```
#-----------------------------------------------------
#-----------------------------------------------------
# Ejemplo: Reconocimiento de Texturas con LBP, Haralick y Gabor
#-----------------------------------------------------

#-----------------------------------------------------
#0. Instalación de PyBalu
#
# PyBalu es una librería creada para extraer características.
#-----------------------------------------------------
# Instalación de Balu
!pip install scipy==1.2
!pip3 install pybalu==0.2.5
print('PyBalu library installed.')
#=====================================================
# Setup de librerías
#-----------------------------------------------------
import numpy as np
import cv2
import matplotlib.pyplot as plt
from   sklearn.metrics import confusion_matrix, accuracy_score
from   sklearn.neighbors import KNeighborsClassifier
from   seaborn import heatmap
from   google.colab.patches import cv2_imshow
from   tqdm.auto import tqdm
from   pybalu.feature_extraction import lbp_features, haralick_features,
gabor_features
from   pybalu.feature_transformation import normalize
#=====================================================


#-----------------------------------------------------
# Carga de base de datos
#-----------------------------------------------------
#=====================================================
# Carga de base de datos
!wget https://www.dropbox.com/.../textures.zip
!unzip textures
#=====================================================


#-----------------------------------------------------
# Funciones necesarias
#-----------------------------------------------------
#=====================================================
def num2fixstr(x,d):
  # example num2fixstr(2,5) returns '00002'
  # example num2fixstr(19,3) returns '019'
  st = '%0*d' % (d,x)
  return st

def ImageLoad(prefix,num_char,num_img,echo='off'):
  st   = prefix + num2fixstr(num_char,3) + '_' + num2fixstr(num_img,3) + '.png'
  if echo == 'on':
    print('loading image '+st+'...')
  img    = plt.imread(st)
  return img
#=====================================================


#-----------------------------------------------------
# Extracción de características
#-----------------------------------------------------
#=====================================================
# ADVERTENCIA: si K es grande y se extraen características de Gabor, este proceso
# puede durar unos 30 minutos!

K     = 11   # número de clases
N     = 9     # número de imágenes por clase
```

```
Mlbp  = 59
MHar1 = 28
MHar3 = 28
MHar5 = 28
Mgab  = 67

Xlbp  = np.zeros((K*N,Mlbp))    # K x N muestras (filas), y Mlbp  características (columnas)
XHar1 = np.zeros((K*N,MHar1))   # K x N muestras (filas), y Mhar1 características (columnas)
XHar3 = np.zeros((K*N,MHar3))   # K x N muestras (filas), y Mhar3 características (columnas)
XHar5 = np.zeros((K*N,MHar5))   # K x N muestras (filas), y Mhar5 características (columnas)
Xgab  = np.zeros((K*N,Mgab))    # K x N muestras (filas), y Mgab  características (columnas)

y     = np.zeros((K*N),'int')           # ground truth (clasificacion ideal)

t = 0
print('Cargando imagenes y extrayendo características...')
for j in tqdm(range(K)):            # para cada clase
  for i in range(N):                # para cada imagen de la clase
    # Lectura de la imagen
    img    = 255*ImageLoad('textures/D',j+1,i+1,echo='off')

    # Extracción de características
    Xlbp[t,:]  = lbp_features(img, hdiv=1, vdiv=1, mapping='nri_uniform')
    XHar1[t,:] = haralick_features(img, distance=1)
    XHar3[t,:] = haralick_features(img, distance=3)
    XHar5[t,:] = haralick_features(img, distance=5)
    #Xgab[t,:]  = gabor_features(img, rotations=8, dilations=8)
    y[t] = j
    t = t+1
#=====================================================


#-----------------------------------------------------
# Selección (Manual) de Características
#-----------------------------------------------------
#=====================================================
# Escoger características a usar
# ------------------------------
#
#X = np.concatenate((XHar1, XHar3, XHar5, Xgab, Xlbp), axis=1)
#X = np.concatenate((XHar5, XHar3), axis=1)
X = Xlbp
#X = Xgab
#X = XHar1
print('X tiene '+str(X.shape[0])+' filas, y '+str(X.shape[1])+' columnas.')
#=====================================================


#-----------------------------------------------------
# Separación Training/Testing
#-----------------------------------------------------
#Antes de diseñar el clasificador, debemos hacer la separación
#de los datos en subset de Training y substets de Testing.
#En este ejemplo usamos una imagen de cada textura para el testing
#(primera imagen) y el resto para el training:

# Training: las primeras 8 imágenes de cada textura

# Testing: la última imagen de cada textura
#-----------------------------------------------------
#=====================================================
# Separación entre training y testing
# Se escoge la primera foto de cada textura para testing, el resto para training

```

```
M = X.shape[1]

Xtrain = np.zeros((K*(N-1),M))  # 8 fotos por textura para training
ytrain = np.zeros((K*(N-1)),'int')
Xtest  = np.zeros((K,M))        # 1 foto por textura para testing
ytest  = np.zeros((K),'int')

itest  = 8                      # imagen escogida para testing, puede ser 0,1,..8

ktrain = 0                      # contador para Xtrain
ktest  = 0                      # contador para Xtest
for i in range(K*N):
  if np.mod(i,N)==itest:
    Xtest[ktest,:] = X[i,:]
    ytest[ktest]   = y[i]
    ktest = ktest+1
  else:
    Xtrain[ktrain,:] = X[i,:]
    ytrain[ktrain]   = y[i]
    ktrain = ktrain+1

print('Separación Training/Testing realizada.')


print('Xtrain tiene '+str(Xtrain.shape[0])+' filas, y '+str(Xtrain.shape[1])+'
columnas.')
print('Xtest  tiene '+str(Xtest.shape[0])+' filas, y '+str(Xtest.shape[1])+'
columnas.')

Xtrain, a, b = normalize(Xtrain)
Xtest = Xtest * a + b

# print('Normalización realizada.')
#=====================================================


#-----------------------------------------------------
# Clasificación usando KNN
#-----------------------------------------------------
# En este ejemplo se usa la implementación de sklearn para KNN
#-----------------------------------------------------
#=====================================================
numero_vecinos = 1
knn = KNeighborsClassifier(n_neighbors=numero_vecinos)
knn.fit(Xtrain, ytrain)
ypred = knn.predict(Xtest)
print('Clasificación usando KNN-'+str(numero_vecinos)+ ' en '+str(K)+' imágenes de
testing realizada.')
#=====================================================


#-----------------------------------------------------
# Evaluación de desempeño
#-----------------------------------------------------
#=====================================================
acc = accuracy_score(ytest,ypred)
print('Testing Accuracy = '+str(acc*100)+'%')
#=====================================================
#=====================================================
C = confusion_matrix(ytest,ypred)
print('Matriz de Confusión:')
heatmap(C, cmap="YlGnBu")
#=====================================================


#-----------------------------------------------------
# Visualización de imágenes mal clasificadas
#-----------------------------------------------------
#=====================================================
```

```
199   for i in range(K):
200     for j in range(K):
201       if i!=j and C[i,j]>0:
202         # Imagen de testing
203         I     = ImageLoad('textures/D',i+1,itest+1,echo='off')
204         # Imágenes de training
205         ini = 1
206         for k in range(N):
207           if k!=itest:
208             Jk  = ImageLoad('textures/D',j+1,k+1,echo='off')
209             if ini==1:
210               ini = 0
211               J = Jk
212             else:
213               J = cv2.hconcat([J,Jk])
214         print('Error de Clasificación: se confundió texture '+str(i+1)+' (foto de
      testing)...')
215         plt.imshow(I*255,cmap='gray')
216         plt.show()
217         print('... ya que fue clasificada como textura '+str(j+1)+' (fotos de
      training):')
218         #implot = plt.imshow(IJ,cmap='gray')
219         plt.imshow(J*255,cmap='gray')
220         plt.show()
221         print('-------------------------------------------')
222   #=====================================================
223
224   #=====================================================
225   #=====================================================
226   #=====================================================
```