Jack Le
CS 4v98

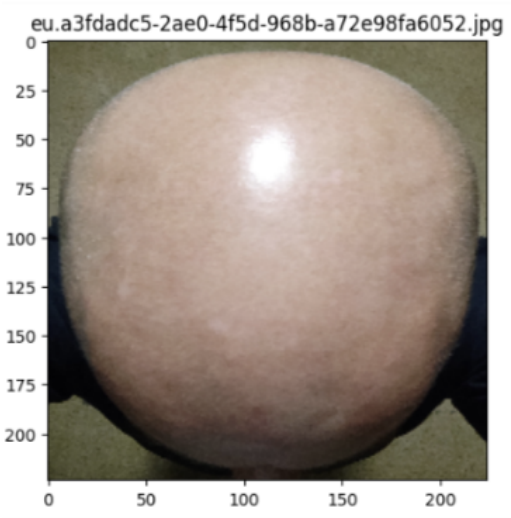
Dataset Creation

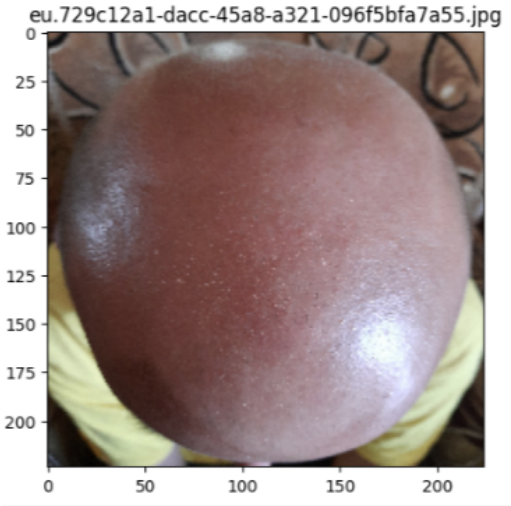
I refined my dataset to focus exclusively on overhead views of bald individuals, leading to a curated collection of approximately one hundred photographs. To ensure uniformity, each image was meticulously cropped and rotated, aligning them such that the front of each head faced the bottom of the photo, creating a standardized orientation. Following these adjustments, I employed data augmentation techniques to enhance the dataset's robustness and variability.

Jack Le
CS 4v98

Preprocessing

To prepare the photos for processing, I proprecessed the images in the dataset. Each image is opened using OpenCV, which initially loads the images in BGR format. They are then converted to the more standard RGB format, commonly used in various image processing and machine learning applications. Each image is resized to a uniform size of 224x224 pixels, which helps reduce computational load. Finally, each processed image and its corresponding filename are stored in separate lists, allowing for organized access to the image data and associated metadata, facilitating further analysis or processing.

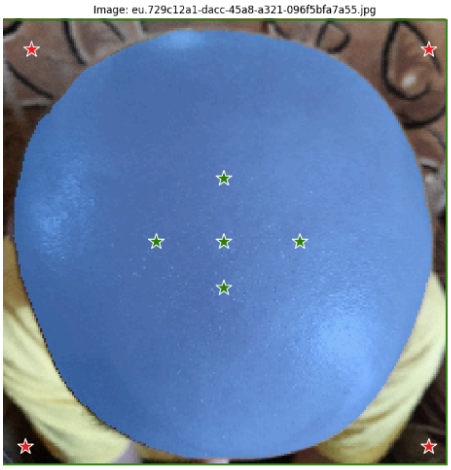


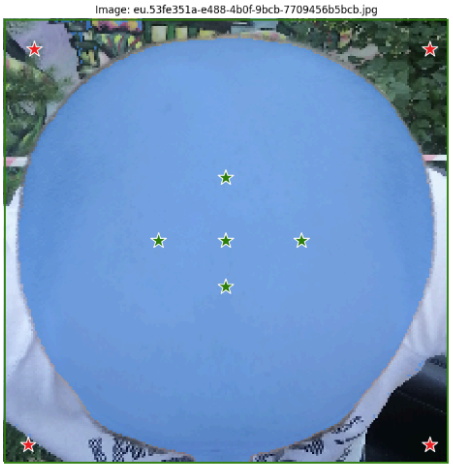


Jack Le
CS 4v98

Creating Segmentation Masks

I have been utilizing a new model called Segment Anything, which allows customization of segmentation of objects. The model efficiently creates segmentation masks for images by utilizing a machine learning predictor that processes batched input points and bounding boxes. For each image, the predictor is set with predefined input points and a bounding box encompassing the full image dimensions (224x224 pixels). These inputs are used to generate segmentation masks, alongside associated scores and logits, with the ability to output multiple masks per image. The process involves visualizing each segmented image along with its mask, bounding box, and marked input points, enhancing interpretability and verification of the segmentation quality. After processing, the model selects and stores the best-performing mask for each image based on the highest score, compiling these into a list. This creates accurate segmentation masks through batch processing.



