Creación e implementación de servicios de aprendizaje automático con AWS

Building and Deploying Machine Learning Services with AWS

Jorge Alejandro Kamlofsky
CAETI – Universidad Abierta Interamericana – Facultad de Tecnología Informática
Av. Montes de Oca 745, Ciudad de Buenos Aires, Argentina.
GIA: Grupo de Investigación en Inteligencia Artificial – Universidad Tecnológica Nacional – Facultad Regional Haedo. Paris 532, Haedo, Provincia de Buenos Aires, Argentina.

DOI *https://doi.org/10.59471/raia2025222*
Enviado: junio 2025. **Aceptado**: octubre 2025. **Publicado:** diciembre 2025
Como citar: Kamlofsky, J. A. (2025). Creación e implementación de servicios de aprendizaje automático con AWS. Revista Abierta De Informática Aplicada, 9(1). *https://doi.org/10.59471/raia2025222*

Resumen

El desarrollo explosivo de la Inteligencia Artificial en los últimos años ha ido de la mano del desarrollo de la computación en la nube: la disponibilidad de grandes volúmenes de datos y recursos informáticos configurables ha favorecido su evolución. Por lo tanto, es sencillo para los proveedores de servicios de computación en la nube desarrollar y proporcionar servicios de Aprendizaje Automático (ML según sus siglas en inglés) e Inteligencia Artificial (IA) desde sus plataformas. Siendo AWS el proveedor líder mundial de servicios de computación en la nube, este documento detalla la implementación y el despliegue de los servicios de ML más destacados que ofrece. Se incluyen ejemplos reproducibles de los servicios descritos.

PALABRAS CLAVES: Machine Learning en AWS, ML en Amazon, ML en cloud, Servicios de ML.

Abstract

The explosive development of Artificial Intelligence in recent years came hand in hand with the development of cloud computing: Naturally, the availability of large volumes of data and configurable computing resources favored its evolution. It is therefore straightforward for cloud computing service providers to develop and provide Machine Learning (ML) and Artificial Intelligence (AI) services from their platforms. AWS being the leading global provider of cloud computing services, this paper details the implementation and deployment of the most prominent ML services provided by AWS. Reproducible examples of the services described are included.

KEYWORDS: Machine Learning in AWS, ML in Amazon, ML in the cloud, ML services.

Introducción

Trabajos Relacionados

Fue quizás la implementación de IPV6 uno de los hechos más importantes que ayudó al Crecimiento del Big Data. Gracias a IPV6 se conectaron miles de millones de dispositivos a internet, generando una inmensa masa de datos heterogéneos, que debían ser almacenados y procesados. Este problema es conocido como Big Bata (Sarabia Jacome, 2020). La computación en la nube ofreció el ambiente adecuado para su almacenamiento y procesamiento (Kamlofsky, 2024).

Algoritmos y técnicas de Machine Learning (ML) hace muchos años que están siendo estudiados. Pero en los años más recientes, las técnicas de Inteligencia Artificial (IA) han ganado mayor importancia por ser útiles para obtener información relevante a partir de los datos y sobre todo del Big Data: se los utiliza para identificar patrones, generar predicciones, basados en estos volúmenes de datos. Dentro de ML, especialmente las técnicas deep learning han tenido un desarrollo destacado beneficiadas por el Big Data (Sarabia Jacome, 2020).

La disponibilidad de grandes volúmenes de datos y de recursos de cómputo configurables ofrecidos por Cloud Computing, favorecieron la evolución de ML e IA. Como consecuencia natural, los proveedores de servicios de computación en la nube desarrollaron y hoy proveen servicios de Machine Learning (ML) e Inteligencia Artificial (IA) desde la nube (Kamlofsky, 2024).

En el trabajo de Borra (2024), se presentan los principales productos de Google (GCP[1]) para ML, En el de Wang etal (2024) se analiza la usabilidad, performance y eficiencia de desplegar modelos de ML en dos plataformas de Google: BigQuery ML y Vertex AI a través de dos casos. En el trabajo de Sayers (2022) se presentan características generales de los servicios de ML propuestos por Amazon (AWS[2]). En el trabajo del Azure Machine Learning Team (Dorard etal, 2016) se presenta a Azure[3] como plataforma para el trabajo de científicos de datos: Se presenta a Azure ML Studio.

Siendo que, AWS es actualmente la plataforma cloud más adoptada y completa en el mundo (Rivas, 2024), se detalla en este trabajo, la implementación de los servicios de ML más destacados provistos por esta plataforma. Este trabajo está basado en el trabajo hecho por Sayers (2022) de AWS: consiste en un conjunto de instrucciones para implementar los principales servicios de ML en el ambiente de AWS. Se complementa al trabajo mencionado,

[1] Sitio oficial de GCP: https://cloud.google.com/
[2] Sitio oficial de AWS: https://aws.amazon.com/
[3] Sitio oficial de Azure: https://azure.microsoft.com/

incluyendo críticas, implementaciones más prácticas y diferentes y con despliegues hechos en Python[4][5]. Se presentan ejemplos reproducibles, para facilitar su asimilación.

Motivación y Alcance

El trabajo de Sayers (2022) consiste en un muy buen material de entrenamiento hecho por AWS. Sin embargo, por ser un curso hecho por AWS, su contenido está muy refinado, sin errores ni fallos, mostrando un impecable abordaje.

En este trabajo se presentan instrucciones para abordar la implementación de los principales servicios de ML de AWS, abordados desde la propia consola de AWS y también se presentan despliegues, desarrollos e implementaciones hechos con Python. Está hecho con un enfoque más crítico, pragmático y efectivo.

Estructura de este trabajo

El Primer Capítulo posee una introducción y presentación de este trabajo. El Capítulo Segundo propone un breve marco teórico presentando conceptos como Cloud Computing, ML, y ML en Cloud. El Capítulo tercero presenta el Desarrollo Técnico: la implementación de los principales servicios de ML en AWS. Finaliza con las conclusiones.

Marco Teórico

Nociones Básicas acerca de la Computación en la Nube

La definición más estricta es la de NIST que dice la computación en la nube es un modelo que permite el acceso a la red, ubicuo, conveniente y bajo demanda, a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de administración o interacción con el proveedor de servicios (Mell, 2011).

Cloud computing, o computación en la nube, se refiere a la entrega de servicios de computación a través de internet. Incluye almacenamiento, procesamiento, bases de datos, redes, software y análisis. Se presenta como un concepto novedoso que hace uso eficiente de los recursos TI distribuidos en el mundo. El crecimiento exponencial de usuarios de Internet y de centros de análisis de datos que obtienen datos de Internet (hoy conocido como Big Data), son la demanda que apalancó el crecimiento de la Computación en la Nube (Kamlofsky, 2022).

Servicios de ML en la Nube

El flujo de trabajo en un proyecto de ML posee cierta complejidad: desde la obtención del conjunto de datos hasta el despliegue y puesta en producción de los modelos, hay muchos

[4] Sitio oficial de Python: https://www.python.org/
[5] Códigos disponibles en: https://github.com/jkamlofsky/ML_Services_in_AWS

aspectos a considerar (Simon, 2020). Y la infraestructura disponible por los proveedores de servicios Cloud ayuda a que esto se realice eficientemente.

Desde el punto de vista de la infraestructura, las tres plataformas cloud ofrecen servicios de cómputo: máquinas virtuales (servidores y PCs), contenedores, clusters de máquinas, GPU, memoria RAM, capacidad de almacenamiento, siendo este el eje de su propuesta. Incluso algunos ofrecen servicios de computación cuántica. Ofrecen también servicios de networking varios, servicios de bases de datos (SQL y NoSQL), funciones server-less, apis y herramientas para Big Data (Borra, 2022; Borra, 2024, Kamlofsky, 2024; Simon 2020).

La disponibilidad de grandes volúmenes de datos en estas infraestructuras pueden alimentar algoritmos de ML dentro de las mismas plataformas. Antes, se requiere su pre-procesamiento para que datos de orígenes diverso, puedan integrarse para luego ingresarse a diferentes algoritmos de ML: Databrics[6], Apache Spark[7], Sagemaker, entre otros.

Las tres plataformas poseen ambientes para el entrenamiento y prueba de modelos: Google Collab, AWS Sagemaker y Azure ML Studio. Allí pueden desarrollarse instancias que permitan crear modelos, basados en los algoritmos más usados: redes neuronales, clustering, PCA, Arboles de decisión, Random Forest, reglas de asociación, entre otros. Los modelos entrenados, pueden desplegarse usando infraestructura cloud y dejalos disponibles tras una api para que puedan ser consumidos. Se abre también la posibilidad de monetizar los modelos a través de los marketplaces (Sayers, 2022; Borra, 2022). Todo dentro de a misma plataforma.

Desarrollo Técnico: Implementación de Servicios ML

En este capítulo se desarrollan capacidades y funcionalidades de los principales servicios de Machine Learning ofrecidos por Amazon. Ellos son: Rekognition, Textract, Comprehend, Transcribe, Translate, Lex y Sagemaker. Los códigos para su implementación están disponibles en el repositorio del autor[8].

Rekognition

Es un servicio de reconocimiento de objetos en imágenes, Permite reconocer objetos o etiquetas dentro de imágenes. Funciona como un servicio desde la consola de AWS. Es un servicio de visión artificial, basado en cloud, creado por Amazon Web Services en 2016. Ofrece interesantes capacidades de visión artificial desde modelos pre-entrenados o desde modelos que un cliente puede preparar a partir de imágenes propias (Indla, 2021).

