docker\_1/

├── Dockerfile

├── cloudbuild.yaml

├── main.py # API Flask que carga modelo\_cancer.pkl

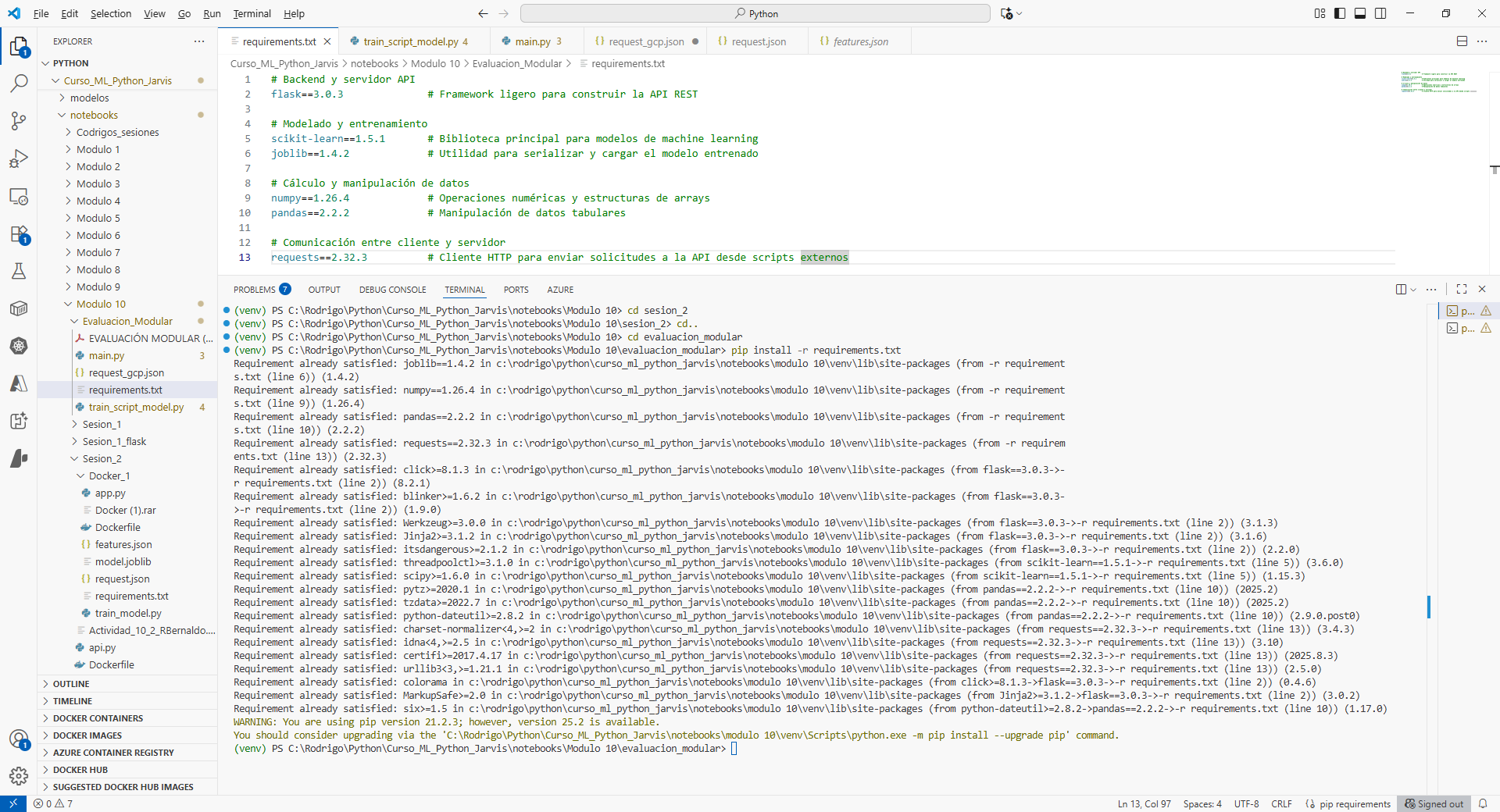
├── train\_script\_cancer.py # Script de entrenamiento

├── modelo\_cancer.pkl # Modelo entrenado (generado por el script)

├── requirements.txt

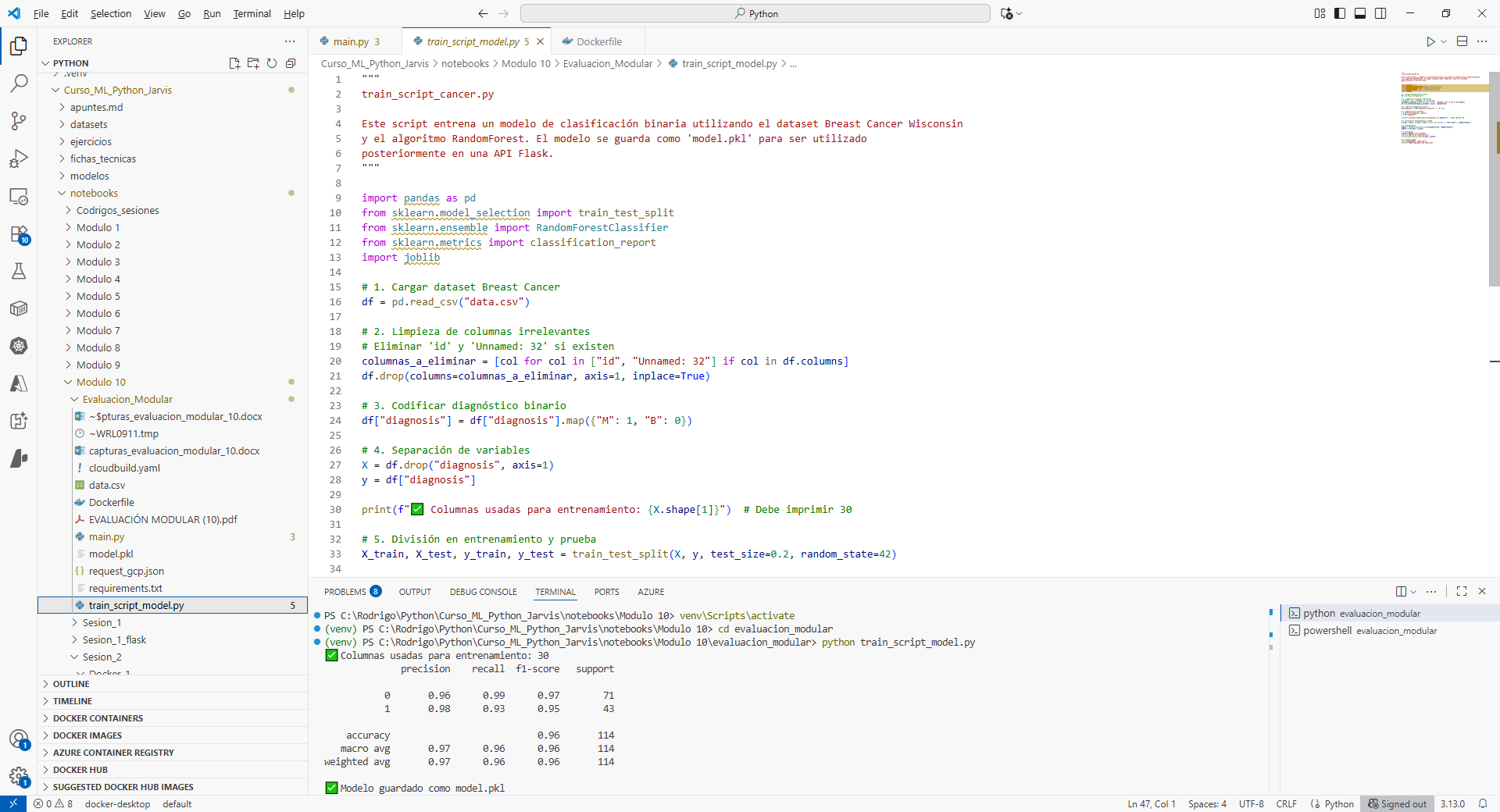
├── request\_gcp.json

├── data.csv # Dataset original (no se sube a producción)

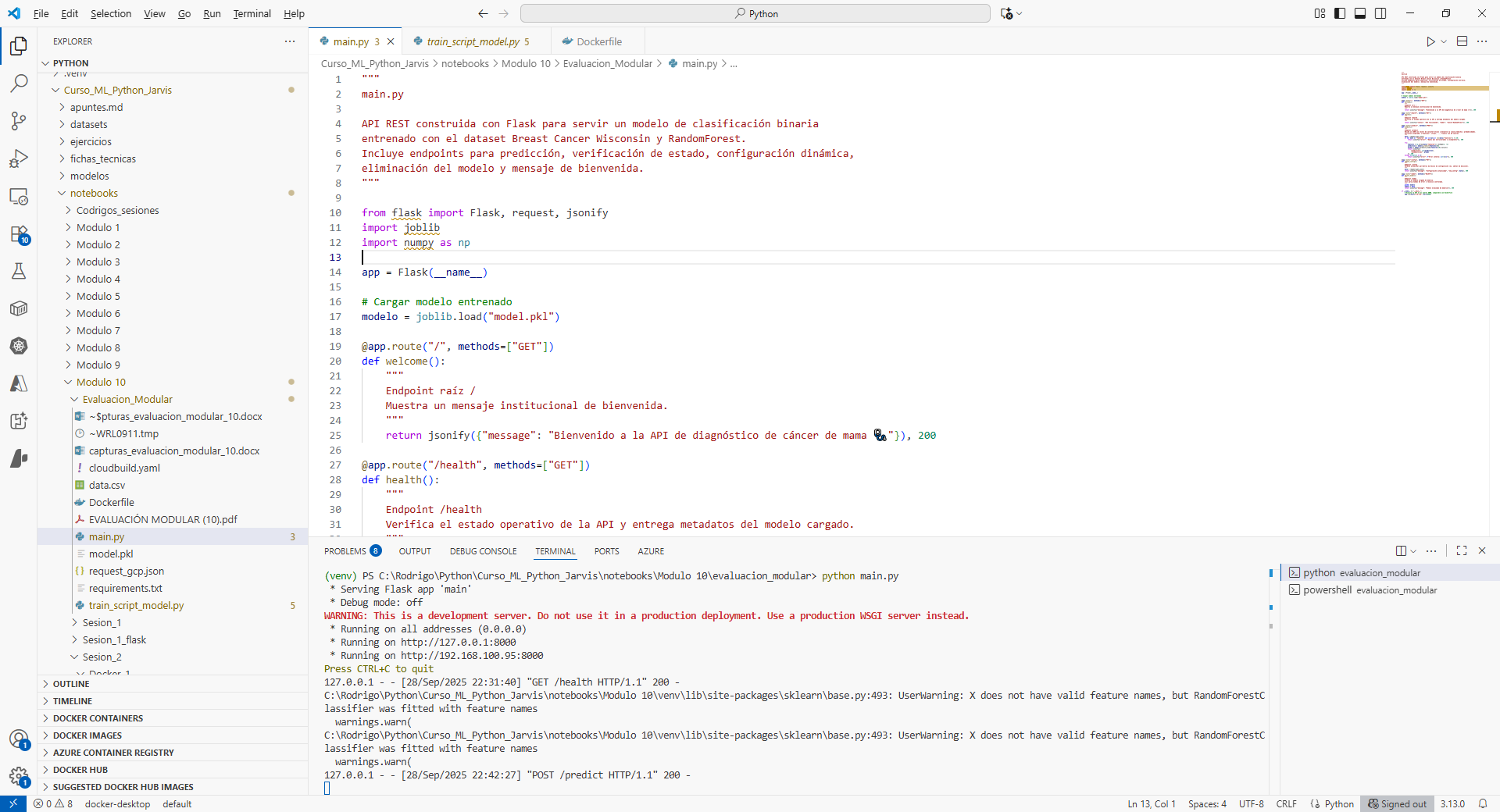


Para esta evaluación modular, prioricé un despliegue reproducible y funcional de la API ML en GCP. Eliminé dependencias visuales como Streamlit y Seaborn en la imagen de producción para optimizar el build y reducir el peso del contenedor. Sin embargo, mantengo herramientas de validación en mi entorno local para análisis complementarios si se requieren en defensa oral.

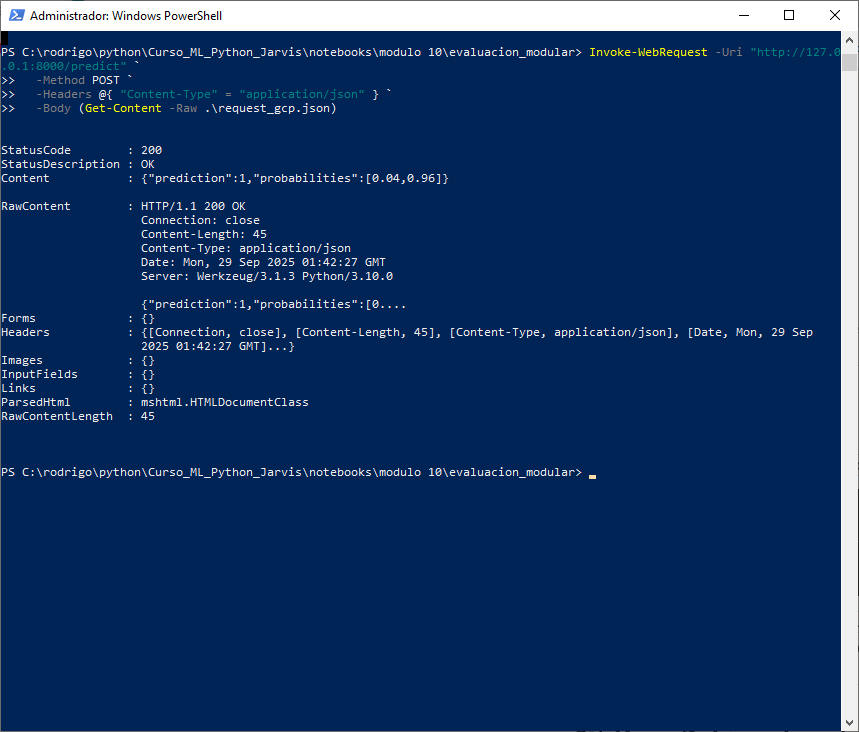
Incluí requests en el archivo requirements.txt porque utilizo scripts de validación en Python que envían solicitudes POST a la API desplegada en Cloud Run. Aunque no es parte del backend, es esencial para pruebas funcionales reproducibles. json no se incluye porque es parte del core de Python



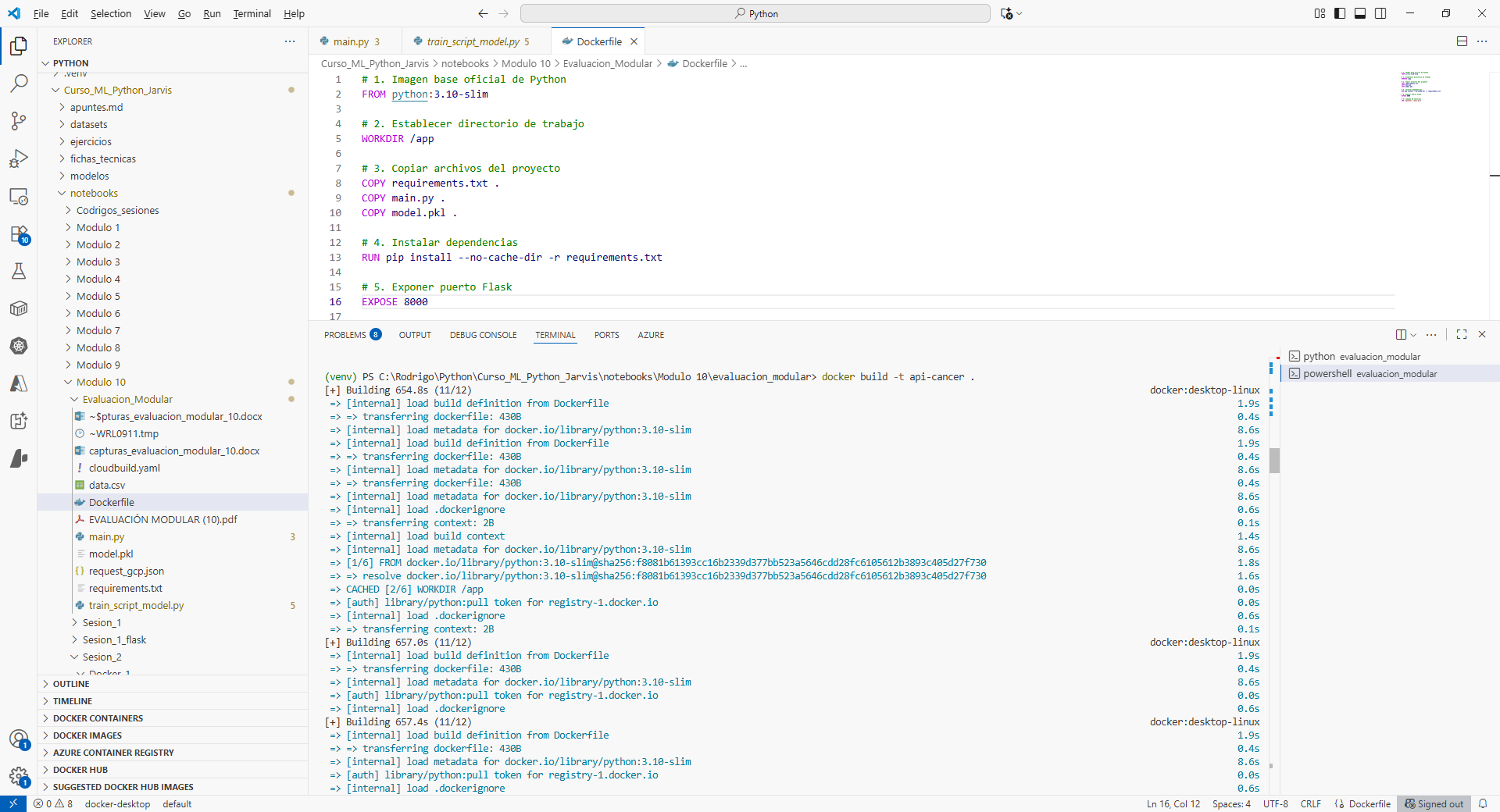
El script elimina dinámicamente las columnas irrelevantes (id, Unnamed: 32) si están presentes, asegurando que el modelo se entrene con las 30 características válidas. Esto evita errores de inferencia en la API y garantiza reproducibilidad en cualquier entorno.

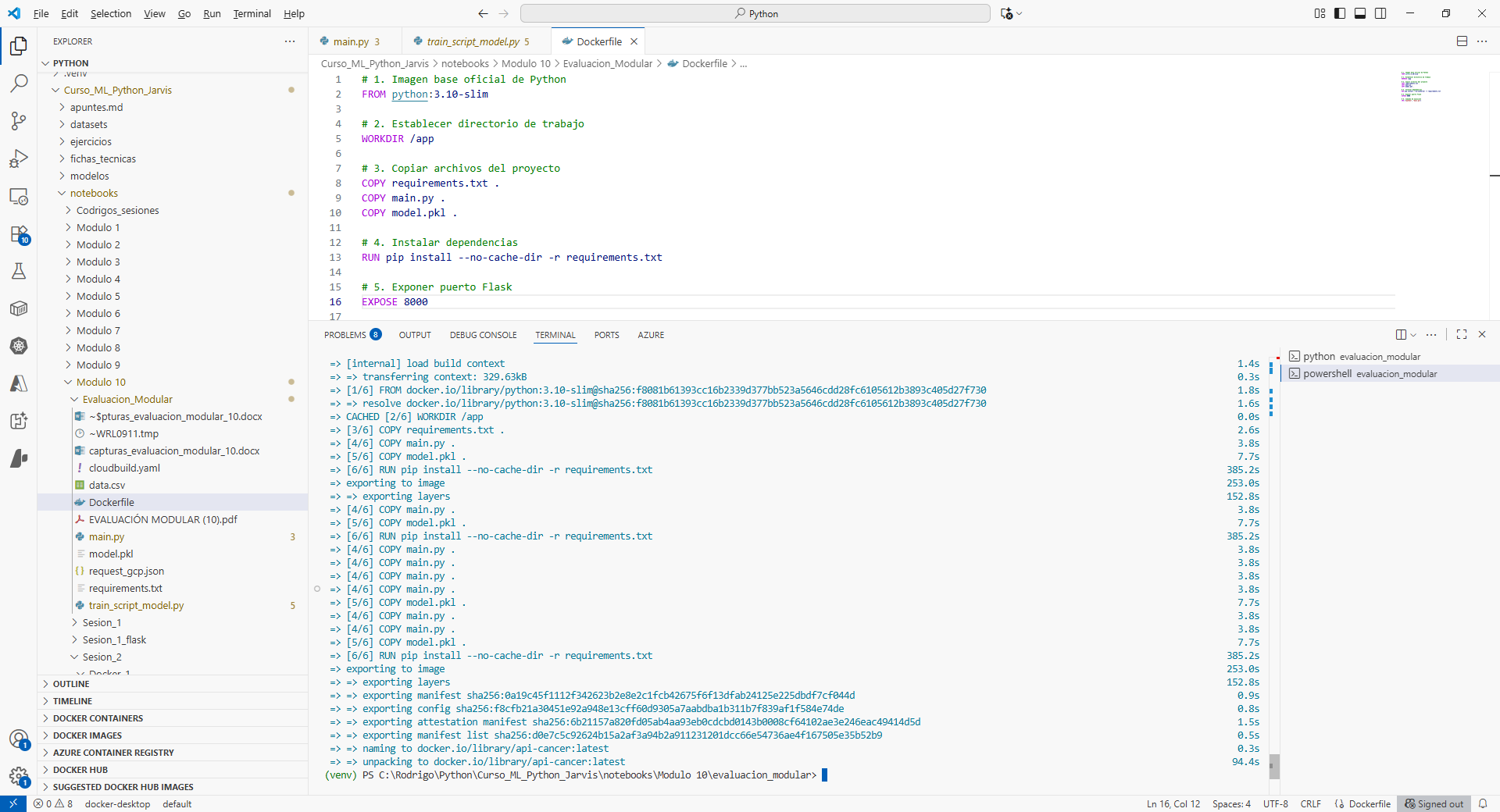


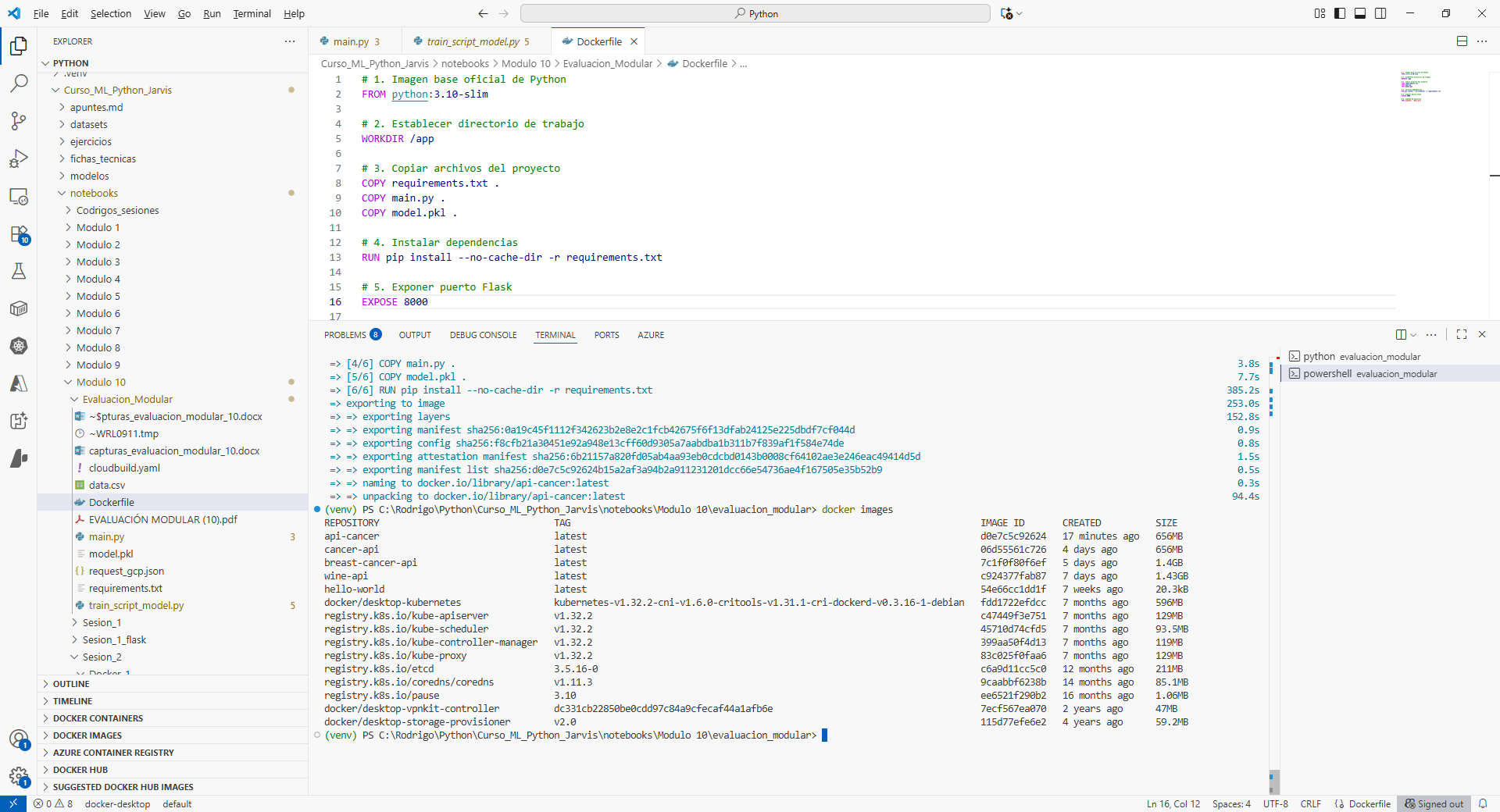
El archivo request\_gcp.json contiene un vector de 30 características extraídas del dataset Breast Cancer Wisconsin. Este formato es compatible con el endpoint /predict desplegado en Cloud Run, y permite validar el funcionamiento de la API con datos reales. La estructura es reproducible, defendible y puede ser utilizada en pruebas automatizadas o manuales.

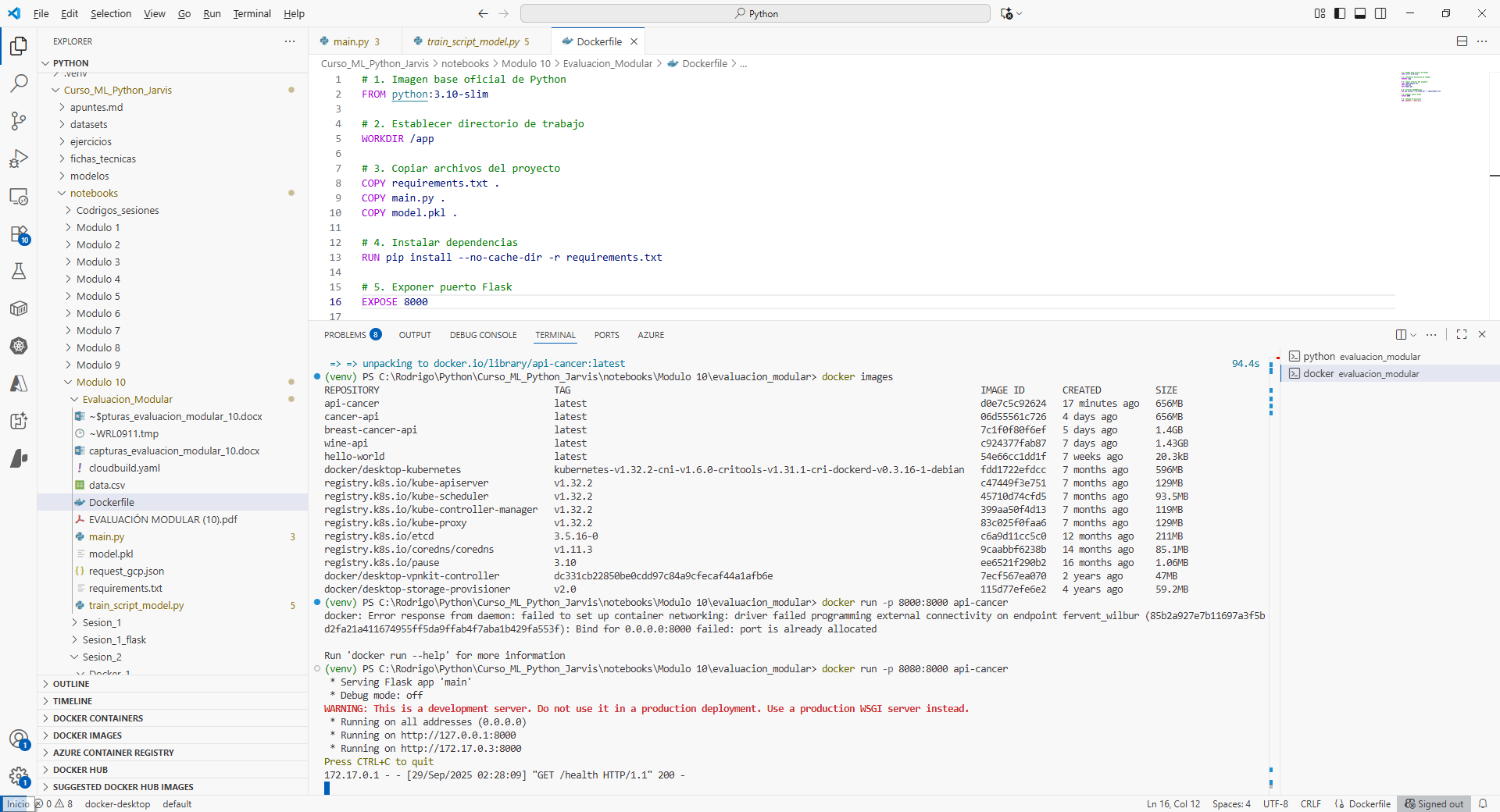


“La API fue validada localmente con una entrada de prueba que contiene las 30 características esperadas. El modelo responde con una predicción binaria y sus probabilidades, confirmando que el entrenamiento, la limpieza de datos y la inferencia están correctamente alineados. El contenedor Docker encapsula todo el entorno, lo que garantiza reproducibilidad y portabilidad para su despliegue en GCP.”

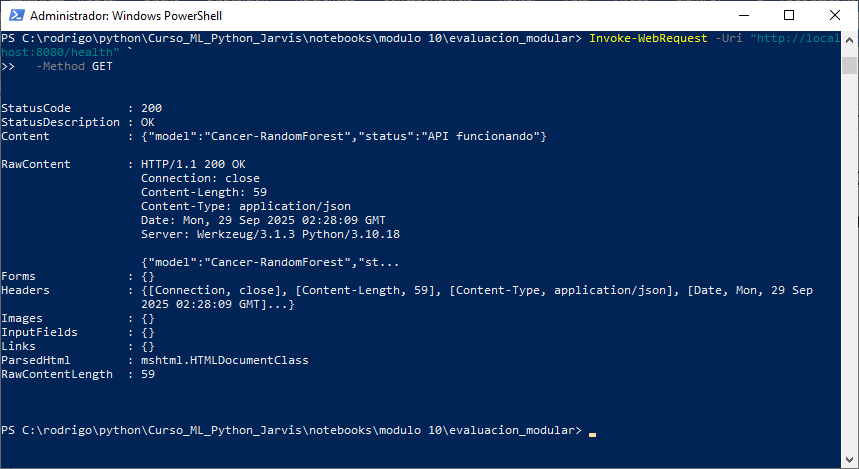


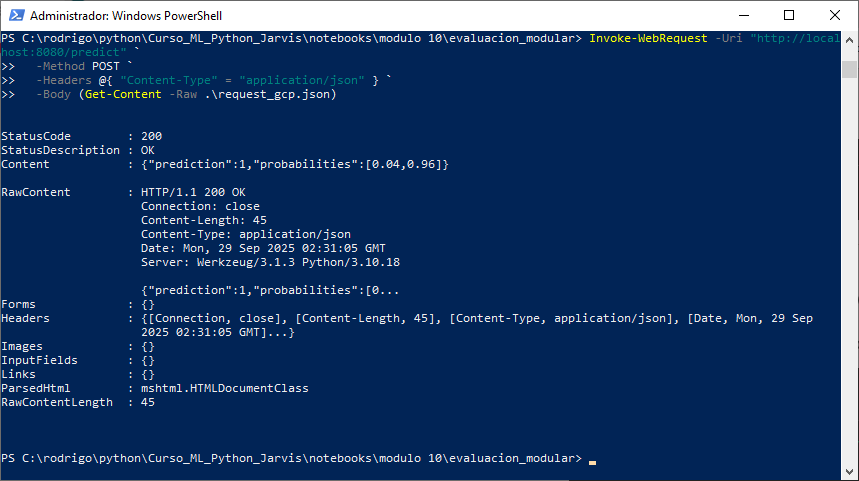




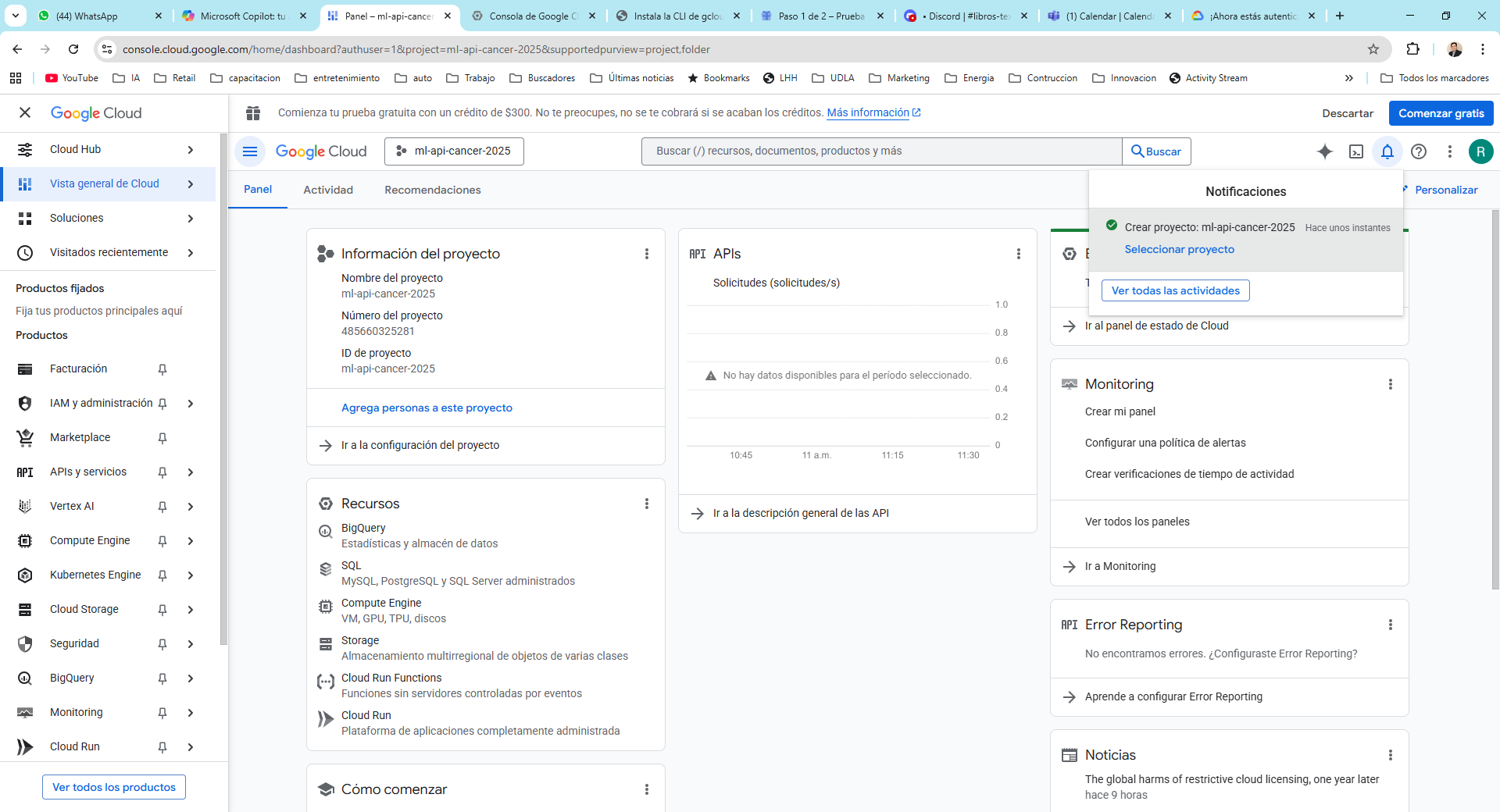
Verificación de la imagen de Docker creada. 

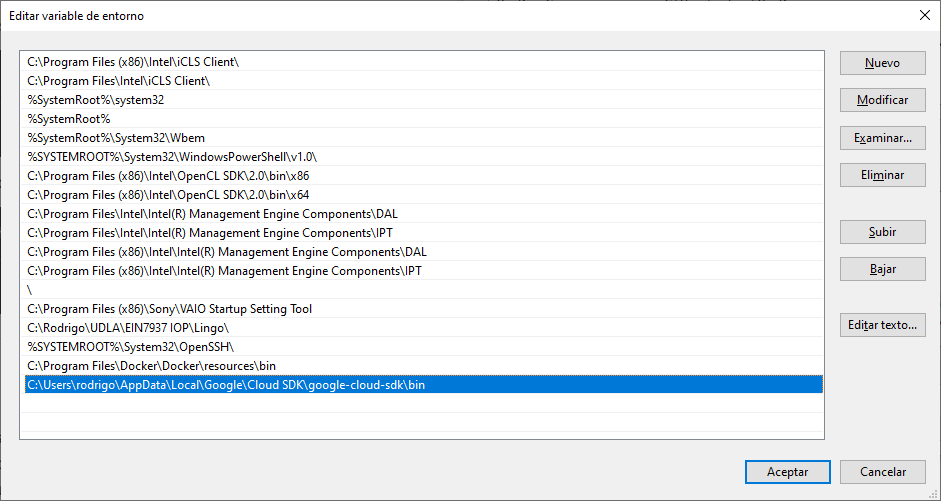
Al probarla, se vio que el puerto 8000 esta ocupado, por lo que se expone externamente con puerto 8080, aunque internamente sigue escuchando el puerto 8000.



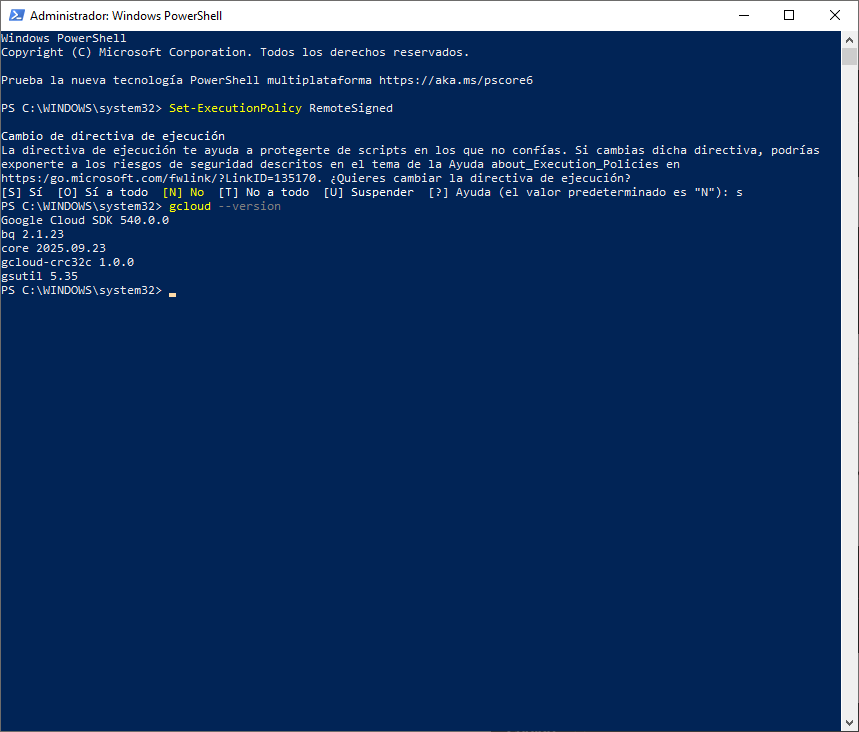


Resultado de prueba en powershell.

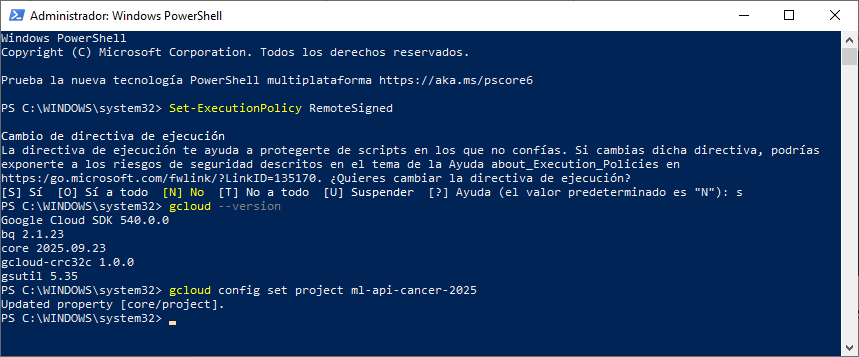




Agregar la ruta del CLI de Google Cloud al Path del sistema permite ejecutar comandos gcloud desde cualquier terminal. Esto habilita la automatización reproducible del despliegue y la interacción con los servicios de GCP desde PowerShell o VS Code.



La política RemoteSigned permite ejecutar scripts locales como gcloud.ps1 sin requerir firma digital, manteniendo la seguridad frente a scripts remotos. Esto habilita el uso completo del CLI de Google Cloud desde PowerShell, necesario para automatizar el despliegue reproducible de la API.



Cambiar la política de ejecución a RemoteSigned permite ejecutar scripts locales como gcloud.ps1 sin comprometer la seguridad del sistema. Esto habilita el uso completo del CLI de Google Cloud desde PowerShell, necesario para automatizar el despliegue reproducible de la API.

| **Archivo** | **Propósito técnico** |
| --- | --- |
| requirements.txt | Define las dependencias necesarias para ejecutar la API y el modelo |
| train\_script\_model.py | Entrena el modelo de cáncer de mama y lo guarda como modelo\_cancer.pkl |
| main.py | API Flask que carga el modelo y expone endpoints /, /health, /predict, etc. |
| request\_gcp.json | JSON de prueba para el endpoint /predict en Cloud Run |

$body = Get-Content -Raw -Path ".\request\_gcp.json"

Invoke-RestMethod -Uri "https://mi-api-modelo-xxxxxx.a.run.app/predict" `

-Method POST `

-Headers @{ "Content-Type" = "application/json" } `

-Body $body