Urban Visual Analysis

Kagioglou Maria

CONTENTS

01 Exploratory Data Analysis



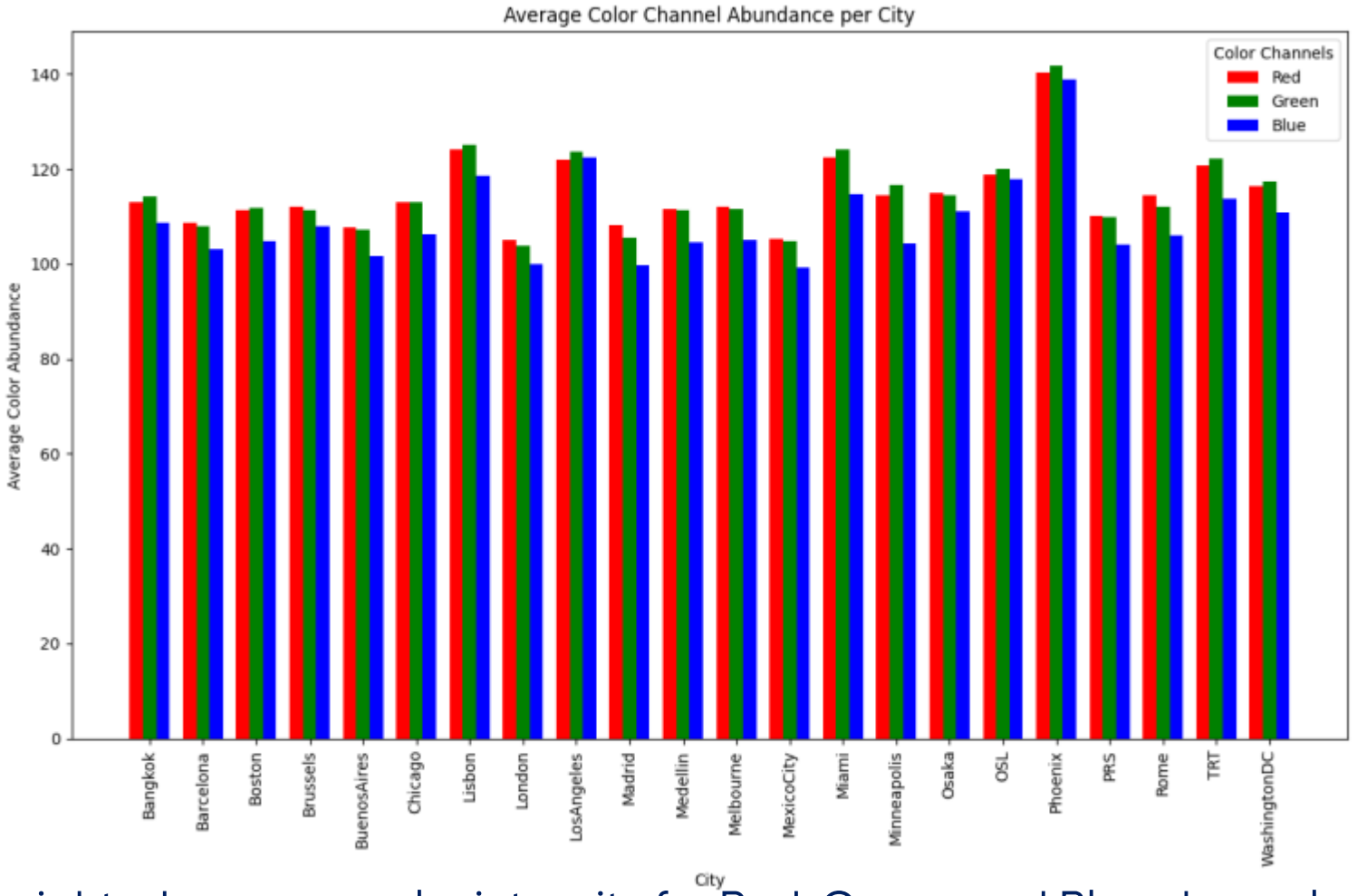






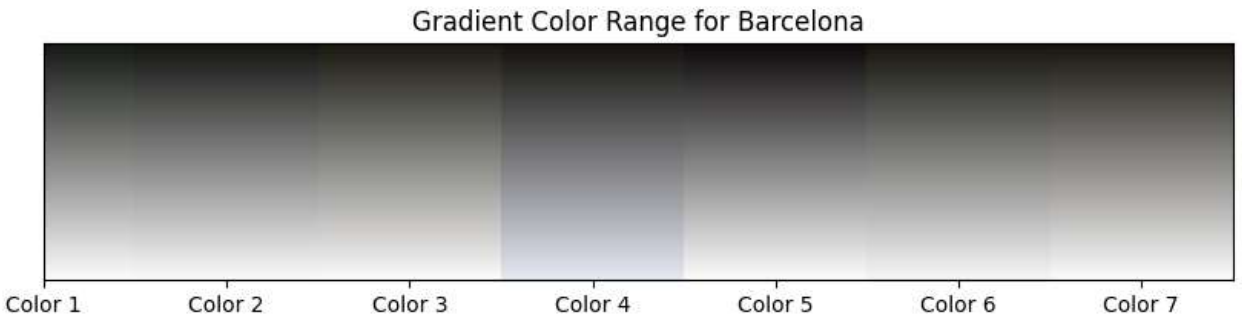

- Understand the data distribution and identify patterns, similarities, and distinctions between cities based on their visual characteristics.

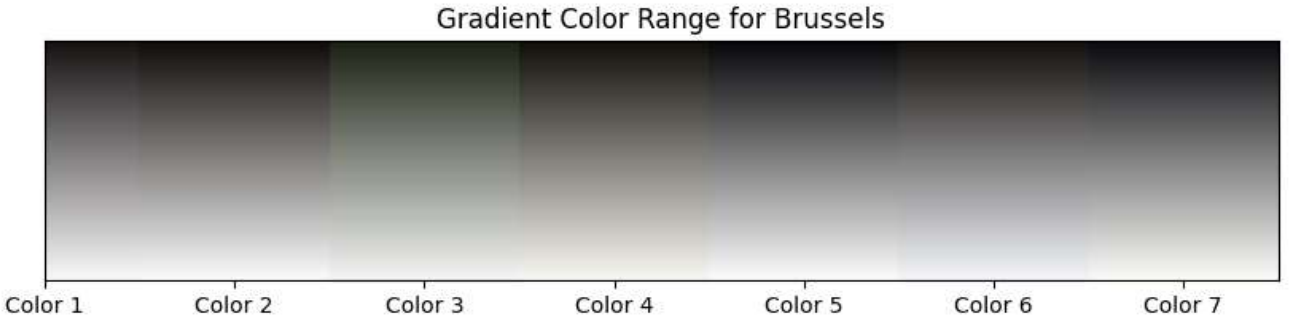
Analysis of Average RGB Color Abundance Across Cities

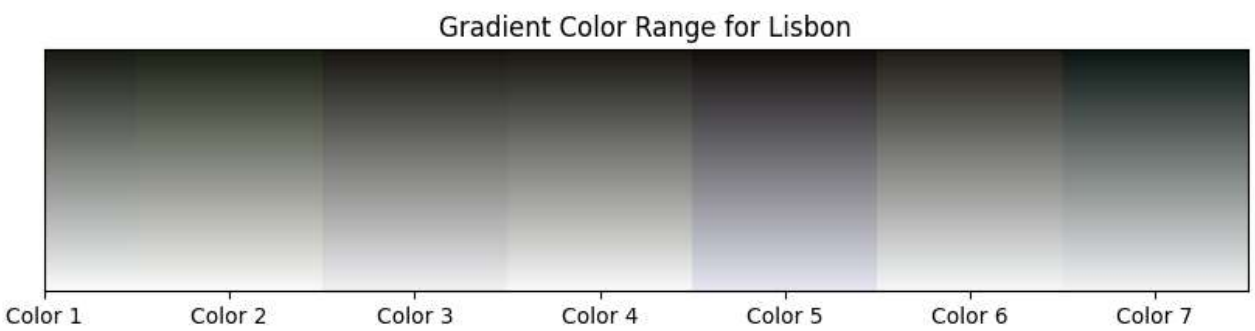


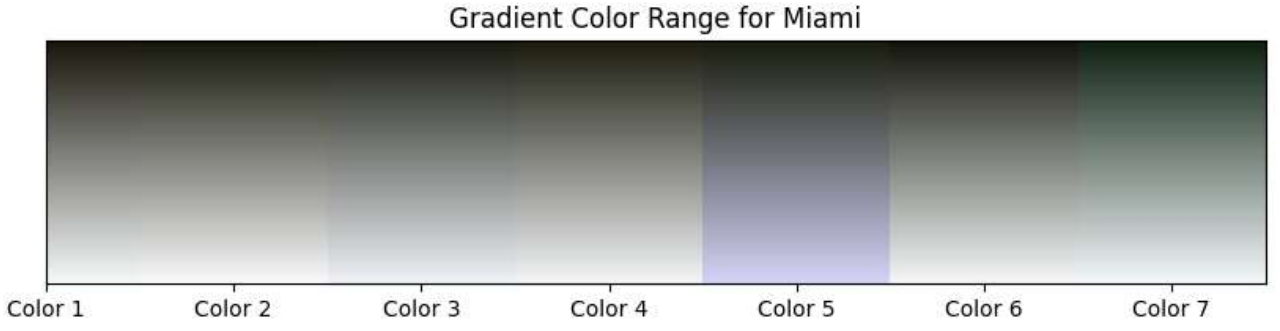
- Summarizes the weighted average color intensity for Red, Green, and Blue channels across cities.
- Uses normalized histograms to compute the mean intensity of each color channel per city.
- The bar plot visually compares the average Red, Green, and Blue channel abundance for each city.

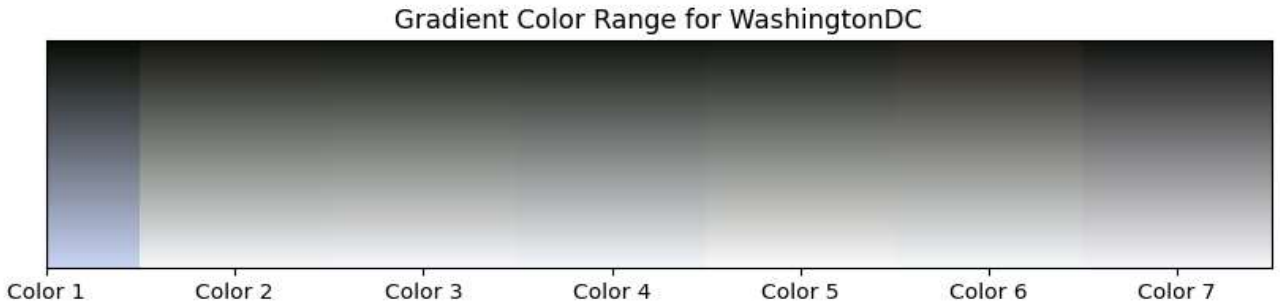
Dominant Color Extraction Across Cities

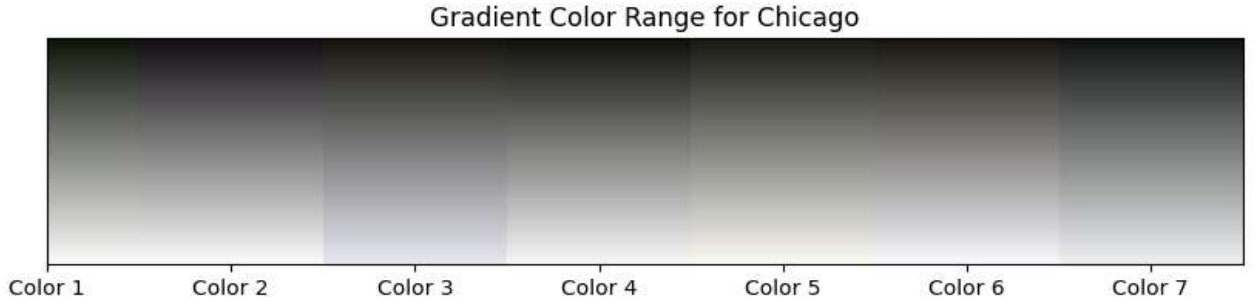






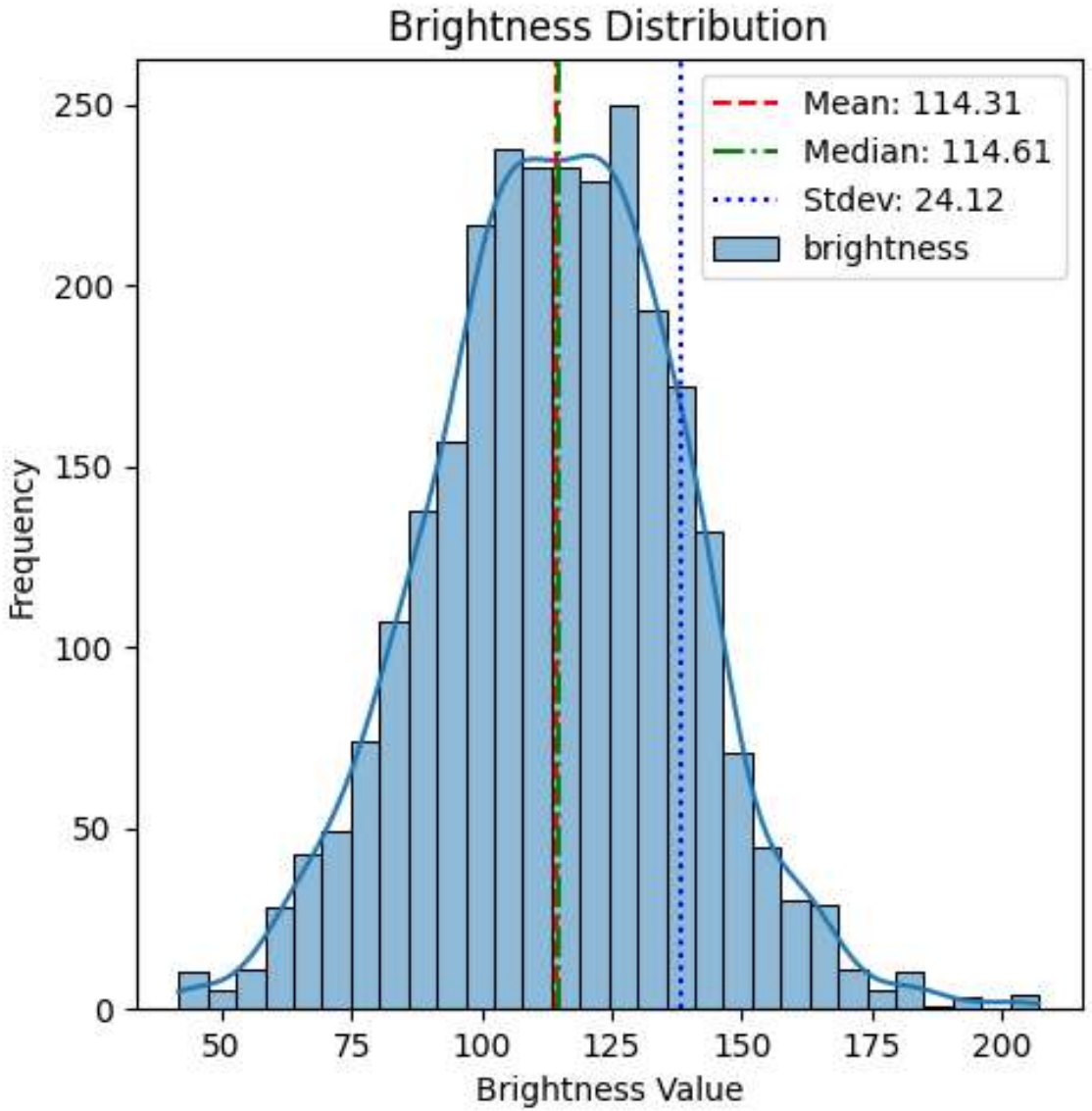






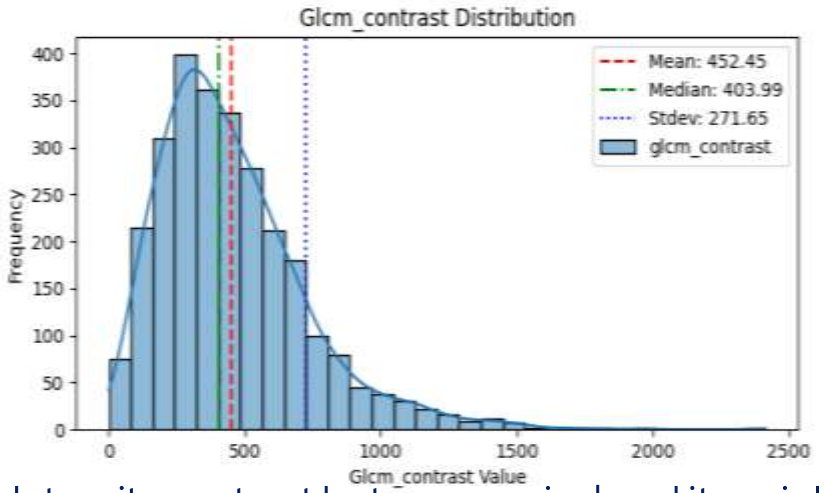
- Uses K-Means clustering to group pixels into clusters based on their RGB values. (k=7).
- Identifies dominant color clusters (centroids) and their relative frequencies.
- Produces a list of k dominant colors and their proportions in the image.

Brightness



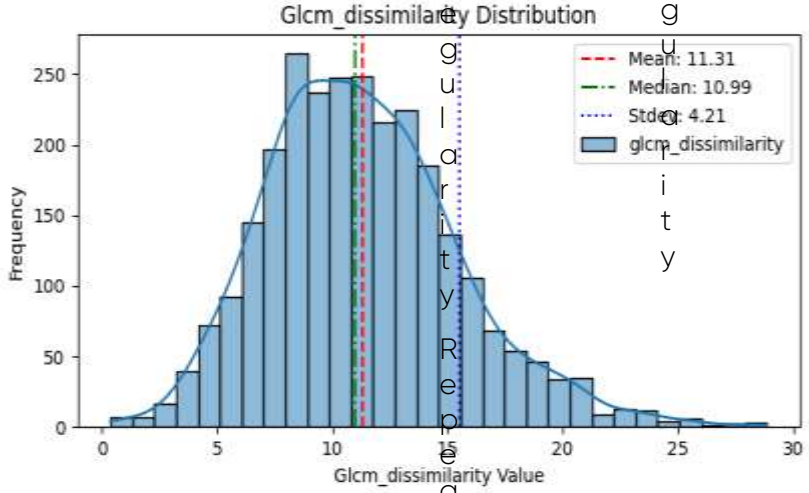
- convert image to grayscale.
- compute the mean pixel intensity.

Contrast-Dissimilarity-Homogeneity-Energy-Correlation

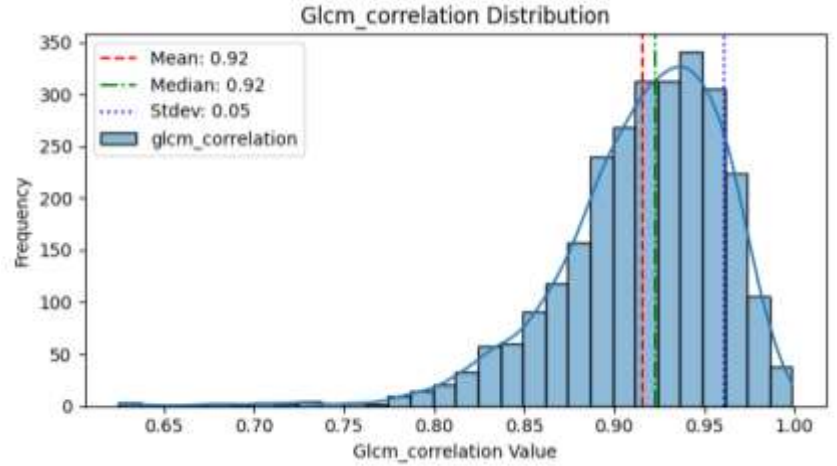


Intensity contrast between a pixel and its neighbor

Edge detection, rough texture analysis

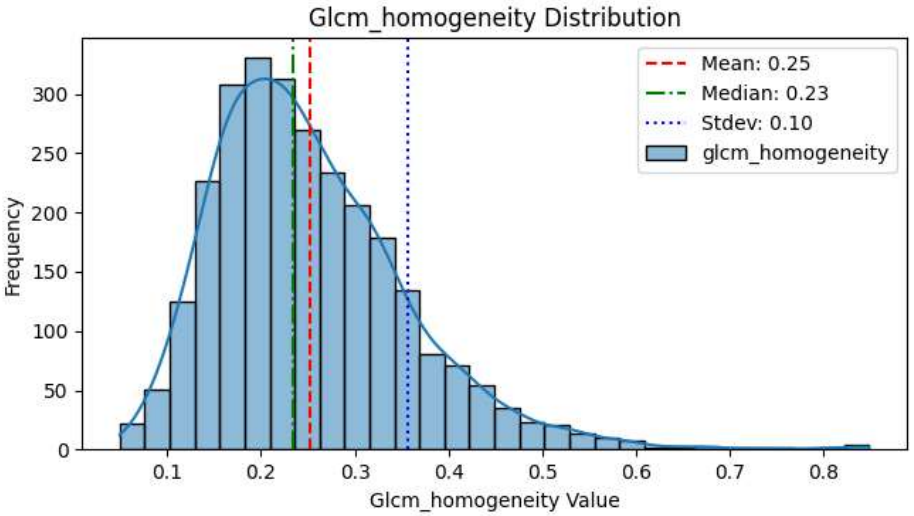


Difference between neighboring pixel intensities



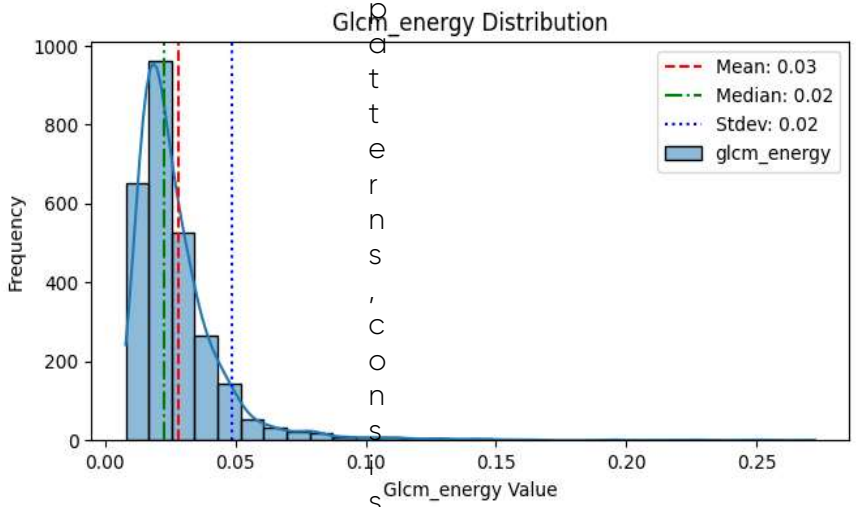
Linear relationship between pixels

Organized textures, repetitive patterns



Uniformity of pixel distribution

Smoothness

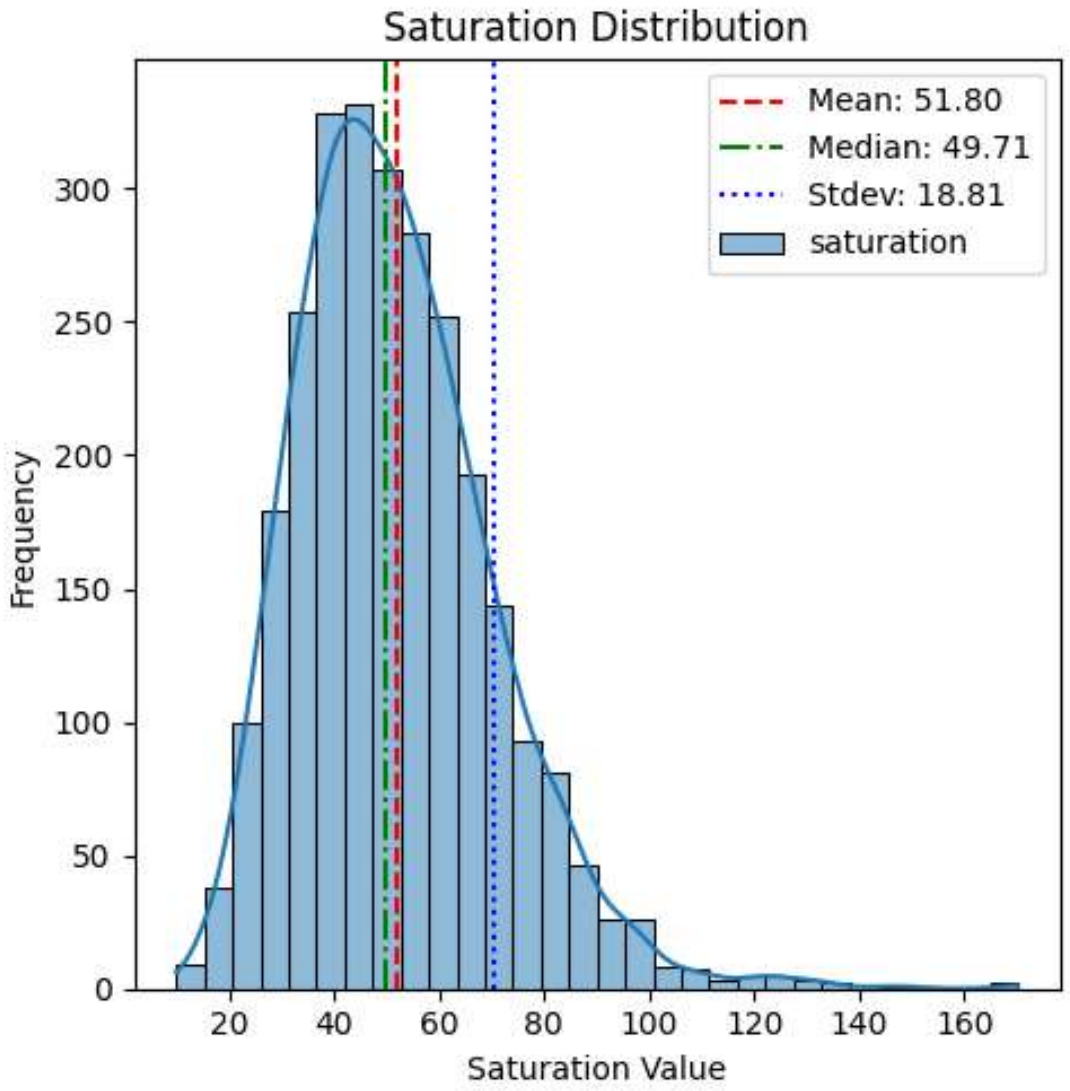


Uniformity or regularity

Repeated patterns, consistent textures

- Use the Gray-Level Co-occurrence Matrix (GLCM): statistical method used to analyze the spatial relationship between pixel intensities.
- skimage.feature.texture

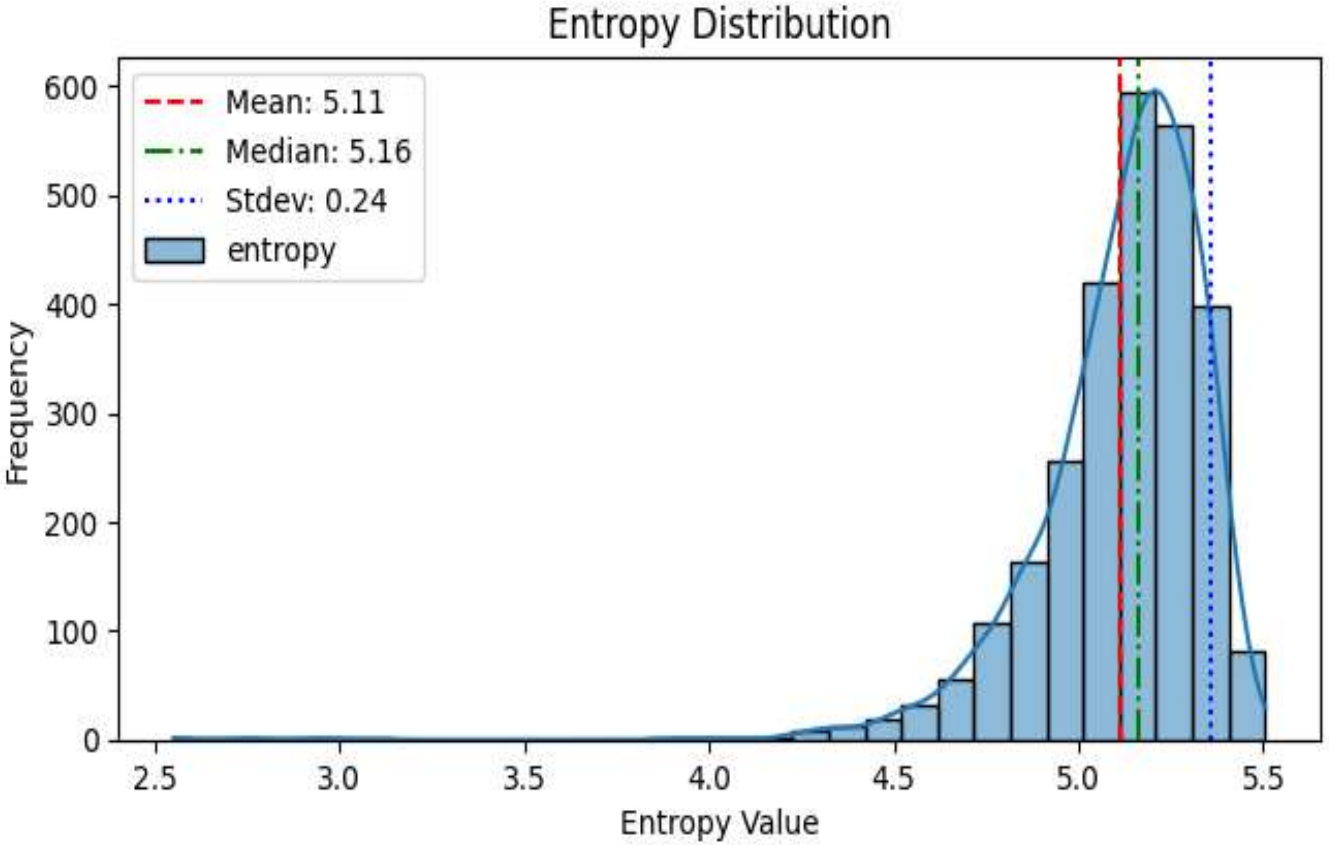
Saturation



- The image is converted to the HSV color space (cv2.cvtColor), where the second channel (H, S, V) represents the saturation.
- np.mean(hsv[:, :, 1])) : quantify the overall color intensity and vividness of the image.

- Measures the intensity of colors-color vibrancy or dullness.

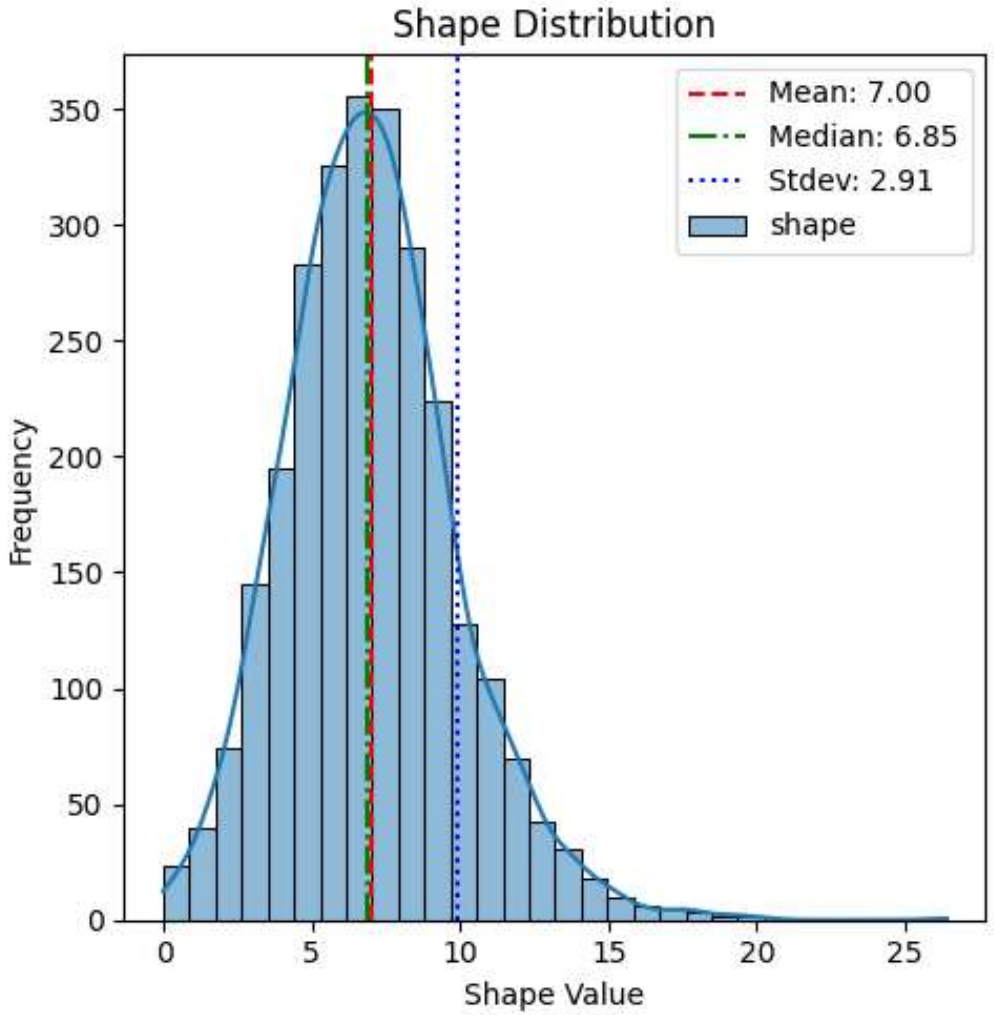
Entropy



- Grayscale image.
- Compute entropy using scipy.stats.entropy.

- Measure of randomness or texture complexity in the image.

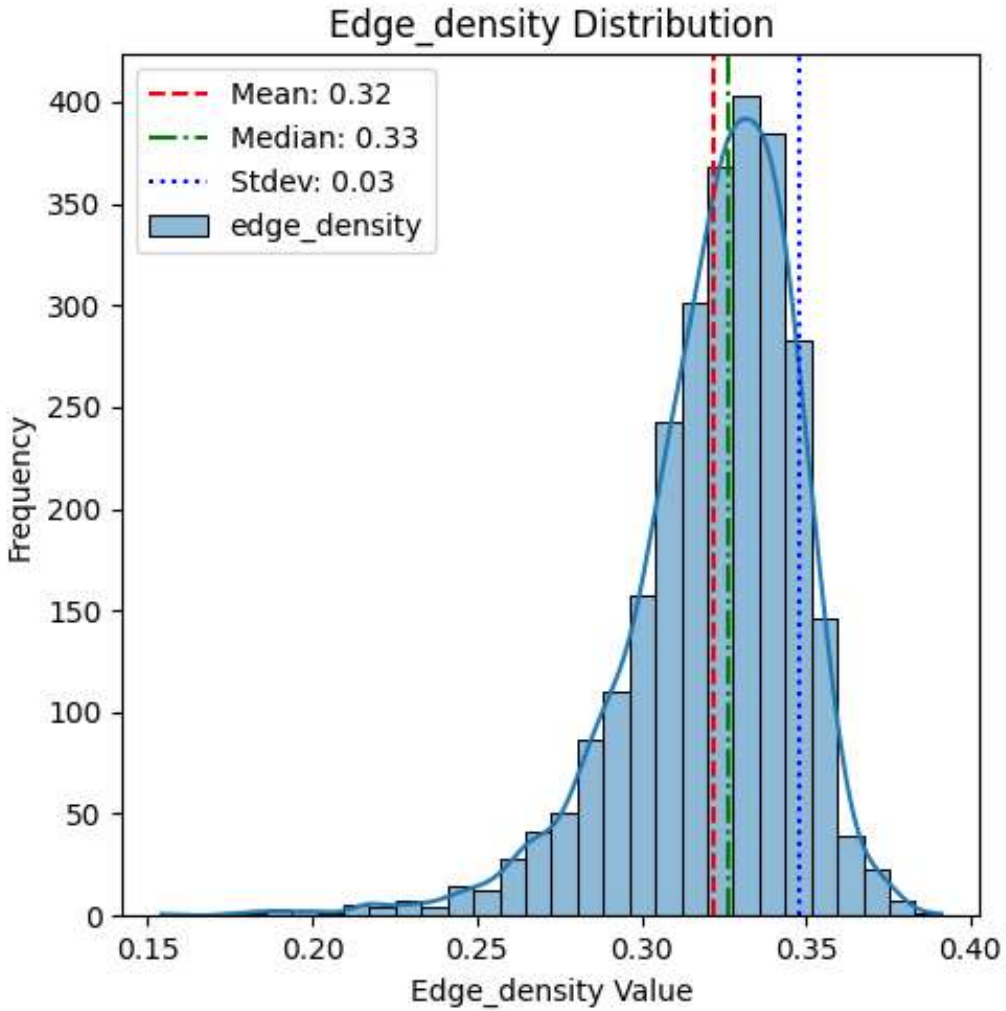
Shape Features



- Converts the Input Image to Grayscale.
- Binary thresholding to the grayscale image using 'cv2.threshold' with a fixed threshold value of 128: above the threshold are set to 255 (white), and those below are set to 0 (black).
- Detects contours in the thresholded binary image using cv2.findContours.
- Number of features to max_contours = 600.

- Compute shape-related features by analyzing image contours.
- Quantify the geometrical properties of objects in an image.
- Detecting circles, polygons, small vs. large objects etc.

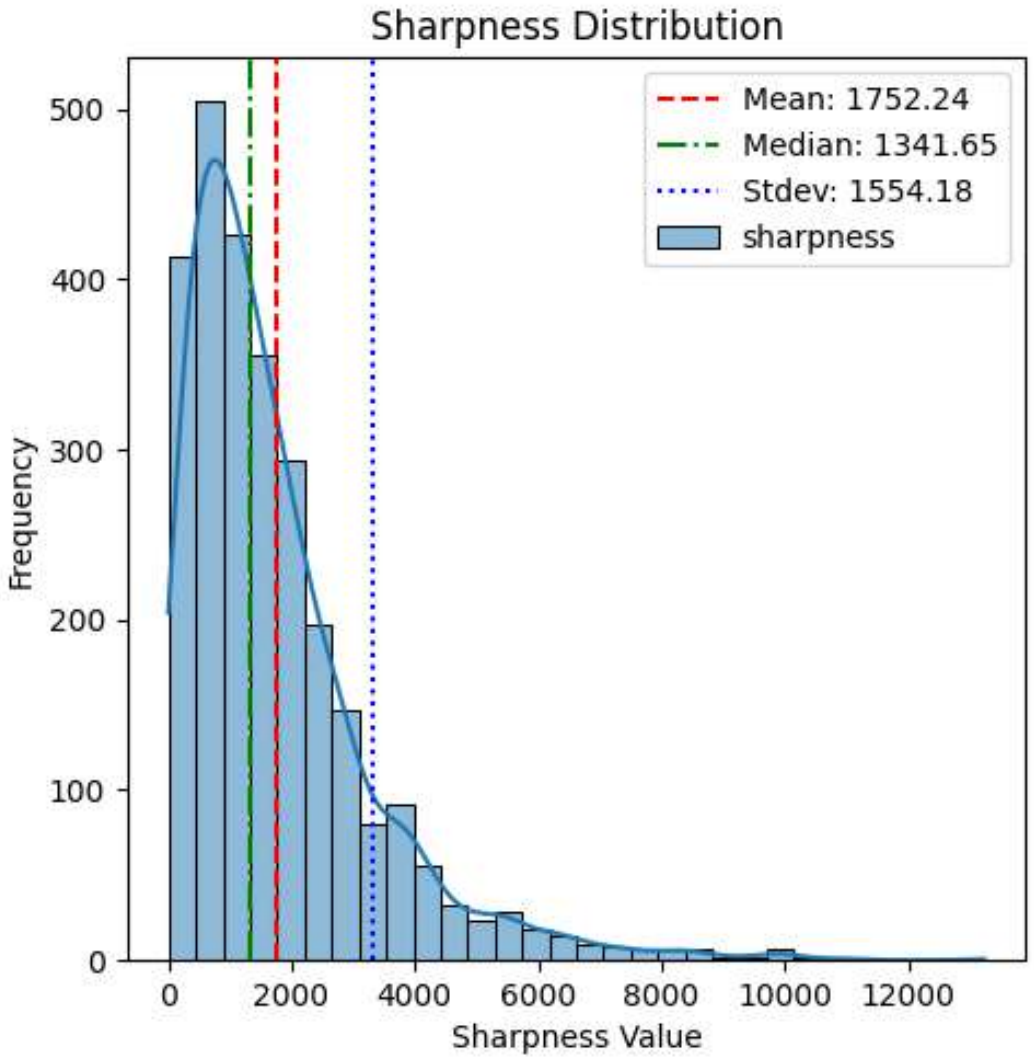
Edge Density



- The Canny edge detection algorithm (cv2.Canny) is applied to the grayscale image with very low thresholds (threshold1=1, threshold2=1) to detect edges.

- Quantifies the amount of structural detail & complexity in the image.
- Object Detection, scene complexity analysis.
- Details or busy scenes vs smoothness or plain regions, urban environments vs. natural scenes.

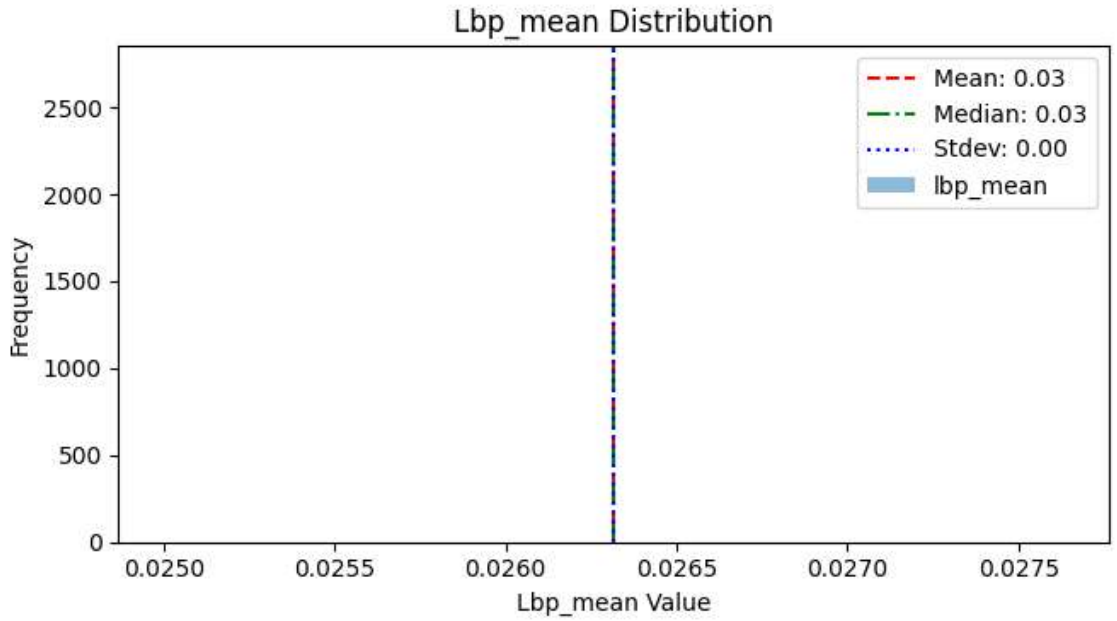
Sharpness

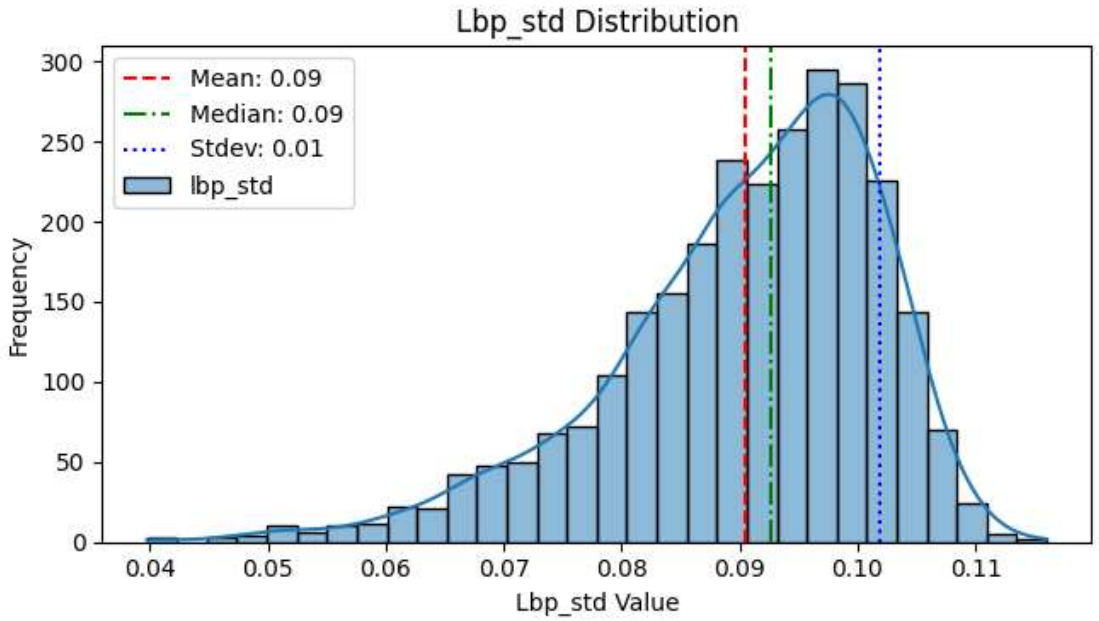


- The Laplacian operator is applied to the grayscale version of the image (cv2.Laplacian), which calculates the second-order derivatives to detect rapid intensity changes (edges).

- Higher variance indicates sharper images due to more intense and frequent changes in pixel intensity.
- Blurry or crisp image => readability or usability of the image.

LBP Features

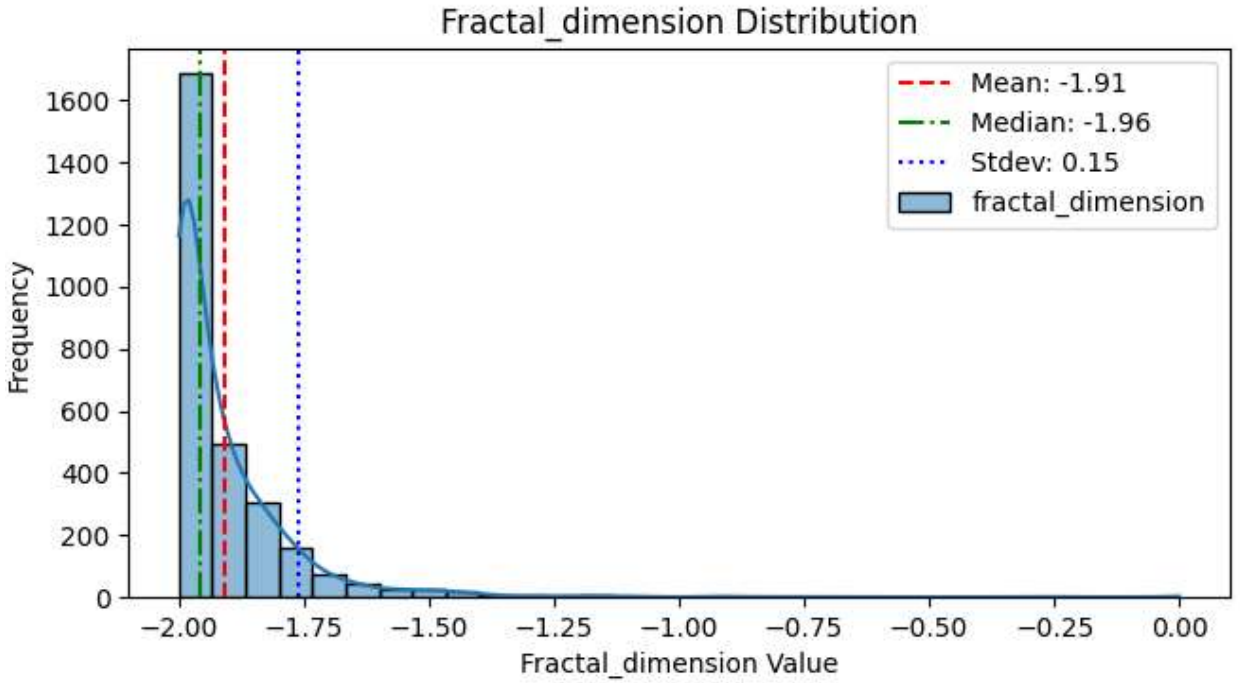




- Grayscale image.
- Parameters for LBP Calculation:
  1. radius = 4: (distance between the central pixel and its neighbors)
  2. n_points = 9 * radius: (total number of neighboring points that will be considered)
- Computes Local Binary Pattern (LBP) : (lbp = local_binary_pattern(img, n_points, radius, method='uniform'))
- Captures local texture patterns by comparing a central pixel to its surrounding neighbors.
- Rough vs. smooth textures, repetitive patterns, or uniform regions, classifying materials like wood, metal etc

Provides the mean and standard deviation of the LBP histogram, describing the texture's uniformity and variability.

Fractal Dimension



$$D = -\frac{log(N(s))}{log(s)}$$

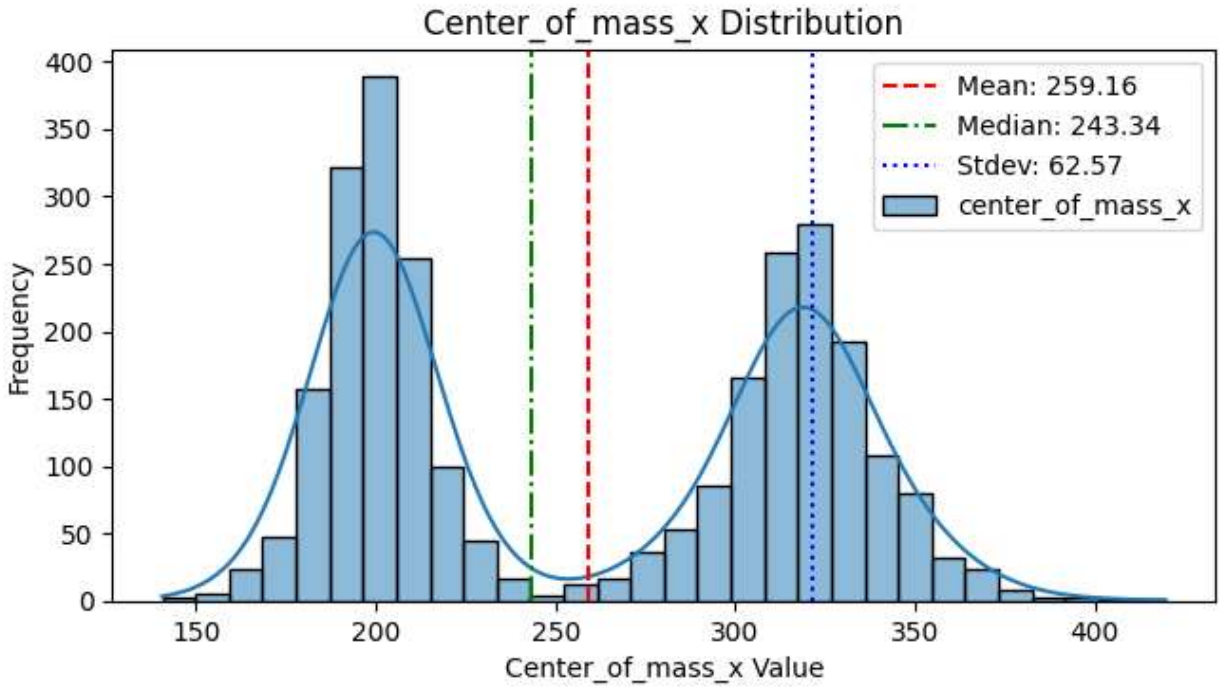
Where:

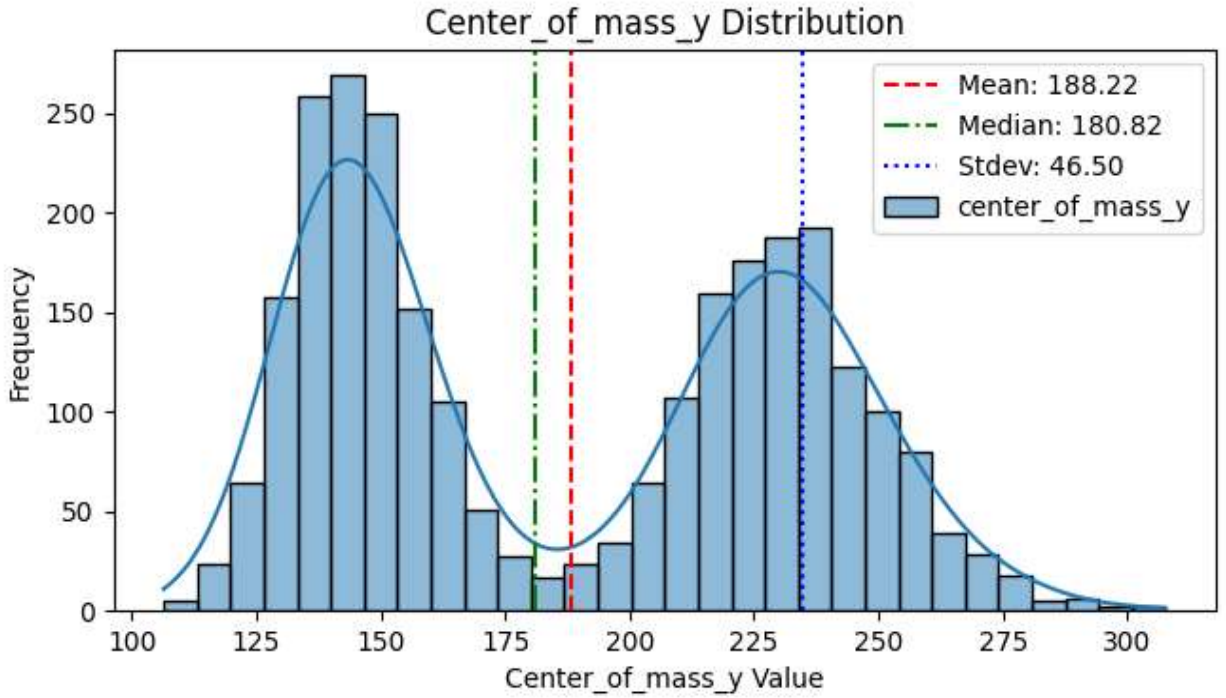
1. D: the fractal dimension.
2. N(s): the number of non-zero pixels at a given box size.
3. s: the size of the box.

- Box Size: divide the image into boxes of varying sizes and count how many boxes contain non-zero pixels.

Measure of its complexity and self-similarity-patterns that are irregular or fragmented, such as natural textures (rivers, mountains) & landscapes.

Spatial Center of Mass





- image in grayscale using cv2.imread(image_path, cv2.IMREAD_GRAYSCALE).
- image moments cv2.moments(img): geometric and intensity-related information about the image.
- Symmetry Analysis, object localization.

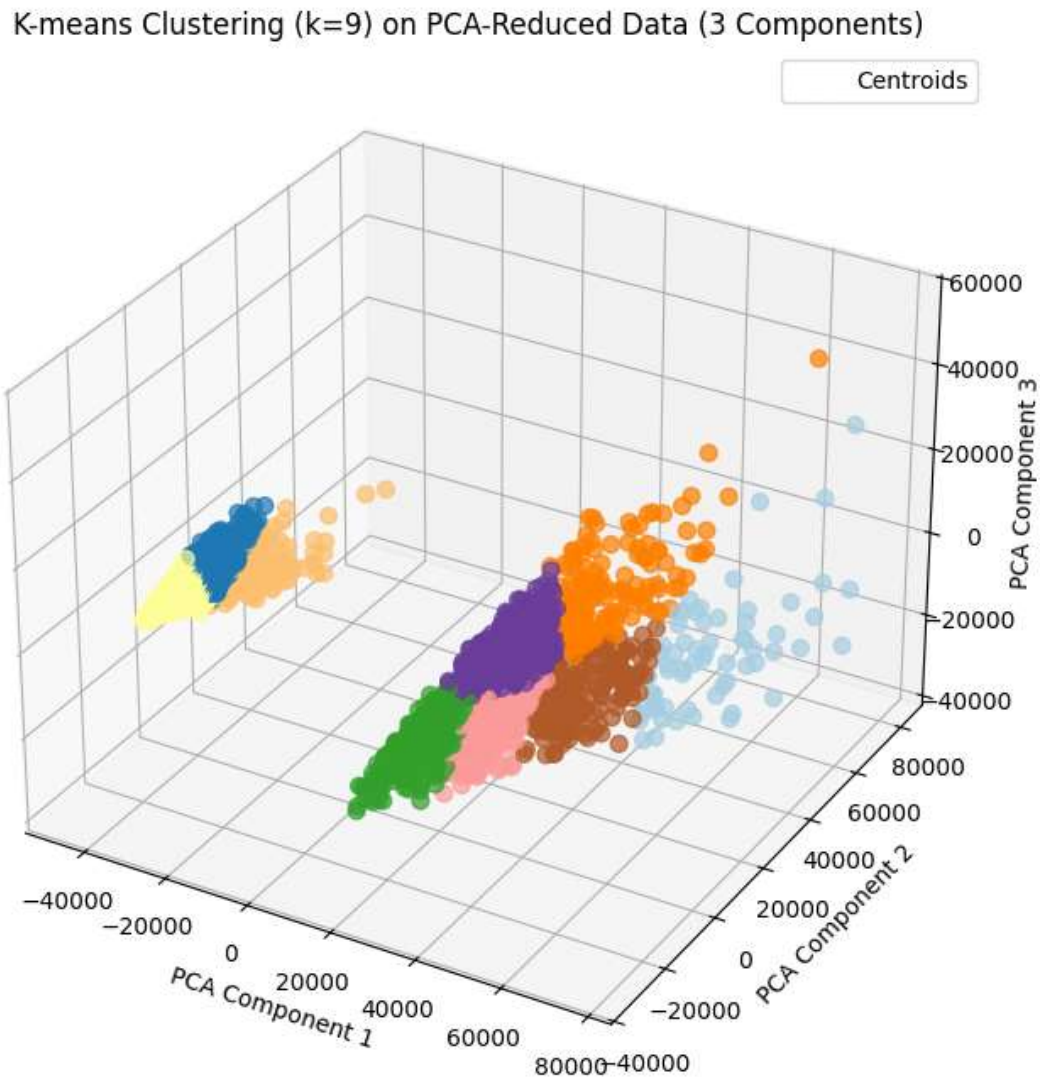
02 Unsupervised Learning

Silhouette Score

- Evaluates the quality of the clustering.
- A higher Silhouette Score indicates that the clusters are well-separated and compact.
- Measures how similar an object is to its own cluster compared to other clusters. A score close to +1 indicates well-defined clusters, 0 indicates overlapping clusters, and negative scores suggest misclassified points.

- The KMeans algorithm is initialized with:
  - n_init=10: The number of times the KMeans algorithm will run with different initializations to find the optimal clustering.
  - max_iter=4000: The maximum number of iterations to refine the clusters.
  - tol=1e-5: The tolerance for convergence. The algorithm stops if the improvement between iterations is smaller than this value.

Clustering Visualisation



PCA (Principal Component Analysis)

- PCA is a dimensionality reduction technique used to reduce the number of features while retaining as much variance as possible in the dataset. It transforms the data into a new coordinate system based on the directions of maximum variance, known as principal components.

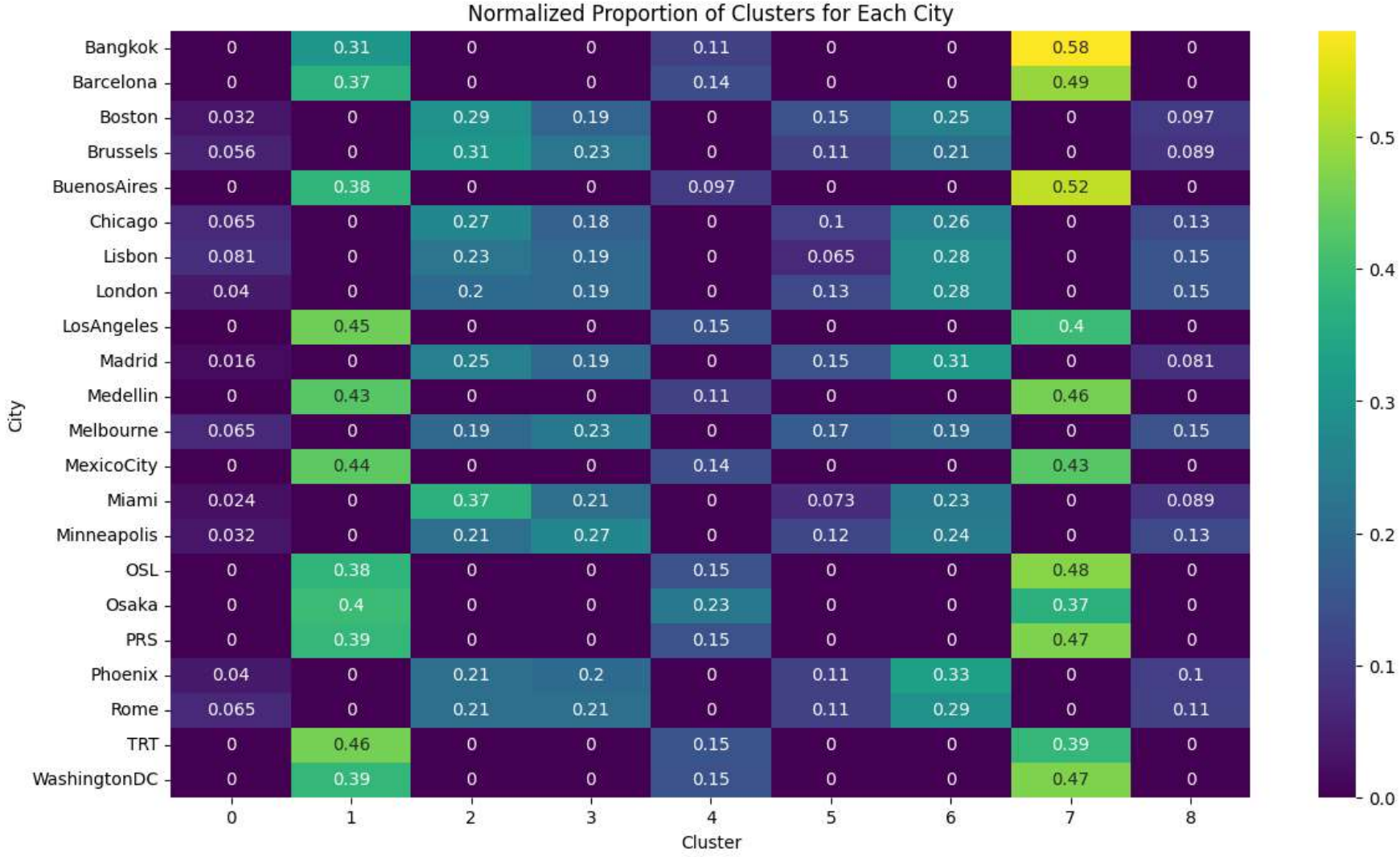
- **explained variance ratio:** The proportion of the total variance explained by each principal component.

- Number of Components to Explain 95% Variance:

```
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
d_optimal = np.argmax(cumulative_variance >= 0.95) + 1
print(f"Number of components explaining 95% variance: {d_optimal}")
```

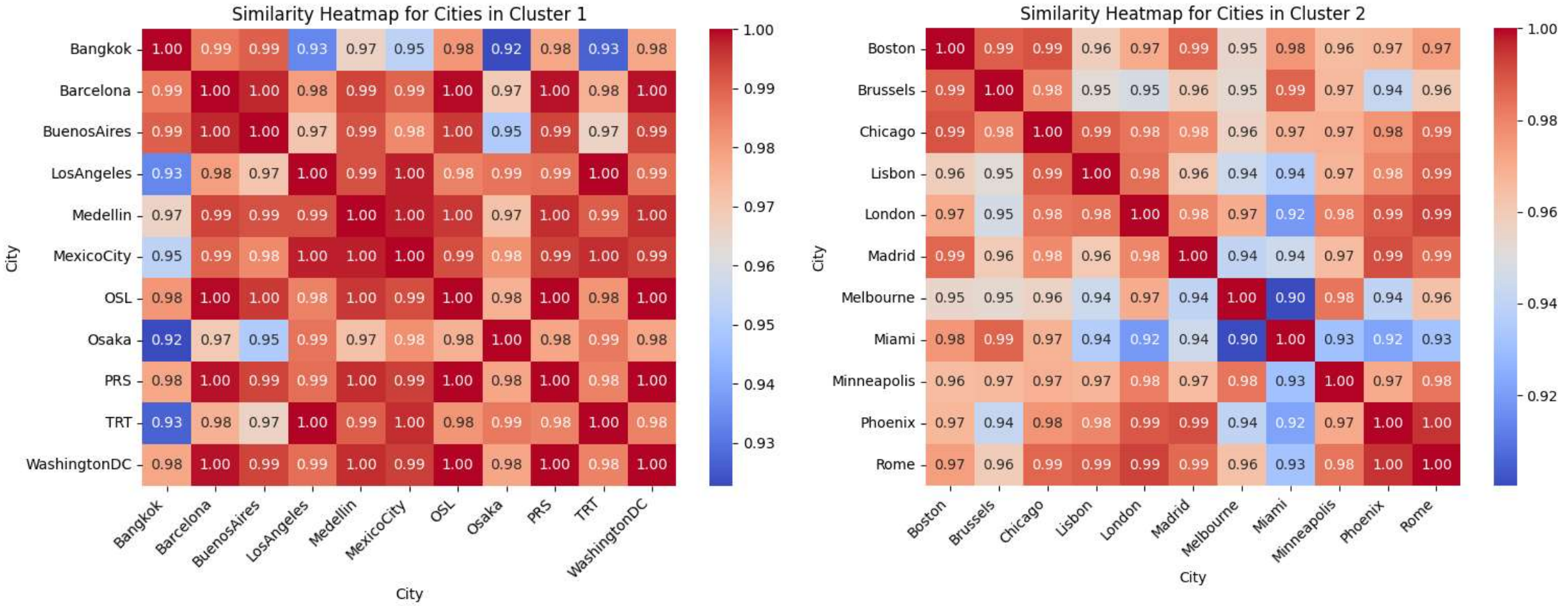
```
Number of components explaining 95% variance: 3
```

Clustering Analysis



- Normalized proportions of images in each cluster for each city.
- identify which clusters are dominant in each city.

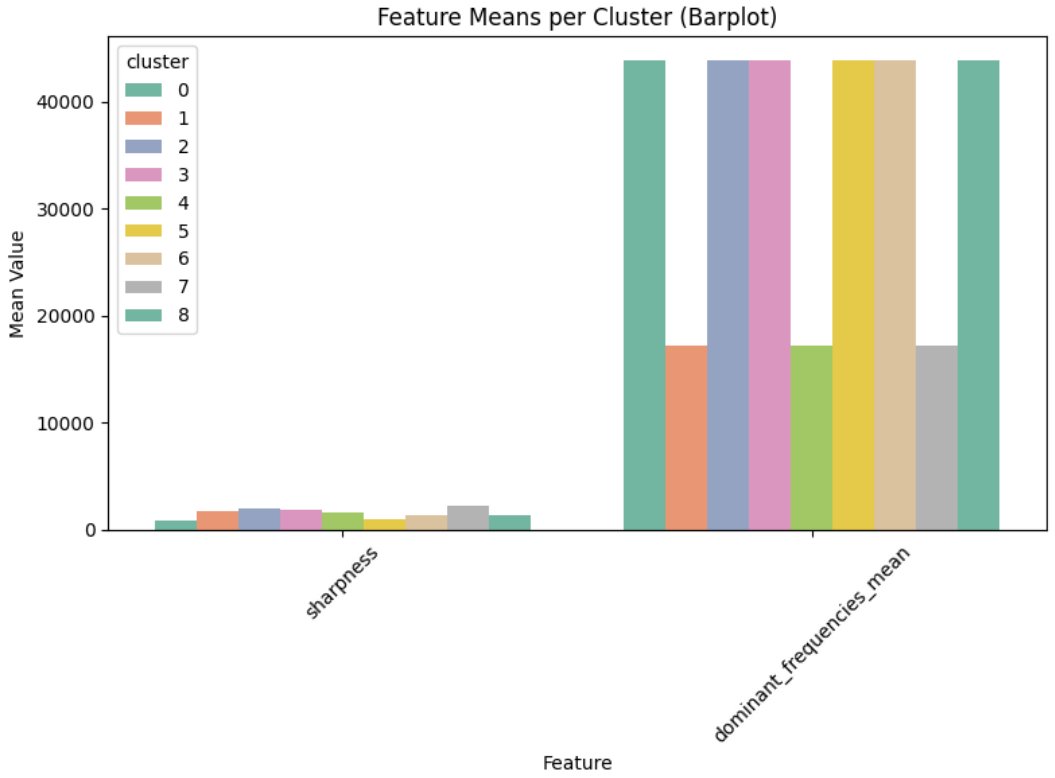
Clustering Analysis

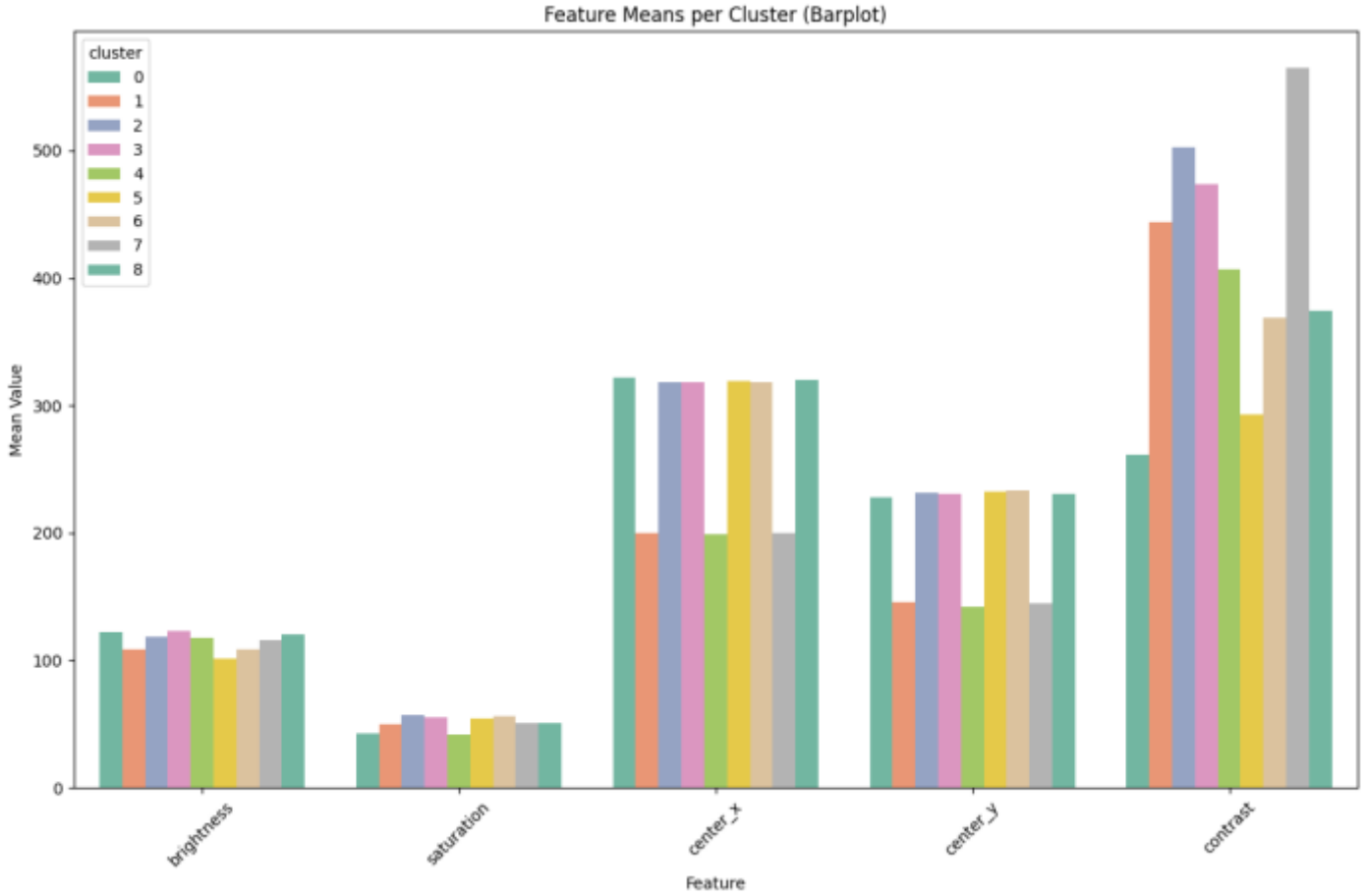


Cosine Similarity

- threshold = 0.1 is applied to filter cities that have significant proportions in each cluster.
- calculates the pairwise cosine similarity between cities. Cosine similarity measures the similarity of two vectors based on their orientation (ignoring their magnitude), which is useful for comparing the cluster proportions of different cities.

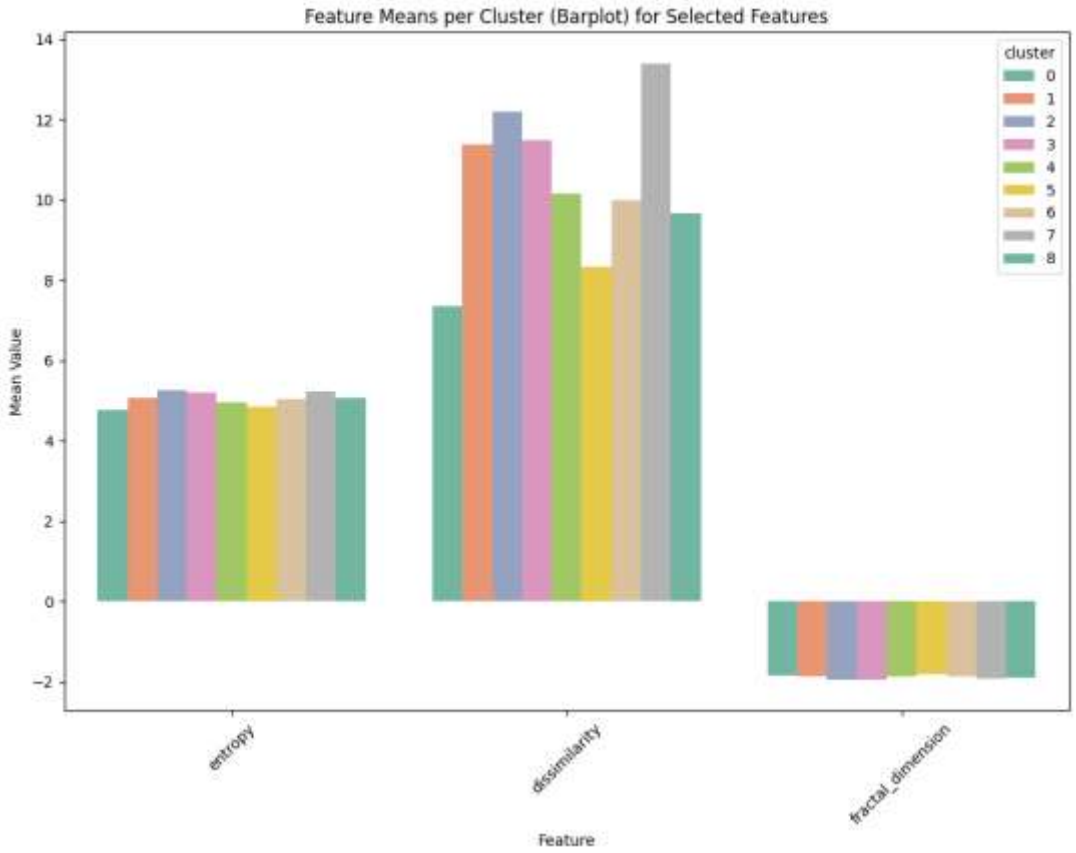
Barplots of Feature Means per Cluster

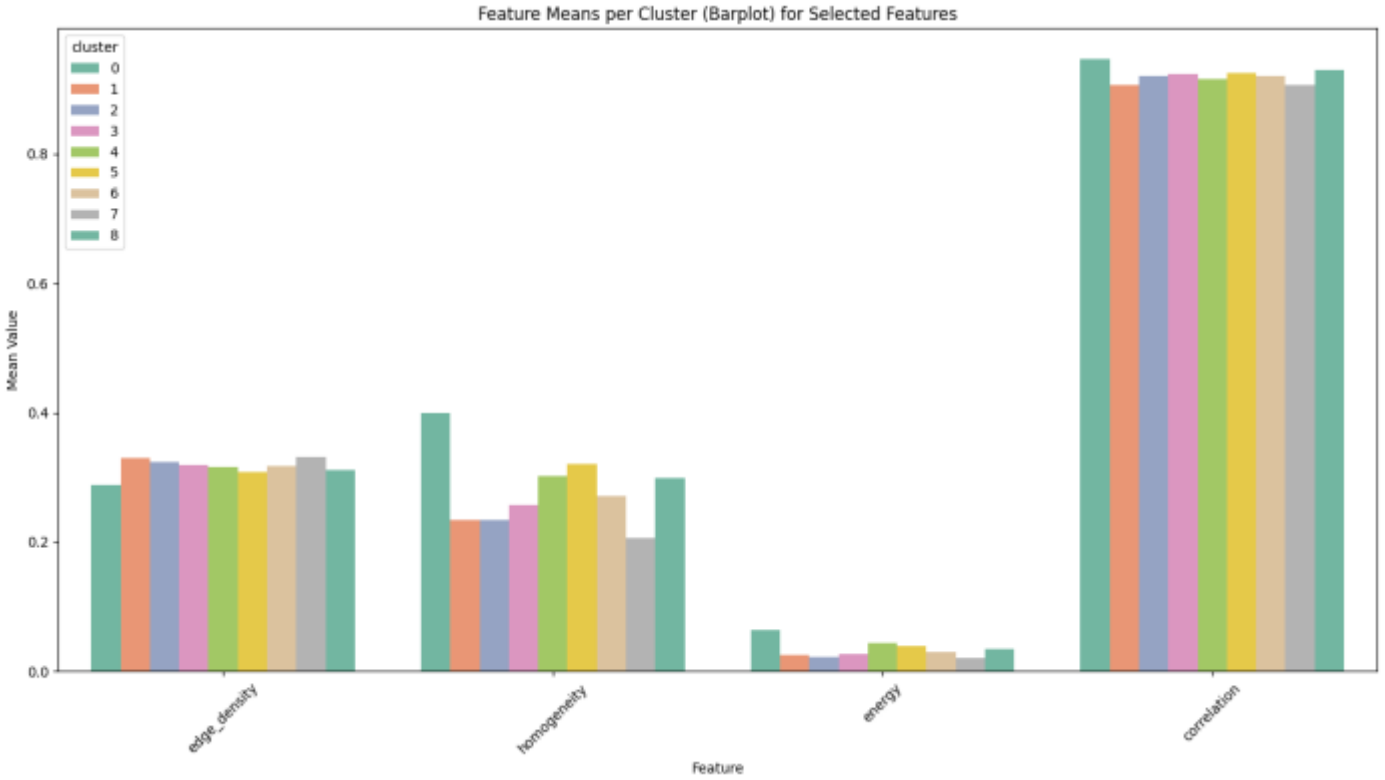




- Features with significant differences in mean values across clusters, such as contrast, likely contribute more to cluster separation.

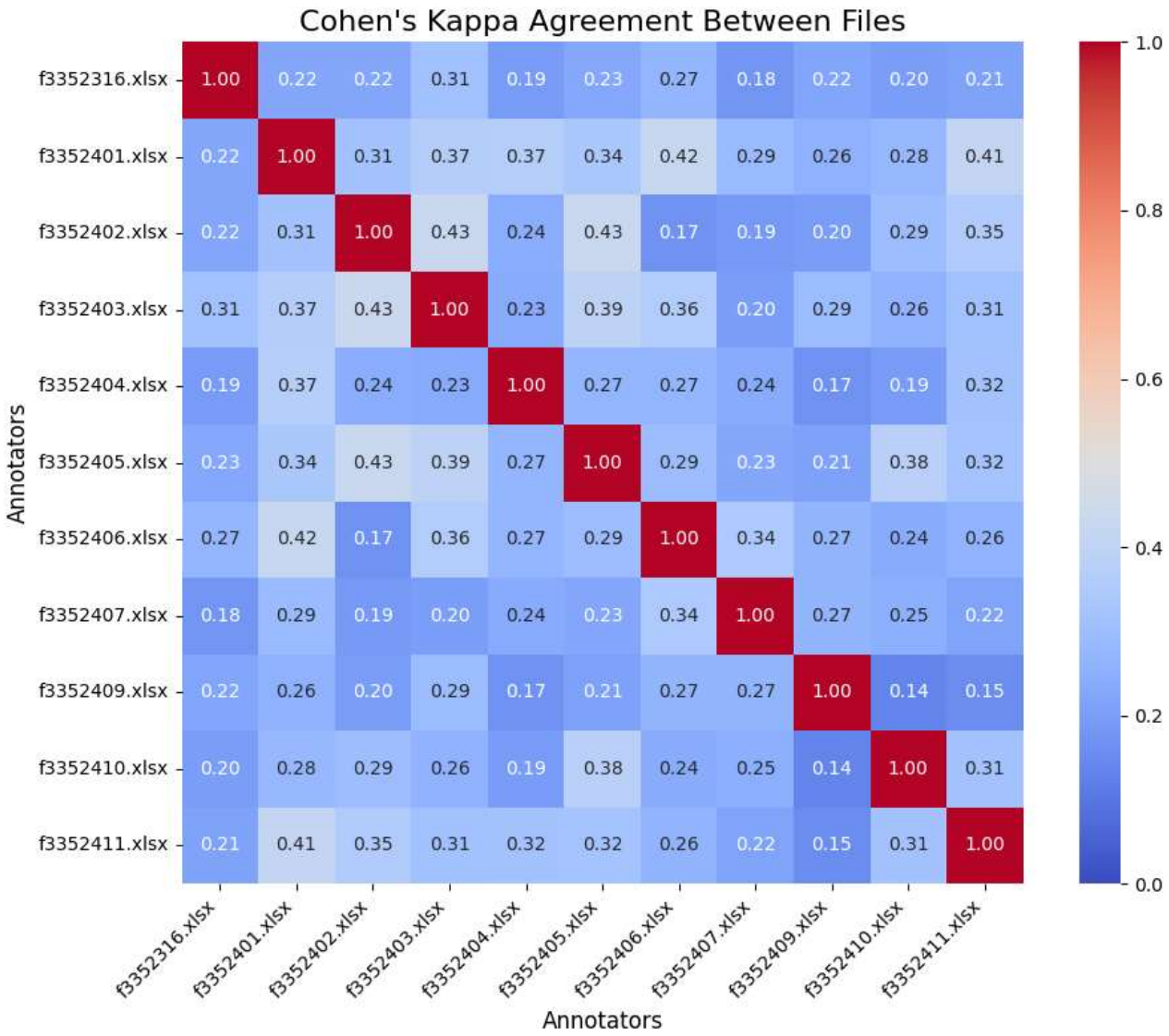
Barplots of Feature Means per Cluster





- Features with significant differences in mean values across clusters, such as homogeneity & dissimilarity, likely contribute more to cluster separation.

03 Image Annotation



Cohen's Kappa

$$k = \frac{P_a - P_e}{1 - P_e}$$

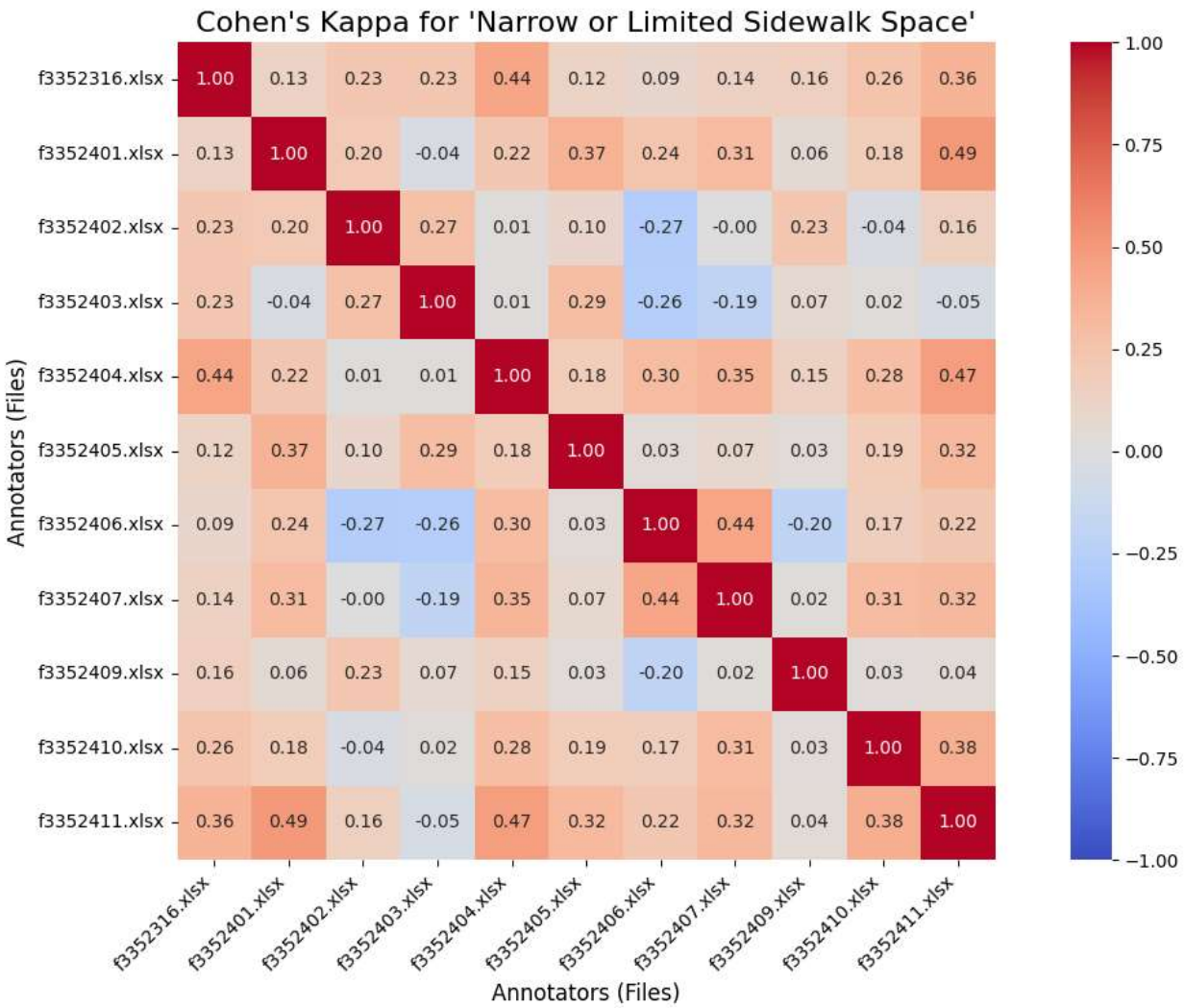
Where:

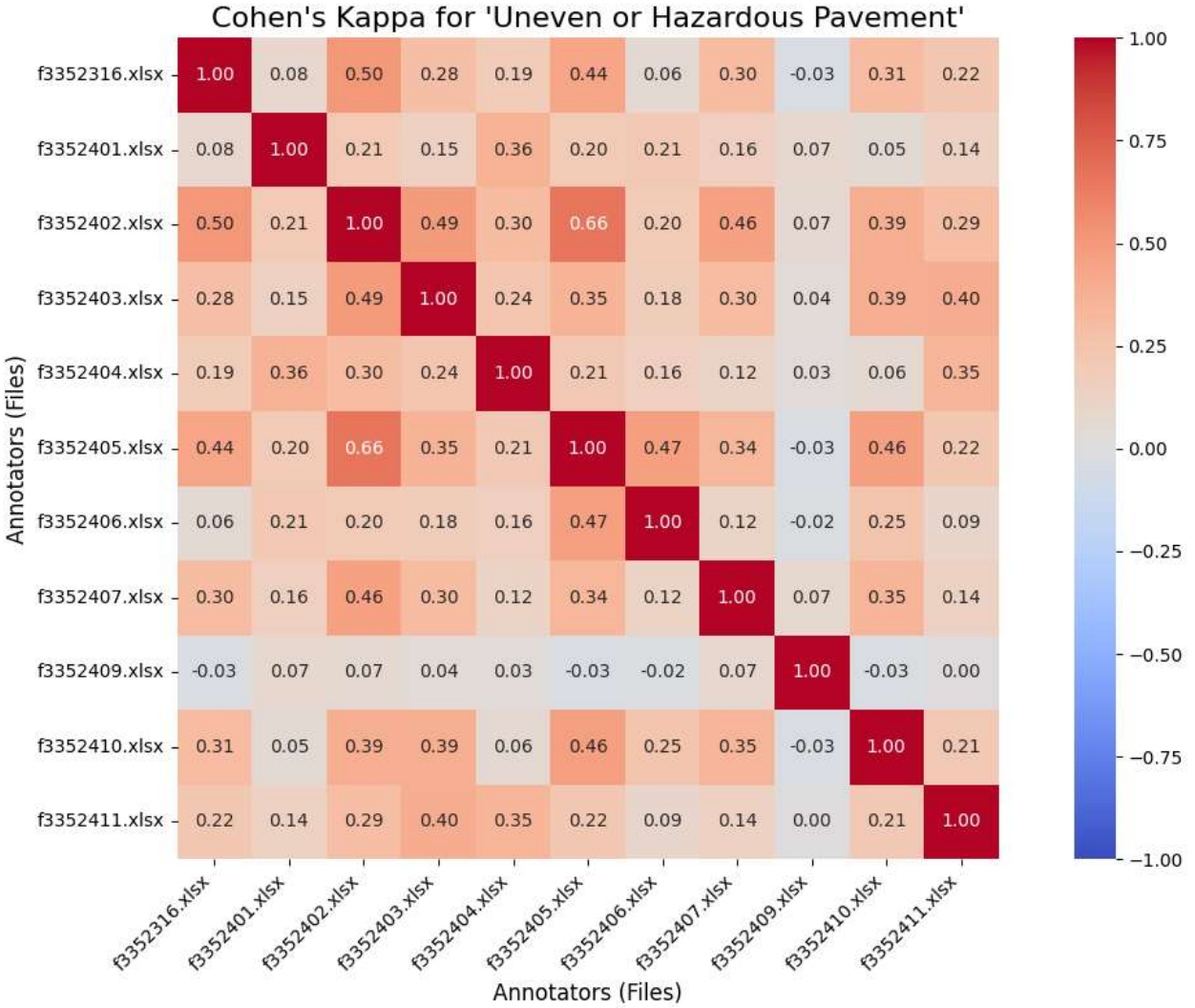
- **Pa:** The observed agreement (proportion of times the annotators agree).
- **Pe:** The expected agreement by chance (calculated based on the distribution of annotations).

Cohen's Kappa range: from -1 to 1:

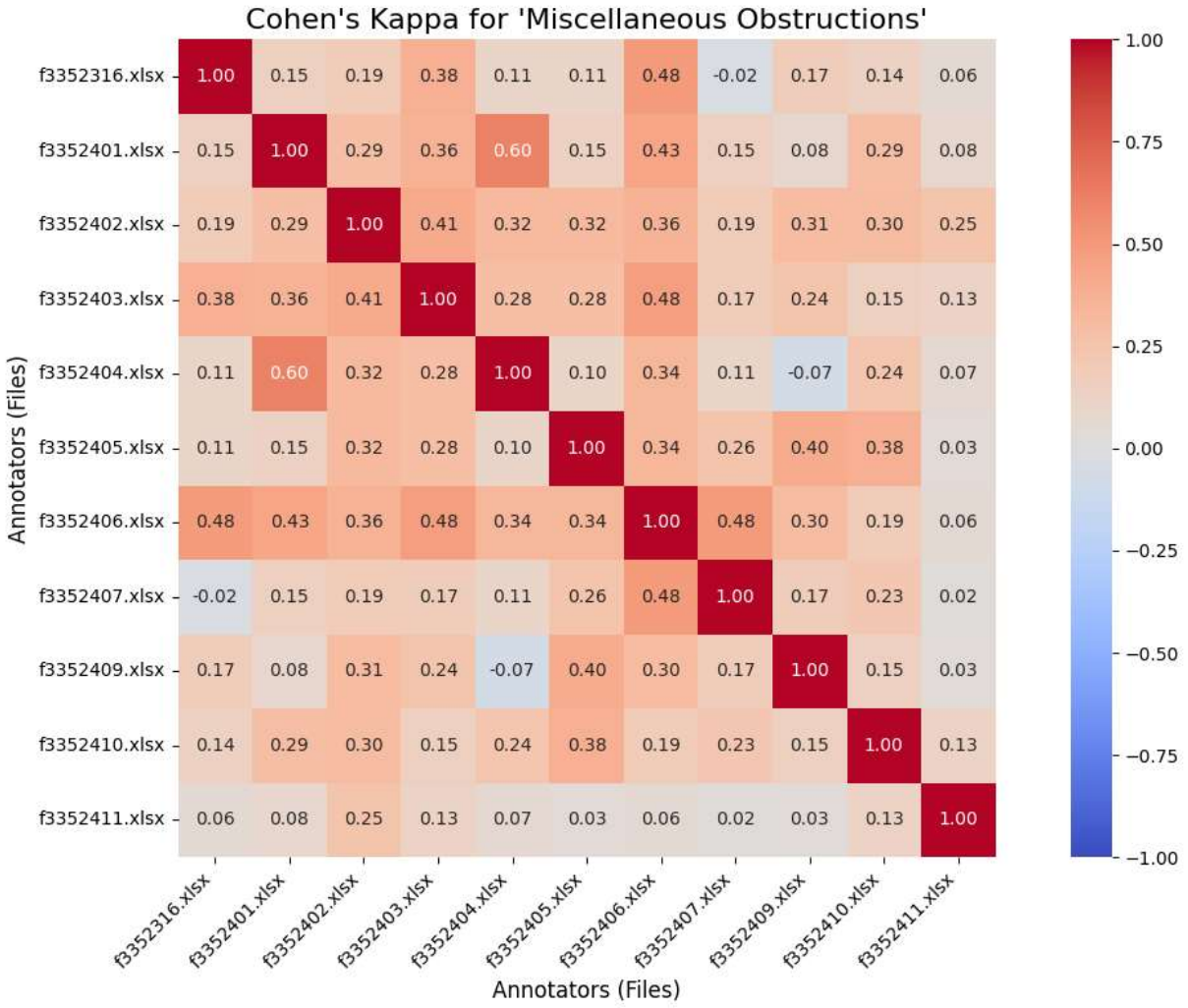
1. 1: Perfect agreement between annotators.
2. 0: No agreement beyond what would be expected by chance.
3. <0: Indicate less agreement than would be expected by chance.

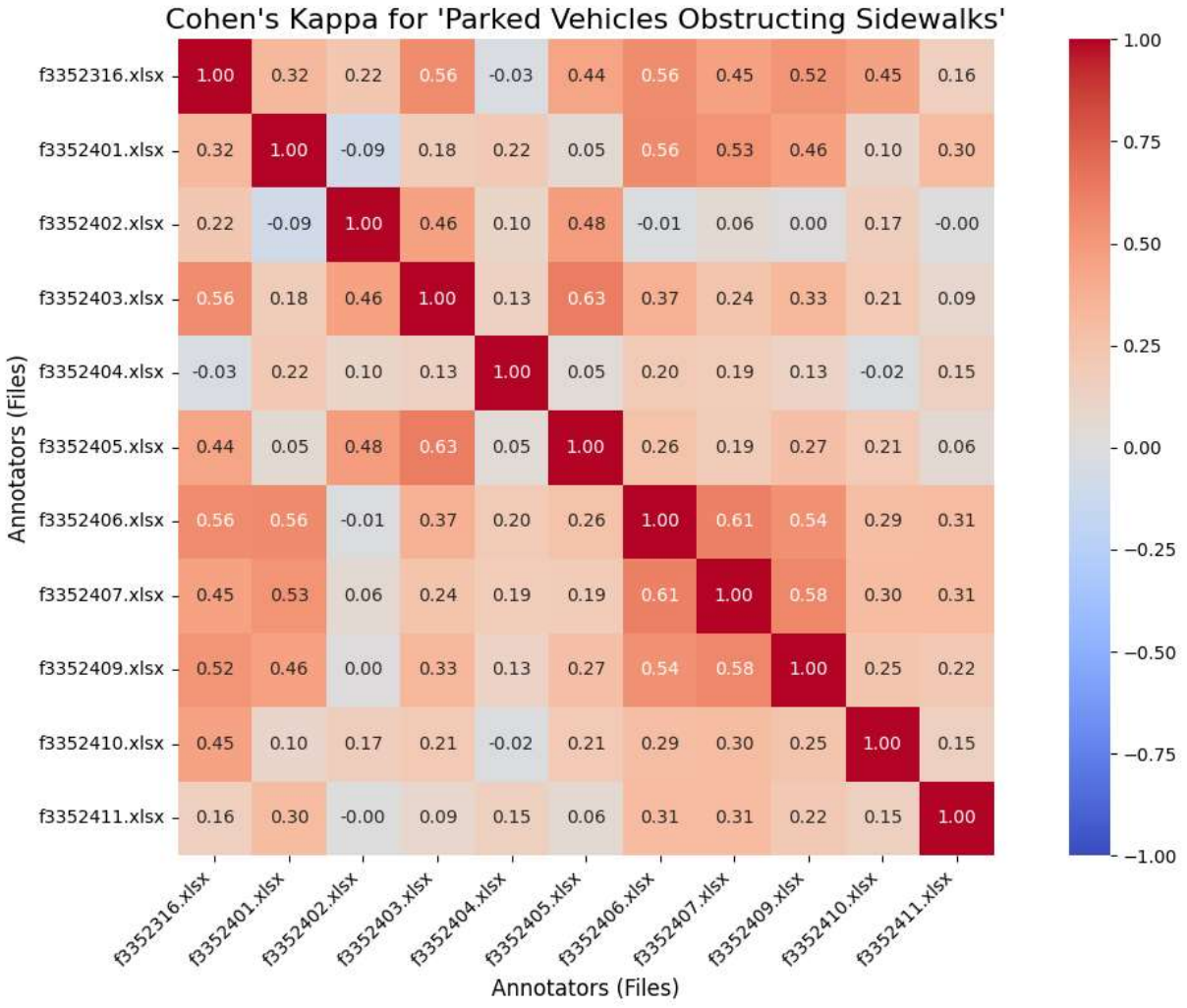
Cohen’s Kappa by Category



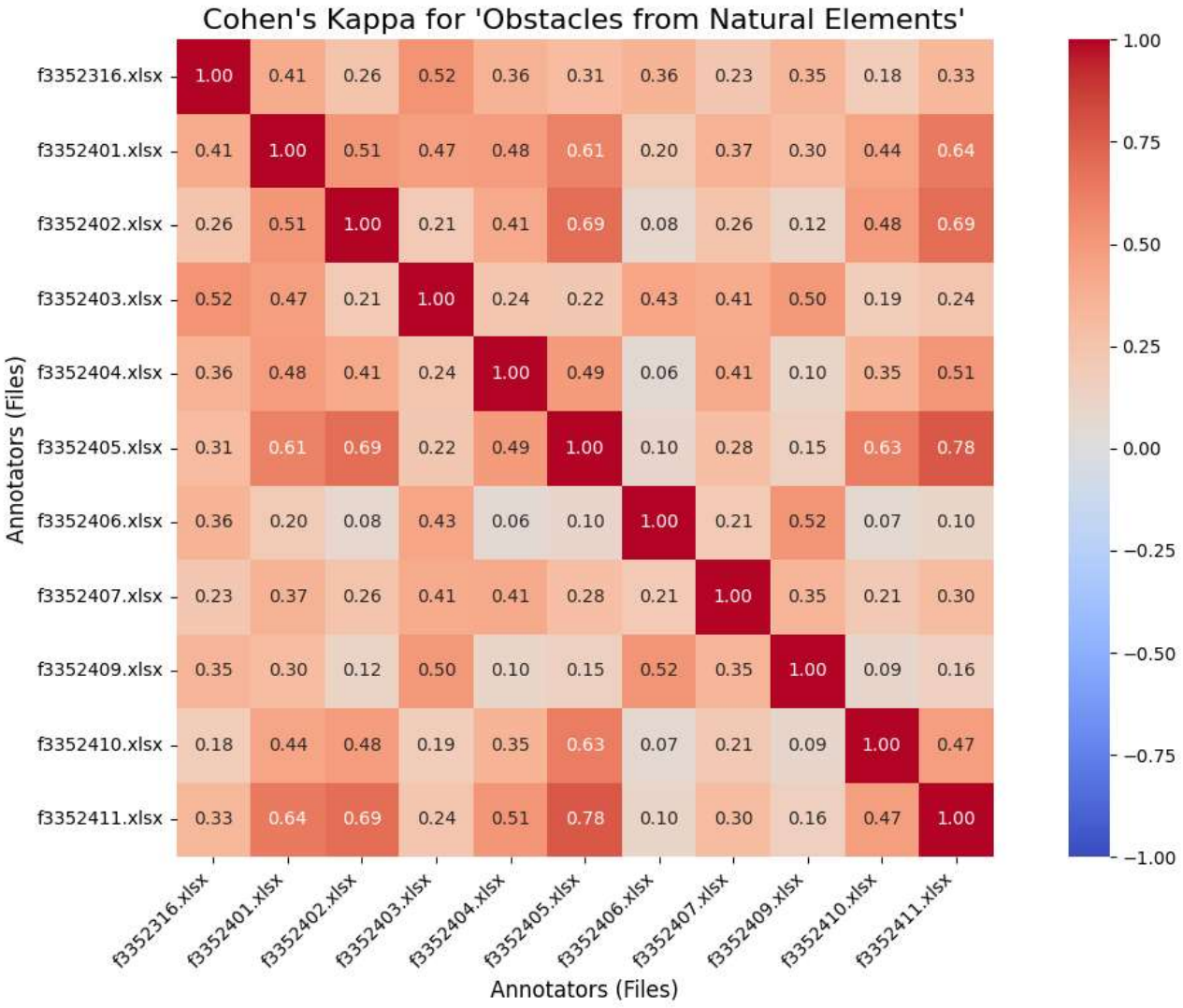


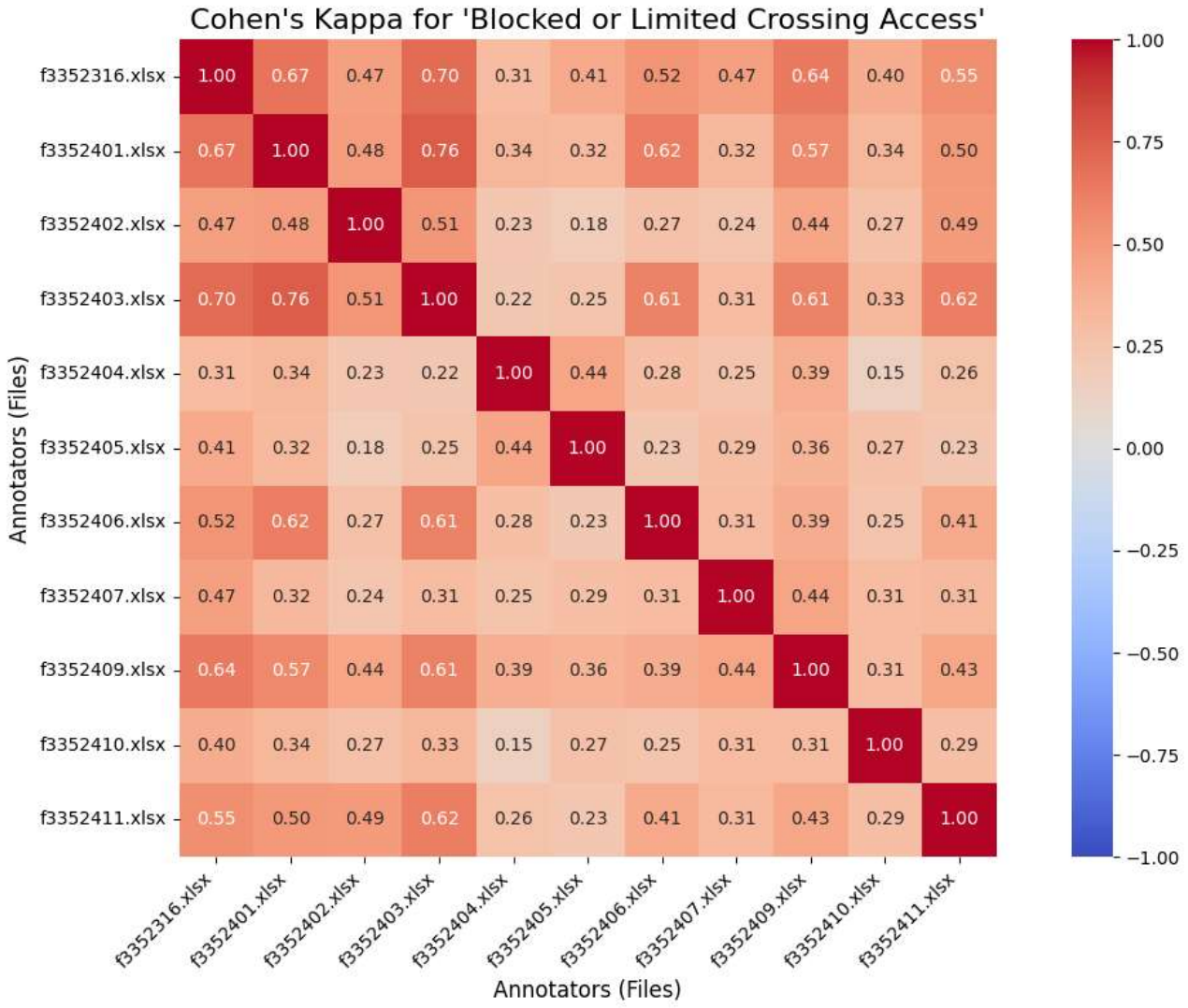
Cohen’s Kappa by Category



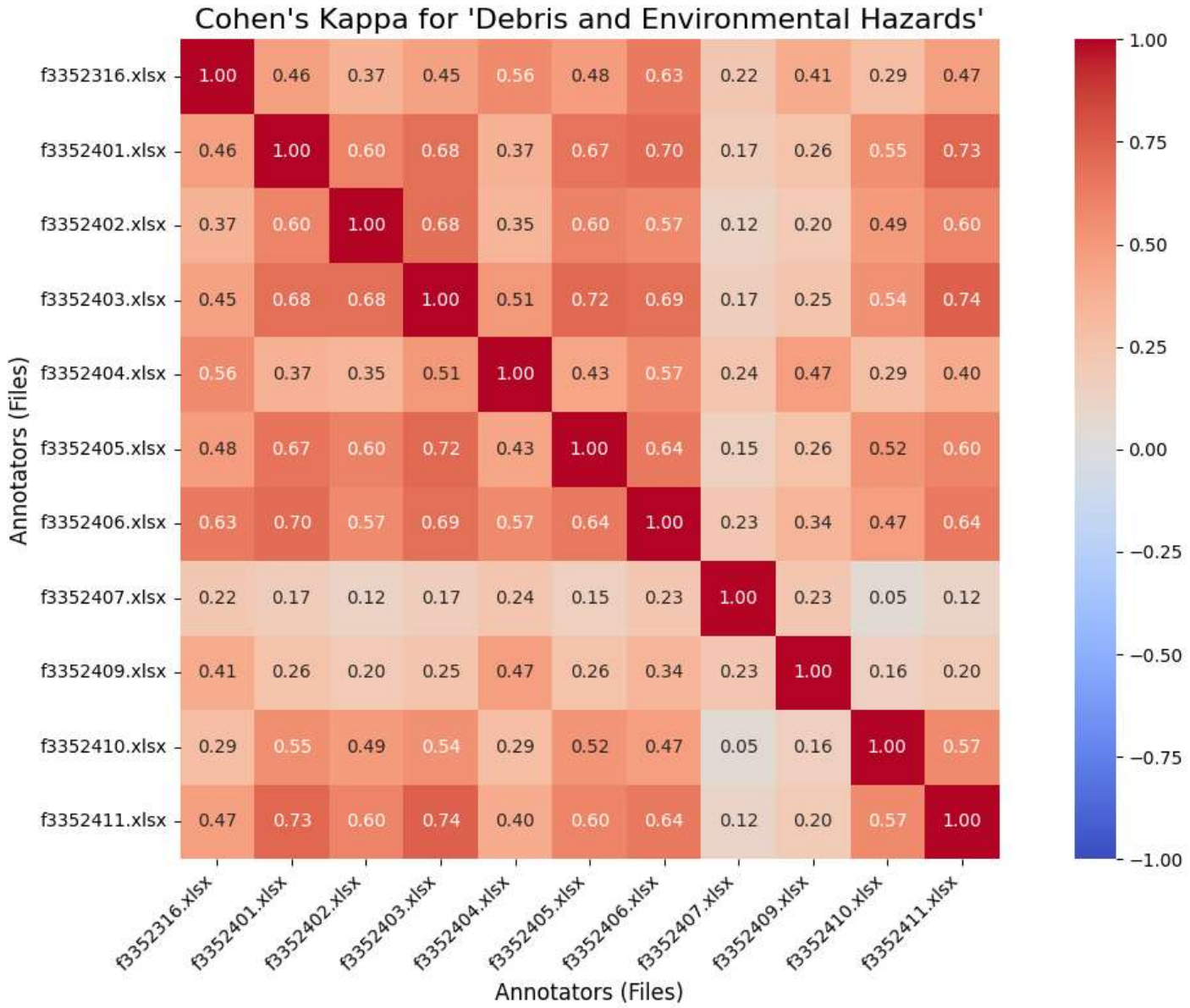


Cohen’s Kappa by Category

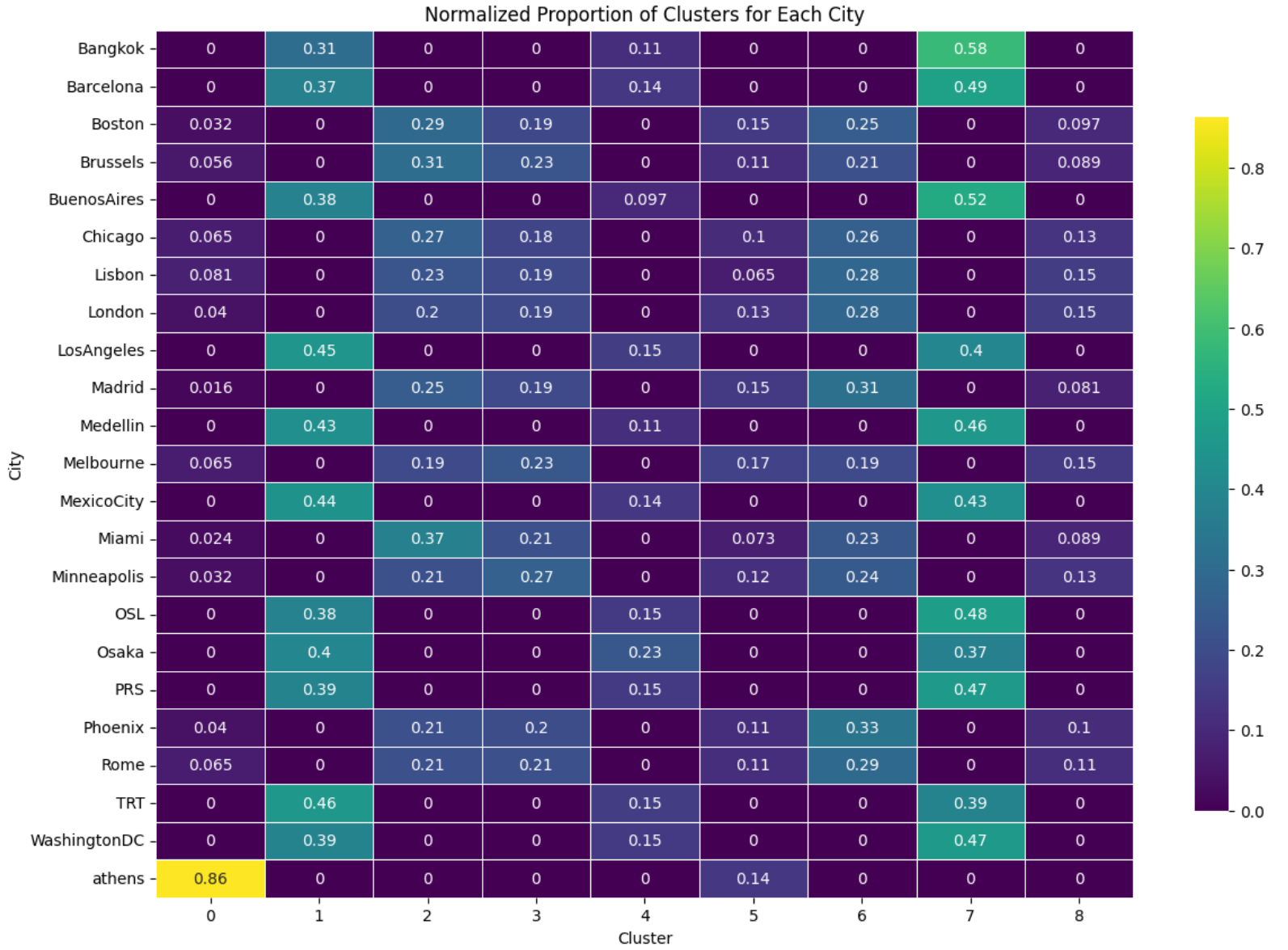




Cohen’s Kappa by Category



04 Conclusion

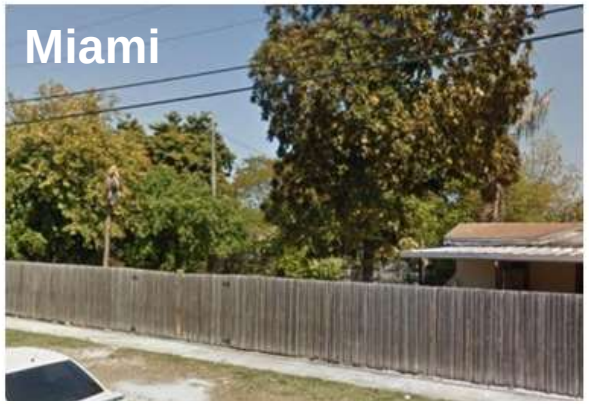


Dominant Features-Cluster 0:

- Energy
- Contrast
- Brightness
- Homogeneity
- Dissimilarity

- Analysis for Athens: Cluster Membership and City Similarity

Image Comparison



Limitations-Suggestions for Improvement

1. Memory Constraints and Image Selection:

- **Due to memory limitations, it was not feasible to load a large number of images simultaneously. As a result, the analysis was restricted to 124 images, corresponding to the number of available images for Athens.**

Suggestions for Improvement

1. Adopting More Efficient Color Extraction Algorithms:

- **Gaussian Mixture Models (GMMs)**. GMMs are capable of modeling overlapping clusters, making them better suited for identifying soft transitions and subtle variations in color. This method would improve the identification of nuanced colors and provide a richer representation of the images' color profiles.**

2.Optimizing Memory Usage:

- **Handling a larger dataset without exceeding memory limits: resizing or batching.**

Thank you
For watching

Limitations-Suggestions for Improvement

1. Memory Constraints and Image Selection:

- **Due to memory limitations, it was not feasible to load a large number of images simultaneously. As a result, the analysis was restricted to 124 images, corresponding to the number of available images for Athens.**

2. K-Means Clustering Parameter Restrictions:

- **The dominant color extraction was performed using K-means clustering. However, due to memory and computational constraints, the number of clusters (k) was limited to 7.**

Suggestions for Improvement

1. Adopting More Efficient Color Extraction Algorithms:

- **Gaussian Mixture Models (GMMs)**. GMMs are capable of modeling overlapping clusters, making them better suited for identifying soft transitions and subtle variations in color. This method would improve the identification of nuanced colors and provide a richer representation of the images' color profiles.**

2.Optimizing Memory Usage:

- **Handling a larger dataset without exceeding memory limits: resizing or batching.**