DEEP
LEARNING
INSTITUTE

DLI Accelerated Data Science Teaching Kit

Lecture 14.13 - k-NN with RAPIDS

What is KNN?

- Stands for K-Nearest Neighbors
- Machine learning technique that predicts an unknown observation:
  - Uses k most similar known observations in the training dataset
- No training time
  - All computation takes place during inference using nearest datapoints in training set

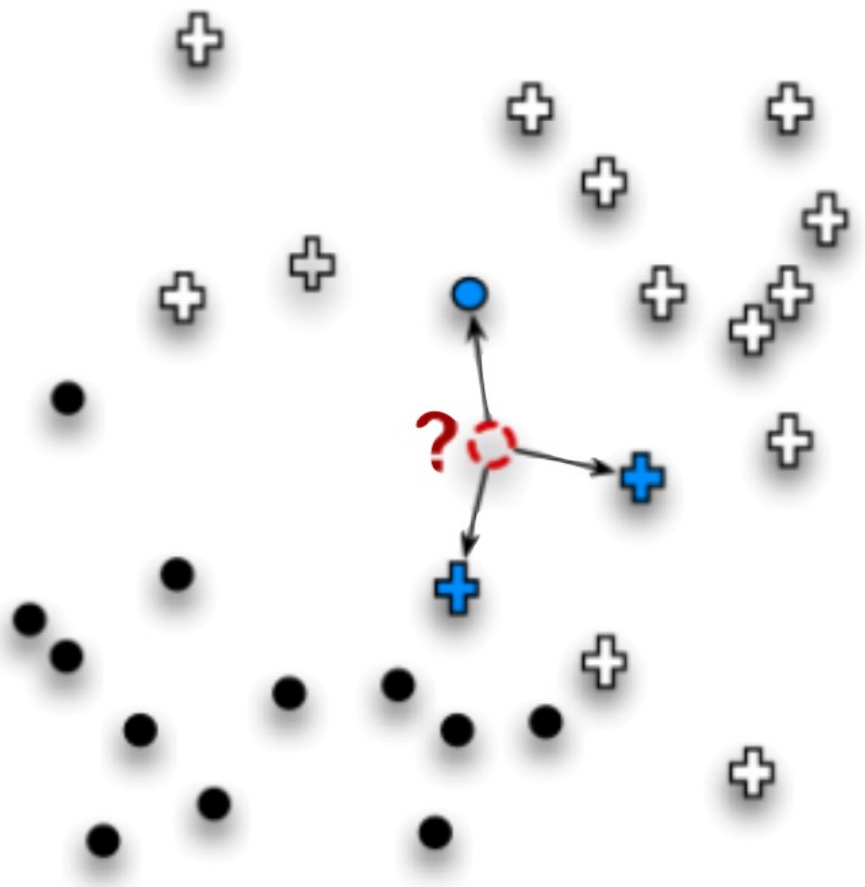


Image Credit: Data Science for Business

Example KNN Classification

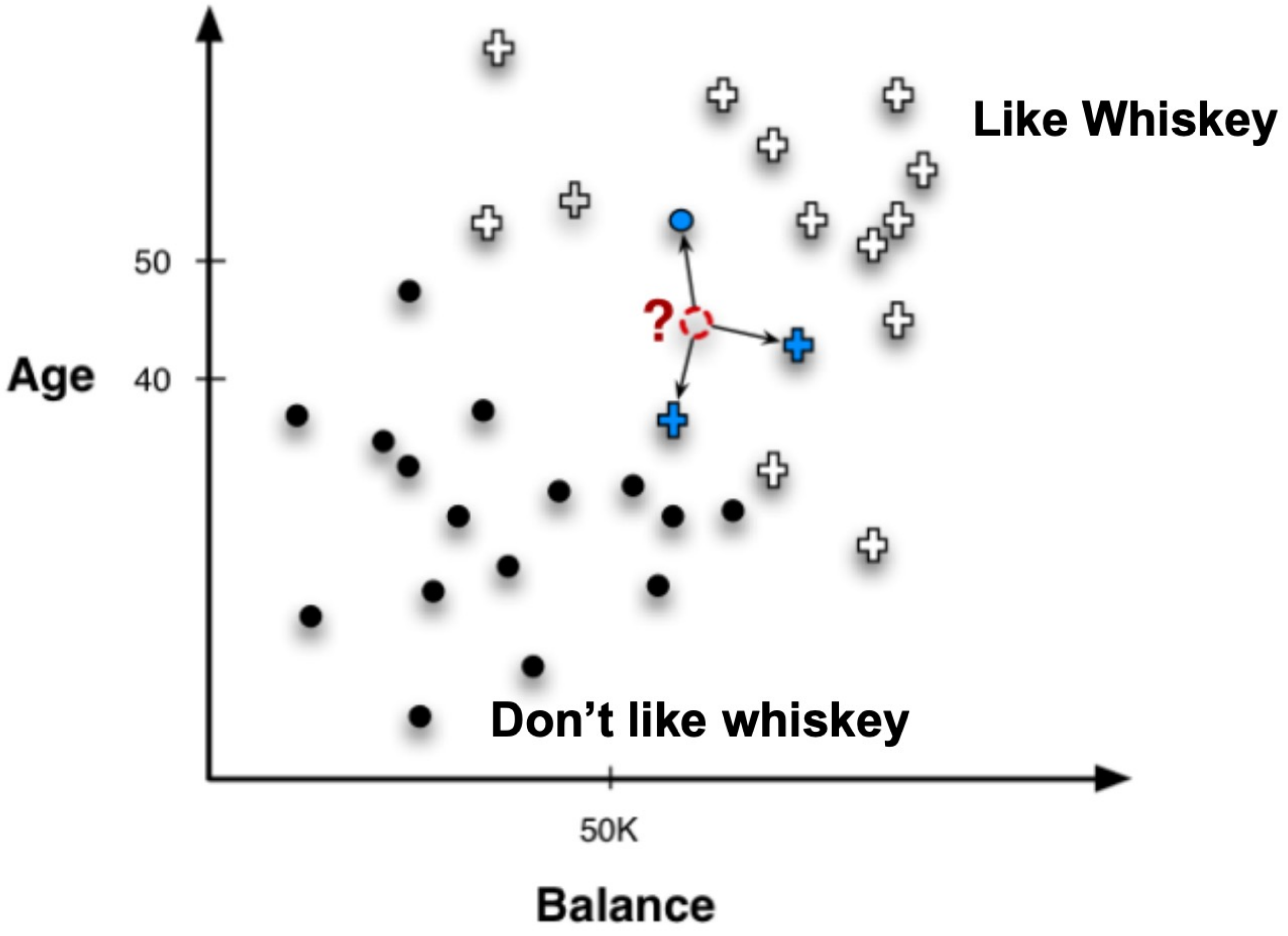
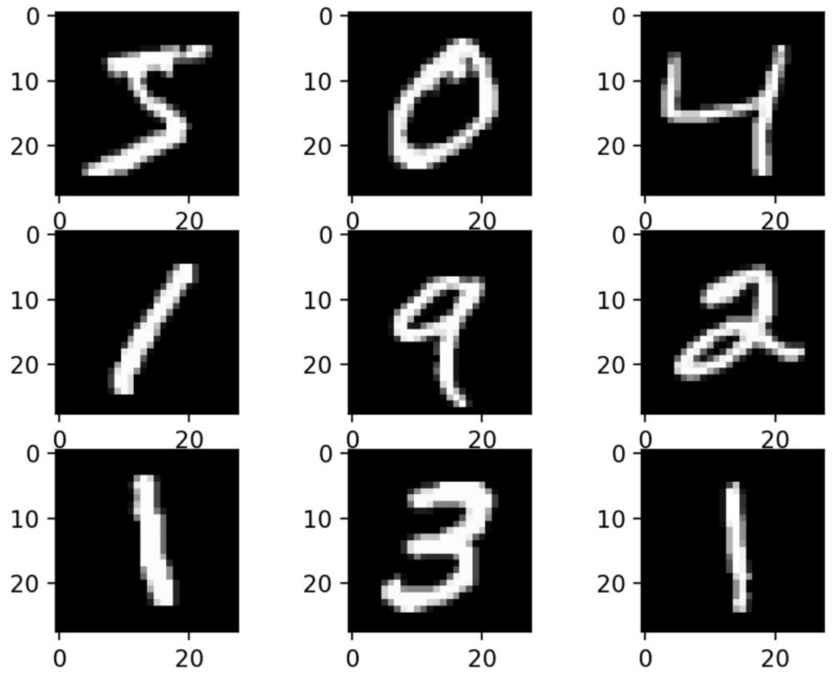


Image Credit: Data Science for Business

KNN Example using RAPIDS

- To show the speed improvement on RAPIDS compared to other platforms, we will explore an example KNN
  - Uses famous MNIST digit image dataset
- Comparison of performance between Nvidia Tesla P100 GPU and classic CPU



Reference: Medium - Accelerating k-nearest neighbors 600x using RAPIDS cuML

Example: Setting up RAPIDS

```
import sys
!conda create -n rapids -c rapidsai -c nvidia -c conda-forge\
rapids=0.11 python=3.6 cudatoolkit=10.1 -y
sys.path = [“/opt/conda/envs/rapids/lib/python3.6/site-packages”]\
+ sys.path
sys.path = [“/opt/conda/envs/rapids/lib/python3.6”] + sys.path
sys.path = [“/opt/conda/envs/rapids/lib”] + sys.path
!cp /opt/conda/envs/rapids/lib/libxgboost.so /opt/conda/lib/
```

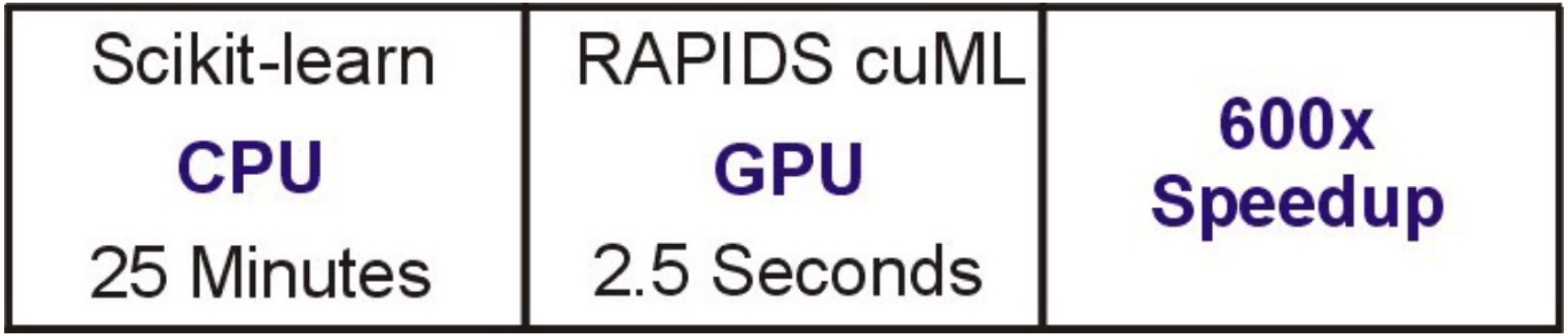
Example: Performing KNN Training

```
# RAPIDS cuML kNN model
import cudf, cuml
from cuml.neighbors import KNeighborsClassifier as cuKNeighbors
train = cudf.read_csv(‘../input/digit-recognizer/train.csv’)
test = cudf.read_csv(‘../input/digit-recognizer/test.csv’)

# Run k-NN training and predictions
model = cuKNeighbors(n_neighbors=7)
model.fit(train.iloc[:,1:785], train.iloc[:,0])
y_hat = model.predict(test)
```

Example: Performing Scikit Version

```
# Scikit-learn kNN model
import pandas
from sklearn.neighbors import KNeighborsClassifier as skKNeighbors
train = pandas.read_csv('../input/digit-recognizer/train.csv')
test = pandas.read_csv('../input/digit-recognizer/test.csv')
model = skKNeighbors(n_neighbors=7)

# Run scikit-learn training and predictions
model.fit(train.iloc[:,1:785], train.iloc[:,0])
y_hat = model.predict(test)
```

Difference in Performance



Scikit-learn **CPU** 25 Minutes	RAPIDS cuML **GPU** 2.5 Seconds	**600x** **Speedup**

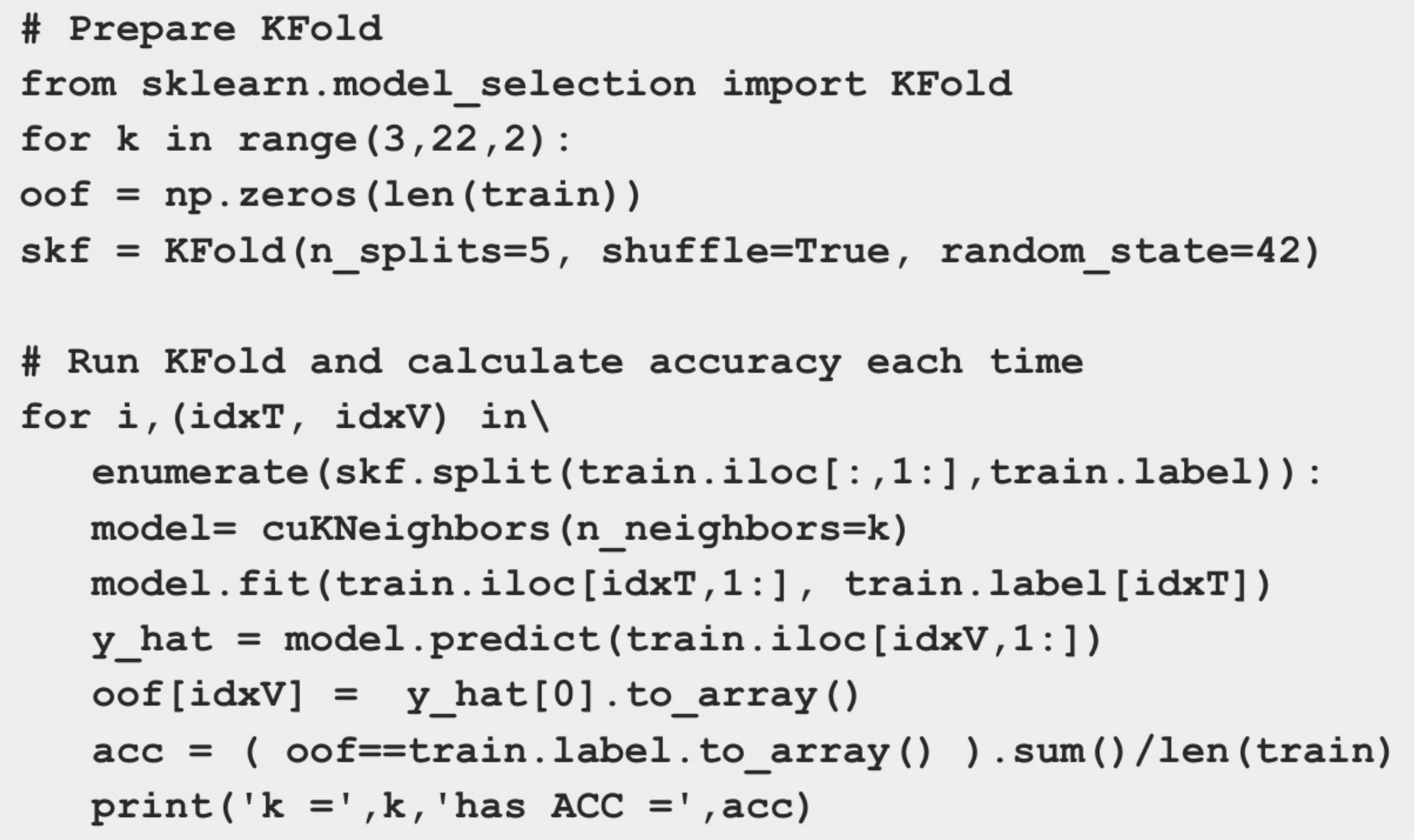
Reference: Medium - Accelerating k-nearest neighbors 600x using RAPIDS cuML

Additional KNN Task

- To further improve our KNN model performance, we can perform the following tasks using RAPIDS:
  - Hyperparameter Tuning
    - Finding the ideal hyperparameters for the model
  - Data Augmentation
    - Adding more training data by manipulating the current data

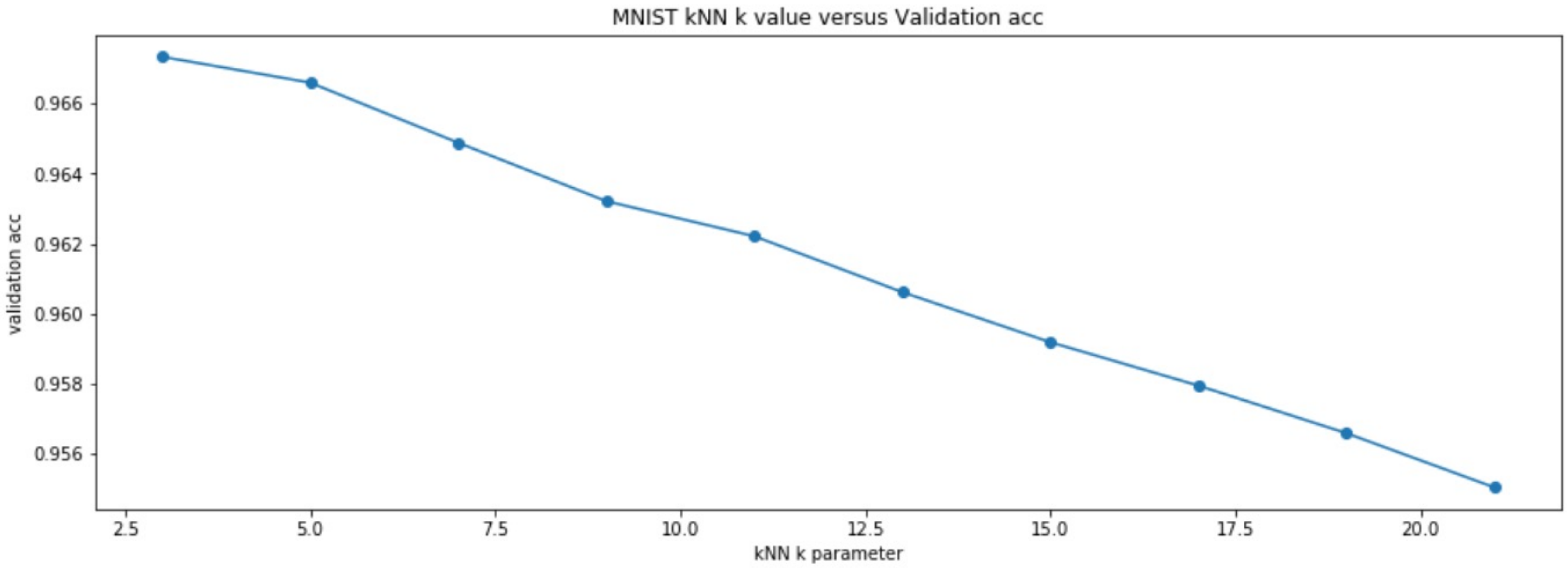
Reference: Medium - Accelerating k-nearest neighbors 600x using RAPIDS cuML

Example: Hyperparameter Tuning



```
# Prepare KFold
from sklearn.model_selection import KFold
for k in range(3,22,2):
oof = np.zeros(len(train))
skf = KFold(n_splits=5, shuffle=True, random_state=42)

# Run KFold and calculate accuracy each time
for i,(idxT, idxV) in\
   enumerate(skf.split(train.iloc[:,1:],train.label)):
   model= cuKNeighbors(n_neighbors=k)
   model.fit(train.iloc[idxT,1:], train.label[idxT])
   y_hat = model.predict(train.iloc[idxV,1:])
   oof[idxV] =  y_hat[0].to_array()
   acc = ( oof==train.label.to_array() ).sum()/len(train)
   print('k =',k,'has ACC =',acc)
```

Example: Improvement in Performance



Reference: Medium - Accelerating k-nearest neighbors 600x using RAPIDS cuML

Example: Data Augmentation

```
# Prepare Image Generator for data augmentation
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(rotation_range=10, zoom_range = 0.10,\
   width_shift_range=0.1, height_shift_range=0.1)
da = 50; bs=4200
train2 = np.zeros((train.shape[0]*da,train.shape[1]),\
   dtype=np.float32)

# Run data augmentation generation
for k,(X,Y) in enumerate( datagen.flow(\
   train[:,1:].reshape((-1,28,28,1)),\
   train[:,0].reshape((-1,1)) ,batch_size=bs ) ):
   train2[bs*k:bs*(k+1),1:] = X.reshape((-1,784))
   train2[bs*k:bs*(k+1),0] = Y.reshape((-1))
   if k==train2.shape[0]//bs-1: break

# Re-run model training using RAPIDS kNN
model.fit(train2[:,1:785], train2[:,0])
y_hat = model.predict(test)
```

Benefits of KNN Rapids

- Improved Speed
  - Accelerated training time by 600x
- Improved Accuracy with Hyperparameter Tuning
  - Increased accuracy with only 2.5 seconds of additional training time
- Further improvement with Data Augmentation
  - Generates new training data and improves accuracy from 96.9% to 98.5%

Reference: Medium - Accelerating k-nearest neighbors 600x using RAPIDS cuML
Image Credit: Nvidia RAPIDS

DLI Accelerated Data Science Teaching Kit

Thank You