**CHAPTER 1**

**INTRODUCTION**

Clustering is a division of data into groups of similar objects. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. In other words, the goal of a good image clustering scheme is to minimize intra-cluster distances between documents, while maximizing inter-cluster distances (using an appropriate distance measure between documents). A distance measure (or, dually, similarity measure) thus lies at the heart of image clustering. Clustering is the most common form of unsupervised learning and this is the major difference between clustering and classification. No super-vision means that there is no human expert who has assigned documents to classes. In clustering, it is the distribution and makeup of the data that will determine cluster membership. Clustering is sometimes erroneously referred to as automatic classification; however, this is inaccurate, since the clusters found are not known prior to processing whereas in case of classification the classes are pre-defined. In clustering, it is the distribution and the nature of data that will determine cluster membership, in opposition to the classification where the classifier learns the association between objects and classes from a so called training set, i.e. a set of data correctly labeled by hand, and then replicates the learnt behavior on unlabeled data.

Clustering is one of the most commonly used methods of data mining, and it has been widely used in feature extraction and data classification. Among all the clustering algorithms, hierarchical clustering is the most widely used one, which mainly has two kinds of aggregation and splitting [1]. The agglomerative hierarchical clustering has been a hot topic in the research of experts and scholars all the time, the socalled cohesion hierarchical clustering is a process of using bottom-up strategy to merge clusters layer by layer, and finally forming the hierarchical clustering tree. In order to get a good clustering effect, the first problem that we have to solve is the distance measurement. Traditional hierarchical clustering algorithms are only dependent on the Euclidean distance, without considering the influence of the distribution of data points around the object, and the different data structure on the clustering results. It is clear that clustering accuracy is often not good. In order to solve the above problems, many experts and scholars have carried on this research for many years. Literature [2] defines the concept of the coherence within a cluster and the separation between clusters respectively, used to describe the similarity within a cluster or between clusters. But the similarity measurement depends only on the number of shared neighbors, does not apply to the two objects that belong to the same cluster but slightly far distance. Literature [3] combines the ant colony optimization algorithm with condensed hierarchical clustering algorithm, proposes a hybrid clustering method HCAA, using the intelligent ant colony optimization algorithm for the global optimal. But it only uses Euclidean distance as the similarity measurement, so the calculation accuracy is low. Literature [4] proposes a hierarchical clustering algorithm (ABHCURE) with the approximate binary of representative points, solves the influence of stray data points on the result of clustering and the problem that the number of clustering is difficult to determine. But it used Euclidean distance to measure similarity between two data objects, without considering the environment and structure characteristics of the data objects themselves on the effect of similarity calculation. The next issue that needs to be addressed is how to select the appropriate clusters to merge. Literature [2] solves the problems of clusters merging, but it chooses the two clusters with minimum separability to merge, this may lead to a higher number of iterations in large amounts of data. Literature [5] uses single connection, all connections, average connections algorithm to merge clusters. For example,

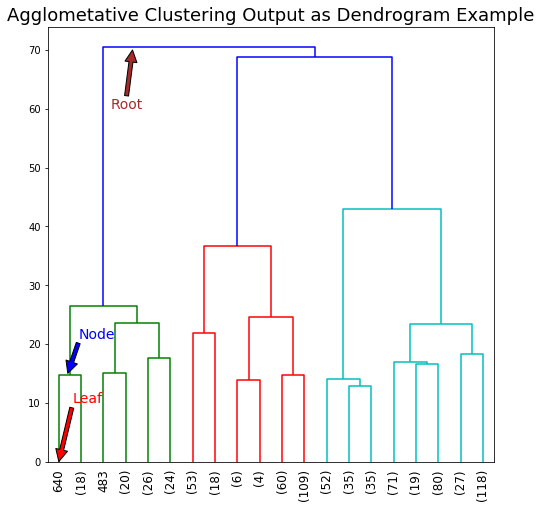
the single connection algorithm provides that the minimum distance between any two elements in two clusters reaches a threshold value for the purpose of merging clusters, but this lacks of consideration of the whole structure of clusters. It can be seen that agglomerative hierarchical clustering algorithm and the some improved algorithms have more or less deficiencies in the distance metric and clusters merging, and cant solve the existing problems well. In this paper, we propose an agglomerative hierarchical clustering algorithm based on global distance measurement. Firstly, it proposes a global distance measurement method for global optimal; secondly, it defines the internal coherence and external separability according to the obtained global distance, merging the clusters reasonably. We conduct experiments on the artificial datasets and real datasets. The results show that the proposed algorithm overcomes the difficulties of the Euclidean distance in mining data hidden information and has higher accuracy.

**1.1 AGGLOMERATION CLUSTERING PROCESS**

In machine learning, unsupervised learning is a machine learning model that infers the data pattern without any guidance or label. Many models are included in the unsupervised learning family, but one of my favorite models is Agglomerative Clustering.

Agglomerative Clustering or bottom-up clustering essentially started from an individual cluster (each data point is considered as an individual cluster, also called (leaf), then every cluster calculates their distancewith each other. The two clusters with the shortest distance with each other would mergecreating what we called node. Newly formed clusters once again calculating the member of their cluster distance with another cluster outside of their cluster. The process is repeated until all the data points assigned to one cluster called root. The result is a tree-based representation of the objects called dendrogram.

Although, we are presented with the result of how the data should be clustered; Agglomerative Clustering does not present any exact number of how our data should be clustered. It is up to us to decide where is the cut-off point.



**1.2 PROBLEM STATEMENT**

the raw datasets from the maps need to be analysis to that the geo graphical patterns can be clustered using agglomerative clustering process this can be helpful to identify the geographical differences based on the on the images by separating the clusters on images based on its value , this will help to classify the clusters only to a certain range , which can help in identifying the geographical features of an satellite geographical datasets.

In Data Analytics we often have very large data (many observations - “rows in a flat file”), which are however similar to each other hence we may want to organize them in a few clusters with similar observations within each cluster. For example, in the case of customer data, even though we may have data from millions of customers, these customers may only belong to a few segments: customers are similar within each segment but different across segments. We may often want to analyse each segment separately, as they may behave differently (e.g. different market segments may have different product preferences and behavioural patterns).

In such situations, to identify segments in the data one can use statistical techniques

broadly called Clusteringtechniques. Based on how we define “similarities” and “differences” between data observations (e.g. customers or assets), which can also be defined mathematically using distance metrics, one can find different segmentation solutions. A key ingredient of clustering and segmentation is exactly the definition of these distance metrics (between observations), which need to be defined creatively based on contextual knowledge and not only using “black box” mathematical equations and techniques.

**CHAPTER 2**

**LITERATURE REVIEW**

A literature review is a scholarly paper, which includes the current knowledge including the substantive findings, as well as theoretical and methodological contribution to a particular topic. Its basis for research in nearly every academic field and it may also be part of graduate and post-graduate student works, including the preparation of thesis, dissertation, or a journal article.

