2_Create_Graph_Dataset

December 12, 2025

0.1 2. Create Graph Dataset

We install the needed libraries to use GraphStorm following the guide in: https://graphstorm.readthedocs.io/en/v0.4.2/install/env-setup.html#setup-pip

[1]:
```
import os
```

[2]:
```
!pip install torch==2.3.0 --index-url https://download.pytorch.org/whl/cpu
!pip install dgl==2.3.0 -f https://data.dgl.ai/wheels/torch-2.3/repo.html
```

```
/bin/bash: switchml: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `switchml'
/bin/bash: module: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `module'
Looking in indexes: https://download.pytorch.org/whl/cpu
Requirement already satisfied: torch==2.3.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (2.3.0+cpu)
Requirement already satisfied: filelock in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch==2.3.0) (3.20.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch==2.3.0) (4.15.0)
Requirement already satisfied: sympy in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch==2.3.0) (1.14.0)
Requirement already satisfied: networkx in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch==2.3.0) (3.4.2)
Requirement already satisfied: jinja2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch==2.3.0) (3.1.6)
Requirement already satisfied: fsspec in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch==2.3.0) (2025.10.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
jinja2->torch==2.3.0) (3.0.3)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sympy->torch==2.3.0) (1.3.0)
/bin/bash: switchml: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `switchml'
/bin/bash: module: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `module'
Looking in links: https://data.dgl.ai/wheels/torch-2.3/repo.html
Requirement already satisfied: dgl==2.3.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (2.3.0)
Requirement already satisfied: numpy<2.0.0,>=1.14.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (1.26.4)
Requirement already satisfied: scipy>=1.1.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (1.15.3)
Requirement already satisfied: networkx>=2.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (3.4.2)
Requirement already satisfied: requests>=2.19.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (2.32.5)
Requirement already satisfied: tqdm in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (4.67.1)
Requirement already satisfied: psutil>=5.8.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (7.1.3)
Requirement already satisfied: torchdata>=0.5.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (0.9.0)
Requirement already satisfied: pandas in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
dgl==2.3.0) (2.3.3)
Requirement already satisfied: charset_normalizer<4,>=2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests>=2.19.0->dgl==2.3.0) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests>=2.19.0->dgl==2.3.0) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests>=2.19.0->dgl==2.3.0) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests>=2.19.0->dgl==2.3.0) (2025.10.5)
Requirement already satisfied: torch>=2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torchdata>=0.5.0->dgl==2.3.0) (2.3.0+cpu)
```

```
Requirement already satisfied: filelock in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=2->torchdata>=0.5.0->dgl==2.3.0) (3.20.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=2->torchdata>=0.5.0->dgl==2.3.0) (4.15.0)
Requirement already satisfied: sympy in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=2->torchdata>=0.5.0->dgl==2.3.0) (1.14.0)
Requirement already satisfied: jinja2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=2->torchdata>=0.5.0->dgl==2.3.0) (3.1.6)
Requirement already satisfied: fsspec in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=2->torchdata>=0.5.0->dgl==2.3.0) (2025.10.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
jinja2->torch>=2->torchdata>=0.5.0->dgl==2.3.0) (3.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pandas->dgl==2.3.0) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pandas->dgl==2.3.0) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pandas->dgl==2.3.0) (2025.2)
Requirement already satisfied: six>=1.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from python-
dateutil>=2.8.2->pandas->dgl==2.3.0) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sympy->torch>=2->torchdata>=0.5.0->dgl==2.3.0) (1.3.0)
```

[3]:
```
!pip install graphstorm
```

```
/bin/bash: switchml: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `switchml'
/bin/bash: module: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `module'
Requirement already satisfied: graphstorm in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (0.5.0.post1)
Requirement already satisfied: h5py in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (3.11.0)
Requirement already satisfied: pyarrow in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (22.0.0)
```

```
Requirement already satisfied: transformers==4.48.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (4.48.0)
Requirement already satisfied: pandas in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (2.3.3)
Requirement already satisfied: scikit-learn in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (1.7.2)
Requirement already satisfied: ogb==1.3.6 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (1.3.6)
Requirement already satisfied: packaging in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (24.2)
Requirement already satisfied: psutil in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (7.1.3)
Requirement already satisfied: torchdata==0.9.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (0.9.0)
Requirement already satisfied: pydantic in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
graphstorm) (2.12.3)
Requirement already satisfied: torch>=1.6.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
ogb==1.3.6->graphstorm) (2.3.0+cpu)
Requirement already satisfied: numpy>=1.16.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
ogb==1.3.6->graphstorm) (1.26.4)
Requirement already satisfied: tqdm>=4.29.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
ogb==1.3.6->graphstorm) (4.67.1)
Requirement already satisfied: six>=1.12.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
ogb==1.3.6->graphstorm) (1.17.0)
Requirement already satisfied: urllib3>=1.24.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
ogb==1.3.6->graphstorm) (2.5.0)
Requirement already satisfied: outdated>=0.2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
ogb==1.3.6->graphstorm) (0.2.2)
Requirement already satisfied: requests in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torchdata==0.9.0->graphstorm) (2.32.5)
Requirement already satisfied: filelock in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers==4.48.0->graphstorm) (3.20.0)
```

```
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers==4.48.0->graphstorm) (0.36.0)
Requirement already satisfied: pyyaml>=5.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers==4.48.0->graphstorm) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers==4.48.0->graphstorm) (2025.11.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers==4.48.0->graphstorm) (0.21.4)
Requirement already satisfied: safetensors>=0.4.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers==4.48.0->graphstorm) (0.7.0)
Requirement already satisfied: fsspec>=2023.5.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
huggingface-hub<1.0,>=0.24.0->transformers==4.48.0->graphstorm) (2025.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
huggingface-hub<1.0,>=0.24.0->transformers==4.48.0->graphstorm) (4.15.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
huggingface-hub<1.0,>=0.24.0->transformers==4.48.0->graphstorm) (1.2.0)
Requirement already satisfied: setuptools>=44 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
outdated>=0.2.0->ogb==1.3.6->graphstorm) (80.9.0)
Requirement already satisfied: littleutils in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
outdated>=0.2.0->ogb==1.3.6->graphstorm) (0.2.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pandas->graphstorm) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pandas->graphstorm) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pandas->graphstorm) (2025.2)
Requirement already satisfied: scipy>=1.8.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from scikit-
learn->graphstorm) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from scikit-
learn->graphstorm) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from scikit-
learn->graphstorm) (3.6.0)
```

```
Requirement already satisfied: sympy in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=1.6.0->ogb==1.3.6->graphstorm) (1.14.0)
Requirement already satisfied: networkx in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=1.6.0->ogb==1.3.6->graphstorm) (3.4.2)
Requirement already satisfied: jinja2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=1.6.0->ogb==1.3.6->graphstorm) (3.1.6)
Requirement already satisfied: MarkupSafe>=2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
jinja2->torch>=1.6.0->ogb==1.3.6->graphstorm) (3.0.3)
Requirement already satisfied: annotated-types>=0.6.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pydantic->graphstorm) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pydantic->graphstorm) (2.41.4)
Requirement already satisfied: typing-inspection>=0.4.2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pydantic->graphstorm) (0.4.2)
Requirement already satisfied: charset_normalizer<4,>=2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->torchdata==0.9.0->graphstorm) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->torchdata==0.9.0->graphstorm) (3.11)
Requirement already satisfied: certifi>=2017.4.17 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->torchdata==0.9.0->graphstorm) (2025.10.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sympy->torch>=1.6.0->ogb==1.3.6->graphstorm) (1.3.0)
```

We will also clone the GraphStorm repository to get access to a set of scripts, and tools that facilitate the use of the framework:

[3]:
```
!git clone https://github.com/awslabs/graphstorm.git
```

```
/bin/bash: switchml: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `switchml'
/bin/bash: module: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `module'
Cloning into 'graphstorm'…
remote: Enumerating objects: 13062, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 13062 (delta 16), reused 6 (delta 0), pack-reused 13003 (from
2)
```

```
Receiving objects: 100% (13062/13062), 9.63 MiB | 41.42 MiB/s, done.
Resolving deltas: 100% (9142/9142), done.
```

[4]:
```
!pip install sentence_transformers
```

```
/bin/bash: switchml: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `switchml'
/bin/bash: module: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `module'
Requirement already satisfied: sentence_transformers in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (5.1.2)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (4.48.0)
Requirement already satisfied: tqdm in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (2.3.0+cpu)
Requirement already satisfied: scikit-learn in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (1.7.2)
Requirement already satisfied: scipy in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (1.15.3)
Requirement already satisfied: huggingface-hub>=0.20.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (0.36.0)
Requirement already satisfied: Pillow in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (12.0.0)
Requirement already satisfied: typing_extensions>=4.5.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sentence_transformers) (4.15.0)
Requirement already satisfied: filelock in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (3.20.0)
Requirement already satisfied: numpy>=1.17 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (2025.11.3)
Requirement already satisfied: requests in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (2.32.5)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (0.21.4)
Requirement already satisfied: safetensors>=0.4.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (0.7.0)
Requirement already satisfied: fsspec>=2023.5.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
huggingface-hub>=0.20.0->sentence_transformers) (2025.10.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
huggingface-hub>=0.20.0->sentence_transformers) (1.2.0)
Requirement already satisfied: sympy in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=1.11.0->sentence_transformers) (1.14.0)
Requirement already satisfied: networkx in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=1.11.0->sentence_transformers) (3.4.2)
Requirement already satisfied: jinja2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
torch>=1.11.0->sentence_transformers) (3.1.6)
Requirement already satisfied: MarkupSafe>=2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
jinja2->torch>=1.11.0->sentence_transformers) (3.0.3)
Requirement already satisfied: charset_normalizer<4,>=2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->transformers<5.0.0,>=4.41.0->sentence_transformers) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->transformers<5.0.0,>=4.41.0->sentence_transformers) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->transformers<5.0.0,>=4.41.0->sentence_transformers) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->transformers<5.0.0,>=4.41.0->sentence_transformers) (2025.10.5)
Requirement already satisfied: joblib>=1.2.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from scikit-
learn->sentence_transformers) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from scikit-
learn->sentence_transformers) (3.6.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sympy->torch>=1.11.0->sentence_transformers) (1.3.0)
```

[5]:
```
from sklearn.model_selection import train_test_split
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq
import pyarrow.fs as pa_fs
import pyarrow.csv as pa_csv
from joblib import Parallel, delayed
from collections import defaultdict
from pathlib import Path
from sentence_transformers import SentenceTransformer
from utils import split_dataframe, write_compatible_csv
from pandas import Float32Dtype, Float64Dtype, Int32Dtype, Int64Dtype,␣
 ↪StringDtype
import logging
from typing import (
    Any,
    Literal,
    Mapping,
    Optional,
    Sequence,
    TypedDict,
    Union,
    cast,
)
import boto3
import json
import os
import re
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

[ ]:
```
#transaction_df = pd.read_csv("./input_data/train_transaction.csv")
#identity_df = pd.read_csv("./input_data/train_identity.csv")
```

[6]:
```
GRAPH_NAME = "ieee-cis-fraud-detection"

PROCESSED_PREFIX = f"./{GRAPH_NAME}"

ID_COLS =␣
 ↪"card1,card2,card3,card4,card5,card6,ProductCD,addr1,addr2,P_emaildomain,R_emaildomain"
CAT_COLS = "M1,M2,M3,M4,M5,M6,M7,M8,M9"
```

```
# Columns we will use for training
COLS_TO_KEEP = {
    "transaction.csv": (
        ID_COLS.split(",")
        + CAT_COLS.split(",")
        +
        [f"C{idx}" for idx in range(1, 15)]
        + ["TransactionID", "TransactionAmt", "TransactionDT", "isFraud"]
    ),
    "identity.csv": ["TransactionID", "DeviceType"],
}
```

[7]:
```
transaction_cols_to_keep = COLS_TO_KEEP['transaction.csv']
identity_cols_to_keep = COLS_TO_KEEP['identity.csv']
```

[8]:
```
read_options = pa_csv.ReadOptions(use_threads=True)
parse_options = pa_csv.ParseOptions(delimiter=",")
convert_options = pa_csv.
 ↪ConvertOptions(include_columns=transaction_cols_to_keep)

transaction_df = pa_csv.read_csv(
            "./input_data/train_transaction.csv",
            read_options=read_options,
            parse_options=parse_options,
            convert_options=convert_options,
        ).to_pandas()
```

[9]:
```
convert_options = pa_csv.ConvertOptions(include_columns=identity_cols_to_keep)

identity_df = pa_csv.read_csv(
            "./input_data/train_identity.csv",
            read_options=read_options,
            parse_options=parse_options,
            convert_options=convert_options,
        ).to_pandas()
```

0.1.1 2.1 Create test/train/validation split

[10]:
```
#We create the test and train split having an equal split of isFraud for all␣
 ↪parts
train_df, test_val_df = train_test_split(
    transaction_df,
    train_size=0.8,
    random_state=42,
    stratify=transaction_df["isFraud"],
)
```

[11]:
```
test_df, val_df = train_test_split(
    test_val_df, train_size=0.5, random_state=42,␣
 ↪stratify=test_val_df["isFraud"]
)
```

[16]:
```
os.makedirs("./data_splits", exist_ok=True)
```

[17]:
```
for df_name, df_data in [
    ("train_ids.parquet", train_df[["TransactionID"]]),
    ("val_ids.parquet", val_df[["TransactionID"]]),
    ("test_ids.parquet", test_df[["TransactionID"]]),
]:
    df_data: pd.DataFrame = df_data.reset_index(drop=True)
    out_path = os.path.join("./data_splits", df_name)
    table = pa.Table.from_pandas(df_data.rename(columns={"TransactionID":␣
 ↪"nid"}))

    pq.write_table(table, out_path)
```

[18]:
```
def get_fraud_frac(series):
    return 100 * sum(series) / len(series)
```

[19]:
```
print(
    "Percent fraud for train/val/test transactions: {:.2f}% / {:.2f}% / {:.
 ↪2f}%".format(
        get_fraud_frac(train_df.isFraud),
        get_fraud_frac(val_df.isFraud),
        get_fraud_frac(test_df.isFraud),
    )
)
```

```
Percent fraud for train/val/test transactions: 3.50% / 3.50% / 3.50%
```

0.1.2 2.2 Create vertex files, features and labels

[20]:
```
non_feature_cols = [
    "isFraud",
    "TransactionDT",
    "TransactionID",
] + ID_COLS.split(",")

feature_cols = [
    col for col in transaction_df.columns if col not in non_feature_cols
]
cat_cols = CAT_COLS.split(",")
```

[21]:
```
feature_dict = {}
for col in feature_cols:
    if col in cat_cols:
        feature_dict[f"{col}:String"] = transaction_df[col]
    else:
        # Check if the column contains integer values
        if pd.api.types.is_integer_dtype(transaction_df[col]):
            feature_dict[f"{col}:Int"] = transaction_df[col]
        else:
            feature_dict[f"{col}:Float"] = transaction_df[col]
```

[22]:
```
transaction_vertices = pd.DataFrame(
    {
        "~id": transaction_df["TransactionID"].astype(str),
        "~label": "Transaction",
        "isFraud:Int": transaction_df["isFraud"],
        "TransactionDT:String": transaction_df["TransactionDT"].astype(str),
    }
)
```

[23]:
```
# We add all features to a dataframe
if feature_dict:
    features_df = pd.DataFrame(feature_dict)
    transaction_vertices = pd.concat([transaction_vertices, features_df],␣
 ↪axis=1)
```

[24]:
```
special_cols = ["~id", "~label", "~from", "~to"]

# For Neptune we need to add valid suffixes to the columns
# https://docs.aws.amazon.com/neptune-analytics/latest/userguide/
 ↪property-column-headers.html
renamed_cols = {}
for col in transaction_vertices.columns:
    # Skip special columns and columns that already have type information
    if col in special_cols or ":" in col:
        continue

    # Add default type suffix based on data type
    if pd.api.types.is_integer_dtype(df[col]):
        renamed_cols[col] = f"{col}:Int"
    elif pd.api.types.is_float_dtype(df[col]):
        renamed_cols[col] = f"{col}:Float"
    else:
        renamed_cols[col] = f"{col}:String"

# Apply the renames if any
if renamed_cols:
```

```
    transaction_vertices = transaction_vertices.rename(columns=renamed_cols)
```

[25]:
```
num_chunks = 1
```

[26]:
```
if len(transaction_vertices) > num_chunks:
    chunk_size = len(transaction_vertices) // num_chunks
    chunks = []
    for i in range(0, num_chunks - 1):
        chunks.append(transaction_vertices.iloc[i * chunk_size : (i + 1) *␣
 ↪chunk_size])

    chunks.append(transaction_vertices.iloc[(num_chunks - 1) * chunk_size :])
    df_chunks = [chunk for chunk in chunks if not chunk.empty]
else:
    df_chunks = [transaction_vertices]
```

[27]:
```
csv_options = pa_csv.WriteOptions(
    include_header=True,
    delimiter=",",
    quoting_style="needed",
)
```

[28]:
```
def write_df_chunk(chunk_idx, chunk_df):
    # From DataFrame to PyArrow table
    table = pa.Table.from_pandas(chunk_df.reset_index(drop=True))
    os.makedirs("./graph_data/vertices", exist_ok=True)
    chunk_path = os.path.join("./graph_data/vertices",␣
 ↪f"Vertex_Transaction_{chunk_idx}.csv")

    with pa_fs.LocalFileSystem().open_output_stream(chunk_path) as f:
        pa_csv.write_csv(table, f, write_options=csv_options)
```

[29]:
```
cpu_count = min(os.cpu_count() or 8, 16)
```

[30]:
```
with Parallel(n_jobs=cpu_count, prefer="threads", verbose=0) as parallel:
    parallel(
        delayed(write_df_chunk)(i, chunk_df) for i, chunk_df in␣
 ↪enumerate(df_chunks)
    )
```

0.1.3 2.3 Create vertex files for node types other than Transaction

We will create vertex files for Card[1-6], Product code (ProductCD), Address[1-2] and EmailDomain

We will use one functionality from utils that we created. `write_compatible_csv` is a function that replicates what we did for the transaction vertices. It splits the dataframe into chunks and writes it into .csv.

[31]:
```
for col in ID_COLS.split(","):
    if col not in transaction_df.columns:
        logging.warning(f"ID column '{col}' not found in transactions table")
```

[32]:
```
def encode_nodes(node_df: pd.DataFrame):
    # Graph machine-learning models require node features, but vertices other␣
 ↪than transaction have no node features
    # To enable training using these nodes we will use ~id and converts it to a␣
 ↪list of strings

    model = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings = model.encode(node_df["~id"].tolist(), batch_size=64)

    def emb_to_str(emb):
        return "|".join(f"{x:.8f}" for x in emb)
    # Create an embedding per each node

    node_df["id_embeddings"] = [emb_to_str(emb) for emb in embeddings]

    return node_df
```

[33]:
```
for card_num in range(1, 7):  # card1 through card6
    card_vertices = pd.DataFrame()
    card_vertices["~id"] = f"card{card_num}:" + transaction_df[
        f"card{card_num}"
    ].astype(str)
    card_vertices["~label"] = f"Card{card_num}"
    card_vertices = card_vertices.drop_duplicates()
    card_vertices = encode_nodes(card_vertices)
    os.makedirs("./graph_data", exist_ok=True)
    write_compatible_csv(
        card_vertices,
        os.path.join("./graph_data", f"Vertex_Card{card_num}"),
        num_chunks=num_chunks,
    )
```

[34]:
```
productcd_vertices = pd.DataFrame()
productcd_vertices["~id"] = "ProductCD:" + transaction_df["ProductCD"].
 ↪astype(str)
productcd_vertices["~label"] = "ProductCD"
productcd_vertices = productcd_vertices.drop_duplicates()
productcd_vertices = encode_nodes(productcd_vertices)
os.makedirs("./graph_data", exist_ok=True)
write_compatible_csv(
    productcd_vertices,
    os.path.join("./graph_data", "Vertex_ProductCD"),
    num_chunks=num_chunks,
```

```
)
```

[35]:
```
for addr_num in range(1, 3):
    addr_vertices = pd.DataFrame()
    addr_vertices["~id"] = f"addr{addr_num}:" + transaction_df[
        f"addr{addr_num}"
    ].astype(str)
    addr_vertices["~label"] = f"Address{addr_num}"
    addr_vertices = addr_vertices.drop_duplicates()
    addr_vertices = encode_nodes(addr_vertices)
    os.makedirs("./graph_data", exist_ok=True)
    write_compatible_csv(
        addr_vertices,
        os.path.join("./graph_data", f"Vertex_Address{addr_num}"),
        num_chunks=num_chunks,
    )
```

[36]:
```
for email_type in ["P_emaildomain", "R_emaildomain"]:
    email_vertices = pd.DataFrame()
    email_vertices["~id"] = f"{email_type}:" + transaction_df[email_type].
 ↪astype(
        str
    )
    email_vertices["~label"] = email_type
    email_vertices = email_vertices.drop_duplicates()
    email_vertices = encode_nodes(email_vertices)
    os.makedirs("./graph_data", exist_ok=True)
    write_compatible_csv(
        email_vertices,
        os.path.join("./graph_data", f"Vertex_{email_type}"),
        num_chunks=num_chunks,
    )
```

[37]:
```
id_cols: list[str] = ["TransactionID"] + ID_COLS.split(",")
full_identity_df = transaction_df[id_cols].merge(
    identity_df, on="TransactionID", how="left"
)
```

[38]:
```
nan_count = full_identity_df["DeviceType"].isna().sum()
print(nan_count)
full_identity_df["DeviceType"] = full_identity_df["DeviceType"].fillna(
    "UNKNOWN_DEVICE"
)
```

```
446307
```

[39]:
```
edge_types = ID_COLS.split(",")
for etype in edge_types:
    edges = pd.DataFrame()
    edges["~id"] = (
        "transaction_"
        + full_identity_df["TransactionID"].astype(str)
        + f"-identified-by-{etype}"
    )
    edges["~from"] = full_identity_df["TransactionID"].astype(str)
    edges["~to"] = f"{etype}:" + full_identity_df[etype].astype(str)

    # Map the edge type to the correct vertex label
    if etype.startswith("card"):
        vertex_type = f"Card{etype[4:]}"
    elif etype == "ProductCD":
        vertex_type = "ProductCD"
    elif etype.startswith("addr"):
        vertex_type = f"Address{etype[4:]}"
    elif etype.endswith("emaildomain"):
        vertex_type = etype
    else:
        vertex_type = etype.capitalize()

    associated_edge_label = f"Transaction,identified_by,{vertex_type}"

    edges["~label"] = associated_edge_label
    edges = edges.drop_duplicates().dropna()
    os.makedirs("./graph_data", exist_ok=True)
    write_compatible_csv(
        edges,
        os.path.join("./graph_data/", f"Edge_{associated_edge_label}"),
        num_chunks=num_chunks,
    )
```

0.1.4 GConstruct creation from processed data

To create the train data for GraphStorm we need to create a JSON file that describes the tabular graph data.

[40]:
```
client = boto3.client("neptune-graph")
```

[234]:
```
graph_id = 'g-ahtxjvmkq4'

query = "CALL neptune.graph.pg_schema() YIELD schema RETURN schema"
response = client.execute_query(
    graphIdentifier=graph_id, queryString=query, language="OPEN_CYPHER"
)
```

```
payload = response["payload"]
schema_json = json.loads(payload.read().decode("utf-8"))
```

[235]:
```
neptune_schema = schema_json["results"][0]["schema"]
print(neptune_schema)
```

```
{'edgeLabelDetails': {}, 'edgeLabels': [], 'nodeLabels': [], 'labelTriples': [],
'nodeLabelDetails': {}}
```

[49]:
```
graph_info = client.get_graph(
    graphIdentifier=graph_id,
)
```

[51]:
```
neptune_schema["vectorSearchConfiguration"] = graph_info.get(
    "vectorSearchConfiguration", None
)
```

[41]:
```
vertex_types: set[str] = set()
relation_types: set[str] = set()
vertex_files = defaultdict(list)
edge_files = defaultdict(list)
```

[42]:
```
path_obj = Path('./graph_data')
csv_files = [str(f) for f in path_obj.rglob('*.csv')]
print(csv_files)
```

```
['graph_data/Edge_Transaction,identified_by,Address2_0.csv',
'graph_data/Vertex_Card2_0.csv',
'graph_data/Edge_Transaction,identified_by,P_emaildomain_0.csv',
'graph_data/Vertex_Address2_0.csv', 'graph_data/Vertex_Card4_0.csv',
'graph_data/Vertex_Card6_0.csv',
'graph_data/Edge_Transaction,identified_by,Address1_0.csv',
'graph_data/Edge_Transaction,identified_by,Card3_0.csv',
'graph_data/Edge_Transaction,identified_by,Card6_0.csv',
'graph_data/Vertex_Address1_0.csv',
'graph_data/Edge_Transaction,identified_by,Card2_0.csv',
'graph_data/Vertex_Card3_0.csv',
'graph_data/Edge_Transaction,identified_by,Card4_0.csv',
'graph_data/Vertex_Card1_0.csv', 'graph_data/Vertex_R_emaildomain_0.csv',
'graph_data/Vertex_Transaction_0.csv', 'graph_data/Vertex_P_emaildomain_0.csv',
'graph_data/Edge_Transaction,identified_by,Card5_0.csv',
'graph_data/Vertex_Card5_0.csv',
'graph_data/Edge_Transaction,identified_by,ProductCD_0.csv',
'graph_data/Edge_Transaction,identified_by,Card1_0.csv',
'graph_data/Edge_Transaction,identified_by,R_emaildomain_0.csv',
'graph_data/Vertex_ProductCD_0.csv']
```

[43]:
```
for f in csv_files:
    fname = Path(f).name

    if fname.startswith("Vertex_"):
        # match everything between "Vertex_" and the last underscore followed␣
 ↪by a number:
        vtype = re.match(r"Vertex_(.+?)_(\d+)\.(parquet|csv)", fname)
        if vtype:
            vertex_type = vtype.group(1)
            assert isinstance(vertex_type, str)
            vertex_types.add(vertex_type)
            # We add just the filename to use relative paths in config
            vertex_files[vtype.group(1)].append(fname)

    elif fname.startswith("Edge_"):
        # Pattern to capture edge type until the last underscore
        relation_match = re.match(r"Edge_(.+)_\d+\.(parquet|csv)", fname)
        if relation_match:
            relation_str = relation_match.group(1)
            assert isinstance(relation_str, str)
            relation_types.add(relation_str)
            # We add just the filename to use relative paths in config
            edge_files[relation_str].append(fname)
```

[44]:
```
vertex_types
```

[44]:
```
{'Address1',
 'Address2',
 'Card1',
 'Card2',
 'Card3',
 'Card4',
 'Card5',
 'Card6',
 'P_emaildomain',
 'ProductCD',
 'R_emaildomain',
 'Transaction'}
```

[45]:
```
relation_types
```

[45]:
```
{'Transaction,identified_by,Address1',
 'Transaction,identified_by,Address2',
 'Transaction,identified_by,Card1',
 'Transaction,identified_by,Card2',
 'Transaction,identified_by,Card3',
 'Transaction,identified_by,Card4',
```

```
 'Transaction,identified_by,Card5',
 'Transaction,identified_by,Card6',
 'Transaction,identified_by,P_emaildomain',
 'Transaction,identified_by,ProductCD',
 'Transaction,identified_by,R_emaildomain'}
```

[46]:
```
vertex_files
```

[46]:
```
defaultdict(list,
            {'Card2': ['Vertex_Card2_0.csv'],
             'Address2': ['Vertex_Address2_0.csv'],
             'Card4': ['Vertex_Card4_0.csv'],
             'Card6': ['Vertex_Card6_0.csv'],
             'Address1': ['Vertex_Address1_0.csv'],
             'Card3': ['Vertex_Card3_0.csv'],
             'Card1': ['Vertex_Card1_0.csv'],
             'R_emaildomain': ['Vertex_R_emaildomain_0.csv'],
             'Transaction': ['Vertex_Transaction_0.csv'],
             'P_emaildomain': ['Vertex_P_emaildomain_0.csv'],
             'Card5': ['Vertex_Card5_0.csv'],
             'ProductCD': ['Vertex_ProductCD_0.csv']})
```

[47]:
```
edge_files
```

[47]:
```
defaultdict(list,
            {'Transaction,identified_by,Address2':
['Edge_Transaction,identified_by,Address2_0.csv'],
             'Transaction,identified_by,P_emaildomain':
['Edge_Transaction,identified_by,P_emaildomain_0.csv'],
             'Transaction,identified_by,Address1':
['Edge_Transaction,identified_by,Address1_0.csv'],
             'Transaction,identified_by,Card3':
['Edge_Transaction,identified_by,Card3_0.csv'],
             'Transaction,identified_by,Card6':
['Edge_Transaction,identified_by,Card6_0.csv'],
             'Transaction,identified_by,Card2':
['Edge_Transaction,identified_by,Card2_0.csv'],
             'Transaction,identified_by,Card4':
['Edge_Transaction,identified_by,Card4_0.csv'],
             'Transaction,identified_by,Card5':
['Edge_Transaction,identified_by,Card5_0.csv'],
             'Transaction,identified_by,ProductCD':
['Edge_Transaction,identified_by,ProductCD_0.csv'],
             'Transaction,identified_by,Card1':
['Edge_Transaction,identified_by,Card1_0.csv'],
             'Transaction,identified_by,R_emaildomain':
['Edge_Transaction,identified_by,R_emaildomain_0.csv']})
```

[48]:
```
NEPTUNE_TO_PD_DTYPES = {
    "Byte": pd.Int8Dtype(),
    "Short": pd.Int16Dtype(),
    "Int": Int32Dtype(),
    "Long": Int64Dtype(),
    "Float": Float32Dtype(),
    "Double": Float64Dtype(),
    "String": StringDtype(),
    "Vector": StringDtype(),
    "vector": StringDtype(),
}
```

[49]:
```
vertex_schemas = {}

for vertex_type in vertex_types:
    dfs = []
    directory = Path("./graph_data/")
    pattern = re.compile(rf"Vertex_{vertex_type}_(\d+)$")
    matching_files = [f for f in directory.iterdir() if pattern.match(f.stem)]
    for file in matching_files:
        df = pd.read_csv(
            file.resolve(),
            dtype=str,
            na_values=["", "nan", "NaN", "NULL", "null", "None", "none"],
            keep_default_na=True,
        )
        dfs.append(df)
    final_df = pd.concat(dfs, ignore_index=True) if len(dfs) > 1 else dfs[0]

    for column in final_df.columns:
        numeric_series = pd.to_numeric(
            final_df[column], errors="coerce"
        )
        if numeric_series.notna().any():
            non_null_values = numeric_series[numeric_series.notna()]
            if (non_null_values == non_null_values.astype(int)).all():
                final_df[column] = numeric_series.astype(
                    pd.Int64Dtype()
                )
            else:
                final_df[column] = numeric_series.astype(
                    pd.Float64Dtype()
                )
        else:
            final_df[column] = final_df[column].astype(pd.StringDtype())

    neptune_types = {}
```

```
    pandas_type_candidates = {}

    for col_name in final_df.columns.values.tolist():
        if ":" in col_name:
            neptune_type_candidate = col_name.split(":")[-1]
            if neptune_type_candidate in NEPTUNE_TO_PD_DTYPES:
                neptune_type = neptune_type_candidate
                neptune_types[col_name] = neptune_type
                pandas_type_candidates[col_name] = NEPTUNE_TO_PD_DTYPES[
                    neptune_type
                ]
            else:
                continue

    if len(pandas_type_candidates) == len(df.columns) - 2:
        pandas_types = pandas_type_candidates
    else:
        pandas_types = df.dtypes.astype(str).to_dict()

    vertex_schemas[vertex_type] = {
        "properties": list(df.columns),
        "pandas_dtypes": pandas_types,
        "neptune_types": neptune_types if neptune_types else None,
    }
```

[50]:
```
edge_schemas = {}

for relation_str in relation_types:
    if "," in relation_str:
        src_type, relation, dst_type = relation_str.split(",")
        relation_name = relation_str

    dfs = []
    directory = Path("./graph_data/")
    pattern = re.compile(rf"Edge_{relation_str}_(\d+)$")
    matching_files = [f for f in directory.iterdir() if pattern.match(f.stem)]

    for file in matching_files:
        df = pd.read_csv(
            file.resolve(),
            dtype=str,
            na_values=["", "nan", "NaN", "NULL", "null", "None", "none"],
            keep_default_na=True,
        )
        dfs.append(df)
    final_df = pd.concat(dfs, ignore_index=True) if len(dfs) > 1 else dfs[0]
```

