Evaluación 1

Machine Learning

NOMBRES: José Fuenzalida, Esteban Tripaili
ASIGNATURA: Machine Learning
FECHA: 16/09/2025


Inacap

1 INTRODUCCIÓN

El aprendizaje automático (Machine Learning) se ha convertido en una disciplina fundamental dentro de la inteligencia artificial, permitiendo a las máquinas aprender patrones complejos a partir de datos sin ser programadas explícitamente para cada tarea específica. Esta capacidad ha revolucionado múltiples sectores, desde la medicina hasta las finanzas, proporcionando herramientas poderosas para la toma de decisiones basada en datos.

El presente informe tiene como objetivo principal proporcionar una comprensión integral de los fundamentos del aprendizaje automático utilizando los contenidos de AWS Academy – Machine Learning Foundations (modulo 3), en el cual se desarrollan las etapas específicas del desarrollo de modelos. Se explorará el pipeline completo de desarrollo de un modelo de machine learning utilizando las herramientas entregadas por Amazon Sage Maker, desde la carga inicial de datos hasta la evaluación final del modelo. Se prestará especial atención al análisis de XGBoost, un algoritmo de gradient boosting ampliamente utilizado en la industria, y se comparará con las redes neuronales para entender cuándo es más apropiado utilizar cada enfoque.

Se finalizará con una comparativa de desempeño entre XGBoost y Redes Neuronales en un caso de referencia para analizar la competitividad y las falencias de cada modelo.

2 DESARROLLO

2.1 TIPOS DE APRENDIZAJE AUTOMÁTICO

2.1.1 Aprendizaje Supervisado

El aprendizaje supervisado es el paradigma más común en machine learning, donde el algoritmo aprende a partir de un conjunto de datos etiquetados. En este enfoque, disponemos tanto de las variables de entrada (features) como de las salidas deseadas (targets o labels), permitiendo al modelo aprender la relación entre ambas.

Características principales:

- Requiere datos etiquetados para el entrenamiento
- El objetivo es predecir valores para nuevos datos no vistos
- Se evalúa comparando las predicciones con los valores reales conocidos

Tipos de problemas:

- **Clasificación:** Predice categorías discretas (ej: spam vs no spam)
- **Regresión:** Predice valores continuos (ej: precios de casas)

Ejemplos prácticos:

- **Detección de fraude:** Clasificar transacciones como fraudulentas o legítimas
- **Diagnóstico médico:** Clasificar imágenes médicas para detectar enfermedades
- **Predicción de ventas:** Estimar ventas futuras basándose en datos históricos
- **Reconocimiento de voz:** Convertir audio en texto

2.1.2 Aprendizaje No Supervisado

El aprendizaje no supervisado trabaja con datos que no tienen etiquetas conocidas. El objetivo es descubrir patrones ocultos, estructuras o relaciones dentro de los datos sin conocer la respuesta correcta de antemano.

Características principales:

- No requiere datos etiquetados
- Busca patrones inherentes en los datos
- Es más exploratorio y descriptivo

Tipos de problemas:

- **Clustering:** Agrupa datos similares
- **Reducción de dimensionalidad:** Simplifica datos complejos
- **Detección de anomalías:** Identifica datos atípicos

Ejemplos prácticos:

- **Segmentación de clientes:** Agrupar clientes por comportamiento de compra
- **Sistemas de recomendación:** Sugerir productos basándose en patrones de usuarios similares
- **Análisis de mercado:** Identificar tendencias en datos financieros
- **Compresión de imágenes:** Reducir el tamaño de archivos manteniendo calidad

2.1.3 Aprendizaje por Refuerzo

El aprendizaje por refuerzo se basa en el concepto de aprender a través de la interacción con un ambiente. Un agente toma acciones en un entorno y recibe recompensas o castigos, aprendiendo gradualmente qué acciones maximizan la recompensa acumulada.

Características principales:

- Aprendizaje basado en recompensas y castigos
- No requiere datos etiquetados inicialmente
- El agente mejora su comportamiento a través de la experiencia

Componentes clave:

- **Agente:** El sistema que aprende
- **Ambiente:** El contexto donde opera el agente
- **Acciones:** Las decisiones que puede tomar el agente
- **Recompensas:** Señales que indican qué tan buena fue una acción

Ejemplos prácticos:

- **Juegos:** AlphaGo dominando el juego de Go
- **Vehículos autónomos:** Aprender a conducir mediante simulaciones
- **Trading algorítmico:** Optimizar estrategias de inversión
- **Control de robots:** Enseñar a robots a caminar o manipular objetos

3 AMAZON SAGEMAKER

3.1 Dataset

El conjunto de datos biomédicos utilizado en los modulos de aplicación y entrenamiento corrsponden a informacion acerca de problemas ortopedicos, la cual se organiza en dos tareas de clasificación diferentes, pero relacionadas.

La primera tarea consiste en clasificar a los pacientes como pertenecientes a una de tres categorías:

- *Normal* (100 pacientes)
- *Hernia de disco* (60 pacientes)
- *Espondilolistesis* (150 pacientes)

Para la segunda tarea, las categorías *Hernia de disco* y *Espondilolistesis* se fusionaron en una sola categoría etiquetada como *anormal*. Por lo tanto, la segunda tarea consiste en clasificar a los pacientes como pertenecientes a una de dos categorías: ***Normal* (100 pacientes) o *Anormal* (210 pacientes).**

Información de atributos:

Cada paciente está representado en el conjunto de datos por seis atributos biomecánicos que se derivan de la forma y orientación de la pelvis y la columna lumbar (en este orden):

- Incidencia pélvica
- Inclinación pélvica
- Ángulo de lordosis lumbar
- Inclinación del sacro
- Radio pélvico
- Grado de espondilolistesis

La siguiente convención se utiliza para las etiquetas de clase:

- Hernia Disco (HD)
- Espondilolistesis (EL)
- Normal (NO)
- Anormal (AN)

3.2 Carga de Datos

La primera etapa consiste en obtener y cargar los datos desde diversas fuentes. Esta fase es fundamental ya que la calidad y cantidad de datos determinará en gran medida el éxito del modelo.

Aspectos importantes:

- **Fuentes de datos:** Bases de datos, APIs, archivos CSV, JSON, etc.
- **Formato de datos:** Estructurados, semi-estructurados o no estructurados
- **Volumen:** Desde datasets pequeños hasta big data
- **Conectividad:** Asegurar acceso estable a las fuentes de datos

El código asociado a la carga de datos en SageMaker AI es el siguiente:

```
[1]: bucket='c174660a4519471l11586574t1w471112865724-labbucket-iigpbbmplwzq'
```

```
[2]: import warnings, requests, zipfile, io
warnings.simplefilter('ignore')
import pandas as pd
from scipy.io import arff

import os
import boto3
import sagemaker
from sagemaker.image_uris import retrieve
from sklearn.model_selection import train_test_split
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

Los datos por utilizar vienen en un archivo .zip el cual se extrae en un ambiente virtual (maquina EC2 Linux) para ser utilizado en los pasos siguientes.

3.3 Preparación de Datos

En este paso los datos son transformados a un formato valido para ser utilizado por el modelo de machine learning a utilizar, XGBoost. Los datos se organizan en formato .csv.

```
[3]: f_zip = 'http://archive.ics.uci.edu/ml/machine-learning-databases/00212/vertebral_column_data.zip'
     r = requests.get(f_zip, stream=True)
     Vertebral_zip = zipfile.ZipFile(io.BytesIO(r.content))
     Vertebral_zip.extractall()

     data = arff.loadarff('column_2C_weka.arff')
     df = pd.DataFrame(data[0])

     class_mapper = {b'Abnormal':1,b'Normal':0}
     df['class']=df['class'].replace(class_mapper)

     cols = df.columns.tolist()
     cols = cols[-1:] + cols[:-1]
     df = df[cols]

     train, test_and_validate = train_test_split(df, test_size=0.2, random_state=42, stratify=df['class'])
     test, validate = train_test_split(test_and_validate, test_size=0.5, random_state=42, stratify=test_and_validate['class'])

     prefix='lab3'

     train_file='vertebral_train.csv'
     test_file='vertebral_test.csv'
     validate_file='vertebral_validate.csv'
```

Los datos se encuentran en columnas las cuales presentan los siguientes encabezados:

```
[6]: df.columns

[6]: Index(['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle',
            'sacral_slope', 'pelvic_radius', 'degree_spondylolisthesis', 'class'],
           dtype='object')
```

Características del ambiente virtual:

- Fuente de datos: CSV alojado en un recurso S3
- Máquina virtual EC2 Linux (ml.m4. xlarge)

```
s3_resource = boto3.Session().resource('s3')
def upload_s3_csv(filename, folder, dataframe):
    csv_buffer = io.StringIO()
    dataframe.to_csv(csv_buffer, header=False, index=False )
    s3_resource.Bucket(bucket).Object(os.path.join(prefix, folder, filename)).put(Body=csv_buffer.getvalue())

upload_s3_csv(train_file, 'train', train)
upload_s3_csv(test_file, 'test', test)
upload_s3_csv(validate_file, 'validate', validate)

container = retrieve('xgboost',boto3.Session().region_name,'1.0-1')

hyperparams={"num_round":"42",
             "eval_metric": "auc",
             "objective": "binary:logistic"}

s3_output_location="s3://{}/{}/output/".format(bucket,prefix)
xgb_model=sagemaker.estimator.Estimator(container,
                                        sagemaker.get_execution_role(),
                                        instance_count=1,
                                        instance_type='ml.m4.xlarge',
                                        output_path=s3_output_location,
                                         hyperparameters=hyperparams,
                                         sagemaker_session=sagemaker.Session())
```

3.4 Entrenamiento

La herramienta Sage Maker realiza el procesamiento de los datos utilizando xgboost a forma de entrenamiento para ser comparados en los pasos siguientes. Para ello, el algoritmo debe ser capaz de identificar y clasificar los resultados biomédicos en las categorías normal/anormal y la ubicación dentro del cuerpo

```
train_channel = sagemaker.inputs.TrainingInput(
    "s3://{}/{}/train/".format(bucket,prefix,train_file),
    content_type='text/csv')

validate_channel = sagemaker.inputs.TrainingInput(
    "s3://{}/{}/validate/".format(bucket,prefix,validate_file),
    content_type='text/csv')

data_channels = {'train': train_channel, 'validation': validate_channel}

xgb_model.fit(inputs=data_channels, logs=False)

batch_X = test.iloc[:,1:];

batch_X_file='batch-in.csv'
upload_s3_csv(batch_X_file, 'batch-in', batch_X)

batch_output = "s3://{}/{}/batch-out/".format(bucket,prefix)
batch_input = "s3://{}/{}/batch-in/{}".format(bucket,prefix,batch_X_file)

xgb_transformer = xgb_model.transformer(instance_count=1,
                                        instance_type='ml.m4.xlarge',
                                        strategy='MultiRecord',
                                        assemble_with='Line',
                                        output_path=batch_output)

xgb_transformer.transform(data=batch_input,
                          data_type='S3Prefix',
                          content_type='text/csv',
                          split_type='Line')
xgb_transformer.wait()

s3 = boto3.client('s3')
obj = s3.get_object(Bucket=bucket, Key="{}/batch-out/{}".format(prefix,'batch-in.csv.out'))
target_predicted = pd.read_csv(io.BytesIO(obj['Body'].read()),names=['class'])
```

```
2025-09-14 08:41:43 Starting - Starting the training job......
2025-09-14 08:42:18 Starting - Preparing the instances for training...
2025-09-14 08:42:38 Downloading - Downloading input data.....
2025-09-14 08:43:08 Downloading - Downloading the training image.............
2025-09-14 08:44:14 Training - Training image download completed. Training in progress....
2025-09-14 08:44:35 Uploading - Uploading generated training model.
2025-09-14 08:44:48 Completed - Training job completed
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2025-09-14-08-44-49-713
INFO:sagemaker:Creating transform job with name: sagemaker-xgboost-2025-09-14-08-44-50-231
.....................................
...
```

Los resultados del entrenamiento entregan los siguientes datos:

[4]:
```
def binary_convert(x):
    threshold = 0.3
    if x > threshold:
        return 1
    else:
        return 0

target_predicted_binary = target_predicted['class'].apply(binary_convert)

print(target_predicted_binary.head(5))
test.head(5)
```

```
0    1
1    1
2    1
3    1
4    1
Name: class, dtype: int64
```

[4]:

	class	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
136	1	88.024499	39.844669	81.774473	48.179830	116.601538	56.766083
230	0	65.611802	23.137919	62.582179	42.473883	124.128001	-4.083298
134	1	52.204693	17.212673	78.094969	34.992020	136.972517	54.939134
130	1	50.066786	9.120340	32.168463	40.946446	99.712453	26.766697
47	1	41.352504	16.577364	30.706191	24.775141	113.266675	-4.497958

3.5 Resultados

Los resultados obtenidos por xgboost son comparados con la fuente original de datos para realizar la evaluación de rendimiento del algoritmo. Se puede ver a continuación que xgboost da resultados erróneos. En la imagen anterior, el modelo muestra 230 casos en los cuales se encontraros patologías ortopédicas, siendo esto erróneo ya que solo hay 210 en la fuente original de datos. Además, declara 136 datos normales, siendo originalmente 100.

3.6 Evaluación

Para realizar la evaluación de desempeño del algoritmo xgboost primero debemos analizar la matriz de confusión:

[5]:
```
test_labels = test.iloc[:,0]
test_labels.head()
```

[5]:
```
136    1
230    0
134    1
130    1
47     1
Name: class, dtype: int64
```

Ahora puede usar la biblioteca *scikit-learn*, que contiene una función para crear una matriz de confusión.

[6]:
```
from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(test_labels, target_predicted_binary)
df_confusion = pd.DataFrame(matrix, index=['Nnormal','Abnormal'],columns=['Normal','Abnormal'])

df_confusion
```

[6]:

	Normal	Abnormal
Nnormal	7	3
Abnormal	2	19

[8]:
```
from sklearn.metrics import roc_auc_score, roc_curve, auc

TN, FP, FN, TP = confusion_matrix(test_labels, target_predicted_binary).ravel()

print(f"True Negative (TN) : {TN}")
print(f"False Positive (FP): {FP}")
print(f"False Negative (FN): {FN}")
print(f"True Positive (TP) : {TP}")
```

```
True Negative (TN) : 7
False Positive (FP): 3
False Negative (FN): 2
True Positive (TP) : 19
```

Se puede observar en la matriz que xgboost entrega resultados falsos positivos (3) y falsos negativos (2). Esto quiere decir que existen errores en como el algoritmo procesa y clasifica los resultados, ya que resultados anormales son clasificados como normales y viceversa.

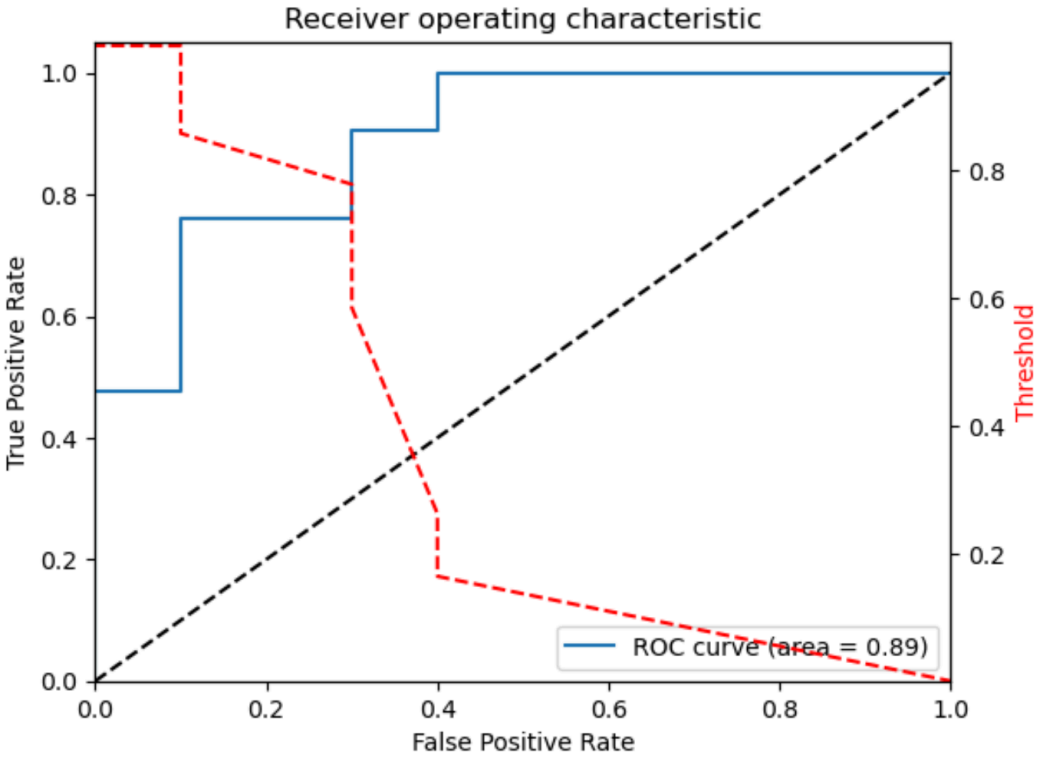
Realizando un estudio mas preciso utilizando las herramientas del notebook podemos obtener un análisis más detallado del desempeño:

[17]:
```
print(f"Sensitivity or TPR: {Sensitivity}%")
print(f"Specificity or TNR: {Specificity}%")
print(f"Precision: {Precision}%")
print(f"Negative Predictive Value: {NPV}%")
print( f"False Positive Rate: {FPR}%")
print(f"False Negative Rate: {FNR}%")
print(f"False Discovery Rate: {FDR}%" )
print(f"Accuracy: {ACC}%")
```

```
Sensitivity or TPR: 90.47619047619048%
Specificity or TNR: 70.0%
Precision: 86.36363636363636%
Negative Predictive Value: 77.77777777777779%
False Positive Rate: 30.0%
False Negative Rate: 9.523809523809524%
False Discovery Rate: 13.636363636363635%
Accuracy: 83.87096774193549%
```

Se puede observar en los porcentajes que XGBoost no es preciso ni exacto a la hora de analizar el conjunto de datos, de hecho, presenta precisión y exactitud menores al 90%, siendo muy notorio los bajos valores en la especificidad en pacientes normales del 70%, es decir, el 70% de los pacientes clasificados como normal presentan en realidad un diagnostico normal, dando un 30% de posibilidad del que el diagnostico sea erróneo y en realidad el paciente si presente un problema ortopédico.

En grafica ROC se puede observar el umbral de incertidumbre (área sobre la línea azul) lo cual confirma la inexactitud de los resultados obtenidos por xgboost.



4 XGBOOST COMO ALGORITMO DE MACHINE LEARNING SUPERVISADO

¿Por qué es un algoritmo supervisado?

Un algoritmo de machine learning se clasifica como supervisado cuando cumple los siguientes criterios:

1. **Disponibilidad de datos etiquetados:** Contamos con un dataset donde cada ejemplo tiene una etiqueta o valor objetivo conocido

2. **Objetivo predictivo claro:** Queremos predecir un valor específico (clasificación o regresión) para nuevos datos

3. **Evaluación objetiva:** Podemos medir el rendimiento comparando predicciones con valores reales

Selección de XGBoost:
XGBoost fue seleccionado por las siguientes razones:

Ventajas técnicas:
- **Rendimiento superior:** Consistentemente obtiene buenos resultados en datos tabulares
- **Eficiencia:** Optimizado para velocidad y uso de memoria
- **Robustez:** Maneja bien valores faltantes y outliers
- **Regularización:** Incluye técnicas para evitar overfitting

Ventajas prácticas:
- **Interpretabilidad:** Proporciona un realce de características importantes y permite análisis SHAP
- **Flexibilidad:** Funciona bien tanto para clasificación como regresión
- **Estabilidad:** Menos sensible a hiperparámetros que otros algoritmos

5 REDES NEURONALES VS XGBOOST

¿Qué son las Redes Neuronales?
Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Consisten en capas interconectadas de nodos (neuronas artificiales) que procesan información de manera distribuida y paralela.

Componentes básicos:
- **Neuronas:** Unidades de procesamiento que reciben inputs, los procesan y generan un output
- **Conexiones:** Enlaces entre neuronas con pesos que determinan la fuerza de la señal
- **Capas:** Organizaciones de neuronas (input, hidden, output)
- **Función de activación:** Determina si una neurona se "activa" o no

¿Cómo Funcionan?

Proceso de forward propagation:
1. Los datos de entrada se alimentan a la capa de input
2. Cada neurona multiplica los inputs por sus pesos y suma un sesgo
3. El resultado pasa por una función de activación
4. El output se propaga a la siguiente capa
5. El proceso continúa hasta la capa de output

Proceso de aprendizaje (backpropagation):
1. Se calcula el error entre la predicción y el valor real
2. El error se propaga hacia atrás a través de la red
3. Los pesos se ajustan para minimizar el error
4. El proceso se repite iterativamente

Cuando usar Redes Neuronales vs XGBoost:

***Redes Neuronales son mejores para*:**

Datos no estructurados:
- **Imágenes:** Reconocimiento facial, clasificación de imágenes médicas
- **Texto:** Procesamiento de lenguaje natural, traducción automática
- **Audio:** Reconocimiento de voz, clasificación de música
- **Series temporales complejas:** Patrones no lineales en el tiempo

Problemas con grandes volúmenes de datos:
- Cuando hay millones de ejemplos de entrenamiento

- Relaciones muy complejas y no lineales
- Necesidad de aprendizaje de representaciones automáticas

XGBoost es mejor para:

Datos tabulares estructurados:

- Datasets con features numéricas y categóricas bien definidas
- Problemas de clasificación/regresiones tradicionales
- Cuando la interpretabilidad es importante

Recursos limitados:

- Datasets pequeños a medianos (< 100k ejemplos)
- Tiempo de desarrollo limitado
- Recursos computacionales restringidos

6 Ejemplo Investigado: Diagnóstico de Cáncer de Mama

Contexto: El Wisconsin Breast Cancer Dataset es un ejemplo clásico donde tanto redes neuronales como XGBoost pueden aplicarse, pero con diferentes ventajas.

Comparación en este caso:

XGBoost:
- Exactitud: ~97%
- Tiempo de entrenamiento: 2-3 minutos
- Interpretabilidad: Alta
- Recursos: Mínimos

Red Neuronal:
- Exactitud: ~98%
- Tiempo de entrenamiento: 10-15 minutos
- Interpretabilidad: Baja (caja negra)
- Recursos: Moderados

Conclusión del ejemplo: Para este caso específico, XGBoost sería la mejor opción debido a:
1. Desempeño comparable con menor complejidad
2. Mayor interpretabilidad (crucial en medicina)
3. Menor tiempo de desarrollo y entrenamiento
4. Menor riesgo de overfitting con dataset pequeño

Sin embargo, si tuviéramos imágenes de mamografías en lugar de datos tabulares, las redes neuronales (específicamente CNNs) serían claramente superiores.

7 CONCLUSION

Este informe ha explorado de manera integral los fundamentos del aprendizaje automático, proporcionando una base sólida para comprender tanto los aspectos teóricos como prácticos de esta disciplina.

Los tres paradigmas principales del aprendizaje automático (supervisado, no supervisado y por refuerzo) abordan diferentes tipos de problemas y requieren enfoques distintos. El aprendizaje supervisado resulta especialmente útil cuando disponemos de datos etiquetados y objetivos predictivos claros, mientras que el no supervisado destaca en la exploración de patrones ocultos y el refuerzo en problemas de toma de decisiones secuenciales.

Las herramientas entregadas por Amazon Sage Maker y la utilización de un Notebook en JupyterLabs facilita la preparación de datos, además de ser estos procesados dentro del mismo ambiente lo que facilita la obtención de las métricas de desempeño.

XGBoost se ha consolidado como una opción robusta para problemas con datos tabulares, ofreciendo un balance excelente entre rendimiento, eficiencia e interpretabilidad. Su capacidad para manejar datos faltantes, resistencia al overfitting y velocidad de entrenamiento lo convierten en una herramienta valiosa para aplicaciones industriales.

En la comparación entre XGBoost y redes neuronales, no existe un algoritmo universalmente superior. XGBoost sobresale en datos estructurados con recursos limitados cuando se requiere interpretabilidad, mientras que las redes neuronales son superiores para datos no estructurados y problemas complejos con grandes volúmenes de datos.

El éxito en machine learning no radica únicamente en elegir el algoritmo más sofisticado, sino en comprender profundamente el problema, preparar adecuadamente los datos y seleccionar las herramientas apropiadas para cada contexto específico. La combinación de

conocimiento teórico sólido y experiencia práctica resulta fundamental para desarrollar soluciones efectivas y confiables.

REFERENCIAS

1. XGBoost Documentation. (2024). *XGBoost Python Package*. https://xgboost.readthedocs.io/
2. AWS Academy. Machine Learning Foundations, Module 3
3. Dheeru Dua and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml