

TRABAJO DE FIN DE MÁSTER

MÁSTER EN BIG DATA & DATA ENGINEERING

2023/2024

TEMA: PIPELINE END-TO-END



MIGUEL ÁNGEL GONZÁLEZ-ALBO CAMPILLO

70578744-R

ÍNDICE

1.-Introducción.....	3
1.1.- Objetivos del proyecto	3
1.1.1.-Objetivos funcionales:	3
1.1.2.-Objetivos técnicos:	3
1.2.- Tipos de datos tratados.....	3
1.3.- Explotación de los datos	3
2.-Análisis de Requisitos	4
2.1.-Identificación de datos y fuentes de datos.....	4
2.2.-Selección de tecnologías y herramientas	5
2.2.1.-Batch vs Streaming	5
3.-Diseño del Pipeline	5
3.1.-Arquitectura del sistema.....	5
3.1.1.-Ingesta de datos	6
3.1.2.-Transformación y limpieza de datos a la capa silver	7
3.1.3.-Trigger.....	8
4.-Machine Learning	9
5.-Power BI	11
6.-Escalabilidad y Rendimiento	12
7.-Futuras mejoras.....	13
8.-Monitorización.....	13
9.-Resultados y Conclusión	14
9.1.-Análisis de Resultados	14
9.2.-Conclusiones del Proyecto	14
9.3.-Glosario de Términos.....	15

1.-Introducción

Este Trabajo de Fin de Máster tiene como objetivo desarrollar un pipeline de datos end-to-end utilizando Azure Synapse, que es capaz de procesar grandes cantidades de datos provenientes de diversas fuentes. El propósito principal es demostrar cómo una arquitectura en la nube permite la ingesta, transformación, almacenamiento y explotación de los datos, incluyendo la creación de un modelo de Machine Learning para predecir tendencias en el inventario de productos y generar visualizaciones a través de Power BI, muy útil para la toma de decisiones de negocio.

1.1.- Objetivos del proyecto

1.1.1.-Objetivos funcionales:

- Automatizar el proceso de datos: Diseñar e implementar un pipeline automatizado que permita la ingesta, transformación y explotación de datos de ventas online, inventarios y soporte técnico.
- Predecir: Utilizar técnicas de Machine Learning para predecir niveles de inventario, con el fin de anticipar niveles de stock futuros.
- Visualización de datos: Mediante un dashboard interactivo en Power BI que permita a los usuarios analizar tendencias de ventas, niveles de inventario y rendimiento de soporte técnico.

1.1.2.-Objetivos técnicos:

- Integrar datos de diferentes fuentes, como Azure Cosmos DB y Azure Data Lake Gen2, en un pipeline automatizado, y realizar su procesamiento en capas (bronze, silver, gold).
- Implementar un modelo de Machine Learning con redes neuronales utilizando Keras/TensorFlow para predecir niveles de stock.
- Asegurar la escalabilidad y el rendimiento del pipeline con Azure Synapse, monitoreado mediante Azure Logs Workspace.

1.2.- Tipos de datos tratados

Los datos procesados en este proyecto provienen de diversas fuentes y formatos:

- Ventas online: Datos relacionados con las ventas de productos en una tienda online.
- Inventarios: Información sobre el stock disponible.
- Soporte técnico: Datos de soporte recibidos de los clientes.

Los formatos de los datos incluyen:

- JSON: Para los datos no estructurados de inventario y soporte técnico.
- CSV: Para los datos de ventas.

1.3.- Explotación de los datos

La explotación de los datos será realizada de las siguientes maneras:

- Modelo de Machine Learning: Mediante un modelo basado en redes neuronales para predecir si el stock de productos excederá un umbral determinado (500 unidades).
- Power BI: Los datos agregados en la capa Gold serán utilizados para crear dashboards en Power BI.

2.-Análisis de Requisitos

Entre los requisitos para la creación del pipeline se incluye la capacidad de procesar datos desde diferentes fuentes, la transformación de estos en múltiples capas (bronze, silver, gold), y el desarrollo de modelos predictivos.

