forma mais intuitive?

**def** traduzRegra(antecedente,consequente):  
 textoAntecedente = **""  
  
 for** item **in** antecedente:  
 textoAntecedente+=codigosLivros.getNomeLivro(item)+**","** textoAntecedente += **" --> "  
  
 for** item **in** consequente:  
 textoAntecedente+=codigosLivros.getNomeLivro(item)+**","  
 return** textoAntecedente

**import** csv  
**import** codigosLivros  
  
**def** carregarBaseDeDados():  
 print(**"Leitura da base de dados"**)  
 **with** open(**'bd.csv'**,**'r'**) **as** arquivo:  
 dados = []  
 cabecalho = next(arquivo) *#o comando next pula o primeiro registro (O cabeçalho)* **for** linha **in** arquivo: *#para cada linha da base de dados lida temos que salvar no banco de dados da memória* itens = linha.split(**','**)  
 itens = itens[1:]  
 linhatemporaria = []  
 **for** item **in** itens:  
 linhatemporaria.append(item.rstrip())  
 dados.append(linhatemporaria)  
 arquivo.close()  
 **return** dados  
  
  
**def** carregaPadroes():  
 listaDePadroes = []  
 listaDeSuportes = []  
 print(**"Leitura dos padrões identificados"**)  
 **with** open(**'padroesEncontrados.csv'**,**'r'**) **as** conjuntoDePadroes:  
 **for** linha **in** conjuntoDePadroes:  
 valores = linha.split(**':'**)  
 padrao = valores[0].split(**','**)  
 suporte = valores[1]  
 listaDePadroes.append(padrao)  
 listaDeSuportes.append(suporte)  
  
  
 **return** listaDePadroes, listaDeSuportes  
  
**def** calculaConfianca(antecedente,consequente,BD,frequencia):  
 contadorDeOcorrenciasdoAntecedente =0  
 **for** linha **in** BD:  
 print(**'Verificando se o antecedente %s faz parte da linha %s'** %(antecedente,linha))  
  
 setAntecedente = set(sorted(antecedente))  
 setLinhaBD = set(sorted(linha))  
  
 **if** setAntecedente.issubset(setLinhaBD):  
 print(**'ENcontrado'**)  
 contadorDeOcorrenciasdoAntecedente +=1  
 **else**:  
 print(**'não faz parte'**)  
 **return** (float(frequencia)/float(contadorDeOcorrenciasdoAntecedente/len(BD)))  
  
  
**def** gravaNoArquivo(arquivo,regra,suporte):  
 **with** open(arquivo,**'a+'**) **as** resultado:  
 resultado.writelines(str(regra)+**':'**+str(suporte)+**'\n'**)  
 **return  
  
  
  
def** traduzListaLivros(listaLivros):  
 texto = **""  
  
 for** item **in** listaLivros:  
 texto+=codigosLivros.getNomeLivro(item)+**","  
 return** texto  
  
  
  
**def** traduzRegra(antecedente,consequente):  
 textoAntecedente = **""  
  
 for** item **in** antecedente:  
 textoAntecedente+=codigosLivros.getNomeLivro(item)+**","** textoAntecedente += **" --> "  
  
 for** item **in** consequente:  
 textoAntecedente+=codigosLivros.getNomeLivro(item)+**","  
 return** textoAntecedente  
  
  
**def** limpaRegras(antecedente,consequente):  
 regraLimpa = **','**.join(antecedente)  
 regraLimpa += **"-->"** regraLimpa += **','**.join(consequente)  
 **return** regraLimpa  
  
  
**def** geraRegras(padrao,BD,confianca,suportePadrao):  
 print(**'combinando itens do padrao %s'** %padrao)  
 tamPadrao = len(padrao)  
 **for** i **in** range(tamPadrao-1):  
 print(**'Cabeça: %s'**%padrao[:i+1])  
 antecedente = padrao[:i+1]  
 consequente = padrao[i+1:]  
 print(**'Cauda %s'**%padrao[i+1:])  
 confRegra = calculaConfianca(antecedente,consequente,BD,suportePadrao)  
 print(confRegra)  
 **if** confRegra >= confianca:  
 regraTraduzida = traduzRegra(antecedente,consequente)  
 gravaNoArquivo(**'regrasEncontradas-Traduzidas.csv'**,str(regraTraduzida), confRegra)  
  
 regraLimpa = limpaRegras(antecedente,consequente)  
 gravaNoArquivo(**'regrasEncontradas.csv'**,str(regraLimpa), confRegra)  
  
  
  
  
   
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 confianca = 0.015  
 BD = carregarBaseDeDados()  
 todosPadroes, todosSuportes = carregaPadroes()  
  
 *# print(todosPadroes)* **for** i **in** range(len(todosPadroes)):  
 print(todosPadroes[i])  
 **if** len(todosPadroes[i]) > 1:  
 geraRegras(todosPadroes[i],BD,confianca,todosSuportes[i])