**Como empezar con un entrenamiento para la red neuronal**

**Necesitamos los siguientes requerimientos:**

* **Python 3.5**
* **Tensorflow 1.12 – 1.14**

**Para empezar vamos a seguir los siguientes pasos:**

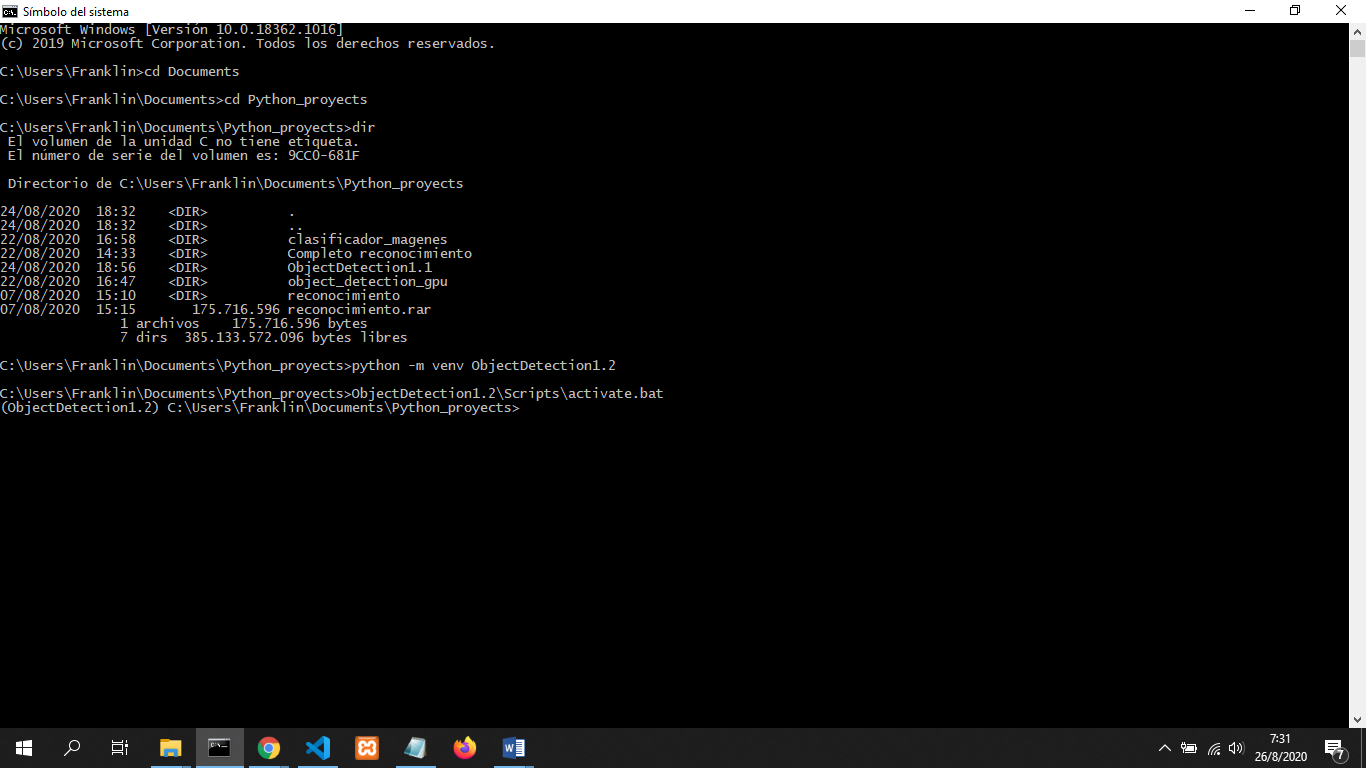
1. **Crear un entorno virtual**

* python -m venv NombreEntornoVirtual

1. **Acceder al entorno virtual**

* ObjectDetection1.2\Scripts\activate.bat

Para identificar que estamos dentro del entorno virtual



1. **Actualizar el paquete pip para poder ejecutar los comandos sin ningún problema**

* python -m pip install --upgrade pip

1. **Instalar todos los paquetes necesarios**

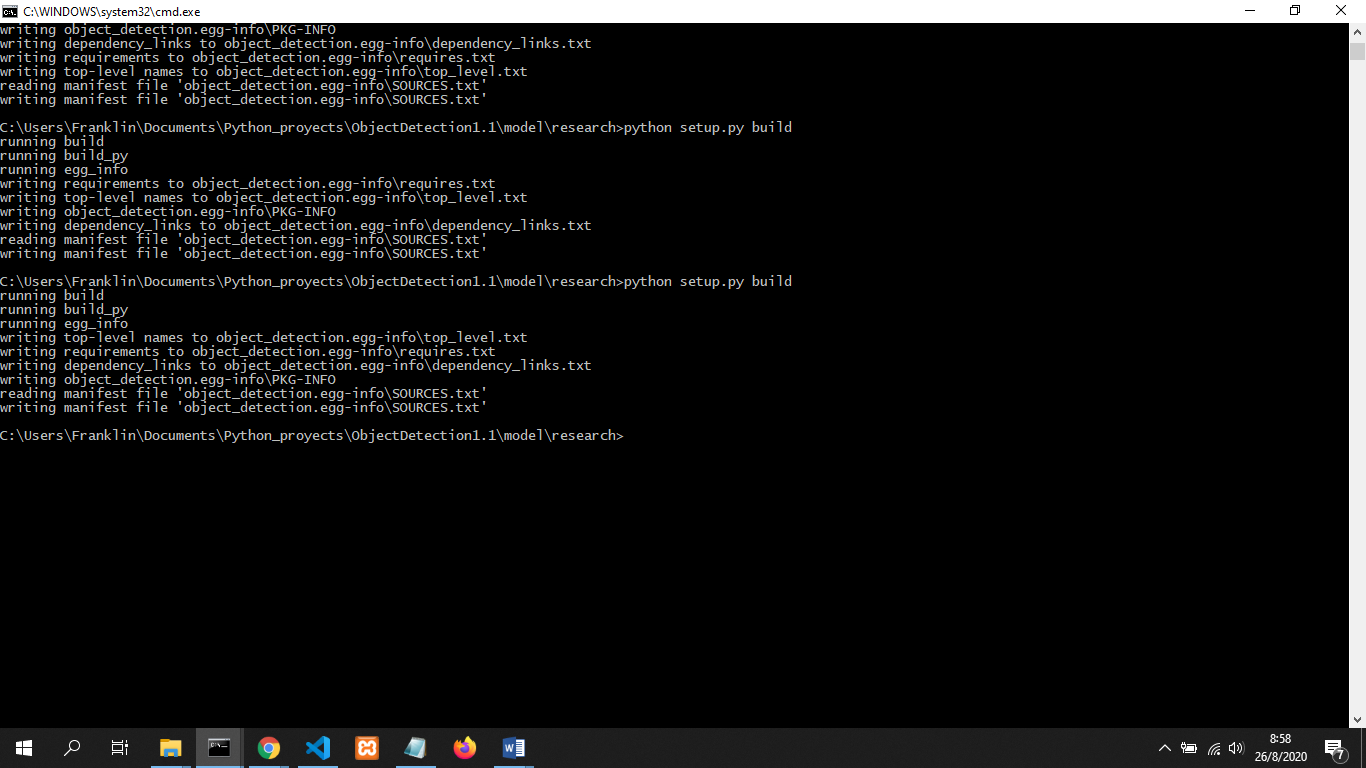
* pip install tensorflow==1.14
* pip install protobuf
* pip install pillow
* pip install lxml
* pip install Cython
* pip install contextlib2
* pip install matplotlib
* pip install pandas
* pip install opencv-python
* pip install numpy

