PS
Perceção Sensorial

Sessão prática – Árvores de decisão

Objetivo: familiarizar os alunos com as técnicas de indução de árvores de decisão apresentadas nas aulas teóricas

1 – Faça o download do exemplo "German Credit Risk" a partir do portal do Kagle (https://www.kaggle.com/uciml/german-credit#german_credit_data.csv) ou a partir da página Moddle de PS. Deverá obter um ficheiro CSV que deverá colocar no folder em que pretende trabalhar.

2 – Abra o ficheiro de dados com a biblioteca Pandas criando um dataframe com os dados.

Sugestão: utilizar a função pd.read_csv

3 - Visualize os dados assim obtidos. Como poderá constatar tem mil linhas com 10 colunas. Uma vez que a primeira coluna não tem qualquer significado apague-a. A última coluna (Purpose) é a variável que se pretende estimar.

Sugestão: *data.drop(data.columns[0],1,inplace=True)*

4 – Verifique o número de classes e o número de valores em cada classe existentes nos dados. Mantenha a classe dos exemplos *radio/TV* e *car* e transforme os restantes valores numa nova classe identificada por *Other*. Deverá ficar com três classes contendo: Other-383, car-337 e radio/TV-280 valores.

Sugestão: para ver o número de exemplos de cada classe - *data['Purpose'].value_counts()*

5 – Uma vez que trabalhar com 1000 valores é demasiado complicado, especialmente para confirmar resultados, selecione apenas os primeiros 10 exemplos e trabalhe apenas com eles.

6 – Uma vez que este problema apresenta valores contínuos e valores discretos construa um dicionário que traduza o nume de cada coluna na categoria respetiva ('Cont' ou 'Disc').

Sugestão: dataDesc={'Age':'Cont', ... }

7 – Como poderá constatar, os dados apresentam *missing values* em duas colunas (*Saving accounts* e *Checking account*). Faça imputação desses valores em falta utilizando o critério de *valor médio dentro da classe* para atributos contínuos e *valor maioritário dentro da classe* para valores discretos.

Sugestão: se estiver atrasado utilize o ficheiro *inputedData.csv* que contém os 10 primeiros exemplos com os valores em falta já imputados. Se tiver (ou quando tiver) possibilidade de fazer esta alínea utilize os valores do ficheiro para verificar os seus valores. Se possível desenvolva código que permita imputar qualquer *dataframe* com qualquer tipo de valores (discretos ou contínuos) detetando automaticamente quais as colunas com *missing values* e fazendo a sua imputação tendo em conta o tipo de valores em causa.

df[c].isnull().values.any() – indica se existem valores em falta na coluna c da dataframe df

PS
Perceção Sensorial

8 – Pretende-se agora construir uma função capaz de criar uma árvore de decisão utilizando o critério da Entropia ou o critério de Gini e fazendo as partições por pesquisa exaustiva no caso dos atributos contínuos ou por características ordenadas no caso dos atributos discretos.

a) Crie uma função que dada uma lista de probabilidades de classes calcule o valor de impuridade correspondente de acordo com o critério de Gini. Teste a sua função com o vetor [0.25,0.25,0.25,0.25] que corresponde a quatro classes equiprováveis e cujo resultado deverá ser 0.75.
b) Repita a alínea anterior criando agora uma função para o critério da Entropia. O resultado com o mesmo vetor de teste deverá ser 2.
c) Crie uma função com o seguinte cabeçalho abaixo indicado. Os três parâmetros são o *dataframe* a processar, as variáveis do dataframe que se pretende utilizar como *predictors*, a identificação da coluna que contém a classe a prever e a função de avaliação de impuridade que se pretende utilizar.

def CalcDT(df,vars,classe,ef):

Uma possível chamada a esta função para incluir na análise todas as colunas de variáveis independentes será:

DT=CalcDT(data, data.columns.drop('Purpose'),'Purpose',Entropy)

d) Comece por chamar a função anterior passando apenas uma variável na lista de variáveis para simplificar. Inicie o desenvolvimento da função calculando os níveis de decisão da variável passada como parâmetro tendo em conta o tipo de variável (discreta ou contínua).
Para a variável *Age*, por exemplo, deverá ter os valores 25 – 31.5 – 40 - 47 – 51 – 57 – 64.
Para a variável *Housing* deverá ter os valores *own* e *rent*.
Note que os testes que se está a pensar utilizar serão do tipo <x | >=x

e) Calcule o valor de impuridade de cada um dos testes anteriores e escolha o teste que apresente a menor impuridade resultante.

f) Inclua agora todas as variáveis disponíveis (chamando a função com a lista completa de varáveis independentes) e calcule qual o melhor teste (qual a variável e o valor de teste melhor) para efetuar a partição do conjunto de dados.

g) Aplique recursivamente a função com cada um dos sub-conjuntos de dados resultantes da melhor partição até que a impuridade de entrada seja zero (e nesse caso temos uma folha com o valor da classe existente no conjunto de dados em causa). Com os resultados destas chamadas recursivas construa uma lista de testes em árvore que represente a árvore de decisão final. Deverá obter algo do género:

```
In [396]: DT

Out[396]: ['Age',
           57.0,
           ['Checking account',
            'moderate',
            ['Age',
             51.0,
             ['NODE', 'Other'],
             ['Housing', 'own', ['NODE', 'car'], ['NODE', 'Other']]],
            ['Age', 25.0, ['NODE', 'radio/TV'], ['NODE', 'car']]],
           ['NODE', 'radio/TV']]
```

Nesta descrição da árvore poderá incluir mais ou menos informação de acordo com o seu gosto pessoal.

PS
Perceção Sensorial

9 – Experimente agora a biblioteca *sklearn* que permite gerar uma árvore de decisão de uma forma muito simples. Para tal terá que transformar todas as variáveis discretas em variáveis numéricas para o que se recomenda a utilização de um dicionário do tipo:

tt={'male':0, 'female':1, 'own':0, 'free':1, 'rent':2, 'little':0, 'quite rich':1, 'rich':2, 'moderate':1, 'radio/TV':0, 'Other':1, 'car':2}

Para gerar a árvore de decisão do *sklearn* basta fazer:

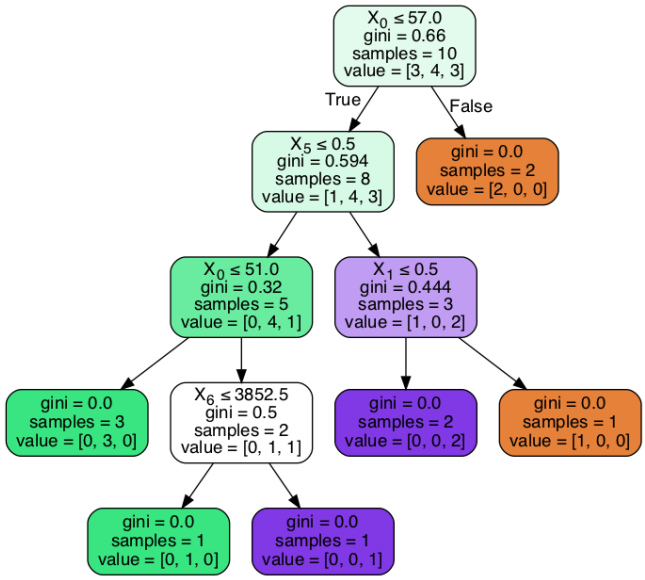
```
from sklearn.tree import DecisionTreeClassifier as dtClass
from sklearn.model_selection import cross_val_score
…
clf=dtClass(random_state=0)
clf.fit(GCdata, GCTarget)    # GCdata – predictors  GCTarget – class
```

10 – Uma vez obtida a árvore de decisão visualize-a utilizando o código que se apresenta seguidamente (ou outro qualquer). Compare a árvore de decisão gerada pelo *sklearn* com a árvore de decisão gerada pelo código que desenvolveu nas alíneas anteriores.

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
import pydotplus
from IPython.display import Image

dot_data = StringIO()

export_graphviz(clf, out_file=dot_data, filled=True, rounded=True, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())    # eventually you need to do: conda install python-graphviz
```



Árvore gerada pelo *sklearn* depois da transformação de variáveis mencionada no ponto 9.