MEMORIA PROYECTO FINAL

Despliegue de un proyecto, instalación de dependencias y ejecución.

Introducción

En este proyecto se va a trabajar con una máquina virtual de Alpine Linux y la instalación de repositorios a descargar y hacer funcionar. Emplearemos herramientas vistas a lo largo como entornos virtuales, la programación en Basch y la manipulación de datos.

El objetivo principal es automatizar mediante scripts de Basch la realización de peticiones HTTP desde Python, con el trabajo previo de la instalación de dependencias del sistema, clonación de repositorios, uso de un entorno virtual, ejecución de servicios de Flask, descarga de datos y estadísticas y generación de mapas.

Desarrollo y Documentación

1. Instalación de paquetes Alpine

En este ejercicio desarrollamos el script *apartado1_Nuria_Belen.sh*, ejecutado desde el usuario root de la máquina virtual. Este script instalará los paquetes necesarios mediante el comando apk add. Estos paquetes son:

- python3
- git
- sudo pantallazos (etc/apk/…)
- bash
- gcc, libc-dev, g++, musl-dev, linux-headers
- wget, curl
- htop

Para hacer que la descarga de sudo funcione, debimos modificar el fichero */etc/apk/repositories* y descomentar la url que contiene community. Podemos observar los cambios en la Figura 1.

***Figura 1**: modificación manual de sudo*

Además, otorgaremos permisos de super usuario al usuario alumnoimat. Para ello, lo añadimos al grupo Wheel, que es el que permite usar sudo en Alpine. Después, comprobamos mediante el comando *sudo -lU alumnoimat* que el usuario ya podía ejecutar cualquier comando como root. Podemos ver los comandos usados en la Figura 2.

```
/home/alumnoimat # sudo -lU alumnoimat
User alumnoimat is not allowed to run sudo on alps.
/home/alumnoimat # echo '%wheel ALL=(ALL) ALL' > /etc/sudoers.d/wheel
/home/alumnoimat # adduser alumnoimat wheel
/home/alumnoimat # sudo -lU alumnoimat
Matching Defaults entries for alumnoimat on alps:
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

Runas and Command-specific defaults for alumnoimat:
    Defaults!/usr/sbin/visudo env_keep+="SUDO_EDITOR EDITOR VISUAL"

User alumnoimat may run the following commands on alps:
    (ALL) ALL
/home/alumnoimat # _
```

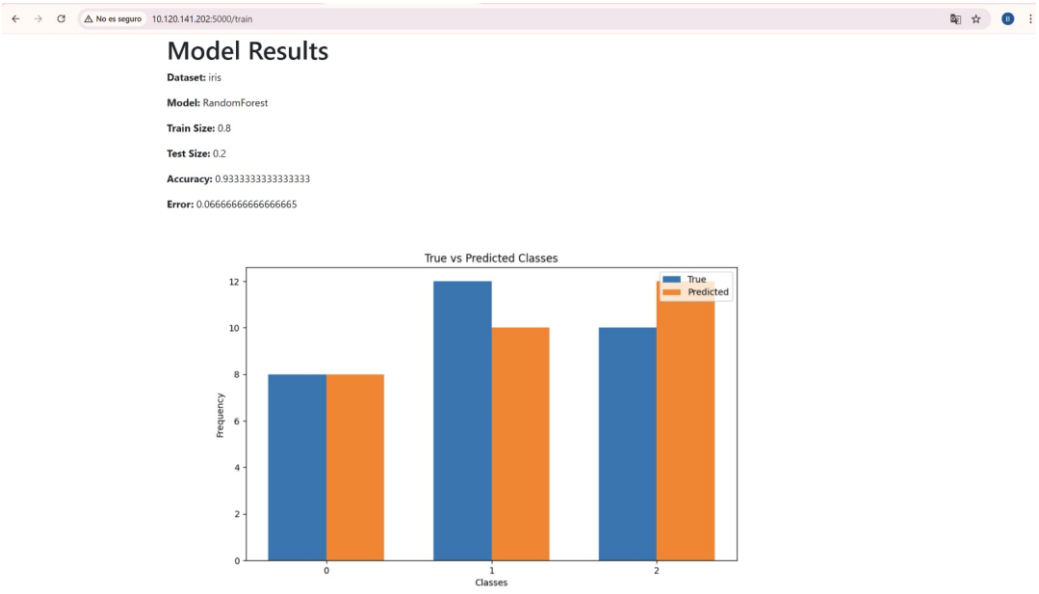
***Figura 2**: permisos de sudo del usuario alumnoimat*

2. Funcionamiento de un repositorio y un despliegue.

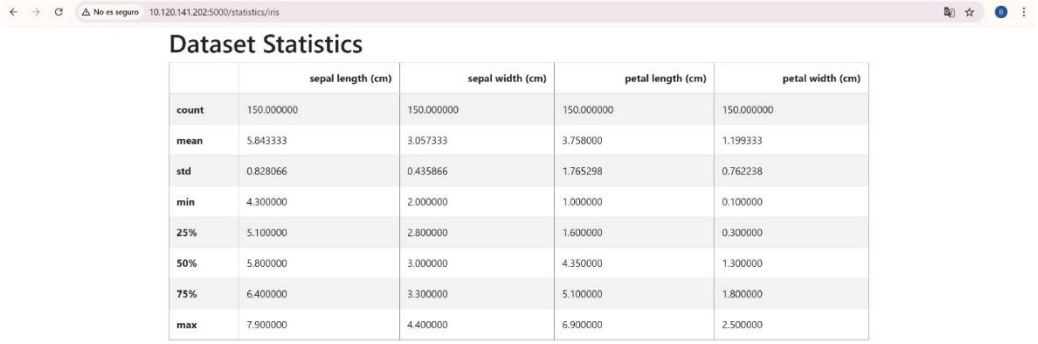
Desde el usuario alumnoimat, de ahora en adelante, desarrollamos un código Bash *apartado_despliegue_Bash_Nuria_Belen.sh*, que desplegará un proyecto de un repositorio git y activará el entorno virtual creado anteriormente, instalando las correspondientes librerías y poniéndolo en ejecución.

Así, se abrirá una aplicación Flask que nos permitirá realizar una seria de tareas. Para abrirla, deberemos cambiar, en el script de Bash adjuntado en la tarea, la dirección IP de la máquina virtual escrita a la dirección IP de la máquina desde la que se vaya a ejecutar el proyecto. Los servicios disponibles en la url son:

- Train and Evaluate

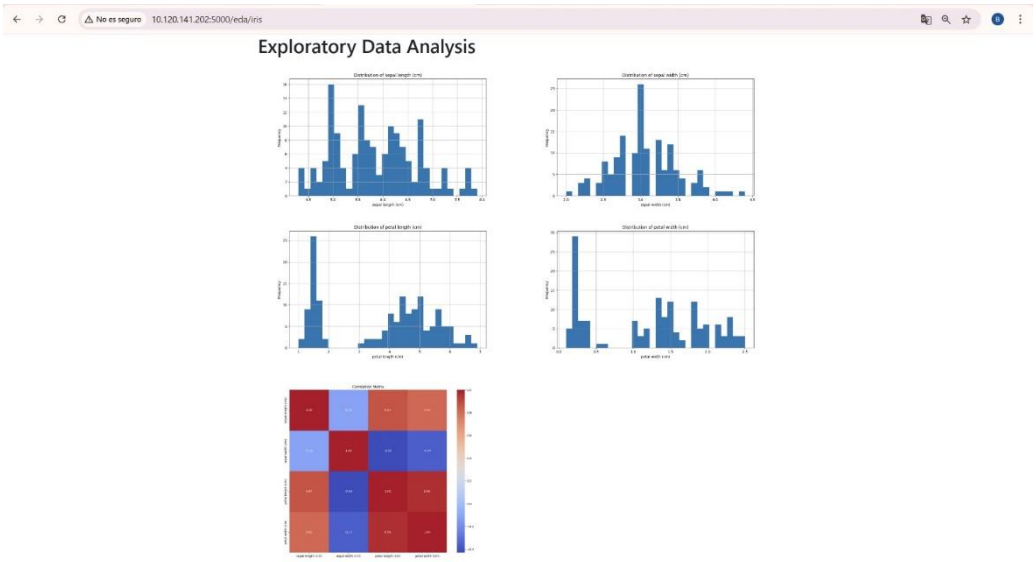


- Dataset Statistics

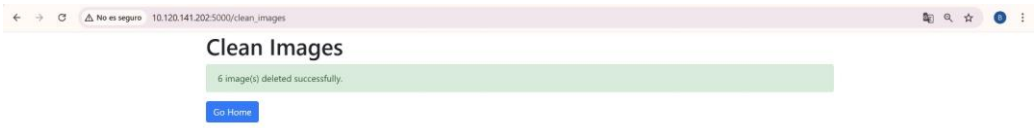


	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

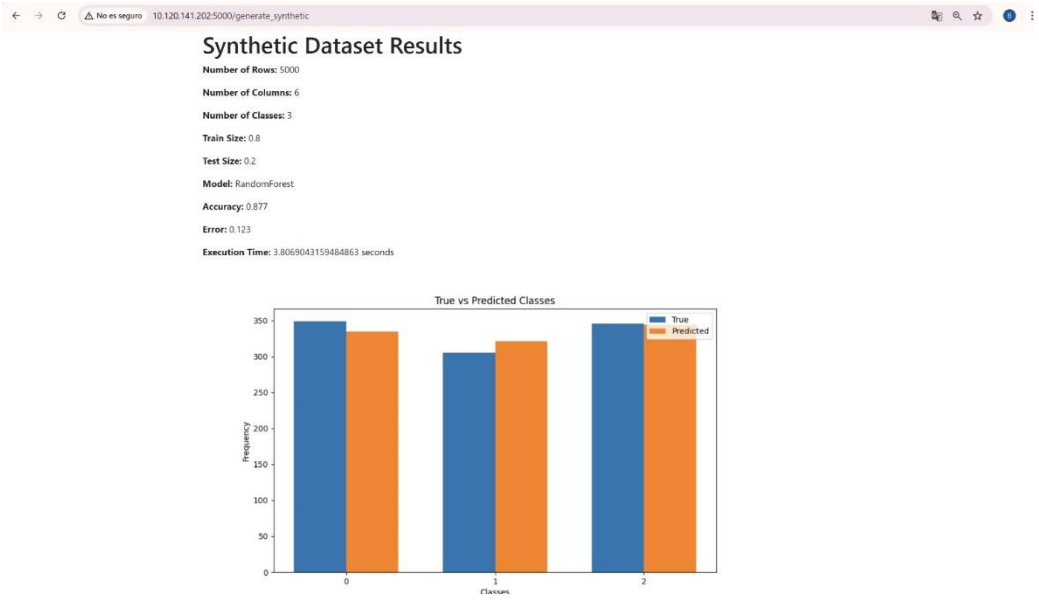
- Exploratory Data Analysis



- Clean Images



- Generate Synthetic Dataset



- Compare Execution: nos permite ejecutar el benchmarking de la multiplicación de matrices. El código Python de esta función, modificado por nosotras, está adjuntado en la tarea.

- Show HTML files

Tras analizar el código proporcionado del proyecto descargado desde github, se proponen algunas posibles **mejoras** de este. Entre ellas, destacamos la falta de comentarios en algunos módulos, ya que a veces se nos ha complicado el entender lo que hacían especialmente los modelos de ML. Así, con una documentación más detallada del código, usuarios con menor conocimiento podrían trabajar con ellos con mayor facilidad.

Además, habría sido conveniente que se nos incluyera un fichero de texto con todos los repositorios necesarios a descargar para la ejecución del apartado 2. Nos hubiera facilitado el no tener que ir uno por uno, ejecutando el código y descubriendo en cada ejecución una nueva librería que descargar.

Por último, mejoraría la organización del directorio descargado del proyecto. En varias ocasiones, siguiendo el enunciado, se nos mencionaban ficheros determinados, pero no adjuntaban la ruta del fichero, por lo que nos veíamos obligadas a ir abriendo uno a uno los subdirectorios hasta localizarlos.

Por último, tras ejecutar el código desarrollado de la función *compare_execution()*, ejecutamos htop y describimos lo observado en la Figura 3.

Durante la prueba de comparación de tiempos observamos como, cuando ejecutamos el proceso de manera secuencial, solo ocupa un núcleo de la CPU, mientras que cuando ejecutamos de manera paralela, ambos núcleos se activan simultáneamente por la creación de procesos extra. Vemos varias instancias de python3, correspondientes a la ejecución de la aplicación Flask y a los procesos paralelos que multiplican las matrices. El uso de la memoria no aumenta mientras se realizan los cálculos de matrices , por lo que afirmamos que el algoritmo usado es eficiente y funciona correctamente, aprovechando más de un núcleo.

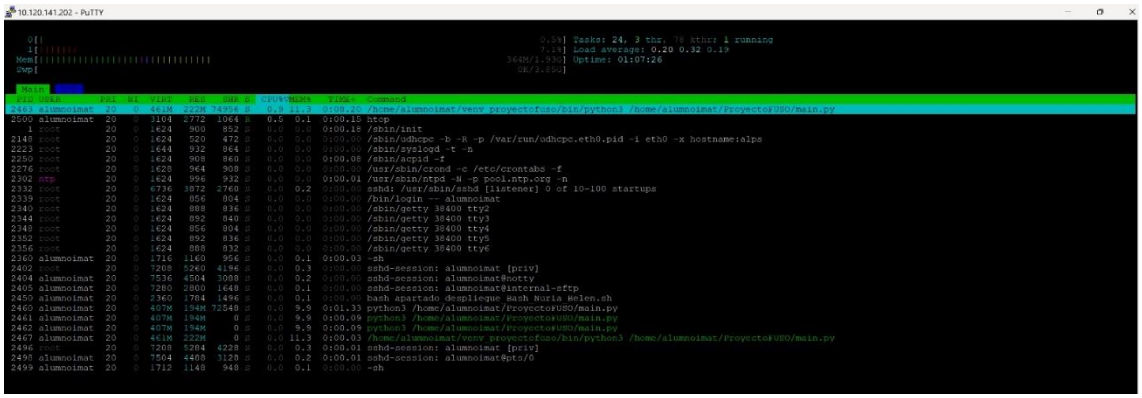


Figura 3: *Monitorización del uso de CPU mediante htop durante la ejecución de la función compare_execution()*

3. Descarga de datos (show HTML files)

En el script de Bash *apartado3_Nuria_Belen.sh*, automatizamos la descarga del conjunto de datos especificado, y los cuatro ficheros pertenecientes a las ciudades de El Paso, Glasgow, Manchester y Washington DC, que contienen la información sobre las visitas

de cada usuario, indicando el identificador de usuario, el momento de la visita y las coordenadas e identificador del lugar de visita.

Además, el código calculará unas estadísticas por ciudad en Bash y llamará a un código Python que las calculará haciendo un uso exclusivo de las librerías sys y os.

Así mismo, el script llamará a los ficheros generate_maps.py y generate_individual_maps.py, obtenidos del repositorio del proyecto, que calcularán los mapas solicitados. Estos mapas serán visualizados desde la aplicación Flask: podemos observar algunos ejemplos en las Figuras 4 y 5.

Por último, desarrollamos un archivo Python que calculará el top 5 usuarios que más visitas realizaron a cada ciudad.

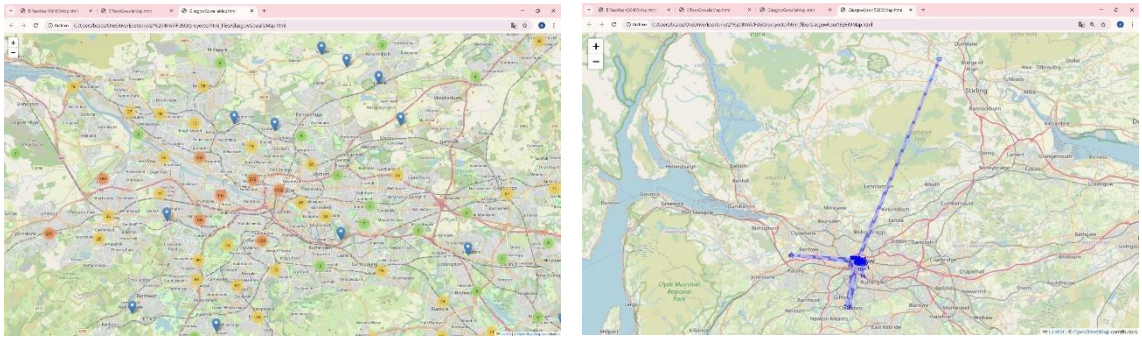


***Figura 4**: mapa y mapa individual del usuario 152639 de la ciudad de Glasgow*

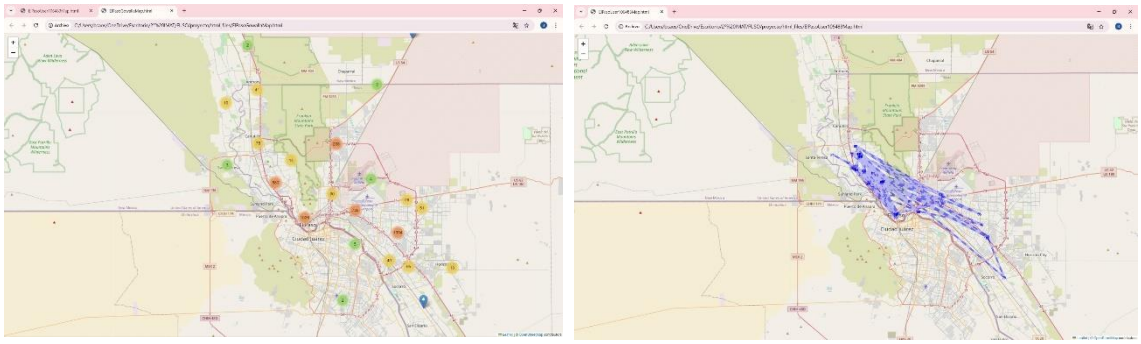


***Figura 5**: mapa y mapa individual del usuario 106483 de la ciudad de El Paso*

4. Peticiones de Python

En este último apartado, desarrollamos un archivo Python apartado4_Nuria_Belen.py que nos permite automatizar peticiones GET y POST al servicio /train de la aplicación Flask con distintos valores de train_size y descargar las imágenes (archivos html) generadas.

Para hacer funcionar este script, es indispensable tener corriendo la aplicación Flask en otra terminal. En nuestro caso, ejecutamos el código Python en la terminal de Windows, mientras que dejábamos correr la aplicación en la terminal Linux de Alpine.

Un ejemplo de las imágenes generadas lo podemos observar en la Figura 6.

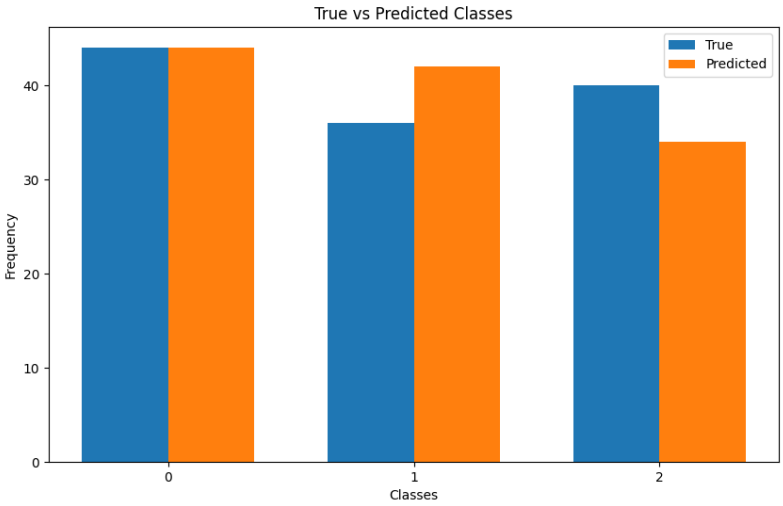


Figura 4: *petición de train para train_size = 0.2 y test_size = 0.8*

Conclusiones

En este proyecto hemos desarrollado de forma automática y estructurada el completo de la instalación, automatización y despliegue dentro de un entorno desarrollado en la máquina virtual de Linux Alpine, haciendo uso principalmente de scripts Bash, entornos virtuales de Python y una aplicación Flask que, ahora, funciona como se ha solicitado.

Hemos instalado y configurado paquetes del sistema, desplegado un repositorio de git, generado estadísticas y mapas a partir de datos, automatizado peticiones de generación de imágenes y analizado el rendimiento del procesamiento en paralelo y secuencial, haciendo uso de herramientas del sistema como htop.