[6] Sitio oficial: https://www.databricks.com/
[7] Sitio oficial: https://spark.apache.org/
[8] Acceso público al Repositorio: https://github.com/jkamlofsky/ML_Services_in_AWS

La Figura 1, extraída del trabajo de (Sayers, 2022) esquematiza las capacidades de AWS Rekognition[9]. En particular, dicha figura se extrajo de un video, mostrando sus capacidades de extracción tanto estáticas como dinámicas.

Funciones

Principales operaciones en Rekognition (Sayers, 2022):

CompareFaces: Dentro de una colección de caras, identifica aquella que corresponde con la detectada

DetectFaces: genero, rango etario, sonrisa, ojos y/o boca abierta, etc.

DetectLabels: En una imagen le pone etiqueta a objetos reconocidos, presenta clase, con cierta confianza (p/ej.: clase: construcción. Label: puente. Confidence: 0.998).

Otras funciones: DetectModerationLabels, DetectProtectiveEquipments, DetectText: Identifica textos, RecognizeCelebrities, entre otras.

Aunque en general Rekognition usa modelos pre-entrenados, es posible agregar imágenes y re-entrenar modelos.

Ejemplo de su uso en la consola

Diríjase a Rekognition: En la barra de búsqueda escriba “Rekognition”. Luego haga click sobre el ícono.

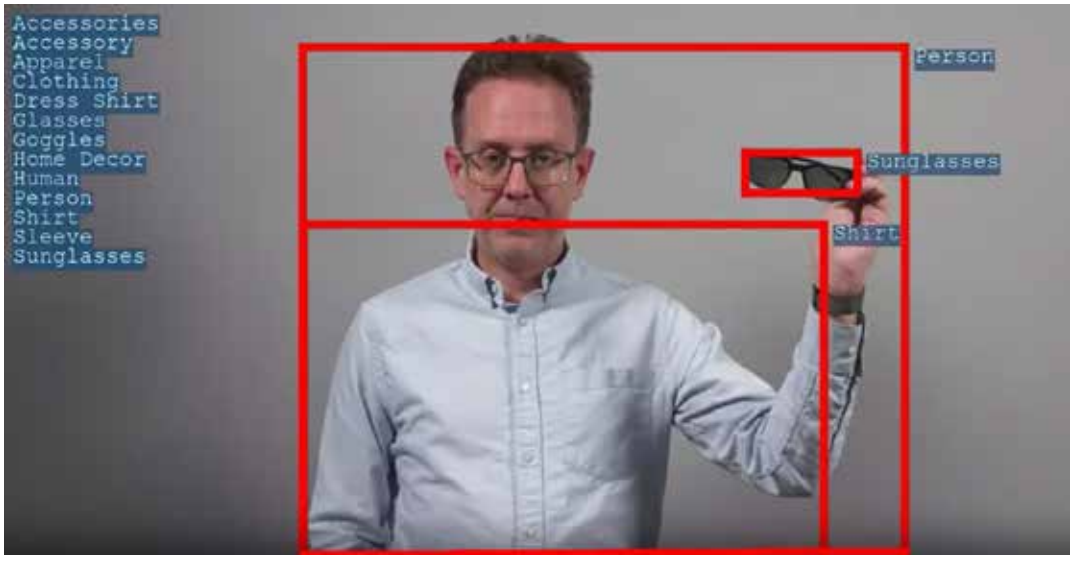


Figura 1. *AWS Rekognition en acción*

[9] Nota del autor: Si bien el video de Sayers (2022) del cual se extrajo la imagen no está hecho totalmente con AWS Rekognition, para su creación, su importancia fue esencial para su creación.

Ingrese una imagen: Puede usar una imagen de ejemplo, o bien analizar una imagen propia. Para ello, puede subirla desde su equipo, o ingresar URL. Se ingresa una imagen llamada: foto03.png

Label Detection: Se hace click en el menú que aparece a la izquierda. La Figura 2 (captura de pantalla) muestra el etiquetado que Rekognition hizo sobre los objetos de la imagen cargada.

Figura 2: *Label Detection aplicada a una imagen*

Se presentan resultados relevantes en la consola. Pero también pueden descargarse todos los resultados haciendo click en: “Download full list”.

Image Properties: Presenta propiedades de la imágenes: colores dominantes, calidad de la imagen, brillo y luminocidad, entre otros. No se presentan características de los objetos contenidos en la imagen. Algunas propiedades presentes en la imagen usada de ejemplo.

- Colores dominantes; #808080 (23,17%), #214f4f (21,58%) y más.
- Calidad de la imagen: Brillo 73.98, nitidez 94.08 y más.

Facial Analysis: Si en la imagen existieran caras, con Facial Analysis puede hacerse un análisis de las mismas. En la imagen de ejemplo (ver Figura 2), se presentan detalles del rostro de la persona agachada (la más cercana). Algunos detalles que se presentan: Parece ser un rostro: 99,8%. Aparentemente es un rostro femenino: 99,8%. Rango de edad: 24 a 30 años. No está sonriendo: 99%. Parece estar triste: 92,1%. No tiene anteojos: 97,5%.

Un Ejemplo de su Uso con Python

En la Tabla 1 se presenta un ejemplo de uso de Rekognition con Python.

TABLA 1. UN EJEMPLO CON PYTHON

Imagen de origen: foto.png

Código: rekognition_build_json.py

```
import glob
import boto3
import json

client = boto3.client('rekognition')
combined = []
for filename in glob.glob('public/photos/*.jpeg'):
    with open(filename, 'rb') as fd:
        response = client.detect_labels(Image={'Bytes': fd.read()})
        entry = { "Filename": filename.replace("public/", "") }
        entry["Labels"] = [
            {
                "Name": label['Name'],
                "Confidence": label['Confidence'],
                "Parents": label['Parents']
            }
            for label in response['Labels']
        ]
        combined.append(entry)

print(json.dumps(combined, indent=2))
```

Salida Json:

```
[
    "Filename": "fotos\\foto03.png",
    "Labels": [
      {
        "Name": "Animal",
        "Confidence": 99.94088745117188,
        "Parents": []
      },
      {
        "Name": "Bird",
        "Confidence": 99.94088745117188,
        "Parents": [
          {
            "Name": "Animal"
          }
        ]
      },
      {
        "Name": "Penguin",
        "Confidence": 99.94088745117188,
        "Parents": [
          {
            "Name": "Animal"
          },
          {
            "Name": "Bird"
          }
        ]
]
… (hay más)
```

Textract

Es un servicio de aprendizaje automático que analiza documentos de texto y automáticamente extrae textos y estructuras. Permite digitalizar texto automáticamente, desde escritura a mano o texto presente en archivos de imágenes o documentos escaneados. Textract identifica la estructura del texto automáticamente. Extrae su geometría (cuadro delimitador) y ubicación de las líneas y palabras de un documento y lo retorna como un objeto cuando se llama a la función Textract. Crea una lista de objetos y relaciones, así como el contexto entre el texto detectado (Kodali et al, 2023).

Funciones

Se destacan las funciones de Textract que se presentan a continuación.

AnalyzeDocument: está hecho para el análisis de documentos en general, identificando en los formularios del mismo: campos prioritarios, encabezados y contenido al que puede accederse con consultas.

AnalizeExpense: está diseñado para el análisis de documentos comerciales: identifica fecha y hora, contenido, importes, y demás.

AnalyzeId: Funcionalidad diseñada para detectar campos y contenidos en documentos identificatorios. Detecta Nombre y Apellido, Número de Documento, imagen, Domicilio, entre otros.

Ejemplo de uso en la Consola de AWS

En la Figura 3 se muestra un documento que se analizó: consiste en la captura de pantalla de una oferta en un marketplace.

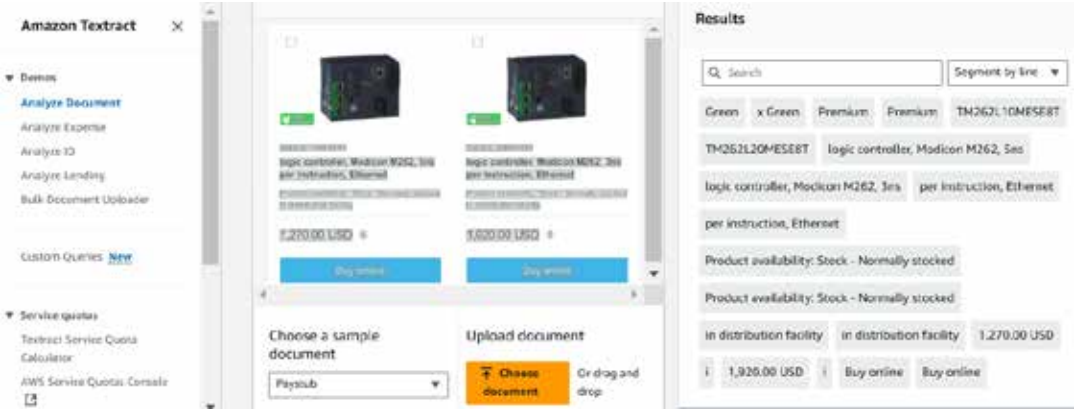


Figura 3. *Análisis de Documentos con Textract*

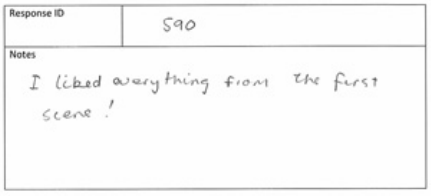
Se observa que se ha detectado correctamente la totalidad del texto presentado dentro de la imagen.

Un ejemplo de uso con Python

En la Tabla 2 se presenta un ejemplo del uso con Python de la función de Textract: analize_document, analizando un documento.

TABLA 2. EJEMPLO DE ANÁLISIS DE DOCUMENTO CON TEXTRACT EN PYTHON

Documento de Origen: 001.jpg



Código: textract_main.py

```
# Librerías
import glob
import boto3
import json
import csv
import sys

csv_array = []
client = boto3.client('textract')
for filename in glob.glob('raw_images/001.jpg'):
    csv_row = []
    print(f"Processing: {filename}")
    with open(filename, 'rb') as fd:
        file_bytes = fd.read()
    # Procesamiento de imágenes
    response = client.analyze_document(
        Document={'Bytes': file_bytes},
        FeatureTypes=["QUERIES"],
        QueriesConfig={
            "Queries": [
                {"Text": "what is the response id", "Alias": "ResponseId"},
                {"Text": "what are the notes?", "Alias": "Notes"},
            ]
        }
    )
    # Response completa a Textract
    print(json.dumps(response, indent=4))

    for block in response["Blocks"]:
        if block["BlockType"] == "QUERY":
            query_alias = block["Query"]["Alias"]
            answer_id = next(rel["Ids"] for rel in block["Relationships"] if rel["Type"] == "ANSWER")[0]
            answer_text = next(b for b in response["Blocks"] if b["Id"] == answer_id)["Text"]
            csv_row[query_alias] = answer_text
    csv_array.append(csv_row)
# Escritura de CSV
writer = csv.DictWriter(sys.stdout, fieldnames=["ResponseId", "Notes"], dialect='excel')
writer.writeheader()
for row in csv_array:
    writer.writerow(row)
```

Salida:

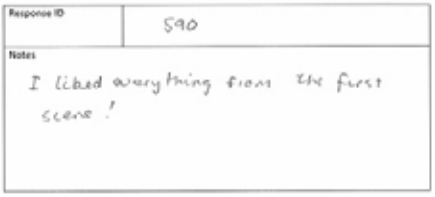

```
{
"BlockType": "QUERY",
"Id":                "5b95effl-7b10-4611-9aab-
0ffbeleadlbe",
"Relationships": [
  {
  "Type": "ANSWER",
  "Ids":             ["2705cd0e-bb73-40df-8954-
b91b6ed0d77c"],
  "Query": {
  "Text": "What is the response id",
  "Alias": "ResponseId"}
},
{
"BlockType": "QUERY_RESULT",
"Confidence": 99.0,
"Text": "590",
"Geometry": {
  "BoundingBox": {
    "Width": 0.06050015240907669,
    "Height": 0.01886042021214962,
    "Left": 0.4029218554496765,
    "Top": 0.09675661474466324},
  "Id":              "2705cd0e-bb73-40df-8954-
b91b6ed0d77c"
},
{
"BlockType": "QUERY",
"Id":                "8301f8c4-2eb4-4596-88dl-
f06d6df622e9",
"Relationships": [
  {
  "Type": "ANSWER",
  "Ids":            ["cbe23a09-2fle-4972-bdfl-
54bcef0f0bb0"],
  "Query": {
  "Text": "What are the notes?",
  "Alias": "Notes"}
},
{
"BlockType": "QUERY_RESULT",
"Confidence": 98.0,
"Text":"I liked every thing from the first
scene",
  "Geometry": {
    "BoundingBox": {
      "Width": 0.6340733766555786,
      "Height": 0.06546258926391602,
      "Left": 0.16090142726898193,
      "Top": 0.16538071632385254},
}
```


En el ejemplo presentado en la Tabla 2 (imagen obtenida del trabajo de Sayers, 2022), *Textract* busca la respuesta a una pregunta presentada en el código. Por ello, la estrategia es aquí, buscar la “query” y su correspondiente “query_result”. En la Tabla 3 se presenta un ejemplo del uso con Python de la función de Textract: detect_text.

El uso de *detect_text* puede resultar más sencillo, ya que trae todo el texto detectado. Luego es posible separar los contenidos según necesidad.

TABLA 3. EJEMPLO DE DETECCIÓN DE TEXTO CON TEXTRACT EN PYTHON

Documento de Origen: 001.jpg



Código: textract_detect_text.py

```
import glob
import boto3
import json
import csv
import sys

csv_array = []
client = boto3.client('textract')
for filename in glob.glob('raw_images/001.jpg'):
    csv_row = {}
    print(f"Processing: {filename}")
    with open(filename, 'rb') as fd:
        file_bytes = fd.read()

    # Procesamiento de imágenes sin consultas
    response = client.detect_document_text(
        Document={'Bytes': file_bytes}
    )

    # Response completa a textract
    print(json.dumps(response, indent=4))

    # Procesamiento de la respuesta
    extracted_text = ""
    for block in response["Blocks"]:
        if block["BlockType"] == "LINE":
            extracted_text += block["Text"] + "\n"

    csv_row["ExtractedText"] = extracted_text.strip()
    csv_array.append(csv_row)

# Escritura de CSV
writer = csv.DictWriter(sys.stdout, fieldnames=["ExtractedText"], dialect='excel')
writer.writeheader()
for row in csv_array:
    writer.writerow(row)
```

Salida:


```
{
"BlockType": "LINE",
"Confidence": 99.8954849243164,
"Text": "Response ID",
"Geometry": {
  "BoundingBox": {
    "Width": 0.0989999920129776,
    "Height": 0.013172809965908527,
    "Left": 0.12449004501104355,
    "Top": 0.08601831644773483},],
  "Polygon": [
    {"X": 0.12449190020561218,
     "Y": 0.08608761429786682},
    {"X": 0.22349004447460175,
     "Y": 0.08601831644773483},
    {"X": 0.22348883748054504,
     "Y": 0.09912172704935074},
    {"X": 0.12449004501104355,
     "Y": 0.09919112920761108} ]
},
{
"BlockType": "LINE",
"Confidence": 90.7627102006036,
"Text": "590",
...
},
{
"BlockType": "LINE",
"Confidence": 99.92236328125,
"Text": "Notes",
...
},
{
"BlockType": "LINE",
"Confidence": 95.04239654541016,
"Text": "I liked every thing from the
first",
...
},
```


Comprehend

AWS Comprehend es un servicio de procesamiento de lenguaje natural (NLP según sus siglas en inglés) basado en aprendizaje profundo que permite hallar y extraer enfoques y relaciones de interés a partir de un texto. Pretende soportar a los desarrolladores y a las empresas para obtener información a partir de datos de texto no estructurados, tales como los que se hallan en sitios web de empresas, documentos varios, reseñas de clientes y posts en redes sociales. *AWS Comprehend* realiza tareas diversas en el proceso de lenguaje natural a partir de modelos de aprendizaje automático que ayudan a obtener información de utilidad a partir de fuentes variadas de texto (Movinuddin, 2023).

Funciones Básicas

Las principales funcionalidades de AWS Comprehend se enuncian a continuación: Detección de Entidades, Frases principales, Idioma, Información de Identificadores Personales

(PII), Análisis de sentimientos, Análisis Sintáctico, Modelos personalizados de clasificación de textos y entidades. Algunas de éstas se explayan más abajo.

Detección de entidades: En un texto, identifica palabras o frases que permiten reconocer objetos, personas, del mundo real.

Frases Principales: Dado un texto, se identifican a las frases más importantes.

Idioma: Identifica el idioma de un texto. En la Figura 4 (extraída del trabajo de Sayers, 2022) se muestra un ejemplo de identificación de idioma.

PII: Dentro de un texto puede hallarse cierta información que puede ser usada para identificar a una entidad. Por ejemplo: número de pasaporte, número de placa patente, dirección, etc.

Análisis de Sentimientos: Dado un texto, se lo clasifica como positivo, negativo o neutro, con una cierta probabilidad. Se trata de identificar la carga emotiva que allí se presenta.

Análisis Sintáctico: En una oración, puede identificarse el núcleo, sujeto y predicado, sustantivos, verbos, etc.

Modelos Personalizados: Combinando los anteriores, y con datasets propios, pueden lograrse modelos propios, a medida.

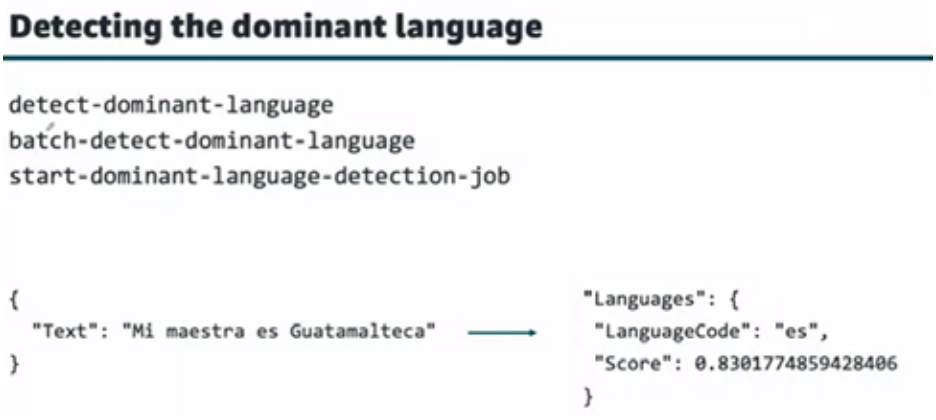


Figura 4. *Detección de idioma dominante*

Un ejemplo desafiante en la Consola de AWS Comprehend

Se introduce un texto en español que incluye información personal, sentimientos variados e incluso ironía, entidades, y demás. Este texto se propone intencionalmente a los fines de poner a prueba aspectos específicos de este sistema.

El texto: "Estimado Señor director de la Dirección Impositiva Regional Señor Juan Melchor de los Ríos. Quien le escribe, Don Carlos Schmith con DNI 12.345.678 fue demandado por su institución por pagos fuera de término de varias cuotas del impuesto y a consecuencia embargada la totalidad de mis depósitos en cuentas bancarias, lo cual me hizo mandarle esta carta, con mis profundos deseos que Ud. arda en el infierno con toda su familia. Por otro lado, me alegra descubrir la eficiencia de sus empleados que han hallado y embargado mi automóvil patente AB123CD. Cordialmente, Carlos."

Entidades: Identificó: "director" (tipo "Persona"), "Dirección Impositiva Regional" (tipo "Organización"), "Señor" (tipo "Persona"), "Melchor de los Ríos" (tipo "Persona"), "Don Carlos Schmith" (tipo: "Persona"), y otros. Observar que se identifica adecuadamente personas y organizaciones. Notar que el nombre "Juan Melchor de los Ríos" incorpora el sustantivo "Ríos", lo cual fue integrado correctamente dentro del Nombre en "Persona".

Idioma: Para el texto presentado en este ejemplo, el idioma detectado es "Español – ES" (confianza: 0,99)

PII: Información de Identificación Personal. Al momento de hechas estas pruebas, la detección de PII no está disponible en Español.

Análisis de Sentimientos: El resultado de Análisis de Sentimientos hecho sobre el texto del ejemplo indica que domina el sentimiento neutral (41%), seguido por sentimiento mixto (34%), seguido por el sentimiento Positivo (21%). Al sentimiento negativo le atribuye un 1% de confianza, cuando por la ironía contenida, es un texto fuertemente negativo. Nótese también, que el texto "Ud. arda en el infierno con toda su familia" es evidentemente violento, sin ironía alguna, lo cual sin discusión, este análisis no resulta correcto.

Un ejemplo de Uso con Python

En la Tabla 4, se muestra un ejemplo del uso de AWS Comprehend con código Python, basado en el trabajo de Sayers (2022).

Transcribe

En el marco del progreso general de la inteligencia artificial, el reconocimiento automático del habla (ASR según sus siglas en inglés) ha mejorado maravillosamente en los últimos tiempos y, en varios aspectos, iguala o mejora aún más la calidad de los expertos humanos. Estas mejoras nos permiten identificar similitudes en el habla a partir de transcripciones generadas automáticamente (Fauss et al, 2024). Se presenta Amazon Transcribe como un servicio que convierte voz a texto. Toma la palabra hablada y la convierte en palabra de texto.

Funciones Básicas

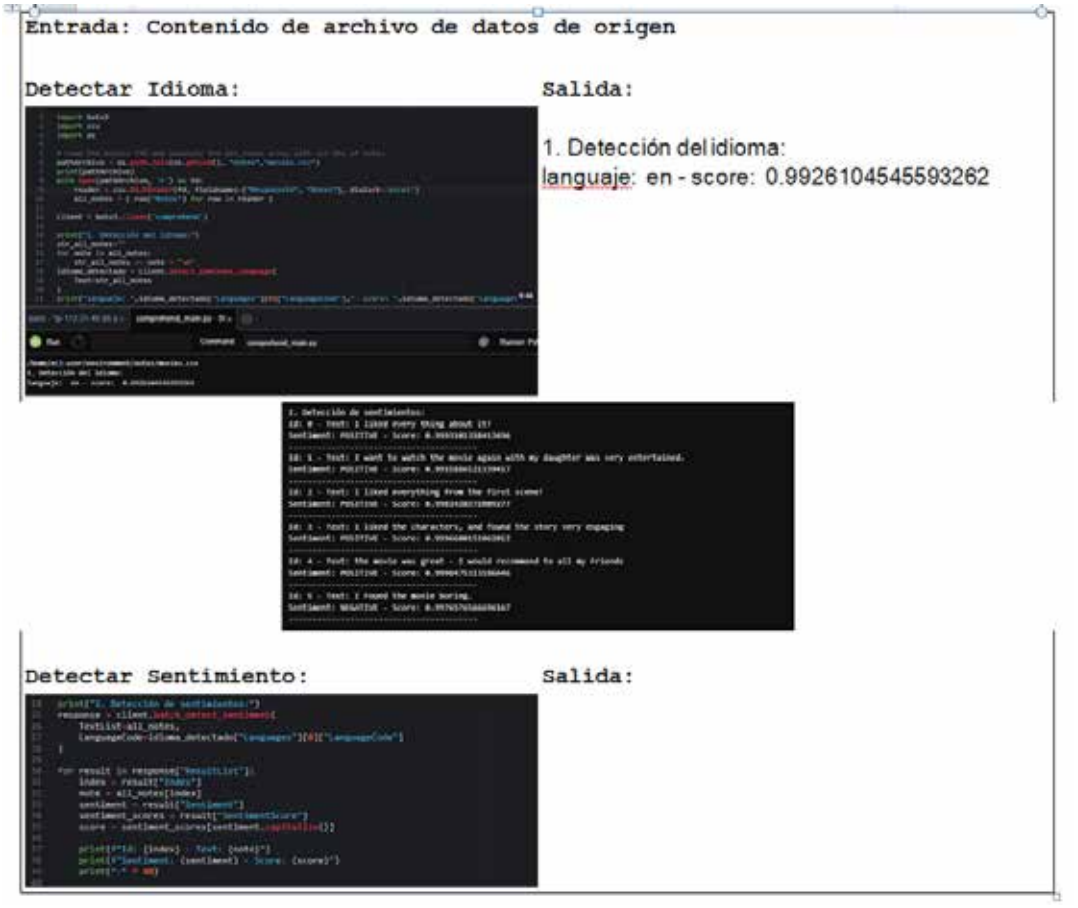
Se describe a continuación, las funcionalidades básicas de Transcribe.

Real Time Transcription: Permite que se genere el texto automáticamente a medida que fluye una conversación.

Transcription Jobs: Genera una transcripción a partir de un archivo de sonido de una conversación.

TABLA 4. ANÁLISIS DE SENTIMIENTOS CON PYTHON

	A	B	C
1	511	I liked every thing about it!	
2	450	I want to watch the movie again with my daughter Was very entertained.	
3	590	I liked everything from the first scene!	
4	643	I liked the characters, and found the story very engaging	
5	206	the movie was great - I would recommend to all my Friends	
6	226	I Found the movie boring.	
7			
8			
9			



Un Ejemplo de Real Time Transcription en la Consola de AWS

Real Time Transcription: En la caja de búsqueda escribir “Transcribe” y hacer click en el ícono cuando aparezca. Antes de iniciar una transcripción en tiempo real, seleccionar el idioma. Luego puede iniciar la transcripción de una conversación. Haciendo click en el botón amarillo “Start” iniciará un flujo de conversación. Haciendo click en el botón amarillo “Stop” detendrá el flujo de conversación.

Se hizo un ejemplo siguiendo estas instrucciones: Se hizo click en “start” y se narró un texto, luego se hizo click en “stop”. En la caja “Transcription output” apareció el texto narrado: “Esta aplicación de Machine Learning es muy interesante y muy útil”. Una vez finalizada la transcripción puede disponerse en cloud o descargarse.

Un Ejemplo de Transcription job en la Consola de AWS

En el menú de la izquierda haga click en “Transcription Jobs”. Una vez en la página, hacer click en el botón amarillo “Create a Job”. Dar nombre y Setear idioma de la transcripción. Seleccionar archivo de audio. Luego presionar Next y luego Create Job. La Figura 5 muestra un ejemplo de uso de Transciption Job: se transcribe una narración de voz incluida en un archivo de sonido previamente grabado.

Aparece el Job en el listado. Esperar hasta que la transcripción se complete. Luego abrir el Job y se mostrará el detalle, tal como se observa en la Figura 5. Análogamente a otros trabajos, el resultado puede descargarse.

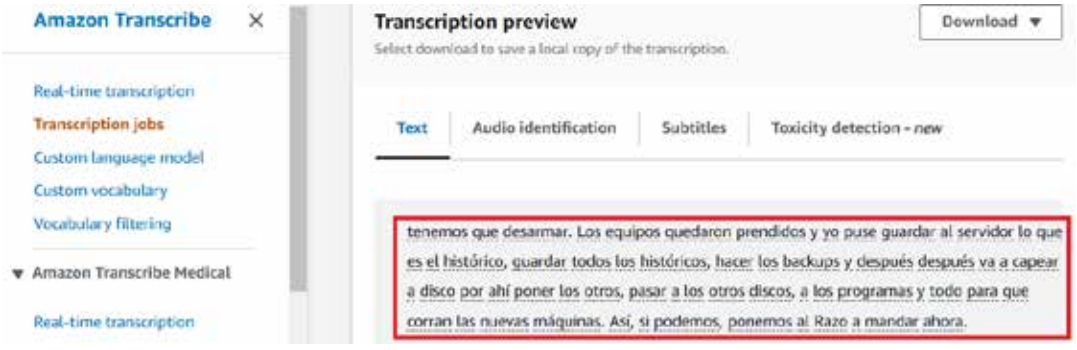


Figura 5. *Transcripción de un archivo de sonido*

Un ejemplo de Uso de Transcript en Python

En ta Tabla 5 se muestra un ejemplo del servicio Transcript con código Python.

TABLA 5: Implementación de transcript en Python

Entrada: "Audio 2024-11-25_1528.ogg" (Audio de Whatsapp)
Porción principal del código:

```
# Path del archivo
s3_uri="s3://bucket-pruebas-ml/Audio 2024-11-25_1528.ogg"
#s3_uri="s3://bucket-pruebas-ml/Audio 2024-11-21_1402.ogg"
# Lenguaje-Code: es-ES
language_code="es-ES"
# Output Bucket:
bucket_name="bucket-pruebas-ml"
# job_name
job_name="mi_tanscripcion_audio_"
# Audio file
audio_file="Audio 2024-11-25_1528.ogg"
# UUID para identificacion unica
import uuid
```

✓ 0.0s

```
transcribe_audio(bucket_name, audio_file, job_name)
```

✓ 18.3s

Salida:

```
Transcription:
Bajate lo que todo lo que encuentres. Estás separado por clases. Creo que
por unidades, también. Bájate básicamente las presentaciones de clase que
se tienen un resumen teórico de lo que cada tema trata en modo
presentación. Así que es un poquito más afable.
Transcription job 'mi_tanscripcion_audio_-1732646381-789998' deleted.
```

Translate

AWS Translate es un servicio cloud que traduce textos ente idiomas. Permite traducciones tanto en tiempo real como por lotes. Permite traducir más de 75 idiomas de forma rápida y efectiva. Proporciona traducciones más precisas y naturales al utilizar la traducción automática neuronal (NMT) como infraestructura de traducción. Gracias a la gran potencia de la infraestructura de AWS, Amazon Translate proporciona un servicio de traducción simultáneo a millones de usuarios en todo el mundo (Oncu & Kuflu, 2024).

Operaciones Básicas

Las operaciones básicas que posee Translate son: Real Time Translation y Batch Translation. Admite también configurar trato formal o no, en varios idiomas: francés, Inglés, español, Italiano, entre otros. Admite también, customizar o adaptar terminología. Para ello, admite archivos csv, tmx y tsv.

Un Ejemplo de Uso de Translate en Consola

Las operaciones básic Realizaremos una traducción en tiempo real en la consola. Para ello, acceder a la consola, buscar “Translation”. Una vez que se accedió, seleccionar “Real Time Translation”, ingresar un texto en la caja de texto, y seleccionar los idiomas. Se observará que en la caja de salida “Translated text” aparece automáticamente el texto traducido al idioma elegido.

En la consola, se ingresó el siguiente texto: “Una anomalía sorpresiva del clima a pocos días del verano rompe el pronóstico y deja una pregunta. Nuevos datos oficiales”. Seleccione el idioma objetivo. En este ejemplo: English EN. En tiempo real se presenta en la caja de texto “Target Languaje”: “A surprising weather anomally a few days into summer breaks the forecast and leaves a question. New official data”.

Luego de realizada la traducción en tiempo real, se muestra información de cantidad de caracteres e idioma detectado (si en el idioma de fuente se dejó “Auto”).

Un Ejemplo de Uso con Python

Se dispone de un texto que se obtuvo de una transcripción de un archivo de audio. Ese texto se encuentra en Portugués Basilero, y se obtuvo del trabajo de Sayers (2022). En la Tabla 6 se presenta: el texto de entrada, el código y el texto de salida.

TABLA 6. IMPLEMENTACIÓN DE AWS TRANSLATE

Código usado:

```
# Función para traducir el texto
def translate_text(text, source_language_code, target_language_code):
    try:
        response = translate_client.translate_text(
            Text=text,
            SourceLanguageCode=source_language_code,
            TargetLanguageCode=target_language_code
        )
        return response['TranslatedText']
    except Exception as e:
        print(f"Error translating text: {e}")
        return None

# Parámetros y ejecución de Función:
lenguaje_destino="es-ES"
lenguaje_origen="pt-BR"
translated_text = translate_text(transcript, lenguaje_origen, lenguaje_destino)
print("Traduccion: ",translated_text)
```

Texto de Entrada (Portugués Brasilero):

Olá, seja bem-vindo. Você quer aprender mais sobre os serviços da AWS? É um prazer imenso pra mim te ensinar tudo o que eu sei, meu nome é Rafael e eu trabalho na Amazon Web surce, criando treinamentos como esse aqui que você tá assistindo, esperamos que você esteja curtindo essa jornada de aprendizado usando os serviços de inteligência artificial que a nuvem da Amazon oferece. Você sabia que a Amazon oferece serviços para te ajudar com visão computacional, detecção de defeitos em linhas de montagem, reconhecimento de dados em texto e conteúdo falado, métricas de negócios, setor de saúde, melhoria de código fonte e até mesmo inteligência artificial no ramo industrial? Tudo isso para melhorar a experiência de seus clientes.

Texto de Salida (Traducido a Español):

Hola, bienvenidos. ¿Desea obtener más información sobre los servicios de AWS?. Es un inmenso placer para mí enseñarte todo lo que sé, me llamo Rafael y trabajo en Amazon Web Source, creando capacitaciones como la que estás viendo. Esperamos que esté disfrutando de este viaje de aprendizaje utilizando los servicios de inteligencia artificial que ofrece la nube de Amazon. ¿Sabías que Amazon ofrece servicios que te ayudan con la visión artificial, la detección de defectos en las líneas de montaje, el reconocimiento de datos en contenido textual y oral, las métricas empresariales, el sector de la salud, la mejora del código fuente e incluso la inteligencia artificial en el sector industrial? Todo esto para mejorar la experiencia de sus clientes.

Amazon Lex

Es un servicio de chat que soporta conversaciones en interfases de voz o de texto. Usa el mismo motor de conversaciones que tiene Alexa. Ello garantiza el uso de capacidades conversacionales ampliamente probadas. Los chatbots permiten obtener ágilmente información que el cliente provee de una transacción.

AWS Lex admite tanto chats de voz como de texto y es posible implementarlos en plataformas móviles y de mensajería. Es posible que casi cualquiera que lo desee pueda crear chatbots conversacionales rápidamente ya que con AWS Lex, ya que no se necesita mucho conocimiento en aprendizaje profundo: para crear un bot, los usuarios pueden simplemente especificar el flujo de conversación básico en la consola. Lex administra el diálogo y ajusta dinámicamente las respuestas en la conversación (Sreeharsha et al, 2022).

Conceptos Básicos Relacionados con Chatbots

Intención (Intent): Es una acción que se quiere completar. La intención puede presentarse en gran cantidad de formas. Por ello, el uso de NLP permite relacionar múltiples intenciones con una acción.

Slot: Son los parámetros necesarios para completar la acción. En un chatbot pueden definirse diferentes tipos de slots.

promptSpecification: Una intención tiene asociada una especificación del prompt: se presenta la estructura de la oración de la acción completa. Entre llaves se presenta el slot de la respuesta. Ejemplo: “OK, Reservo una habitación por {cuatro} noches a partir de {fecha de ingreso}. Confirma la reserva?”

Declination: Se presenta una respuesta en caso que el prompt no es aceptado.

DialogCodeHook: Puede agregarse mayor lógica relacionando al chatbot con una función lambda. Normalmente, para efectivizar una reserva se llama a una función Lambda.

Slot resolution: Puede restringirse a un conjunto de valores pre-definidos, o bien puede permitirse respuestas abiertas, definidas mediante un modelo NLP. Por ejemplo, Slot_name: tipo de habitación; Slot_type: Los tipos de habitación disponibles. Slot_resolution: restringido a un listado pre-definido o no. Slot_values: {simple, doble, presidencial}.

Intenciones integradas: Lex posee múltiples intenciones integradas como ser: ayuda (AMAZON.HelpIntent), pausa (AMAZON.PauseIntent), repetir (AMAZON.RepeatIntent), cancelar (AMAZON.CancelIntent).

Slots Integrados: Fecha: AMAZON.Date puede identificar fechas de formas múltiples. Por ejemplo: “Primero de Febrero”, “El próximo jueves”, “Este fin de semana”.

Expresiones (Utterances): son frases o expresiones de ejemplo que los usuarios podrían decir para activar un intent específico. Ayudan a entrenar el modelo de lenguaje natural para comprender las variaciones en el lenguaje del usuario.

Un Ejemplo de Chatbot en la Consola de AWS Lex

El ejemplo consiste en implementar un chatbot en Lex que consulte a una Lambda el horario de funcionamiento de un negocio. A continuación se enuncian e ilustran los pasos. A grandes rasgos: En consola Lex, Configurar el bot, crear los slots, crear los intents, integrarlos con una función Lambda[10] para manejar la lógica del negocio. Ver los pasos a seguir en la Tabla 7.

TABLA 7. PASOS A SEGUIR PARA LA CREACIÓN DEL BOT DE EJEMPLO EN LEX

- *Ir a consola Lex:*
- *Configurar el bot: Click en "Create Bot". En "Configue bot settings": darle nombre (bot name). En: IAM permissions: "Create a role with basic Lex permissions". En COPPA (children Privacy Protection Act), selecciona "No". Todo el resto dejarlo que está por default y seleccionar "Next". Si no está su Idioma, agregarlo.*
- *Crear Slots: Click en Slot Types. Agregar nuevo tipo de Slot. Acá puede insertar un Slot customizado, propio (Add blanck slot type), como es "sucursal", o bien insertar slots ya integrados dentro de Amazon (Use built-in type) como ser: fecha, hora, etc. Blanck slot type: En "Resolution" elegimos "Restrict..." para que la respuesta se limite a los valores propuestos. Abajo se listan los valores. En este ejemplo, se agregaron las siguientes sucursales: Buenos Aires, Córdoba y Mendoza. Es decir, se les asignó los valores "Buenos Aires", "Córdoba" y "Mendoza" al slot type "sucursal".*
- *Crear Intents: En el menú de la izquierda, arriba de "Slot types" usado previamente, se encuentra "Intents". Al hacer click, se presenta la posibilidad de crear un nuevo Intent. Crear un nuevo Intent desde cero: Se elige: Add empty intent. Darle nombre. Agregar expresiones (utterances) y sintaxis de los Intents.*

En este ejemplo de bot creado, se desea conocer el horario de funcionamiento de las diferentes sucursales. Observar las expresiones incorporadas, junto con su sintaxis. La Figura 6 presenta las diferentes expresiones o utterances.

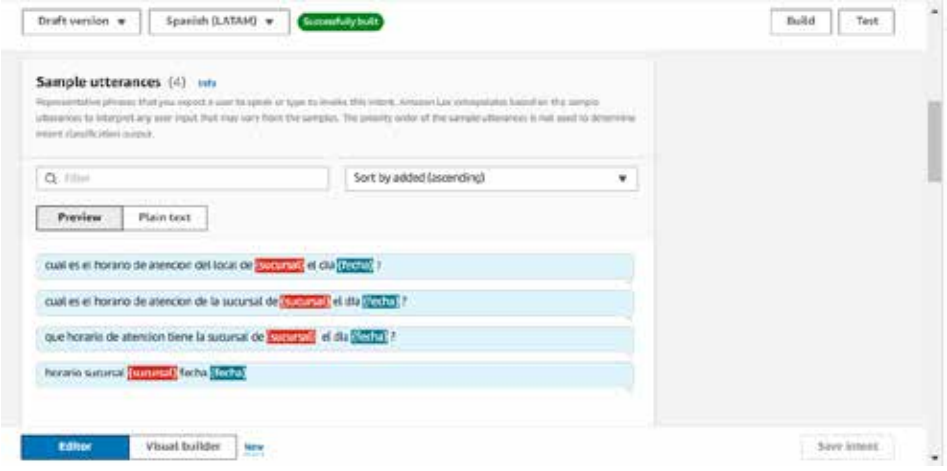


Figura 6. *Ejemplo de listado de expresiones en AWS Lex*

[10] Función Lambda en AWS: "es un servicio informático que puede utilizar para crear aplicaciones sin aprovisionar servidores". Fuente: https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html

En el ejemplo mostrado en la Figura 6, se tiene el intent llamado “horarios_de_funcionamiento”, con las expresiones “A qué hora abre el negocio de {sucursal} ?”, “A qué hora abre el local de {sucursal}?” y “A qué hora abre el local de {sucursal} el {fecha} ?”. Mientras que el slot sucursal es del tipo custom, el slot fecha es tipo built-in.

Una vez creado el bot en Lex, se requiere su integración con una función Lamba para que se despliegue y se deje disponible al usuario. Los pasos se enuncian a continuación. Integración con Función Lambda: En “Fullfilment” Seleccionar “Active”, crear la función Lamda: Ir a Lambda, crear la función en Python, insertar el código: Ver Anexo lex_lambda_horarios.py. Luego hacer Deploy para que esté disponible. Esta función lambda se encarga de aspectos de la lógica del negocio del chatbot y actúa complementando al bot de Lex. Puede hacer un Test de la función Lambda para verificar su correcto funcionamiento. Ello puede observarse en la Figura 7.

Figura 7. *Respuesta de la Función Lambda “lex_lambda_horarios”*

La Figura 7, se muestra como prueba de funcionamiento de la función Lambda. Se ve a la sucursal Mendoza y la fecha “2024-12-20” que es viernes. El horario esperado es: “Por la mañana: 8 a 12 - Por la tarde: 16 a 20”, que es lo que se observa en el body de la respuesta de la prueba (en rojo). Lo que es lo esperado.

Para la integración del chatbot con la función Lambda recientemente creada, se enuncian aquí los pasos. Asociar la función lambda con el chatbot. Ir a Aliases, que es una propiedad que aparece en el menú de la izquierda del chatbot. En Alias: click en “testBotAlias”. En Language: “Spanish Latin-American”. Allí seleccionar la función Lambda y grabar los cambios. Integrar los cambios: En el menú de la izquierda ir a Spanish (LATAM). Hacer click en el botón “Built”. Una vez que el bot pudo integrarse, aparece el respectivo mensaje, en verde.

Una vez integrados puede verificarse su funcionamiento. Para ello: Posicionarse en el bot, y en el intent. Click en Test. Ingresar consulta (en linea con las utterances). En este caso, como ejemplo, se consulta por horario de funcionamiento de la sucursal de Mendoza el día 2024-12-20 (que es viernes). La Figura 8 muestra a la interacción exitosa con el bot integrado de Lex.

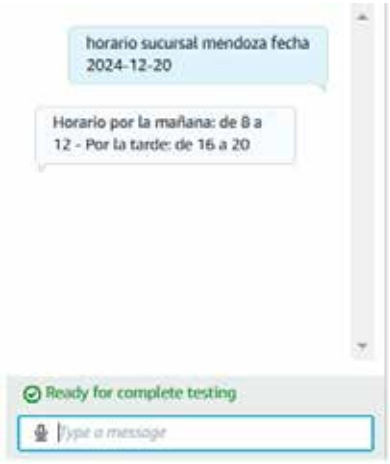


Figura 8. *Prueba exitosa del bot de Lex integrado*

Chatbot como Servicio

Si bien Amazon Lex no proporciona una URL directa para interactuar con el bot, puede crearse una API o un front-end para hacerlo. En este apartado se propone crear una API en el entorno de AWS para poder exponer al chatbot de Lex como un servicio.

La API se conectará con Lex vía una función Lambda y retornará desde la API la respuesta de Lex. La Figura 9 esquematiza la arquitectura del servicio de Chatbot usando a Postman como un servicio de prueba.

Vía Postman pueden pasarse los requerimientos al servicio (a través de la API), y se recibe la respuesta, tal como lo presenta la Figura 9: Se ilustran componentes y conexiones. A continuación se describen algunos de esos componentes.

La función Lambda apiDelBot: Se crea una función Lambda que se encargará de obtener disponer de una dirección web y re-dirigir el tráfico al servicio de Lex.

Api-Gateway apiDelBot-API: Las funciones Lambda requieren de una api-gateway que actúa como iniciador o catalizador para su lanzamiento. Para su creación, hacer click en Add trigger. Seleccionar Api-gateway. En resumen, los pasos son: Crear una api tipo Rest, Crear un recurso y un método (POST) e integrar con la función Lambda apiDelBot. Por último, configurar encabezados (consulta y User-Id) y deployar. Para más información, consultar el trabajo de (Poccia, 2016)[11]. Obtener el Endpoint de la api para ser usado por Postman.

[11] Nota del autor: El trabajo referido presenta en un amplio detalle cómo implementar api-gateways en AWS. Sin embargo, considere que desde la fecha de publicación de su trabajo hasta hoy, varias cosas cambiaron.

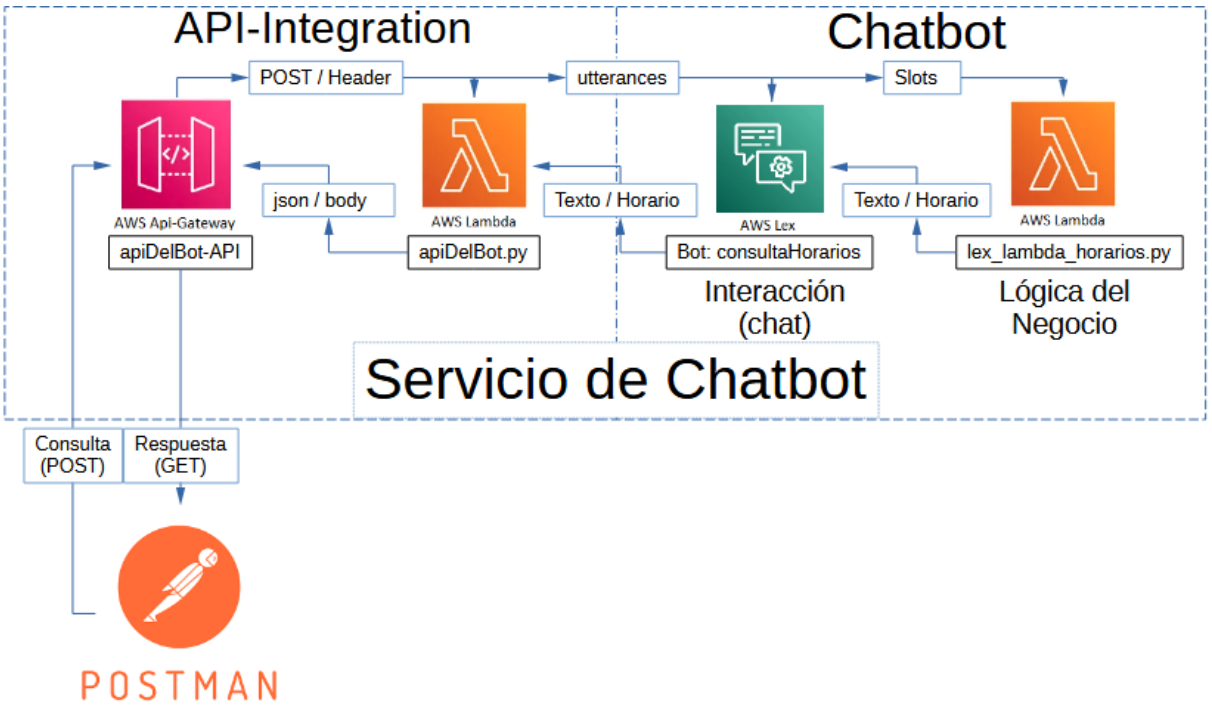


Figura 9. *Arquitectura del Servicio de Chatbot*

Postman[12] es una herramienta de desarrollo que permite diseñar, probar, automatizar y documentar APIs mediante una interfaz amigable para enviar solicitudes HTTP y analizar sus respuestas. Postman es un software que se describe como una plataforma API que se puede utilizar para crear API y utilizarlas. Postman puede utilizarse para probar dichas API de Postman y otras APIs (Renta, 2023). Con Postman, puede diagramarse y probarse la interacción con el chatbot. La Figura 10 muestra un ejemplo de una consulta hecha desde Postman al servicio del chatbot.

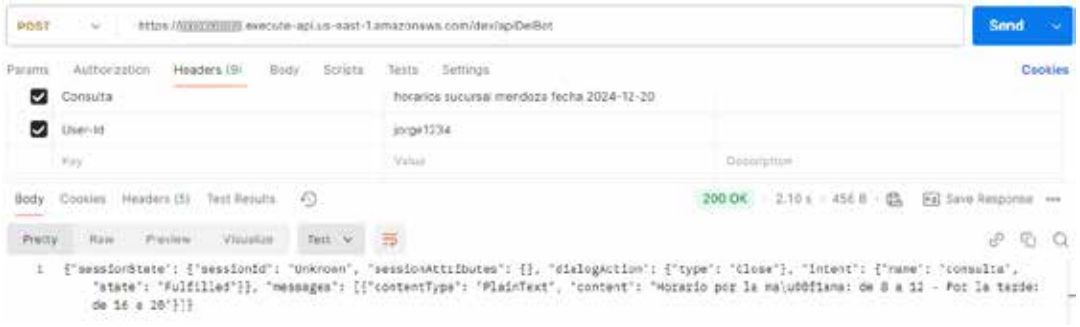


Figura 10. *Consulta de Postman al Servicio de Chatbot*

[12] Sitio oficial de Postman: https://www.postman.com/

En el ejemplo presentado en la Figura 10, Se pasa por header una consulta: “horarios sucursal mendoza fecha 2024-12-20” y se recibe la respuesta: “Horario por la mañana de 8 a 12 – Por la tarde: de 16 a 20”.

Sagemaker

Entrenar modelos de ML con grandes datasets en constante evolución es hoy un gran desafío para muchas empresas e instituciones. AWS Sagemaker es un modelo computacional que aborda este, y otros desafíos. SageMaker es una plataforma de ML escalable provista por Amazon. Sagemaker admite entrenamiento incremental, aprendizaje reanudable y elástico, optimización automática de hiperparámetros, entre otros (Liberty et al, 2020). Amazon Sagemaker es un servicio cloud para preparar, construir, entrenar y desplegar modelos de ML de alta calidad (Sayers, 2022).

Aspectos de Relevantes de Interés de Sagemaker

Instancias Notebooks: AWS Sagemaker Notebook es una instancia de AWS ML donde se puede compilar y compartir código, documentación y visualizar modelos. Con AWS Sagemaket Notebooks puede: Lanzar instancias de Jupyter Notebooks, preparar y procesar data, escribir código y entrenar modelos, desplegar y validar modelos.

Entrenamiento de Modelos en AWS Sagemaker: AWS Sagemaker puede usarse tanto para re-entrenar modelos pre-entrenados como para crear un algoritmo propio.

Algunos Algoritmos disponibles en Sagemaker: Puede usarse internamente: XGBoost, Object2Vec, K-Means, entre otros. Puede proveerse su propio algoritmo y también pueden utilizarse algoritmos disponibles en el AWS marketplace.

Configuraciones: Sagemaker puede presentarse en configuración estándar, en configuración optimizada para cómputo o bien, en configuraciones de instancias aceleradas (con disponibilidad de GPUs).

Hiper-parámetros: Se pasan hiper-parámetros para un control refinado sobre las configuraciones de algoritmos y modelos entrenados.

Canales de entrada de datos: Los datos pueden ingresarse vía S3 buckets o EFS para entrenamiento y validación (testing). Pueden ingresarse archivos csv para datasets tubulares o bien archivos png para procesamiento de imágenes.

AWS Marketplace: En el AWS Marketplace puede comprarse y venderse algoritmos, modelos y paquetes de ML para inferir o para entrenamiento.

Inferencias: Finalmente, en ML se desea crear modelos robustos para realizar inferencias a partir de un conjunto de datos de origen. Las inferencias pueden hacerse en tiempo real, como servicio serverless, o por lotes.

Ejemplo en la Consola de Sagemaker: Modelos del Marketplace

Se crea un modelo en AWS Sagemaker a partir de un modelo obtenido del AWS Marketplace. Se buscará crear un servicio de ML en tiempo real exponiendo un endpoint para inferencias y también, un servicio de ML para analizar por lotes. Se inicia accediendo a la consola Amazon Sagemaker AI.

Acceso al AWS Marketplace: En el menú de la izquierda, abajo, suele hallarse AWS Marketplace. A su vez, AWS Marketplace tiene los siguientes apartados: Model Packages, Algorithms, AWS Data Exchange y AI Products. En este caso, accedemos a Model Packages.

Ejemplo de Uso de un modelo pre-entrenado: Face and License Plate Anonymizer. Se elige a modo de ejemplo, el modelo llamado Face and License Plate Anonymizer, hecho por Navinfo Europe B.V.[13] versión 4,0. Este modelo tiene un "Free-Trial" de 5 días a partir de la suscripción.

Resumen de la aplicación presentado en el Marketplace: "El anonimizador de rostros y matrículas de NavInfo Europe detecta y difumina rostros y matrículas reconocibles en imágenes. El desenfoque de rostros y matrículas ayuda a alcanzar los estándares de privacidad globales. Los modelos se entrenan utilizando imágenes tomadas desde una cámara de tablero."

Una vez hecha la suscripción, se accede a la pantalla de configuración y lanzamiento. Aparecen las opciones para métodos de lanzamiento: AWS Cloud Formation, Sagemaker console, AWS CLI Command-Line Interface. En este caso, se elige: Sagemaker Console.

Las opciones de Sagemaker: Create a real-time inference endpoint y Create a batch transform job. En este ejemplo, se cargarán imágenes de autos con sus respectivas patentes, y se realizará su anonimización. Las imágenes se encuentran en una carpeta llamada cars dentro de un bucket en S3 del repositorio de este trabajo.

Creación del modelo: En Amazon Sagemaker AI hacer click en el botón amarillo Create Model. Allí completar: nombre del modelo, IAM role (SageMaker-ExecutionRole). En container definition seleccionar Use a model package subscription from AWS Marketplace. Aparecerán sus suscripciones. Elegir Face and License Plate Anonymizer. Luego: Create. Aparecerá el modelo creado en la lista.

Create a Job: Al hacer click en el modelo, puede crearse un Job haciendo click en Create batch transform job. Elegir Instant type e Instant count. Dar nombre al job. En Input data configuration, content type: image/jpeg. Ingresar la ubicación de los archivos de origen. En Output data configuration, ingresar path. Luego: Create job.

[13] Sitio oficial de Navinfo Europe: https://www.navinfo.eu/

La Tabla 8 muestra un ejemplo[14] de la ejecución de un batch transform job: A la izquierda se presentan imágenes de entrada del batch y a la izquierda, las imágenes de salida.

Entrenamiento y despliegue de modelos listos para usarse

AWS Sagemaker dispone de instancias Jupyter Notebooks mediante las cuales pueden ejecutarse tareas. En este caso, la Jupyter Notebook tratará los datos y con ellos entrenará el modelo y lo desplegará, listo para ser usado como servicio. Luego, vía url, o Postman, puede consultarse acerca de una necesidad de predicción o bien acerca de los indicadores de calidad del modelo desplegado. El ejemplo utiliza el dataset creditcard descargado desde Kaggle[15], que permite desplegar un servicio en línea de predicción de fraude con tarjetas de crédito entrenado con Random Forest.

Entrenamiento y despliegue del modelo desde Instancia Jupyter Notebook: En AWS Sagemaker, en el menú de la izquierda se encuentra Applications and IDEs. Seleccionar Notebooks. Click en Create notebook instance. Crea una nueva instancia de Notebook y esperar a que esté en estado InService. Allí puede abrirse la Jupyter Notebook.

TABLA 8. ANONIMIZACIÓN DE PATENTES HECHA POR NAVINFO

[14] Repositorio de imágenes de Navinfo: https://github.com/navinfoeurope/anonymizer/tree/main/data
[15] Link para descarga del Dataset: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

En este ejemplo, la Notebook contiene los códigos para: lectura de dataset, entrenamiento y despliegue del modelo. Las celda de entrenamiento y despliegue presentan como entry_point scripts de Python: train.py y predict.py. Dichos archivos deben estar disponibles en el ambiente Sagemaker de AWS. La Tabla 9 muestra un ejemplo de cómo podría construirse la Jupyter Notebook.

Luego ejecutar la celda de entrenamiento del modelo, puede verificar que el modelo se haya entrenado correctamente. Para ello, vaya a Training / Training Jobs y debería ver el trabajo enviado en su estado como Completado. También debería verificar que el modelo finalmente se haya creado. Para ello, vaya a Inference / Models y debería observar el modelo creado con su arn. Finalmente, luego de ejecutar la celda del despliegue del modelo, Ud. necesita asegurarse que el Endpoint se haya creado. Para ello, vaya a Inference / Endpoints y podrá verificar que el mismo se haya creado y se encuentre en estado inService.

Predicción y calidad del modelo: A esta altura, el modelo está entrenado y en servicio, disponible para realizar inferencias. Para ello, es necesario hacer pedidos por get y post a la url del endpoint recientemente creado.

TABLA 9. JUPYTER NOTEBOOK PARA ENTRENAMIENTO Y DESPLIEGUE DE MODELO

```
# Imports y Settings
import boto3
import sagemaker
import pandas as pd
from sagemaker import get_execution_role
from sagemaker.sklearn.estimator import SKLearn

# Configuración
s3_bucket = 'bucket.curso.ml'
s3_prefix = 'credit-card-fraud'
s3_input_data = 'input'
s3_filename = 'creditcard.csv'
role = get_execution_role()
boto3.Session().client('s3', region_name='us-east-2', endpoint_url='https://s3.amazonaws.com')

# Carga de datos de entrada
data_uri = f"s3://{s3_bucket}/{s3_prefix}/{s3_input_data}/{s3_filename}"
# Cargar dataset localmente para preprocesar
df = pd.read_csv(data_uri)
# Verificar datos
print(df.shape)

# Entrenamiento
# Configurar el estimador de scikit-learn
sklearn_estimator = SKLearn(entry_point='creditcard_train.py',role=role, instance_type='ml.m4.xlarge',
    framework_version='0.23-1', py_version='py3', hyperparameters={'max_leaf_nodes': 30})
# Iniciar el entrenamiento
sklearn_estimator.fit({'train': data_uri})

#.Despliegue.del.modelo
from sagemaker.sklearn.model import SKLearnModel
#.Crear.el.modelo
model = SKLearnModel(model_data=sklearn_estimator.model_data, role=role,
    entry_point='creditcard_predict.py')
#.Desplegar.el.modelo
predictor = model.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

Predicción y calidad del modelo: A esta altura, el modelo está entrenado y en servicio, disponible para realizar inferencias. Para ello, es necesario hacer pedidos por get y post a la url del endpoint recientemente creado.

Prueba del Modelo Dentro de AWS: Una primera prueba puede realizarse aún dentro del ambiente AWS. Para ello acceda a la Jupyter Notebook en Sagemaker. Como el modelo ya está desplegado, no es necesario volver a ejecutar las celdas anteriores. Se agrega una celda que contiene los datos de la conexión y el pedido.

Configuraciones para Pedido Utilizando Postman:

- Armado de la URL: La URL a la cual se invocará para acceder al modelo para requerir una predicción se obtiene con: https://runtime.sagemaker + <region> + .amazonaws.com / endpoints / + <Endpoint-Name> + / invocations
- Método: Post
- Headers: Content-Type: application/json, X-Amz-Target: “SageMaker.InvokeEndpoint”
- Body: Dato tipo Raw y Json. Debajo colocar un json con el pedido en body (payload).

La figura 10 muestra el pedido, y la respuesta, utilizando Postman. Recuadrado en rojo, el resultado de la predicción.

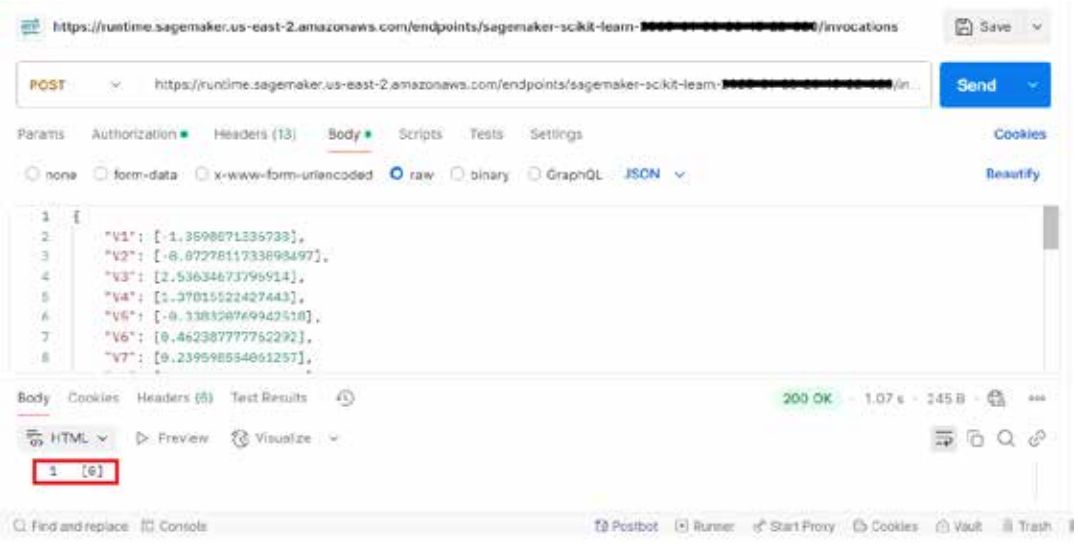


Figura 10. *Uso de Postman para probar y obtener predicciones*

Conclusiones

El uso de las plataformas cloud para el desarrollo y despliegue de aplicaciones de ML resulta conveniente. En particular, el ambiente que ofrece AWS contiene herramientas que simplifican estas tareas.

En este trabajo se presentaron diferentes enfoques para implementar los servicios de AWS ML más destacables: su uso directamente en consola, o bien, mediante código Python, para ser usado fuera de la consola, dentro de su propio ambiente. Para ambos enfoques, y para cada servicio de ML se presentaron ejemplos que pueden ser reproducidos para así, facilitar su entendimiento y uso por parte de diferentes tipos de usuarios: usuarios esporádicos que solo necesitan usar eventualmente estos servicios (la consola), o desarrolladores de aplicaciones y soluciones basadas en AWS ML

Este trabajo, muestra ser en consecuencia, un práctico y efectivo instructivo para el uso, desarrollo, implementación y/o despliegue de servicios de ML confiables.

Referencias

Borra, Praveen. (2022). “Exploring Microsoft Azure’s Cloud Computing: A Comprehensive Assessment. International Journal of Advanced Research”. In Science, Communication and Technology (IJARSCT) Volume, 2.

Borra, Praveen. (2024). “A Survey of Google Cloud Platform (GCP): Features, Services, and Applications”. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume, 4.

Louis Dorard, Mark D. Reid and Francisco J. Martin (Azure Machine Learning Team), (2016). AzureML: Anatomy of a machine learning service. In Conference on Predictive APIs and Apps (pp. 1-13). PMLR.

Fauss, M., Hao, J., Li, C., Palmer, M., & Choi, I. (2024). AutoSSD: A system for automated detection of similar speech responses in language tests.

Indla, R. K. (2021). An overview on amazon rekognition technology.

Kamlofsky, J. A. (2022). Computación en la Nube: Fundamentos, Críticas y Desafíos. Revista Abierta de Informática Aplicada, 6(2), 3-30.

Kamlofsky, J. A. (2024). Una Reseña Acerca de Servicios de Machine Learning en ambientes Cloud. Researchgate.net

Kodali, R. K., Shekhar, T., & Boppana, L. (2023). Automated Plagiarism Detection in Moodle. In TENCON 2023-2023 IEEE Region 10 Conference (TENCON) (pp. 176-181). IEEE.

Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., ... & Smola, A. (2020, June). Elastic machine learning algorithms in amazon sagemaker. In Proceedings of the 2020 ACM

SIGMOD International Conference on Management of Data (pp. 731-737).

Mell, P. (2011). The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology.

Movinuddin. (2023). HEALTHCARE TEXT ANALYTICS USING RECENT ML TECHNIQUES AND DATA CLASSIFICATION USING AWS CLOUD ML SERVICES (Doctoral dissertation, Middle Tennessee State University).

Oncu, M. H., & Kutlu, Y. (2024). Performance Comparison of Known AI Translation Tools. Tethys Env. Sci, 1(3), 117-126.

Poccia, D. (2016). AWS Lambda in Action: Event-driven serverless applications. Simon and Schuster.

Ranta, J. (2023). Testing AWS hosted Restful APIs with Postman.

Rivas, E. R. H. (2024). IMPLEMENTACIÓN DE APLICACIÓN WEB DE GESTIÓN DE COMERCIO EN AMAZON WEB SERVICES (AWS) PARA MULTISERVICIOS SANTA ELENA. Universidad de San Carlos de Guatemala.

Sarabia Jácome, D. F. (2020). Arquitectura de análisis de datos generados por el internet de las cosas IoT en tiempo real (Doctoral dissertation, Universitat Politècnica de València).

Sayers, Russell (2022). "Introducción al aprendizaje automático en AWS", AWS. Curso en línea: *https://www.coursera.org/learn/machine-learning-on-aws*

Simon Julien (2020). "Building, training and deploying machine learning models with Amazon SageMaker", Youtube. En línea: *https://www.youtube.com/watch?v=sOUhLiI85sU*

Sreeharsha, A., Kesapragada, S. M., & Chalamalasetty, S. P. (2022). Building chatbot using amazon lex and integrating with a chat application. Interantional journal of scientific research in engineering and management, 6(04), 1-6.

Wang, H., Yang, J., Liang, G., Lee, Y., & Cao, Z. (2024, August). Analyzing the Usability, Performance, and Cost-Efficiency of Deploying ML Models on BigQuery ML and Vertex AI in Google Cloud. In Proceedings of the 2024 8th International Conference on Cloud and Big Data Computing (pp. 15-25).