Documents are here:

GRAPH THEORY

2 - Graph Traversal Algorithms

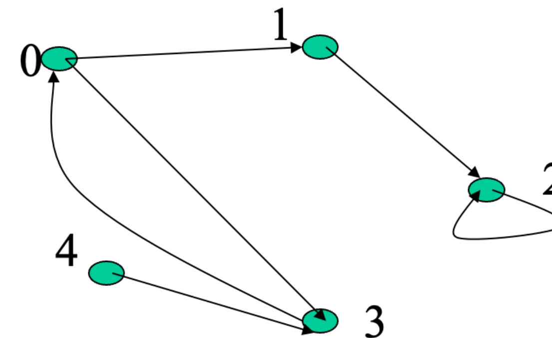
https://www-l2ti.univ-paris13.fr/~viennet/ens/2024-USTH-Graphs

Emmanuel Viennet
emmanuel.viennet@univ-paris13.fr

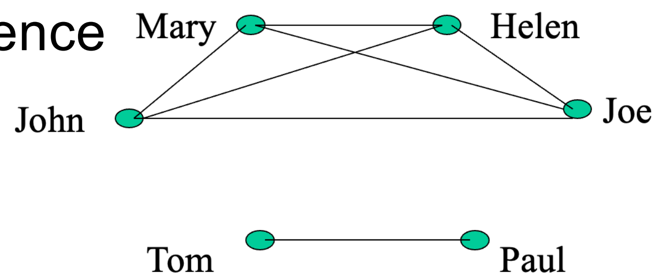
Paths

A path in a graph G is a **sequence of nodes** $x_1, x_2, \ldots, x_k$,
such that there is an edge from each node the next one in the sequence

- For example, the sequence 3, 0, 1, 2 is a path,
  but the sequence 0, 3, 2 is not a path because (0,2) is not an edge



- In this graph, the sequence **John, Mary, Joe, Helen** is a path, but the sequence **Helen, Tom, Paul** is not a path



Undirected Graphs

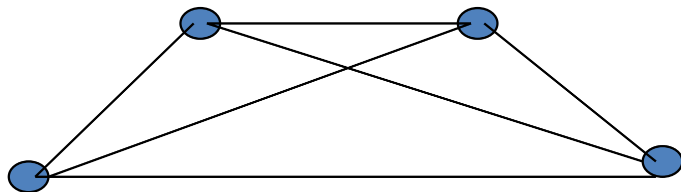
In the following, unless explicitly stated, we will focus on undirected graphs.

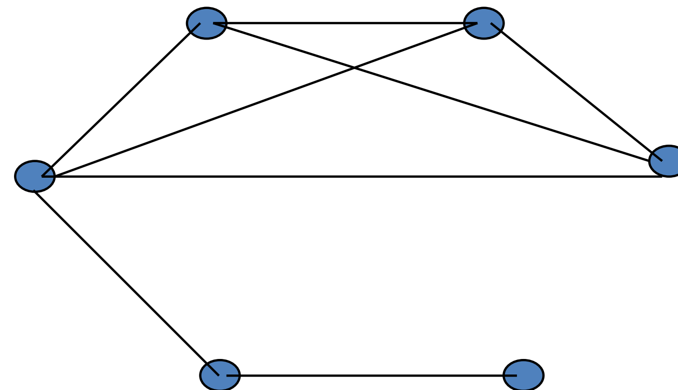
Most results and algorithms can be generalized to directed graphs.

Graph Connectivity

- A graph is said to be *connected* if there is a path between every pair of nodes.
  Otherwise, the graph is *disconnected*

Informally, a graph is connected if it hangs in one piece





Disconnected *Connected*

Connected Components

- If a graph is not connected, then each "piece" is called a connected component.
  - A piece in itself is connected, but if you bring any other node to it from the graph, it is no longer connected

- If the graph is connected, then the whole graph is one single connected component

- Of Interest: Given any undirected graph G,
  - Is G connected?
  - If not, find its connected components.

