In [1]:
```
from dask_cuda import LocalCUDACluster
from dask.distributed import Client

max_depth = 18
n_trees = 30

cluster = LocalCUDACluster(threads_per_worker=1)
if 'c' in globals():
    c.close()
c = Client(cluster)

workers = c.has_what().keys()
n_workers = len(workers)
n_streams = 8
```

In [2]:
```
import numpy as np
from sklearn import model_selection
import pandas as pd


df = pd.read_csv('test.csv').drop('Unnamed: 0', axis=1)

print(f'Is there any empty value? {df.isnull().values.any()}')
X = df.drop(['C2'],1).to_numpy().astype(np.float32)
y = df['C2'].astype(np.int32)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
test_size=0.2)
```

```
Is there any empty value? False
```

In [3]:
```
import cudf
import dask_cudf
from cuml.dask.common import utils as dask_utils

n_partitions = n_workers
X_train_cudf = cudf.DataFrame.from_pandas(pd.DataFrame(X_train))
y_train_cudf = cudf.Series(y_train)


X_train_dask = dask_cudf.from_cudf(X_train_cudf, npartitions=n_partitions)
y_train_dask = dask_cudf.from_cudf(y_train_cudf, npartitions=n_partitions)
X_train_dask, y_train_dask = \
    dask_utils.persist_across_workers(c, [X_train_dask, y_train_dask], wor
kers=workers)
```

In [4]:
```
from cuml.dask.ensemble import RandomForestRegressor as cumlDaskRF
from dask.distributed import wait

cuml_model = cumlDaskRF(max_depth=max_depth, n_estimators=n_trees,
                        n_streams=n_streams,
                        workers = workers
                        )
cuml_model.fit(X_train_dask, y_train_dask)

wait(cuml_model.rfs)
```

Out[4]:
```
DoneAndNotDoneFutures(done={<Future: status: finished, type: RandomForestR
egressor, key: 0c0d6a0c-0089-11ea-8145-0242ac11000a-1>, <Future: status: f
inished, type: RandomForestRegressor, key: 0c0d6a0c-0089-11ea-8145-0242ac1
1000a-2>, <Future: status: finished, type: RandomForestRegressor, key: 0c0
d6a0c-0089-11ea-8145-0242ac11000a-0>, <Future: status: finished, type: Ran
domForestRegressor, key: 0c0d6a0c-0089-11ea-8145-0242ac11000a-7>, <Future:
 status: finished, type: RandomForestRegressor, key: 0c0d6a0c-0089-11ea-814
5-0242ac11000a-6>, <Future: status: finished, type: RandomForestRegressor,
 key: 0c0d6a0c-0089-11ea-8145-0242ac11000a-4>, <Future: status: finished, t
ype: RandomForestRegressor, key: 0c0d6a0c-0089-11ea-8145-0242ac11000a-5>,
<Future: status: finished, type: RandomForestRegressor, key: 0c0d6a0c-0089
-11ea-8145-0242ac11000a-3>}, not_done=set())
```

In [5]:
```
from sklearn.metrics import mean_squared_error

cuml_y_pred = cuml_model.predict(X_test)
print("CuML accuracy:     ", mean_squared_error(y_test, cuml_y_pred))
```

```
CuML accuracy:      783.576
```

In [6]:
```
from sklearn.ensemble import RandomForestRegressor as sklRF

skl_model = sklRF(max_depth=max_depth, n_estimators=n_trees, n_jobs=-1)
skl_model.fit(X_train, y_train)

skl_y_pred = skl_model.predict(X_test)
print("SKLearn accuracy:  ", mean_squared_error(y_test, skl_y_pred))

skl_model = sklRF(max_depth=max_depth, n_estimators=n_trees, n_jobs=-1)
skl_model.fit(X_train, y_train)
```

```
SKLearn accuracy:   95.17307864889293
```

Out[6]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=18,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=30, n_job
s=-1,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False)
```