```
    for column in final_df.columns:
        numeric_series = pd.to_numeric(
            final_df[column], errors="coerce"
        )
        if numeric_series.notna().any():
            non_null_values = numeric_series[numeric_series.notna()]
            if (non_null_values == non_null_values.astype(int)).all():
                final_df[column] = numeric_series.astype(
                    pd.Int64Dtype()
                )
            else:
                final_df[column] = numeric_series.astype(
                    pd.Float64Dtype()
                )
        else:
            final_df[column] = final_df[column].astype(pd.StringDtype())

    edge_schemas[relation_name] = {
    "properties": list(df.columns),
    "pandas_dtypes": df.dtypes.astype(str).to_dict(),
    }
```

[51]:
```
graph_schema = {}
graph_schema = {
    "vertex_types": list(vertex_types),
    "relation_types": list(relation_types),
    "vertex_files": dict(vertex_files),
    "edge_files": dict(edge_files),
    "vertex_schemas": vertex_schemas,
    "edge_schemas": edge_schemas,
    "input_format": "csv"
}
```

[52]:
```
relationships = {}

for relation_str in graph_schema['edge_files'].keys():
    print(relation_str)
    parts = relation_str.split(",")
    src_type, _, dst_type = parts
    edge_type = (src_type, relation, dst_type)
    relationships[edge_type] = {
        "source_type": src_type,
        "dest_type": dst_type,
        "relation": relation,
    }
```

```
Transaction,identified_by,Address2
Transaction,identified_by,P_emaildomain
```

```
Transaction,identified_by,Address1
Transaction,identified_by,Card3
Transaction,identified_by,Card6
Transaction,identified_by,Card2
Transaction,identified_by,Card4
Transaction,identified_by,Card5
Transaction,identified_by,ProductCD
Transaction,identified_by,Card1
Transaction,identified_by,R_emaildomain
```

[53]:
```
label_column = 'isFraud:Int'
target_type = 'Transaction'
learning_task = 'classification'

feature_suggestions = {}
feature_suggestions["vertices"] = {}
feature_suggestions["edges"] = {}

for vtype, ntype_info in graph_schema['vertex_schemas'].items():
    first_feature = True
    for col, dtype in ntype_info['pandas_dtypes'].items():
        if col in ["~id", "~label"]:
            continue

        feature_col = col
        feature_name = col.split(":")[0]

        if pd.api.types.is_numeric_dtype(dtype):
            if first_feature:
                feature_suggestions["vertices"][vtype] = []
                first_feature = False
            feature_suggestions["vertices"][vtype].append({
                "feature_col": feature_col,
                "feature_name": feature_name,
                "transform": {"name": "max_min_norm"},
            })
        elif pd.api.types.is_string_dtype(dtype):
            if first_feature:
                feature_suggestions["vertices"][vtype] = []
                first_feature = False
            if feature_col.endswith("embeddings:String"):
                feature_suggestions["vertices"][vtype].append({
                    "feature_col": feature_col,
                    "feature_name": feature_name,
                    "transform": {"name": "no-op", "separator": "|"},
                })
            else:
```

```
                feature_suggestions["vertices"][vtype].append({
                    "feature_col": feature_col,
                    "feature_name": feature_name,
                    "transform": {"name": "to_categorical"},
                })
```

[54]:
```
feature_suggestions["vertices"]
```

[54]:
```
{'Card5': [{'feature_col': 'id_embeddings:String',
   'feature_name': 'id_embeddings',
   'transform': {'name': 'no-op', 'separator': '|'}}],
 'ProductCD': [{'feature_col': 'id_embeddings:String',
   'feature_name': 'id_embeddings',
   'transform': {'name': 'no-op', 'separator': '|'}}],
 'Transaction': [{'feature_col': 'isFraud:Int',
   'feature_name': 'isFraud',
   'transform': {'name': 'max_min_norm'}},
  {'feature_col': 'TransactionDT:String',
   'feature_name': 'TransactionDT',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M1:String',
   'feature_name': 'M1',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M2:String',
   'feature_name': 'M2',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M3:String',
   'feature_name': 'M3',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M4:String',
   'feature_name': 'M4',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M5:String',
   'feature_name': 'M5',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M6:String',
   'feature_name': 'M6',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M7:String',
   'feature_name': 'M7',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M8:String',
   'feature_name': 'M8',
   'transform': {'name': 'to_categorical'}},
  {'feature_col': 'M9:String',
   'feature_name': 'M9',
   'transform': {'name': 'to_categorical'}},
```

```
 {'feature_col': 'C1:Float',
  'feature_name': 'C1',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C2:Float',
  'feature_name': 'C2',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C3:Float',
  'feature_name': 'C3',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C4:Float',
  'feature_name': 'C4',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C5:Float',
  'feature_name': 'C5',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C6:Float',
  'feature_name': 'C6',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C7:Float',
  'feature_name': 'C7',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C8:Float',
  'feature_name': 'C8',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C9:Float',
  'feature_name': 'C9',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C10:Float',
  'feature_name': 'C10',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C11:Float',
  'feature_name': 'C11',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C12:Float',
  'feature_name': 'C12',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C13:Float',
  'feature_name': 'C13',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'C14:Float',
  'feature_name': 'C14',
  'transform': {'name': 'max_min_norm'}},
 {'feature_col': 'TransactionAmt:Float',
  'feature_name': 'TransactionAmt',
  'transform': {'name': 'max_min_norm'}}],
'Address2': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
```

```
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'Card3': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'R_emaildomain': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'Card4': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'Card2': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'Address1': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'Card1': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'P_emaildomain': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}],
 'Card6': [{'feature_col': 'id_embeddings:String',
  'feature_name': 'id_embeddings',
  'transform': {'name': 'no-op', 'separator': '|'}}]}
```

[55]:
```
cols_to_keep = {
        "Transaction":
        [
            "~id",
            "~label",
            "isFraud:Int",
        ]
        +
        [f"C{idx}:Float" for idx in range(1, 15)]
        + ["TransactionAmt:Float"]
        +
        [f"{CAT_COL}:String" for CAT_COL in CAT_COLS.split(",")]
    }

cols_to_skip = None
confirmed_features_final = {}
confirmed_features_final['vertices'] = {}
labels_config_final = {}
labels_config_final['vertices'] = {}

for vtype, properties in sorted(feature_suggestions["vertices"].items()):
```

```
    confirmed_features = []
    labels_list = []

    for gc_feature_dict in properties:
        should_skip = (cols_to_skip and gc_feature_dict["feature_col"] in␣
↪cols_to_skip)
        should_not_keep = (cols_to_keep and gc_feature_dict["feature_col"] not␣
↪in cols_to_keep)
        if (
                vtype == target_type
                and gc_feature_dict["feature_col"] == label_column
            ):
            split_config = {
                        "custom_split_filenames": {
                            "train": os.path.join("./data_splits", "train_ids.
↪parquet"),
                            "valid": os.path.join("./data_splits", "val_ids.
↪parquet"),
                            "test": os.path.join("./data_splits", "test_ids.
↪parquet"),
                            "column": ["nid"],
                        }
            }

            labels_list.append(
                    {
                        "task_type": learning_task,
                        "label_col": label_column,
                        "label_name": label_column.split(":")[0],
                        **split_config,
                    }
            )
        else:
            confirmed_features.append(gc_feature_dict)

    confirmed_features_final['vertices'][vtype] = confirmed_features
    labels_config_final['vertices'][vtype] = labels_list
```

[56]:
```
relation_to_schema = {}

for relation_str, schema in graph_schema['edge_schemas'].items():
    relation_to_schema[relation_str] = schema

    if "," in relation_str:
        parts = relation_str.split(",")
        src_type, relation, dst_type = parts
```

```
        if {src_type, dst_type}.issubset(graph_schema['vertex_types']):
            relation_to_schema[relation] = schema
```

[57]:
```
for etype, _ in relationships.items():
    _, relation, _ = etype
    etype_str = ",".join(etype)

    if relation in relation_to_schema:
        etype_info = relation_to_schema[relation]

    for col, dtype in etype_info['pandas_dtypes'].items():
        if col in ["~id", "~from", "~to", "~label"]:
            continue

confirmed_features_final['edges'] = dict()
```

[58]:
```
GSConfig = TypedDict("GSConfig", {"version": str, "nodes": list, "edges": list})
gs_config = GSConfig(version="gconstruct-v0.1", nodes=[], edges=[])
```

[59]:
```
for vtype in graph_schema['vertex_types']:
    format_name = os.path.
 ↪splitext(graph_schema['vertex_files'][vtype][0])[-1][1:]
    format_dict = {"name": format_name, "separator": ","}

    vertex_config: dict[str, Any] = {
            "node_type": vtype,
            "format": format_dict,
            "files": graph_schema['vertex_files'][vtype],
            "node_id_col": "~id",
        }

    if confirmed_features_final["vertices"].get(vtype):
        vertex_config["features"] = confirmed_features_final["vertices"][vtype]

    if labels_config_final["vertices"].get(vtype):
        vertex_config["labels"] = labels_config_final["vertices"][vtype]

    gs_config["nodes"].append(vertex_config)
```

[60]:
```
edge_config = {}
```

[61]:
```
for src, rel, dst in relationships.keys():
    canonical_etype = ",".join([src, rel, dst])

    if canonical_etype in graph_schema['edge_files']:
        efiles = graph_schema['edge_files'][canonical_etype]
```

```
    format_name = os.path.splitext(efiles[0])[-1][1:]
    format_dict = {"name": format_name, "separator": ","}

    edge_config = {
            "relation": [
                src,
                rel,
                dst,
            ],
            "format": format_dict,
            "files": efiles,
            "source_id_col": "~from",
            "dest_id_col": "~to",
        }


    gs_config["edges"].append(edge_config)

    edge_config = {
                "relation": [
                    dst,
                    rel + "-rev",
                    src,
                ],
                "format": format_dict,
                "files": efiles,
                "source_id_col": "~to",
                "dest_id_col": "~from",
    }
    gs_config["edges"].append(edge_config)
```

[62]:
```
gs_config["edges"]
```

[62]:
```
[{'relation': ['Transaction', 'identified_by', 'Address2'],
  'format': {'name': 'csv', 'separator': ','},
  'files': ['Edge_Transaction,identified_by,Address2_0.csv'],
  'source_id_col': '~from',
  'dest_id_col': '~to'},
 {'relation': ['Address2', 'identified_by-rev', 'Transaction'],
  'format': {'name': 'csv', 'separator': ','},
  'files': ['Edge_Transaction,identified_by,Address2_0.csv'],
  'source_id_col': '~to',
  'dest_id_col': '~from'},
 {'relation': ['Transaction', 'identified_by', 'P_emaildomain'],
  'format': {'name': 'csv', 'separator': ','},
  'files': ['Edge_Transaction,identified_by,P_emaildomain_0.csv'],
  'source_id_col': '~from',
```

```
 'dest_id_col': '~to'},
{'relation': ['P_emaildomain', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,P_emaildomain_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Address1'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Address1_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Address1', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Address1_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Card3'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card3_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Card3', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card3_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Card6'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card6_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Card6', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card6_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Card2'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card2_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Card2', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card2_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Card4'],
```

```
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card4_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Card4', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card4_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Card5'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card5_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Card5', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card5_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'ProductCD'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,ProductCD_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['ProductCD', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,ProductCD_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'Card1'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card1_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['Card1', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,Card1_0.csv'],
 'source_id_col': '~to',
 'dest_id_col': '~from'},
{'relation': ['Transaction', 'identified_by', 'R_emaildomain'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,R_emaildomain_0.csv'],
 'source_id_col': '~from',
 'dest_id_col': '~to'},
{'relation': ['R_emaildomain', 'identified_by-rev', 'Transaction'],
 'format': {'name': 'csv', 'separator': ','},
 'files': ['Edge_Transaction,identified_by,R_emaildomain_0.csv'],
```

```
  'source_id_col': '~to',
  'dest_id_col': '~from'}]
```

[63]:
```
CONFIG_FILENAME = "ieee-cis-gconstruct-node-classification.json"

with open(CONFIG_FILENAME, "w") as f:
    json.dump(gs_config, f, indent=2)
```

[4]:
```
import sys

PYTHON = sys.executable
```

[5]:
```
%cd graph_data
```

```
/home/ec2-user/SageMaker/graph_data
```

[6]:
```
!{PYTHON} -m graphstorm.gconstruct.construct_graph \
          --conf-file  ieee-cis-gconstruct-node-classification.json \
          --output-dir ../ieee_gs \
          --num-parts 1 \
          --graph-name ieee-cis
```

```
/bin/bash: switchml: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `switchml'
/bin/bash: module: line 1: syntax error: unexpected end of file
/bin/bash: error importing function definition for `module'
DGL backend not selected or invalid.  Assuming PyTorch for now.
Setting the default backend to "pytorch". You can change it in the
~/.dgl/config.json file or export the DGLBACKEND environment variable.  Valid
options are: pytorch, mxnet, tensorflow (all lowercase)
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-
packages/outdated/__init__.py:36: UserWarning: pkg_resources is deprecated as an
API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The
pkg_resources package is slated for removal as early as 2025-11-30. Refrain from
using this package or pin to Setuptools<81.
  from pkg_resources import parse_version
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/runpy.py:126:
RuntimeWarning: 'graphstorm.gconstruct.construct_graph' found in sys.modules
after import of package 'graphstorm.gconstruct', but prior to execution of
'graphstorm.gconstruct.construct_graph'; this may result in unpredictable
behaviour
  warn(RuntimeWarning(msg))
INFO:root:Parsing config file as GConstruct config
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-
packages/graphstorm/gconstruct/transform.py:649: RuntimeWarning: invalid value
encountered in cast
  max_val[max_val > self._max_bound] = self._max_bound
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-
```

```
packages/graphstorm/gconstruct/transform.py:656: RuntimeWarning: invalid value
encountered in cast
  min_val[min_val < self._min_bound] = self._min_bound
WARNING:root:Overwrote the existing ../ieee_gs/data_transform_new.json file,
which was generated in the previous graph construction command. Use the
--output-conf-file argument to specify a different location if not want to
overwrite the existing configuration file.
INFO:root:The graph has 12 node types and 22 edge types.
INFO:root:Node type Address1 has 333 nodes
INFO:root:Node type Address2 has 75 nodes
INFO:root:Node type Card1 has 13553 nodes
INFO:root:Node type Card2 has 501 nodes
INFO:root:Node type Card3 has 115 nodes
INFO:root:Node type Card4 has 5 nodes
INFO:root:Node type Card5 has 120 nodes
INFO:root:Node type Card6 has 5 nodes
INFO:root:Node type P_emaildomain has 60 nodes
INFO:root:Node type ProductCD has 5 nodes
INFO:root:Node type R_emaildomain has 61 nodes
INFO:root:Node type Transaction has 590540 nodes
INFO:root:Edge type ('Address1', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Address2', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Card1', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Card2', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Card3', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Card4', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Card5', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('Card6', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('P_emaildomain', 'identified_by-rev', 'Transaction') has
590540 edges
INFO:root:Edge type ('ProductCD', 'identified_by-rev', 'Transaction') has 590540
edges
INFO:root:Edge type ('R_emaildomain', 'identified_by-rev', 'Transaction') has
590540 edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Address1') has 590540
edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Address2') has 590540
edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Card1') has 590540 edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Card2') has 590540 edges
```

```
INFO:root:Edge type ('Transaction', 'identified_by', 'Card3') has 590540 edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Card4') has 590540 edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Card5') has 590540 edges
INFO:root:Edge type ('Transaction', 'identified_by', 'Card6') has 590540 edges
INFO:root:Edge type ('Transaction', 'identified_by', 'P_emaildomain') has 590540
edges
INFO:root:Edge type ('Transaction', 'identified_by', 'ProductCD') has 590540
edges
INFO:root:Edge type ('Transaction', 'identified_by', 'R_emaildomain') has 590540
edges
INFO:root:Node type Card5 has features: ['id_embeddings'].
INFO:root:Node type ProductCD has features: ['id_embeddings'].
INFO:root:Node type Transaction has features: ['TransactionDT', 'M1', 'M2',
'M3', 'M4', 'M5', 'M6', 'M7', 'M8', 'M9', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6',
'C7', 'C8', 'C9', 'C10', 'C11', 'C12', 'C13', 'C14', 'TransactionAmt',
'train_mask', 'val_mask', 'test_mask', 'isFraud:Int'].
INFO:root:Train/val/test on Transaction with mask train_mask, val_mask,
test_mask: 472432, 59054, 59054
INFO:root:Note: Custom train, validate, test mask information for nodes are not
collected.
INFO:root:Node type Address2 has features: ['id_embeddings'].
INFO:root:Node type Card3 has features: ['id_embeddings'].
INFO:root:Node type R_emaildomain has features: ['id_embeddings'].
INFO:root:Node type Card4 has features: ['id_embeddings'].
INFO:root:Node type Card2 has features: ['id_embeddings'].
INFO:root:Node type Address1 has features: ['id_embeddings'].
INFO:root:Node type Card1 has features: ['id_embeddings'].
INFO:root:Node type P_emaildomain has features: ['id_embeddings'].
INFO:root:Node type Card6 has features: ['id_embeddings'].
The graph has 12 node types and balance among 14 types
Converting to homogeneous graph takes 0.179s, peak mem: 327.768 GB
Save partitions: 1.350 seconds, peak memory: 328.022 GB
There are 12991880 edges in the graph and 0 edge cuts for 1 partitions.
INFO:root:Graph construction generated new node IDs for 'Card5'. The ID map is
saved under ../ieee_gs/raw_id_mappings/Card5.
INFO:root:Graph construction generated new node IDs for 'ProductCD'. The ID map
is saved under ../ieee_gs/raw_id_mappings/ProductCD.
INFO:root:Graph construction generated new node IDs for 'Transaction'. The ID
map is saved under ../ieee_gs/raw_id_mappings/Transaction.
INFO:root:Graph construction generated new node IDs for 'Address2'. The ID map
is saved under ../ieee_gs/raw_id_mappings/Address2.
INFO:root:Graph construction generated new node IDs for 'Card3'. The ID map is
saved under ../ieee_gs/raw_id_mappings/Card3.
INFO:root:Graph construction generated new node IDs for 'R_emaildomain'. The ID
map is saved under ../ieee_gs/raw_id_mappings/R_emaildomain.
INFO:root:Graph construction generated new node IDs for 'Card4'. The ID map is
saved under ../ieee_gs/raw_id_mappings/Card4.
INFO:root:Graph construction generated new node IDs for 'Card2'. The ID map is
```

```
saved under ../ieee_gs/raw_id_mappings/Card2.
INFO:root:Graph construction generated new node IDs for 'Address1'. The ID map
is saved under ../ieee_gs/raw_id_mappings/Address1.
INFO:root:Graph construction generated new node IDs for 'Card1'. The ID map is
saved under ../ieee_gs/raw_id_mappings/Card1.
INFO:root:Graph construction generated new node IDs for 'P_emaildomain'. The ID
map is saved under ../ieee_gs/raw_id_mappings/P_emaildomain.
INFO:root:Graph construction generated new node IDs for 'Card6'. The ID map is
saved under ../ieee_gs/raw_id_mappings/Card6.
```

[ ]: