417526: Computer Laboratory II

Information Retrieval

Teaching Scheme:

PR: 02 Hours/Week

Credit 02

Examination Scheme and Marks

Term Work (TW): 50

Marks Practical (PR): 25 Marks

List of Assignments: -

1. Write a program for pre-processing of a text document such as stop word removal, stemming.
2. Implement a program for retrieval of documents using inverted files.
3. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API.
4. Implement e-mail spam filtering using text classification algorithm with appropriate dataset.
5. Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.
6. Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).
7. Build the web crawler to pull product information and links from an e-commerce website.

Course Objectives:

- Understand the concepts of information retrieval and web mining
- Understand information retrieval process using standards available tools

Course Outcomes:

CO1: Apply various tools and techniques for information retrieval and web mining

CO2: Evaluate and analyze retrieved information

Learning Resources

Text Books:

1. C. Manning, P. Raghavan, and H. Schütze, “Introduction to Information Retrieval”, Cambridge University Press, 2008
2. Ricardo Baeza-Yates, Berthier Riberio–Neto, “Modern Information Retrieval”, Pearson Education, ISBN: 81-297-0274-6

3. C.J. Rijsbergen, “Information Retrieval”, 2nd edition, ISBN: 978-408709293
4. Ryan Mitchell, “Web Scraping with Python”, O’reilly

Reference Books:

1. S. Buttcher, C. Clarke and G. Cormack, “Information Retrieval: Implementing and Evaluating Search Engines” MIT Press, 2010, ISBN: 0-408-70929-4
2. Amy N. Langville and Carl D. Meyer, “Google's PageRank and Beyond: The Science of Search Engine Rankings”, Princeton University Press, ISBN: 9781400830329

e-Books:

1. http://nlp-iiith.vlabs.ac.in/

CO-PO MAPPING												
Course Outcome (COs)	**Program Outcomes (POs)**											
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	1	2	3	2	-	-	-	-	-	-	1
CO2	1	1	2	3	2	-	-	-	-	-	-	1

Assignment No. 1

Title:
Stop Word Removal

Problem Statement:
Write a program for pre-processing of a text document such as stop word removal, stemming.

Software Requirement:
Python 3.9 and Jupiter/PyCharm/Any Equivalent Editor

Input:
Text Document

Output:
Text Document without stop words and stemming.

Theory:

Natural Language Processing (NLP) is the branch of Artificial Intelligence that allows machines to interpret human language. However, the same cannot be used directly by the machine, and we need to pre-process the same first.

Text pre-processing is the process of preparing text data so that machines can use the same to perform tasks like analysis, predictions, etc. There are many different steps in text pre-processing but in this article, we will only get familiar with stop words, why do we remove them, and the different libraries that can be used to remove them.

What is stemming?

Stemming is the process of reducing a word to its stem that affixes to suffixes and prefixes or to the roots of words known as "lemmas". Stemming is important in natural language understanding (NLU) and natural language processing (NLP).

Stemming is a part of linguistic studies in morphology as well as artificial intelligence (AI) information retrieval and extraction. Stemming and AI knowledge extract meaningful information from vast sources like big data or the internet since additional forms of a word related to a subject may need to be searched to get the best results. Stemming is also a part of queries and internet search engines.

What are stop words?

The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the text. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”.

Why do we remove stop words?

Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to

the important information. In order words, we can say that the removal of such words does not show any negative consequences on the model we train for our task.

Removal of stop words definitely reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.

What are the different libraries to remove stop words?

Some of the libraries used for the removal of English stop words, the stop words list along with the code are given below.

- **Natural Language Toolkit (NLTK):**
  NLTK is an amazing library to play with natural language. When you will start your NLP journey, this is the first library that you will use.

- **spaCy:**
  spaCy is an open-source software library for advanced NLP. This library is quite popular now and NLP practitioners use this to get their work done in the best way.
- **Gensim:**
  Gensim (Generate Similar) is an open-source software library that uses modern statistical machine learning.
- **Scikit-Learn:**
  Scikit-Learn needs no introduction. It is a free software machine learning library for Python. It is probably the most powerful library for machine learning.

Conclusion:

Thus, we have successfully implemented a program for text preprocessing.

Outcome:

Upon completion students will be able to:

Understand the stop word removal and stemming from a given text document.

Assignment No. 2

Title:
Retrieval of Inverted Files

Problem Statement:
Implement a program for retrieval of documents using inverted files.

Software Requirement:
Python 3.9 and Jupiter/PyCharm/Any Equivalent Editor

Input:
Text Files

Output:
Indexed Inverted Files.

Theory:

An Inverted Index is a data structure used in information retrieval systems to efficiently retrieve documents or web pages containing a specific term or set of terms. In an inverted index, the index is organized by terms (words), and each term points to a list of documents or web pages that contain that term.

Inverted indexes are widely used in search engines, database systems, and other applications where efficient text search is required. They are especially useful for large collections of documents, where searching through all the documents would be prohibitively slow.

An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. In simple words, it is a hashmap-like data structure that directs you from a word to a document or a web page.

Example: Consider the following documents.

Document 1: The quick brown fox jumped over the lazy dog.
Document 2: The lazy dog slept in the sun.

To create an inverted index for these documents, we first tokenize the documents into terms, as follows.

Document 1: The, quick, brown, fox, jumped, over, the lazy, dog.
Document 2: The, lazy, dog, slept, in, the, sun.

Next, we create an index of the terms, where each term points to a list of documents that contain that term, as follows.

The -> Document 1, Document 2

Quick -> Document 1
Brown -> Document 1
Fox -> Document 1
Jumped -> Document 1
Over -> Document 1
Lazy -> Document 1, Document 2
Dog -> Document 1, Document 2
Slept -> Document 2
In -> Document 2
Sun -> Document 2

To search for documents containing a particular term or set of terms, the search engine queries the inverted index for those terms and retrieves the list of documents associated with each term. The search engine can then use this information to rank the documents based on relevance to the query and present them to the user in order of importance.

Suppose we want to search the texts “hello everyone, ” “this article is based on an inverted index, ” and “which is hashmap-like data structure“. If we index by (text, word within the text), the index with a location in the text is:

hello	(1, 1)
everyone	(1, 2)
this	(2, 1)
article	(2, 2)
is	(2, 3); (3, 2)
based	(2, 4)
on	(2, 5)
inverted	(2, 6)
index	(2, 7)
which	(3, 1)
hashmap	(3, 3)
like	(3, 4)
data	(3, 5)
structure	(3, 6)

The word “hello” is in document 1 (“hello everyone”) starting at word 1, so has an entry (1, 1), and the word “is” is in documents 2 and 3 at ‘3rd’ and ‘2nd’ positions respectively (here position is based on the word).

The index may have weights, frequencies, or other indicators.

Steps to Build an Inverted Index

Fetch the Document: Removing of Stop Words: Stop words are the most occurring and useless words in documents like “I”, “the”, “we”, “is”, and “an”. Stemming of Root Word: Whenever I want to search for “cat”, I want to see a document that has information about it. But the word present in the document is called “cats” or “catty” instead of “cat”. To relate both words, I’ll chop some part of every word I read so that I could get the “root word”. There are standard tools for performing this like “Porter’s Stemmer”.

Record Document IDs: If the word is already present add a reference of the document to index else creates a new entry. Add additional information like the frequency of the word, location of the word, etc.

Example:

Words	Document
ant	doc1
demo	doc2
world	doc1, doc2

Conclusion:

Thus, we have successfully implemented a program for retrieval of Inverted files.

Outcome:

Upon completion students will be able to:

Understand the retrieval of Inverted files.

Assignment No. 3

Title:
construct a Bayesian network considering medical data.

Problem Statement:
Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API.

Software Requirement:
Python 3.9 and Jupiter/PyCharm/Any Equivalent Editor

Input:
Standard Heart Disease Data Set

Output:
Probability of Heart Disease.

Theory:

A Bayesian network is a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable. Bayesian network consists of two major parts: a directed acyclic graph and a set of conditional probability distributions

- The directed acyclic graph is a set of random variables represented by nodes.
- The conditional probability distribution of a node (random variable) is defined for every possible outcome of the preceding causal node(s).

For illustration, consider the following example. Suppose we attempt to turn on our computer, but the computer does not start (observation/evidence). We would like to know which of the possible causes of computer failure is more likely. In this simplified illustration, we assume only two possible causes of this misfortune: electricity failure and computer malfunction.

The corresponding directed acyclic graph is depicted in below figure.

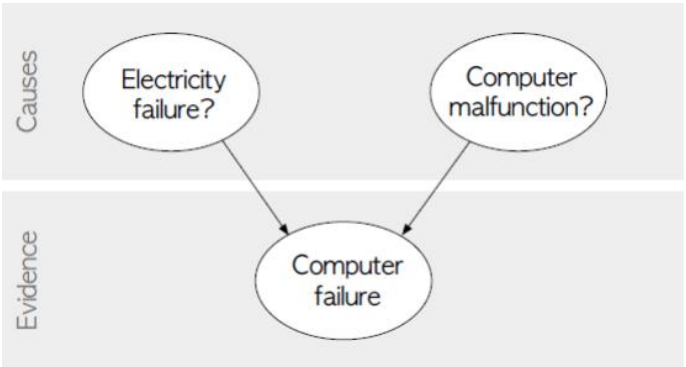


Fig: Directed acyclic graph representing two independent possible causes of a computer failure.

The goal is to calculate the posterior conditional probability distribution of each of the possible unobserved causes given the observed evidence, i.e. P [Cause | Evidence].

Data Set:

Title: Heart Disease Databases

The Cleveland database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "Heartdisease" field refers to the presence of heart disease in the patient.

It is integer valued from 0 (no presence) to 4.

Database:	0	1	2	3	4	Total
Cleveland:	164	55	36	35	13	303

Attribute Information:

1. age: age in years
2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type
    - Value 1: typical angina
    - Value 2: atypical angina
    - Value 3: non-anginal pain
    - Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholestoral in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results
    - Value 0: normal
    - Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    - Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment
    - Value 1: upsloping
    - Value 2: flat
    - Value 3: downsloping
12. ca = number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
14. Heartdisease: It is integer valued from 0 (no presence) to 4. Diagnosis of heart disease (angiographic disease status)

Some instance from the dataset:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Heartdisease
63	1	1	145	233	1	2	150	o	2.3	3	o	6	o
67	1	4	160	286	o	2	108	1	1.5	2	3	3	2
67	1	4	120	229	o	2	129	1	2.6	2	2	7	1
41	o	2	130	204	o	2	172	o	1.4	1	o	3	o
62	o	4	140	268	o	2	160	o	3.6	3	2	3	3
60	1	4	130	206	o	2	132	1	2.4	2	2	7	4

Conclusion:

Thus, we have successfully implemented a program to construct a Bayesian network considering medical data.

Outcome:

Upon completion students will be able to:

Understand the to construct a Bayesian network using medical data set.

Assignment No. 4

Title:
Email Spam Filtering.

Problem Statement:
Implement e-mail spam filtering using text classification algorithm with appropriate dataset.

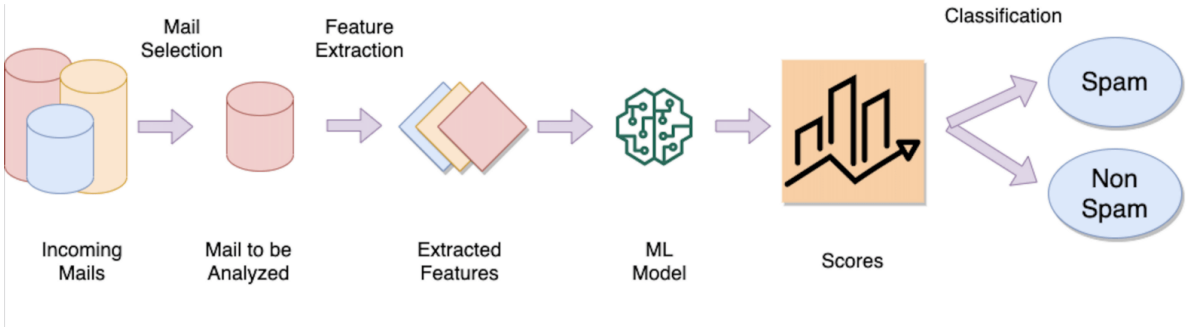
Software Requirement:
Python 3.9 and Jupiter/PyCharm/Any Equivalent Editor

Input:
Data Set

Output:
Spam and Not Spam Classification.

Theory:

Email spam filtering using text classification algorithms is a common and effective approach to automatically identify and filter out unwanted or malicious emails from your inbox. Text classification algorithms use machine learning techniques to analyze the content of emails and assign them to one of two classes: "spam" or "not spam" (also known as "ham").



Here's a high-level overview of how this process works:

1. Data Collection: To train a text classification model, you need a labeled dataset that contains a substantial number of emails categorized as spam or not spam. This dataset is used to train and evaluate the model.

2. Preprocessing: The first step is to preprocess the text data. This includes tasks like tokenization (breaking the text into words or tokens), lowercasing, removing punctuation, and handling special characters.

3. Feature Extraction: Transform the text data into numerical features that can be used by machine learning algorithms. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings (e.g., Word2Vec, GloVe).

4. Model Selection: Choose an appropriate machine learning algorithm for text classification. Common choices include:

    - Naive Bayes: A simple and fast algorithm that works well for spam classification.
    - Support Vector Machines (SVM): Effective for separating data into two classes.
    - Decision Trees: Useful for creating interpretable models.
    - Random Forest: An ensemble method that combines multiple decision trees. Neural Networks: Deep learning models like CNNs or RNNs can be used for more complex and accurate classification.
    - Model Training: Train the selected model on the preprocessed email data. This involves feeding the model with the labeled dataset and adjusting its parameters to optimize its performance.

5. Evaluation: Use a separate dataset (or cross-validation) to evaluate the model's performance. Common evaluation metrics include accuracy, precision, recall, F1-score, and ROC curves.

6. Deployment: Once the model performs satisfactorily, it can be deployed into your email system to automatically classify incoming emails as spam or not spam.

7. Ongoing Maintenance: Spam email characteristics change over time, so it's important to continuously update and retrain your spam filter model to adapt to new spamming techniques.

8. User Feedback: Incorporate user feedback to improve the model. Users can report false positives (legitimate emails mistakenly classified as spam) and false negatives (spam emails that weren't caught). This feedback can be used to fine-tune the model.

9. Additional Techniques: Consider using additional techniques like blacklists, whitelists, and heuristic rules (e.g., checking for suspicious links) to enhance the accuracy of the spam filter.

It's worth noting that spam filters are a cat-and-mouse game. As spammers get more sophisticated, spam filters need to evolve and adapt to new tactics. Machine learning models can help with this adaptability but require ongoing maintenance and monitoring to stay effective.

Conclusion:

Thus, we have successfully implemented a program for Email Spam Filtering using Text Classification.

Outcome:

Upon completion students will be able to:

Understand the text classification for email filtering.

Assignment No. 5

Title:
Agglomerative hierarchical clustering.

Problem Statement:
Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.

Software Requirement:
Python 3.9 and Jupiter/PyCharm/Any Equivalent Editor

Input:
Data Set

Output:
Clusters of given data.

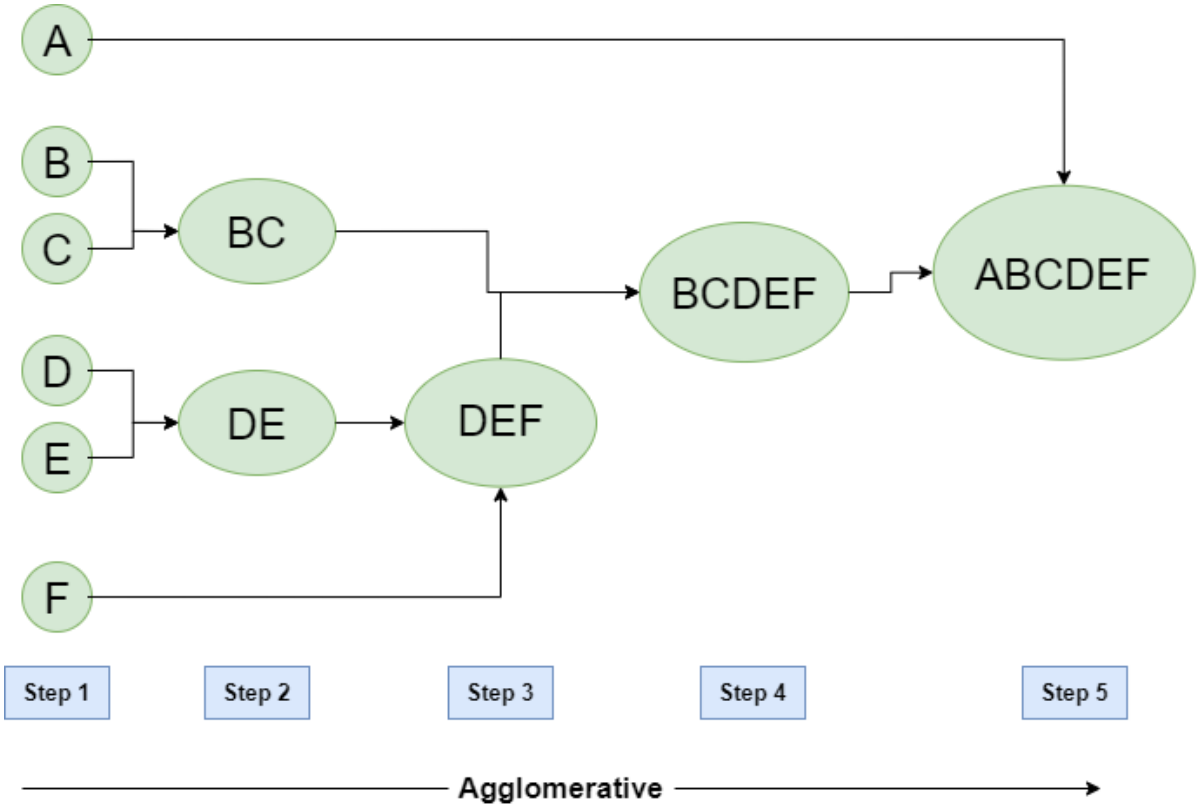
Theory:

The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the bottom-up approach. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.
This hierarchy of clusters is represented in the form of the dendrogram.

Algorithm :

```
given a dataset (d1, d2, d3, ....dN) of size N
# compute the distance matrix
for i=1 to N:
  # as the distance matrix is symmetric about
  # the primary diagonal so we compute only lower
  # part of the primary diagonal
  for j=1 to i:
    dis_mat[i][j] = distance[di, dj]
each data point is a singleton cluster
repeat
  merge the two cluster having minimum distance
  update the distance matrix
until only a single cluster remains
```

How the Agglomerative Hierarchical clustering Work?
The working of the AHC algorithm can be explained using the below steps:



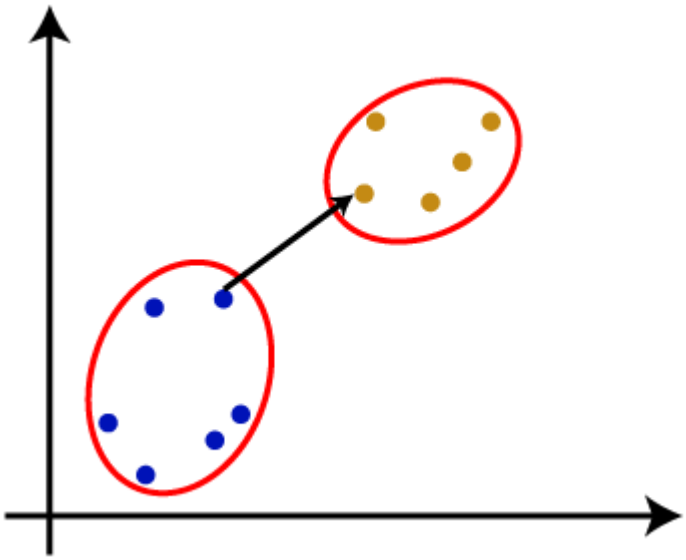
Steps:

- Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.
- In the second step, comparable clusters are merged together to form a single cluster. Let's say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly to cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]
- We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]
- Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We're now left with clusters [(A), (BCDEF)].
- At last, the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

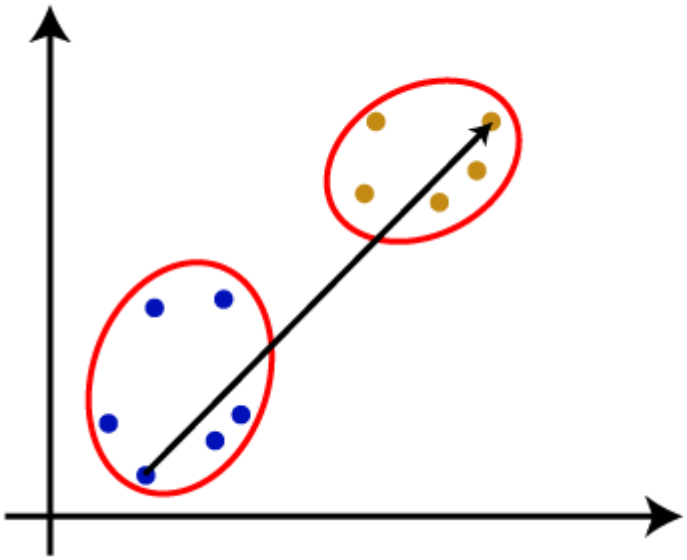
Measure for the distance between two clusters

As we have seen, the closest distance between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called Linkage methods. Some of the popular linkage methods are given below:

Single Linkage: It is the Shortest Distance between the closest points of the clusters. Consider the below image:

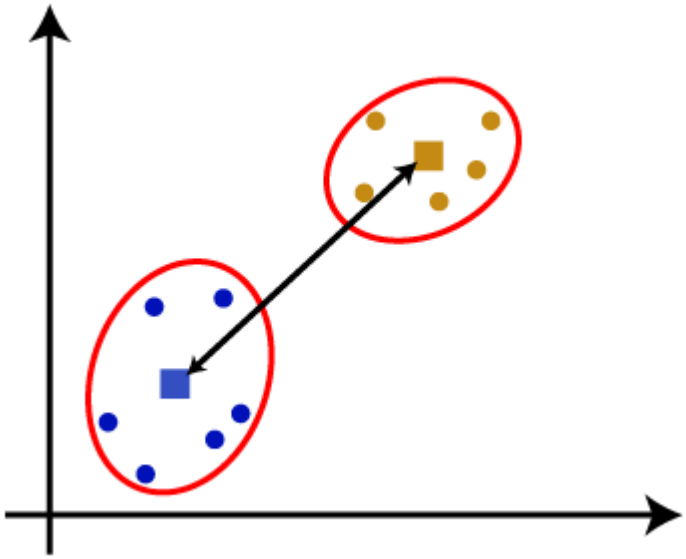


Complete Linkage: It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



Average Linkage: It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

Centroid Linkage: It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



Steps for implementation of AHC using Python:

The steps for implementation will be the same as the k-means clustering, except for some changes such as the method to find the number of clusters. Below are the steps:

- Data Pre-processing
    - Importing the libraries
    - Importing the dataset
    - Extracting the matrix of features
- Finding the optimal number of clusters using the Dendrogram
- Training the hierarchical clustering model
- Visualizing the clusters

Conclusion:

Thus, we have successfully implemented a program for Agglomerative hierarchical clustering algorithm.

Outcome:

Upon completion students will be able to:

Understand the Agglomerative hierarchical clustering algorithm.