Arquitectura para Grandes Volúmenes de Datos

Proyecto Final

Prof. Wilmer Efrén Pereira

198002 Diego Arellano Zamudio

196950 Aldo Carmona Jimenez

Descripción del proyecto

Se desarrolló una aplicación en Spark que captura datos a través de un flujo de datos de Spark Structure Streaming, simulando una llegada de datos en tiempo real. Los datos que se utilizaron en este proyecto son referentes a partidos de Hockey de la liga NHL del año 1918 al 2022. Estos datos se utilizaron para entrenar y evaluar un modelo de regresión logística cuyo objetivo es clasificar si el partido lo ganó el equipo local o no. Este proyecto se ejecutó en Databricks y en el entorno local de Spark, con el objetivo de comparar los tiempos de ejecución.

Obtención de los datos

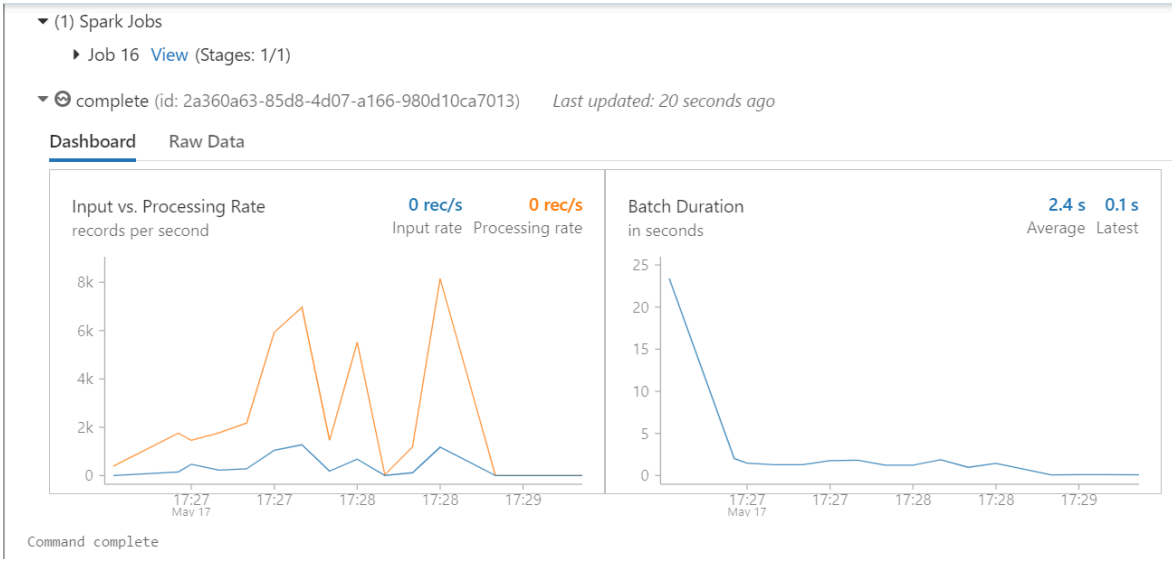
La lectura de los datos se llevó a cabo por conjuntos que abarcan los partidos de cada década. A continuación presentamos el esquema de los datos:

- La columna season es de tipo IntegerType y representa la temporada en la que se jugó el partido, típicamente un año o un conjunto de años, como 2023 para la temporada 2022-2023.
- La columna date es de tipo TimestampType y almacena la fecha y la hora exacta del partido.
- Las columnas home_team_abbr y away_team_abbr son de tipo StringType y contienen las abreviaturas de los nombres de los equipos local y visitante, respectivamente, como "TOR" para los Toronto Maple Leafs y "MTL" para los Montreal Canadiens.
- Las columnas home_team_pregame_rating y away_team_pregame_rating, ambas de tipo FloatType, representan las calificaciones previas al juego de los equipos local y visitante, respectivamente. Estas calificaciones se basan en el rendimiento pasado y otras estadísticas relevantes.
- Las columnas home_team_winprob y away_team_winprob, también de tipo FloatType, indican las probabilidades de victoria de los equipos local y visitante, expresadas como valores decimales entre 0 y 1, donde 0.75 significa un 75% de probabilidad de ganar.
- La columna overtime_prob, de tipo FloatType, muestra la probabilidad de que el partido vaya a tiempo extra, igualmente expresada como un valor decimal entre 0 y 1.

