Graph Features

April 29, 2022

1 Creating a feature matrix from a networkx graph

In this notebook we will look at a few ways to quickly create a feature matrix from a networkx graph.

```
In [ ]: import networkx as nx
        import pandas as pd

        G = nx.read_gpickle('major_us_cities')
```

1.1 Node based features

```
In [ ]: G.nodes(data=True)
```

```
In [ ]: # Initialize the dataframe, using the nodes as the index
        df = pd.DataFrame(index=G.nodes())
```

1.1.1 Extracting attributes

Using `nx.get_node_attributes` it's easy to extract the node attributes in the graph into DataFrame columns.

```
In [ ]: df['location'] = pd.Series(nx.get_node_attributes(G, 'location'))
        df['population'] = pd.Series(nx.get_node_attributes(G, 'population'))

        df.head()
```

1.1.2 Creating node based features

Most of the networkx functions related to nodes return a dictionary, which can also easily be added to our dataframe.

```
In [ ]: df['clustering'] = pd.Series(nx.clustering(G))
        df['degree'] = pd.Series(G.degree())

        df
```

2 Edge based features

```
In [ ]: G.edges(data=True)
```

```
In [ ]: # Initialize the dataframe, using the edges as the index
        df = pd.DataFrame(index=G.edges())
```

2.0.1 Extracting attributes

Using `nx.get_edge_attributes`, it's easy to extract the edge attributes in the graph into DataFrame columns.

```
In [ ]: df['weight'] = pd.Series(nx.get_edge_attributes(G, 'weight'))

        df
```

2.0.2 Creating edge based features

Many of the networkx functions related to edges return a nested data structures. We can extract the relevant data using list comprehension.

```
In [ ]: df['preferential attachment'] = [i[2] for i in nx.preferential_attachment(G, df.index)]

        df
```

In the case where the function expects two nodes to be passed in, we can map the index to a lamda function.

```
In [ ]: df['Common Neighbors'] = df.index.map(lambda city: len(list(nx.common_neighbors(G, city[

        df
```