

App de ML

Por:

Ignacio Ortega

30 de Septiembre del 2025

Índice general

1. Leeme	1
1.1. Instrucciones de instalación	1
1.2. Descripción de archivos importantes	1
1.3. Repositorios	2
2. Evaluación Módulo 10	3
2.1. Entrenamiento y serialización del modelo	3
2.2. Desarrollo de API con Flask	3
2.3. Dockerización del sistema	4
2.4. Automatización CI/CD (opcional)	4
3. Reflexiones	9

Capítulo 1

Leeme

1.1. Instrucciones de instalación

Simplemente descomprimir el archivo `.zip`.

1.2. Descripción de archivos importantes

Dentro del `.zip` encontrará:

- `./requirements.txt`
Este archivo contiene los módulos de python necesarios para hacer funcionar la API y entrenar el modelo random forest classifier.
- `./Dockerfile`
Contiene la configuración que debe tener la máquina virtual como archivos, directorios y puertos, para el correcto funcionamiento de la API dentro de esta.
- `./serializar_modelo`
Es la carpeta que contiene el contenido para serializar el modelo y el modelo serializado.
- `./serializar_modelo/Modelo_Breast_Cancer.pkl`
Es el modelo serializado.

1.3. Repositorios

- Repositorio GitHub
Link
- Repositorio Docker
Link

Capítulo 2

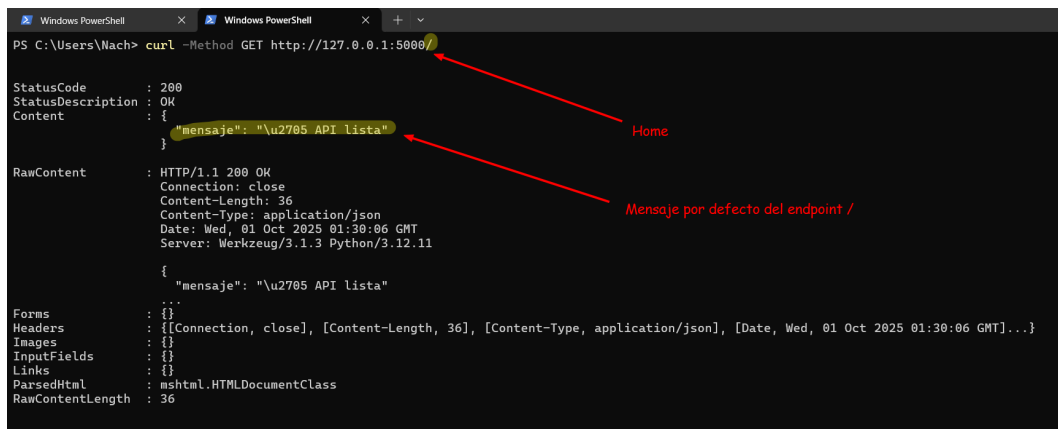
Evaluación Módulo 10

2.1. Entrenamiento y serialización del modelo

Los archivos asociados a la serialización del modelo están en la ubicación `./Eva_Mod10/serializar_modelo`. El archivo `serializar_modelo.ipynb` carga los datos, crea un modelo de clasificación de tipo random forest y serializa el modelo entrenado. Notar que el modelo serializado tiene un rendimiento decente, pero este no es el objetivo de la evaluación.

2.2. Desarrollo de API con Flask

El archivo `./Eva_Mod10/app.py` se encarga de crear la API asociada al modelo de clasificación del ítem anterior. Esta API tiene dos endpoints, uno raíz que se encarga de hacer notar que está funcionando la API,



```
PS C:\Users\Nach> curl -Method GET http://127.0.0.1:5000/

StatusCode      : 200
StatusDescription : OK
Content         : {
  "mensaje": "\u2705 API lista"
}

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 36
                  Content-Type: application/json
                  Date: Wed, 01 Oct 2025 01:30:06 GMT
                  Server: Werkzeug/3.1.3 Python/3.12.11

                  {
                    "mensaje": "\u2705 API lista"
                  }
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 36], [Content-Type, application/json], [Date, Wed, 01 Oct 2025 01:30:06 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 36
```

y otro, para hacer las predicciones necesarias por el usuario.

```
PS C:\Users\Nacho> curl -XPOST http://127.0.0.1:5000/predict -H 'Content-Type: application/json' -d '{"features": [1.93490586, 0.99373946, 1.93309591, 2.81678415, 0.30883801, 1.06619158, 2.29003532, 2.1171921, 1.43621312, -0.54118592, 2.16129581, -0.71857712, 1.735548, 2.14655106, -0.58601673, 0.76820176, 2.09838476, 1.11014072, 0.41986408, 0.45658573, 1.92810613, 0.21540599, 1.72873394, 1.98541627, -0.49396886, 0.40835425, 2.04811805, 1.46813593, 0.36439584, -0.30233857]}'

StatusCode      : 200
StatusDescription: OK
Content         : {"prediction": 1}

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 22
                  Content-Type: application/json
                  Date: Wed, 01 Oct 2025 01:31:58 GMT
                  Server: Werkzeug/3.1.3 Python/3.12.11

                  {
                    "prediction": 1
                  }

Forms           : {}
Headers         : [{"Connection", "close"}, [{"Content-Length", 22}, [{"Content-Type", "application/json"}, [{"Date", "Wed, 01 Oct 2025 01:31:58 GMT"}...]]
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength: 22
```

2.3. Dockerización del sistema

Para realizar la dockerización del sistema usamos el archivo `./Eva_Mod10/Dockerfile`. Este archivo tiene la función de crear el directorio, abrir puertos, copiar archivo e instalar las librerías necesarias en el entorno virtual. En la siguiente imagen podemos el contenedor asociado a la API del ejercicio anterior corriendo.

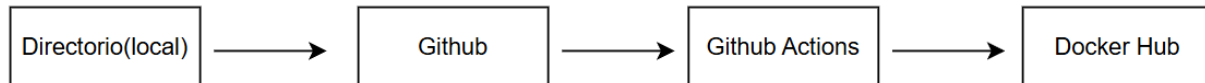
```
Digest: sha256:0e36ab4dbca6599aa1d0a68a1d4d8e7ca04fbca2b44cbc8bcb3a7478b1ee2c8
Status: Downloaded newer image for ignacio763/eva_mod10:2025.09.30-56faece
docker.io/ignacio763/eva_mod10:2025.09.30-56faece
PS C:\Users\Nacho> docker run -p 5000:6874 4b78b2cf3f5d
* Serving Flask app 'app'
* Debug mode: on
PS C:\Users\Nacho> docker run -p 6000:6874 4b78b2cf3f5d
* Serving Flask app 'app'
* Debug mode: on
PS C:\Users\Nacho> docker pull ignacio763/eva_mod10:2025.09.30-578afcd
2025.09.30-578afcd: Pulling from ignacio763/eva_mod10
15b1d8a5ff03: Already exists
22718812f617: Already exists
401a98f7495b: Already exists
ad446e7df19a: Already exists
3ea8285073a3: Already exists
9263f10fdf97: Already exists
280a8213ce16: Already exists
0fdc3f926e88: Pull complete
7f04110acc42: Pull complete
2a476c496acb: Pull complete
14e5144238f3: Pull complete
a885b953dc10: Pull complete
d6e3707fd6dc: Pull complete
Digest: sha256:346d216d7b09304a2fdbaedee8596a0f0d2fd7ce2d232862102888ee84bde42
Status: Downloaded newer image for ignacio763/eva_mod10:2025.09.30-578afcd
docker.io/ignacio763/eva_mod10:2025.09.30-578afcd
PS C:\Users\Nacho> docker run -p 5000:6875 sha256:e986785d8bebe7178abf7032d7cf71609015062b7b34c0cb45aa1f07a43cbc50
* Serving Flask app 'app'
* Debug mode: on
```

2.4. Automatización CI/CD (opcional)

Nuestra evaluación consiste en un mini-proyecto que tiene por objetivo servir una API. Esta API es cargada desde una imagen de docker la cuál obviamente estará dispuesta a presentar cambios a lo largo del tiempo. Para esto, se registraran las versiones en el siguiente repositorio

https://github.com/IgnOrtega/Eva_Mod10. También, en el caso que se apliquen cambios a este repositorio se activará un trigger que creará una nueva imagen de docker para actualizar el entorno de trabajo y la API. Esta nueva versión de la imagen de docker se subirá en el siguiente repositorio de Docker Hub https://hub.docker.com/repository/docker/ignacio763/eva_mod10/general.

En resumen, se tendrá el siguiente flujo de trabajo:



A continuación se mostrará un ejemplo de cambio en el repositorio con el propósito de mostrar como funciona el flujo de trabajo. En una primera instancia, nuestra API trabaja con el puerto 5248 dentro de la máquina virtual tal y como lo muestra las siguientes imagenes:

```
PS C:\Users\Nach> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ignacio763/eva_mod10 2025.09.30-a4be64a 4b78b2cf3f5d        12 minutes ago     1.46GB
ignacio763/eva_mod10 latest              4b78b2cf3f5d        12 minutes ago     1.46GB
breast-cancer-api    latest             be7f181d5ca1        21 hours ago       1.46GB
<none>               <none>             165c036e49b9        21 hours ago       1.46GB
iris-api             latest             bf3453f68de2        2 days ago         1.46GB
<none>               <none>             b15ff4a4342f        2 days ago         1.46GB
<none>               <none>             67856a481afb        5 days ago         1.46GB
<none>               <none>             83c3f13fd134        6 days ago         1.46GB
<none>               <none>             778079742a22        6 days ago         1.46GB
<none>               <none>             f45624d985fd        6 days ago         1.46GB
<none>               <none>             772ba58fb22b        6 days ago         1.46GB
gcr.io/k8s-minikube/kicbase v0.0.48            c6b5532e987b        3 weeks ago        1.31GB
tensorflow/tensorflow latest-gpu-jupyter db5d82e97e1d        6 weeks ago        7.92GB
nvidia/cuda          13.0.0-cudnn-devel-rockylinux9 638501ba0eae        7 weeks ago        7.78GB
nvidia/cuda          12.8.1-cudnn-runtime-rockylinux8 ca7b40e49c44        6 months ago       4.8GB
PS C:\Users\Nach> docker run -p 5000:5248 4b78b2cf3f5d
* Serving Flask app 'app'
* Debug mode: on
```

Version de la imagen docker

Contenedor Docker

Puerto máquina local

```
PS C:\Users\Nach> curl -Method POST http://127.0.0.1:5000/predict `
>> -Headers @{ "Content-Type" = "application/json" } `
>> -Body '{"features": [1.23498586, 0.99272946, 1.93109591, 2.01678415, 0.30883801, 1.06619158, 2.29003532, 2.1171921, 1.43621312, -0.54118592, 2.16
129581, -0.71857712, 1.735548, 2.14655106, -0.58601671, 0.76020176, 2.09838476, 1.11014072, 0.41986408, 0.45658573, 1.92810613, 0.21540599, 1.728733
94, 1.98541627, -0.49396886, 0.40035425, 2.04811805, 1.46613593, 0.36439584, -0.30233857]}'

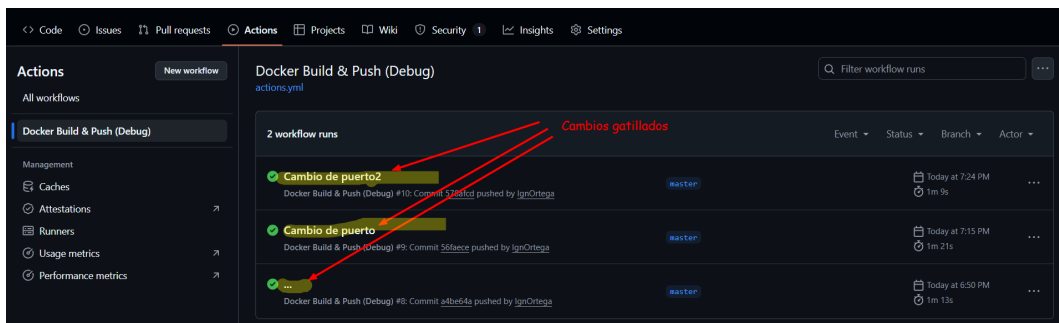
StatusCode      : 200
StatusDescription : OK
Content         : {"prediction": 1}
RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 22
                  Content-Type: application/json
                  Date: Tue, 30 Sep 2025 22:06:42 GMT
                  Server: Werkzeug/3.1.3 Python/3.12.11
                  {"prediction": 1}
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 22], [Content-Type, application/json], [Date, Tue, 30 Sep 2025 22:06:42 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 22

PS C:\Users\Nach>
```

La idea será cambiar este puerto y mostrar como fluye este flujo de trabajo semi-automatizado. Ahora, haremos editaremos el archivo `./Eva_Mod10/Dockerfile`, haciendo EXPOSE 6874 y en el archivo `./Eva_Mod10/app.py` haciendo:

```
app.run(debug=True, host='0.0.0.0', port=6874)
```

Luego, hacemos el push hacia el repositorio remoto, haciendo que se active el trigger de GitHub actions:



Este trigger sube a DockerHub una imagen al repositorio de DockerHub <https://hub.docker.com/repositorio> la siguiente imagen muestra las distintas imagenes que ha subido este trigger.

Tags DOCKER SCOUT INACTIVE [Activate](#)

This repository contains 4 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	3 minutes
2025.09.30-578afcd		Image	less than 1 day	3 minutes
2025.09.30-56faece		Image	less than 1 day	12 minutes
2025.09.30-a4be64a		Image	less than 1 day	37 minutes
2025.09.30-b58488a		Image	less than 1 day	about 2 hours

[See all](#)

Imágenes docker de la evaluación

Finalmente, debemos bajar la última imagen de DockerHub del repositorio, entonces cargaremos los cambios hechos al inicio que se pueden ver en la siguiente imagen.

```
Digest: sha256:0e36ab4dbca6599aa1d0a68a1d4d8e7ca04fbca2b44cbc8bcba3a7478b1ee2c8
Status: Downloaded newer image for ignacio763/eva_mod10:2025.09.30-56faece
docker.io/ignacio763/eva_mod10:2025.09.30-56faece
PS C:\Users\Nacho> docker run -p 5000:6874 4b78b2cf3f5d
* Serving Flask app 'app'
* Debug mode: on
PS C:\Users\Nacho> docker run -p 6000:6874 4b78b2cf3f5d
* Serving Flask app 'app'
* Debug mode: on
PS C:\Users\Nacho> docker pull ignacio763/eva_mod10:2025.09.30-578afcd
2025.09.30-578afcd: Pulling from ignacio763/eva_mod10
15b1d8a5ff03: Already exists
22718812f617: Already exists
401a98f7495b: Already exists
ad446e7df19a: Already exists
3ea8285073a3: Already exists
9263f10fdf97: Already exists
280a8213ce16: Already exists
0fdc3f926e88: Pull complete
7f04110acc42: Pull complete
2a476c496acb: Pull complete
14e5144238f3: Pull complete
a885b953dc10: Pull complete
d6e3707fd6dc: Pull complete
Digest: sha256:346d216d7b09304a2fdbaedeee8596a0f0d2fd7ce2d232862102888ee84bde42
Status: Downloaded newer image for ignacio763/eva_mod10:2025.09.30-578afcd
docker.io/ignacio763/eva_mod10:2025.09.30-578afcd
PS C:\Users\Nacho> docker run -p 5000:6875 sha256:e986785d8bebe7178abf7032d7cf71609015062b7b34c0cb45aa1f07a43cbc50
* Serving Flask app 'app'
* Debug mode: on
```

Notando, que el puerto de la maquina virtual cambio. La siguiente imagen muestra que funciona la API con el cambio.

```
Windows PowerShell X Windows PowerShell X + -
PS C:\Users\Nacho> curl -Method POST http://127.0.0.1:5000/predict '
>> -Headers @{ "Content-Type" = "application/json" } '
>> -Body '{"features": [1.93498586, 0.99373946, 1.93309591, 2.01678415, 0.30883801, 1.06619158, 2.29003532, 2.1171921, 1.43621312, -0.54118592, 2.16
129581, -0.71857712, 1.735548, 2.14655106, -0.58601673, 0.76020176, 2.09838476, 1.11014072, 0.41986408, 0.45658573, 1.92810613, 0.21540599, 1.728733
94, 1.98541627, -0.49396886, 0.40035425, 2.04811805, 1.46813593, 0.76439584, -0.30233857]}'

StatusCode      : 200
StatusDescription : OK
Content         : {"prediction": 1}
RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 22
                  Content-Type: application/json
                  Date: Tue, 30 Sep 2025 23:52:44 GMT
                  Server: Werkzeug/3.1.3 Python/3.12.11

                  {
                    "prediction": 1
                  }
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 22], [Content-Type, application/json], [Date, Tue, 30 Sep 2025 23:52:44 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 22
```

Capítulo 3

Reflexiones

En los módulos anteriores vimos como entrenar un modelo, como medir su rendimiento y como preentrenarlos. En esta evaluación trabajamos realizando un trabajo más de producción y automatización para que cierto usuarios pudieran consumir nuestros modelos. Los modelos se actualizan de forma semi-automática, lo cuál sirve para una cantidad de modelos pequeña que debemos mantener. Para una cantidad de modelos más alta, quizás sea mejor usar otras herramientas como kubernetes, que es capaz de reiniciar un contener de docker en el caso de que exista una versión de la imagen asociada al contenedor.