3.5 Bonus: Node Embedding and Prediction

In this section, we will create node embeddings using the Node2Vec algorithm. Then, we will train a classifier to try to predict the region of a node (name_1).

3.5.1 First, create a Cypher Projection of the name of your choice including (:Area_4) nodes and [:trip] edges. Don't forget to include the 'NB' attribute which will be used as a weight for the graph traversal by Node2Vec.

3.5.2 Then, use the Node2Vec algorithm to create node's embeddings. Make sure to specify the dimensions of embeddings (embeddingDimension), the length of the walks and the number of iterations per node (walkLength and iterations), the probabilities (or weights) of returning to the previous node and exploring a new node (returnFactor and inOutFactor) and, above all, the attribute that will be used as a weight for relationships (relationshipWeightProperty).

3.5.3 You should obtain the following table, which you can import in CSV format. This table contains the names of the region's attributes (name_1) with their corresponding embeddings for each node in the graph.

	city	region	embedding
1	"Yssingeaux"	"Haute-Loire"	[0.32235315442085266, 0.36172258853912354, -0.24609294533729553, 0.36301079392433167, -0.053950235247612, 0.16393014788627625, 0.03348840773105621, 0.1918618232
2	"Saint-Lô"	"Manche"	[0.5904310345649719, -0.06822948902845383, 0.038585860282182693, 0.06505212932825089, -0.27604103088378906, -0.1503499448299408, 0.3430739939212799, -0.172385409
3	"Alençon"	"Orne"	[0.6049037575721741, -0.0782601460814476, -0.27620941400527954, -0.06567700952291489, -0.041889749467372894, -0.2928217649459839, 0.4771932065486908, 0.103934347(
4	"Béthune"	"Pas-de-Calais"	[0.8334031105041504, 0.2682449519634247, 0.06596867740154266, 0.1863444596529007, -0.08619867265224457, 0.2873293459415436, 0.07791266590356827, 0.1095872819423

Figure 3.2: Nodes Embeddings

3.5.4 You now need to switch to a python editor (the one of your choice for the rest of the tutorial). First, make the following imports:

```
1 import pandas as pd
2 import ast
3 import numpy as np
4 from sklearn.model_selection import train_test_split
5 from sklearn.ensemble import RandomForestClassifier
6 from sklearn.metrics import top_k_accuracy_score
7
```

3.5.5 Then, load the CSV using pandas and create a dataframe. Make sure to convert columns types if needed (e.g. embeddings as numpy arrays).

3.5.6 Create inputs (embeddings) and labels (regions) and make a split between a training set and a test set.

3.5.7 Create inputs (embeddings) and labels (regions) and make a split between a training set and a test set.

3.5.8 Create a ML model and train it! (make sure to chose your hyper-parameters wisely)

3.5.9 Finally, evaluate your model on the test set. Select an appropriate metric (I personally chose Top-K Accuracy and varied the value of K).

If you completed the steps correctly, you should get results comparable to mine (73% Top-1 Accuracy and 95% Top-5 Accuracy). You can try to improve these results by modifying the embedding method (e.g. GraphSage or FastRP). You can also try to change the values of the hyper-parameters (for both embedding step or for the model) or chose another prediction model (e.g. a neural network instead of machine learning). Good Luck!