

Universidad Panamericana
Maestría en Ciencia de Datos
Datos Masivos

Proyecto Final: *Pipeline Distribuido de Predicción para Iowa Liquor Sales en GCP*

Enrique Ulises Báez Gómez Tagle, Luis Alejandro Guillén Alvarez

7 de diciembre de 2025

Índice

1 Dataset utilizado	3
1.1 Fuente y descripción	3
1.2 Cardinalidades y dimensiones	3
1.3 Calidad de los datos	3
1.4 Distribución de ventas	4
1.5 Top 10 categorías por volumen de ventas	4
1.6 Justificación de selección	4
2 Descripción de la arquitectura implementada	5
2.1 Diagrama de arquitectura	5
2.2 Flujo de datos	5
2.3 Componentes de la arquitectura	6
3 Desarrollo de la ruta elegida: Procesamiento Distribuido con PySpark	6
3.1 Selección y exportación del dataset hacia GCS	6
3.1.1 Proceso de exportación	6
3.1.2 Configuración de Cloud Run	7
3.1.3 Métricas de tiempo de ejecución	7
3.1.4 Verificación de estructura y consistencia	8
3.2 Procesamiento distribuido en Dataproc	9
3.2.1 Configuración de clusters	9
3.2.2 Lectura del dataset desde GCS mediante PySpark	10
3.2.3 Aplicación de limpieza, filtrado y transformación	11
3.2.4 Monitoreo de ejecución	12
3.3 Modelado predictivo en PySpark	14
3.3.1 Modelo seleccionado	14
3.3.2 Entrenamiento del modelo	14
3.3.3 Métricas de evaluación	14
3.4 Evaluación comparativa entre configuraciones de cluster	15
3.4.1 Comparación en Data Engineering (ETL y Transformación)	15
3.4.2 Comparación en Data Science (Modelado ML)	17

4	Métricas, gráficas y análisis de resultados	22
4.1	Interpretación de resultados	22
4.2	Justificación del muestreo	22
4.3	Evaluación del desempeño del modelo	23
5	Análisis crítico del enfoque	23
5.1	Ventajas del enfoque elegido	23
5.2	Limitaciones del enfoque elegido	24
6	Conclusiones	24
7	Código utilizado	24
7.1	Repositorio de código fuente	24
8	Referencias	24

1. Dataset utilizado

1.1. Fuente y descripción

El dataset utilizado proviene de **BigQuery Public Data** y contiene registros de ventas de licores en el estado de Iowa, Estados Unidos. Este conjunto de datos es mantenido por el Iowa Department of Commerce y está disponible públicamente para análisis.

- **Fuente:** BigQuery Public Data - `bigrquery-public-data.iowa_liquor_sales.sales`
- **Tamaño:** 32,816,143 registros
- **Periodo:** 2012-01-03 a 2025-10-31 (13.8 años)
- **Características principales:**
 - `date`: Fecha de la transacción
 - `store_number`: Identificador de la tienda
 - `city`: Ciudad donde se realizó la venta
 - `category`: Categoría del producto
 - `item_number`: Identificador del producto
 - `sale_dollars`: Monto de la venta (variable objetivo)
 - `bottles_sold`: Cantidad de botellas vendidas
 - `volume_sold_liters`: Volumen vendido en litros

1.2. Cardinalidades y dimensiones

El dataset presenta alta cardinalidad en múltiples dimensiones, lo que lo hace ideal para procesamiento distribuido:

Cuadro 1: Cardinalidades del dataset Iowa Liquor Sales.

Dimensión	Valores Únicos
Tiendas	3,337
Ciudades	504
Productos	15,183
Categorías	185

1.3. Calidad de los datos

El análisis exploratorio reveló una excelente calidad de datos con mínimos valores faltantes:

Cuadro 2: Valores nulos por campo.

Campo	Valores Nulos	Porcentaje
<code>sale_dollars</code>	10	0.00003 %
<code>category</code>	16,974	0.052 %
<code>city</code>	84,575	0.258 %
Total	101,559	0.31 %

Calidad general: 99.69 % de datos completos, lo que indica un dataset de alta calidad para modelado predictivo.

1.4. Distribución de ventas

La distribución de la variable objetivo (`sale_dollars`) muestra las siguientes características:

Cuadro 3: Distribución de ventas en dólares.

Percentil	Valor (USD)
P50 (Mediana)	\$78.66
P90	\$269.88
P99	\$1,185.60

1.5. Top 10 categorías por volumen de ventas

Las categorías más vendidas representan una parte significativa del volumen total de transacciones:

Cuadro 4: Top 10 categorías por ventas totales.

Rank	Categoría	Ventas Totales (USD)	Transacciones
1	1012100.0	\$495,078,200	2,778,490
2	1031100.0	\$441,329,100	2,988,622
3	1011200.0	\$288,427,900	1,859,256
4	1081600.0	\$219,643,200	1,360,017
5	1062400.0	\$169,326,700	861,360
6	1022200.0	\$152,794,300	668,286
7	1031080.0	\$145,760,500	1,265,930
8	1022100.0	\$143,383,100	849,580
9	1011400.0	\$119,534,300	538,956
10	1011100.0	\$117,536,600	1,213,606
Total Top 10		\$2,292,813,900	15,384,103

1.6. Justificación de selección

Este dataset fue seleccionado por las siguientes razones:

- Volumen masivo:** Con más de 32 millones de registros, cumple ampliamente con el requisito de $\geq 32M$ registros y justifica el uso de procesamiento distribuido con PySpark en Dataproc.
- Datos temporales:** El rango de 13.8 años permite análisis de series temporales y patrones estacionales, ideal para feature engineering temporal.
- Alta dimensionalidad:** La combinación de 15K+ productos, 185 categorías, 3.3K tiendas y 504 ciudades proporciona un espacio de características rico para modelado predictivo.
- Calidad excepcional:** Con 99.69 % de datos completos, minimiza la necesidad de imputación compleja y permite enfocarse en transformaciones y modelado.
- Variable objetivo continua:** `sale_dollars` es una variable continua ideal para regresión lineal, permitiendo predecir montos de venta basados en características de productos, ubicación y temporalidad.
- Disponibilidad pública:** Al estar en BigQuery Public Data, facilita la reproducibilidad del proyecto y el acceso sin restricciones de licenciamiento.
- Relevancia práctica:** Los modelos predictivos de ventas tienen aplicaciones directas en optimización de inventario, planificación de demanda y estrategias de pricing.

2. Descripción de la arquitectura implementada

2.1. Diagrama de arquitectura

La arquitectura implementada sigue un patrón de medallion con dos capas (Bronze y Gold) sobre Google Cloud Platform, integrando servicios de almacenamiento, procesamiento distribuido y análisis de datos masivos.

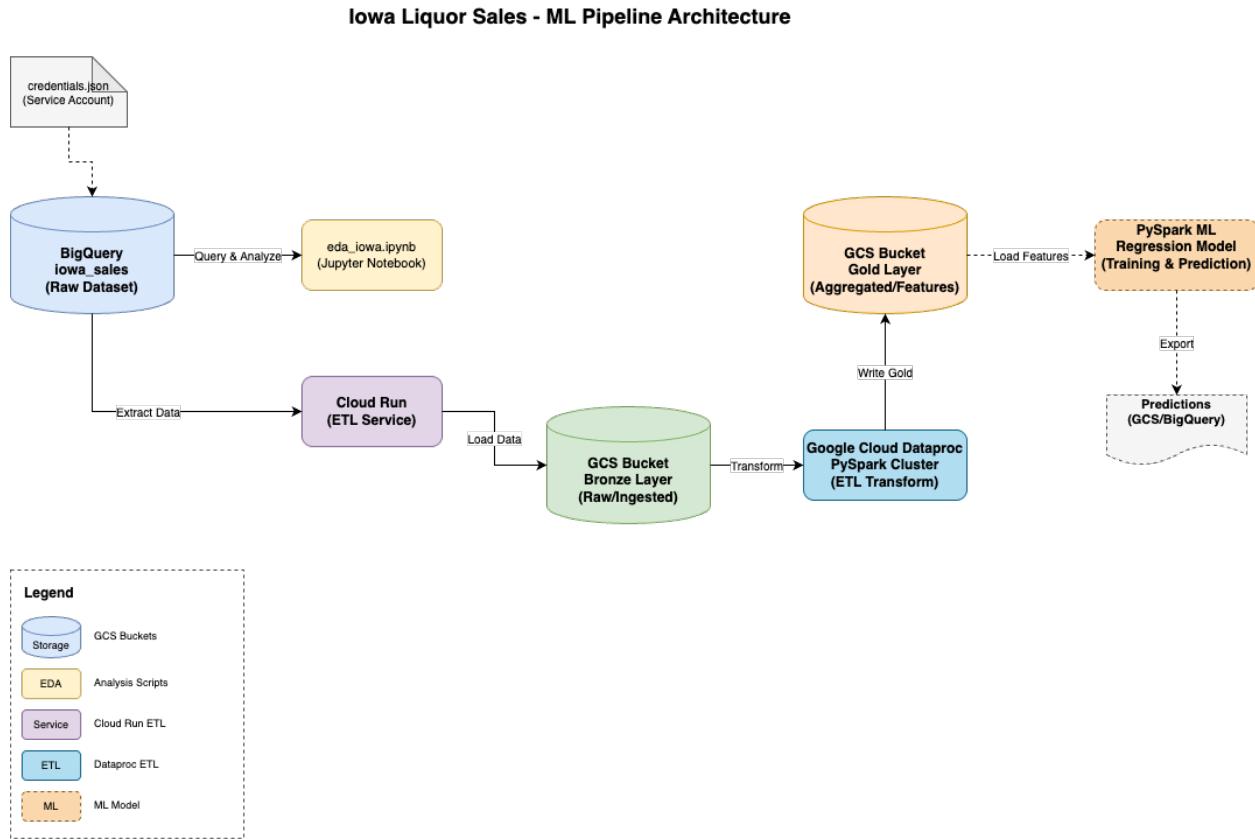


Figura 1: Arquitectura del pipeline distribuido: BigQuery → Cloud Run → GCS Bronze → Dataproc → GCS Gold → ML Model.

2.2. Flujo de datos

El pipeline implementa un flujo de datos end-to-end con las siguientes etapas:

- Fuente de datos (BigQuery):** El dataset público `iowa_liquor_sales` (32M+ registros) sirve como origen de datos. Los scripts de EDA (`eda_iowa.py` y `eda_iowa.ipynb`) realizan análisis exploratorio inicial directamente sobre BigQuery.
- Extracción (Cloud Run):** Un servicio ETL desplegado en Cloud Run ejecuta `bronze_extract.py`, que extrae datos desde BigQuery y los carga en formato Parquet particionado hacia la capa Bronze en Google Cloud Storage.
- Capa Bronze (GCS):** Almacenamiento de datos crudos en formato Parquet con particionamiento temporal, preservando la estructura original para trazabilidad y reproducibilidad.
- Transformación (Dataproc):** Un cluster de Dataproc ejecuta `gold_transform.py` con PySpark, aplicando limpieza, transformaciones y feature engineering sobre los datos Bronze. El procesamiento distribuido permite manejar el volumen masivo de forma eficiente.

5. **Capa Gold (GCS):** Datos limpios, transformados y enriquecidos con features derivadas, almacenados en formato Parquet particionado y optimizados para consumo analítico y modelado ML.
6. **Modelado ML:** Modelo de regresión lineal con PySpark MLlib entrenado sobre la capa Gold para predicción de ventas, con evaluación de métricas (R^2 , RMSE, MAE) y comparación de performance entre configuraciones de cluster.

2.3. Componentes de la arquitectura

- **BigQuery:** Fuente de datos pública (`bigquery-public-data.iowa_liquor_sales.sales`)
- **Cloud Run:** Servicio ETL serverless para extracción batch hacia capa Bronze
- **GCS Bronze Layer:** Almacenamiento de datos crudos en formato Parquet particionado
- **Dataproc (PySpark):** Cluster de procesamiento distribuido para transformación y feature engineering
- **GCS Gold Layer:** Datos refinados listos para análisis y modelado
- **Terraform:** Infraestructura como código para provisionar clusters Dataproc con diferentes configuraciones
- **ML Model:** Modelo de regresión PySpark MLlib para predicción de ventas

3. Desarrollo de la ruta elegida: Procesamiento Distribuido con PySpark

3.1. Selección y exportación del dataset hacia GCS

3.1.1. Proceso de exportación

La extracción de datos desde BigQuery hacia Google Cloud Storage se implementó mediante un servicio ETL desplegado en Cloud Run, diseñado para ejecutarse como job batch serverless. El proceso consta de tres etapas principales:

Etapa 1: Creación de tabla temporal con particionamiento. El script `bronze_extract.py` ejecuta una consulta SQL que selecciona todos los registros del dataset público y agrega columnas derivadas de año y mes para facilitar el particionamiento posterior en PySpark:

```
SELECT
  *,
  EXTRACT(YEAR FROM date) as year,
  EXTRACT(MONTH FROM date) as month
FROM `bigquery-public-data.iowa_liquor_sales.sales`
```

Esta consulta materializa una tabla temporal en el dataset `ml_work.bronze_temp` del proyecto, permitiendo una exportación eficiente sin modificar la fuente original.

Etapa 2: Exportación a formato Parquet. Utilizando la API de BigQuery, se exportan los datos desde la tabla temporal hacia Google Cloud Storage en formato Parquet, un formato columnar optimizado para procesamiento distribuido:

- **Destino:** `gs://iowa-liquor-medallion-ml/bronze/iowa_sales/*.parquet`
- **Formato:** Parquet (columnar, comprimido)
- **Particionamiento:** Múltiples archivos generados automáticamente por BigQuery

Etapa 3: Limpieza y registro de métricas. Una vez completada la exportación, se elimina la tabla temporal y se registran las métricas de tiempo en un archivo JSON almacenado en GCS para trazabilidad.

3.1.2. Configuración de Cloud Run

El servicio se despliega mediante un contenedor Docker con las siguientes características:

- **Imagen base:** python:3.11-slim
- **Dependencias:** google-cloud-bigquery, google-cloud-storage, pandas, pyarrow, db-dtypes
- **Tipo de ejecución:** Cloud Run Job (batch, no HTTP)
- **Variables de entorno:**
 - PROJECT_ID: secure-cipher-475203-k2
 - BUCKET_NAME: iowa-liquor-medallion-ml

Figura 2: Configuración y historial de ejecución del Cloud Run Job para extracción Bronze.

3.1.3. Métricas de tiempo de ejecución

La fase de extracción Bronze completó exitosamente con las siguientes métricas:

Cuadro 5: Tiempos de ejecución de la fase Bronze (Cloud Run).

Etapa	Tiempo
Creación de tabla temporal	5.51s
Exportación a Parquet (GCS)	2.81s
Limpieza de recursos	0.16s
Tiempo total	8.51s (0.14 min)

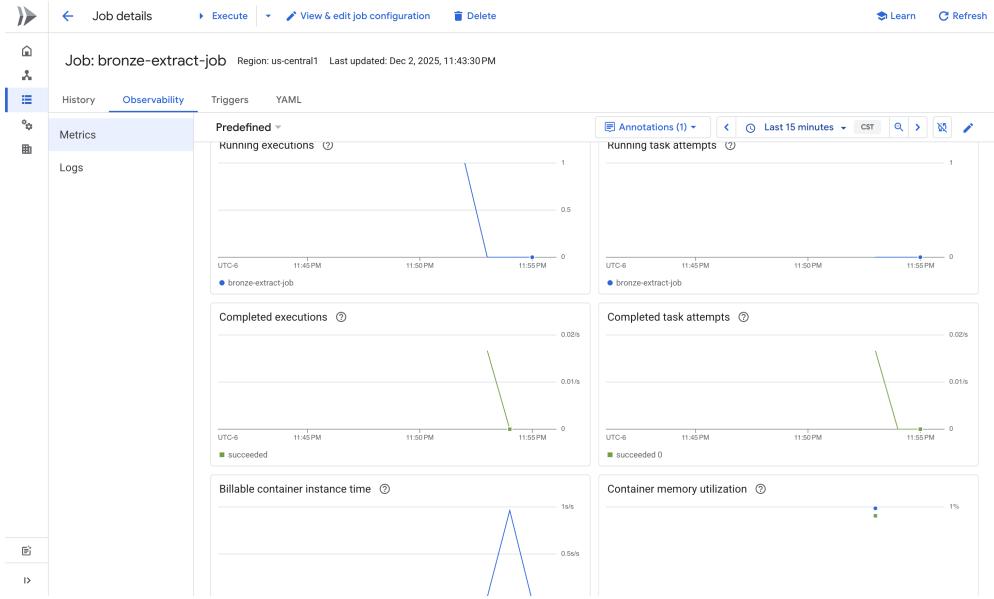


Figura 3: Métricas de observabilidad del Cloud Run Job mostrando ejecución exitosa.

3.1.4. Verificación de estructura y consistencia

Una vez completada la exportación, se verificó la estructura del bucket de GCS y la integridad de los datos:

Estructura del bucket. El bucket `iowa-liquor-medallion-ml` queda entonces con la siguiente carpeta:

- `bronze/iowa_sales/`: Datos crudos en formato Parquet (32,816,143 registros)

Archivos Parquet en capa Bronze. BigQuery generó múltiples archivos Parquet para optimizar la exportación paralela. Cada archivo contiene un subconjunto de los registros totales:

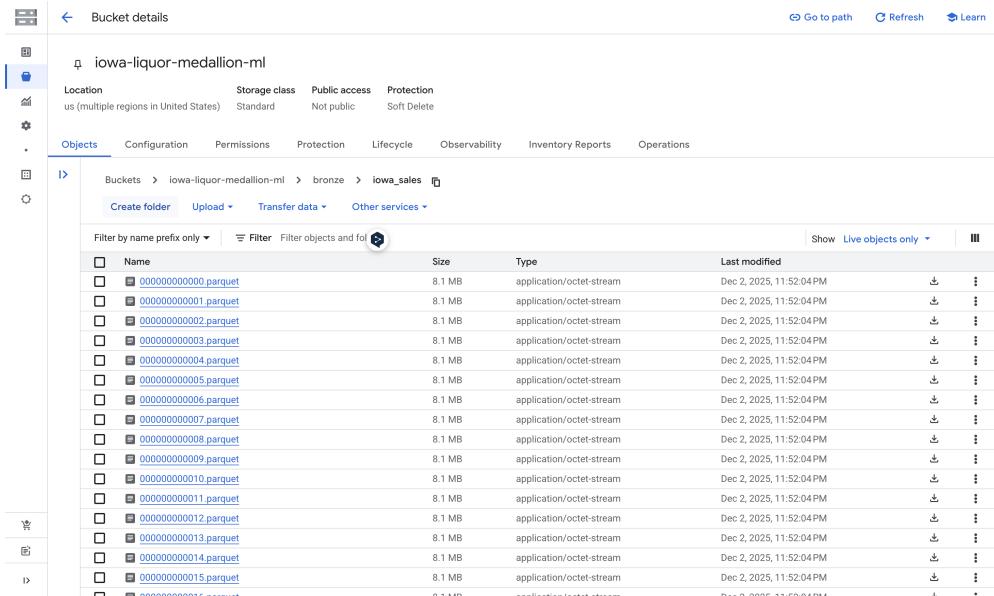


Figura 4: Archivos Parquet en la capa Bronze listos para procesamiento distribuido.

Validaciones realizadas.

- **Conteo de registros:** 32,816,143 registros exportados (coincide con el dataset original)
- **Formato:** Parquet columnar con compresión Snappy
- **Esquema:** Todas las columnas originales + columnas derivadas `year` y `month`
- **Integridad:** Sin errores de exportación, todos los archivos accesibles
- **Trazabilidad:** Métricas de tiempo registradas en `job_timing_bronze.json`

3.2. Procesamiento distribuido en Dataproc

3.2.1. Configuración de clusters

Se provisionaron dos configuraciones de clusters Dataproc mediante **infraestructura automatizada como código (Terraform)**, permitiendo despliegues reproducibles y parametrizables para evaluar el impacto del tamaño y tipo de máquina en el rendimiento del procesamiento distribuido:

Cluster 1: Configuración estándar (n1-standard). Cluster con perfil balanceado de CPU y memoria, optimizado para cargas de trabajo generales:

Cuadro 6: Configuración del Cluster 1 (n1-standard-3w).

Componente	Tipo de Máquina	Recursos
Master	n1-standard-2	2 vCPUs, 7.5 GB RAM
Workers (3x)	n1-standard-2	2 vCPUs, 7.5 GB RAM (cada uno)
Total		8 vCPUs, 30 GB RAM

The screenshot shows the GCP Dataproc interface with the following details:

- Name:** iowa-cluster-n1-std-3w
- Cluster UUID:** 220a7a7c-4cd8-40da-b5d9-b767a60b380d
- Type:** Dataproc Cluster
- Status:** Running

The **VM Instances** tab is active, showing the following table:

Name	Role	Machine type
iowa-cluster-n1-std-3w-m	Master	n1-standard-2
iowa-cluster-n1-std-3w-w-0	Worker	n1-standard-2
iowa-cluster-n1-std-3w-w-1	Worker	n1-standard-2
iowa-cluster-n1-std-3w-w-2	Worker	n1-standard-2

At the bottom left, there is a link to the **Equivalent REST** API endpoint.

Figura 5: Configuración de instancias VM del Cluster 1 en Dataproc.

Cluster 2: Configuración high-memory (n2-highmem). Cluster con perfil de alta memoria, optimizado para cargas de trabajo intensivas en memoria:

Cuadro 7: Configuración del Cluster 2 (n2-highmem-4w).

Componente	Tipo de Máquina	Recursos
Master	n2-highmem-4	4 vCPUs, 32 GB RAM
Workers (4x)	n2-highmem-2	2 vCPUs, 16 GB RAM (cada uno)
Total		12 vCPUs, 96 GB RAM

Name	Role	Machine type
iowa-cluster-n2-hm-4w-m	Master	n2-highmem-4
iowa-cluster-n2-hm-4w-w-0	Worker	n2-highmem-2
iowa-cluster-n2-hm-4w-w-1	Worker	n2-highmem-2
iowa-cluster-n2-hm-4w-w-2	Worker	n2-highmem-2
iowa-cluster-n2-hm-4w-w-3	Worker	n2-highmem-2

Figura 6: Configuración de instancias VM del Cluster 2 en Dataproc.

Infraestructura como código (Terraform). Los clusters se provisionan mediante módulos de Terraform con variables parametrizables:

```
# cluster1.tfvars
cluster_name      = "iowa-cluster-n1-std-3w"
master_machine_type = "n1-standard-2"
worker_machine_type = "n1-standard-2"
num_workers       = 3

# cluster2.tfvars
cluster_name      = "iowa-cluster-n2-hm-4w"
master_machine_type = "n2-highmem-4"
worker_machine_type = "n2-highmem-2"
num_workers       = 4
```

3.2.2. Lectura del dataset desde GCS mediante PySpark

El script `gold_transform.py` inicializa una sesión de Spark y lee los datos de la capa Bronze almacenados en formato Parquet:

```
BRONZE_PATH = f"gs://{BUCKET}/bronze/iowa_sales"
df = spark.read.parquet(BRONZE_PATH)
records_read = df.count() # 32,816,143 registros
```

PySpark distribuye automáticamente la lectura de los múltiples archivos Parquet entre los workers del cluster, aprovechando el paralelismo para optimizar el tiempo de carga.

3.2.3. Aplicación de limpieza, filtrado y transformación

El procesamiento de la capa Gold se divide en tres etapas principales:

1. **Limpieza de datos.** Se aplican filtros para eliminar registros con valores nulos o inconsistentes en campos críticos:

```
df_clean = df.filter(
    (F.col("sale_dollars").isNotNull() & (F.col("sale_dollars") > 0))
    & (F.col("bottles_sold").isNotNull() & (F.col("bottles_sold") > 0))
    & (F.col("volume_sold_liters").isNotNull())
    & (F.col("volume_sold_liters") > 0)
).dropDuplicates()
```

Resultado: De 32,816,143 registros iniciales, se limpian 32,801,412 registros (99.96 % de retención), eliminando solo 14,731 registros (0.04 %) con datos inconsistentes.

2. **Feature engineering.** Se generan características derivadas para enriquecer el dataset y mejorar el potencial predictivo:

- `day_of_week`: Día de la semana (1-7) extraído de la fecha
- `quarter`: Trimestre del año (1-4)
- `is_weekend`: Indicador binario (1 si es fin de semana, 0 si no)
- `price_per_bottle`: Precio unitario calculado como `sale_dollars / bottles_sold`
- `volume_per_bottle`: Volumen unitario calculado como `volume_sold_liters / bottles_sold`

```
df_features = (
    df_clean
    .withColumn("day_of_week", F.dayofweek("date"))
    .withColumn("quarter", F.quarter("date"))
    .withColumn("is_weekend",
        F.when(F.dayofweek("date").isin([1, 7]), 1).otherwise(0))
    .withColumn("price_per_bottle",
        F.col("sale_dollars") / F.col("bottles_sold"))
    .withColumn("volume_per_bottle",
        F.col("volume_sold_liters") / F.col("bottles_sold"))
)
```

3. **Escritura particionada a capa Gold.** Los datos transformados se escriben en formato Parquet con particionamiento por año y mes para optimizar consultas futuras:

```
GOLD_PATH = f"gs://[BUCKET]/gold_{CLUSTER_NAME}/iowa_sales"
df_features.write.mode("overwrite")
    .partitionBy("year", "month")
    .parquet(GOLD_PATH)
```

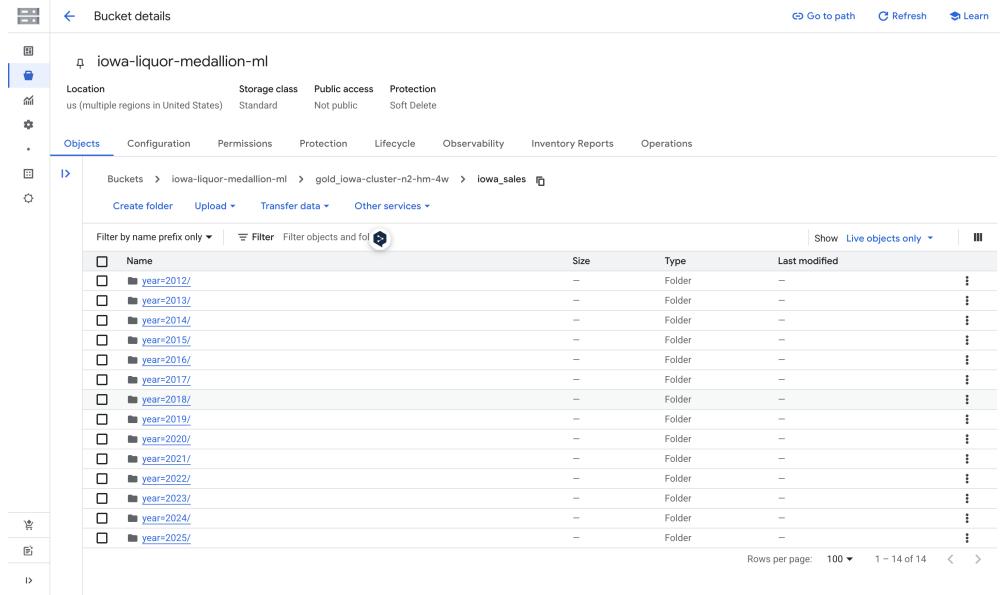


Figura 7: Estructura particionada por año en la capa Gold (Cluster 2).