Makrehchi,[1] M Hierarchical clustering has been well-studied in the community of machine learning. Hierarchical clustering algorithms are deterministic, stable, and do not need a pre-determined number of clusters as input. However, they are not scalable for very large data due to their non-linear complexity. In this paper, a new approach is proposed to reduce the complexity of Hierarchical Clustering, improve the purity of the clustering algorithm, and reduce the chaining factor. The proposed method has the following components: (i) A new combination similarity based on common-neighbours of graph theory is proposed, (ii) In every iteration, instead of calculating the centroids for new clusters, new centroids are estimated from centroids in previous iteration, and (iii) In each iteration, instead of merging only one pair of objects, multiple pairs are merged at the same time. In addition to the proposed combination similarity, four well-known methods including centroid-based, group-based, complete-link, and single-link, have been also implemented. All five methods are tested and evaluated using two metrics: purity and imbalance or chaining factor. We show that our proposed algorithm outperforms other classic methods.

[Zahra Nazari](https://ieeexplore.ieee.org/author/37085727700),[2]The purpose of data clustering algorithm is to form clusters (groups) of data points such that there is high intra-cluster and low inter-cluster similarity. There are different types of clustering methods such as hierarchical, partitioning, grid and density based. Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. A hierarchical clustering method can be thought of as a set of ordinary (flat) clustering methods organized in a tree structure. These methods construct the clusters by recursively partitioning the objects in either a top-down or bottom-up fashion. In this paper we present a new hierarchical clustering algorithm using Euclidean distance. To validate this method we have performed some experiments with low dimensional artificial datasets and high dimensional fMRI dataset. Finally the result of our method is compared to some of existing clustering methods.

[Thuy-Diem Nguyen](https://ieeexplore.ieee.org/author/37085651075),[3] Cluster analysis is an important data mining technique widely used for pattern recognition and information retrieval. In the literature, over a hundred clustering algorithms have been developed to target input datasets with different characteristics. Among these algorithms, the hierarchical clustering method is particularly useful for analyzing genetic datasets in evolutionary biology studies because of the inherent hierarchical relationships amongst the genetic sequences extracted from related organisms. However, this algorithm is computational expensive in terms of both execution time and particularly memory usage. This paper summarizes our experience in using parallel computing technologies with new algorithms to perform hierarchical sequence clustering in a more effective way without compromising the accuracy of the results.

R.J. Gil Gaacia **,**[4]This paper presents a general framework for agglomerative hierarchical clustering based on graphs. Different hierarchical agglomerative clustering algorithms can be obtained from this framework, by specifying an inter-cluster similarity measure, a subgraph of the 13-similarity graph, and a cover routine. We also describe two methods obtained from this framework called hierarchical compact algorithm and hierarchical star algorithm. These algorithms have been evaluated using standard document collections. The experimental results show that our methods are faster and obtain smaller hierarchies than traditional hierarchical algorithms while achieving a similar clustering quality.

Sakshi patel, [5] Clustering algorithm plays a vital role in organizing large amount of information into small number of clusters which provides some meaningful information. Clustering is a process of categorizing set of objects into groups called clusters. Hierarchical clustering is a method of cluster analysis which is used to build hierarchy of clusters. This paper focuses on hierarchical agglomerative clustering. In this paper, we also explain some agglomerative algorithms and their comparison.

Masoud Makrehchi, [6] Hierarchical clustering has been well-studied in the community of machine learning. Hierarchical clustering algorithms are deterministic, stable, and do not need a pre-determined number of clusters as input. However, they are not scalable for very large data due to their non-linear complexity. In this paper, a new approach is proposed to reduce the complexity of Hierarchical Clustering, improve the purity of the clustering algorithm, and reduce the chaining factor. The proposed method has the following components: (i) A new combination similarity based on common-neighbours of graph theory is proposed, (ii) In every iteration, instead of calculating the centroids for new clusters, new centroids are estimated from centroids in previous iteration, and (iii) In each iteration, instead of merging only one pair of objects, multiple pairs are merged at the same time. In addition to the proposed combination similarity, four well-known methods including centroid-based, group-based, complete-link, and single-link, have been also implemented. All five methods are tested and evaluated using two metrics: purity and imbalance or chaining factor. We show that our proposed algorithm outperforms other classic methods.

Zahra Nazari, [7] The purpose of data clustering algorithm is to form clusters (groups) of data points such that there is high intra-cluster and low inter-cluster similarity. There are different types of clustering methods such as hierarchical, partitioning, grid and density based. Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. A hierarchical clustering method can be thought of as a set of ordinary (flat) clustering methods organized in a tree structure. These methods construct the clusters by recursively partitioning the objects in either a top-down or bottom-up fashion. In this paper we present a new hierarchical clustering algorithm using Euclidean distance. To validate this method we have performed some experiments with low dimensional artificial datasets and high dimensional fMRI dataset. Finally the result of our method is compared to some of existing clustering methods.

Thuy-Diem-Nguyen, [8] Cluster analysis is an important data mining technique widely used for pattern recognition and information retrieval. In the literature, over a hundred clustering algorithms have been developed to target input datasets with different characteristics. Among these algorithms, the hierarchical clustering method is particularly useful for analysing genetic datasets in evolutionary biology studies because of the inherent hierarchical relationships amongst the genetic sequences extracted from related organisms. However, this algorithm is computational expensive in terms of both execution time and particularly memory usage. This paper summarizes our experience in using parallel computing technologies with new algorithms to perform hierarchical sequence clustering in a more effective way without compromising the accuracy of the results.

R.J.Gil-Garcia, [9] This paper presents a general framework for agglomerative hierarchical clustering based on graphs. Different hierarchical agglomerative clustering algorithms can be obtained from this framework, by specifying an inter-cluster similarity measure, a subgraph of the 13-similarity graph, and a cover routine. We also describe two methods obtained from this framework called hierarchical compact algorithm and hierarchical star algorithm. These algorithms have been evaluated using standard document collections. The experimental results show that our methods are faster and obtain smaller hierarchies than traditional hierarchical algorithms while achieving a similar clustering quality

Hamidreza mirzari, [10] In this paper, we propose a new algorithm for extending the hierarchical clustering methods and introduce a Multi-View Agglomerative Clustering approach to handle multi-view represented objects. Experiments on real world datasets indicate that our algorithm considering the relationship among multiple views can provide a solution with improved quality in multi-view setting. We find empirically that the multi-view version of our Agglomerative Clustering, independent of linkage method and given any number of views, greatly improves on its single-view counterparts.

**2.1 PROPOSED OBJECTIVE**

Lack of a Global Objective Function: agglomerative hierarchical clustering techniques perform clustering on a local level and as such there is no global objective function like in the K-Means algorithm. This is actually an advantage of this technique because the time

and space complexity of global functions tends to be very expensive.

Ability to Handle Different cluster Sizes:

we have to decide how to treat clusters of various sizes that are merged together.

Merging Decisions Are Final:

one downside of this technique is that once two clusters have been merged they cannot be split up at a later time for a more favourable union.

**CHAPTER 3**

**SYSTEM REQUIREMENT**

The system requirements specify features, components and behaviour of system which is to be developed. The following sections describe about functional, non-functional, performance related, features and behaviour of the solution. This includes the detailed description of the solution to be developed.

**3.1 Functional Requirement**

In software engineering, a functional requirement defines a system or its component. It describes the functions software must perform. A function is nothing but inputs, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

Functional Requirements are those requirements which show the working and functionality of a system and the expected behaviour of a system based on certain situations and inputs. It defines specific functionality of a system.

Functional requirements of system are:

* A new decentralized supply chain that detects counterfeiting attacks using blockchain and Near Field Communication (NFC) technologies.
* Block-supply chain replaces the centralized supply chain design and utilizes a new proposed consensus protocol.
* Manufacture is able to prove their product is authentic and is also able to track their product ‘s pathway.
* The setup is easy to implement and requires less operation cost.

**3.2 Non-functional Requirement**

* **Reliability** The framework ought to be dependable and solid in giving the functionalities. When a client has rolled out a few improvements, the progressions must be made unmistakable by the framework. The progressions made by the Programmer ought to be unmistakable both to the Project pioneer and in addition the Test designer.
* **Security** Aside from bug following the framework must give important security and must secure the entire procedure from smashing. As innovation started to develop in quick rate the security turned into the significant concern of an association. A great many dollars are put resources into giving security. Bug following conveys the greatest security accessible at the most noteworthy execution rate conceivable, guaranteeing that unapproved clients can't get to imperative issue data without consent. Bug following framework issues diverse validated clients their mystery passwords so there are limited functionalities for all the clients.
* **Maintainability** The framework observing and upkeep ought to be basic and target in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.
* **Performance** The framework will be utilized by numerous representatives all the while. Since the framework will be facilitated on a solitary web server with a solitary database server out of sight, execution turns into a noteworthy concern. The framework ought not to succumb when numerous clients would be utilizing it all the while. It ought to permit quick availability to every last bit of its clients. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.
* **Portability** The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.
* **Scalability** The framework should be sufficiently adaptable to include new functionalities at a later stage. There ought to be a typical channel, which can oblige the new functionalities.
* **Flexibility** is the capacity of a framework to adjust to changing situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change.

**3.3 Hardware and Software Requirement**

**Hardware Requirements**

Processor : i3 2.5mhz or above

RAM : 8GB DDR4

Hard Disk : 20 GB

Graphics : 4gb or above

**Software Requirements**

Coding language : Python

IDE Tool used: jupyter notebook, colab

Operating System : Windows 10

**3.4 Basic Operational Requirement**

Operational requirements are those statements that “identify the essential capabilities, associated requirements, performance measures, and the process or series of actions to be taken in effecting the results that are desired in order to address mission area deficiencies, evolving applications or threats, emerging technologies, or system cost improvements.” The operational requirements assessment starts with the Concept of Operations and goes to a greater level of details in identifying mission performance assumptions and constrains and current deficiencies of or enhancements needed for operations and mission success. Operational requirements are the basis for system requirements. Blockchain systems engineers (SEs) are expected to be able to understand the user’s needs based on the operational needs‟ assessment (i.e., what mission area capability gaps need to be addressed). They must be able to analyze the needs identified by the capability gaps and develop or assist in defining the operational and top-level characteristics or requirements of the system. They also should use the concept of operation (CONOPS) to understand the operational needs, desires, vision expectations, performance requirements, and challenges of the system. Proposed system is expected to be able to lay out an evolutionary strategy for the requirements that identify and prioritize initial capabilities and subsequent capability increments to be implemented over time. This approach allows for rapid delivery of initial capabilities and enables agility in delivering future capabilities that are responsive to changes in the operational environment.

**CHAPTER 4**

**SYSTEM ANALYSIS**

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Analysts in the field of engineering look at requirements, structures, mechanisms, and systems dimensions. Analysis is an exploratory activity. The Analysis Phase is where the project lifecycle begins. The Analysis Phase is where you break down the deliverables in the high level Project Charter into the more detailed business requirements. The Analysis Phase is also the part of the project where you identify the overall direction that the project will take through the creation of the project strategy documents.

**Feasibility Study**

The achievability of the scheme is scrutinized throughout this part and commerce tender is place onward with a terribly general organize for the project and a few prices approximate. All through system psychotherapy the viability study of the predictable system is to be prearranged this is habitually to make sure that the anticipated system isn't a lumber to the commercial. For viability analysis, some indulgent of the chief provisions for the structure is imperative.

Three key anxieties anxious within the achievability analysis are:

* Economic Feasibility
* Technical Feasibility
* Social Feasibility

**Economic Feasibility**

This study is mete out to foresee the financially viable crash that the scheme can show off the union the magnitude of endowment that the communal will decant into the psychoanalysis and enlargement of the scheme is prohibited. The expenditures ought to be even consequently the residential system in totalling inside the financial statement and this was pull off accordingly of the largest part of the equipment’s used are unreservedly on the souk solely the specially made commodities had to be procure.

**Technical Feasibility**

This cram is functional to see the procedural feasibleness, to be precise, the nominal chucks of the structure. Any organism urbanized should not have a lofty stipulate on the accessible procedural where withal. this can ground high strains on the accessible procedural possessions this can cause lofty strains being sited on the punter. The urbanized organism should have an unpretentious stipulate, as exclusively ostensible or unsound amend are looked for executing this routine.

**Social Feasibility**

The visage of cram is to make certain the coverage of receiving of the scheme by the abuser. This embrace the routine of schooling the abuser to use the organism with competence. The abuser should not undergo defenceless by the organism, as an alternative should reconcile for it as a stipulation the coverage of approval by the abusers unaided depends on the traditions that are used to instruct the abuser concerning the organism and to figure him au fait with it. His echelon of assurance should be elevated so he's conjointly proficient to craft some productive denigration, that is salutation, as he's the definitive abuser of the system.

**Proposed System**

The proposed system use’s agglomerate clustering algorithm by Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

1. **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
2. **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

**Summary**

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram.

The main purpose of this project is to get an in depth understanding of how the Divisive and Agglomerative hierarchical clustering algorithms work.

**CHAPTER 5**

**SYSTEM DESIGN**

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modelling software systems and is increasingly used for high designing non-software systems and organizations.

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. The design will contain the specification of all these modules, their interaction with other modules and the desired output from each module. The output of the design process is a description of the software architecture.

The design phase is followed by two sub phases

* High Level Design
* Detailed Level Design

**5.1 Fundamental Design Concepts**

Fundamental design is developed in course of recent years. As year’s passes, enthusiasm of creating new designs is evolved and each design has been tested. Software designer gets new ideas and foundation to build and test new design concepts. Fundamental framework is design to "getting it right". A major plan idea, for example, deliberation, and refinement, modularity, and programming engineering and data encryption is applied to meet the requirement of proposed work.

**5.1.1 High Level Design**

In the high level design, the proposed functional and non functional requirements of the software are depicted. Overall solution to the architecture is developed which can handle those needs. This chapter involves the following consideration.

* System Architecture

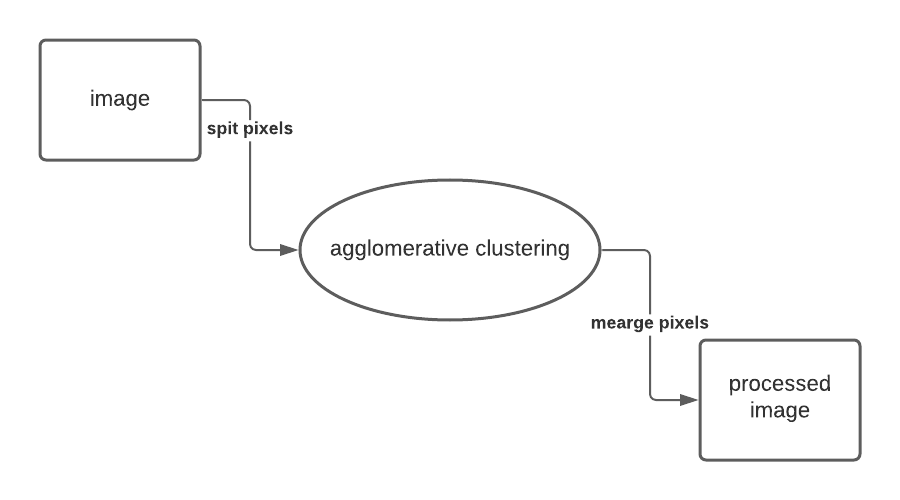
**5.1.2 Low Level Design**

During the detailed phase, the view of the application developed during the high level design is broken down into modules and programs. Logic design is done for every program and then documented as program specifications. For every program, a unit test plan is created.

* Data flow Diagram.
* The information stream outline demonstrates the graphical portrayal, similar to game  
  plans it is utilized to speak to the information through the sources of info, different sorts of information examination will be completed and the coveted yield will be produced.
* These parts will be utilized to demonstrate the framework and it will be displayed by to contemplate quickly regarding the information. In the framework outline the DFD will demonstrate the stream of whole parts. The stream of data will in arrangement of change utilizing this framework.
* During the detailed phase, the view of the application developed during the high-level design is broken down into modules and programs. Logic design is done for every program and then documented as program specifications. Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

**5.1.2.1 Data Flow Diagram- Level 0**

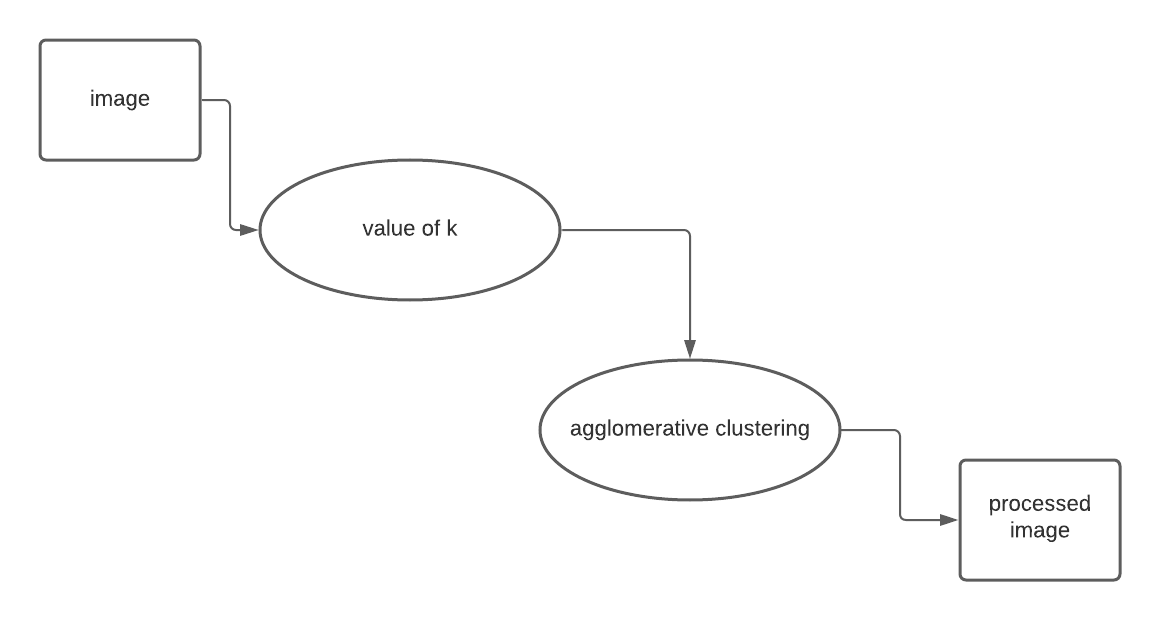
* It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities.
* It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

**Figure 5.1: Data Flow diagram Level-0**

* In dataflow diagram level zero (Figure 5.1) we can observe first we resize an image and pre process the image , this include the calculation of pixels and merge the similar clusters.
* Then these clusters will be merged and the clusters will be plotted on to a 2d graph

**5.1.2.2 Data Flow Diagram- Level 1**

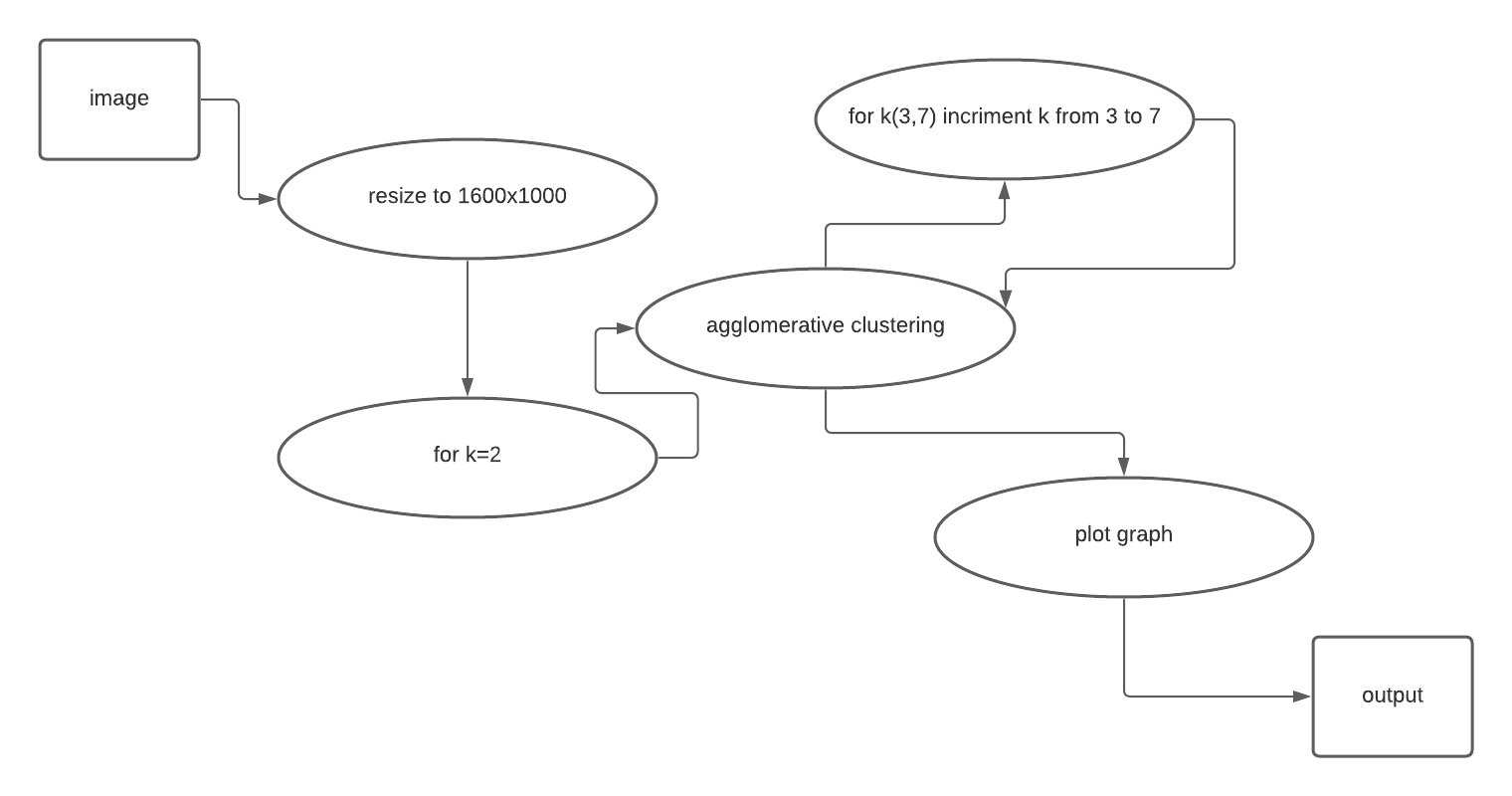
* In 1-level DFD (Figure 5.2), the context diagram is decomposed into multiple bubbles/processes.
* In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub processes.

**Figure 5.2: Data Flow diagram Level-1**

* The image will set with different values of the ‘k’, which specifies the intensity of the pixels in RGB scale before the processing using agglomerative clustering.
* Then the image will be pre process the image , this include the calculation of pixels and merge the similar clusters.

**5.1.2.3 Data Flow Diagram- Level 2**

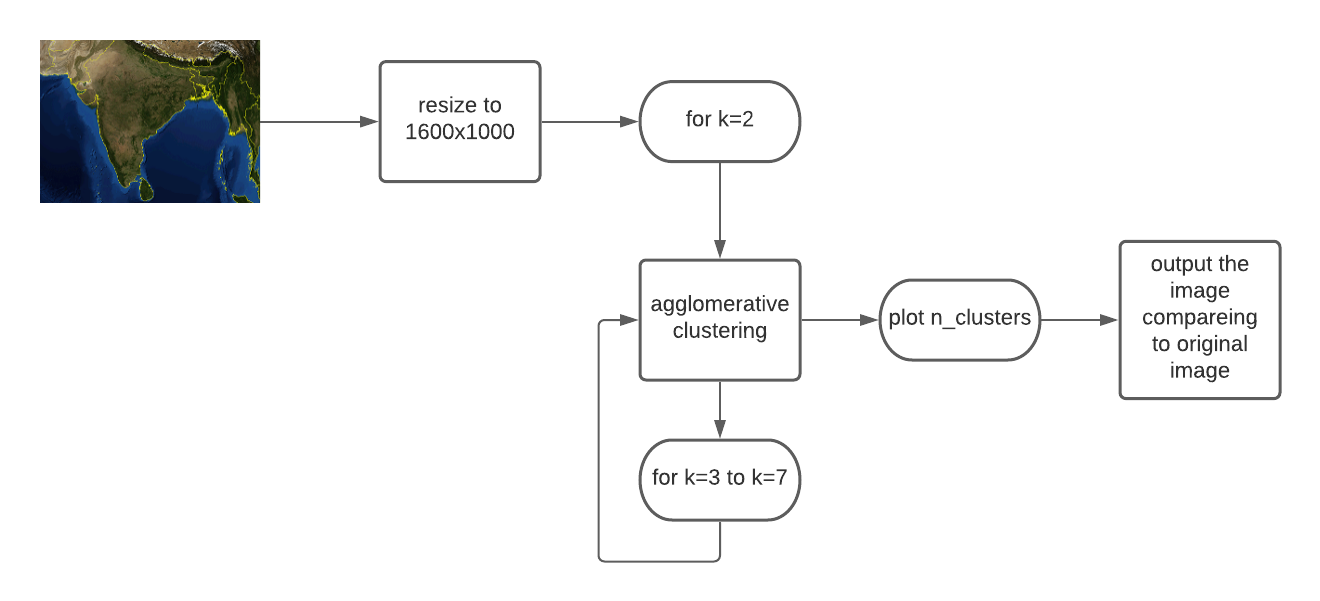
* 2-level DFD (Figure 5.3) goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning.
* The entry criteria for this will be the High Level Document (HLD). And the exit criteria will the program specification and unit test plan i.e. Low Level Document (LLD).

**Figure 5.3: Data Flow Diagram Level-2**

* First the imaged will be resized to be compatible with our code , our code is built for 1600x1000 dimensions , means the image what we process shell have a specific number of pixel’s which is 1600000 pixels in an image
* Then the cluster’s RGB intensity shell be assumed for the instal value of k , which shell be 2 in our case, and then perform the agglomerative clustering algorithm on it.
* Then the value of the k will be incremented to 3 to 7 which has good change of differentiating the clusters of different colours and margining them , the graph will be plotted based on each value of k and this shell be the resulted output.

**5.2 Software Architecture**

The below figure 5.4 shows a general block diagram describing the activities performed by this project. Before you build a computer, you should conceptualize the purpose and the structure. It’s a conceptual method that defines the structure, behaviour, and more views of a system.

**Figure 5.4 software process**

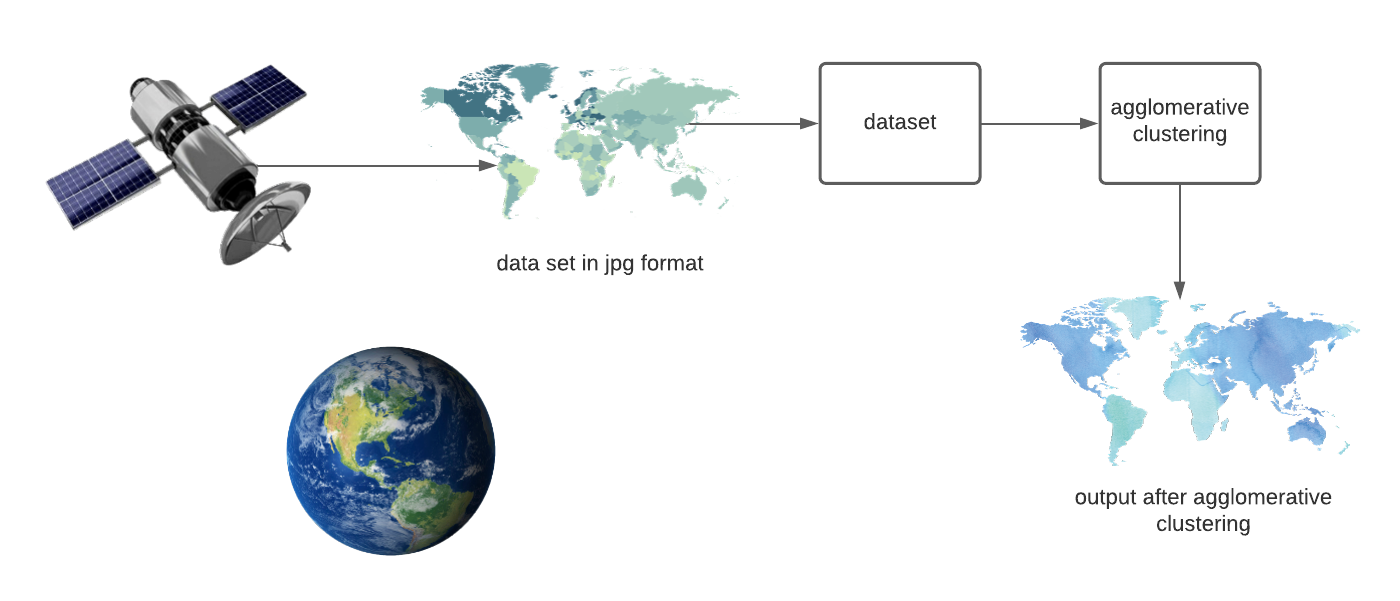
The image dataset , which is mostly a geographical map of area ,or a satellite taken image , this image will be resized to the size compact to our code , which is 1600x1000= 1600000 pixels in the image , these pixels will be assume’s some clusters which will be ‘k’ this value will be assumed for k=2 for the primary clustering .

Then the k will be incriminated to 3 and will be incremented till 7 , this can be increased to more number depending of the complexity of the colours present in the image , and the clusters will be plotted on to the 2d graph, and same process will be repeated still the final value of ‘k’.

**5.3 Use Case Diagram:**

A utilization case in programming designing and frameworks building is a portrayal of a framework's conduct as it reacts to a demand that starts from outside of that framework. As it were, a utilization case depicts "who" can do "what" with the framework being referred to. The utilization case method is utilized to catch a framework's behavioural necessities by specifying situation driven strings through the useful prerequisites.

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

**Figure 5.5: Use case Diagram of Authentication Module**

the raw data sets from the GPS or any image (JPG) will be considered as the dataset for our code , this data set will be given to our code which will calculate the clusters based on the similar pixels and assume any two clusters randomly , then calculate the distance between them and predict the centre , the clusters will be written to n\_ clusters and plots the graph based on the parameters.

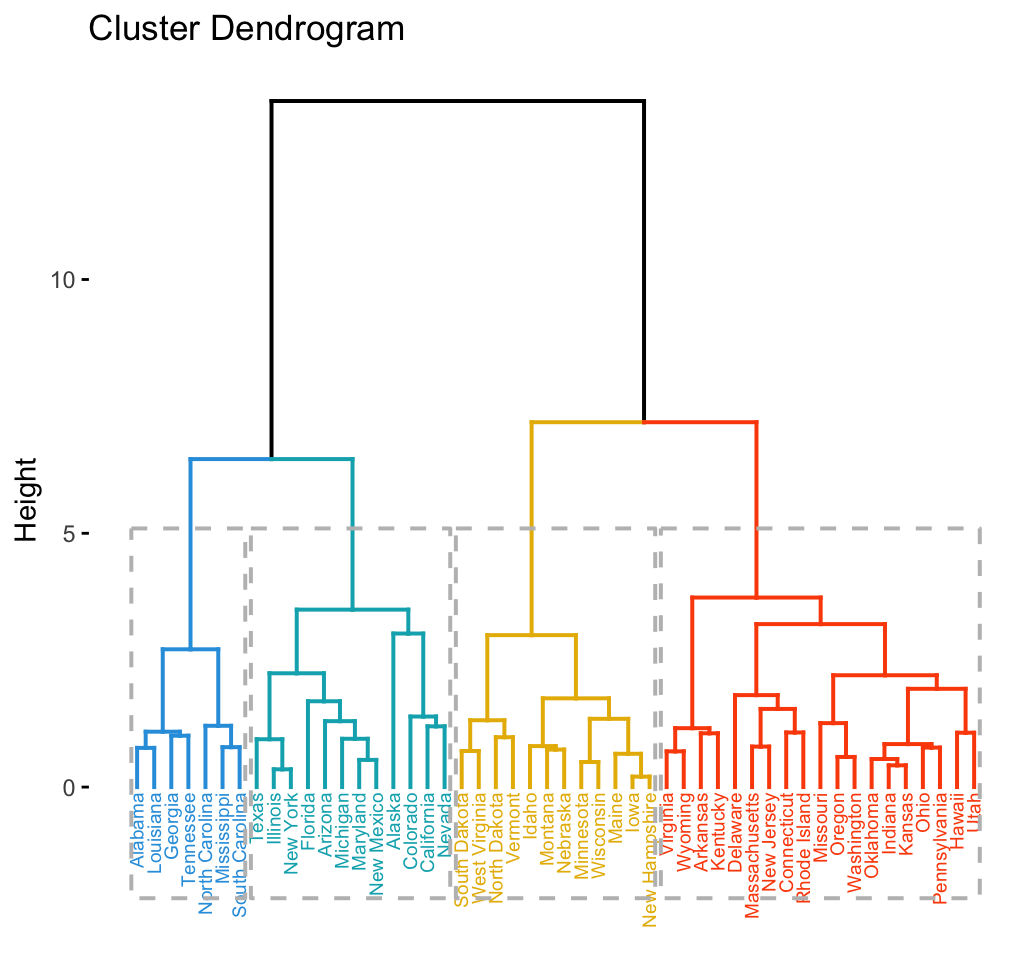
**5.4 Sequence Diagram**

A sequence diagram (Figure 5.7) shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances.

Messages, written with horizontal [arrows](https://en.wikipedia.org/wiki/Arrow_(symbol)) with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent [asynchronous messages](http://www.uml-diagrams.org/sequence-diagrams.html), and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn’t have to wait for a response. Asynchronous calls are present in multithreaded applications, event-driven applications and in [message-oriented middleware](https://en.wikipedia.org/wiki/Message-oriented_middleware). Activation boxes, or [method](https://en.wikipedia.org/wiki/Method_(computer_science))-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language)).

**Figure 5.7: Sequence diagram of agglomerative clustering**

We can see that the clustering pattern for complete linkage distance tends to create compact clusters of clusters, while single linkage tends to add one point at a time to the cluster, creating long stringy clusters. As we might expect from our discussion of distances, Euclidean distance and correlation distance produce very different dendrograms.

Hierarchical clustering does not tell us how many clusters there are, or where to cut the dendrogram to form clusters. In R there is a function cluster which will cut a tree into clusters at a specified height. However, based on our visualization, we might prefer to cut the long branches at different heights. In any case, there is a fair amount of subjectivity in determining which branches should and should not be cut to form separate clusters.

**CHAPTER 6**

**IMPLEMENTATION**

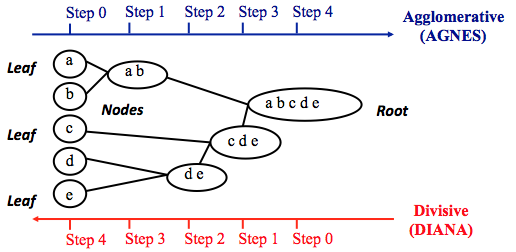
Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. In other words, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment.

Implementation is one of the most important phases of the Software Development Life Cycle (SDLC). It encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. Specifically, it involves coding the system using a particular programming language and transferring the design into an actual working system.

## **Agglomerative Hierarchical Clustering**

Agglomerative clustering works in a “bottom-up” manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root) (see figure below).

The inverse of agglomerative clustering is *divisive clustering*, which is also known as DIANA (*Divise Analysis*) and it works in a “top-down” manner. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster (see figure below).

**Fig 6.1** Agglomerative clustering

## **Steps to agglomerative hierarchical clustering**

We’ll follow the steps below to perform agglomerative hierarchical clustering using R software:

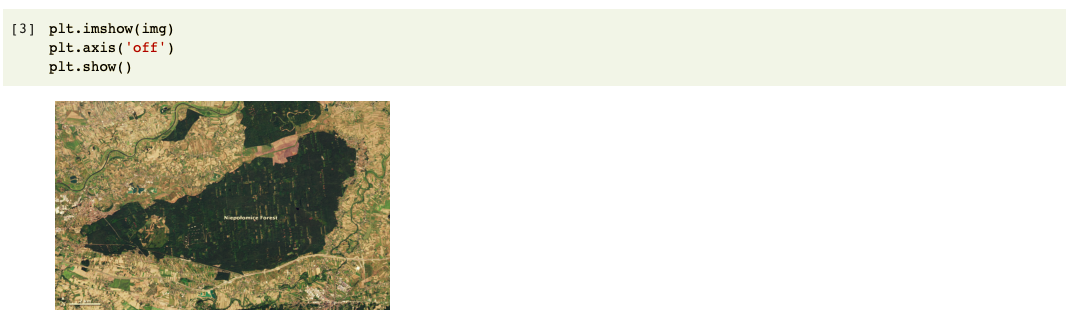
1. Preparing the data
2. Computing (dis)similarity information between every pair of objects in the data set.
3. Using linkage function to group objects into hierarchical cluster tree, based on the distance information generated at step 1. Objects/clusters that are in close proximity are linked together using the linkage function.
4. Determining where to cut the hierarchical tree into clusters. This creates a partition of the data.

### **Data structure and preparation**

The data should be a numeric matrix with:

* rows representing observations (individuals);
* and columns representing variables.

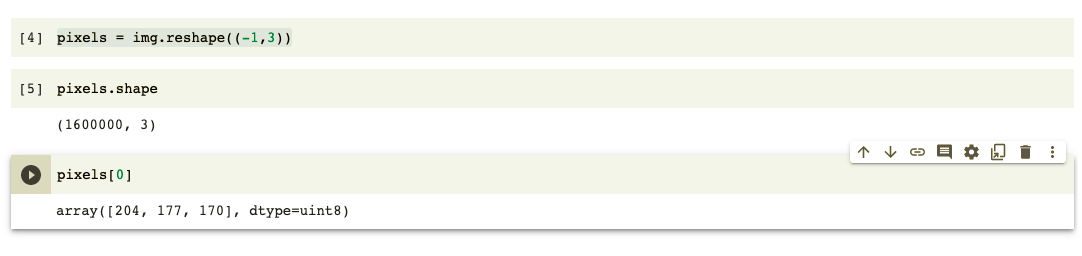
Here, we’ll use the image (jpg ) data sets

****

**Resizing the dataset**

The reshape() function is used to give a new shape to an array without changing its data.

Read the elements of a using this index order, and place the elements into the reshaped array using this index order. ‘C’ means to read / write the elements using C-like index order, with the last axis index changing fastest, back to the first axis index changing slowest. ‘F’ means to read / write the elements using Fortran-like index order, with the first index changing fastest, and the last index changing slowest. Note that the ‘C’ and ‘F’ options take no account of the memory layout of the underlying array, and only refer to the order of indexing. ‘A’ means to read / write the elements in Fortran-like index order if a is Fortran contiguous in memory, C-like order otherwise.



## **Verify the cluster tree**

After linking the objects in a data set into a hierarchical cluster tree, you might want to assess that the distances (i.e., heights) in the tree reflect the original distances accurately.

One way to measure how well the cluster tree generated by the function reflects your data is to compute the correlation between the *cophenetic* distances and the original distance data generated by the function. If the clustering is valid, the linking of objects in the cluster tree should have a strong correlation with the distances between objects in the original distance matrix.

The closer the value of the correlation coefficient is to 1, the more accurately the clustering solution reflects your data. Values above 0.75 are felt to be good. The “average” linkage method appears to produce high values of this statistic. This may be one reason that it is so popular.

The R base function can be used to compute the co-phenetic distances for hierarchical clustering.

## **Summary**

Hierarchical clustering is a cluster analysis method, which produce a tree-based representation (i.e.: dendrogram) of a data. Objects in the dendrogram are linked together based on their similarity.

To perform hierarchical cluster analysis in R, the first step is to calculate the pairwise distance matrix using the function. Next, the result of this computation is used by the function to produce the hierarchical tree. Finally, you can use the function [in factoextra R package] to plot easily a beautiful dendrogram.

**6.4 Codeing**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import random

from copy import deepcopy

path\_to\_jpg\_file = "/content/sample\_data/sample.jpg"

img = mpimg.imread(path\_to\_jpg\_file)

img.shape

plt.imshow(img)

plt.axis('off')

plt.show()

pixels = img.reshape((-1,3))

pixels.shape

pixels[0]

def euclidean\_distance(point1, point2):

"""

Computes euclidean distance of point1 and point2.

point1 and point2 are lists.

"""

return np.linalg.norm(np.array(point1) - np.array(point2))

def clusters\_distance(cluster1, cluster2):

"""

Computes distance between two clusters.

cluster1 and cluster2 are lists of lists of points

"""

return max([euclidean\_distance(point1, point2) for point1 in cluster1 for point2 in cluster2])

def clusters\_distance\_2(cluster1, cluster2):

"""

Computes distance between two centroids of the two clusters

cluster1 and cluster2 are lists of lists of points

"""

cluster1\_center = np.average(cluster1, axis=0)

cluster2\_center = np.average(cluster2, axis=0)

return euclidean\_distance(cluster1\_center, cluster2\_center)

class AgglomerativeClustering:

def \_\_init\_\_(self, k=2, initial\_k=25):

self.k = k

self.initial\_k = initial\_k

def initial\_clusters(self, points):

"""

partition pixels into self.initial\_k groups based on color similarity

"""

groups = {}

d = int(256 / (self.initial\_k))

for i in range(self.initial\_k):

j = i \* d

groups[(j, j, j)] = []

for i, p in enumerate(points):

if i%100000 == 0:

print('processing pixel:', i)

go = min(groups.keys(), key=lambda c: euclidean\_distance(p, c))

groups[go].append(p)

return [g for g in groups.values() if len(g) > 0]

def fit(self, points):

# initially, assign each point to a distinct cluster

print('Computing initial clusters ...')

self.clusters\_list = self.initial\_clusters(points)

print('number of initial clusters:', len(self.clusters\_list))

print('merging clusters ...')

while len(self.clusters\_list) > self.k:

# Find the closest (most similar) pair of clusters

cluster1, cluster2 = min([(c1, c2) for i, c1 in enumerate(self.clusters\_list) for c2 in self.clusters\_list[:i]],

key=lambda c: clusters\_distance\_2(c[0], c[1]))

# Remove the two clusters from the clusters list

self.clusters\_list = [c for c in self.clusters\_list if c != cluster1 and c != cluster2]

# Merge the two clusters

merged\_cluster = cluster1 + cluster2

# Add the merged cluster to the clusters list

self.clusters\_list.append(merged\_cluster)

print('number of clusters:', len(self.clusters\_list))

print('assigning cluster num to each point ...')

self.cluster = {}

for cl\_num, cl in enumerate(self.clusters\_list):

for point in cl:

self.cluster[tuple(point)] = cl\_num

print('Computing cluster centers ...')

self.centers = {}

for cl\_num, cl in enumerate(self.clusters\_list):

self.centers[cl\_num] = np.average(cl, axis=0)

def predict\_cluster(self, point):

"""

Find cluster number of point

"""

# assuming point belongs to clusters that were computed by fit functions

return self.cluster[tuple(point)]

def predict\_center(self, point):

"""

Find center of the cluster that point belongs to

"""

point\_cluster\_num = self.predict\_cluster(point)

center = self.centers[point\_cluster\_num]

return center

n\_clusters = 2

agglo = AgglomerativeClustering(k=n\_clusters, initial\_k=25)

agglo.fit(pixels)

new\_img = [[agglo.predict\_center(list(pixel)) for pixel in row] for row in img]

new\_img = np.array(new\_img, np.uint8)

plt.figure(figsize=(15,15))

plt.subplot(1,2,1)

plt.imshow(img)

plt.axis('off')

plt.title('Original image')

plt.subplot(1,2,2)

plt.imshow(new\_img)

plt.axis('off')

plt.title(f'Segmented image with k={n\_clusters}')

plt.show()

def cluster\_and\_plot(n\_clusters):

agglo = AgglomerativeClustering(k=n\_clusters, initial\_k=25)

agglo.fit(pixels)

new\_img = [[agglo.predict\_center(pixel) for pixel in row] for row in img]

new\_img = np.array(new\_img, np.uint8)

plt.figure(figsize=(15,15))

plt.subplot(1,2,1)

plt.imshow(img)

plt.axis('off')

plt.title('Original image')

plt.subplot(1,2,2)

plt.imshow(new\_img)

plt.axis('off')

plt.title(f'Segmented image with k={n\_clusters}')

plt.show()

n\_clusters = range(3,7)

for k in n\_clusters:

print(f'Processing k={k}:')

cluster\_and\_plot(k)

**CHAPTER 7**

**TESTING**

Testing is an important phase in the development life cycle of the product. This is the phase, where the remaining errors, if any, from all the phases are detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program was performing as expected. Errors were found and corrected by using the below stated testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system, before it was ready for implementation.

It is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stake holders, i.e., intended to reveal the quality-related information about the product with respect to context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors.

The quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; Testing furnishes a ‘criticism’ or comparison that compares the state and behaviour of the product against specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing. There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation not merely a matter of creating and following routine procedure.

Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product-putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one, which reveals an error; however, more recent thinking suggest that a good test is one which reveals information of interest to someone who matters within the project community.

**7.1 Types of Testing**

* **Unit Testing**

Individual component are tested to ensure that they operate correctly. Each component is tested independently, without other system component. This system was tested with the set of proper test data for each module and the results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module. This is also known as MODULE TESTING. This testing is carried out during phases, each module is found to be working satisfactory as regards to the expected output from the module.

* **Integration Testing**

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with flow of data across interfaces. The unit-tested modules are grouped together and tested in small segment, which make it easier to isolate and correct errors. This approach is continued unit I have integrated all modules to form the system as a whole.

* **System Testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration testing. System testing is based on process description and flows, emphasizing pre-driver process and integration points

* **Performance Testing**

The performance testing ensures that the output being produced within the time limits and time taken for the system compiling, giving response to the users and request being send to the system in order to retrieve the results.

* **Validation Testing**

**The validation testing can be defined in many ways, but a simple definition is that. Validation succeeds when the software functions in a manner that can be reasonably expected by the end user.**

**Black Box testing**

Black box testing is done to find the following

* Incorrect or missing functions
* Interface errors
* Errors on external database access
* Performance error
* Initialization and termination error

**White Box Testing**

This allows the tests to

* Check whether all independent paths within a module have been exercised at least once
* Exercise all logical decisions on their false sides
* Execute all loops and their boundaries and within their boundaries
* Exercise the internal data structure to ensure their validity
* Ensure whether all possible validity checks and validity lookups have been provided to validate data entry.
* **Acceptance Testing**

This is the final stage of testing process before the system is accepted for operational use. The system is tested within the data supplied from the system procurer rather than simulated data.

**SNAPSHOTS**