

Desarrollo de Productos de Datos

Tarea 1: Modelo de ML Cliente-Servidor

1. Descripción General

Vamos a expandir lo realizado en el [Laboratorio de Implementación](#) de un Modelo de Machine Learning visto en [clase](#). Para desarrollar la Tarea debe basarse en el código proporcionado y empezar a construir las mejoras y nuevas funcionalidades sugeridas en las siguientes preguntas.

2. 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](#))

```
# @app.post("/countObjects")
@app.post("/predict")
def prediction(model: Model, confidence=0.2, file: UploadFile = File(...)):

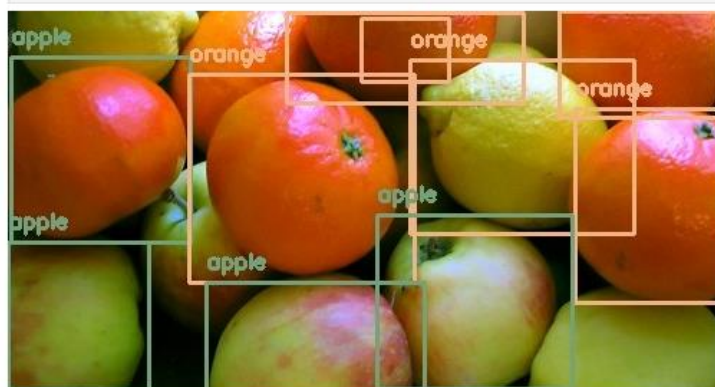
    # 1. Validar el archivo de entrada
    filename = file.filename
```

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:

```
[9]: display_image_from_response(prediction, 0.4)
```



La cantidad de objetos en esta imagen es: 10

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.

```


@app.post("/countObjects")
#@app.post("/predict")
def prediction(model: Model, confidence=0.2, file: UploadFile = File(...)):

    # Crear una imagen que contenga las cajas delimitadoras y etiquetas
    output_image = draw_bbox(image, bbox, label, conf)
    print("Number of objects in this image are " + str(len(label)))
  
```

```

INFO:      127.0.0.1:53280 - "POST /countObjects?model=yolov3-tiny&param=0.9 HTTP/1.1" 200 OK
Number of objects in this image are 3
INFO:      127.0.0.1:53282 - "POST /countObjects?model=yolov3-tiny&param=0.9 HTTP/1.1" 200 OK
Number of objects in this image are 1
INFO:      127.0.0.1:53284 - "POST /countObjects?model=yolov3-tiny&param=0.9 HTTP/1.1" 200 OK
Number of objects in this image are 1
INFO:      127.0.0.1:53286 - "POST /countObjects?model=yolov3-tiny&param=0.9 HTTP/1.1" 200 OK
Number of objects in this image are 1
INFO:      127.0.0.1:53288 - "POST /countObjects?model=yolov3-tiny&param=0.9 HTTP/1.1" 200 OK
Number of objects in this image are 4
  
```

server.ipynb client.ipynb



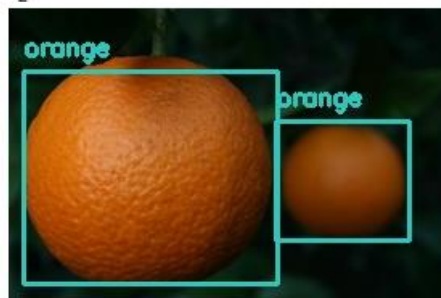
```

=====
Imagen procesada: oranges.jpg

La cantidad de objetos en esta imagen es: 2
Objeto detectado: orange, con un nivel de confianza de 0.6185588836669922

Objeto detectado: orange, con un nivel de confianza de 0.5561690330505371

2
  
```



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.

```
[2]: base_url = 'http://localhost:8000'
    #endpoint = '/predict'
    endpoint = "/countObjects"
    #model = 'yolov3-tiny'
    model = 'yolov3'
    #param = '0.6'
    param = str(float(0.9))
```

Para consumir el modelo, vamos a agregar el endpoint a la URL base para obtener la URL completa.



La cantidad de objetos en esta imagen es: 1

```
: display_image_from_response(prediction, 0.3)
```

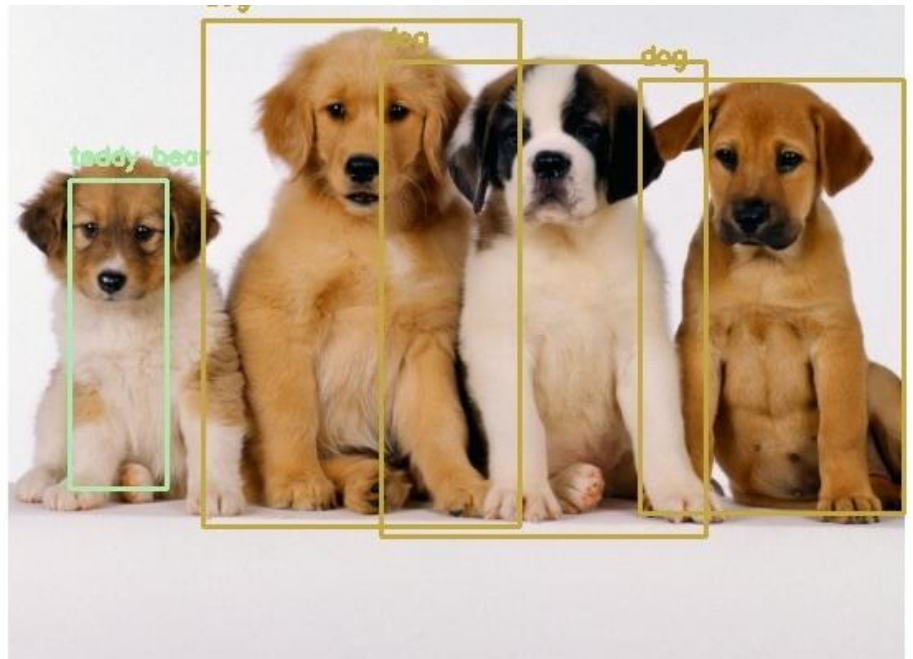


La cantidad de objetos en esta imagen es: 2

```
[167]: display_image_from_response(prediction, 0.3)
```



La cantidad de objetos en esta imagen es: 3



La cantidad de objetos en esta imagen es: 4

¡Todo funcionó bien!



La cantidad de objetos en esta imagen es: 5

3. 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-. El investigador cree que de por si el modelo funciona bien, sin embargo, un propósito útil sería ponerlo a prueba en circunstancias de baja visibilidad como la fauna marina al hacer investigaciones científicas. Es decir de todo el material que puedan reunir los biólogos marinos puede después ser procesado utilizando el contador de objetos, tal vez los algoritmos puedan detectar algo que por fatiga o falta de tiempo el ojo humano pasa por alto. Además, no tendría que ser preciso en clasificar lo que detecta, en primera instancia bastaría con que lo detecte, por lo tanto, no habría que setear el confidence tan alto. El investigador estima que esto sería de gran utilidad para procesar el material de cada expedición con posterioridad o en tiempo real a medida que se generan los registros. En ese sentido, la métrica de desempeño indicada sería la de precisión.

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) Lo ideal sería que el servidor se encuentre hospedado en la nube, públicamente, así se podrá acceder a él desde cualquier lugar, sin necesidad de vpn o filtros especiales. Esa sería la primera dificultad, pero la más simple de superar.

B) No se evaluó, pero el investigador presume que el modelo no es muy eficiente procesando imágenes de fauna marina, o basura, o sedimentos, por lo que también sería una excelente oportunidad para que una vez el contador de objetos discrimine las imágenes, se puedan utilizar nuevamente técnicas de ML para entrenar un nuevo modelo que clasifique la biodiversidad que hay en el mar.

C) El investigador supone que una herramienta de estas características que es pública y de libre acceso, será muy utilizada, de rápida adopción y si cumple su propósito, muchas otras investigaciones comenzarán a hacer uso de él. Lo que generará un alto consumo de cpu, bw, memoria, espacio en disco, logs, etc. Esto debería compensarse con una arquitectura en alta disponibilidad, con balanceo de carga y sistemas de almacenamiento, ya que con lo que se está tratando son precisamente: fotos.

D) El producto tal vez esté siendo utilizado mucho por comunidades enteras, lo que pueda impedir su uso para otros propósitos, por lo que habría que levantar más servidores, lo que implicará mayor demanda de servicios, lo que

finalmente se traducirá en que se convertirá en una app crítica, y generará gastos.

E) Varios: imprevistos, mal uso de la aplicación, sabotaje, robo de información, etc.

4. 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.

5. Recursos

Recuerde que todo el material del código 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 parámetro 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 extendiéndola 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*