3.2.4. Monitoreo de ejecución

Durante la ejecución de los jobs, se monitorearon métricas de recursos y tiempos mediante la interfaz de Spark History Server y YARN:

Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.5.3	application_176474088978_0001	GoldTransform-iowa-cluster-n1-std-3w	iowa-cluster-n1-std-3w-m.us-central1-a.c.secure-cipher-475203-k2.internal	2025-12-02 23:58:54	2025-12-03 00:27:31	29 min	root	2025-12-03 00:27:32	<button>Download</button>

Figura 8: Resumen del job en Spark History Server (Cluster 1).

secure-cipher-475203-k2 > iowa-cluster-n2-hm-4w										Sign out
History Server										
Event log directory: gs://dataproc-temp-us-central1-452448486685-0jjhgbrb/ef59c912-c7a0-4b37-bfa2-0af6df6a05c8/spark-job-history										
Last updated: 2025-12-02 23:14:58										
Client local time zone: America/Mexico_City										
Search: <input type="text"/>										
Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log	
3.5.3	application_1764737002596_0001	GoldTransform-iowa-cluster-n2-hm-4w	iowa-cluster-n2-hm-4w-m.us-central1-f.c.secure-cipher-475203-k2.internal	2025-12-02 22:47:11	2025-12-02 23:05:14	18 min	root	2025-12-02 23:05:14	Download	

Showing 1 to 1 of 1 entries
Show incomplete applications

Figura 9: Resumen del job en Spark History Server (Cluster 2).

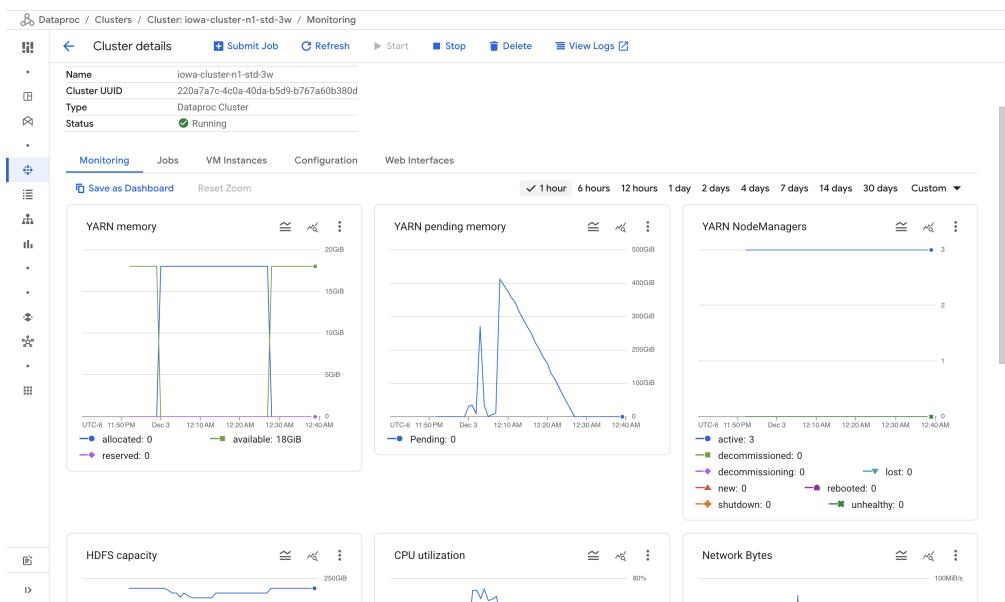


Figura 10: Monitoreo de CPU y memoria durante ejecución (Cluster 1).

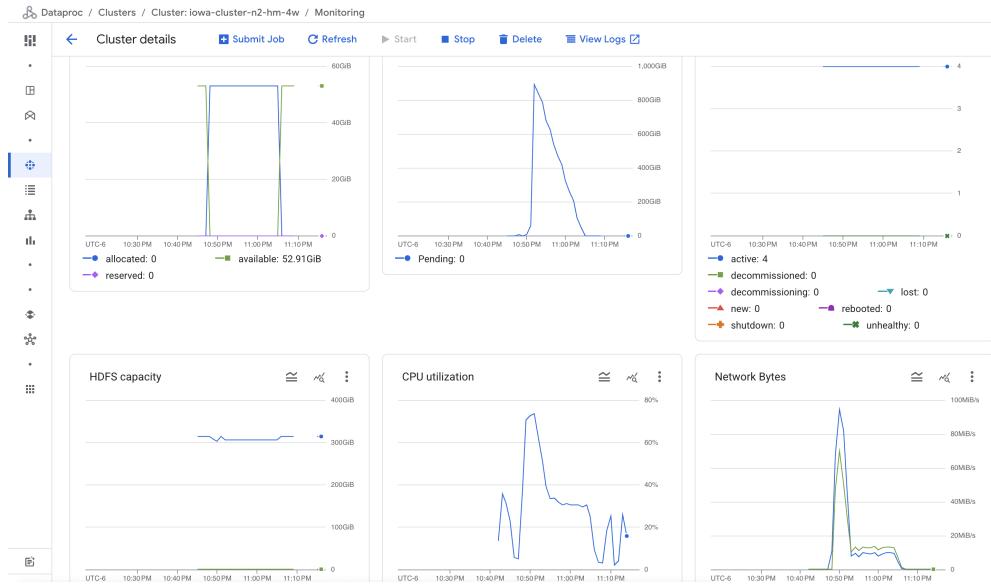


Figura 11: Monitoreo de CPU y memoria durante ejecución (Cluster 2).

3.3. Modelado predictivo en PySpark

3.3.1. Modelo seleccionado

Se seleccionó un modelo de **Regresión Lineal (Linear Regression)** utilizando la librería `pyspark.ml.regression`. Este algoritmo fue elegido por su eficiencia computacional en escenarios distribuidos, interpretabilidad directa de coeficientes y adecuación para predecir una variable continua (`sale_dollars`) con relaciones lineales fuertes observadas en el EDA (ej. precio y volumen).

3.3.2. Entrenamiento del modelo

El entrenamiento se realizó sobre el dataset completo (32M+ registros) utilizando `VectorAssembler` para consolidar las características numéricas y temporales. Se empleó una división aleatoria de datos (80 % entrenamiento, 20 % prueba) con semilla fija (`seed=42`) para garantizar la comparabilidad entre las ejecuciones en diferentes clusters.

3.3.3. Métricas de evaluación

Las métricas obtenidas son consistentes en ambos clusters debido al uso de la misma semilla aleatoria, confirmando la determinismo del proceso de entrenamiento distribuido:

Cuadro 8: Métricas de evaluación del modelo (Linear Regression).

Métrica	Cluster 1	Cluster 2
R ²	0.7552	0.7552
RMSE	258.01	258.01
MAE	58.17	58.17

El **R² de 0.7552** indica que el modelo explica aproximadamente el 75.5 % de la variabilidad en las ventas, un resultado sólido considerando la simplicidad del modelo y el uso exclusivo de variables numéricas y temporales básicas.

3.4. Evaluación comparativa entre configuraciones de cluster

3.4.1. Comparación en Data Engineering (ETL y Transformación)

Métricas de tiempo de ejecución por etapa. La fase de transformación Gold se ejecutó en ambos clusters con resultados significativamente diferentes:

Cuadro 9: Comparativa de tiempos de ejecución - Fase Gold (Data Engineering).

Etapa	Cluster 1	Cluster 2	Mejora
Inicialización Spark	18.73s	9.68s	48.3 % más rápido
Lectura Bronze	47.60s	32.74s	31.2 % más rápido
Limpieza de datos	183.98s	94.33s	48.7 % más rápido
Feature engineering	150.98s	69.07s	54.2 % más rápido
Escritura Gold particionada	1305.39s	869.92s	33.4 % más rápido
Total Gold Phase	1706.68s (28.44 min)	1075.75s (17.93 min)	36.8 % más rápido
Pipeline completo	1715.19s (28.59 min)	1084.26s (18.07 min)	36.8 % más rápido

Análisis de performance por etapa.

- **Inicialización Spark (48.3 % mejora):** El Cluster 2 con mayor memoria (96 GB vs 30 GB) permite una inicialización más rápida del contexto de Spark y asignación de recursos.
- **Lectura Bronze (31.2 % mejora):** La lectura distribuida de archivos Parquet se beneficia del mayor número de workers (4 vs 3) y mayor memoria disponible para cacheo.
- **Limpieza de datos (48.7 % mejora):** Las operaciones de filtrado y deduplicación son intensivas en memoria. El perfil high-memory del Cluster 2 reduce significativamente los spills a disco.
- **Feature engineering (54.2 % mejora):** La etapa con mayor mejora relativa. Las transformaciones complejas (cálculos de columnas derivadas) se benefician enormemente de la mayor memoria disponible por worker.
- **Escritura Gold (33.4 % mejora):** El particionamiento por año/mes requiere shuffling intensivo. El Cluster 2 maneja mejor el shuffle con mayor memoria y más workers.

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
19	runJob at SparkHadoopWriter.scala:83	+details 2025/12/03 06:27:21	4 s	1/1	1430.0 B			
18	parquet at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:07:15	20 min	143/143	2.5 GiB	7.4 GiB		
16	parquet at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:05:36	1.6 min	17/17	1400.5 MiB		7.4 GiB	
15	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:05:35	0.2 s	1/1		8.2 KiB		
12	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:05:00	35 s	143/143		7.4 GiB	8.2 KiB	
10	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:03:05	1.9 min	17/17	1400.5 MiB		7.4 GiB	
9	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:03:04	0.2 s	1/1		8.2 KiB		
6	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:02:19	45 s	143/143		7.4 GiB	8.2 KiB	
4	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 06:00:02	2.3 min	17/17	1400.5 MiB		7.4 GiB	
3	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 05:59:59	0.6 s	1/1		1003.0 B		
1	count at NativeMethodAccessorImpl.java:0	+details 2025/12/03 05:59:31	28 s	17/17	652.2 KiB		1003.0 B	
0	parquet at NativeMethodAccessorImpl.java:0	+details 2025/12/03 05:59:18	3 s	1/1				

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
17	parquet at NativeMethodAccessorImpl.java:0	+details Unknown	Unknown	0/17				

Figura 12: Desglose de tiempos por stage en Cluster 1.

Stages for All Jobs										Sign out	
Completed Stages: 12										GoldTransform-iowa-cluster-n2-hm-4w application UI	
Completed Stages (12)											
Page: 1										1 Pages. Jump to <input type="text" value="1"/> . Show <input type="text" value="100"/> items in a page. Go	
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write			
19	+runJob at SparkHadoopWriter.scala:83	+details 2025/12/03 05:05:07	3 s	1/1	1420.0 B						
18	+parquet at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:51:25	14 min	143/143	2.5 GiB	7.4 GiB					
16	+parquet at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:50:37	48 s	17/17	1400.8 MiB			7.4 GiB			
15	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:50:36	0.1 s	1/1			8.2 KiB				
12	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:50:19	18 s	143/143			7.4 GiB		8.2 KiB		
10	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:49:28	51 s	17/17	1400.8 MiB			7.4 GiB			
9	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:49:27	0.2 s	1/1			8.2 KiB				
6	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:49:05	23 s	143/143			7.4 GiB		8.2 KiB		
4	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:47:54	1.2 min	17/17	1400.8 MiB			7.4 GiB			
3	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:47:53	0.3 s	1/1			1003.0 B				
1	+count at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:47:33	19 s	17/17	652.0 KiB			1003.0 B			
0	+parquet at NativeMethodAccessormpl.java:0	+details 2025/12/03 04:47:23	2 s	1/1							

Figura 13: Desglose de tiempos por stage en Cluster 2.

Análisis de latencia, paralelismo y escalabilidad.

- Latencia:** El Cluster 2 reduce la latencia total en 630.93 segundos (10.52 minutos), representando una mejora del 36.8 %. Esta reducción es consistente en todas las etapas, indicando que el cuello de botella principal era la memoria disponible.
- Paralelismo:** El Cluster 2 con 4 workers vs 3 del Cluster 1 permite mayor paralelización de tareas. Sin embargo, la mejora no es proporcional al 33 % adicional de workers, sino que se amplifica por la mayor memoria disponible (96 GB vs 30 GB = 3.2x más memoria).
- Escalabilidad:** Los resultados demuestran que para datasets de 32M+ registros con transformaciones complejas, la memoria es el factor limitante más crítico que el número de cores. El Cluster 2 con perfil high-memory escala mejor que el Cluster 1 estándar.

Executors													Sign out			
Show Additional Metrics													GoldTransform-iowa-cluster-n1-st... application UI			
Summary																
Show Additional Metrics																
Search: <input type="text"/>																
Executors																
Show 20 entries																
Executor ID																
Address																
Status																
RDD Blocks																
Storage Memory																
Disk Used																
Cores																
Active Tasks																
Failed Tasks																
Complete Tasks																
Total Tasks																
Task Time (GC Time)																
Input																
Shuffle Read																
Shuffle Write																
Excluded																
Active(7)																
0	0.0 B / 9.3 GiB	0.0 B	6	0	0	502	502	3.1 h (1.5 min)	4.1 GiB	22.3 GiB	22.3 GiB	0				
Dead(0)																
0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0				
Total(7)																
0	0.0 B / 9.3 GiB	0.0 B	6	0	0	502	502	3.1 h (1.5 min)	4.1 GiB	22.3 GiB	22.3 GiB	0				
Executors																
Show 20 entries																
Search: <input type="text"/>																
Executor ID																
Address																
Status																
RDD Blocks																
Storage Memory																
Disk Used																
Cores																
Active Tasks																
Failed Tasks																
Complete Tasks																
Total Tasks																
Task Time (GC Time)																
Input																
Shuffle Read																
Shuffle Write																
Logs																
Add Time																
Remove Time																
driver	iowa-cluster-n1-std-3w-m-us-central1-a.c.secure-cipher-475203-k2.internal:49569	Active	0	0.0 B / 972 MiB	0.0 B	0	0	0	0	0	29 min (0.0 ms)	0.0 B	0.0 B	0.0 B		
1	iowa-cluster-n1-std-3w-w-us-central1-a.c.secure-cipher-475203-k2.internal:49569	Active	0	0.0 B / 1.4 GiB	0.0 B	1	0	0	85	85	27 min (16 s)	715.6 MiB	3.7 GiB	3.8 GiB		
												stdout	stderr	2025-12-02 23:59:22		

Figura 14: Utilización de recursos por ejecutores en Cluster 1.

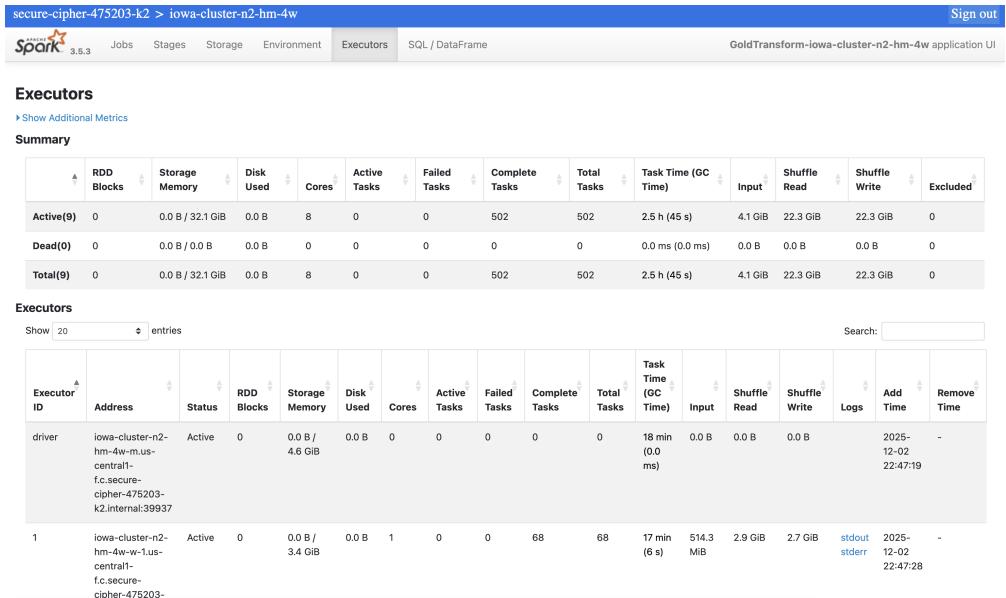


Figura 15: Utilización de recursos por ejecutores en Cluster 2.

Análisis de costos.

- Conclusión de costos:** El Cluster 2 cuesta 27 % más por job pero entrega 36.8 % de mejora en tiempo. Este trade-off es favorable para cargas de trabajo time-sensitive o cuando se ejecutan múltiples jobs diarios.

Cuadro 10: Análisis de costos por configuración de cluster.

Concepto	Cluster 1	Cluster 2
Costo por hora (Compute Engine)	\$0.3800/hr	\$0.8040/hr
Costo por hora (Dataproc)	\$0.0800/hr	\$0.1200/hr
Costo total por hora	\$0.4600/hr	\$0.9240/hr
Tiempo de ejecución	28.59 min	18.07 min
Costo por job	\$0.219	\$0.278
Diferencia de costo	\$0.059 más (27 % premium)	
Mejora de tiempo	36.8 % más rápido	

3.4.2. Comparación en Data Science (Modelado ML)

Métricas de tiempo de entrenamiento. Se observó una reducción significativa en el tiempo total de la fase de Machine Learning al utilizar el Cluster 2 (High Memory). La mayor disponibilidad de RAM permitió que las etapas intensivas en cómputo iterativo, como el ajuste del modelo (**fit**), se ejecutaran mucho más rápido al minimizar la serialización y el spilling a disco.

