Universidad Tecnológica de Panamá

Maestría en Analítica de Datos

Curso:

Modelos Predictivos

Avances del Proyecto Final:

Análisis y Predicción del Consumo de Recursos en Entornos de Computación en la Nube.

Facilitador:

Juan Marcos Castillo, PhD

De Gracia, Evaristo 8-788-1186

Año

2025

Introducción

Los modelos predictivos son una herramienta poderosa que permite a usuarios y empresas realizar evaluaciones sobre sus sistemas para identificar información valiosa que les apoye en la toma de decisiones de una manera oportuna.

Durante el desarrollo de este proyecto estaré aplicando los conocimientos adquiridos en la materia de modelos predictivos de tal forma que me ayude a analizar y determinar de una manera más exacta que modelo se adecua más al estudio que quiero realizar.

El tema que he escogido para mi proyecto es la evaluación de los recursos al utilizar computación en la nube. La computación en la nube tiene la ventaja que dependiendo de la necesidad te permite escalar en recursos, pero si no se utilizan de una manera óptima y eficiente pueden llevar a generar costos innecesarios a las empresas.

Justificación

Actualmente me encuentro trabajando en el área de ingeniería de datos y dentro de mi último proyecto laboral nos encontramos en la migración de un sistema local a computación en la nube. Este proyecto surge de la necesidad de predecir y gestionar eficientemente el consumo de recursos en un entorno dinámico, donde la sobre provisión o subutilización de capacidades impacta directamente en la facturación operativa y la calidad del servicio. Con este análisis mi meta es poder entender y crear modelos predictivos que ayuden a la toma de decisiones y presentar un informe que sirva como base para la configuración y utilización de recursos en la nube enfocado en los costos eficientes.

Antecedentes

La computación en la nube ha experimentado un auge en la última década, cambiando la manera como las organizaciones manejan sus recursos tecnológicos, esto ha permitido reducir los costos operativos a las empresas ya que pueden delegar el aprovisionamiento de sus recursos computacionales a proveedores dedicados.

En este contexto, la aplicación de técnicas de modelado predictivo se ha convertido en una herramienta fundamental para las organizaciones que buscan maximizar la inversión de sus recursos en la nube. Al analizar el comportamiento histórico de sus datos como lo son consumo de CPU, memoria y/o GPU, estos modelos pueden ser utilizados de manera proactiva en vez de reactivo.

El objetivo de estos modelos y de nuestro proyecto es lograr un equilibrio entre costo-eficiencia para garantizar que los recursos asignados a un proyecto sean los correctos eliminando gastos innecesarios.

1. **Definición de Problema**

Con nuestro dataset actual queremos predecir los tiempos de ejecución en función del uso de recursos computacionales como CPU, memoria, GPU y como se relacionan las variables con la eficiencia de los recursos.

2. **Determinación de la base de datos**

La base de datos utilizada para este proyecto se extrajo del repositorio público de Alibaba: clusterdata/cluster-trace-gpu-v2023 at master · alibaba/clusterdata. Esta base de datos, como se describe en el archivo README adjunto (clusterdata/README.md at master · alibaba/clusterdata), contiene trazas de clústeres de GPU de Alibaba. Para este análisis predictivo, se tomó como base la información correspondiente al año 2023, proporcionando un conjunto de datos relevante y actualizado para la modelización.

Variables independientes:

- cpu_milli: Uso de CPU en milinúcleos.
- memory_mib: Memoria utilizada en mebibytes.
- num_gpu: Número de GPUs asignadas.
- gpu_milli: Tiempo de uso de GPU en milisegundos.

3. **Pre-procesamiento y limpieza**

La primera parte del proyecto consistió en consolidar todos los csv del periodo 2023 que tengan atributos relacionados a las variables dependientes. Para esta consolidación se utilizó el archivo notebook\utils\concatenate.py.

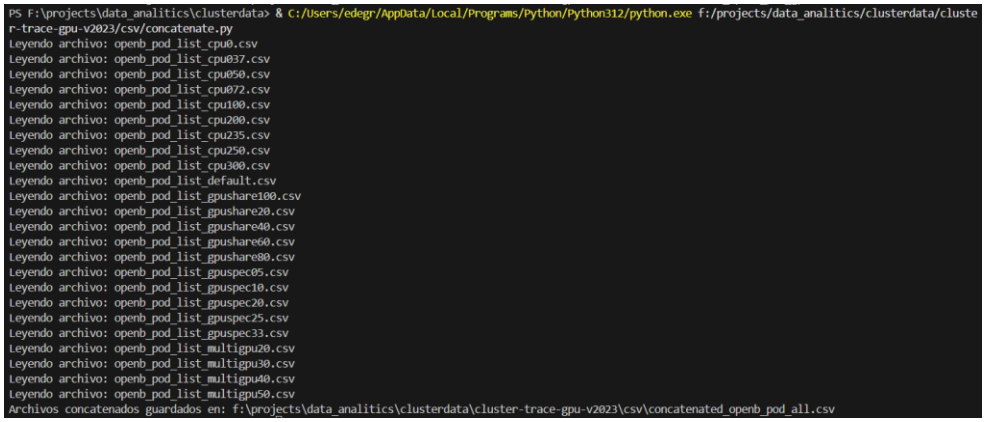


Figura 1: Consolidación de archivos csv.

En esta etapa, los datos extraídos fueron sometidos a un proceso de pre-procesamiento y limpieza. Esto incluyó las siguientes acciones

- Filtrado de datos: se identificó que los datos contenía diferentes estatus que incluían: fallidos, en ejecución, exitosos y no iniciados. Como el enfoque de nuestro estudio es el consumo de recursos, filtramos nuestros datos a los siguientes estatus: fallidos y exitosos. Esto nos dejó con un total de 42,615 registros de un total de 199,197. Esta decisión se tomó por los siguientes motivos:
  Los procesos en ejecución tenían un tiempo de duración muy corto.
  Los procesos que no habían iniciado no tenían tiempos de duración.

- Análisis de Correlación de Variables: Se utilizó este análisis para identificar las variables que se relacionaban entre sí para nuestro modelo. En este análisis se identifcaron las siguientes variables que se relacionan entre sí: cpu, memoria, numero de gpu.

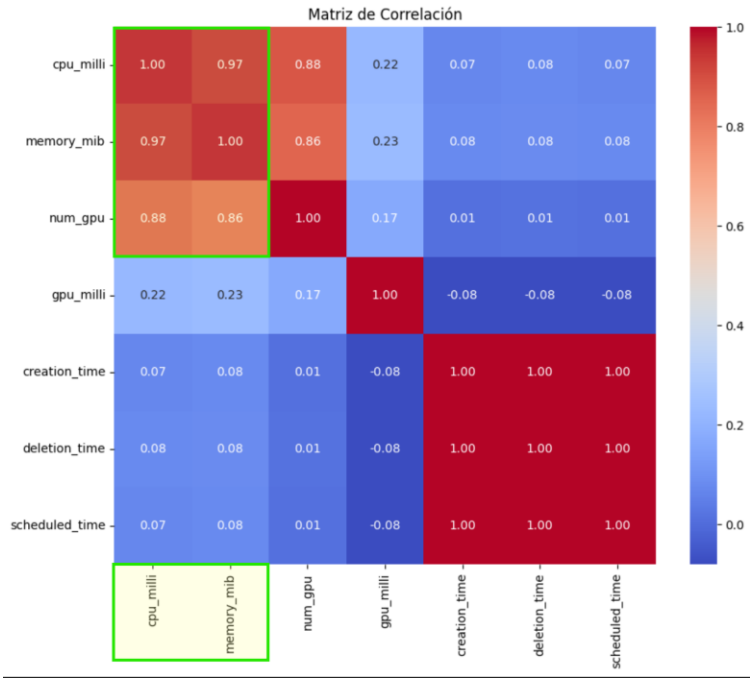


Figura 2: Correlación de variables.

- Validación de duplicados: No se identificaron duplicados a nivel de todas las columnas.

```
# Mantener la primera versión de un registro duplicado en un nuevo dataframe
no_dups_df = filtered_data.drop_duplicates(keep='first', inplace=False)

# Contar duplicados
dups_rows = filtered_data[filtered_data.duplicated(keep='first')]

# Calcular el porcentaje de duplicados
porcentaje_dups = (dups_rows.shape[0] / filtered_data.shape[0]) * 100

# Imprimir resultados:
print(f'Cantidad de Registros Originales: {filtered_data.shape[0]}')
print(f"Cantidad de Registros duplicados: {dups_rows.shape[0]}")
print(f"Porcentaje de duplicados sobre la muestra original: {porcentaje_dups:.2f}%")
print(f"Cantidad de Registros sin dups  : {no_dups_df.shape[0]}")
```

✓ 0.0s

```
Cantidad de Registros Originales: 42615
Cantidad de Registros duplicados: 0
Porcentaje de duplicados sobre la muestra original: 0.00%
Cantidad de Registros sin dups  : 42615
```

Figura 3. Validación de duplicados.

- Creación de nuevas columnas:
    - Conversión de cpu_milli, gpu_milli y memory_mib se realizó para expresar el uso de recursos en una unidad más interpretable.
    - total_usage se creó para tener una medida agregada del consumo de total de recursos (cpu y memoria)
    - La variable cpu_gpu_ratio se calculó para analizar cómo se balancea el uso de CPU y GPU, lo que puede proporcionar información sobre la eficiencia de las tareas.

```
# Convierte miliCPU a cores
data["cpu_cores"] = data["cpu_milli"] / 1000
# Convierte miliGPU a cores
data["gpu_cores"] = data["gpu_milli"] / 1000
# Convertir MiB a GiB
data['memory_gbs'] = data['memory_mib'] / 1024
# Calcula el uso total de CPU y GPU en cores
data["total_usage"] = data["cpu_cores"] + data["gpu_cores"]
# Calcula la relación entre el uso de CPU y GPU (evitando la división por cero)
data["cpu_gpu_ratio"] = data["cpu_cores"] / (data["gpu_cores"] + 1e-6)
```

Figura 4: Nuevas variables para el análisis.

4. **Análisis descriptivo:**

Se llevó a cabo un análisis descriptivo exhaustivo para comprender las características principales del conjunto de datos de 2023. Puntos relevantes del análisis:

- El 75% de las instancias usan hasta 4 Cores, pero hay casos con hasta 120 Cores (posibles cargas críticas o errores de configuración). Esto nos indica que puede haber sobre provisionamiento de recursos que pueden elevar los costos.
- En memoria la media es 19 Gbs pero con valores extremo de 720 Gbs lo cual muestra un similar comportamiento al uso del CPU.

```
data[['cpu_cores', 'memory_gbs','gpu_cores','total_usage', 'cpu_gpu_ratio']].describe()
```

✓ 0.0s Python

	cpu_cores	memory_gbs	gpu_cores	total_usage	cpu_gpu_ratio
count	42597.000000	42597.000000	42597.000000	42597.000000	4.259700e+04
mean	6.238897	18.672637	0.848055	7.086952	6.095022e+04
std	11.314430	50.143444	0.154221	11.349417	1.138557e+06
min	3.152000	5.468750	0.000000	3.742000	3.151997e+00
25%	3.152000	5.468750	0.810000	3.962000	3.891353e+00
50%	3.152000	5.468750	0.810000	3.962000	3.891353e+00
75%	4.000000	10.351562	1.000000	4.522000	3.999996e+00
max	120.200000	720.000000	1.000000	121.200000	3.200000e+07

Figura 5: Análisis descriptivo.

5. Visualizaciones

Se generaron gráficos y visualizaciones para explorar las distribuciones de las variables individuales (histogramas, diagramas de caja), las relaciones entre pares de variables (diagramas de dispersión) y las posibles tendencias o patrones en los datos.

Luego de la primera distribución de los datos nos percatamos que las variables que nos interesaban (memoria y cpu) estaban sezgadas a la derecha. Se utilizó logaritmo natural de 1 (log1p) para evitar errores y mitigar el impacto de los outliers.

