School of Science & Engineering

Data Engineering and Visualization

CSC 3356

Project #3

Asmaa Dalil

Instructor : Prof. Tajjeeddine Rachidi

Project Statement:

In this project you are required to identify 2 very large datasets (Tabular + (Network or Field)) and perform their visualization. For this, you will need to follow the What/Why/How steps/methodology seen in class, namely describe the dataset, identify 2 tasks (action and target) for each dataset using the exact terminology seen in class specifying the encoding, manipulations, eventual facets, encoding and channels.

The What/Why/How methodology

In data visualization, it provides a structured approach to understanding and communicating the characteristics and potential of a dataset:

What:

- *Definition*: Understand the dataset's key components, attributes, and structure.
- *Objective*: Identify data types, dimensions, and basic statistics.

Why:

- *Definition*: Uncover the dataset's significance or relevance.
- *Objective*: Highlight unique insights, trends, or patterns within the data.
- *Visual Representation*: Showcase interesting aspects through visualizations like trend plots, correlation matrices, or outlier analysis.

How:

- *Definition*: Outline potential applications or analyses using the dataset.
- *Objective*: Provide guidance on tasks, questions, or hypotheses the dataset can address.
- *Visual Representation*: Develop visualizations demonstrating different analytical approaches, such as interactive dashboards, exploratory plots, or predictive models.

This method can dissect a dataset, revealing its constituent elements and enabling observers to discern patterns or connections within. One of its most notable advantages lies in its capacity to elucidate complex datasets, bringing to the forefront aspects that were previously unclear or unknown.

Geospatial Dataset :

The dataset consists of six files , the file we used is the world.shp file. It is a world map.

The metadata:

FIPS: The FIPS (Federal Information Processing Standards) code for the country.

ISO2: The ISO 3166-1 alpha-2 code for the country.

ISO3: The ISO 3166-1 alpha-3 code for the country.

UN: The United Nations code for the country.

NAME: The name of the country.

AREA: The area of the country in square units.

POP2005: The population of the country in the year 2005.

REGION: The region code of the country.

SUBREGION: The subregion code of the country.

LON: The longitude coordinate of the country.

LAT: The latitude coordinate of the country.

geometry: The geometric representation of the country, either as a Polygon or MultiPolygon.

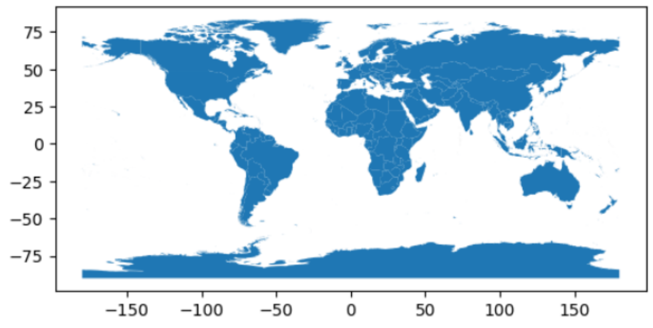
I tried to explore the dataset first and plot it in order to understand it:

```
print("\nSummary statistics of the GeoDataFrame's geometry:")
print(world_data.geometry.describe())
```

```
Summary statistics of the GeoDataFrame's geometry:
count                                                   246
unique                                                  246
top       MULTIPOLYGON (((-61.686668 17.024441000000138,...
freq                                                      1
Name: geometry, dtype: object
```

```
world_data.plot()
```

```
<Axes: >
```



Task 1: Geospatial Data Area Calculation and Visualization with Bar Chart

What: calculate the area for each geometry in the GeoSeries world_data and assigns the result to a new column named 'area' in the DataFrame world_data.

<u>Why</u>: To provide a visual understanding of the relative sizes of different countries based on their area. Data represented in the world_data *GeoSeries* or *GeoDataFrame*. Calculating the area of each geometry can be useful for various spatial analysis tasks, such as determining the size of countries or regions.

<u>How</u>:

Target: Calculate and visualize country areas.

Why: To provide a visual understanding of the relative sizes of different countries based on their area.

<u>How</u>:

Target: Calculate and visualize country areas.

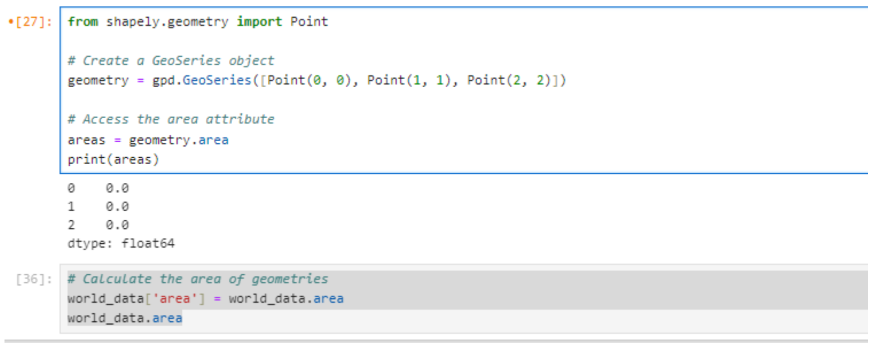
Action:

- Import geospatial data using GeoPandas and create a GeoSeries object representing point geometries.
- Calculate the area of *geometries* and add it as a new attribute to the dataset.
- Convert the dataset to the *EPSG:3857* projection for web mapping compatibility.
- Sort the data by area to facilitate visualization.
- Plot the bar chart with countries on the x-axis and their respective areas on the y-axis.

