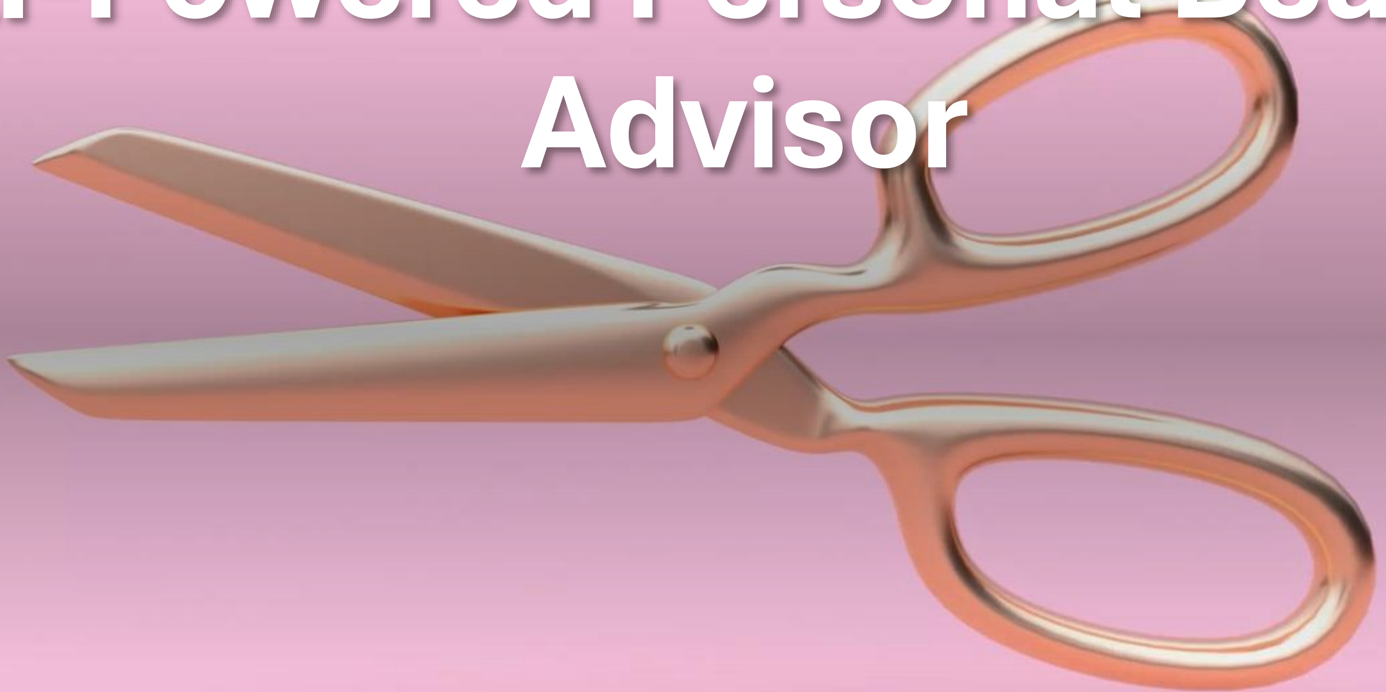
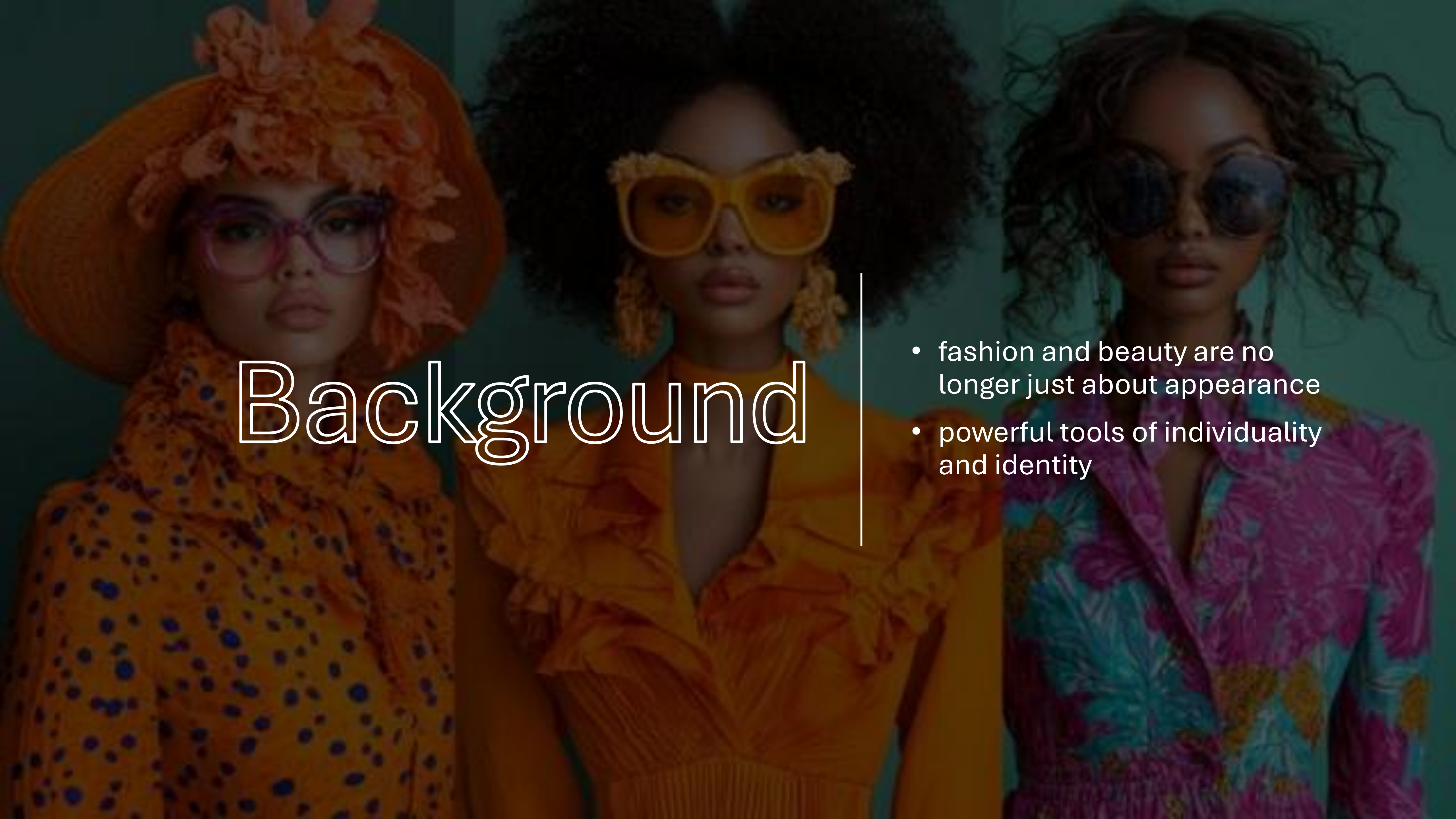
부산대학교 정보컴퓨터공학부 2025전기 졸업과제

AI-Powered Personal Beauty Advisor



분과 A - 11 <Underdog> 팀
지도교수: 전상률
Han Nwae Nyein, Nyi Nyi Htun, 전재원



Background

- fashion and beauty are no longer just about appearance
- powerful tools of individuality and identity

Personal Color Analysis

- Popular method for beauty & feature
- Matches colors to natural features
- Limitation
  - Subjective Judgment
  - High Cost
  - Limited Visualization

How do we determine personal color?

Armocromia

- Conventional
- 12 seasonal types
- based on the overall face

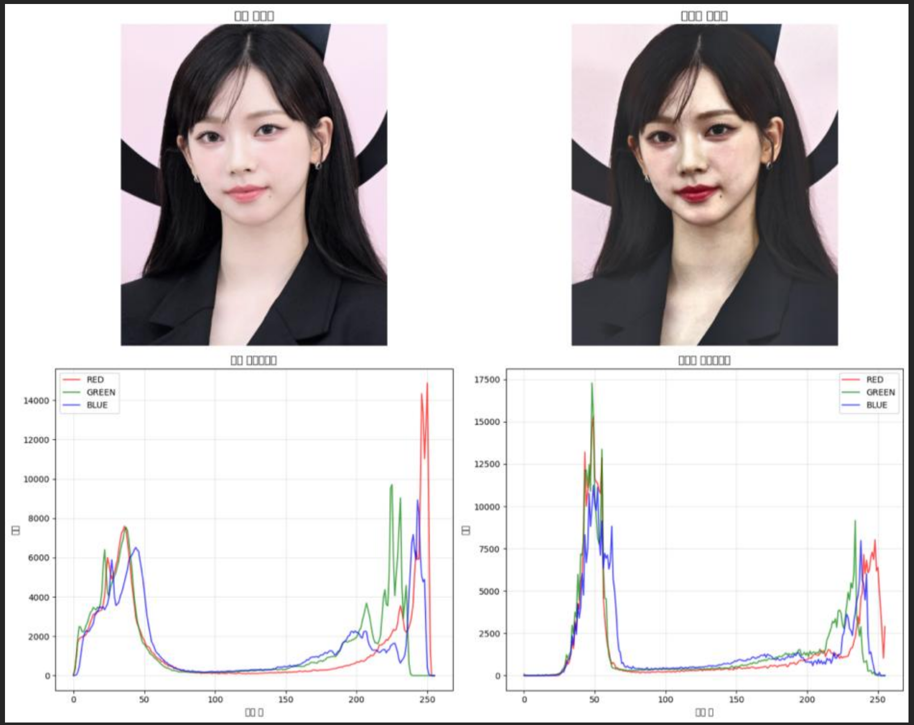
But this loses relevance!!!

But this loses relevance if the user changes their lenses or hair color.

Solution: New Personal Color System

- we created an entirely new personal color system
- based only on skin tone
- clustering using K-Means.
- gives a more stable and practical foundation for recommendations

Data Preprocessing



Dataset: 5000 images from ArmocromiaDataset

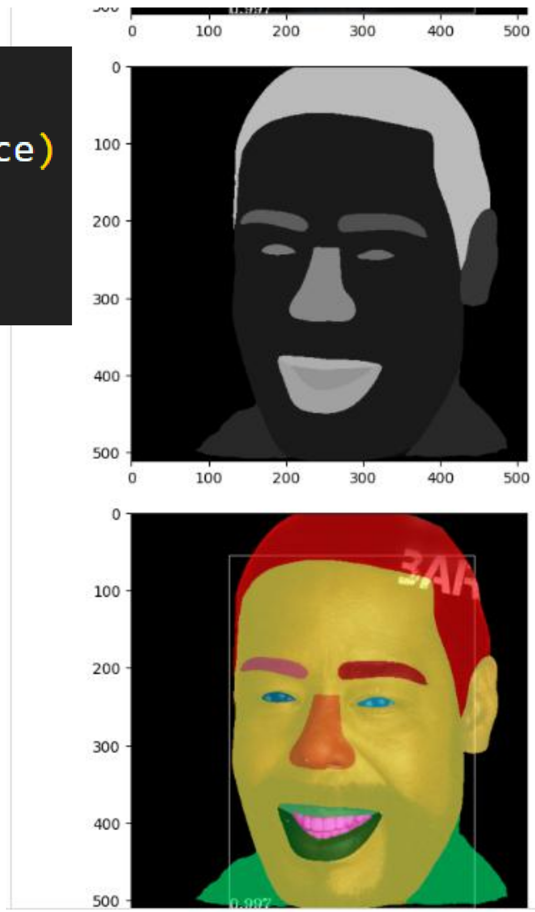
```python
def analyze_lighting_conditions(image_np):
    """이미지의 조명 상태를 분석하여 보정 전략을 결정"""
    lab = cv2.cvtColor(image_np, cv2.COLOR_RGB2LAB)
    l_channel = lab[:, :, 0]
    mean_brightness = np.mean(l_channel)
    std_brightness = np.std(l_channel)
    dark_pixels = np.sum(l_channel < 85) / l_channel.size
    bright_pixels = np.sum(l_channel > 170) / l_channel.size
    return {
        'mean_brightness': mean_brightness, 'std_brightness': std_brightness,
        'dark_ratio': dark_pixels, 'bright_ratio': bright_pixels,
        'is_underexposed': mean_brightness < 120 and dark_pixels > 0.3,
        'is_overexposed': mean_brightness > 180 and bright_pixels > 0.2,
        'has_low_contrast': std_brightness < 25, 'has_uneven_lighting': std_brightness > 50
    }
```

```python
def comprehensive_lighting_correction(image_np, lighting_info=None):

    if lighting_info['is_underexposed']:
        corrected = shadow_highlight_correction(corrected, shadow_amount=0.3, highlight_amount=-0.1)
        corrected = gamma_correction(corrected, 0.7)
        correction_log.append("Underexposure correction: shadow lift + gamma 0.7")
    elif lighting_info['is_overexposed']:
        corrected = shadow_highlight_correction(corrected, shadow_amount=0.0, highlight_amount=-0.4)
        corrected = gamma_correction(corrected, 1.3)
        correction_log.append("Overexposure correction: highlight recovery + gamma 1.3")

    if lighting_info['has_low_contrast']:
        clip_limit = 4.0 if lighting_info['std_brightness'] < 15 else 2.5
        corrected = adaptive_histogram_equalization(corrected, clip_limit=clip_limit)
        correction_log.append(f"Low contrast correction: CLAHE (clip_limit={clip_limit})")
    elif lighting_info['has_uneven_lighting']:
        corrected = adaptive_histogram_equalization(corrected, clip_limit=2.0, tile_grid_size=(6, 6))
        correction_log.append("Uneven lighting correction: Soft CLAHE")

    if lighting_info['std_brightness'] < 30:
        corrected = unsharp_masking(corrected, strength=0.3, radius=1.2)
        correction_log.append("Sharpening applied")
```

Face Detection and Segmentation

```
# Facer 라이브러리의 얼굴 관련 모델들
face_detector = facer.face_detector('retinaface/mobilenet', device=device)
face_parser = facer.face_parser('farl/celebm/448', device=device)
print("✓ Facer models loaded successfully.")
```

1. **Face Detection:** First, a face detector identifies the location of the face.
2. **Face Parsing:** A semantic segmentation model then analyzes the facial region, creating a detailed map where every pixel is assigned a label (e.g., skin, hair, eyes, lips).



Feature Extraction (Mini Kmeans)

```python
# --- Mini K-Means for Feature Extraction ---

# 'pixels' contains thousands of Lab color values from one person's skin
if len(pixels) < 10:
    # Handle cases with insufficient skin pixels
    representative_colors = np.zeros((10, 3))
else:
    # 1. Initialize K-Means to find 10 clusters (representative colors)
    mini_kmeans = KMeans(n_clusters=10, n_init='auto', random_state=0)

    # 2. Sort the skin pixels into 10 groups
    mini_kmeans.fit(pixels)

    # 3. Get the center of each cluster, which is the representative color
    representative_colors = mini_kmeans.cluster_centers_
```

Feature Scaling

The entire feature matrix is standardized using *StandardScaler* from *scikit-learn*.

```
# This code is located within the 'extract_skin_features' function

# 'color_features' is the raw table of 30D feature vectors before scaling.

# 1. Initialize the StandardScaler object.
scaler = StandardScaler()

# 2. Fit the scaler to the data and then transform it in one step.
#    This command learns the properties of your data (mean, std dev)
#    and applies the scaling formula.
scaled_features = scaler.fit_transform(color_features)

# 'scaled_features' is the final, standardized data used for clustering.
return scaled_features, labels, paths_list, scaler
```

StandardScaler: The *fit_transform* Method

- fit() (Learn):
  - Calculates the mean and standard deviation of the input data.
- transform() (Apply):
  - Scales each value using the formula:

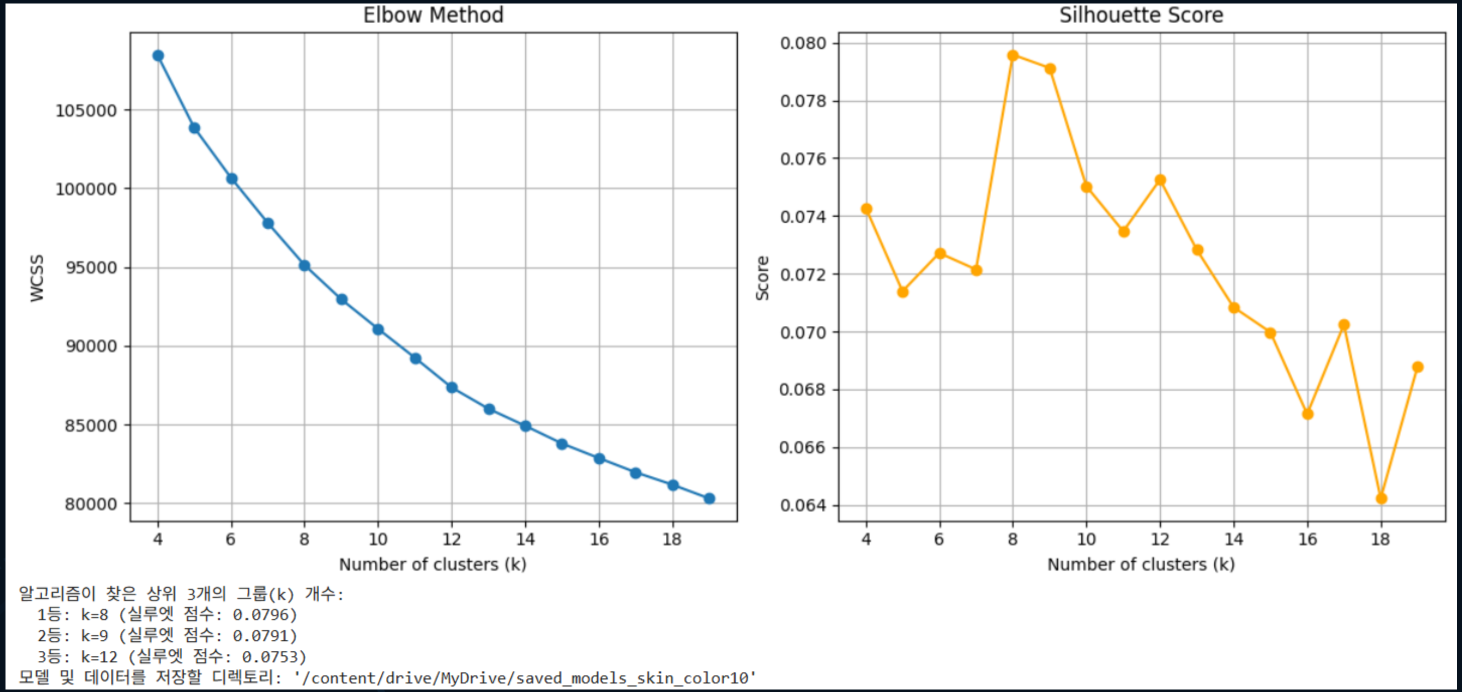
$$z = \frac{(x - \mu)}{\sigma}$$

- Result: Features are centered at 0 with a standard deviation of 1.

```python
# ... inside the main() function ...

# The 'final_features' variable holds the feature vectors for all images.
# 'optimal_k' is the best number of clusters you found (e.g., 8).

# This is the "big K-Means" part
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', n_init=12, random_state=0)
cluster_labels = kmeans.fit_predict(final_features)

# After this, the model is saved
joblib.dump(kmeans, os.path.join(MODEL_DIR, 'kmeans_model.joblib'))
```

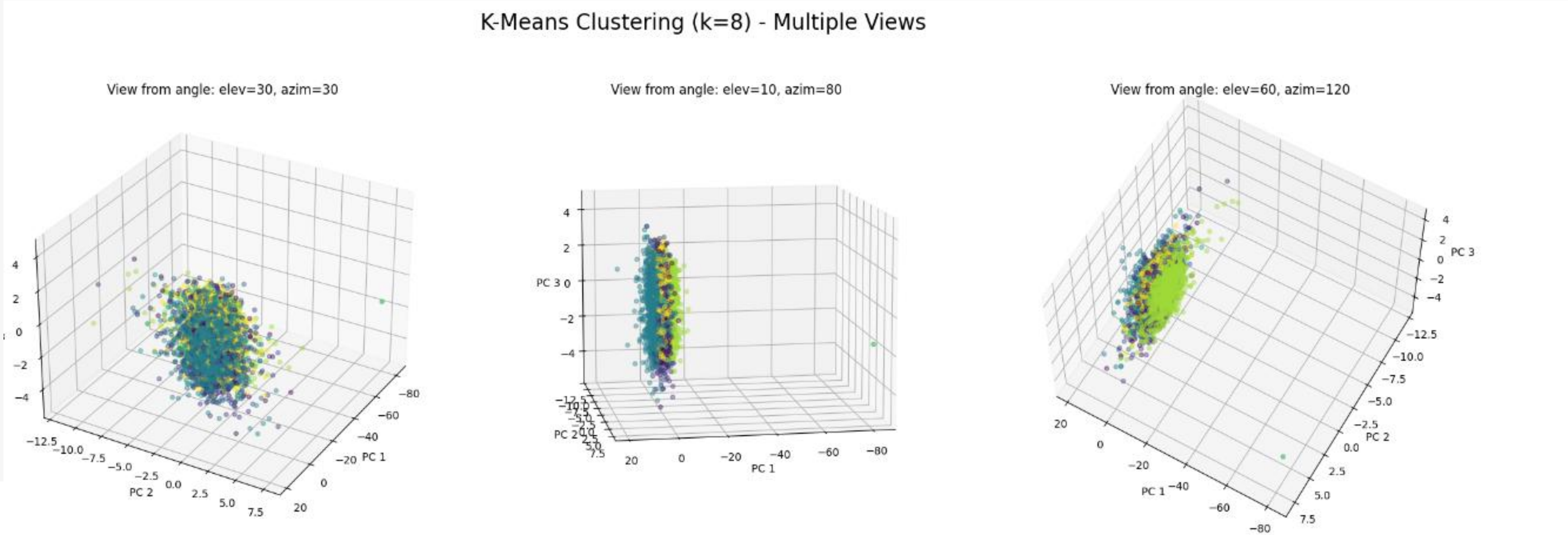
Model Training (Big Kmeans)

- Trained on **30D skin-tone scaled feature vectors**
- Used **K-Means++ initialization**
- Tested multiple **k values (4–20)**
- Model saved with **joblib**



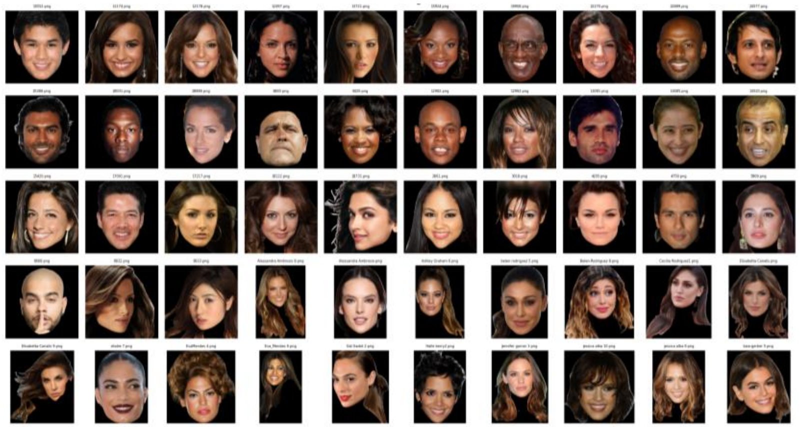
Selection of K Value and Clustering Evaluation

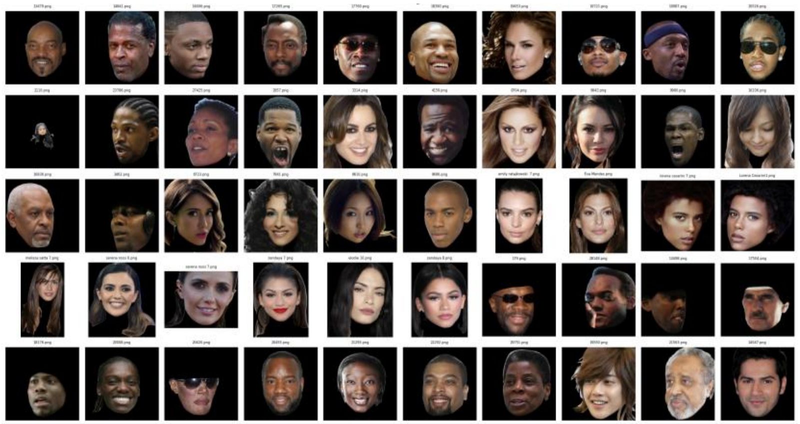
- The **Elbow Method** visualizes the within-cluster sum of squares (WCSS) against K, and identifies the point wher e increasing the number of clusters yields diminishing i mprovements.
- The **Silhouette Score** measures how similar each sampl e is to its own cluster compared to other clusters, with hi gher scores indicating clearer and more well-defined clu sters.