2.1.-Identificación de datos y fuentes de datos

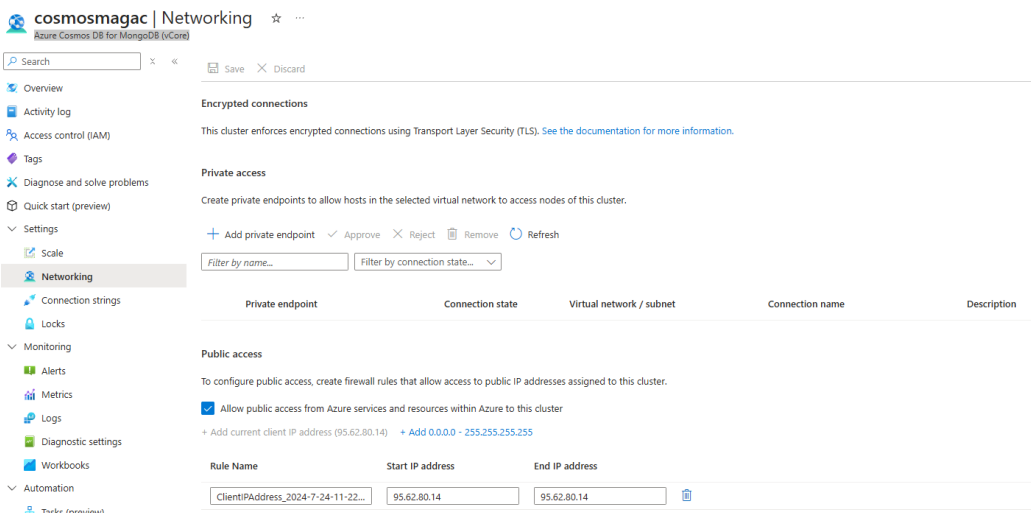
Las fuentes de datos son 2, y los archivos utilizados son 3. Los archivos los he generado yo, y están relacionados con las ventas de una tienda online, sus inventarios y el soporte técnico.

El volumen de los datos se encuentra entre 10.000 y 250.000 filas.

1. Azure Cosmos DB for MongoDB: Base de datos NoSQL utilizada para almacenar datos no estructurados. En este caso he usado 2 archivos .json:
 - a. InventarioProductos.json

Para cargar este archivo, he creado una tabla InventarioProductos.

He tenido que cambiar la variable de entorno Path para poder leer el archivo, así como cambiar las reglas en ComosDB para que me permita conectar con mi IP:



The screenshot shows the 'Networking' tab in the Azure Cosmos DB for MongoDB (vCore) portal. The 'Public access' section is expanded, showing a table of IP addresses and a rule named 'ClientIPAddress_2024-7-24-11-22...'. The table has columns for Rule Name, Start IP address, and End IP address. The rule is configured to allow public access from the client IP address 95.62.80.14 to the end IP address 95.62.80.14.

Rule Name	Start IP address	End IP address
ClientIPAddress_2024-7-24-11-22...	95.62.80.14	95.62.80.14

Usando MongoDB Command Line Database Tools Download (<https://www.mongodb.com/try/download/database-tools>) y el connection string proporcionado por Cosmos, he importado el .json

```
PS C:\Users\Miguel> mongoimport --uri "mongodb+srv://[redacted]@cosmosmagac.mongodb.cluster.cosmos.azure.com/?tls=true&authMechanism=SCRAM-SHA-256&retrywrites=false&maxIdleTimeMS=120000" --collection InventarioProductos --file "C:\Users\Miguel\Desktop\MASTER INGENIERIA DE DATOS\TFM\Sources\InventarioProductos.json"
2024-07-24T11:25:44.040+0200 connected to: mongodb+srv://[redacted]@cosmosmagac.mongodb.cluster.cosmos.azure.com/?tls=true&authMechanism=SCRAM-SHA-256&retrywrites=false&maxIdleTimeMS=120000
2024-07-24T11:25:46.158+0200 1000 document(s) imported successfully. 0 document(s) failed to import.
```

Desde MongoDB Compass uso también este string para comprobar que los datos son correctos

b. SoporteTecnico.json

En este caso los pasos son similares, lo único que cambia es el comando ya que tiene otro formato, por lo que añado -jsonArray

```
PS C:\Users\Miguel> mongoimport --uri "mongodb+srv://[REDACTED]@cosmosmagac.mongodb.cluster.cosmos.azure.com/?tls=true&authMechanism=SCRAM-SHA-256&retryWrites=false&maxIdleTimeMS=120000" --collection SoporteTecnico --file "C:\Users\Miguel\Desktop\MASTER INGENIERIA DE DATOS\TFM\Sources\SoporteTecnico.json" --jsonArray
connected to: mongodb+srv://[REDACTED]@cosmosmagac.mongodb.cluster.cosmos.azure.com/?tls=true&authMechanism=SCRAM-SHA-256&retryWrites=false&maxIdleTimeMS=120000
2024-07-24T13:20:51.386+0200 [.....] test.SoporteTecnico 817KB/39.0MB (2.0%)
2024-07-24T13:20:54.398+0200 [.....] test.SoporteTecnico 2.74MB/39.0MB (7.0%)
2024-07-24T13:21:00.396+0200 [#####] test.SoporteTecnico 5.86MB/39.0MB (15.0%)
2024-07-24T13:21:03.389+0200 [#####] test.SoporteTecnico 8.59MB/39.0MB (22.0%)
2024-07-24T13:21:06.395+0200 [#####] test.SoporteTecnico 11.7MB/39.0MB (30.0%)
2024-07-24T13:21:09.394+0200 [#####] test.SoporteTecnico 14.4MB/39.0MB (37.0%)
2024-07-24T13:21:12.392+0200 [#####] test.SoporteTecnico 17.6MB/39.0MB (45.0%)
2024-07-24T13:21:15.387+0200 [#####] test.SoporteTecnico 20.3MB/39.0MB (52.0%)
2024-07-24T13:21:18.398+0200 [#####] test.SoporteTecnico 23.4MB/39.0MB (60.0%)
2024-07-24T13:21:21.394+0200 [#####] test.SoporteTecnico 26.1MB/39.0MB (67.0%)
2024-07-24T13:21:24.394+0200 [#####] test.SoporteTecnico 28.9MB/39.0MB (74.0%)
2024-07-24T13:21:27.394+0200 [#####] test.SoporteTecnico 32.0MB/39.0MB (82.1%)
2024-07-24T13:21:30.390+0200 [#####] test.SoporteTecnico 34.7MB/39.0MB (89.1%)
2024-07-24T13:21:33.390+0200 [#####] test.SoporteTecnico 37.5MB/39.0MB (96.1%)
2024-07-24T13:21:35.139+0200 [#####] test.SoporteTecnico 39.0MB/39.0MB (100.0%)
100000 document(s) imported successfully. 0 document(s) failed to import.
```

2. Azure Data Lake Gen2: En este caso, he importado los datos desde otro Datalake.

El archivo es:

a. VentasTiendaOnline.txt

Los datos son transferidos a la capa bronze para su procesamiento posterior.

2.2.-Selección de tecnologías y herramientas

He elegido para realizar mi trabajo:

- Azure Synapse: por su capacidad de integrarse fácilmente con otras herramientas de Azure. Aquí he realizado el pipeline.
- Azure Cosmos DB: que servirá como fuentes de datos no estructurados.
- Data Lake Gen2: como almacenamiento en la nube.
- Azure ML: para productivizar el modelo de machine learning.
- Logs Analytics Workspace: para analizar los logs del pipeline.
- Power BI: para realizar las visualizaciones.

2.2.1.-Batch vs Streaming

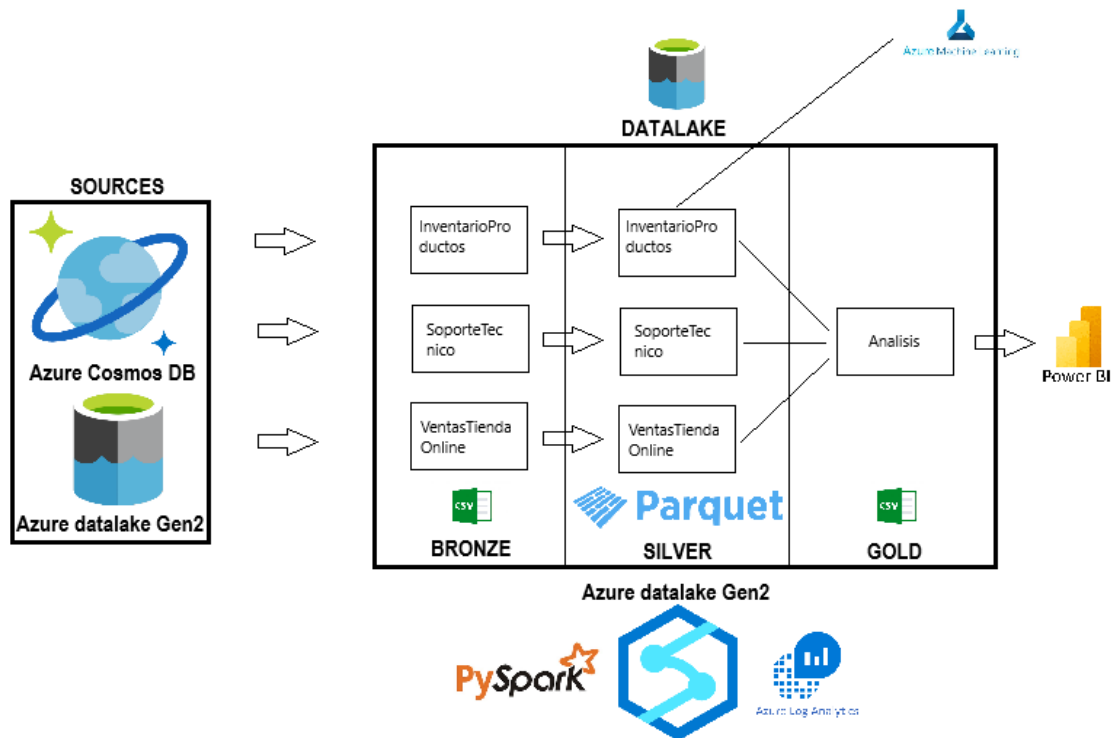
Dado que los datos se procesan en intervalos periódicos, he optado por un enfoque de procesamiento por lotes (batch). He establecido un trigger que ejecuta el pipeline cada 5 horas.