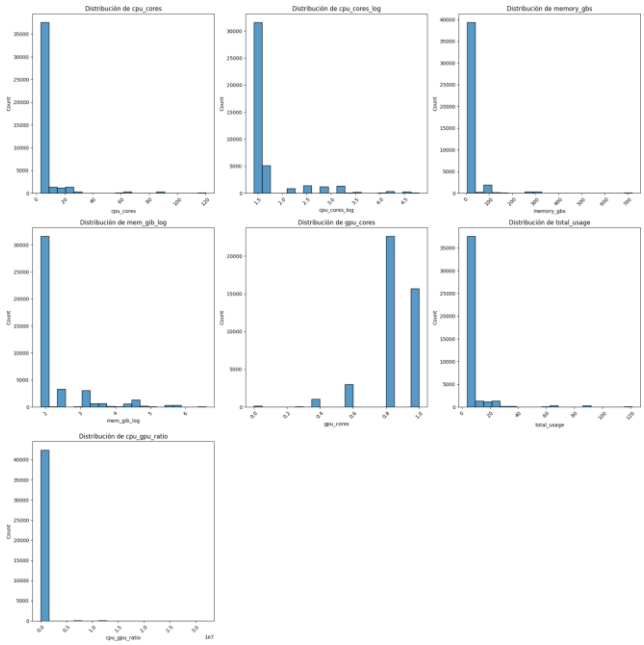


Figura 6: Distribución de variables incluyendo logaritmo natural de 1 para outliers.

Al aplicar el logaritmo natura, también se puede apreciar una mejor distribución en los diagramas de cajas.

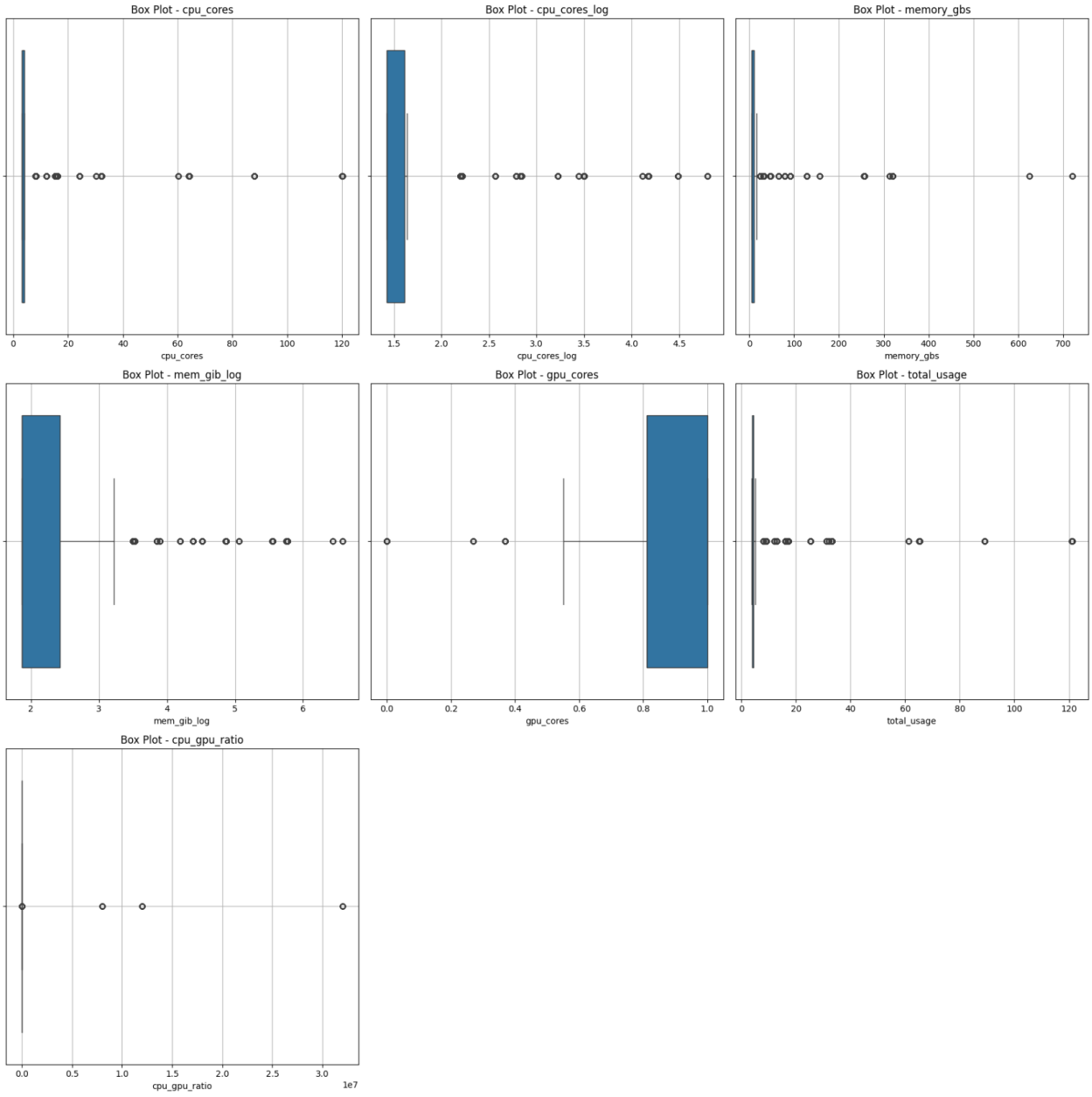


Figura 7: Diagrama de caja

6. Selección de variables:

La selección de las variables predictoras más relevantes fue un paso crucial para construir modelos predictivos eficientes y precisos. Se aplicaron diversas técnicas para identificar y seleccionar las variables que tienen una mayor influencia en la variable objetivo. Estas técnicas pudieron incluir:

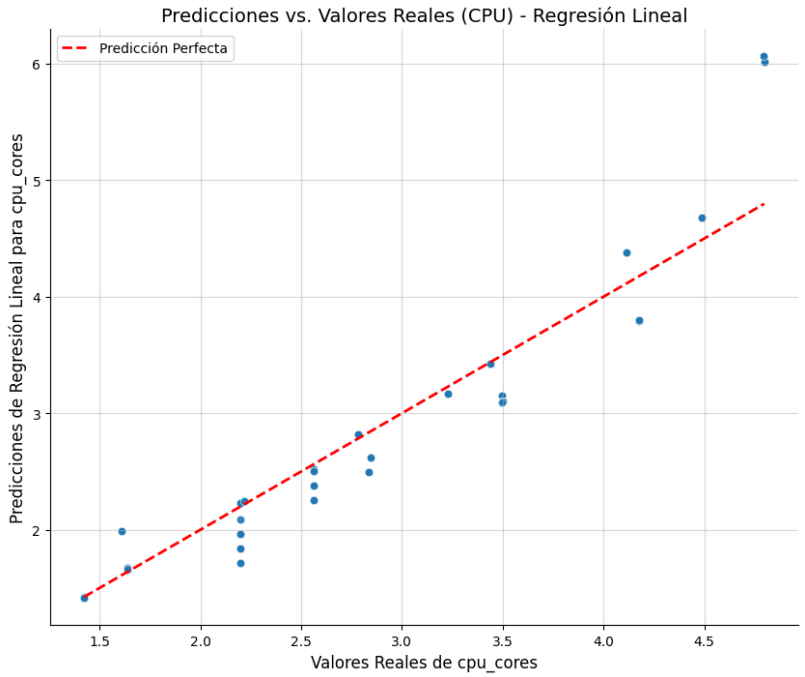
- Análisis de correlación: Se calcularon las correlaciones entre las variables predictoras y la variable objetivo para identificar aquellas con una relación lineal más fuerte.
- Aplicar logaritmo natural: Con este método pudimos aplanar los datos y evitar sesgos en la distribución.
- Conocimiento del dominio: Se consideró el conocimiento del dominio de los datos del clúster para seleccionar variables que teóricamente tienen una influencia en la variable objetivo.

La selección de variables se realizó con el objetivo de maximizar la capacidad predictiva del modelo, reducir la complejidad, evitar el sobreajuste y mejorar la interpretabilidad.

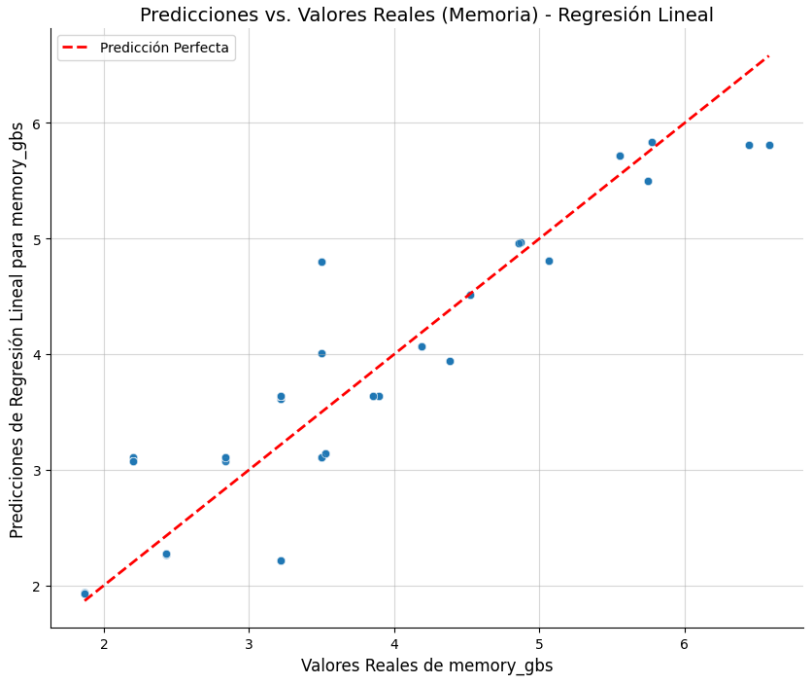
7. Selección de Modelos:

En esta etapa, se consideraron y evaluaron diversos modelos de aprendizaje automático para la tarea de predicción específica. La selección de los modelos se basó en la naturaleza de la variable objetivo (continua o categórica), las características del conjunto de datos y los objetivos del proyecto. Algunos de los modelos que pudieron haber sido considerados incluyen:

- Modelos de regresión: Iniciamos nuestro análisis con el modelo de regresión lineal en donde evaluamos las variables de cpu y memoria. Nuestros primeros resultados fueron prometedores para el uso de cpu, pero con la variable de la memoria no fue muy preciso nuestro modelo.



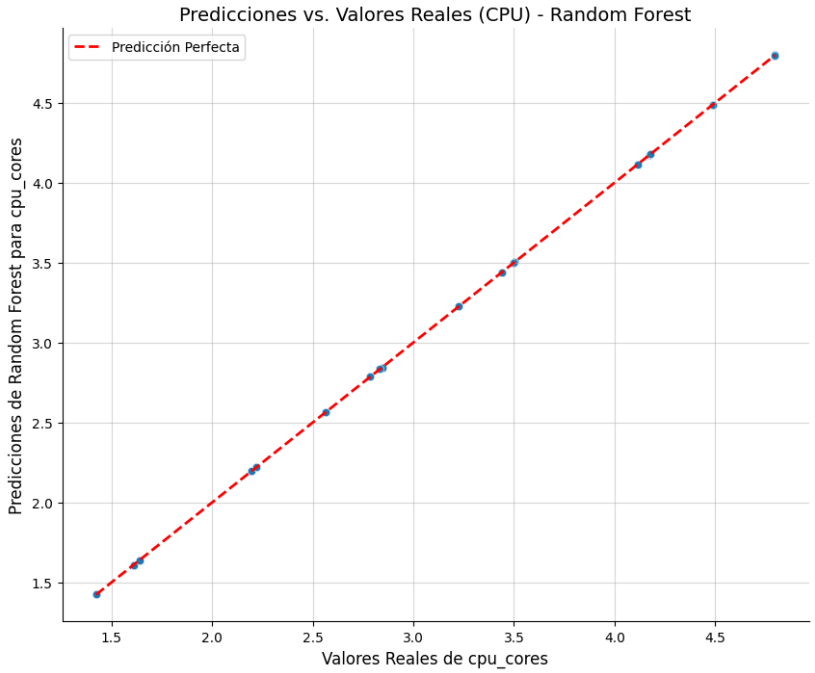
Error Cuadrático Medio (MSE): 0.0175 R^2: 0.9514



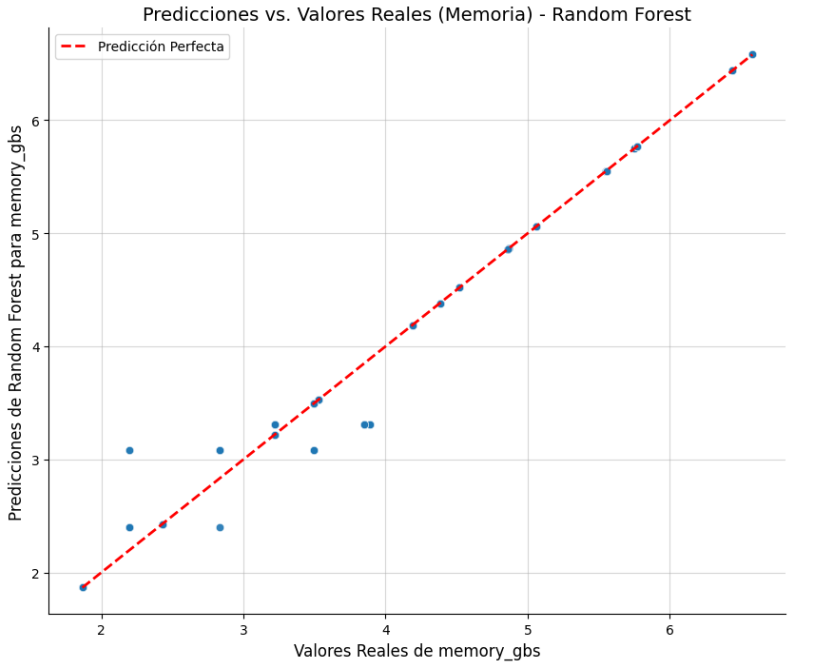
Error Cuadrático Medio (MSE): 0.0676 R^2: 0.9109

- Random Forest:

  Al utilizar el algoritmo de random forest, el modelo fue más preciso para ambas variables.



Error Cuadrático Medio (MSE): 6.0749 R^2: 1.0



Error Cuadrático Medio (MSE): 0.00191 R^2: 0.9974

8. Comparación de modelos

Memoria	**Regresión Lineal**	**Random Forest**
Error Cuadrático Medio (MSE)	0.068	0.002
R^2	0.911	0.997

CPU	**Regresión Lineal**	**Random Forest**
Error Cuadrático Medio (MSE)	0.018	**0.000**
R^2	**0.951**	1.000

Como podemos observar en la gráfica tabular nuestro modelo de Random Forest fue más preciso que nuestro modelo de Regresión Lineal.

Conclusiones

Durante el desarrollo de este proyecto pude aplicar los conocimientos adquiridos en la materia y entender un poco mas como funcionan los algoritmos de modelos predictivos. Es importante recalcar que el trabajo previo en la limpieza de datos, manejo de outliers y entender las estadísticas básicas es lo que nos puede ayudar a mejorar significativamente la precisión y la confiabilidad de nuestros modelos. Este proceso iterativo de exploración, limpieza y análisis es lo que nos puede ayudar a mejorar la precisión de nuestros modelos y, en última instancia, proporcionar información valiosa para la gestión y optimización de estos sistemas complejos o cualquier otro modelo que quisiéramos aplicar a futuro.

BIBLIOGRAFÍA

Alibaba Group. (2023). *cluster-trace-gpu-v2023* [Conjunto de datos]. GitHub. Recuperado de https://github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2023

ANEXOS

Campos en el dataset

- cpu_milli: Uso de CPU en milinúcleos.
- memory_mib: Memoria utilizada en mebibytes.
- num_gpu: Número de GPUs asignadas.
- gpu_milli: Tiempo de uso de GPU en milisegundos.

Repositorio en github: edegraciab/ma-modelos-predictivos-poyecto-final: Proyecto Final de la materia Modelos Predictivos dentro de la Maestría en Analítica de Datos