# Desarrollo de Productos de Datos

## Tarea 1: Modelo de ML Cliente-Servidor

### Descripción General

Vamos a expandir lo realizado en el [Laboratorio de Implementación](https://drive.google.com/file/d/1WyDwDL7HNADH-hdM17TIk4BL1iQXvrdR/view?usp=sharing) de un Modelo de Machine Learning visto en [clase](https://drive.google.com/file/d/1tOk1g4D4tTvJDGv5sLIXFxr1a8hGPKgW/view?usp=sharing). Para desarrollar la Tarea debe basarse en el código proporcionado y empezar a construir las mejoras y nuevas funcionalidades sugeridas en las siguientes preguntas.

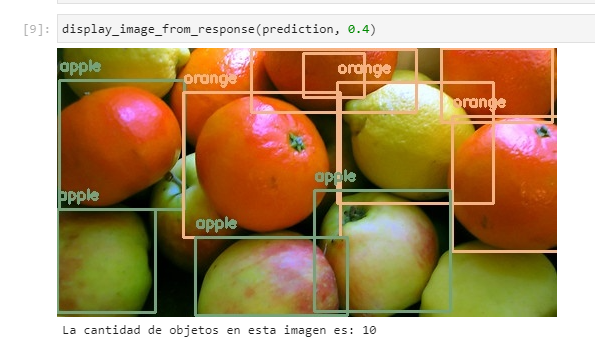
### Preguntas Prácticas (Código)

**2.1 (1 punto)** Agregue al programa servidor la opción de consultar al endpoint */predict* agregando como parámetro el nivel de confianza (si necesita orientación, lea el punto [5: Recursos](#_a00xhgt7k724))



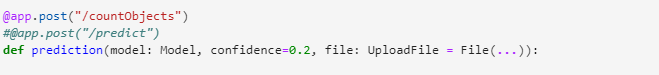
**2.2 (1 punto)** Realice los cambios necesarios en el programa cliente para comprobar que la consulta con nivel de confianza funciona correctamente. Pruebe con la imagen *fruits.jpg* para distintos niveles de confianza y comente los resultados. ¿Se le ocurre alguna otra imagen que sea interesante evaluar en este escenario? (puede buscar cualquier imagen en Internet, no necesariamente las adjuntas en los archivos).

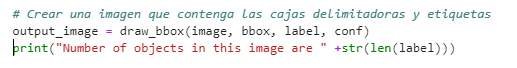
**R-.** El investigador creyó pertinente poner a prueba el modelo y quiso utilizar una fotografía con pelotas de distintos deportes, sin tener el resultado esperado, pero era bastante obvio lo que pasaría, era sólo para setear expectativas. El modelo no supo identificarlas.

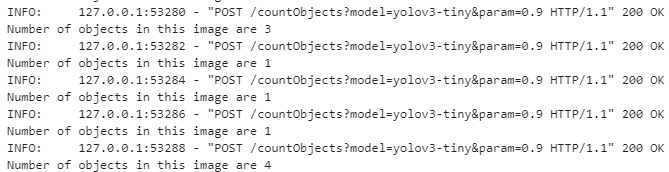
Por otro lado, se intentó con algo en la misma línea y el modelo identificó realizando un conteo de mejor manera:

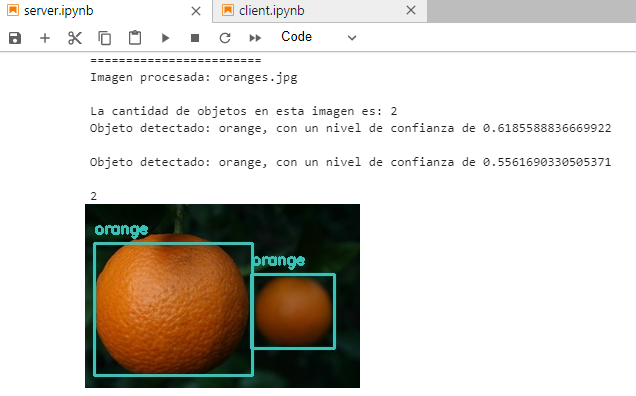
**2.3 (1 punto)** En el programa servidor implemente un nuevo endpoint */countObjects* para realizar un contador de un objeto en particular, por ejemplo un contador de naranjas (o de cualquier objeto que usted desee) apuntando al endpoint */countOranges*. La idea es que esta funcionalidad reciba un modelo, nivel de confianza y modelo pero entregue como salida la cantidad de naranjas que existen en la imagen.

**R-.**Se genera un nuevo endpoint y se realizan las modificaciones para que el modelo haga el conteo de los objetos que detecta.



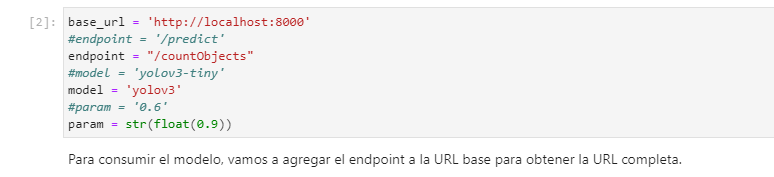


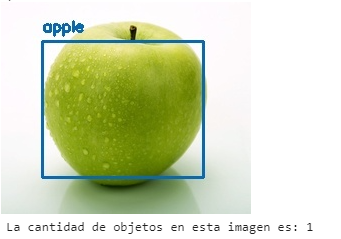


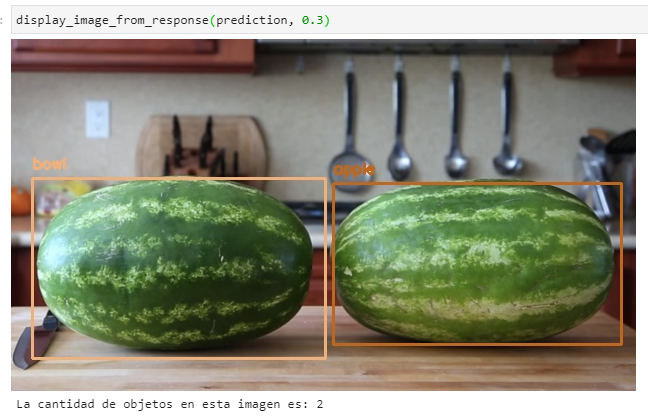


**2.4 (1 punto)** Agregue al cliente las funcionalidades necesarias para probar el funcionamiento del endpoint */countObjects.* Pruebe con imágenes que presenten: 1, 2, 3, 4, 5 o más ocurrencias del objeto en cuestión.

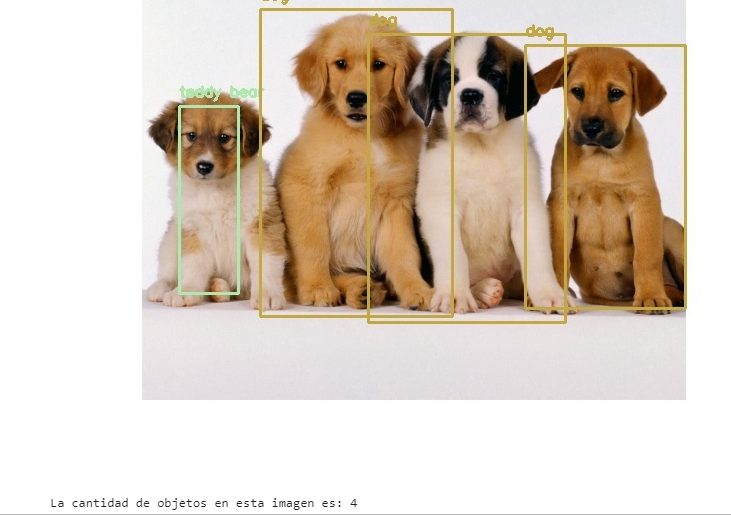
**R-.** Se apunta al endpoint generado en el servidor.













### Preguntas Teóricas (No Código)

**3.1 (1 punto)** En base a la disponibilidad de un servidor con el endpoint */countObjects,* es decir, un modelo de Machine Learning que realice conteo de objetos en imágenes, plantee una aplicación cliente que pueda crearse para construir un Producto de Datos. Detalle qué tipo de problema resolvería, si se puede aplicar a alguna industria en particular, que usuarios tendría y cómo podría entregar valor a dichos usuarios. ¿Qué métricas de desempeño para el Producto de Datos serían adecuadas en este escenario?

**R-.** Por el perfil profesional que tiene el investigador, ha llegado a la conclusión de que un contador de objetos de estas características y con la implementación de otras sería muy útil en los centros de monitoreo, donde cada vez se automatizan más las tareas. Hoy en día se depende mucho del capital humano para discriminar alertas basadas en colores, con esta solución se podría reducir al mínimo la participación humana en centros de monitoreos. Esto es cada vez es más requerido en la industria de telecomunicaciones e infraestructura de datacenter. El tipo de problemas que se resolvería sería que no se vería afectado por la fatiga de las extensas horas de turno de los agentes de monitoreo, o pérdida de atención a los paneles por sueño, necesidades básicas, etc.

Por las características de los paneles de monitoreo, que son web a través de https, en donde todo lo sano y operativamente normal se manifiesta en color verde, se podrían generar screenshot de estos, para que en cuanto se vea alguna alarma en rojo, se tome alguna acción.

Esto debería complementarse con un scheduler que ejecute una captura de pantalla cada cierto tiempo, y la foto resultante pase por el modelo para discriminar cuantas alarmas existen en ese momento. Esto se podría hacer con Python y Selenium.

**3.2 (1 punto)** En el escenario que su producto se implemente y comience a tener usuarios. En base a lo visto en clases: ¿Qué dificultades puede tener en el futuro? Enumere 5 de esas dificultades y comente cuál es la importancia de cada una.

**R-.** A) Según lo visto en clases, y por como están organizado los NOC, convergen redes de varios clientes, y algunos segmentos podrían ser más restrictivos con ciertos protocolos, puertos o servicios que no funcionarán ya que podrían ser catalogados como una amenaza.

**B)** El investigador piensa que las dificultades no vendrían por el lado de la performance del Modelo, ya que resuelve muy bien. Sin embargo, por el lado de la implementación podrían generarse ciertos requisitos por parte de arquitectura: Alta Disponibilidad, Hardening.

### Entrega

**La entrega es por Canvas**, no por correo electrónico.

El plazo de entrega es el **domingo 8 de agosto a las 23.59 hrs**. Se deben entregar todos los archivos realizados para responder la Tarea mediante un enlace a Google Drive. Para responder las preguntas Teóricas puede usar celdas de texto en el programa cliente o adjuntar un informe breve en formato pdf. El trabajo es **individual** (una entrega por estudiante).  
  
Se aceptarán entregas atrasadas con una penalización de 5 décimas en la nota final por cada día no feriado de atraso.

### Recursos

Recuerde que todo el material del codigo visto en clases está acá: <https://drive.google.com/drive/folders/1RY4S2kWtH2h3izBLX8FKoUaYqBMDHP03?usp=sharing>  
  
Al momento de querer probar los cambios en el código siga los siguientes pasos como guía:

* Detener el programa servidor interrumpiendo el Kernel del notebook.
* Realizar los cambios que desee en el código del servidor.
* Volver a correr las celdas de código que tienen las definiciones del programa servidor usando *FastApi*.
* Volver a correr el servidor usando la celda que ejecuta *uvicorn*.
* Realizar los cambios necesarios en el programa cliente.
* Probar con algunas imágenes y analizar si las respuestas del servidor son correctas.

Algunas indicaciones que pueden ayudar a resolver las preguntas de código:

**En el servidor:**

La función de predicción en el endpoint */predict* necesita tener un parámetro adicional para incorporar el nivel de confianza. Agregue este parámetro antes del parámetro File. Esto es necesario porque File tiene un valor por defecto y eso debe ser declarado al final.

*cv.detect\_common\_objects* acepta el párametro de nivel de confianza, el cual es de tipo número floating point (*float* en Python).

**En el cliente:**

Se puede agregar un nuevo parámetro a la URL extendiendola con un *&*, seguido del nombre del parámetro y su valor. El nombre de este nuevo parámetro debe ser idéntico al usado dentro de la función de predicción del servidor. Un ejemplo de esto sería: *misupermodelo.com/predict?model=yolov3-tiny&newParam=value*