MAKE ME UP

AI를 활용한 생활/뷰티 추천 서비스

Team

최민*

프로젝트 총괄
퍼스널 컬러 분석
데이터 가공 및 분류
머신러닝 모델 학습

김효*

퍼스널 컬러 분석
데이터 수집
딥러닝 모델 학습
자료 정리

조혁*

두피 데이터 분석
데이터 수집
데이터 가공 및 분류
딥러닝 모델 학습

이서*

얼굴형 데이터 분석
데이터 수집
데이터 가공 및 분류
딥러닝 모델 학습

이재*

프론트엔드
데이터 가공 및 분류

유광*

백엔드
데이터베이스 연동
Docker 작업

Contents

01

프로젝트 개요

프로젝트 배경

자신에게 맞는 스타일링과 건강을 위한 제품 및 서비스를 찾기 위해 많은 시간과 비용을 투자해야함

두피 건강

두피 질환 진단을 위해 병원 방문을 꺼려하거나 비용이 걱정 되는 사람이 많음

⇌

헤어스타일

자신에게 어울리는 헤어스타일을 찾기 위해 많은 시행착오가 필요

⇌

퍼스널컬러

퍼스널 컬러 진단을 위한 비용과 시간이 많이 필요

프로젝트 **목적**

“

AI를 활용한

종합 뷰티 진단 서비스”

+

피부톤, 얼굴형, 두피 상태 AI 분석
분석 결과와 스타일링 팁 제공
관련 제품 추천으로 사용자들에게 높은 만족도 제공

프로젝트 기대효과

손쉬운 개인 맞춤형 진단 서비스로 고객의 시간과 비용을 절약

두피 건강

두피질환에 대해 미리 파악하고 상담과 진단에 드는 비용을 절약 가능

헤어스타일

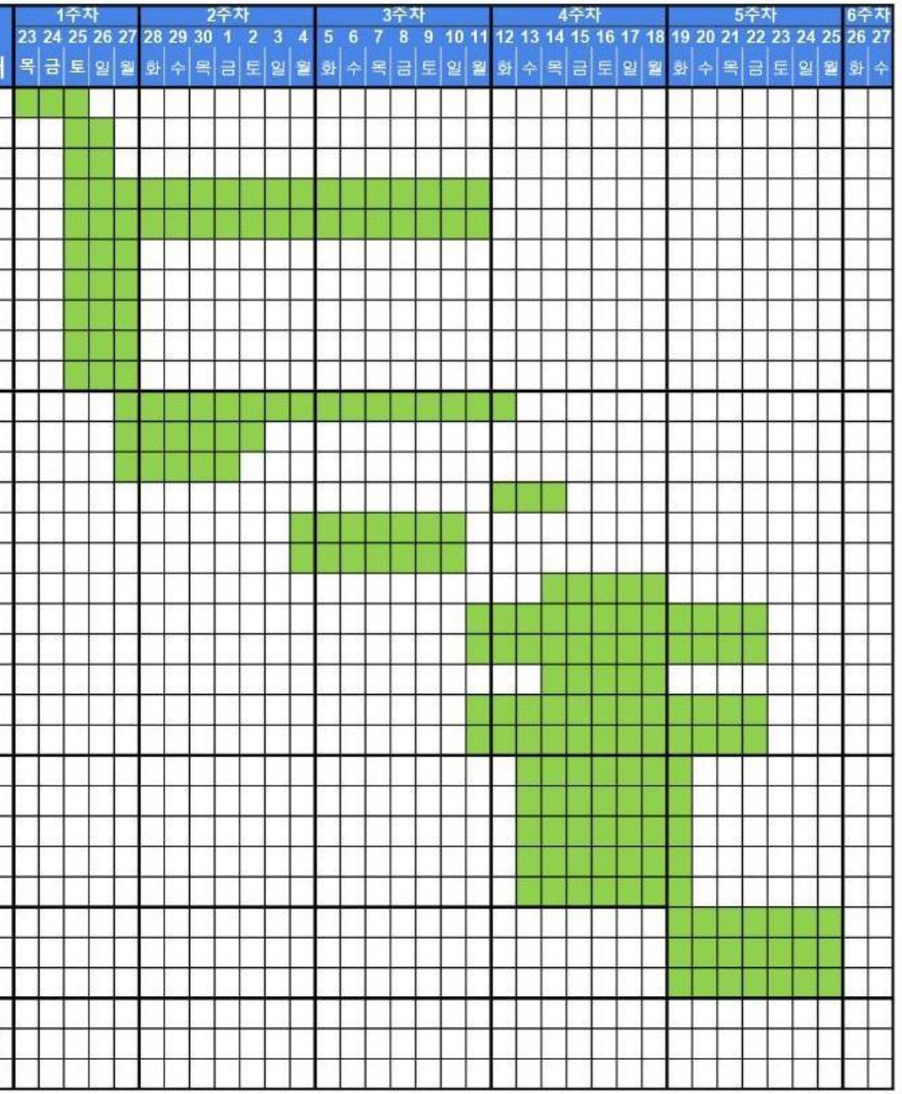
자신에게 맞는 헤어스타일을 찾는 시행착오를 덜어 줄 수 있음

퍼스널컬러

오프라인 진단에 비해 시간과 비용이 크게 절약됨

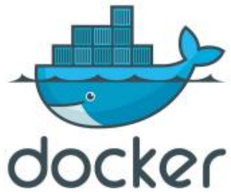
WBS 프로젝트 일정

구분	역할분류	작업	기능상세	진행상태
기획 및 설계	공통	기획서작성		완료
	공통	WBS 작성		완료
	공통	시스템구성도작성		완료
	공통	ERD 작성		완료
	공통	Sequence Diagram 작성		완료
	FE	UI/UX 디자인		완료
	FE	기술 스택 선정		완료
	BE	기술 스택 선정		완료
	FE	프로젝트 디렉터리 구조 설정		완료
	BE	프로젝트 디렉터리 구조 설정		완료
분석	데이터분석	데이터수집	퍼스널 컬러 색상 데이터 수집	완료
	데이터분석		두피 진단 이미지 수집	완료
	데이터분석		얼굴형 이미지 수집	완료
	데이터분석	데이터 전처리	퍼스널 컬러 색상 데이터 전처리	완료
	데이터분석		두피 진단 이미지 전처리	완료
	데이터분석		얼굴형 이미지 전처리	완료
	데이터분석	데이터 분류 및 학습	퍼스널 컬러 색상 데이터 학습	완료
	데이터분석		두피 진단 이미지 학습	완료
	데이터분석		얼굴형 이미지 학습	완료
	데이터분석	데이터 모델링	퍼스널 컬러 분석 모델링	완료
	데이터분석		두피 진단 모델링	완료
	데이터분석		얼굴형 분석 모델링	완료
페이지 구현	FE	메인페이지 구현	서비스 메인페이지 구현	완료
	FE	서비스 페이지 구현	퍼스널 컬러 서비스 페이지 구현	완료
	FE		두피 진단 서비스 페이지 구현	완료
	FE		얼굴형 분석 서비스 페이지 구현	완료
	BE	Logic 구현		완료
검수 및 배포	공통	기능 테스트		완료
	공통	호환성 테스트		완료
	공통	배포		완료
유지보수				작업대기
				작업대기
				작업대기



개발 도구

Back-End



Front-End

Analysis Tools

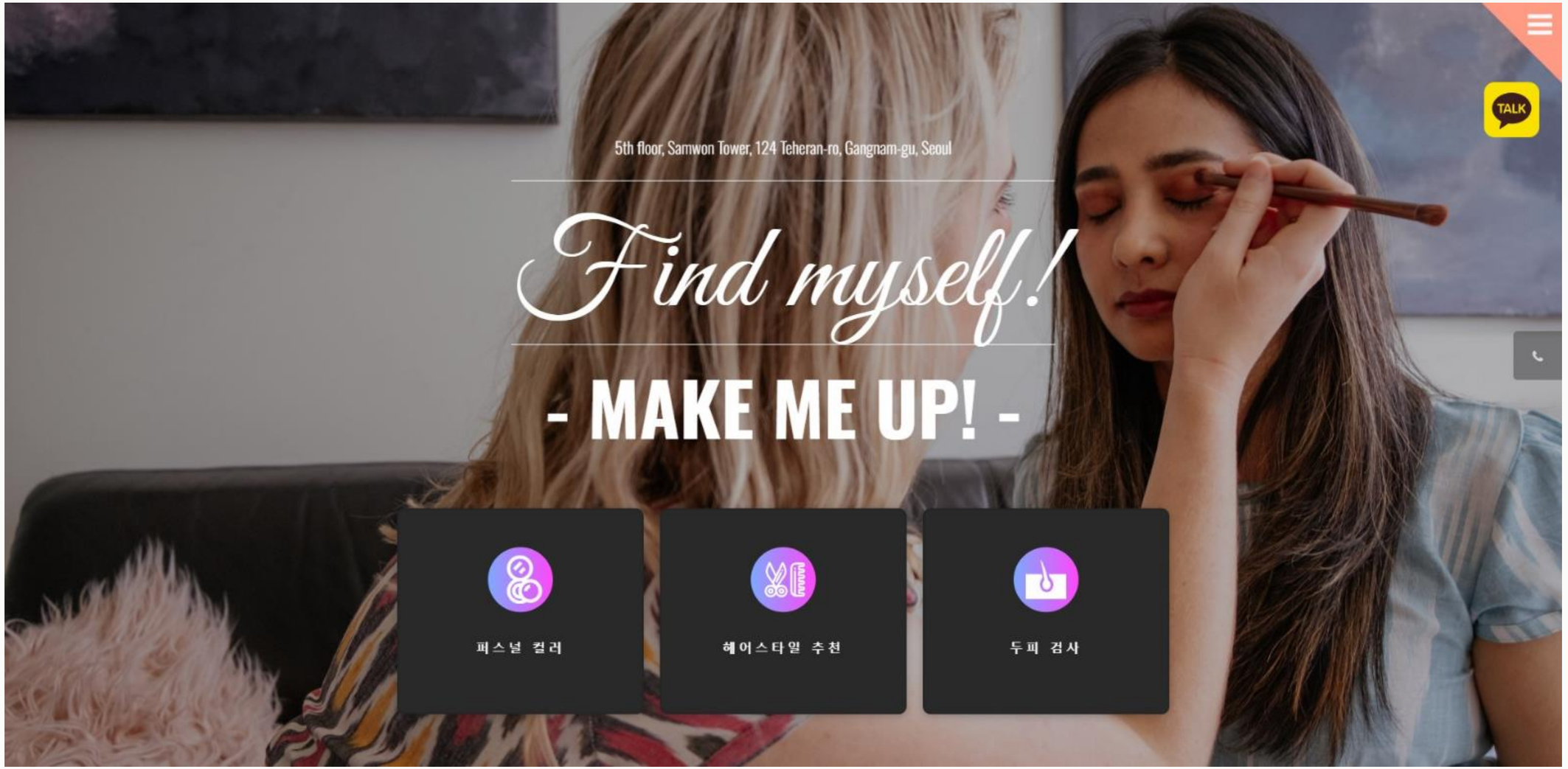
Manage Tools

DB

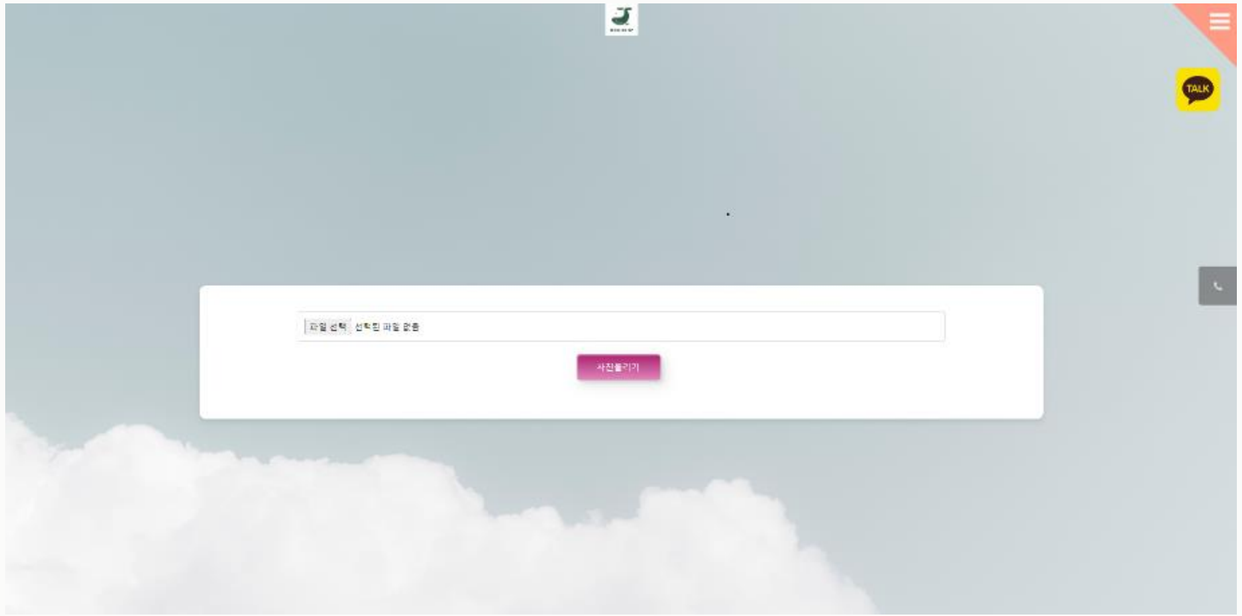
02

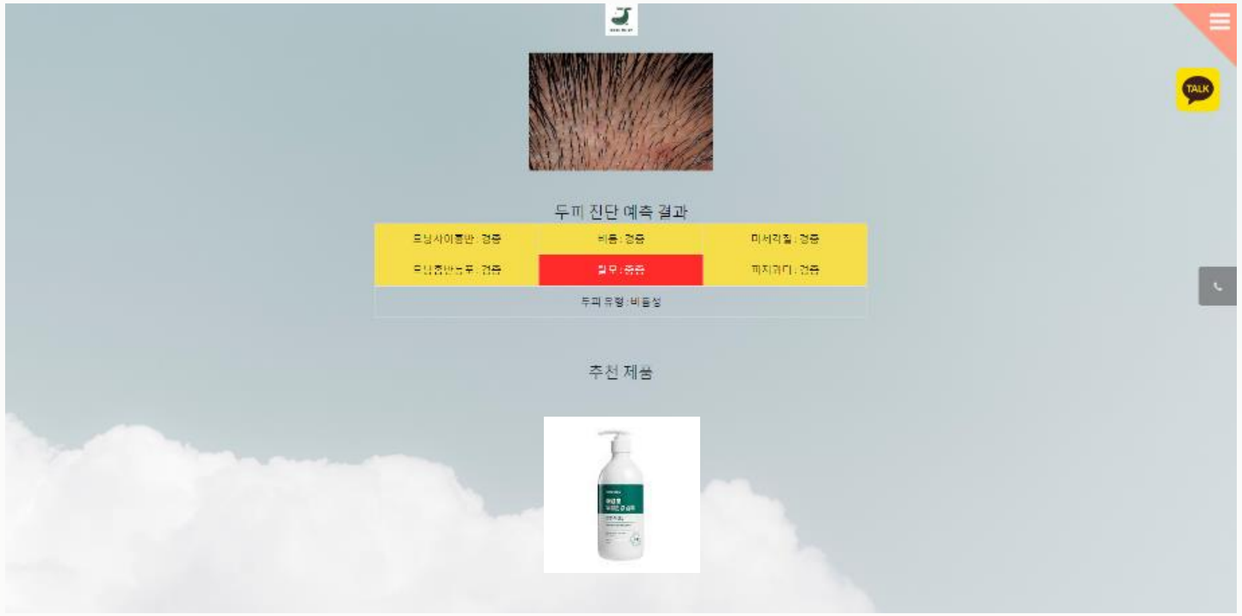
서비스 기획

서비스 구성


TALK
5th floor, Samwon Tower, 124 Teheran-ro, Gangnam-gu, Seoul
Find myself!
- MAKE ME UP! -
퍼스널 컬러
헤어스타일 추천
두피 검사

서비스 구성 두피 유형 찾기

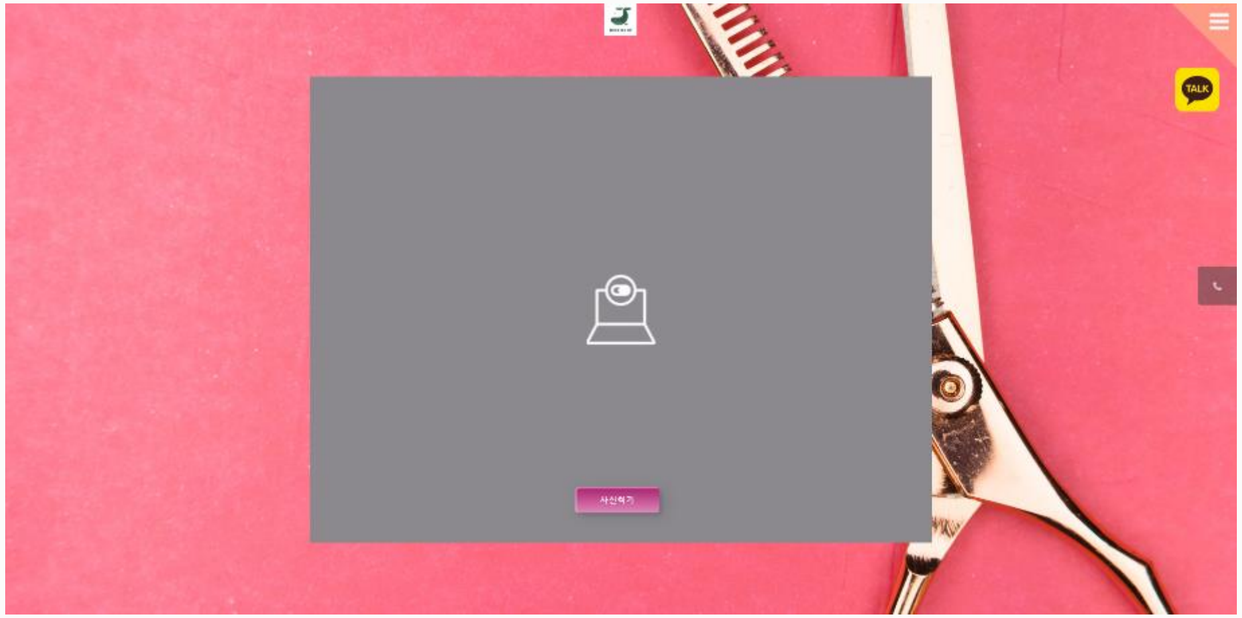




두피 이미지 분류 서비스로 본인 두피 유형 찾기

두피유형 맞춤 건강정보 제공 및 두피 케어 제품 추천

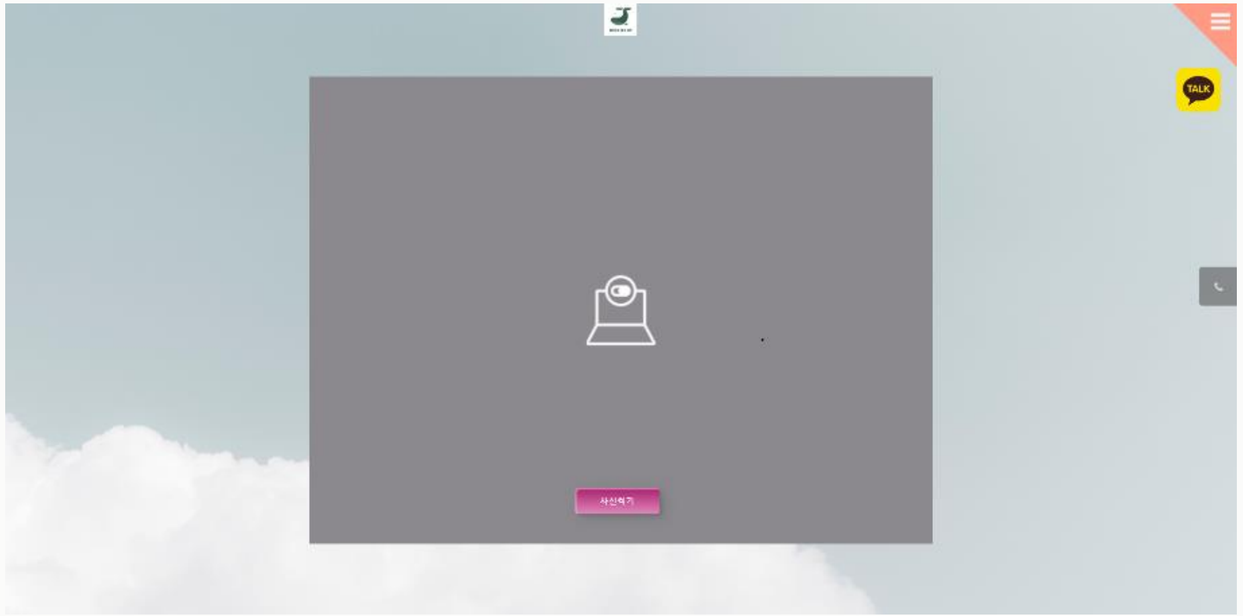
서비스 구성 헤어스타일 추천



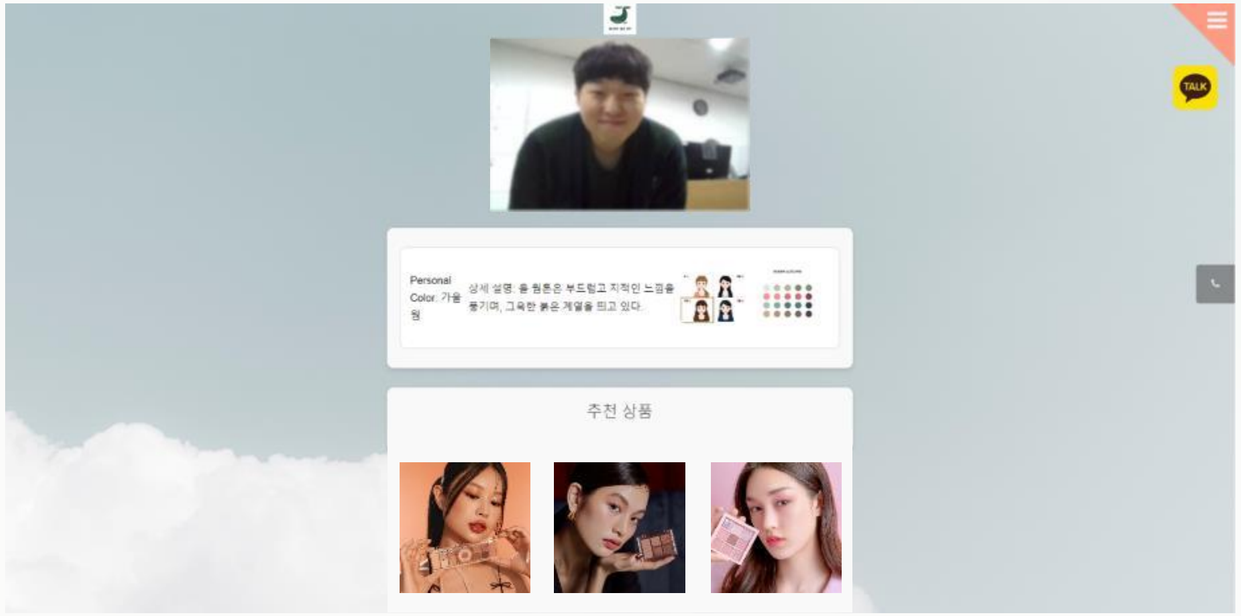
카메라 인식을 통해 사용자의 얼굴형을 분류

얼굴형 분석을 통해 각 사용자에게 어울리는 헤어스타일 추천

서비스 구성 퍼스널 컬러 진단



사용자의 피부톤을 인식하여 퍼스널컬러 찾기



피부톤 분석 결과 제공, 메이크업 및 제품 추천

03

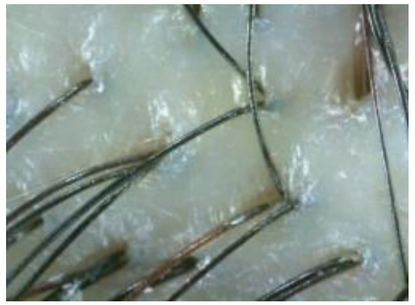
데이터 수집 및 분석

데이터 수집 - 두피 데이터

데이터 구성

1. 두피 이미지
2. 어노테이션 데이터 (두피 증상 데이터)
3. 메타 데이터 (설문조사 데이터)

두피 이미지 데이터



출처: AI 허브 (aihub.or.kr) 유형별 두피 이미지

두피 증상 6가지

a. 미세각질　　b. 피지과다

c. 모낭사이홍반　　d. 모낭홍반/농포

e. 비듬　　f. 탈모

두피 유형 9가지

A. 양호　　B. 건성　　C. 지성

D. 민감성　　E. 지루성　　F. 염증성

G. 비듬성　　H. 탈모성　　I. 복합성

데이터 수집 - 두피 데이터 활용

두피 증상 6가지와 구축·활용 가이드라인을 기준으로 두피유형 9가지 분류

구분	미세각질	피지과다	모낭사이홍반	모낭홍반/농포	비듬	탈모
양호	-	-	-	-	-	-
건성	+	-	-	-	-	-
지성	-	+	-	-	-	-
민감성	+ -	-	+	-	-	-
지루성	+ -	+	+	-	+ -	-
염증성	+ -	+ -	-	+	+ -	-
비듬성	+ -	+ -	-	-	+	-
탈모성	-	-	-	-	-	+
복합성	위의 8가지 두피유형에 해당하지 않는 경우					

(+ : 해당 증상이 있음 / - : 해당 증상이 없음 / + - : 해당 증상이 있거나 없을 수 있음)

데이터 전처리 - 두피 데이터

클래스 불균형 처리

1. 데이터 증강(Data Augmentation):

- 이미지 회전, 이동, 확대/축소, 뒤집기 등

2. 데이터 언더샘플링(Data Undersampling):

- 데이터를 줄여서 클래스 간 데이터 양을 맞추는 방법

Train data	Test data
332 ~ 29960	95 ~ 8560
1000	200
3000	600

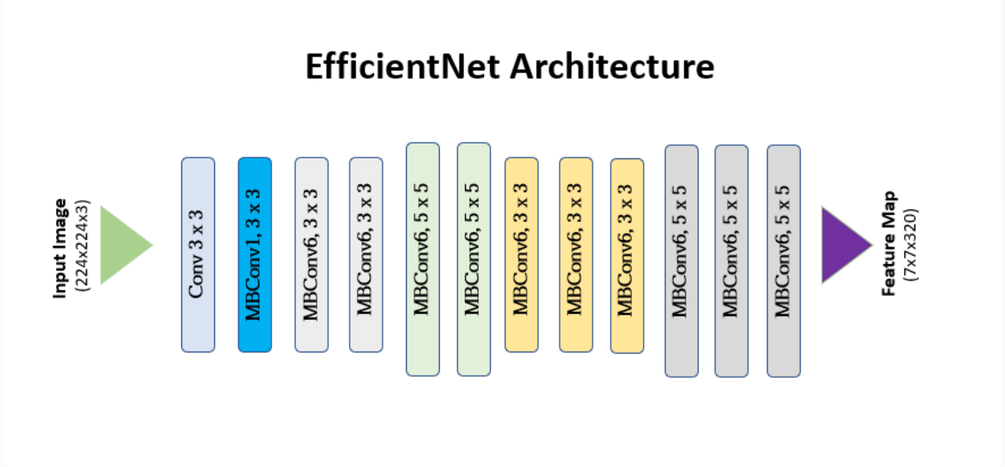
단위: 개

데이터 분석 - 전이학습 모델

EfficientNet

- 더 적은 파라미터들로 더 높은 정확도 도출 가능

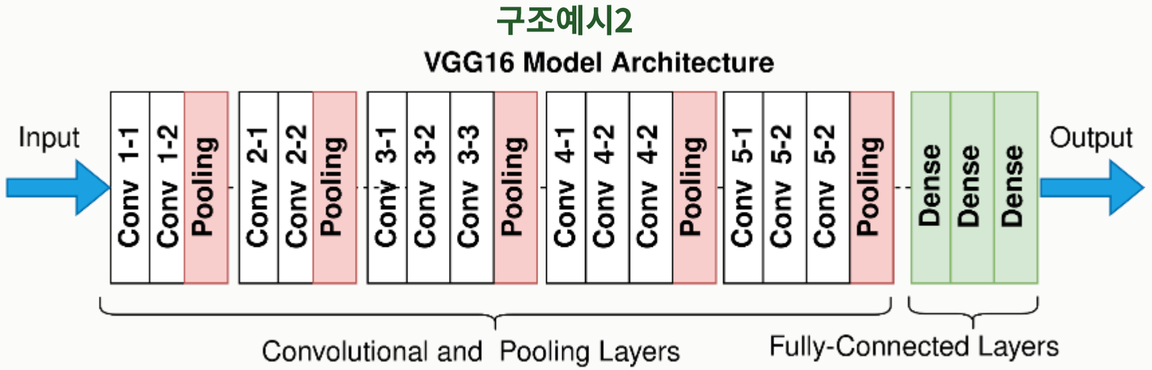
구조예시1



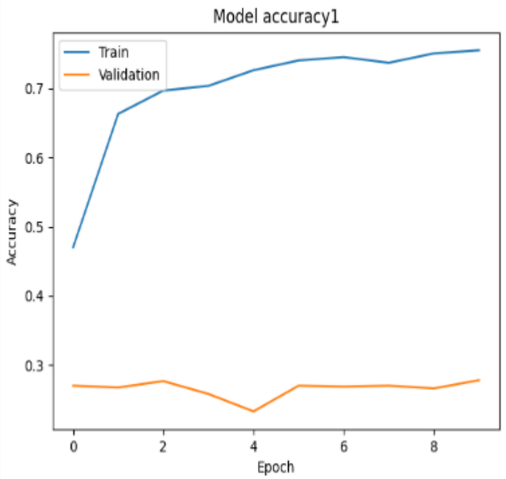
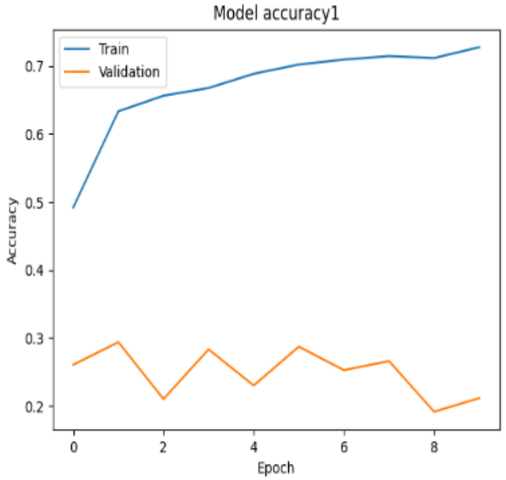
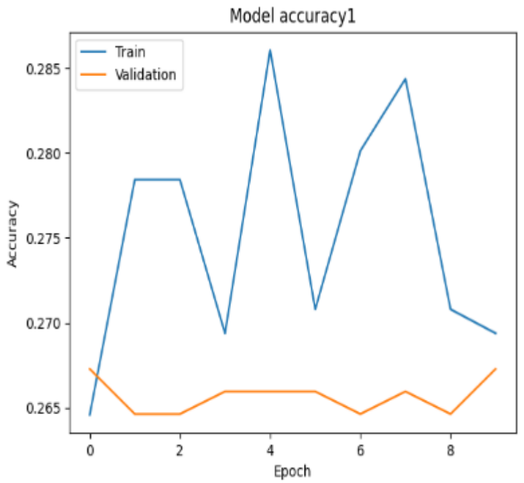
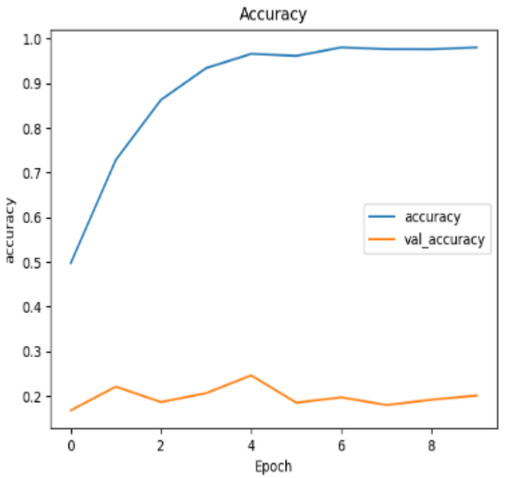
VGG16

- (ILSVRC) 2014에서 우수한 성과를 보여 기준 모델로 인정

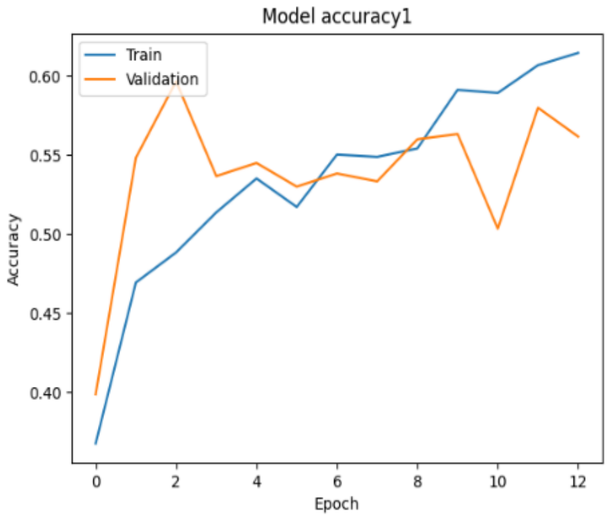
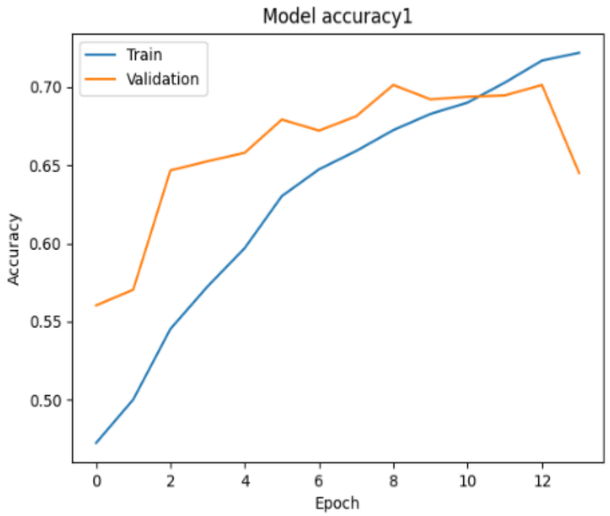
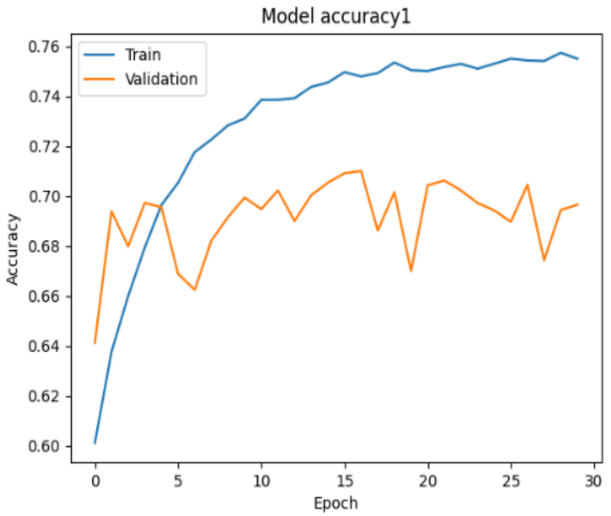
구조예시2



데이터 분석 - 모델 학습 진행 과정(EfficientNet)

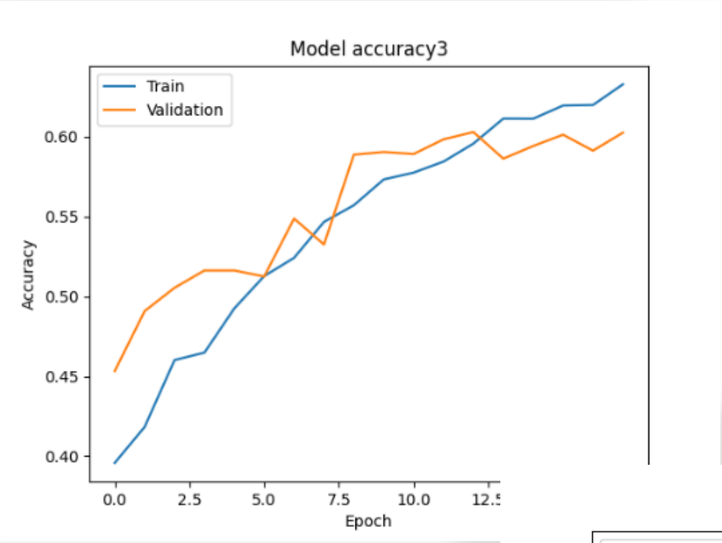
1차(27.79%)	2차(28.72%)	3차(26.73%)	4차(24.07%)
	Convolutional 레이어 , Dropout 레이어 추가	learning_rate 값 변경	augmentation 미적용
			

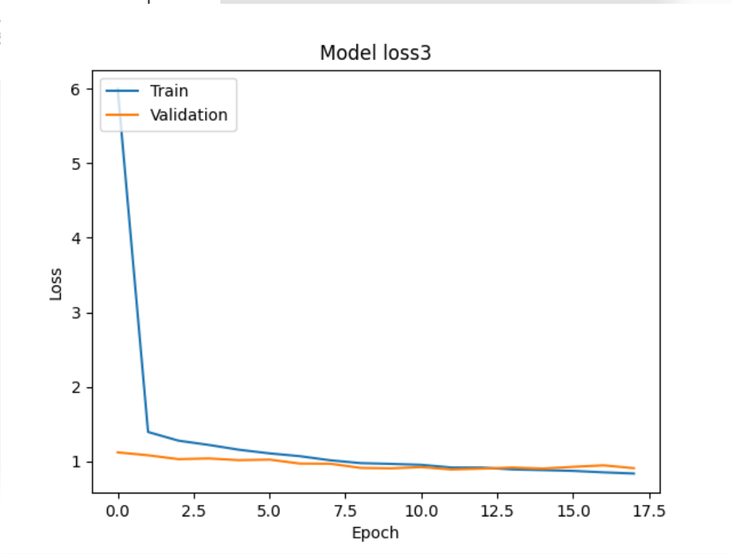
데이터 분석 - 모델 학습 진행 과정(VGG16)

5차(58.04%)	6차(70.13%)	7차(70.62%)
Train 1000, Test 200	Train 3000, Test 600	원본데이터
		

데이터 분석 - 모델 학습 결과(VGG16)

Model	Accuracy	Validation Accuracy
모낭사이홍반	75.16 %	70.62 %
모낭홍반농포	71.15 %	66.92 %
미세각질	63.27 %	60.25 %
비듬	67.55 %	66.21 %
탈모	77.93 %	75.04 %
피지과다	62.63 %	58.00 %
평균	69.55 %	66.17 %

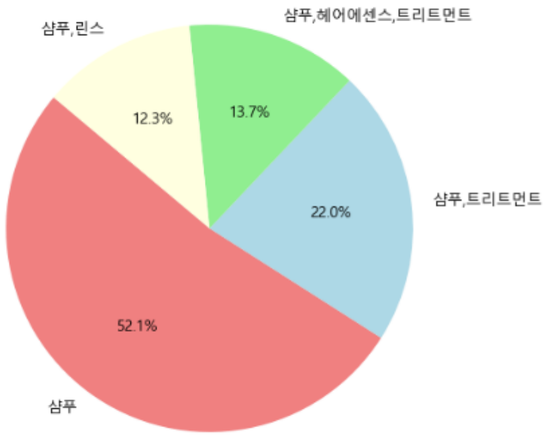




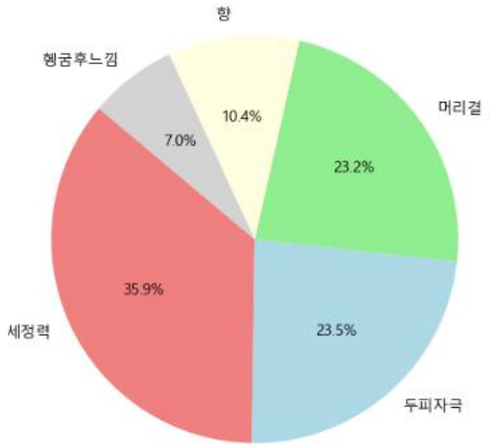
데이터 분석 - 추천 서비스

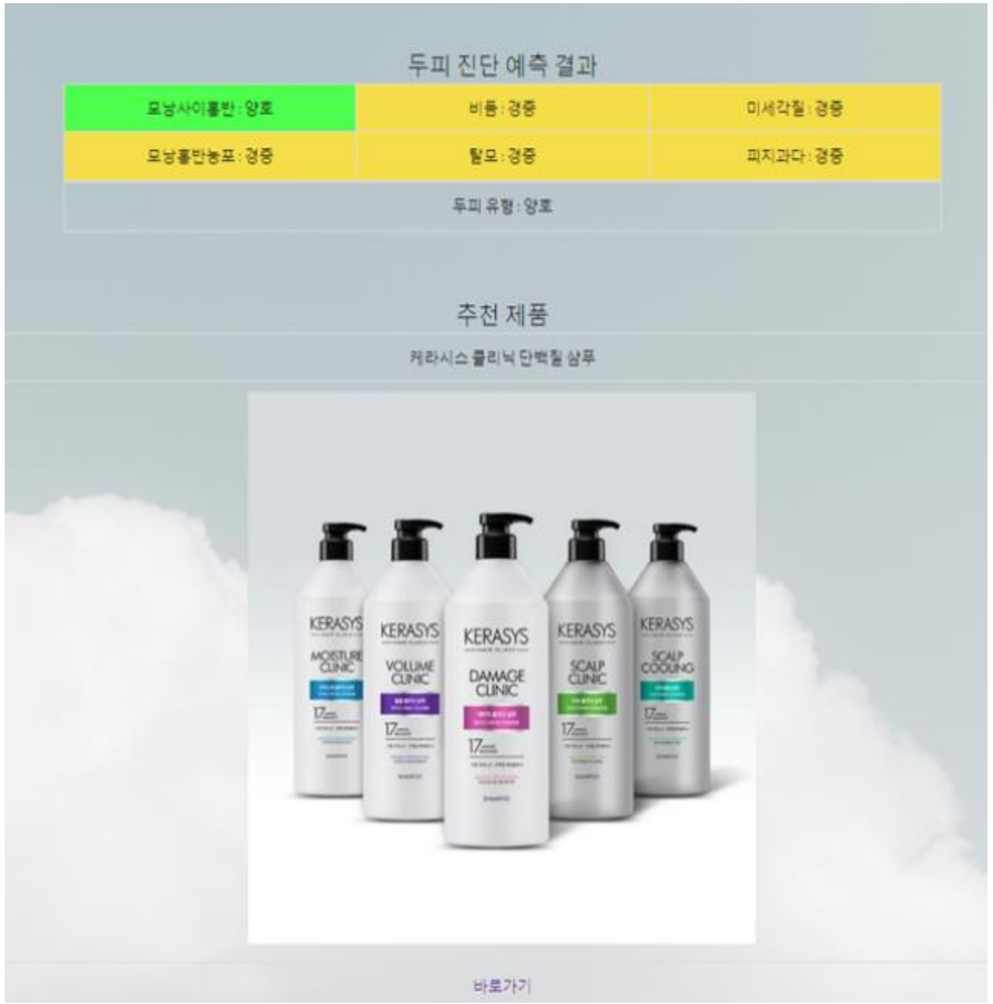
약10만명 설문조사(메타 데이터)를 통해 제품 추천에 활용

현재 사용하고 있는 두피모발용 제품



샴푸 구매시 고려하는 부분





데이터 수집 - 얼굴형 데이터

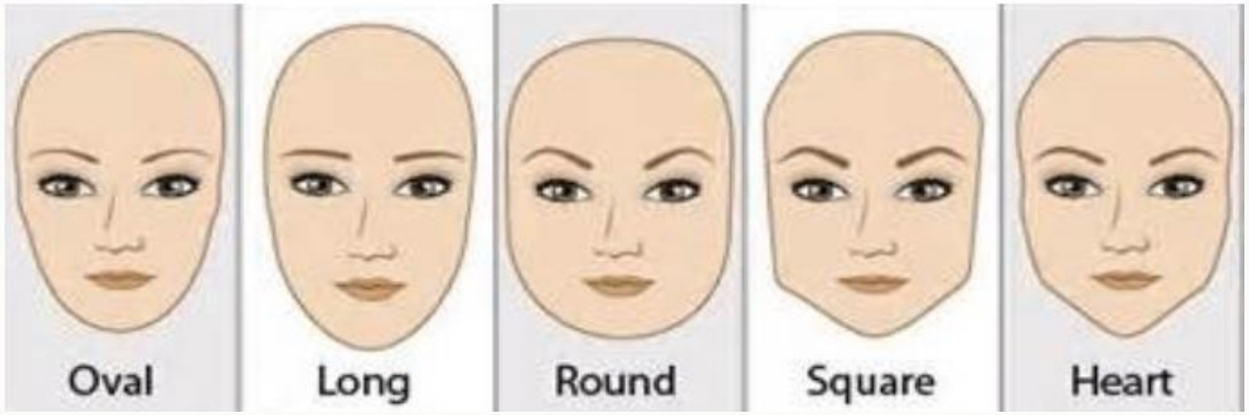
데이터 구성

얼굴 모양에 따라 분류된 유명인의 총 5000개 이미지로 구성

출처: kaggle의 Face Shape Dataset

얼굴형 분류 기준

얼굴형 분류 데이터셋 : 하트형, 계란형, 둥근형, 사각형, 긴얼굴형



출처: Google Image Search

데이터 전처리 - 얼굴형 데이터

전처리 1 – Face-Recognition

face-recognition 라이브러리 사용

얼굴 영역 crop 처리

전처리 2 – Data Augmentation

```
train_datagen = ImageDataGenerator(
    preprocess_input
    rotation_range=30,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode = 'nearest'
)
```

preprocess_input()
- VGG16 모델에 입력하기 위해 이미지를 전처리 하는 함수

<과정>
1. 이미지를 RGB 채널 순서로 바꾼다.
2. 각 채널마다 평균값을 뺀다.
3. 이미지를 0과 1 사이의 값으로 스케일링한다.

데이터 분석 - 학습에 사용된 모델 비교

VGG16

Layer (type)	Output Shape	Param #
ImageInput (InputLayer)	[(None, 224, 224, 3)]	0
Conv1_1 (Conv2D)	(None, 224, 224, 64)	1792
Conv1_2 (Conv2D)	(None, 224, 224, 64)	36928
pool1 (MaxPooling2D)	(None, 112, 112, 64)	0
Conv2_1 (SeparableConv2D)	(None, 112, 112, 128)	8896
Conv2_2 (SeparableConv2D)	(None, 112, 112, 128)	17664
pool2 (MaxPooling2D)	(None, 56, 56, 128)	0
Conv3_1 (SeparableConv2D)	(None, 56, 56, 256)	34176
bn1 (BatchNormalization)	(None, 56, 56, 256)	1024
Conv3_2 (SeparableConv2D)	(None, 56, 56, 256)	68096
bn2 (BatchNormalization)	(None, 56, 56, 256)	1024
Conv3_3 (SeparableConv2D)	(None, 56, 56, 256)	68096
pool3 (MaxPooling2D)	(None, 28, 28, 256)	0
Conv4_1 (SeparableConv2D)	(None, 28, 28, 512)	133888
bn3 (BatchNormalization)	(None, 28, 28, 512)	2048
Conv4_2 (SeparableConv2D)	(None, 28, 28, 512)	267264
bn4 (BatchNormalization)	(None, 28, 28, 512)	2048
Conv4_3 (SeparableConv2D)	(None, 28, 28, 512)	267264
pool4 (MaxPooling2D)	(None, 14, 14, 512)	0
flatten (Flatten)	(None, 100352)	0
fc1 (Dense)	(None, 1024)	102761472
dropout1 (Dropout)	(None, 1024)	0
fc2 (Dense)	(None, 512)	524800
dropout2 (Dropout)	(None, 512)	0
fc3 (Dense)	(None, 5)	2565

Total params: 104199045 (397.49 MB)
Trainable params: 104195973 (397.48 MB)
Non-trainable params: 3072 (12.00 KB)

ResNet50V2

Layer (type)	Output Shape	Param #
resnet50v2 (Functional)	(None, 7, 7, 2048)	23564800
flatten_1 (Flatten)	(None, 100352)	0
dense_2 (Dense)	(None, 512)	51380736
dense_3 (Dense)	(None, 5)	2565

Total params: 74948101 (285.90 MB)
Trainable params: 51383301 (196.01 MB)
Non-trainable params: 23564800 (89.89 MB)

데이터 분석 - 모델 학습

VGG16 모델 학습
- crop처리와 정규화 처리를 train data만 실행

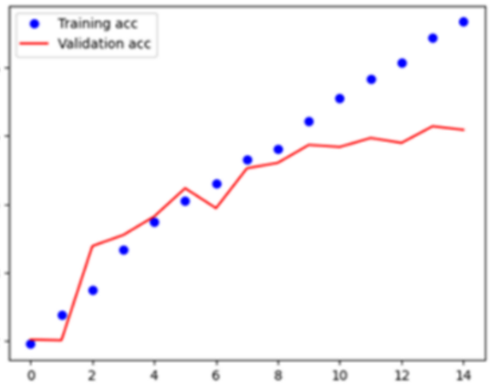
```
Epoch 20/100
359/359 - 64s - loss: 1.6026 - accuracy: 0.2237 - 64s/epoch - 177ms/step
Epoch 21/100
359/359 - 64s - loss: 1.6035 - accuracy: 0.2257 - 64s/epoch - 178ms/step
Epoch 22/100
359/359 - 64s - loss: 1.5991 - accuracy: 0.2277 - 64s/epoch - 179ms/step
Epoch 23/100
359/359 - 64s - loss: 1.6017 - accuracy: 0.2302 - 64s/epoch - 177ms/step
Epoch 24/100
359/359 - 64s - loss: 1.6014 - accuracy: 0.2248 - 64s/epoch - 177ms/step
Epoch 25/100
359/359 - 65s - loss: 1.6028 - accuracy: 0.2255 - 65s/epoch - 182ms/step
Epoch 26/100
```

→ 모델성능이 약 23%

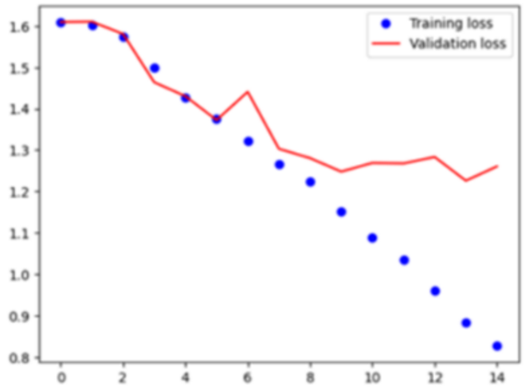
VGG16 모델 학습
- 전처리가 안된 원본 데이터로 다시 학습
  train, test data 모두 crop처리를 하지 않고 정규화만 실행

```
Epoch 11/30
250/250 [==============================] - 1029s 4s/step - loss: 1.0
Epoch 12/30
250/250 [==============================] - 1036s 4s/step - loss: 1.0
Epoch 13/30
250/250 [==============================] - 1025s 4s/step - loss: 0.9
Epoch 14/30
250/250 [==============================] - 1032s 4s/step - loss: 0.8
Epoch 15/30
250/250 [==============================] - 1025s 4s/step - loss: 0.8
```

→ 모델성능이 약 50%



훈련데이터 정확도와 검증데이터 정확도



훈련데이터 손실값과 검증데이터 손실값

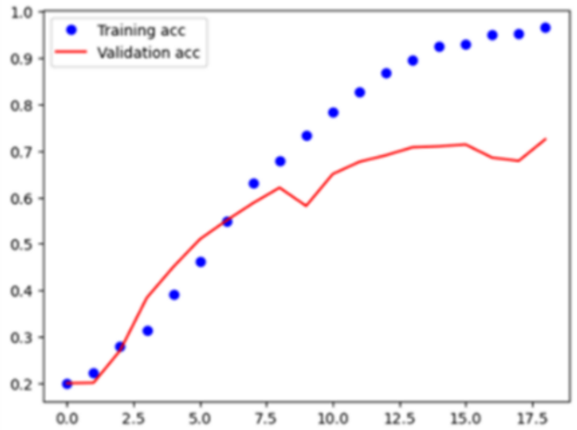
데이터 분석 - 모델 학습(VGG16모델 사용)

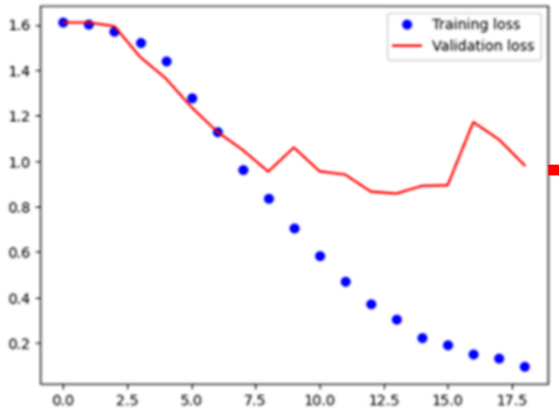
VGG16 모델 학습
- train, test 모두 crop처리와 정규화까지 진행

```
In [3]: from keras.models import load_model
        model = load_model('C:/Users/SEOHO/Desktop/wfacedata/shape_vgg16_process1.h5')

        model.evaluate(test_data)
```

```
63/63 [==============================] - 28s 426ms/step - loss: 0.9848 - accuracy:
0.7250
Out [3]: [0.9847874641418457, 0.7250000238418579]
```





→ 모델성능이 72.5%

데이터 분석 - 모델 학습

ResNet50V2모델 학습
- train, test 모두 crop처리와 정규화까지 진행

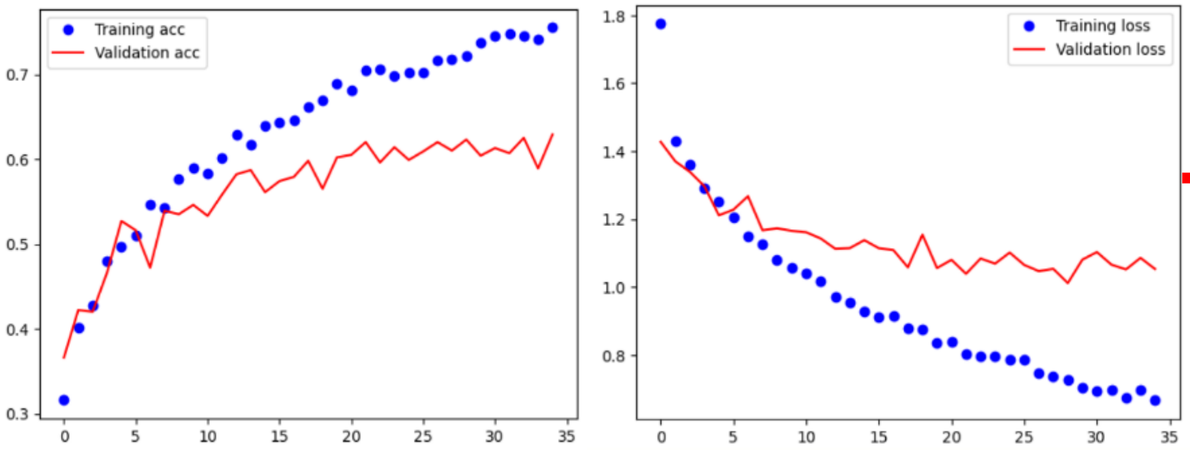
In [4]:
```
from keras.models import load_model
model = load_model('C:/Users/SEOHO/Desktop/wfacedata/model_checkpoint.h5')

model.evaluate(test_generator)
```

```
WARNING:tensorflow:From C:₩anaconda3₩Lib₩site-packages₩keras₩src₩utils₩tf_utils.py:492: The
ated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

50/50 [==============================] - 29s 553ms/step - loss: 1.0525 - accuracy: 0.6290
```

Out[4]: [1.0525226593017578, 0.6290000081062317]



→ 모델성능이 약 63%

데이터 분석 - VGG16 모델 평가

평가1

– 얼굴형별 200장의 이미지들에 대한 평가 결과

```
63/63 [==============================] - 32s 500ms/step

Round     254
Oblong    206
Heart     199
Oval      189
Square    152
Name: count, dtype: int64
```

평가2

– 사각형 얼굴형으로 유명한 여자 연예인을 83%의 확률로 얼굴형을 사각형으로 예측

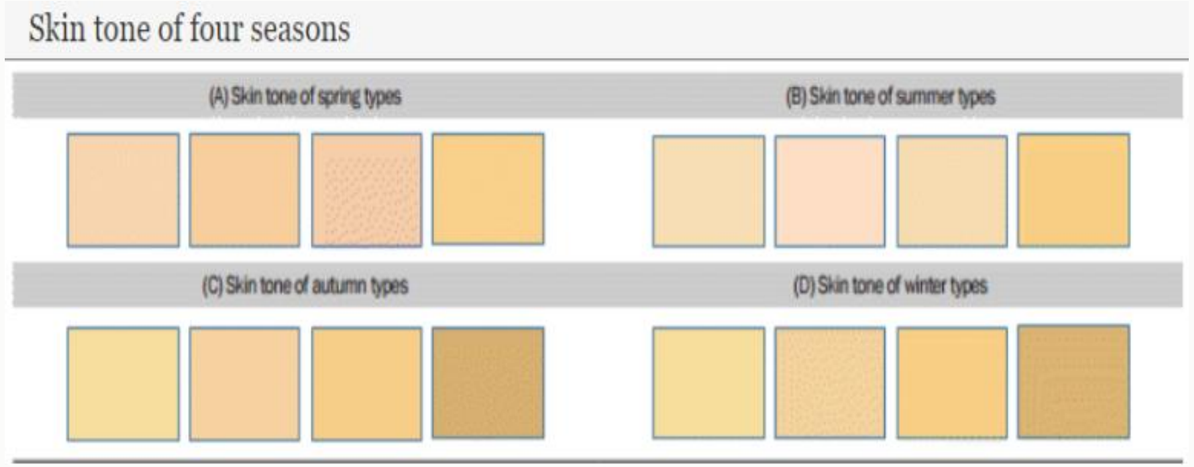

```
1/1 [==============================] - 0s 329ms/step
Predicted Class: 사각형
Prediction Percentages: [0. 0. 0. 16. 83]
```

데이터 분석 - 분류 결과 출력

결과 : 얼굴형에 맞는 헤어스타일을 남녀 모두에게 추천

데이터 수집 - 퍼스널 컬러 데이터

퍼스널 컬러별 피부톤



출처 :퍼스널 컬러 진단 가이드 요인간 상관관계와 뇌파분석(김민경, 2016)

퍼스널 컬러별 색상 팔레트

출처 :퍼스널 컬러 진단 가이드 요인간 상관관계와 뇌파분석(김민경, 2016)

봄 타입 (R, G, B)
(251, 211, 168), (255, 202, 149),
(253, 197, 161), (252, 204, 130)

가을 타입 (R, G, B)
(255, 221, 150), (247, 206, 152),
(249, 201, 128), (212 169, 101)

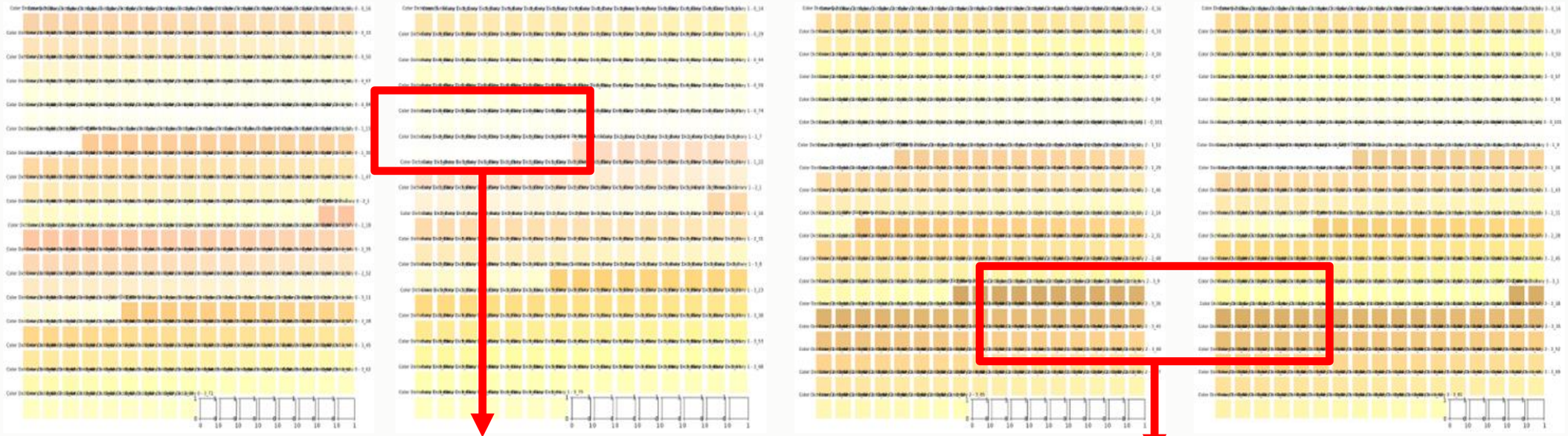
여름 타입 (R, G, B)
(253, 231, 174), (255, 219, 192),
(254, 217, 170), (254, 210, 122)

겨울 타입 (R, G, B)
(255, 220, 147), (242, 206, 148),
(247, 207, 121), (216, 173, 102)

데이터 전처리 - 퍼스널 컬러 데이터

① 4개의 RGB 값을 총 1111개의 데이터로 증강

봄 타입 | 여름 타입 | 가을 타입 | 겨울 타입



② 흰색에 가까운 값 제거

③ 중복값 제거 총 색상값 875개

데이터 분석 - 모델 설계 및 학습

옵션 1: 머신 러닝 분류 모델을 활용한 stacking ensemble 방식

Soft Voting

Hard Voting

CatBoostClassfier

옵션 2: 간단한 딥러닝 분류 모델 사용

DNN 모델

CNN 모델

데이터 분석 - 모델 설계 및 학습

옵션 1: 머신 러닝 분류 모델을 활용한 stacking ensemble 방식

1. Minmaxscaler 적용한 ensemble 모델 + PolynomialFeatures로 피처 수 확장 voting 방식 : soft

```
# MinMaxScaler를 사용하여 데이터 정규화
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
print('X_scaled : ', X_scaled)
```

```
# PolynomialFeatures를 사용하여 피처 확장 (예시로 degree=2를 선택)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X_scaled)
print('X_poly: ', X_poly)
print(X_poly.shape)
```

```
# 각 모델을 VotingClassifier에 추가 (소프트 보팅 설정)
ensemble_model_soft = VotingClassifier(estimators=models, voting='soft')
```

9개 모델 사용

SVM Model:
Accuracy: 0.5257142857142857
KNN11 Model:
Accuracy: 0.49142857142857144
KNN9 Model:
Accuracy: 0.52
KNN7 Model:
Accuracy: 0.5371428571428571
KNN5 Model:
Accuracy: 0.5771428571428572
LDA Model:
Accuracy: 0.5885714285714285
LR_L2 Model:
Accuracy: 0.5028571428571429
RF Model :
Accuracy : ?
GR Model :
Accuracy : ?

데이터 분석 - 모델 설계 및 학습

옵션 1: 머신 러닝 분류 모델을 활용한 stacking ensemble 방식

2. Minmaxscaler 적용한 ensemble 모델 + PolynomialFeatures로 피처 수 확장 voting 방식 : hard

```
# MinMaxScaler를 사용하여 데이터 정규화
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
print('X_scaled : ', X_scaled)
```

```
# PolynomialFeatures를 사용하여 피처 확장 (예시로 degree=2를 선택)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X_scaled)
print('X_poly: ', X_poly)
print(X_poly.shape)
```

```
# 각 모델을 VotingClassifier에 추가
ensemble_hard_model = VotingClassifier(estimators=models, voting='hard
```

9개 모델 사용

SVM Model:
Accuracy: 0.5257142857142857
KNN11 Model:
Accuracy: 0.49142857142857144
KNN9 Model:
Accuracy: 0.52
KNN7 Model:
Accuracy: 0.5371428571428571
KNN5 Model:
Accuracy: 0.5771428571428572
LDA Model:
Accuracy: 0.5885714285714285
LR_L2 Model:
Accuracy: 0.5028571428571429
RF Model :
Accuracy : ?
GR Model :
Accuracy : ?

데이터 분석 - 모델 설계 및 학습

옵션 1: 머신 러닝 분류 모델을 활용한 stacking ensemble 방식

3. CatBoostClassfier 모델

```
# MinMaxScaler를 사용하여 데이터 정규화
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
print('X_scaled : ', X_scaled)
```

```
# PolynomialFeatures를 사용하여 피처 확장 (예시로 degree=2를 선택)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X_scaled)
print('X_poly: ', X_poly)
print(X_poly.shape)
```

```
# CatBoost 모델 생성 및 훈련
catboost_model = CatBoostClassifier(iterations=500, depth=10, learning_rate=0.01, loss_function='MultiClass')
catboost_model.fit(X_train, y_train)
```

데이터 분석 - 모델 설계 및 학습

옵션 1: 머신 러닝 분류 모델을 활용한 stacking ensemble 방식

soft voting

Ensemble Model (Soft Voting) Accuracy: 0.697142857

Ensemble Model (Soft Voting) Classification Report:

	precision	recall	f1-score	support
0	0.69	0.65	0.67	54
1	0.55	0.59	0.57	39
2	0.73	0.79	0.76	56
3	0.91	0.77	0.83	26
accuracy			0.70	175
macro avg	0.72	0.70	0.71	175
weighted avg	0.70	0.70	0.70	175

가장 높은 정확도

hard voting

Ensemble Model Accuracy: 0.6342857142857142

Ensemble Model Classification Report:

	precision	recall	f1-score	support
0	0.56	0.65	0.60	54
1	0.54	0.51	0.53	39
2	0.67	0.66	0.67	56
3	0.90	0.73	0.81	26
accuracy			0.63	175
macro avg	0.67	0.64	0.65	175
weighted avg	0.64	0.63	0.64	175

CatBoostClassfier

CatBoost Model Accuracy: 0.6857142857142857

CatBoost Model Classification Report:

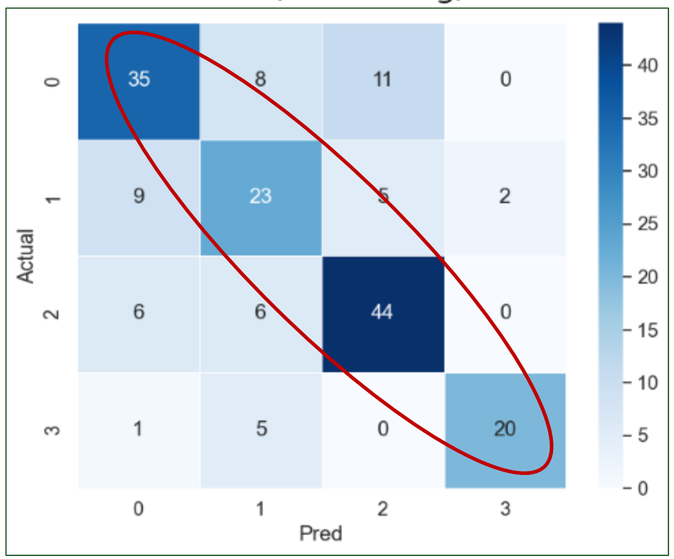
	precision	recall	f1-score	support
0	0.68	0.67	0.67	54
1	0.63	0.56	0.59	39
2	0.66	0.79	0.72	56
3	0.90	0.69	0.78	26
accuracy			0.69	175
macro avg	0.72	0.68	0.69	175
weighted avg	0.69	0.69	0.69	175

데이터 분석 - 모델 설계 및 학습

옵션 1: 머신 러닝 분류 모델을 활용한 stacking ensemble 방식

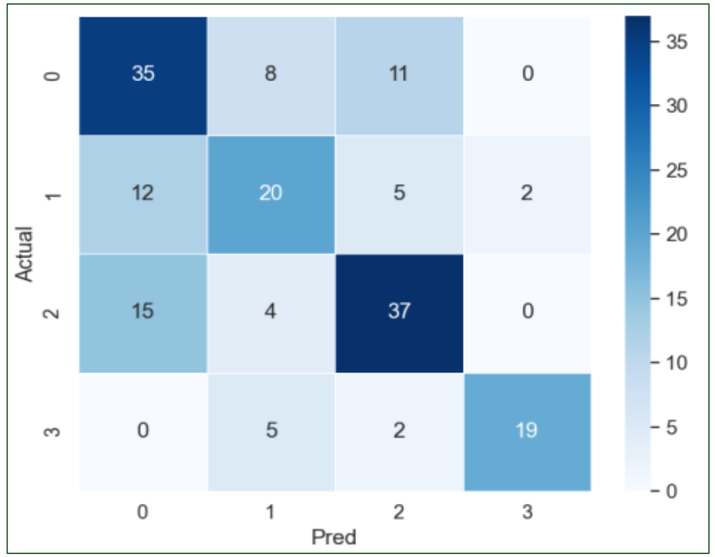
soft voting

Ensemble Model (Soft Voting) Confusion Matrix:



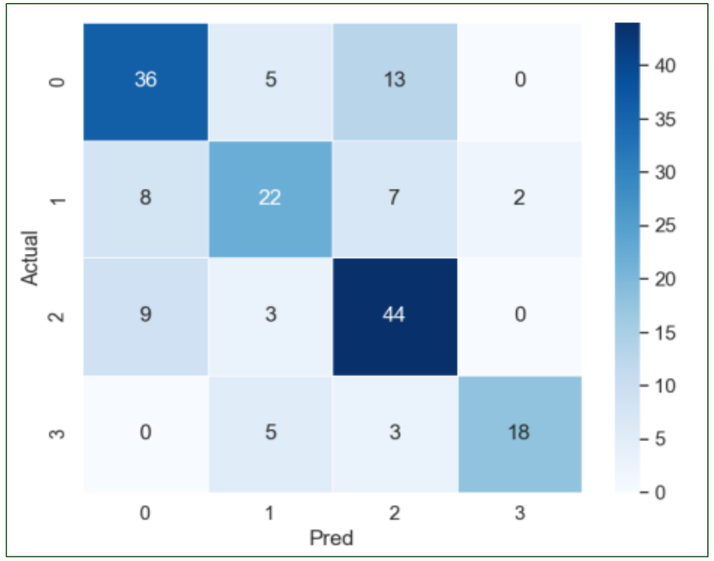
hard voting

Ensemble Model Confusion Matrix:



CatBoostClassfier

CatBoost Model Confusion Matrix:



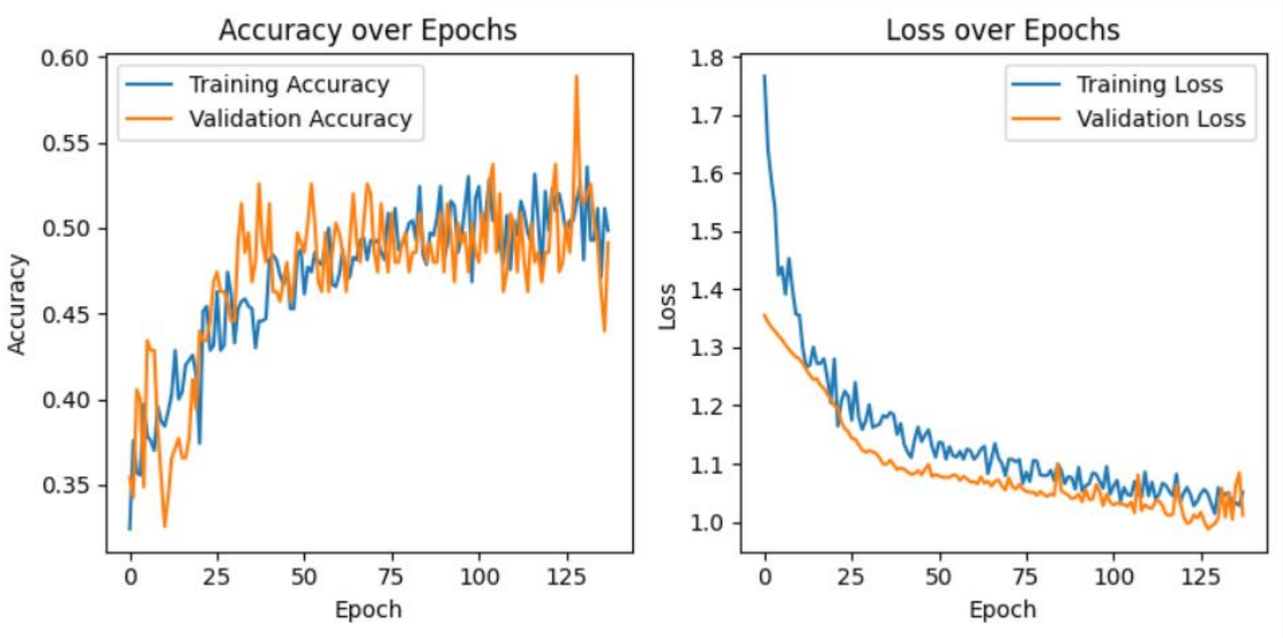
데이터 분석 - 모델 설계 및 학습

옵션 2: 딥러닝 분류 모델 사용

```
Epoch 121/200
22/22 [==============================] - 0s 8ms/step - loss: 1.0465 - accuracy: 0.4943 - val_loss: 0.9915 - val_accuracy: 0.4971
Epoch 122/200
22/22 [==============================] - 0s 8ms/step - loss: 1.0325 - accuracy: 0.5086 - val_loss: 1.0053 - val_accuracy: 0.4743
Epoch 123/200
22/22 [==============================] - 0s 8ms/step - loss: 1.0490 - accuracy: 0.4943 - val_loss: 1.0046 - val_accuracy: 0.5086
Epoch 124/200
19/22 [========================>.....] - ETA: 0s - loss: 1.0098 - accuracy: 0.5230Restoring model weights from the end of the best epoch: 114.
22/22 [==============================] - 0s 9ms/step - loss: 1.0184 - accuracy: 0.5257 - val_loss: 0.9959 - val_accuracy: 0.4971
Epoch 124: early stopping
6/6 [==============================] - 0s 4ms/step - loss: 0.9908 - accuracy: 0.5143
Test accuracy: 51.43%
```

DNN 모델

완전 연결 계층을 중심으로 한 Dense 신경망 사용
relu 대신 swish 함수 사용



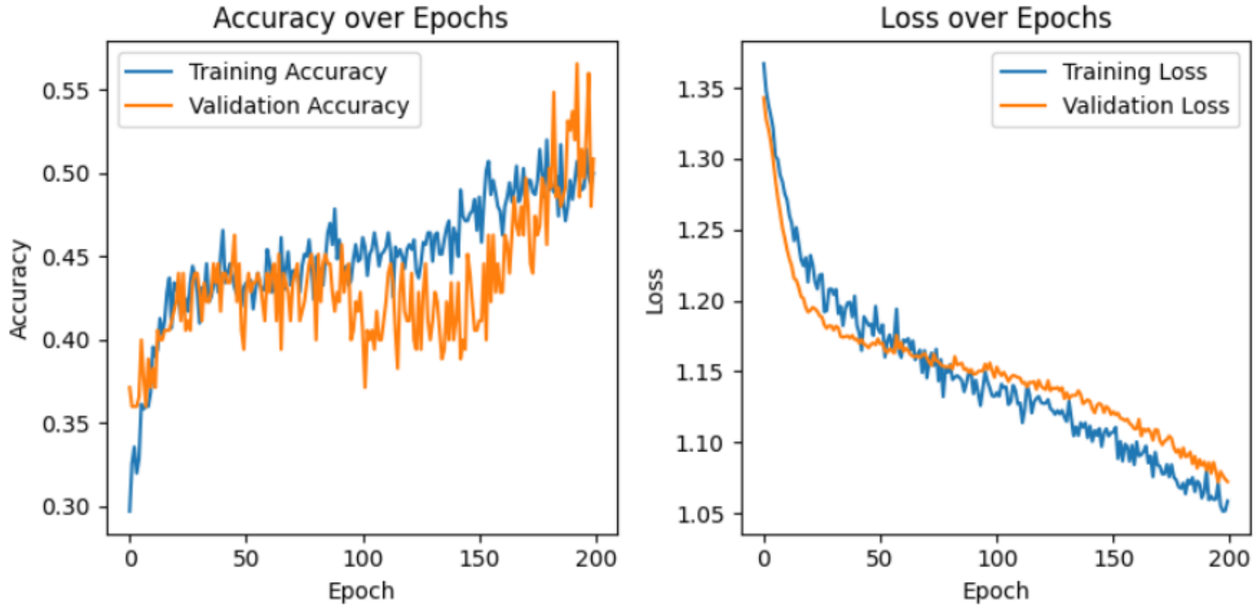
데이터 분석 - 모델 설계 및 학습

옵션 2: 딥러닝 분류 모델 사용

```
22/22 [==============================] - 0s 3ms/step - loss: 1.0555 - accuracy: 0.5143 - val_loss: 1.0795 - val_accuracy: 0.5257
Epoch 198/200
22/22 [==============================] - 0s 2ms/step - loss: 1.0515 - accuracy: 0.4971 - val_loss: 1.0766 - val_accuracy: 0.5600
Epoch 199/200
22/22 [==============================] - 0s 2ms/step - loss: 1.0518 - accuracy: 0.4929 - val_loss: 1.0741 - val_accuracy: 0.4800
Epoch 200/200
22/22 [==============================] - 0s 2ms/step - loss: 1.0586 - accuracy: 0.5000 - val_loss: 1.0724 - val_accuracy: 0.5086
6/6 [==============================] - 0s 2ms/step - loss: 1.0724 - accuracy: 0.5086
Test accuracy: 50.86%
```

CNN 모델

합성곱 신경망 사용
1D 합성곱 계층, 맥스 풀링 계층,
Flatten 및 Dense 계층을 포함



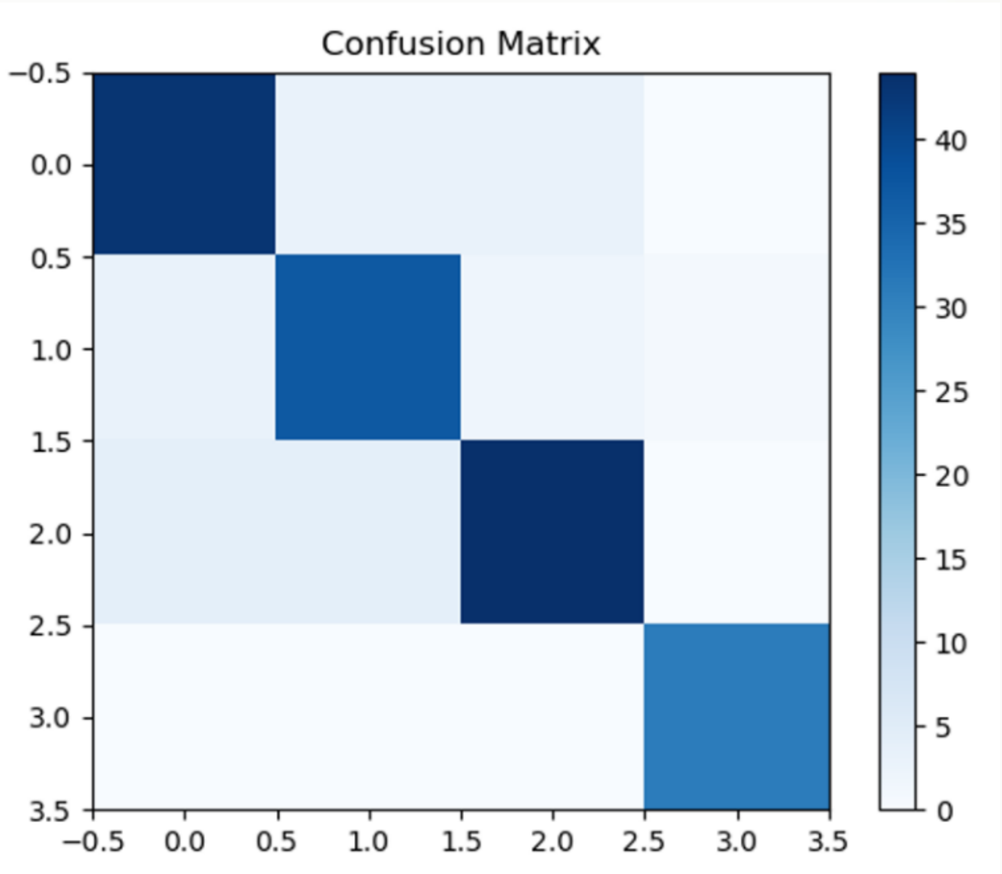
데이터 분석 - 성능 평가

앙상블 모델(soft voting) : 테스트 결과

	precision	recall	f1-score	support
0	0.86	0.88	0.87	49
1	0.84	0.86	0.85	43
2	0.90	0.85	0.87	52
3	0.97	1.00	0.98	31
accuracy			0.89	175
macro avg	0.89	0.90	0.89	175
weighted avg	0.89	0.89	0.89	175

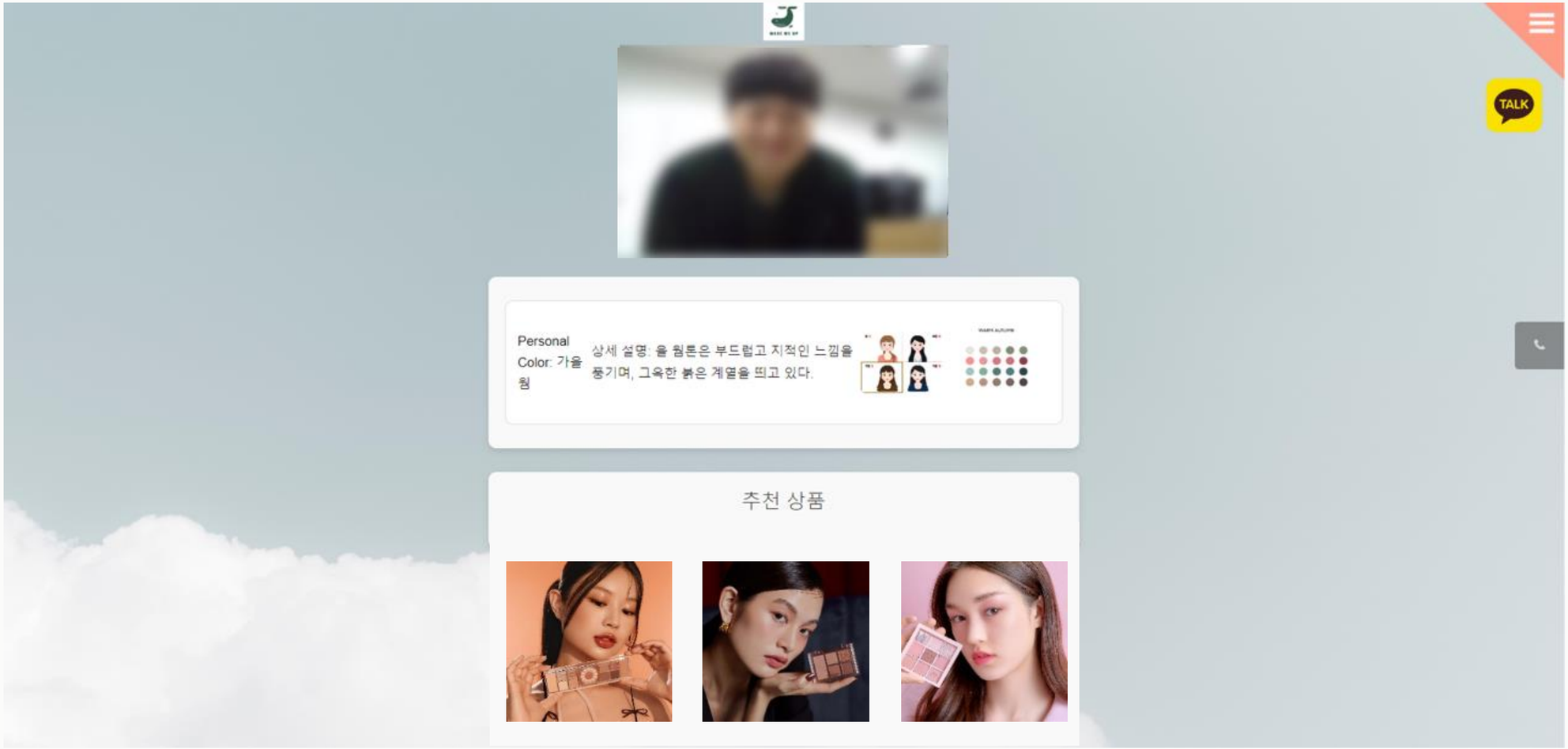
Accuracy: 88.57%

Confusion Matrix:
[[43 3 3 0]
[3 37 2 1]
[4 4 44 0]
[0 0 0 31]]



데이터 분석 - 분류 결과 출력

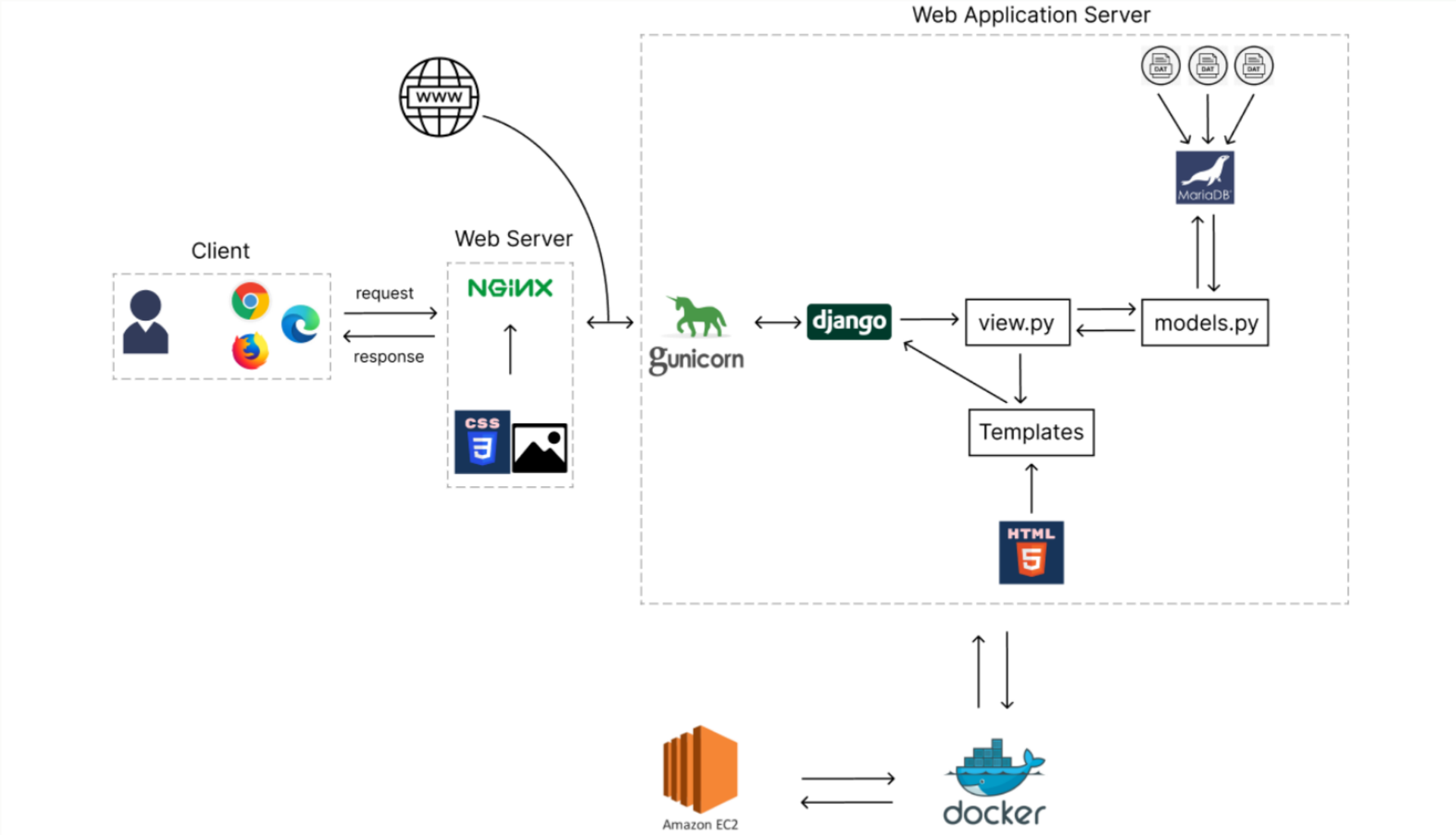
결과 : 각 퍼스널 컬러 유형별 색상 팔레트를 참고하여 제품 추천



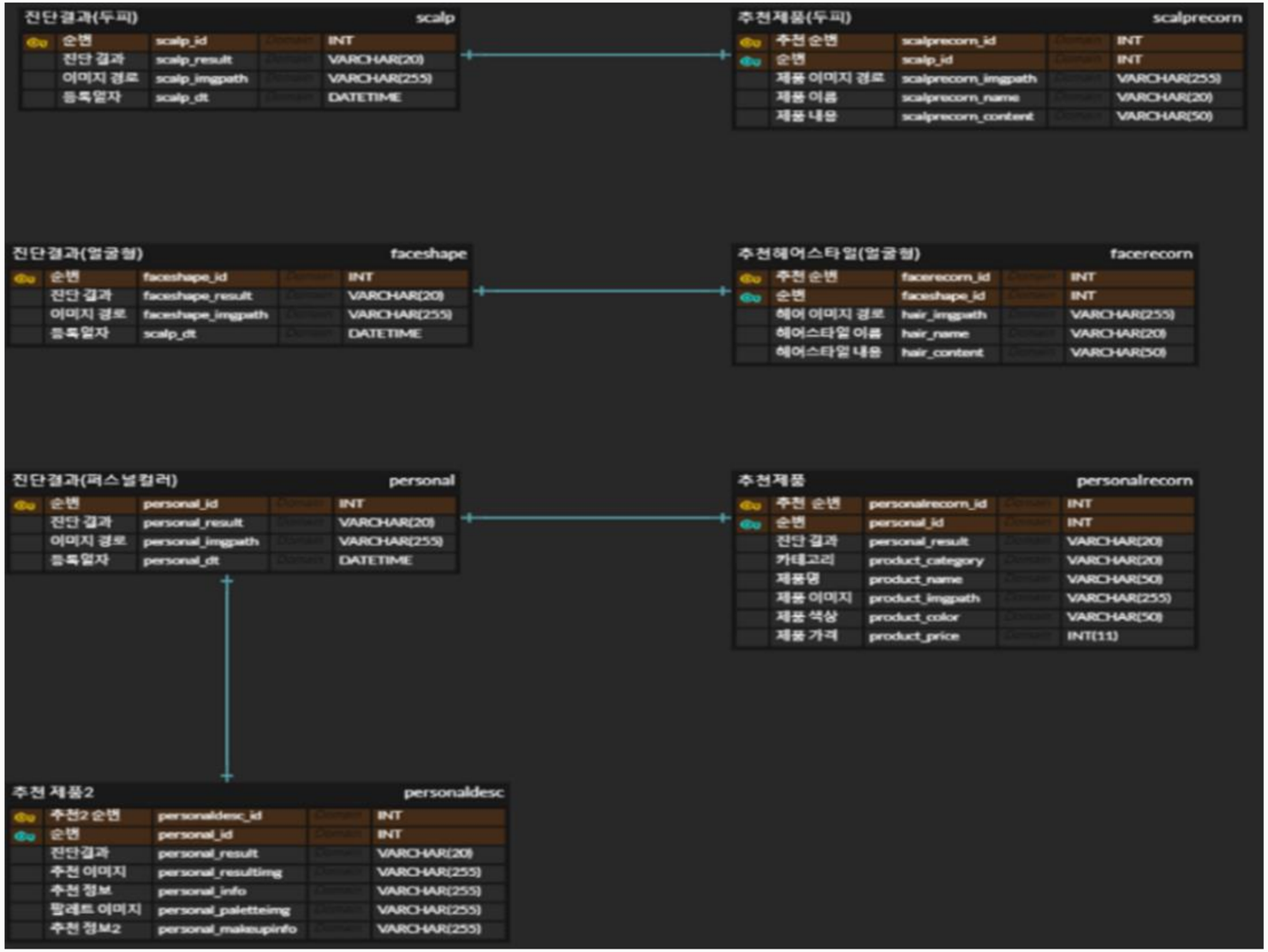
04

서비스 설계 및 구현

서비스 설계도 – 시스템 구성도

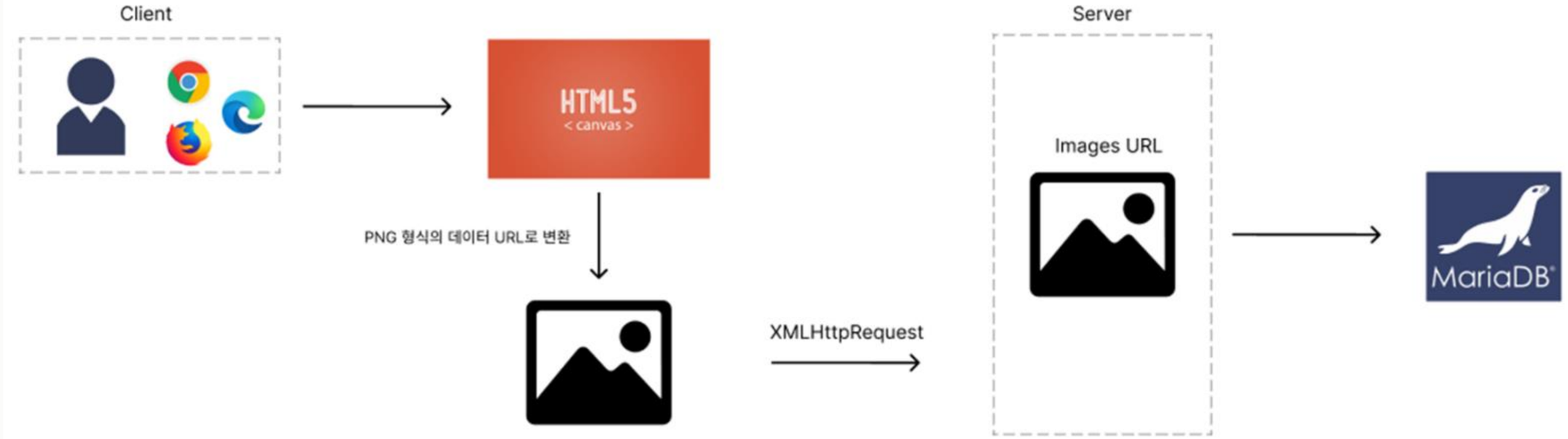


서비스 설계도 – ERD


진단결과(두피)
scalp
순번 scalp_id INT
진단 결과 scalp_result VARCHAR(20)
이미지 경로 scalp_imgpath VARCHAR(255)
등록일자 scalp_dt DATETIME
추천제품(두피)
scalprecorn
추천 순번 scalprecorn_id INT
순번 scalp_id INT
제품 이미지 경로 scalprecorn_imgpath VARCHAR(255)
제품 이름 scalprecorn_name VARCHAR(20)
제품 내용 scalprecorn_content VARCHAR(50)
진단결과(얼굴형)
faceshape
순번 faceshape_id INT
진단 결과 faceshape_result VARCHAR(20)
이미지 경로 faceshape_imgpath VARCHAR(255)
등록일자 scalp_dt DATETIME
추천헤어스타일(얼굴형)
facerecorn
추천 순번 facerecorn_id INT
순번 faceshape_id INT
헤어 이미지 경로 hair_imgpath VARCHAR(255)
헤어스타일 이름 hair_name VARCHAR(20)
헤어스타일 내용 hair_content VARCHAR(50)
진단결과(퍼스널컬러)
personal
순번 personal_id INT
진단 결과 personal_result VARCHAR(20)
이미지 경로 personal_imgpath VARCHAR(255)
등록일자 personal_dt DATETIME
추천제품
personalrecorn
추천 순번 personalrecorn_id INT
순번 personal_id INT
진단 결과 personal_result VARCHAR(20)
카테고리 product_category VARCHAR(20)
제품명 product_name VARCHAR(50)
제품 이미지 product_imgpath VARCHAR(255)
제품 색상 product_color VARCHAR(50)
제품 가격 product_price INT(11)
추천 제품2
personaldesc
추천2 순번 personaldesc_id INT
순번 personal_id INT
진단결과 personal_result VARCHAR(20)
추천 이미지 personal_resultimg VARCHAR(255)
추천 정보 personal_info VARCHAR(255)
팔레트 이미지 personal_paletteimg VARCHAR(255)
추천 정보2 personal_makeupinfo VARCHAR(255)

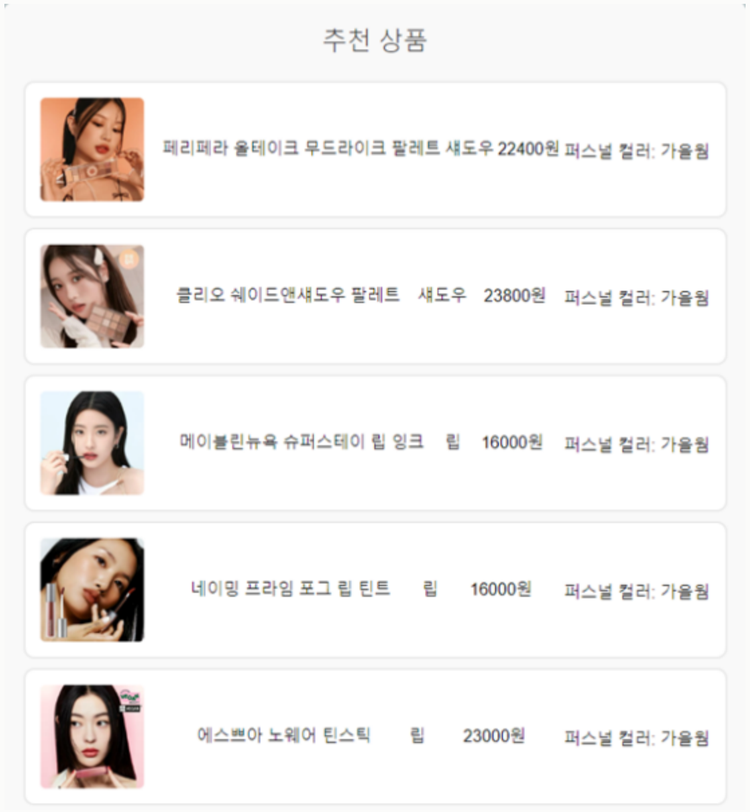
서비스 설계도

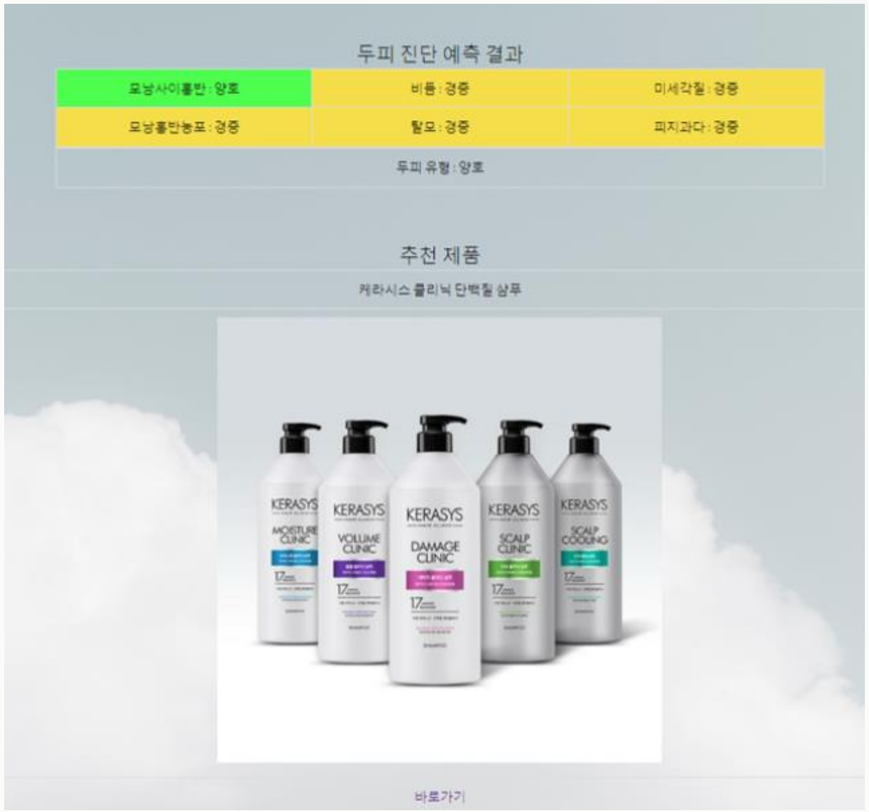
실시간 웹 캠으로 찍은 사진들을 ajax처리 방식으로 DB에 저장



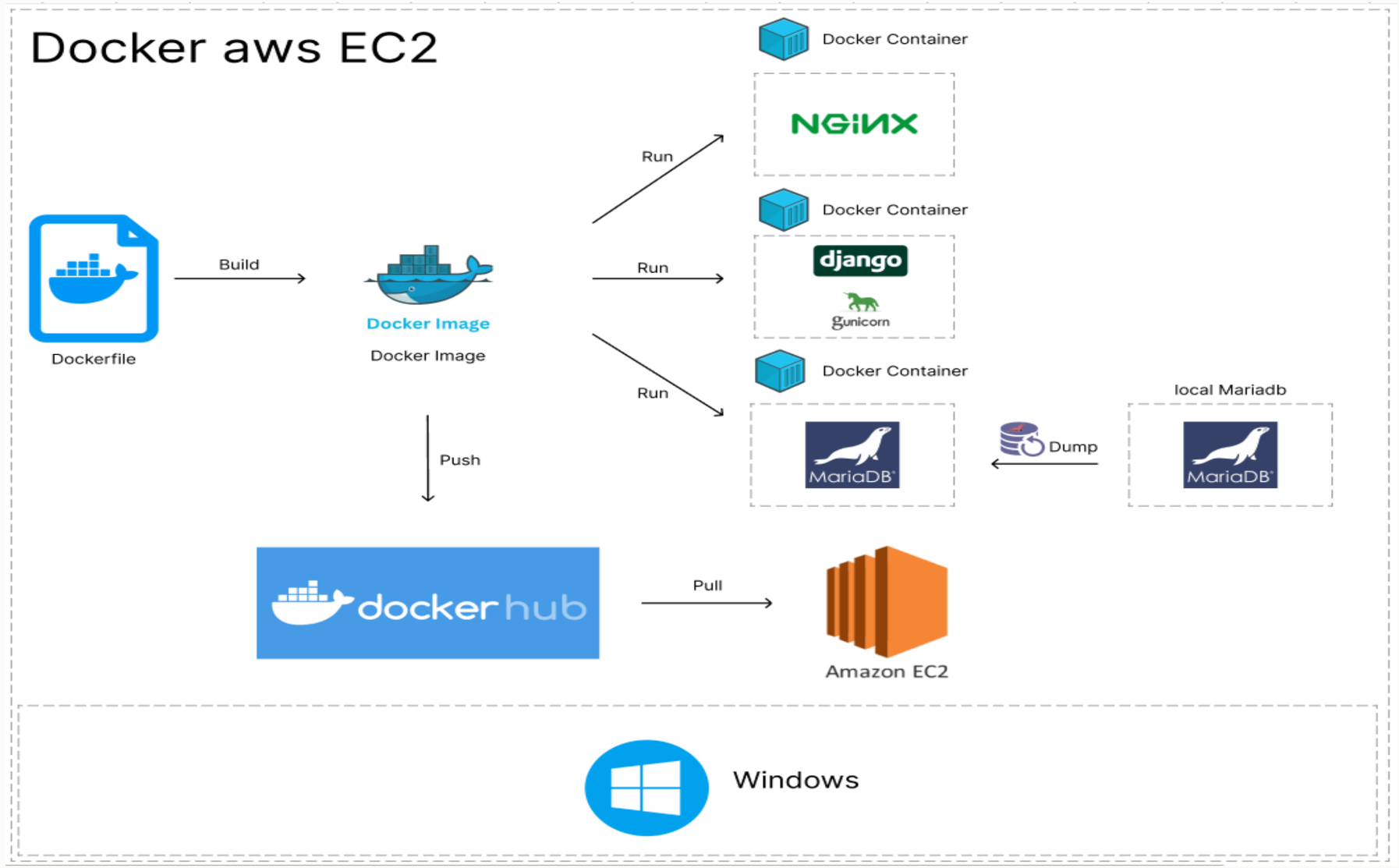
서비스 구현 – 추천 서비스

DB에 저장된 분석 결과를 기반으로 추천 서비스 제공

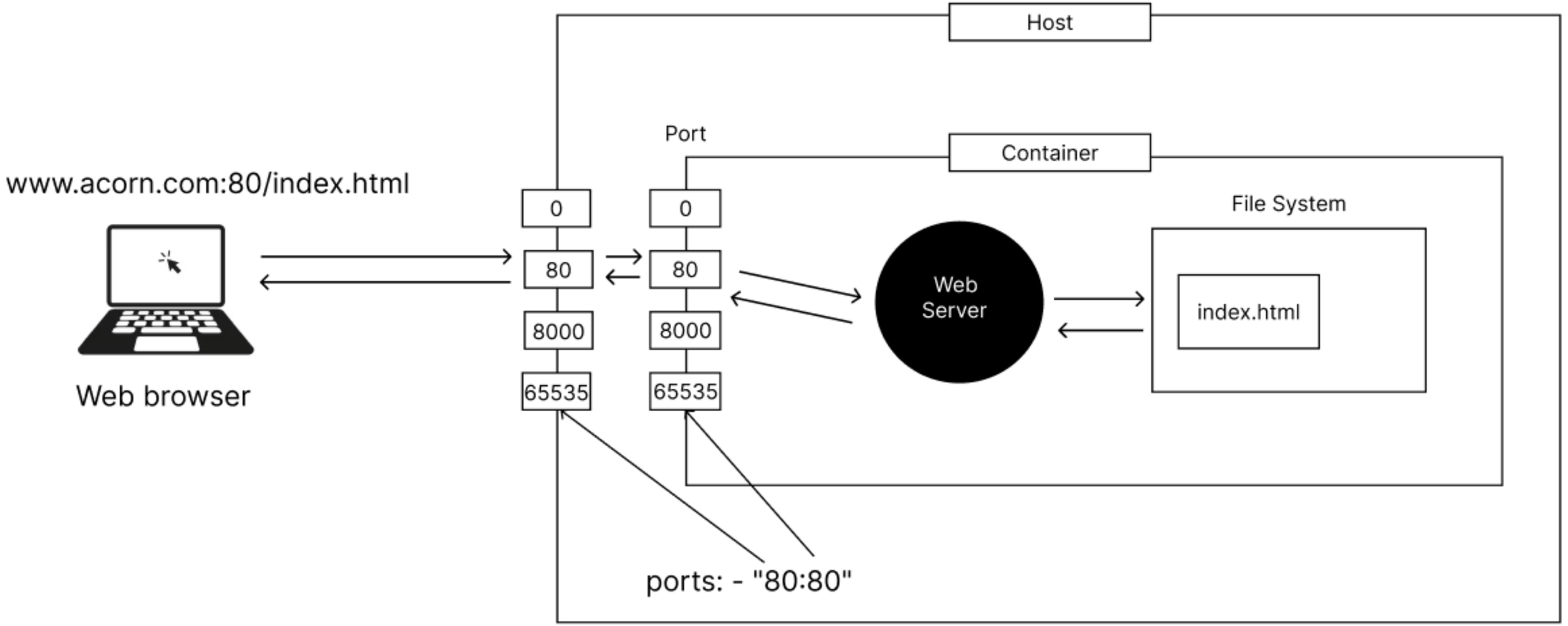




Docker


Docker aws EC2
Docker Container
NGINX
Run
Docker Container
Build
Run
django
gunicorn
Dockerfile
Docker Image
Docker Image
Docker Container
Run
local Mariadb
Push
MariaDB
Dump
MariaDB
dockerhub
Pull
Amazon EC2
Windows

Docker


Host
Port
Container
www.acorn.com:80/index.html
0
80
8000
65535
0
80
8000
65535
File System
Web
Server
index.html
Web browser
ports: - "80:80"

05

시연

06

질의응답

1팀 뷰티팔레트

THANK YOU

경청해 주셔서 감사합니다.