Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Disciplina: Inteligência Artificial
Atividade: Lista 6 - IA

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

Questão 01

Fórmulas:

- Ganho(atributo) = Entropia(Classe) - Entropia (Atributo)

A heurística usada na função $ESCOLHER - ATRIBUTO$ é *simplesmente escolher o atributo com o maior ganho.*

- $Entropia(s) = -\sum_{i=1}^{c} p_i \times log_2(p_i)$

Legenda:

Pi: $Refere - se$ à $probabilidade\ de\ um\ dado\ elemento\ pertencer$ à $classe\ i.$
$Log2(pi)$: $Refere - se\ ao\ logaritmo\ de\ base\ 2\ da\ probabilidade\ pi$
$Sinal\ negativo\ (-)$: $Garante\ que\ a\ entropia\ seja\ um\ valor\ positivo$

Cálculos de ganho sobre os atributos base de dados `Restaurante.csv`:

Entropia da Classe:

$$Entropia(Classe) = -((\frac{9}{17} \times log_2(\frac{9}{17})) + (\frac{8}{17}\ log_2(\frac{8}{17})))$$

Entropia (Classe) = -(0,48575 + 0,511747)

Entropia (Classe) = 0,997447

- **Experiência** [Alta, baixa, Média,]

Ganho (Experiência) = 0,997447 - (5/17 * I(4/5,1/5) + 6/17 * I(1/6,5/6) + 6/17 * I(4/6,2/6))

I(4/5,1/5) = - (4/5 * log2(4/5) + 1/5 * log2(1/5)) = 0.212332
I(1/6,5/6) = - (1/6 * log2(1/6) + 5/6 * log2(5/6)) = 0.229420
I(4/6,2/6) = - (4/6 * log2(4/6) + 2/6 * log2(2/6)) = 0.324104

Ganho (Experiência) = 0,997447 - (5/17 * 0.212332 + 6/17 * 0.229420+ 6/17 * 0.324104)

Ganho (Experiência) ≃ 0.231647

- **Interesse**

Ganho (Interesse) = 0,997447 - (7/17 * I(6/7,1/7) + 10/17 * I(3/10,7/10))

I(6/7,1/7) = - (6/7 * log2(6/7) + 1/7 * log(1/7))
I(6/7,1/7) = 0.243630

I(3/10,7/10) = - (3/10 * log2(3/10) + 7/10 * log(7/10))
I(3/10,7/10) = 0.518406

Ganho (Interesse) = 0,997447 - (7/17 * 0.243630 + 10/17 * 0.518406)
Ganho (Interesse) ≃ 0.235466

- **Horas**

Ganho (Horas) = 0,997447 - (8/17 * I(5/8,3/8) + 9/17 * I(4/9,5/9))

I(5/8,3/8) = - (5/8 * log2(5/8) + 3/8 * lg2(3/8))
I(5/8,3/8) = 0.449145

I(4/9,5/9) = - (4/9 * log2(4/9) + 5/9 * log2(5/9))
I(4/9,5/9) = 0.524687

Ganho (Horas) = 0,997447 - (8/17 * 0.449145 + 9/17 * 0.524687)
Ganho (Horas) ≃ 0.023670

Logo de acordo com os cálculos:

- Ganho (Experiência) = 0.231647
- **Ganho (Interesse) = 0.235466 ← Escolhido**
- Ganho Horas (Cliente) = 0.023670

Portanto o atributo que será usado na raiz da árvore, e que dará um maior grau de ganho, ou seja, melhor contribuirá para a classificação dos atributos nessa etapa é o atributo **Interesse com um ganho de 0.235466.**

Portanto resposta correta: b) A raiz da árvore é o atributo Interesse com ganho de 0,235

Questão 02

Gosta de IA (Classe)	Experiência (Alta)	Interesse (Alto)	Horas (Baixas)
Gosta: 9/17	4/9	6/9	4/9
Não Gosta: 8/17	1/8	1/8	5/8

Gostar: 9/17 x 4/9 x 6/9 x 4/9 = 0,069716775
Não Gostar: 8/17 x 1/8 x 1/8 x 5/8 = 0,004595588

Soma = 0,074312363

P (Gostar) = 0,069716775/0,074312363 = 0,9381585 * 100 ≃ 93,81 %

P (Não Gostar) = 0,004595588/0,074312363 = 0,0618415 * 100 ≃ 06,18 %

Portanto resposta correta: **a) 93,81 % e 06,18 %**

Questão 03

Todos os códigos do exercício se encontram em:

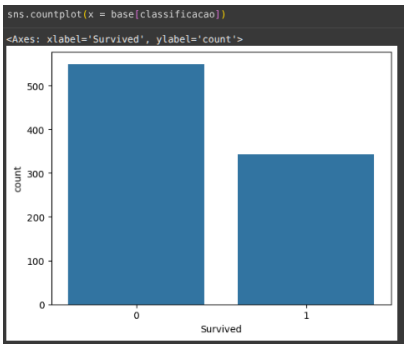
Tratamento:
https://colab.research.google.com/drive/19jW_GD-Rd8OPXInksdNWPxMruT4QEjYT?usp=sharing
Treinamento:
https://colab.research.google.com/drive/1cPuWKV-KdnPMlf2oqJpzLs0m_8pY4bym?usp=sharing

1) Foi primeiramente feito a o tratamento dos dados do DataBase do titanic:

Tratamento:

O tratamento envolveu primeiramente a seleção das colunas dariam mais sentido/impacto na tratativa do problema, sendo portanto as colunas selecionadas:
'Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'

Onde ‘Suvived’ foi a classe a qual se classificou. Após a visualização da distribuição dos dados, observou-se que um maior o problema continha em sua maioria um maior numero de não sobreviventes, observe abaixo:



Posteriormente tratou-se os dados, os transformando em numéricos, onde:

Sex: 0 - male , 1 - famale
Embarked: 0 - S , 1 - C e 2 - Q, demais 3

E para finalizar dropou-se com o método .dropna(axis=0, how="any"), todas as linhas que possuíam dados ausentes, obs: devido ao grande número de dados não encontrou-se problema em executar essa decisão

Separou-se o conjunto em: X_treino, X_teste, y_treino, y_teste, atravez do método train_test_split(X_prev, y_classe, test_size = 0.20, random_state = 42), o semparando entre dados de teste, treino, e classe de teste e treino

Após a visualização da consistência dos dados se exportou o arquivo .pkl

Treinamento:

Importou-se as bibliotecas necessárias para o treinamento e a visualização dos resultados, com destaque as bibliotecas de cada método:

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

Importou-se a divisão dos dados de treino e teste do arquivo .pkl, e após uma pequena visualização dos dados se início a chamada e definição dos parâmetros de cada metodo

modeloNaive = GaussianNB()
modeloDecisionTree = DecisionTreeClassifier(criterion='entropy')
modeloRandomForest = RandomForestClassifier(n_estimators=100, max_features=3, criterion='entropy', random_state = 42)

Selecionou-se as colunas que fazeriam parte dos treinos:

columns_selected = ['Pclass', 'Sex_encoded', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_encoded']

E em fiz iniciou-se o treinamento:

modeloNaive.fit(X_treino_sel, y_treino)
modeloDecisionTree.fit(X_treino_sel, y_treino)
modeloRandomForest.fit(X_treino_sel, y_treino)

Após o treinamento se fez uma predição a partir dos dados de teste:

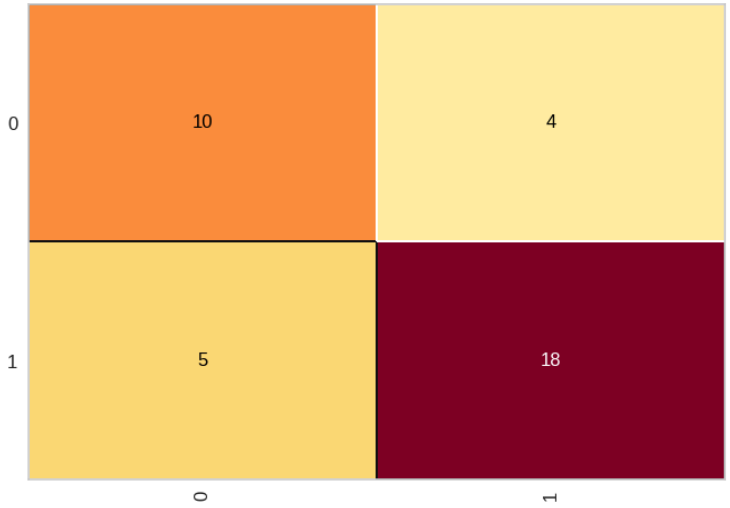
previsoes_naive = modeloNaive.predict(X_teste_sel)
previsoes_decisionTree = modeloDecisionTree.predict(X_teste_sel)
previsoes_RandomForest = modeloRandomForest.predict(X_teste_sel)

Após a visualização das métricas relacionadas a cada predição, pode se concluir que a acurácia de cada método foi:

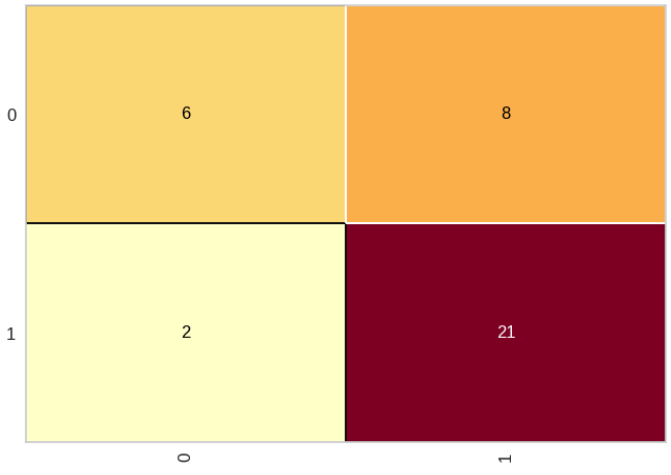
previsoes_naive = 0.7567567567567568 (Maior acurácia)
previsoes_decisionTree = 0.7297297297297297
previsoes_RandomForest = 0.7297297297297297

E as matrizes de confusão foram:

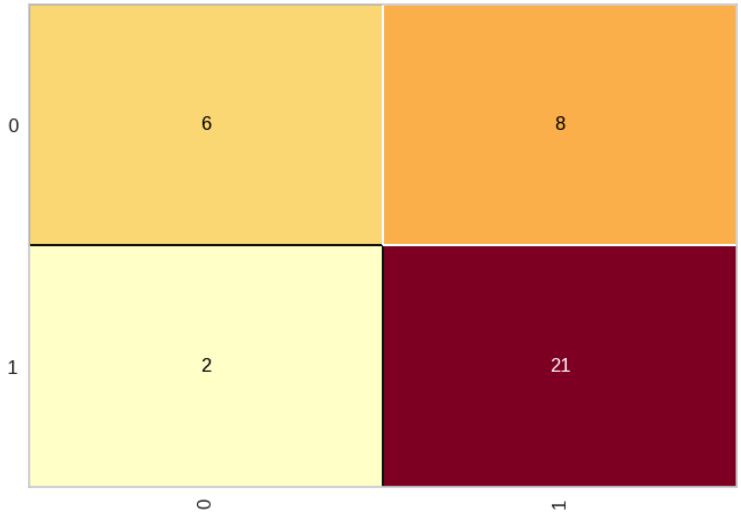
Naive:



DecisionTree:



RandomFlorest:



Conclusão: Pode-se observar resumidamente que todos os três métodos possuiram resultados similares, no entanto com o método do Naive Gaussian possuir um numero de acertos maior quando a decisão de classificar os não sobreviventes, se saindo um pouco pior ao errar mais quanto aos sobreviventes em comparação aos métodos de árvore, vale ressaltar que tanto o RandomFlorest quanto a utilização de uma simples decisionTree resultados nas mesmas classificações.

