c6-clasificacion-multiple-image

February 4, 2024

IMAGE CLASSIFICATION

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import os
from tqdm import tqdm
import warnings
import os
```

[6]:
```
warnings.filterwarnings('ignore')
print(os.listdir("../Clasificacion"))
```

```
['.ipynb_checkpoints', '6l5tue4rl95l7cod0m6m4.zip', 'C5-Mapa_de_Colores-
Clase.ipynb', 'C5-Mapa_de_Colores.ipynb', 'C6-Clasificacion-Clase.ipynb',
'C6-Clasificacion.ipynb', 'C6-Clasificacion_IMG.ipynb',
'C6-Clasificacion_Multiple_IMG.ipynb', 'C6-Objetos_y_Figuras.ipynb', 'cats',
'City_car', 'city_car.jpeg', 'city_car2.jpeg', 'coef.yaml', 'cuboA.jpg',
'cuboB.jpg', 'dogs', 'images.ipynb', 'Links.txt', 'Logo.jpg', 'Motor_bike',
'test_set', 'training_set', 'Truck', 'Truck.jpeg', 'Truck2.jpg']
```

[7]:
```
dataset_truck = "../Clasificacion/Truck"
dataset_city_car = "../Clasificacion/City_car"
dataset_motor_bike = "../Clasificacion/Motor_bike"
```

[8]:
```
class_1 = dataset_truck
class_2 = dataset_city_car
class_3 = dataset_motor_bike
image_size = 128
```

[11]:
```
def dataset_data():
    dataset_class_1 = []
    dataset_class_2 = []
    dataset_class_3 = []
    for image1 in tqdm(os.listdir(class_1)):
        path = os.path.join(class_1, image1)
        img1 = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
```

```
        img1 = cv2.resize(img1, (image_size, image_size))
        dataset_class_1.append(img1)
    for image2 in tqdm(os.listdir(class_2)):
        path = os.path.join(class_2, image2)
        img2 = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        img2 = cv2.resize(img2, (image_size, image_size))
        dataset_class_2.append(img2)
    for image3 in tqdm(os.listdir(class_3)):
        path = os.path.join(class_3, image3)
        img3 = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        img3 = cv2.resize(img3, (image_size, image_size))
        dataset_class_3.append(img3)

    dataset_classes = np.concatenate((np.asarray(dataset_class_1),
                                      np.asarray(dataset_class_2),
                                      np.asarray(dataset_class_3)), axis=0)
    return dataset_classes
```

[12]:
```
x_data = dataset_data() # Recibir los datos del dataset
x_data = (x_data-np.min(x_data))/(np.max(x_data)-np.min(x_data)) #Normalización␣
 ↪de los datos
```

```
100%|
 | 1150/1150 [00:00<00:00, 3133.37it/s]
100%|
 | 1150/1150 [00:00<00:00, 3999.33it/s]
100%|
 | 1150/1150 [00:04<00:00, 279.48it/s]
```

[108]:
```
print(x_data.shape)
```

```
(3450, 128, 128)
```

[14]:
```
Y_0 = np.zeros(1150)
Y_1 = np.ones(1150)
Y_2 = np.ones(1150)*2
y_data = np.concatenate((Y_0, Y_1, Y_2), axis=0).reshape(x_data.shape[0],1)
```

[15]:
```
print("X tamaño: " , x_data.shape)
print("Y tamaño: " , y_data.shape)
```

```
X tamaño:  (3450, 128, 128)
Y tamaño:  (3450, 1)
```

[16]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data,
                                                    test_size = 0.2,
```

```
                                                    random_state = 42)
number_of_train = x_train.shape[0]
number_of_test = x_test.shape[0]
```

[17]:
```
x_train_flatten = x_train.reshape(number_of_train, x_train.shape[1]*x_train.
 ↪shape[2]) # Distribuir la imagen
x_test_flatten = x_test.reshape(number_of_test, x_test.shape[1]*x_test.
 ↪shape[2]) # Distribuir la imagen
print("X train distribuida", x_train_flatten.shape)
print("X test distribuida", x_test_flatten.shape)
```

```
X train distribuida (2760, 16384)
X test distribuida (690, 16384)
```

[21]:
```
x_train = x_train_flatten
x_test = x_test_flatten
y_test = y_test
y_train = y_train
print("x train: ", x_train.shape)
print("x test: ", x_test.shape)
print("y train: ", y_train.shape)
print("y test: ", y_test.shape)
```

```
x entrenamiento:  (2760, 16384)
x testeo:  (690, 16384)
y entrenamiento:  (2760, 1)
y testeo:  (690, 1)
```

[101]:
```
from sklearn import linear_model

logreg = linear_model.LogisticRegression(multi_class='ovr')
print("Precisión del Test: {} ".format(logreg.fit(x_train, y_train).
 ↪score(x_test, y_test)))
print("Precisión del Training: {} ".format(logreg.fit(x_train, y_train).
 ↪score(x_train, y_train)))
```

```
Precisión del Test: 0.855072463768116
Precisión del Training: 1.0
```

[107]:
```
skl_inter = logreg.intercept_
skl_coef = logreg.coef_

cv_file = cv2.FileStorage("coef.yaml", cv2.FILE_STORAGE_WRITE)
cv_file.write("skl_inter", skl_inter)
cv_file.write("skl_coef", skl_coef)
cv_file.release()
```

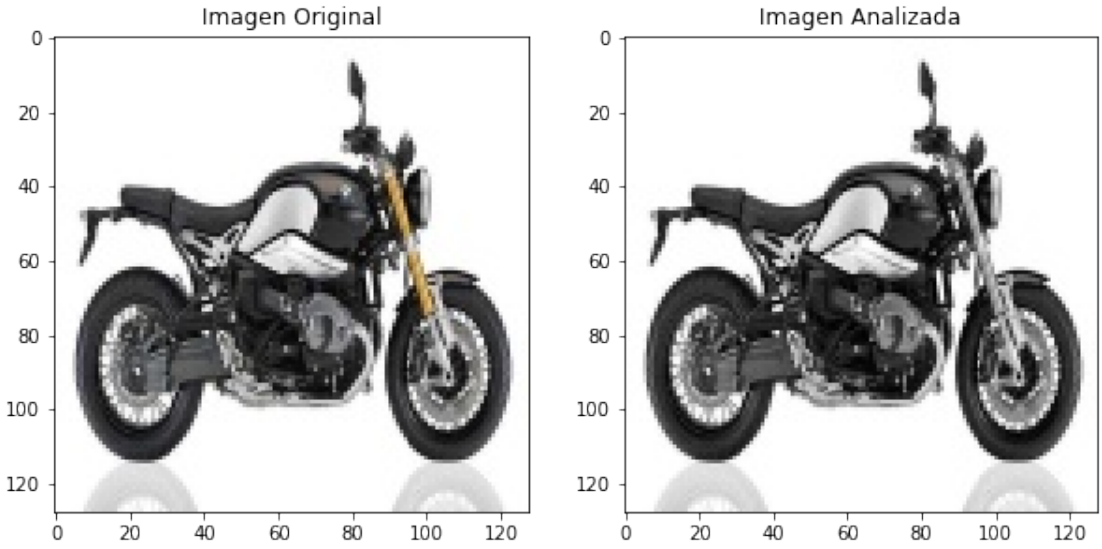
[104]:
```
def logistic_regression_skl(img_test, coef, bias):
    img_test_size = cv2.cvtColor(img_test, cv2.COLOR_BGR2GRAY)
    img_test_size = cv2.resize(img_test_size, (image_size, image_size))
    x_test = img_test_size.reshape(1, 128*128)
    score = (np.dot(x_test, coef.T) + bias)*0.001
    result = np.exp(score) / np.sum(np.exp(score), axis=1)

    plt.figure(figsize=(10,5))
    plt.subplot(121)
    plt.imshow(cv2.cvtColor(img_test, cv2.COLOR_BGR2RGB))
    plt.title("Imagen Original")
    plt.subplot(122)
    plt.imshow(img_test_size, cmap='gray')
    plt.title("Imagen Analizada")
    plt.show()

    print("Truck probility : "+str(result[0,0]))
    print("Car Probability : "+str(result[0,1]))
    print("Motorcycle probability  : "+str(result[0,2]))
```

[106]:
```
img_test = cv2.imread("Test_Truck_Car_Bike/11.jpg")
logistic_regression_skl(img_test, skl_coef, skl_inter)
```

```
(1, 16384)
```




```
Probabilidad de ser un camion : 0.00024474272079702834
Probabilidad de ser un auto de ciudad : 0.0014823211401776171
Probabilidad de ser una motocicleta  : 0.9982729361390253
```

[ ]: