Mini-teste 3 **Vitor Peixoto - A79175**
Mineração de Dados 26/11/2018
Mestrado em Engenharia Informática Universidade do Minho

Exercício 1

O teste exato de *Fisher* é usado para estimar a qualidade das regras na medida em que permite definir regras redundantes, levando ao aparecimento de regras de associação com um número mínimo de atributos (regras mais generalizadas).

No exercício dado, um ramo da árvore é descrito pela seguinte regra:

$$X = v1 \quad \& \quad Y = a \quad \& \quad Age < 55 \implies classe = c1$$

Que, após aplicado um algoritmo de regras de associação, se conseguiu derivar a seguinte regra em conjunto:

$$X = v1 \quad \& \quad Y = a \implies classe = c1$$

O teste de *Fisher* entre a primeira e segunda regra origina um *p-value*=0.078155.

Segundo o guião do *Caren*, este implementa uma metodologia usando o teste exato de *Fisher* entre uma regra e as suas generalizações imediatas. Se obtivermos um *p-value* superior a *alpha* então a regra é descartada. Este teste avalia então a significância da melhoria da confiança de uma regra em relação à confiança das suas generalizações imediatas. A problemática do teste de *Fisher* resume-se assim à resposta para a seguinte questão: *Novos itens adicionados a uma regra para obter uma versão especializada também contribuem para a sua confiança?*

Sendo assim, temos de avaliar o *p-value* obtido pelo teste de *Fisher* que foi 0.078155. Este valor verifica-se maior que o valor de *alpha* (grau de significância) que tradicionalmente é 0.05. Assim sendo, a regra mais específica é descartada pela sua generalização, ou seja, o atributo Age<55 acaba por ser redundante e pouco significativo para a confiança da regra.

Respondendo à pergunta do enunciado, sugere-se assim que se efetue uma troca da árvore de decisão relativamente à regra do ramo pela sua generalização uma vez que a especificação era redundante.

Exercício 2

Este exercício pedia regras redundantes, produtivas e significativas para o *dataset student_courses.bas*.

Em primeiro lugar é importante traçar a definição do que são regras redundantes, produtivas e significativas.

- **Regras redundantes:** Uma regra é redundante se a sua generalização tem o mesmo valor de suporte.
- **Regras produtivas:** Uma regra é produtiva se a sua melhoria (*improvement*) for positiva, ou seja, uma regra mais especifica produz uma mais valia em termos de valor de medida de interesse.
- **Regras significativas:** Uma regra é significativa se é estatísticamente significante relativamente às suas generalizações. As regras significativas podem ser filtradas no *CareN* através do switch *-fisher*.

Utilizando o *CareN* podemos obter todas as regras com o seguinte comando:

```
$ java caren student_courses.bas 0.1 0.5 -s, -d
```

Facilmente encontrámos algumas regras redundantes, como as seguintes:

```
Sup = 0.10900  Conf = 1.00000   GEST400    <--    CC422  &  CC432  &  CC412  &  DIP461
Sup = 0.10900  Conf = 1.00000   GEST400    <--    CC422  &  CC432  &  CC412  &  DIP461  &  DIP463
```

De facto, a regra de baixo é redundante visto apresentar mais um atributo (maior complexidade) sem tal se traduzir no valor de suporte, uma vez que é igual ao da sua generalização.

Os regras produtivas são aquelas cuja melhoria é positiva. O *CareN* permite filtrar as regras produtivas com o switch *-imp*:

```
$ java caren student_courses.bas 0.1 0.5 -s, -d -imp0.0001
```

O *-imp* indica um valor mínimo para o valor de melhoria (*improvement*), neste caso, queríamos que a melhoria fosse maior que 0, logo escolhemos 0.0001 como valor mínimo para a melhoria. Este comando reduziu o número de regras de 36.607 para 3.402, um decréscimo bastante significativo.

Para obter agora apenas as regras significativas, utilizamos então o switch *-fisher* como já aqui referido:

```
$ java caren student_courses.bas 0.1 0.5 -s, -fisher -d
```

Este switch usa o valor de *alpha* por defeito a 0.05 para rejeição da hipótese nula. De facto, utilizando o teste de *Fisher*, obtivemos 671 regras, um número também muito inferior ao número de regras produtivas, visto termos filtrado apenas as regras estatísticamente significantes relativamente às suas generalizações.

Exercício 3

A complexidade de um algoritmo de geração de regras de associação pode ser afetada pelos seguintes fatores:

- **Valor mínimo de suporte:** Diminuir o valor mínimo de suporte frequentemente resulta em mais itens frequentes, isto implica que sejam gerados e contabilizados, aumentando a complexidade computacional do algoritmo.
- ***Datasets* densos:** Para *datasets* densos, o comprimento das transações pode ser grande, afetando o número máximo de itens frequentes, que, como já vimos no ponto anterior, aumenta a complexidade do algoritmo.

Há mais fatores que podem influenciar a complexidade computacional deste tipo de algoritmos, mas estes dois são os mais influentes, principalmente o valor mínimo de suporte onde efetuamos também alguns testes: Corremos o algoritmo com um valor de *minsup*=0.1 onde obtivemos um tempo de 1,901 segundos, depois com *minsup*=0.05 com o tempo de 3,897 segundos e finalmente com *minsup*=0.01 com o tempo de 26,639 segundos. Através dos tempos obtidos conseguimos observar claramente a influência deste *threshold* no tempo de execução do algoritmo.