Best Practices in Feature Engineering for Tabular Data with GPU Acceleration

by Chris Deotte & Ronay Ak

Goals

- **Improve model performance for Tabular datasets**
  - Learn and apply best practices for data preprocessing and feature engineering

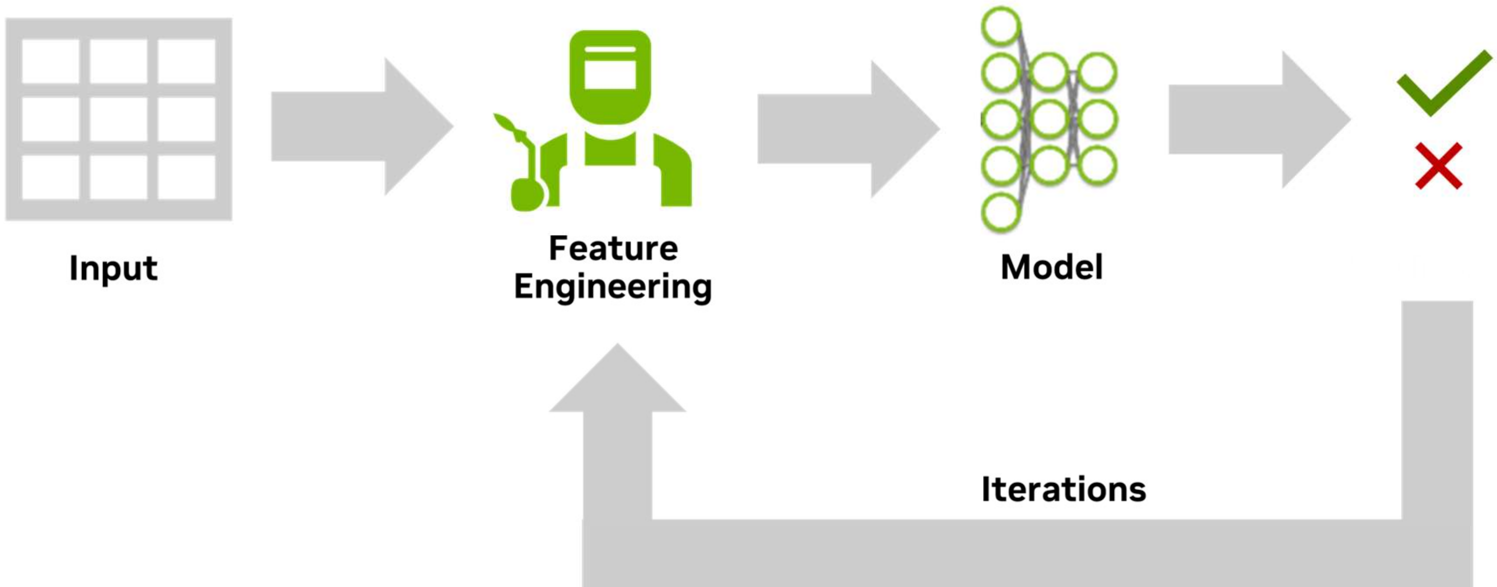
- **Accelerate training pipelines**
  - Leverage GPUs acceleration for quick experimentation

Agenda

- **Experimentation Pipeline for Tabular Datsets**
  - Acceleration is Important for Tabular Datasets
  - Overview of Feature Types
  - Accuracy Improvements with Target Encoding

- **NVIDIA RAPIDS: Accelerating Data Science End-to-End**
  - NVIDIA cuDF: Automatic pandas Acceleration
  - NVIDIA cuML: Accelerated scikit-learn

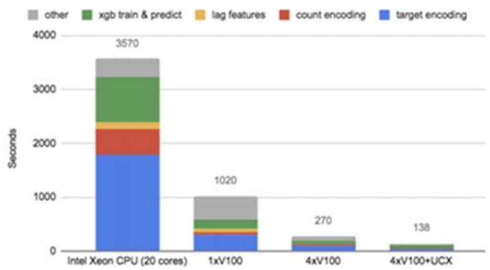
Experimentation Pipeline for Tabular Datasets



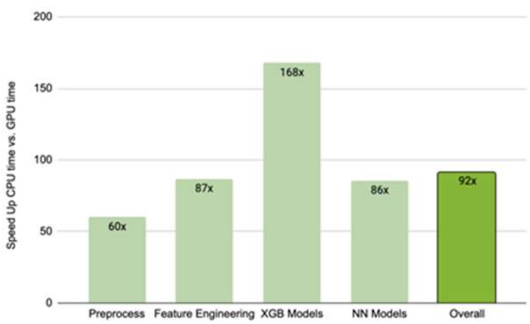
Developing high accuracy model requires a cycle to run multiple experiments to iterate on feature engineering, model types and architectures.

Accelerating pipelines enables more experiments

NVIDIA @RecSys Challenge 2020

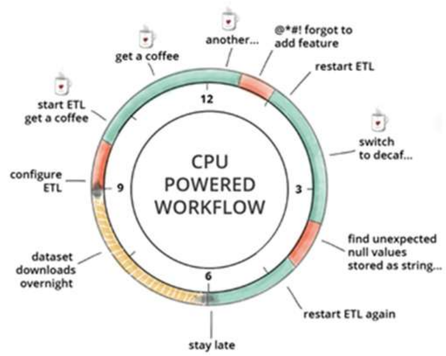


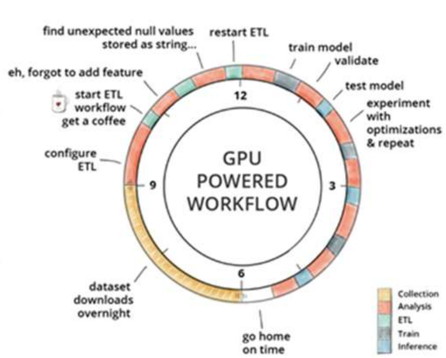
NVIDIA @RecSys Challenge 2021



Source: https://medium.com/rapids-ai/rapids-accelerates-data-science-end-to-end-afda1973b65d
Source: https://github.com/NVIDIA-Merlin/competitions/tree/main/RecSys2021_Challenge
Source: https://github.com/NVIDIA-Merlin/competitions/tree/main/RecSys2020_Challenge

Illustrative day as a data scientist w/wo GPU acceleration





- RecSys2020 - 28x faster than optimized CPU code
- RecSys2021 - 92x faster than initial CPU code
- Less computation time enables more experiments

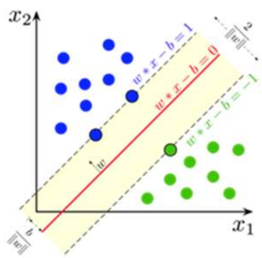
Overview Feature Types

Bold techniques in focus

Feature Type	Example	Feature Engineering
Categorical	User ID / Item ID Brand Main Category	Target Encoding Count Encoding Categorify + Combining Categories
Unstructured list	Keywords Subcategories Colors	Target Encoding Count Encoding Categorify
Numeric	Price Deliver time Avg. reviews	Binning Normalization Gauss Rank
Timestamp	Timestamp	Extract month, weekday, weekend, hour
Timeseries	Events in order Time since last event	# of events in past X Difference in time (lag)
Image	Product image	Extract latent representation with deep learning
Text	Description	Extract latent representation with deep learning Term frequency-inverse document frequency
Social graph	Follower/Following graph	Link analysis
Geo location	Addresses	Distances to point of interest

Overview Some Models

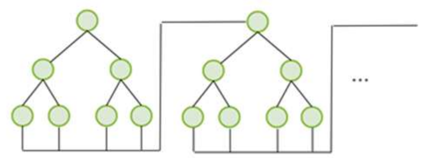
Support Vector Machines



Types:
- SVM

Components:
- Maximizes distance between decision boundary and data
- May require normalization
- Good for high dimensional data

Tree Based

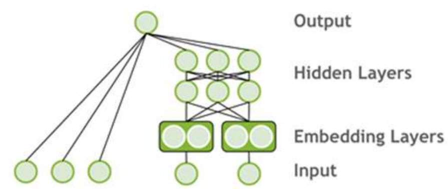


Types:
- CatBoost
- XGBoost
- LightGBM

Components:
- Defines split by information gain per feature
- Does not require normalization of input features

XGBoost cannot handle raw categorical features

Deep Learning



Types:
- Wide And Deep
- DeepFM
- DLRM

Components:
- Embedding Layers
- Feed-Forward Layers
- Requires normalization of input features

Dataset of the Tutorial

Dataset: Amazon Review Dataset - Category Electronics
URL: https://jmcauley.ucsd.edu/data/amazon/

Events: Ratings
Timeframe: Jun 1999 - July 2014

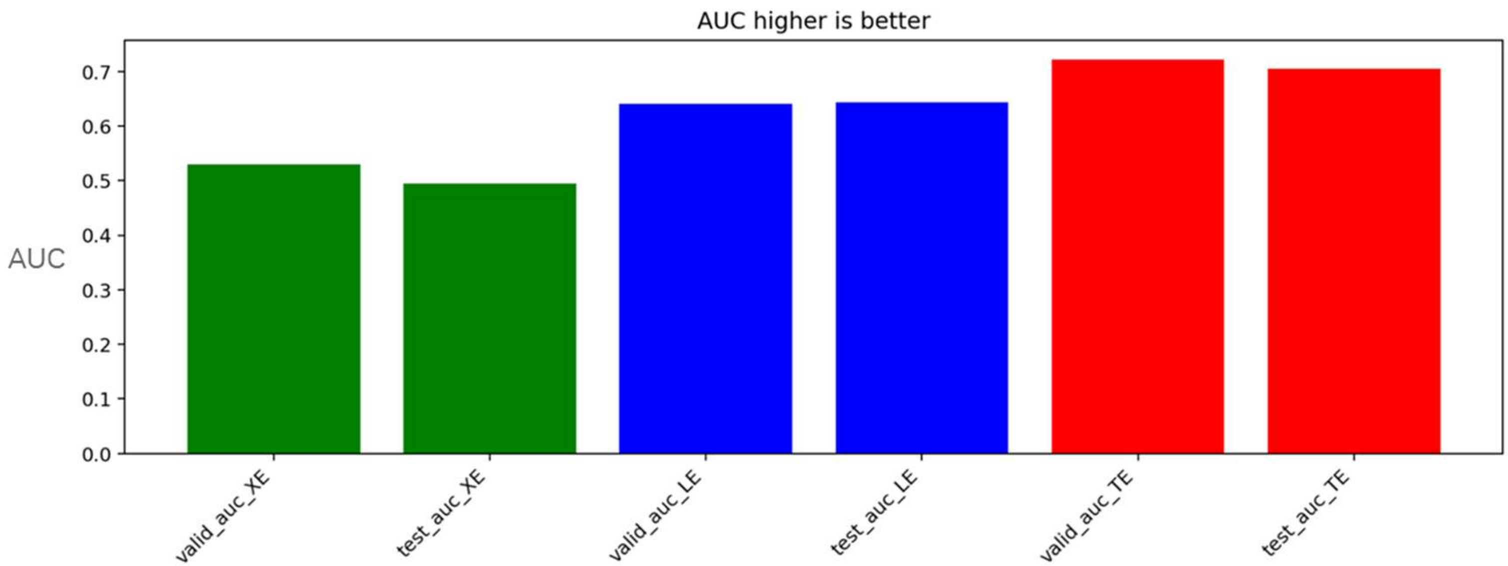
Goal:
- **Positive target**: Rating>=4
- **Negative target**: Rating<=3

Dataset split:
- **Training**: June-1999 - May-2014 (1.6 Mio samples)
- **Validation**: June-2014 (43k samples)
- **Test**: July-2014 (33k Mio samples)

Baseline: ~80% of events are high ratings

Features:
- userID, productID
- price
- timestamp
- category
- brand

Performance improvement of 25% with Target Encoding



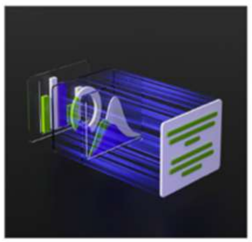
- XE: XGB Built-In Enable Categorical
- LE: Label encoding
- TE: Target Encoding

GPU Computing & NVIDIA CUDA-X for Data Processing

Modern Applications Need Accelerated Computing

Petabye scale data | Massive models | Real-time performance

Recommenders



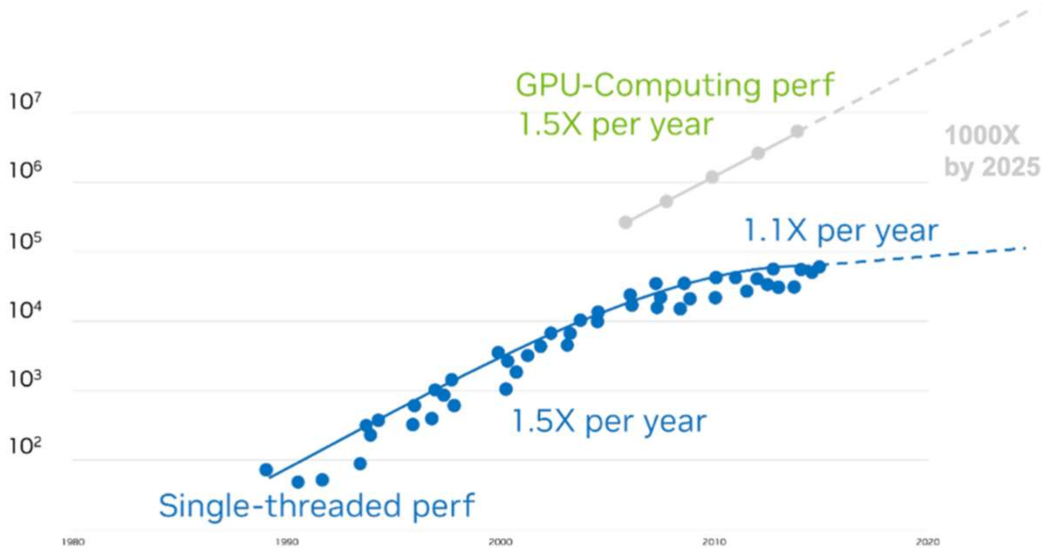
LLMs

Forecasting

Fraud Detection

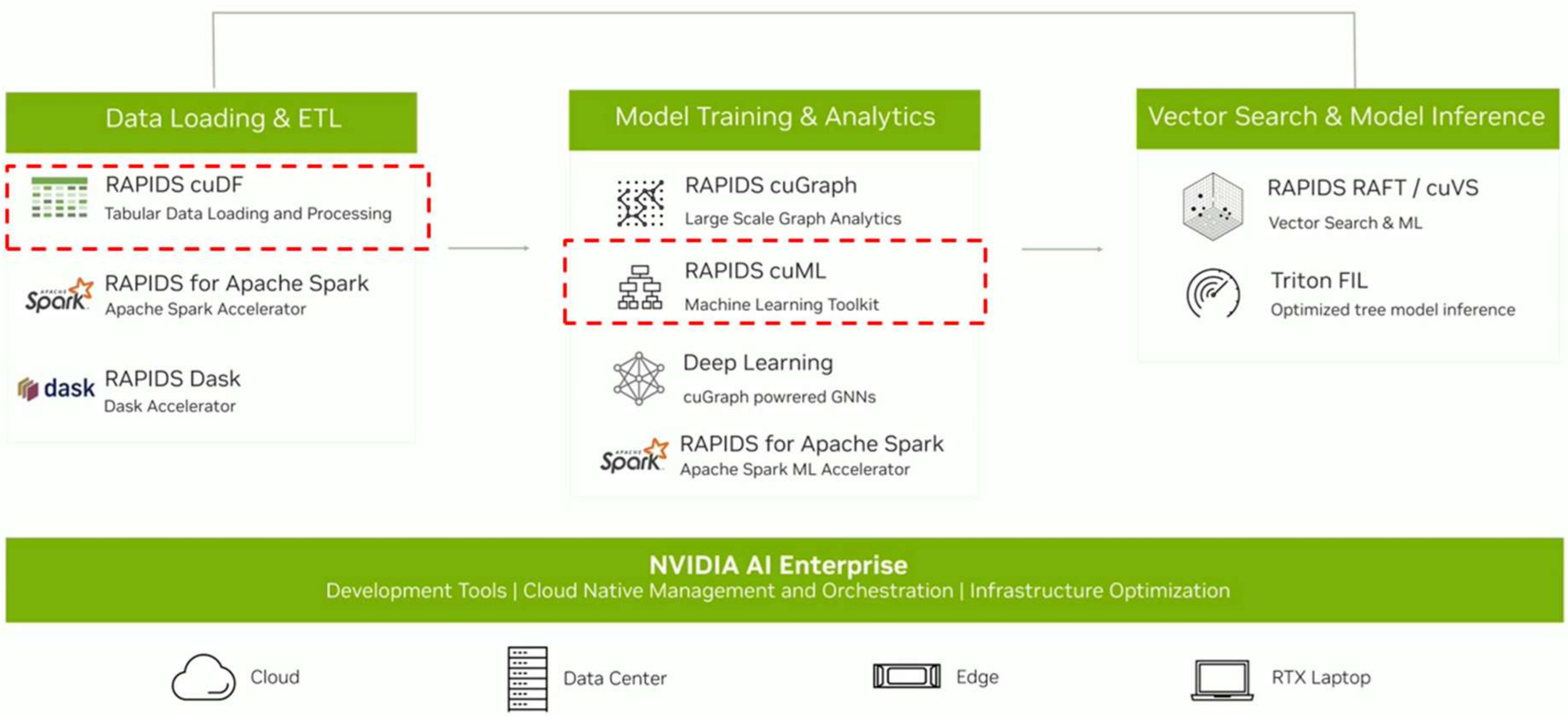
Genomic Analysis

Cybersecurity



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp

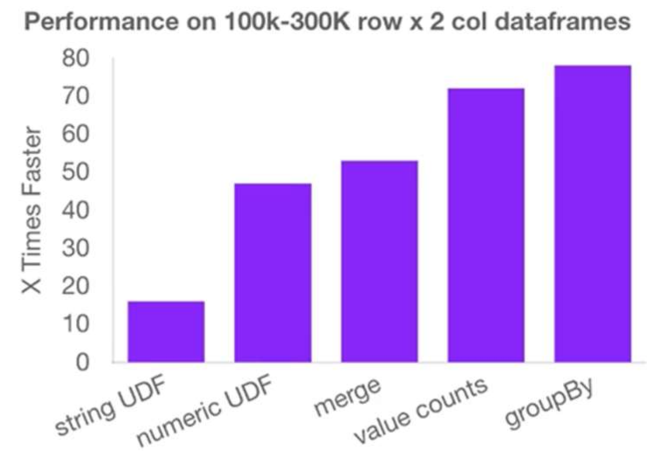
NVIDIA CUDA-X Libraries: Accelerating Data Science End-to-End



NVIDIA cuDF

Background: What is NVIDIA cuDF?

- pandas-like data processing library for the GPU
  - Core functions for loading, filtering, aggregating,
  - Numeric, datetime, categorical, string and nested data
  - GPU accelerated I/O (e.g., CSV, parquet, json)
  - 10s-100s times faster than pandas*
- Built upon the libcudf C++/CUDA library
- Part of the wider RAPIDS ecosystem - see NVIDIA cuML, cuGraph, cuSpatial, RAFT, etc,
- Two modes of usage:
  - Standalone library (classic)
  - **cudf.pandas**

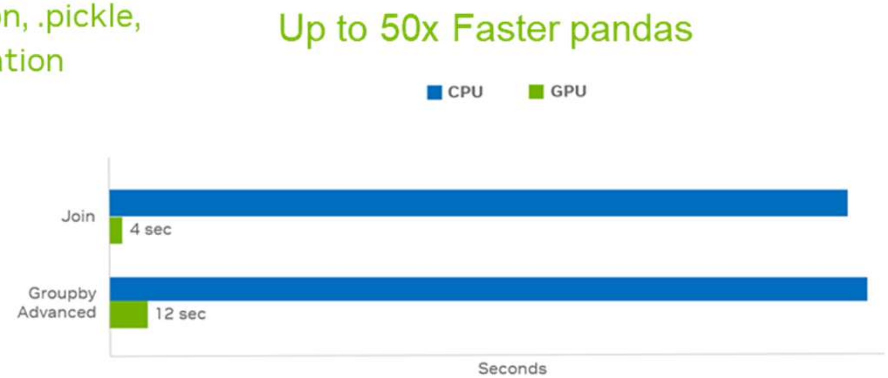


* Benchmark on AMD EPYC 7642 (using 1x 2.3GHz CPU core) w/ 512GB and NVIDIA A100 80GB (1x GPU) w/ pandas v1.5 and cuDF v23.02

Accelerating pandas with ZERO Code Change

pandas Acceleration with NVIDIA cuDF

- Enables the acceleration of pandas workflows using GPUs with minimal code changes
- Significantly speeds up data processing tasks, especially with larger datasets
- cuDF synchronizes data between the GPU and CPU as needed, managing the complexities of heterogeneous execution.
- cuDF pandas supports different file formats such as .csv, .json, .pickle, .paraquet, and hence enables GPU-accelerated data manipulation



Standard DuckDB Data Benchmark (5 GB) on cudf.pandas, pandas v2.2
HW: NVIDIA L4, CPU: Intel Xeon 8480CL
SW: pandas v2.2.1, RAPIDS cuDF 24.02

Automatic pandas Acceleration

- Requires no changes to existing pandas code. Just install cudf and:

```
%load_ext cudf.pandas

                or

$ python -m cudf.pandas <script.py>
```

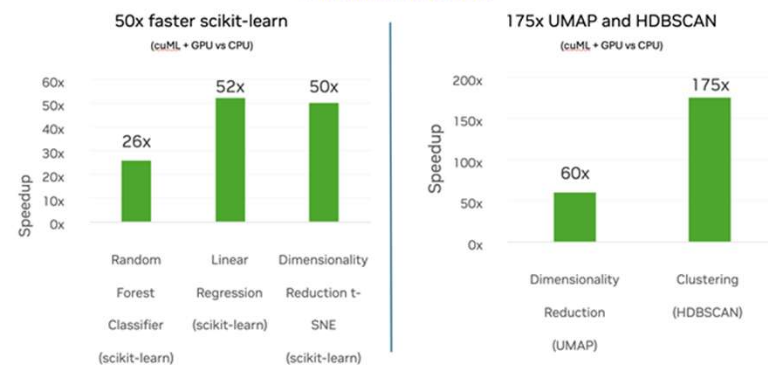
- Suports 100% of the pandas API
- Accelerates operations by 10-100x using the GPU
- Falls back to using pandas on the CPU for unsupported functions and methods

```
[ ]: %load_ext cudf.pandas

     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt

     data = pd.read_parquet("data.parquet")
     subset = data.index.indexer_between_time("09:30", "16:00")
     data = data.iloc[subset]
     results = data.groupby(pd.Grouper(freq="1D")).mean()

     sns.lineplot(results)
     plt.xticks(rotation=30)
```

Zero Code Change for scikit-learn with NVIDIA cuML

- Accelerates popular ML algorithms used in scikit-learn, plus UMAP and HDBSCAN
- Zero code changes required to existing scikit-learn code – just load the extension
  - **%load_ext cuml.accel**
  - **$ python -m cuml.accel script.py**
- Speedups ranging from 5-200x depending on the algorithm
- Compatible with third party libraries

Now available in Open Beta



Specs: NVIDIA cuML 25.02 on NVIDIA H100 80GB HBM3 |scikit-learn v1.5.2 on Intel Xeon Platinum 8480CL

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X_train, y_test)
preds = clf.predict_proba(X_test)
```

cuML OFF

```
%load_ext cuml.accel
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X_train, y_test)
preds = clf.predict_proba(X_test)
```

cuML ON

Deploying NVIDIA CUDA-X Python Libraries

Documentation to get you and up and running RAPIDS anywhere

Local Machine

Use RAPIDS on your local workstation or server.

docker conda pip WSL2

Cloud

Use RAPIDS on the cloud.

Amazon Web Services
Google Cloud Platform
Microsoft Azure IBM Cloud

HPC

Use RAPIDS on high performance computers and supercomputers.

SLURM

Platforms

Use RAPIDS on compute platforms.

Kubernetes Kubeflow Coiled
Databricks

Tools

There are many tools to deploy RAPIDS.

containers dask-kubernetes
dask-operator dask-helm-chart
dask-gateway

Cloud ML Examples

See our example notebooks repo with opinionated deployments of RAPIDS to boost machine learning workflows.

xgboost optuna mlflow
ray tune

Guides

Detailed guides on how to deploy and optimize RAPIDS.

Microsoft Azure Infiniband MIG

RAPIDS Deployment Documentation

Hands-On Lab

Structure of the Hands-On Lab

- **Part 1: Accelerated Feature Engineering with RAPIDS cuDF Pandas on GPU**
  - **Target Encoding**
  - **Count Encoding**
- **Part 2: Train ML models on GPU**
  - **Train an XGBoost model on GPU**
  - **Train an SVC with RAPIDS cuML on GPU**

Lab Details

Use !nvidia-smi to check GPU memory - we are using 1x NVIDIA Tesla T4 with 16GB memory

```
!nvidia-smi
```

```
Thu Feb 27 19:46:18 2025
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.12             Driver Version: 535.104.12   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       On  | 00000001:00:00.0 Off |                  Off |
| N/A   32C    P8               9W /  70W |      2MiB / 16384MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

Shutdown notebooks in the end to free GPU memory

```
import IPython

app = IPython.Application.instance()
app.kernel.do_shutdown(False)
```

```
{'status': 'ok', 'restart': False}
```

Some notebooks automatically restart kernel to free GPU memory and reset DataFrame

```
import IPython

app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

```
{'status': 'ok', 'restart': True}
```