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

DML-004 - GUÍA DE LABORATORIO 3

CLASIFICACIÓN BINARIA USANDO AZURE ML DESIGNER (TITANIC)

1. Objetivo

Construir un pipeline reproducible en Azure ML Designer (modo classic prebuilt components) para resolver un problema de clasificación binaria con el dataset Titanic. El flujo incluye limpieza de datos, codificación de variables categóricas, normalización de variables numéricas, partición en conjuntos de entrenamiento y prueba, entrenamiento de dos algoritmos (Two-Class Logistic Regression y Two-Class Boosted Decision Tree), evaluación comparativa y (opcional) ajuste de hiperparámetros.

2. Resultados de aprendizaje

- Entender el propósito de cada etapa del pipeline de ML y su impacto en el rendimiento del modelo.
- Configurar correctamente componentes classic prebuilt en Azure ML Designer.
- Interpretar métricas de clasificación (AUC, F1, Precision, Recall) y analizar trade-offs.
- Aplicar técnicas básicas de tuning de hiperparámetros y reproducibilidad experimental.

3. Requisitos previos

- Cuenta activa en Azure y un Workspace de Azure Machine Learning creado.
- Acceso a Azure ML Studio (interfaz web).
- Dataset Titanic en formato CSV (delimitado por coma).
- Conocimientos básicos de estadística y clasificación binaria.

4. Dataset

Dataset sugerido: Titanic (variable binaria objetivo Survived = 0/1).

Columnas típicas: Pclass, Sex, Age, SibSp, Parch, Fare, Embarked, (y otras como Name, Ticket, Cabin).

Características: faltantes en Age y, ocasionalmente, Embarked; variables categóricas en Sex y Embarked; Cabin puede eliminarse por alta cardinalidad y valores faltantes.

Alternativas (flujo idéntico): Adult Income (>50K), Bank Marketing (y=yes/no).

5. Preparación del entorno

1. Entra a Azure ML Studio → Designer → Create a new pipeline (classic prebuilt components).
2. Crea el Data Asset: Data → +Create → From local files → Tabular dataset → separador coma (,).
3. Confirma el esquema detectado (tipos de datos) y guarda con una versión (facilita reproducibilidad).

6. Esquema general del pipeline (DAG)

Diagrama lógico de alto nivel:

DATA ASSET → SELECT COLUMNS → CLEAN MISSING DATA → CONVERT TO INDICATOR VALUES (ONE-HOT) → NORMALIZE DATA → SPLIT DATA → [(LOGISTIC REGRESSION → TRAIN) || (BOOSTED DECISION TREE → TRAIN)] → SCORE MODEL → EVALUATE MODEL

Variante con tuning: Reemplaza el bloque de entrenamiento por Tune Model Hyperparameters → Best model → Score → Evaluate.

7. Detalle paso a paso con configuración y justificación

7.1 SELECT COLUMNS IN DATASET

Objetivo: reducir ruido y evitar fugas de información (data leakage).

- Incluir: Survived (label), Pclass, Sex, Age, SibSp, Parch, Fare, Embarked.
- Excluir: PassengerId, Name, Ticket, Cabin (irrelevantes/alto % faltantes).

7.2 CLEAN MISSING DATA

Configura Mode = Replace missing values.

- Numéricas (Age, Fare): usar Mediana (robusta a outliers).
- Categóricas (Embarked): usar Moda.

Justificación: la imputación simple mantiene el tamaño de muestra. Cabin puede eliminarse por alto nivel de nulos.

7.3 CONVERT TO INDICATOR VALUES (ONE-HOT ENCODING)

- Columnas a convertir: Sex, Embarked.

Justificación: los algoritmos requieren variables numéricas; one-hot evita orden ficticio de categorías.

7.4 NORMALIZE DATA

- Método recomendado: Z-score (alternativa: MinMax).
- Aplica solo a columnas numéricas (excluye Survived).

Justificación: estandarizar escalas acelera y estabiliza el entrenamiento, especialmente para regresión logística.

7.5 SPLIT DATA

- Fraction: 0.7 (70% train / 30% test).
- Random seed: fija un número para reproducibilidad (ej. 42).
- Stratified split: ON con Survived para preservar proporción de clases.

Justificación: partición robusta y reproducible; el estratificado evita sesgos al evaluar.

7.6 TWO-CLASS LOGISTIC REGRESSION (+ TRAIN MODEL)

- Conecta el algoritmo a Train Model y el puerto de salida de train (Split) al Train Model.
- En Train Model, selecciona la columna Label: Survived.
- Hiperparámetros útiles: número máximo de iteraciones, tipo y peso de regularización (L2/L1).

Sugerencias iniciales: max iters 100–500; comenzar con L2; ajustar el peso de regularización para balancear sesgo-varianza.

7.7 TWO-CLASS BOOSTED DECISION TREE (+ TRAIN MODEL)

- Parámetros de inicio: Number of trees = 100; Max depth = auto.
- Otros parámetros a explorar: learning rate, número mínimo de instancias por hoja, número de hojas.

Justificación: boosting reduce sesgo combinando muchos árboles débiles; sensibilidad a hiperparámetros como learning rate.

7.8 SCORE MODEL Y EVALUATE MODEL

- Score Model: conecta el modelo entrenado y el conjunto de test (salida de Split).
- Evaluate Model: inspecciona AUC, F1, Precision, Recall y la matriz de confusión.

Buenas prácticas: comparar ambos modelos con la misma partición y preprocesamiento; interpretar tanto umbral fijo como curvas ROC/PR.

8. Entregables

4. Captura del DAG completo del pipeline.
5. Captura de configuración de Clean Missing Data.
6. Tabla comparativa de métricas (Logistic vs Boosted).
7. Conclusión breve: modelo ganador, métricas clave, mejoras futuras (tuning/ingeniería de variables).

9. Troubleshooting

- No aparece 'Convert to Indicator Values': verifica que el pipeline esté en 'classic prebuilt components'.
- El Label no se reconoce: en Edit Metadata marca 'Survived' como label y re-ejecuta desde ese punto.
- Métricas incoherentes/muy bajas: confirma Split estratificado y que Survived sea 0/1 sin valores faltantes.
- Demora excesiva: usa una muestra (Sample) para depurar, luego vuelve al dataset completo.
- Error por tipos: revisa que Sex/Embarked estén como categóricas antes del one-hot.
- Data leakage: evita columnas como Name, Ticket, Cabin en el entrenamiento.

10. Reflexión y análisis crítico

- ¿Qué variables aportan más a la predicción? (importancias del árbol, coeficientes de la LR).
- ¿Existe desbalance de clases? Considera ajustar umbral o usar métricas por clase.
- ¿Qué pasaría si no normalizas? Observa el efecto en la convergencia de la LR.
- ¿Cómo cambia el rendimiento con diferentes random seeds? Evalúa estabilidad.