- Las columnas home_team_expected_points y away_team_expected_points, ambas de tipo FloatType, contienen los puntos esperados que se anticipa que los equipos local y visitante consigan en el partido, según algún modelo predictivo.
- La columna home_team_won es de tipo IntegerType y es un indicador binario que muestra si el equipo local ganó el partido, donde 1 indica una victoria y 0 indica que no ganó.
- Finalmente, la columna decade, de tipo IntegerType, representa la década en la que se jugó el partido, como 2020 para la década de los 2020s, lo cual es útil para realizar análisis históricos y comparativos a lo largo de diferentes décadas. Este esquema estructurado permite un análisis detallado y eficiente de los datos de partidos de hockey.

Finalmente, analizamos el flujo de obtención de datos y obtuvimos los siguientes resultados:

Flujo de entrada mínimo: 0 rec/s
Flujo de entrada máximo:1516 rec/s
Flujo de entrada promedio: 321.65 rec/s
Varianza del flujo de entrada: 511.24



Visualización de métricas interesantes

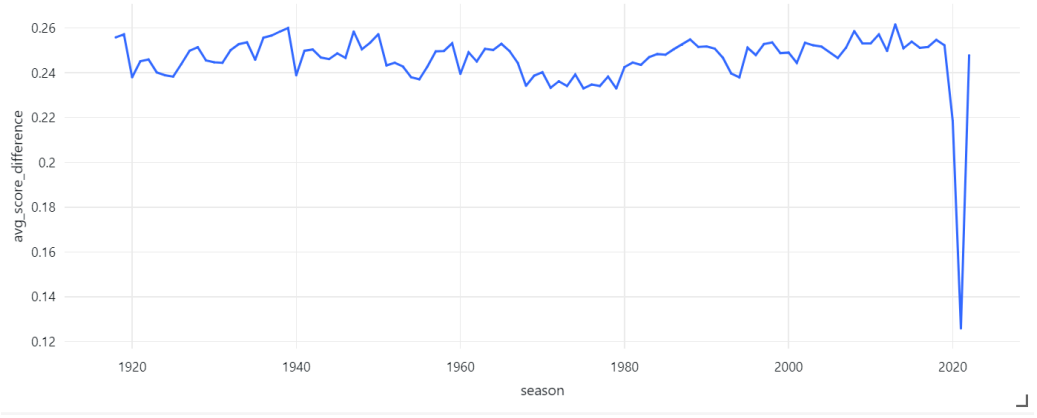
La media y desviación estándar de los puntos esperados del equipo local

	avg_home_expected_points	stddev_home_expected_points
1	1.2383869539350782	0.17989008719773278

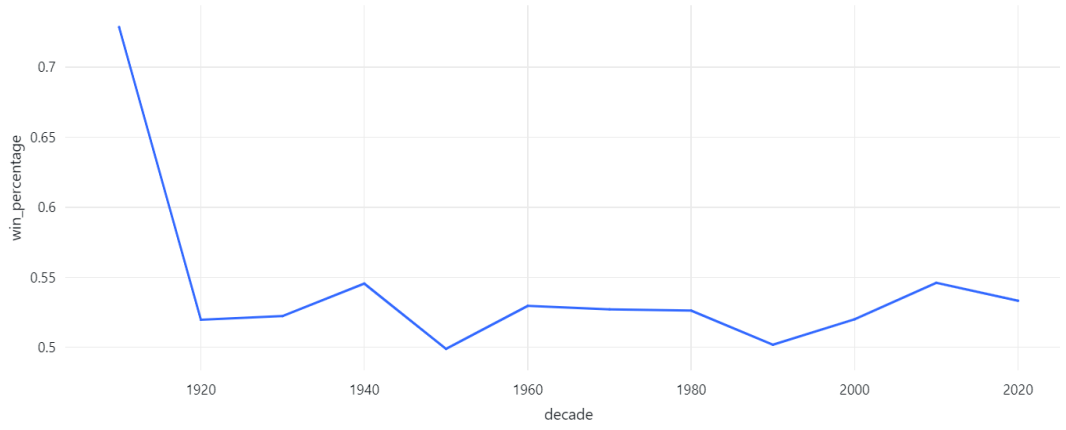
La fecha mínima y la fecha máxima

	min_date	max_date
1	1917-12-19T00:00:00.000+0000	2022-06-26T00:00:00.000+0000

La diferencia de puntos esperada por cada temporada



Porcentaje de juegos ganados por los equipos locales por cada década

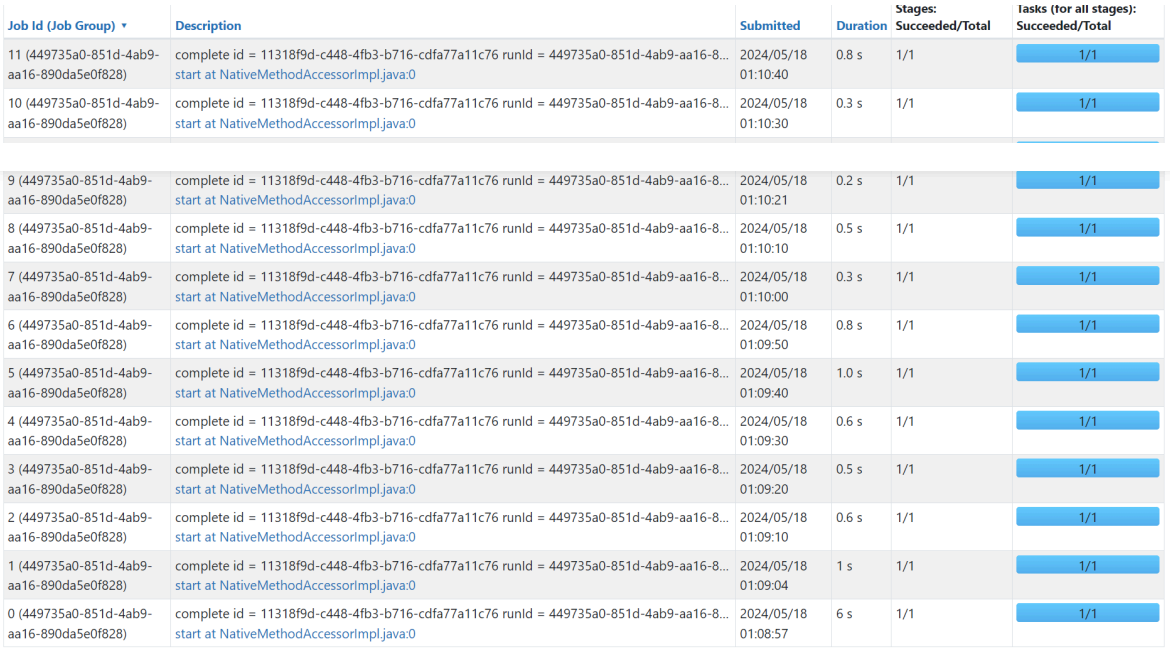


Explicación de los *jobs* y tareas

Un job en Spark es una unidad completa de trabajo que se envía para ejecutar una acción sobre un DataFrame. Las acciones en Spark son operaciones que devuelven un valor después de correr un cálculo sobre los datos, como count(), collect() o save(). Cada acción que se llama en un programa Spark inicia un job.

Los jobs se dividen en stages. Los stages se forman según las operaciones que requieren que los datos sean shuffleados (reorganizados).

Un *stage* contiene tareas que pueden ejecutarse en paralelo. Cada *stage* se compone de tareas, que son las unidades de trabajo más pequeñas que se envían a los ejecutores para su procesamiento. Una tarea en Spark corresponde a la ejecución de una operación en una partición del DataFrame.
El Driver en Spark crea el plan de ejecución del job, dividiéndolo en stages y tareas. Luego, el Driver envía las tareas a los ejecutores que están en los nodos del clúster. Los ejecutores son procesos que ejecutan las tareas y devuelven los resultados al Driver.



Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:10:40	0.8 s	1/1	1/1
10 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:10:30	0.3 s	1/1	1/1
9 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:10:21	0.2 s	1/1	1/1
8 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:10:10	0.5 s	1/1	1/1
7 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:10:00	0.3 s	1/1	1/1
6 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:09:50	0.8 s	1/1	1/1
5 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:09:40	1.0 s	1/1	1/1
4 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:09:30	0.6 s	1/1	1/1
3 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:09:20	0.5 s	1/1	1/1
2 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:09:10	0.6 s	1/1	1/1
1 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:09:04	1 s	1/1	1/1
0 (449735a0-851d-4ab9-aa16-890da5e0f828)	complete id = 11318f9d-c448-4fb3-b716-cdfa77a11c76 runId = 449735a0-851d-4ab9-aa16-8... start at NativeMethodAccessorImpl.java:0	2024/05/18 01:08:57	6 s	1/1	1/1

La primera columna, "Job Id (Job Group)", muestra el ID del trabajo y el ID del grupo de trabajo al que pertenece. El ID del trabajo es un identificador único para el trabajo, mientras que el ID del grupo de trabajo es un identificador único para el grupo de trabajos al que pertenece el trabajo. Los trabajos se agrupan en grupos de trabajos para facilitar su gestión y organización.

La segunda columna, "Description", muestra una descripción del trabajo. La descripción proporciona información sobre lo que hace el trabajo. Se puede ver que los trabajos están relacionados con el procesamiento de datos.

La tercera columna, "Submitted", muestra la fecha y hora en que se envió el trabajo. La fecha y hora de envío es la fecha y hora en que el trabajo se puso en cola para su ejecución.

La cuarta columna, "Duration", muestra la duración del trabajo. La duración es el tiempo que tardó el trabajo en ejecutarse. Las duraciones de los trabajos son cortas, lo que indica que los trabajos se ejecutaron rápidamente.

La quinta columna, "Succeeded/Total", muestra el número de tareas que se completaron con éxito y el número total de tareas del trabajo. En la imagen, todos los trabajos tienen una proporción de tareas completadas con éxito de 1/1, lo que indica que todas las tareas de los trabajos se completaron con éxito.

La sexta columna, "Stages:", muestra el número de etapas del trabajo. Las etapas son unidades lógicas de trabajo que se pueden ejecutar en paralelo. En la imagen, todos los trabajos tienen una sola etapa.

La séptima columna, "Tasks (for all stages):", muestra el número de tareas de todas las etapas del trabajo. Las tareas son unidades de trabajo que se ejecutan en una sola etapa. En la imagen, todos los trabajos tienen una sola tarea.

La octava columna, "Storage", muestra el almacenamiento utilizado por el trabajo. El almacenamiento utilizado es la cantidad de almacenamiento que utiliza el trabajo para almacenar sus datos y resultados. En la imagen, el almacenamiento utilizado por los trabajos es bajo.

La novena columna, "Environment", muestra el entorno en el que se ejecutó el trabajo. El entorno es el conjunto de recursos que utiliza el trabajo para ejecutarse. En la imagen, los entornos de los trabajos son "DBC/ODBC Server".

La décima columna, "Structured Streaming", muestra si el trabajo es un trabajo de streaming estructurado. Los trabajos de streaming estructurado son trabajos que procesan datos de streaming en tiempo real. En la imagen, todos los trabajos son trabajos de streaming estructurado.

La última columna, "ESP", muestra la hora de finalización del trabajo. La hora de finalización es la fecha y hora en que el trabajo se completó. Las horas de finalización de los trabajos son las mismas que las horas de envío, lo que indica que los trabajos se completaron rápidamente.

Resumen:

- La tabla muestra que los trabajos de Databricks son de streaming estructurado que se completaron rápidamente y sin errores.
- Los trabajos se ejecutaron en un entorno DBC/ODBC Server y utilizaron una cantidad baja de almacenamiento.
- Los trabajos tardaron entre 0,2 y 1,0 segundos en completarse.
- Los trabajos completaron una tarea cada uno.
- Los trabajos utilizaron entre 2,5 y 12,5 MB de almacenamiento.