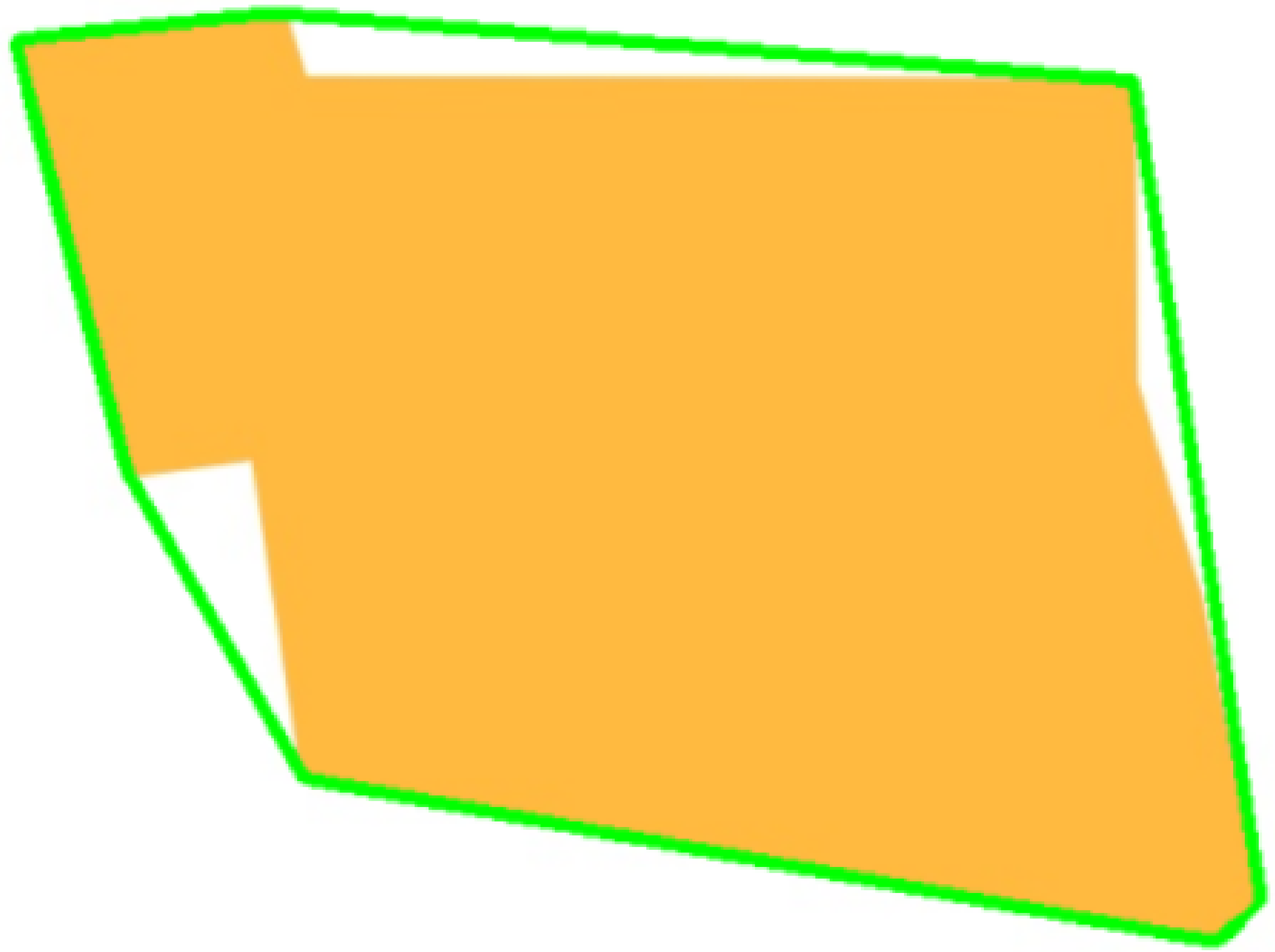
DS 203

ARCHITECTURAL DESIGN OPTIMIZATION

ABHISHEK KUMAR (22B2210)
BHIRUD RAJ MANOHAR (22B2214)
HARDIK KHARIWAL (22B3954)
JATIN GUPTA (22B3967)

Executive Overview:

- Problem Overview
- Key findings and insights obtained from the analysis.
- Procedure to solve each problem.
- Learnings and Experiences
- Conclusion



DS 203

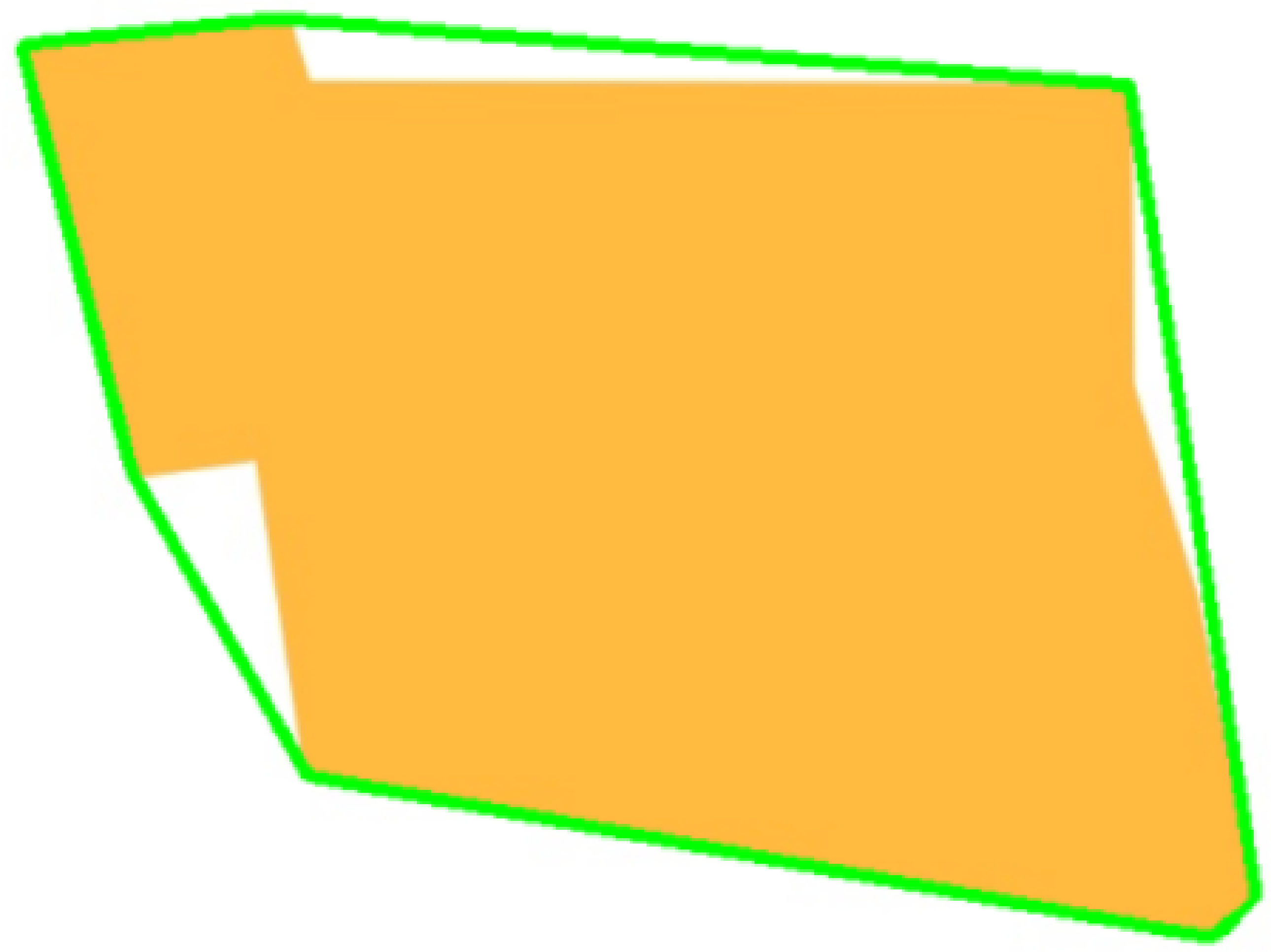
About The Project

PROBLEM OVERVIEW

Given the 1183 bitmaps of the building views, we aim to group those bitmaps into families based on their shape, classify them based on their complexity and predict design family for a specific layout.

Additionally, we seek to accelerate the layout design by obtaining relevant prior layouts based on specific parameters.

In short, we aim to streamline the process and improve insights into the designs and respond swiftly to the company requirements.

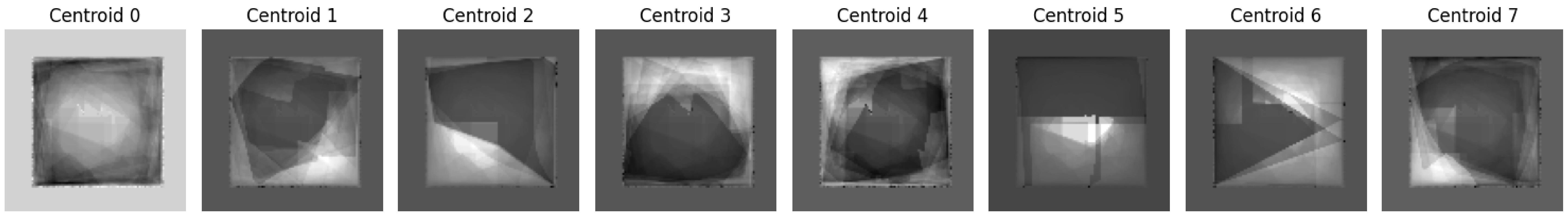


KEY FINDINGS AND INSIGHTS OBTAINED FROM THE ANALYSIS.

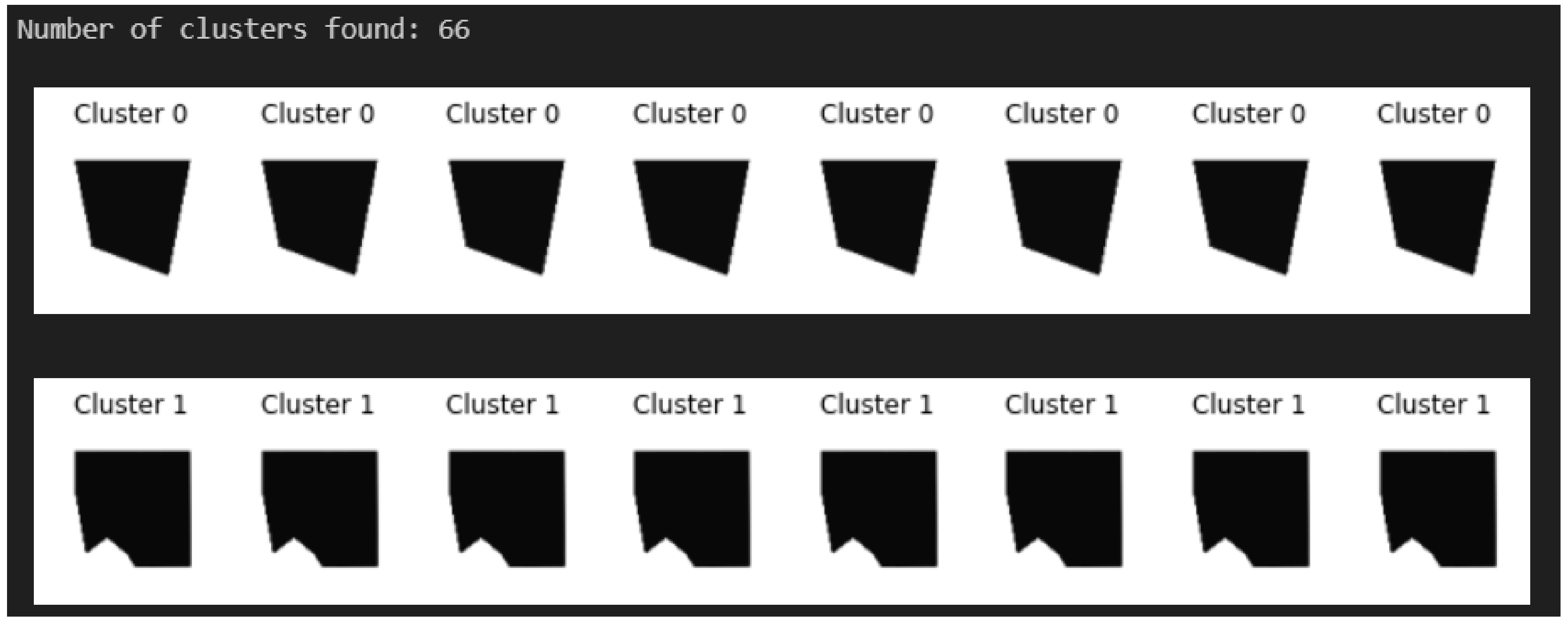
Shape-Based Clustering

Outputs

K-Means



DBSCAN

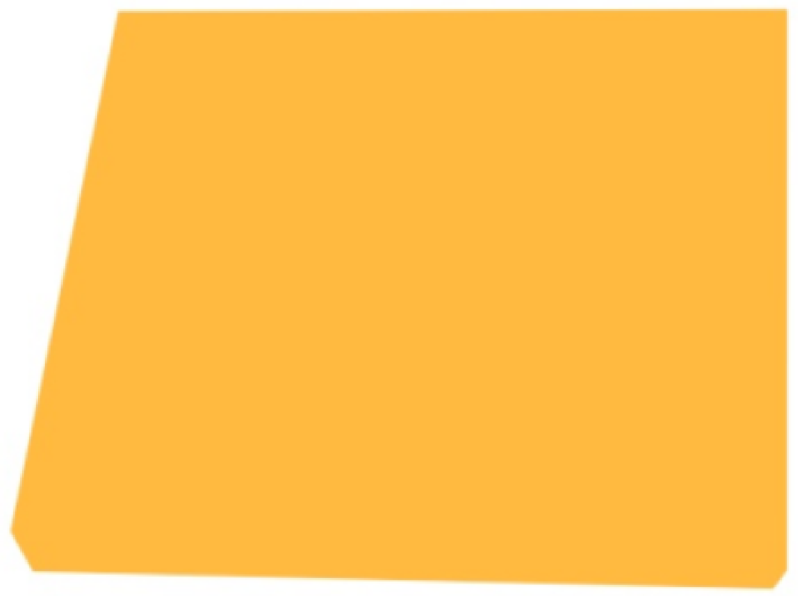


```
Number of clusters found: 66
```

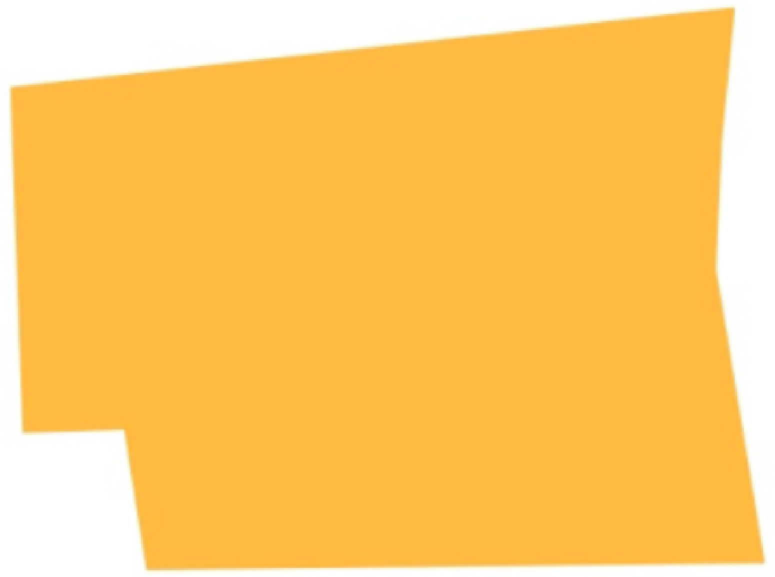
Complexity Classification

Outputs

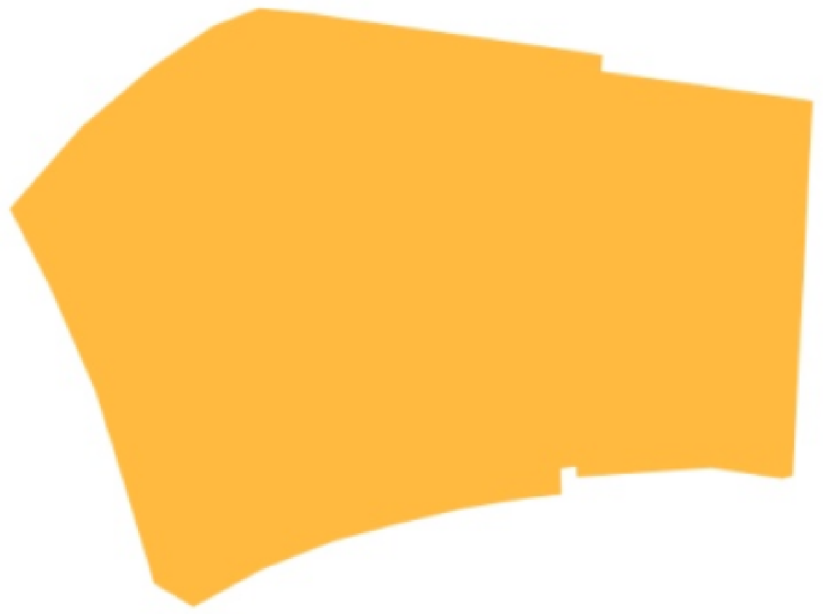
Complexity: 123.50
Category: Low



Complexity: 214.50
Category: Medium



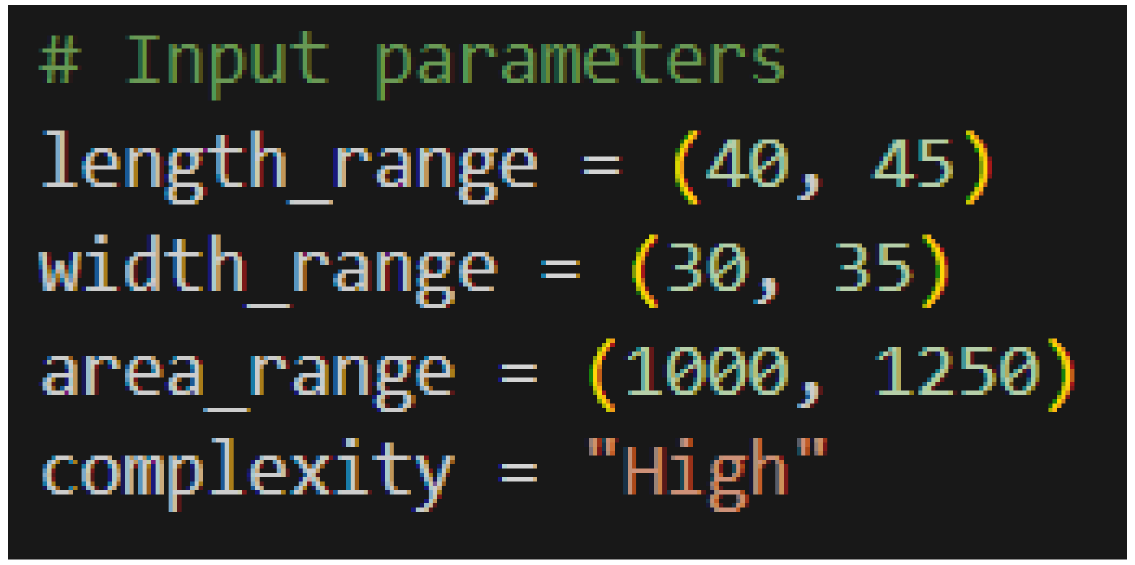
Complexity: 391.00
Category: High



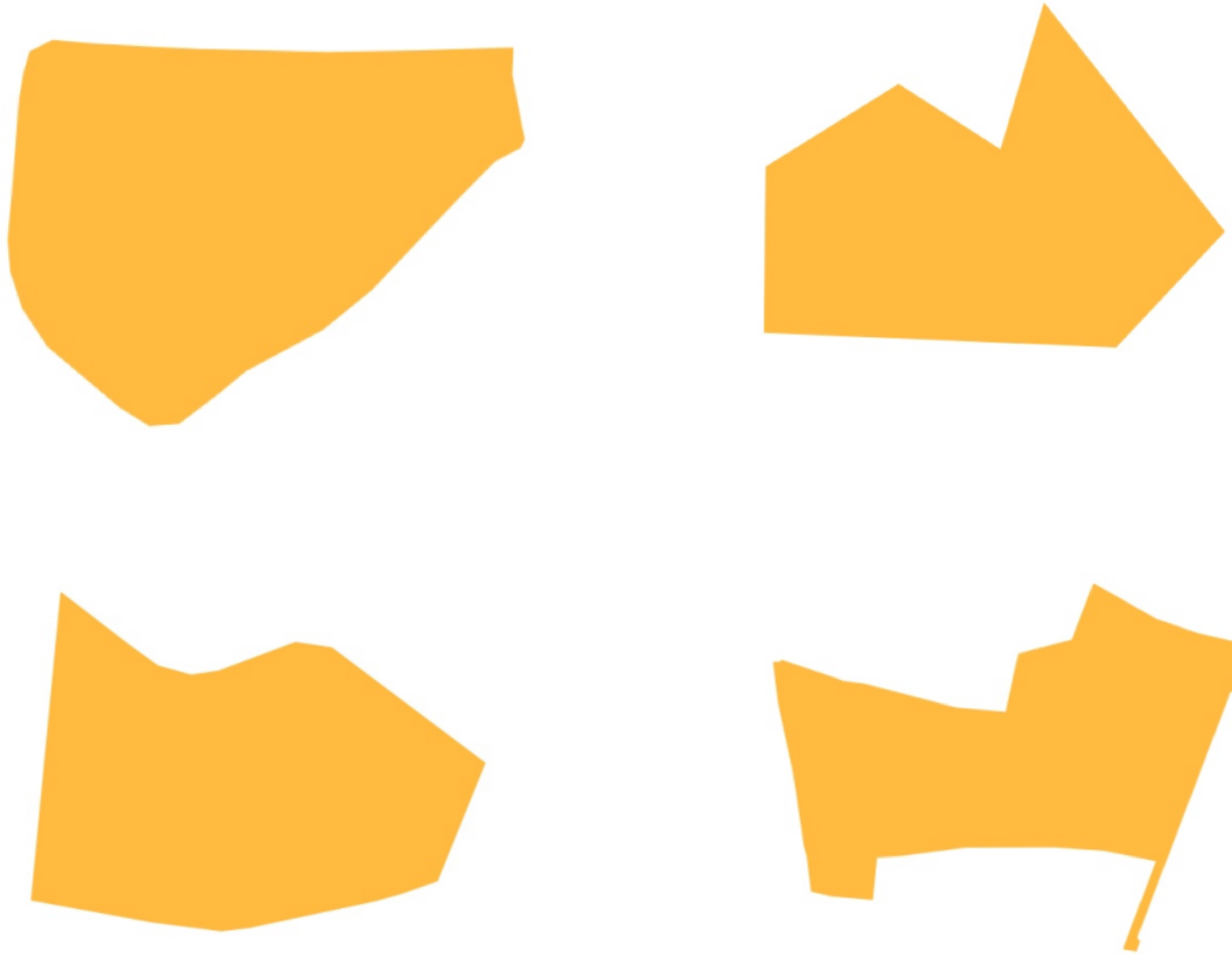
Layout Prediction

Outputs

input



```
# Input parameters
length_range = (40, 45)
width_range = (30, 35)
area_range = (1000, 1250)
complexity = "High"
```



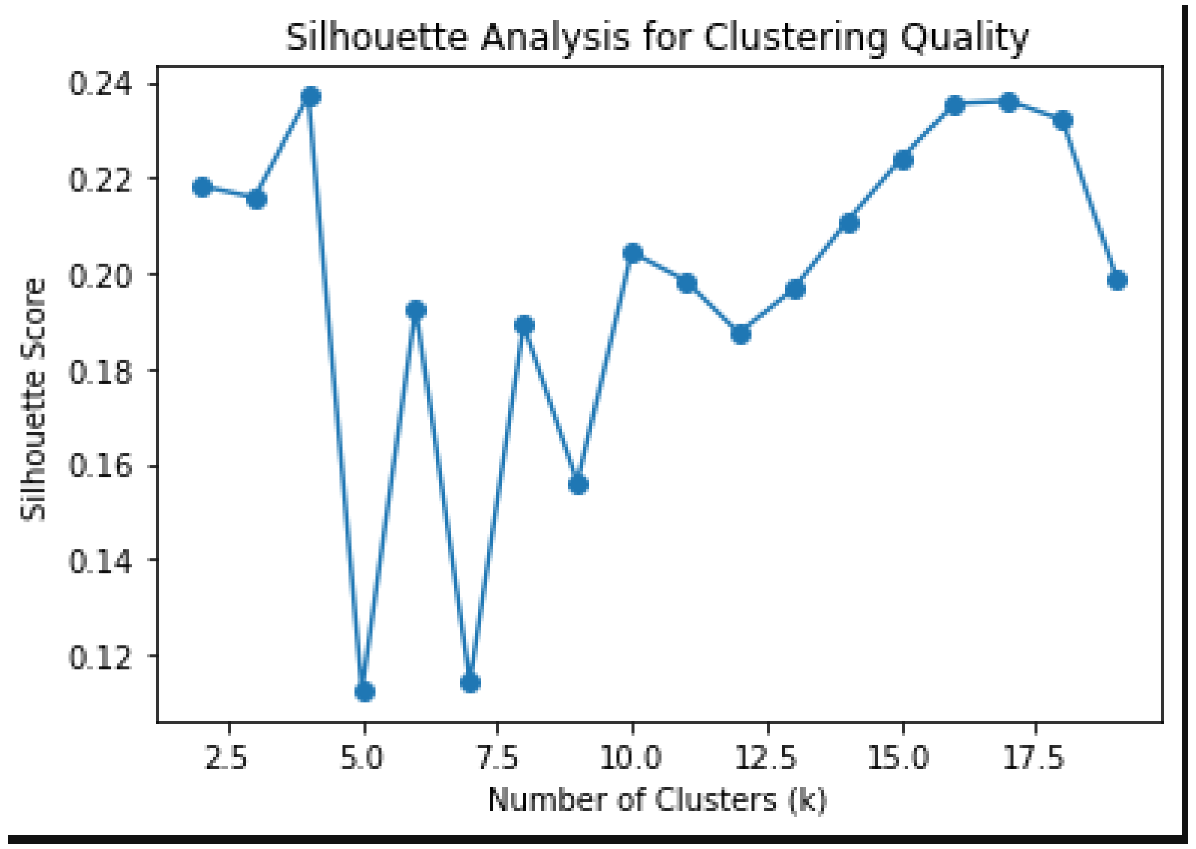
PROCEDURE TO SOLVE EACH PROBLEM.

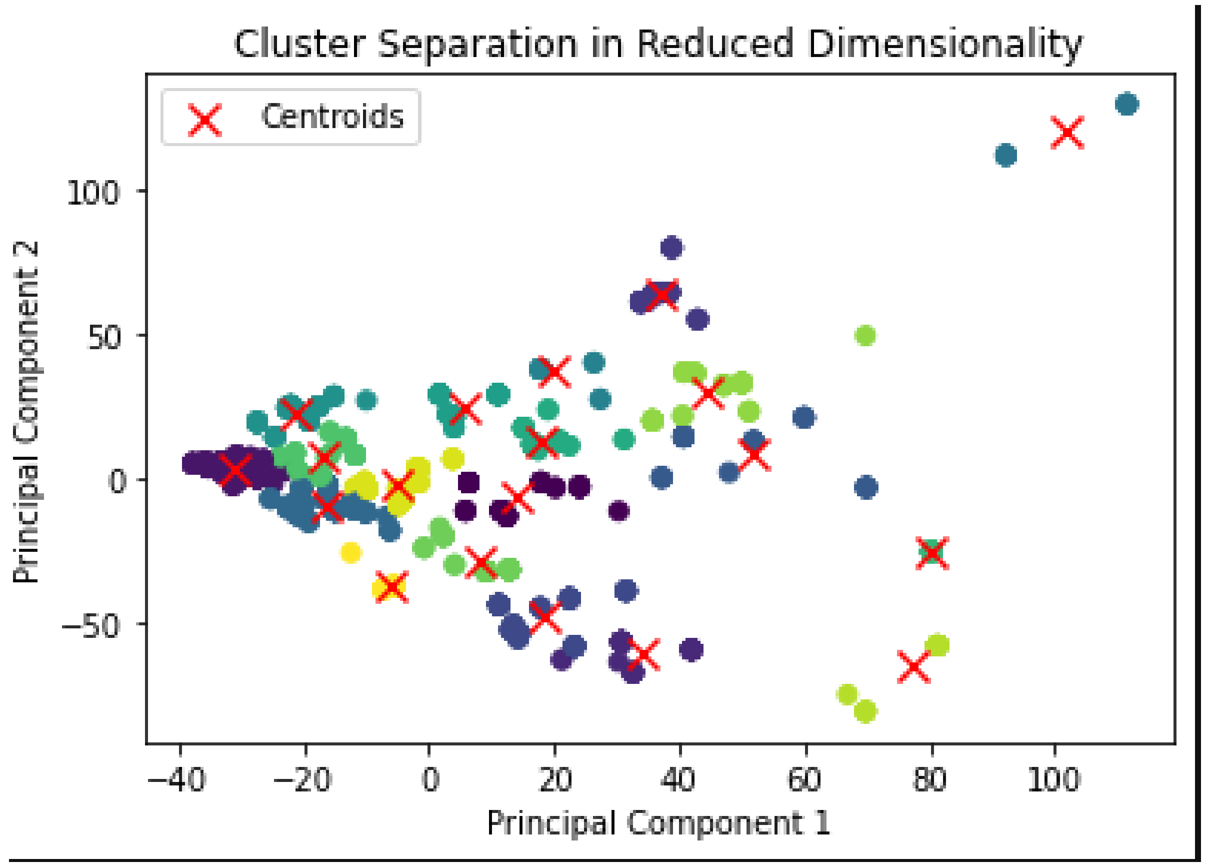
DS 203

Shape-Based Clustering

K-Means Clustering (Approach-1):

- Used the **K-means clustering** method to group images by their shapes and complexities, convert images into grayscale extract features from them, and make sure they're consistent.
- Then, it uses K-means with different cluster sizes to group similar layouts based on their features.
- Visualizing the clusters and centroids of the images which shows shape variations and complexity levels for different clusters
- Calculates silhouette scores for varying cluster sizes using KMeans. Plot silhouette scores against the number of clusters for visualization, optimal cluster selection based on silhouette score peaks comes for some **14**, **15, 16 clusters**.
- Also applies PCA to reduce feature dimensions to 2 for visualization. Uses KMeans clustering on reduced features to identify clusters and Plot clusters and centroids in reduced 2D space for cluster separation analysis.

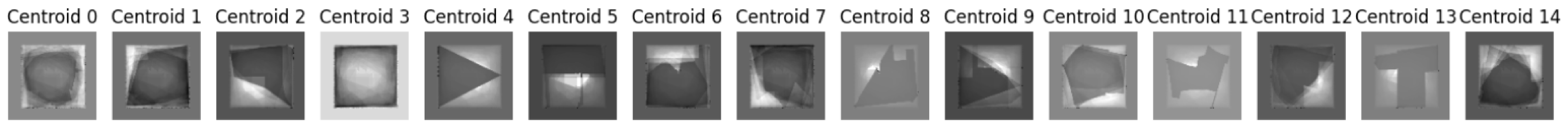




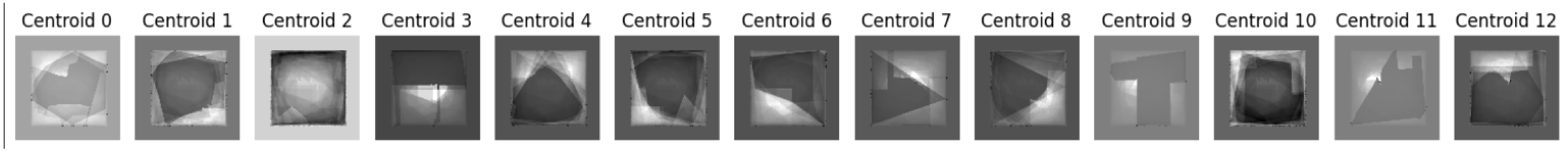
Shape-Based Clustering

K-Means Clustering (Approach-1):

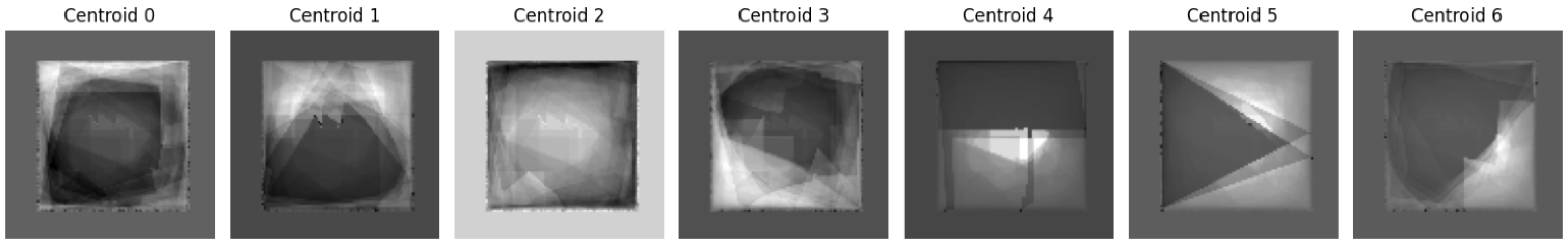
For_no_of _cluster=15



For_no_of _cluster=13



For_no_of _cluster=7



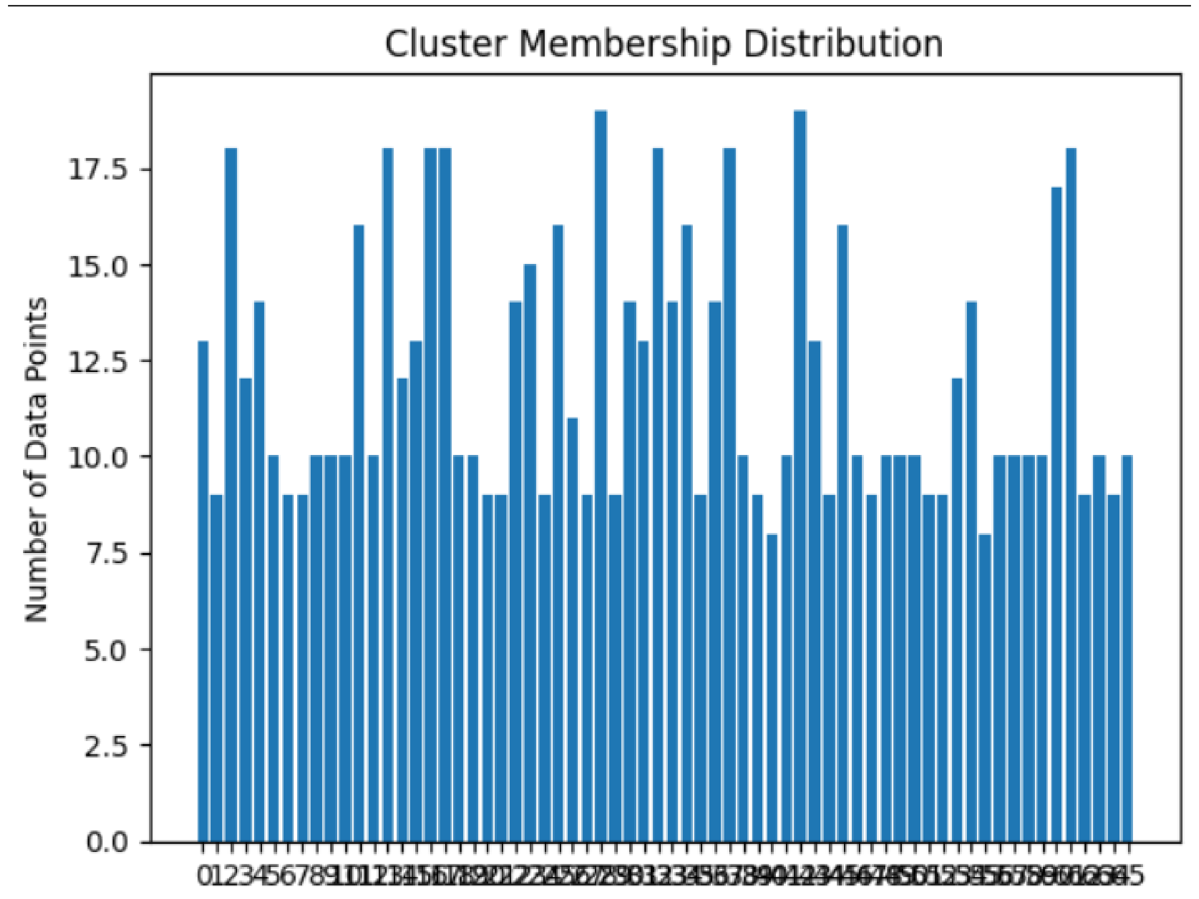
Shape-Based Clustering

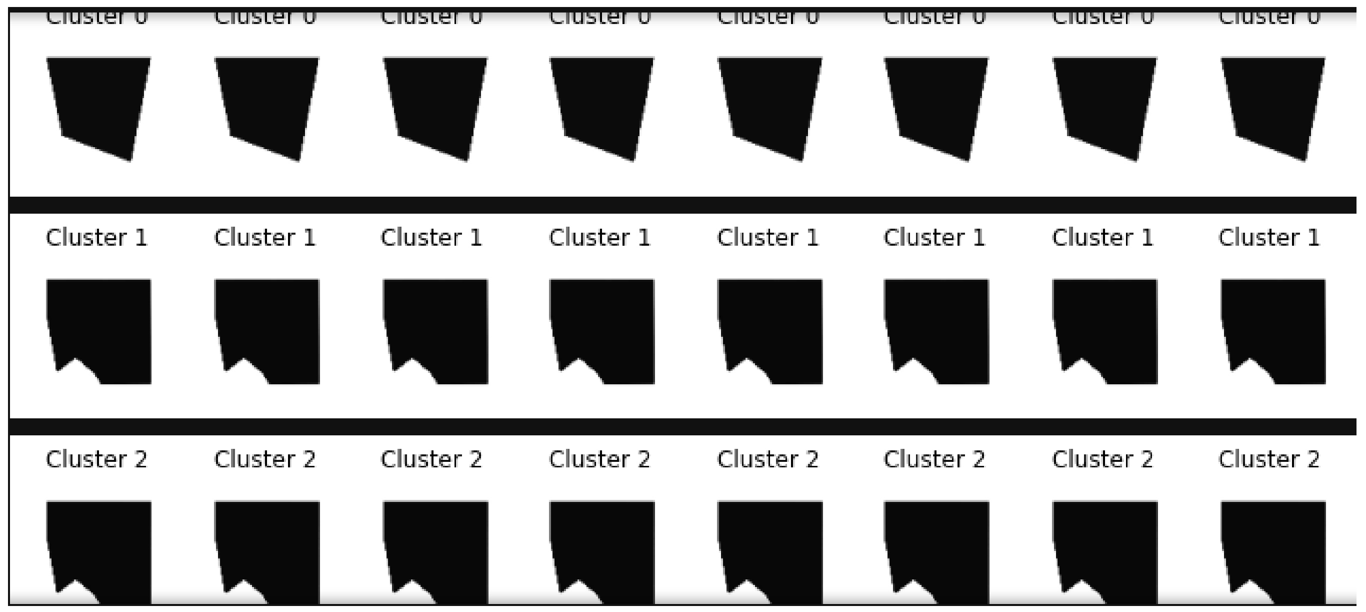
DBSCAN (Approach -2

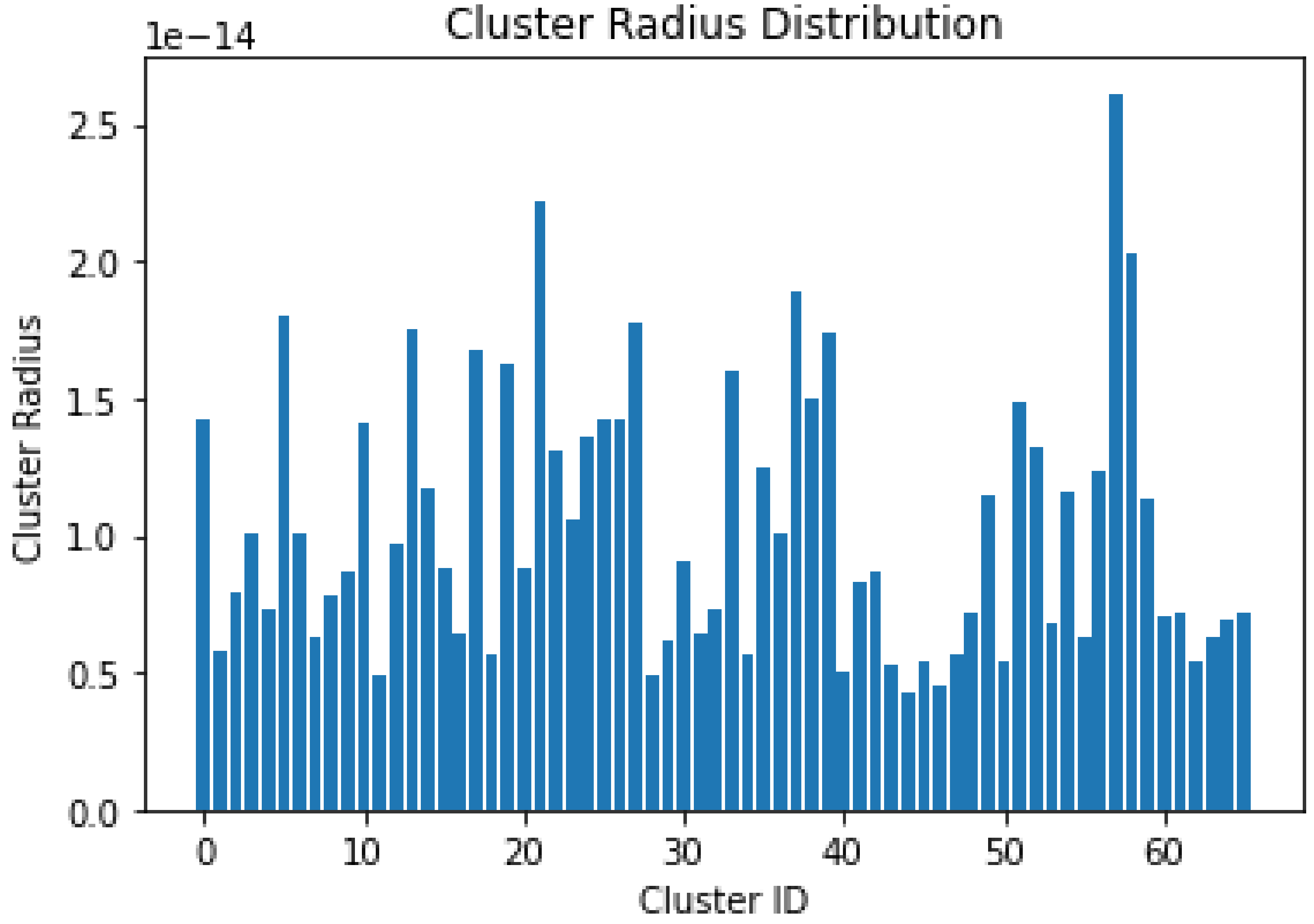
- DBSCAN creates group building layouts by structural similarities, and forms clusters based on density, revealing various shapes and complexities and it can adjust its cluster size based on density.

- It ends up with **no_of_cluster=66**, and all the clusters have similar images.

- Also Calculated the cluster radius for each cluster in a dataset, It computes the maximum distance from each cluster's center to its points,







Complexity Classification

- To distinguish the images based on the amount of yellow regions, we converted the image to grayscale and applied a Gaussian blur to reduce noise and then finally creating a binary mask for the yellow regions.

- Using this mask, we found the contours representing these regions and calculated their **total area** which is one of the **features** we used in the further analysis.

- Following the above calculations we also calculated the number of **edge pixels** and number of **vertices**. these are also the **features** used for further evaluation.

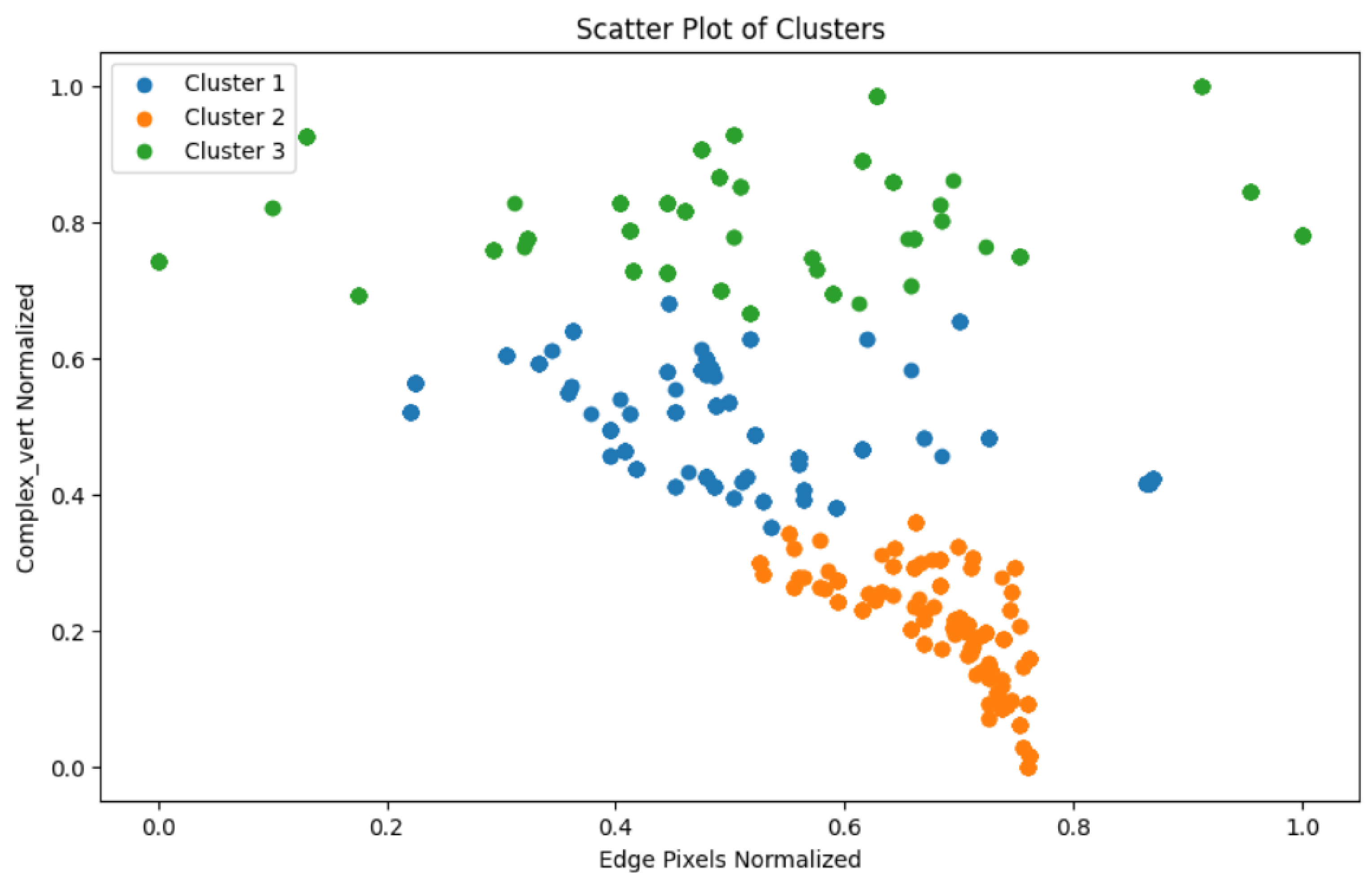
- After extracting these features for each image, we then normalized the value to ensure the code normalizes the values to ensure they fall within a consistent range (0 to 1) which in turn will result in improved performance.

- We then Cluster the images using K-means based on the normalized features, each image is assigned to a cluster based on the total area, edge pixels and number of vertices. We also create directories for each cluster and organize the images accordingly.

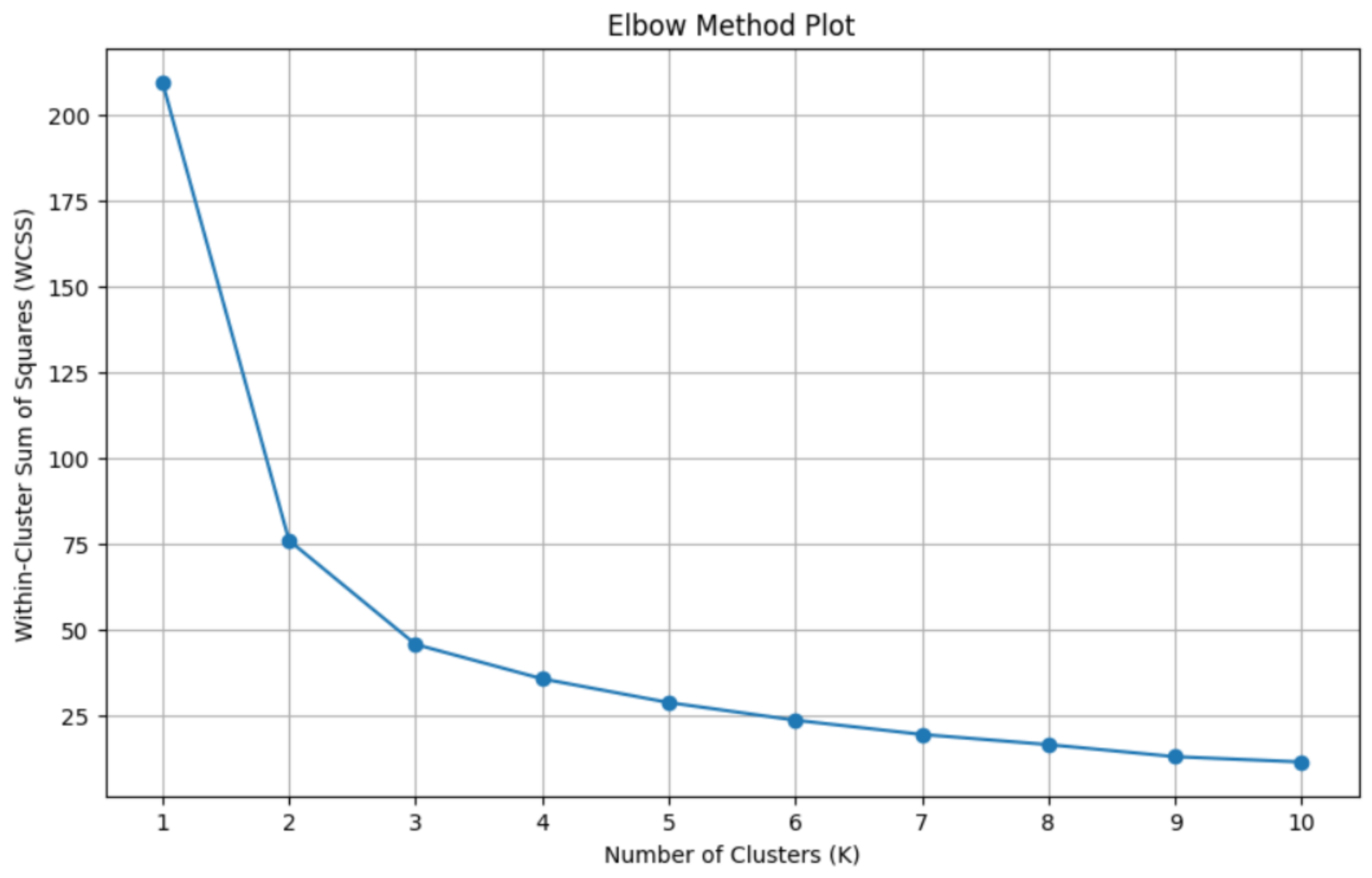
- Finally, we classify the clusters into complexity of three categories: low, medium, or high, based on predefined thresholds.

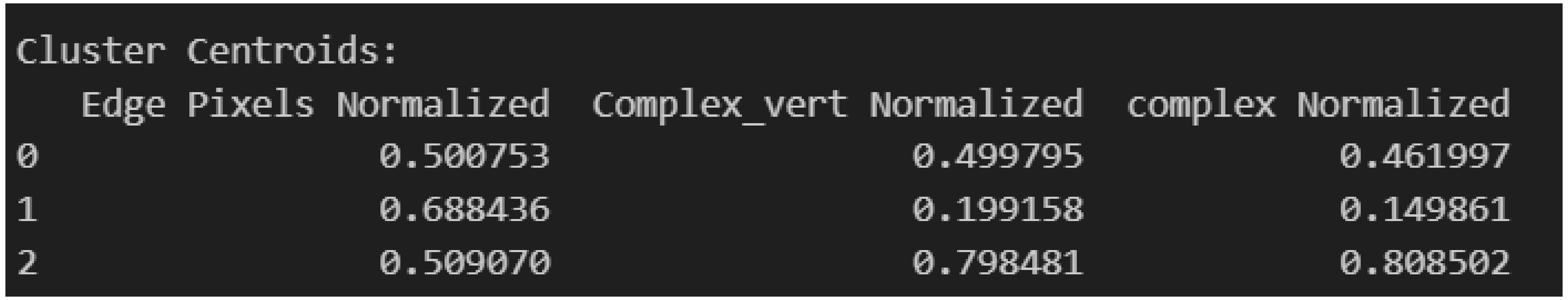
Complexity Classification (Plots)

DS203



Elbow is somewhat created at **no_of_cluster=3**, so taking 3 clusters will not be a bad idea

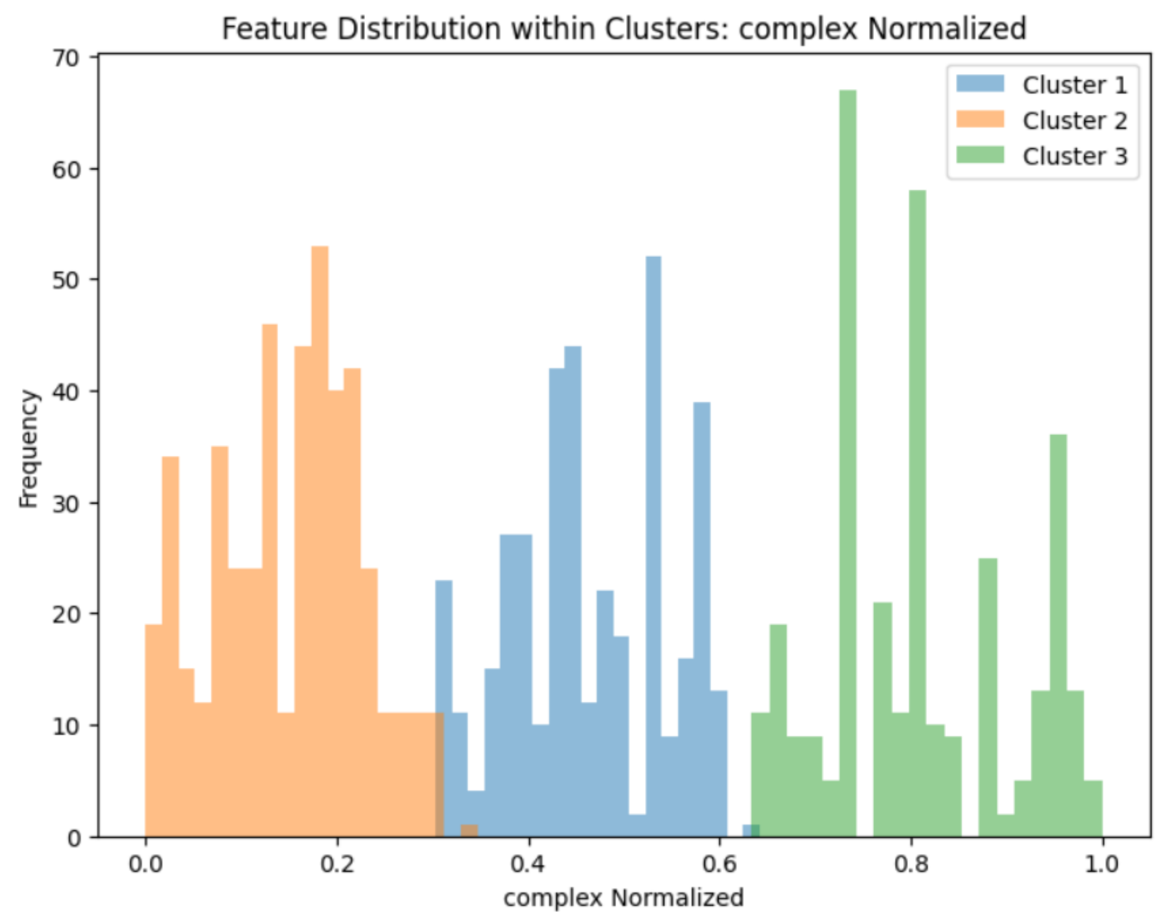




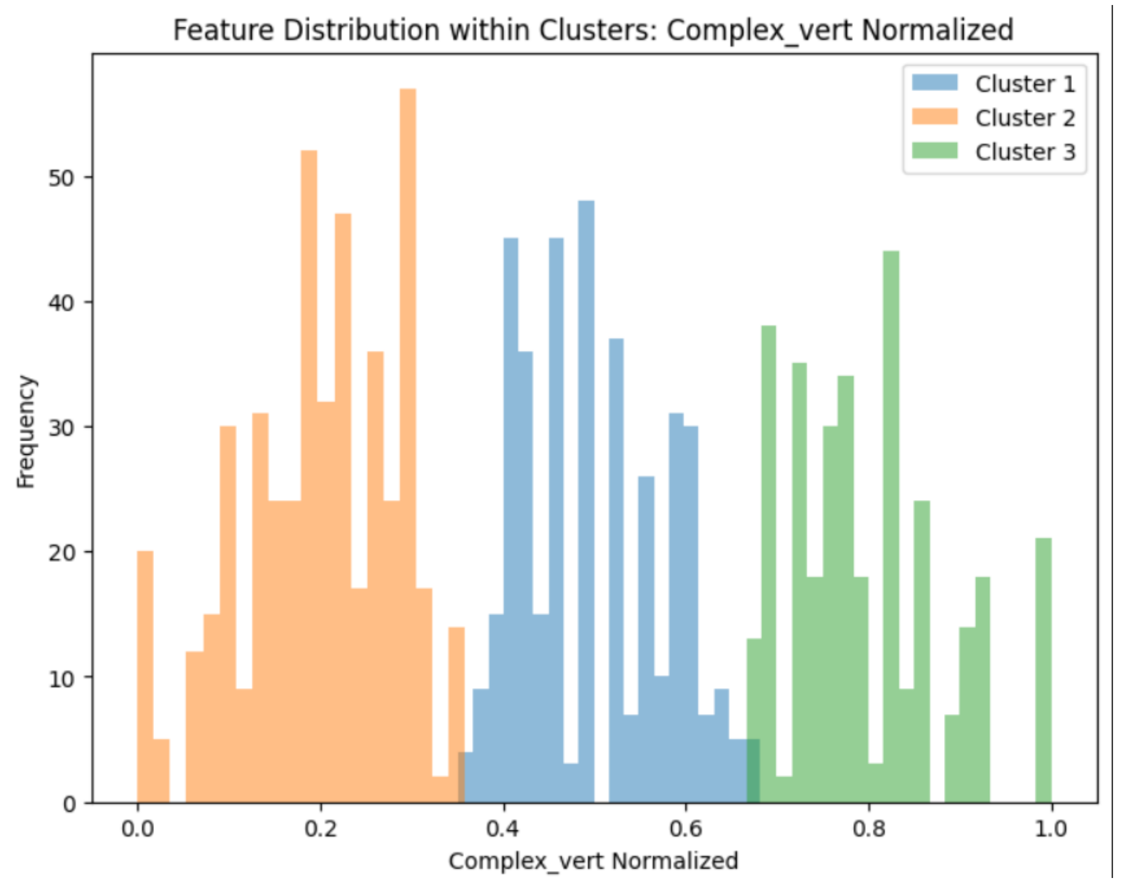
```
Cluster Centroids:
   Edge Pixels Normalized  Complex_vert Normalized  complex Normalized
0                0.500753                 0.499795            0.461997
1                0.688436                 0.199158            0.149861
2                0.509070                 0.798481            0.808502
```

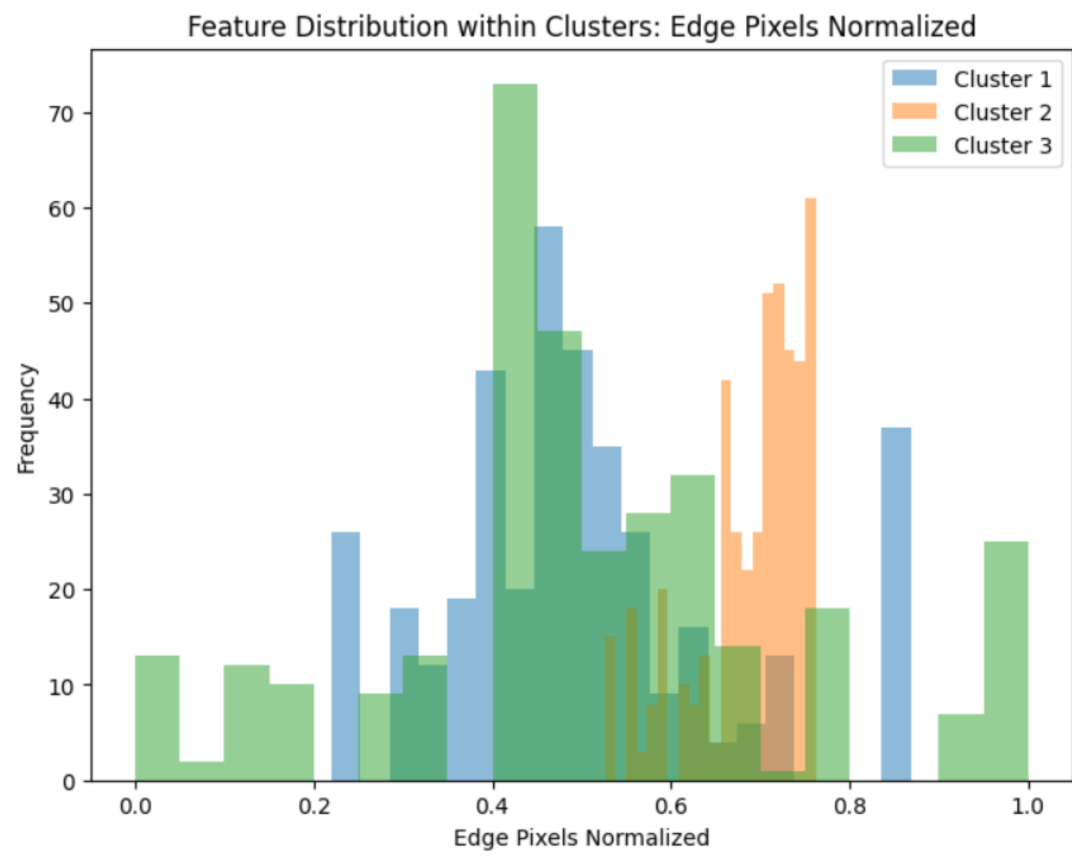
Complexity Classification (Plots)

DS203



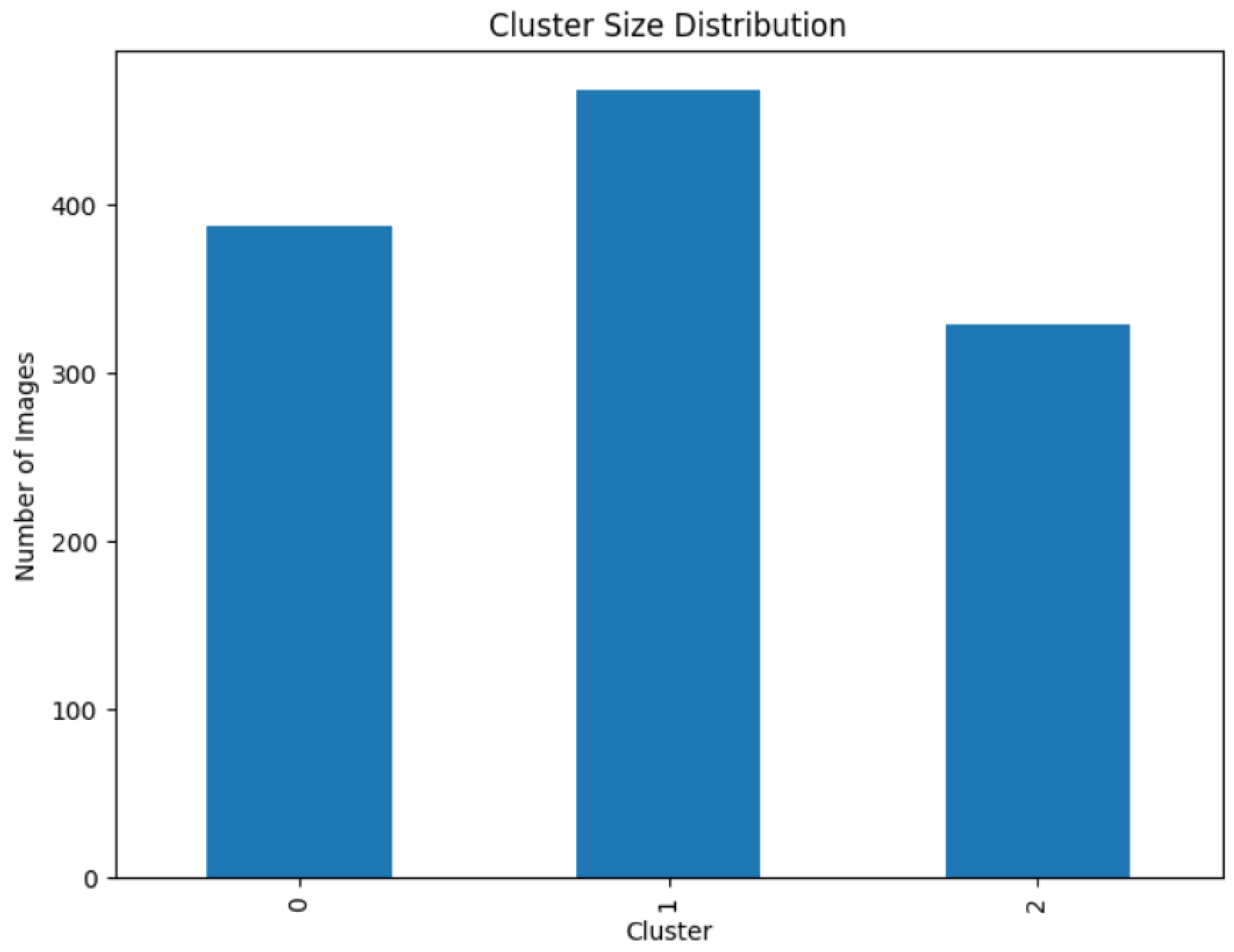
Complexity using **Area** and **Vertices** are very good features for classification but for edge pixels there will be no harm to drop it





Complexity Classification

Almost **equal**
number of images
in each cluster



Layout Prediction

- For each image, we convert it to the HSV color space and define a threshold for yellow regions. Using contour detection, the function identifies these regions and proceeds to calculate either the minimum area **bounding rectangle** (for bounding boxes) or the **convex hull** (for bounding polygons).

- Calculate the **Area** for the convex hull bounding image (to be more accurate) and the **Length**, **Width** for the bounding rectangle box and to extract parameters, and we also took shape(parameter/area) and area **complexity**(from the previous problem) are two other parameter.

- Then this data is stored in a DataFrame with the image labels. Now customer will specify the parameters of the building architecture, program will filter the DataFrame based on the given conditions (range of length, width, area, and complexity).

- Retrieve the image label names from the filtered DataFrame, Iterate through the folder containing the images that match the image names from the DataFrame, and display the images that meet the specified conditions.

Image with Bounding Box

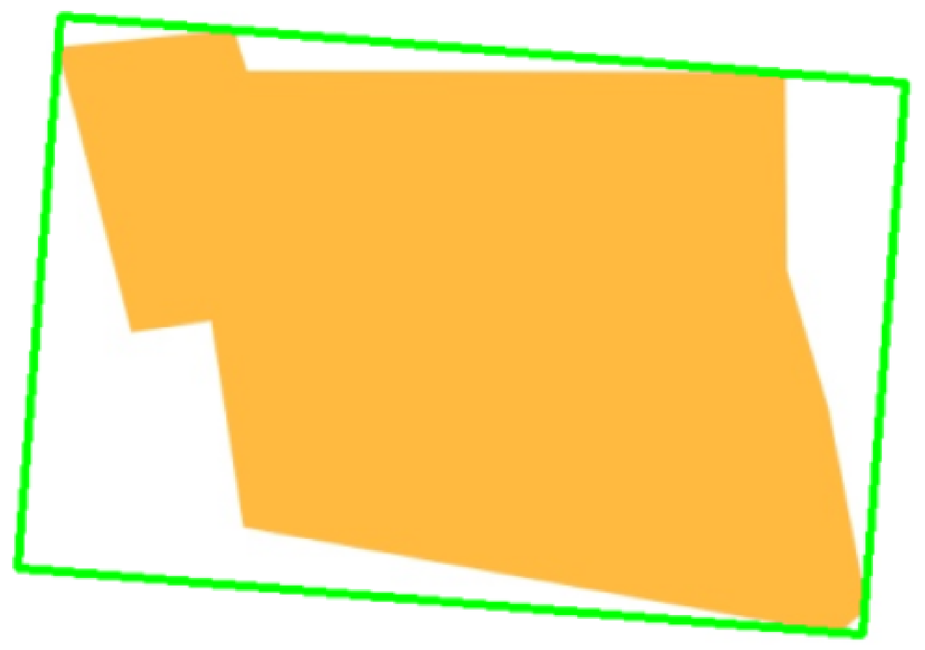
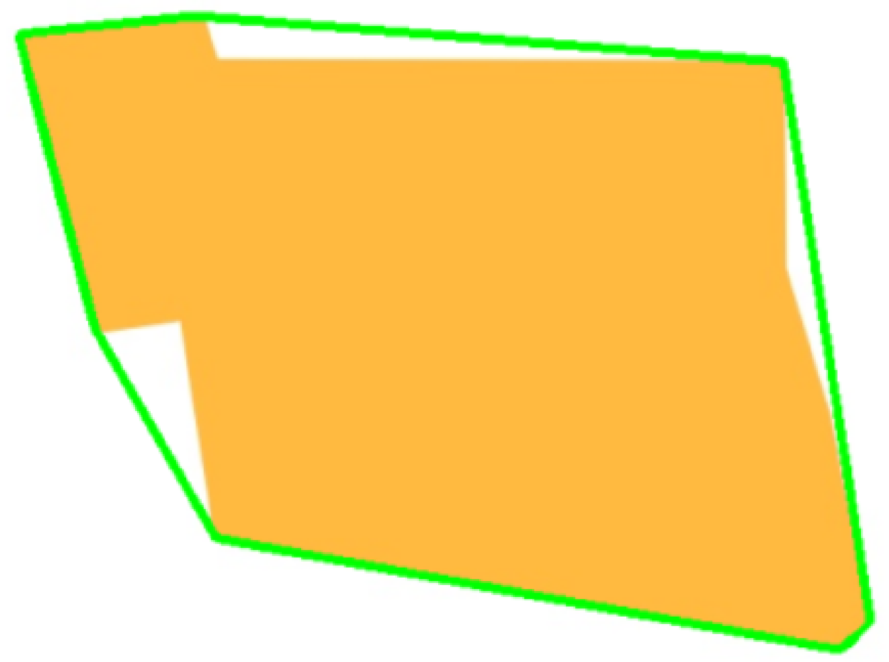
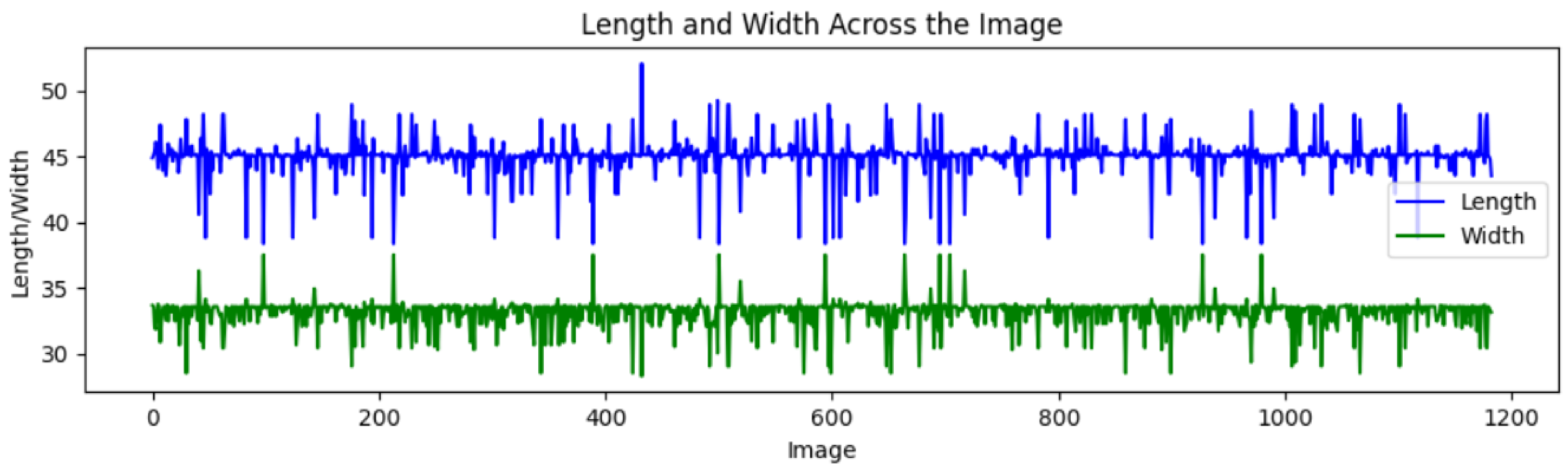


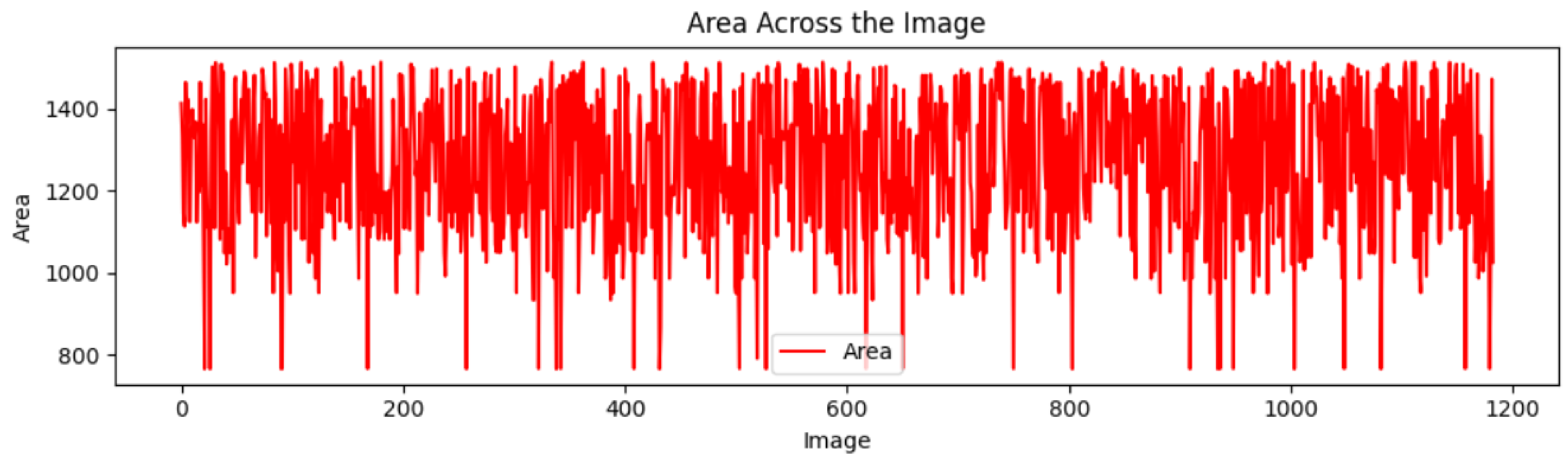
Image with Bounding Polygon



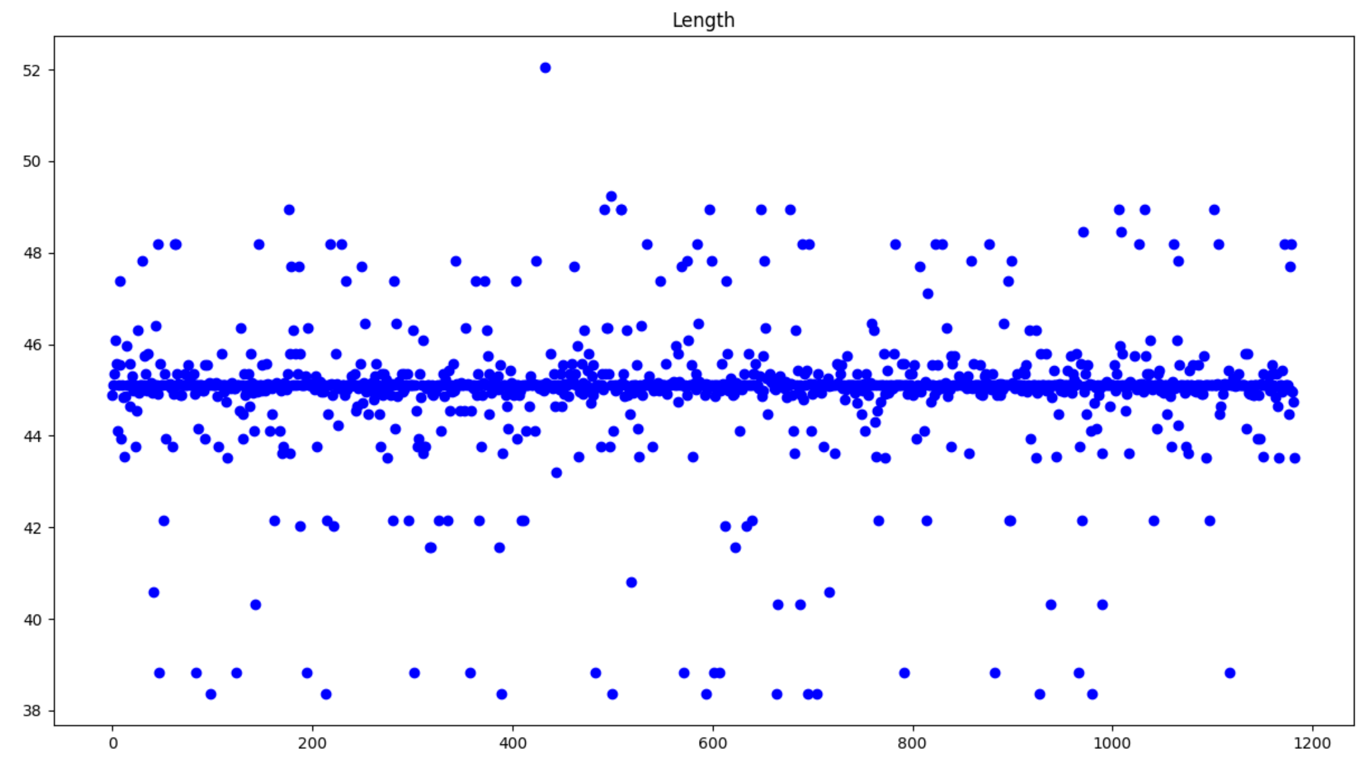
Layout Prediction (Plots)

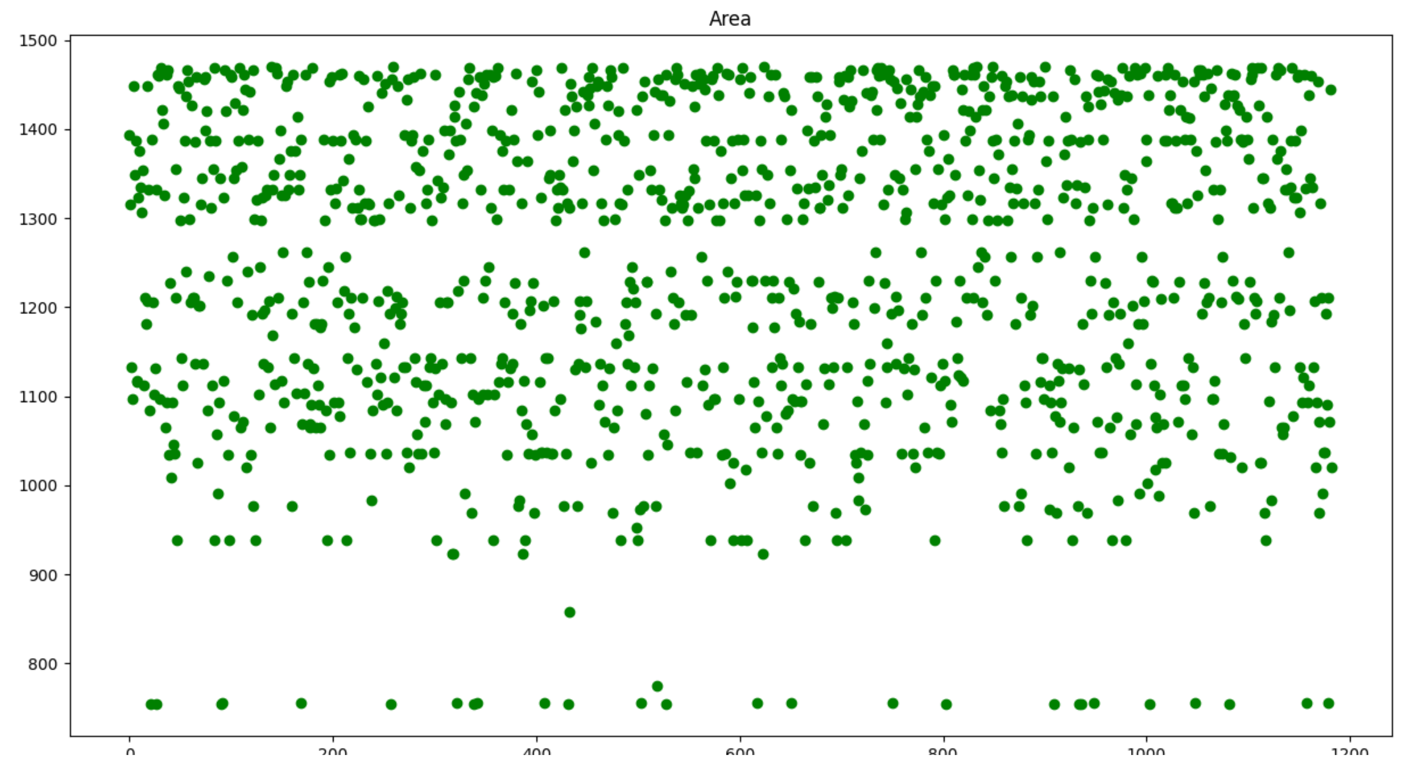
Ranges of value that Length ,Widrth and Area can take

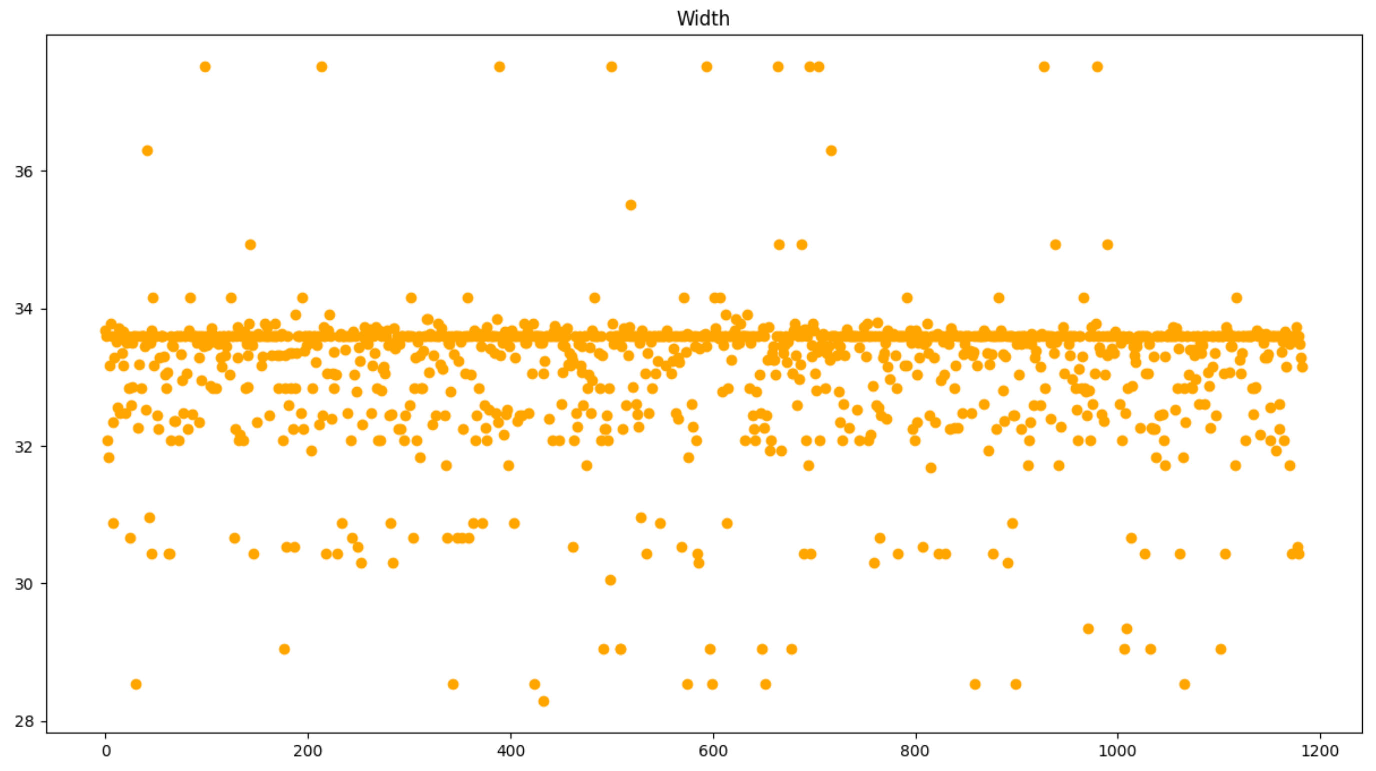


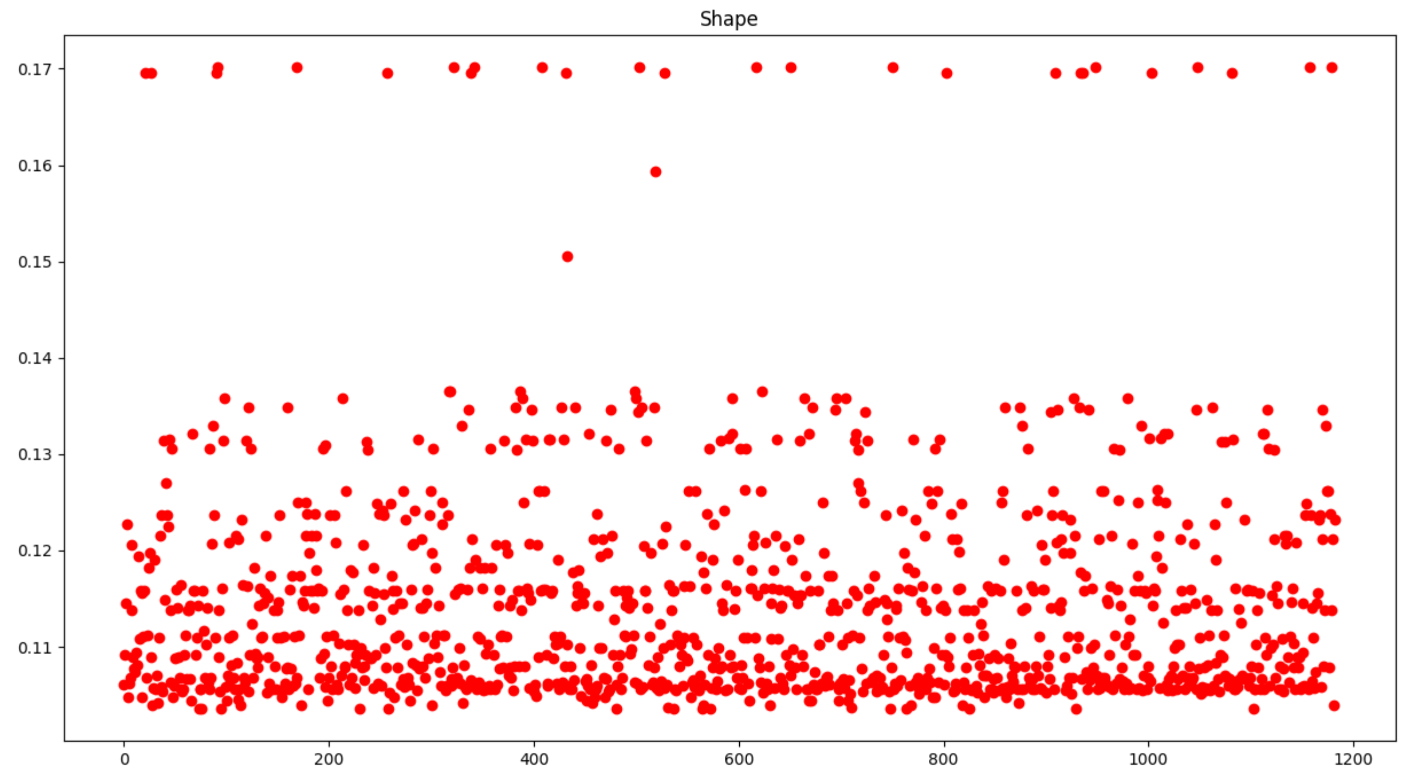


Layout Prediction (Plots)



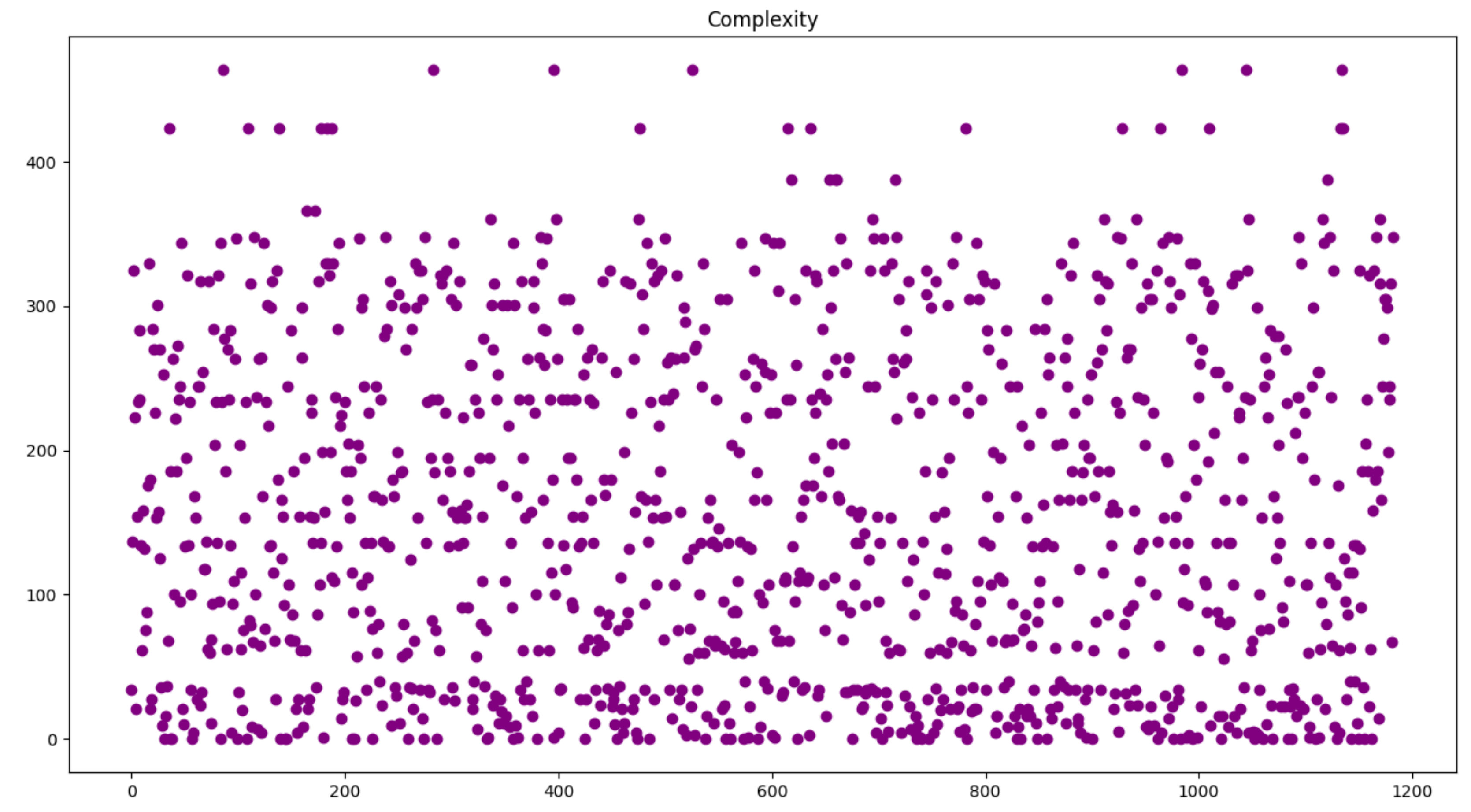


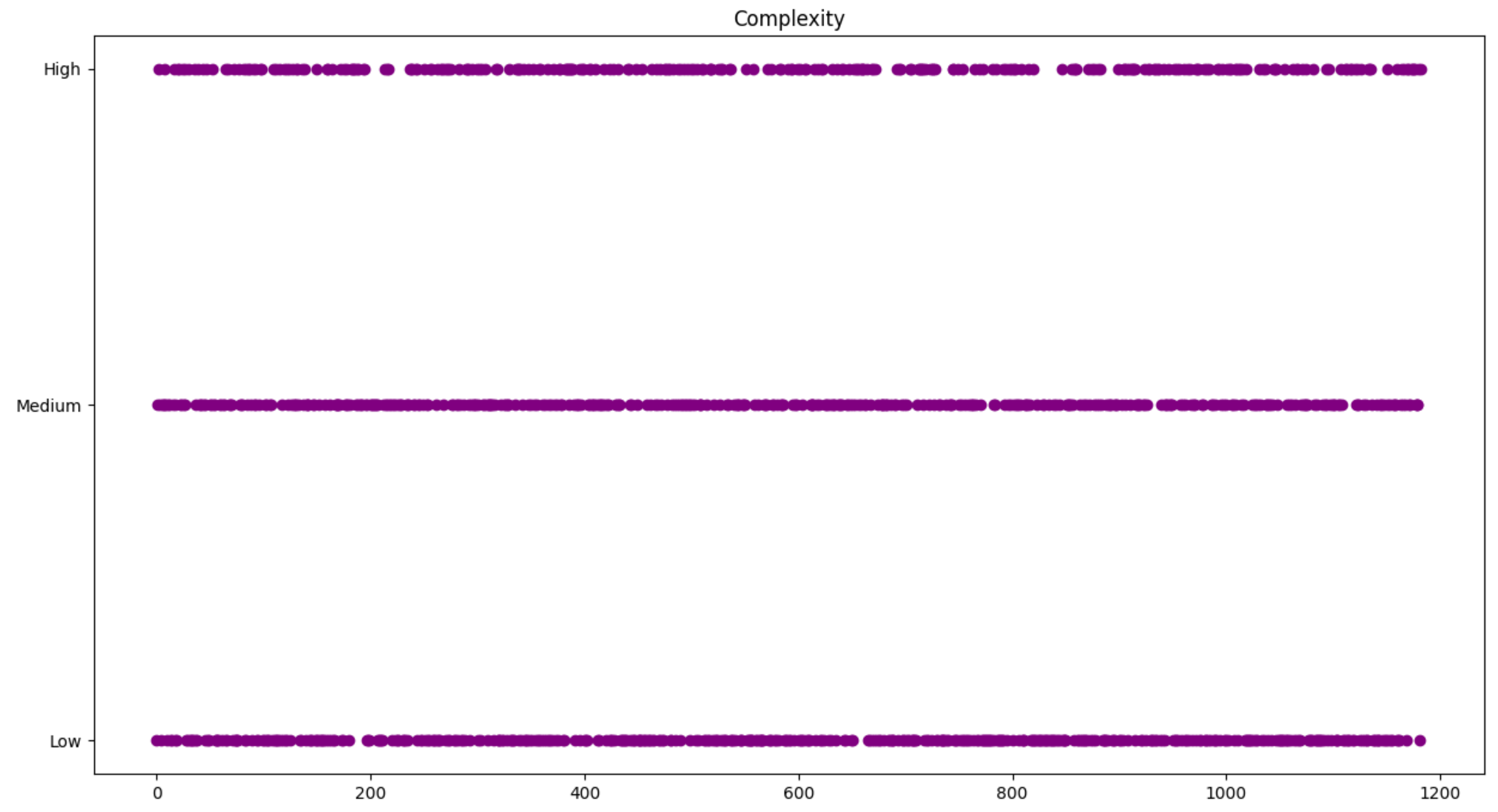




Layout Prediction (Plots)

Complexity and area show a **good variation** (in a big range) in their values

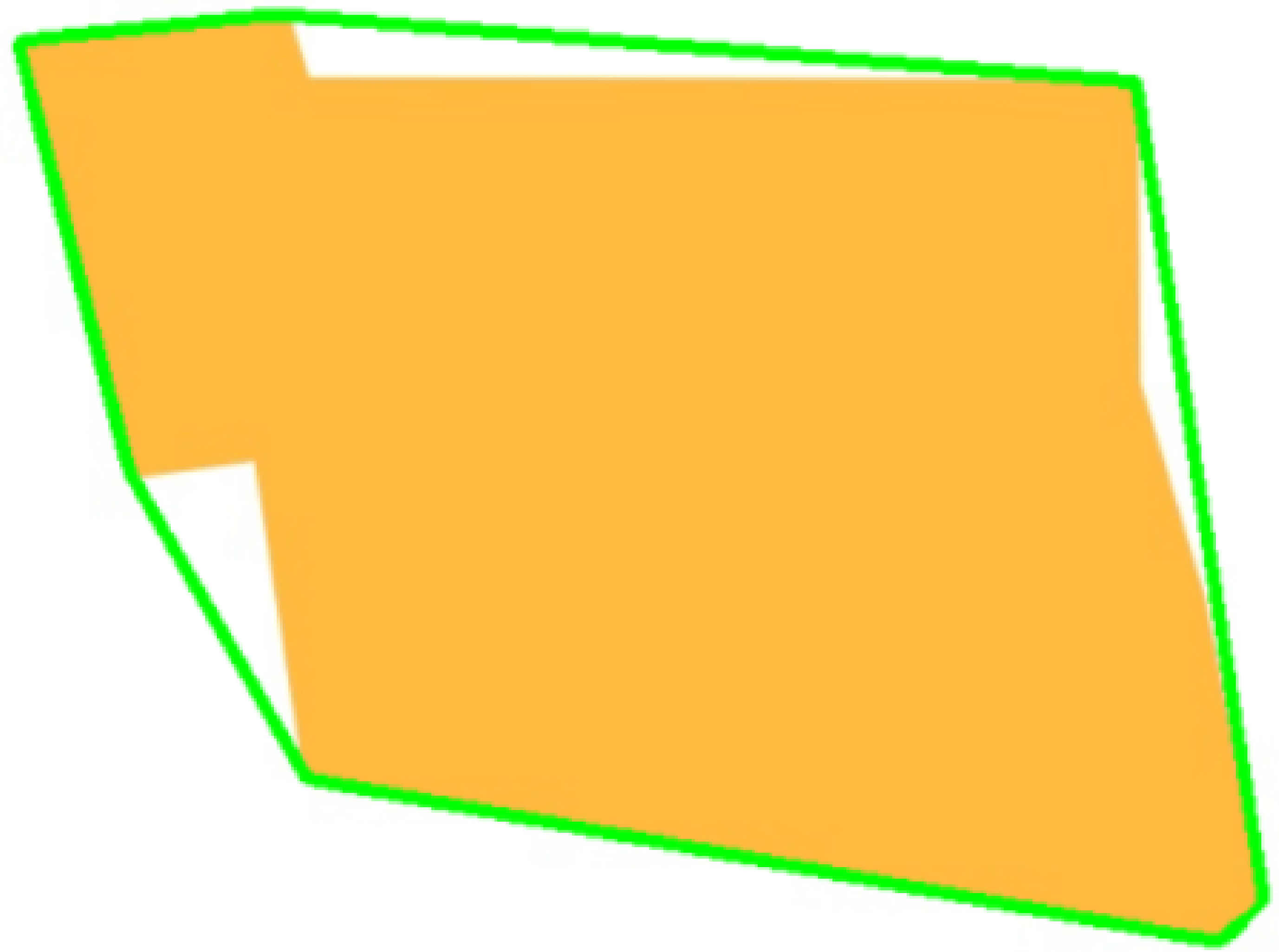




Almost **equal** number of images in each cluster

Learning and Experiences

Using data science and image processing to solve real-world problems in housing architecture. From clustering, classification, and predictive modeling techniques to processing images, and how they can help us to analyze architectural designs and extract features from the raw images,leading to meaningful insights.



Conclusion

Companies use data science and machine learning techniques to analyze their building layout designs can help them work more efficiently and meet customer needs better, enhancing productivity, facilitating robust designs, and responding effectively to customer requirements. Through the exploration of shape-based clustering through image processing for layout design is not that precise but it can definitely help to get valuable insights and basic classification.