Caso 2 – Dataset Metro de Santiago, con PySpark

Hans Jorge Contreras Robledo

Fecha: 15/09/2025

Introducción

En este ejercicio trabajamos con un dataset simulado de afluencia en el Metro de Santiago. El objetivo fue demostrar cómo aplicar PySpark para procesar datos de forma distribuida, hacer consultas rápidas, realizar un análisis de clustering con KMeans y finalmente simular un caso de procesamiento en streaming.

1. Inicialización de Spark

Comenzamos Instalando PySpark en entorno de Google Collab y creando la sesión de Spark (`SparkSession`). Esto nos permitió usar Spark SQL, DataFrames y librerías de Machine Learning distribuidas en MLlib.

2. Carga de datos y muestra

Se cargó el CSV con información del Metro de Santiago. Para simplificar el análisis, se tomó una muestra del 20% de los datos. Esto nos permitió trabajar más rápido en el ejemplo pero manteniendo representatividad.

3. Consultas con RDD, DataFrame y SQL

Ejecutamos diferentes tipos de consultas:

- Con RDDs: contamos la frecuencia de estaciones en la muestra.
- Con DataFrames: agrupamos registros según columnas.
- Con SQL: creamos la vista `metro` y consultamos el total de registros por línea.

4. Clustering con KMeans

Preparamos un conjunto de características numéricas y aplicamos el algoritmo KMeans para dividir los datos en 3 grupos. Antes de entrenar, usamos `VectorAssembler` y `StandardScaler` para preparar los datos. Luego evaluamos el modelo con la métrica Silhouette, que nos indicó qué tan bien formados estaban los grupos.

5. Procesamiento en Streaming simulado

Se configuró un pipeline de Structured Streaming para simular cómo Spark puede procesar datos que van llegando de forma continua, en este caso con columnas de `timestamp`, `linea`, `estacion` y `afluencia`. Agrupamos los datos en ventanas de 5 minutos y calculamos el total de pasajeros por estación.

6. Guardado de artefactos y modelo

El modelo KMeans entrenado se guardó en la carpeta `models/`, y las métricas de Silhouette se almacenaron en `artifacts/metro_metrics.json`. Esto permite conservar los resultados del análisis y reutilizar el modelo sin necesidad de volver a entrenarlo.

Conclusiones

En este caso aprendimos cómo aplicar PySpark para un escenario urbano como el Metro de Santiago. Usamos diferentes enfoques de consulta (RDD, DataFrame, SQL), realizamos clustering con KMeans y simulamos un flujo de streaming en tiempo real. Esto refuerza la importancia de Spark como herramienta para manejar grandes volúmenes de datos, tanto en procesamiento batch como en tiempo real.