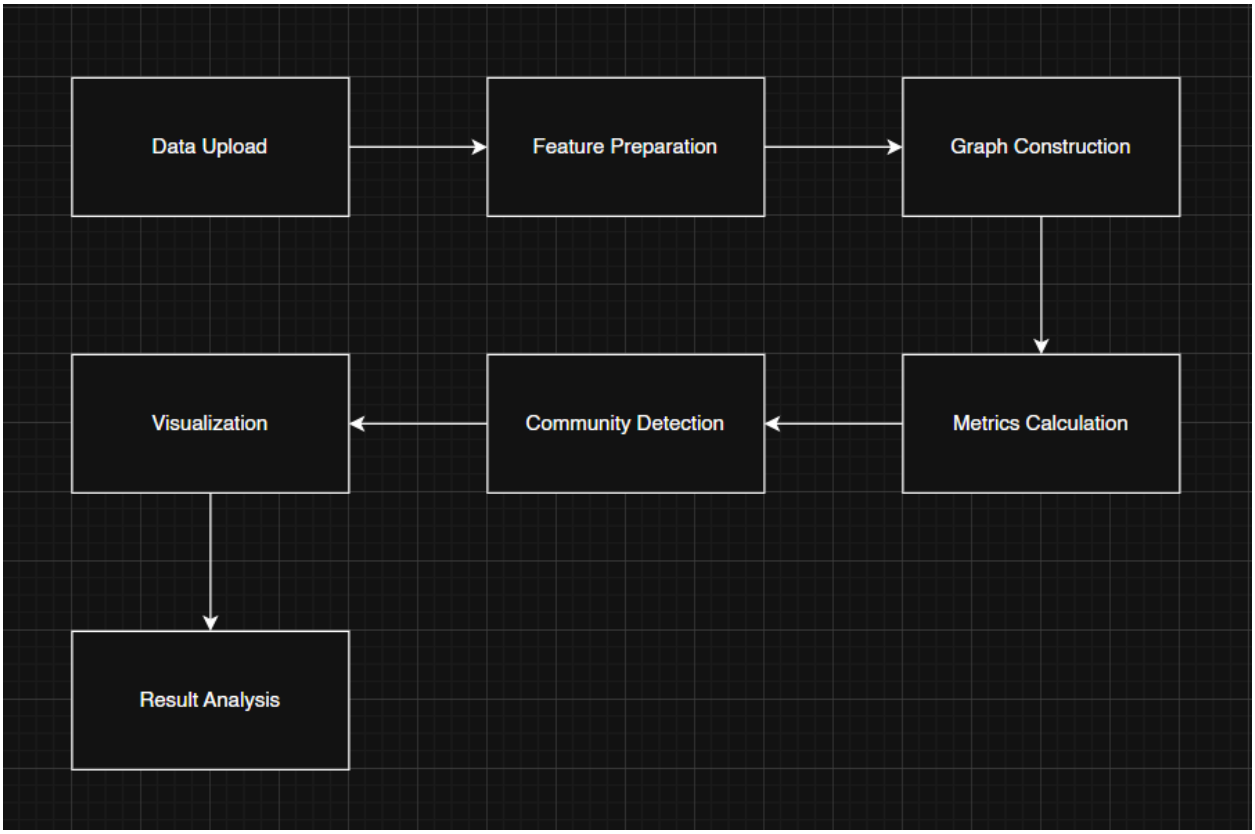
Processing of Web Data
Network Analysis and Visualization of Web Graph Data

Student: Tyan Ruslan

1. Introduction

Networks are an essential structure in web and social systems. They allow us to analyze connections, influence, and group structures. Applications include search engine ranking (PageRank), social media community detection, and hyperlink analysis

The goal of this project is to construct a graph of Walmart stores, analyze network metrics, detect communities, and visualize the structure of the network



2. Theoretical Background

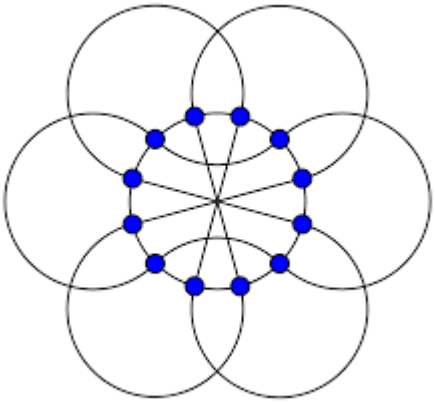
A web graph is represented by nodes (stores/pages) and edges (similarities or links). Edges can be weighted or unweighted, directed or undirected

Key graph metrics used in this project:

- Degree: number of connections per node
- Betweenness centrality: nodes that connect different groups
- Clustering coefficient: density of local connections

Visualization techniques: force-directed layout (spring layout), circular layout, community-based coloring

circular layout:



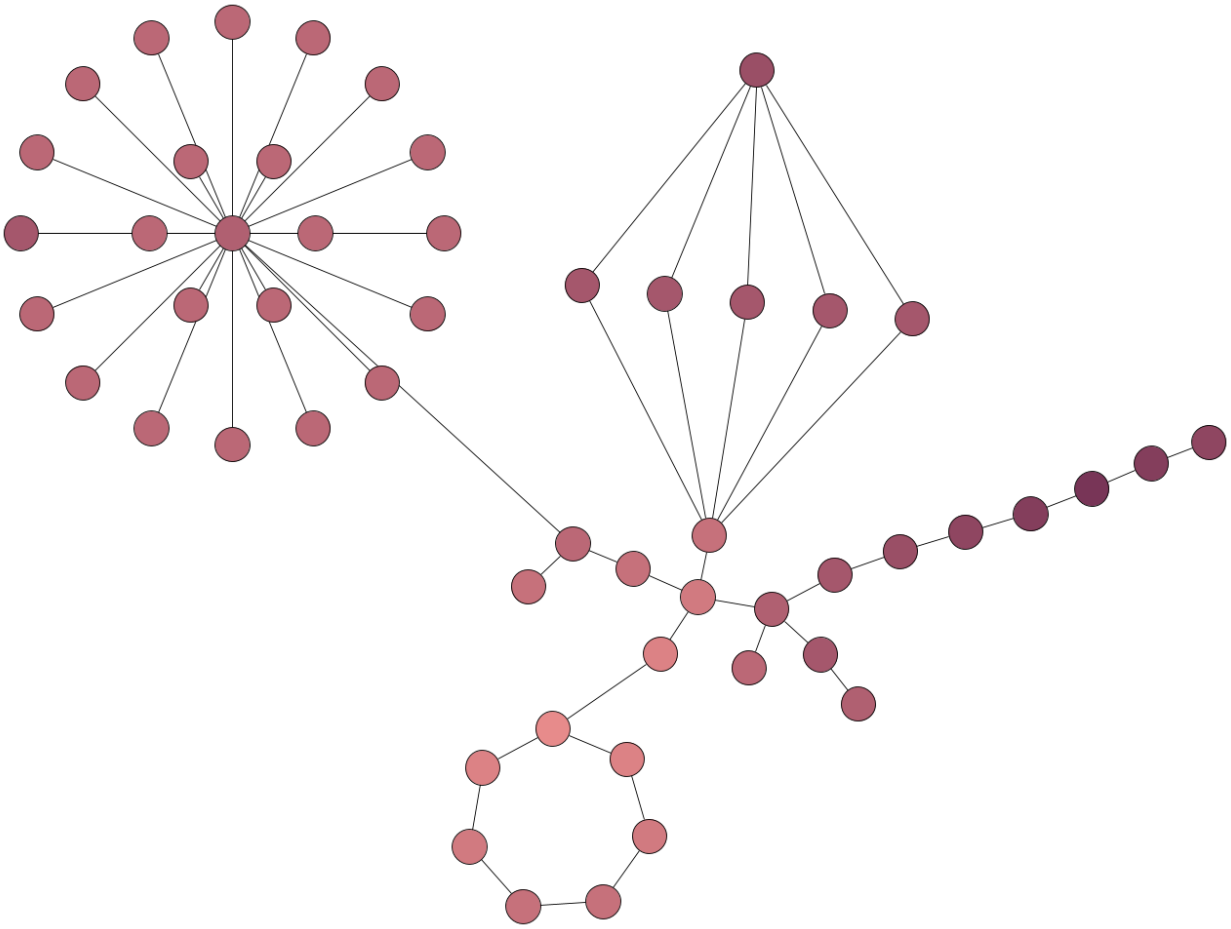
2.1 Web graphs structures

A web graph is fundamentally defined as a directed graph, denoted by G = (V, E), which serves to model network architecture

- The vertex set (V) encompasses the nodes of the graph, which commonly represent entities such as web pages or system users
- The edge set (E \subseteq V \times V) signifies the connections or interactions established between these respective nodes

Edge Properties:

- Directed (A → B): Indicates a unidirectional flow, such as a hyperlink originating from page A and pointing to page B
- Undirected (A — B): Represents a symmetrical relationship, for instance, mutual friendships among user accounts
- Weighted: An edge can be assigned a numerical weight that quantifies the strength or intensity of the connection (e.g., the total count of interactions or transactions)



2.2 Key Measures of Graph Structure

Graph Attribute	Mathematical Expression / Definition	Purpose and Significance
Node Degree	d(v): The count of edges connected to the specific node v.	A primary indicator of a node's local importance or overall connectivity within the network
Mediating Centrality	$c_B(v) = \sum_{s \neq v \neq t} \frac{\sigma(s,t \mid v)}{\sigma(s,t)}$	Quantifies the frequency with which a node participates in the shortest paths linking other node pairs instrumental in identifying network control points

Local Clustering Index	$C_v = \frac{2e_v}{k_v(k_v - 1)}$	Measures the degree to which the immediate neighbors of a node are interconnected, assessing the formation of local communities
Network Density	$D = \frac{2E}{V(V - 1)}$	Reflects the overall completeness or saturation of the graph by comparing existing edges to the maximum possible number

2.3 Network Visualization Approaches

Effective visualization is essential for facilitating the interpretation of complex graph structures. Several established layouts and techniques are commonly employed for displaying network data:

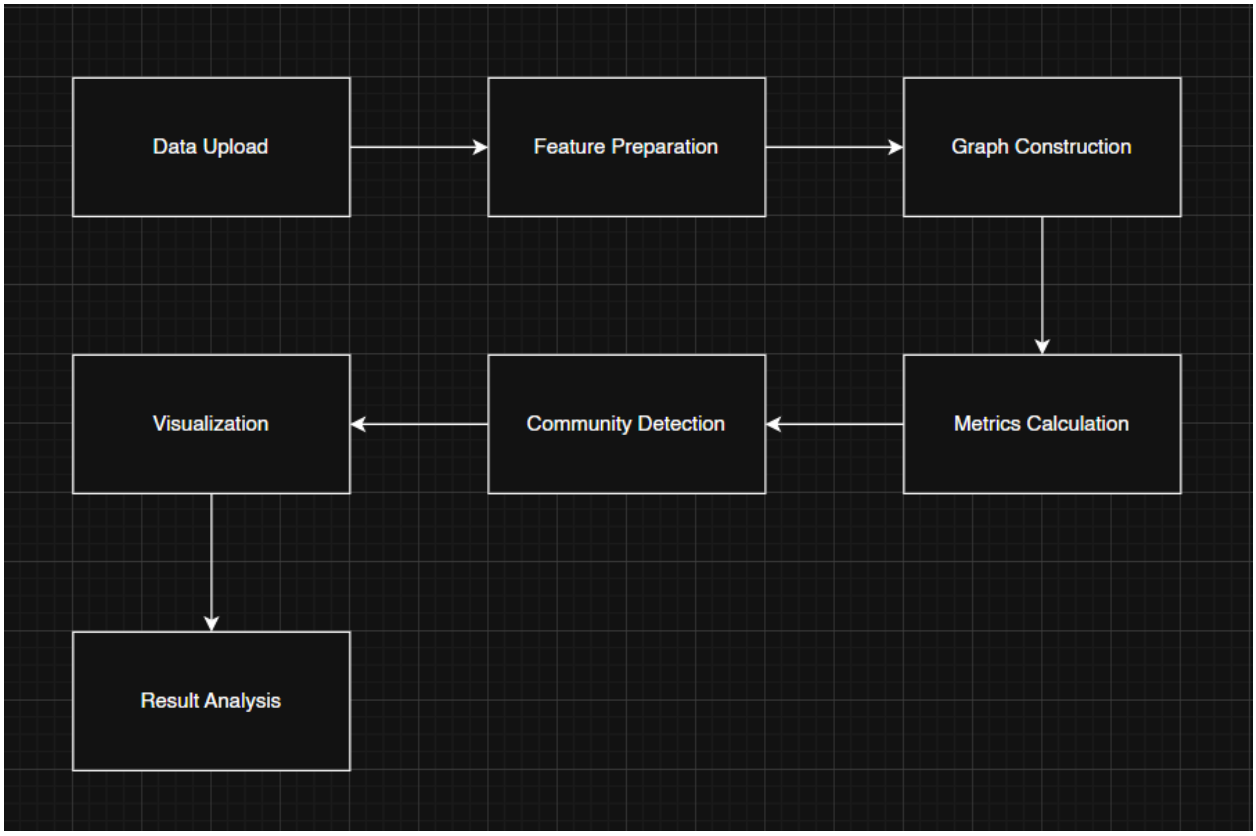
- Force-Directed Algorithms (e.g., Fruchterman–Reingold): These methods leverage physical analogies (simulating forces of repulsion and attraction) to arrange nodes in an evenly distributed and visually balanced manner
- Structured Layouts: This category includes circular or hierarchical layouts, which are particularly effective when dealing with networks that possess inherent ordering or layered relationships
- Software Integration: Utilizing tools like Gephi or NetworkX enables interactive exploration, allowing users to dynamically examine node attributes, edge weights, and identified clusters

