RELATÓRIO TÉCNICO

Wesley de Oliveira Mendes - 828.507

08 - Classificação de Formas

Prof. Rodrigo de Oliveira Plotze

Ribeirão Preoto - SP
2021

Processamento de Imagens e Imagens

Engenharia da Computação - 2021.01

Wesley de Oliveira Mendes, 828.507

Tarefa 08 - Classificação de Formas

- Objetivo
  - Desenvolver um sistema de visão computacional para classificação de formas.

Download das imagens

Imports

In [90]:
```
import cv2 as cv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

Code

Exercício

In [91]:
```
path = '/content/drive/MyDrive/Colab Notebooks/College/Signal-Image-Processing/data/frutas_dataset'
```

In [92]:
```
classification = (
    ('001', '030', 'Maça'),
    ('031', '060', 'Abacaxi'),
    ('061', '090', 'Banana'),
    ('091', '120', 'Pêssego'),
    ('121', '150', 'Pitanga'),
    ('151', '180', 'Laranja'),
    ('181', '210', 'Morango'),
    ('211', '240', 'Pera'),
    ('241', '270', 'Limão'),
    ('271', '300', 'Uva')
)
```

In [93]:
```
# dicionario para salvar dados das imagens
dataset = {
    'Diameter': [],
    'Perimeter': [],
    'Area': [],
    'Compactness': [],
    'Eccentricity': [],
    'Rectangularity': [],
    'Solidity': [],
    'Classes': [],
}

# loop para extrar dados de indices
for start, end, classes in classification:
    # loop para extrar dados das imagens
    for i in range(int(start), int(end) + 1):
        image = cv.imread(f'{path}/{i:03d}.bmp')
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

        _, image = cv.threshold(image, 127, 255, cv.THRESH_BINARY_INV)
        contour, order = cv.findContours(image, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)

        # extrair diametro
        diameter = np.sqrt(4 * cv.contourArea(contour[0]) / np.pi)
        dataset['Diameter'].append(float(diameter))

        # extrair perimetro
        perimeter = cv.countNonZero(cv.Canny(image, 50, 100))
        dataset['Perimeter'].append(float(perimeter))

        # extrair area
        area = cv.countNonZero(image)
        dataset['Area'].append(float(area))

        # extrair compacidade
        compactness = np.square(perimeter / area)
        dataset['Compactness'].append(float(compactness))

        # extrair excentricidade
        eccentricity = 0

        if len(contour[0]) > 5:
            (x, y), (minor_axis, major_axis), angle = cv.fitEllipse(contour[0])
            eccentricity = major_axis / minor_axis

        dataset['Eccentricity'].append(float(eccentricity))

        # extrair retangularidade
        x, y, width, height = cv.boundingRect(contour[0])
        rectangularity = area / (width * height)
        dataset['Rectangularity'].append(float(rectangularity))

        # extrair solidez
        area_obj = cv.contourArea(contour[0])
        area_fechoconvexo = cv.contourArea(cv.convexHull(contour[0]))
        solidity = area_obj / area
        dataset['Solidity'].append(float(solidity))

        # especificar classe da imagem
        dataset['Classes'].append(classes)
```

In [94]:
```
# criar datafram pandas a partir do dicionario gerado
df = pd.DataFrame(dataset)
df.head()
```

Out[94]:

	Diameter	Perimeter	Area	Compactness	Eccentricity	Rectangularity	Solidity	Classes
0	0.000000	1595.0	37892.0	0.001772	0.0	37892.00	0.000000	Maça
1	0.000000	1583.0	37340.0	0.001797	0.0	18670.00	0.000000	Maça
2	0.000000	1634.0	37661.0	0.001882	0.0	37661.00	0.000000	Maça
3	0.000000	1595.0	37326.0	0.001826	0.0	37326.00	0.000000	Maça
4	0.797885	1595.0	37587.0	0.001801	0.0	9396.75	0.000013	Maça

In [95]:
```
# extrair label de classificacao
labels = df['Classes'].astype('category').cat.categories.tolist()
labels_to_replace = {'Classes' : {k: v for k, v in zip(labels, list(range(1, len(labels) + 1)))}}
print(labels_to_replace)
```

```
{'Classes': {'Abacaxi': 1, 'Banana': 2, 'Laranja': 3, 'Limão': 4, 'Maça': 5, 'Morango': 6, 'Pera': 7, 'Pitanga': 8, 'Pêssego': 9, 'Uva': 10}}
```

In [96]:
```
# substituir label texto de classificacao para numerico
df.replace(labels_to_replace, inplace=True)
dataframe = df.copy()
dataframe.head()
```

Out[96]:

	Diameter	Perimeter	Area	Compactness	Eccentricity	Rectangularity	Solidity	Classes
0	0.000000	1595.0	37892.0	0.001772	0.0	37892.00	0.000000	5
1	0.000000	1583.0	37340.0	0.001797	0.0	18670.00	0.000000	5
2	0.000000	1634.0	37661.0	0.001882	0.0	37661.00	0.000000	5
3	0.000000	1595.0	37326.0	0.001826	0.0	37326.00	0.000000	5
4	0.797885	1595.0	37587.0	0.001801	0.0	9396.75	0.000013	5

In [97]:
```
# remover colunas desnecessarias
df.drop(['Diameter', 'Solidity'], axis=1, inplace=True)
df.head()
```

Out[97]:

	Perimeter	Area	Compactness	Eccentricity	Rectangularity	Classes
0	1595.0	37892.0	0.001772	0.0	37892.00	5
1	1583.0	37340.0	0.001797	0.0	18670.00	5
2	1634.0	37661.0	0.001882	0.0	37661.00	5
3	1595.0	37326.0	0.001826	0.0	37326.00	5
4	1595.0	37587.0	0.001801	0.0	9396.75	5

In [98]:
```
# separar as caracteristicas e os rotulos
X = df.iloc[:, :3]  # caracteristicas
y = df.iloc[:, -1]  # rotulos
```

In [99]:
```
# PROCESSAMENTO
# dividir o conjunto de dados em treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=25)
```

In [100]:
```
# normalizacao dos dados
normalize = StandardScaler()
normalize.fit(X_train)

X_train = normalize.transform(X_train)
X_test = normalize.transform(X_test)
```

In [101]:
```
# classificador
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
Y = knn.predict(X_test)
```

In [102]:
```
# resultado
acc = accuracy_score(y_test, Y)
print(f'Accuracy {acc:.2f} or {acc * 100:.0f}%\n')

# Matriz confusao
print(pd.crosstab(y_test, Y, rownames=['True'], colnames=['Predicao'], margins=True))
```

```
Accuracy 0.87 or 87%

Predicao  1  2   3  4   5   6   7  8  9  10  All
True
1         9  0   0  0   0   0   0  0  0   0    9
2         0  9   0  0   0   0   0  0  0   0    9
3         0  0  10  0   0   0   0  0  0   0   10
4         0  0   0  4   0   0   0  0  0   0    4
5         0  0   0  0  10   0   0  0  0   0   10
6         0  0   0  0   0   5   5  0  0   0   10
7         0  0   0  0   0   2   5  0  0   0    7
8         0  0   0  0   0   0   0  9  0   0    9
9         0  0   0  0   2   0   0  0  9   0   11
10        0  0   0  0   0   3   0  0  0   8   11
All       9  9  10  4  12  10  10  9  9   8   90
```

In [102]: