AI DRAWING USING CLUSTERING ALGORITHM

INDEX

Sr. No.	Title	Page No.
1	**Acknowledgement**	**3**
2	**Synopsis**	**4**
2.1	Introduction	5
	Types of Learning	8
2.2	Technologies Used	16
3	**Main Report**	**17**
3.1	Objective & Scope of Project	19
3.2	Theoretical Background	20
3.3	Definition of Problems	29
3.4	System Analysis & Design	30
3.5	System Planning	49
4	**Methodology Adopted,Details of Hardware and Software**	**53**
5	**Life Cycle Of The Project**	**54**
5 .1	Block Diagram	54
5.2	Input and Output Design	55
	Code	58

5.3	Process Involved	72
5.4	Methodology used for testing	73
6	Future Scope	81
7	Conclusion	82
8	**Bibliography**	**83**

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards **the Information Technology Department of Sathaye College.**

After months of hard work, finally we are very happy to present our final year Project. The Project making was full of new experiences and learning and difficult one too. Though a difficult job it was made simpler by the timely guidance received, which helped us greatly in the completion of our project .But it wouldn't be right to do so without thanking to those who have helped us in converting our thought into reality. So we would like to take full advantage of this opportunity to thank each and every person who has helped us throughout the completion of our project.

We are obliged to our parents & family members who always support us greatly and encouraged us in each and every step. We give our special thanks and sincere gratitude towards the Principal Mrs KavitaRege, Head of Department **Prof.Arvind Khadye.**

I owe my sincere thanks to our Project guide **Mr. Dhanraj jadhav sir** for his constant support and encouragement without which the successful completion of this project would have been impossible. He has been instrumental for making us concentrate and focus our effort in this project.

We will be forever indebted to all friends who really encouraged us. And last but not the least the entire college staff. Finally we would like to thank each and every individual who was directly or indirectly contributing for this project.

-Thank you.

SYNOPSIS

AI Drawing Tool using Clustering Algorithm

The huge amounts of data generated by media sensors in health monitoring systems, by medical diagnosis that produce media (audio, video, image, and text) content, and from health service providers are too complex and voluminous to be processed and analyzed by traditional methods.

Data mining approaches offer the methodology and technology to transform these heterogeneous data into meaningful information for decision making. This paper studies data mining applications in healthcare. Mainly, we study *k*-means clustering algorithms on large datasets and present an enhancement to *k*-means clustering, which requires *k* or a lesser number of passes to a dataset.

The proposed algorithm, which we call *G*-means, utilizes a greedy approach to produce the preliminary centroids and then takes *k* or lesser passes over the dataset to adjust these center points.

The project implements the clustering algorithm, as an idea to utilize the transformation of datasets into points and replicate the same. **(Hierarchial Clustering Algo.)**

INTRODUCTION

Nowadays,

healthcare data are received from various healthcare service providers including sensory environment to provide better healthcare services. This data contains details about patients, medical tests, and treatment. The received data is very vast and complex; thus, it is difficult to quickly analyze the data in order to make important decision regarding patient health. Data mining technique is one of the important research areas in identifying meaningful information from huge datasets. In healthcare application, such as a heart monitoring system, it is an important method for efficiently detecting unknown and valuable information from huge heterogeneous health data. In a healthcare application, data mining techniques can be used to detect unknown diseases, causes of diseases, and identification of medical treatment methods. It also helps medical researchers in making efficient healthcare policies, constructing drug recommendation systems, and developing health profiles of individuals.

In Machine Learning, the types of **Learning** can broadly be classified into three types: **1. Supervised Learning, 2. Unsupervised Learning and 3. Semi-supervised Learning**. Algorithms belonging to the family of **Unsupervised Learning** have no variable to predict tied to the data. Instead of having an output, the data only has an input which would be multiple variables that describe the data. This is where clustering comes in.

Clustering is the task of grouping together a set of objects in a way that objects in the same cluster are more similar to each other than to objects in other clusters. Similarity is a metric that reflects the strength of relationship between two data objects. Clustering is mainly used for exploratory data mining.

It has manifold usage in many fields such as machine learning, pattern recognition, image analysis, information retrieval, bio-informatics, data compression, and computer graphics.

However, this post tries to unravel the inner workings of K-Means, a very popular clustering technique. There's also a very good DataCamp post on K-Means, which explains the types of clustering (hard and soft clustering), types of clustering methods (connectivity, centroid, distribution and density) with a case study. The algorithm will help you to tackle unlabeled datasets (i.e. the datasets that do not have any class-labels) and draw your own inferences from them with ease.

K-Means falls under the category of centroid-based clustering. A centroid is a data point (imaginary or real) at the center of a cluster. In centroid-based clustering, clusters are represented by a central vector or a centroid. This centroid might not necessarily be a member of the dataset. Centroid-based clustering is an iterative algorithm in which the notion of similarity is derived by how close a data point is to the centroid of the cluster.

K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

K-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
This results in a partitioning of the data space into Voronoi cells.

K-Means minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. Better Euclidean solutions can for example be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both *k-means* and *Gaussian mixture modeling*.

They both use cluster centers to model the data; however, *k*-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the *k*-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with *k*-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by *k*-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

TYPES OF LEARNING

1. Supervised Learning :-

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.

It infers a function from *labeled* training data consisting of a set of training examples.

In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances.

This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

The parallel task in human and animal psychology is often referred to as concept learning.

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.

4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

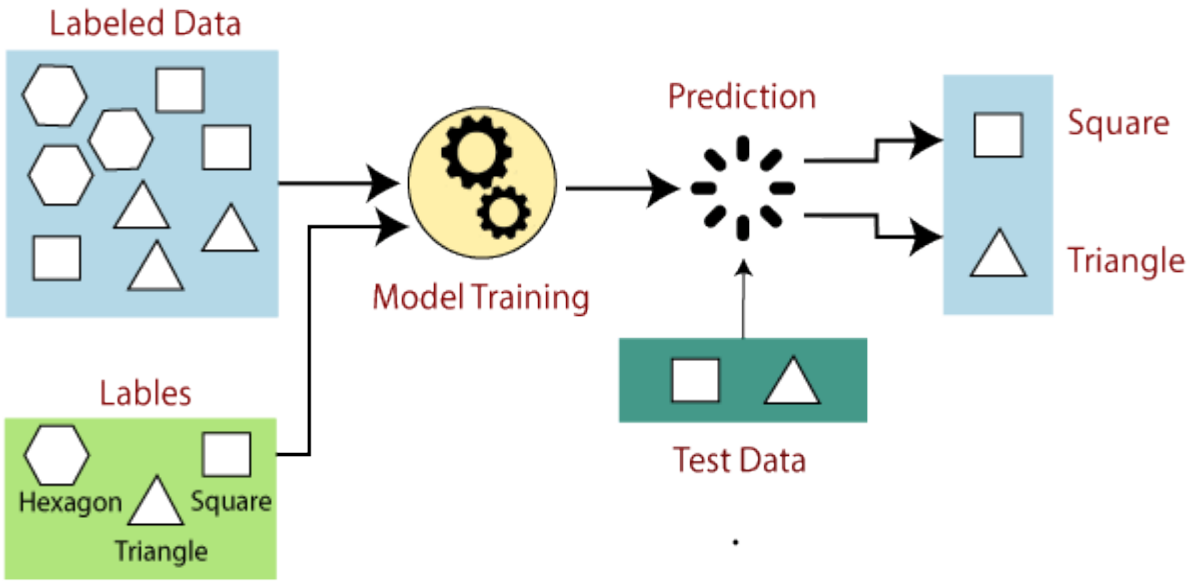
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

A wide range of supervised learning algorithms are available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems

1. Bias-variance tradeoff
2. Function complexity and amount of training data
3. Dimensionality off input space
4. Noise in the output values

EXAPMLE:-


Labeled Data
Prediction
Square
Triangle
Model Training
Lables
Hexagon
Square
Triangle
Test Data

2. Unsupervised Learning :-

Unsupervised learning is a type of self-organized Hebbian learning that helps find previously unknown patterns in data set without pre-existing labels.

It is also known as self-organization and allows modeling probability densities of given inputs.

It is one of the main three categories of machine learning, along with supervised and reinforcement learning.

Semi-supervised learning has also been described, and is a hybridization of supervised and unsupervised techniques.

Two of the main methods used in unsupervised learning are principal component and cluster analysis.

Cluster analysis is used in unsupervised learning to group, or segment, datasets with shared attributes in order to extrapolate algorithmic relationships.

Cluster analysis is a branch of machine learning that groups the data that has not been labelled, classified or categorized.

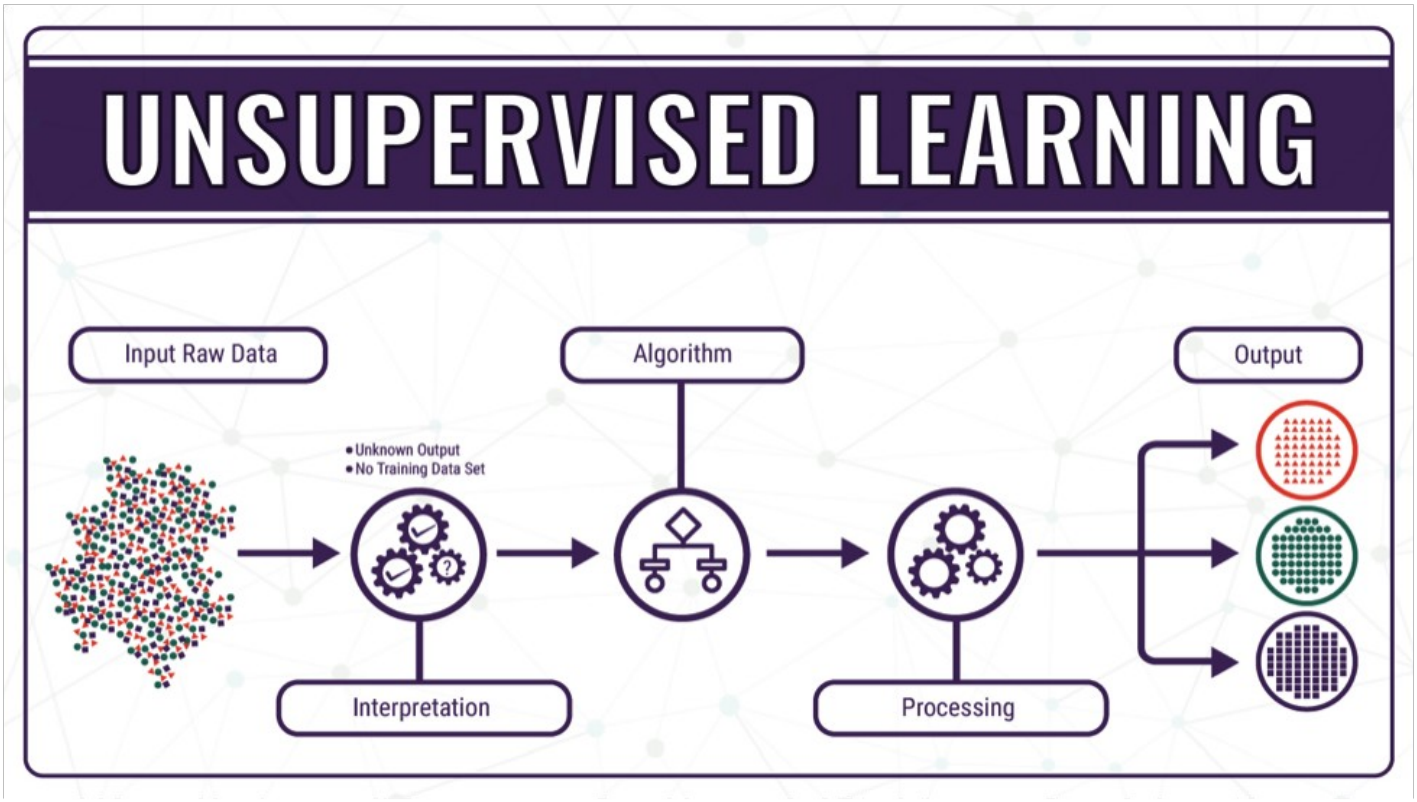
Instead of responding to feedback, cluster analysis identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data.

This approach helps detect anomalous data points that do not fit into either group.

A central application of unsupervised learning is in the field of density estimation in statistics, though unsupervised learning encompasses many other domains involving summarizing and explaining data features.

It could be contrasted with supervised learning by saying that whereas supervised learning intends to infer a conditional probability distribution px(x/y) conditioned on the label y of input data; unsupervised learning intends to infer an a priori probability distribution px(x).

EXAPMLE:-


UNSUPERVISED LEARNING
Input Raw Data
Unknown Output
No Training Data Set
Interpretation
Algorithm
Processing
Output

3. Semi-Supervised Learning :-

Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training.

Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data).

Unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy.

The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location).

The cost associated with the labeling process thus may render large, fully labeled training sets infeasible, whereas acquisition of unlabeled data is relatively inexpensive.

In such situations, semi-supervised learning can be of great practical value.

Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning.

In order to make any use of unlabeled data, some relationship to the underlying distribution of data must exist.

Semi-supervised learning algorithms make use of at least one of the following assumptions**:-**

1. Continuity Assumptions
2. Cluster Assumptions
3. Manifold assumptions

EXAMPLE :-

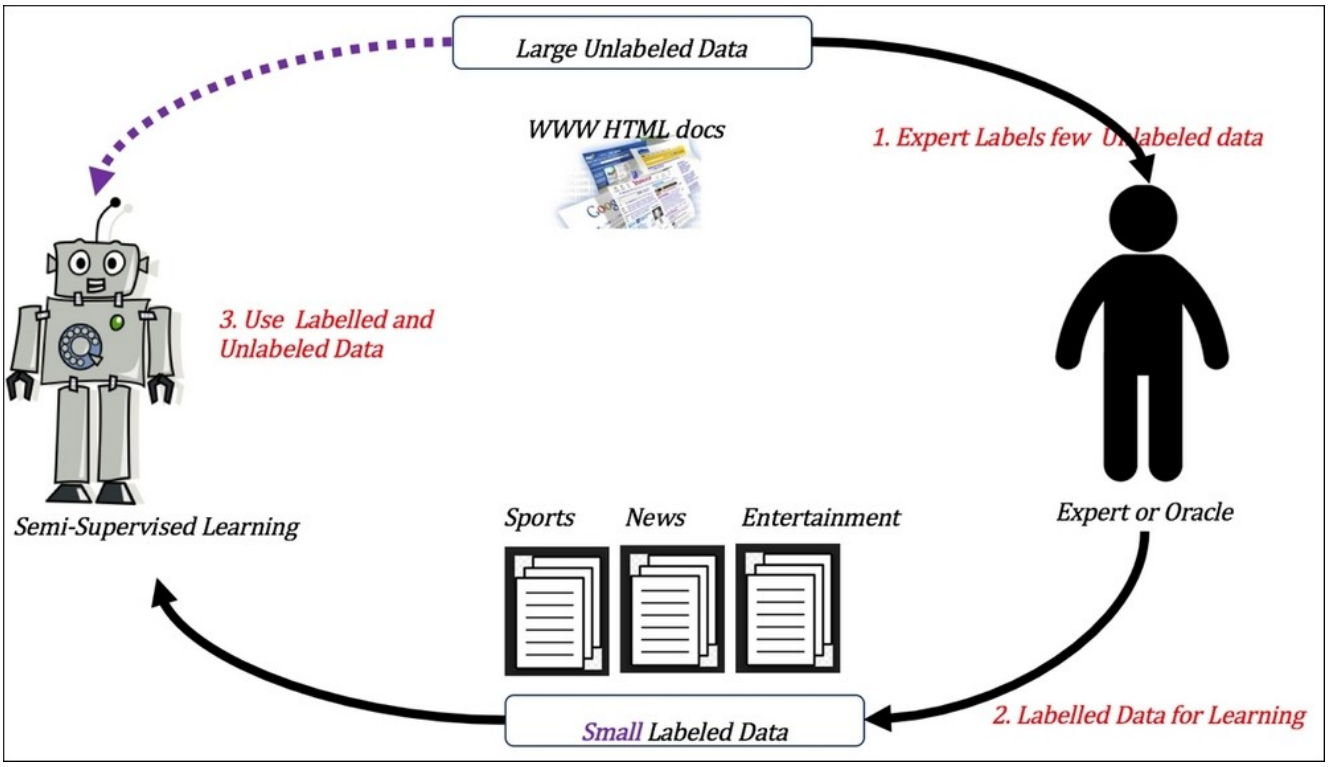
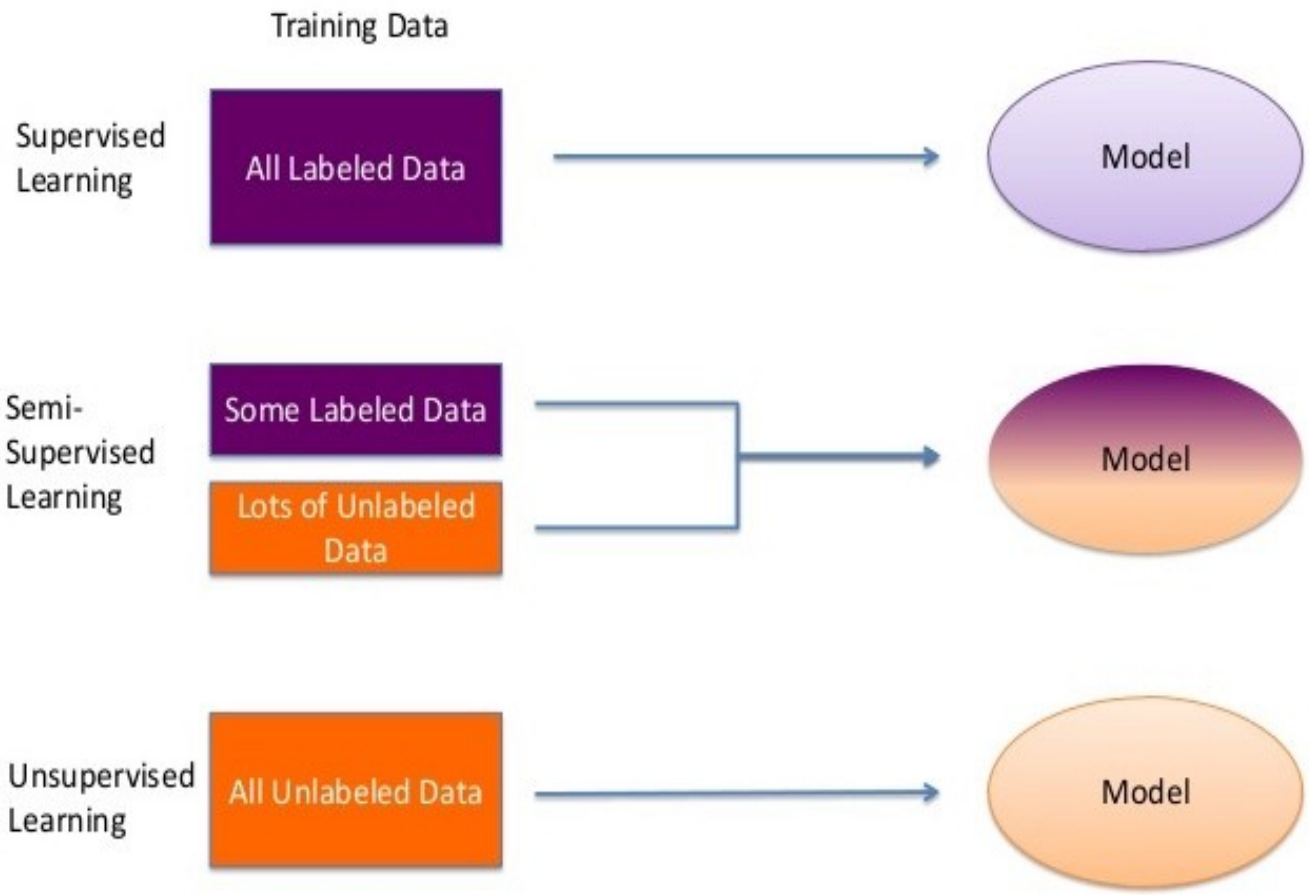

Large Unlabeled Data
WWW HTML docs
1. Expert Labels few Unlabeled data
3. Use Labelled and Unlabeled Data
Semi-Supervised Learning
Sports
News
Entertainment
Expert or Oracle
Small Labeled Data
2. Labelled Data for Learning

DIAGRAM:-


Training Data
Supervised Learning
All Labeled Data
Model
Semi-Supervised Learning
Some Labeled Data
Lots of Unlabeled Data
Model
Unsupervised Learning
All Unlabeled Data
Model

TECHNOLOGY USED

This project will be a AI Drawing Based on Desktop Application to be developed in C#.

Front End

- Form Design (VB.NET & C# Visual)
- Back Coding (VB.NET & C#)
- Testing (VB.NET & C#)

MAIN REPORT

Objective & Scope of Project

Objective of Project: -

In my Project I Primarily concentrated on the clustering algorithm and how it can be used for automatic drawing and painting using AI techniques.

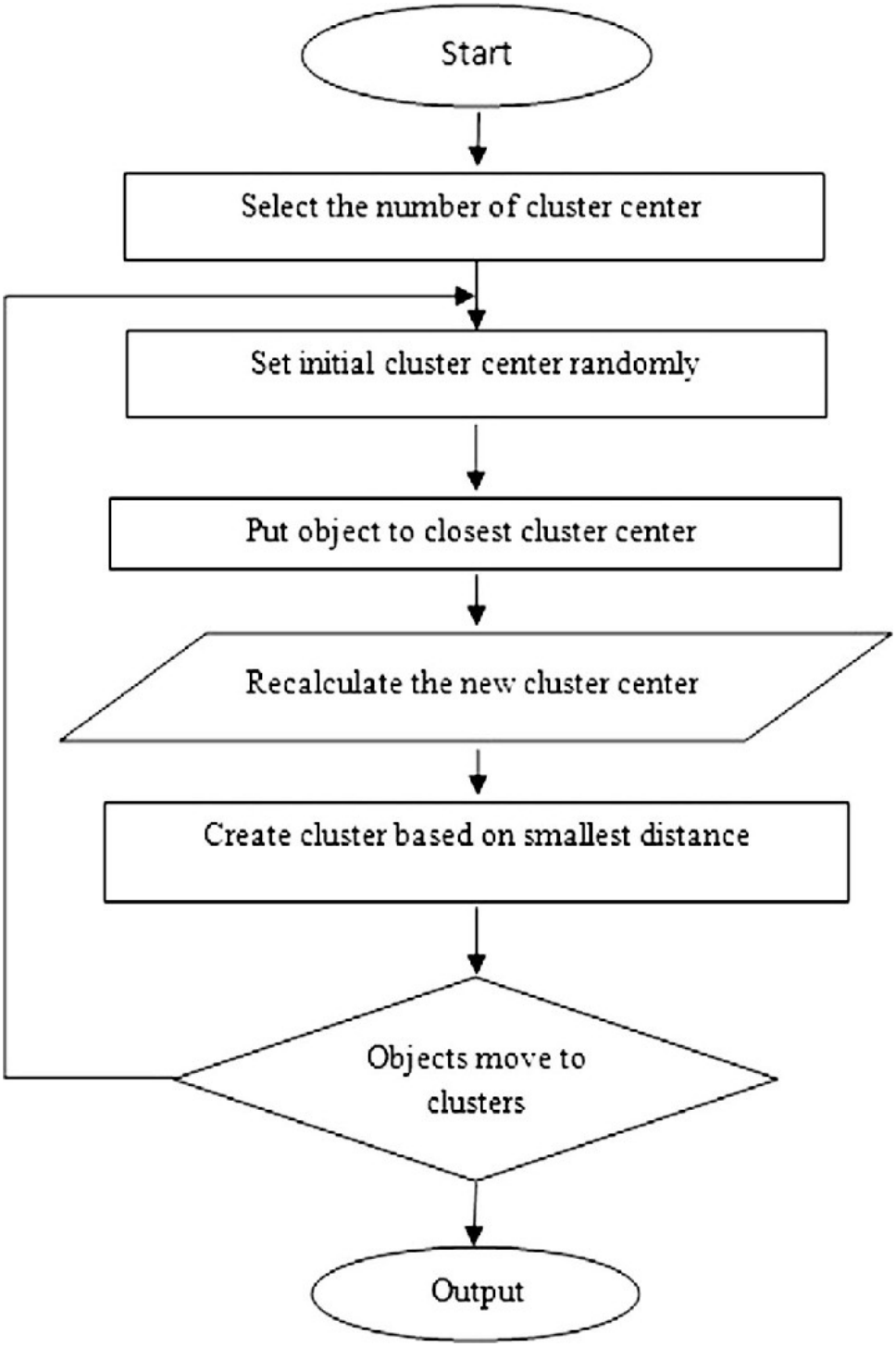
K-Means falls under the category of centroid-based clustering.

The algorithm has a loose relationship to the *k*-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with *k*-means due to the name.

Applying the 1-nearest neighbor classifier to the cluster centers obtained by *k*-means classifies new data into the existing clusters.

This is known as nearest centroid classifier or Rocchio algorithm.

FLOWCHART OF K-MEANS CLUSTERING ALGORITHAM:-



PROJECT SCOPE

This project is developed for developing and creating pattern using a drawing window. The scope of the project is implementation of clustering algorithm for automatic drawing and painting technique

The scope of the project is to use k means clustering algorithm to draw images and patterns. To meet the requirement, I use the simple and basic approach of AI clusters.

In these project the proposed approach find the suitable algorithm for developing a pattern, and replicate it automatically by the system.

THEORETICAL BACKGROUND

What is Clusteing Algorithm?

K-Means falls under the category of centroid-based clustering. A centroid is a data point (imaginary or real) at the center of a cluster. In centroid-based clustering, clusters are represented by a central vector or a centroid. This centroid might not necessarily be a member of the dataset. Centroid-based clustering is an iterative algorithm in which the notion of similarity is derived by how close a data point is to the centroid of the cluster.

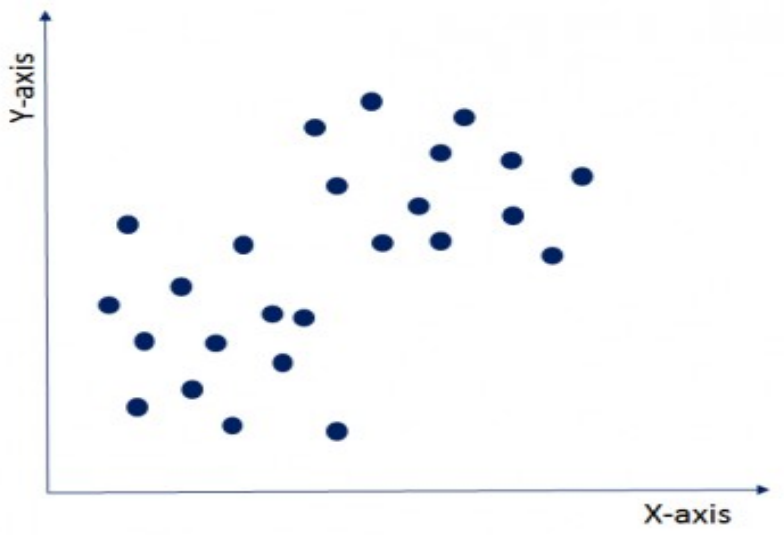
Theory of K-means Clustering

k-means clustering is one of the simplest algorithms which uses unsupervised learning method to solve known clustering issues. k-means clustering require following two inputs.

k = number of clusters

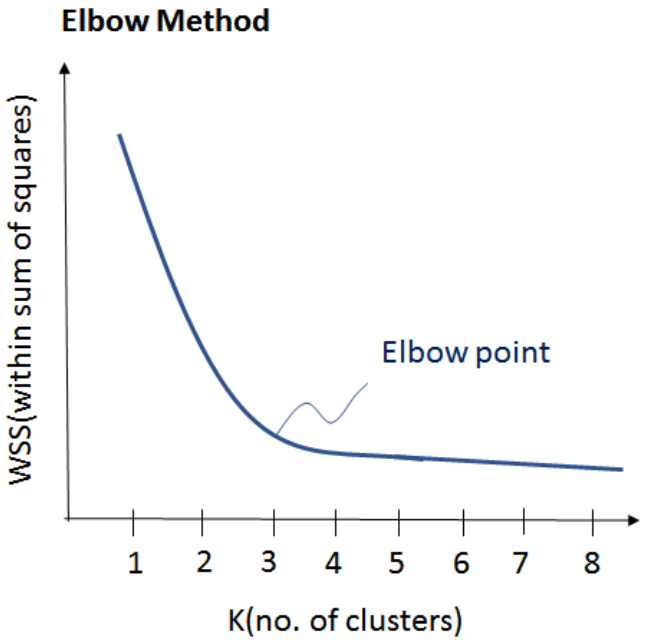
Training set(m) = {x1, x2, x3,……., xm}

Let’s say you have an unlabeled data set like the one shown below and you want to group this data into clusters.


Y-axis
X-axis

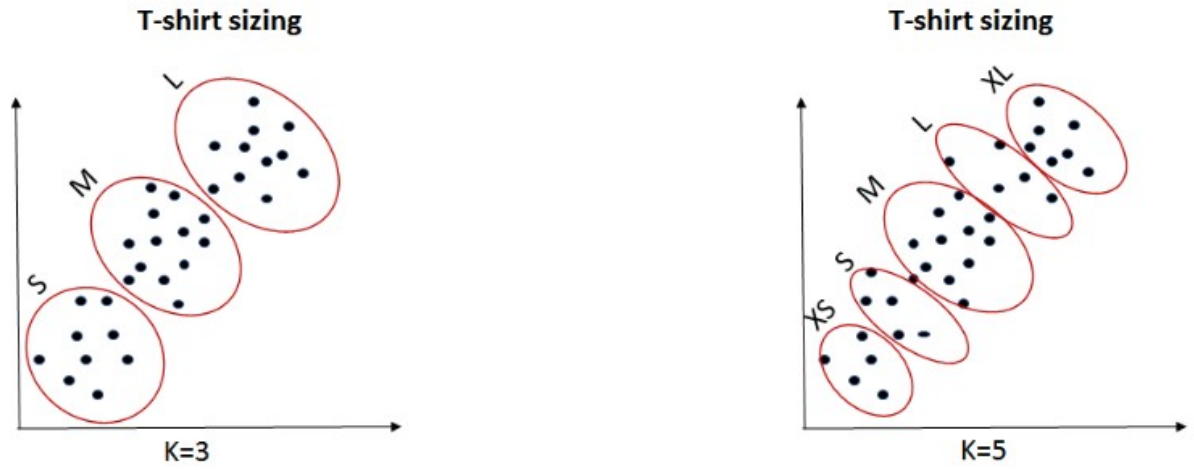
Now, the important question is how should you choose the optimum number of clusters? There are two possible ways for choosing the number of clusters.

(i) Elbow Method: Here, you draw a curve between WSS (within sum of squares) and the number of clusters. It is called elbow method because the curve looks like a human arm and the elbow point gives us the optimum number of clusters. As you can see that after the elbow point, there is a very slow change in the value of WSS, so you should take the elbow point value as the final number of clusters.



(ii) Purpose Based: You can run k-means clustering algorithm to get different clusters based on a variety of purposes. You can partition the data on different metrics and see how well it performs for that particular case. Let's take an example of marketing T-shirts of different sizes. You can partition the dataset into different number of clusters depending upon the purpose that you want to meet. In the following example, I have taken two different criteria, price and comfort.

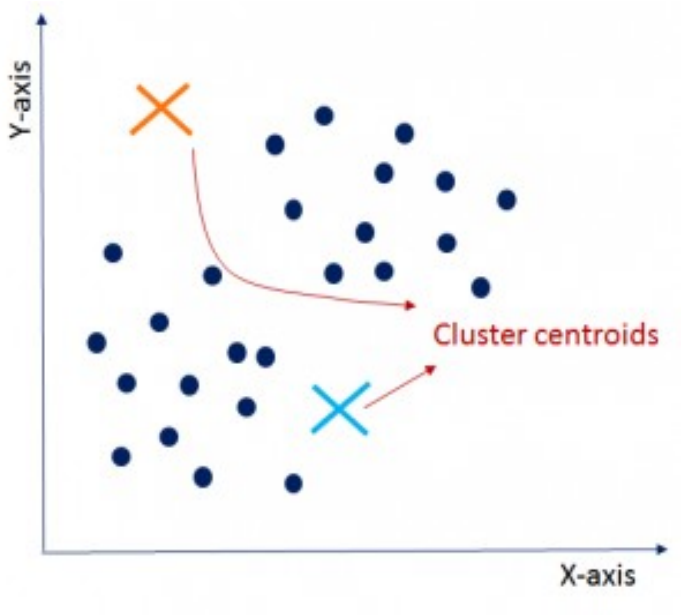
Let's see these two possibilities as shown in the image below.



K=3: If you want to provide only 3 sizes(S, M, L) so that prices are cheaper, you will divide the data set into 3 clusters.

K=5: Now, if you want to provide more comfort and variety to your customers with more sizes (XS, S, M, L, XL), then you will divide the data set into 5 clusters.

Now, once we have the value of k with us, let's understand it's execution.



Initialisation:-

Firstly, you need to randomly initialise two points called the cluster centroids. Here, you need to

Make sure that your cluster centroids depicted by an orange and blue cross as shown in the image are less than the training data points depicted by navy blue dots. k-means clustering

algorithm is an iterative algorithm and it follows next two steps iteratively. Once you are done with the initialization, let's move on to the next step.

Cluster Assignment:-

In this step, it will go through all the navy blue data points to compute the distance between the data points and the cluster centroid initialised in the previous step. Now, depending upon the minimum distance from the orange cluster centroid or blue cluster centroid, it will group itself into that particular group. So, data points are divided into two groups, one represented by orange color and the other one in blue color as shown in the graph. Since these cluster formations are not the optimised clusters, so let's move ahead and see how to get final clusters.

Move Centroid:

Now, you will take the above two cluster centroids and iteratively reposition them for optimization. You will take all blue dots, compute their average and move current cluster

centroid to this new location. Similarly, you will move orange cluster centroid to the average of orange data points. Therefore, the new cluster centroids will look as shown in the graph. Moving forward, let's see how can we optimize clusters which will give us better insight.

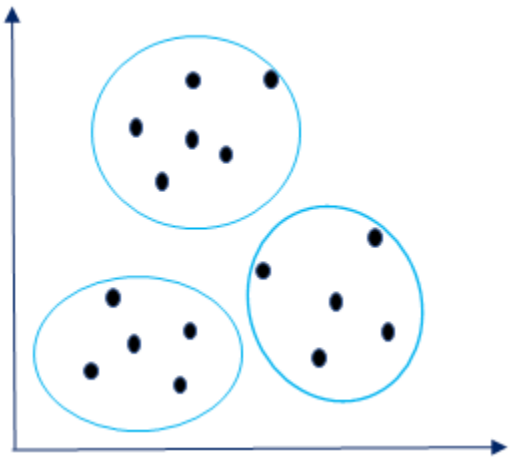
Optimization:

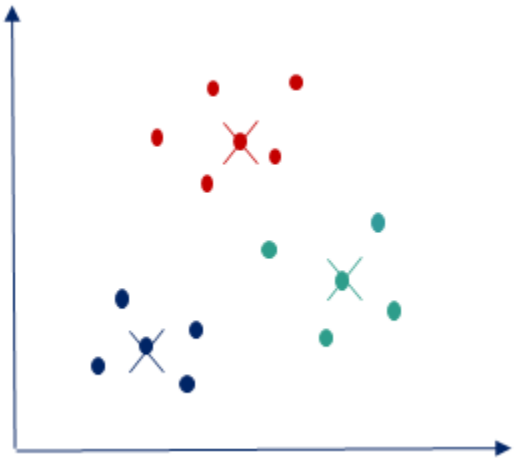
You need to repeat above two steps iteratively till the cluster centroids stop changing their positions and become static. Once the clusters become static, then k-means clustering algorithm is said to be converged.

Convergence:

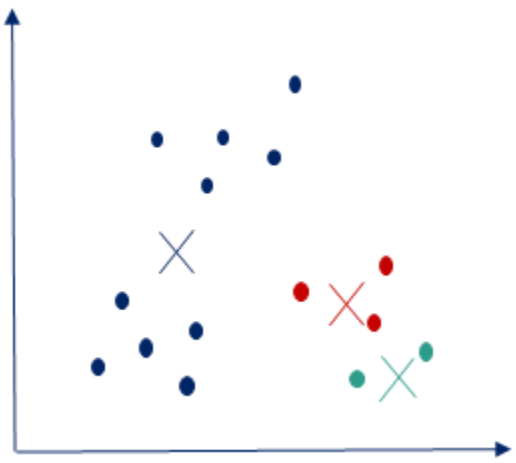
Finally, k-means clustering algorithm converges and divides the data points into two clusters clearly visible in orange and blue. It can happen that k-means may end up converging with different solutions depending on how the clusters were initialised.

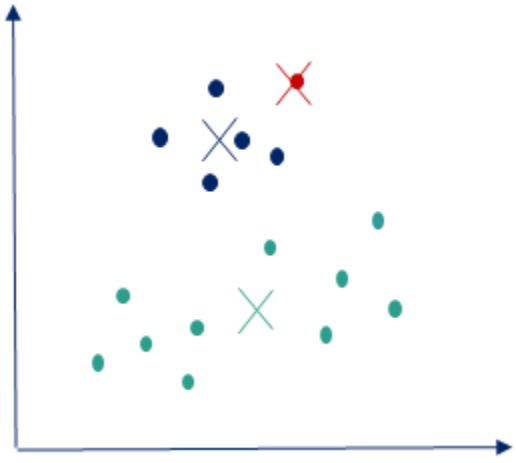
As you can see in the graph below, the three clusters are clearly visible but you might end up having different clusters depending upon your choice of cluster centroids.





Below shown are some other possibilities of cluster partitioning based on the different choice of cluster centroids. You may end up having any of these groupings based on your requirements and the goal that you are trying to achieve.





Definition of Problems

The project AI Drawing using clustering algorithm System deals with the problems on creating and developing potraits without the use of human intelligence or physical efforts.

And hence there is a lot other software which is been used .the main question is the efficiency and speed.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user-friendly.

We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

System Analysis and Design

Algorithm 1: The *k*-means algorithm and its output.

Algorithm: k-means.

The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of
The objects in the cluster.

Input:

K: the number of clusters,

D: a data set containing *n* objects.

Output:

A set of *k* clusters.

Method:

(1) randomly choose *k* objects from *D* as the initial cluster centers;

(2) repeat

(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects
in the cluster;

(4) update the cluster means, i.e., calculate the mean value of the objects for each clusters;

(5) until no change;

Figure 1 :-

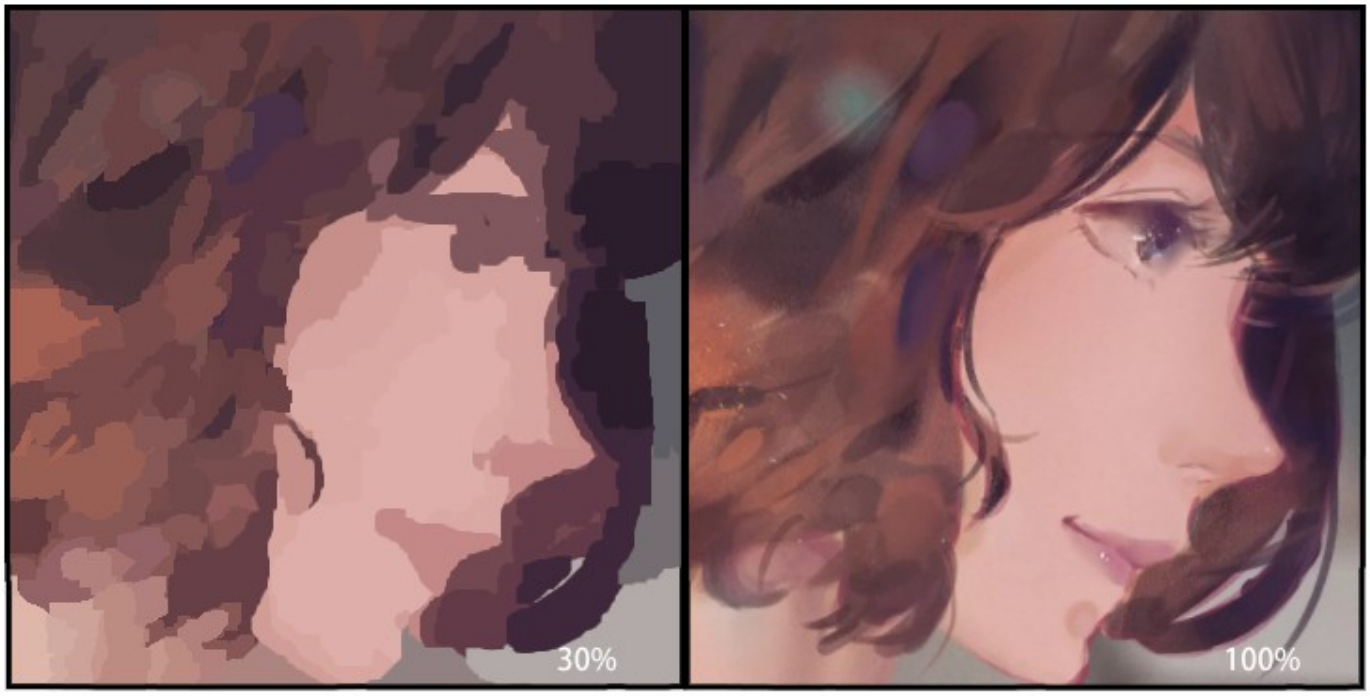
Illustrates how the algorithm is run one step at a time. It demonstrates three groups in a space where *k* is equal to three, and they are distinguished according to the colors with their represented data elements (blue, brown, and green). For additional clarification they are isolated along three parts.

Figure 1(a) is the initial centroids selection at random where *k* number of initial centroids have been depicted

And accordingly coloring their entire group with the same color.

The centroids are characterized by the plus sign.

A desktop application that can draw images automaticlly.



Hierarchical clustering Technique

Hierarchical clustering is one of the popular and easy to understand clustering technique.

This clustering technique is divided into two types:

1) Agglomerative
2) Divisive

Agglomerative Hierarchical clustering Technique:

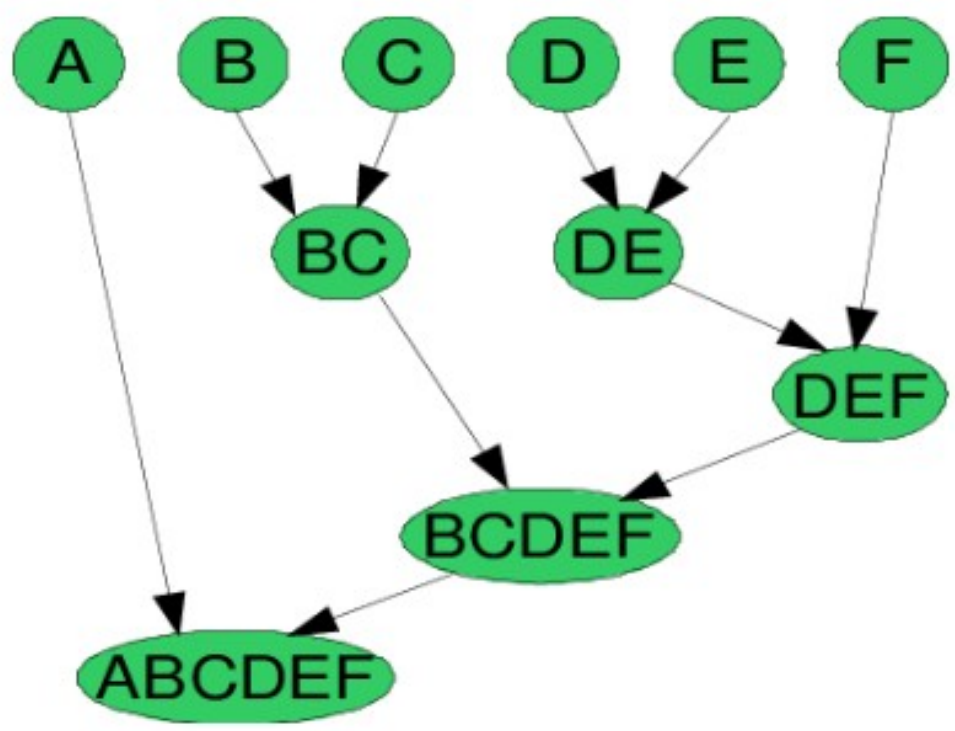
In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

The basic algorithm of Agglomerative is straight forward.
Compute the proximity matrix
Let each data point be a cluster
Repeat: Merge the two closest clusters and update the proximity matrix
Until only a single cluster remains

Key operation is the computation of the proximity of two clusters

To understand better let's see a pictorial representation of the Agglomerative Hierarchical clustering Technique. Lets say we have six data points {A,B,C,D,E,F}.

Step- 1: In the initial step, we calculate the proximity of individual points and consider all the six data points as individual clusters as shown in the image below.



Agglomerative Hierarchical Clustering Technique

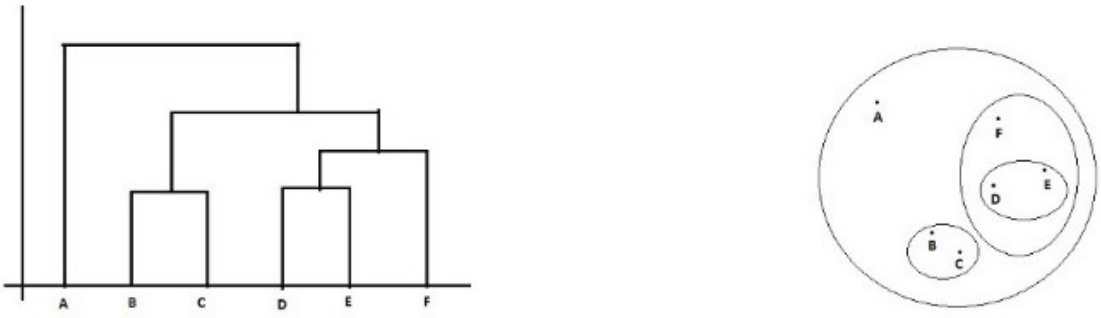
Step- 2: In step two, similar clusters are merged together and formed as a single cluster. Let's consider B,C, and D,E are similar clusters that are merged in step two. Now, we're left with four clusters which are A, BC, DE, F.

Step- 3: We again calculate the proximity of new clusters and merge the similar clusters to form new clusters A, BC, DEF.

Step- 4: Calculate the proximity of the new clusters. The clusters DEF and BC are similar and merged together to form a new cluster.
We're now left with two clusters A, BCDEF.

Step- 5: Finally, all the clusters are merged together and form a single cluster.
The Hierarchical clustering Technique can be visualized using a **Dendrogram.**

A **Dendrogram** is a tree-like diagram that records the sequences of merges or splits.



Dendrogram representation

Limitations of Hierarchical clustering Technique

1) There is no mathematical objective for Hierarchical clustering.

2) All the approaches to calculate the similarity between clusters has its own disadvantages.

3) High space and time complexity for Hierarchical clustering.

4) Hence this clustering algorithm cannot be used when we have huge data.

Divisive Hierarchical clustering Technique

Since the Divisive Hierarchical clustering Technique is not much used in the real world, I'll give a brief of the Divisive Hierarchical clustering Technique.

In simple words, we can say that the Divisive Hierarchical clustering is exactly the opposite of the **Agglomerative Hierarchical clustering.**

In Divisive Hierarchical clustering, we consider all the data points as a single cluster and in each iteration, we separate the data points from the cluster which are not similar.

Each data point which is separated is considered as an individual cluster.
In the end, we'll be left with n clusters.

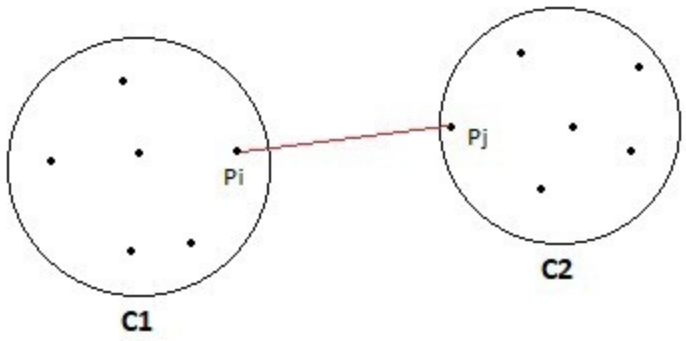
As we're dividing the single clusters into n clusters, it is named as **Divisive Hierarchical clustering.**

HOW DO WE CALCULATE THE SIMILARITY BETWEEN TWO CLUSTERS ?

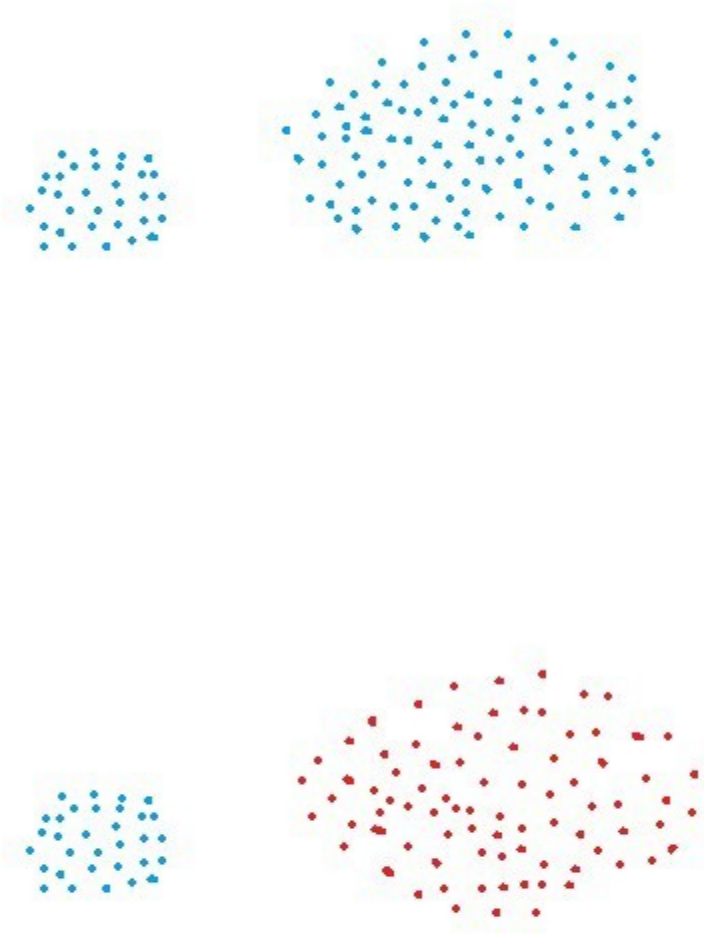
Calculating the similarity between two clusters is important to merge or divide the clusters.
There are certain approaches which are used to calculate the similarity between two clusters:

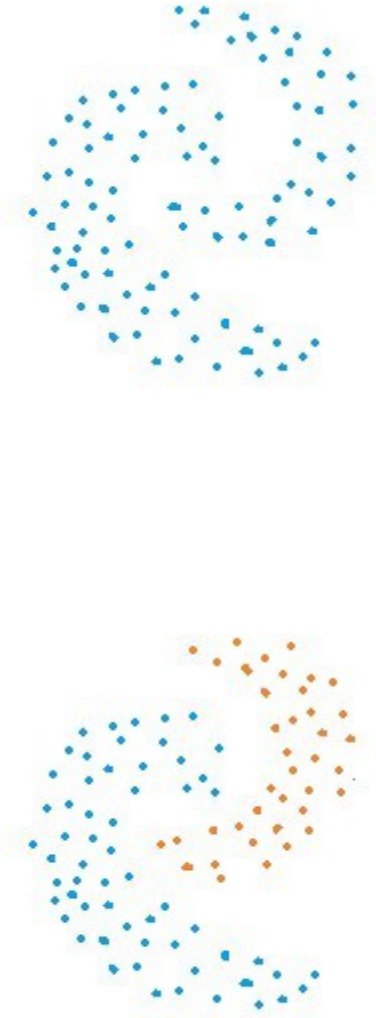
- MIN

- MAX
- Group Average
- Distance Between Centroids
- Ward's Method
- **MIN:** Also known as single linkage algorithm can be defined as the similarity of two clusters C1 and C2 is equal to the **minimum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.



-

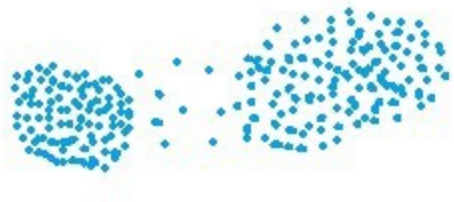


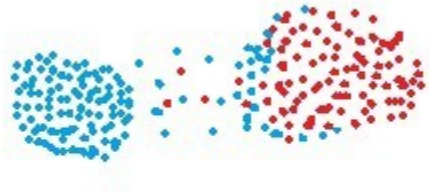


Original data vs Clustered data using MIN approach

Cons of MIN:

- MIN approach cannot separate clusters properly if there is noise between clusters.





MAX:

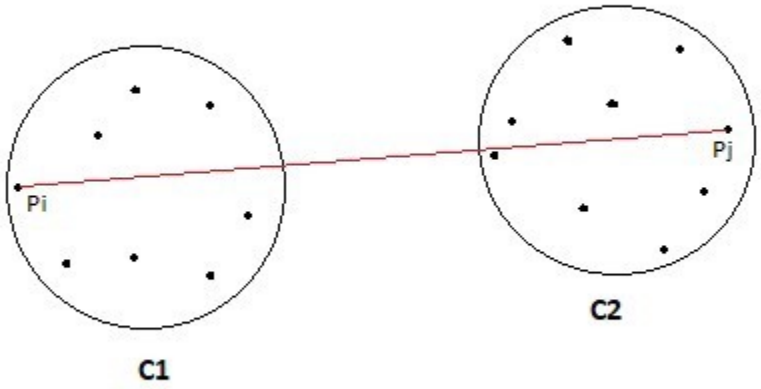
Also known as the complete linkage algorithm, this is exactly opposite to the **MIN** approach.

The similarity of two clusters C1 and C2 is equal to the **maximum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.

Mathematically this can be written as,

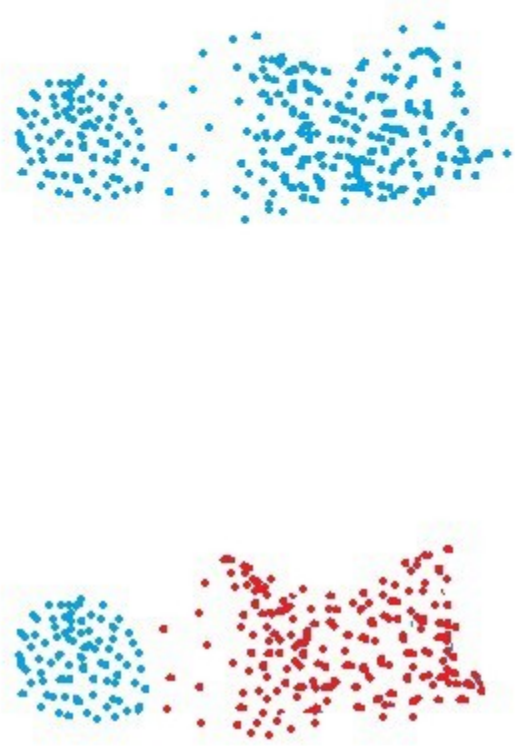
$Sim(C1,C2) = Max\ Sim(Pi,Pj)$ such that $Pi \in C1$ & $Pj \in C2$

In simple words, pick the two farthest points such that one point lies in cluster one and the other point lies in cluster 2 and take their similarity and declare it as the similarity between two clusters.



Pros of MAX:

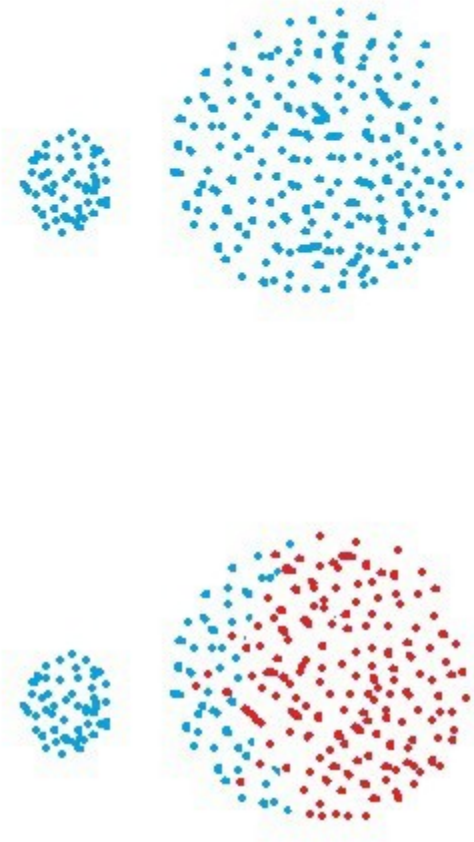
- MAX approach does well in separating clusters if there is noise between clusters.



Original data vs Clustered data using MAX approach

Cons of Max:

- Max approach is biased towards globular clusters.
- Max approach tends to break large clusters.



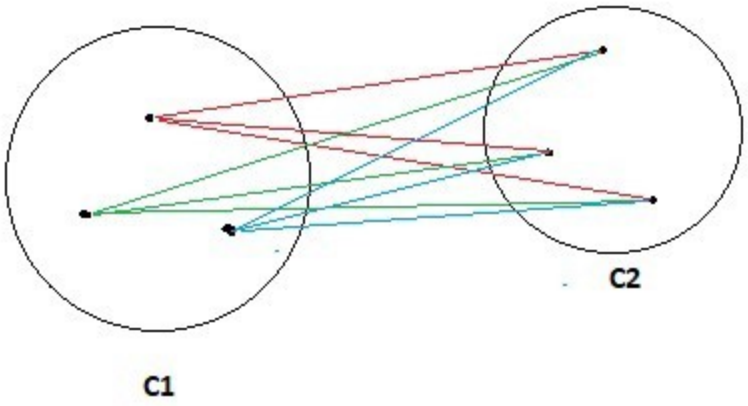
Original data vs Clustered data using MAX approach

Group Average: Take all the pairs of points and compute their similarities and calculate the average of the similarities.

Mathematically this can be written as,

$sim(C1,C2) = \sum sim(Pi, Pj)/|C1|*|C2|$

where, $Pi \in C1$ & $Pj \in C2$

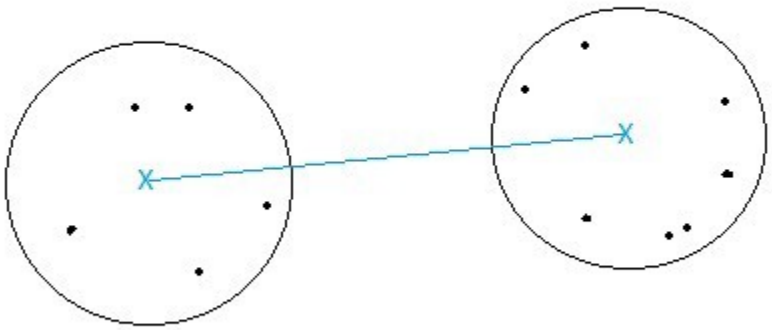


Pros of Group Average:

- The group Average approach does well in separating clusters if there is noise between clusters.

Cons of Group Average:

- The group Average approach is biased towards globular clusters.
- **Distance between centroids:** Compute the centroids of two clusters C1 & C2 and take the similarity between the two centroids as the similarity between two clusters.
- This is a less popular technique in the real world.



Ward's Method: This approach of calculating the similarity between two clusters is exactly the same as Group Average except that Ward's method calculates the sum of the square of the distances Pi and PJ.

Mathematically this can be written as,

$sim(C1,C2) = \sum (dist(Pi, Pj))^2/|C1|*|C2|$

Pros of Ward's method:

- Ward's method approach also does well in separating clusters if there is noise between clusters.

Cons of Ward's method:

- Ward's method approach is also biased towards globular clusters.

Space and Time Complexity of Hierarchical clustering Technique:-

Space complexity:

The space required for the Hierarchical clustering Technique is very high when the number of data points are high as we need to store the similarity matrix in the RAM. The space complexity is the order of the square of n.

Space complexity = $O(n^2)$ where n is the number of data points.

Time complexity:

Since we've to perform n iterations and in each iteration, we need to update the similarity matrix and restore the matrix, the time complexity is also very high. The time complexity is the order of cube of n.

Time complexity = $O(n^3)$ where n is the number of data points.

SYSTEM PLANNING

Gantt chart

Gantt chart is a type of bar chart that illustrates a project schedule, named after its inventor, Henry Gantt (1861–1919), who designed such a chart around the years 1910–1915. Modern Gantt charts also show the dependency relationships between activities and current schedule status.

Definition:

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line as shown here.

Gantt charts are sometimes equated with bar charts.

Gantt charts are usually created initially using an *early start time approach*, where each task is scheduled to start immediately when its prerequisites are complete. This method maximizes the float time available for all tasks.

History

Although now regarded as a common charting technique, Gantt charts were considered revolutionary when first introduced. The first known tool of this type was developed in 1896 by Karol Adamiecki, who called it a *harmonogram*. Adamiecki did not publish his chart until 1931, however, and only in Polish, which limited both its adoption and recognition of his authorship.

In 1912, Hermann Schürch [de] published what would be considered Gantt charts while discussing a construction project. It appears that Schürch's charts were not notable but rather routine in Germany at the time they were published.The prior development leading to Schürch's work is unknown. Unlike later Gantt charts, Schürch's charts did not display interdependencies, leaving them to be inferred by the reader. These were also static representations of a planned schedule.

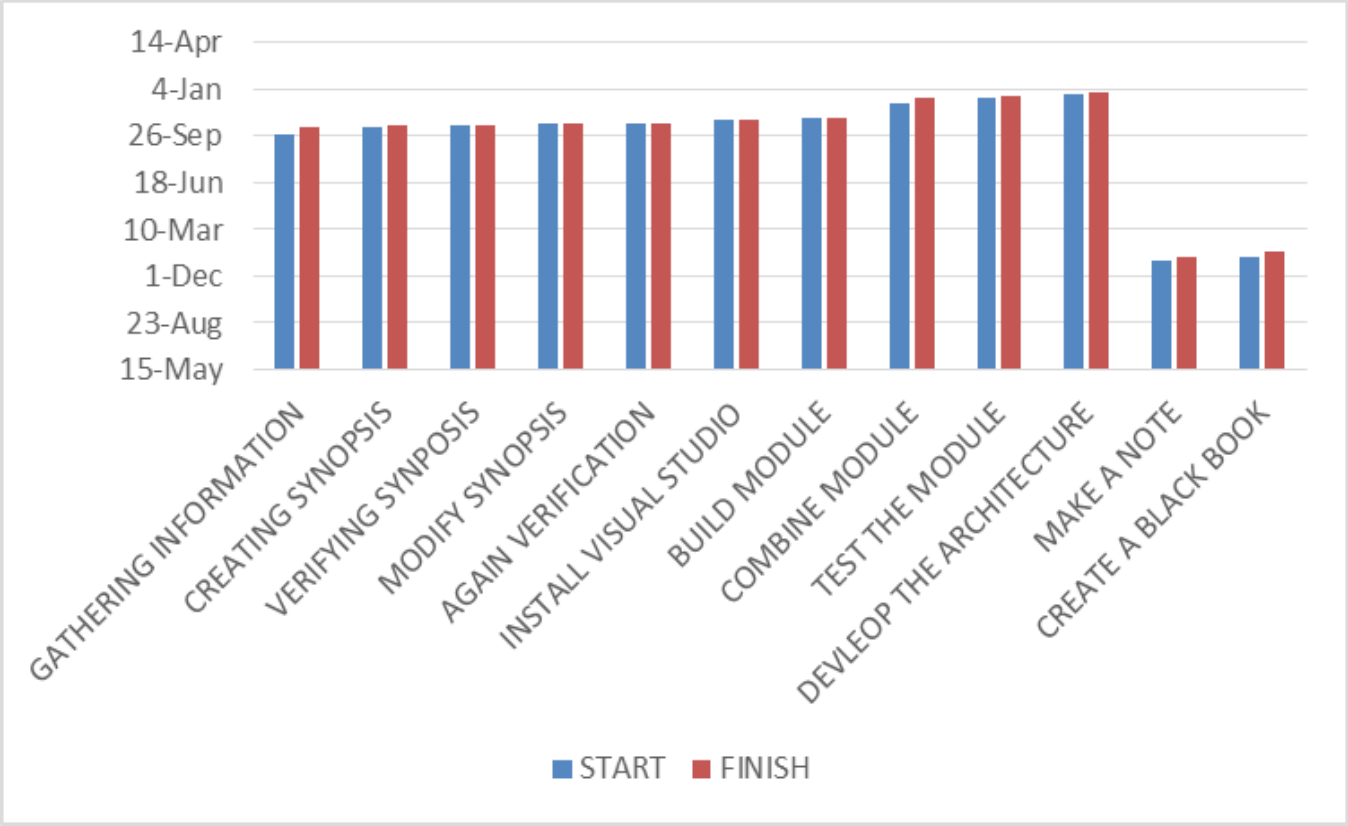
The chart is named after Henry Gantt (1861–1919), who designed his chart around the years 1910–1915.

One of the first major applications of Gantt charts was by the United States during World War I, at the instigation of General William Crozier.

The earliest Gantt charts were drawn on paper and therefore had to be redrawn entirely in order to adjust to schedule changes. For many years, project managers used pieces of paper or blocks for Gantt chart bars so they could be adjusted as needed. Gantt's collaborator, Walter Polakov introduced Gantt charts to the Soviet Union in 1929 when he was working for the Supreme Soviet of the National Economy. They were used in developing the First Five Year Plan, supplying Russian translations to explain their use. In the 1980s, personal computers allowed widespread creation of complex and elaborate Gantt charts.

The first desktop applications were intended mainly for project managers and project schedulers. With the advent of the Internet and increased collaboration over networks at the end of the 1990s, Gantt charts became a common feature of web-based applications, including collaborative groupware. By 2012, almost all Gantt charts were made by software which can easily adjust to schedule changes.In 1999, Gantt charts were identified as "one of the most widely used management tools for project scheduling and control.

Gantt chart


14-Apr
4-Jan
26-Sep
18-Jun
10-Mar
1-Dec
23-Aug
15-May
GATHERING INFORMATION
CREATING SYNOPSIS
VERIFYING SYNPOSIS
MODIFY SYNOPSIS
AGAIN VERIFICATION
INSTALL VISUAL STUDIO
BUILD MODULE
COMBINE MODULE
TEST THE MODULE
DEVLEOP THE ARCHITECTURE
MAKE A NOTE
CREATE A BLACK BOOK
START
FINISH

METHODOLOGY ADOPTED, DETAILS OF HARDWARE AND SOFTWARE USED

Hardware and Software tools to be used

1. Microsoft Visual Studio 2012

Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native codeand managed code.

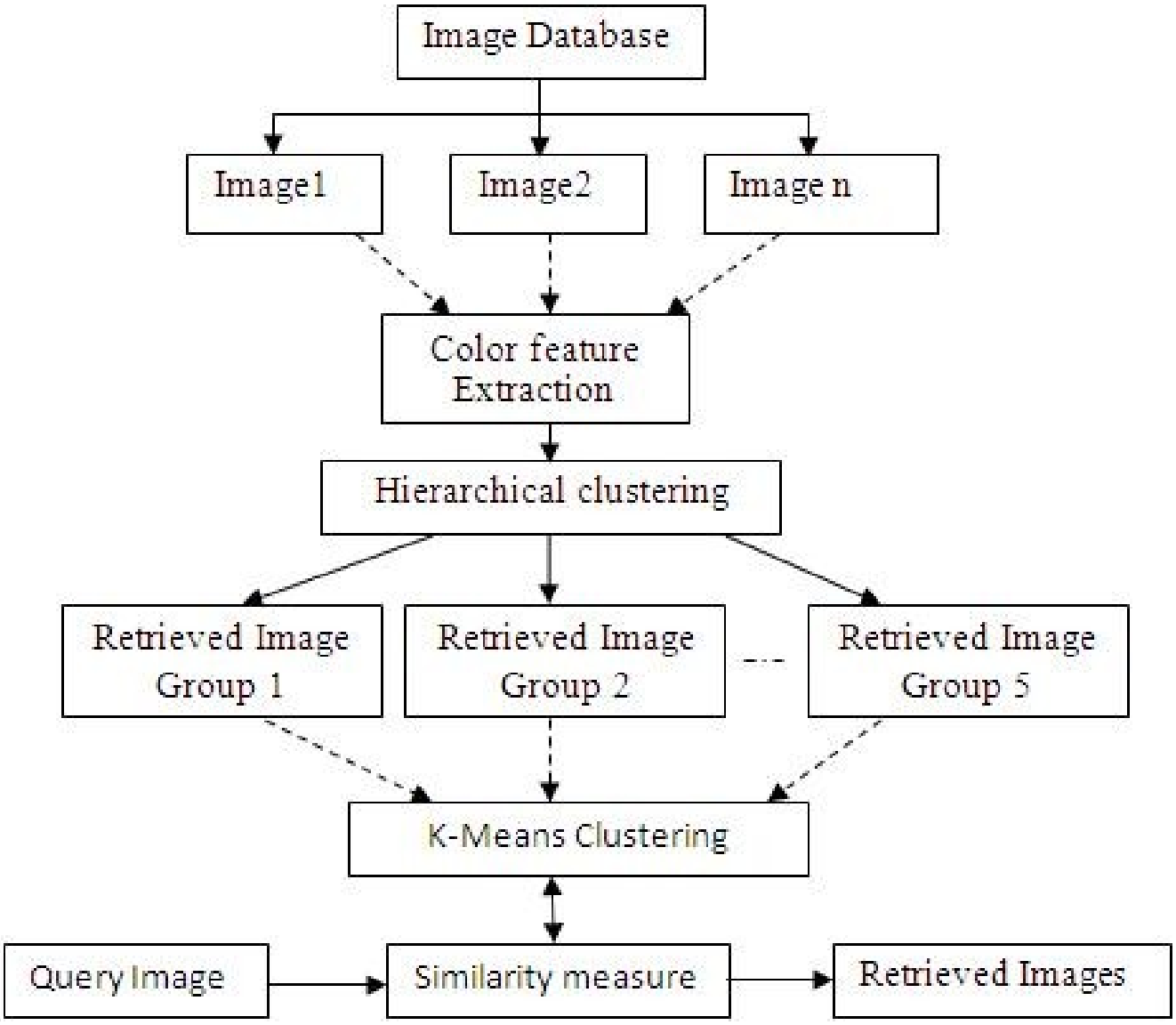
Visual Studio includes a code editor supporting IntelliSense (the code completioncomponent) as well as code refactoring.

The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, classdesigner, and database schema designer.

It accepts plug-ins that enhance the functionality at almost every level—including adding support for source controlsystems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle

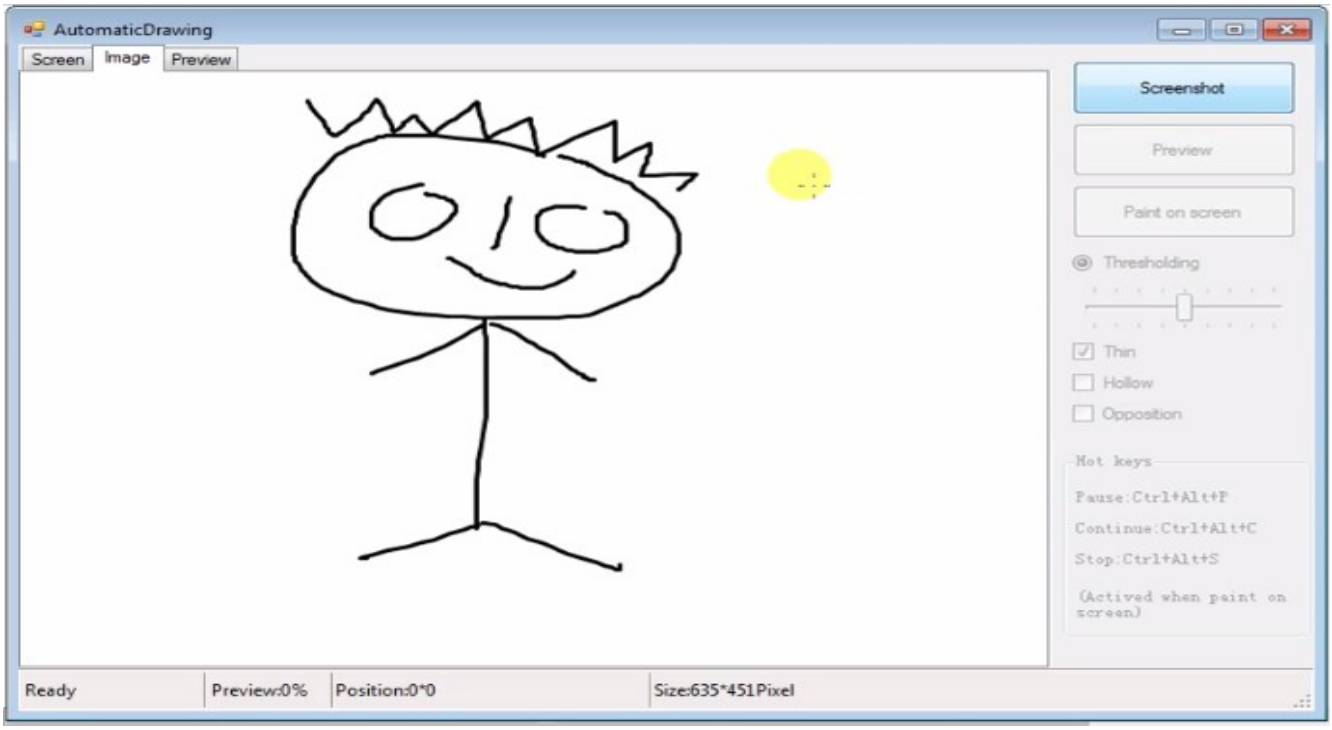
Life Cycle of the Project

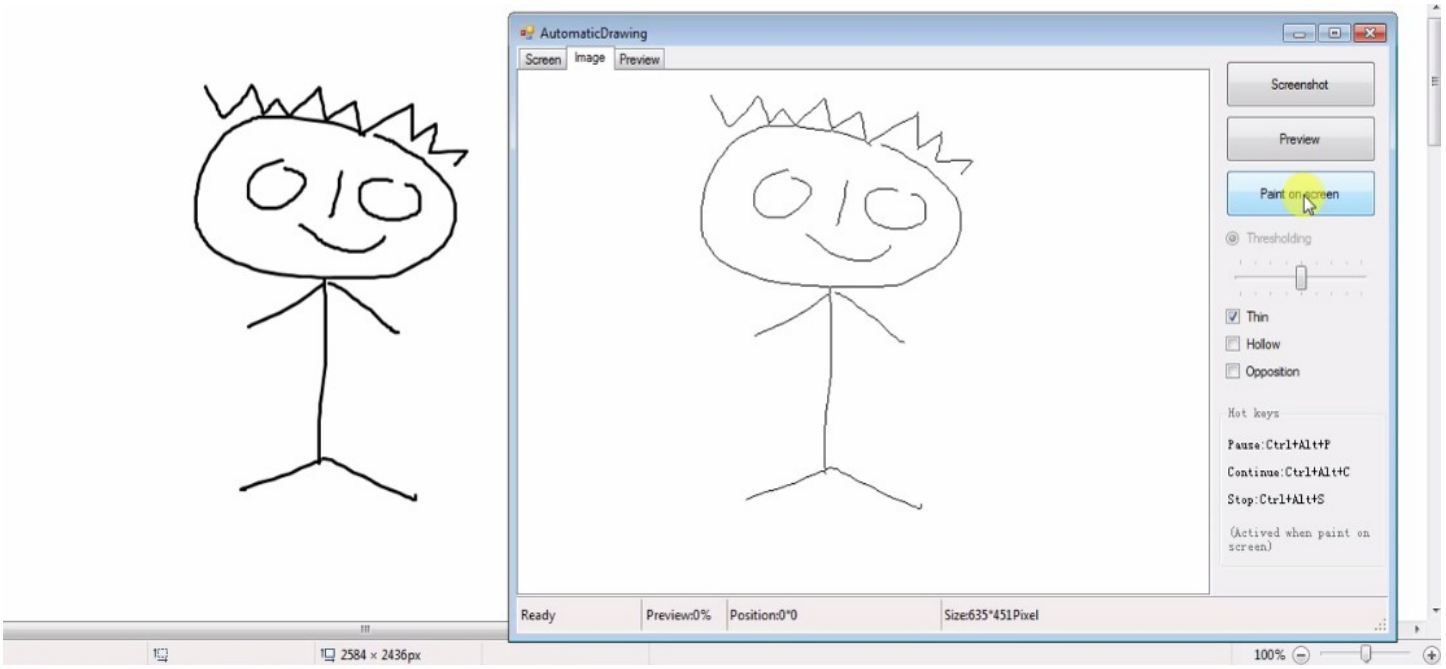
Block Diagram



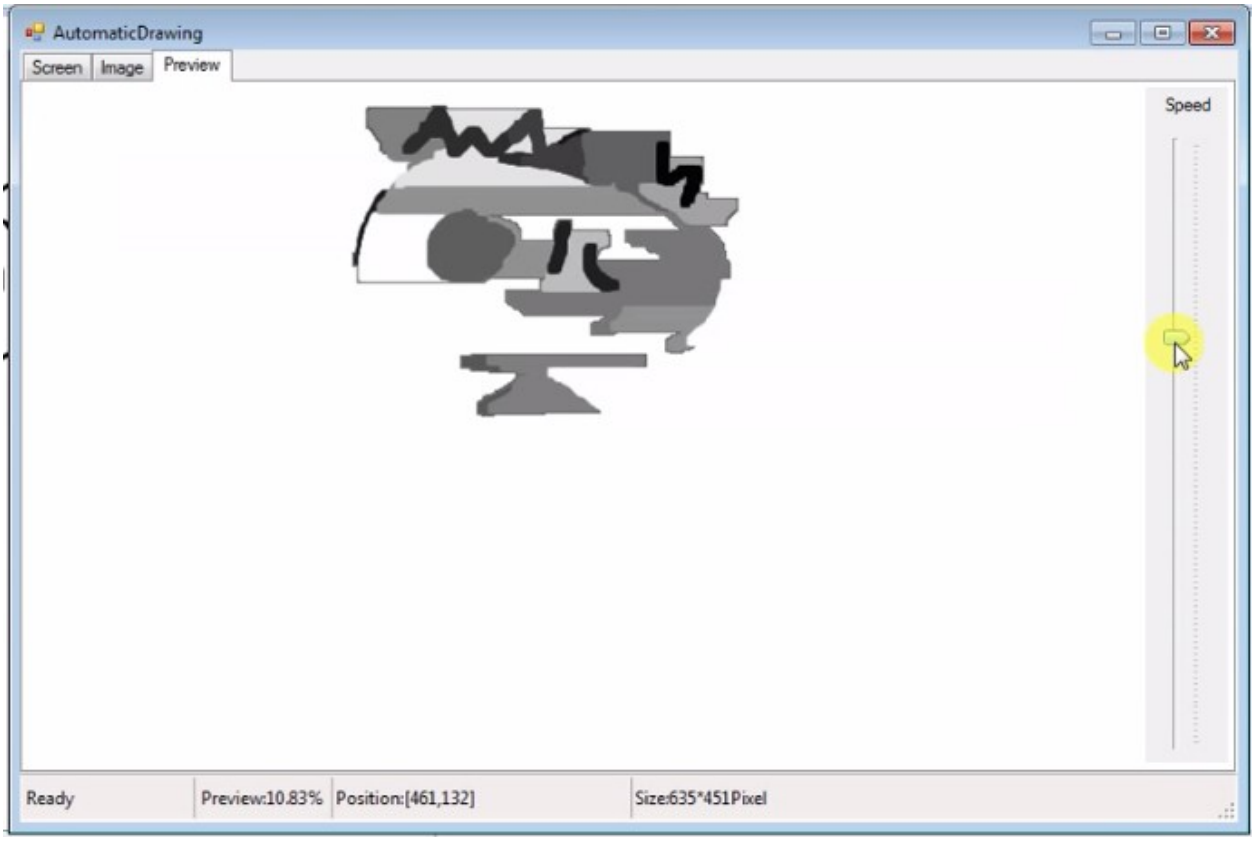
INPUT OUTPUT DESIGN

Draw Window




AutomaticDrawing
Screen
Image
Preview
Screenshot
Preview
Paint on screen
Thresholding
Thin
Hollow
Opposition
Hot keys
Pause:Ctrl+Alt+P
Continue:Ctrl+Alt+C
Stop:Ctrl+Alt+S
(Actived when paint on screen)
Ready
Preview:0%
Position:0*0
Size:635*451Pixel
2584 × 2436px
100%

Clustering Process and Developing the Image



CODE

```
Imports System.Drawing
Imports System.Numerics
Imports AutoPaint.Core
''' <summary>
''' </summary>
Public Class ClusterAI
    ''' <summary>
    ''' </summary>
    Public Property Lines As New List(Of ILine)
    ''' <summary>
    ''' </summary>
    Public Property Hierarchies As New List(Of IHierarchy)

    ''' <summary>
    ''' </summary>
    Public Sub New(pixels As PixelData, Optional maxRank As Integer = 10)
        Dim start As DateTime = DateTime.Now
        Debug.WriteLine("Initialize Start")
        Hierarchies.Add(GridHierarchy.CreateFromPixels(pixels))
        Debug.WriteLine($"Initialize Over,TimeConsuming:{DateTime.Now - start}")
        Debug.WriteLine($"Generation Start")
        For i = 0 To maxRank - 1
            start = DateTime.Now
```

```
        Hierarchies.Add(Hierarchies.Last.Generate())


    Debug.WriteLine($"Total:{maxRank},Current:{i + 1},Time Consuming:{DateTime.Now -
start},Hierarchy[{Hierarchies.Last.ToString()}]")
      Next
      Debug.WriteLine(pixels.Colors.Length)
      For i = maxRank - 1 To 0 Step -1
        Lines.AddRange(GenerateLines(Hierarchies(i)))
      Next
    End Sub

    Private Function GenerateLines(hierarchy As IHierarchy) As List(Of ILine)
      Dim result As New List(Of ILine)
      Dim count As Integer
      For Each SubCluster In hierarchy.Clusters
        Dim line As New Line
        For Each SubLeaf In SubCluster.Leaves
          Dim c As Color = SubCluster.Color
          Dim p As Color = Color.FromArgb(CInt(c.A / (hierarchy.Rank + 1.0F)), c.R, c.G,
c.B)
          line.Vertices.Add(New Vertex With {.Color = SubCluster.Color, .Position =
SubLeaf.Position, .Size = hierarchy.Rank + 1.0F})

        Next
        result.Add(line)
        count += line.Vertices.Count
      Next
```

```
        Debug.WriteLine(count)



    Return result
     End Function

End Class
```

SequenceAI.cs

```
using AutoPaint.Core;
using System;
using System.Collections.Generic;
using System.Numerics;
using System.Linq;
using System.Text;

namespace AutoPaint.Recognition
{
    /// <summary>
    /// </summary>
    public class FastAI : IDisposable
    {
        /// <summary>
        /// </summary>
        public List<ILine> Lines { get; set; } = new List<ILine>();
        /// <summary>
        /// </summary>
        public ScanMode ScanMode { get; set; } = ScanMode.Rect;
```

```
        /// <summary>


    /// </summary>
        public int DepthMax { get; set; } = 1000;
        /// <summary>
        /// </summary>
        public bool IsCheckAround { get; set; } = false;
        /// <summary>

        /// </summary>
        public int AroundLower { get; set; } = 3;
        /// <summary>
        /// </summary>
        public int AroundUpper { get; set; } = 7;

        private static readonly int[] OffsetX = new[] { -1, 0, 1, 1, 1, 0, -1, -1 };
        private static readonly int[] OffsetY = new[] { -1, -1, -1, 0, 1, 1, 1, 0 };
        private bool NewLine;

        public FastAI(int[,] bools, ScanMode mode = ScanMode.Rect, bool isCheckAround =
false)
        {
            this.ScanMode = mode;
            this.IsCheckAround = isCheckAround;
            CalculateSequence(bools);
        }
```

```
    /// <summary>


 /// </summary>
    private void CreateNewSequence()
    {
        Lines.Add(new Line());
    }
    /// <summary>
    /// </summary>
    private void AddPoint(Vector2 position)
    {
        Lines.Last().Vertices.Add(new Vertex() { Position = position, Size = 1 });
    }
    /// <summary>
    /// </summary>
    private void CalculateSequence(int[,] bools)
    {
        if (ScanMode == ScanMode.Rect)
            ScanRect(bools);
        else
            ScanCircle(bools);
    }
    /// <summary>
    /// </summary>
    private void ScanCircle(int[,] bools)
    {
        int xlength = bools.GetLength(0);
```

```
        int ylength = bools.GetLength(1);



    Vector2 center = new Vector2(System.Convert.ToSingle(xlength / (double)2),
System.Convert.ToSingle(ylength / (double)2));
        int radius = 0;
        var loopTo = System.Convert.ToInt32(Math.Sqrt(xlength * xlength + ylength *
ylength));
        for (radius = 0; radius <= loopTo; radius++)
        {
           for (float Theat = 0; Theat <= Math.PI * 2; Theat += 1 / (float)radius)
           {
              int dx = System.Convert.ToInt32(center.X + radius * Math.Cos(Theat));
              int dy = System.Convert.ToInt32(center.Y + radius * Math.Sin(Theat));
              if (!(dx >= 0 && dy >= 0 && dx < xlength && dy < ylength))
                 continue;
              if (bools[dx, dy] == 1)
              {
                 bools[dx, dy] = 0;
                 this.CreateNewSequence();
                 this.AddPoint(new Vector2(dx, dy));
                 MoveNext(bools, dx, dy, 0);
                 NewLine = true;
              }
           }
        }
     }
     /// <summary>
```

```
    /// </summary>


 private void ScanRect(int[,] BolArr)
    {
        int xCount = BolArr.GetUpperBound(0);
        int yCount = BolArr.GetUpperBound(1);
        var loopTo = xCount - 1;
        for (var i = 0; i <= loopTo; i++)
        {
            var loopTo1 = yCount - 1;
            for (var j = 0; j <= loopTo1; j++)
            {
                int dx = i;
                int dy = j;

 if (!(dx > 0 && dy > 0 && dx < xCount && dy < yCount))
                    continue;
                if (BolArr[dx, dy] == 1)
                {
                    BolArr[dx, dy] = 0;
                    this.CreateNewSequence();
                    this.AddPoint(new Vector2(dx, dy));
                    MoveNext(BolArr, dx, dy, 0);
                    NewLine = true;
                }
            }
        }
```

```
    }


/// <summary>
    /// </summary>
    private void MoveNext(int[,] bools, int x, int y, int depth)
    {
        if (depth > DepthMax)
            return;
        int xBound = bools.GetUpperBound(0);
        int yBound = bools.GetUpperBound(1);
        int dx, dy;
        if (IsCheckAround)
        {
            int around = GetAroundValue(bools, x, y);
            if (around >= AroundLower && around <= AroundUpper)
                return;
        }
        for (var i = 0; i <= 7; i++)
        {
            dx = x + OffsetX[i];
            dy = y + OffsetY[i];
            if (!(dx >= 0 && dy >= 0 && dx <= xBound && dy <= yBound))
                return;
            else if (bools[dx, dy] == 1)
            {
                bools[dx, dy] = 0;
                if (NewLine == true)
```

```
            {


    this.CreateNewSequence();
                this.AddPoint(new Vector2(dx, dy));
                NewLine = false;
            }
            else
                this.AddPoint(new Vector2(dx, dy));
            MoveNext(bools, dx, dy, depth + 1);
            NewLine = true;
        }
      }
    }
    /// <summary>
    /// </summary>
    private int GetAroundValue(int[,] bools, int x, int y)
    {
      int dx, dy, result = 0;

int xBound = bools.GetUpperBound(0);
      int yBound = bools.GetUpperBound(1);
      for (var i = 0; i <= 7; i++)
      {
        dx = x + OffsetX[i];
        dy = y + OffsetY[i];
        if (dx >= 0 && dy >= 0 && dx <= xBound && dy <= yBound)
        {
```

```
            if (bools[dx, dy] == 1)


        result += 1;
              }
           }
           return result;
        }

        void IDisposable.Dispose()
        {
           Lines.Clear();
        }
    }
}
```

BitmapHelper.cs

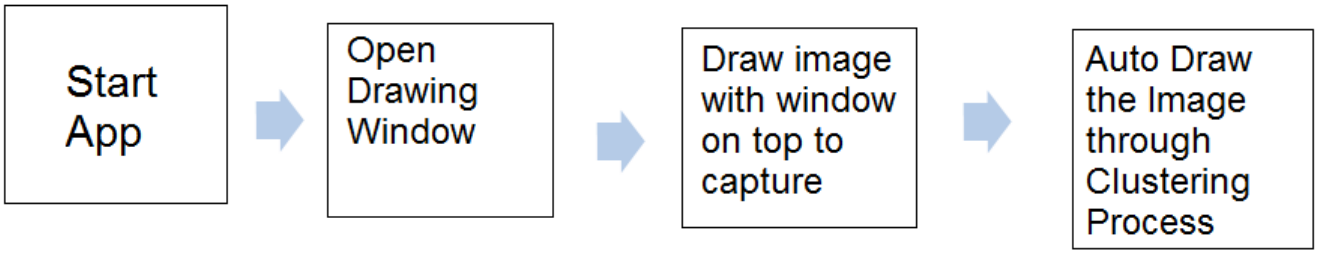
```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using AutoPaint.Core;

namespace AutoPaint.Utilities
{
   public class BitmapHelper
   {
      /// <summary>

      /// </summary>
      public static Bitmap GetScreenImage(Rectangle rect)
      {
         Bitmap result = new Bitmap(rect.Width, rect.Height);
         using (Graphics pg = Graphics.FromImage(result))
         {
            pg.CopyFromScreen(rect.X, rect.Y, 0, 0, new Size(rect.Width, rect.Height));
         }
         return result;
      }

      /// <summary>
```

```
    /// </summary>
    public static Bitmap GetTextImage(string text, Font font, int width, int height)
    {
        Bitmap result = new Bitmap(width, height);
        using (var pg = Graphics.FromImage(result))
        {
            pg.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias; // 抗
锯齿
            pg.DrawString(text, font, Brushes.Black, 0, 0);
        }
        return result;
    }

    /// <summary>

    /// </summary>
    public static PixelData GetPixelDataFromBitmap(Bitmap bmp)
    {
        Color[,] colors = new Color[bmp.Width - 1 + 1, bmp.Height - 1 + 1];
        var loopTo = bmp.Width - 1;
        for (var i = 0; i <= loopTo; i++)
        {
            var loopTo1 = bmp.Height - 1;
            for (var j = 0; j <= loopTo1; j++)
                colors[i, j] = bmp.GetPixel(i, j);
        }
```

```
    return PixelData.CreateFromColors(colors);
        }
        /// <summary>

        /// </summary>
        public static Bitmap GetBitmapFromPixelData(PixelData pixels)
        {
            return GetBitmapFromColors(pixels.Colors);
        }

        /// <summary>

        /// </summary>
        public static Color[,] GetColorsFromBitmap(Bitmap bmp)
        {
            Color[,] result = new Color[bmp.Width - 1 + 1, bmp.Height - 1 + 1];
            var loopTo = bmp.Width - 1;
            for (var i = 0; i <= loopTo; i++)
            {
                var loopTo1 = bmp.Height - 1;
                for (var j = 0; j <= loopTo1; j++)
                    result[i, j] = bmp.GetPixel(i, j);
            }
            return result;
        }
```

```
 /// <summary>

      /// </summary>
      public static Bitmap GetBitmapFromColors(Color[,] colors)
      {
         int w = colors.GetUpperBound(0) + 1;
         int h = colors.GetUpperBound(1) + 1;
         Bitmap result = new Bitmap(w, h);
         var loopTo = w - 1;
         for (var i = 0; i <= loopTo; i++)
         {
            var loopTo1 = h - 1;
            for (var j = 0; j <= loopTo1; j++)
               result.SetPixel(i, j, colors[i, j]);
         }
         return result;
      }
   }
}
```

Process Involved



Methodology Used For Testing

There are different methods that can be used for software testing. This chapter briefly describes the methods available.

Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

Advantages	Disadvantages
Well suited and efficient for large code segments.	Limited coverage, since only a selected number of test scenarios is actually performed.
Code access is not required.	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
Clearly separates user's perspective from the developer's perspective through visibly defined roles.	Blind coverage, since the tester cannot target specific code segments or errorprone areas.
Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.	The test cases are difficult to design.

White-box testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**.

In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of white-box testing.

Advantages	Disadvantages
As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.	Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
It helps in optimizing the code.	Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested.
Extra lines of code can be removed which can bring in hidden defects.	It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.

Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	

Grey-box testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Advantages	Disadvantages
Offers combined benefits of black-box and white-box testing wherever possible.	Since the access to source code is not available, the ability to go over the code and test coverage is limited.
Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.	The tests can be redundant if the software designer has already run a test case.
Based on the limited information available, a grey-box tester can design excellent test scenarios especially around communication protocols and data type handling.	Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested.
The test is done from the point of view of the user and not the designer.	

A Comparison of Testing Methods

The following table lists the points that differentiate black-box testing, grey-box testing, and white-box testing.

Black-Box Testing	Grey-Box Testing	White-Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
It is exhaustive and the least time-consuming.	Partly time-consuming and exhaustive.	The most exhaustive and time-consuming type of

		testing.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

FUTURE SCOPE

1) Improvement in clustering algorithm to create more accurate drawings.

2) Future enhancements like additional set of drawing tools can be added.

3) Multi Tool drawing with customized colour pallete.

4) Increasing the processing power of the algorithm to incorporate thousands of pictures, so that it is able to learn about certain aspects of the world and make decisions in a way that resembles the human way of thinking.

5) Features like non-deterministic and semi-autonomous behavior can be implemented.

6) Multiple Options for choosing a specific algorithm to perform the drawing.

CONCLUSION

In recent years, the field of artificial intelligence has experienced a boom.

By processing thousands of pictures, a computer is able to learn about certain aspects of the world and make decisions in a way that resembles the human way of thinking.

Machine learning is ubiquitous and comes with new ways to program and interact with machines.

To me, this was an opportunity to teach machines how to draw by observing images and sketches of real objectS so they can be participants in the drawing process and not just mere transcribers.

I believe this approach will enable a new branch of creative mediation with machines that can develop their own aesthetics and gather knowledge from previously seen images and representations of the world.

Bibliography

Books Reference:

1. Vb.Net Wrox Publications.

2. Unified Modeling Language Grady Booch

3. Software Engineering Fairely, Pressman

4. Roger S Pressmen, Software Engineering A Practitioner's Approach, McGraw Hill Publications, Fifth Edition

5. Herbert Schildt, the Complete Reference Java 2.0, Tata McGraw Hills, Fifth Edition

Website Reference:

http://www.msdn.microsoft.com/vbasic

http:// www.vbdotnetheaven.com

http:// www.startvbdotnet.com

http:// www.devarticles.com/c/b/VB.Net

http:// www.accelebrate.com/vb.net/default.htm

http:// www.programmersheaven.com/2/VB-NET-School

http:// www.harding.edu/fmccown/vbnet_csharp_comparison.html

http:// www.a1vbcode.com