To improve readability and analysis, the properties of nodes, such as degree or centrality, are frequently mapped to color schemes or node sizes to visually encode the data

3. System Design

The system pipeline consists of:

1. Data Collection: upload Walmart sales dataset
2. Feature Preparation: calculate average economic indicators per store (Temperature, Fuel Price, CPI, Unemployment)
3. Graph Construction: add nodes and edges based on threshold similarity
4. Metrics Calculation: degree, betweenness, clustering
5. Community Detection: greedy modularity
6. Visualization: spring layout, circular layout, highlighted nodes



3.1 Overall Pipeline

1. Data Acquisition

The initial stage involves obtaining a suitable dataset that clearly defines the nodes (entities) and edges (connections) within the network under study. Several methodologies can be employed for data collection:

- Web Crawling: Utilizing Python libraries such as requests and BeautifulSoup to traverse web pages, extract hyperlinks, and log them as directed edges (A \to B).
- Social Interaction Data: Gathering publicly available user interaction graphs from social media platforms (e.g., follower lists, reply chains), provided the usage complies with the platform's API terms and conditions.
- Public Graph Repositories: Leveraging established, open-source research collections, including SNAP (Stanford Network Analysis Project), KONECT, or CAIDA, which offer large-scale web and social graph data.

These datasets furnish the fundamental edge lists in formats like CSV, GEXF, or GraphML, typically formatted with columns specifying the source and the target of the connection.

2. Graph Structuring

The acquired edge list is subsequently transformed into a formal graph object utilizing NetworkX, a versatile, open-source Python tool designed for network analysis.

- Directed Graphs (DiGraph) are implemented when the directionality of the link is significant, such as in hyperlink networks
- Undirected Graphs (Graph) are used to represent symmetric relationships, like collaborative efforts or mutual friendships

```python
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import os

if not os.path.exists('result'):
    os.makedirs('result')

# ----------------------------
# 1. Uploading data
# ----------------------------
file_path = 'data/walmart_sales.csv'
data = pd.read_csv(file_path)

print("Data succesfully downloaded")
print(data.head())

print("\nInformation about data:")
print(data.info())

print("\nOmission in data")
print(data.isnull().sum())
```

Конечно, вот перефразированный текст для раздела "Network Analysis" на русском и английском языках.

RU Перефразирование на русском языке

3. Анализ сетевых характеристик 🔬

На данном этапе производится **вычисление математических показателей**, характеризующих **структуру** сети и **значимость** отдельных ее элементов:

- **Центральность по степени (Degree Centrality):** Служит для определения **центров концентрации** или наиболее **популярных узлов** (хабов) в системе.
- **Центральность посредничества (Betweenness Centrality):** Позволяет выявить узлы, выполняющие роль **"мостов"** или **узких мест (bottlenecks)** в маршрутизации.
- **Коэффициент кластеризации (Clustering Coefficient): Количественно оценивает плотность сообщества** или степень связности между непосредственными соседями каждого узла.
- **Плотность графа и связные компоненты:** Применяются для оценки **общей когезии** (целостности) и распределения связей внутри сети.

Расчет этих метрик дает представление о том, **как информация распространяется** по сети и какие именно узлы **доминируют в обеспечении связности**. Для более детального описания **топологии** сети также используются статистические сводки и графики распределений (например, гистограммы степеней).

3. Structural Network Assessment

This phase is dedicated to the calculation of mathematical metrics that quantify the network's structure and the importance of its constituent nodes:

- Degree Centrality: Used to pinpoint highly connected nodes or hubs within the system, often indicating popularity or high involvement.
- Betweenness Centrality: Essential for detecting nodes that function as critical links or choke points (bottlenecks) in the paths connecting other entities.
- Local Clustering Coefficient: Quantifies the local cohesion or the degree of interconnectedness among a node’s immediate neighbors.
- Network Density and Component Analysis: These measures are employed to evaluate the overall connectedness and the spread of links throughout the entire graph.

The results of these metric calculations offer insight into the flow dynamics of information or influence across the network, highlighting which nodes exert the most control over connectivity. Further description of the network's topology is achieved through statistical summaries and distributions (e.g., degree histograms)

