+ Código + Texto

Desarrollo descarga imágenes. Dataset.

```
!pip install mtcnn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting mtcnn
  Downloading mtcnn-0.1.1-py3-none-any.whl (2.3 MB)
     |████████████████████████████████| 2.3 MB 7.1 MB/s
Requirement already satisfied: keras>=2.0.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: opencv-python>=4.1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages
Installing collected packages: mtcnn
Successfully installed mtcnn-0.1.1
```

```
import mtcnn
# print version
print(mtcnn.__version__)
```

```
0.1.0
```

```
import numpy as np
import pandas as pd
import cv2 # opencv
from mtcnn.mtcnn import MTCNN
from matplotlib import pyplot as plt
from keras.models import load_model
from PIL import Image

import os


# load the face dataset
data = np.load ('/content/celebrity-faces-dataset.npz')
trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'], data['arr_3']
print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)
```

```
Loaded:  (831, 160, 160, 3) (831,) (337, 160, 160, 3) (337,)
```

```
facenet_model = load_model('/content/facenet_keras.h5')
print('Loaded Model')
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was *
Loaded Model
```

```
def get_embedding(model, face):
    # scale pixel values
```

```
        face = face.astype('float32')
        # standardization
        mean, std = face.mean(), face.std()
        face = (face-mean)/std
        # transfer face into one sample (3 dimension to 4 dimension)
        sample = np.expand_dims(face, axis=0)
        # make prediction to get embedding
        yhat = model.predict(sample)
        return yhat[0]

    # convert each face in the train set into embedding
    emdTrainX = list()
    for face in trainX:
        emd = get_embedding(facenet_model, face)
        emdTrainX.append(emd)

    emdTrainX = np.asarray(emdTrainX)
    print(emdTrainX.shape)

    # convert each face in the test set into embedding
    emdTestX = list()
    for face in testX:
        emd = get_embedding(facenet_model, face)
        emdTestX.append(emd)
    emdTestX = np.asarray(emdTestX)
    print(emdTestX.shape)

    # save arrays to one file in compressed format
    np.savez_compressed('celebrity-faces-embeddings.npz', emdTrainX, trainy, emdTestX, testy)
```

```
(831, 128)
(337, 128)
```

```
    from sklearn.metrics import accuracy_score
    from sklearn.preprocessing import LabelEncoder
    from sklearn.preprocessing import Normalizer
    from sklearn.svm import SVC

    print("Dataset: train=%d, test=%d" % (emdTrainX.shape [0], emdTestX.shape [0]))
    # normalize input vectors
    in_encoder = Normalizer()
    emdTrainX_norm = in_encoder.transform(emdTrainX)
    emdTestX_norm = in_encoder.transform(emdTestX)
    # label encode targets
    out_encoder = LabelEncoder()
    out_encoder.fit(trainy)
    trainy_enc = out_encoder.transform(trainy)
    testy_enc = out_encoder.transform(testy)
    # fit model
    model = SVC(kernel='linear', probability=True)
```

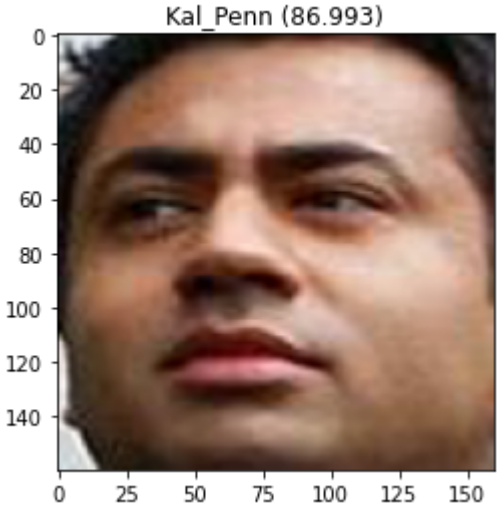
```
model.fit(emdTrainX_norm, trainy_enc)
# predict
yhat_train = model.predict(emdTrainX_norm)
yhat_test = model.predict(emdTestX_norm)
# score
score_train = accuracy_score(trainy_enc, yhat_train)
score_test = accuracy_score(testy_enc, yhat_test)
# summarize
print('Accuracy: train=%.3f, test=%.3f' % (score_train*100, score_test*100))
```

```
Dataset: train=831, test=337
Accuracy: train=100.000, test=99.407
```

```
from random import choice
# select a random face from test set
selection = choice([i for i in range(testX.shape[0])])
random_face = testX[selection]
random_face_emd = emdTestX_norm[selection]
random_face_class = testy_enc[selection]
random_face_name = out_encoder.inverse_transform([random_face_class])

# prediction for the face
samples = np.expand_dims(random_face_emd, axis=0)
yhat_class = model.predict(samples)
yhat_prob = model.predict_proba(samples)
# get name
class_index = yhat_class[0]
class_probability = yhat_prob[0,class_index] * 100
predict_names = out_encoder.inverse_transform(yhat_class)
all_names = out_encoder.inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11,12])
#print('Predicted: %s (%.3f)' % (predict_names[0], class_probability))
print('Predicted: \n%s \n%s' % (all_names, yhat_prob[0]*100))
print('Expected: %s' % random_face_name[0])
# plot face
plt.imshow(random_face)
title = '%s (%.3f)' % (predict_names[0], class_probability)
plt.title(title)
plt.show()
```

```
Predicted:
['Adam_Brody' 'Adam_McKay' 'Adam_Sandler' 'Billy_Zane' 'Brad_Garrett'
 'Brad_Pitt' 'Bradley_Cooper' 'Colin_Hanks' 'Justin_Long' 'Kal_Penn'
 'Tobey_Maguire' 'Victor_Garber' 'Woody_Allen']
[ 0.38143001  1.28406571  0.39231664  1.46649024  1.80585261  0.54328201
  1.13259968  2.71804325  0.82186845 86.99297323  0.28828843  1.61527496
  0.55751479]
Expected: Kal_Penn
```



✓ 0 s completado a las 3:44