3.-Diseño del Pipeline

3.1.-Arquitectura del sistema

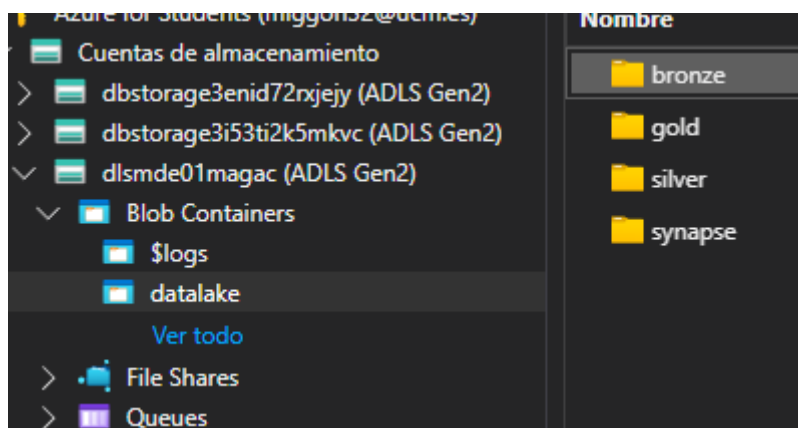
El sistema está compuesto por un pipeline que incluye múltiples etapas, desde la ingesta hasta la agregación y almacenamiento final.

La arquitectura general es así:

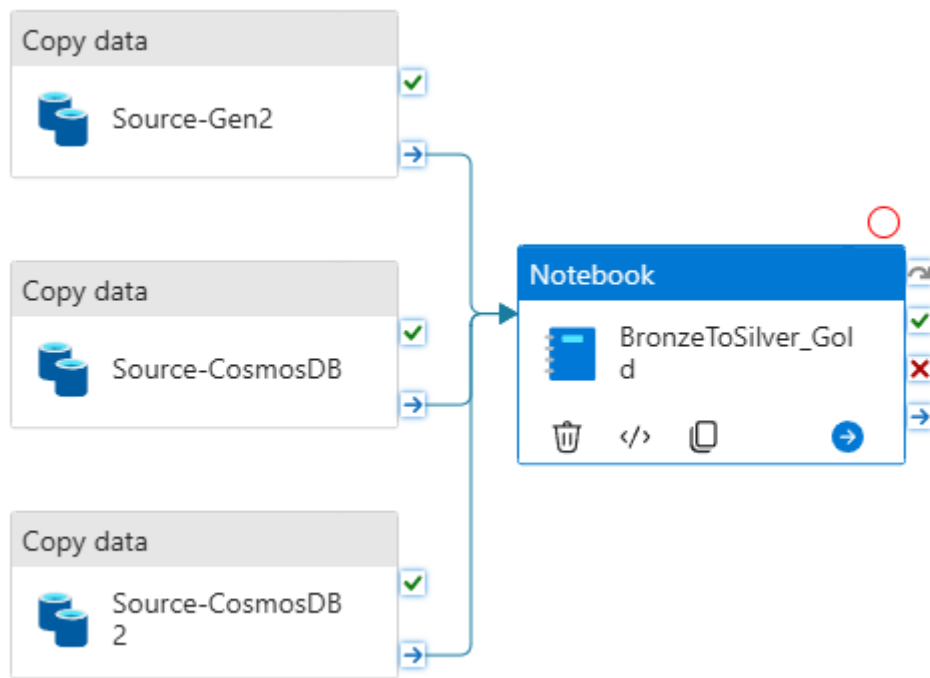


3.1.1.-Ingesta de datos

En primer lugar, he creado una cuenta de almacenamiento para mi datalake, en la que he creado las 3 carpetas de bronze, silver y gold:



Para la ingesta de datos he utilizado Azure Synapse. Para ello, he generado 3 módulos de copiar datos, dos de ellos desde Cosmos, y el otro desde un datalake Gen2 que tengo en otro grupo de recursos. El pipeline completo es el siguiente:



Estos datos, serán copiados de forma raw en la capa bronze de mi datalake, siendo la única transformación el de pasar los .json a .csv, para que todos los datos estén en el mismo formato:

← → ∨ ↑ Blobs activos (predeterminado) datalake > bronze	
Nombre	Nivel de acceso
InventarioProductos.csv	Nivel de acceso frecuente (inferido)
SoporteTecnico.csv	Nivel de acceso frecuente (inferido)
Ventas.csv	Nivel de acceso frecuente (inferido)

3.1.2.-Transformación y limpieza de datos a la capa silver

He usado un script en un notebook dentro de Synapse con Pyspark (el notebook se llama BronzeToSilver_Gold).

Nota: En un principio, mi idea era generar 2 notebooks, un para silver, y otro para gold. Sin embargo, las limitaciones en los vCores de la cuenta de almacenamiento de estudiante no me permitían crear 2 Spark Pools, por lo que solo puedo ejecutar un único notebook. Es por ello por lo que he unificado tanto el paso de bronze a silver como el de silver a gold en un único notebook. Sin embargo, se puede ver que las partes están diferenciadas.

Este script sirve para procesar:

1. Datos de bronze a silver: los va a almacenar en la carpeta silver en formato delta. Carga los archivos, elimina duplicados y los guarda.

← → ∨ ↑ Blobs activos (predeterminado) ▼ datalake > silver > SoporteTecnico	
Nombre	Nivel de acceso
📁 _delta_log	
📄 part-00000-7e4e03d0-cb5b-4c2a-b64c-0ec6844c012b-c000.snappy.parquet	Nivel de acceso frecuente
📄 part-00000-8fd749c7-4a08-4af4-b475-48a1af84e9ba-c000.snappy.parquet	Nivel de acceso frecuente
📄 part-00000-180fdc18-0ba0-4fbf-b84b-36f8bcefc32e-c000.snappy.parquet	Nivel de acceso frecuente
📄 part-00000-3907bb22-bf24-448f-8abf-169b447bb3a9-c000.snappy.parquet	Nivel de acceso frecuente
📄 part-00000-04018cfe-504f-4486-ae7e-4c893c897933-c000.snappy.parquet	Nivel de acceso frecuente

2. Datos de silver a gold: lee las tablas de silver, realiza unos joins entre las tres tablas y hace agregaciones. Por último, se ordena y se guarda en la capa gold en un .csv.

← → ∨ ↑ Blobs activos (predeterminado) ▼ datalake > gold > Analisis.csv	
Nombre	Nivel de acceso
📄 _SUCCESS	Nivel de acceso frecuente
📄 part-00000-7120fcff-41ba-4e66-b97a-5587dddb281b-c000.csv	Nivel de acceso frecuente

3.1.3.-Trigger

El pipeline está orquestado mediante un trigger que ejecuta las ingestas y transformaciones cada 5 horas, asegurando que los datos siempre estén actualizados y listos para su análisis en Power BI o la ejecución del modelo de machine learning.

Edit trigger

Name *
Trigger-1

Description

Type *
ScheduleTrigger

Start date * ⓘ
9/16/2024, 10:06:00 AM

Time zone * ⓘ
Brussels, Copenhagen, Madrid, Paris (UTC+2)

ⓘ This time zone observes daylight savings. Trigger will auto-adju:

Recurrence * ⓘ
Every Hour(s)

☒ Specify an end date

End On * ⓘ
9/20/2024, 8:01:00 AM

Annotations
[+ New](#)

Status ⓘ
☒ Started ☐ Stopped

4.-Machine Learning

Para la parte de Machine Learning, he decidido usar una red neuronal para predecir datos desde la carpeta de InventarioProductos de la capa silver. El modelo ha sido entrenado en MLFlow en local. Tiene métricas de validación como la precisión y la curva ROC.

**Nota: la cuenta de estudiante de Azure requería vCores adicionales para usar Azure ML, por lo que he optado por hacerlo en local*

El notebook lo he llamado “TFM-ML1”.

En este script se hace lo siguiente:

1. Conectar a mi cuenta de Azure Data Lake, utilizando la cadena de conexión, concretamente nos conectamos a la carpeta que necesitamos y usamos el archivo más reciente.
2. Descargar el archivo en formato Parquet y lo cargamos en un DataFrame de Pandas.
3. Preparar los datos para el modelo:
 - a. Convertir columnas a otro formato.
 - b. Seleccionar características.
 - c. Definir columna objetivo.
 - d. Dividimos en train y test.
 - e. Escalamos los datos usando StandardScaler.

4. Entrenar el modelo de red neuronal con TensorFlow/Keras.
 - a. Crear la red neuronal básica, con capas Dense, activaciones relu y sigmoid.
 - b. Utilizar la función de pérdida `binary_crossentropy` para clasificación binaria y optimizador Adam.
 - c. Entrenar con los datos de train.
5. MLFlow
 - a. Configurar MLFlow para registrar los modelos en el directorio local.
6. Crear API REST con Flask
 - a. Crear la ruta `/predict` para hacer las predicciones, que serán el endpoint que reciba los datos en Json, los convierte en un dataframe y realiza predicciones que devuelve.
7. En otro notebook ("TFM-ML2"), hacer una llamada de prueba.

```
[15]: import requests
import pandas as pd

# Crear un ejemplo de datos para enviar
data = {
    'Stock': [400, 600],
    'Precio_Costo': [10.5, 12.0],
    'Precio_Venta': [15.0, 18.0]
}

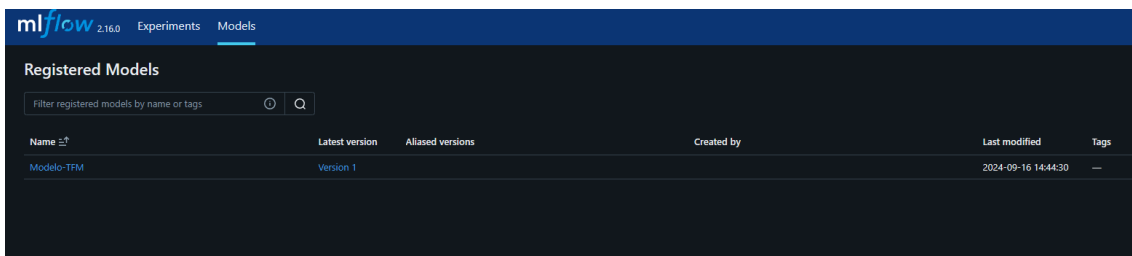
# Hacer la solicitud POST al servidor Flask
response = requests.post("http://localhost:5003/predict", json=data)

# Mostrar la respuesta
print(response.json())

{'predictions': [[1.0], [1.0]]}
```

Ahora conviene iniciar la interfaz web de MLFlow, mediante `mlflow ui --backend-store-uri file:///mlruns --port 5005`

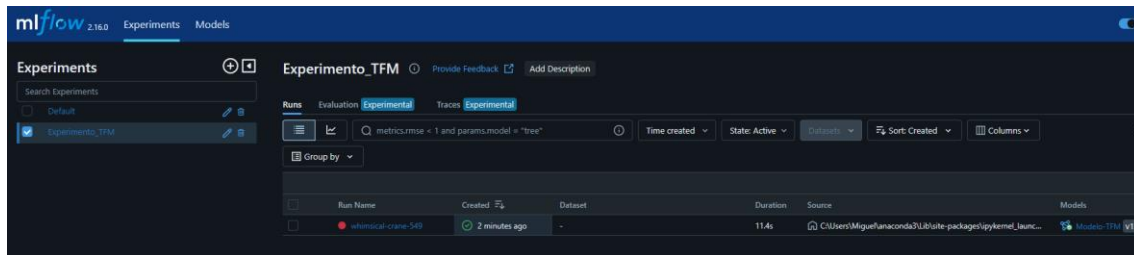
El modelo se ha registrado en MLFlow:



The screenshot shows the MLFlow web interface with the 'Models' tab selected. Below the header, there is a search bar for registered models. A table lists the registered models with columns for Name, Latest version, Aliased versions, Created by, Last modified, and Tags.

Name	Latest version	Aliased versions	Created by	Last modified	Tags
Modelo-TFM	Version 1			2024-09-16 14:44:30	—

Y los experimentos:



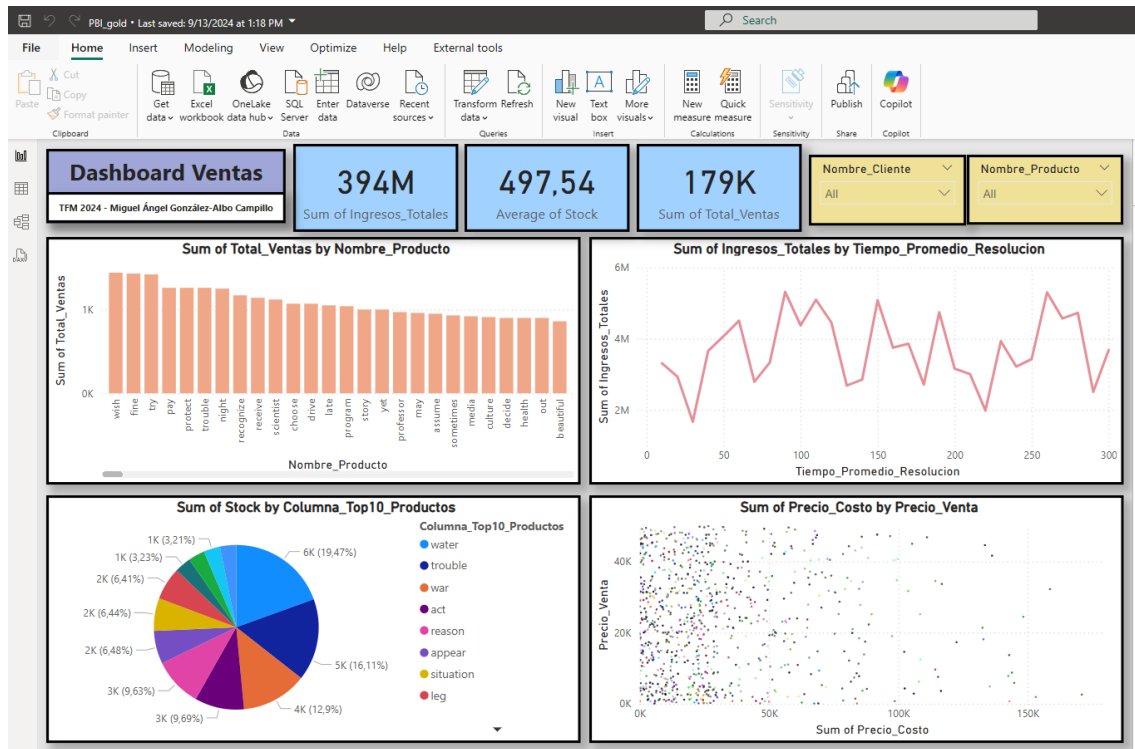
Respecto al model de Machine Learning, para interpretar los resultados:

- Features:
 - Stock
 - Coste del producto
 - Precio de venta
- Es un problema de clasificación binaria donde predecimos si el producto tendrá un alto nivel de stock (que pongo en el umbral de 500 unidades). No estamos prediciendo la cantidad exacta del stock, estamos prediciendo:
 - > 500: será un 1.
 - <=500: será un 0.

5.-Power BI

Los datos procesados en la capa gold se conectan a Power BI para la creación de un dashboard interactivo, permitiendo la visualización en tiempo real de las métricas.

Este Power BI (llamado PBI_gold) se ha conectado a Azure mediante la cadena de conexión, directamente recogiendo el último archivo que se encuentre en la carpeta gold.



Este Power BI podrá subirse a la nube con una cuenta profesional, y hacer que se actualice periódicamente.

6.-Escalabilidad y Rendimiento

El pipeline desarrollado para ser escalable y manejar grandes volúmenes de datos, tanto estructurados como no estructurados, provenientes de múltiples fuentes. La escalabilidad se produce al usar tecnologías como Synapse o Data Lake Gen2, que permiten el procesamiento paralelo, lo que lleva a optimizar el rendimiento según las necesidades del momento.

Optimizaciones que impactan en el rendimiento:

- Datalake con particiones: datos en formato Delta en capa Silver, lo que facilita operaciones más rápidas mediante la paralelización y la optimización en las consultas.
- Transformaciones eficaces: Las transformaciones de datos han sido diseñadas para minimizar el uso de recursos, realizando filtrados y agregaciones en una sola operación cuando es posible.
- Triggers eficientes: El trigger cada 5 horas hace que el pipeline no esté sobrecargado con ingestas continuas.

7.-Futuras mejoras

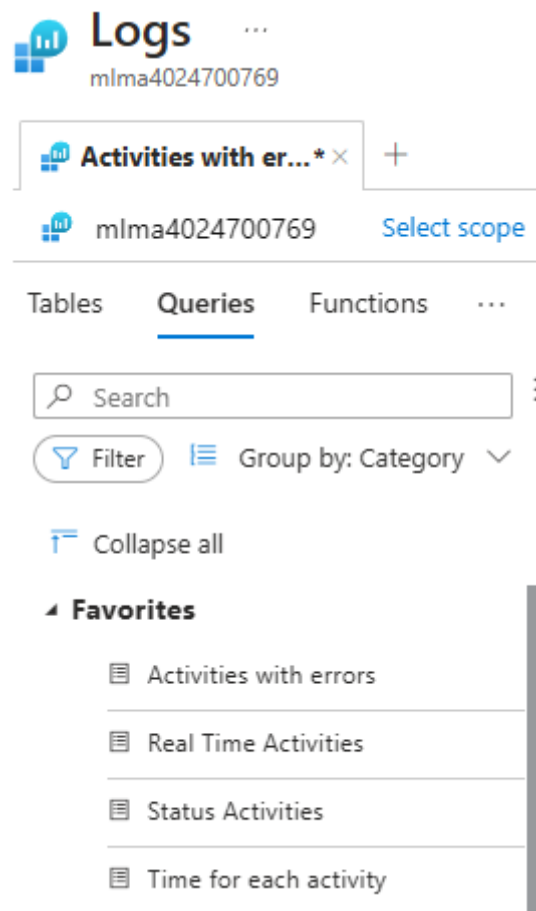
Como mejoras futuras, se podría escalar el sistema mediante:

- Añadir más Spark Pools: considerando que es una cuenta de estudiante y sus limitaciones, se podría ampliar el uso de Spark Pools para dividir el procesamiento de datos entre diferentes etapas, mejorando la eficiencia.
- Autoscaling en Azure ML: se podría agregar Azure ML para ajustar automáticamente las necesidades del modelo según el tamaño de los datos y la complejidad del propio modelo.
- Optimizar el modelo de ML.
- Incorporación de datos en streaming.

8.-Monitorización

El pipeline cuenta con una monitorización continua mediante Azure Logs Workspace, lo que permite identificar problemas en tiempo real y aplicar soluciones. Para ello, he creado algunas queries que permiten ver en tiempo real, el estado del pipeline. Como futuros pasos, se podrían crear alertas.

Queries:



Ejemplos de resultados de una de las queries:

The screenshot shows the Microsoft Azure Logs interface for the resource group 'rg-practicacloud-magac' and the log 'Logs-TFM-MA'. A query is running, filtering for 'SynapseIntegrationActivityRuns' where 'TimeGenerated' is greater than 'ago(1h)'. The results table shows various pipeline activities with their names, pipeline names, statuses, and timestamps.

ActivityName	PipelineName	Status	TimeGenerated [UTC]
BronzeToSilver_Gold	PipelineTFM	Succeeded	9/18/2024, 9:27:07.020 AM
BronzeToSilver_Gold	PipelineTFM	InProgress	9/18/2024, 9:21:30.649 AM
BronzeToSilver_Gold	PipelineTFM	Queued	9/18/2024, 9:21:29.542 AM
Source-CosmosDB2	PipelineTFM	Succeeded	9/18/2024, 9:21:28.588 AM
Source-Gen2	PipelineTFM	Succeeded	9/18/2024, 9:19:01.221 AM
Source-Gen2	PipelineTFM	InProgress	9/18/2024, 9:18:56.747 AM
Source-CosmosDB	PipelineTFM	Succeeded	9/18/2024, 9:18:45.875 AM
Source-CosmosDB2	PipelineTFM	InProgress	9/18/2024, 9:18:35.460 AM
Source-CosmosDB	PipelineTFM	InProgress	9/18/2024, 9:18:32.951 AM
Source-CosmosDB	PipelineTFM	Queued	9/18/2024, 9:18:23.888 AM
Source-CosmosDB2	PipelineTFM	Queued	9/18/2024, 9:18:23.879 AM
Source-Gen2	PipelineTFM	Queued	9/18/2024, 9:18:23.849 AM

9.-Resultados y Conclusión

9.1.-Análisis de Resultados

El pipeline implementado es capaz de procesar y transformar grandes volúmenes de datos con éxito, desde la ingesta hasta la visualización en Power BI.

Además, las métricas de rendimiento del modelo de Machine Learning tienen un alto porcentaje de accuracy y curva ROC, lo que significa que tiene una alta capacidad predictiva con los datos provistos.

9.2.-Conclusiones del Proyecto

En este trabajo he logrado implementar un pipeline de datos robusto, automatizado y escalable en Azure. La arquitectura puede procesar datos en diferentes capas del datalake, desde la ingesta hasta el análisis mediante machine learning. La integración con Power BI permite visualizar de una forma efectiva los resultados.

9.3.-Glosario de Términos

- Azure Synapse: Plataforma de análisis en la nube que combina integración de datos, big data y almacenamiento de datos empresariales.
- Azure Data Lake Gen2: Servicio de almacenamiento en la nube optimizado para el análisis de grandes volúmenes de datos.
- MLFlow: Herramienta de código abierto para gestionar el ciclo de vida completo del machine learning, incluyendo experimentos, reproducibilidad y despliegue de modelos.
- Power BI: Herramienta de visualización interactiva de datos que permite crear dashboards y reportes.
- Flask: Framework de microservicios para Python que permite crear aplicaciones web de manera sencilla.