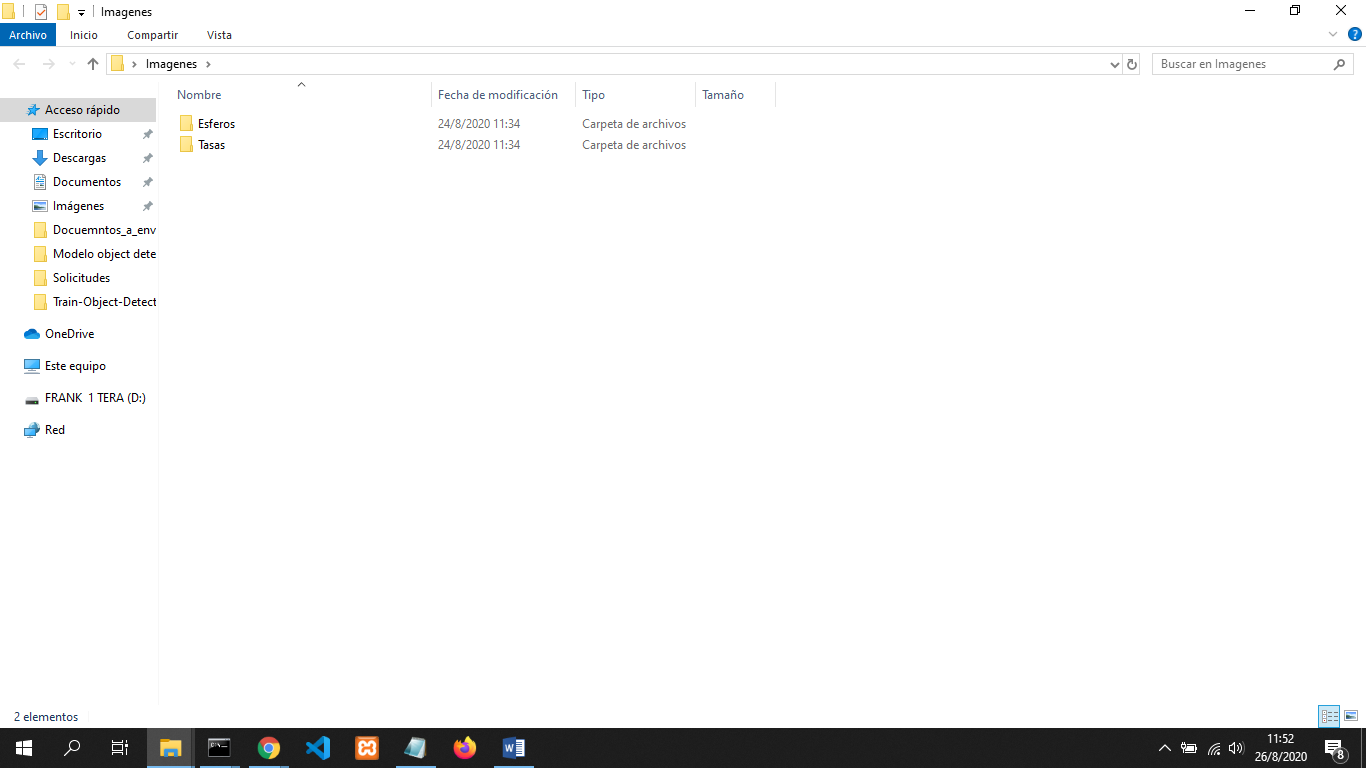
**Como empezar con un entrenamiento para la red neuronal**

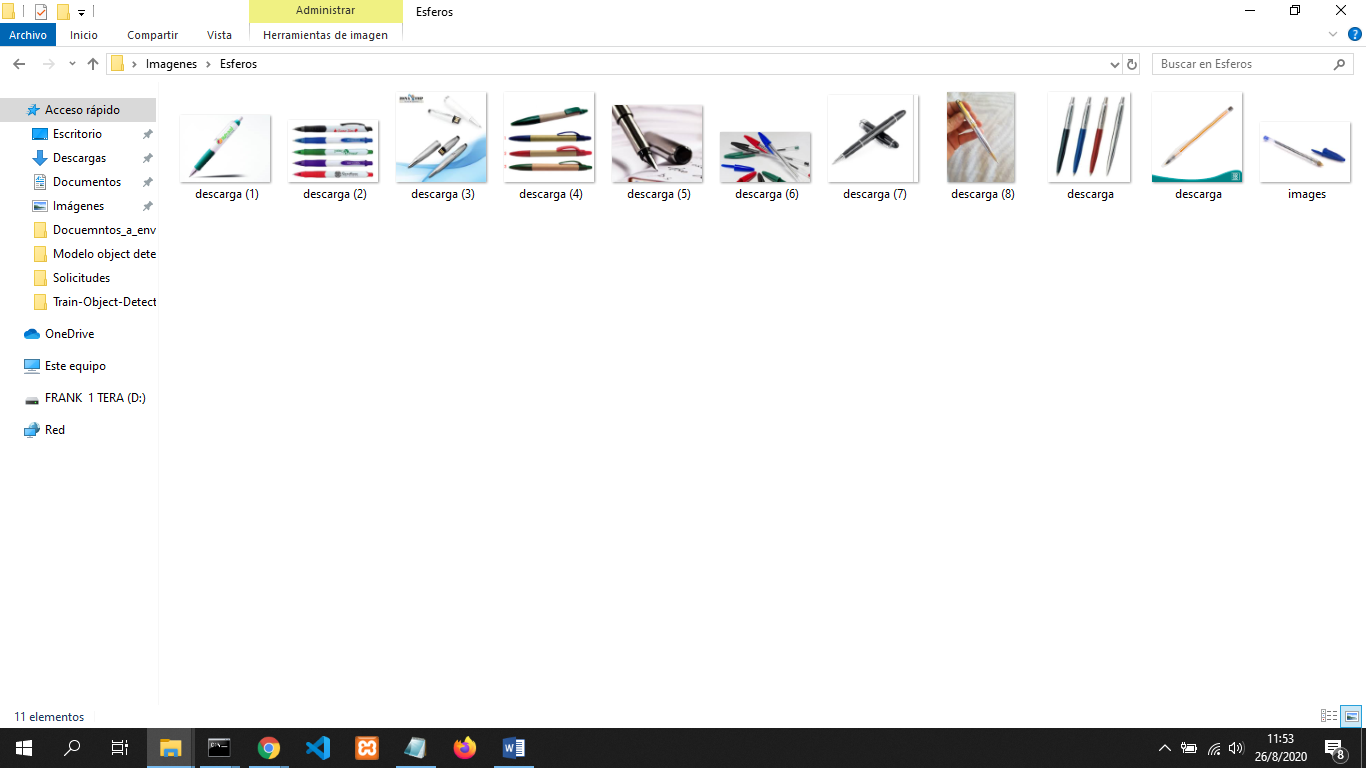
**Necesitamos los siguientes requerimientos:**

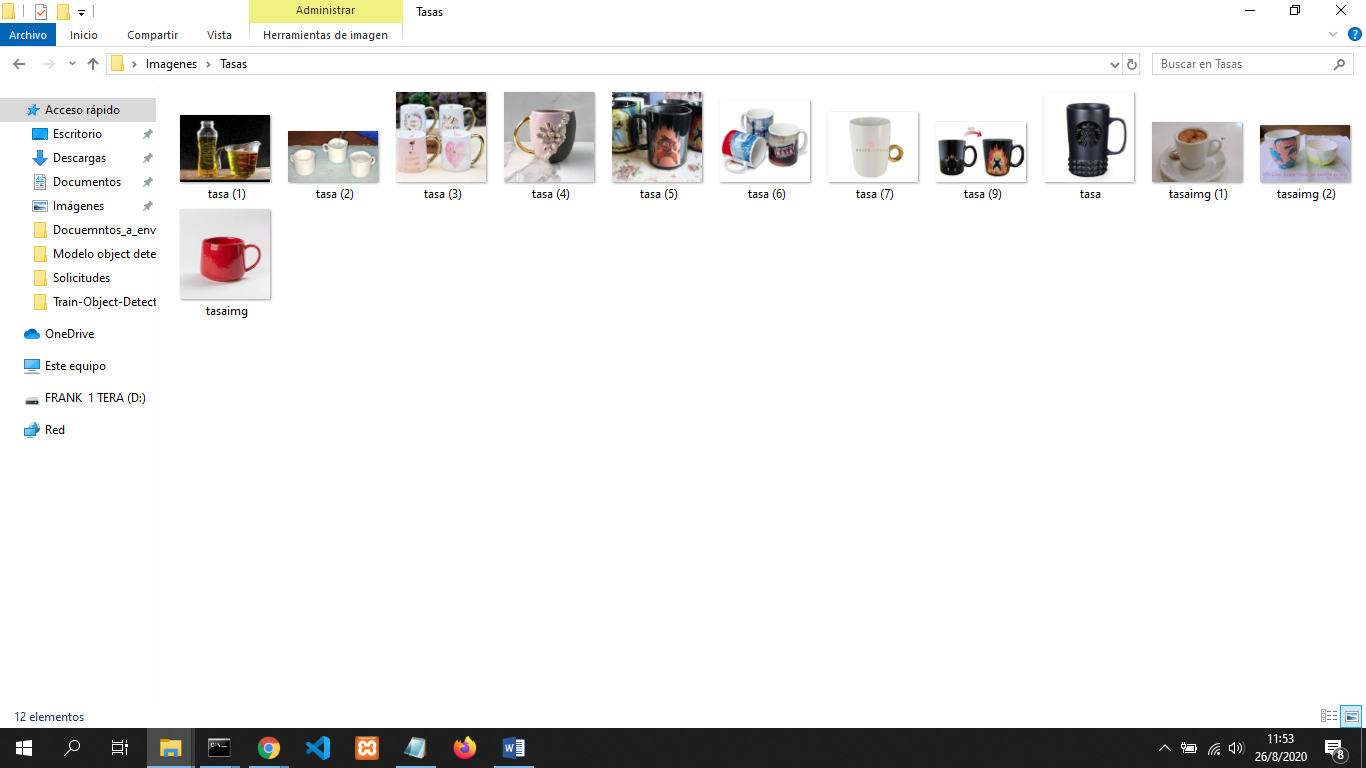
* **Python 3.5**
* **Tensorflow 1.12 – 1.14**

**Para empezar, necesitamos saber cuántos objetos va a reconocer nuestro programa. En este tutorial lo aremos con 2 objetos (tasa y esfero).**

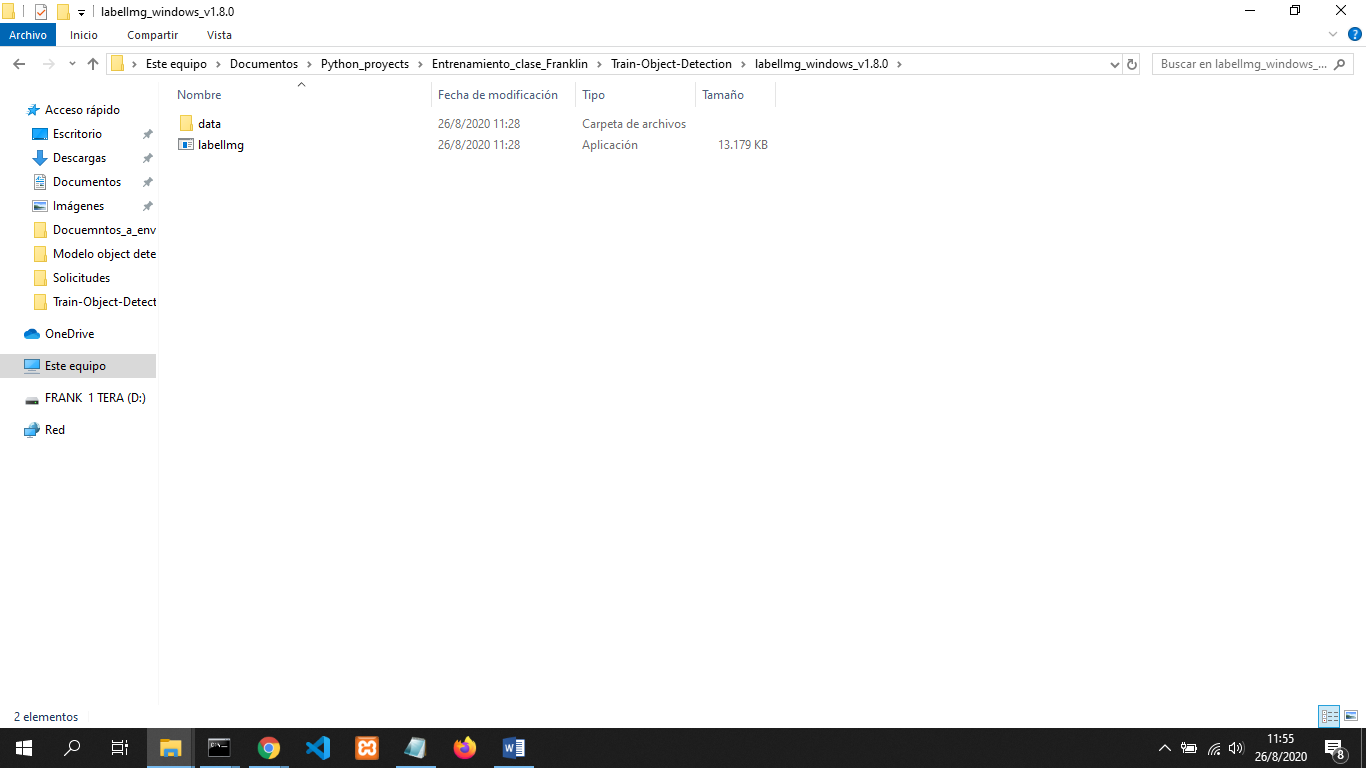
1. **Creamos una carpeta que va a contener nuestras imágenes por carpetas como se muestra en la imagen. Todas las imágenes deben tener nombres diferentes, ya que será un problema después si una imagen de una carpeta tiene el mismo nombre que el de una imagen de otra carpeta.**