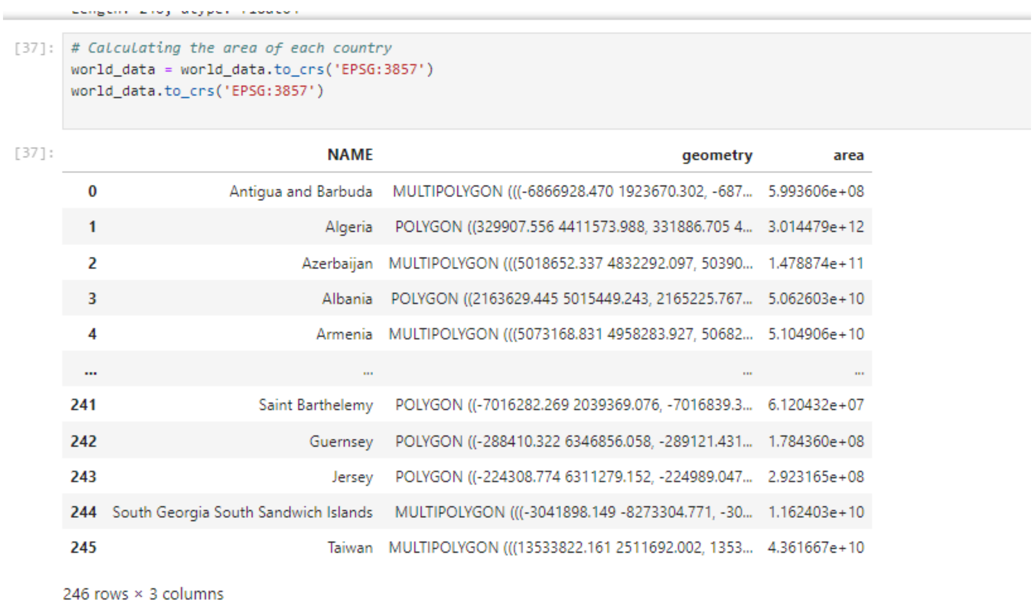
<u>Terminology</u>:

- Task Actions: Calculate and visualize country areas.
- Targets: Provide a clear representation of country areas.
- Attributes: Country names, area size.

<u>Source Code:</u>



```
•[27]: from shapely.geometry import Point

       # Create a GeoSeries object
       geometry = gpd.GeoSeries([Point(0, 0), Point(1, 1), Point(2, 2)])

       # Access the area attribute
       areas = geometry.area
       print(areas)

       0    0.0
       1    0.0
       2    0.0
       dtype: float64

 [36]: # Calculate the area of geometries
       world_data['area'] = world_data.area
       world_data.area
```



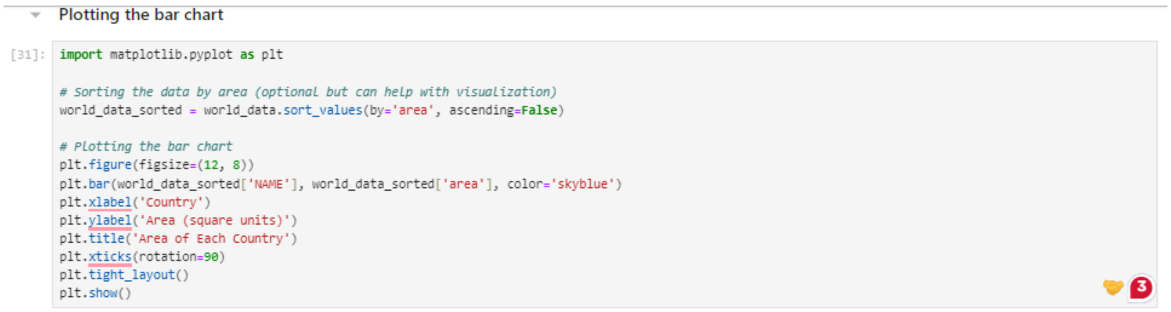
```
[37]: # Calculating the area of each country
      world_data = world_data.to_crs('EPSG:3857')
      world_data.to_crs('EPSG:3857')
```

[37]:

	NAME	geometry	area
0	Antigua and Barbuda	MULTIPOLYGON (((-6866928.470 1923670.302, -687...	5.993606e+08
1	Algeria	POLYGON ((329907.556 4411573.988, 331886.705 4...	3.014479e+12
2	Azerbaijan	MULTIPOLYGON (((5018652.337 4832292.097, 50390...	1.478874e+11
3	Albania	POLYGON ((2163629.445 5015449.243, 2165225.767...	5.062603e+10
4	Armenia	MULTIPOLYGON (((5073168.831 4958283.927, 50682...	5.104906e+10
...	...	...	...
241	Saint Barthelemy	POLYGON ((-7016282.269 2039369.076, -7016839.3...	6.120432e+07
242	Guernsey	POLYGON ((-288410.322 6346856.058, -289121.431...	1.784360e+08
243	Jersey	POLYGON ((-224308.774 6311279.152, -224989.047...	2.923165e+08
244	South Georgia South Sandwich Islands	MULTIPOLYGON (((-3041898.149 -8273304.771, -30...	1.162403e+10
245	Taiwan	MULTIPOLYGON (((13533822.161 2511692.002, 1353...	4.361667e+10

