USP - EACH

SIN5007 - Reconhecimento de Padrões

Grupo 8-H

Atividade 05 - Função para Validação Cruzada

Abril de 2020

- A função implementada recebe três parâmetros: **data**: um conjunto de dados (no caso, a database composta pelos vetores de características); **target**: um conjunto com as classes de cada amostra de data; e, **k**: a quantidade de conjuntos (folds) em que se deseja dividir a database.
- Optamos por incluir o parâmetro **target**, pois nossa database possui, originalmente, três classes (SAN - Sem Anomalia; CMH - Cardiomiopatia Hipertrófica; e, CMD - Cardiomiopatia Dilatada). Dessa forma, poderemos generalizar a função futuramente para trabalhar com todas as classes.
- A função separa a database em k conjuntos de treinamento/teste.
- Em cada conjunto se respeita ao máximo possível a proporção das classes em referência a database original.
- Em cada conjunto, a quantidade de amostras entre as classes poderão ter, no máximo, 1 elemento a mais ou a menos em relação a outro conjunto.
- Ao final, os dados de cada conjunto são impressos, seguindo o seguinte exemplo/modelo:

k = 3, Dataset: 80 positivas e 40 negativas (66% x 33%)
Fold 1: Pos: 27, Neg: 14, Total: 41, Proporção: 66%; 33%
Fold 2: Pos: 27, Neg: 13, Total: 40, Proporção: 67%; 32%
Fold 3: Pos: 26, Neg: 13, Total: 39, Proporção: 66%; 33%

In [41]:

```
# Bibliotecas
import numpy as np
import pandas as pd
import random
import math
import matplotlib.pyplot as plt

from sklearn.model_selection import KFold
from sklearn.model_selection import GroupKFold
from sklearn.model_selection import StratifiedKFold
```

In [42]:

```
# Leitura do arquivo de entrada

df = pd.read_csv('CMCT_20200503.csv', )
```

In [43]:

```
# Seleção somente dos TARGET (Classes) 0 - Sem Anomalia e 2 - Cardiomiopatia Dilata
dfx = df.copy()

masc = dfx['TARGET'] != 1
dfy = dfx.loc[masc, :]

X = dfy.drop(['TARGET'], axis=1)
Y = dfy['TARGET']
```

In [44]:

```
##############################################
#### Separa uma database em k conjuntos ####
##############################################
def splitFolds(data, target, k=10):

    # Contadores para apresentação
    ldata = len(data)              # Quantidade de linhas da base
    numel = int(ldata / k)         # Quantidade de amostras por fold

    uclass = target.value_counts() # Classes e suas quantidades
    uclass.sort_index(inplace=True)
    nclass = uclass.index          # Classes
    qclass = uclass.values         # Quantidade de cada classe

    # Junção das classes e suas quantidades para um dicionário
    zclass = zip(nclass, qclass)
    dclass = dict(zclass)

    # Separação dos conjuntos
    partesK = []    # Conterá todos os conjuntos k de índices, cada um proporcional
                    #a cada classe
    for i in range(k):
        pk = []
        if (i < 7):
            nelem = numel + 1
        else:
            nelem = numel

        ntot = nelem

        for nc, qc in dclass.items():
            # Captura de todos os índices da coluna target
            masc = target == nc
            idclass = list(target[masc].index)

            # Montagem dos k conjuntos com a proporção o mais próxima possível da
            #database completa
            propclass = int(round(nelem * qc / ldata))
            if (ntot > propclass):
                ntot -= propclass
            else:
                propclass = ntot

            rs = random.sample(idclass, propclass)
            pk = pk + rs

        partesK.append(pk)

    return (partesK)
```

In [45]:

```
##########################################
#### Execução e impressão do relatório ####
##########################################

k = 10
partesK = splitFolds(X, Y, k)

qtdDS = len(X)
vc = Y.value_counts()
qtdT0 = vc.values[1]
qtdT2 = vc.values[0]
perT0 = round(qtdT0/qtdDS * 100, 2)
perT2 = round(qtdT2/qtdDS * 100, 2)

print('**** Dataset: CMCT_20200503.csv ****')
print('k =', k, ', Dataset:', qtdT0, 'SAN e', qtdT2, 'CMD (', perT0, '%  x ',
      perT2, '% )')
print()

qtd0 = 0
qtd2 = 0
per0 = 0.0
per2 = 0.0

for f in range(len(partesK)):
    dx = Y[partesK[f]]
    vc = dx.value_counts()
    qtd0 = vc.values[1]
    qtd2 = vc.values[0]
    per0 = round(qtd0/(qtd0 + qtd2) * 100, 2)
    per2 = round(qtd2/(qtd0 + qtd2) * 100, 2)

    print(f'Fold {f+1}: SAN: {qtd0}, CMD: {qtd2}, Total: {qtd0+qtd2},',
          f'Proporção: {per0}%; {per2}%')
```

```
**** Dataset: CMCT_20200503.csv ****
k = 10 , Dataset: 101 SAN e 116 CMD ( 46.54 %  x  53.46 % )

Fold 1: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 2: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 3: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 4: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 5: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 6: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 7: SAN: 10, CMD: 12, Total: 22, Proporção: 45.45%; 54.55%
Fold 8: SAN: 10, CMD: 11, Total: 21, Proporção: 47.62%; 52.38%
Fold 9: SAN: 10, CMD: 11, Total: 21, Proporção: 47.62%; 52.38%
Fold 10: SAN: 10, CMD: 11, Total: 21, Proporção: 47.62%; 52.38%
```

In [ ]:

In [ ]:

In [ ]:

```

```