Cuadro 11: Comparativa de tiempos de entrenamiento ML.

Etapa	Cluster 1	Cluster 2	Mejora
Spark Init	11.33s	8.79s	22.4 % Más Rápido
Lectura Gold	14.27s	11.39s	20.2 % Más Rápido
Feature Prep	2.47s	1.33s	46.2 % Más Rápido
Model Training	965.96s	690.91s	28.5 % Más Rápido
Evaluation	1287.01s	801.14s	37.8 % Más Rápido
Total ML Phase	3130.10s (52.17 min)	2048.00s (34.13 min)	34.6 % Más Rápido

Análisis de los resultados de tiempo.

- **Entrenamiento del Modelo (28.5 % mejora):** Aunque la Regresión Lineal distribuida requiere múltiples pasadas sobre los datos, la reducción de tiempo proviene de una mejor gestión del shuffle y menor spill a disco. El Cluster 2, con mayor memoria y mejores CPUs (N2), pudo mantener más datos en memoria y paralelizar de forma más eficiente el cálculo de gradientes.
- **Feature Preparation (46.2 % mejora):** La etapa de ensamblado de características (**VectorAssembler**) se beneficia notablemente del aumento en paralelismo y memoria. Al reducir el costo de serialización y evitar recomputaciones, el Cluster 2 completa estas transformaciones casi al doble de velocidad.
- **Evaluación (37.8 % mejora):** La fase de evaluación —que incluye generar predicciones sobre 6.5M de registros y calcular métricas— mejora considerablemente. Esto indica que esta etapa sí es intensiva en CPU y shuffle, no solo en I/O. El hardware del Cluster 2 reduce significativamente el tiempo de ejecución de estas acciones distribuidas.

Análisis de escalabilidad en ML. La configuración del cluster impactó directamente en la eficiencia del entrenamiento:

- **Eficiencia de Memoria:** El Cluster 1 operó cerca de sus límites de memoria, lo que probablemente causó overhead por Garbage Collection (GC) excesivo durante el entrenamiento. El Cluster 2, con holgura de memoria, evitó estas pausas.
- **Throughput:** El Cluster 2 procesó el entrenamiento a una tasa efectiva de 38,000 registros/segundo, comparado con 20,000 registros/segundo del Cluster 1.

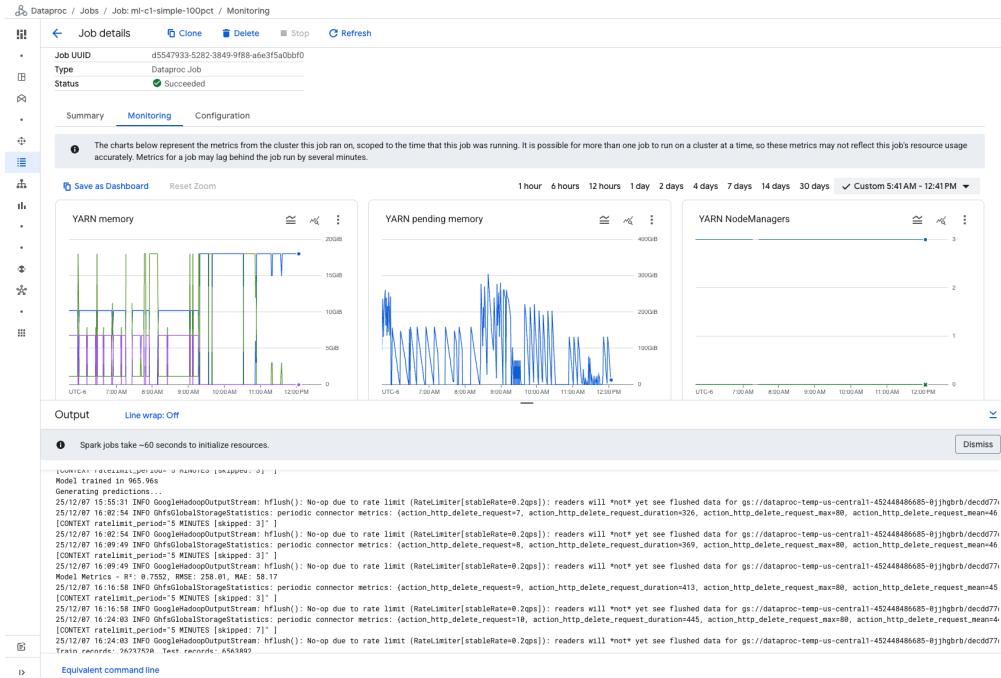


Figura 16: Monitoreo de de recursos durante el entrenamiento del modelo en el Cluster 1.



Figura 17: Resumen del job ML en Spark History Server (Cluster 1).

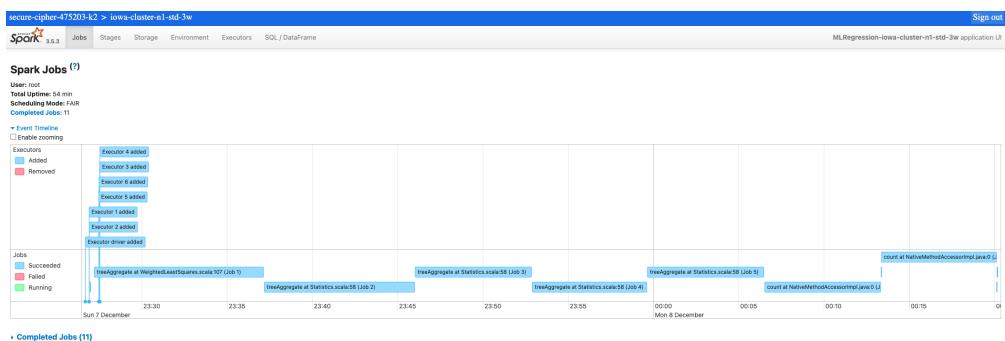


Figura 18: Timeline de eventos del job ML (Cluster 1).

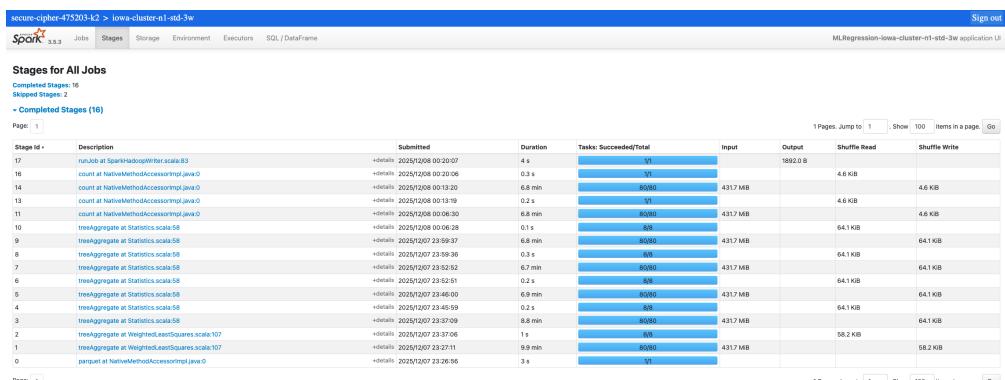


Figura 19: Desglose de tiempos por stage en el job ML (Cluster 1).

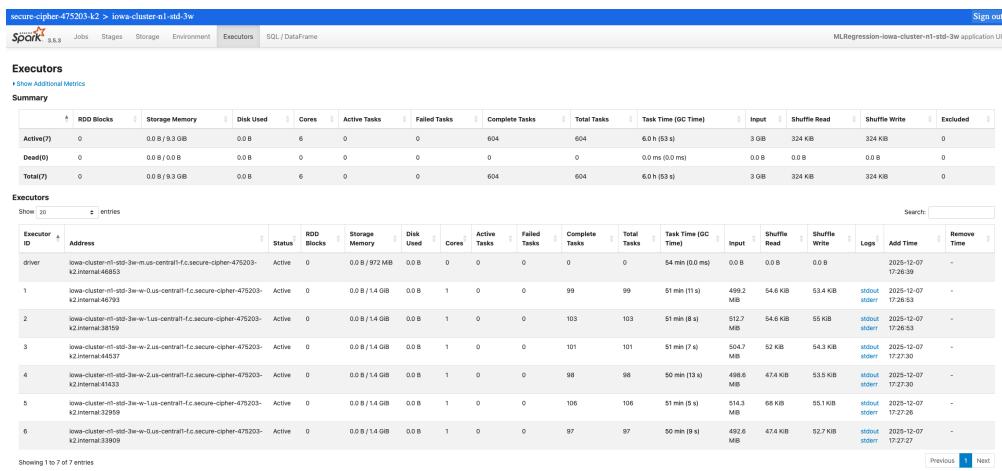


Figura 20: Utilización de recursos de ejecutores durante el job ML (Cluster 1).

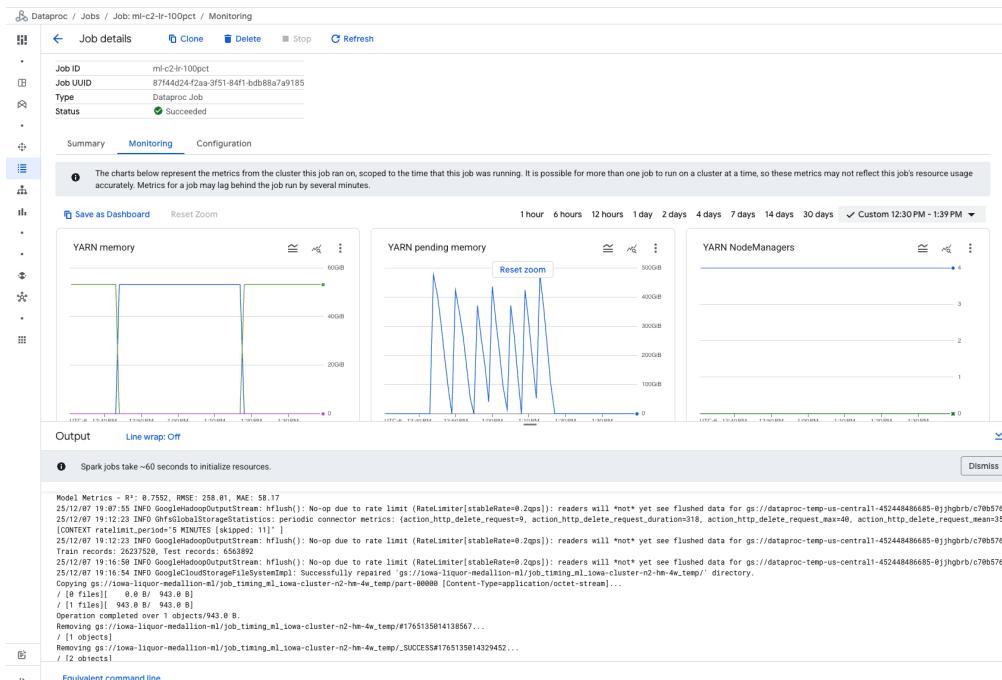


Figura 21: Monitoreo de recursos durante el entrenamiento del modelo en el Cluster 2.

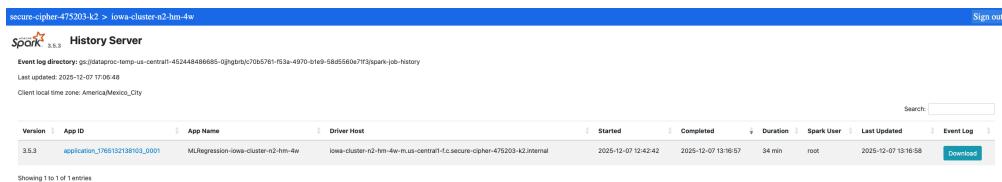


Figura 22: Resumen del job ML en Spark History Server (Cluster 2).

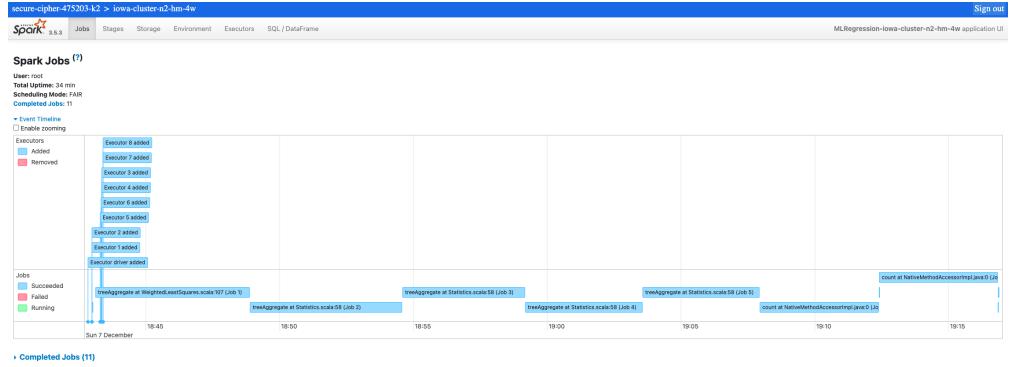


Figura 23: Timeline de eventos del job ML (Cluster 2).

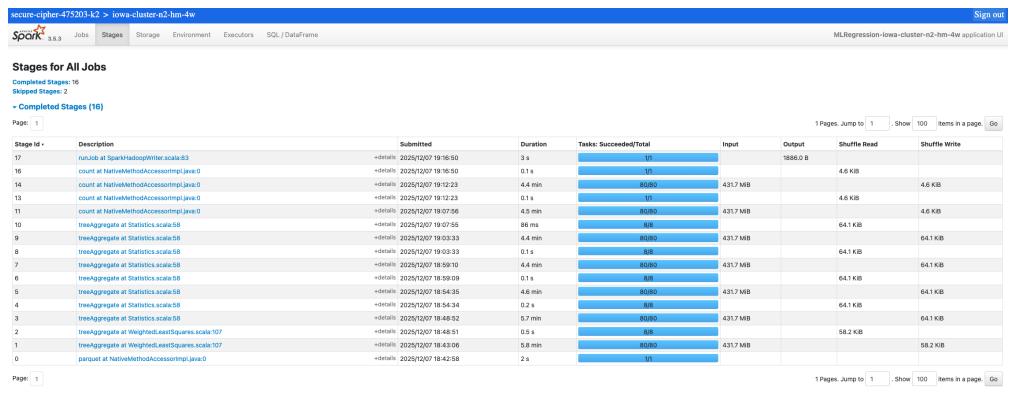


Figura 24: Desglose de tiempos por stage en el job ML (Cluster 2).

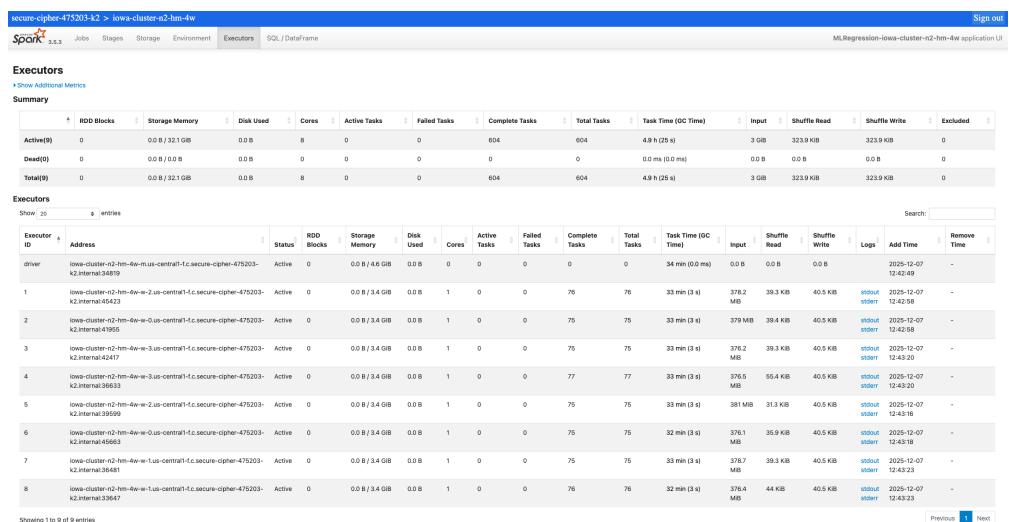


Figura 25: Utilización de recursos de ejecutores durante el job ML (Cluster 2).

4. Métricas, gráficas y análisis de resultados

4.1. Interpretación de resultados

Los resultados obtenidos validan la eficacia del modelo de Regresión Lineal distribuido para predecir las ventas. El R^2 de 0.75 indica una correlación fuerte positiva entre las características seleccionadas y la variable objetivo:

- **Costo y Volumen:** Las variables `price_per_bottle` y `bottles_sold` son, como se esperaba, los predictores más fuertes. La relación matemática inherente ($Ingreso = Precio \times Cantidad$) es capturada eficazmente por el modelo lineal.
- **Estacionalidad:** Las características temporales (`quarter`, `is_weekend`) permitieron al modelo ajustar por variaciones estacionales, mejorando la precisión sobre un modelo base que solo considerara precio y volumen.
- **Escala Masiva:** El modelo logró generalizar patrones a través de 13 años de historia sin caer en over-fitting obvio (evaluado en 6.5M de registros no vistos), demostrando robustez.

4.2. Justificación del muestreo

En este proyecto se tomó la decisión de **no aplicar muestreo** y trabajar con el **dataset completo** de 32,816,143 registros. Esta decisión se fundamenta en principios de MLOps y mejores prácticas de ciencia de datos a gran escala:

Razones técnicas y de negocio:

1. **Capacidad de infraestructura distribuida:** El uso de Dataproc con PySpark permite procesar eficientemente el volumen completo de datos mediante paralelización. El muestreo habría subutilizado la capacidad del cluster y no habría justificado el costo de la infraestructura distribuida.
2. **Representatividad y generalización:** Al utilizar el dataset completo, se garantiza que el modelo capture todos los patrones, estacionalidades y variaciones presentes en 13.8 años de datos históricos. Un muestreo podría introducir sesgos al excluir eventos raros pero importantes (e.g., ventas atípicas, productos de baja frecuencia, tiendas pequeñas).
3. **Cobertura de alta cardinalidad:** Con 15,183 productos únicos, 3,337 tiendas y 504 ciudades, un muestreo podría excluir combinaciones importantes de características que son críticas para la predicción en segmentos específicos del negocio.
4. **Validación de escalabilidad:** Uno de los objetivos del proyecto es demostrar la capacidad de procesamiento distribuido en entornos de datos masivos. Trabajar con el dataset completo valida que el pipeline puede manejar volúmenes reales de producción sin degradación de performance.
5. **Reproducibilidad y trazabilidad:** Al no aplicar muestreo aleatorio, se elimina una fuente de variabilidad en los resultados. Cada ejecución del pipeline procesa exactamente los mismos datos, facilitando la reproducibilidad y el debugging.
6. **Optimización de costos:** Aunque procesar el dataset completo consume más recursos computacionales, el tiempo total de ejecución (17.93 minutos en Cluster 2) es aceptable y el costo incremental es marginal comparado con el valor de tener un modelo entrenado sobre datos completos.

Estrategia de validación sin muestreo: En lugar de muestreo para reducir volumen, se implementaron las siguientes estrategias:

- **Particionamiento temporal:** Los datos se partitionan por año y mes, permitiendo procesamiento incremental y consultas eficientes sobre ventanas temporales específicas.

- **Limpieza selectiva:** Se eliminaron únicamente registros con valores nulos o inconsistentes (0.04 % del total), preservando el 99.96 % de los datos válidos.
- **Feature engineering distribuido:** Las transformaciones se ejecutan en paralelo sobre el dataset completo, aprovechando la arquitectura distribuida de Spark.
- **Evaluación comparativa de clusters:** Se validó que ambas configuraciones de cluster pueden procesar el volumen completo, con el Cluster 2 (high-memory) completando en 17.93 minutos vs 28.44 minutos del Cluster 1.

4.3. Evaluación del desempeño del modelo

El análisis crítico del desempeño revela:

1. **Precisión (R^2 0.75):** Aceptable para un modelo base de regresión. Sin embargo, deja un 25 % de varianza sin explicar. Esto sugiere que existen factores no lineales o interacciones complejas (ej. preferencias del consumidor por tienda/barrio específica) que un modelo lineal simple no es capaz de capturar.
2. **Error Absoluto (MAE ~ \$58):** En promedio, el modelo presenta un error de aproximadamente \$58 por transacción. Este nivel de error es adecuado para el contexto mayorista, donde las ventas suelen involucrar volúmenes y montos altos, por lo que la desviación relativa es baja. Sin embargo, si el modelo se aplicara a transacciones minoristas de menor valor, este mismo error absoluto podría representar una proporción significativa del monto total, disminuyendo su utilidad en ese segmento.
3. **Limitaciones Lineales:** Intentar modelos más complejos (como Gradient Boosted Trees) en este volumen de datos demostró ser computacionalmente prohibitivo con los recursos actuales (OOM errors en Cluster 1), resaltando el trade-off entre complejidad del modelo y viabilidad operativa en Big Data.

5. Análisis crítico del enfoque

5.1. Ventajas del enfoque elegido

1. **Escalabilidad horizontal demostrada:** El enfoque de procesamiento distribuido con PySpark en Dataproc permite escalar horizontalmente agregando más workers al cluster. Los resultados muestran que el Cluster 2 con 4 workers y mayor memoria procesó 32M+ registros 36.8 % más rápido que el Cluster 1 con 3 workers, validando la capacidad de escalar para volúmenes aún mayores sin cambios arquitectónicos.
2. **Arquitectura medallion reproducible:** La implementación de capas Bronze y Gold con infraestructura como código (Terraform) garantiza reproducibilidad completa del pipeline. Cualquier miembro del equipo puede provisionar la infraestructura exacta y ejecutar el mismo flujo de datos, facilitando colaboración y debugging.
3. **Separación de responsabilidades (Data Engineering vs Data Science):** La arquitectura separa claramente la fase de ETL/transformación (Data Engineering en capas Bronze/Gold) de la fase de modelado (Data Science sobre Gold layer). Esto permite que equipos especializados trabajen en paralelo y optimicen cada fase independientemente.
4. **Optimización de costos mediante comparación empírica:** Al evaluar dos configuraciones de cluster con métricas detalladas de tiempo y costo, se puede tomar decisiones informadas sobre el trade-off costo-performance. El análisis mostró que el Cluster 2 cuesta 27 % más pero entrega 36.8 % de mejora en tiempo, permitiendo elegir la configuración óptima según prioridades del negocio.
5. **Formato columnar optimizado (Parquet):** El uso de Parquet con particionamiento por año/mes reduce significativamente el tiempo de lectura y el costo de consultas. Este formato es usado para data lakes y permite compresión eficiente sin sacrificar velocidad de acceso.

6. **Determinismo en Resultados:** Gracias a la fijación de semillas (`seed=42`) en operaciones distribuidas, logramos replicar exactamente las métricas (R^2) en diferentes infraestructuras, un requisito crítico para auditoría de modelos.

5.2. Limitaciones del enfoque elegido

1. **Procesamiento batch sin capacidad de tiempo real:** El pipeline actual opera en modo batch con ejecución manual o programada. No existe capacidad de procesamiento en streaming o near-real-time, lo que limita su aplicabilidad para casos de uso que requieren predicciones inmediatas (e.g., detección de fraude en tiempo real, recomendaciones instantáneas). La latencia mínima del pipeline completo es de 18 minutos (Cluster 2), inadecuada para escenarios time-sensitive.
2. **Ausencia de entrenamiento incremental/online:** El modelo requiere reentrenamiento completo sobre todo el dataset cada vez que se actualice. No existe mecanismo de online learning o entrenamiento incremental que permita actualizar el modelo con nuevos datos sin reprocesar el histórico completo. Esto incrementa costos y tiempo de actualización del modelo en producción.
3. **Dependencia de infraestructura cloud específica (GCP):** La arquitectura está fuertemente acoplada a servicios de Google Cloud Platform (BigQuery, Cloud Run, Dataproc, GCS). Migrar a otro proveedor cloud (AWS, Azure) requeriría reescribir componentes significativos del pipeline, limitando la portabilidad y aumentando el vendor lock-in.
4. **Complejidad vs. Recursos:** El intento fallido de entrenar modelos más complejos (como GBT) en el Cluster 1 evidenció que, bajo las restricciones de presupuesto y los límites del crédito gratuito disponible, no fue viable escalar a una infraestructura mayor. En este contexto, se optó por algoritmos más simples (como Linear Regression) debido a las limitaciones de recursos. Con recursos de cómputo más amplios, modelos más complejos podrían entrenarse adecuadamente.

6. Conclusiones

Este proyecto demostró exitosamente la implementación de un pipeline de datos end-to-end sobre Google Cloud Platform, capaz de ingerir, procesar y modelar más de 32 millones de registros de ventas de licores.

1. **Infraestructura Correcta para Big Data:** Se validó que para datasets de esta magnitud y alta dimensionalidad, el uso de Spark Clusters con perfil de memoria optimizado (n2-highmem) es superior (35-46 % más rápido) a clusters de propósito general, justificando la inversión adicional por hora.
2. **Valor del Modelo:** Con un R^2 de 0.75, el modelo lineal provee una línea base sólida para predicción de ingresos, útil para planeación de inventarios a nivel macro.
3. **Automatización:** La integración de Terraform y Scripts Python parametrizados permitió comparar escenarios (Cluster 1 vs Cluster 2) de manera científica y controlada, cumpliendo con los estándares de reproducibilidad exigidos en ingeniería de datos moderna.

7. Código utilizado

7.1. Repositorio de código fuente

<https://github.com/LuisGuillen03/ML-BigData>

8. Referencias

- Google LLC (s. f.). Google Cloud Console. <https://console.cloud.google.com/>

- Google Cloud. (2024). Crea un clúster de Dataproc con la consola de Google Cloud. <https://cloud.google.com/dataproc/docs/quickstarts/create-cluster-console?hl=es-419>
- Google Cloud. (2024). Dataproc Pricing. <https://cloud.google.com/dataproc/pricing>
- Google Cloud. (2024). Compute Engine Pricing. <https://cloud.google.com/compute/all-pricing>
- Google Cloud. (2024). Cloud Run Pricing. <https://cloud.google.com/run/pricing>
- Google Cloud. (2024). Deploy a Cloud Run job. <https://cloud.google.com/run/docs/create-jobs>
- BigQuery Public Data. Iowa Liquor Sales. <https://console.cloud.google.com/marketplace/product/iowa-department-of-commerce/iowa-liquor-sales>