Aprendizagem Automática 2020/2021

Trabalho 1

Pretende-se com este trabalho implementar em Python 3 uma classe para gerar árvores de decisão com "pruning". O trabalho pode ser realizado em grupos de dois e deve ser submetido através moodle **até dia 21 de dezembro.**

Até ao fim do prazo deve submeter um ficheiro .zip, em que o nome do ficheiro tem os números dos alunos (e.g. "44444_33333.zip") e o conteúdo é o código fonte do trabalho, adequadamente comentado, e um PDF com um relatório sucinto com a análise dos conjuntos de dados fornecidos (deve comparar o uso de árvores de decisão com e sem pruning, e usando diversas funções de homogeneidade/pureza).

A classe deverá ser parcialmente compatível com as classes dos classificadores do sklearn, e implementar uma árvore de decisão de acordo com o algoritmo apresentado nos slides das aulas teóricas (algoritmo ID3) seguida de pruning com o método REP (Reduced Error Pruning).

A classe deverá cumprir os seguintes requisitos:

- ter uma parametrização para decidir a medida de pureza (opção entre gini, entropia, e erro) e indicação se terá ou não pruning, e.g. ***DecisionTreeREPrune(criterion='gini', prune=True)*;**

- implementar o método ***fit(x,y)*** que gera uma árvore de decisão em função dos dados de treino x e y, em que X é um array bidimensional com conjunto de dados para o treino com a dimensão *(n_samples, n_features) e* y é um array unidimensional com as classes de cada exemplo, com a dimensão *(n_samples);*

- no método ***fit,*** a geração da árvore deverá ser feita com 75% dos dados de treino e o pruning (Reduced Error Pruning) deverá ser feito com 25%. Poderão, para este efeito, usar a função de *train_test_split()* do sklearn.

- implementar um método ***score(x, y)*** que devolve o valor da exatidão (accuracy) com o conjunto de dados de teste x e y.

- considere que todos os atributos são nominais (e se trata de um problema de classificação).

Note que para a análise dos resultados com os diversos conjuntos de dados disponibilizados (weather.nominal.csv; contact-lenses.csv; soybean.csv; e vote.csv) deve efectuar sempre uma divisão entre treino e teste, usando por exemplo a função *train_test_split()* do sklearn.

Para carregar e testar os dados pode usar algo como ...

```
data=numpy.genfromtxt("weather.nominal.csv", delimiter=",", dtype=None, encoding=None)
xdata=data[1:,0:-1]          #  dados: da segunda à ultima linha, da primeira à penúltima coluna
ydata=data[1:,-1]            # classe: da segunda à ultima linha, só última coluna
x_train, x_test, y_train, y_test = train_test_split(xdata, ydata, random_state=0)

classifier = myDecisionTreeREPrune()
classifier.fit(x_train, y_train)
result = classifier.score(x_test, y_test)
print("Percentagem de casos corretamente classificados {:2.2%}".format(result))
```