DLI Accelerated Data Science Teaching Kt

Lecture 14.4 - RAPIDS Acceleration: Linear Regression

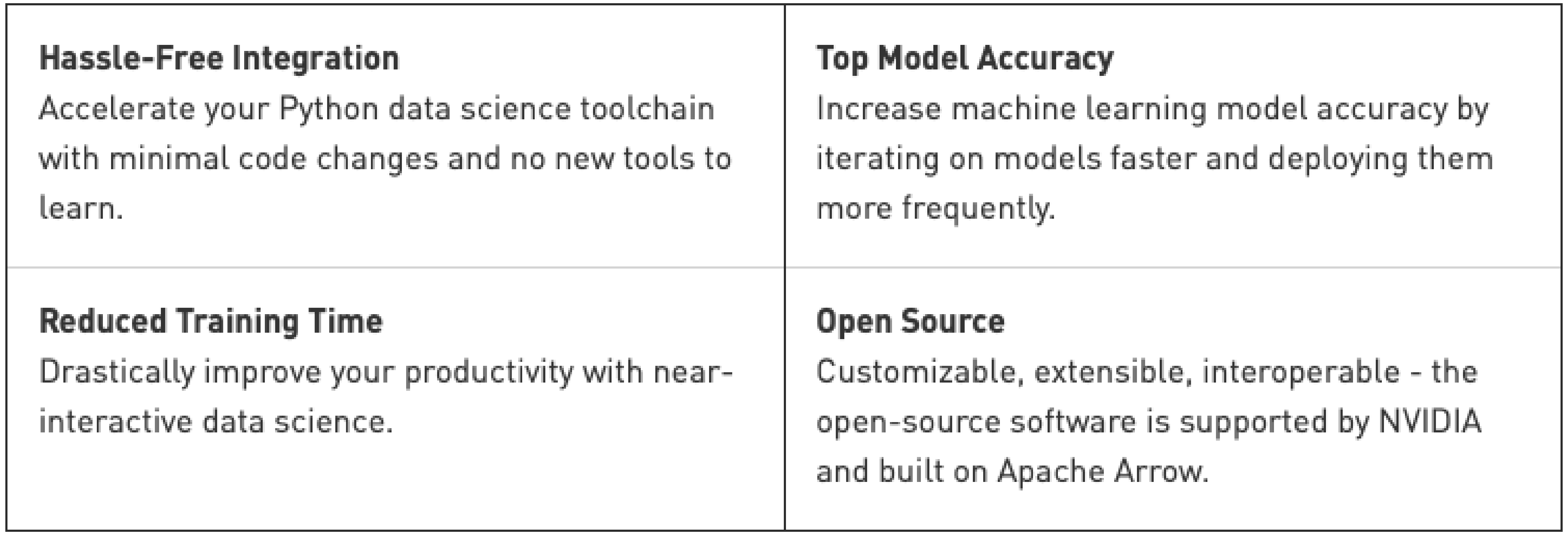
RAPIDS

The RAPIDS data science framework includes a collection of libraries for executing end-to-end data science pipelines completely in the GPU.

It is designed to have a familiar look and feel to data scientists working in Python.

RAPIDS

Features



Hassle-Free Integration Accelerate your Python data science toolchain with minimal code changes and no new tools to learn.	**Top Model Accuracy** Increase machine learning model accuracy by iterating on models faster and deploying them more frequently.
Reduced Training Time Drastically improve your productivity with near-interactive data science.	**Open Source** Customizable, extensible, interoperable - the open-source software is supported by NVIDIA and built on Apache Arrow.

Source: https://rapids.ai/start.html

Speed Up Learning of Linear Regression

Linear Regression is a simple machine learning model where the response y is modelled by a linear combination of the predictors in X.

The model can take array-like objects, either in host as NumPy arrays or in device (as Numba or cuda_array_interface-compliant), as well as pandas or cuDF DataFrames as the input. You can also use the pandas GPU accelerator extension, cuDF.pandas to speed up the processing.

Linear Regression vs Linear Regression cuML

Import packages

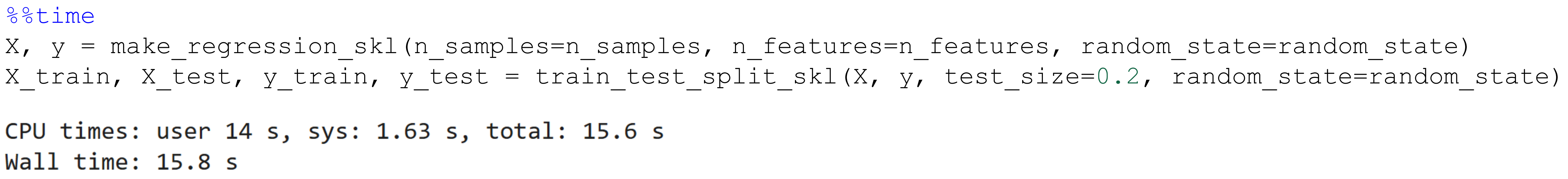
```
# load the cuDF GPU extension for Pandas
%load_ext cudf.pandas
import pandas

# Import CPU based libraries
from sklearn.linear_model import LinearRegression
from sklearn.datasets import make_regression as make_regression_skl
from sklearn.model_selection import train_test_split as train_test_split_skl
from sklearn.metrics import r2_score as r2_score_skl
from sklearn.linear_model import LinearRegression as skLinearRegression

# Import GPU accelerated libraries
from cuml import make_regression as make_regression_cuml, train_test_split as
train_test_split_cuml, LinearRegression as LinearRegression_cuml
from cuml.linear_model import LinearRegression as cuLinearRegression
from cuml.metrics.regression import r2_score as r2_score_cuml
```

Setting parameters

```
n_samples = 2**19 #Change depending on the size of your GPU
n_features = 399
random_state = 23
```

Linear Regression vs Linear Regression cuML

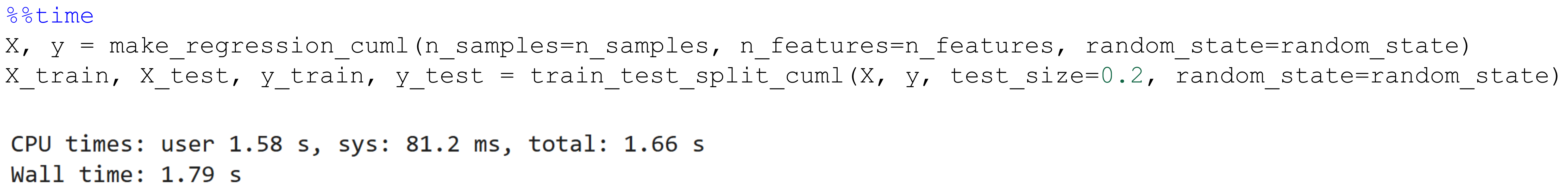
Generating Data with Sklearn



```
%%time
X, y = make_regression_skl(n_samples=n_samples, n_features=n_features, random_state=random_state)
X_train, X_test, y_train, y_test = train_test_split_skl(X, y, test_size=0.2, random_state=random_state)
```

```
CPU times: user 14 s, sys: 1.63 s, total: 15.6 s
Wall time: 15.8 s
```

Generating Data with GPU Acceleration



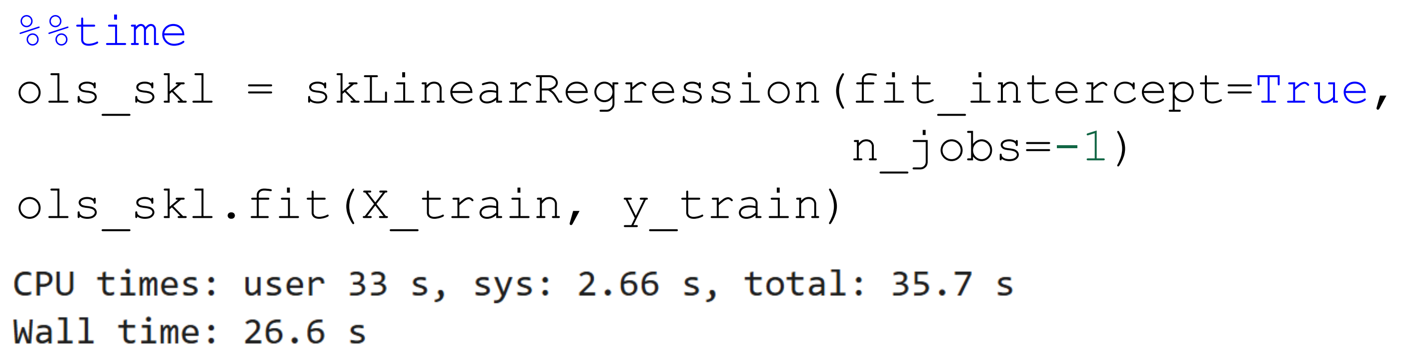
```
%%time
X, y = make_regression_cuml(n_samples=n_samples, n_features=n_features, random_state=random_state)
X_train, X_test, y_train, y_test = train_test_split_cuml(X, y, test_size=0.2, random_state=random_state)
```

```
CPU times: user 1.58 s, sys: 81.2 ms, total: 1.66 s
Wall time: 1.79 s
```

Linear Regression vs Linear Regression cuML

Sklearn Linear Regression

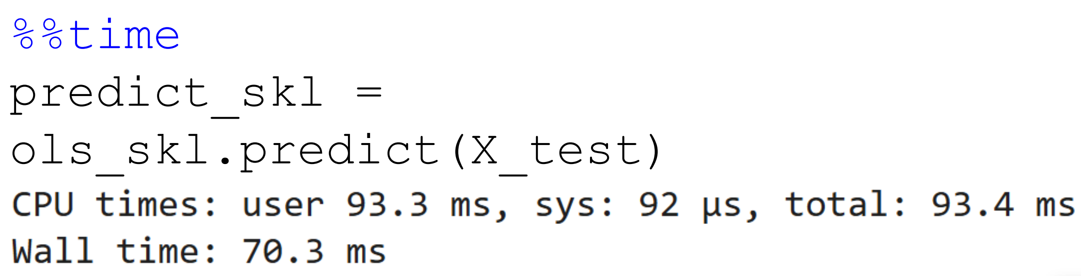
Fit



```
%%time
ols_skl = skLinearRegression(fit_intercept=True,
                             n_jobs=-1)
ols_skl.fit(X_train, y_train)
```

```
CPU times: user 33 s, sys: 2.66 s, total: 35.7 s
Wall time: 26.6 s
```

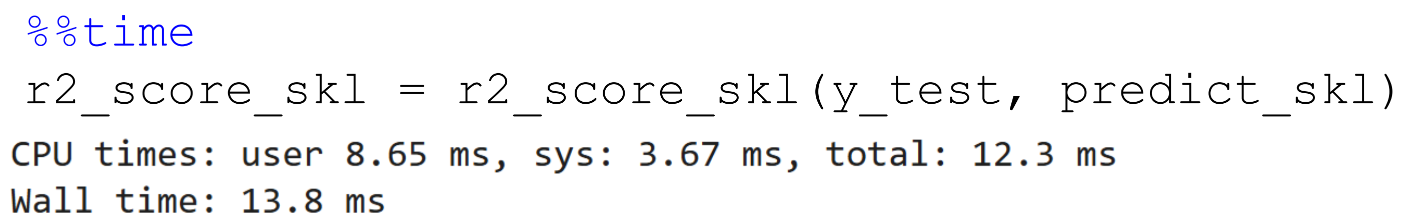
Predict



```
%%time
predict_skl =
ols_skl.predict(X_test)
```

```
CPU times: user 93.3 ms, sys: 92 µs, total: 93.4 ms
Wall time: 70.3 ms
```

Evaluate

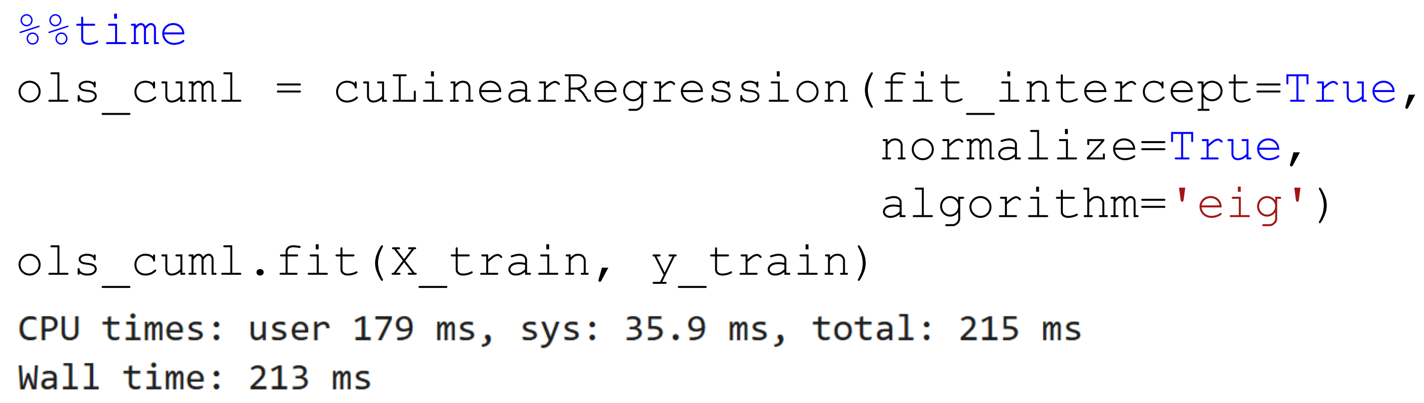


```
%%time
r2_score_skl = r2_score_skl(y_test, predict_skl)
```

```
CPU times: user 8.65 ms, sys: 3.67 ms, total: 12.3 ms
Wall time: 13.8 ms
```

cuML

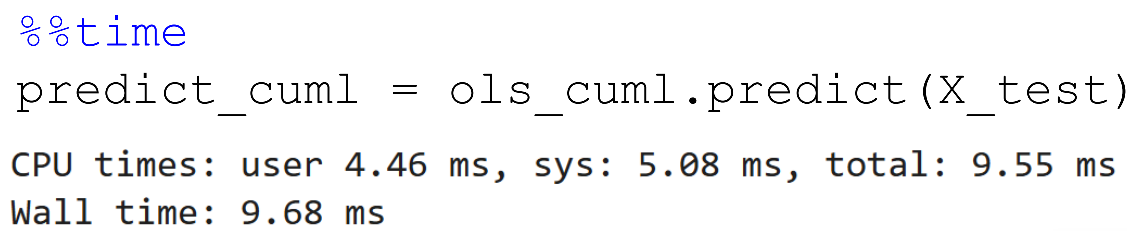
Fit



```
%%time
ols_cuml = cuLinearRegression(fit_intercept=True,
                              normalize=True,
                              algorithm='eig')
ols_cuml.fit(X_train, y_train)
```

```
CPU times: user 179 ms, sys: 35.9 ms, total: 215 ms
Wall time: 213 ms
```

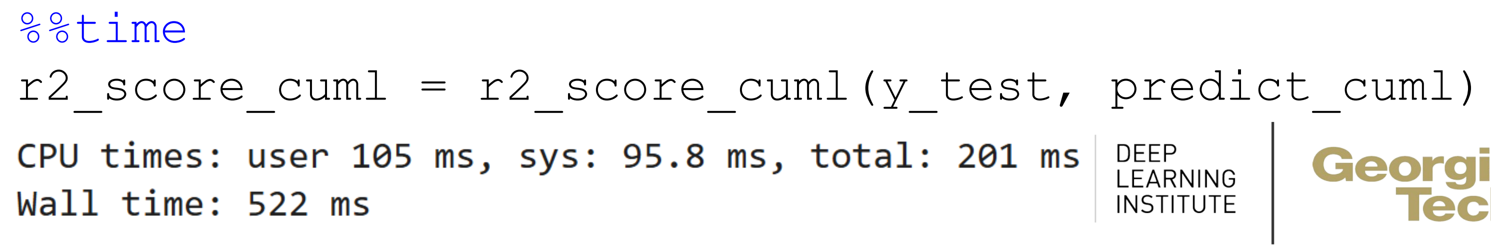
Predict



```
%%time
predict_cuml = ols_cuml.predict(X_test)
```

```
CPU times: user 4.46 ms, sys: 5.08 ms, total: 9.55 ms
Wall time: 9.68 ms
```

Evaluate



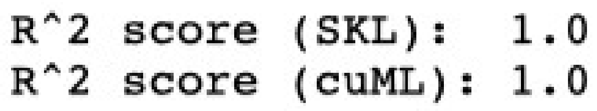
```
%%time
r2_score_cuml = r2_score_cuml(y_test, predict_cuml)
```

```
CPU times: user 105 ms, sys: 95.8 ms, total: 201 ms
Wall time: 522 ms
```

Linear Regression vs Linear Regression cuML

Compare Results

```
print("R^2 score (SKL): %s" r2_score_skl)
print("R^2 score (cuML): %s" r2_score_cuml)
```



```
R^2 score (SKL):  1.0
R^2 score (cuML): 1.0
```

DLI Accelerated Data Science Teaching Kit

Thank You