2) **Nesta etapa se foi refeito o treinamento com otimizadores de hiperparametros**

Em sema se pode observar que o uso de otimizadores de hiperparametros trouxeram resultados melhores as previsões pós treinamento dos dados, tendo como novos resultados das acurácias:

previsoes_naive = 0.7567567567567568
previsoes_decisionTree = 0.7567567567567568
previsoes_RandomForest = 0.7837837837837838 (Nova maior acurácia)

Utilizou-se o otimizador GridSearchCV para encontrar os melhores parâmetros para o método Naive, resultado nos parâmetros: {'var_smoothing': 1e-06}

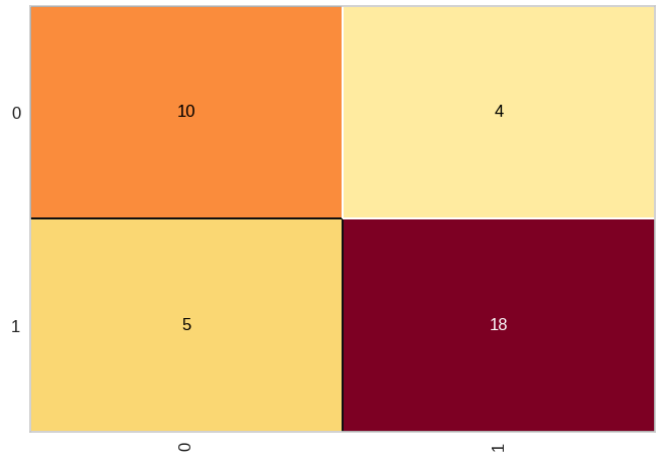
Já para DecisionTree e RandomFlorest utilizou-se o RandomizedSearchCV, pois observou-se que o uso do GridSearchCV resultava em tempos de execução muito grandes, sendo uma melhor escolha o uso do RandomizedSearchCV, postando sendo os melhores parâmetros:

DecisionTree: {'splitter': 'random', 'min_weight_fraction_leaf': 0.1, 'min_samples_split': 2, 'min_samples_leaf': 10, 'min_impurity_decrease': 0.0, 'max_leaf_nodes': 10, 'max_features': None, 'max_depth': None, 'criterion': 'gini', 'class_weight': 'balanced'}

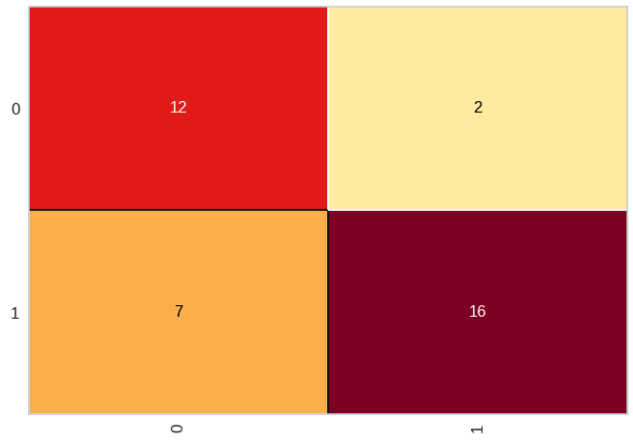
RandomForest: {'oob_score': True, 'n_jobs': -1, 'n_estimators': 300, 'min_weight_fraction_leaf': 0.0, 'min_samples_split': 10, 'min_samples_leaf': 1, 'min_impurity_decrease': 0.0, 'max_leaf_nodes': 10, 'max_features': 'sqrt', 'max_depth': None, 'criterion': 'gini', 'class_weight': 'balanced', 'bootstrap': True}

E as novas matrizes de confusão:

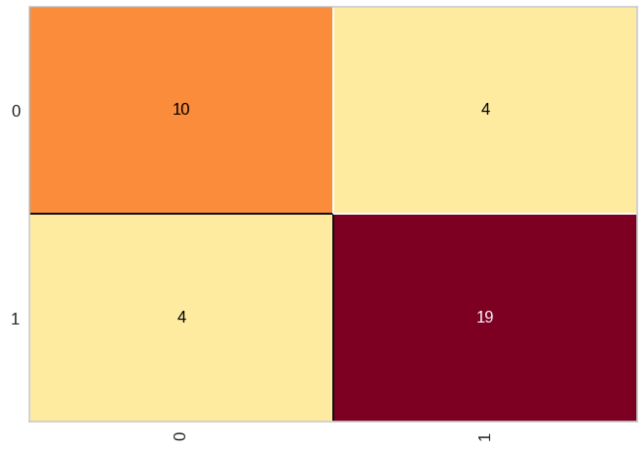
Naive:



DecisionTree:



RandomFlorest:



Questão 04

Suporte mínimo aceitável de 0.3 e confiança de 0.8:

Cálculos dos suportes: (Suporte mínimo 0.3)
ItemSet 1:
Leite: 2/10 = 0.2 x
Café: 3/10 = 0.3
Cerveja: 2/10 = 0.2 x
Pão: 5/10 = 0.5
Manteiga: 5/10 = 0.5
Arroz: 2/10 = 0.2 x
Feijão: 2/10 = 0.2 x

ItemSet 2:

{ Café, Pão } = 3/10 = 0.3
{ Café, Manteiga } = 3/10 = 0.3
{ Pão, Manteiga } = 4/10 = 0.3

ItemSet 3:

{ Café, Pão, Manteiga } = 3/10 = 0.3

Cálculo das Regras (Associações): (Confiança mínima 0.8)

{ Café, Pão } :
Café -> Pão = 3/3 = 1
Pão -> Café = 3/5 = 0.6 x

{ Café, Manteiga } :
Café -> Manteiga = 3/3 = 1
Manteiga -> Café = 3/5 = 0.6 x

{ Pão, Manteiga } :
Pão -> Manteiga = 4/5 = 0.8
Manteiga -> Pão = 4/5 = 0.8

{ Café, Pão, Manteiga }:
Café + Pão -> Manteiga = 3/3 = 1
Café + Manteiga -> Pão = 3/3 = 1
Pão + Manteiga -> Café = 3/4 = 0.75 x

Café -> Pão + Manteiga = 3/3 = 1
Pão -> Café + Manteiga = 3/5 = 0.6 x
Manteiga -> Pão + Café = 3/5 = 0.6 x

Portanto teremos um total de 7 regras as quais são:

1. Café -> Pão
2. Café -> Manteiga
3. Pão -> Manteiga
4. Manteiga -> Pão
5. Café + Pão -> Manteiga
6. Café + Manteiga -> Pão
7. Café -> Pão + Manteiga

Questão 05 e 06

Códigos disponíveis em:
https://colab.research.google.com/drive/10HxmdStVIR-X9_bKb7blvU_Lz8GglyDB?usp=sharing

Executou-se o algoritmo e obteve se o seguintes resultados:

```
RegrasFinais.sort_values(by='lift', ascending =False)
```

	Antecedente	Consequente	suporte	confianca	lift
6	[Cafe]	[Pao, Manteiga]	0.3	1.00	2.5
11	[Pao, Manteiga]	[Cafe]	0.3	0.75	2.5
0	[Cafe]	[Manteiga]	0.3	1.00	2.0
1	[Manteiga]	[Cafe]	0.3	0.60	2.0
3	[Pao]	[Cafe]	0.3	0.60	2.0
2	[Cafe]	[Pao]	0.3	1.00	2.0
8	[Pao]	[Manteiga, Cafe]	0.3	0.60	2.0
7	[Manteiga]	[Pao, Cafe]	0.3	0.60	2.0
9	[Manteiga, Cafe]	[Pao]	0.3	1.00	2.0
10	[Pao, Cafe]	[Manteiga]	0.3	1.00	2.0
4	[Manteiga]	[Pao]	0.4	0.80	1.6
5	[Pao]	[Manteiga]	0.4	0.80	1.6

Dos quais ao se filtrar as regras com confiança >= 0.8 e suporte >= 0.3, se obtém:

```
RegrasFinais_filtradas = RegrasFinais[
    (RegrasFinais['confianca'] >= 0.8) &
    (RegrasFinais['suporte'] >= 0.3)
]
RegrasFinais_filtradas
```

	Antecedente	Consequente	suporte	confianca	lift
0	[Cafe]	[Manteiga]	0.3	1.0	2.0
2	[Cafe]	[Pao]	0.3	1.0	2.0
4	[Manteiga]	[Pao]	0.4	0.8	1.6
5	[Pao]	[Manteiga]	0.4	0.8	1.6
6	[Cafe]	[Pao, Manteiga]	0.3	1.0	2.5
9	[Manteiga, Cafe]	[Pao]	0.3	1.0	2.0
10	[Pao, Cafe]	[Manteiga]	0.3	1.0	2.0

Questão 07

Códigos disponíveis em:
https://colab.research.google.com/drive/1jAsZQ7dZD-TbwmXZpCes7or7h_DDvIyi?usp=sharing

Alterações necessárias:

1) Se Alterou a linha de código responsável por retirar do conjunto de regras as quais associavam NaN, ou seja, faziam a associação de negação com demais atributos, veja abaixo:

   # MODIFICAÇÃO AQUI: Remover apenas regras vazias, mas manter as com 'nan'
   # if 'nan' in a or 'nan' in b: continue

Sendo portanto as regras com ausência de produto:

REGRAS COM AUSÊNCIA DE PRODUTOS:
==========

Antecedente_Traduzido	Consequente_Traduzido	suporte	confianca	lift
COMPRA Cafe	NÃO COMPRA Manteiga	0.3	1.00	2.0
COMPRA Manteiga	NÃO COMPRA Cafe	0.3	0.60	2.0
NÃO COMPRA Cafe	COMPRA Manteiga	0.3	1.00	2.0
NÃO COMPRA Manteiga	COMPRA Cafe	0.3	0.60	2.0
COMPRA Cafe	NÃO COMPRA Pao	0.3	1.00	2.0
COMPRA Pao	NÃO COMPRA Cafe	0.3	0.60	2.0
NÃO COMPRA Cafe	COMPRA Pao	0.3	1.00	2.0
NÃO COMPRA Pao	COMPRA Cafe	0.3	0.60	2.0
COMPRA Manteiga	NÃO COMPRA Pao	0.4	0.80	1.6
COMPRA Pao	NÃO COMPRA Manteiga	0.4	0.80	1.6
NÃO COMPRA Manteiga	COMPRA Pao	0.4	0.80	1.6
NÃO COMPRA Pao	COMPRA Manteiga	0.4	0.80	1.6
COMPRA Cafe	NÃO COMPRA Pao	0.3	1.00	2.5
COMPRA Manteiga	NÃO COMPRA Cafe	0.3	0.60	2.0
COMPRA Pao	NÃO COMPRA Cafe	0.3	0.60	2.0
COMPRA Cafe, Manteiga	NÃO COMPRA Pao	0.3	1.00	2.0
COMPRA Cafe, Pao	NÃO COMPRA Manteiga	0.3	1.00	2.0
NÃO COMPRA Cafe	COMPRA Pao, Manteiga	0.3	1.00	2.5
COMPRA Pao, Manteiga	NÃO COMPRA Cafe	0.3	0.75	2.5
NÃO COMPRA Manteiga	COMPRA Cafe, Pao	0.3	0.60	2.0
NÃO COMPRA Pao	COMPRA Cafe, Manteiga	0.3	0.60	2.0
NÃO COMPRA Cafe	COMPRA Pao	0.3	1.00	2.0
NÃO COMPRA Cafe	COMPRA Manteiga	0.3	1.00	2.0
NÃO COMPRA Pao	COMPRA Cafe	0.3	0.75	2.5

Logo o conjunto de todas as regras possíveis sera:

index	Antecedente	Consequente	suporte	confianca	lift	Antecedente_Traduzido	Consequente_Traduzido
0	Cafe	Manteiga	0.3	1.0	2.0	COMPRA Cafe	COMPRA Manteiga
2	Cafe	Pao	0.3	1.0	2.0	COMPRA Cafe	COMPRA Pao
4	Cafe	nan	0.3	1.0	1.0	COMPRA Cafe	ITENS AUSENTES
5	Manteiga	Pao	0.4	0.8	1.6	COMPRA Manteiga	COMPRA Pao
6	Pao	Manteiga	0.4	0.8	1.6	COMPRA Pao	COMPRA Manteiga
7	Manteiga	nan	0.5	1.0	1.0	COMPRA Manteiga	ITENS AUSENTES
8	Pao	nan	0.5	1.0	1.0	COMPRA Pao	ITENS AUSENTES
9	Cafe	Pao,Manteiga	0.3	1.0	2.5	COMPRA Cafe	COMPRA Pao, Manteiga
12	Cafe,Manteiga	Pao	0.3	1.0	2.0	COMPRA Cafe, Manteiga	COMPRA Pao
13	Cafe,Pao	Manteiga	0.3	1.0	2.0	COMPRA Cafe, Pao	COMPRA Manteiga
15	Cafe	Manteiga,nan	0.3	1.0	2.0	COMPRA Cafe	NÃO COMPRA Manteiga
17	Cafe,Manteiga	nan	0.3	1.0	1.0	COMPRA Cafe, Manteiga	ITENS AUSENTES
18	Cafe,nan	Manteiga	0.3	1.0	2.0	NÃO COMPRA Cafe	COMPRA Manteiga
20	Cafe	Pao,nan	0.3	1.0	2.0	COMPRA Cafe	NÃO COMPRA Pao
22	Cafe,Pao	nan	0.3	1.0	1.0	COMPRA Cafe, Pao	ITENS AUSENTES
23	Cafe,nan	Pao	0.3	1.0	2.0	NÃO COMPRA Cafe	COMPRA Pao
25	Manteiga	Pao,nan	0.4	0.8	1.6	COMPRA Manteiga	NÃO COMPRA Pao
26	Pao	Manteiga,nan	0.4	0.8	1.6	COMPRA Pao	NÃO COMPRA Manteiga

index	Antecedente	Consequente	suporte	confianca	lift	Antecedente_Traduzido	Consequente_Traduzido
27	Pao,Manteiga	nan	0.4	1.0	1.0	COMPRA Pao, Manteiga	ITENS AUSENTES
28	Manteiga,nan	Pao	0.4	0.8	1.6	NÃO COMPRA Manteiga	COMPRA Pao
29	Pao,nan	Manteiga	0.4	0.8	1.6	NÃO COMPRA Pao	COMPRA Manteiga
30	Cafe	Pao,Manteiga,nan	0.3	1.0	2.5	COMPRA Cafe	NÃO COMPRA Pao
33	Cafe,Manteiga	Pao,nan	0.3	1.0	2.0	COMPRA Cafe, Manteiga	NÃO COMPRA Pao
34	Cafe,Pao	Manteiga,nan	0.3	1.0	2.0	COMPRA Cafe, Pao	NÃO COMPRA Manteiga
35	Cafe,nan	Pao,Manteiga	0.3	1.0	2.5	NÃO COMPRA Cafe	COMPRA Pao, Manteiga
39	Cafe,Pao,Manteiga	nan	0.3	1.0	1.0	COMPRA Cafe, Pao, Manteiga	ITENS AUSENTES
40	Cafe,Manteiga,nan	Pao	0.3	1.0	2.0	NÃO COMPRA Cafe	COMPRA Pao
41	Cafe,Pao,nan	Manteiga	0.3	1.0	2.0	NÃO COMPRA Cafe	COMPRA Manteiga