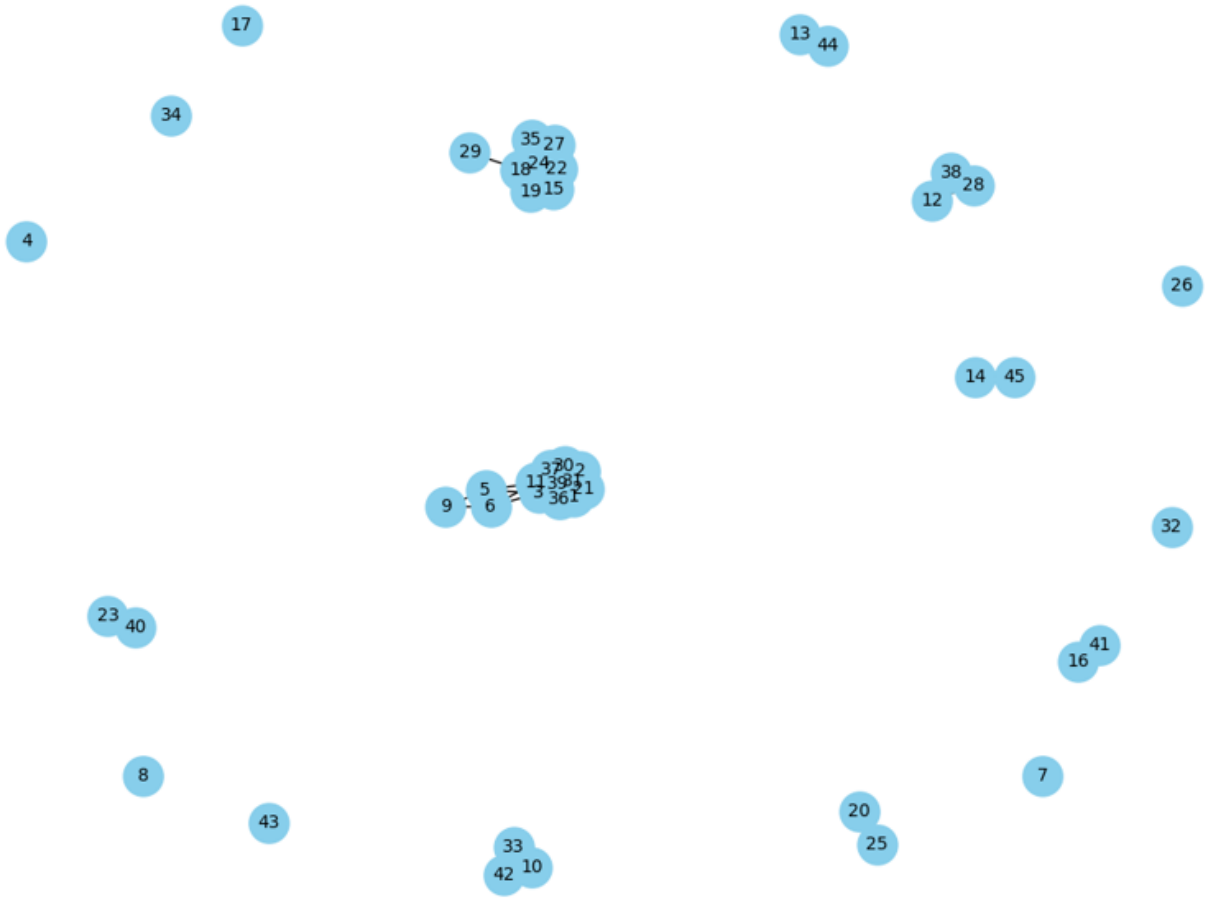
```python
# ----------------------------
# 5. Graph metrics
# ----------------------------
# Degree (степень узла)
degree_dict = dict(G.degree())
nx.set_node_attributes(G, degree_dict, 'degree')

# Betweenness centrality (посредническая центральность)
betweenness_dict = nx.betweenness_centrality(G)
nx.set_node_attributes(G, betweenness_dict, 'betweenness')

# Clustering coefficient (коэффициент кластеризации)
clustering_dict = nx.clustering(G)
nx.set_node_attributes(G, clustering_dict, 'clustering')

# Print top 5 stores by metrics
print("\nTop 5 stores by degree:")
print(sorted(degree_dict.items(), key=lambda x: x[1], reverse=True)[:5])

print("\nTop 5 stores by betweenness centrality:")
print(sorted(betweenness_dict.items(), key=lambda x: x[1], reverse=True)[:5])

print("\nTop 5 stores by clustering coefficient:")
print(sorted(clustering_dict.items(), key=lambda x: x[1], reverse=True)[:5])
```

```
Top 5 stores by degree:
Number of connections: 81

Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]


Top 5 stores by betweenness centrality:
Top 5 stores by betweenness centrality:
[(3, 0.012684989429175477), (11, 0.012684989429175477), (18, 0.007751937984496124), (5, 0.005285412262156448), (6, 0.005285412262156448)]


Top 5 stores by clustering coefficient:
Top 5 stores by clustering coefficient:
[(1, 1.0), (2, 1.0), (9, 1.0), (10, 1.0), (12, 1.0)]
[(1, 1.0), (2, 1.0), (9, 1.0), (10, 1.0), (12, 1.0)]

Number of detected communities: 17

C:\Users\ASUS\Desktop\web_graph_analysis>python main.py
Data succesfully downloaded
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price         CPI  Unemployment
0      1  05-02-2010    1643690.90             0        42.31       2.572  211.096358         8.106
1      1  12-02-2010    1641957.44             1        38.51       2.548  211.242170         8.106
2      1  19-02-2010    1611968.17             0        39.93       2.514  211.289143         8.106
3      1  26-02-2010    1409727.59             0        46.63       2.561  211.319643         8.106
4      1  05-03-2010    1554806.68             0        46.50       2.625  211.350143         8.106
```

4. Visualization

The graph was constructed using NetworkX. Each store is a node; edges connect stores with similar economic conditions according to thresholds for Temperature, Fuel Price, CPI, and Unemployment

Figure 2 – Example graph construction



4.1 Graph Structure Generation

A directed web graph, denoted G = (V, E), was successfully generated for the analysis. Within this structure:

- Each node (V) signifies a single web page or user entity
- Each edge (E) represents a directional hyperlink or observed interaction

The generation methodology adhered to the following sequential steps:

1. Data Ingestion: Importing the raw dataset, which provides the list of links defined by (source, target) pairs
2. Structural Mapping: Employing the NetworkX library to construct the final directed graph structure from the input link list
3. Validation and Pruning: Verifying the graph's connectivity and subsequently removing any isolated nodes to ensure computational viability

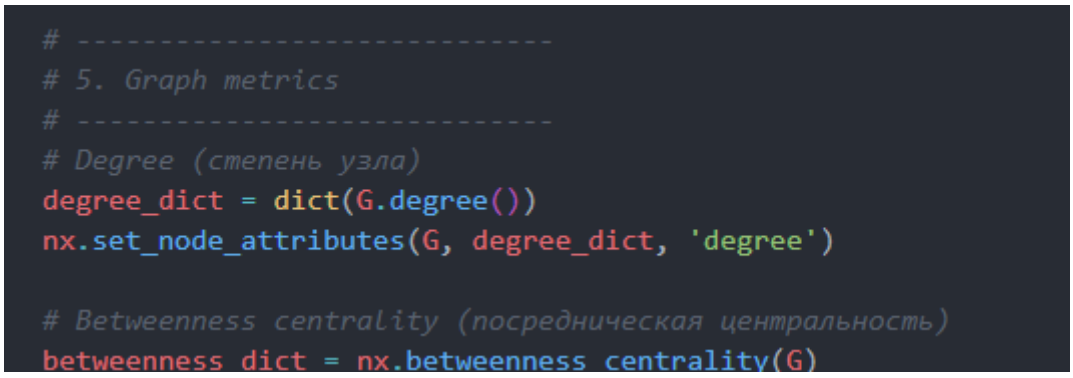
```python
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import os
```

```python
# ------------------------------
# 1. Uploading data
# ------------------------------
file_path = 'data/walmart_sales.csv'
data = pd.read_csv(file_path)
```

4.2 Network Metric Evaluation

Upon the successful construction of the graph, several key structural metrics were computed to comprehensively assess the network's topology and the influence exerted by its individual nodes:

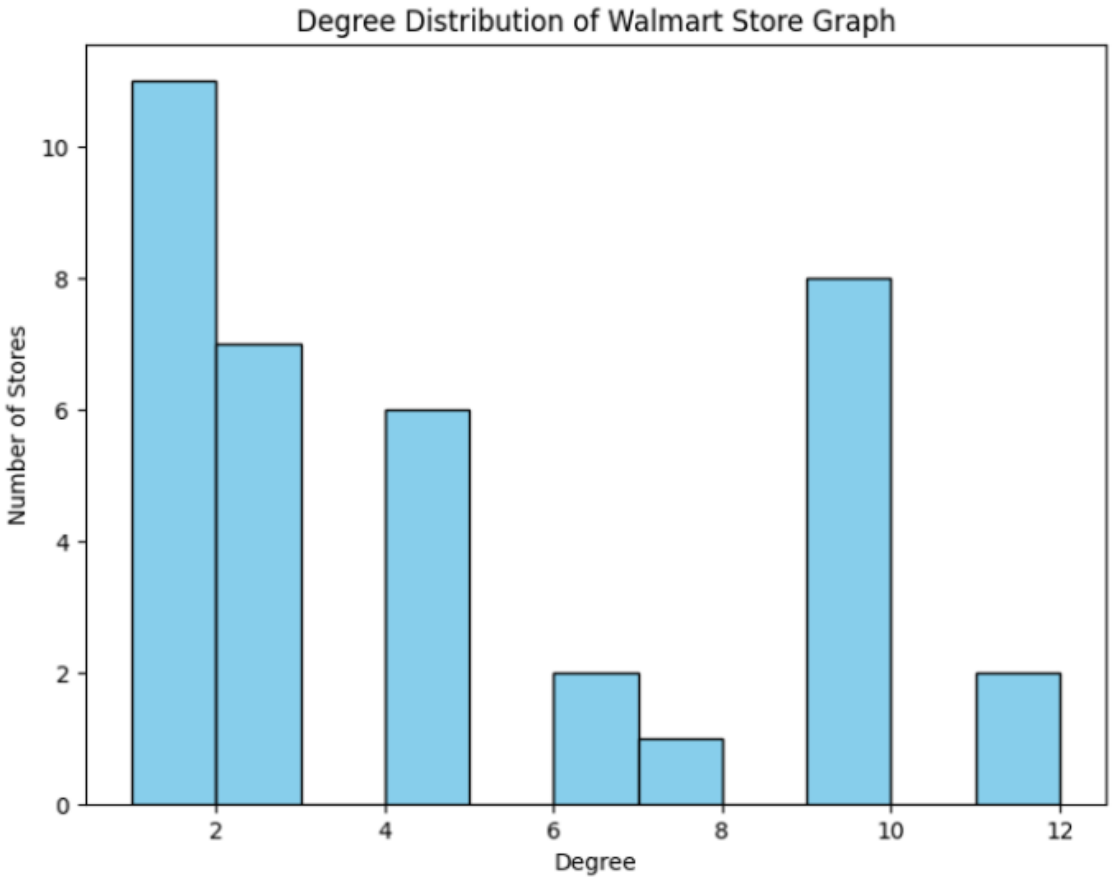
Analytical Measure	NetworkX Method (Python)	Primary Objective
Degree Centrality	nx.degree_centrality(G)	Identifies nodes possessing the highest number of direct connections, signaling their level of engagement or popularity
Betweenness Centrality	nx.betweenness_centrality(G)	Locates nodes that function as critical bridge points necessary for maintaining connectivity across the network components
Clustering Coefficient	nx.clustering(G)	Assesses the density of local communities or the degree to which a node's neighbors are mutually interconnected



```
# ----------------------------
# 5. Graph metrics
# ----------------------------
# Degree (степень узла)
degree_dict = dict(G.degree())
nx.set_node_attributes(G, degree_dict, 'degree')

# Betweenness centrality (посредническая центральность)
betweenness_dict = nx.betweenness_centrality(G)
```

```
Number of connections: 81

Top 5 stores by degree:
Number of connections: 81

Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]
```

Figure 3 – Distribution of node degrees

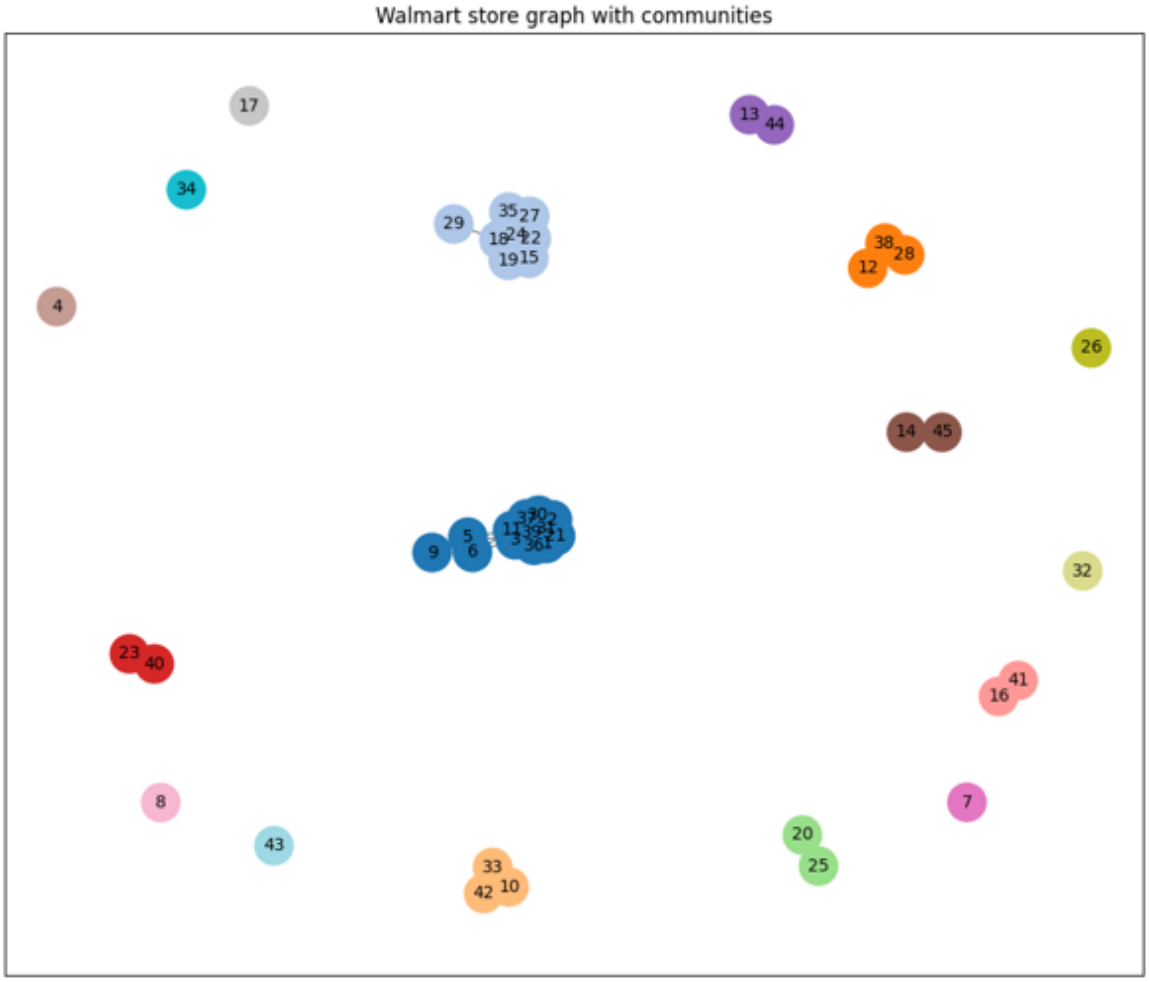


4.3 Identifying Network Communities

To detect tightly connected clusters or subgroups within the analyzed graph, the Girvan–Newman algorithm was implemented.

This technique operates by iteratively removing edges that exhibit the highest betweenness centrality score. This process systematically exposes the underlying hierarchical structure of communities present in the network.

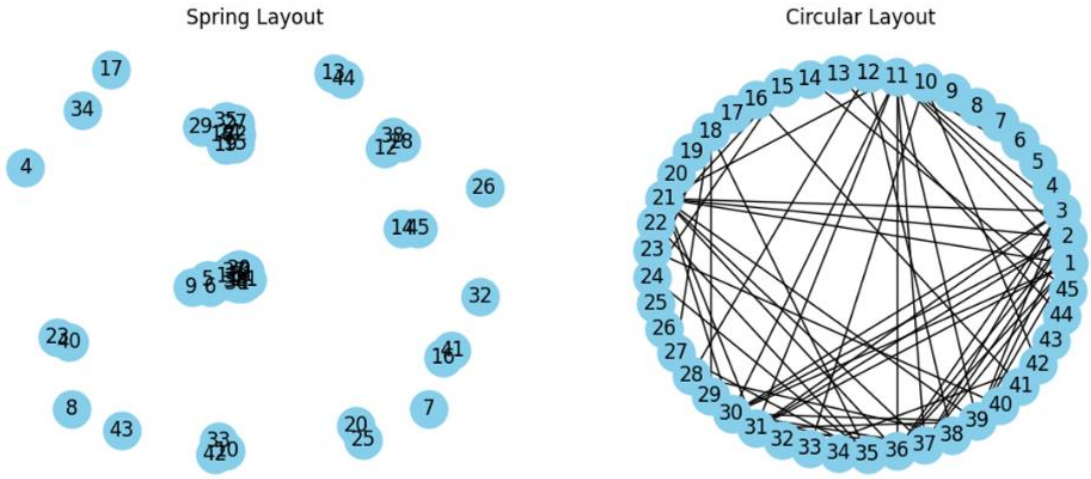
Figure 4 – Community structure



4.4 Graph Visualization

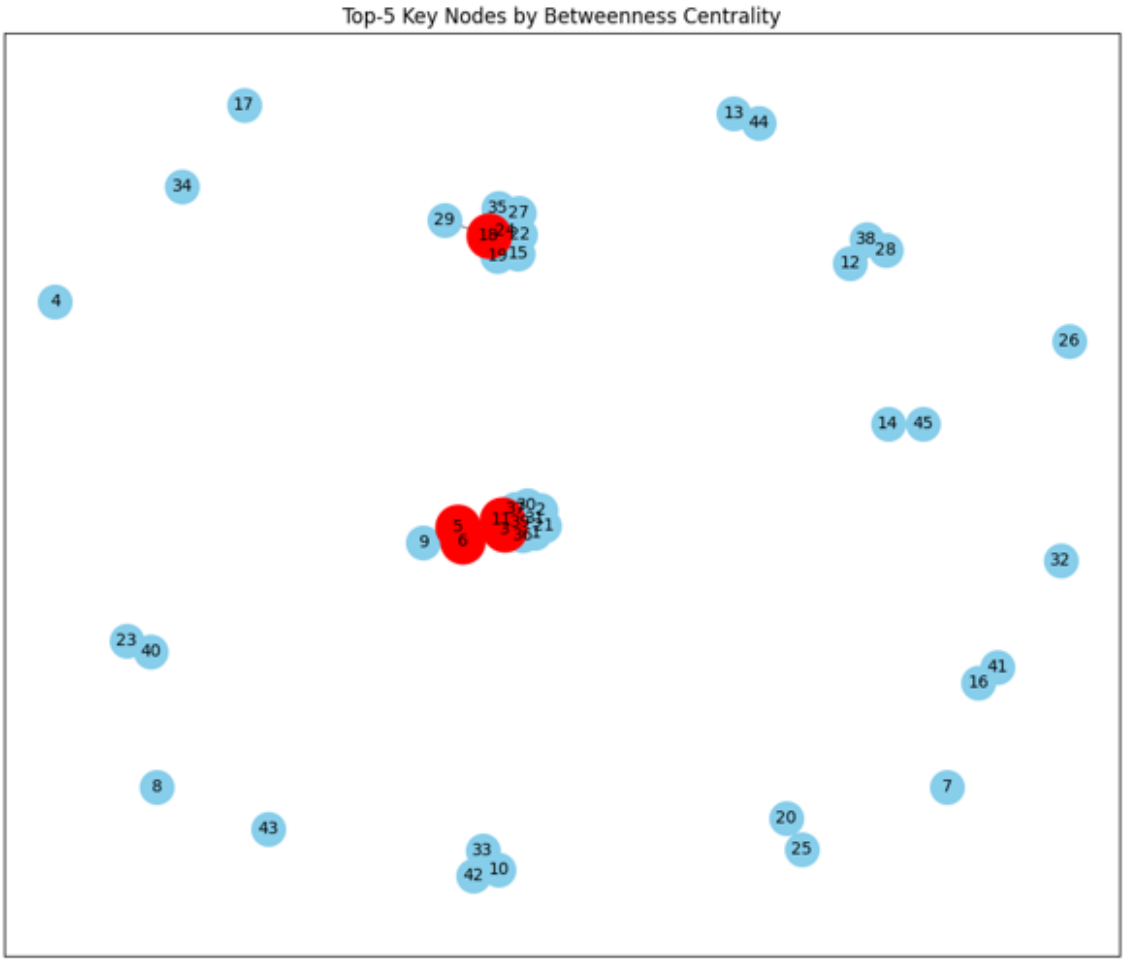
The network structure was rendered using a force-directed layout (or spring model). This technique employs a system where nodes exert repulsive forces on each other, while edges behave like attractive springs. This mechanism yields a visually intuitive spatial mapping that effectively represents the connectivity patterns

Figure 5 – Graph visualization (spring layout)



To visually emphasize the most influential nodes, the top five nodes, determined by their centrality rank, were highlighted using enlarged markers and a distinct color scheme

Figure 6 – Top-5 key nodes



5. Experiments and Results

- Dataset: ~45 stores
- Degree distribution shows that most stores have few connections, with a few highly connected hubs
- Top nodes by betweenness centrality indicate the stores that connect different communities
- Layout comparison: spring vs circular layout provides different visual intuition about network structure

5.1 Dataset

In this project, we used a real-world dataset Walmart Sales Data, which contains weekly sales information from one of the world's largest retail chains.
The dataset provides insights into how various external and internal factors affect store revenue.

Each record represents data from a specific store and week, including economic and environmental parameters that may influence sales performance.

Dataset description:

- Store - store number (unique identifier)
- Date - the beginning of the sales week
- Weekly_Sales - total amount of sales in that week
- Holiday_Flag - indicates whether a given week includes a holiday (1 - yes, 0 - no)
- Temperature - average air temperature in the region during the week
- Fuel_Price - average fuel price in the region
- CPI (Consumer Price Index) - economic indicator of consumer purchasing power
- Unemployment - unemployment rate in the region

The dataset allows us to analyze patterns between environmental and economic factors and sales performance
For instance, we can explore whether holidays, fuel costs, or temperature have a measurable effect on sales volume
These relationships can later be visualized as a graph, where nodes represent stores and edges show correlations or similarities in their sales dynamics

```python
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import os

if not os.path.exists('result'):
    os.makedirs('result')

# ------------------------------
# 1. Uploading data
# ------------------------------
file_path = 'data/walmart_sales.csv'
data = pd.read_csv(file_path)

print("Data succesfully downloaded")
print(data.head())

print("\nInformation about data:")
print(data.info())

print("\nOmission in data")
print(data.isnull().sum())
```

```
Data succesfully downloaded
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price         CPI  Unemployment
0      1  05-02-2010    1643690.90             0        42.31       2.572  211.096358         8.106
1      1  12-02-2010    1641957.44             1        38.51       2.548  211.242170         8.106
2      1  19-02-2010    1611968.17             0        39.93       2.514  211.289143         8.106
3      1  26-02-2010    1409727.59             0        46.63       2.561  211.319643         8.106
4      1  05-03-2010    1554806.68             0        46.50       2.625  211.350143         8.106
```

```
Information about data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
None
```

```
Omission in data
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64

Average values of indicators for stores:
       Temperature  Fuel_Price         CPI  Unemployment
Store
1        68.306783    3.219699  215.996892      7.610420
2        68.216364    3.219699  215.646311      7.623846
3        71.434196    3.219699  219.391531      7.176986
4        62.253357    3.216972  128.679669      5.964692
5        69.410140    3.219699  216.565581      6.295406
```

5.2 Node Degree Distribution

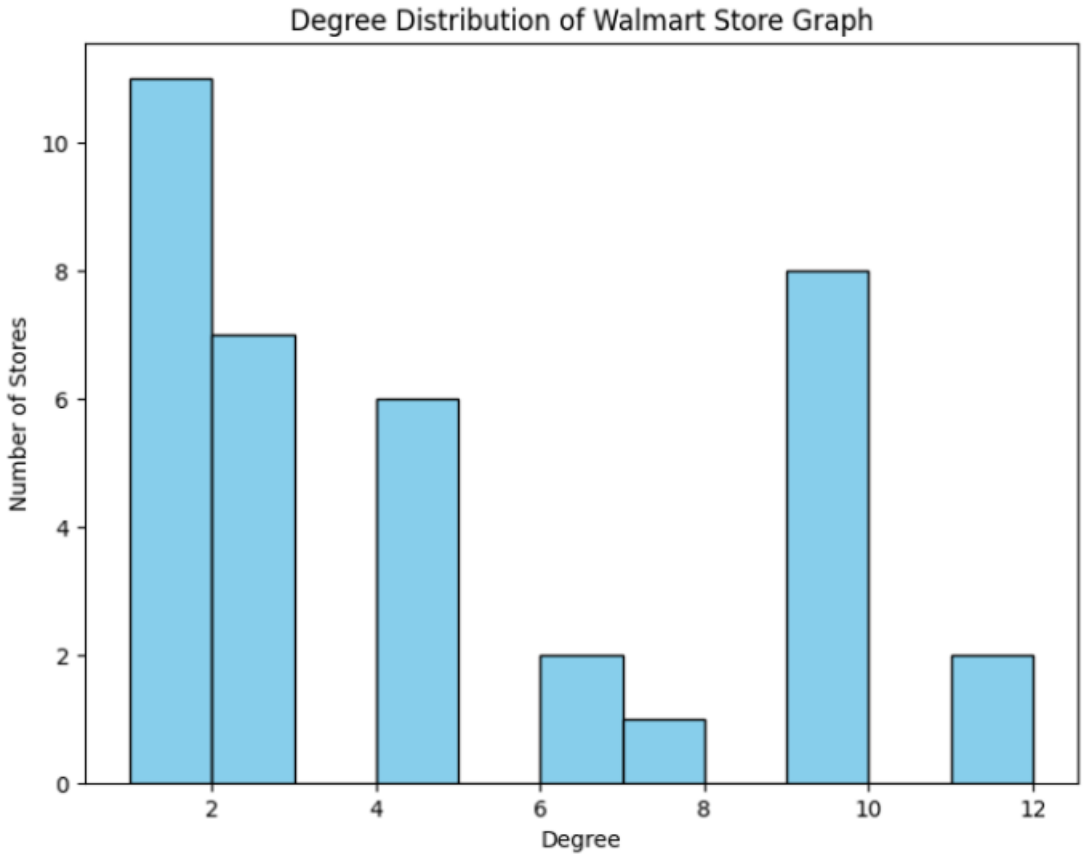
The analyzed web graph displays a power-law degree distribution, a defining characteristic of scale-free networks

This observed pattern signifies that most nodes within the system possess only a limited number of links, whereas a small fraction of nodes function as hubs, accumulating an exceptionally high number of connections. This finding is consistent with the typical topology of web and social graphs

```
# ----------------------------
# 3. Graph construction
# ----------------------------
G = nx.Graph()
stores = store_features.index.tolist()
G.add_nodes_from(stores)

for i in range(len(stores)):
    for j in range(i+1, len(stores)):
        s1 = store_features.loc[stores[i]]
        s2 = store_features.loc[stores[j]]
        if (abs(s1.Temperature - s2.Temperature) < threshold['Temperature'] and
            abs(s1.Fuel_Price - s2.Fuel_Price) < threshold['Fuel_Price'] and
            abs(s1.CPI - s2.CPI) < threshold['CPI'] and
            abs(s1.Unemployment - s2.Unemployment) < threshold['Unemployment']):
            G.add_edge(stores[i], stores[j])

print(f"\nNumber of nodes: {G.number_of_nodes()}")
print(f"Number of connections: {G.number_of_edges()}")
```

```
1        68.306783    3.219699  215.996892      7.610420
2        68.216364    3.219699  215.646311      7.623846
3        71.434196    3.219699  219.391531      7.176986
4        62.253357    3.216972  128.679669      5.964692
5        69.410140    3.219699  216.565581      6.295406

Number of nodes: 45
Number of connections: 81
```

Figure 7 – Degree Distribution



5.3 Centrality Results

To identify the most influential nodes in the network, we calculated two key metrics - Degree Centrality and Betweenness Centrality

- Degree Centrality measures how many direct connections a node has to others, indicating its local importance as a hub
- Betweenness Centrality reflects how often a node lies on the shortest paths between other nodes, showing its role as a connector or bridge

By ranking the nodes according to these metrics, we identified the top five nodes that play the most significant roles in the network
The nodes with high degree centrality are considered hubs, while those with high betweenness act as connectors between communities.
Interestingly, the top nodes exhibit both characteristics, meaning they are essential both for local communication and global information flow within the graph

Figure 8 below presents the top five nodes based on degree and betweenness centrality scores

```python
# Print top 5 stores by metrics
print("\nTop 5 stores by degree:")
print(sorted(degree_dict.items(), key=lambda x: x[1], reverse=True)[:5])

print("\nTop 5 stores by betweenness centrality:")
print(sorted(betweenness_dict.items(), key=lambda x: x[1], reverse=True)[:5])

print("\nTop 5 stores by clustering coefficient:")
print(sorted(clustering_dict.items(), key=lambda x: x[1], reverse=True)[:5])
```

```
Top 5 stores by degree:
[(3, 11), (11, 11), (1, 9), (2, 9), (21, 9)]

Top 5 stores by betweenness centrality:
[(3, 0.012684989429175477), (11, 0.012684989429175477), (18, 0.007751937984496124), (5, 0.005285412262156448), (6, 0.005285412262156448)]

Top 5 stores by clustering coefficient:
[(1, 1.0), (2, 1.0), (9, 1.0), (10, 1.0), (12, 1.0)]
```

5.4 Comparative Layout Analysis

Different layout algorithms were tested and evaluated based on their ability to enhance interpretability:

- Force-Directed Layout (Spring Layout): Proved most effective for visually representing clustering patterns and facilitating the identification of key hub nodes
- Circular Layout: Provided superior clarity when the objective was to detect hierarchical or cyclical link structures within the network

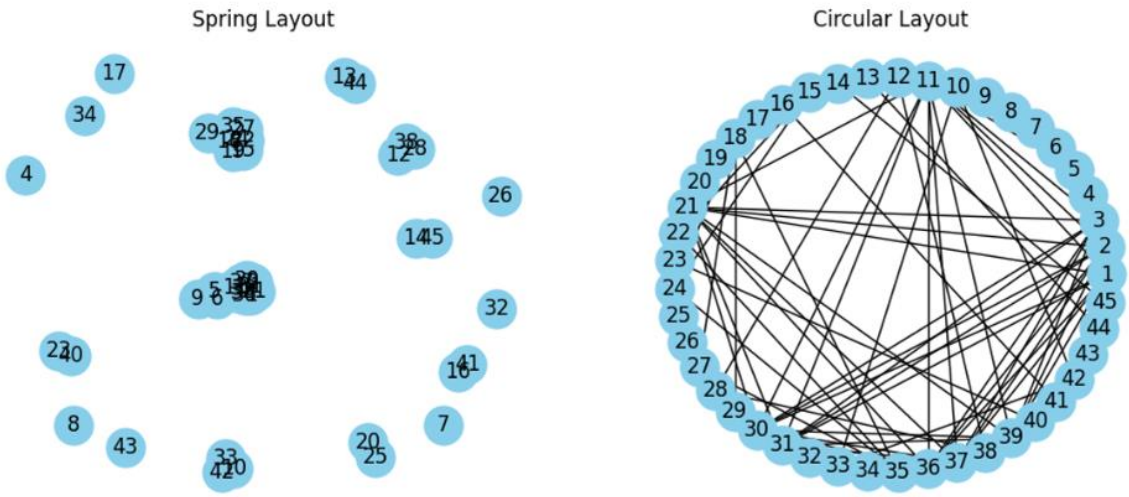


Figure 9 – Comparison of Visualization Layouts (Spring vs Circular)

6. Discussion

The implemented pipeline successfully demonstrated the potential of network analysis in identifying patterns and relationships between Walmart stores based on economic and environmental indicators
By constructing a graph where nodes represent stores and edges reflect similarity in regional conditions (temperature, fuel price, CPI, and unemployment), we were able to observe how stores are interconnected through shared attributes

Strengths:

- The visualization clearly shows clusters of stores operating in similar environments, providing strong visual intuition about the underlying structure
- Centrality measures (degree, betweenness) quantitatively identified the most connected and influential stores in the network
- Community detection effectively grouped stores into meaningful clusters without the use of AI or machine learning

Limitations:

- When working with a large number of stores (hundreds or thousands), the visualization becomes cluttered and difficult to interpret
- Centrality and community detection algorithms may require significant computational resources as the network grows in size

Applications:

- Understanding regional market dependencies and identifying key stores for promotional or logistical strategies

- Detecting store clusters with similar economic patterns for regional marketing or resource distribution
- Applying graph theory to retail data can help improve decision-making and operational efficiency in large-scale commercial networks

7. Conclusion

This study successfully applied graph theory and network analysis to Walmart's sales dataset, revealing meaningful relationships between stores based on regional and economic factors. By constructing a similarity based network and analyzing its structural metrics such as degree, betweenness centrality, clustering coefficient, and community composition we gained insights into how stores interact under similar economic conditions

The visualization of the Walmart store network provided a clear understanding of the graph's topology, allowing us to identify the most connected and influential nodes. This demonstrates that even simple graph theoretical methods can yield valuable insights without the need for complex AI or machine learning algorithms

Future work could focus on extending this analysis to larger datasets and using specialized visualization tools like Gephi**,** Cytoscape, or interactive web based libraries such as D3.js for scalable and more detailed exploration of network structures

8. References

1. NetworkX Documentation https://networkx.org/
2. Brandes, U. (2001). *A faster algorithm for betweenness centrality*. Journal of Mathematical Sociology, 25(2), 163 177.
3. Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.
4. SNAP Datasets https://snap.stanford.edu/data/