1. **Añadir al PATH las variables de entorno en donde se encuentra nuestro entorno virtual**

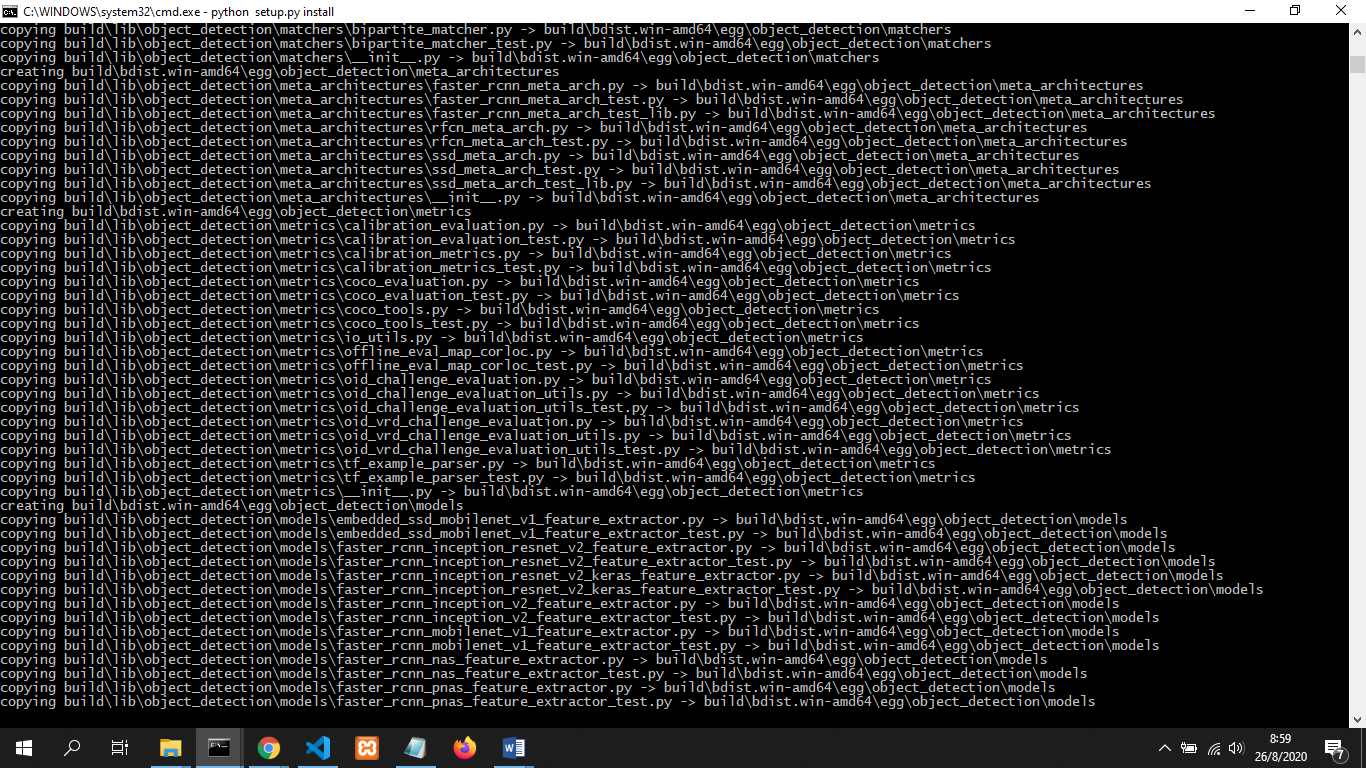
set PYTHONPATH=C:\Users\Franklin\Documents\Python\_proyects\ObjectDetection1.2\modelC:\Users\Franklin\Documents\Python\_proyects\ObjectDetection1.2\model\research;C:\Users\Franklin\Documents\Python\_proyects\ObjectDetection1.2\model\research\slim

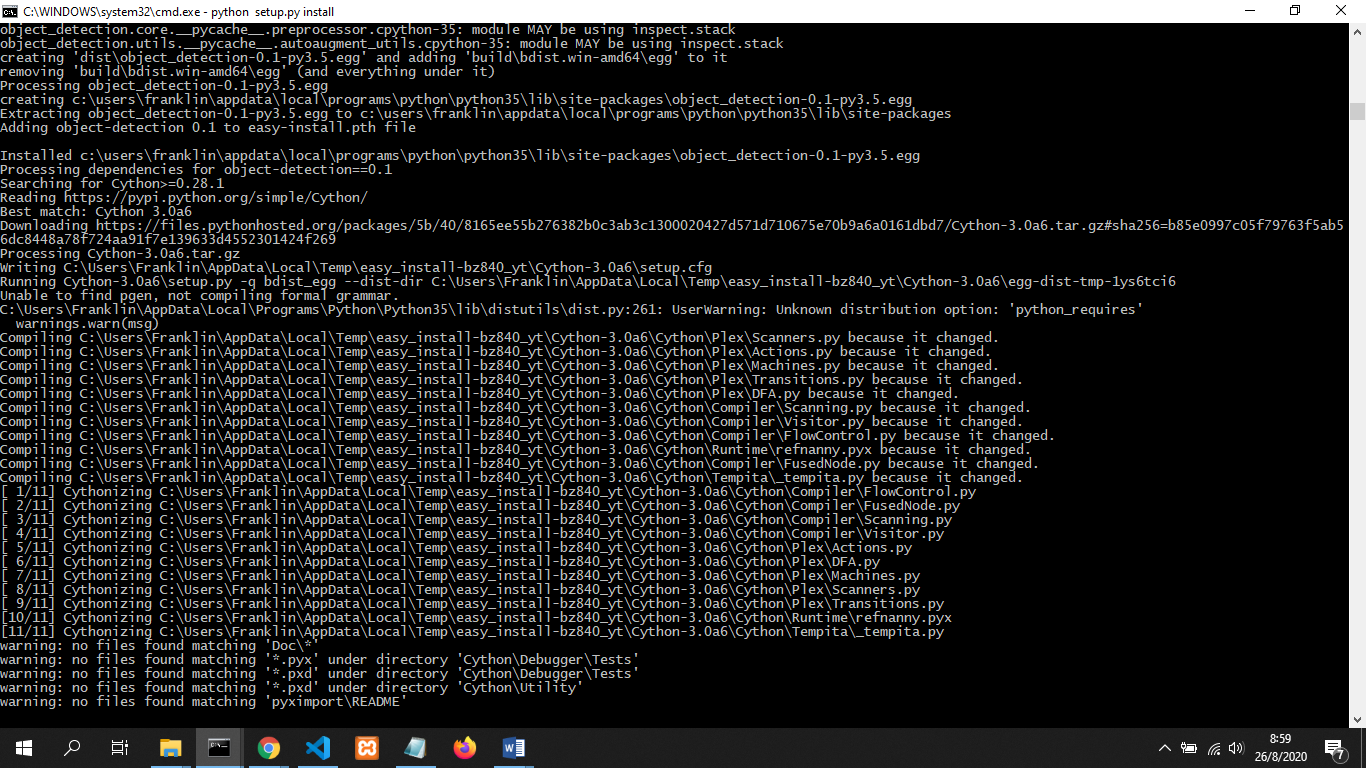
1. **Dentro de la carpeta model/research ejecutar los siguiente comandos**

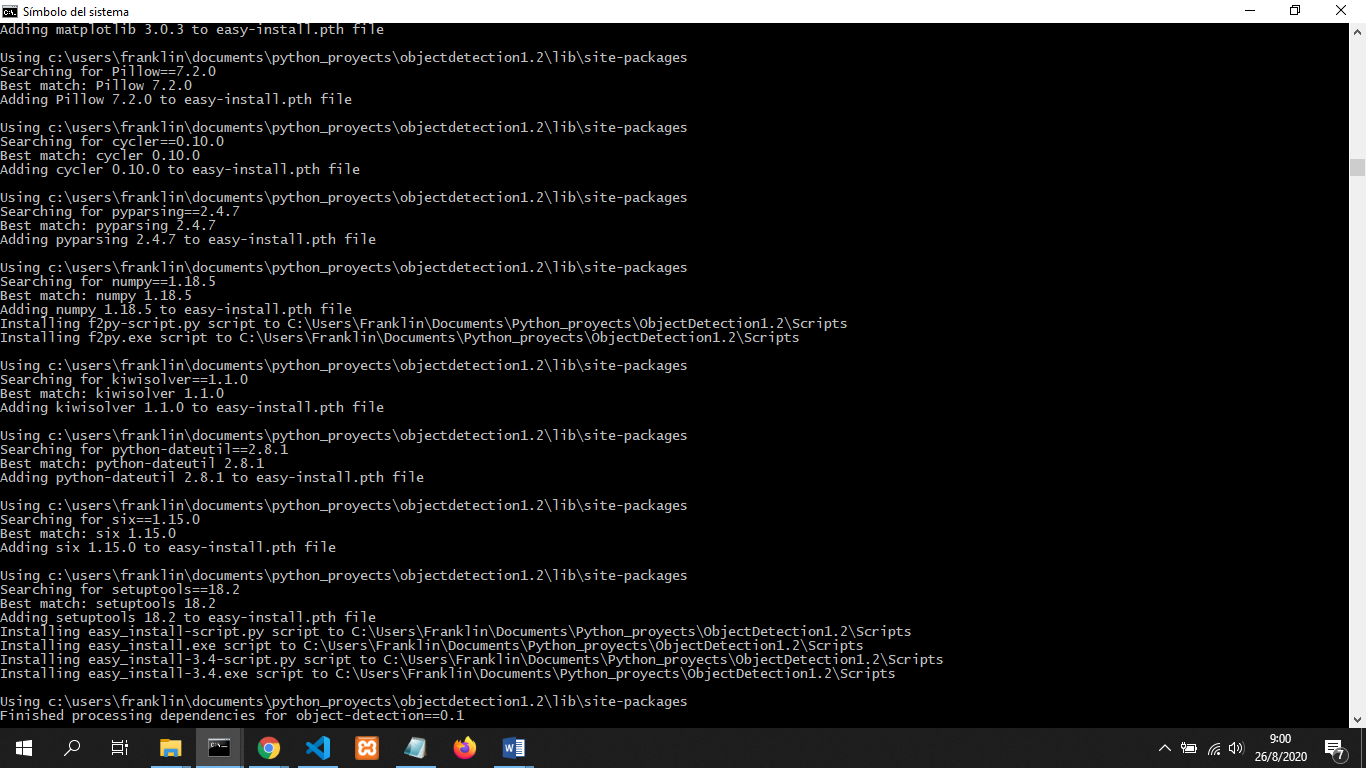
* python setup.py build



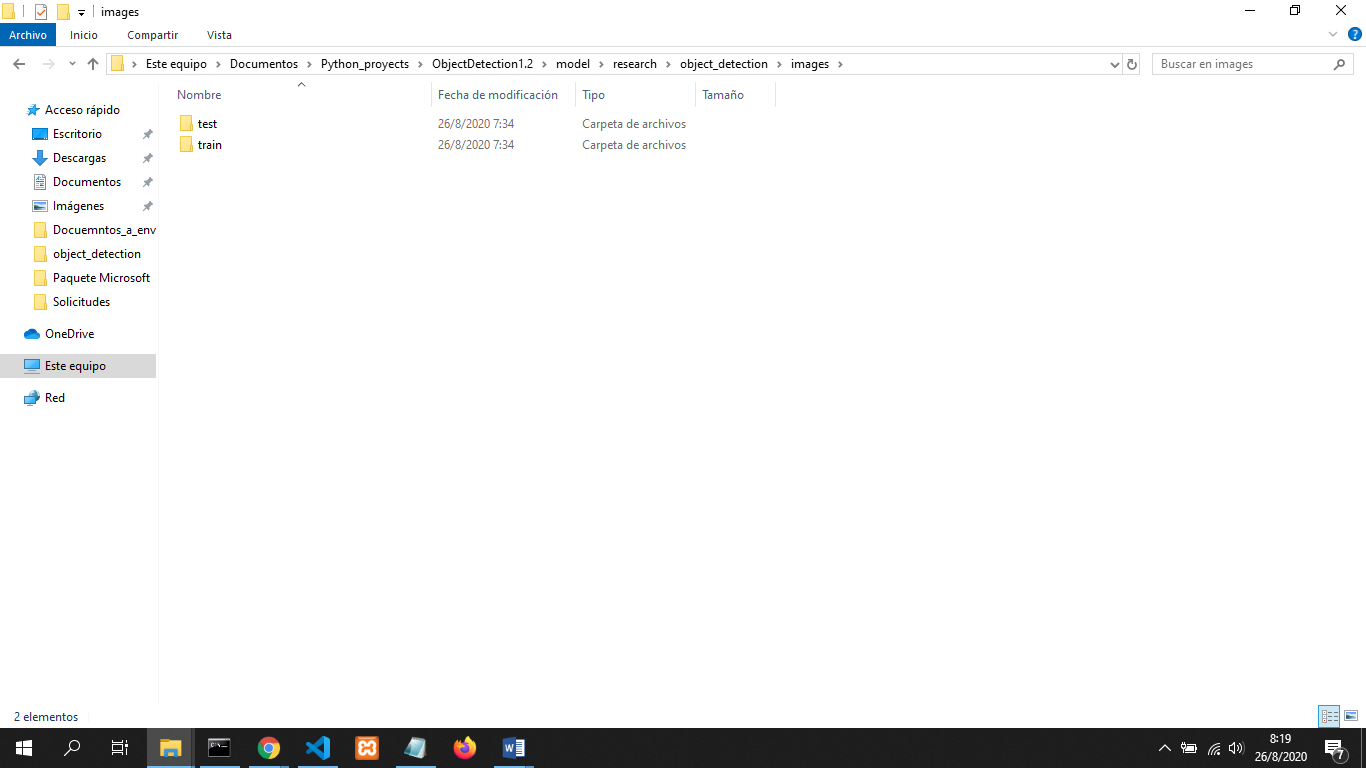
* python setup.py install

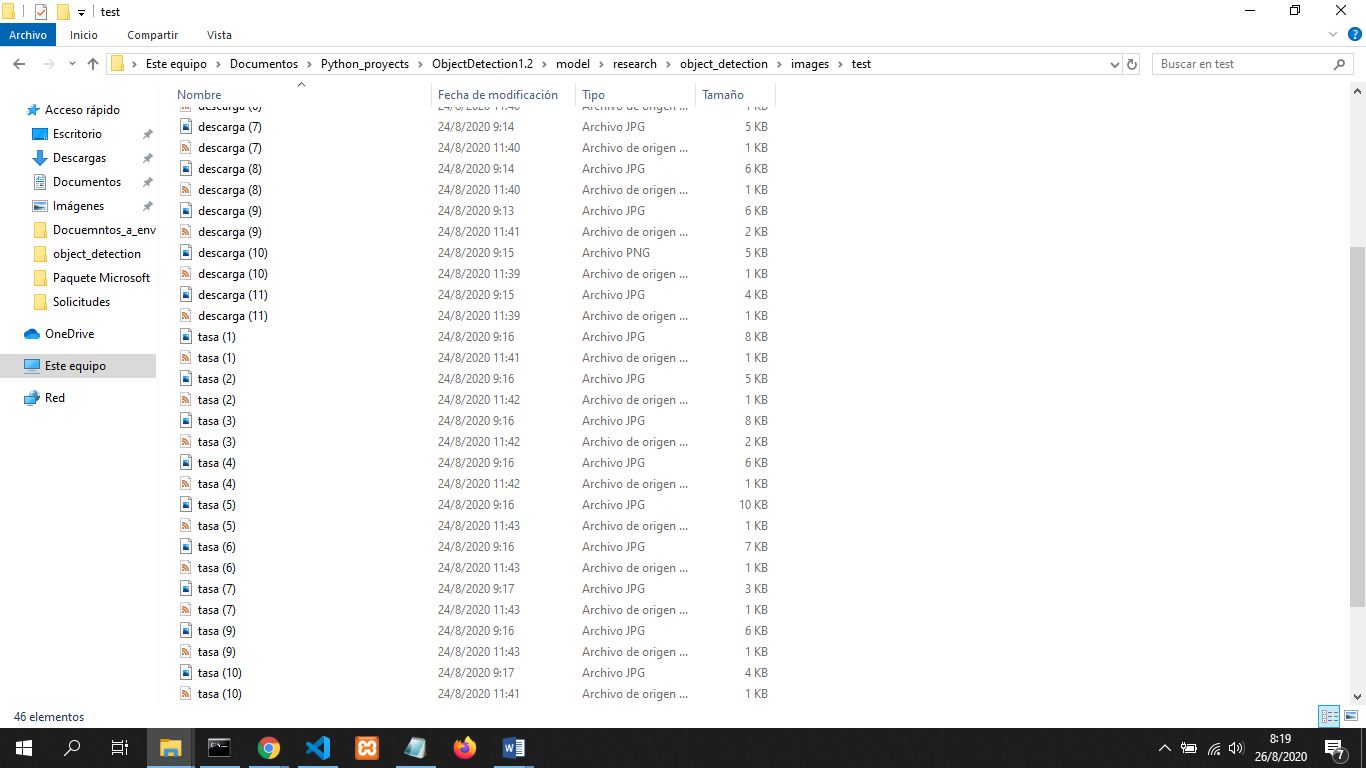






1. **Copiamos nuestras imágenes procesadas con labelimg dentro de las carpetas train y test, Todas las imágenes sin importar el objeto que contengan.**

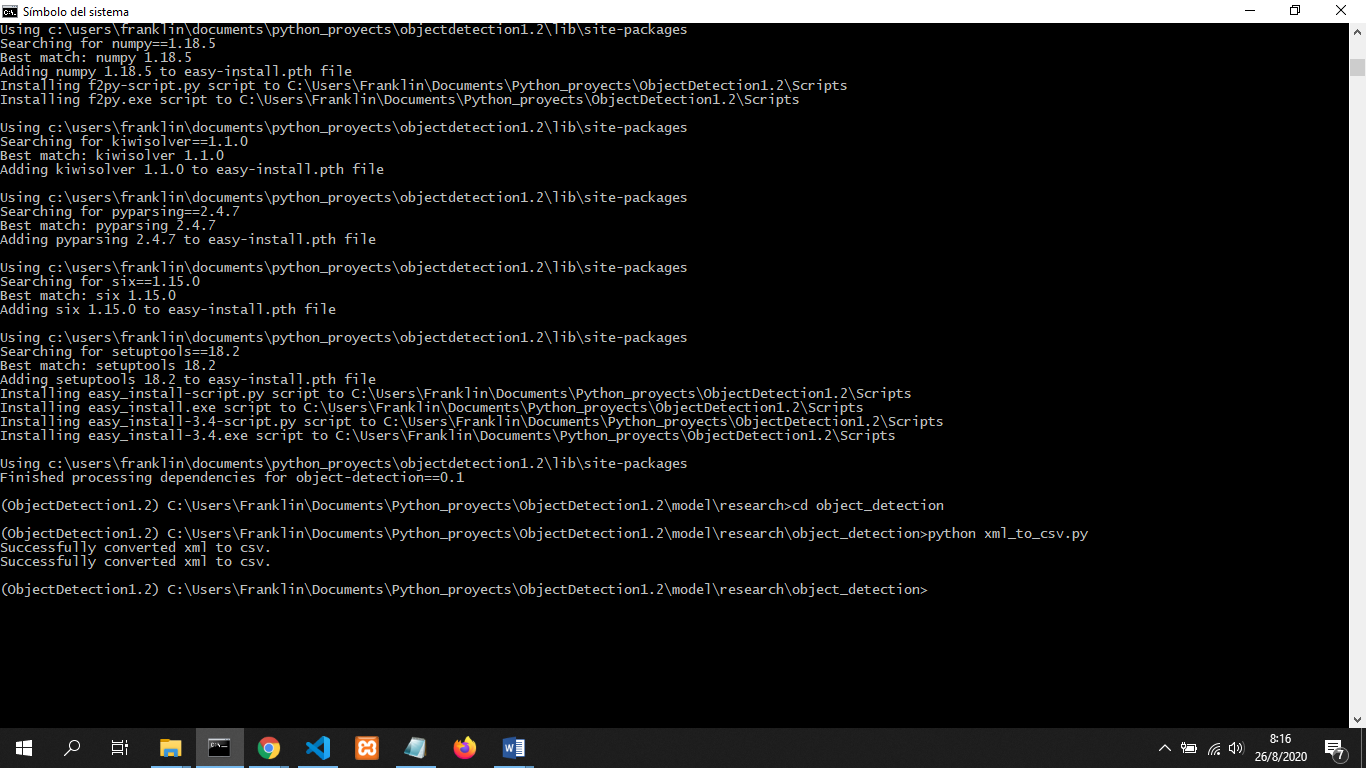


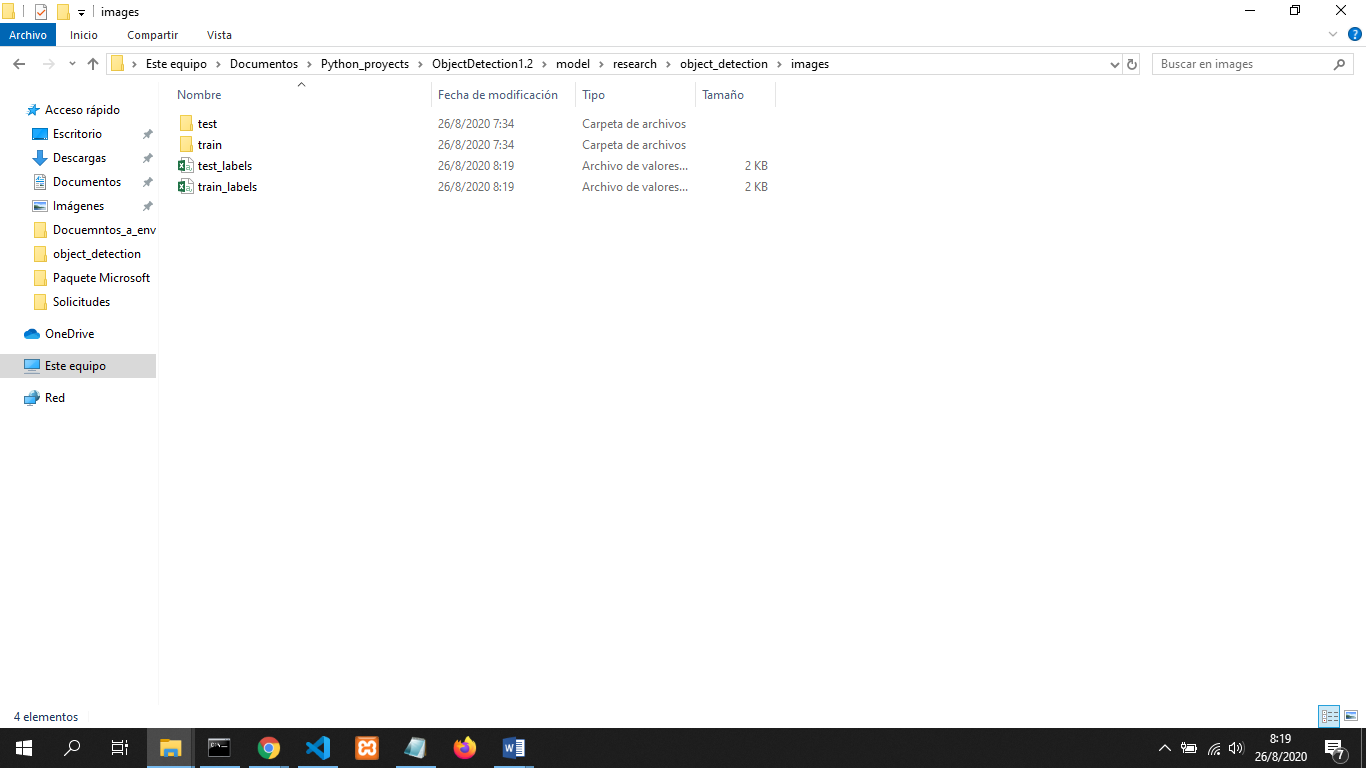


1. **Dentro de la carpeta model/research/object\_detection ejecutamos los siguientes comandos.**

* python xml\_to\_csv.py

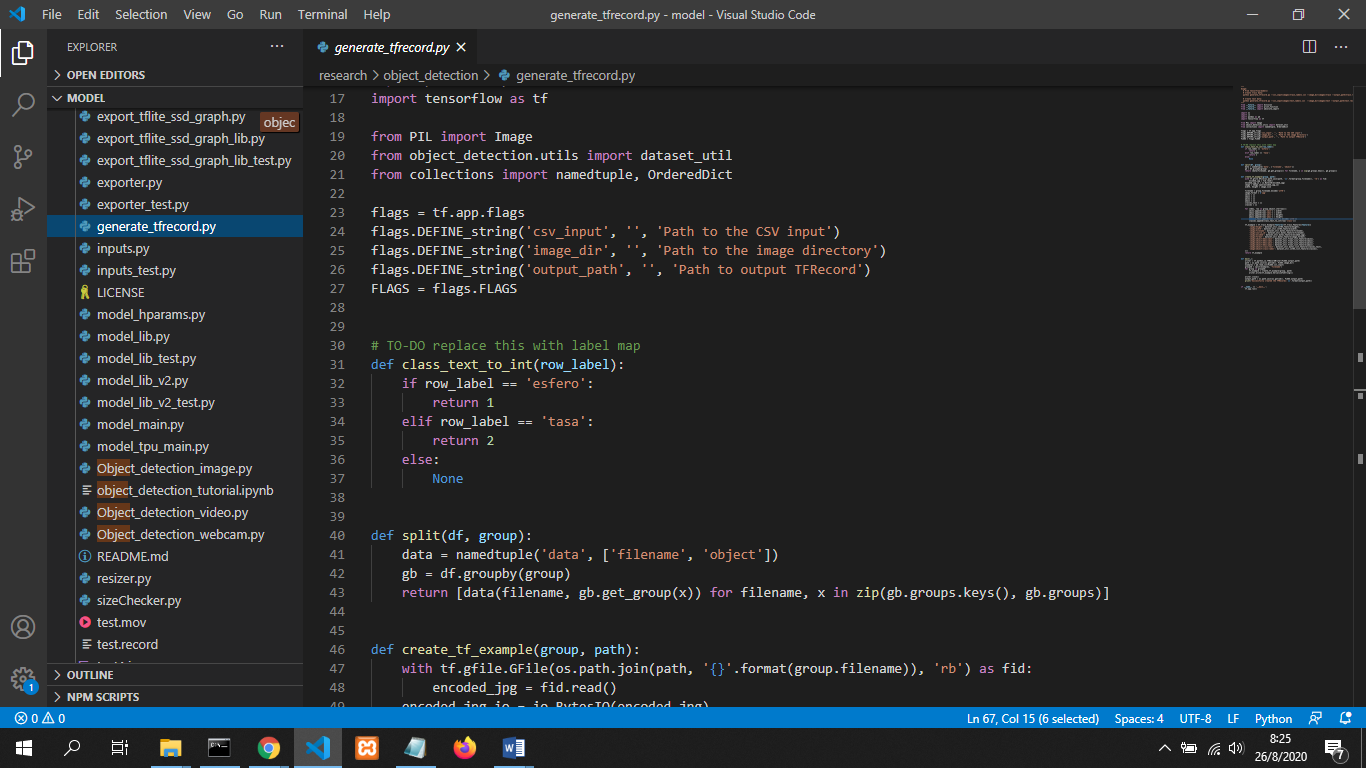
Esto nos convertirá los xml a un formato entendible para la maquina





Esto nos dejara dos archivos en formato csv o formato de Excel.

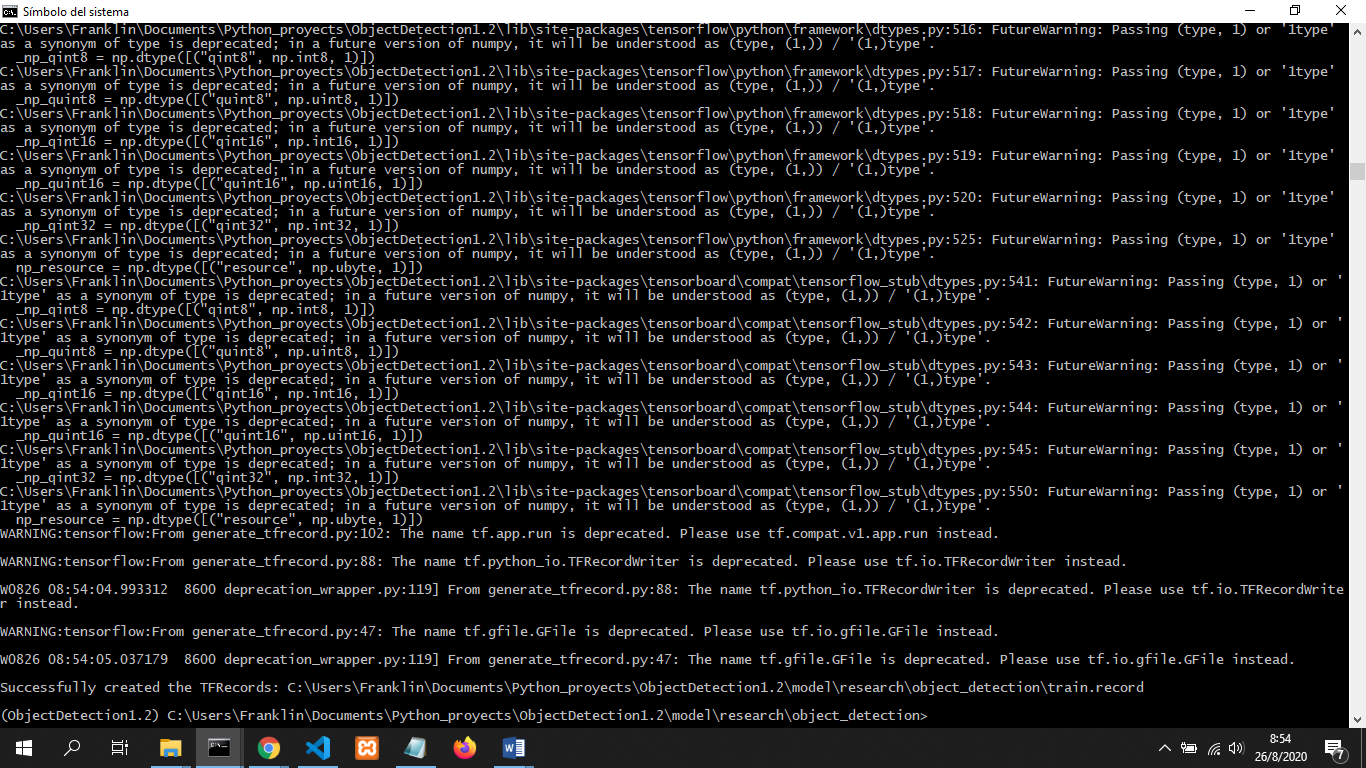
1. **Modificamos el archivo *generate\_tfrecord.py* que está ubicado dentro de la carpeta model/research/object\_detection por el número de objetos que queremos entrenar. Ejemplo: Si queremos identificar una tasa y un esfero los colocamos dentro de este archivo de la siguiente manera:**



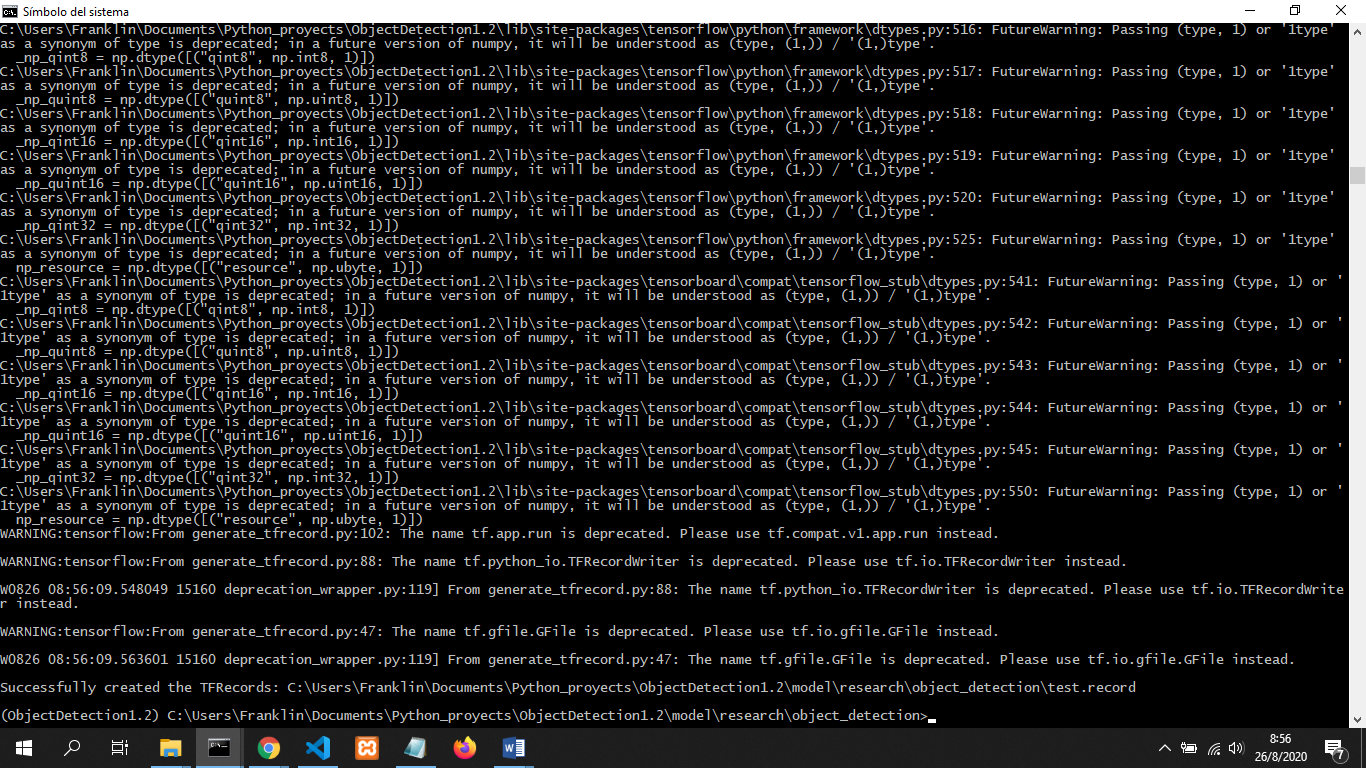
Este nombre debe coincidir con los label que asignamos a nuestras imágenes al procesarlas con labelimg.

1. **Con esta configuración ya podemos ejecutar el siguiente comando**

* python generate\_tfrecord.py --csv\_input=images\train\_labels.csv --image\_dir=images\train --output\_path=train.record



* python generate\_tfrecord.py --csv\_input=images\test\_labels.csv --image\_dir=images\test --output\_path=test.record



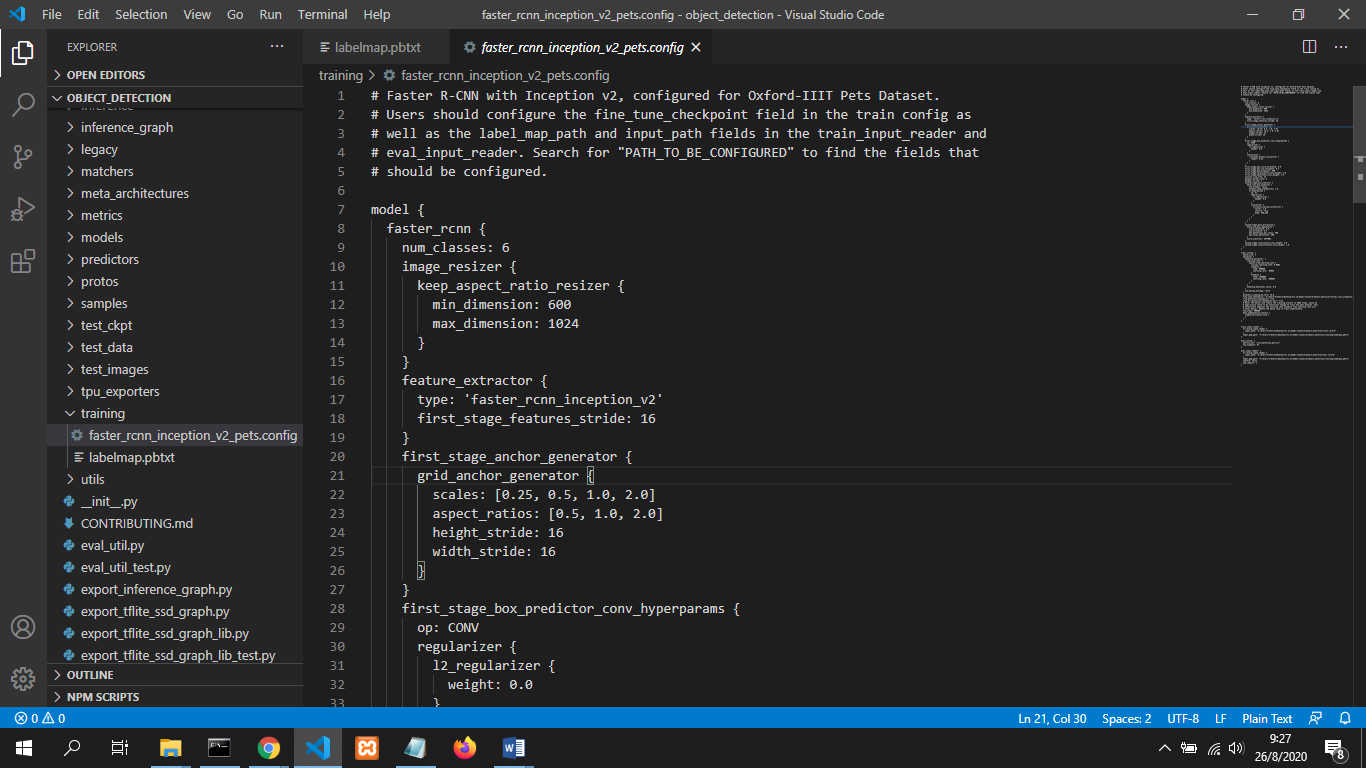
Esto nos generará los archivos con extencion .record (train.record y test.record) estos archivos se obtuvieron de los csv de la carpeta de imágenes, es decir a los labels de las imágenes las procesamos 2 veces.

1. **Modificamos los archivos *faster\_rcnn\_inception\_v2\_pets.config y labelmap.pbtxt* que están ubicados dentro de model/research/object\_detection/training con la siguiente configuración:**

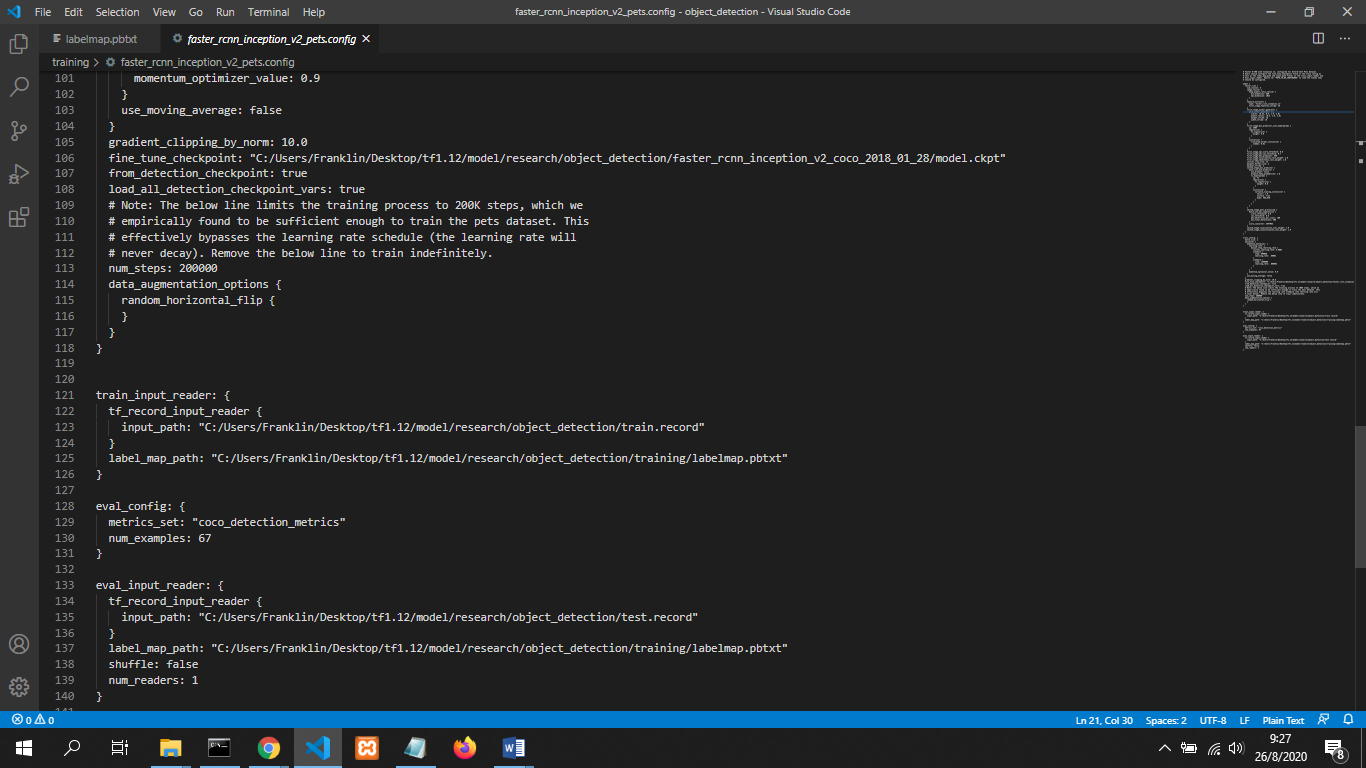
***faster\_rcnn\_inception\_v2\_pets.config***

Modificamos la variable **num\_classes** con el numero de objetos que vamos a entrenar.

Ejemplo: si vamos a entrenar taza y esfero indicamos que son 2 clases

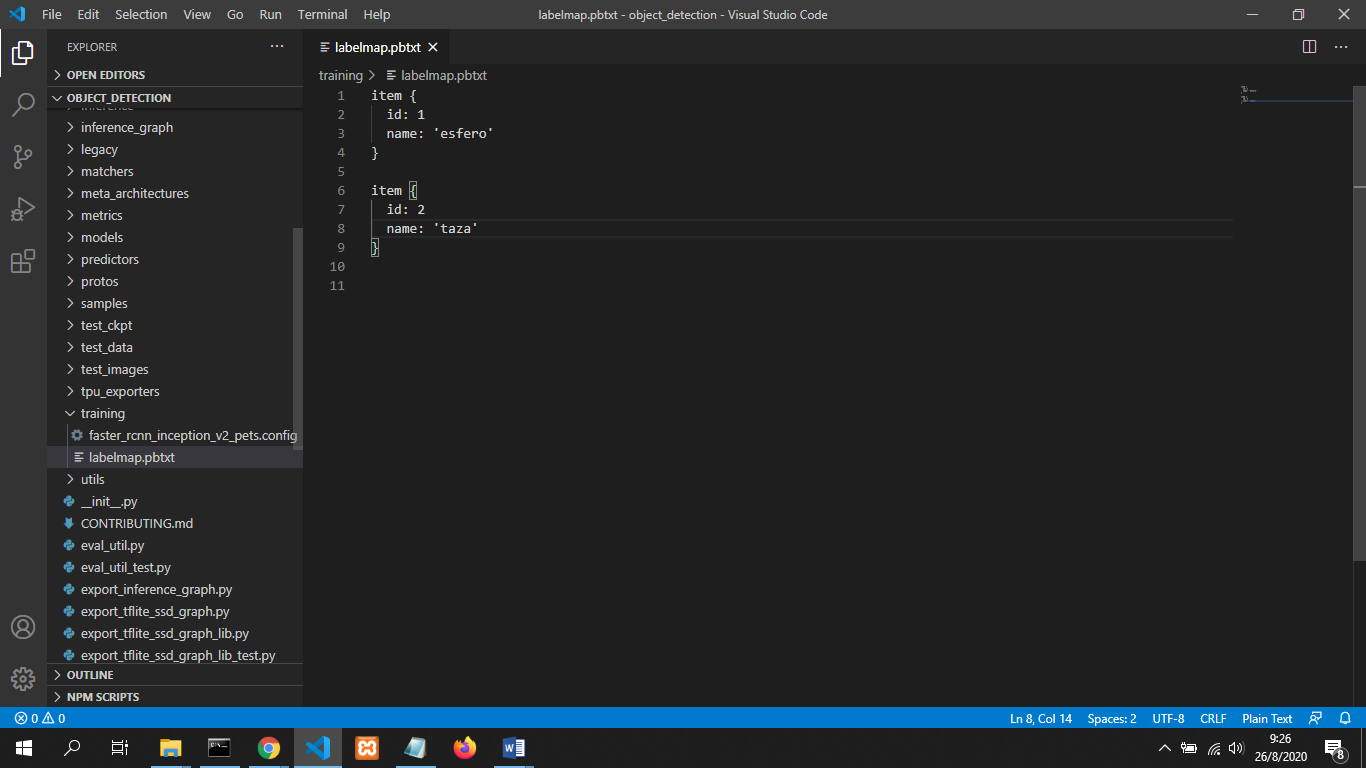


Modificamos todos los path con los que corresponden a nuestra carpeta



***labelmap.pbtxt***

Indicamos el nombre de nuestros objetos al igual que en el **paso 9**



1. **Ejecutamos el entrenamiento con el siguiente comando, esto nos dará el número de pasos que está ejecutando y la perdida que está recibiendo, está perdida (loss) debe ser muy baja entre 1 y 0.x, si supera este número algo está mal en los pasos anteriores. Para ejecutar el entrenamiento ejecutamnos el comnando dentro de model/research/object\_detection:**

* python legacy/train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/faster\_rcnn\_inception\_v2\_pets.config