Universidad Católica Boliviana "San Pablo"
Unidad Académica Cochabamba
Departamento de Posgrado
Ing. Mauricio Alejandro Quezada Bustillo

DML-004 – GUÍA DE LABORATORIO 9

Pipeline de 8 componentes para Heart Disease (SDK v2)

🎯 Objetivos

• Diseñar un pipeline modular (8 pasos) con componentes YAML + pipeline en Python.

• Practicar limpieza, codificación, normalización, partición estratificada, entrenamiento, scoring y evaluación.

• Ejecutar y monitorear el pipeline en Azure ML Studio/SDK y dejar outputs versionados.

🧱 Arquitectura (8 componentes)

1) select_cols – Selección de columnas relevantes.
2) impute – Imputación de nulos.
3) encode – One-hot encoding de categóricas.
4) scale – Estandarización (Z-score).
5) split – Split 80/20 estratificado.
6) train_lr – Entrenamiento (Logistic Regression).
7) score – Predicción en test.
8) eval – Métricas (accuracy, precision, recall, F1, AUC).

📂 Estructura de proyecto sugerida

```
heart-pipeline/
├── components/
│   ├── select_cols/
│   ├── impute/
│   ├── encode/
│   ├── scale/
│   ├── split/
│   ├── train_lr/
│   ├── score/
│   └── eval/
├── envs/
├── data/
└── pipelines/
```

🧪 Entorno (envs/conda.yml)

```
name: heart-mlops
channels: [conda-forge, defaults]
dependencies:
  - python=3.10
  - pip
  - pip:
      - azure-ai-ml
      - azure-identity
      - pandas
      - numpy
      - scikit-learn
      - matplotlib
```

🧩 Plantilla de componente YAML

```
$schema: https://azuremlschemas.azureedge.net/latest/commandComponent.schema.json
name: select_cols
display_name: Select Columns (Heart)
type: command

inputs:
  raw_data: uri_file
outputs:
  selected: uri_file

code: .
environment:
  image: mcr.microsoft.com/azureml/openmpi4.1.0-ubuntu20.04:latest
  conda_file: ../../envs/conda.yml

command: >-
  python select_cols.py
  --data ${{inputs.raw_data}}
  --out ${{outputs.selected}}
```

Universidad Católica Boliviana "San Pablo"
Unidad Académica Cochabamba
Departamento de Posgrado
Ing. Mauricio Alejandro Quezada Bustillo

🧠 Scripts por componente (resúmenes)

- select_cols.py: selecciona columnas relevantes.
- impute.py: imputación numéricas=mediana, categóricas=moda.
- encode.py: one-hot encoding.
- scale.py: normalización Z-score.
- split.py: división 80/20 estratificada.
- train.py: entrena Logistic Regression.
- score.py: genera predicciones.
- eval.py: calcula métricas y guarda JSON.

📄 YAML de I/O por paso

select_cols → impute → encode → scale → split → train → score → eval

Pipeline en Python

```
@dsl.pipeline(compute=COMPUTE_NAME, description="Heart pipeline: 8 steps")
def heart_pipeline(raw: Input):
  s  = select_cols(raw_data=raw)
  imp = impute(selected=s.outputs.selected)
  enc = encode(imputed=imp.outputs.imputed)
  sc  = scale(encoded=enc.outputs.encoded)
  sp  = split(scaled=sc.outputs.scaled)

  tr  = train_lr(train=sp.outputs.train)
  sc2 = score(model_dir=tr.outputs.model_dir, test=sp.outputs.test)
  ev  = evalc(predictions=sc2.outputs.predictions)

  return {"metrics": ev.outputs.metrics, "model": tr.outputs.model_dir}
```

✅ Entregables

- Capturas del DAG con 8 componentes.
- Captura de split estratificado y environment.
- Artefactos: modelo, predictions.csv, metrics.json.
- Breve informe interpretando métricas.