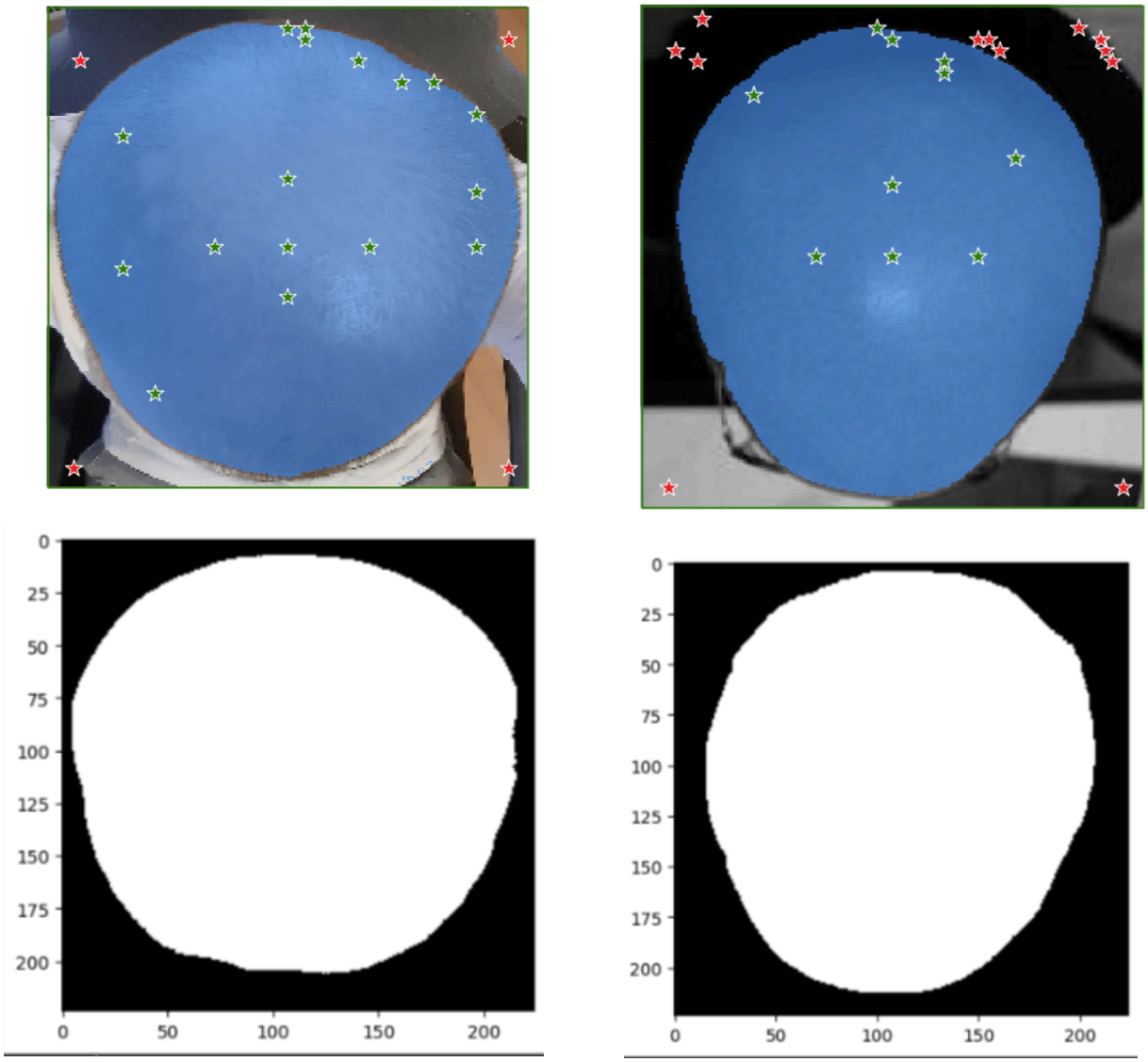


However, obstructions such as glasses, objects in the foreground, or complex backgrounds lead to inaccurate segmentation masks in some photos. These elements can introduce noise and distort the visual cues that the model relies on to delineate and classify different regions of an image effectively. This leads to unusable segmentation masks.

To address this, I implemented a strategy of iterative refinement. The process involves re-engaging the model multiple times for the same image, each time selecting new and potentially more informative points based on the inadequacies observed in the previous output. This method of selectively adjusting input points allows the model to reconsider the image with a slightly different perspective or focus, thereby potentially capturing details it missed earlier. This iterative adjustment and reprocessing help in fine-tuning the segmentation results until a satisfactory mask is obtained. By repeatedly modifying the input points and allowing the model to reassess the image, the accuracy of the

Jack Le
CS 4v98

segmentation masks is significantly improved, ensuring that they are both precise and reliable.



Jack Le
CS 4v98

Extracting Head Shape

This section will discuss the methods of extracting head shape features that I utilized on the segmentation masks.

Head Aspect Ratio

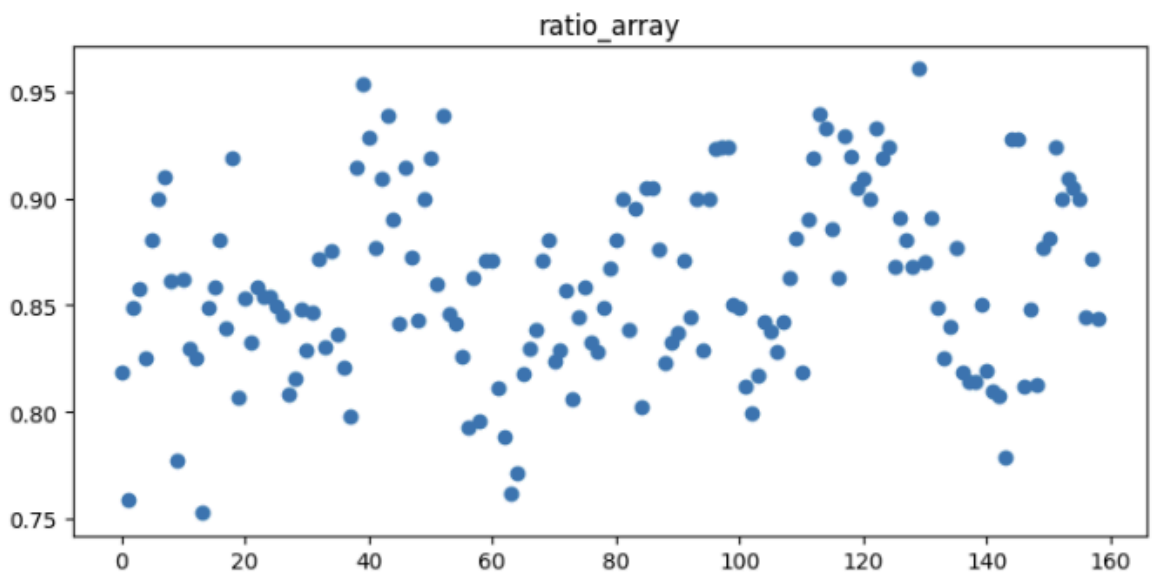
I calculate the width and length of each segmentation mask in the segmentation_masks array and store these dimensions in the widths and lengths lists. For valid masks, I determine the horizontal and vertical ranges of the true indices (object boundaries) to calculate the width and length. The ratio between them is then calculated.

We can represent this mathematically:

Width = $\max(x_{indices}) - \min(x_{indices})$

Length = $\max(y_{indices}) - \min(y_{indices})$

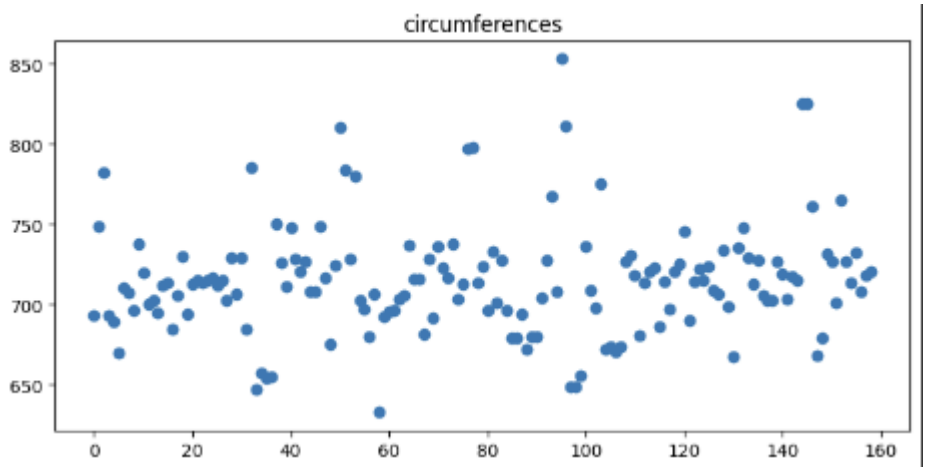
Head Aspect Ratio = Width / Length



Head Circumference

Jack Le
CS 4v98

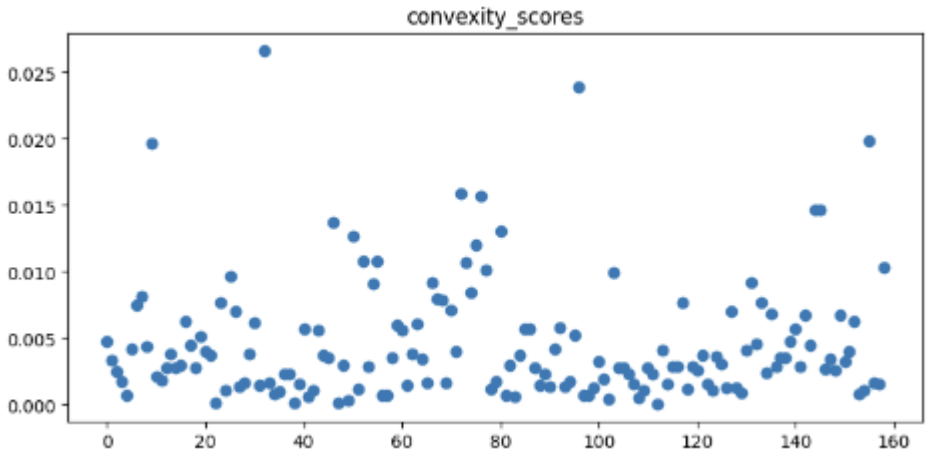
To calculate the circumference of each segmentation mask, I first convert the mask to a binary format, marking pixels of the mask as 1 and the rest as 0. Using OpenCV's findContours method, I identify the external boundaries of the mask and select the largest contour based on area for precise measurement. The circumference is then determined by calculating the perimeter of this contour with OpenCV's arcLength function. I append the resulting circumference to a list, or add zero if no contours are found, ensuring an accurate representation of each mask's perimeter.



