EDIX EDUCACIÓN
MASTER FP IA Y BIG DATA

Reto final. Módulo 2. Aprendizaje automático

Alumno: Benjamín Pastor

Se debe entregar un informe con la descripción detallada de la actividad y las respuestas a todo lo que se solicita en el enunciado. El informe será un PDF con imágenes y texto donde se vean los pasos que se siguieron. Adjunto al informe debe aparecer el notebook con el código Python.

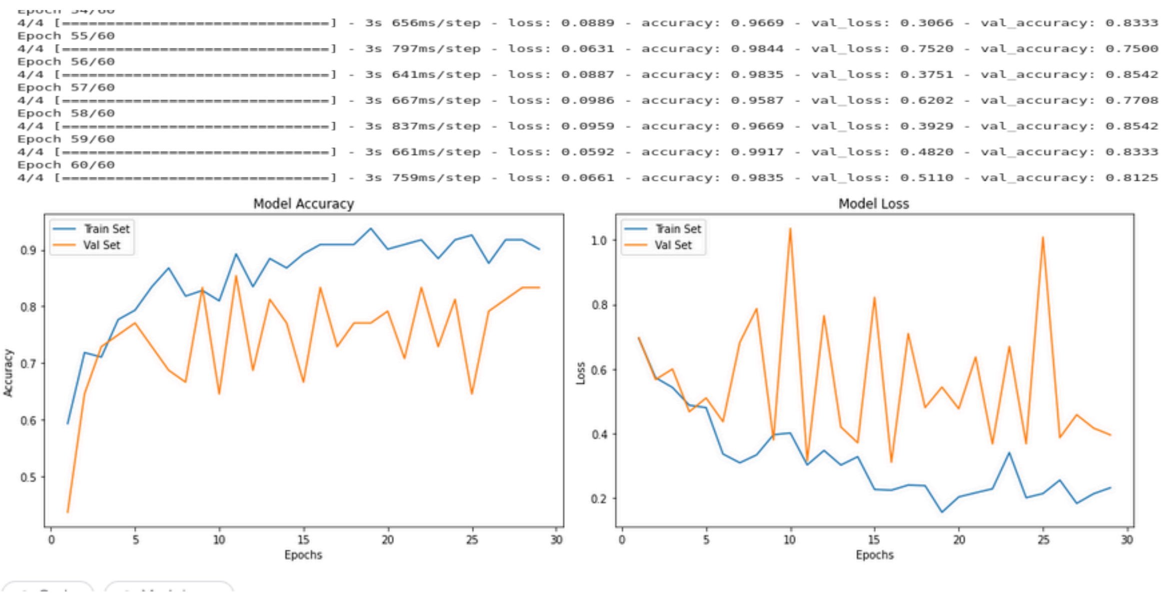
La aplicación permite mediante una entrada del usuario la elección de uno de los siguientes modelos: VGG16, Xception y ResNet50.

Como optimizadores de Keras se ha probado con Adam, RMSprop y SGD
y varios learning_rate desde 1e-4 hasta 1e-2, obteniéndose mejores resultados con Adam.

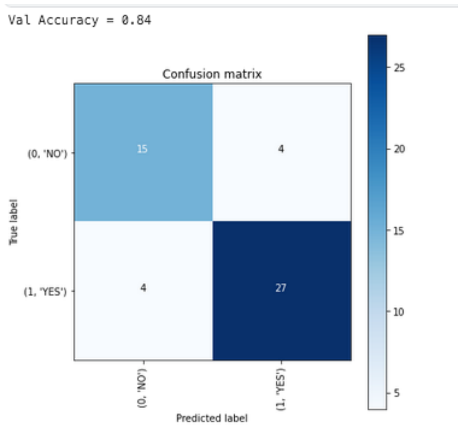
Además para mejorar la predicción se ha añadido una capa densa con 2048 neuronas y activación Relu para todos los modelos.

El único modelo que supera el 90% sobre el conjunto de prueba es el modelo VGG16.

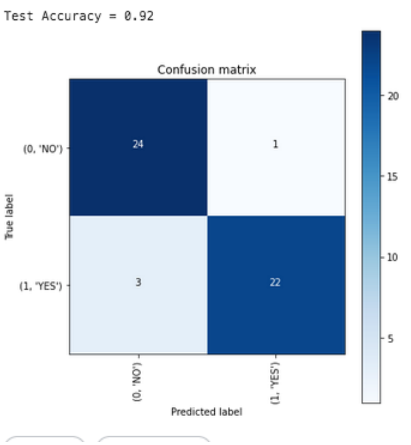
Mediante la utilización del modelo VGG16 se obtienen los siguientes resultados:



```
Epoch 54/60
4/4 [==============================] - 3s 656ms/step - loss: 0.0889 - accuracy: 0.9669 - val_loss: 0.3066 - val_accuracy: 0.8333
Epoch 55/60
4/4 [==============================] - 3s 797ms/step - loss: 0.0631 - accuracy: 0.9844 - val_loss: 0.7520 - val_accuracy: 0.7500
Epoch 56/60
4/4 [==============================] - 3s 641ms/step - loss: 0.0887 - accuracy: 0.9835 - val_loss: 0.3751 - val_accuracy: 0.8542
Epoch 57/60
4/4 [==============================] - 3s 667ms/step - loss: 0.0986 - accuracy: 0.9587 - val_loss: 0.6202 - val_accuracy: 0.7708
Epoch 58/60
4/4 [==============================] - 3s 837ms/step - loss: 0.0959 - accuracy: 0.9669 - val_loss: 0.3929 - val_accuracy: 0.8542
Epoch 59/60
4/4 [==============================] - 3s 661ms/step - loss: 0.0592 - accuracy: 0.9917 - val_loss: 0.4820 - val_accuracy: 0.8333
Epoch 60/60
4/4 [==============================] - 3s 759ms/step - loss: 0.0661 - accuracy: 0.9835 - val_loss: 0.5110 - val_accuracy: 0.8125
```

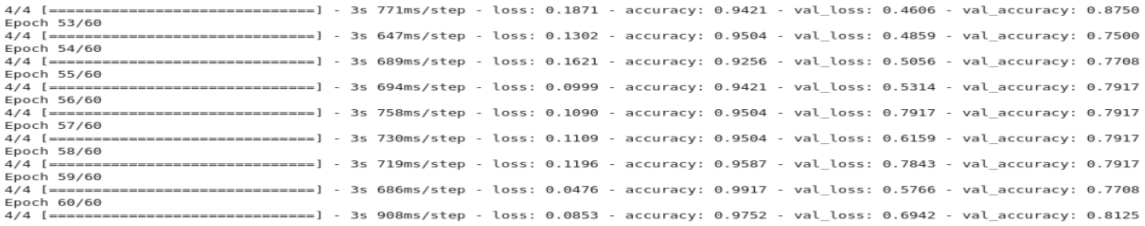


```
Val Accuracy = 0.84
```

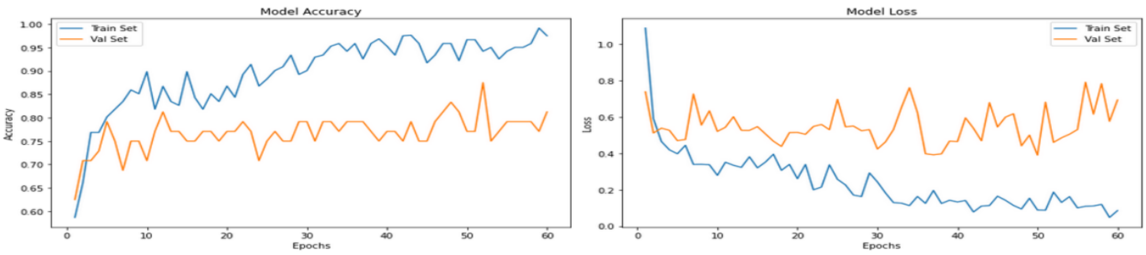


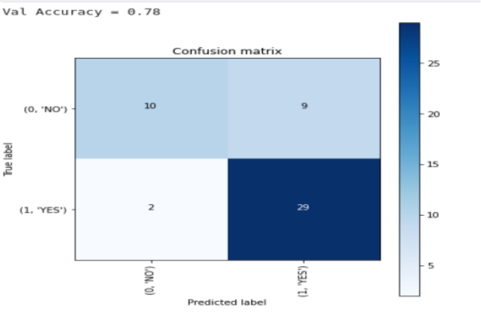
```
Test Accuracy = 0.92
```

Mediante la utilización del modelo Xception se obtienen los siguientes resultados:

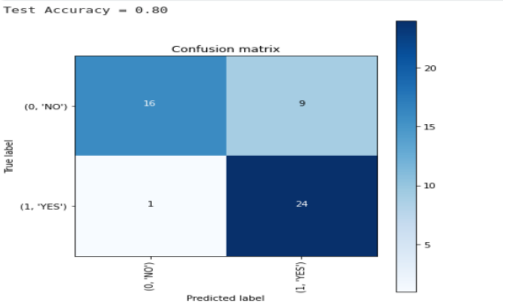


```
4/4 [==============================] - 3s 771ms/step - loss: 0.1871 - accuracy: 0.9421 - val_loss: 0.4606 - val_accuracy: 0.8750
Epoch 53/60
4/4 [==============================] - 3s 647ms/step - loss: 0.1302 - accuracy: 0.9504 - val_loss: 0.4859 - val_accuracy: 0.7500
Epoch 54/60
4/4 [==============================] - 3s 689ms/step - loss: 0.1621 - accuracy: 0.9256 - val_loss: 0.5056 - val_accuracy: 0.7708
Epoch 55/60
4/4 [==============================] - 3s 694ms/step - loss: 0.0999 - accuracy: 0.9421 - val_loss: 0.5314 - val_accuracy: 0.7917
Epoch 56/60
4/4 [==============================] - 3s 758ms/step - loss: 0.1090 - accuracy: 0.9504 - val_loss: 0.7917 - val_accuracy: 0.7917
Epoch 57/60
4/4 [==============================] - 3s 730ms/step - loss: 0.1109 - accuracy: 0.9504 - val_loss: 0.6159 - val_accuracy: 0.7917
Epoch 58/60
4/4 [==============================] - 3s 719ms/step - loss: 0.1196 - accuracy: 0.9587 - val_loss: 0.7843 - val_accuracy: 0.7917
Epoch 59/60
4/4 [==============================] - 3s 686ms/step - loss: 0.0476 - accuracy: 0.9917 - val_loss: 0.5766 - val_accuracy: 0.7708
Epoch 60/60
4/4 [==============================] - 3s 908ms/step - loss: 0.0853 - accuracy: 0.9752 - val_loss: 0.6942 - val_accuracy: 0.8125
```



