"Análisis de Recomendaciones de Hoteles con Machine Learning y Cómputo Paralelo"

Ignacio Cortés Atzin Maxela, Ríos Rivera Fernanda Anahí
Ingeniería en Inteligencia Artificial, Escuela superior de Cómputo.
Instituto Politécnico Nacional.

RESUMEN

Este proyecto presenta un análisis de datos masivos aplicando técnicas de aprendizaje automático y cómputo paralelo. Utilizando un conjunto grande de datos de recomendaciones de hoteles de Expedia, se entrenó un modelo de clasificación y se optimizaron los tiempos de procesamiento mediante herramientas como **PySpark, multiprocessing y PyCUDA.**

OBJETIVO

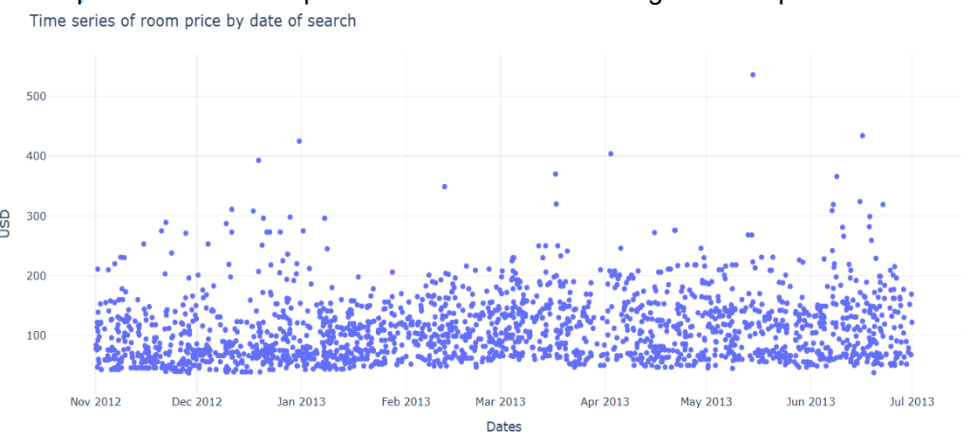
Demostrar cómo el uso combinado de estas tecnologías permite analizar grandes volúmenes de información de forma eficiente y escalable, con aplicaciones potenciales en el sector turístico y otros ámbitos donde se generan grandes cantidades de datos.

INTRODUCCIÓN

El avance de la tecnología ha generado enormes cantidades de datos en sectores como el turismo, donde millones de usuarios consultan y reservan hoteles en línea. Analizar esta información permite descubrir patrones y tomar decisiones más inteligentes. Sin embargo, procesar grandes volúmenes de datos representa un reto técnico. En este proyecto se combinan técnicas de análisis de datos y cómputo paralelo para agilizar este proceso, utilizando herramientas como Spark (para datos distribuidos), PyCUDA (para procesamiento en GPU) y algoritmos de aprendizaje automático como Random Forest.

EJEMPLO

Comportamiento de los precios de los hoteles a lo largo del tiempo.



DESARROLLO

Se utilizó el lenguaje Python y un dataset de reseñas de hoteles obtenido en Kaggle. El análisis se dividió en las siguientes etapas:

- **Carga y limpieza de datos:** usando pandas y PySpark para manejar el volumen masivo de datos.
- **Análisis exploratorio:** visualizaciones con seaborn y matplotlib para identificar relaciones y patrones.
- **Clasificación:** entrenamiento de un modelo Random Forest para predecir el grupo de hotel más probable.
- **Optimización con cómputo paralelo:**
  - *multiprocessing* y *concurrent.futures* para paralelizar tareas en CPU.
  - *PySpark* para operaciones distribuidas.
  - *PyCUDA* para acelerar cálculos vectorizados en GPU.

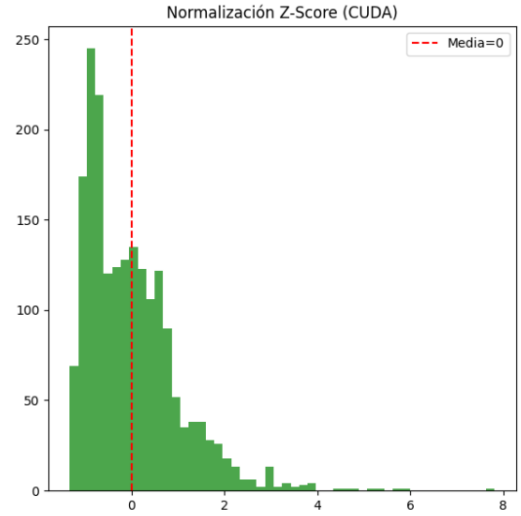
Se cargaron **1,953 registros filtrados** desde el archivo train.csv usando **Spark**, específicamente búsquedas hechas por usuarios del país 219, en un hotel con ID 104517, y con precios válidos.

Esto indica que el dataset final es **pequeño**, lo cual influye en el rendimiento de los métodos paralelos.

Normalización Z-Score (CUDA)

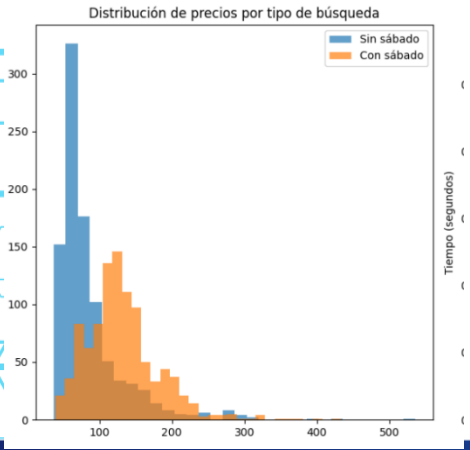
Normalización Z-Score

La mayoría de los valores están a la izquierda del eje (negativos), lo cual indica que **muchos precios están por debajo del promedio**.
La línea roja punteada marca la media (0 después de la normalización), que es la referencia estándar en Z-Score.
El uso de CUDA permite que esta transformación se aplique **rápidamente y en paralelo**, incluso con millones de precios.



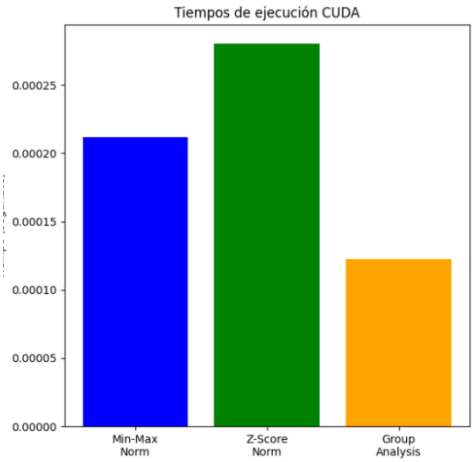
Distribución de precios por tipo de búsqueda

Los precios sin sábado están más concentrados en valores bajos (pico más alto a la izquierda).
Esto sugiere que incluir el sábado en la estancia eleva el costo promedio del hospedaje, probablemente por mayor demanda en fines de semana.



Tiempos de ejecución CUDA

El análisis por grupos fue el más rápido (menos carga computacional).
Las normalizaciones tardaron un poco más, especialmente la Z-Score por involucrar media y desviación estándar.
Aun así, todos los procesos fueron completados en milisegundos, lo cual demuestra la eficiencia del cómputo paralelo en GPU.



También se comparó la normalización intensiva de 6 millones de datos entre CPU y GPU usando PyCUDA, midiendo tiempo, precisión y consumo de memoria para evaluar el beneficio del cómputo paralelo en GPU.
La GPU ejecuta la normalización mucho más rápido y con menor uso de memoria que la CPU, demostrando las ventajas del cómputo paralelo en PyCUDA para grandes volúmenes de datos.

```
[GPU] Normalización optimizada...

============================================================
RESULTADOS COMPARATIVOS CPU vs GPU
============================================================
CPU - Tiempo:      0.3605 s
GPU - Tiempo:      0.0074 s
Speedup GPU/CPU:   48.85x
Error máximo:      4.172325e-07

RAM usada en CPU:    191.94 MB
VRAM usada en GPU:   48.0 MB

Min GPU: 0.000000
Max GPU: 0.891327
Media GPU: 0.448839
Resultados consistentes
```

CONCLUSIÓN

En este proyecto se integraron técnicas avanzadas de procesamiento paralelo con algoritmos de aprendizaje automático, lo cual permitió trabajar con grandes volúmenes de datos de manera eficiente.
Este aprendizaje se traduce en beneficios directos para los usuarios: Se ofrecen respuestas más rápidas, precisas y personalizadas, incluso cuando se manejan grandes cantidades de datos.