DOCUMENTACIÓN AI MODEL

JUAN DAVID PEÑA ROLDÁN

20222209141

ELECTIVA CIENCIAS DE LA COMPUTACIÓN I

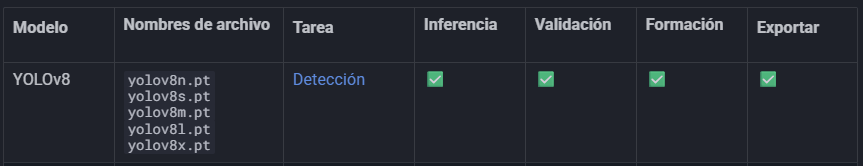
UNIVERSIDAD SURCOLOMBIANA

NEIVA, HUILA

2025

Una vez definida la estructura del proyecto, necesitamos entrenar un modelo de ia que fuese capaz de distinguir cuando se le muestre un cultivo sano de lechuga o por lo contrario un cultivo que presentara algún tipo de enfermedad o indicios de empezar a malograrse.

El modelo escogido para el model detection fue YoloV8 [Explorar Ultralytics YOLOv8 - Ultralytics YOLO Docs](https://docs.ultralytics.com/es/models/yolov8/#supported-tasks-and-modes), un modelo bastante robusto para detección de objetos según entrenamiento, dentro de YoloV8 existen varios modelos de entrenamiento, unos más pesados que otros, el usado para este proyecto fue “yolov8m.pt” un modelo de entrenamiento de exigencia media, acá está la ejemplificación de los distintos modelos:



Para poder guiar al modelo en el transcurso del proyecto se siguieron estos pasos:

* **Obtención de imágenes de muestra**
* **Creación del dataset**
* **Preparación del entorno de entrenamiento**
* **Entrenamiento del modelo**
* **Pruebas y resultados**

Todo esto con la finalidad de entrenar el modelo de la mejor manera y fuese capaz de poder detectar entre estos dos casos según sea requerido

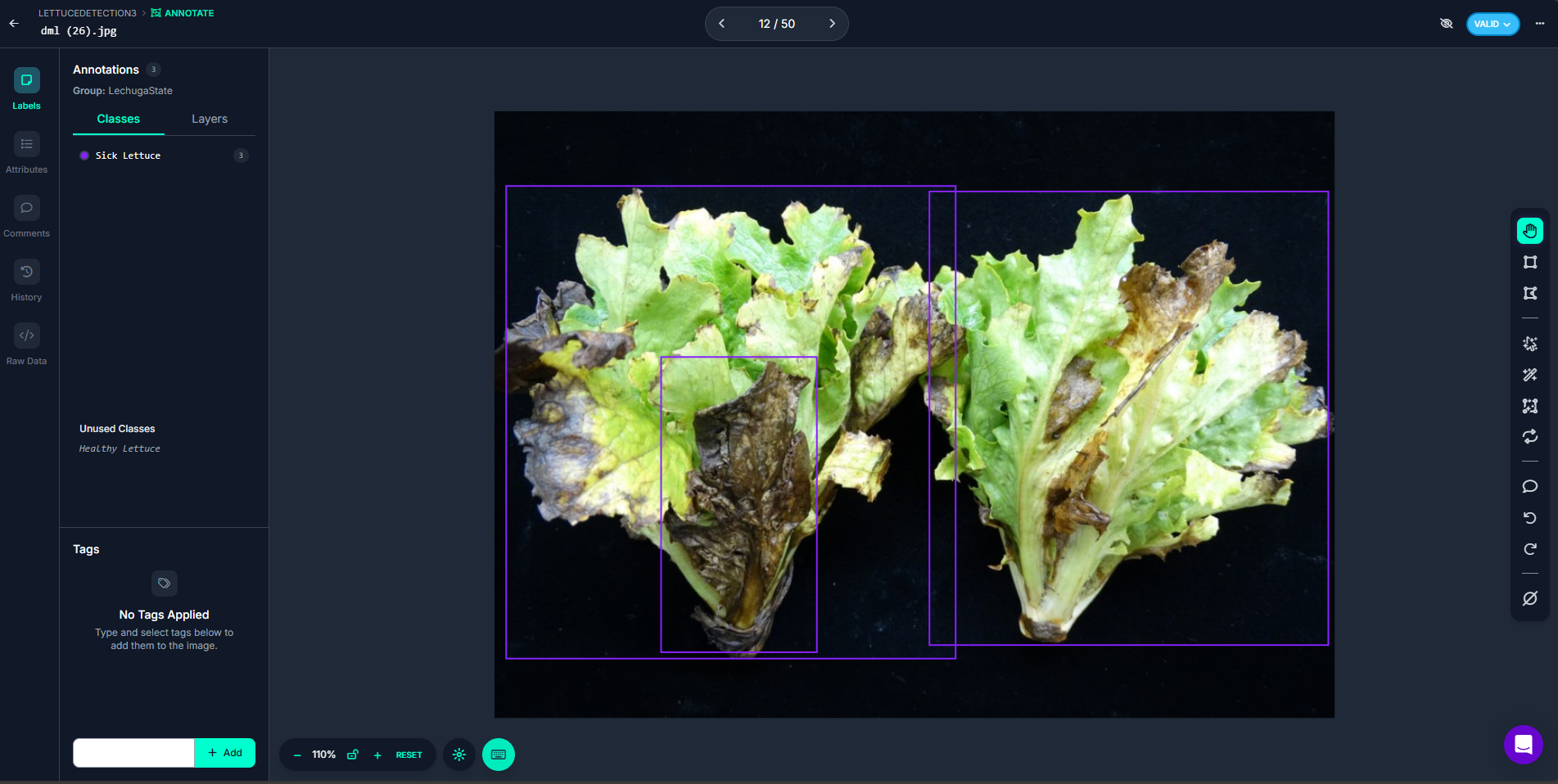
1. **Obtención de imágenes de muestra**

Para el modelo necesitamos una serie de imágenes de buena resolución que contenga ejemplo de como se vería el cultivo de lechuga en los dos casos, sea sano o no, para sacar estas imágenes se usó un dataset de imágenes de lechuga ya existente en Kaggle ([Lettuce NPK dataset](https://www.kaggle.com/datasets/baronn/lettuce-npk-dataset/data)), a partir de estas imágenes fue donde se sacaron las muestras para entrenar el modelo

1. **Creación del dataset**

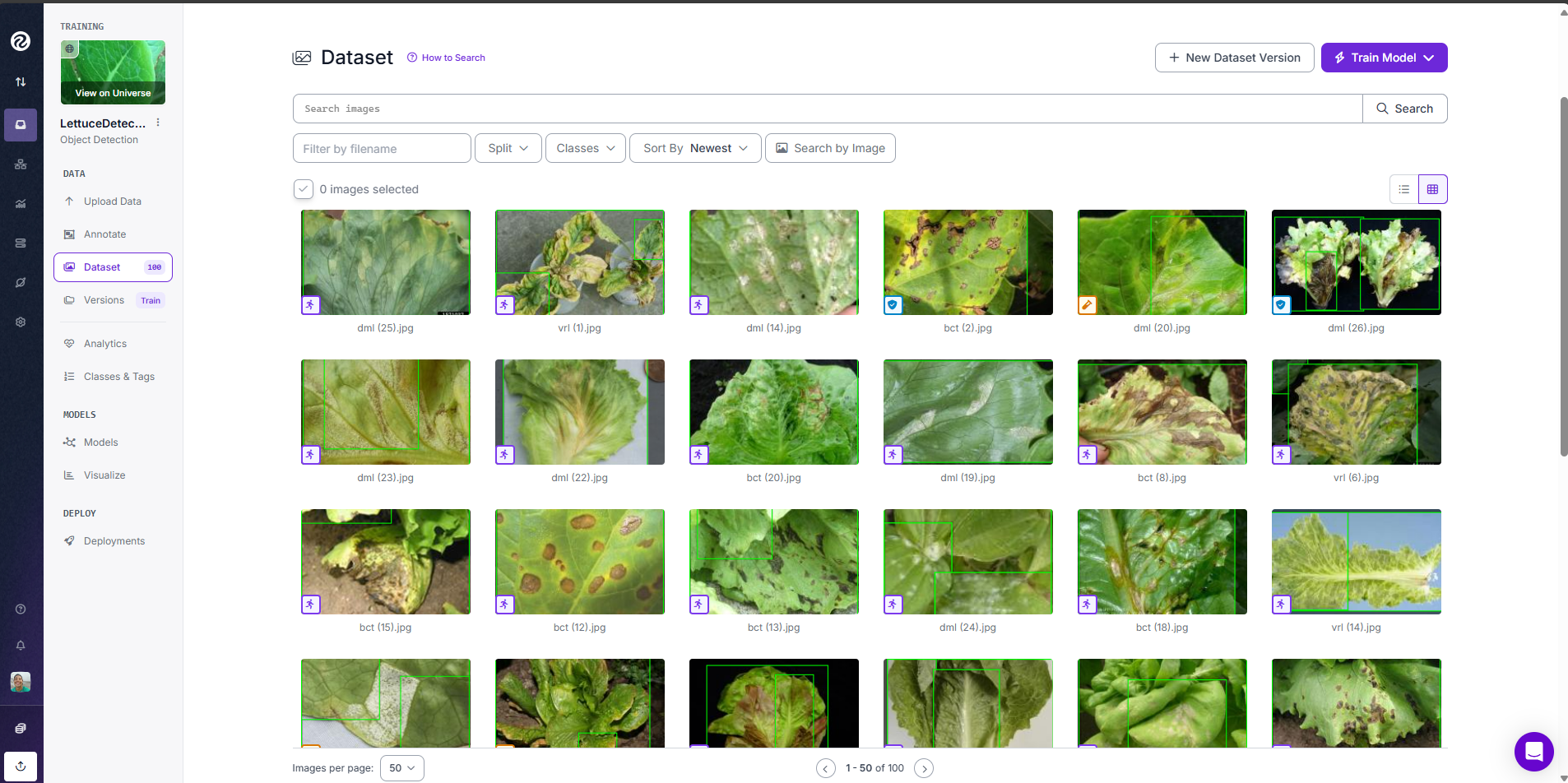
Para temas de productividad se optó por usar Roboflow como herramienta para crear el dataset con el cual se iba a entrenar el modelo, porque no solo basta con tener las imágenes, hay que realizar un labeling para que el modelo comprenda mejor que es lo que está observando.

La muestra total fue de 100 imágenes, repartidas entre el labeling de “Healthy Lettuce” para definir a las lechugas sanas y “Sick Lettuce” para definir a las que no lo son, el labeling consiste en hacer un tipo de recorte bordado a la muestra, para que el modelo sepa a que enfocar y así mismo tener en cuenta patrones.



En este caso tenemos una lechuga con claros signos de estar podrida, una lechuga clasificada como “Sick Lettuce”, el labeling se hace con el fin de reconocer estos signos, así como se ve en la imagen y por eso mismo es clasificada, resaltando los factores diferenciadores entre los dos casos.

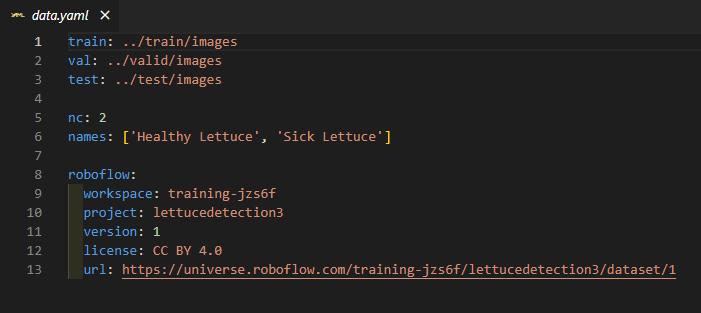
Ya una vez terminado el dataset, se tendría que ver algo así:



Ahora lo que sigue sería exportar el dataset con los dos casos a nuestro proyecto en el entorno de entrenamiento elegido con anotación de imágenes en formato YOLO con el siguiente código:



Una vez puesto el código se importará la información necesaria a nuestro proyecto con la información del dataset, con un archivo importante llamado en este caso “data.yaml”, que contiene la siguiente información:



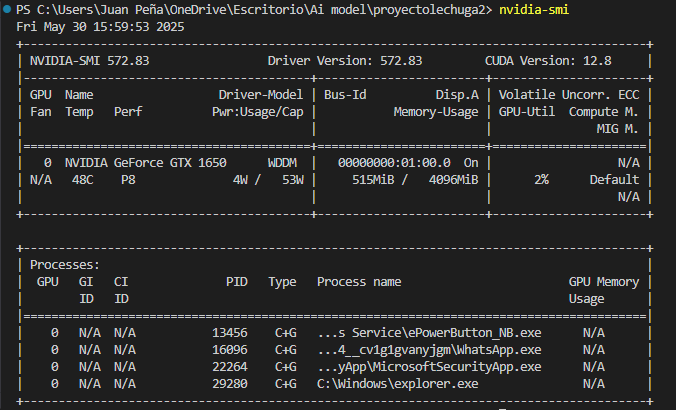
Acá se puede observar que el dataset está correcto con los dos labeling esperados.

1. **Preparación del entorno de entrenamiento**

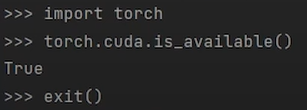
El entorno usado fue VsCode con un entorno virtual de Python 3.8, donde para preparalo para entrenar el modelo se hizo lo siguiente; primeramente, se creó una carpeta con el nombre que le quisiéramos dar al proyecto, en este caso “proyectolechuga2”, donde ya con el paso anterior de importación del dataset, queda instalar las librerías necesarias de Yolov8 para empezar con el entrenamiento, se instalan las librerías de ultralytics con el siguiente comando:



Una vez instalada esta librería, se pone el siguiente comando para empezar a entrenar el modelo, pero antes de esto se tiene que tener en cuenta un factor importante, el modelo debería ser entrenado con una GPU para términos de un entrenamiento más eficiente y ya que se está haciendo en local, en este caso, se cuenta con una GPU *Nvidia Gforce GTX 1650*, pero debemos comprobar que es reconocida y que el modelo va a ser entrenado bajo ella para evitar un estrés innecesario al procesador y su gráfica integrada, ya que en este caso se cuenta con una GPU dedicada para ello, se ponen los siguientes comandos:



Esto para comprobar que en efecto se reconoce la GPU con sus drivers, dependiendo de la versión de CUDA de la GPU, así mismo varia la versión que se instala de torch para el entrenamiento con la gráfica, se pone el siguiente comando en Python para comprobar el usaje de la GPU dentro del entorno



Cuando nos arroja el valor de true es porque se está usando la GPU dentro del entorno, ahora si se puede proceder al entrenamiento del modelo.

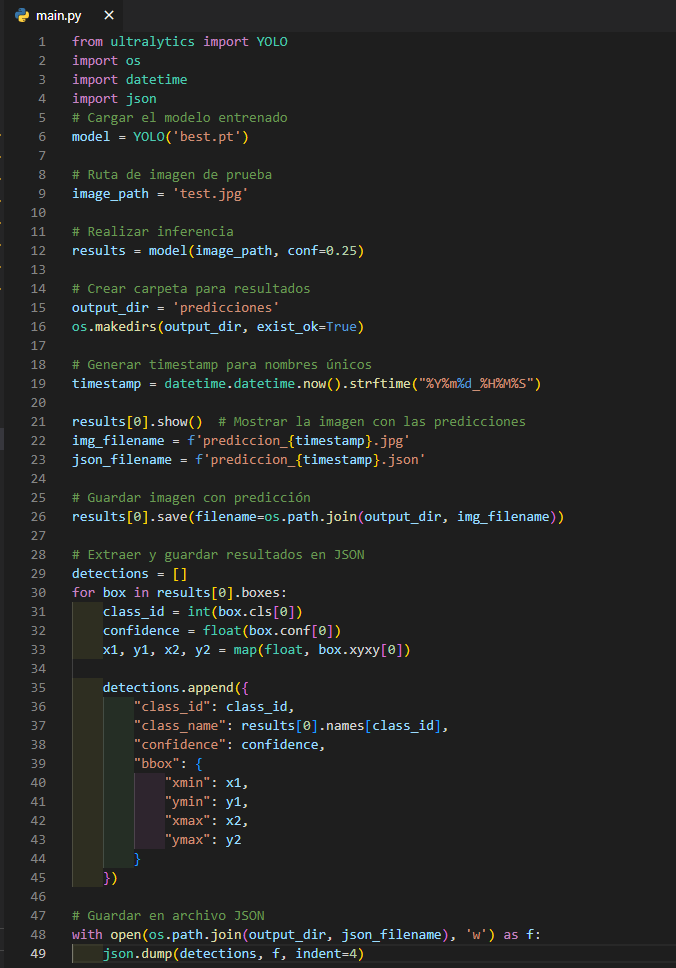
1. **Entrenamiento del modelo**



El comando utilizado sirve para entrenar un modelo de detección de objetos usando YOLOv8. Está configurado para realizar una tarea de detección (task=detect) en modo entrenamiento (mode=train) durante 30 épocas (epochs=30). El entrenamiento se basa en la información proporcionada en el archivo data.yaml, que contiene las rutas a tus imágenes de entrenamiento y validación, así como las clases que quieres detectar. El modelo de partida que estás utilizando es yolov8m.pt, una versión de tamaño medio del modelo YOLO preentrenado. Además, las imágenes se redimensionan a 640x640 píxeles (imgsz=640) y se procesan en lotes de 4 imágenes a la vez (batch=4), lo cual ayuda a optimizar el uso de memoria durante el entrenamiento. Al finalizar, se obtiene el modelo entrenado (guardado como best.pt junto con estadísticas de rendimiento como precisión y exactitud

1. **Pruebas y resultados**

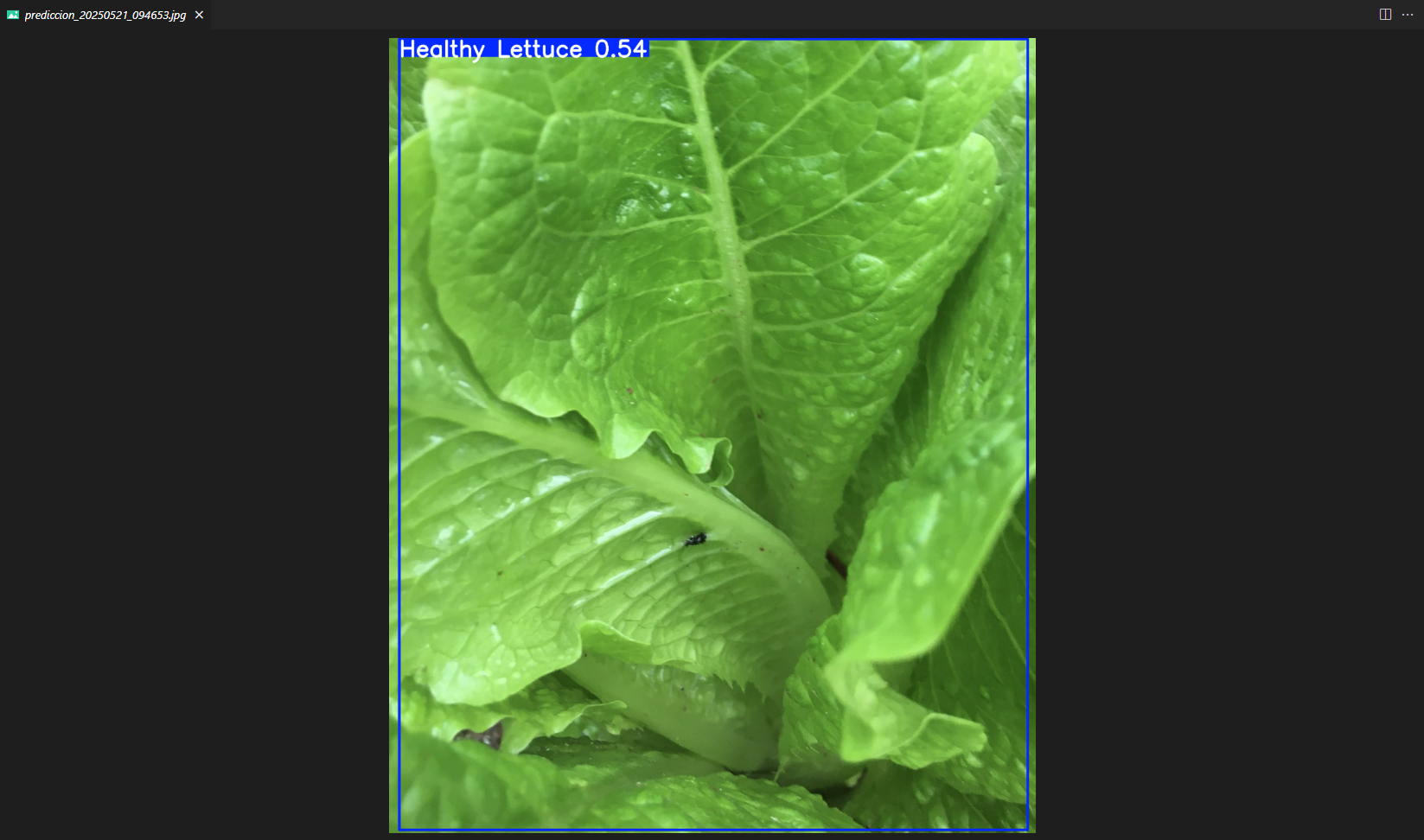
Ya teniendo el modelo entrenado se procede a hacer las pruebas para comprobar que el proceso fue exitoso y en realidad el modelo es capaz de hacer su cometido, se creó un main.py que contiene el siguiente código:



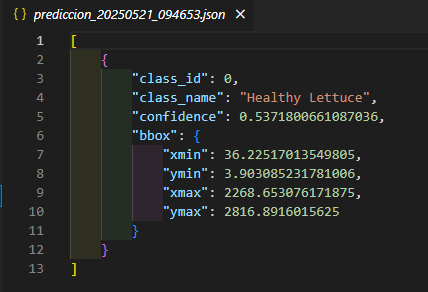
Este código importa las librerías necesarias y carga el modelo entrenado de YOLOv8 llamado best.pt y lo utiliza para hacer la detección de las lechugas sobre una imagen de prueba llamada test.jpg. Luego, crea una carpeta llamada predicciones (si no existe) donde guarda los resultados. Para evitar sobrescribir archivos, genera un nombre único para cada salida usando la fecha y hora actual. El modelo realiza la inferencia con un umbral de confianza mínimo de 0.25 y muestra visualmente la imagen con las detecciones. Posteriormente, guarda esa imagen con las cajas de predicción dibujadas, y además extrae información detallada de cada objeto detectado (como la clase, el nombre de la clase, el nivel de confianza y las coordenadas de la caja delimitadora). Esta información se organiza en una estructura tipo lista y se guarda en un archivo .json dentro de la misma carpeta, usando el mismo nombre único basado en la hora. Así, cada vez que se ejecuta el código, se almacenan tanto la imagen con las predicciones como un archivo JSON con los datos de esas detecciones, sin reemplazar los resultados anteriores.

Una vez es ejecutado el código, estos fueron los resultados

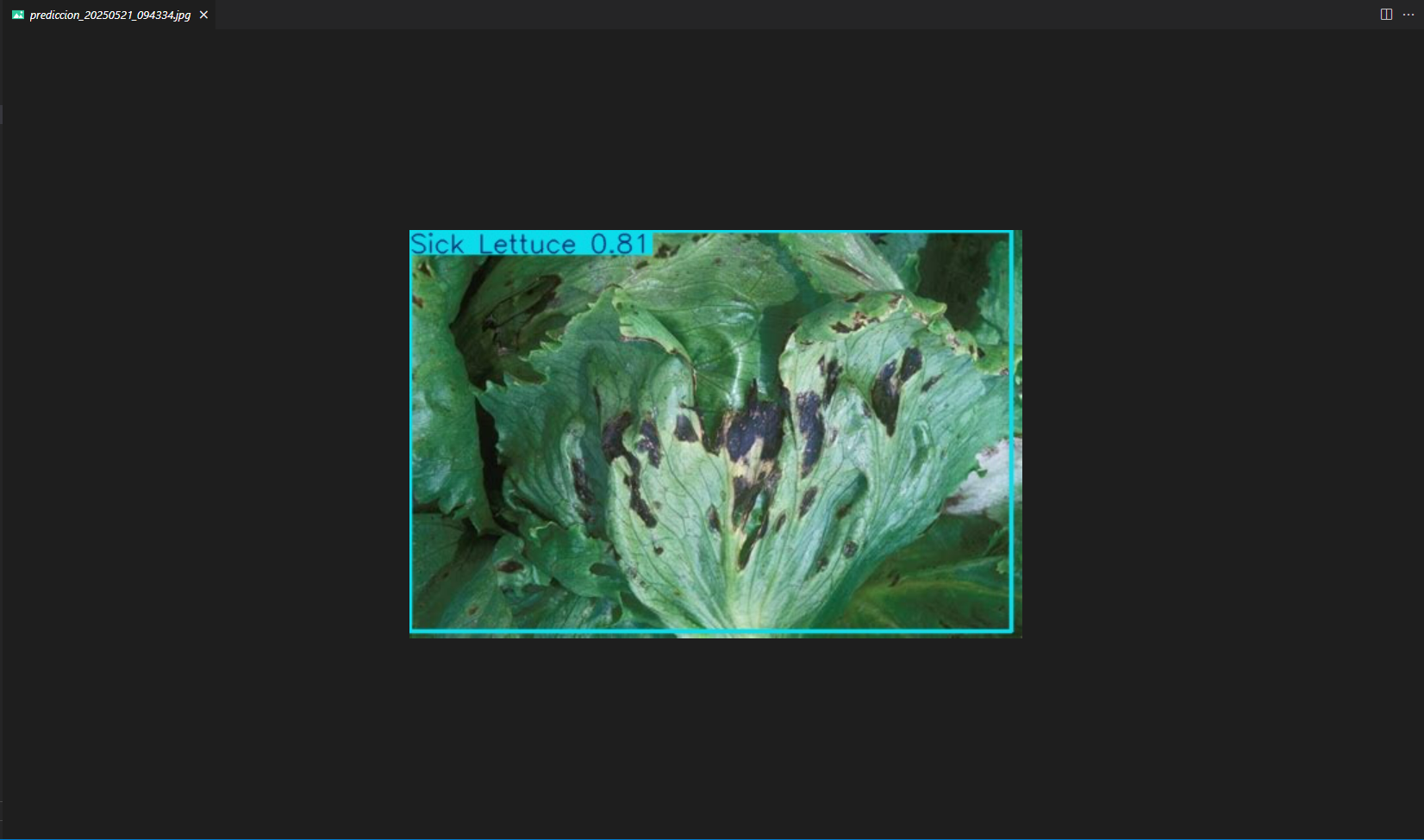
* Healthy lettuce



.json:



* Sick Lettuce



.json:



Con estas pruebas satisfactorias, se puede concluir que en efecto el modelo fue entrenado exitosamente, siendo capaz de diferenciar entre una lechuga en buen y mal estado; con esto ya puede ser aplicado dentro del proyecto.