Cuadro 1: Comparativa: objetivo, hiperparámetros y equivalentes GPU (cuML / XGBoost).

Regresor	**Objetivo** $J(\beta, b)$ **(con** b ***no*** **regularizado)**	**Hiperparámetros clave**	**Implementación en RAPIDS**
LinearRegression (OLS)	$J = \frac{1}{2n} \left\lVert y - X\beta - b\mathbf{1} \right\rVert_2^2$	`fit_intercept`	**cuML** `LinearRegression`
Ridge	$J = \frac{1}{2n} \left\lVert y - X\beta - b\mathbf{1} \right\rVert_2^2 \; + \; \lambda \lVert\beta\rVert_2^2$	`alpha` ($\equiv \lambda$), `fit_intercept`	**cuML** `Ridge`
Lasso	$J = \frac{1}{2n} \left\lVert y - X\beta - b\mathbf{1} \right\rVert_2^2 \; + \; \lambda \lVert\beta\rVert_1$	`alpha` ($\equiv \lambda$), `max_iter`, `tol`, `fit_intercept`	**cuML** `Lasso`
ElasticNet	$J = \frac{1}{2n} \left\lVert y - X\beta - b\mathbf{1} \right\rVert_2^2 \; + \; \lambda\Big((1-\rho)\lVert\beta\rVert_2^2 \; + \; \rho\lVert\beta\rVert_1\Big), \; \rho \in [0,1]$	`alpha` ($\equiv \lambda$), `l1_ratio` ($\equiv \rho$), `max_iter`, `tol`, `fit_intercept`	**cuML** `ElasticNet`
KernelRidge (KRR)	$J(f) \; = \; \frac{1}{2n} \sum_{i=1}^{n} \Big(y_i \; - \; f(x_i)\Big)^2 \quad + \quad \lambda \lVert f\rVert_{\mathcal{H}}^2, \quad \text{con } f \; \in \; \mathcal{H}_k \text{ y } f(\cdot) \; = \; \sum_i \alpha_i k(x_i, \cdot)$	`alpha` ($\equiv \lambda$), `kernel` (rbf/linear/poly), `gamma`, `degree`, `coef0`	**cuML** `KernelRidge`
SGDRegressor	$J = \frac{1}{2n} \sum_{i=1}^{n} \Big(y_i - x_i^\top \beta - b\Big)^2 \; + \; \lambda\,\Omega(\beta), \; \Omega \in \{\lVert\cdot\rVert_1, \lVert\cdot\rVert_2\}$	`loss`, `alpha` ($\equiv \; \lambda$), `penalty` (L1/L2/EN), `learning_rate`, `max_iter`, `tol`	**cuML** `SGD`
GaussianProcessRegressor	$\max\limits_{\theta, \sigma^2} \; \log p(y \mid X, \theta, \sigma^2) = -\frac{1}{2} y^\top (K_\theta + \sigma^2 I)^{-1} y - \frac{1}{2} \log \lvert K_\theta + \sigma^2 I\rvert - \frac{n}{2} \log(2\pi)$	`kernel` (k_θ), `alpha` ($\equiv \quad \sigma^2$), `n_restarts_optimizer`	**cuML** `GaussianProcessRegressor`
SVR (ε-insensitive)	$\min\limits_{w,b,\xi,\xi^*} \; \frac{1}{2}\lVert w\rVert_2^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \text{ s.a. } \begin{cases} y_i - \langle w, \phi(x_i)\rangle - b \le \varepsilon + \xi_i \\ \langle w, \phi(x_i)\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$	`C`, `epsilon`, `kernel`, `gamma`, `degree`, `coef0`	**cuML** `SVR`
RandomForestRegressor	En cada nodo $S \to \{S_L, S_R\}$: $\max\limits_{s} \; \Delta(s) \; = \; \mathrm{Var}(S) - \frac{\lvert S_L\rvert}{\lvert S\rvert}\mathrm{Var}(S_L) - \frac{\lvert S_R\rvert}{\lvert S\rvert}\mathrm{Var}(S_R)$ (reducción MSE, promediado en T árboles)	`n_estimators`, `max_depth`, `max_features`, `min_samples_leaf`, `bootstrap`	**cuML** `RandomForestRegressor`
GradientBoostingRegressor	$\min\limits_{F} \; \sum_{i=1}^{n} \ell\Big(y_i, \; F(x_i)\Big), \quad F_m = F_{m-1} + \nu\, h_m, \quad h_m$ ajusta residuos $-\nabla_F \ell$	`learning_rate` (ν), `n_estimators`, `max_depth`, `subsample`	**XGBoost** `Regressor` (`tree_method=gpu_hist`)