Inteligência artificial aplicada

Arquitetura de dados

Trabalho da disciplina - Questão 2

Nomes: Paulo Sergio Herval Silva Junior, Pedro de Sousa Alves Graça, Matheus Eduardo de Arazão e Anderson Felipe de Paiva

Carregando dados

Dataset: https://archive.ics.uci.edu/dataset/602/dry+bean+dataset

In [ ]:
```
import pandas as pd

df_dry_bean = pd.read_excel('data/Dry_Bean_Dataset.xlsx')

x_orig = df.iloc[:, :16]
y_orig = df['Class']
```

In [ ]:
```
# Trata os dados removendo as colunas 'Area' e 'ConvexArea' e normalizando os dados
from sklearn.preprocessing import StandardScaler

x_tratado = x_orig.drop(columns=['Area', 'ConvexArea'])

scaler = StandardScaler()
x_tratado = scaler.fit_transform(x_tratado)
```

In [ ]:
```
from sklearn.model_selection import train_test_split

# treinamento com os dados originais
x_orig_train, x_orig_test, y_orig_train, y_orig_test = train_test_split(x_orig,
                      y_orig, test_size=0.25, stratify=y_orig,random_state=10)

# treinamento com os dados tratados
x_train, x_test, y_train, y_test = train_test_split(x_tratado,
                      y_orig, test_size=0.25, stratify=y_orig,random_state=10)
```

In [ ]:
```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

treinador = svm.SVC()  #algoritmo escolhido

modelo_orig = treinador.fit(x_orig_train, y_orig_train)

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(x_orig_train)
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
```

```
print('Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(x_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred))
```

```
Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO
[[  56    0  673    0  234    0   29]
 [   0  390    2    0    0    0    0]
 [  42    0 1079    0   98    0    3]
 [   0    0    0 2249    0  319   91]
 [  38    0   72   23  847   47  419]
 [   0    0    0  523   36  426  535]
 [   0    0    0   97  188  261 1431]]
              precision    recall  f1-score   support

    BARBUNYA       0.41      0.06      0.10       992
      BOMBAY       1.00      0.99      1.00       392
        CALI       0.59      0.88      0.71      1222
    DERMASON       0.78      0.85      0.81      2659
       HOROZ       0.60      0.59      0.59      1446
       SEKER       0.40      0.28      0.33      1520
        SIRA       0.57      0.72      0.64      1977

    accuracy                           0.63     10208
   macro avg       0.62      0.62      0.60     10208
weighted avg       0.61      0.63      0.60     10208

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[ 18   0 220   0  77   0  15]
 [  0 130   0   0   0   0   0]
 [  8   0 360   0  38   0   2]
 [  0   0   0 749   0 109  29]
 [ 10   0  28  10 289  11 134]
 [  0   0   0 169  10 136 192]
 [  0   0   0  39  48  89 483]]
              precision    recall  f1-score   support

    BARBUNYA       0.50      0.05      0.10       330
      BOMBAY       1.00      1.00      1.00       130
        CALI       0.59      0.88      0.71       408
    DERMASON       0.77      0.84      0.81       887
       HOROZ       0.63      0.60      0.61       482
       SEKER       0.39      0.27      0.32       507
        SIRA       0.56      0.73      0.64       659

    accuracy                           0.64      3403
   macro avg       0.64      0.63      0.60      3403
weighted avg       0.62      0.64      0.60      3403
```

In [ ]:

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

treinador = svm.SVC()  #algoritmo escolhido

modelo = treinador.fit(x_train, y_train)

# predição com os mesmos dados usados para treinar
y_pred = modelo.predict(x_train)
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO')
print(cm_train)
print(classification_report(y_train, y_pred))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados TRATADOS usados para TESTES')
y2_pred = modelo.predict(x_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred))
```

```
Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO
[[ 912    0   54    0    3    5   18]
 [   0  392    0    0    0    0    0]
 [  26    0 1171    0   16    3    6]
 [   0    0    0 2477    3   36  143]
 [   2    0   16   11 1391    0   26]
 [   5    0    0   18    0 1460   37]
 [   4    0    3  170   25   21 1754]]
              precision    recall  f1-score   support

    BARBUNYA       0.96      0.92      0.94       992
      BOMBAY       1.00      1.00      1.00       392
        CALI       0.94      0.96      0.95      1222
    DERMASON       0.93      0.93      0.93      2659
       HOROZ       0.97      0.96      0.96      1446
       SEKER       0.96      0.96      0.96      1520
        SIRA       0.88      0.89      0.89      1977

    accuracy                           0.94     10208
   macro avg       0.95      0.95      0.95     10208
weighted avg       0.94      0.94      0.94     10208

Matriz de confusão - com os dados TRATADOS usados para TESTES
[[297   0  17   0   2   4  10]
 [  0 130   0   0   0   0   0]
 [  9   0 385   0   6   0   8]
 [  0   0   0 821   1  15  50]
 [  1   0  11   5 457   0   8]
 [  3   0   0  19   0 468  17]
 [  2   0   0  68  10   5 574]]
              precision    recall  f1-score   support

    BARBUNYA       0.95      0.90      0.93       330
      BOMBAY       1.00      1.00      1.00       130
        CALI       0.93      0.94      0.94       408
    DERMASON       0.90      0.93      0.91       887
       HOROZ       0.96      0.95      0.95       482
       SEKER       0.95      0.92      0.94       507
        SIRA       0.86      0.87      0.87       659

    accuracy                           0.92      3403
   macro avg       0.94      0.93      0.93      3403
weighted avg       0.92      0.92      0.92      3403
```

In [ ]: