
# Centro de Informática
# Disciplina: IF1015 - Ciência dos Dados
# Lista de exercícios - 1

1) Considerem a base de dados Câncer de Pulmão e a base de dados Íris da UCI. Vocês irão construir operações de **pré-processamento** sobre essas bases, considerando que elas tiveram algum erro durante a coleta dos dados (os arquivos ruidosos estão na pasta desse exercício).
   Os ruídos presentes no dados são de dados faltantes (nos arquivos da base, o ruído está presente com um traço "-" ). Trate os ruídos dessas bases considerando os seguintes métodos de resolução:
   a) Coloque os valores como sendo o valor médio dos valores de um atributo faltante.
   b) Coloque os valores como sendo a moda dos valores de um atributo faltante.
   c) Coloque os valores como sendo um valor aleatório coerente com o valor do atributo faltante (maior ou igual ao valor mínimo existente ou menor ou igual ao valor máximo existente).
   d) Elimine as linhas que estejam com algum atributo faltante.

Considere também que talvez seja preciso transformar os atributos categóricos em atributos numéricos binários para usar em alguns classificadores. Faça isso considerando a possibilidade de o atributo ser categórico ordinal. Também verifique se os atributos são importantes (pode ser que tenha um atributo identificador que não é útil para a tarefa de classificação).

Para cada base pré-processada, gere um arquivo de texto e armazene.

Agora faça um programa em que cada uma dessas bases tratadas seja treinada por uma rede neural artificial (RNA) com uma camada escondida, e com 20 neurônios na camada escondida, com 70% dos objetos sendo usados para treinamento e 30% para teste e veja a taxa de classificação. Treine a RNA com o algoritmo de backpropagation. Compare com o resultado encontrado com a base de dados original.

Segue **um exemplo** de treinamento e teste usando a biblioteca sklearn de Python:

```
from sklearn.neural_network import MLPClassifier
```

```
X = [[0., 0.], [1., 1.]] #duas entradas
y = [0, 1] #valores esperados para cada uma das entradas
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(5, 2), random_state=1)

#TREINAMENTO
clf.fit(X, y)    #a funcao fit treina a rede neural com o conjunto de entrada X e a saida esperada
y
#TESTE
clf.predict([[2., 2.]]) #prediz qual a classe que pertence a entrada (2,2)
```

Fonte: *https://scikit-learn.org/stable/modules/neural_networks_supervised.html*

2) Use as bases de dados pré-processadas da questão 1 e rode uma versão simplificada do k-Means supervisionado: o Nearest Centroid Classifier. Este classificador pega as amostras de cada classe e gera seus respectivos centróides. Então, para classificar um objeto novo, verifica qual o centróide que está menos distante e rotula-o como sendo da classe desse centróide. Você pode usar as funções da biblioteca sklearn do Python. Segue **um exemplo** abaixo

```
from sklearn.neighbors.nearest_centroid import NearestCentroid
import numpy as np
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
y = np.array([1, 1, 1, 2, 2, 2])
clf = NearestCentroid()
#TREINAMENTO
clf.fit(X, y)
#TESTE
print(clf.predict([[-0.8, -1]]))
```

Fonte: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

Veja a taxa de acerto/erro para cada uma das bases de treinamento e veja o impacto das decisões tomadas nas etapas de pré-processamento.

3) Rode o kNN para as bases pré-processadas da primeira questão. Plote a taxa de erro de médio de classificação para diferentes valores de k.

Aqui está **um exemplo** de um k-NN para objetos de apenas um atributo.

```
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
from sklearn.neighbors import KNeighborsClassifier
```

```
neigh = KNeighborsClassifier(n_neighbors=3)
#TREINAMENTO
neigh.fit(X, y)

#TESTE
print(neigh.predict([[1.1]]))
```

Fonte: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

4) Repita os experimentos considerando agora o método de avaliação de classificador **k-FOLD**. Repita todos os experimentos para valores de **k** variando entre **k**=2 até o tamanho máximo possível de folders. Perceba que cada fold deve ter a mesma proporção de classes ou com o máximo de aproximação para isso, então esse valor máximo pode variar de uma base para outra.