Rockery : Color Clustering Module

Research

Key Concepts

- Color Clustering ⇒ a technique used to group similar colors in an image
  - K-means ⇒ an unsupervised ML algorithm
- image loading and processing libraries
  - PIL - for opening & processing images
  - OpenCV - processing images
  - matplotlib - for visualizing Color Clusters
- Clustering algorithms
  - K-means
  - DBSCAN ⇒ density based clustering
- feature Extraction ⇒ Extracting the most important colors from the image (eg pixel Sampling or histograms)

K Means

- alg iteratively divides data points into K clusters by minimizing the variance in each cluster
- how it works
  1. each point randomly assigned to a K cluster
  2. Compute the centroid (functionally the center) for each cluster
  3. reassign each data point to the cluster w/ the closest center
  4. Repeat until cluster assignments stop changing
- need to select K number of clusters to group the data into
- Elbow method lets us graph the inertia & visualize the point it decreases linearly
  - point referred to as elbow ⇒ good estimate for the best K value

Image Color Extraction using K means Clustering

- two types of graphs: raster & vector
  - vector graphs ⇒ based on geometry shapes
    - zoom in, shape is always clear
    - drawn by clear geometric formulas
  - Raster graphs ⇒ array like data mix of pixels, smallest unit of a dot containing color information
    - number of pixels ⇒ resolution
- Color Spaces
  - Every pixel in raster is given a color
  - image = data set of colors
    - define the colors w/ numbers
  - Color map ⇒ one dimensional color information
  - pallets & Color Swatches ⇒ 2D
  - default for ML is RGB

- deconstruct the photo into pixels w/ RGB data points

- K means Clustering
  - Similarities recognized as Euclidean Distance
  - dynamic process to find the centroid
    - other points distance from the centroid
  - non deterministic alg ⇒ outputs can be adjusted
  - can stop when we find K clusters consistent w/ intuition
  - use the Elbow test or Silhouette to find the optimal K value
    - Elbow not always helpful
  - when the K value is larger than 5 ⇒ not always visually apparent

Kick off

Project: functional Color analyzer written in python

- Published to a custom domain
  - need to wrap it in an API ⇒ probably fast API
    - where would it be hosted? github actions ?
      - Easiest would be to test locally

Getting Started

- if over a certain % of the image is white Remove the background ?

- for the inertia ⇒ list the graph points and find the one where it dips

Day 1 Summary

- Conducted trial runs of finding k means
- Cluster quality
  - Elbow method ⇒ identifies the "elbow" point where the SSE starts to level off
  - Silhouette Score ⇒ how well Seperated the Clusters are
  - Davies-Bouldin ⇒ Balances Cluster Compactness & Seperation
  - Calinski-Harabasz ⇒ Emphasize between Cluster dispersion
- Next Steps
  - find a way to average out the k values
  - try on a more colorful image
  - maybe all 4 graphs w/ the different k values

Day 3

- Created a function that Saves the Scatter plots
- CH is Consistently overkill
- mode either right on the money or Slightly under
- run again Comparing median, mean, mode.
- which one to go with
  - ~~mode is still mostly useless ⇒ Removing~~
  - if they agree on the # of Clusters go w/ that
  - find the Correlation Coefficient
    - if it is extremely close to a line (the abs is close to 1)
      - lowest number

- intermediately close
  - middle number
- all over the place
  - highest number

- low, med, high choice
  - low : 0.99 - 0.97
  - med : 0.96 - 0.6
  - low : 0.6 - 0

if 2 # are the same (low or high)
3 lower
} higher

- all the same # ⇒ go w/ that number

2nd Try Alg

- linear Corr pretty much useless
- Also Ch method pretty much useless ⇒ overkill
- Solid Color Background Skewing Results
- maybe look at main colors and how distinctly different they are?
    - distance formula

Not Gonna Work

3rd Shot

- difference from the mean is typically between 15 and 53 when picked

- distance ranges 110 - 200
  - if no # are in range pick the one closest
  - if more than one w/i range ⇒ pick the lowest

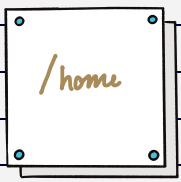
} possible Alg

Fast API

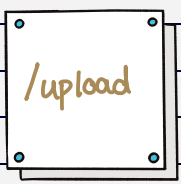
- validation for all data types
- Security already integrated → if we need it

End points

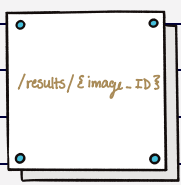
Render Probably the best idea to host this



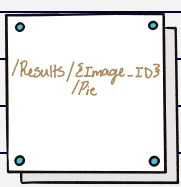
- Erases any images that
- back to the homepage
- GET



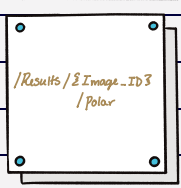
- POST
- upload the image



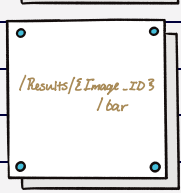
- GET
- endpoint to retrieve processed image
- all graphs



- GET
- original image + Pie chart



- GET
- original image + Circular polar graph



- GET
- Original image + 3D bar graph