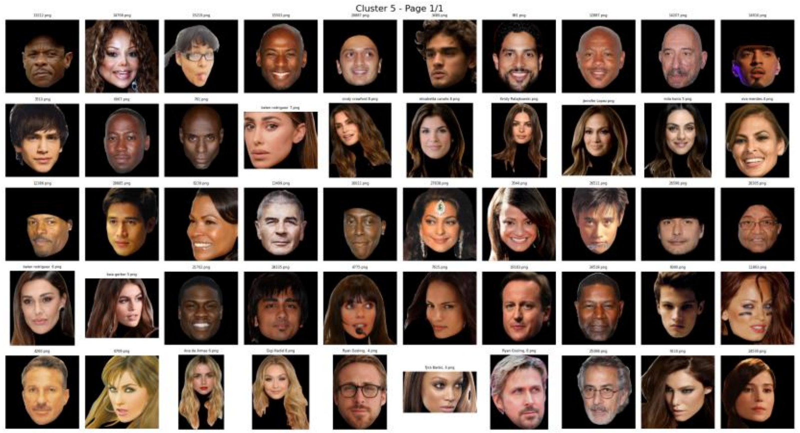


Clustering Result Visualization using Principal Component Analysis(PCA)

Visualization of representative images by cluster







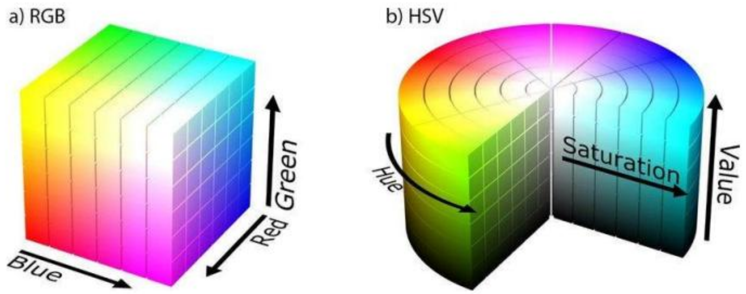
Cluster Labeling and Initial Palette Design

- the average HSV, Lab, and RGB values for each group were comprehensively analyzed names were assigned that matched the visual impression.

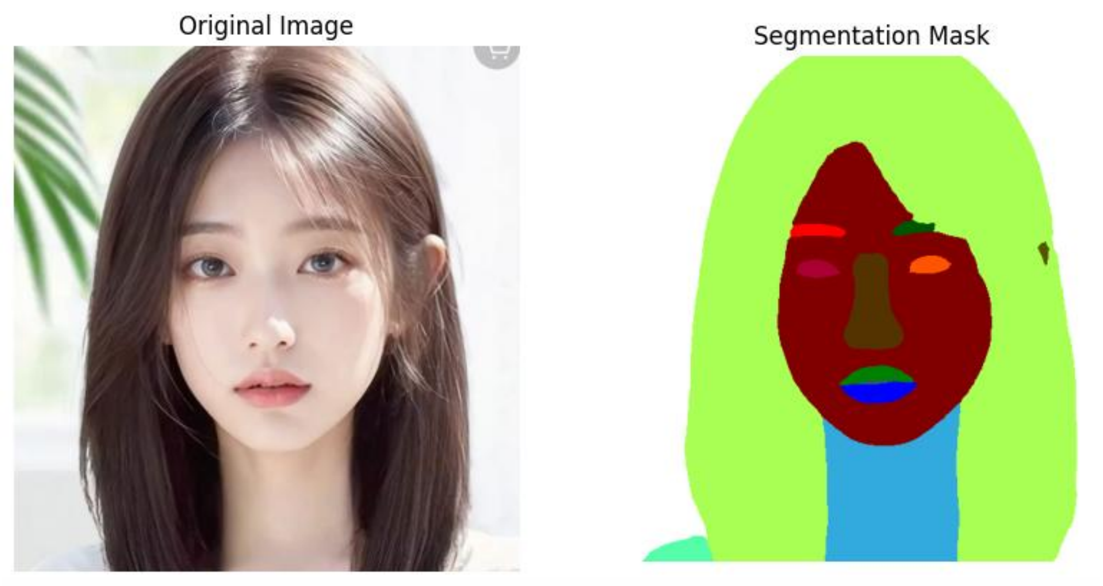
Cluster	Visual Name	Summary of Key Features
0	**Golden**	Highest average hue and saturation. Clearly warm tones.
1	**Warm Beige**	Overall, warm, with a slight olive undertone.
3	**Muted Clay**	Lowest saturation. Calm, muted colors.
4	**Warm Apricot**	Clear orange tones throughout. Stable, warm image.
5	**Peachy Pink**	Red-pink variability exists. Somewhat lovely, vibrant tones.
6	**Honey Buff**	It is a golden beige series that is warm and sweet like honey.
7	**Beige Rose**	Similar to cluster No.6, but slightly softer.

Cluster Labeling and Initial Palette Design

- After adjusting Hue and Value (brightness) to create a palette that harmonizes with the cluster, appropriate colors were manually selected to complete the final palette.



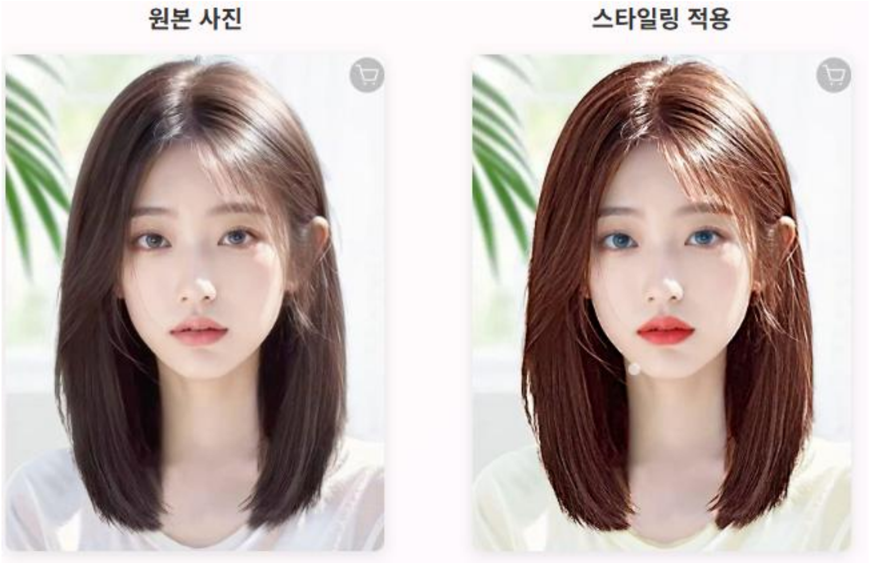
Cluster Labeling and Initial Palette Design



Color Overlay

- **Model**: BiSeNet (pretrained on CelebAMask-HQ, 19 classes)
- **Baseline**: Alpha blending (simple overlay)
- **Limitations**: Flat/unrealistic results in dark areas (e.g., black hair)
- **Improvements**:
    - Soft Light blending → preserves texture, shading, highlights
    - Sharpening → enhances detail

Soft Light Blending



$$\text{Result} = \begin{cases} 2 \cdot B \cdot S + B^2 \cdot (1 - 2S), & \text{if } S < 0.5 \\ 2 \cdot B \cdot (1 - S) + \sqrt{B} \cdot (2S - 1), & \text{if } S \geq 0.5 \end{cases}$$

•B: base pixel brightness

•S: blend pixel brightness

•Preserves shading + highlights → more natural results than alpha blending

Web Design and Implementation

- Backend: Flask (Python)
- Frontend: HTML, CSS, JavaScript
- ML Integration: .joblib models
- Firebase: User login & personalization

Research Results & Evaluation

- **New Skin-Tone Based Personal Color System**
- **7 Clusters Identified** through K-Means & PCA
- **Improved Visualization** with soft light blending
- **Web Prototype Implemented** (Flask + Firebase)
- **Limitations & Future Work** (lighting robustness, wider dataset)

Thank you for listening!

Team Underdog – AI-Powered Personal Beauty Advisor
2025 Graduation Project