Head Convexity

I calculate convexity scores for segmentation masks by first converting each mask into a binary format and identifying the largest contour using OpenCV's findContours function. I then compute the convex hull of this contour and measure its convexity with cv2.matchShapes, which assesses how the shape deviates from perfect convexity. A score close to 0 indicates high convexity, representing the head's roundness, while higher scores suggest increasing irregularities. Scores are categorized as follows: very low (less than 0.01) indicates significant deviation; moderate low (0.01 to 0.05) shows less deviation; relatively higher low (0.05 to 0.1) suggests closer to convex; and scores near or above 0.1 denote minimal concavities.

Jack Le
CS 4v98



Head Eccentricity

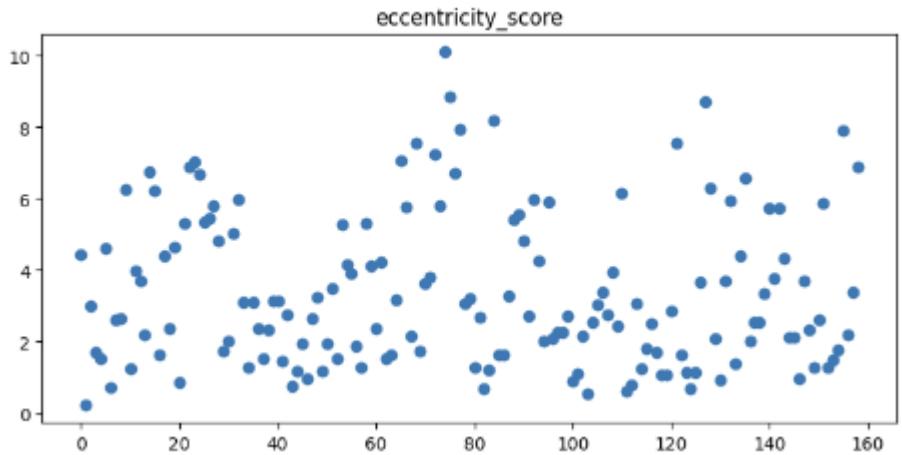
I calculate the centroid of each head from the segmentation masks by determining the geometric center using OpenCV's moments function, which averages all points in the mask's largest contour. This centroid is then compared to the center of the smallest enclosing circle for the head to calculate eccentricity, measuring the Euclidean distance between the two points. This metric assesses how the head's mass is distributed relative to its geometric center, with low eccentricity indicating symmetry and high eccentricity suggesting significant asymmetry or deviations.

This can be represented mathematically:

$$\text{Eccentricity} = \sqrt{(C_x - x)^2 + (C_y - y)^2}$$

- Cx and Cy are the x and y coordinates of the centroid of the head contour, respectively.
- x and y are the x and y coordinates of the center of the minimum enclosing circle that encompasses the head contour.

Jack Le
CS 4v98

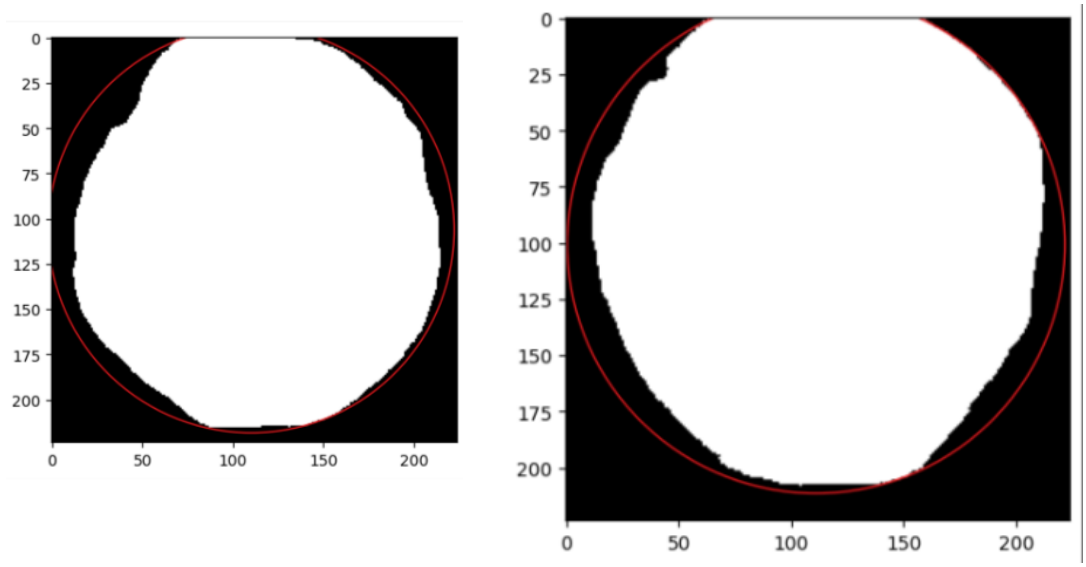


Head Compactness

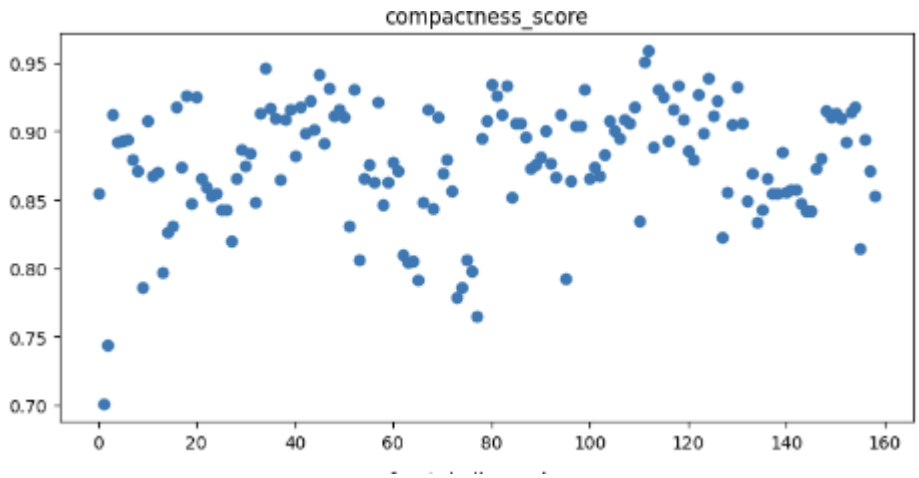
I calculate the area of the smallest circle that can entirely enclose each head using the minimum enclosing circle, computed with OpenCV's minEnclosingCircle function. The compactness of the head is then quantified by dividing the area of the head by the area of this circle. I then process each mask to identify contours, select the largest to fit the circle, calculate the circle's area, and compare this to the mask's area obtained by summing its pixel values. This results in a compactness score for each mask, indicating how well the head fills its bounding circle, which is visualized by overlaying the circles on the masks.

This can be represented mathematically as:

Compactness = $\text{Area}_{\text{mask}}$ / $\text{Area}_{\text{circle}}$



Jack Le
CS 4v98

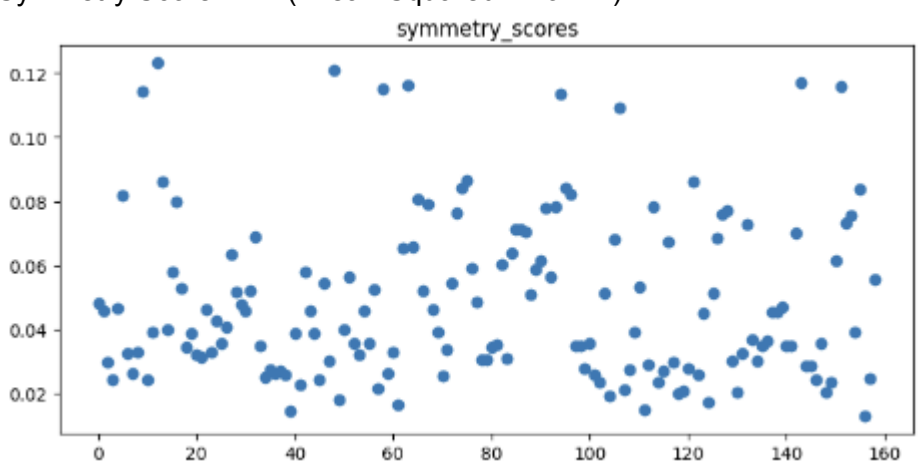


Symmetry

I calculate symmetry scores for segmentation masks by first ensuring each mask is in integer format, then splitting each mask into left and right halves. I mirror the right half to compare it against the left using the Mean Squared Error (MSE) method. This method quantifies the similarity between the two halves by computing the average of the squares of differences between corresponding pixel values. Lower MSE values indicate higher symmetry, showing that the two halves are very similar, while higher values suggest greater asymmetry, which could be due to natural variations in head shape or distortions in the mask. I generate a symmetry score for each mask to assess its bilateral symmetry effectively.

We can represent this mathematically as:

Symmetry Score= 1 / (Mean Squared Error+1)

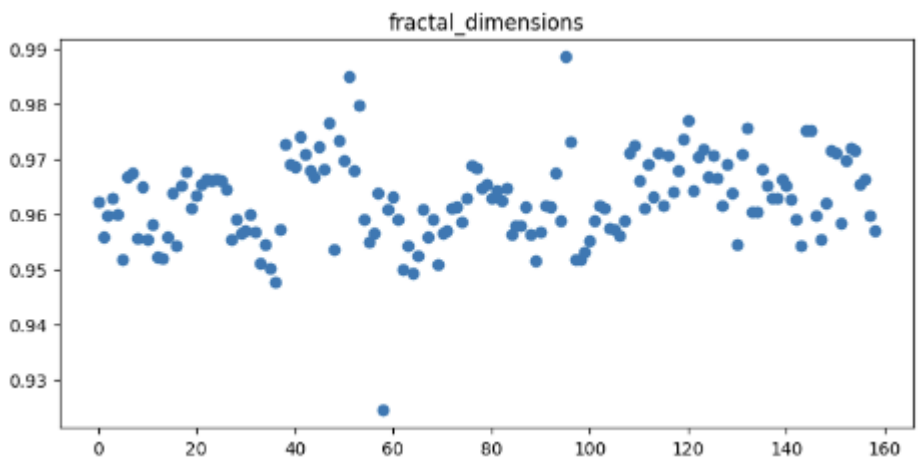


Fractal Dimensions

Jack Le
CS 4v98

Fractal dimension quantifies the complexity of a pattern, reflecting how the outline of a head changes with different measurement scales. For head outlines, a higher fractal dimension indicates a more complex and rough outline, possibly suggesting specific biological traits.

To compute this, I use the Box-Counting Method: I overlay a grid of varying-sized boxes over the image, counting boxes that intersect the head outline, and plot the log of the box count against the log of the box size. The slope of this log-log plot, determined via linear regression, estimates the fractal dimension.

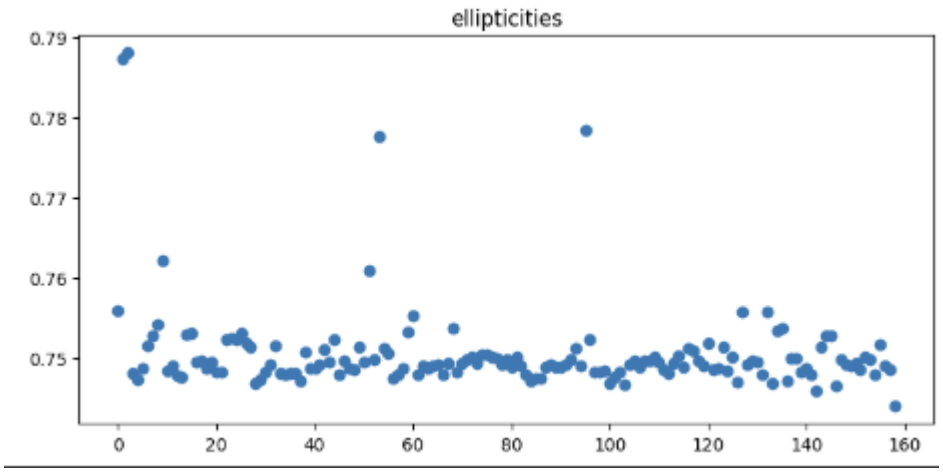


Ellipticity

Head ellipticity measures the extent to which a head outline deviates from a perfect ellipse, serving as an indicator of symmetry and regularity. In this process, each segmentation mask is converted to binary, and the largest contour is used to fit an ellipse using OpenCV's fitEllipse function. The ellipse's geometric properties—center, axes lengths, and rotation angle—are calculated and overlaid onto the original head outline. Ellipticity is then determined by comparing the area of the head and the fitted ellipse, and is expressed as one minus the ratio of the head area to the ellipse area. This quantifies how closely each mask aligns with an elliptical shape, with the results visualized to highlight variations in head shapes.

Ellipticity = $1 - \text{Area}_{\text{mask}} / \text{Area}_{\text{ellipse}}$

Jack Le
CS 4v98



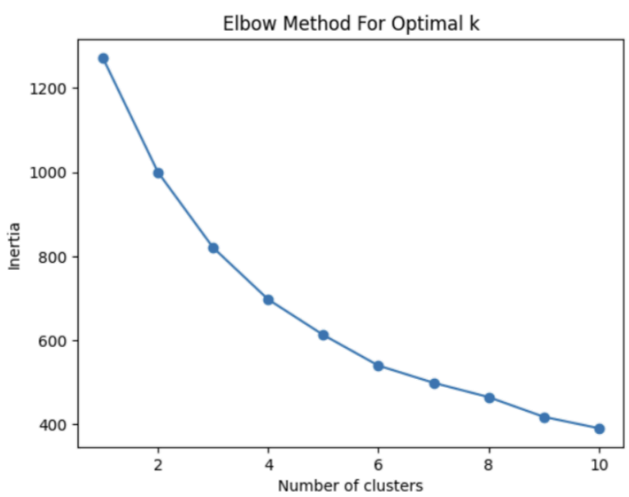
Jack Le
CS 4v98

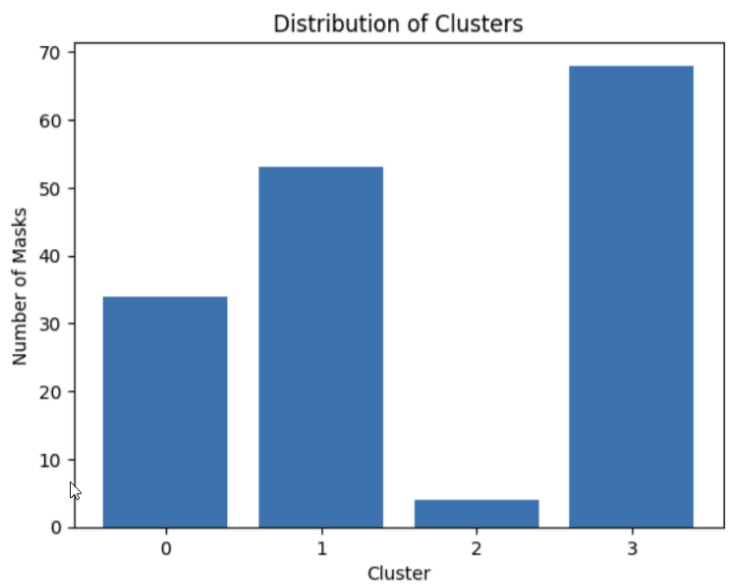
Clustering

I perform a clustering analysis on the extracted features. To determine the optimal number of clusters, I employ the elbow method. This method involves plotting the number of clusters against the inertia, which is the sum of squared distances of samples to their nearest cluster center. As I increase the number of clusters, the inertia decreases. The point where the inertia begins to decrease at a slower rate—visually forming an "elbow" shape—is considered optimal. In this analysis, the elbow method indicates that four clusters are ideal.

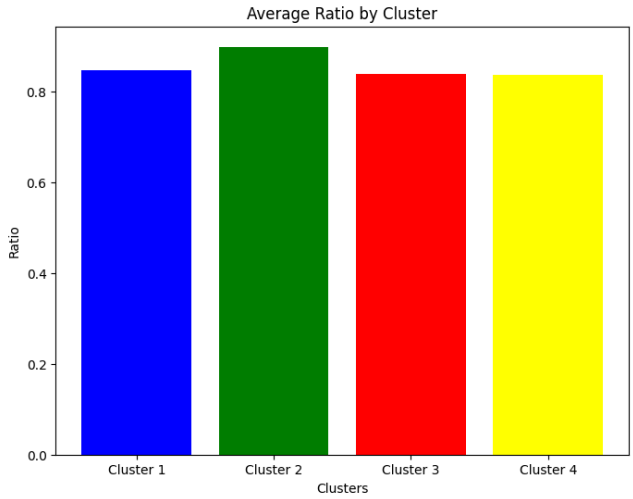
Following this, I apply KMeans clustering, which partitions the data into four clusters by assigning each data point to the nearest cluster center while keeping the inter-cluster variance as low as possible. This iterative algorithm recalculates cluster centers by taking the mean of all data points assigned to a cluster and reassigns data points to the cluster where they are closest to the new center. This process repeats until the cluster assignments no longer change significantly.

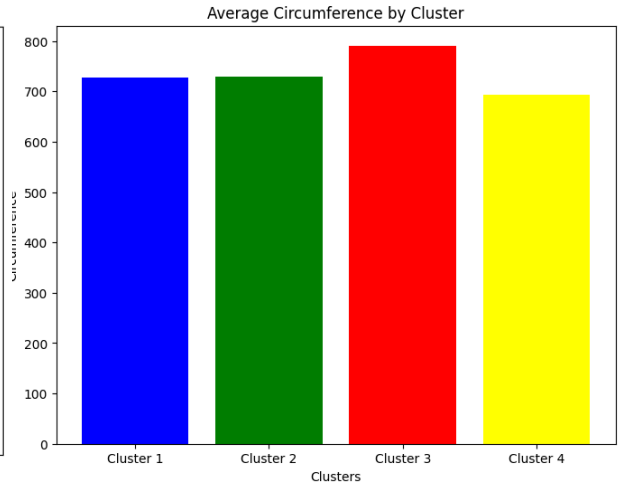
After clustering, I calculate and display the average values of the features within each cluster to delineate their unique characteristics.

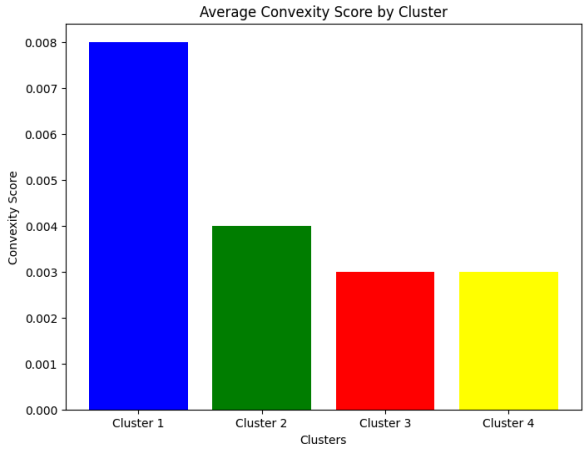


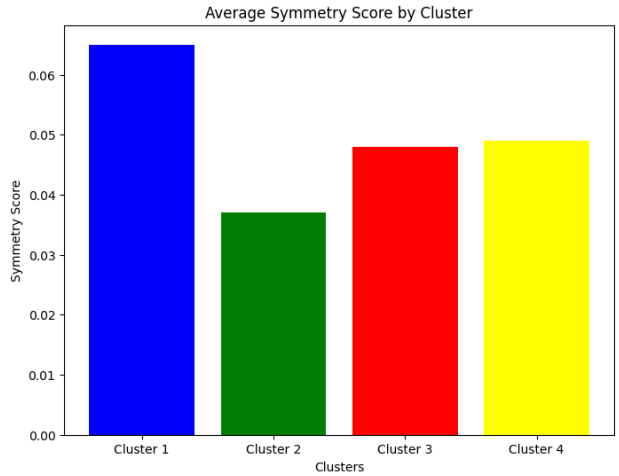


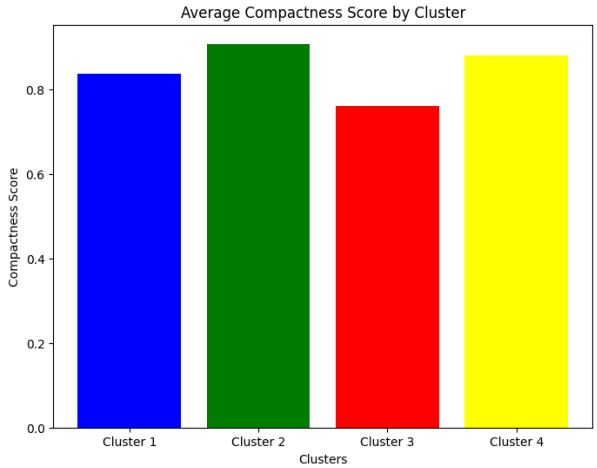
Jack Le
CS 4v98

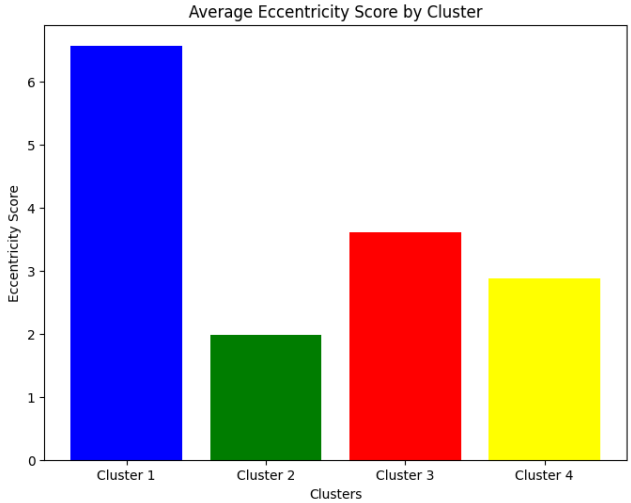
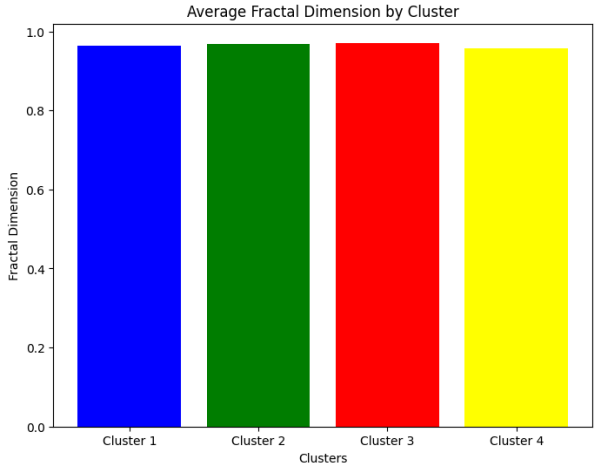
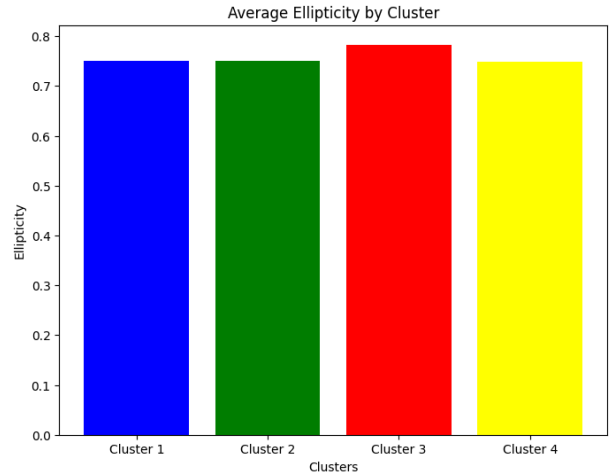












Jack Le
CS 4v98