https://colab.research.google.com/drive/1D7lGLBLeXhnSJep4f\_gtvJfG5k0PpXyj?usp=sharing#scrollTo=GXbZ1vtkOAeV

CARGAMOS LAS BASES DE DATOS

Clases:

* Cargar bases de datos JSON, BASE 64
* Cargar bases de datos CSV
* Preprocesamiento de datos

Utilice /tmp/ esto es en la nube en Google colab

import os

import zipfile

!wget --no-check-certificate https://storage.googleapis.com/platzi-tf2/databasesLoadData.zip \

    -O /tmp/databasesLoadData.zip

--2022-02-17 20:13:30-- <https://storage.googleapis.com/platzi-tf2/databasesLoadData.zip>

Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.204.128, 172.217.203.128, 172.253.123.128, ...

Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.204.128|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 71662510 (68M) [application/zip]

Saving to: ‘/tmp/databasesLoadData.zip’

/tmp/databasesLoadD 100%[===================>] 68.34M 175MB/s in 0.4s

2022-02-17 20:13:31 (175 MB/s) - ‘/tmp/databasesLoadData.zip’ saved [71662510/71662510]

# Descomprimimos

local\_zip = '/tmp/databasesLoadData.zip'

zip\_ref = zipfile.ZipFile(local\_zip, 'r')

zip\_ref.extractall('/tmp/databasesLoadData')

zip\_ref.close()

import json

import codecs

import requests

import numpy as np

from PIL import Image

from io import BytesIO

%matplotlib inline

import matplotlib.pyplot as plt

url = "/tmp/databasesLoadData/sign\_mnist\_json/data.json"

0s

data\_json = []  
with codecs.open(url, 'rU','utf-8') as js:  
  for line in js:  
    data\_json.append(json.loads(line))  
  
  
print("{} imagenes encontradas".format(len(data\_json)) )

9 imagenes encontradas

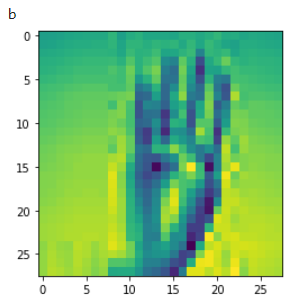
data\_json[0]

{'content': '<https://storage.googleapis.com/platzi-tf2/img_mnist/29_B.jpg>',

'label': 'b'}

images = []  
  
for data in data\_json:  
  response = requests.get(data['content'])  
  img = np.asarray(Image.open(BytesIO(response.content)))  
  images.append([img, data["label"]])

plt.imshow(images[0][0].reshape(28, 28))  
print(images[0][1])



import base64

url = "/tmp/databasesLoadData/sign\_mnist\_base64/data.json"

with open(url) as f:

  data = json.load(f)

print(data)

{'b': ''}

base64\_img\_bytes = data['b'].encode('utf-8')  
path\_img = "/tmp/decoded\_images.png"  
with open(path\_img, "wb") as file\_to\_save:  
  decoded\_image\_data = base64.decodebytes(base64\_img\_bytes)  
  file\_to\_save.write(decoded\_image\_data)

img = Image.open(path\_img)  
img

img = Image.open("/tmp/databasesLoadData/pixeles.png")  
img

import numpy as np  
import pandas as pd  
  
%matplotlib inline  
import matplotlib.pyplot as plt  
from PIL import Image  
import seaborn as sns

train = pd.read\_csv("/tmp/databasesLoadData/sign\_mnist\_train/sign\_mnist\_train.csv")  
test = pd.read\_csv("/tmp/databasesLoadData/sign\_mnist\_test/sign\_mnist\_test.csv")

train.head()

train.shape

(27455, 785)

labels = train['label'].values

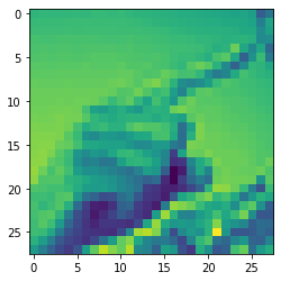
[ ]

train.drop('label', axis = 1, inplace = True)

train.head()

images = train.values

plt.imshow(images[1].reshape(28, 28))



train = pd.read\_csv("/tmp/databasesLoadData/sign\_mnist\_train/sign\_mnist\_train\_clean.csv")