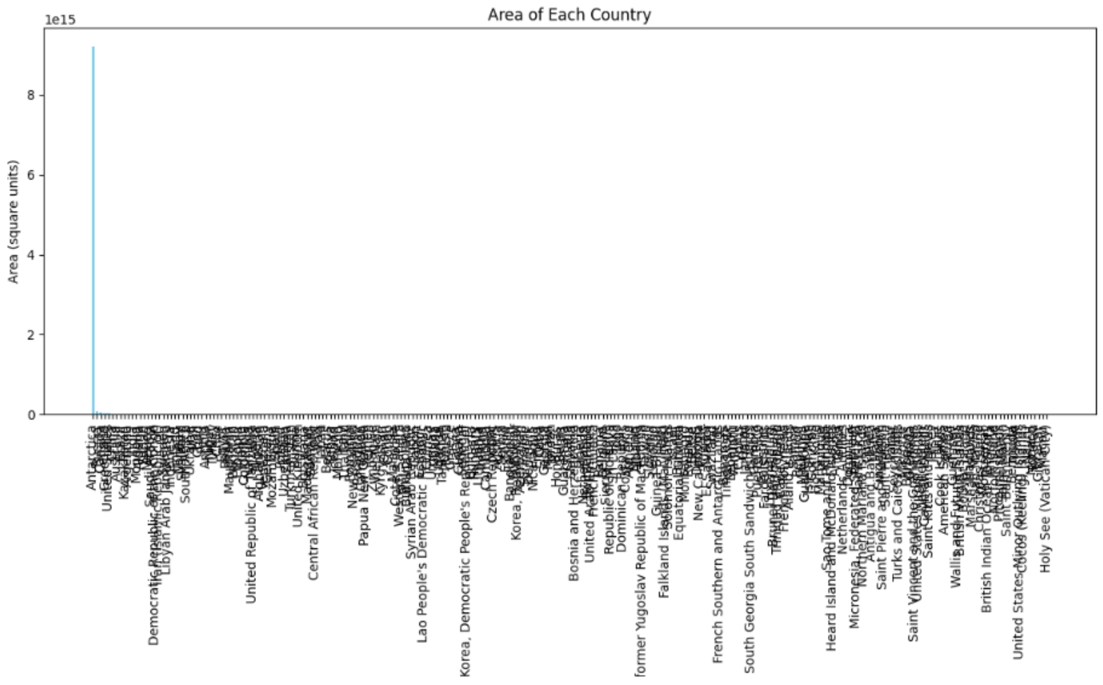
246 rows × 3 columns

Visualization :



Plotting the bar chart

```
[31]: import matplotlib.pyplot as plt

      # Sorting the data by area (optional but can help with visualization)
      world_data_sorted = world_data.sort_values(by='area', ascending=False)

      # Plotting the bar chart
      plt.figure(figsize=(12, 8))
      plt.bar(world_data_sorted['NAME'], world_data_sorted['area'], color='skyblue')
      plt.xlabel('Country')
      plt.ylabel('Area (square units)')
      plt.title('Area of Each Country')
      plt.xticks(rotation=90)
      plt.tight_layout()
      plt.show()
```



Task 2 : Analyze the trend of population density (population per unit area) over time for different regions using a moving average chart.

What: Calculate the population density for each region by dividing the population (POP2005) by the area (AREA) and then compute the moving average of population density over a specified window size.
The moving average will help smooth out fluctuations and identify trends in population density over time.

Why: Understanding population density trends is crucial for various analyses, such as urban planning, resource allocation, and demographic studies.
Using a moving average allows us to focus on long-term trends while filtering out short-term fluctuations, providing a clearer picture of population density dynamics.
How:

Target:

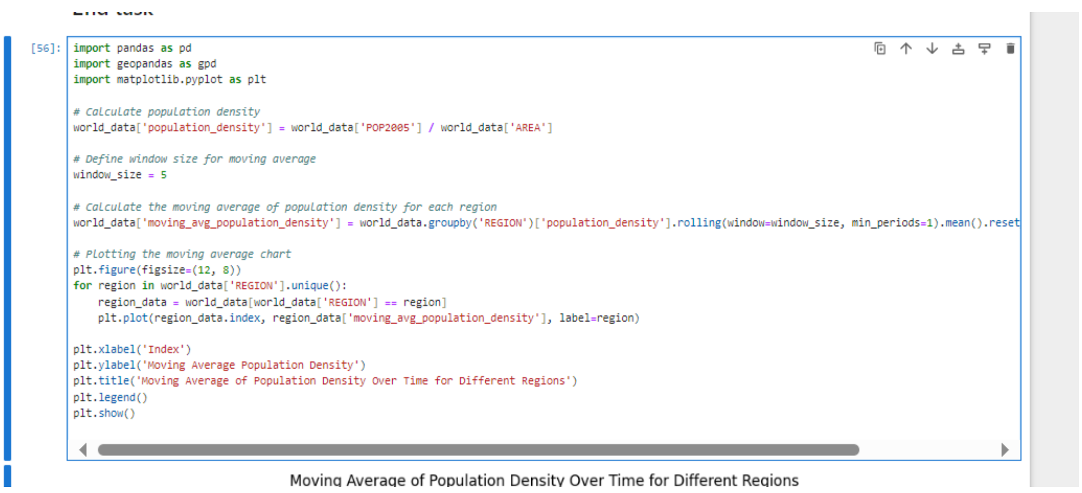
- The target is the dataset containing attributes such as region (REGION), population (POP2005), and area (AREA).

Manipulations: Calculate the population density for each region by dividing population (POP2005) by area (AREA).

- Compute the moving average of population density over a specified window size.
- Eventual Facets:
- The eventual facet is to visualize the trend of population density over time for different regions using a moving average chart.

Encoding and Channels:
- Encoding: Population density data will be encoded along the y-axis (dependent variable), and time (or index) will be encoded along the x-axis (independent variable).
- Channels: Line chart will be used as the channel to represent the trend of population density over time. Color can be used to differentiate between regions.

Source Code:

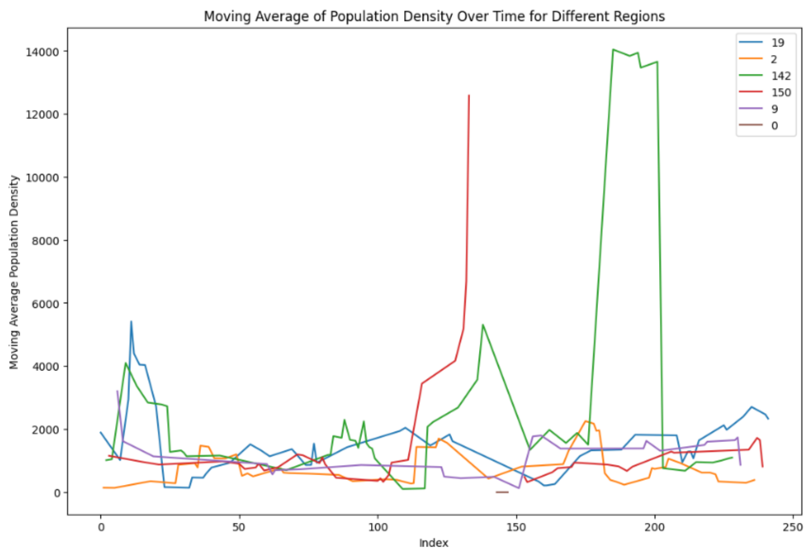


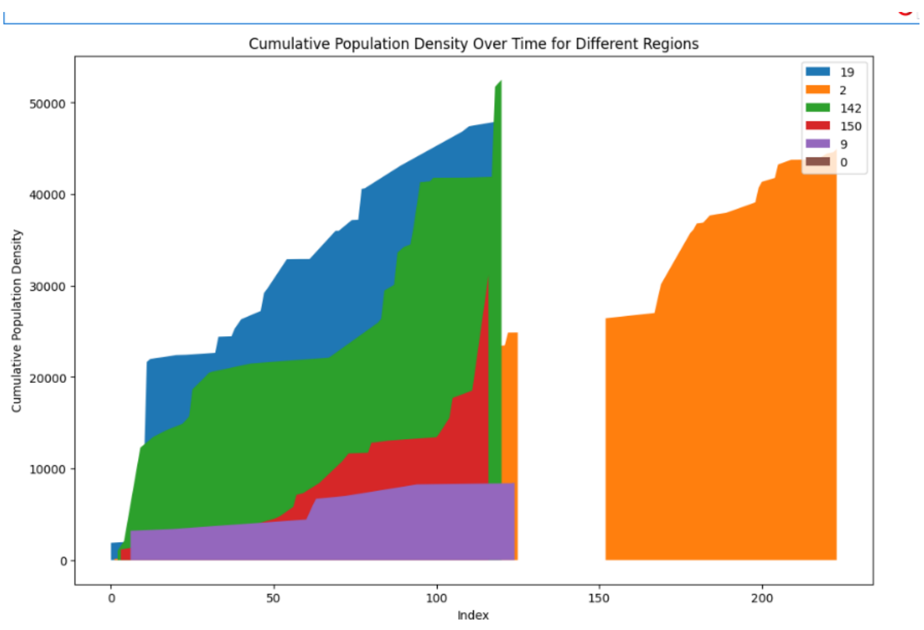
```
[56]: import pandas as pd
      import geopandas as gpd
      import matplotlib.pyplot as plt

      # Calculate population density
      world_data['population_density'] = world_data['POP2005'] / world_data['AREA']

      # Define window size for moving average
      window_size = 5

      # Calculate the moving average of population density for each region
      world_data['moving_avg_population_density'] = world_data.groupby('REGION')['population_density'].rolling(window=window_size, min_periods=1).mean().reset

      # Plotting the moving average chart
      plt.figure(figsize=(12, 8))
      for region in world_data['REGION'].unique():
          region_data = world_data[world_data['REGION'] == region]
          plt.plot(region_data.index, region_data['moving_avg_population_density'], label=region)

      plt.xlabel('Index')
      plt.ylabel('Moving Average Population Density')
      plt.title('Moving Average of Population Density Over Time for Different Regions')
      plt.legend()
      plt.show()
```

Moving Average of Population Density Over Time for Different Regions

- We first calculate the population density by dividing the population (POP2005) by the area (AREA).Then, we calculate the moving average of population density for each region using the rolling() function.Finally, we plot the moving average chart for each region, with the x-axis representing the index and the y-axis representing the moving average population density. Each region's trend is represented by a line on the plot, with a legend to distinguish between regions.

Visualization :





Network dataset:

The dataset named "miserables" seems to consist of the following columns:

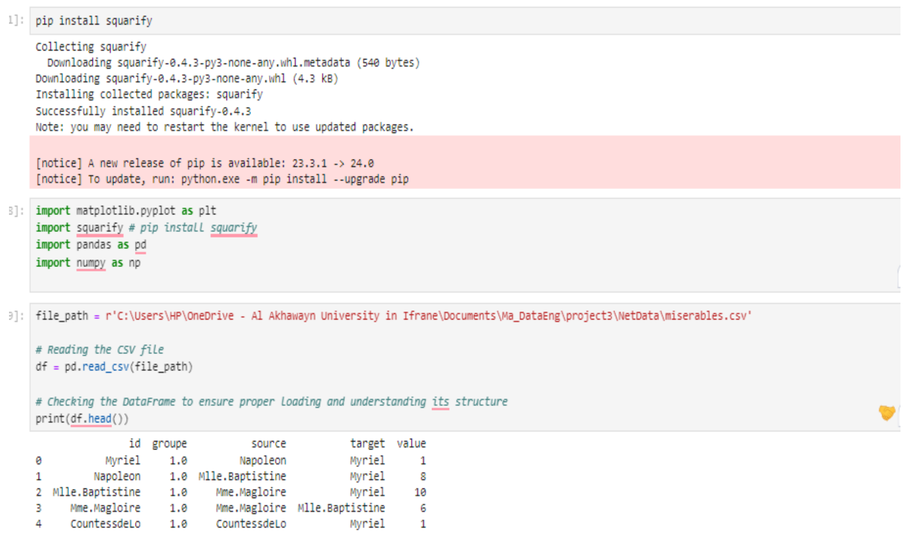
id: Identifier for each node.

groupe: Possibly a categorical attribute indicating the group membership of nodes.

source: Identifier for the source node of a directed edge.

target: Identifier for the target node of a directed edge.

value: Numerical attribute, likely representing the weight or strength of the connection between source and target.



```
1]: pip install squarify

Collecting squarify
  Downloading squarify-0.4.3-py3-none-any.whl.metadata (540 bytes)
Downloading squarify-0.4.3-py3-none-any.whl (4.3 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.3
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
3]: import matplotlib.pyplot as plt
import squarify # pip install squarify
import pandas as pd
import numpy as np
```

```
3]: file_path = r'C:\Users\HP\OneDrive - Al Akhawayn University in Ifrane\Documents\Ma_DataEng\project3\NetData\miserables.csv'

# Reading the CSV file
df = pd.read_csv(file_path)

# Checking the DataFrame to ensure proper loading and understanding its structure
print(df.head())
```

```
             id  groupe         source          target  value
0        Myriel     1.0       Napoleon          Myriel      1
1      Napoleon     1.0  Mlle.Baptistine        Myriel      8
2  Mlle.Baptistine  1.0     Mme.Magloire         Myriel     10
3     Mme.Magloire  1.0     Mme.Magloire  Mlle.Baptistine     6
4     CountessdeLo  1.0     CountessdeLo         Myriel      1
```

Task1 :using adjacency matrix

What:

- Data Types: Network data involving nodes and directed edges.
- Dataset Types: A network with possible categorical attributes for nodes (groups) and quantitative attributes for edges (values).

Why:

Target: Understanding the overall connectivity and group-based interactions within the network.

Actions:

- Discover: Identify prominent patterns or clusters.
- Browse: Explore the interactions between different nodes, especially focusing on group interactions.
- Summarize: Provide an overview of the network structure.

How:

Encoding:

- Nodes will be encoded by their ids.
- Edges will be encoded by their existence and weights between nodes.

Manipulation:

- Filtering to view specific groups.
- Sorting nodes to reveal patterns more clearly.

Facet:

- Multiple matrix views could be used to compare different group interactions.

Channels:

- Color saturation to denote edge weights.
- Positional differences to denote node connections.

Scale:

An adjacency matrix for a network typically scales quadratically with the number of nodes, as it is an n x n matrix where n is the number of nodes.

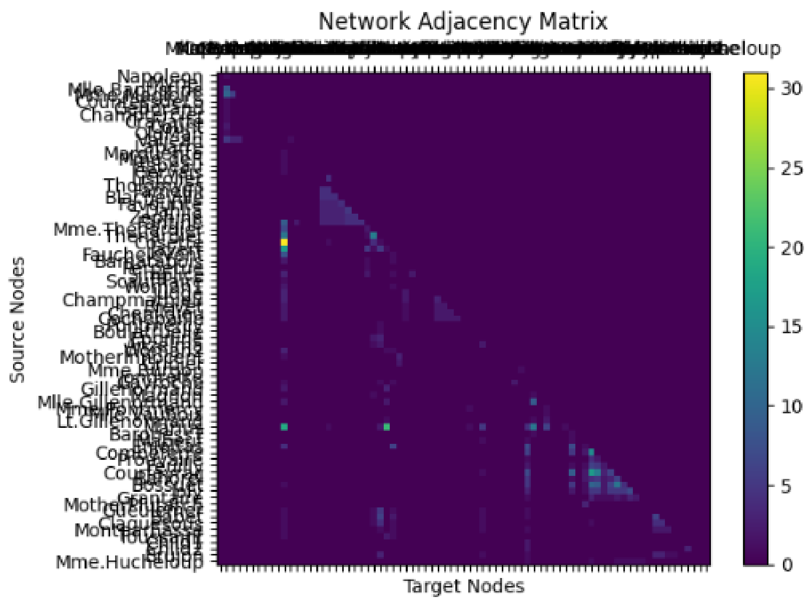
Source Code:

For visualization, I used Python libraries such as Matplotlib and Pandas. Here's a sample implementation:

```
nodes = pd.unique(df[['source', 'target']].values.ravel())
node_index = {node: idx for idx, node in enumerate(nodes)}
n = len(nodes)
adjacency_matrix = np.zeros((n, n))

for _, row in df.iterrows():
    source_idx = node_index[row['source']]
    target_idx = node_index[row['target']]
    adjacency_matrix[source_idx, target_idx] = row['value']

# Step 3: Visualizing the adjacency matrix
fig, ax = plt.subplots()
cax = ax.matshow(adjacency_matrix, cmap='viridis')
fig.colorbar(cax)
ax.set_xticks(np.arange(n))
ax.set_yticks(np.arange(n))
ax.set_xticklabels(nodes)
ax.set_yticklabels(nodes)
ax.set_xlabel('Target Nodes')
ax.set_ylabel('Source Nodes')
plt.title('Network Adjacency Matrix')
plt.show()
```

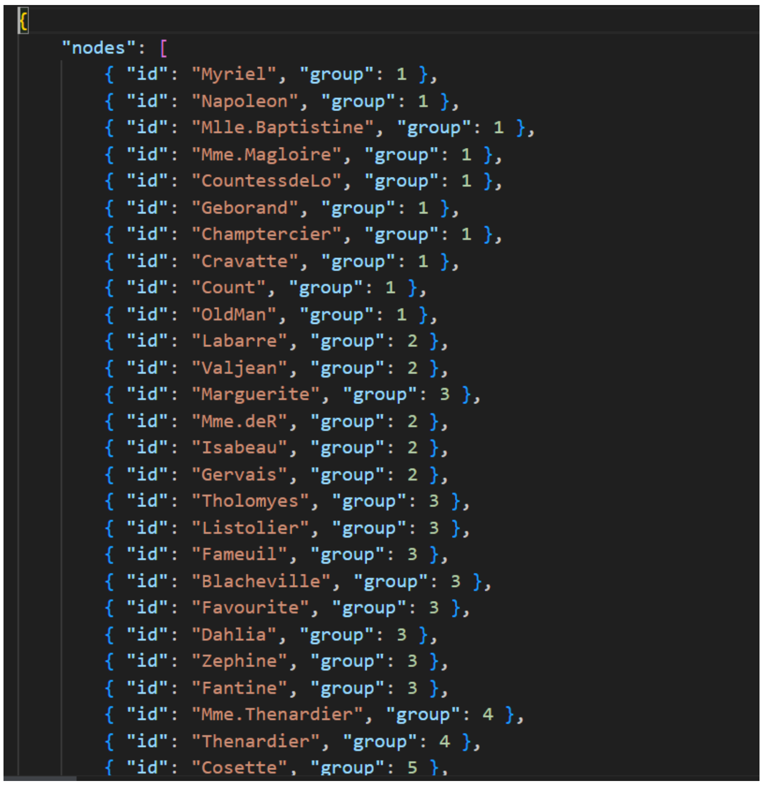
Visualization:

I used matshow from Matplotlib to create the matrix view. Each cell's color depth indicates the edge weight (value), providing a quick visual summary of the connections and their strengths.



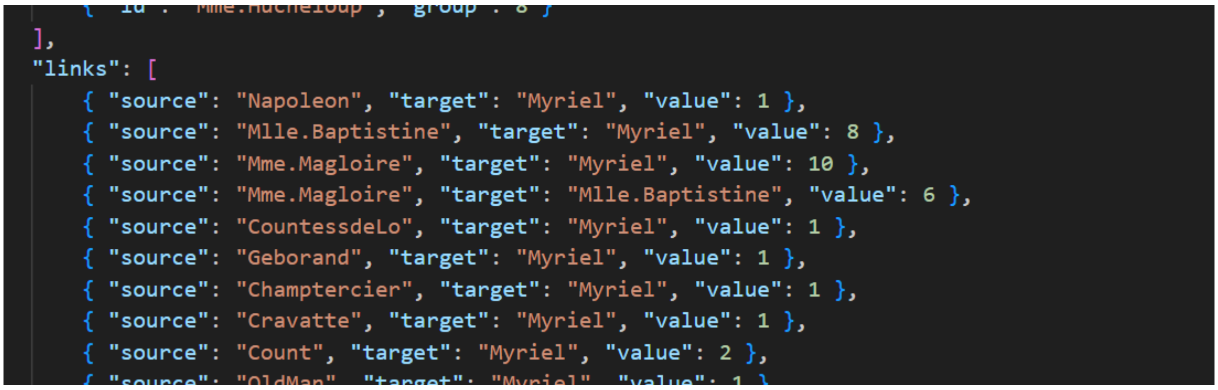
Task2 using Force directed graph

For this task the dataset was in a form of a Jason file :



```
{
    "nodes": [
        { "id": "Myriel", "group": 1 },
        { "id": "Napoleon", "group": 1 },
        { "id": "Mlle.Baptistine", "group": 1 },
        { "id": "Mme.Magloire", "group": 1 },
        { "id": "CountessdeLo", "group": 1 },
        { "id": "Geborand", "group": 1 },
        { "id": "Champtercier", "group": 1 },
        { "id": "Cravatte", "group": 1 },
        { "id": "Count", "group": 1 },
        { "id": "OldMan", "group": 1 },
        { "id": "Labarre", "group": 2 },
        { "id": "Valjean", "group": 2 },
        { "id": "Marguerite", "group": 3 },
        { "id": "Mme.deR", "group": 2 },
        { "id": "Isabeau", "group": 2 },
        { "id": "Gervais", "group": 2 },
        { "id": "Tholomyes", "group": 3 },
        { "id": "Listolier", "group": 3 },
        { "id": "Fameuil", "group": 3 },
        { "id": "Blacheville", "group": 3 },
        { "id": "Favourite", "group": 3 },
        { "id": "Dahlia", "group": 3 },
        { "id": "Zephine", "group": 3 },
        { "id": "Fantine", "group": 3 },
        { "id": "Mme.Thenardier", "group": 4 },
        { "id": "Thenardier", "group": 4 },
        { "id": "Cosette", "group": 5 },
```

And the characters are listed and linked with one another in this part of the file:



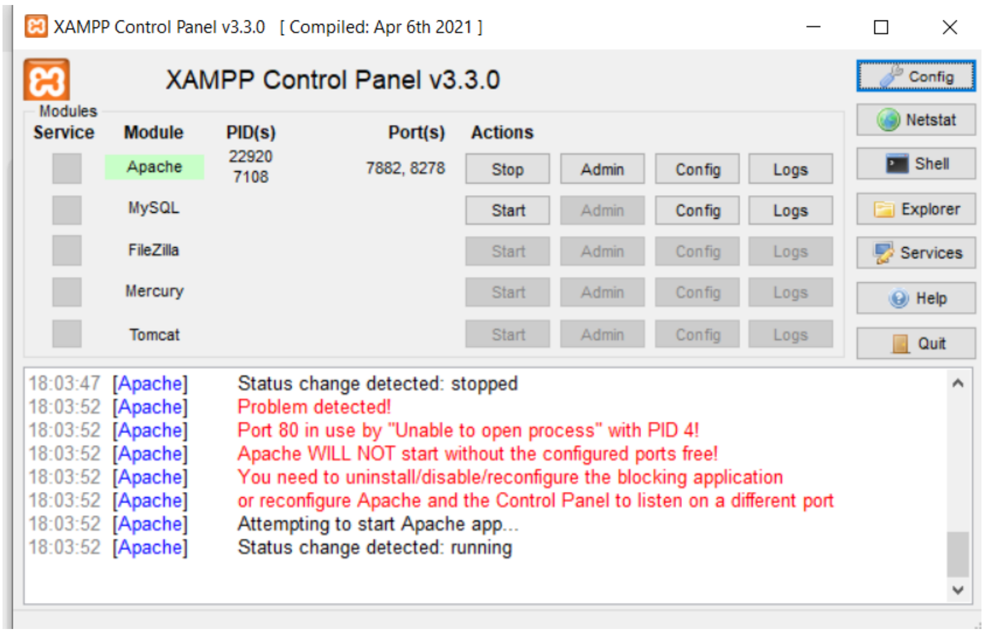
```
        { "id": "Mme.Hucheloup", "group": 8 }
    ],
    "links": [
        { "source": "Napoleon", "target": "Myriel", "value": 1 },
        { "source": "Mlle.Baptistine", "target": "Myriel", "value": 8 },
        { "source": "Mme.Magloire", "target": "Myriel", "value": 10 },
        { "source": "Mme.Magloire", "target": "Mlle.Baptistine", "value": 6 },
        { "source": "CountessdeLo", "target": "Myriel", "value": 1 },
        { "source": "Geborand", "target": "Myriel", "value": 1 },
        { "source": "Champtercier", "target": "Myriel", "value": 1 },
        { "source": "Cravatte", "target": "Myriel", "value": 1 },
        { "source": "Count", "target": "Myriel", "value": 2 },
        { "source": "OldMan", "target": "Myriel", "value": 1 }
```

- *What*: The network being visualized is "Miserables.json", with nodes interconnected by links.

- *How*: Nodes are represented as point marks, while links are depicted using connection marks.

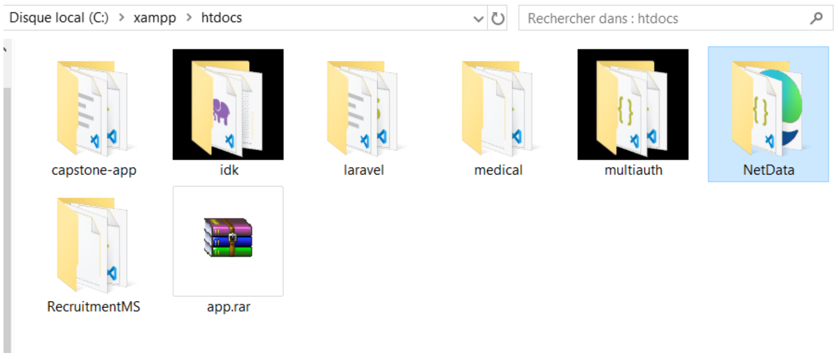
- *Why*: This visualization allows us to discern nodes with more connections, distinguishing between strong and weak nodes.

- *Code*: XAMPP was utilized for visualizing this aspect.

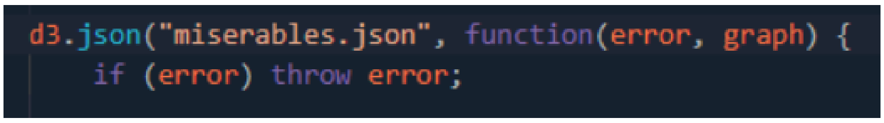

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit
Modules
Stop
Start
Admin
Config
Logs

Service	Module	PID(s)	Port(s)	Actions
	Apache	22920 7108	7882, 8278	Stop Admin Config Logs
	MySQL			Start Admin Config Logs
	FileZilla			Start Admin Config Logs
	Mercury			Start Admin Config Logs
	Tomcat			Start Admin Config Logs

```
18:03:47  [Apache]   Status change detected: stopped
18:03:52  [Apache]   Problem detected!
18:03:52  [Apache]   Port 80 in use by "Unable to open process" with PID 4!
18:03:52  [Apache]   Apache WILL NOT start without the configured ports free!
18:03:52  [Apache]   You need to uninstall/disable/reconfigure the blocking application
18:03:52  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
18:03:52  [Apache]   Attempting to start Apache app...
18:03:52  [Apache]   Status change detected: running
```



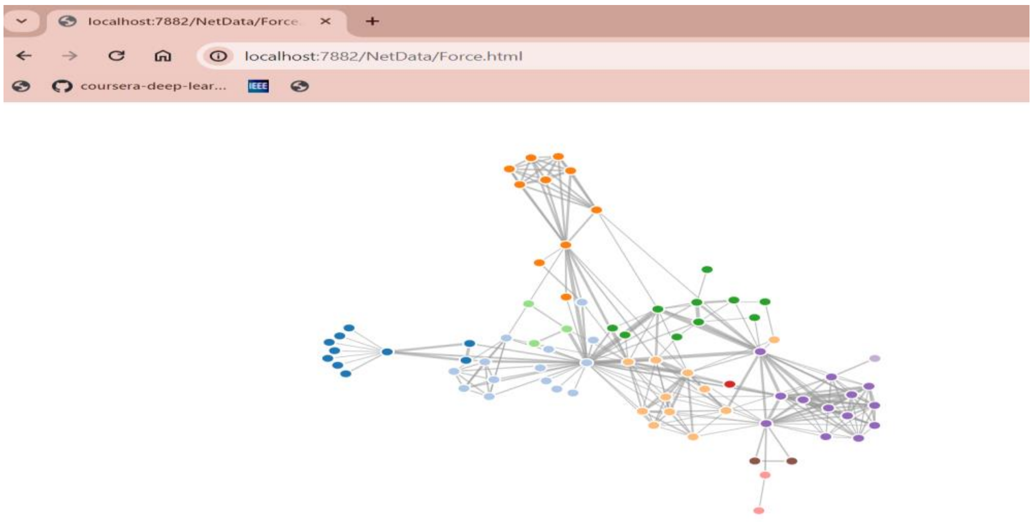
make sure to put our folder containing the source code in C:\\xampp\\htdocs, long side the json file. And load it in the code:



```
d3.json("miserables.json", function(error, graph) {
    if (error) throw error;
```

Visulization :

A Force-directed graph, also known as force-directed graph layout, is a method for drawing graphs where the positioning of nodes is determined by the lengths and directions of edges, indicating potential connections between them. Nodes are positioned closer together when there is a strong correlation between them, and farther apart when the correlation is weak


localhost:7882/NetData/Force.
localhost:7882/NetData/Force.html
coursera-deep-lear...
IEEE

: