3. Consulte en qué consiste y el funcionamiento principal de la librería RAPIDS. Posteriormente, elabore una tabla que compare los métodos e hiperparámetros más relevantes de los regresores mencionados en el punto 2, indicando su implementación o el algoritmo equivalente disponible en RAPIDS.

* Descripción general librería RAPIDS

RAPIDS es un ecosistema de librerías de código abierto desarrollado por NVIDIA que permite realizar flujos completos de data science y machine learning sobre GPU, aprovechando la arquitectura CUDA (Compute Unified Device Architecture), aprovechando para acelerar el procesamiento.

Su objetivo principal es reducir drásticamente los tiempos de ejecución de tareas como la manipulación de datos, entrenamiento de modelos y predicción, manteniendo una API muy similar a la de scikit-learn y pandas, lo cual facilita su adopción

* Componentes principales de RAPIDS

Módulo	Funcionalidad principal	Equivalente en CPU
cuDF	Manipulación de datos tipo dataframe en GPU	Pandas
cuML	Implementación de modelos de machine learning en GPU	scikit-learn
cuGraph	Algoritmos de grafos y redes sobre GPU	NetworkX
cuIO	Lectura y escritura de archivos optimizada para GPU	Pandas .read_*
RMM	Gestión eficiente de memoria en GPU	—

Rapios permite que los datos permanezcan completamente en la GPU desde su carga hasta el modelado, evitando transferencias costosas entre CPU y GPU. Esto permite aceleraciones entre 10x y 50x frente a implementaciones tradicionales en CPU.

* Comparación de regresores entre scikit-learn y RAPIDS

La siguiente tabla resume los regresores solicitados en el punto 2, sus hiperparametros más relevantes y sus implementaciones o equivalentes en RAPIDS (cuML)

Regresor (scikit-learn	Implementación / equivalente en RAPIDS	Principales Hiperparametros	Comentario
LinearRegression	cuml.LinearRegression	fit_intercept, normalize, algorithm (QR o SVD)	Misma API que sklearn, total en GPU
Lasso	cuml.Lasso	alpha, fit_intercept, max_iter, tol	Basado en

/ /

Elastic Net	cuml.ElasticNet	alpha, l1_ratio, tol, fit_intercept, max_iter	combina regularización L1 y L2 en GPU
Kernel Ridge	(no disponible directamente)	—	Puede aproximarse con cuml.SVR(kernel='rbf') o CuPy
SGDRegressor	cuml.SGD	alpha, learning_rate, eta0, max_iter, penalty	Implementación GPU del descenso estocástico
Bayesian Ridge	(no disponible directamente)	—	Se puede aproximar con librerías GPU como PyMC o JAX
Gaussian Process-Regressor	(no disponible en cuML)	—	Puede implementarse con GPyTorch en GPU
SVR (Support Vector Regressor)	cuml.SVR	C, epsilon, kernel, gamma, degree, coef0	Implementación CUDA completa, gran ganancia en velocidad
RandomForest-Regressor	cuml.RandomForestRegressor	n_estimators, max_depth, max_features, min_samples_split	Entrenamiento y predicción paralelos en GPU
GradientBoosting-Regressor / XGBoost	cuml.XGBoost o xgboost.XGBRegressor(tree_method='gpu_hist')	n_estimator, learning_rate, max_depth, subsample, colsample_bytree	XGBoost soporta GPU nativamente; cuML ofrece wrapper compatible

→ Rapids cuML mantiene una interfaz prácticamente idéntica a scikit-learn, lo que permite migrar código con cambios mínimos

→ Los mayores beneficios se observan en datasets grandes (más de 1 millón de muestras), donde la aceleración por GPU es más notoria

→ Algunos modelos probabilísticos y no lineales (como Gaussian Processes o Bayesian Ridge) aún no cuentan con implementación directa, pero pueden abordarse con librerías complementarias que aprovechen GPU, como GPyTorch, TensorFlow Probability o PyMC

→ Para procesamiento distribuido o uso de múltiples GPUs, RAPIDS ofrece Dask-cuML, que permite escalar horizontalmente las tareas de entrenamiento