Despliegue de modelo para detección de deficiencias nutricionales en hojas de café desde Azure

Integrantes:

- ➢ Victor Alberto Lizcano Portilla alberto.lizcano@udea.edu.co
- ➢ David Alberto Rodriguez Muñoz dalberto.rodriguez@udea.edu.co

Trabajo final de Cloud - Cohorte 2-2021

1. DESCRIPCIÓN DEL DATASET

El conjunto de datos utilizado para este dataset es de un antigüo proyecto de tesis de uno de los compañeros del trabajo. Corresponde a datos extraídos de imágenes de hojas de café. El problema que dió origen al dataset es llamado ***"Detección de deficiencias nutricionales en plantas de café utilizando procesamiento digital de imágenes"***. El dataset se encuentra conformado por los vectores patrón que describen las características de cada imagen (características de forma, color y textura).

2. WORKFLOW EN AZURE

Siguiendo la ruta del workflow para Azure machine learning, se obtienen los siguientes pasos para la puesta en producción del modelo.

A. CREACIÓN DEL WORKSPACE

Para la creación del workspace, inicialmente fue necesario crear el grupo de recursos (figura 1), para este caso el *"rg-ml-udea-david-alberto"*, la nomenclatura de cada elemento se realizó siguiendo los lineamientos vistos en clase.

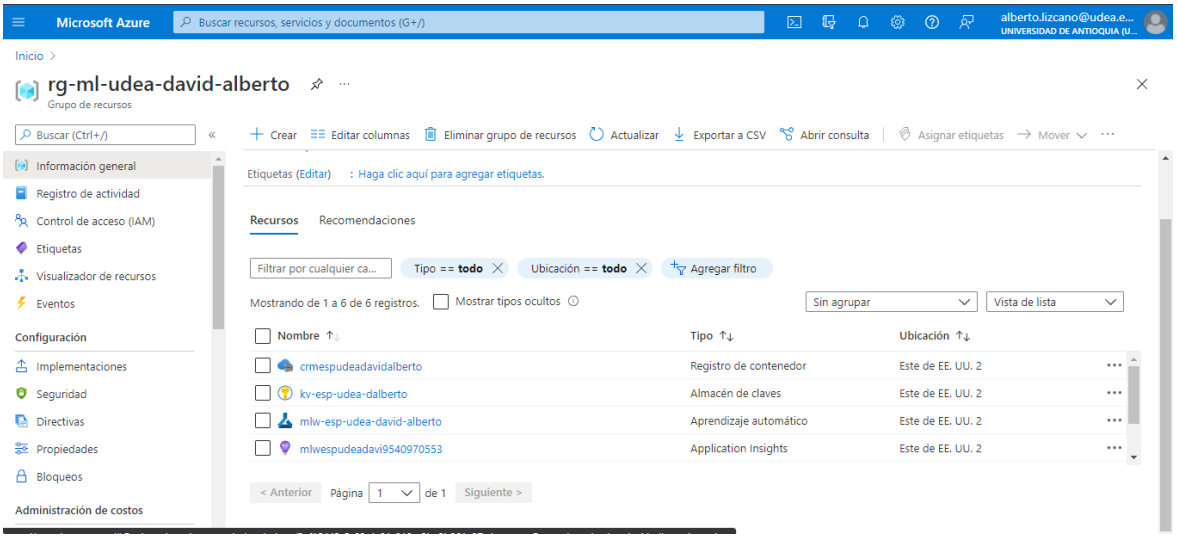


Figura 1. Grupo de recursos rg-ml-udea-david-alberto

Seguido a la creación del grupo de recursos se procede a configurar los diferentes ambientes de trabajo, instalando todos los paquetes necesarios para su correcto funcionamiento, esto se realiza utilizando los archivos yamel *"nutrition-aml-env.yml"* y *"nutrition-local-env.yml".* Posteriormente nos conectamos al espacio de trabajo en azure, utilizando el siguiente script "*01-connect_workspace.py"* (figura 2), el cual genera un archivo de configuración llamado config.json

```python
from azureml.core.authentication import InteractiveLoginAuthentication
from azureml.core import Workspace

interactive_auth = InteractiveLoginAuthentication(force=True, tenant_id='99e1e721-7184-498e-8aff-b2ad4e53c1c2')

ws = Workspace.get(
    name='mlw-esp-udea-david-alberto',
    subscription_id='2ef15443-5c82-4c81-818a-2be2b061a27e',
    resource_group='rg-ml-udea-david-alberto',
    location='eastus2',
    auth=interactive_auth
)
ws.write_config(path='.azureml')
```

Figura 2. Script 01-connect_workspace.py

Finalmente se realizó un test para confirmar la correcta conexión con el espacio de trabajo en azure, el resultado se muestra en la figura 3.

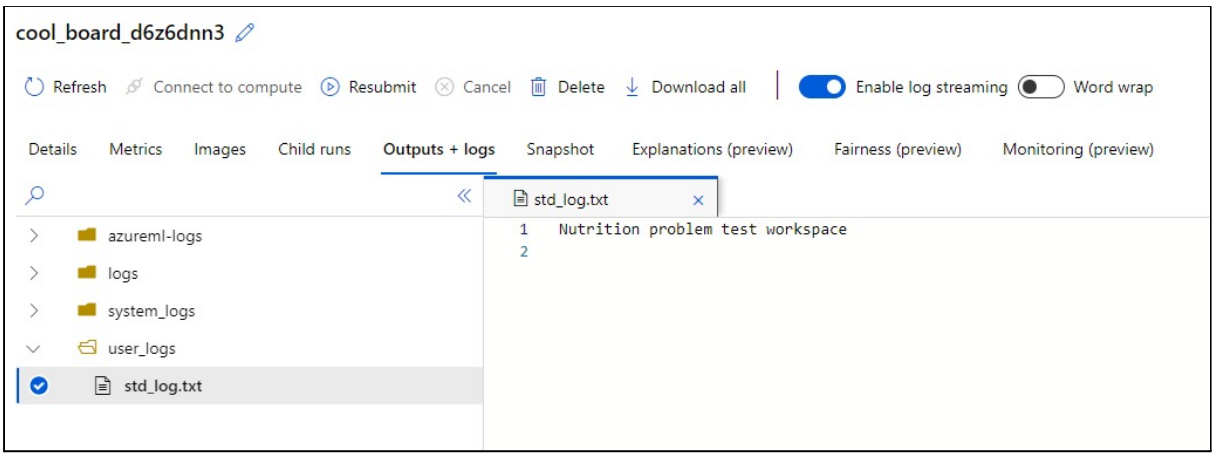


Figura 3. Test workspace

B. ENTRENAMIENTO DEL MODELO

Ya con el workspace creado, se prosiguió a crear el dataset en el workspace, por facilidad se utilizaron los scripts del repositorio original y se acoplaron las rutas de nuestro trabajo (Figura 4). Después de ejecutar el script podemos verificar que el dataset se encuentra ya en nuestro workspace de Azure (Figura 5).

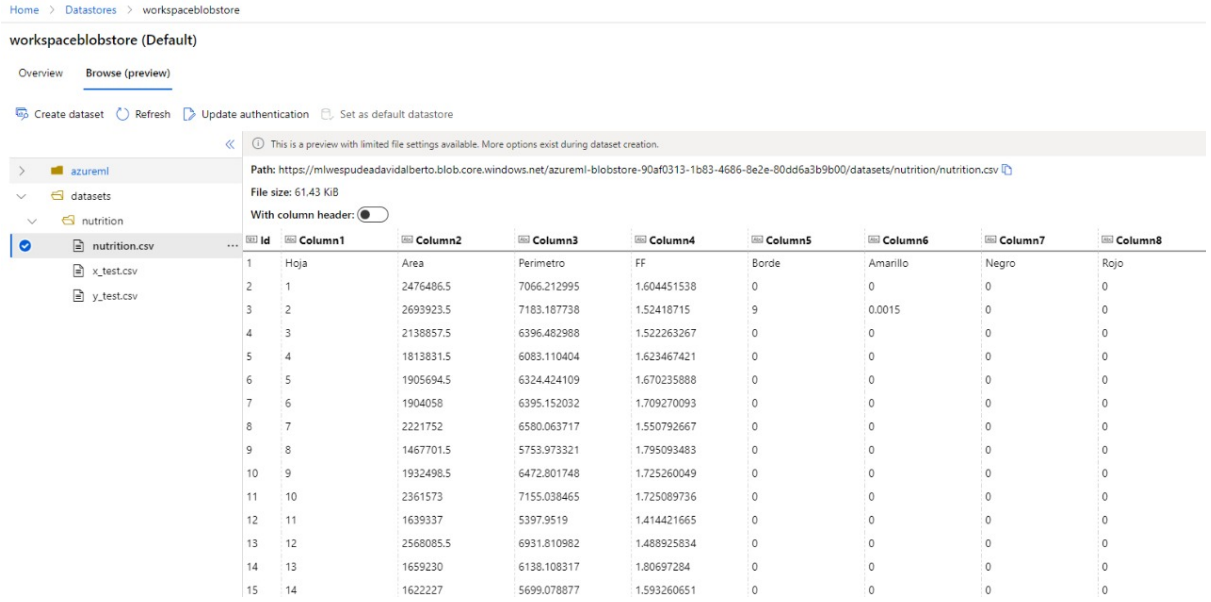
```python
# 04-upload-data.py
from azureml.core import Workspace

ws = Workspace.from_config(path='./.azureml',_file_name='config.json')

datastore = ws.get_default_datastore()

datastore.upload(src_dir='./data',
                 target_path='datasets/nutrition',
                 overwrite=True)
```

Figura 4. Script para subir datos a Azure ML studio.



Id	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8
1	Hoja	Area	Perimetro	FF	Borde	Amarillo	Negro	Rojo
2	1	2476486.5	7066.212995	1.604451538	0	0	0	0
3	2	2693923.5	7183.187738	1.52418715	9	0.0015	0	0
4	3	2138857.5	6396.482988	1.522263267	0	0	0	0
5	4	1813831.5	6083.110404	1.623467421	0	0	0	0
6	5	1905694.5	6324.424109	1.670235888	0	0	0	0
7	6	1904058	6395.152032	1.709270093	0	0	0	0
8	7	2221752	6580.063717	1.550792667	0	0	0	0
9	8	1467701.5	5753.973321	1.795093483	0	0	0	0
10	9	1932498.5	6472.801748	1.725260049	0	0	0	0
11	10	2361573	7155.038465	1.725089736	0	0	0	0
12	11	1639337	5397.9519	1.414421665	0	0	0	0
13	12	2568085.5	6931.810982	1.488925834	0	0	0	0
14	13	1659230	6138.108317	1.80697284	0	0	0	0
15	14	1622227	5699.078877	1.593260651	0	0	0	0

Figura 5. Dataset creado en azure ML studio.

Una vez con los datos en el workspace se continúa con el entrenamiento del modelo de manera remota en la nube de azure. Utilizando el script *"05-train-remote-dataset.py"* para entrenamiento remoto (Figura 6), se crean 4 objetos esenciales en este paso; **dataset** objeto que se especifica en que path se encuentran los datos en nuestro workspace, **experiment** se crea el experimento en el workspace, **config** se crean las configuraciones del script de entrenamiento, y por último, **env** donde especificamos cual es el ambiente virtual que se utilizará para entrenar el modelo en nuestro workspace. Al finalizar las configuraciones este script se ejecuta y podemos ver los resultados del experimento en el Azure ML studio (Figura 7), el cual muestra un score de 0.826.

05-train-remote-dataset.py > ...

```python
# 05-run-pytorch-data.py
from azureml.core import Workspace
from azureml.core import Experiment
from azureml.core import Environment
from azureml.core import ScriptRunConfig
from azureml.core import Dataset

if __name__ == "__main__":
    ws = Workspace.from_config(path='./.azureml',_file_name='config.json')
    datastore = ws.get_default_datastore()
    dataset = Dataset.File.from_files(path=(datastore, 'datasets/nutrition'))

    experiment = Experiment(workspace=ws, name='train-nutrition-model-remote')

    config = ScriptRunConfig(
        source_directory='./src',
        script='train-remote.py',
        compute_target='homework-cluster',
        arguments=['--data_path', dataset.as_named_input('input').as_mount()]
    )
    # set up pytorch environment
    env = Environment.from_conda_specification(
        name='nutrition-aml-env',
        file_path='./.azureml/nutrition-aml-env.yml'
    )
    config.run_config.environment = env

    run = experiment.submit(config)

    aml_url = run.get_portal_url()
    print("Submitted to compute cluster. Click link below")
    print("")
    print(aml_url)

```

Figura 6. Script de entrenamiento remoto.

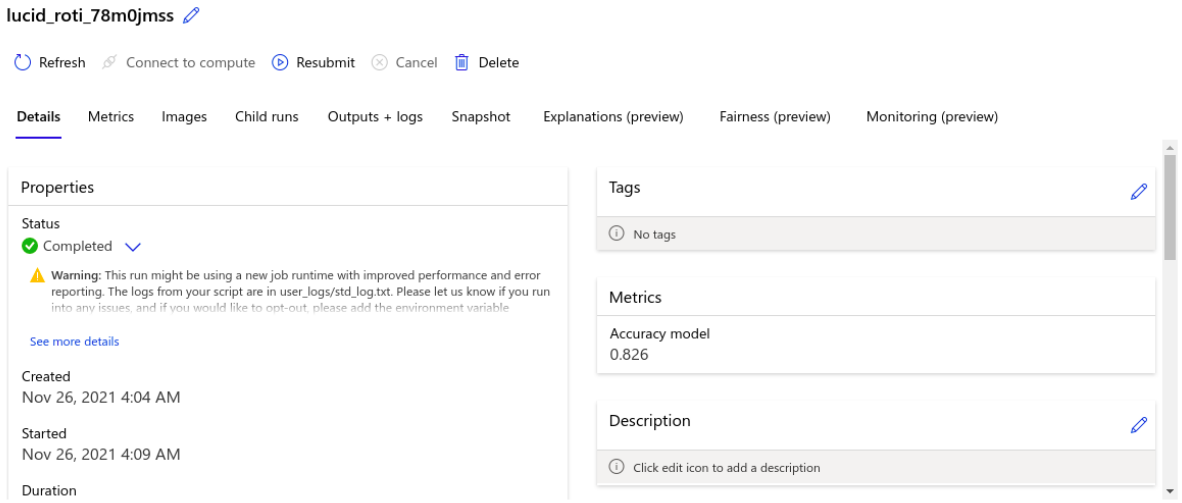


Figura 7. Resultados del entrenamiento remoto.

C. REGISTRO Y DESPLIEGUE DEL MODELO

Luego de haber realizado el entrenamiento del modelo de manera remota y su correspondiente test, se procede a registrar el modelo, para ello se hizo uso del siguiente script "*06-model-registration-azure.py*" (figura 8)

06-model-registration-aml.py > ...

```python
# 06-model-registration-azure.py
from azureml.core import Workspace
from azureml.core import Model

if __name__ == "__main__":
    ws = Workspace.from_config(path='./.azureml',_file_name='config.json')
    model = Model.register(model_name='nutrition_model',
                           tags={'area': 'tarea_udea','scoring' : 0.82576, 'data_set_size': 660},
                           model_path='outputs/nutrition_model.pkl',workspace = ws)
    print(model.name, model.id, model.version, sep='\t')
```

Figura 8. Script registro del modelo

El resultado del registro se muestra en la figura 9

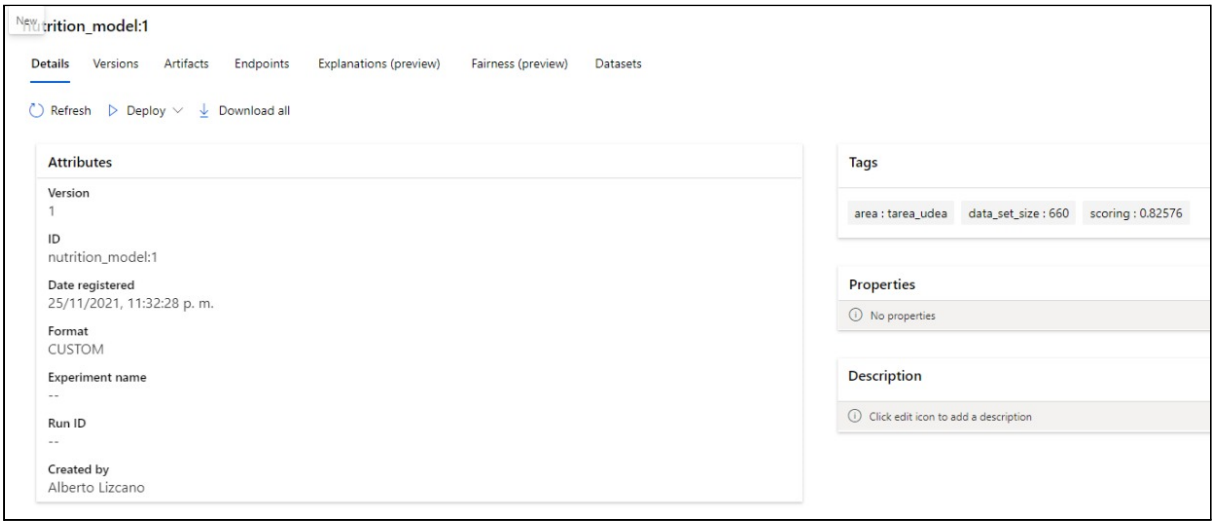


Figura 9. Registro del modelo

Para el despliegue del modelo, etapa en la cual se planea colocar el modelo en producción, se utilizaron los siguientes scripts para realizar el proceso de manera local *"07-deploy-model-local.py"* (figura 10) y de manera remota con ayuda de la nube de Azure, el script *"08-deploy-model-remote.py"* utilizado se muestra en la figura 11.

07-deploy-model-local.py > ...

```python
from azureml.core import environment
from azureml.core.webservice import webservice
from azureml.core.model import InferenceConfig
from azureml.core.environment import Environment
from azureml.core import Workspace, workspace
from azureml.core.model import Model
from azureml.core.webservice import LocalWebservice

ws = Workspace.from_config(path='./.azureml', _file_name='config.json')
model = Model(ws, name='nutrition_model', version=1)

env = Environment.from_conda_specification(
    name='nutrition-aml-env',
    file_path='./.azureml/nutrition-aml-env.yml'
)

inference_config = InferenceConfig(entry_script='./src/score.py', environment=env)

deployment_config = LocalWebservice.deploy_configuration(port=3000)

local_service = Model.deploy(workspace=ws,
                             name='nutrition-model-local',
                             models=[model],
                             inference_config=inference_config,
                             deployment_config=deployment_config
                             )

local_service.wait_for_deployment(show_output=True)
print(f'Scoring URI is: {local_service.scoring_uri}')
```

Figura 10. 07-deploy-model-local.py

08-deploy-model-remote.py > ...

```python
from azureml.core.model import InferenceConfig
from azureml.core.environment import Environment
from azureml.core import Workspace
from azureml.core.model import Model
from azureml.core.webservice import AciWebservice

ws = Workspace.from_config(path='./.azureml', _file_name='config.json')

model = Model(ws, name='nutrition_model', version=1)

env = Environment.from_conda_specification(
    name='nutrition-aml-env',
    file_path='./.azureml/nutrition-aml-env.yml'
)

inference_config = InferenceConfig(entry_script='./src/score.py', environment=env)

deployment_config = AciWebservice.deploy_configuration(cpu_cores=1, memory_gb=1)

aci_service = Model.deploy(workspace=ws, name='nutrition-model-service',
                           models=[model], inference_config=inference_config,
                           deployment_config=deployment_config)

aci_service.wait_for_deployment(show_output=True)
print(f'ACI state: {aci_service.state}')
print(aci_service.get_logs())
```

Figura 11. 08-deploy-model-remote.py

El resultado se muestra en la figura 12.

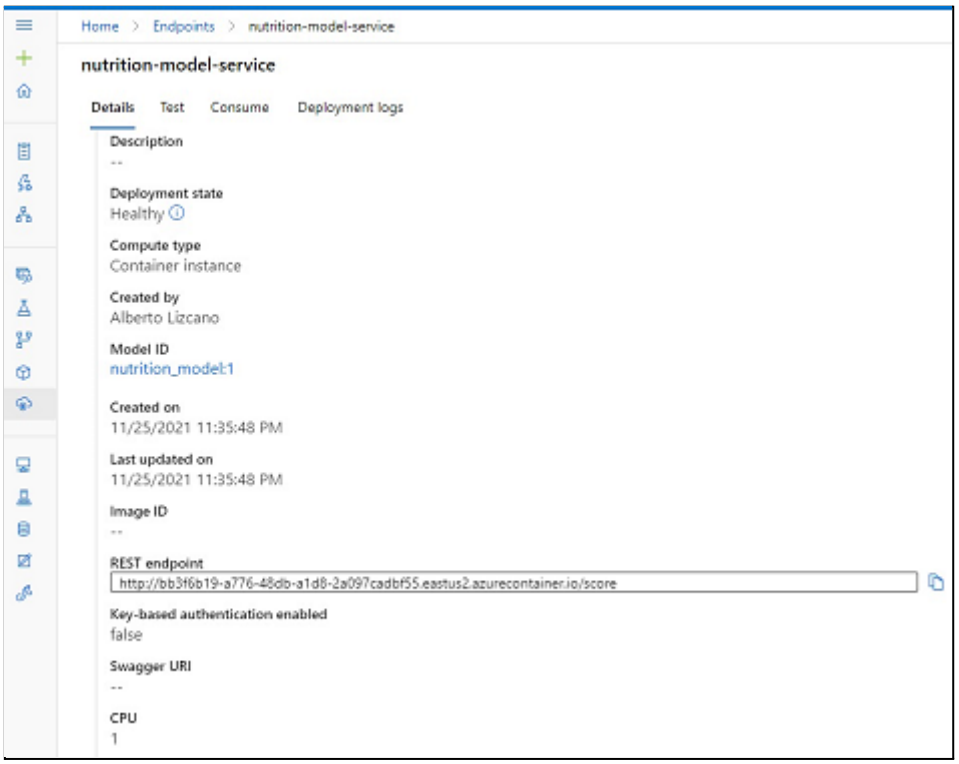


Figura 12. Despliegue del modelo en Azure