Graph Traversal techniques

The previous connectivity problem, as well as many other graph problems, can be solved using *graph traversal techniques*

There are two standard graph traversal techniques:

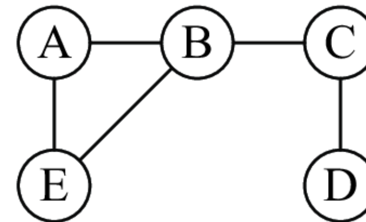
- *Depth-First Search* (DFS)
- *Breadth-First Search* (BFS)

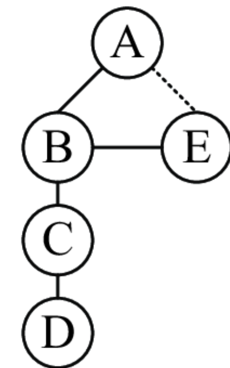
Graph Traversal techniques (2)

In both DFS and BFS, the nodes of the undirected graph are visited in a systematic manner so that **every node is visited exactly one**.

Both BFS and DFS give rise to a **tree**:

- When a node x is visited, it is labeled as visited, and it is added to the tree
- If the traversal got to node x from node y, y is viewed as the parent of x, and x a child of y



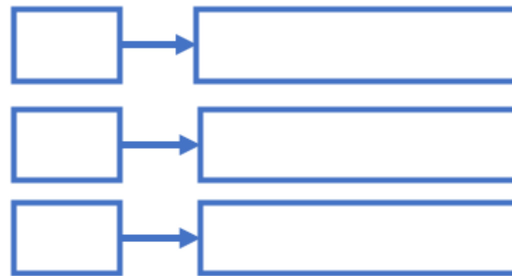


(a) undirected graph **(b)** DFS spanning tree

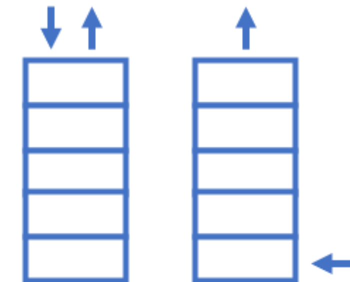
Graphs are a data structure

Array

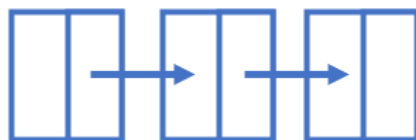
Hash Table



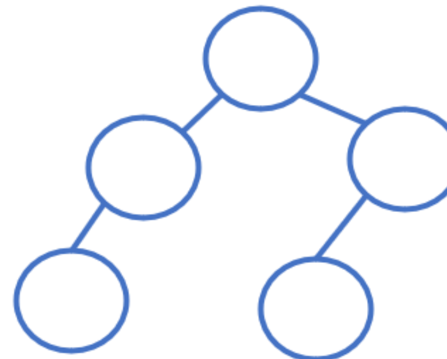
Stack & Queue



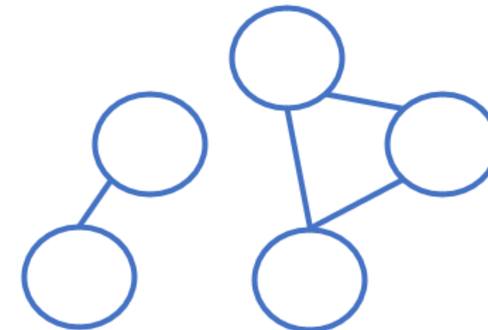
Linked List



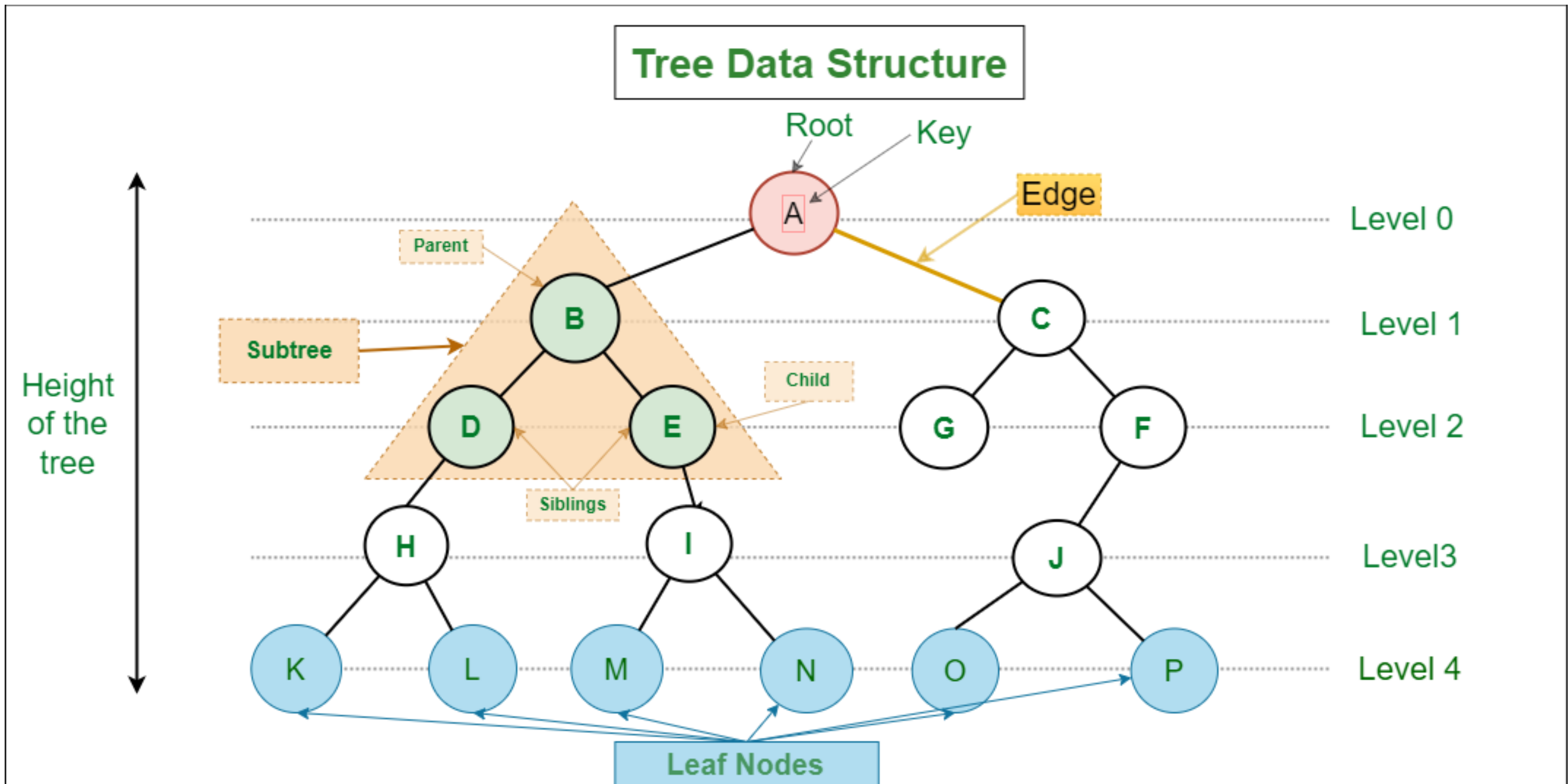
Tree



Graph



https://medium.com/codex/a-dummys-guide-to-linked-lists-part-1-44469f35f65a



https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials

Depth-First Search (DFS)

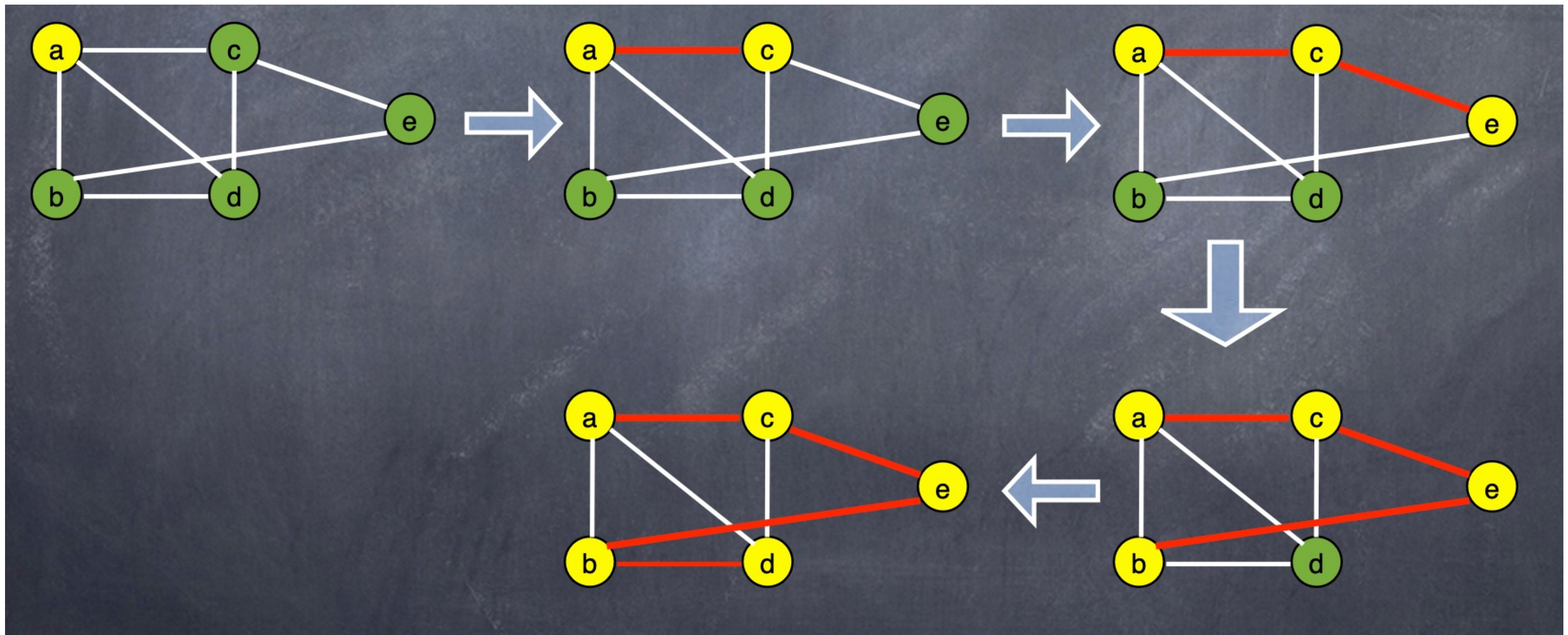
DFS algorithm:

1. Select an unvisited node x, visit it, and treat as the current node
2. Find an unvisited neighbor of the current node, visit it, and make it the new current node
3. If the current node has no unvisited neighbors, backtrack to the its parent, and make that parent the new current node
4. Repeat steps 3 and 4 until no more nodes can be visited
5. If there are still unvisited nodes, repeat from step 1

Depth-First Search (DFS)

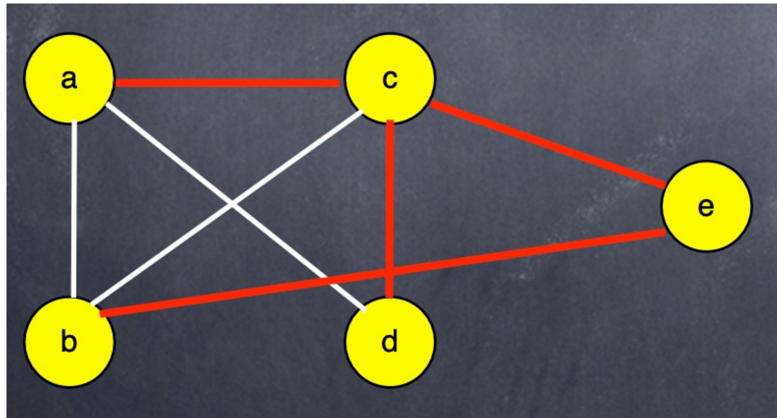
DFS algorithm:

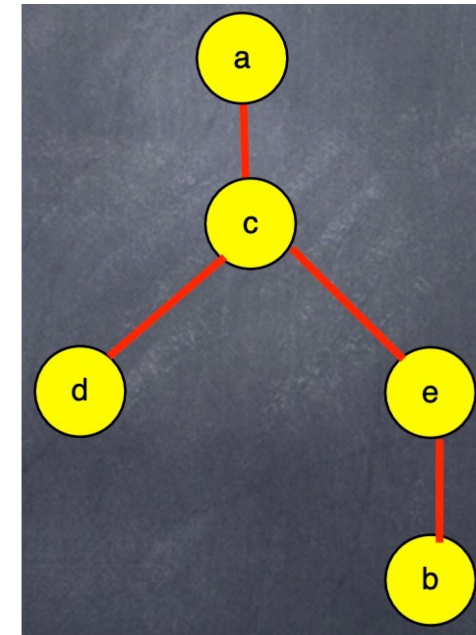
Keep exploring from most recently added node until you have to backtrack



Depth-First Search (DFS)

Theorem: Let T be a depth-first search tree. Let x and y be 2 nodes in the tree. Let (x,y) be an edge that is in G but not in T.
Then either x is an ancestor of y or y is an ancestor of x in T.





Depth-First Search (DFS)

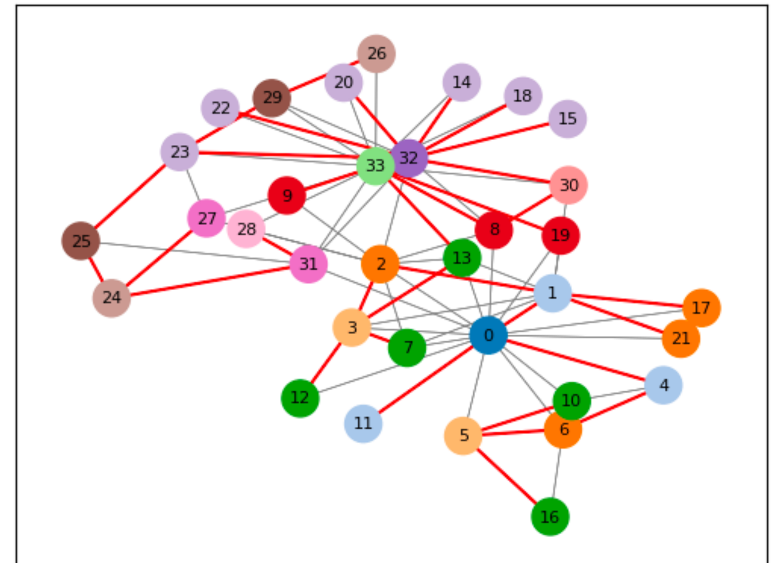
DFS algorithm: recursive formulation

```
def DFS(u):
    mark u as "explored"
    for each edge (u, v) incident to u:
        if v is not marked as "explored":
            DFS(v)
```

Coming next:

1. DFS Algorithm with a stack
2. Implement DFS in Python
3. Algorithms complexity
4. Breadth-First-Search (BFS)

Documents are here:



https://www-l2ti.univ-paris13.fr/~viennet/ens/2024-USTH-Graphs

Conclusion

Graph theory is a fundamental component of computer science with wide-ranging applications

- **Data Structures and Algorithms**: Graphs are essential in representing complex data structures like networks, which are central to various algorithms in computer science, such as those used in searching (like BFS and DFS), shortest path algorithms (like Dijkstra's and Bellman-Ford), and network flow algorithms.
- **Network Analysis**: Analyzing and optimizing computer networks, social networks, and web networks, including understanding the internet's topology, routing protocols, and analyzing social media interactions.
- **Problem Solving and Optimization**: Many complex computer science problems are modeled using graphs, including scheduling problems, resource allocation, and optimization problems (like the *Traveling Salesman Problem*), making graph theory a key tool for developing efficient solutions.
- **Database Theory**: Graphs are used in the modeling of databases, particularly in understanding relationships within network databases and for designing efficient data retrieval algorithms, including the use of graph databases in big data applications.
- **Artificial Intelligence and Machine Learning**: Graph theory plays a role in AI and ML, particularly in areas like semantic networks, neural networks, and in developing algorithms for clustering and pattern recognition, enhancing machine learning models' effectiveness in interpreting complex datasets.