Aprendizagem de Máquina
Prof. Tiago Buarque A. de Carvalho

Exercícios sobre Árvores de Decisão

1. (30 pontos) Utilizando a base Car Evaluation

   https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

   (a) Construa sua própria implementação do algoritmo de treinamento de uma árvore de decisão.

   (b) Realize um experimento do tipo 10-fold-cross-validation para avaliar a taxa de acerto da sua implementação.

```
def gerarArvore(no, x, y, atributosRestantes, erros = -1):
    no.predominancia = predominancia(y,["unacc", "acc", "good", "vgood"])

    if erros == -1:
        erros = calcularErros(y,["unacc", "acc", "good", "vgood"])

    print("Erro:",erros)
    restantes = []
    for u in atributosRestantes:
        restantes.append(u)
    print("Restantes:",restantes)
    atributos = [["vhigh", "high", "med", "low"],["vhigh", "high", "med", "low"],["2", "3", "4", "5more"],["2", "4", "more"],["small", "med", "big"],["low", "med", "h

    XsYs = [ [] for k in range(len(atributos)) ]
    melhor = erros
    melhorIndex = -1

    for k in range(len(atributosRestantes)):
        atributoAtual = atributosRestantes[k]
        errosAtual = 0
        clas = ""
        print("Atributo:",atributoAtual)

        for i in range(len(atributos[atributoAtual])):
            valorAtual = atributos[atributoAtual][i]
            print("Valor:",valorAtual)
            x1, y1 = gerarElementos(x,y,atributoAtual,valorAtual)
            #print(x1[0],x1[1],y1[0],y1[1])
            erro = calcularErros(y1,["unacc", "acc", "good", "vgood"])
            errosAtual += erro
            XsYs[atributoAtual].append([x1,y1,valorAtual,erro])


        print("Erros atual final:",errosAtual)
        if errosAtual < melhor:
            melhorIndex = atributoAtual
            melhor = errosAtual

    #Checar melhor e gerarArvore recursivamente. e gerar filhos
    print(melhorIndex,melhor)
    if melhorIndex != -1:
        filhos = XsYs[melhorIndex]
        restantes.remove(melhorIndex)
        for f in range(len(filhos)):
            atual = filhos[f]
            novoNo = No(melhorIndex,atual[2])
            print(novoNo.atributo,novoNo.valor,novoNo.filhos)
            print(no.filhos)
            no.filhos.append(novoNo)
            gerarArvore(novoNo,atual[0],atual[1],restantes,atual[3])


    return no
```

```
[602, 186, 39, 37]
Erro: 262
Restantes: [0, 1, 2, 3, 4, 5]
Atributo: 0
Valor: vhigh
[178, 35, 0, 0]
Valor: high
[166, 51, 0, 0]
Valor: med
[133, 63, 16, 13]
Valor: low
[125, 37, 23, 24]
Erros atual final: 262
Atributo: 1
Valor: vhigh
[173, 41, 0, 0]
Valor: high
[165, 50, 0, 9]
Valor: med
[130, 49, 11, 11]
Valor: low
[134, 46, 28, 17]
Erros atual final: 262
Atributo: 2
Valor: 2
[164, 36, 8, 6]
Valor: 3
[136, 52, 10, 7]
Valor: 4
[155, 43, 10, 11]
Valor: 5more
[147, 55, 11, 13]
Erros atual final: 262
Atributo: 3
Valor: 2
[280, 0, 0, 0]
Valor: 4
[160, 98, 20, 22]
Valor: more
[162, 88, 19, 15]
Erros atual final: 262
Atributo: 4
Valor: small
[213, 45, 11, 0]
Valor: med
[197, 67, 13, 13]
Valor: big
[192, 74, 15, 24]
Erros atual final: 262
Atributo: 5
Valor: low
[292, 0, 0, 0]
Valor: med
[178, 90, 24, 0]
Valor: high
[132, 96, 15, 37]
Erros atual final: 262
-1 262
unacc
[]
```

```
Holdout 1 -> Taxa de Acerto = 71.41203703703704 %
Holdout 2 -> Taxa de Acerto = 68.63425925925925 %
Holdout 3 -> Taxa de Acerto = 70.02314814814815 %
Holdout 4 -> Taxa de Acerto = 68.75 %
Holdout 5 -> Taxa de Acerto = 70.37037037037037 %
Holdout 6 -> Taxa de Acerto = 69.56018518518519 %
Holdout 7 -> Taxa de Acerto = 71.41203703703704 %
Holdout 8 -> Taxa de Acerto = 70.02314814814815 %
Holdout 9 -> Taxa de Acerto = 70.94907407407408 %
Holdout 10 -> Taxa de Acerto = 70.37037037037037 %

Media = 70.15046296296296 %
```

2. (10 pontos) Compare os resultados da implementação com os resultados da implementação de uma biblioteca de árvores de decisão.

```python
lenc = preprocessing.LabelEncoder()
#itens = itens.apply(lenc.fit_transform)
itens = itens.apply(lenc.fit_transform)

#print(itens[0:10])
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(itens, classificacao, test_size=0.5)

clf = tree.DecisionTreeClassifier()
clf.fit(X_train,y_train)
predicts = clf.predict(X_test)
y_test = y_test.tolist()

erros = 0
acertos = 0
for k in range(len(X_test)):
    pred = predicts[k]
    if pred == y_test[k]:
        acertos += 1
    else:
        erros += 1

total = len(y_test)

taxa = (acertos/total)*100
```

```
Teste 1
ACERTOS: 836
ERROS: 28
Taxa de Acerto: 96.75925925925925 %

Teste 2
ACERTOS: 840
ERROS: 24
Taxa de Acerto: 97.22222222222221 %

Teste 3
ACERTOS: 833
ERROS: 31
Taxa de Acerto: 96.41203703703704 %

Teste 4
ACERTOS: 842
ERROS: 22
Taxa de Acerto: 97.45370370370371 %

Teste 5
ACERTOS: 834
ERROS: 30
Taxa de Acerto: 96.52777777777779 %

Teste 6
ACERTOS: 830
ERROS: 34
Taxa de Acerto: 96.06481481481481 %

Teste 7
ACERTOS: 822
ERROS: 42
Taxa de Acerto: 95.13888888888889 %

Teste 8
ACERTOS: 826
ERROS: 38
Taxa de Acerto: 95.60185185185185 %

Teste 9
ACERTOS: 829
ERROS: 35
Taxa de Acerto: 95.94907407407408 %

Teste 10
ACERTOS: 839
ERROS: 25
Taxa de Acerto: 97.10648148148148 %

----------------------------------
Media: 96.42361111111111 %
```

3. (30 pontos) Utilizando a base Car Evalution e um Holdout 50/50. Teste pelo menos três valores distintos de número mínimo de exemplos por folha e realize os experimentos PAREADOS. Avalie os efeitos de limitar o número mínimo de exemplos por folha:

   (a) Na taxa de acerto para o conjunto de treino.

   (b) Na taxa de acerto para o conjunto de teste.

   (c) Na matriz de confusão.

   (d) No número de folhas da árvore final.

(e) No número de nós da árvore.

```
Teste 1
MIN LEAF 1
Folhas = 68
ACERTOS: 819
ERROS: 45
Taxa de Acerto: 94.79166666666666 %
Matriz:
 ['unacc', 'acc', 'good', 'vgood']
[[608   9   0   0]
 [ 22 153   2   3]
 [  3   6  30   0]
 [  0   0   0  28]]

MIN LEAF 10
Folhas = 30
ACERTOS: 764
ERROS: 100
Taxa de Acerto: 88.42592592592592 %
Matriz:
 ['unacc', 'acc', 'good', 'vgood']
[[580  35   2   0]
 [ 29 142   8   1]
 [  0  10  23   6]
 [  0   9   0  19]]

MIN LEAF 30
Folhas = 13
ACERTOS: 702
ERROS: 162
Taxa de Acerto: 81.25 %
Matriz:
 ['unacc', 'acc', 'good', 'vgood']
[[553  64   0   0]
 [ 38 130   0  12]
 [  0  39   0   0]
 [  0   9   0  19]]
```

```
Teste 30
MIN LEAF 1
Folhas = 69
ACERTOS: 829
ERROS: 35
Taxa de Acerto: 95.94907407407408 %
Matriz:
 ['unacc', 'acc', 'good', 'vgood']
[[597   6   3   0]
 [ 17 171   3   0]
 [  0   1  34   0]
 [  0   5   0  27]]

MIN LEAF 10
Folhas = 29
ACERTOS: 757
ERROS: 107
Taxa de Acerto: 87.61574074074075 %
Matriz:
 ['unacc', 'acc', 'good', 'vgood']
[[554  49   3   0]
 [ 22 159   7   3]
 [  0   9  22   4]
 [  0  10   0  22]]

MIN LEAF 30
Folhas = 13
ACERTOS: 706
ERROS: 158
Taxa de Acerto: 81.71296296296296 %
Matriz:
 ['unacc', 'acc', 'good', 'vgood']
[[566  40   0   0]
 [ 59 120   0  12]
 [  0  35   0   0]
 [  0  12   0  20]]
```

```
Teste
Media 1: 96.40046296296298 %
Media 10: 89.8341049382716 %
Media 30: 82.03703703703704 %

Treino
Media 1: 100.0 %
Media 10: 92.61574074074078 %
Media 30: 84.0895061728395 %
```

4. (30 pontos) Faça o mesmo da questão anterior para a base Breast Cancer. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer

```
Teste 1
MIN LEAF 1
Folhas = 55
ACERTOS: 99
ERROS: 44
Taxa de Acerto: 69.23076923076923 %
Matriz:
 ['no-recurrence-events', 'recurrence-events']
[[80 22]
 [22 19]]

MIN LEAF 10
Folhas = 10
ACERTOS: 108
ERROS: 35
Taxa de Acerto: 75.52447552447552 %
Matriz:
 ['no-recurrence-events', 'recurrence-events']
[[98  4]
 [31 10]]

MIN LEAF 30
Folhas = 3
ACERTOS: 104
ERROS: 39
Taxa de Acerto: 72.72727272727273 %
Matriz:
 ['no-recurrence-events', 'recurrence-events']
[[83 19]
 [20 21]]
```

```
Teste 30
MIN LEAF 1
Folhas = 45
ACERTOS: 92
ERROS: 51
Taxa de Acerto: 64.33566433566433 %
Matriz:
 ['no-recurrence-events', 'recurrence-events']
[[76 26]
 [25 16]]

MIN LEAF 10
Folhas = 9
ACERTOS: 96
ERROS: 47
Taxa de Acerto: 67.13286713286713 %
Matriz:
 ['no-recurrence-events', 'recurrence-events']
[[89 13]
 [34  7]]

MIN LEAF 30
Folhas = 4
ACERTOS: 98
ERROS: 45
Taxa de Acerto: 68.53146853146853 %
Matriz:
 ['no-recurrence-events', 'recurrence-events']
[[75 27]
 [18 23]]
```

```
Teste
Media 1: 62.79720279720281 %
Media 10: 72.14452214452216 %
Media 30: 69.51048951048955 %

Treino
Media 1: 98.7878787878788 %
Media 10: 77.64568764568763 %
Media 30: 73.4965034965035 %
```