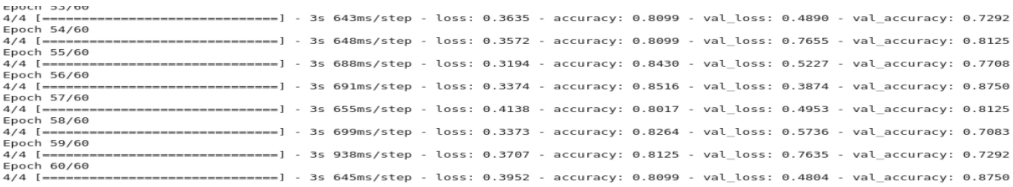


```
Val Accuracy = 0.78
```

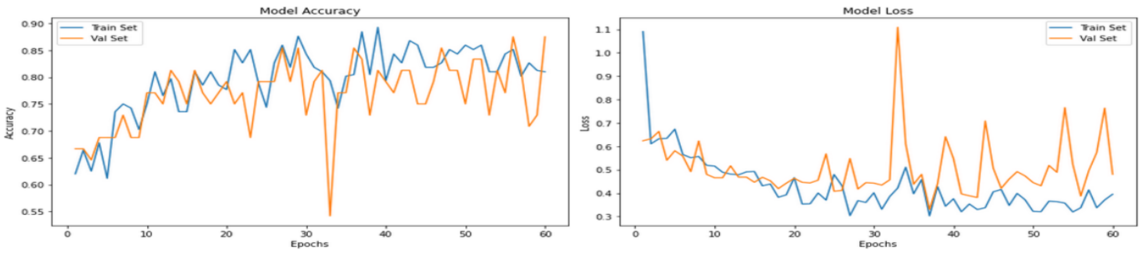


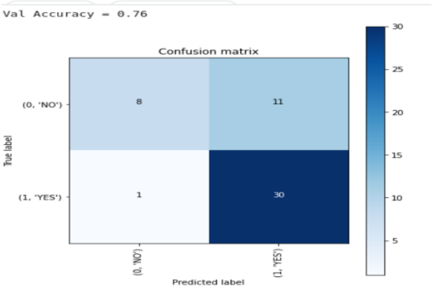
```
Test Accuracy = 0.80
```

Mediante la utilización del modelo ResNet50 se obtienen los siguientes resultados:

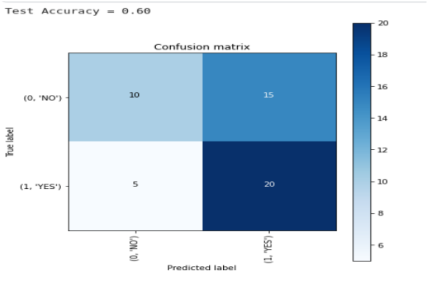


```
Epoch 53/60
4/4 [==============================] - 3s 643ms/step - loss: 0.3635 - accuracy: 0.8099 - val_loss: 0.4890 - val_accuracy: 0.7292
Epoch 54/60
4/4 [==============================] - 3s 648ms/step - loss: 0.3572 - accuracy: 0.8099 - val_loss: 0.7655 - val_accuracy: 0.8125
Epoch 55/60
4/4 [==============================] - 3s 688ms/step - loss: 0.3194 - accuracy: 0.8430 - val_loss: 0.5227 - val_accuracy: 0.7708
Epoch 56/60
4/4 [==============================] - 3s 691ms/step - loss: 0.3374 - accuracy: 0.8516 - val_loss: 0.3874 - val_accuracy: 0.8750
Epoch 57/60
4/4 [==============================] - 3s 655ms/step - loss: 0.4138 - accuracy: 0.8017 - val_loss: 0.4953 - val_accuracy: 0.8125
Epoch 58/60
4/4 [==============================] - 3s 699ms/step - loss: 0.3373 - accuracy: 0.8264 - val_loss: 0.5736 - val_accuracy: 0.7083
Epoch 59/60
4/4 [==============================] - 3s 938ms/step - loss: 0.3707 - accuracy: 0.8125 - val_loss: 0.7635 - val_accuracy: 0.7292
Epoch 60/60
4/4 [==============================] - 3s 645ms/step - loss: 0.3952 - accuracy: 0.8099 - val_loss: 0.4804 - val_accuracy: 0.8750
```





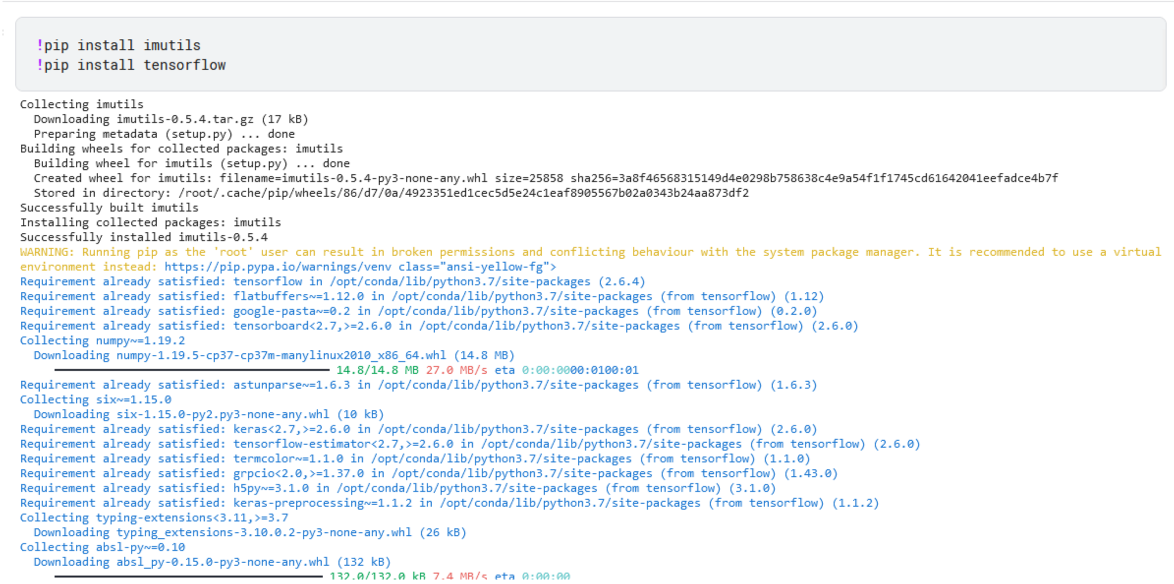
```
Val Accuracy = 0.76
```



```
Test Accuracy = 0.60
```

A continuación breve explicación y capturas del proceso:

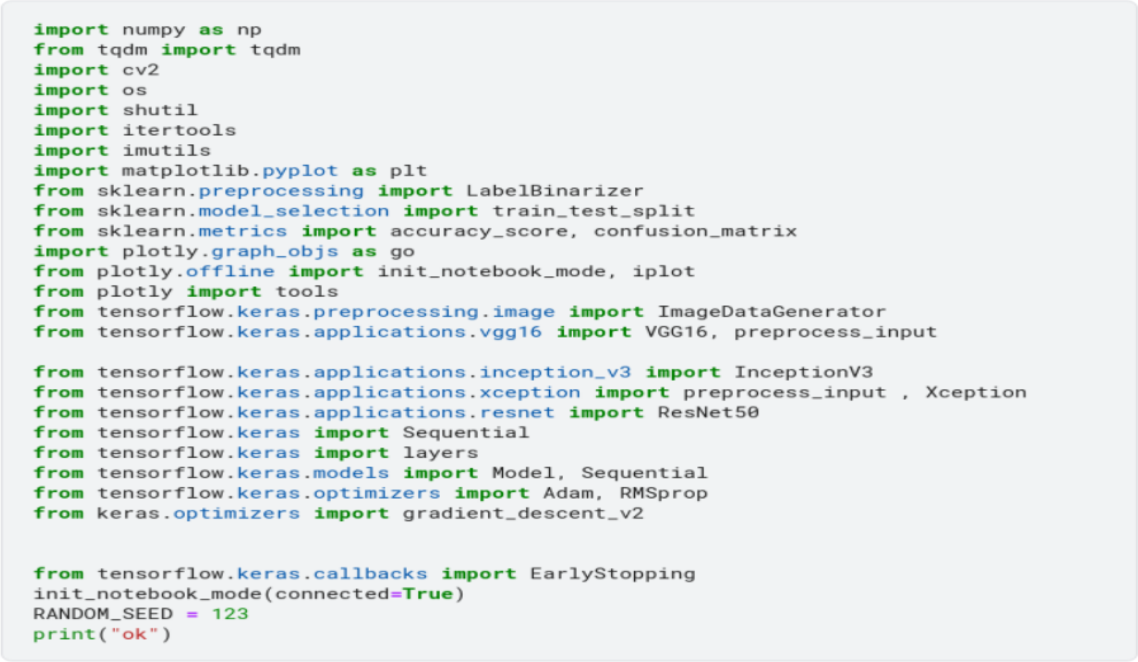
1. Instalar imutils y Tensorflow



```
!pip install imutils
!pip install tensorflow
```

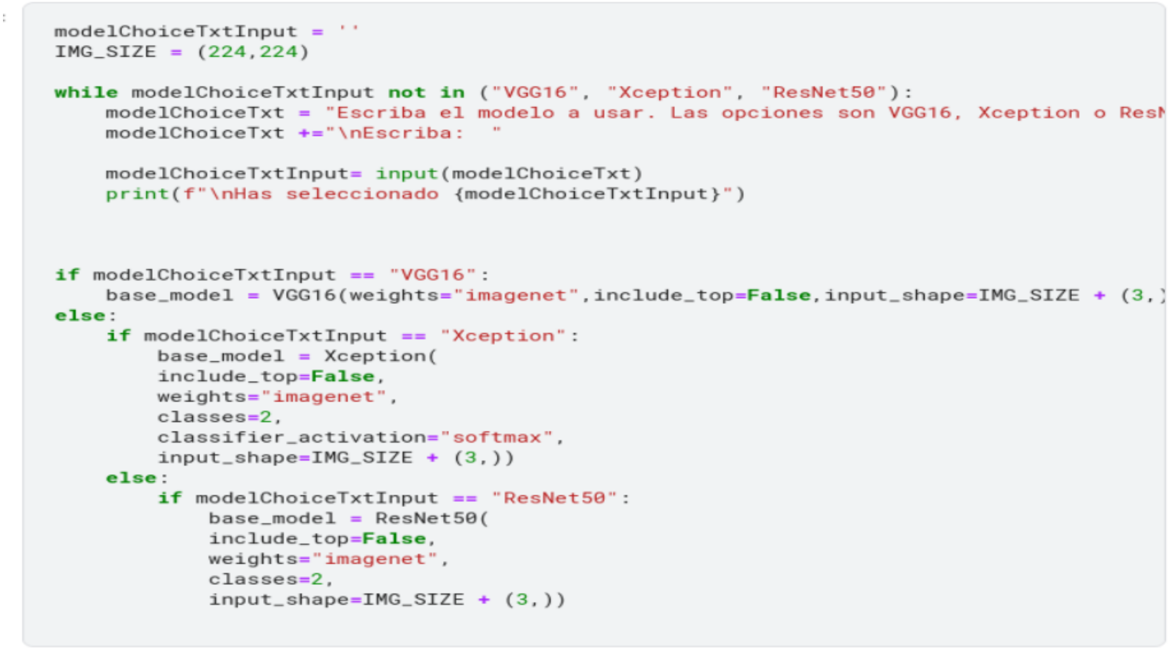
```
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25858 sha256=3a8f46568315149d4e0298b758638c4e9a54f1f1745cd61642041eefadce4b7f
  Stored in directory: /root/.cache/pip/wheels/86/d7/0a/4923351ed1cec5d5e24c1eaf8905567b02a0343b24aa873df2
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual
environment instead: https://pip.pypa.io/warnings/venv class="ansi-yellow-fg">
Requirement already satisfied: tensorflow in /opt/conda/lib/python3.7/site-packages (2.6.4)
Requirement already satisfied: flatbuffers~=1.12.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (1.12)
Requirement already satisfied: google-pasta~=0.2 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard<2.7,>=2.6.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (2.6.0)
Collecting numpy~=1.19.2
  Downloading numpy-1.19.5-cp37-cp37m-manylinux2010_x86_64.whl (14.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.8/14.8 MB 27.0 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: astunparse~=1.6.3 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (1.6.3)
Collecting six~=1.15.0
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: keras<2.7,>=2.6.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (2.6.0)
Requirement already satisfied: tensorflow-estimator<2.7,>=2.6.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (2.6.0)
Requirement already satisfied: termcolor~=1.1.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: grpcio<2.0,>=1.37.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (1.43.0)
Requirement already satisfied: h5py~=3.1.0 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (3.1.0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /opt/conda/lib/python3.7/site-packages (from tensorflow) (1.1.2)
Collecting typing-extensions<3.11,>=3.7
  Downloading typing_extensions-3.10.0.2-py3-none-any.whl (26 kB)
Collecting absl-py~=0.10
  Downloading absl_py-0.15.0-py3-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.0/132.0 kB 7.4 MB/s eta 0:00:00
```

2. Importación librerías



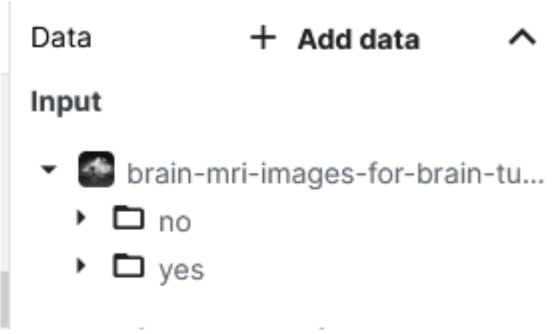
```
import numpy as np
from tqdm import tqdm
import cv2
import os
import shutil
import itertools
import imutils
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
from plotly import tools
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input

from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.xception import preprocess_input , Xception
from tensorflow.keras.applications.resnet import ResNet50
from tensorflow.keras import Sequential
from tensorflow.keras import layers
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.optimizers import gradient_descent_v2


from tensorflow.keras.callbacks import EarlyStopping
init_notebook_mode(connected=True)
RANDOM_SEED = 123
print("ok")
```

3. Selección del modelo

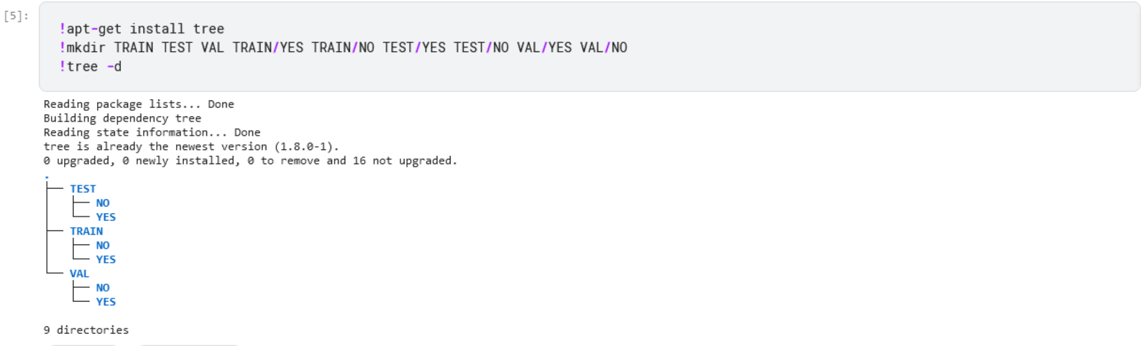


```
modelChoiceTxtInput = ''
IMG_SIZE = (224,224)

while modelChoiceTxtInput not in ("VGG16", "Xception", "ResNet50"):
    modelChoiceTxt = "Escriba el modelo a usar. Las opciones son VGG16, Xception o ResN
    modelChoiceTxt +="\nEscriba:  "

    modelChoiceTxtInput= input(modelChoiceTxt)
    print(f"\nHas seleccionado {modelChoiceTxtInput}")


if modelChoiceTxtInput == "VGG16":
    base_model = VGG16(weights="imagenet",include_top=False,input_shape=IMG_SIZE + (3,)
else:
    if modelChoiceTxtInput == "Xception":
        base_model = Xception(
        include_top=False,
        weights="imagenet",
        classes=2,
        classifier_activation="softmax",
        input_shape=IMG_SIZE + (3,))
    else:
        if modelChoiceTxtInput == "ResNet50":
            base_model = ResNet50(
            include_top=False,
            weights="imagenet",
            classes=2,
            input_shape=IMG_SIZE + (3,))
```

4. Importación dataset



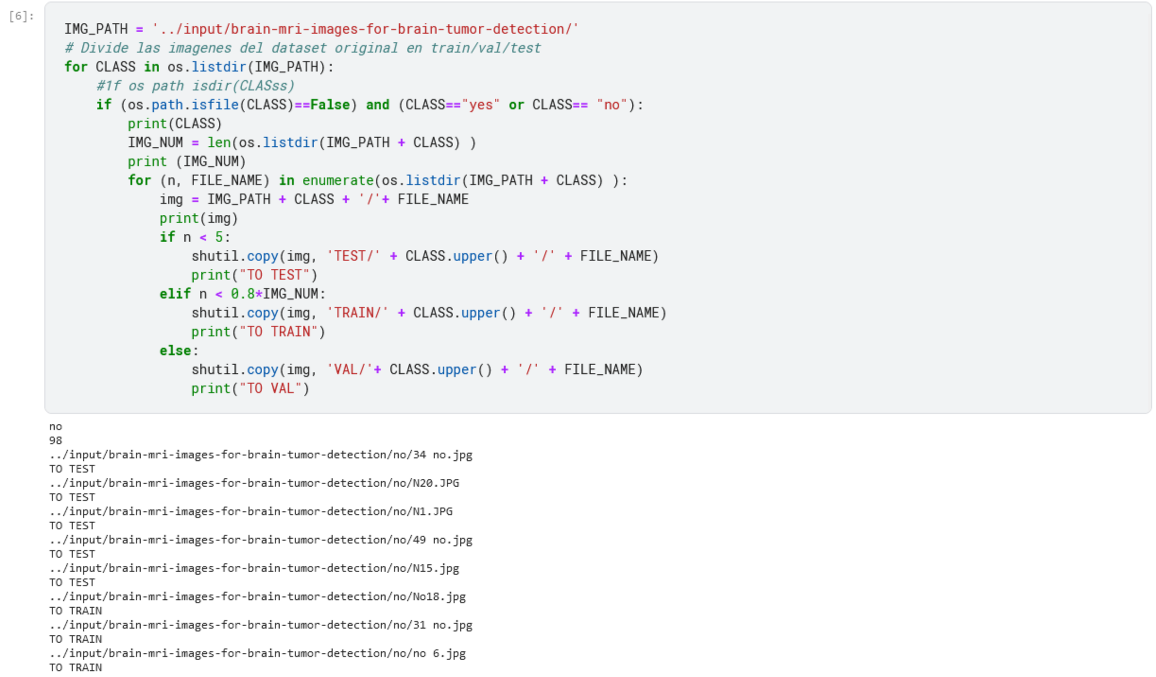
5. División en subconjuntos



```
[5]:
!apt-get install tree
!mkdir TRAIN TEST VAL TRAIN/YES TRAIN/NO TEST/YES TEST/NO VAL/YES VAL/NO
!tree -d
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
tree is already the newest version (1.8.0-1).
0 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.
.
├── TEST
│   ├── NO
│   └── YES
├── TRAIN
│   ├── NO
│   └── YES
└── VAL
    ├── NO
    └── YES

9 directories
```

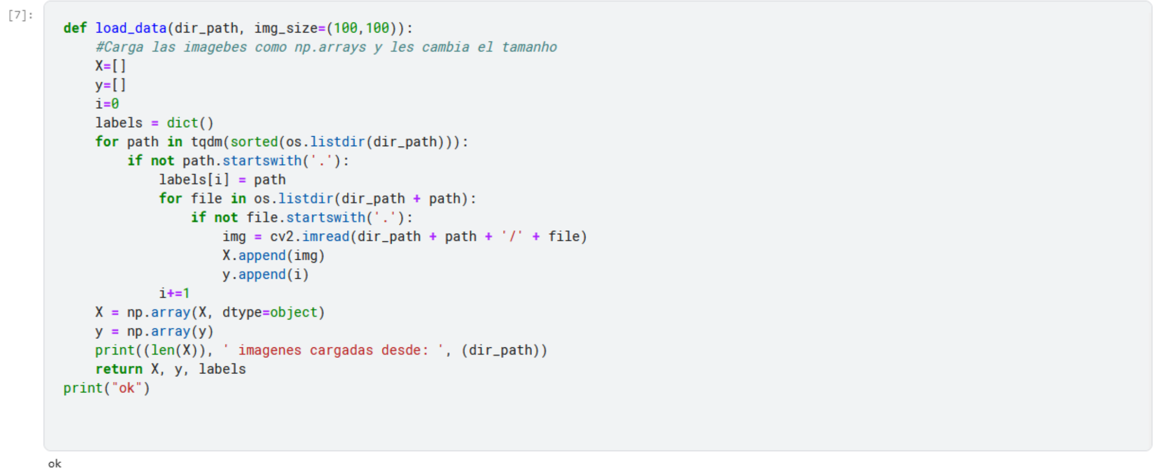
6. División 80-20 dataset



```
[6]:
IMG_PATH = '../input/brain-mri-images-for-brain-tumor-detection/'
# Divide las imagenes del dataset original en train/val/test
for CLASS in os.listdir(IMG_PATH):
    #1f os path isdir(CLASss)
    if (os.path.isfile(CLASS)==False) and (CLASS=="yes" or CLASS== "no"):
        print(CLASS)
        IMG_NUM = len(os.listdir(IMG_PATH + CLASS) )
        print (IMG_NUM)
        for (n, FILE_NAME) in enumerate(os.listdir(IMG_PATH + CLASS) ):
            img = IMG_PATH + CLASS + '/'+ FILE_NAME
            print(img)
            if n < 5:
                shutil.copy(img, 'TEST/' + CLASS.upper() + '/' + FILE_NAME)
                print("TO TEST")
            elif n < 0.8*IMG_NUM:
                shutil.copy(img, 'TRAIN/' + CLASS.upper() + '/' + FILE_NAME)
                print("TO TRAIN")
            else:
                shutil.copy(img, 'VAL/'+ CLASS.upper() + '/' + FILE_NAME)
                print("TO VAL")
```

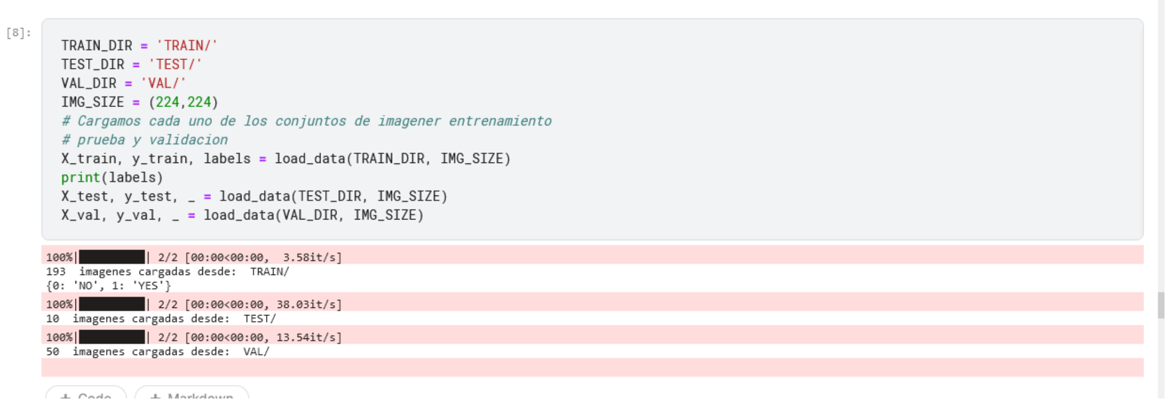
```
no
98
../input/brain-mri-images-for-brain-tumor-detection/no/34 no.jpg
TO TEST
../input/brain-mri-images-for-brain-tumor-detection/no/N20.JPG
TO TEST
../input/brain-mri-images-for-brain-tumor-detection/no/N1.JPG
TO TEST
../input/brain-mri-images-for-brain-tumor-detection/no/49 no.jpg
TO TEST
../input/brain-mri-images-for-brain-tumor-detection/no/N15.jpg
TO TEST
../input/brain-mri-images-for-brain-tumor-detection/no/No18.jpg
TO TRAIN
../input/brain-mri-images-for-brain-tumor-detection/no/31 no.jpg
TO TRAIN
../input/brain-mri-images-for-brain-tumor-detection/no/no 6.jpg
TO TRAIN
```

7. Función carga imágenes y copia a nuevas carpetas



```
[7]:
def load_data(dir_path, img_size=(100,100)):
    #Carga las imagebes como np.arrays y les cambia el tamanho
    X=[]
    y=[]
    i=0
    labels = dict()
    for path in tqdm(sorted(os.listdir(dir_path))):
        if not path.startswith('.'):
            labels[i] = path
            for file in os.listdir(dir_path + path):
                if not file.startswith('.'):
                    img = cv2.imread(dir_path + path + '/' + file)
                    X.append(img)
                    y.append(i)
            i+=1
    X = np.array(X, dtype=object)
    y = np.array(y)
    print((len(X)), ' imagenes cargadas desde: ', (dir_path))
    return X, y, labels
print("ok")
```

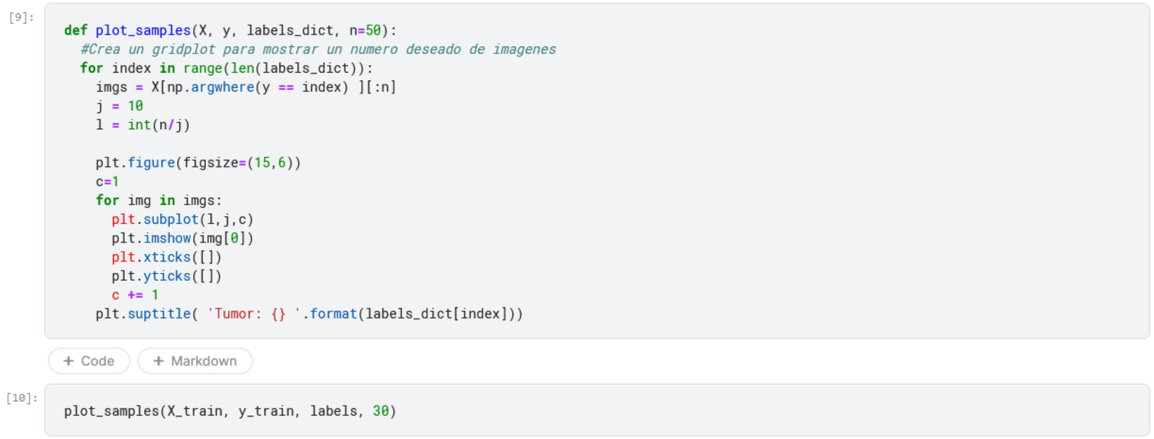
```
ok
```



```
[8]:
TRAIN_DIR = 'TRAIN/'
TEST_DIR = 'TEST/'
VAL_DIR = 'VAL/'
IMG_SIZE = (224,224)
# Cargamos cada uno de los conjuntos de imagener entrenamiento
# prueba y validacion
X_train, y_train, labels = load_data(TRAIN_DIR, IMG_SIZE)
print(labels)
X_test, y_test, _ = load_data(TEST_DIR, IMG_SIZE)
X_val, y_val, _ = load_data(VAL_DIR, IMG_SIZE)
```

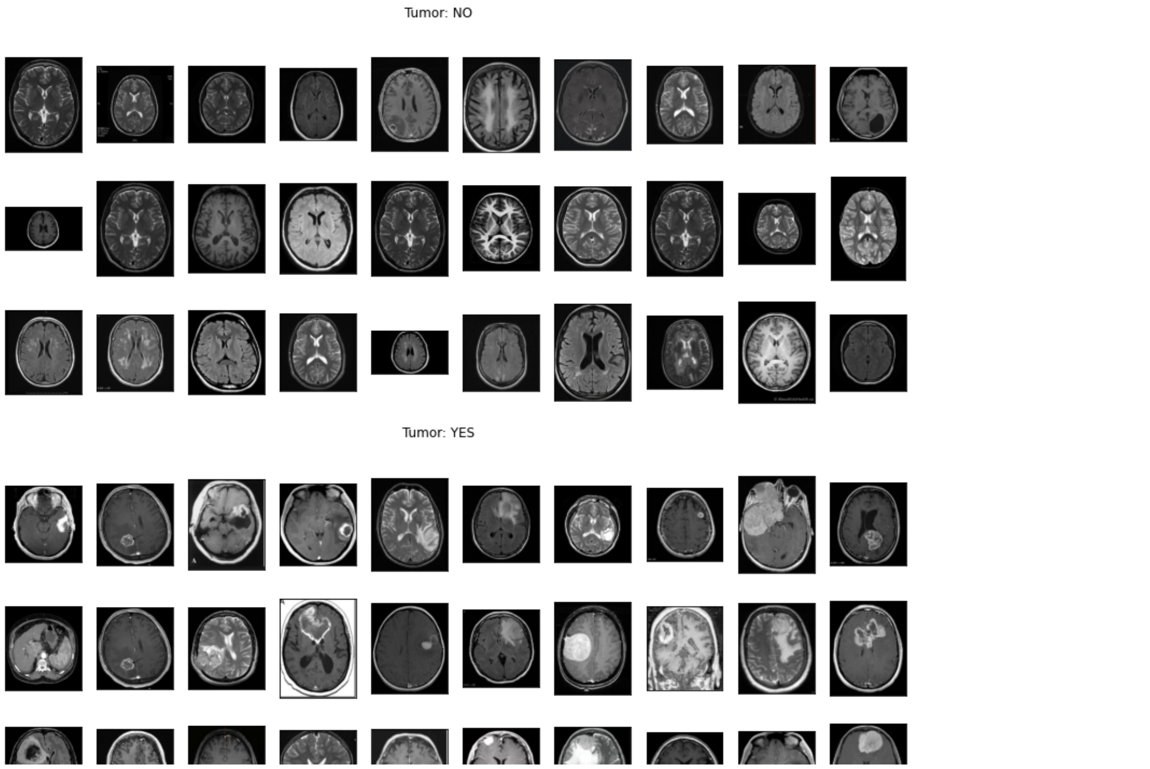
```
100%|██████████| 2/2 [00:00<00:00,  3.58it/s]
193  imagenes cargadas desde:  TRAIN/
{0: 'NO', 1: 'YES'}
100%|██████████| 2/2 [00:00<00:00, 38.03it/s]
10  imagenes cargadas desde:  TEST/
100%|██████████| 2/2 [00:00<00:00, 13.54it/s]
50  imagenes cargadas desde:  VAL/
```

8. Comprobar imágenes

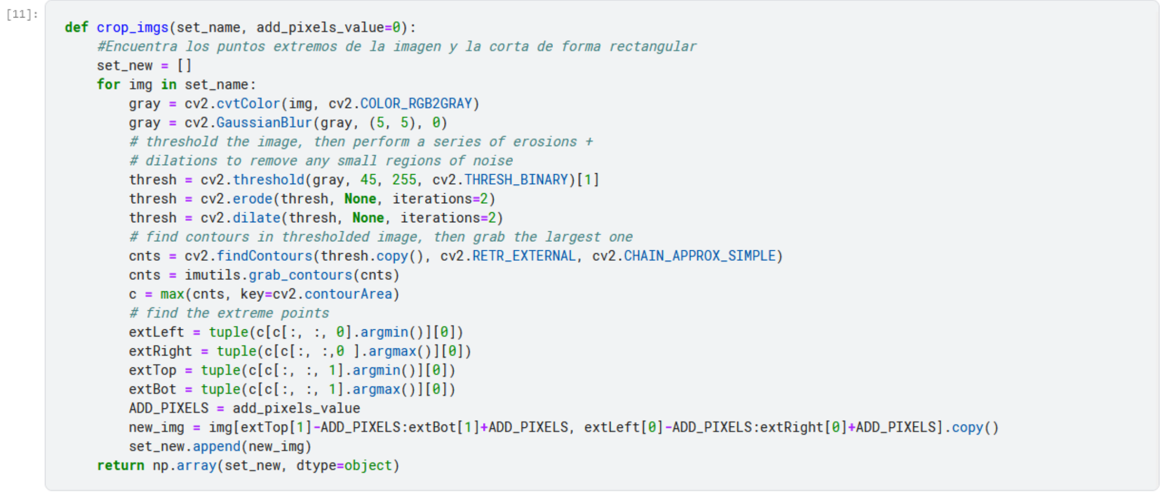


```
[9]:
def plot_samples(X, y, labels_dict, n=50):
  #Crea un gridplot para mostrar un numero deseado de imagenes
  for index in range(len(labels_dict)):
    imgs = X[np.argwhere(y == index) ][:n]
    j = 10
    l = int(n/j)

    plt.figure(figsize=(15,6))
    c=1
    for img in imgs:
      plt.subplot(l,j,c)
      plt.imshow(img[0])
      plt.xticks([])
      plt.yticks([])
      c += 1
    plt.suptitle( 'Tumor: {} '.format(labels_dict[index]))
```

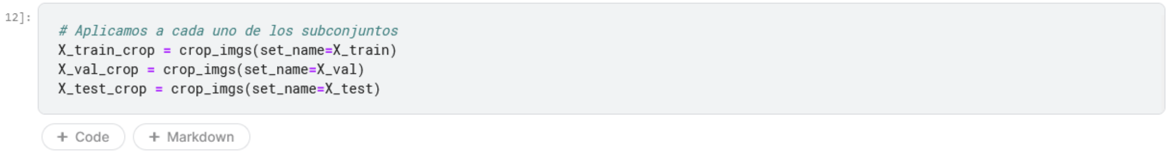
```
[10]:
plot_samples(X_train, y_train, labels, 30)
```



9. Recortar imágenes para mejorar precisión y aplicar a cada subconjunto

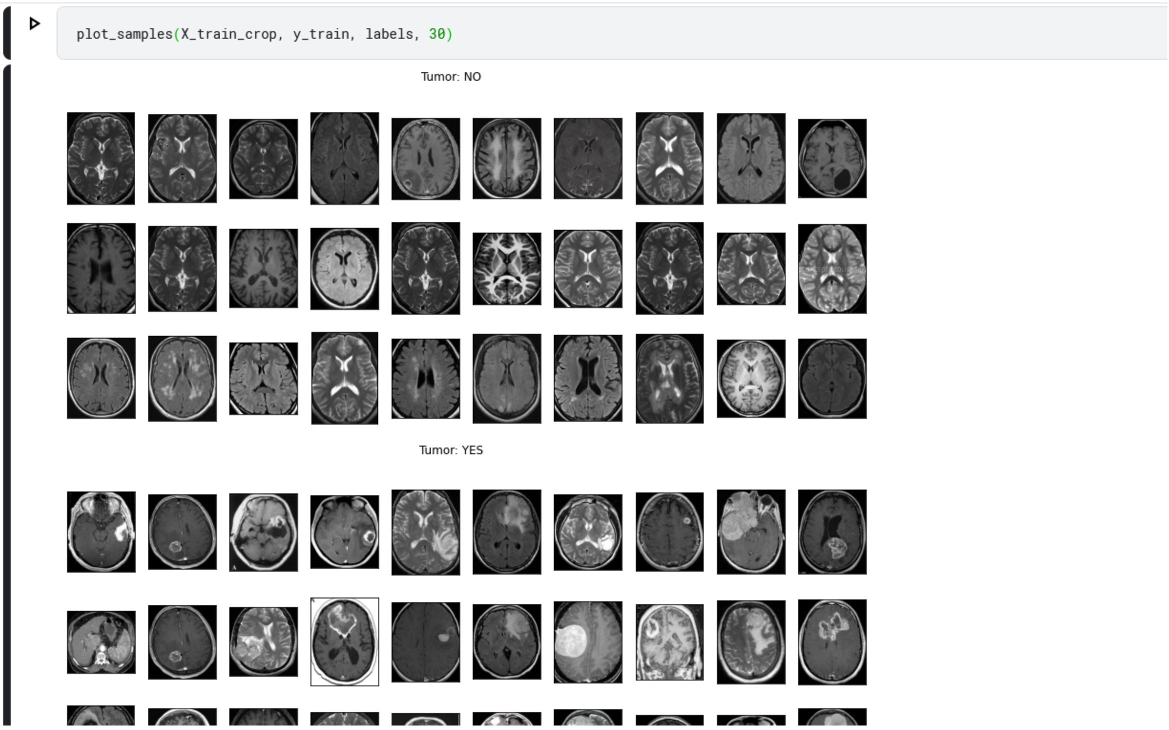


```
[11]:
def crop_imgs(set_name, add_pixels_value=0):
    #Encuentra los puntos extremos de la imagen y la corta de forma rectangular
    set_new = []
    for img in set_name:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)
        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)
        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)
        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :,0 ].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])
        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)
    return np.array(set_new, dtype=object)
```



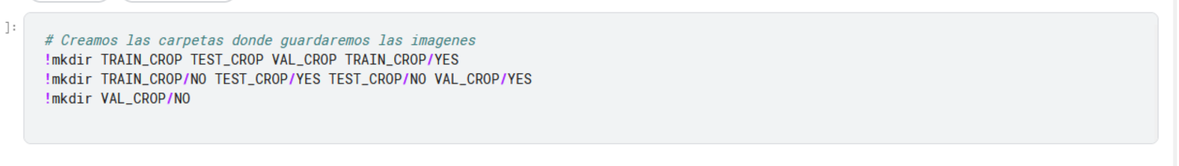
```
# Aplicamos a cada uno de los subconjuntos
X_train_crop = crop_imgs(set_name=X_train)
X_val_crop = crop_imgs(set_name=X_val)
X_test_crop = crop_imgs(set_name=X_test)
```

10. Comprobamos que imágenes están más ajustadas

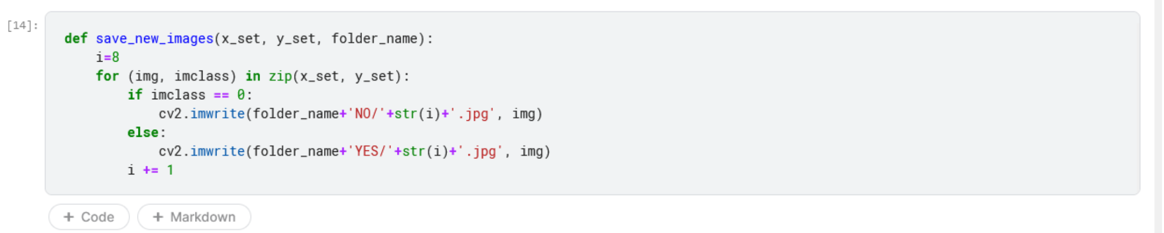


```
plot_samples(X_train_crop, y_train, labels, 30)
```

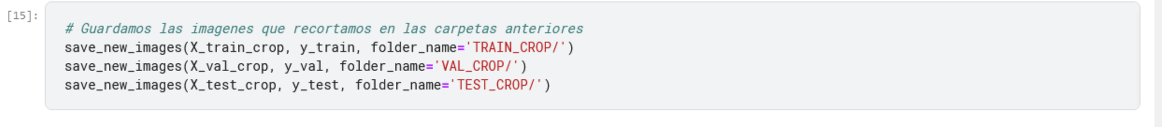
11. Creamos las carpetas donde guardaremos las nuevas imágenes



```
# Creamos las carpetas donde guardaremos las imagenes
!mkdir TRAIN_CROP TEST_CROP VAL_CROP TRAIN_CROP/YES
!mkdir TRAIN_CROP/NO TEST_CROP/YES TEST_CROP/NO VAL_CROP/YES
!mkdir VAL_CROP/NO
```

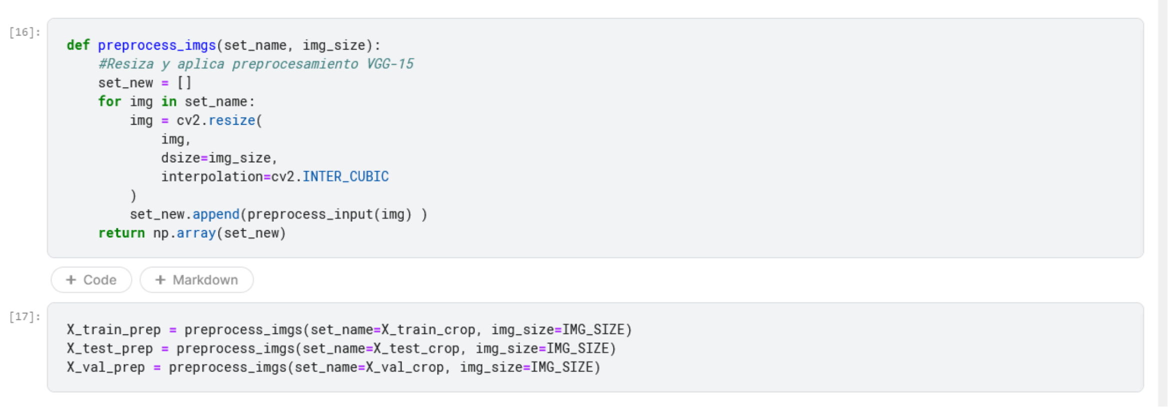


```
def save_new_images(x_set, y_set, folder_name):
    i=8
    for (img, imclass) in zip(x_set, y_set):
        if imclass == 0:
            cv2.imwrite(folder_name+'NO/'+str(i)+'.jpg', img)
        else:
            cv2.imwrite(folder_name+'YES/'+str(i)+'.jpg', img)
        i += 1
```



```
# Guardamos las imagenes que recortamos en las carpetas anteriores
save_new_images(X_train_crop, y_train, folder_name='TRAIN_CROP/')
save_new_images(X_val_crop, y_val, folder_name='VAL_CROP/')
save_new_images(X_test_crop, y_test, folder_name='TEST_CROP/')
```

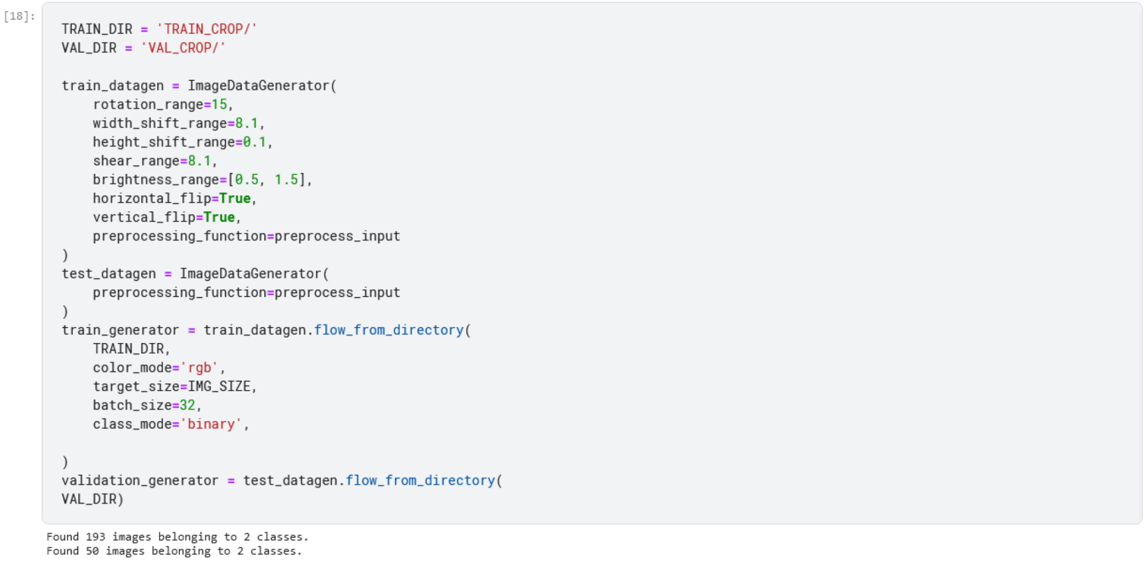
12. Adaptar tamaño



```
def preprocess_imgs(set_name, img_size):
    #Resiza y aplica preprocesamiento VGG-15
    set_new = []
    for img in set_name:
        img = cv2.resize(
            img,
            dsize=img_size,
            interpolation=cv2.INTER_CUBIC
        )
        set_new.append(preprocess_input(img) )
    return np.array(set_new)
```

```
X_train_prep = preprocess_imgs(set_name=X_train_crop, img_size=IMG_SIZE)
X_test_prep = preprocess_imgs(set_name=X_test_crop, img_size=IMG_SIZE)
X_val_prep = preprocess_imgs(set_name=X_val_crop, img_size=IMG_SIZE)
```

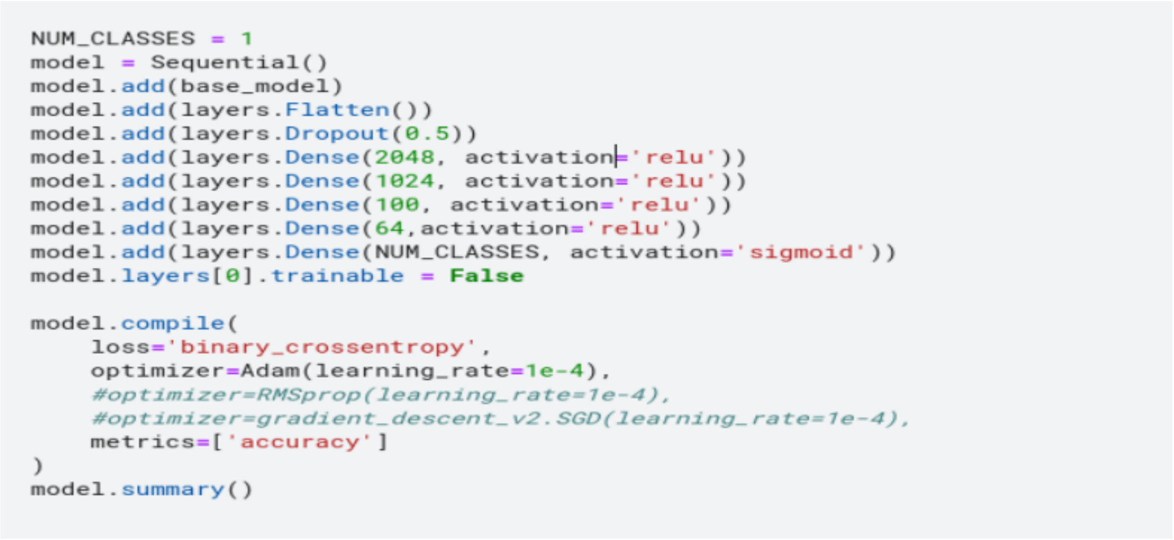
Data Augmentation



```
TRAIN_DIR = 'TRAIN_CROP/'
VAL_DIR = 'VAL_CROP/'

train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)
test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)
train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',

)
validation_generator = test_datagen.flow_from_directory(
VAL_DIR)
```

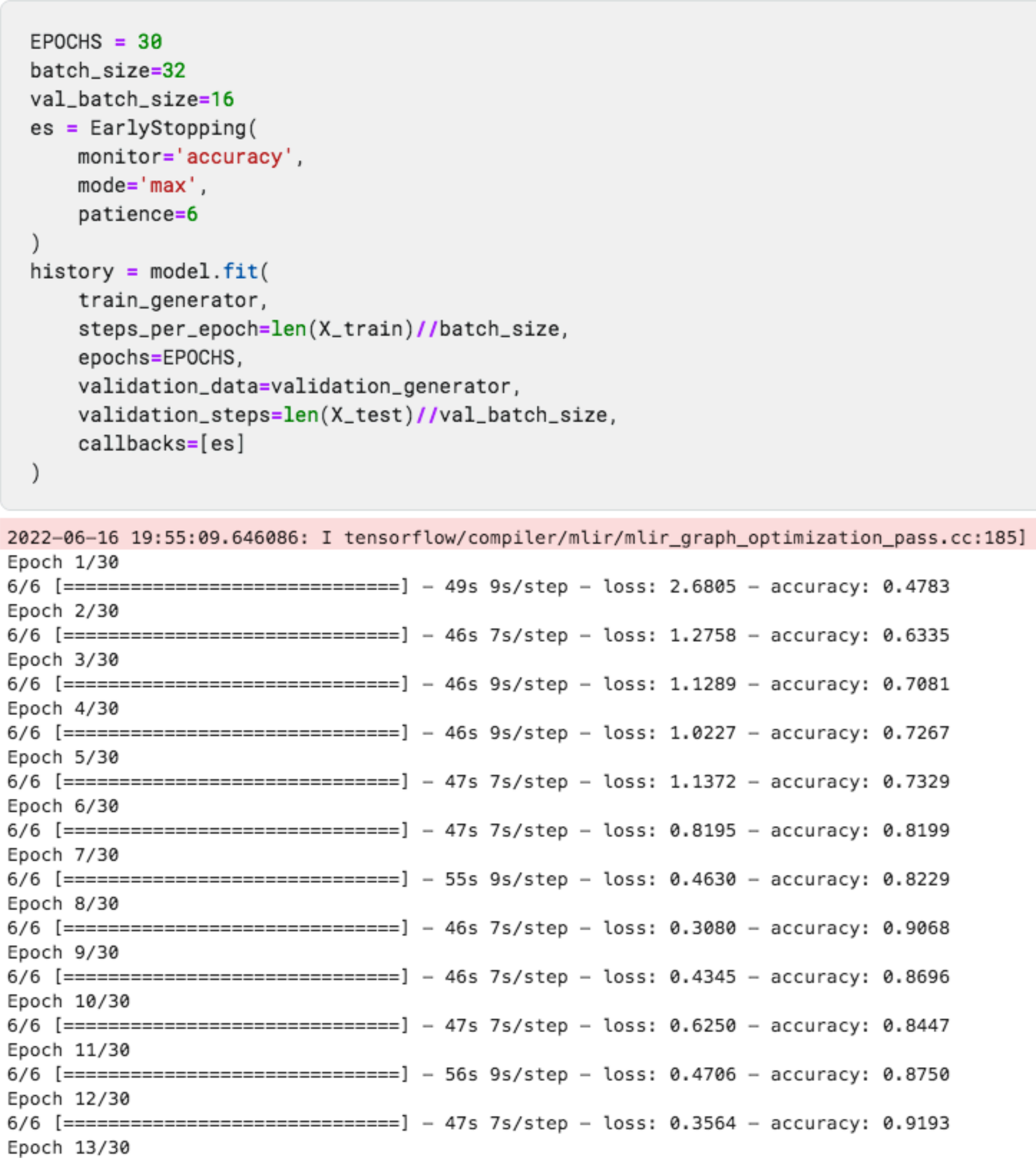
```
Found 193 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
```

13. Cargamos el modelo Base seleccionado



```
NUM_CLASSES = 1
model = Sequential()
model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(2048, activation='relu'))
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dense(100, activation='relu'))
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))
model.layers[0].trainable = False

model.compile(
    loss='binary_crossentropy',
    optimizer=Adam(learning_rate=1e-4),
    #optimizer=RMSprop(learning_rate=1e-4),
    #optimizer=gradient_descent_v2.SGD(learning_rate=1e-4),
    metrics=['accuracy']
)
model.summary()
```

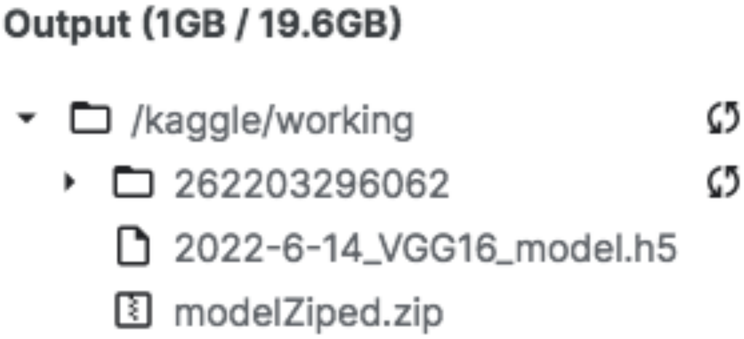
14. Entrenamiento del modelo



```
EPOCHS = 30
batch_size=32
val_batch_size=16
es = EarlyStopping(
    monitor='accuracy',
    mode='max',
    patience=6
)
history = model.fit(
    train_generator,
    steps_per_epoch=len(X_train)//batch_size,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=len(X_test)//val_batch_size,
    callbacks=[es]
)
```

```
2022-06-16 19:55:09.646086: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185]
Epoch 1/30
6/6 [==============================] - 49s 9s/step - loss: 2.6805 - accuracy: 0.4783
Epoch 2/30
6/6 [==============================] - 46s 7s/step - loss: 1.2758 - accuracy: 0.6335
Epoch 3/30
6/6 [==============================] - 46s 9s/step - loss: 1.1289 - accuracy: 0.7081
Epoch 4/30
6/6 [==============================] - 46s 9s/step - loss: 1.0227 - accuracy: 0.7267
Epoch 5/30
6/6 [==============================] - 47s 7s/step - loss: 1.1372 - accuracy: 0.7329
Epoch 6/30
6/6 [==============================] - 47s 7s/step - loss: 0.8195 - accuracy: 0.8199
Epoch 7/30
6/6 [==============================] - 55s 9s/step - loss: 0.4630 - accuracy: 0.8229
Epoch 8/30
6/6 [==============================] - 46s 7s/step - loss: 0.3080 - accuracy: 0.9068
Epoch 9/30
6/6 [==============================] - 46s 7s/step - loss: 0.4345 - accuracy: 0.8696
Epoch 10/30
6/6 [==============================] - 47s 7s/step - loss: 0.6250 - accuracy: 0.8447
Epoch 11/30
6/6 [==============================] - 56s 9s/step - loss: 0.4706 - accuracy: 0.8750
Epoch 12/30
6/6 [==============================] - 47s 7s/step - loss: 0.3564 - accuracy: 0.9193
Epoch 13/30
```

15. Descarga del modelo



Output (1GB / 19.6GB)

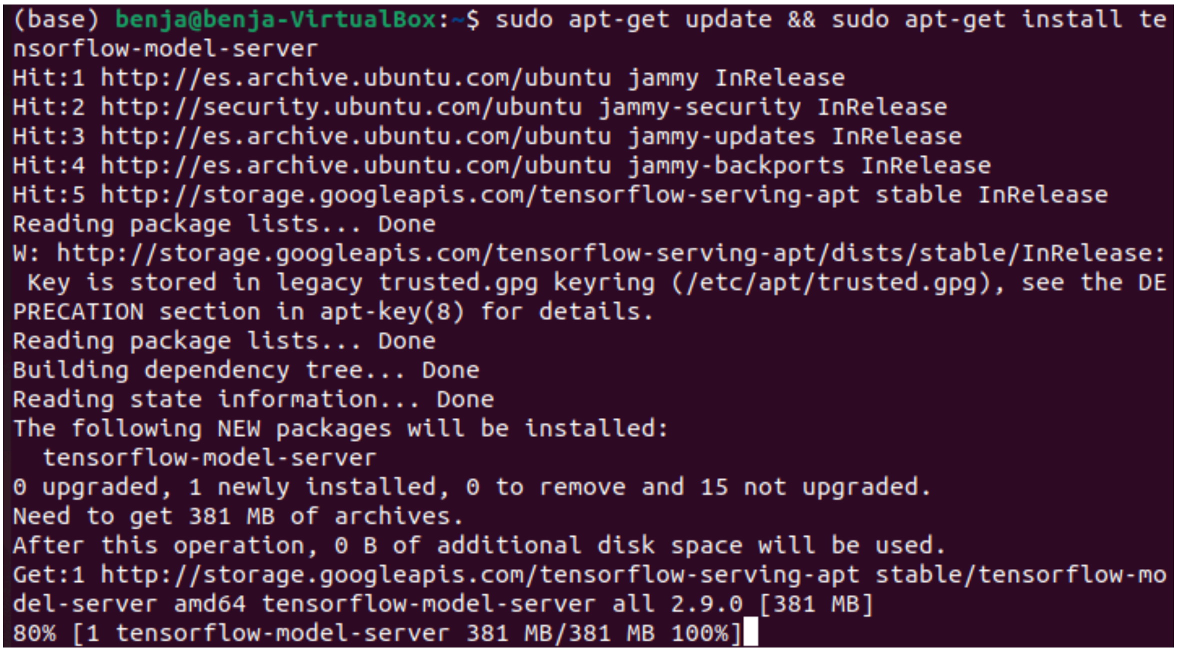
/kaggle/working

262203296062

2022-6-14_VGG16_model.h5

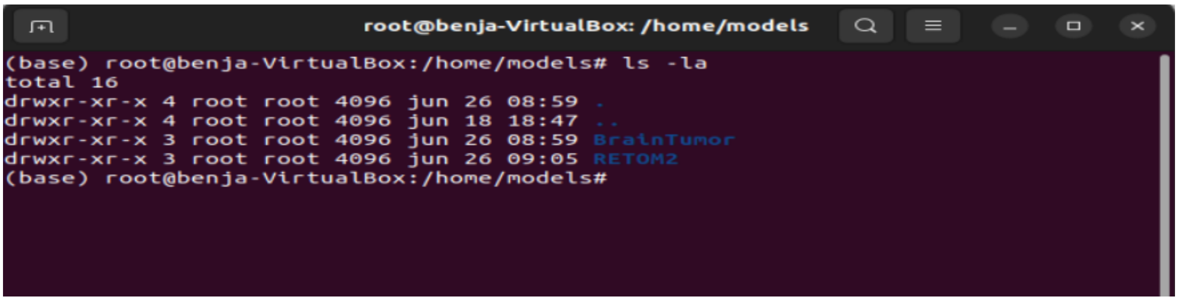
modelZiped.zip

16. Instalación Tensorflow



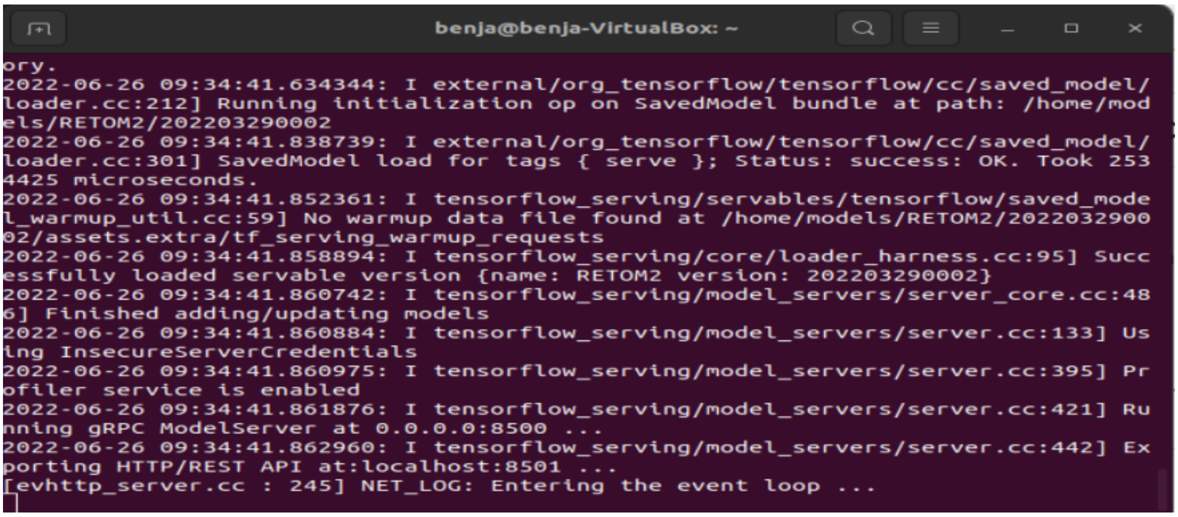
```
(base) benja@benja-VirtualBox:~$ sudo apt-get update && sudo apt-get install te
nsorflow-model-server
Hit:1 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:5 http://storage.googleapis.com/tensorflow-serving-apt stable InRelease
Reading package lists... Done
W: http://storage.googleapis.com/tensorflow-serving-apt/dists/stable/InRelease:
 Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DE
PRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  tensorflow-model-server
0 upgraded, 1 newly installed, 0 to remove and 15 not upgraded.
Need to get 381 MB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://storage.googleapis.com/tensorflow-serving-apt stable/tensorflow-mo
del-server amd64 tensorflow-model-server all 2.9.0 [381 MB]
80% [1 tensorflow-model-server 381 MB/381 MB 100%]
```

17. Creación carpeta



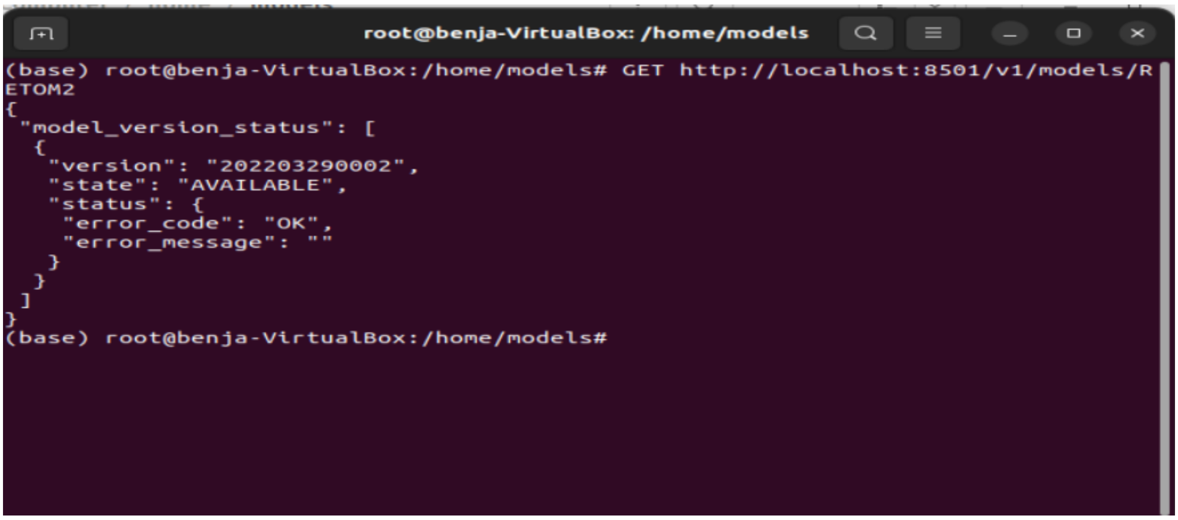
```
(base) root@benja-VirtualBox:/home/models# ls -la
total 16
drwxr-xr-x 4 root root 4096 jun 26 08:59 .
drwxr-xr-x 4 root root 4096 jun 18 18:47 ..
drwxr-xr-x 3 root root 4096 jun 26 08:59 BrainTumor
drwxr-xr-x 3 root root 4096 jun 26 09:05 RETOM2
(base) root@benja-VirtualBox:/home/models#
```

18. TensorFlow.



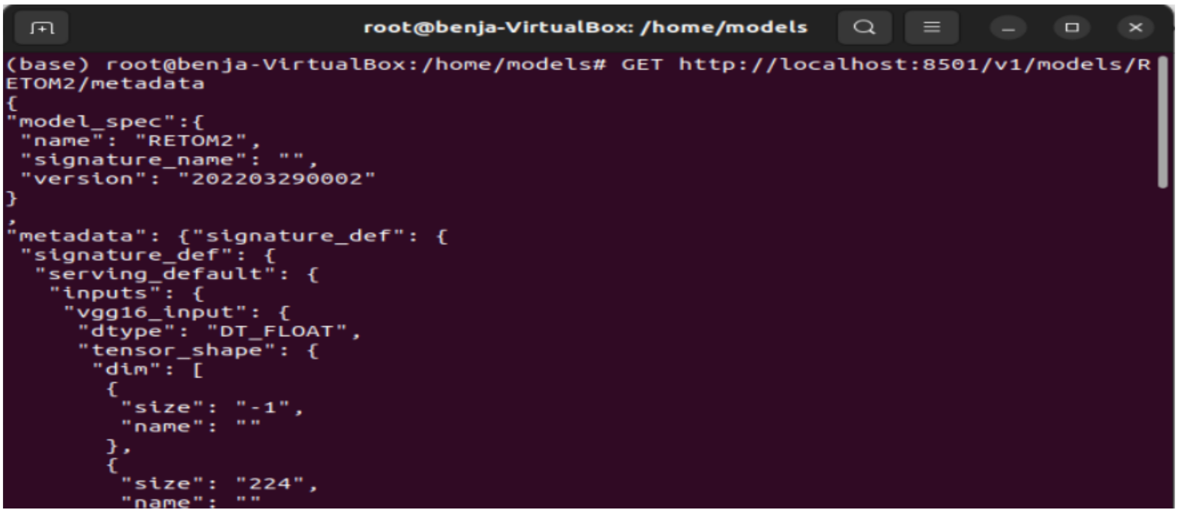
```
ory.
2022-06-26 09:34:41.634344: I external/org_tensorflow/tensorflow/cc/saved_model/
loader.cc:212] Running initialization op on SavedModel bundle at path: /home/mod
els/RETOM2/202203290002
2022-06-26 09:34:41.838739: I external/org_tensorflow/tensorflow/cc/saved_model/
loader.cc:301] SavedModel load for tags { serve }; Status: success: OK. Took 253
4425 microseconds.
2022-06-26 09:34:41.852361: I tensorflow_serving/servables/tensorflow/saved_mode
l_warmup_util.cc:59] No warmup data file found at /home/models/RETOM2/2022032900
02/assets.extra/tf_serving_warmup_requests
2022-06-26 09:34:41.858894: I tensorflow_serving/core/loader_harness.cc:95] Succ
essfully loaded servable version {name: RETOM2 version: 202203290002}
2022-06-26 09:34:41.860742: I tensorflow_serving/model_servers/server_core.cc:48
6] Finished adding/updating models
2022-06-26 09:34:41.860884: I tensorflow_serving/model_servers/server.cc:133] Us
ing InsecureServerCredentials
2022-06-26 09:34:41.860975: I tensorflow_serving/model_servers/server.cc:395] Pr
ofiler service is enabled
2022-06-26 09:34:41.861876: I tensorflow_serving/model_servers/server.cc:421] Ru
nning gRPC ModelServer at 0.0.0.0:8500 ...
2022-06-26 09:34:41.862960: I tensorflow_serving/model_servers/server.cc:442] Ex
porting HTTP/REST API at:localhost:8501 ...
[evhttp_server.cc : 245] NET_LOG: Entering the event loop ...
```

19. GET para comprobar modelo



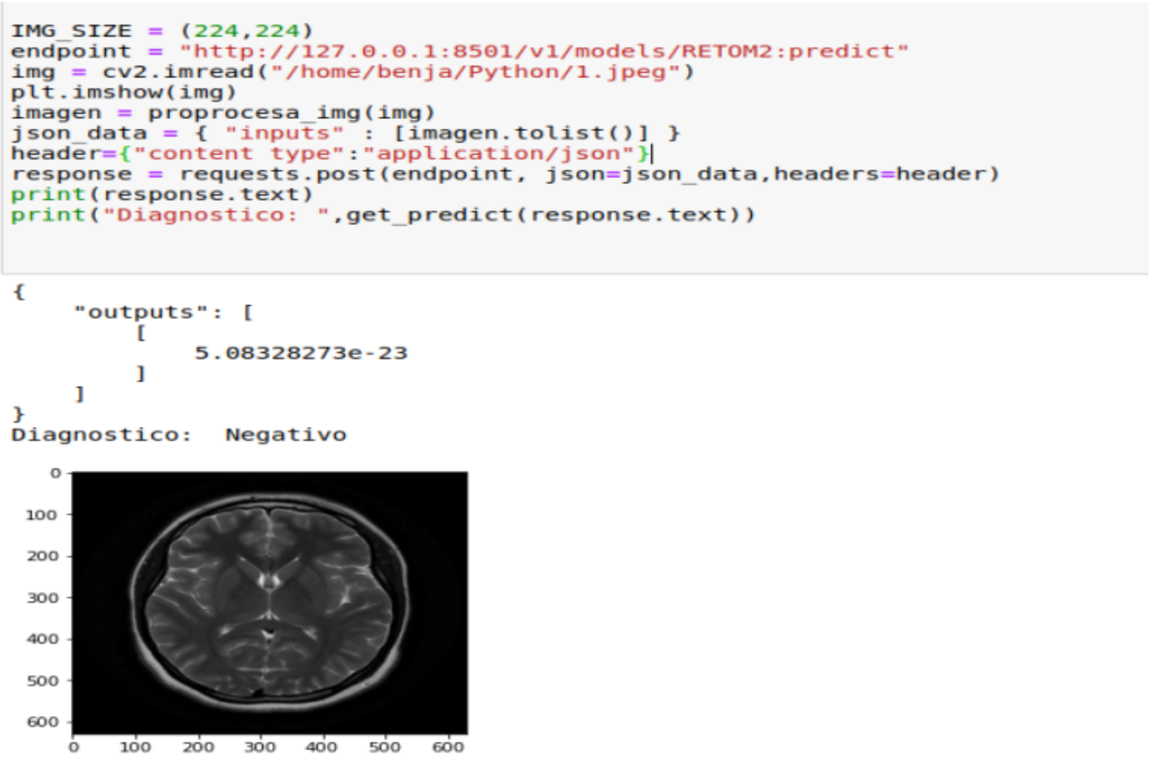
```
(base) root@benja-VirtualBox:/home/models# GET http://localhost:8501/v1/models/R
ETOM2
{
 "model_version_status": [
  {
   "version": "202203290002",
   "state": "AVAILABLE",
   "status": {
    "error_code": "OK",
    "error_message": ""
   }
  }
 ]
}
(base) root@benja-VirtualBox:/home/models#
```

20. GET Metadatos



```
(base) root@benja-VirtualBox:/home/models# GET http://localhost:8501/v1/models/R
ETOM2/metadata
{
"model_spec":{
 "name": "RETOM2",
 "signature_name": "",
 "version": "202203290002"
}
,
"metadata": {"signature_def": {
 "signature_def": {
  "serving_default": {
   "inputs": {
    "vgg16_input": {
     "dtype": "DT_FLOAT",
     "tensor_shape": {
      "dim": [
       {
        "size": "-1",
        "name": ""
       },
       {
        "size": "224",
        "name": ""
```

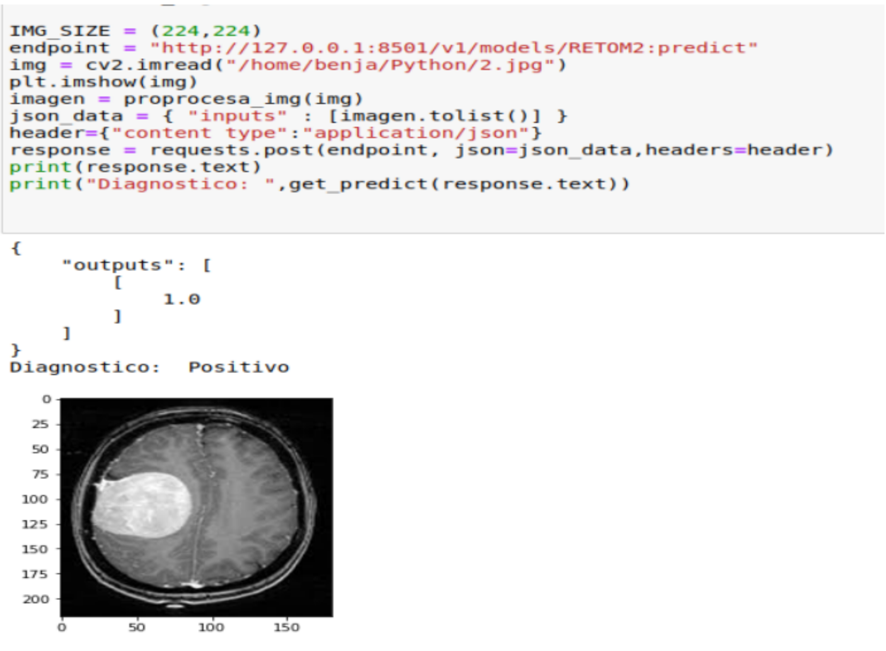
21. Predicción Negativa



```
IMG_SIZE = (224,224)
endpoint = "http://127.0.0.1:8501/v1/models/RETOM2:predict"
img = cv2.imread("/home/benja/Python/1.jpeg")
plt.imshow(img)
imagen = proprocesa_img(img)
json_data = { "inputs" : [imagen.tolist()] }
header={"content type":"application/json"}
response = requests.post(endpoint, json=json_data,headers=header)
print(response.text)
print("Diagnostico: ",get_predict(response.text))
```

```
{
    "outputs": [
        [
            5.08328273e-23
        ]
    ]
}
Diagnostico:  Negativo
```

22. Predicción Positiva



```
IMG_SIZE = (224,224)
endpoint = "http://127.0.0.1:8501/v1/models/RETOM2:predict"
img = cv2.imread("/home/benja/Python/2.jpg")
plt.imshow(img)
imagen = proprocesa_img(img)
json_data = { "inputs" : [imagen.tolist()] }
header={"content type":"application/json"}
response = requests.post(endpoint, json=json_data,headers=header)
print(response.text)
print("Diagnostico: ",get_predict(response.text))
```

```
{
    "outputs": [
        [
            1.0
        ]
    ]
}
Diagnostico:  Positivo
```