KNN

In this notebook you will use GPU-accelerated k-nearest neighbors to identify the nearest road nodes to hospitals.

Objectives

By the time you complete this notebook you will be able to:

- Use GPU-accelerated k-nearest neighbors using a single GPU

Imports

In [1]:
```
import cudf
import cuml
```

Load Data

Road Nodes

We begin by reading our road nodes data.

In [2]:
```
road_nodes = cudf.read_csv('./data/road_nodes_2-06.csv', dtype=['str', 'float32', 'flo
```

In [3]:
```
road_nodes.dtypes
```

Out[3]:
```
node_id     object
east       float32
north      float32
type        object
dtype: object
```

In [4]:
```
road_nodes.shape
```

Out[4]:
```
(3121148, 4)
```

In [5]:
```
road_nodes.head()
```

Out[5]:

	node_id	east	north	type
0	id02FE73D4-E88D-4119-8DC2-6E80DE6F6594	320608.09375	870994.0000	junction
1	id634D65C1-C38B-4868-9080-2E1E47F0935C	320628.50000	871103.8125	road end
2	idDC14D4D1-774E-487D-8EDE-60B129E5482C	320635.46875	870983.8750	junction
3	id51555819-1A39-4B41-B0C9-C6D2086D9921	320648.68750	871083.5625	junction
4	id9E362428-79D7-4EE3-B015-0CE3F6A78A69	320658.18750	871162.3750	junction

Hospitals

Next we load the hospital data.

In [6]:
```
hospitals = cudf.read_csv('./data/hospitals_2-06.csv')
```

In [7]:
```
hospitals.dtypes
```

Out[7]:
```
OrganisationID          int64
OrganisationCode       object
OrganisationType       object
SubType                object
Sector                 object
OrganisationStatus     object
IsPimsManaged          object
OrganisationName       object
Address1               object
Address2               object
Address3               object
City                   object
County                 object
Postcode               object
Latitude              float64
Longitude             float64
ParentODSCode          object
ParentName             object
Phone                  object
Email                  object
Website                object
Fax                    object
northing              float64
easting               float64
dtype: object
```

In [8]:
```
hospitals.shape
```

Out[8]:
```
(1226, 24)
```

In [9]:
```
hospitals.head()
```

Out[9]:

	OrganisationID	OrganisationCode	OrganisationType	SubType	Sector	OrganisationStatus	Isl
0	17970	NDA07	Hospital	Hospital	Independent Sector	Visible	
1	17981	NDA18	Hospital	Hospital	Independent Sector	Visible	
2	18102	NLT02	Hospital	Hospital	NHS Sector	Visible	
3	18138	NMP01	Hospital	Hospital	Independent Sector	Visible	
4	18142	NMV01	Hospital	Hospital	Independent Sector	Visible	

5 rows × 24 columns

K-Nearest Neighbors

We are going to use the k-nearest neighbors algorithm to find the nearest *k* road nodes for every hospital. We will need to fit a KNN model with road data, and then give our trained model hospital locations so that it can return the nearest roads.

Exercise: Prep the KNN Model

Create a k-nearest neighbors model `knn` by using the `cuml.NearestNeighbors` constructor, passing it the named argument `n_neighbors` set to 3.

In [11]:
```
knn = cuml.NearestNeighbors(n_neighbors=3)
```

Solution

In [12]:
```
# %load solutions/prep_knn
knn = cuml.NearestNeighbors(n_neighbors=3)
```

Exercise: Fit the KNN Model

Create a new dataframe `road_locs` using the `road_nodes` columns `east` and `north`. The order of the columns doesn't matter, except that we will need them to remain consistent over multiple operations, so please use the ordering `['east', 'north']`.

Fit the `knn` model with `road_locs` using the `knn.fit` method.

In [14]:
```
road_locs = road_nodes[['east', 'north']]
knn.fit(road_locs)
```

Out[14]:
```
NearestNeighbors()
```

Solution

In [15]:
```
# %load solutions/fit_knn
road_locs = road_nodes[['east', 'north']]
knn.fit(road_locs)
```

Out[15]:
```
NearestNeighbors()
```

Exercise: Road Nodes Closest to Each Hospital

Use the `knn.kneighbors` method to find the 3 closest road nodes to each hospital. `knn.kneighbors` expects 2 arguments: `X`, for which you should use the `easting` and `northing` columns of `hospitals` (remember to retain the same column order as when you fit the `knn` model above), and `n_neighbors`, the number of neighbors to search for--in this case, 3.

`knn.kneighbors` will return 2 cudf dataframes, which you should name `distances` and `indices` respectively.

In [17]:
```
distances, indices = knn.kneighbors(hospitals[['easting', 'northing']], 3) # order has
```

Solution

In [18]:
```
# %load solutions/k_closest_nodes
distances, indices = knn.kneighbors(hospitals[['easting', 'northing']], 3) # order has
```

Viewing a Specific Hospital

We can now use `indices`, `hospitals`, and `road_nodes` to derive information specific to a given hospital. Here we will examine the hospital at index `10`. First we view the hospital's grid coordinates:

In [19]:
```
SELECTED_RESULT = 10
print('hospital coordinates:\n', hospitals.loc[SELECTED_RESULT, ['easting', 'northing'
```

```
hospital coordinates:
easting     260713.17190
northing     56303.21875
Name: 10, dtype: float64
```

Now we view the road node IDs for the 3 closest road nodes:

In [20]:
```
nearest_road_nodes = indices.iloc[SELECTED_RESULT, 0:3]
print('node_id:\n', nearest_road_nodes, sep='')
```

```
node_id:
0    118559
1    118560
2    118678
Name: 10, dtype: int64
```

And finally the grid coordinates for the 3 nearest road nodes, which we can confirm are located in order of increasing distance from the hospital:

In [21]:
```
print('road_node coordinates:\n', road_nodes.loc[nearest_road_nodes, ['east', 'north']
```

```
road_node coordinates:
                 east         north
118559  260697.859375  56322.710938
118560  260722.812500  56207.925781
118678  260540.000000  56105.000000
```

Please Restart the Kernel

In [ ]:
```
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

Next

In the next notebook, you will return to the K-means algorithm, but this time using a multi-node, multi-GPU Dask version that can scale to production size.