1. **Dentro de los archivos descargados de github encontrará una carpeta llamada labelImg\_windows\_v1.8.0, es un ejecutable que permite dar una etiqueta xml a cada imagen, esto es necesario para los siguientes pasos del entrenamiento.**



1. **Ejecutamos la aplicación labelImg y damos clic en open dir y abrimos nuestra carpeta de imágenes**

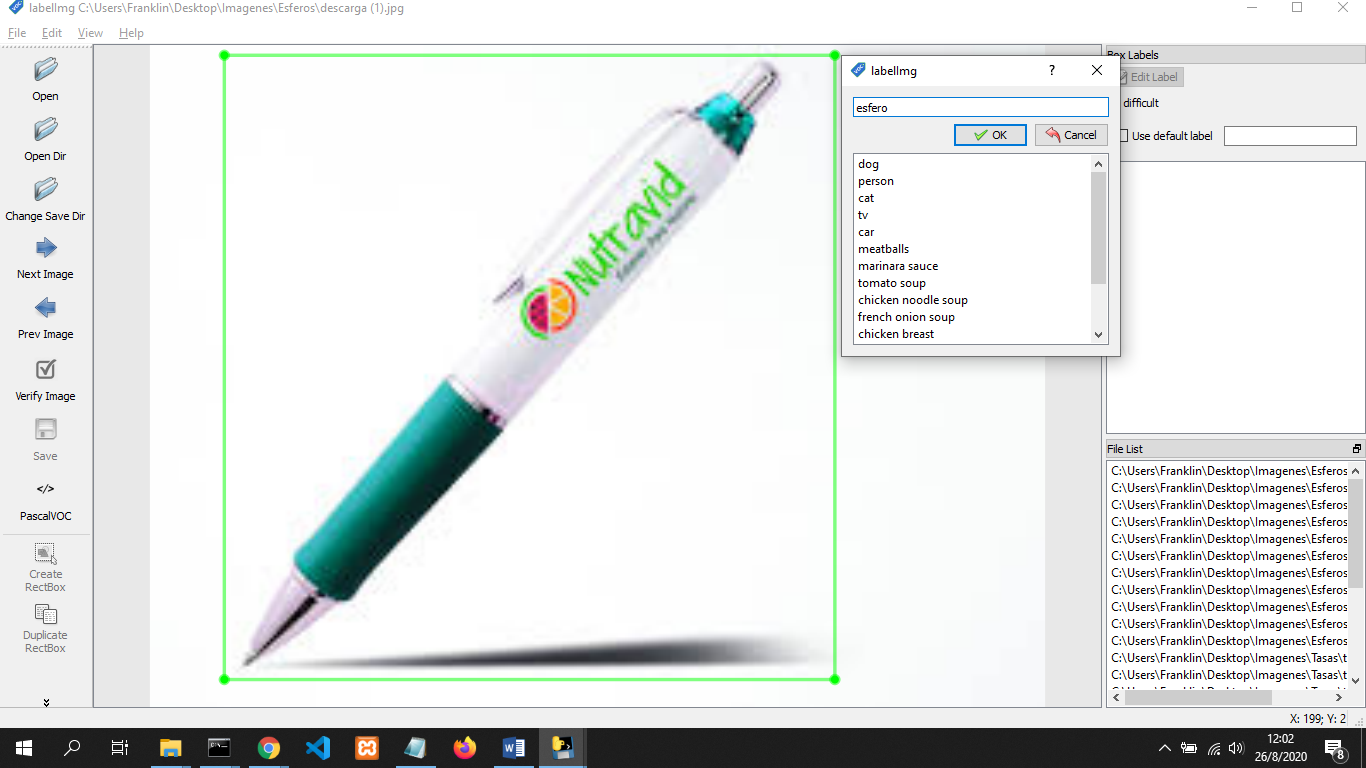


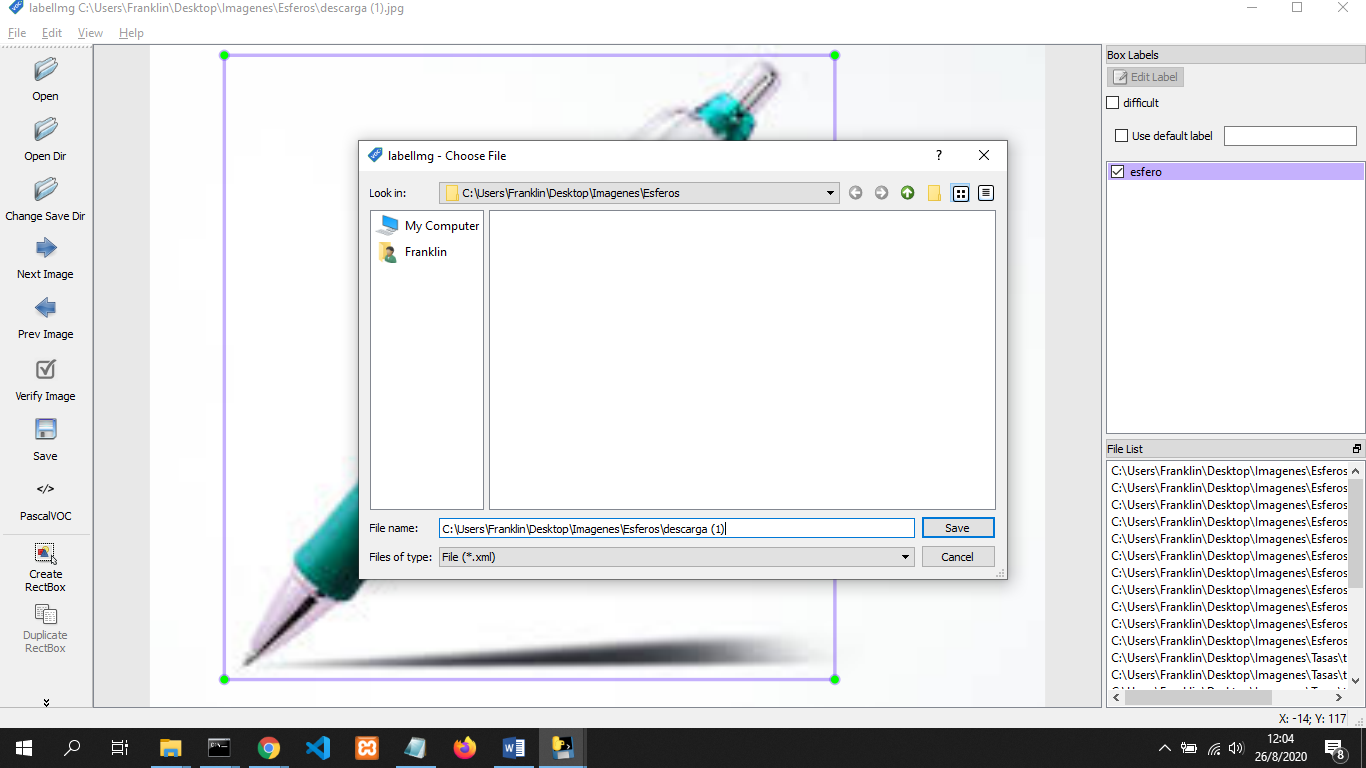


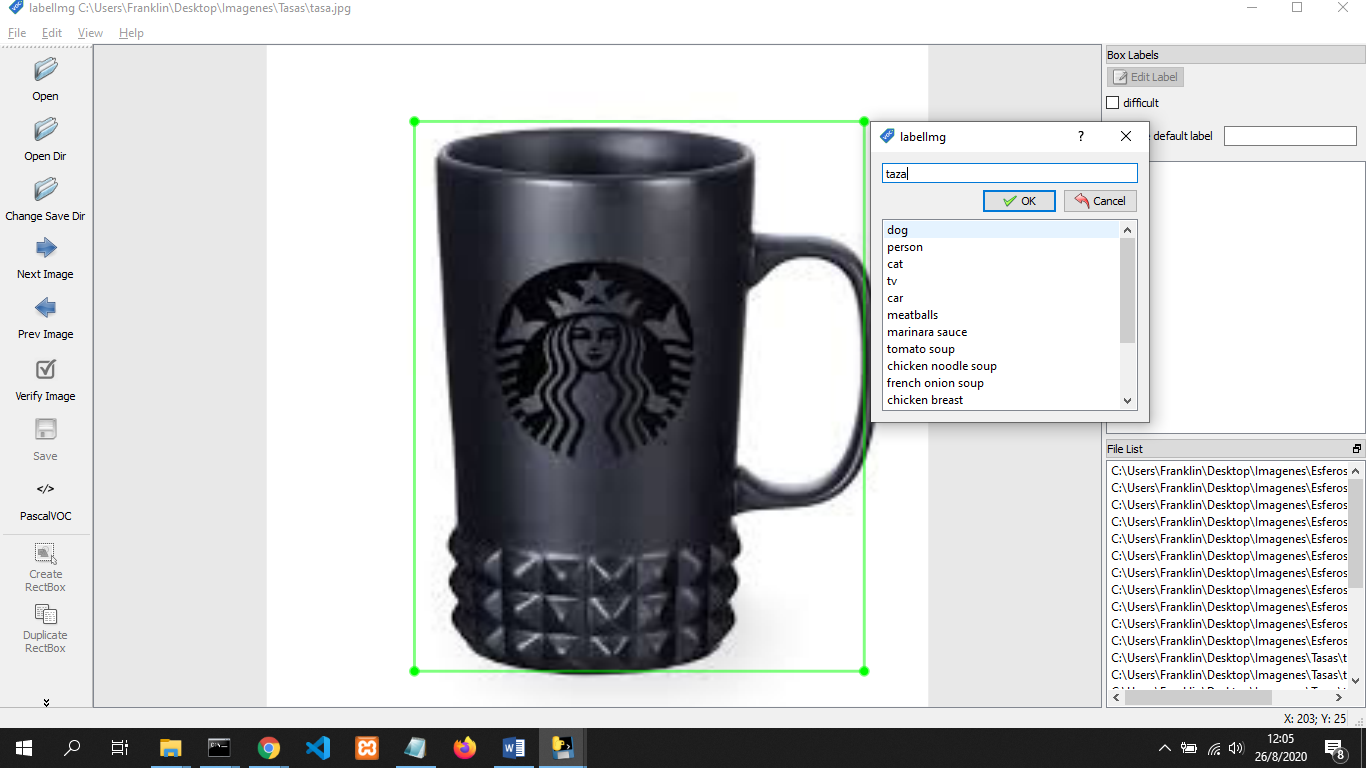
1. **Seleccionamos el ítem Create RectBox que se ubica en el lado izquierdo de nuestro panel y creamos un cuadrado alrededor de nuestro objeto**

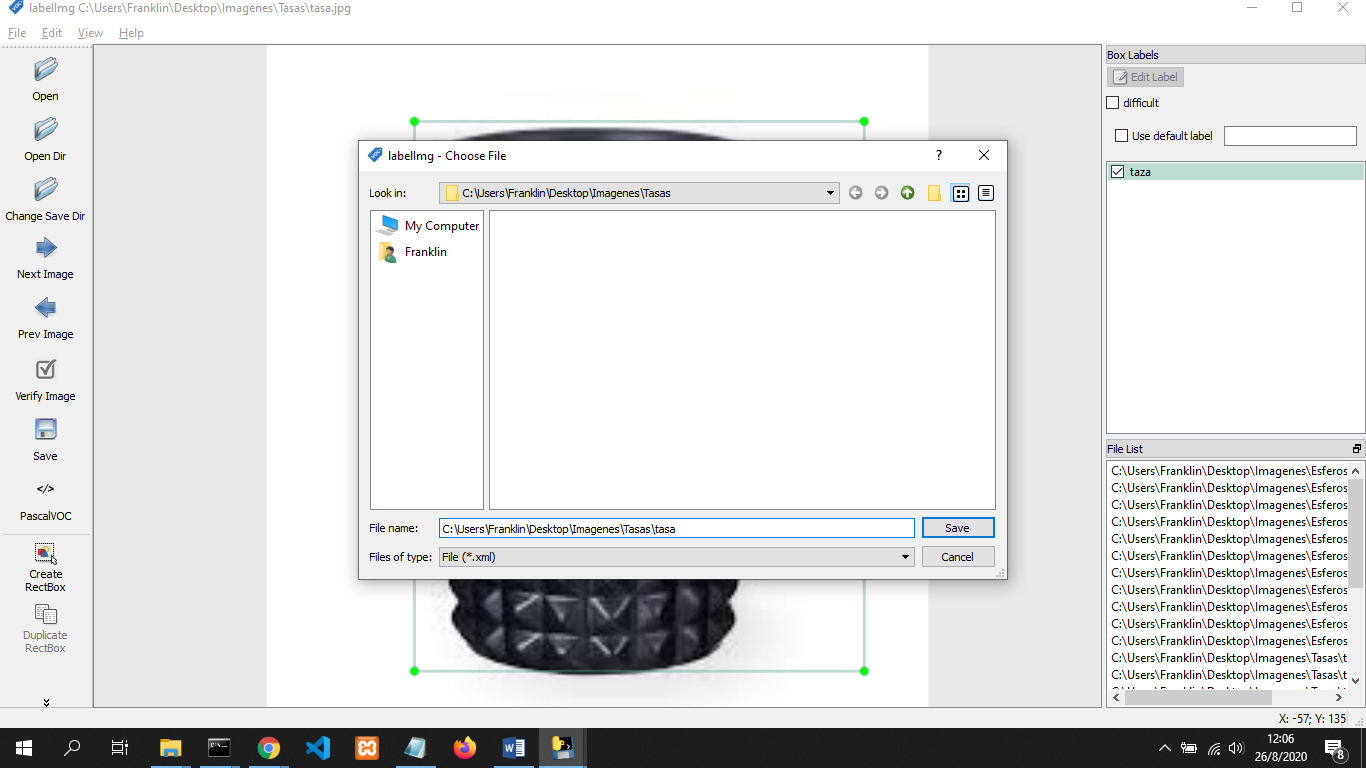


1. **Nos aparecerá una ventana en la que debemos seleccionar el nombre de nuestro label, en este caso todas las imágenes de esferos van a tener el label de esfero y le damos en ok y clic en save que está ubicado igual en el panel izquierdo, esto nos pedirá una ruta en la que se guardara el xml, así que guardaremos en el mismo lugar donde se encuentran nuestras imagenes.**

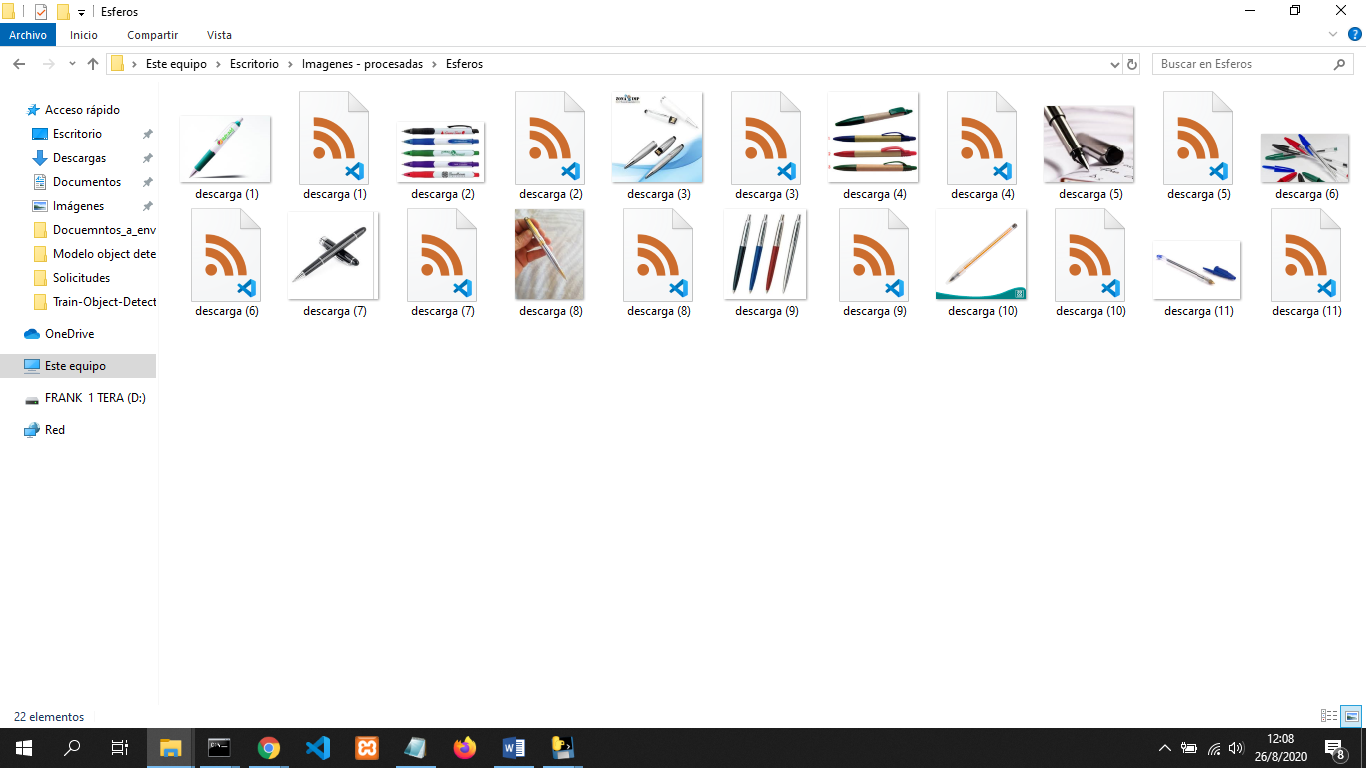


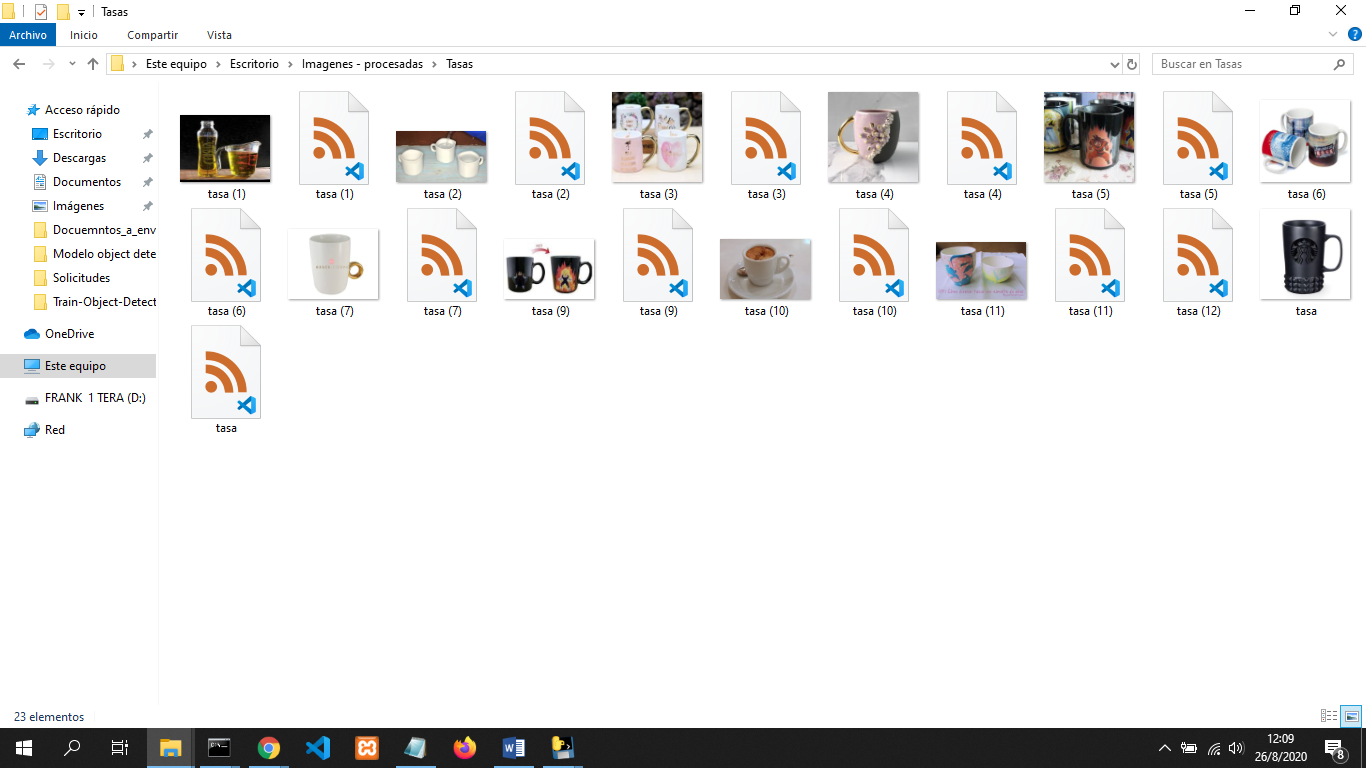






1. **Debemos hacer este proceso con todas las imágenes que tengamos, asegurándonos de darle el mismo nombre del label a todas las imágenes y guardándolas en el mismo directorio de la imagen. Nos quedará algo como esto.**





**Despues pasaremos todo el contenido de la carpeta a model/research/object\_detection/images pero eso lo veremos en el entrenamiento. Estas imágenes que ya contienen los xml se llamaran imágenes precesadas**

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

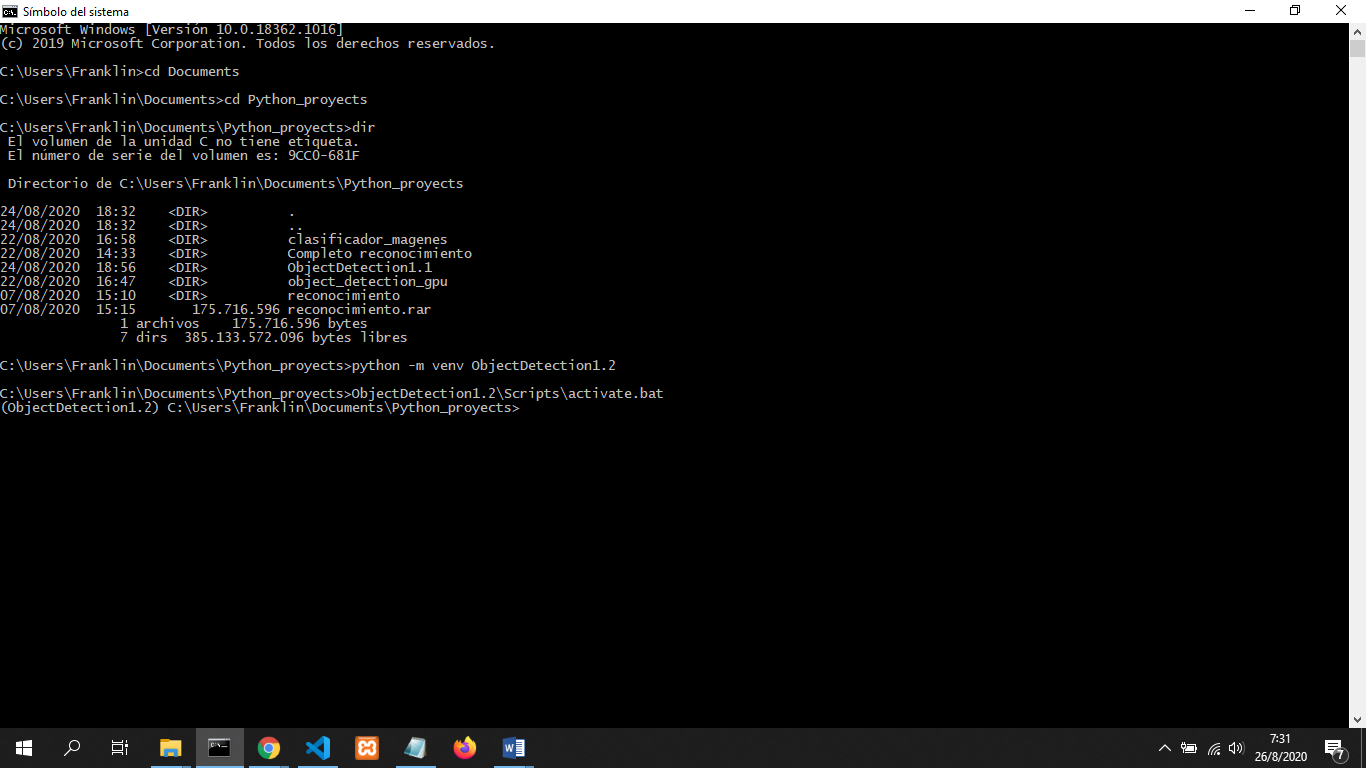
1. **Crear un entorno virtual**

* COMANDO: python -m venv NombreEntornoVirtual

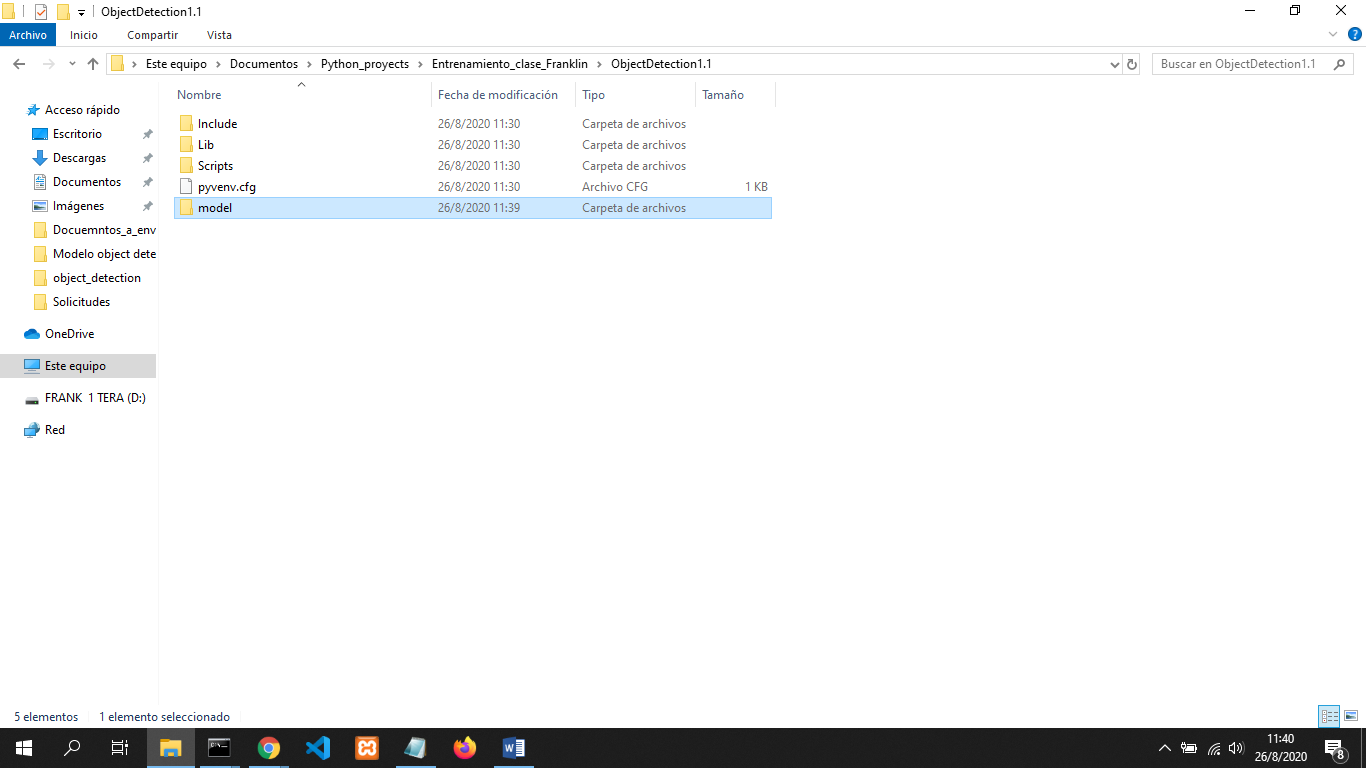
1. **Acceder al entorno virtual**

* COMANDO: NombreEntornoVirtual\Scripts\activate.bat

Para identificar que estamos dentro del entorno virtual



1. **Copiar la carpeta model dentro del entorno virtual que acabamos de crear**



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

* COMANDO: python -m pip install --upgrade pip

1. **Instalar todos los paquetes necesarios**

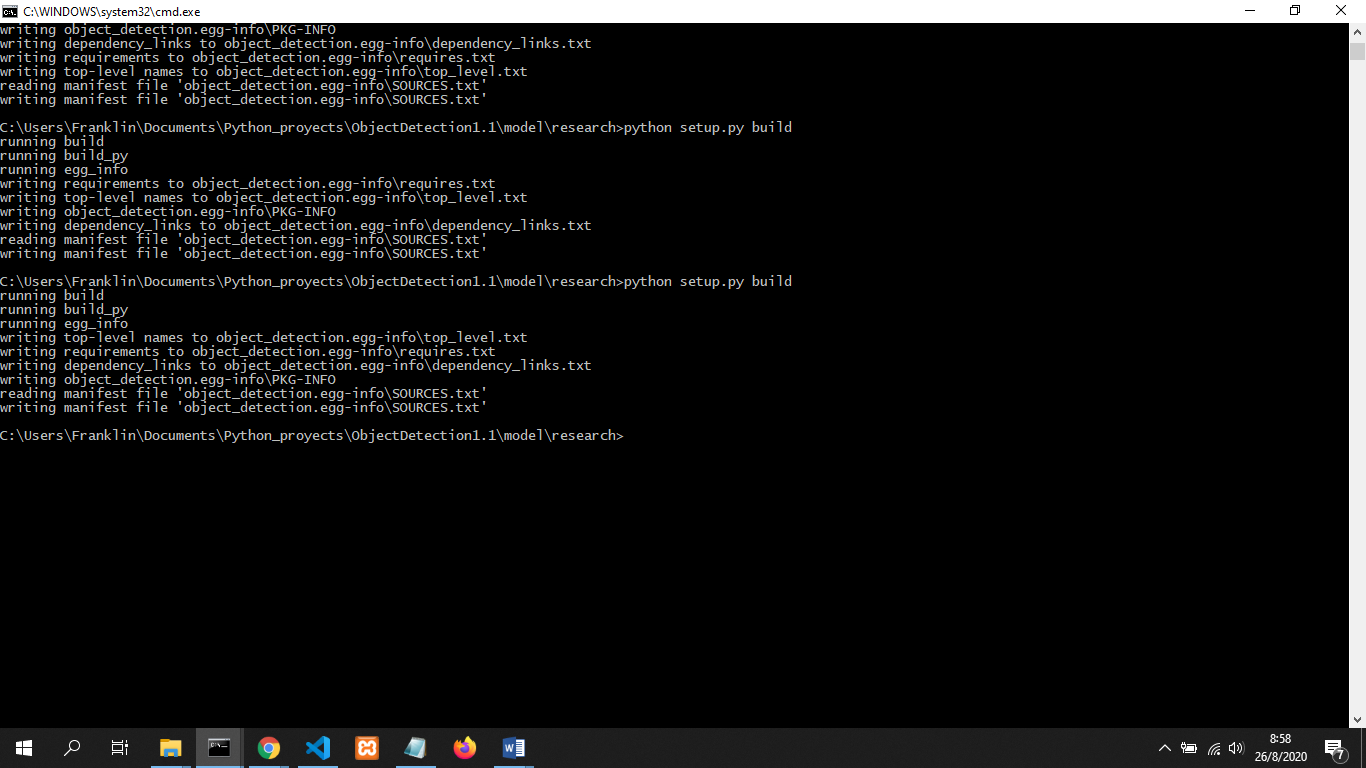
* COMANDO: pip install tensorflow==1.14
* COMANDO: pip install protobuf
* COMANDO: pip install pillow
* COMANDO: pip install lxml
* COMANDO: pip install Cython
* COMANDO: pip install contextlib2
* COMANDO: pip install matplotlib
* COMANDO: pip install pandas
* COMANDO: pip install opencv-python
* COMANDO: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

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

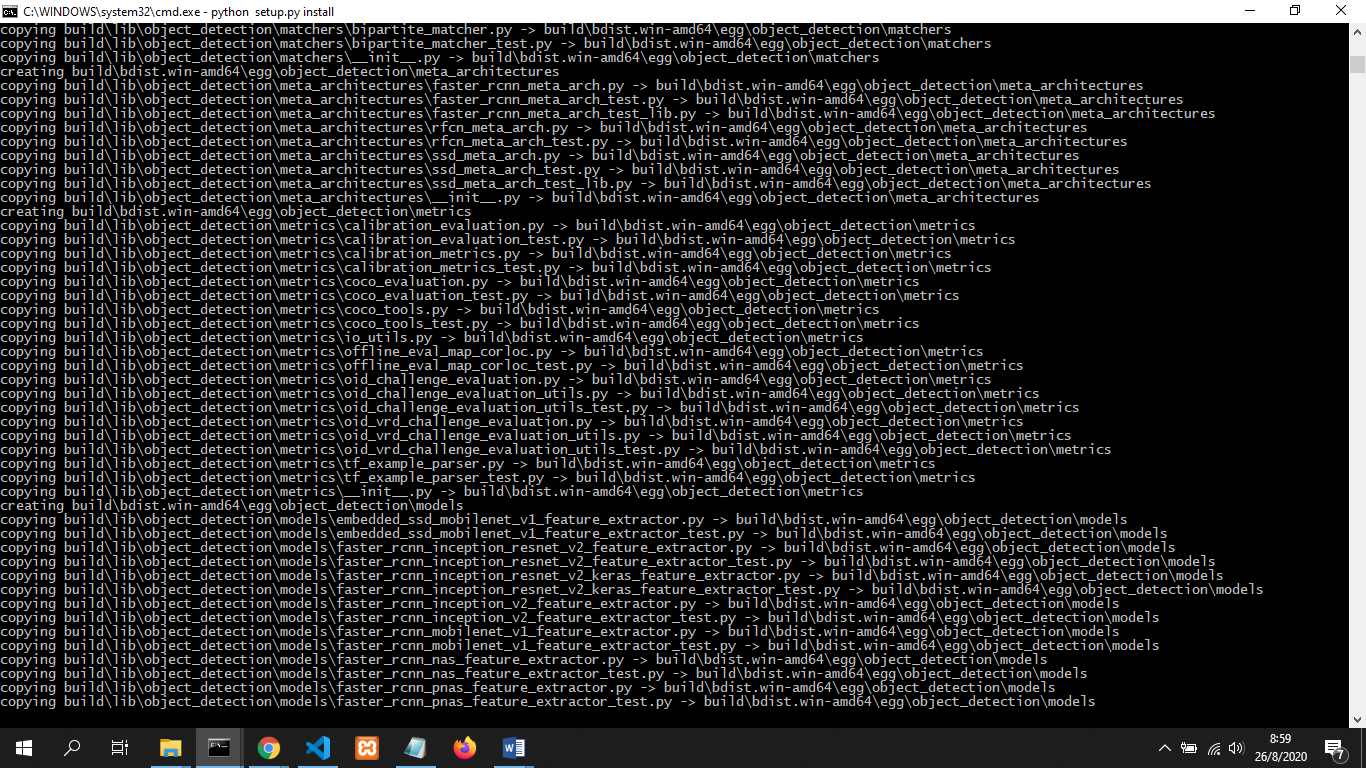
* COMANDO: set PYTHONPATH=C:\Users\Franklin\Documents\Python\_proyects\ObjectDetection1.3\modelC:\Users\Franklin\Documents\Python\_proyects\ObjectDetection1.3\model\research;C:\Users\Franklin\Documents\Python\_proyects\ObjectDetection1.3\model\research\slim

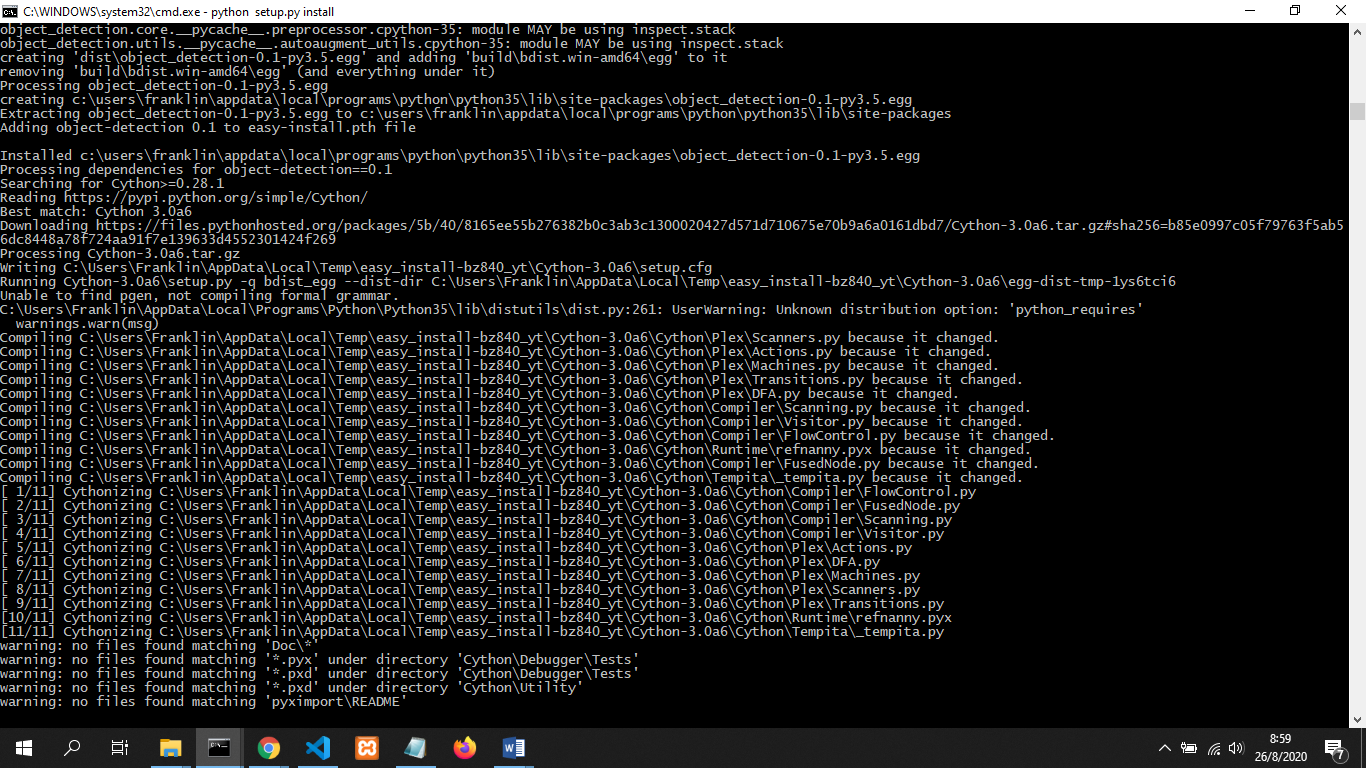
1. **Dentro de la carpeta model/research ejecutar los siguiente comandos**

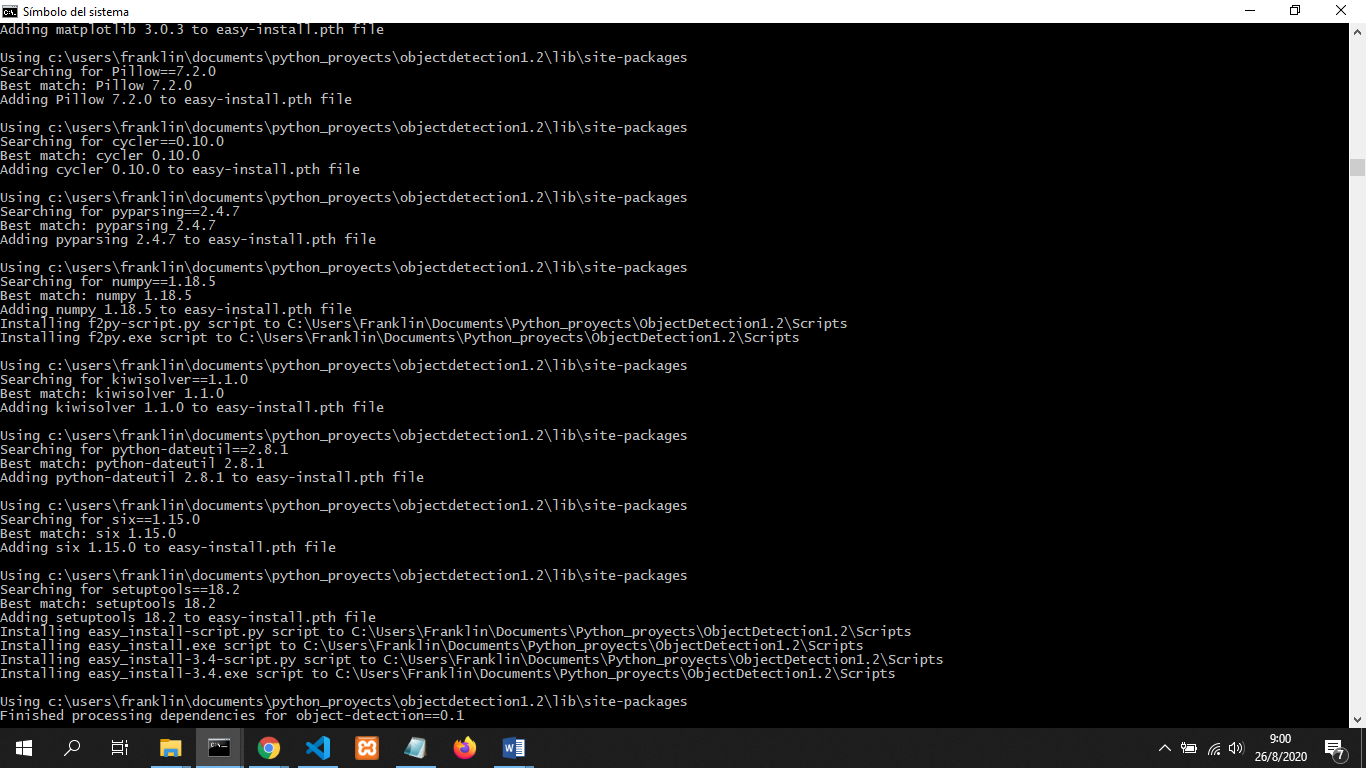
* COMANDO: python setup.py build



* COMANDO: python setup.py install



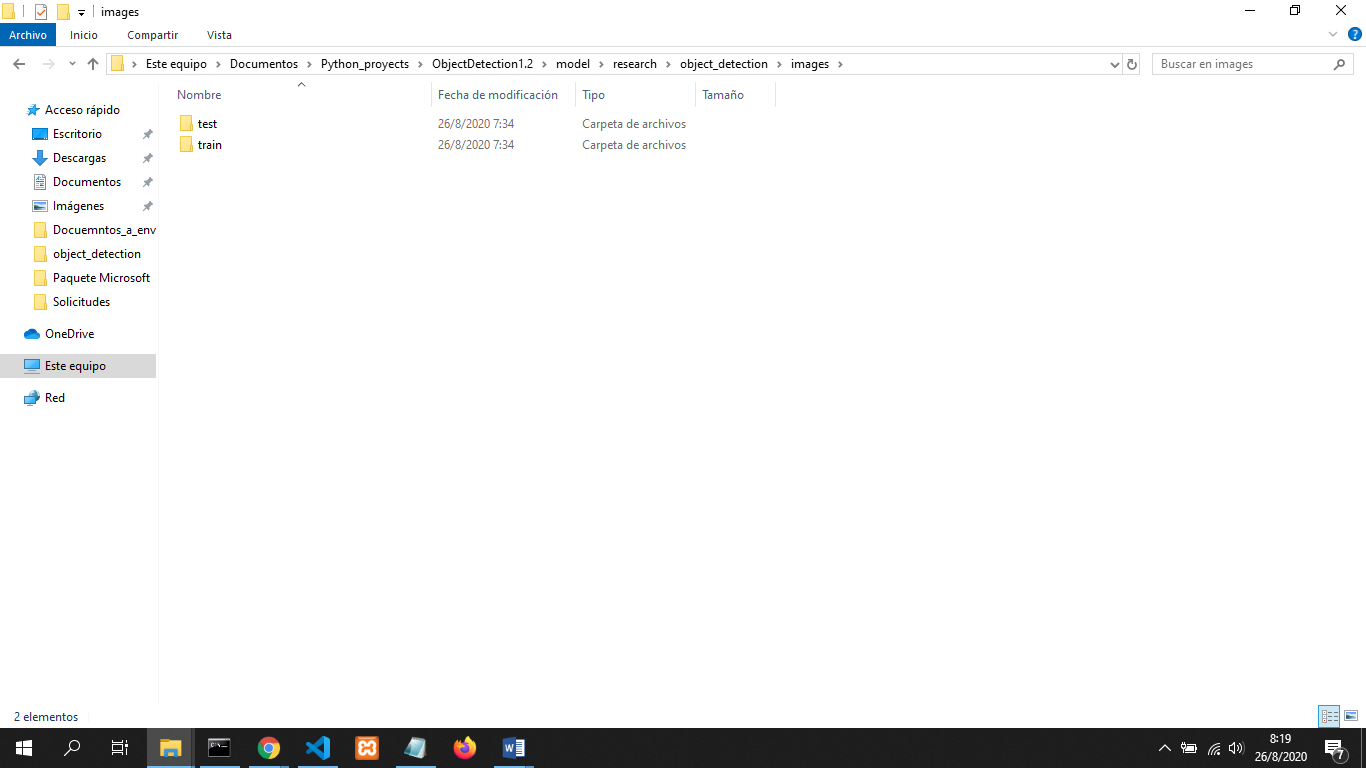




1. **Creamos la carpeta images dentro de model/research/object\_detection, y dentro de la carpeta images creamos otras dos carpetas con los nombres de train y test de la siguiente manera**



1. **Copiamos nuestras imágenes procesadas dentro de las carpetas train y test que está dentro de model/research/object\_detection/images, Todas las imágenes sin importar el objeto que contengan, es decir las dos carpetas de imágenes van dentro de train y test..**

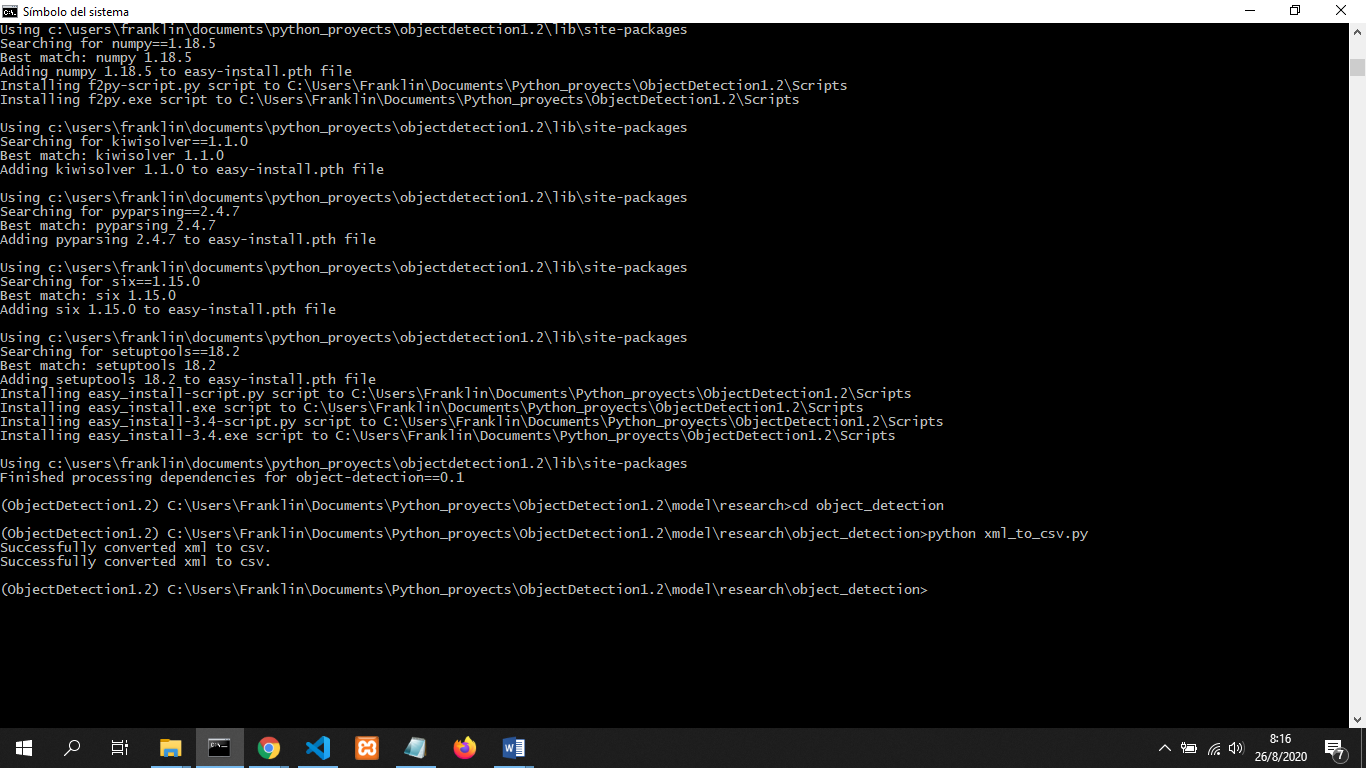


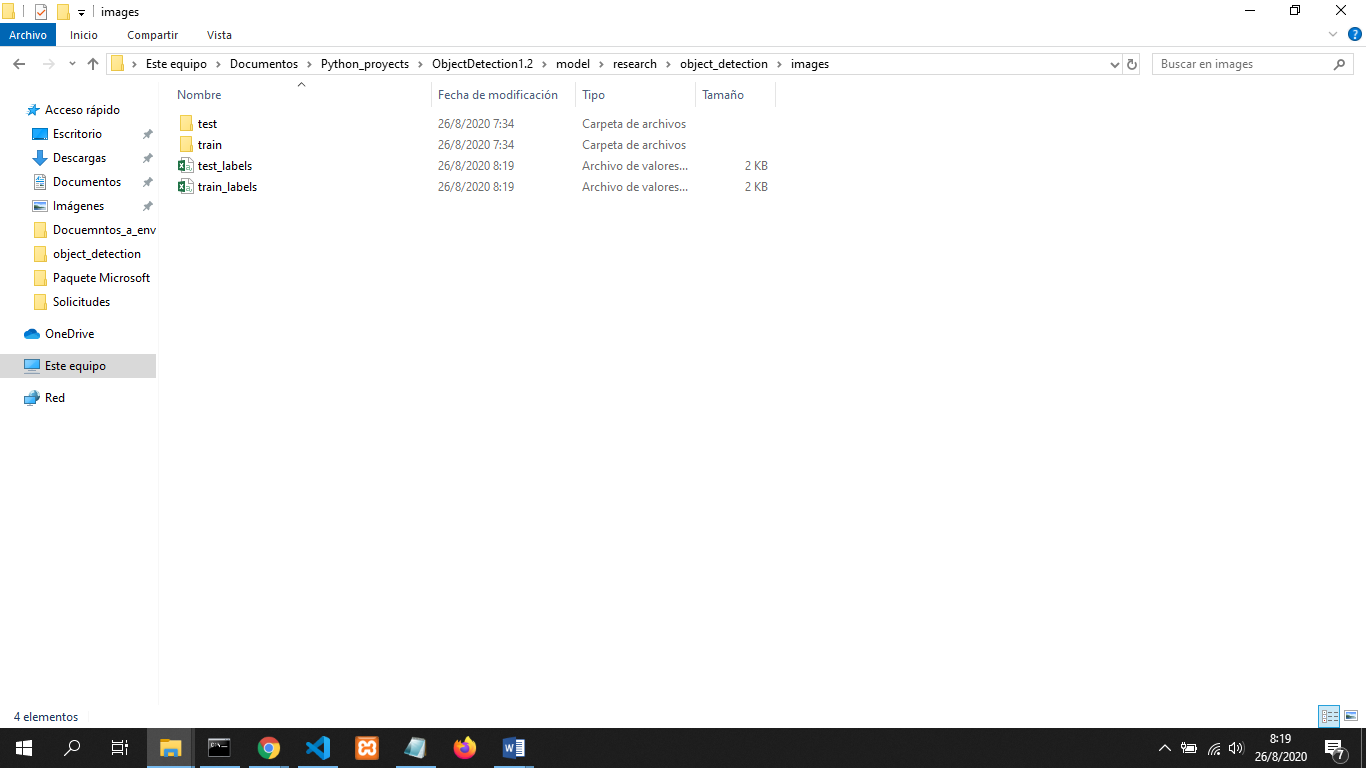


1. **Dentro de la carpeta model/research/object\_detection ejecutamos los siguientes comandos.**

* COMANDO: python xml\_to\_csv.py

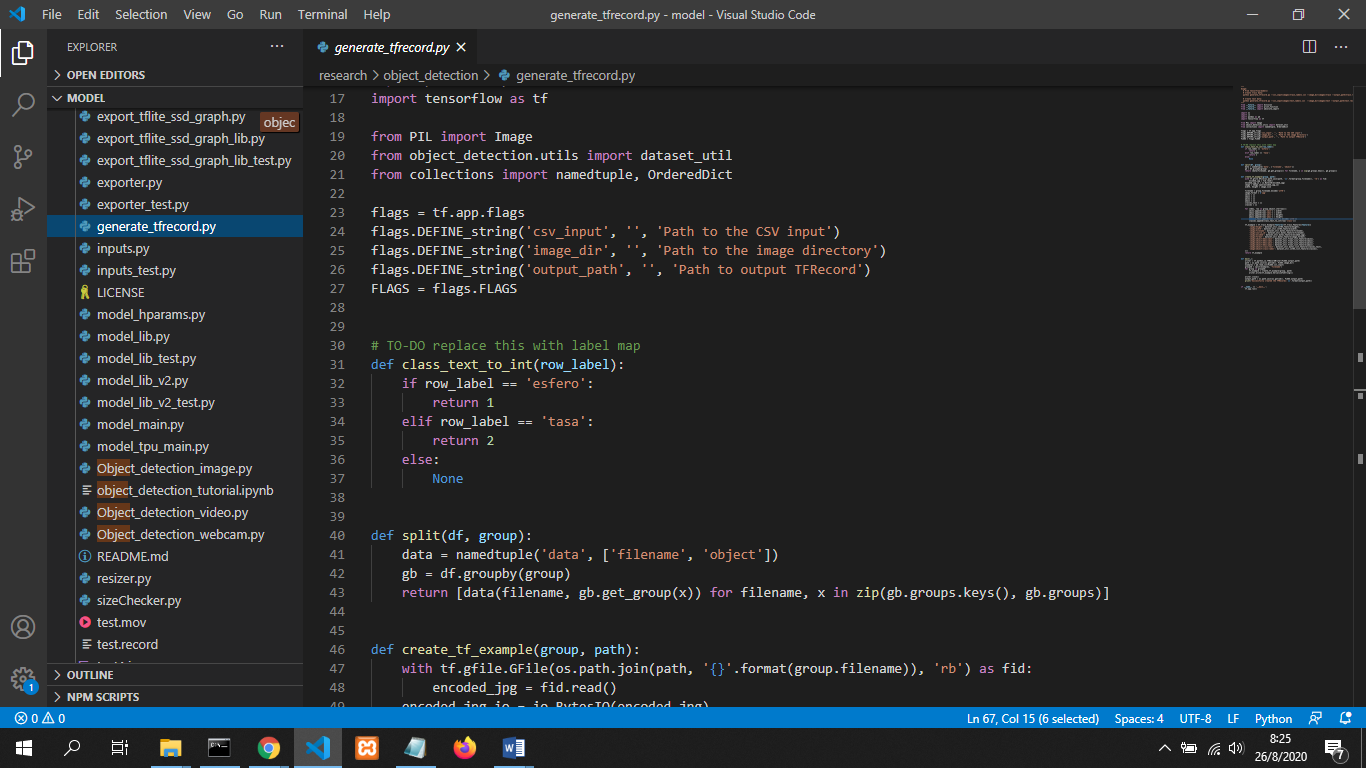
Esto nos convertirá los xml a un formato entendible para la maquina





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

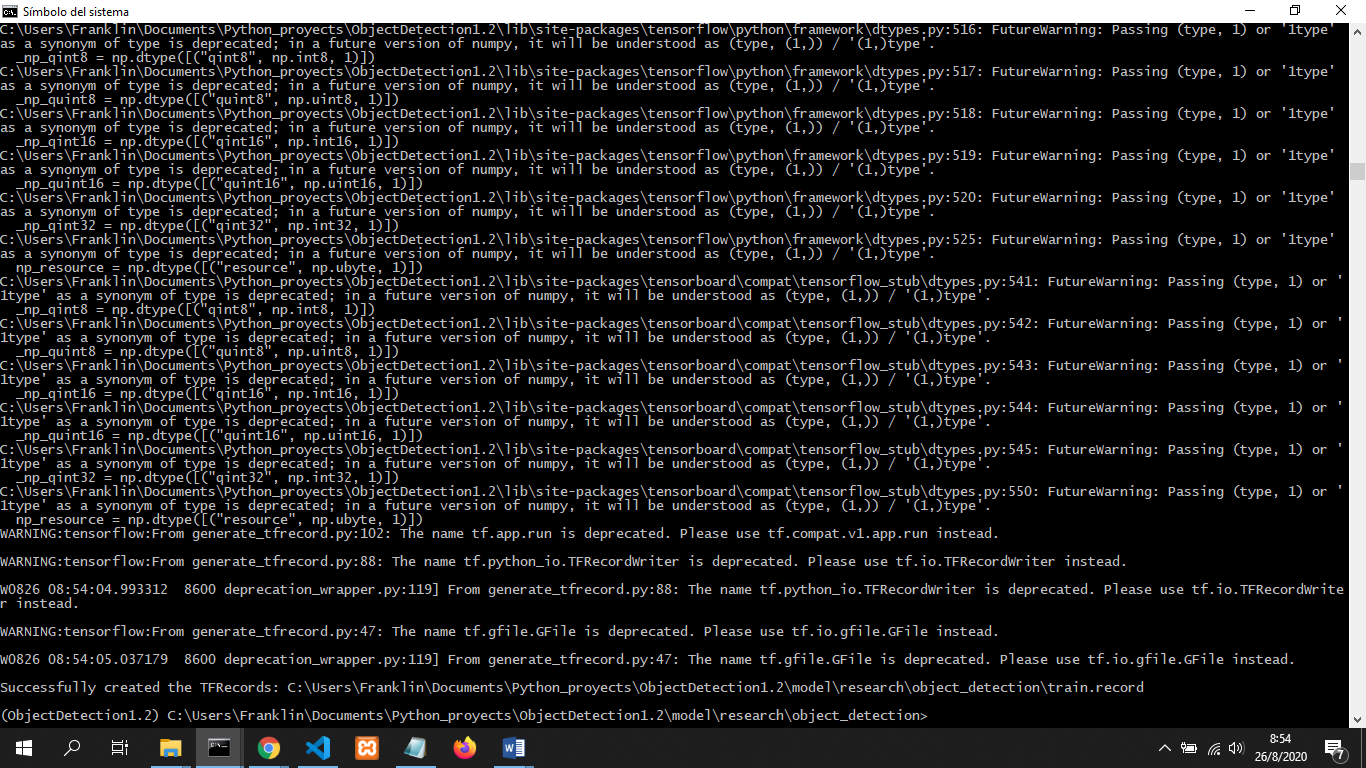
1. **Modificamos el archivo *generate\_tfrecord.py* que está ubicado dentro de la carpeta model/research/object\_detection por el número de objetos que queremos entrenar. Ejemplo: Si queremos identificar una tasa y un esfero los colocamos dentro de este archivo de la siguiente manera:**



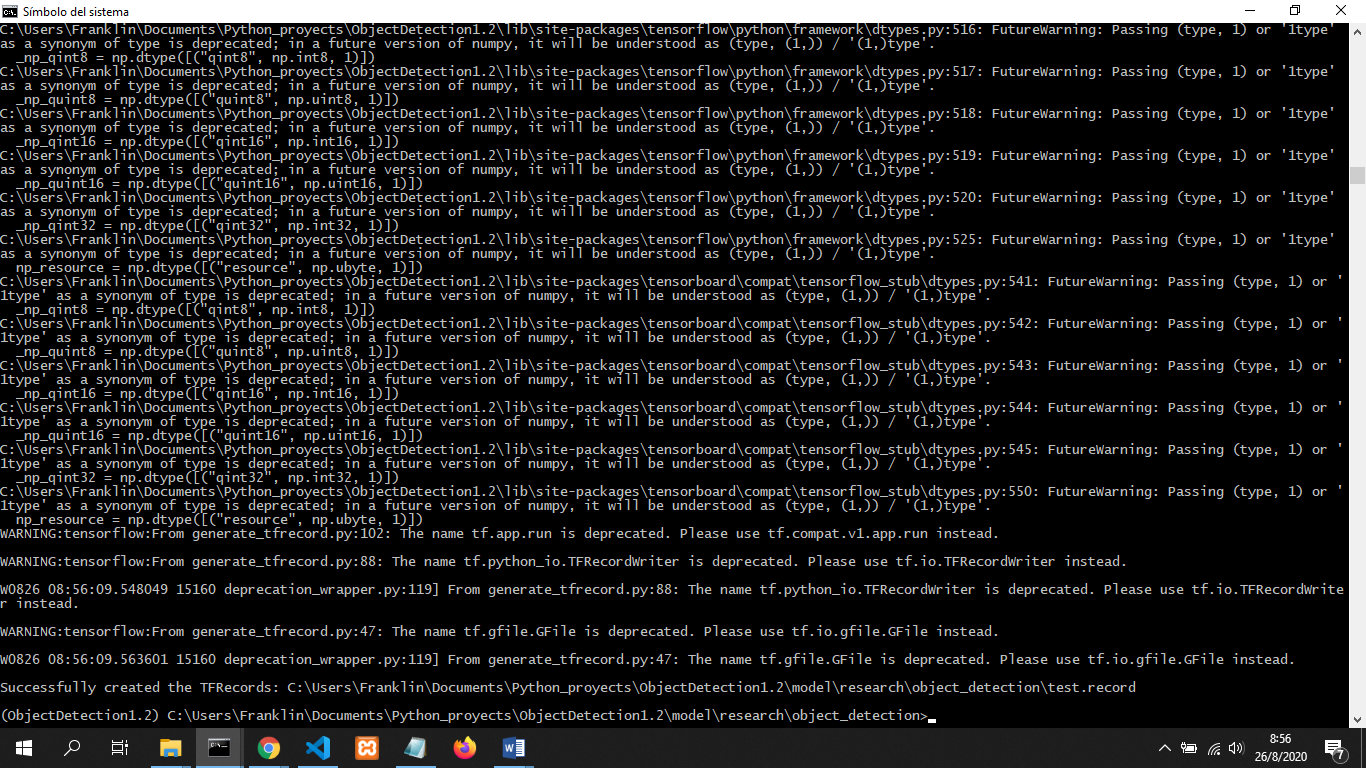
Este nombre debe coincidir con los label que asignamos a nuestras imágenes al procesarlas con labelimg.

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

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



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



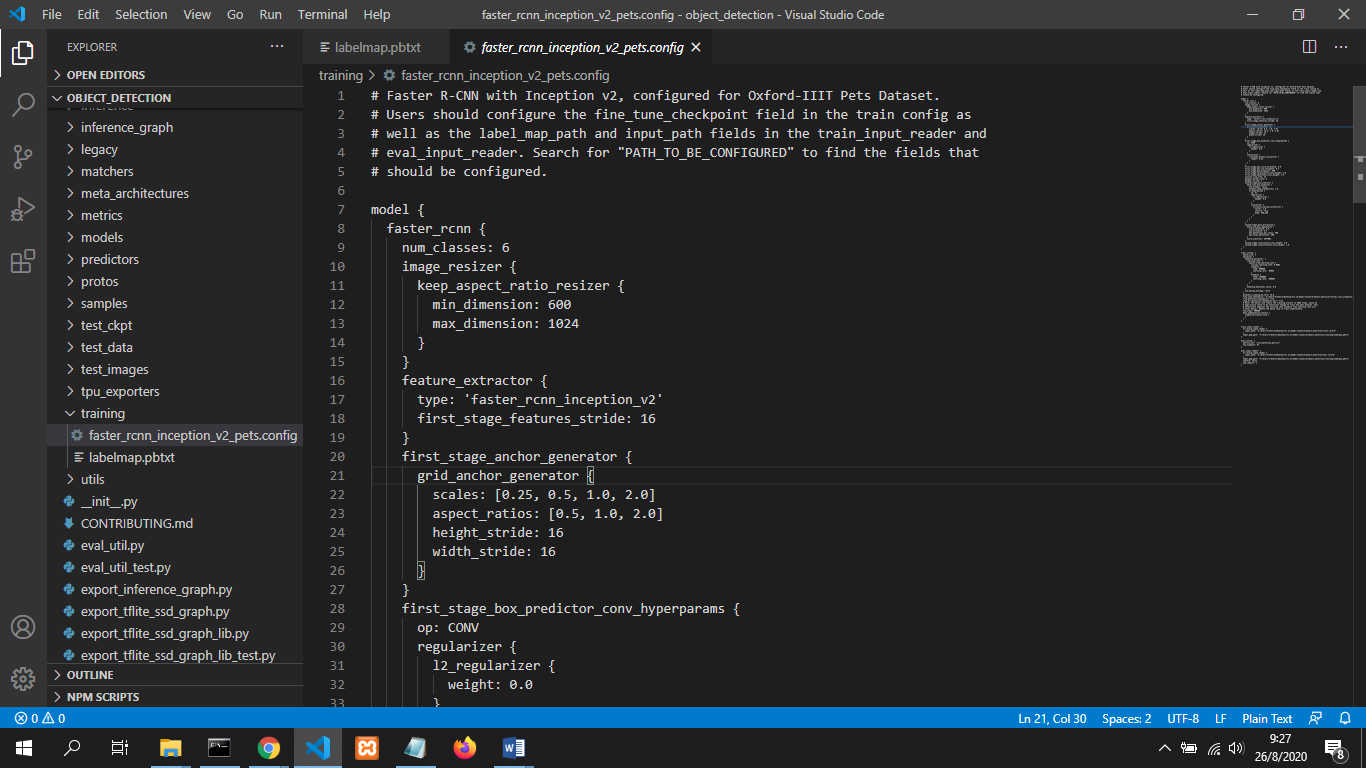
Esto nos generará los archivos con extencion .record (train.record y test.record) estos archivos se obtuvieron de los csv de la carpeta de imágenes, es decir a los labels de las imágenes las procesamos 2 veces.

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:**

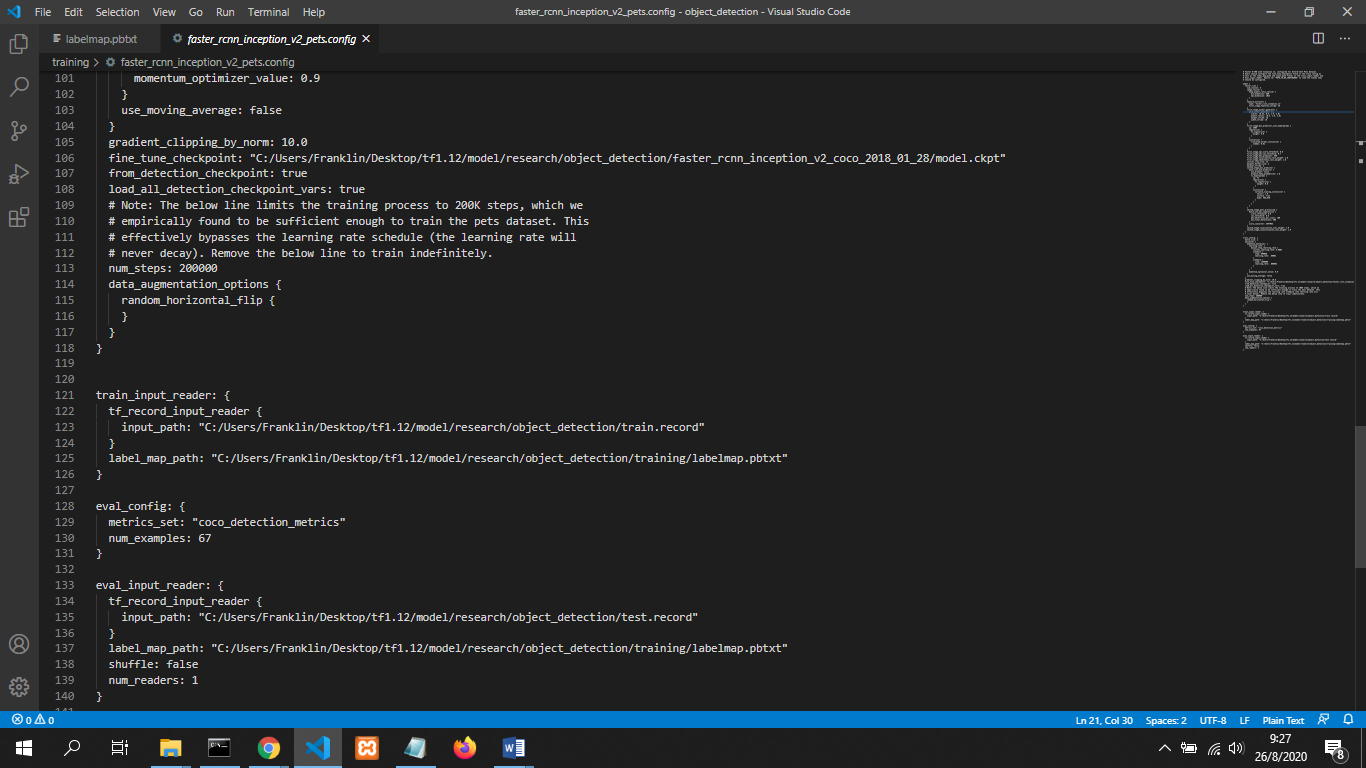
***Archivo: faster\_rcnn\_inception\_v2\_pets.config***

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

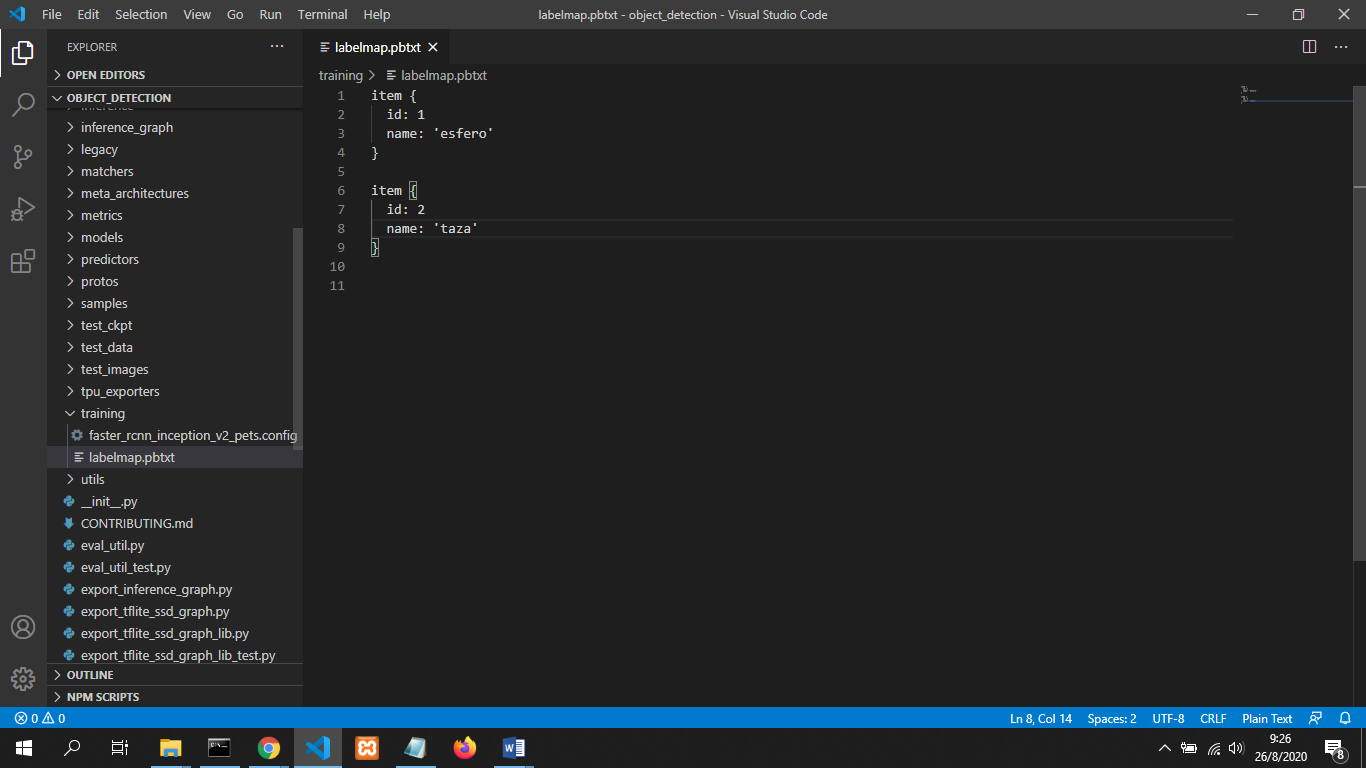


Modificamos todos los path con los que corresponden a nuestra carpeta



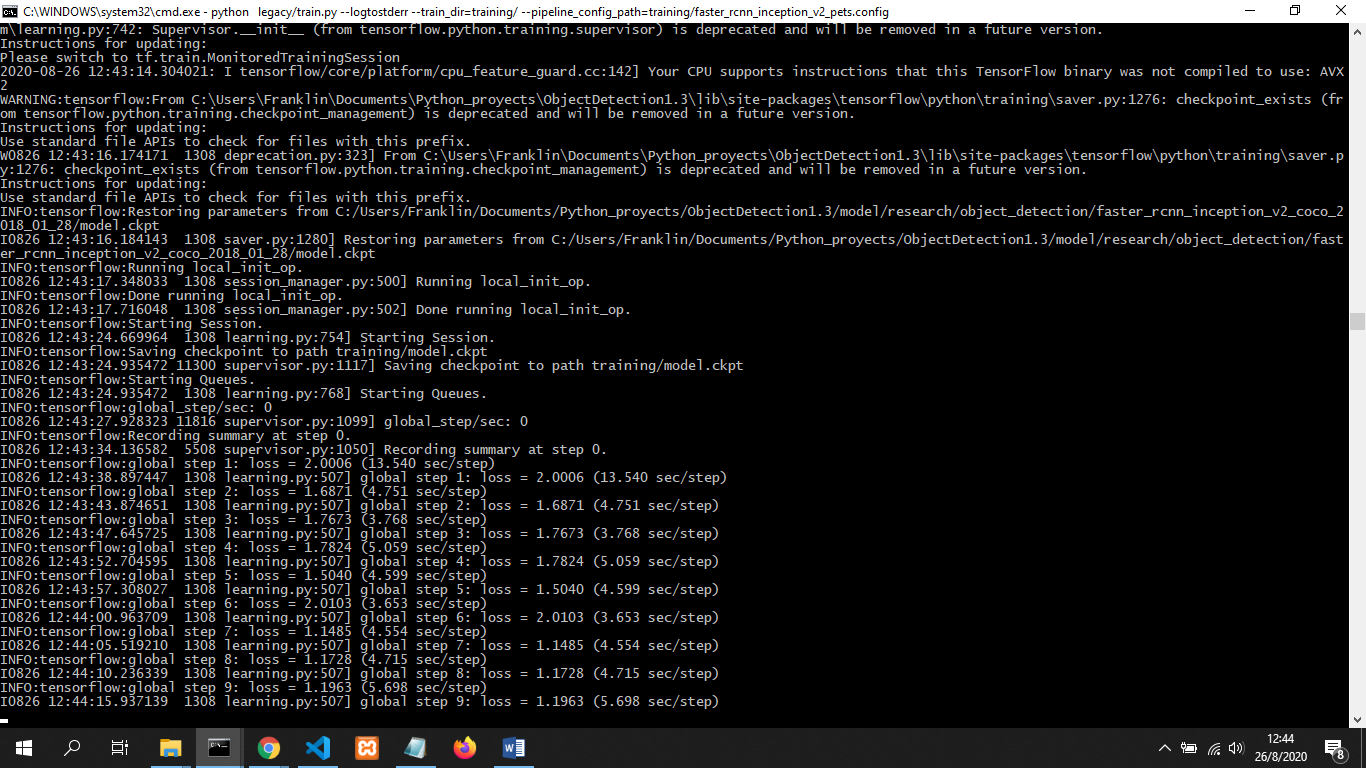
***Archivo: labelmap.pbtxt***

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 ejecutamos el comando dentro de model/research/object\_detection:**

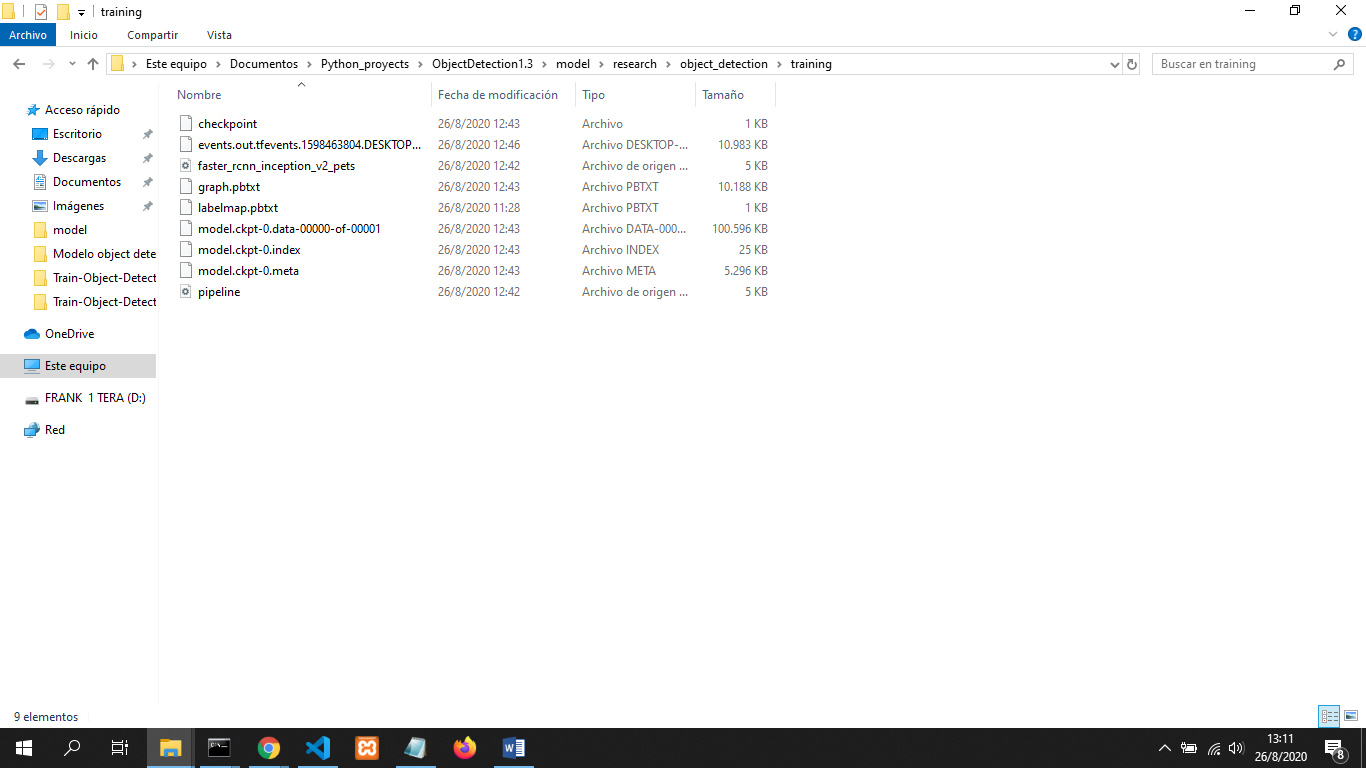
* COMANDO: python legacy/train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/faster\_rcnn\_inception\_v2\_pets.config



1. **Para congelar el entrenamiento paramos el programa con ctrl + c y ejecutamos el comando**

* COMANDO: python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/faster\_rcnn\_inception\_v2\_pets.config --trained\_checkpoint\_prefix training/model.ckpt-907 --output\_directory inference\_graph

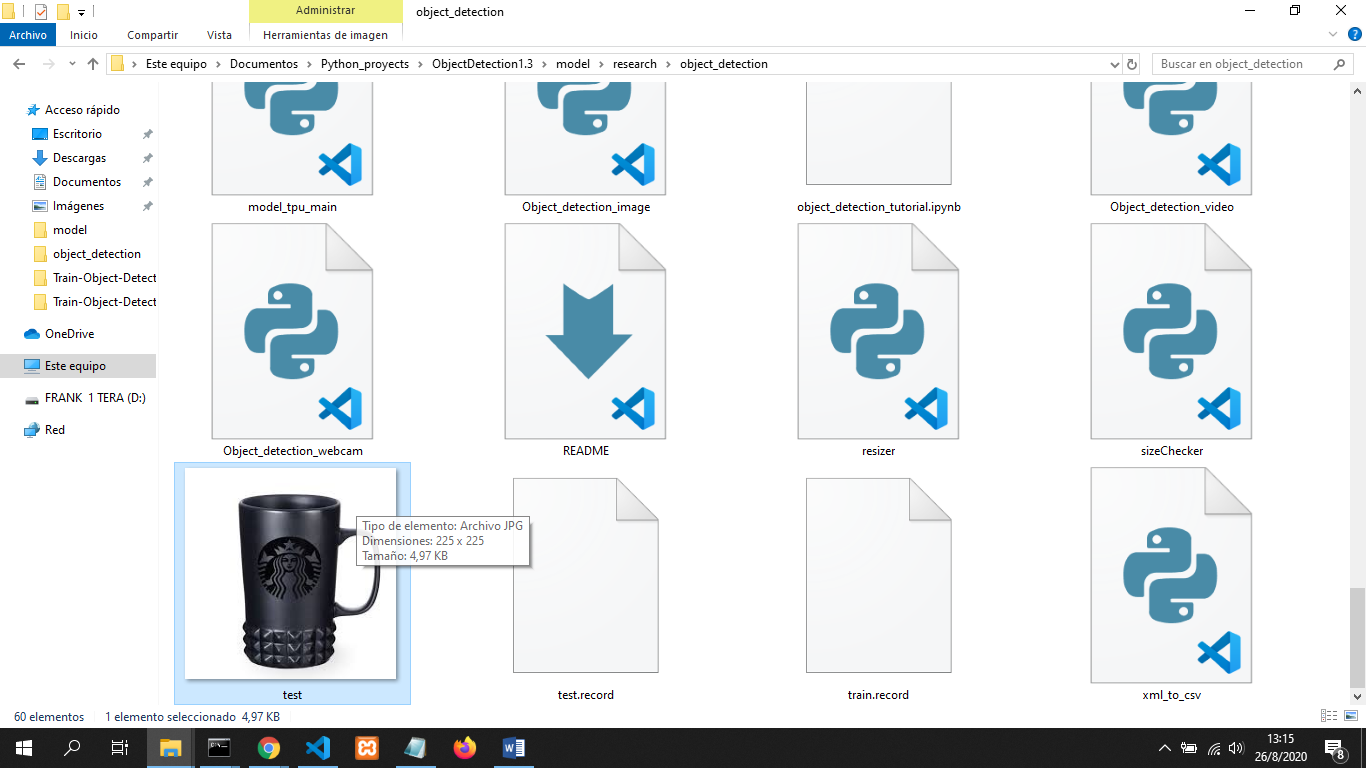
El número 907 hace referencia al proceso en el cual se quedo el entrenamiento. Este numero lo podemos encontrar en model/research/object\_detection/training



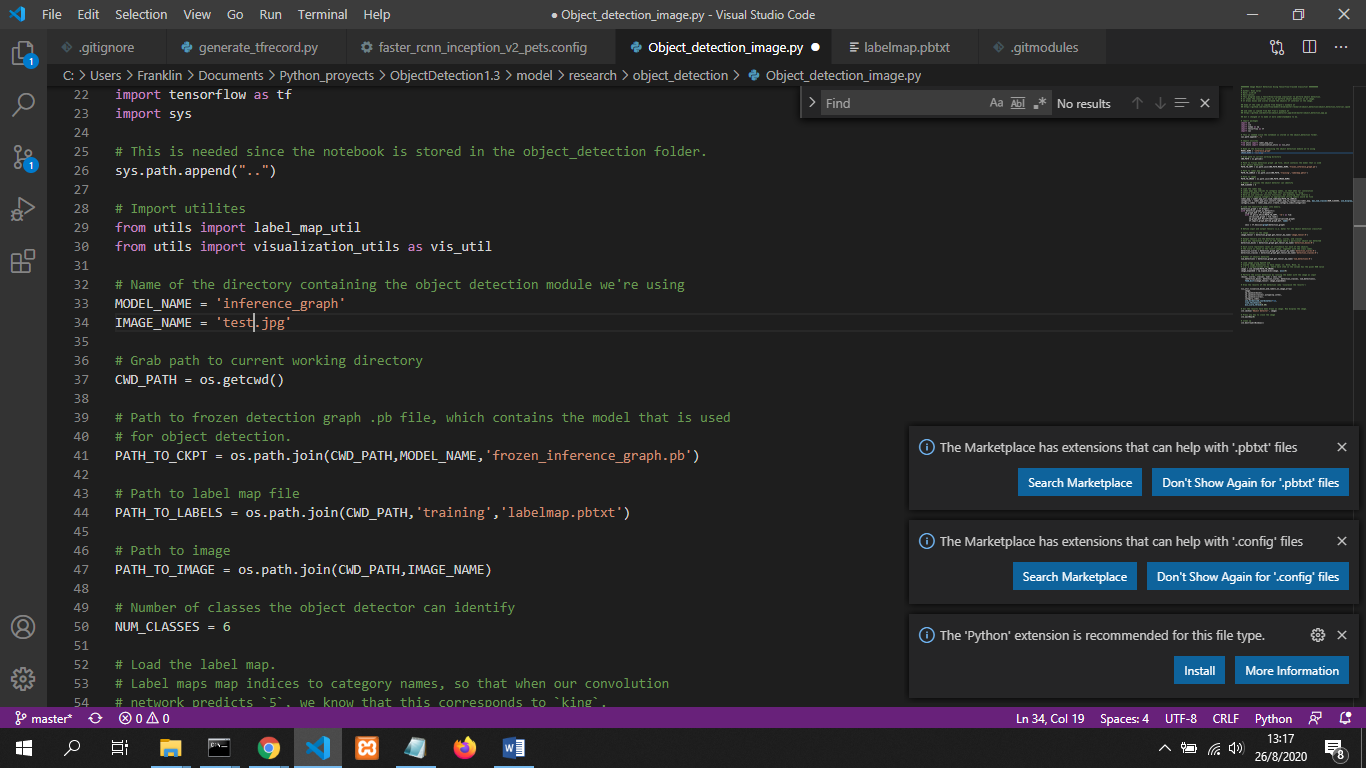
**En este caso el número es el 0 pero entre más entrenemos la red neuronal este número irá incrementando.**

**Pruebas**

**Colocamos una imagen dentro de** model/research/object\_detection **con cuerlaquier nombre.**



**Configuramos el archivo object\_detection\_imagen.py que se encuentra en este mismo directorio y cambiamos IMAGE\_NAME por el nombre de nuestra imagen**



**Solo nos queda ejecutar el archivo y verificar que todo esté funcionando correctamente.**

**La duración del entrenamiento depende de las capacidades de nuestra máquina, entre más pasos se ejecuten en el entrenamiento, más precisa será nuestra red.**