Lecture 9

Go over Exam #1
Clustering vs. Classification
Clustering overview

Data **clustering techniques** are essential in data mining for discovering meaningful patterns and structures in datasets. Clustering aims to group similar data points together while separating dissimilar ones. Here are some common data clustering techniques used in data mining:

K-Means Clustering:
Method: K-Means partitions data into k clusters based on distance from cluster centroids.
Advantages: It is computationally efficient and works well with large datasets.
Considerations: The number of clusters (k) must be specified in advance, and it is sensitive to initial centroid selection.

Hierarchical Clustering:
Method: Hierarchical clustering creates a tree-like structure of clusters, with data points or groups merging step by step.
Advantages: It provides a hierarchy of clusters, allowing different granularity levels for analysis.
Considerations: Hierarchical clustering can be computationally intensive, and the choice of linkage method (single, complete, average, etc.) impacts results.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):
Method: DBSCAN groups data points into clusters based on their density and connectivity.
Advantages: It can discover clusters of arbitrary shapes and is robust to noise.
Considerations: DBSCAN requires setting parameters like the minimum number of points in a neighborhood.

Agglomerative Clustering:
Method: Agglomerative clustering starts with each data point as a single cluster and recursively merges the closest clusters.
Advantages: It is straightforward to implement and allows for a flexible number of clusters.
Considerations: Agglomerative clustering can be computationally demanding for large datasets.

Spectral Clustering:
Method: Spectral clustering transforms data into a lower-dimensional space and performs clustering in that space.
Advantages: It is effective for data with complex structures and works well when clusters have non-convex shapes.
*Consideration*s: Spectral clustering involves eigenvalue decomposition and can be computationally expensive.

Fuzzy C-Means Clustering:
Method: Fuzzy C-Means assigns data points to clusters with membership degrees, allowing points to belong to multiple clusters to varying degrees.
Advantages: It accommodates data points that have ambiguous cluster assignments.
Considerations: It requires the tuning of a fuzziness parameter.

Mean Shift Clustering:
Method: Mean Shift identifies clusters by seeking modes in the density of data points.
Advantages: It is robust to initializations and can discover clusters of varying shapes and sizes.
Considerations: Parameter selection, especially bandwidth, can impact results.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies):
Method: BIRCH is a hierarchical clustering method optimized for large datasets and online clustering.
Advantages: It is memory-efficient and can handle streaming data.
Considerations: BIRCH is sensitive to the choice of parameters.

Self-Organizing Maps (SOM):
Method: SOM is a type of artificial neural network that maps high-dimensional data to a lower-dimensional grid while preserving topological relationships.
Advantages: It can visualize complex data structures and clusters.
Considerations: SOM requires the tuning of grid size and learning rate.

OPTICS (Ordering Points To Identify the Clustering Structure):
Method: OPTICS is an extension of DBSCAN that produces a reachability plot to reveal the cluster structure.
Advantages: It provides a more detailed view of clusters and density.
Considerations: OPTICS can be computationally intensive, and parameter tuning is required.

The choice of clustering technique depends on the characteristics of the data, the desired granularity of clusters, and the goals of the data mining task. It often involves experimenting with different methods and evaluating the quality of clustering results based on domain-specific criteria.

Classification and clustering are two distinct techniques used in data analysis and data mining, each with its own purpose and methodology. Here are the key differences between classification and clustering:

1. ***Purpose***:
Classification: The main goal of classification is to assign data points to predefined categories or classes based on their features or attributes. It is a supervised learning technique, where the algorithm learns from labeled data to make predictions or assign labels to new, unlabeled data.
Clustering: Clustering, on the other hand, aims to group similar data points together based on their inherent similarities or patterns in the data. It is an unsupervised learning technique, and it doesn't require prior knowledge of the classes or labels.

2. Supervision:
Classification: Supervised learning algorithms are used for classification. These algorithms learn from a labeled training dataset and then make predictions on new, unseen data. The presence of labels or classes guides the learning process.
Clustering: Clustering is an unsupervised learning technique. It doesn't rely on labeled data. Instead, it groups data points based on their similarities or patterns, and there are no predefined classes or labels.

3. Data Labeling:
Classification: In classification, data points are assigned to predefined categories or classes. Each data point has a clear label indicating which class it belongs to.

Clustering: In clustering, data points are grouped together based on their similarities, but there are no predefined labels or classes. Clusters are formed based on the inherent structure of the data.

4. Algorithm Types:
Classification: Common classification algorithms include decision trees, random forests, support vector machines, k-nearest neighbors, and neural networks. These algorithms aim to learn the decision boundaries that separate different classes.
Clustering: Common clustering algorithms include K-means, hierarchical clustering, DBSCAN, and spectral clustering. These algorithms group data points based on similarity metrics or distance measures.

5. Evaluation:
Classification: The performance of a classification model can be evaluated using metrics like accuracy, precision, recall, F1 score, and ROC AUC, among others, to assess the model's ability to correctly predict class labels.
Clustering: The evaluation of clustering results is more challenging because there are no predefined classes. Metrics like silhouette score, Davies-Bouldin index, and the sum of squared errors (SSE) can be used to assess the quality of clusters.

6. Use Cases:
Classification: Classification is typically used for tasks like spam email detection, sentiment analysis, fraud detection, image recognition, and any problem where data points need to be assigned to predefined categories.
Clustering: Clustering is used in tasks like customer segmentation, anomaly detection, document grouping, and exploratory data analysis, where patterns and groupings in the data need to be discovered.

Classification and clustering are distinct techniques in data analysis, each suited for different types of use cases. Here are some different use cases for classification and clustering:

Use Cases for Classification:

- Spam Email Detection: Classify incoming emails as either spam or non-spam based on their content and attributes.
- Sentiment Analysis: Determine the sentiment of customer reviews or social media posts, such as whether they are positive, negative, or neutral.
- Credit Risk Assessment: Predict whether a loan applicant is likely to default on a loan based on their credit history, income, and other features.
- Disease Diagnosis: Classify medical images, such as X-rays or MRI scans, to detect diseases like cancer or diabetic retinopathy.
- Image Recognition: Identify objects or patterns in images, such as recognizing faces, animals, or objects.
- Handwriting Recognition: Classify handwritten characters or digits to convert them into machine-readable text.
- Fraud Detection: Identify fraudulent transactions in financial or online systems by categorizing them as fraudulent or legitimate.
- Customer Churn Prediction: Predict whether customers are likely to leave a subscription service or platform based on their behavior and usage patterns.
- Language Identification: Determine the language of a text document or speech sample.

- Document Classification: Automatically categorize documents into predefined classes, such as news articles into topics like politics, sports, or entertainment.

Use Cases for Clustering:

- Customer Segmentation: Group customers with similar behavior and characteristics to target marketing efforts more effectively.
- Anomaly Detection: Detect abnormal or fraudulent patterns in data by identifying clusters of data points that deviate from the norm.
- Image Compression: Use clustering to reduce the size of images while preserving their visual quality.
- Genomic Data Analysis: Cluster genes or genetic data to identify patterns or group genes with similar functions.
- Social Network Analysis: Identify communities or groups within social networks to understand the structure of connections.
- Recommendation Systems: Cluster users or items based on their preferences to provide personalized recommendations.
- Data Preprocessing: Preprocess data by grouping similar data points together before applying more specific analysis techniques.
- Text Document Clustering: Cluster text documents to discover topics or themes in large text corpora.
- Geospatial Analysis: Group geospatial data, such as locations of customers or incidents, to identify patterns or hotspots.
- Market Basket Analysis: Identify patterns of items frequently purchased together in retail data.

Using a clustering technique as a classification method is less common but can be appropriate in specific situations when certain conditions are met. Here are some scenarios when it may be suitable to use clustering for classification:

Lack of Labeled Data: If you have limited or no labeled data, and it's difficult or costly to obtain labels, clustering can be used to create "pseudo-labels." Clustering the data into groups based on similarities allows you to assign labels to the groups, effectively creating a form of unsupervised classification.

Feature Engineering: Clustering can be used as a feature engineering step to generate new features for a classification task. For example, you can cluster data points based on certain features and use the cluster assignments as additional features to enhance the performance of a classification model.

Semi-Supervised Learning: In semi-supervised learning, a small portion of the data is labeled, and the rest is unlabeled. You can use clustering to group unlabeled data points and then propagate labels from the labeled data to the clustered groups. This can help expand the training set for a classification model.

Imbalanced Datasets: In cases of imbalanced datasets, where one class significantly outweighs the others, clustering can be used to create synthetic samples for the minority class. Oversampling techniques, such as Synthetic Minority Over-sampling Technique (SMOTE), can be applied to balance the dataset for classification.