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,524,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,598,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,623,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,672,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,697,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784) have mixed types.Specify dtype option on import or set low\_memory=False.

interactivity=interactivity, compiler=compiler, result=result)

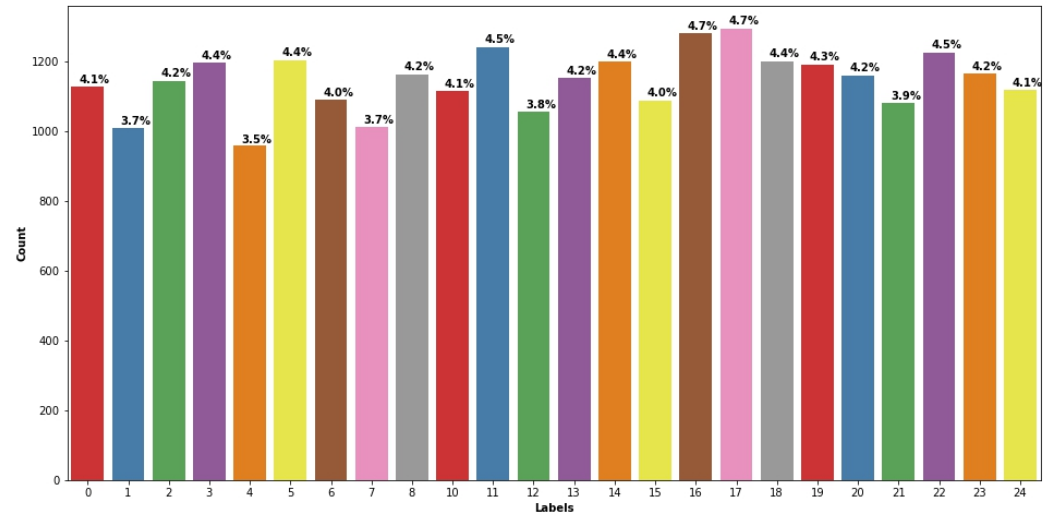
plt.figure(figsize = (10, 10))  
sns.set\_style("darkgrid")  
sns.countplot(train["label"])

MEJOR

fig, ax=plt.subplots(nrows=1, ncols=1, figsize=(16,8))  
ax=sns.countplot(data=train, x=‘label’, palette=‘Set1’)

for p in ax.patches:  
 ax.annotate(f’{round(p.get\_height()/len(train)\*100,1)}%’, (p.get\_x()+p.get\_width()/4, p.get\_height()\*1.01), weight=‘bold’)

plt.xlabel(‘Labels’,weight=‘bold’)  
plt.ylabel(‘Count’, weight=‘bold’)  
plt.show()



**Cuantas letras tiene cada letra, son 24 letras de referencia al lenguaje de señal**

y\_train = train['label']  
y\_test = test['label']

#En los datos a entrenar, y luego para verificar, no necesitamos la columna ‘label’ , esta es la única columna en el target

del train['label']  
del test['label']

train.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 27455 entries, 0 to 27454

Columns: 784 entries, pixel1 to pixel784

dtypes: object(784)

memory usage: 164.2+ MB

train.dtypes

pixel1 object

pixel2 object

pixel3 object

pixel4 object

pixel5 object

...

pixel780 object

pixel781 object

pixel782 object

pixel783 object

pixel784 object

Length: 784, dtype: object

unique\_val = np.array(labels)  
np.unique(unique\_val)

array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17,

18, 19, 20, 21, 22, 23, 24])

**SON 24 CLASE, 24 LETRAS DEL ALFABETO DE SENAS**

para saber cuales son los valores unicos de los labels

**train**['label'].unique()

para hacer un conteo de los labels

**train**.groupby('label')['label'].count()

**train.isnull().values.any()**

**train.isnull().values.any()**

False

**#verifique si no hay datos duplicados, y si lo hay bórrelos**

**train[train.duplicated()]**

train = train.drop([317,487, 595, 689, 802, 861], axis = 0)

#verifique si no hay valores basura como ‘fwefew’, y si los hay bórrelos por el índice de la fila

train[train['pixel1'] == "fwefew"]

train = train.drop([727], axis = 0)

#asegure que todos los datos son numericos

train = train.astype(str).astype(int)

train = train / 255  
test = test / 255

train.head()