Multi-Step Classification: In some cases, a multi-step classification approach can benefit from clustering. For instance, you might first cluster data into groups based on similarities and then apply a specific classifier for each cluster to address the heterogeneity within the data.

Outlier Detection: Clustering can be used as a preliminary step to identify and classify outliers or anomalies. Once outliers are detected, they can be labeled as a separate class, and a classification model can be trained to distinguish between normal and anomalous instances.

Data Exploration and Preprocessing: Clustering can help in the data exploration phase by revealing hidden structures or patterns in the data. This insight can guide the selection of appropriate classification algorithms and preprocessing steps.

It's important to note that using clustering for classification can introduce challenges, such as the choice of the number of clusters (K) and the interpretation of cluster assignments as class labels. Careful consideration and evaluation are required to ensure that the clustering-based classification approach is effective and appropriate for the specific problem at hand.

Clustering and rule mining are two distinct data analysis techniques, but they can be used together for pattern discovery and knowledge extraction in data. The process of using clustering for rule mining typically involves the following steps:

Data Preparation: Begin by collecting and preparing your dataset, ensuring it's in a suitable format for analysis.
Clustering: Apply a clustering algorithm to the dataset to group similar data points together. Common clustering algorithms include K-Means, Hierarchical Clustering, DBSCAN, and OPTICS, among others. The result of clustering is the assignment of data points to clusters or groups. Each cluster represents a set of data points with similar characteristics.
Rule Mining: After clustering, you can perform rule mining within each cluster to extract meaningful patterns, associations, or rules that describe the behavior of data points within that cluster.

The choice of the rule mining algorithm depends on the specific type of rules you want to discover. Common rule mining techniques include Association Rule Mining, Classification Rule Mining, and Decision Tree induction.

Association Rule Mining: In association rule mining, you aim to find interesting associations or co-occurrences among different attributes or items in your data. The Apriori algorithm is a well-known technique for association rule mining. It can identify rules like "If A and B, then C" where A, B, and C are attributes or items.

Classification Rule Mining: If you have labeled data and want to find rules that describe how attributes or features contribute to the classification of data points into different classes, you can use classification rule mining. Common algorithms for classification rule mining include C4.5, CART, and Random Forest.

Decision Tree Induction: Decision trees can be used for rule mining, especially when you want to discover rules for classification or prediction tasks. Decision tree algorithms, such as C4.5 or ID3, create a tree-like structure with rules at each node to partition the data.

Evaluate and Interpret Rules: Once you've mined rules within each cluster, evaluate the rules for significance, support, confidence, or other relevant metrics. Interpret the rules to gain insights into the behavior or characteristics of data points within specific clusters.

Iterate and Refine: Depending on your findings, you may need to iterate the process, adjusting clustering or rule mining parameters and strategies.

The combination of clustering and rule mining can help identify and understand patterns and relationships within different subsets of your data. This can be useful in various applications, including customer segmentation, market basket analysis, anomaly detection, and more. It's important to adapt the process to the specific goals and characteristics of your dataset and problem domain.

Resources:

1. https://www.coveo.com/blog/clustering-and-classification-in-ecommerce/
2. https://www.geeksforgeeks.org/ml-classification-vs-clustering/
3. https://www.simplilearn.com/tutorials/data-analytics-tutorial/classification-vs-clustering
4. https://www.analyticsvidhya.com/blog/2023/05/classification-vs-clustering/
5. https://blog.bismart.com/en/classification-vs.-clustering-a-practical-explanation
6. https://www.educative.io/answers/classification-vs-clustering
7. https://stats.stackexchange.com/questions/474396/classification-vs-clustering-question
8. https://www.datacamp.com/blog/classification-vs-clustering-in-machine-learning
9. https://unstop.com/blog/classification-vs-clustering
10. https://www.geeksforgeeks.org/clustering-in-r-programming/#
11. https://domino.ai/blog/clustering-in-r
12. https://www.statmethods.net/advstats/cluster.html
13. https://www.analyticsvidhya.com/blog/2021/04/beginners-guide-to-clustering-in-r-program/

Lecture 10

Hierarchical Clustering
Agglomerative Clustering
Spectral Clustering

Hierarchical clustering is a popular clustering technique used in data mining and machine learning. It organizes data points into a hierarchical structure of clusters, often represented as a tree-like diagram called a dendrogram. Hierarchical clustering can be categorized into two main approaches: agglomerative (bottom-up) and divisive (top-down). Here, we'll focus on the agglomerative approach, which is more commonly used.

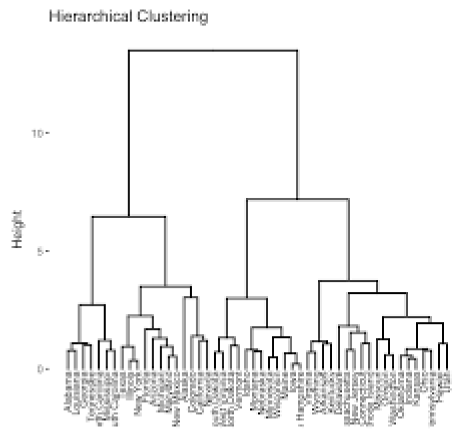
Agglomerative Hierarchical Clustering: Agglomerative hierarchical clustering starts with each data point as its cluster and then recursively merges the closest clusters, forming a hierarchy. Here's a step-by-step overview of how it works:

1. Initialization: Begin with each data point as a separate cluster. If you have n data points, you start with n clusters, each containing a single data point.

2. Merge Closest Clusters: Identify the two closest clusters based on a distance metric, often using methods like single-linkage, complete-linkage, or average-linkage. Merge these two clusters into a new, larger cluster.

3. Update Distance Matrix: Recalculate the distances between this newly formed cluster and all other clusters, resulting in an updated distance matrix.

4. Repeat: Repeat steps 2 and 3 until there is only one cluster left, which contains all data points. The hierarchy of clusters forms a dendrogram. *Dendrogram*: A dendrogram is a tree-like diagram that represents the hierarchical structure of clusters. The vertical lines in the dendrogram indicate the merging of clusters at each step. The height at which clusters merge reflects the dissimilarity between them.



Key Considerations:
Choice of Distance Metric: The choice of distance metric (Euclidean, Manhattan, etc.) significantly affects the results. The method for measuring the distance between clusters also matters, as single-linkage, complete-linkage, and average-linkage can lead to different outcomes.

Number of Clusters: Hierarchical clustering does not require you to specify the number of clusters (k) beforehand. Instead, you can choose the desired level of granularity by cutting the dendrogram at a certain height.

Interpreting the Dendrogram: The dendrogram provides insights into the hierarchical relationships between clusters. By cutting it at various heights, you can obtain different clusterings with varying numbers of clusters.

Scalability: Hierarchical clustering can be computationally intensive, especially for large datasets, as it involves repeated distance calculations and matrix updates.

Applications:
- Hierarchical clustering is widely used in various fields and applications, including:
- Biology: For phylogenetic tree construction and gene expression analysis.
- Image Analysis: For segmentation and object detection.
- Marketing: For customer segmentation and market basket analysis.

- Social Network Analysis: For community detection.
- Document Clustering: For organizing documents into topic hierarchies.

Hierarchical clustering is a versatile technique that provides insights into the structure of data, making it valuable for exploratory data analysis and pattern discovery. It allows you to visualize the relationships between clusters and analyze data at multiple levels of granularity.

An example of hierarchical clustering in R using the hclust function and the iris dataset. In this example, we will perform agglomerative hierarchical clustering on the sepal length and sepal width measurements from the iris dataset. Hierarchical clustering will create a dendrogram, allowing us to explore the hierarchical relationships between data points.

Here's how to perform hierarchical clustering in R:

```
# Load the iris dataset
data(iris)
# Select the relevant features (sepal length and sepal width)
iris_features <- iris[, c("Sepal.Length", "Sepal.Width")]
# Perform hierarchical clustering
hc <- hclust(dist(iris_features), method = "complete")
# Complete-linkage clustering # Plot the dendrogram
plot(hc, main = "Hierarchical Clustering of Iris Data", xlab =
"Species")
```

This code does the following:
Loads the iris dataset and selects the relevant features (sepal length and sepal width). Performs hierarchical clustering using the hclust function. We use the "complete" linkage method, which calculates the distance between clusters as the maximum distance between their individual data points.

You can choose other linkage methods like "single," "average," or "ward.D2" based on your preferences. It then plots the dendrogram using the plot function, which shows the hierarchical structure of the clusters. In the dendrogram, data points are represented as leaves, and the height at which branches merge indicates the dissimilarity between clusters.

When you run this code, you'll see a dendrogram that illustrates the hierarchical clustering of the iris data. Data points are grouped together based on their sepal length and sepal width measurements, and you can choose to cut the dendrogram at a certain height to obtain different clusterings with varying numbers of clusters. This hierarchical approach provides insights into the relationships between clusters and allows you to explore the data at multiple levels of granularity.

Agglomerative clustering is a hierarchical clustering technique that falls under the broader category of hierarchical clustering algorithms. Unlike divisive clustering, which starts with all data points in one cluster and recursively divides them into smaller clusters, agglomerative clustering begins with each data point in a separate cluster and progressively merges them into larger clusters. Here's how agglomerative clustering works:

Basic Procedure:
Initialization: Start with each data point as a single cluster. If you have n data points, you begin with n clusters, each containing one data point.

Merge Closest Clusters: Identify the two closest clusters based on a chosen linkage method, which determines how to measure the distance between clusters. Common linkage methods include:

- *Single Linkage*: Based on the minimum distance between data points in the two clusters.
- *Complete Linkage*: Based on the maximum distance between data points in the two clusters.
- *Average Linkage*: Based on the average distance between data points in the two clusters.
- *Ward's Linkage*: Minimizes the increase in variance within clusters when merging.

Update Distance Matrix: Recalculate the distances between the newly formed cluster and all other clusters, resulting in an updated distance matrix.

Repeat: Continue merging the two closest clusters and updating the distance matrix until there is only one cluster left, which contains all data points. The hierarchy of clusters forms a dendrogram.

Dendrogram: The output of agglomerative clustering is often visualized as a dendrogram, which is a tree-like diagram that represents the hierarchy of clusters. The vertical lines in the dendrogram indicate the merging of clusters at each step, with the height at which they merge reflecting the dissimilarity between them.

Key Considerations:
Agglomerative clustering does not require you to specify the number of clusters (k) in advance. Instead, you can choose the desired level of granularity by cutting the dendrogram at a certain height.

The choice of linkage method can significantly affect the results. Different linkage methods may lead to different cluster shapes and sizes.

The order in which clusters are merged can impact the final clustering result. The algorithm can start with the two closest clusters or the two farthest clusters.
Agglomerative clustering is computationally intensive, especially for large datasets, as it involves repeated distance calculations and matrix updates.

Applications:
Agglomerative clustering is used in various fields and applications, including biology for phylogenetic tree construction, image analysis for segmentation and object detection, marketing for customer segmentation, social network analysis for community detection, and document clustering for organizing documents into topic hierarchies.

Agglomerative clustering is a versatile technique that provides insights into the structure of data and allows for hierarchical exploration of clusters. It is particularly valuable when you want to understand relationships between clusters at different levels of granularity.

An example of agglomerative clustering in R using the agnes function from the cluster package. In this example, we will use a synthetic dataset and apply agglomerative clustering to group the data points into clusters. We'll also visualize the results using a dendrogram.

First, ensure that you have the cluster package installed. You can install it using the following command if it's not already installed:

```
install.packages("cluster")
```

Now, let's create a synthetic dataset and perform agglomerative clustering:

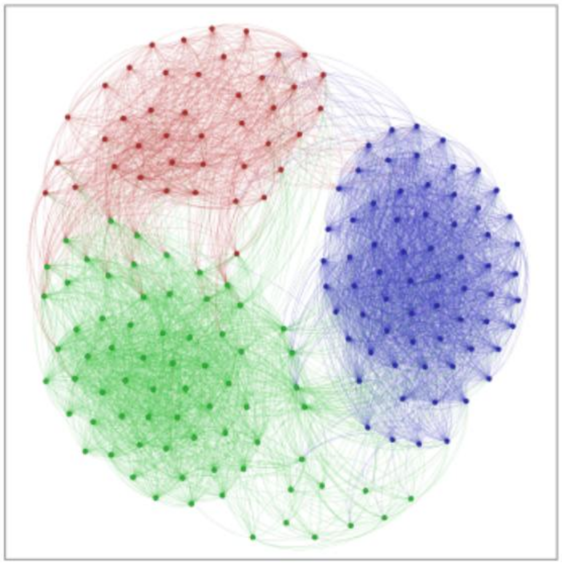
```
# Load the cluster library
library(cluster)
# Generate synthetic data
set.seed(123)
data <- matrix(rnorm(200, mean = 0, sd = 1), ncol = 2)
# Perform agglomerative clustering
agglo_result <- agnes(data, method = "complete")
# Using complete linkage
# Plot the dendrogram
plot(agglo_result, main = "Agglomerative Clustering Example")
```

In this code, we: Load the cluster library. Generate a synthetic dataset with two clusters using random data. Perform agglomerative clustering using the agnes function. We use the "complete" linkage method, which calculates the distance between clusters as the maximum distance between their individual data points. You can choose other linkage methods such as "single" or "average" based on your preferences. Plot the dendrogram using the plot function to visualize the hierarchical clustering results.

When you run this code, you will see a dendrogram that illustrates the hierarchical clustering of the synthetic data. The dendrogram shows the hierarchical structure of clusters, with the height at which branches merge indicating the dissimilarity between clusters.

You can choose to cut the dendrogram at a certain height to obtain different clusterings with varying numbers of clusters. Agglomerative clustering provides a hierarchical view of the data's clustering structure, allowing you to explore the data at different levels of granularity.

Spectral clustering is a powerful and versatile clustering technique that is widely used in data mining and machine learning. It differs from traditional clustering methods like K-Means or hierarchical clustering by embedding the data into a lower-dimensional space and performing clustering in that space. Spectral clustering is particularly useful for discovering clusters with complex shapes, handling non-convex clusters, and finding clusters of varying sizes. Here's how spectral clustering works:



Basic Procedure:
Affinity Matrix: Start with your data, which can be represented as a matrix of pairwise affinities or similarities between data points. The affinity matrix encodes how similar or related each data point is to every other data point. Common similarity measures include Euclidean distance, cosine similarity, or other kernel functions.

Graph Construction: Use the affinity matrix to construct a graph where data points are nodes, and the affinities represent edge weights. You can choose to build a fully connected graph or a k-nearest neighbor graph, depending on the nature of your data.

Spectral Embedding: Compute the Laplacian matrix of the graph, which characterizes the connectivity and structure of the data. This matrix can be used to derive the graph Laplacian eigenvectors.

Eigenvector Decomposition: Compute the eigenvectors and eigenvalues of the Laplacian matrix. These eigenvectors represent the embedded representation of the data in a lower-dimensional space.

Cluster Assignment: After obtaining the eigenvectors, perform clustering on the embedded data using conventional clustering techniques like K-Means or normalized cuts. You can choose the number of clusters (k) or use techniques like the eigengap heuristic to determine the optimal k.

Key Considerations:
- Spectral clustering can handle complex cluster shapes and is less sensitive to the initialization of cluster centers compared to K-Means.
- It can uncover clusters with varying densities and sizes.
- The choice of the similarity metric and the graph construction method is crucial, and it can affect the results significantly. Different choices can lead to different cluster structures.
- Spectral clustering may require parameter tuning, such as the number of neighbors in the k-nearest neighbor graph or the number of eigenvectors to use.
- It is suitable for both numerical and categorical data.
- Spectral clustering is computationally intensive, especially for large datasets, due to the eigenvector decomposition step.

Applications: Spectral clustering is used in various applications, including:

- *Image segmentation*: For partitioning images into regions with similar properties.
- *Document clustering*: For organizing documents into meaningful groups based on their content.
- *Biological data analysis*: For clustering genes or proteins in genomic and proteomic studies.
- *Community detection in social networks*: For identifying communities or groups of nodes with similar properties.
- *Anomaly detection*: For identifying data points that deviate from the norm in a dataset.

Resources:

1. https://www.r-bloggers.com/2016/01/hierarchical-clustering-in-r-2/
2. https://uc-r.github.io/hc_clustering
3. https://www.datacamp.com/tutorial/hierarchical-clustering-R
4. https://www.geeksforgeeks.org/hierarchical-clustering-in-r-programming/
5. https://www.statology.org/hierarchical-clustering-in-r/
6. https://bookdown.org/rdpeng/exdata/hierarchical-clustering.html
7. https://medium.com/@sukmaanindita/wards-hierarchical-clustering-method-using-r-studio-a47b5a79cb4d
8. https://online.stat.psu.edu/stat508/lesson/12/12.8
9. https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/
10. https://mhahsler.github.io/Introduction_to_Data_Mining_R_Examples/book/clustering-analysis.html
11. https://www.di.fc.ul.pt/~jpn/r/spectralclustering/spectralclustering.html
12. https://rpubs.com/nurakawa/spectral-clustering
13. https://rpubs.com/gargeejagtap/SpectralClustering