



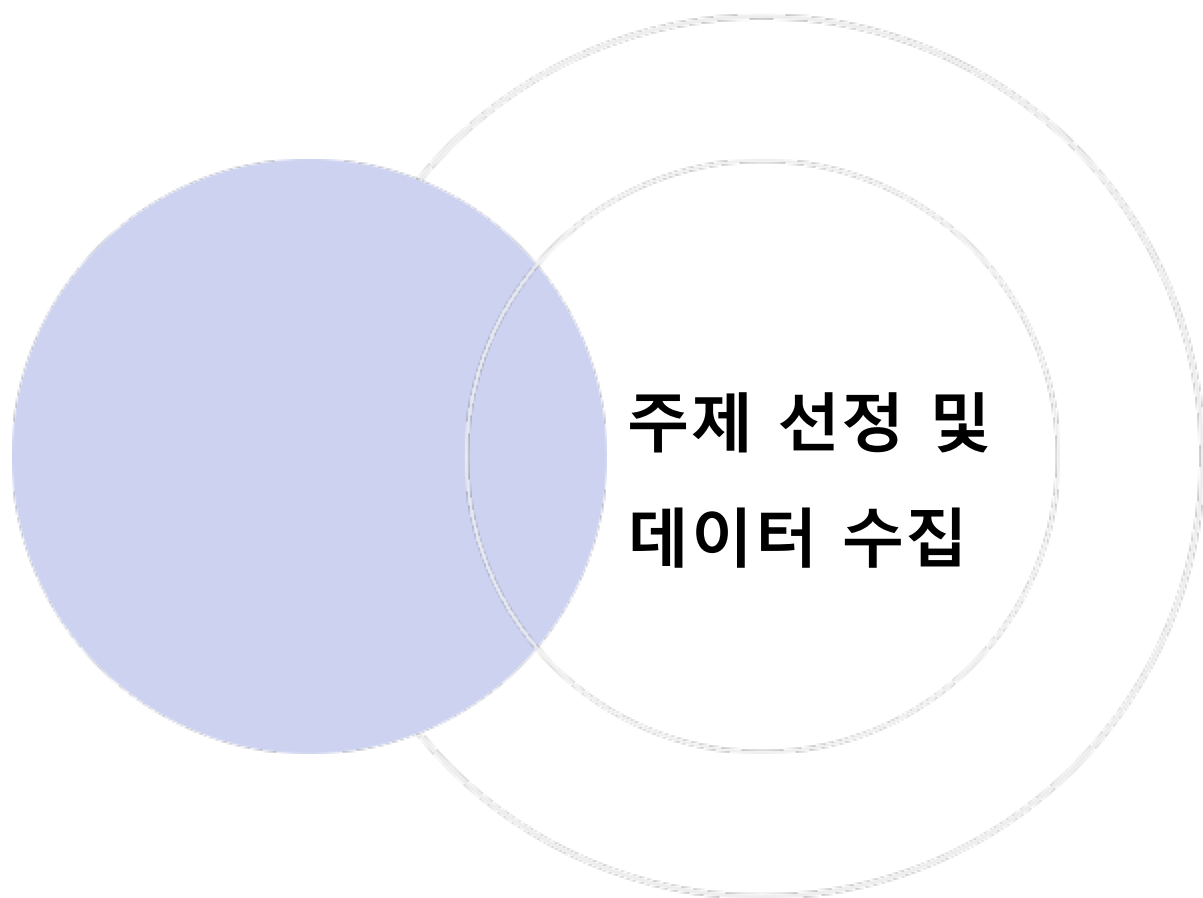
시퀀스데이터 분석 프로젝트

이미지를 통한 알약 분석



Contents

1. 주제 선정 및 데이터 수집
2. EDA 및 Data Preprocessing
3. Modeling
4. 최종 결과
5. 참고 문헌

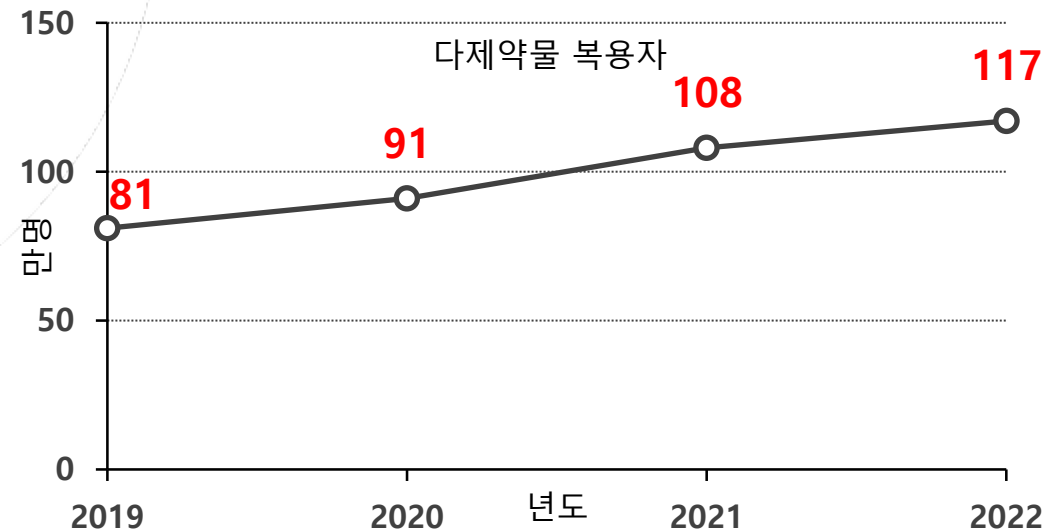




약물 복용

OECD 발표 자료에 의하면 우리나라 75세 이상 환자 중 5개 이상의 약물을 복용 중인 노인 비율은 2021년 기준 64.2%로, OECD 평균(48.6%) 보다 높고 포르투갈(73.0%), 이탈리아(64.7%)에 이어 3번째

2022년 상반기 건강보험 가입자 진료기준 10종 이상의 다제약물을 2달 이상 복용 중인 환자 117만 5130명



2019년 ~ 2022년 해마다 증가

주제 선정 동기



01

66세 인구의 53.7%에서 1종 이상의 '노인 부적절 약물'을 복용, 1인당 평균 2.4개를 복용하는 것으로 분석



02

약품 처방이 매우 쉽고 많은 수가 전문의약품 및 일반의약품을 혼용, 부작용을 겪는 사례들이 발생



03

의약품 혼용 금지 조합 및 부작용에 대한 정보 접근성이 떨어짐

프로젝트 목표

정확한 정보 전달

약품 인식 및 정보
제공을 통해
약품의 정확한
정보를 전달

편리한 정보 획득

어떤 사용자 층도
정보 획득이
쉽고 편리

오남용 피해 최소화

최종적으로 약물에 대한
정보를 인지하여
약물 오남용을 방지하고
피해를 최소화



데이터



공공데이터 정보

공공데이터

• 개요

• [공공데이터 공개](#)

주요 통계

특허등재 통계

공공데이터 공개

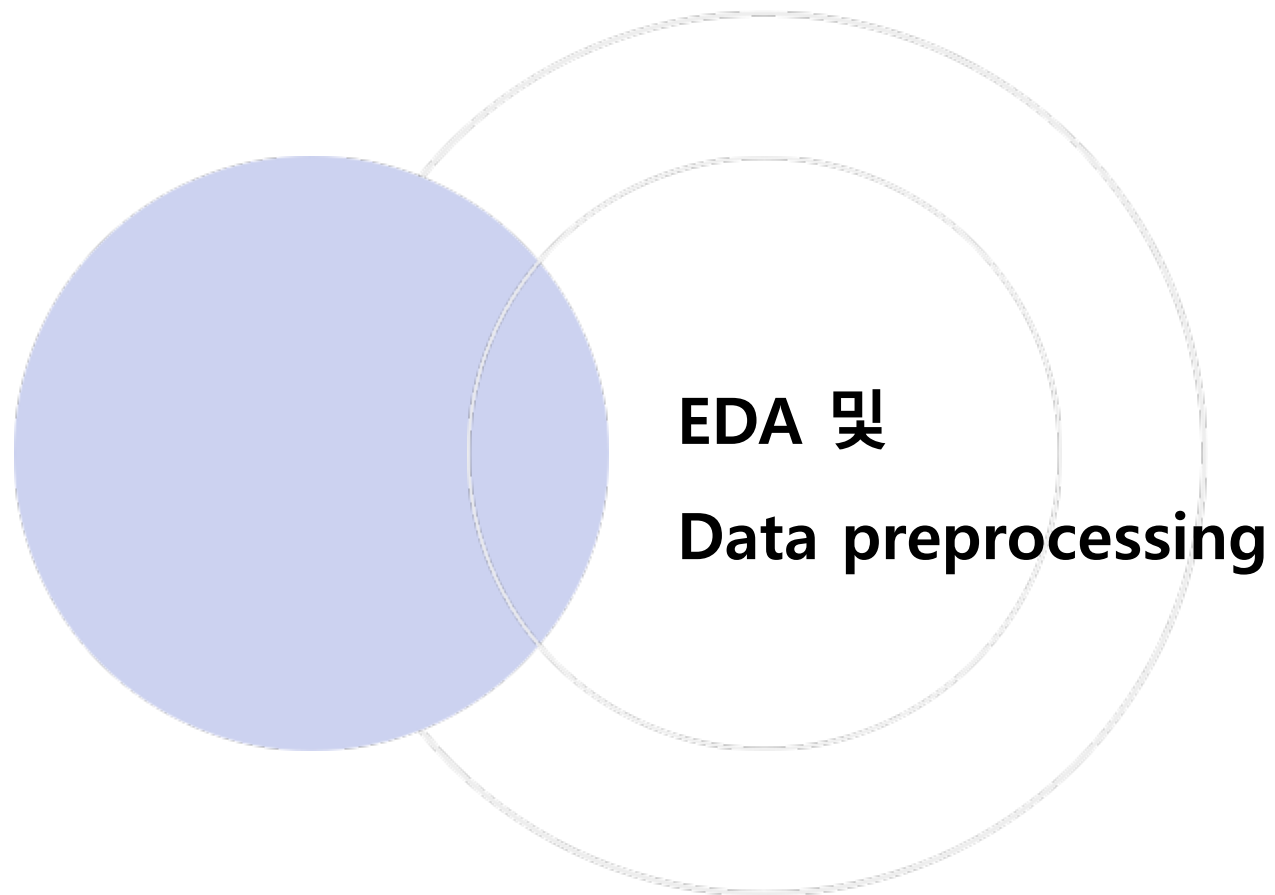
상세정보

서비스명	의약품 낱알식별		
제공형태	EXCEL/CSV	제공주기	매일 04:00 AM
다운로드	엑셀다운로드 CSV다운로드		

목록



의약품안전나라 의약품 낱알식별 데이터 사용



COLUMN INFORMATION

(Row: 25,125 / Column: 30)

#	Column	Dtype
1	품목일련번호	int64
2	품목명	object
3	업소일련번호	int64
4	업소명	object
5	성상	object
6	큰제품이미지	object
7	표시앞	object
8	표시뒤	object
9	의약품제형	object
10	색상앞	object

#	Column	Dtype
11	색상뒤	object
12	분할선앞	object
13	분할선뒤	int64
14	크기장축	object
15	크기단축	object
16	크기두께	object
17	이미지생성일자	int64
18	분류번호	object
19	분류명	object
20	전문일반구분	object

#	Column	Dtype
21	품목허가일자	int64
22	제형코드명	object
23	표기내용앞	int64
24	표기내용뒤	object
25	표기이미지앞	object
26	표기이미지뒤	object
27	표기코드앞	object
28	표기코드뒤	object
29	변경일자	object
30	사업자번호	int64

SAMPLE DATA

#	Column	Value
1	품목일련번호	200808877
2	품목명	페라트라정2.5밀리그램(레트로졸)
3	업소일련번호	19560004
4	업소명	(주)유한양행
5	성상	어두운황색의원형필름코팅정
6	큰제품이미지	https://nedrug.mfds.go.kr/pbp/cmn/itemImageDownload/147426403087300107
7	표시앞	YH
8	표시뒤	LT
9	의약품제형	원형
10	색상앞	노랑

SAMPLE DATA

#	Column	Value
11	색상뒤	-
12	분할선앞	-
13	분할선뒤	-
14	크기장축	6.1
15	크기단축	6.1
16	크기두께	3.5
17	이미지생성일자	20100429
18	분류번호	4210
19	분류명	항악성종양제
20	전문일반구분	전문의약품

SAMPLE DATA

#	Column	Value
21	품목허가일자	20080820
22	제형코드명	필름코팅정
23	표기내용앞	-
24	표기내용뒤	-
25	표기이미지앞	-
26	표기이미지뒤	-
27	표기코드앞	-
28	표기코드뒤	-
29	변경일자	-
30	사업자번호	1188100601

MISSING VALUES (#)

(전체 25,125개의 데이터)

#	Column	Nan
1	품목일련번호	0
2	품목명	0
3	업소일련번호	0
4	업소명	0
5	성상	0
6	큰제품이미지	0
7	표시앞	330
8	표시뒤	12598
9	의약품제형	2
10	색상앞	1

#	Column	Nan
11	색상뒤	21168
12	분할선앞	25194
13	분할선뒤	25016
14	크기장축	58
15	크기단축	76
16	크기두께	90
17	이미지생성일자	0
18	분류번호	168
19	분류명	168
20	전문일반구분	0

#	Column	Nan
21	품목허가일자	0
22	제형코드명	31
23	표기내용앞	22899
24	표기내용뒤	24607
25	표기이미지앞	22889
26	표기이미지뒤	24606
27	표기코드앞	22889
28	표기코드뒤	24606
29	변경일자	5933
30	사업자번호	0

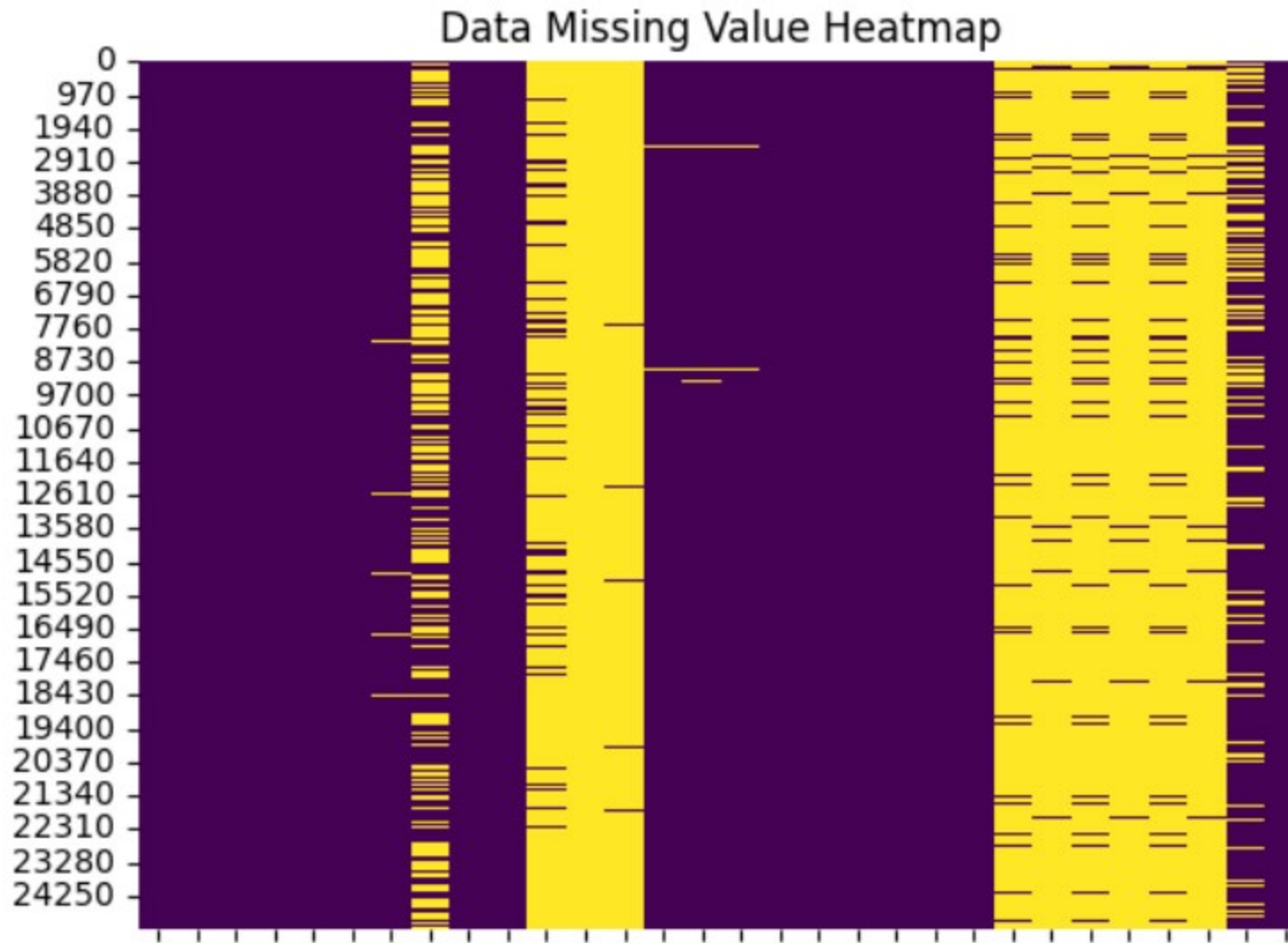
MISSING VALUES (%)

(소수점 첫째 자리에서 반올림)

#	Column	%
1	품목일련번호	0
2	품목명	0
3	업소일련번호	0
4	업소명	0
5	성상	0
6	큰제품이미지	0
7	표시앞	1
8	표시뒤	50
9	의약품제형	0
10	색상앞	0

#	Column	%
11	색상뒤	84
12	분할선앞	100
13	분할선뒤	99
14	크기장축	0
15	크기단축	0
16	크기두께	0
17	이미지생성일자	0
18	분류번호	1
19	분류명	1
20	전문일반구분	0

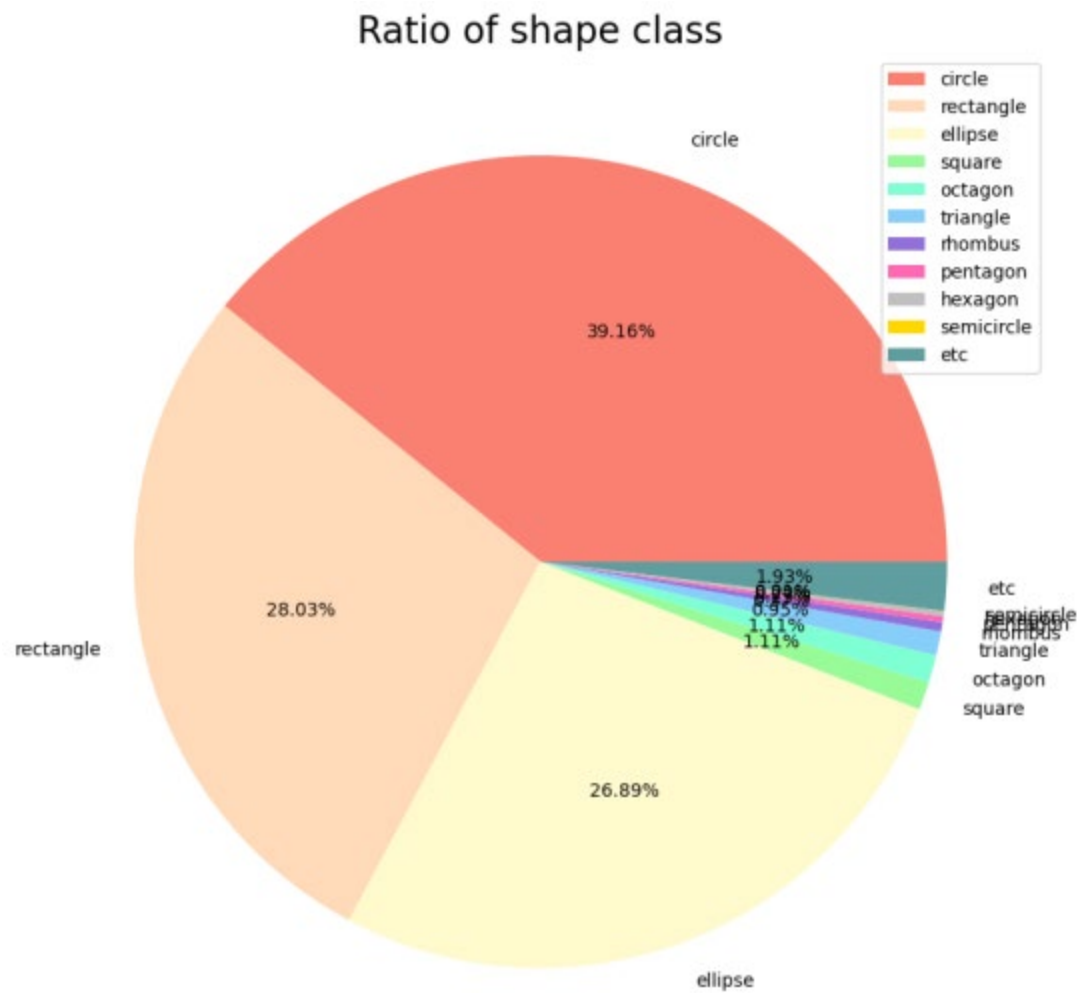
#	Column	%
21	품목허가일자	0
22	제형코드명	0
23	표기내용앞	91
24	표기내용뒤	98
25	표기이미지앞	91
26	표기이미지뒤	98
27	표기코드앞	91
28	표기코드뒤	98
29	변경일자	24
30	사업자번호	0



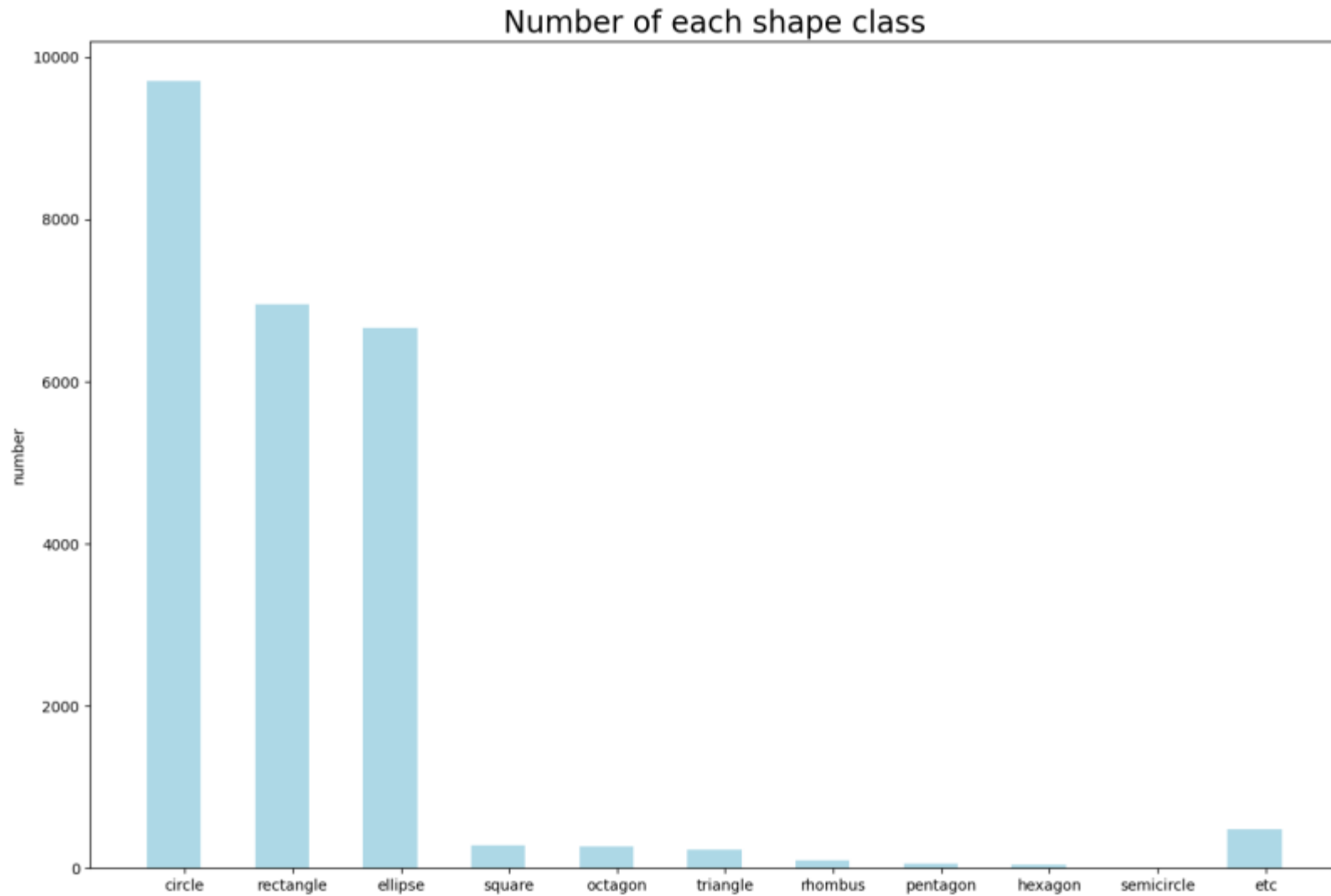
SHAPE

(소수점 셋째 자리에서 반올림)

의약품제형	개수	비율(%)
원형	9706	39.16
장방형	6948	28.03
타원형	6666	26.89
사각형	276	1.11
팔각형	274	1.11
삼각형	235	0.95
마름모형	91	0.37
오각형	58	0.23
육각형	50	0.20
반원형	3	0.01
기타	479	1.93



SHAPE



COLOR(FRONT)

색상앞	개수
노랑 투명	162
빨강 투명	99
파랑 투명	95
초록 투명	89
주황 투명	79
청록 투명	45
하양 노랑	30
갈색 투명	23
보라 투명	21
하양 투명	14
하양 파랑	12
분홍 투명	6

색상앞	개수
하양 갈색	5
연두 투명	5
파랑 열은	1
갈색 진한	1
하양 초록	1
하양 주황 투명	1
하양 빨강	1
분홍 진한	1
분홍 열은	1
자주 투명	1
하양 청록	1
-	1

두 개 이상의 색상 값을 갖는 행 694개
→ 가장 앞의 1개 색상 값만 사용

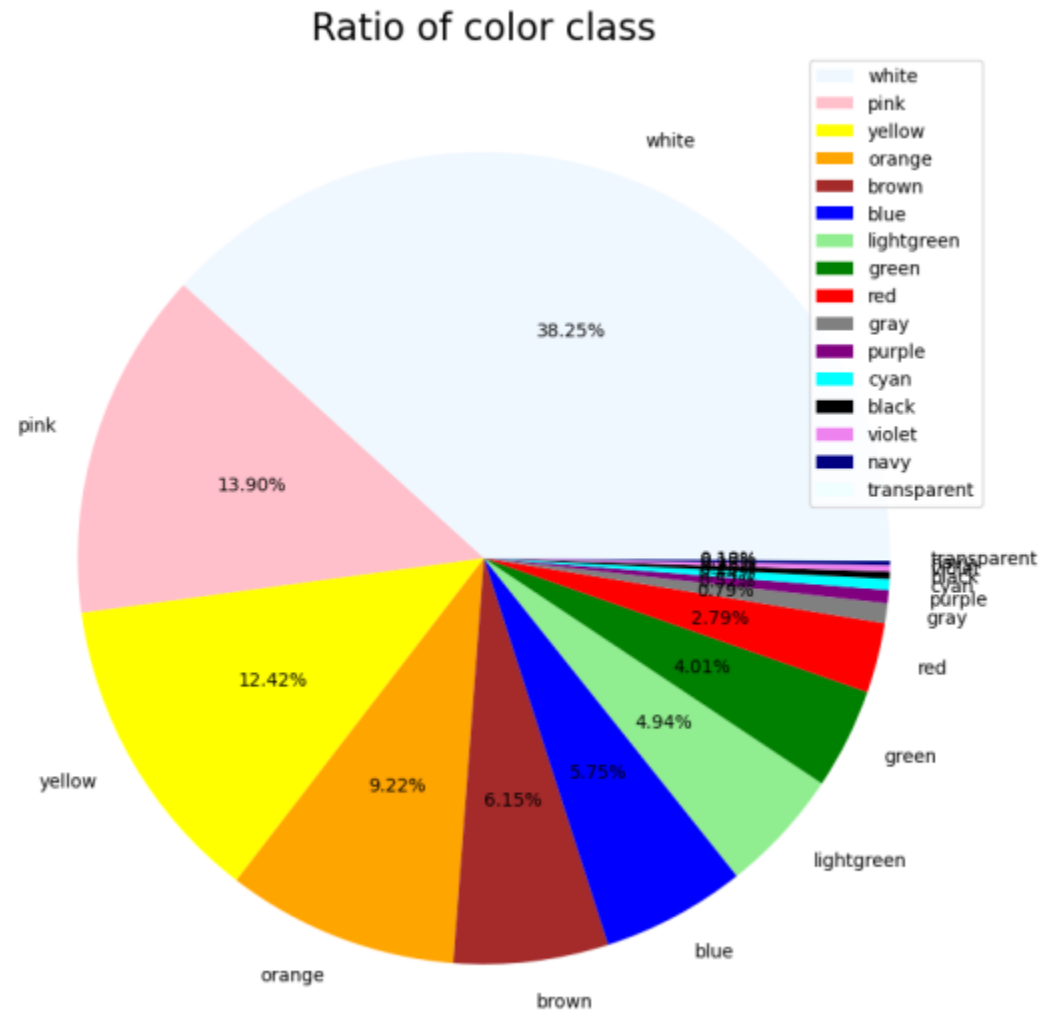
'-' 값을 갖는 행 1개
→ 실물 이미지 확인 후 '노랑'으로 대체



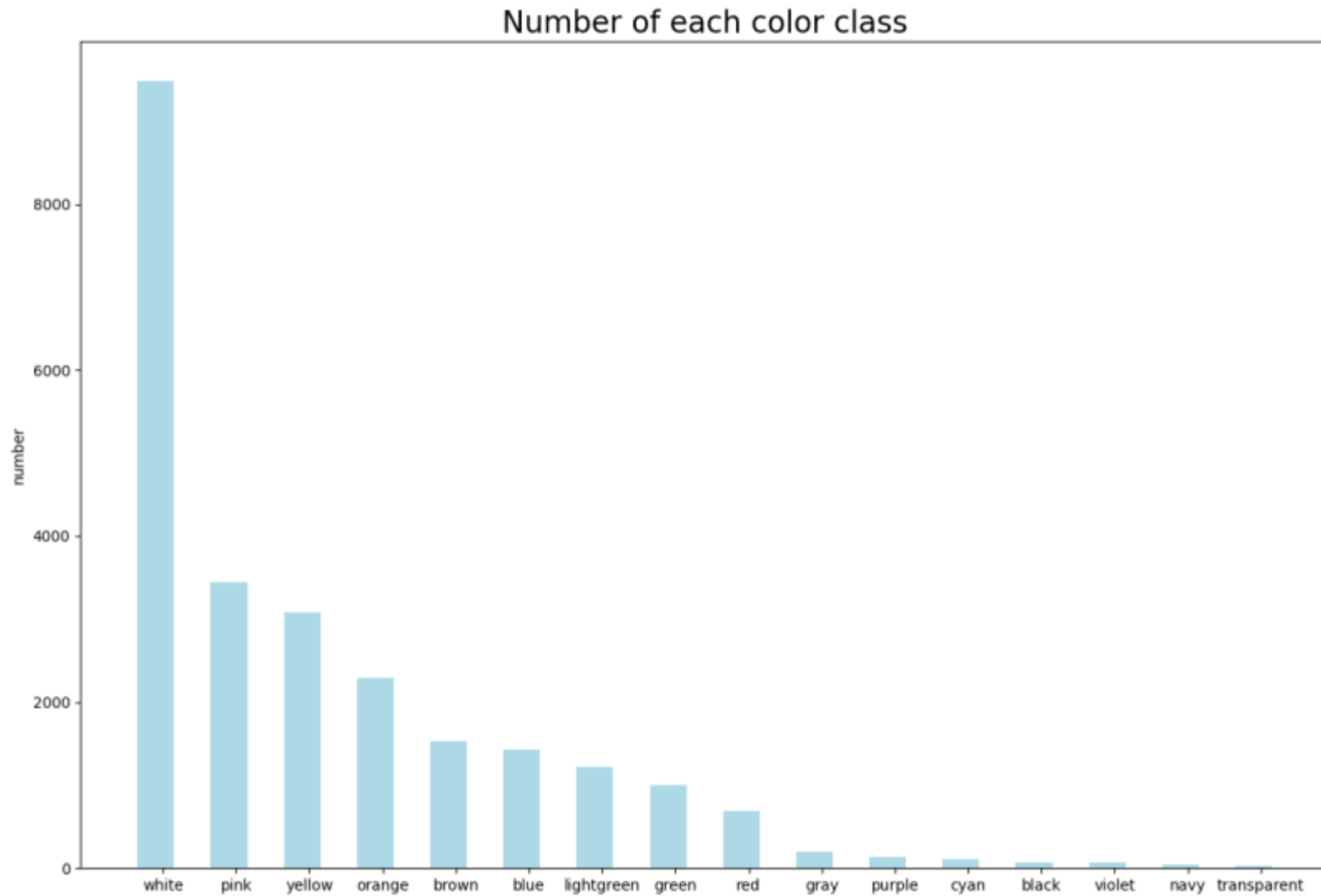
COLOR(FRONT)

(소수점 셋째 자리에서 반올림)

색상앞	개수	비율(%)
하양	9482	38.25
분홍	3445	13.90
노랑	3079	12.42
주황	2286	9.22
갈색	1525	6.15
파랑	1426	5.75
연두	1224	4.94
초록	994	4.01
빨강	691	2.79
회색	196	0.79
보라	129	0.52
청록	109	0.44
검정	71	0.29
자주	63	0.25
남색	42	0.17
투명	25	0.10



COLOR(FRONT)



COLOR(BACK)

색상앞	개수
하양 투명	24
분홍 투명	17
노랑 옐은	17
주황 투명	14
파랑 옐은	14
하양 파랑	12
하양 노랑	8
갈색 옐은	5
갈색 진한	4
노랑 투명	4
회색 진한	4
초록 투명	2
청록 투명	2
분홍 옐은	1

색상앞	개수
노랑 진한	1
초록 진한	1
초록 옐은	1
검정 투명	1
청록 옐은	1
파랑 진한	1
주황 옐은	1
자주 진한	1
하양 청록	1
하양 초록 투명	1
자주 옐은	1
연두 진한	1
-	21168

두 개 이상의 색상 값을 갖는 행 140개
→ 가장 앞의 1개 색상 값만 사용

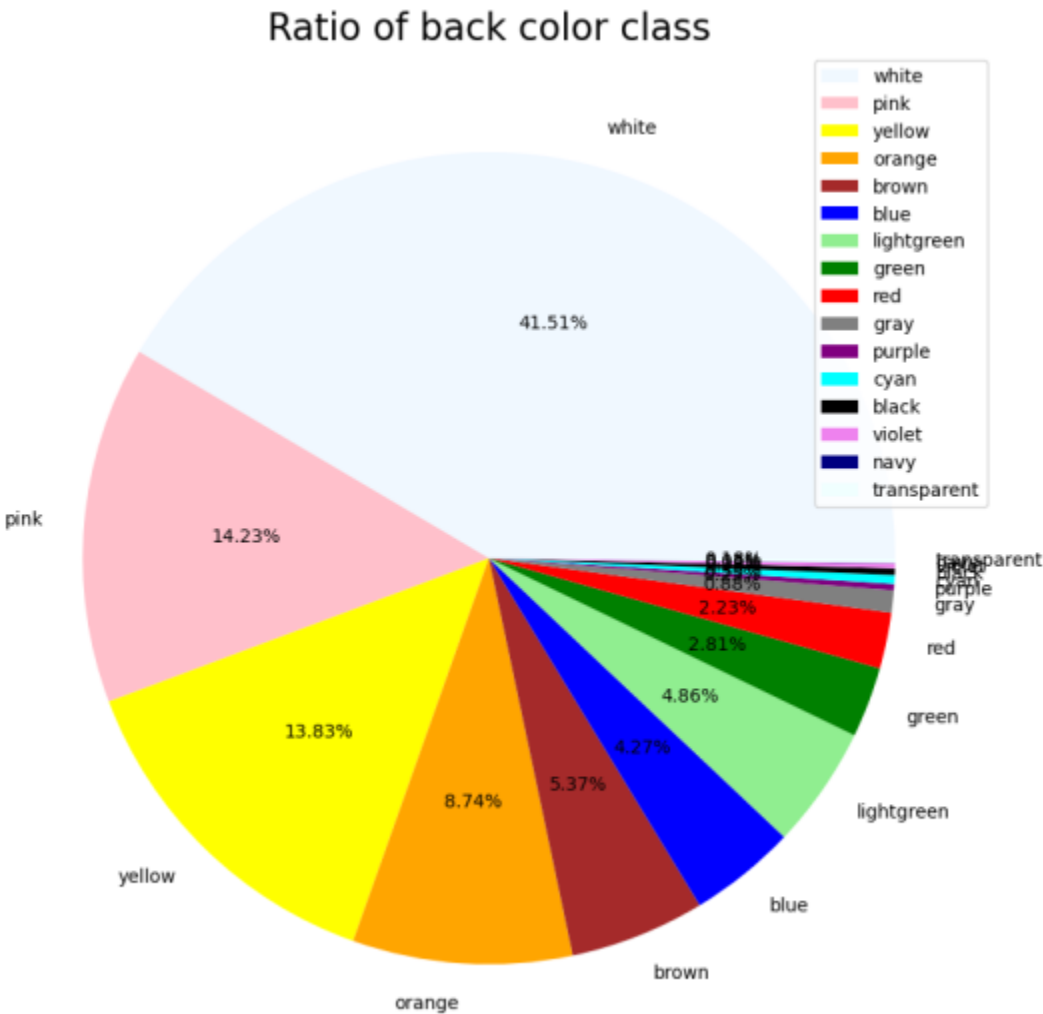
'-' 값을 갖는 행 1개
→ 앞면 색상으로 각각 대체



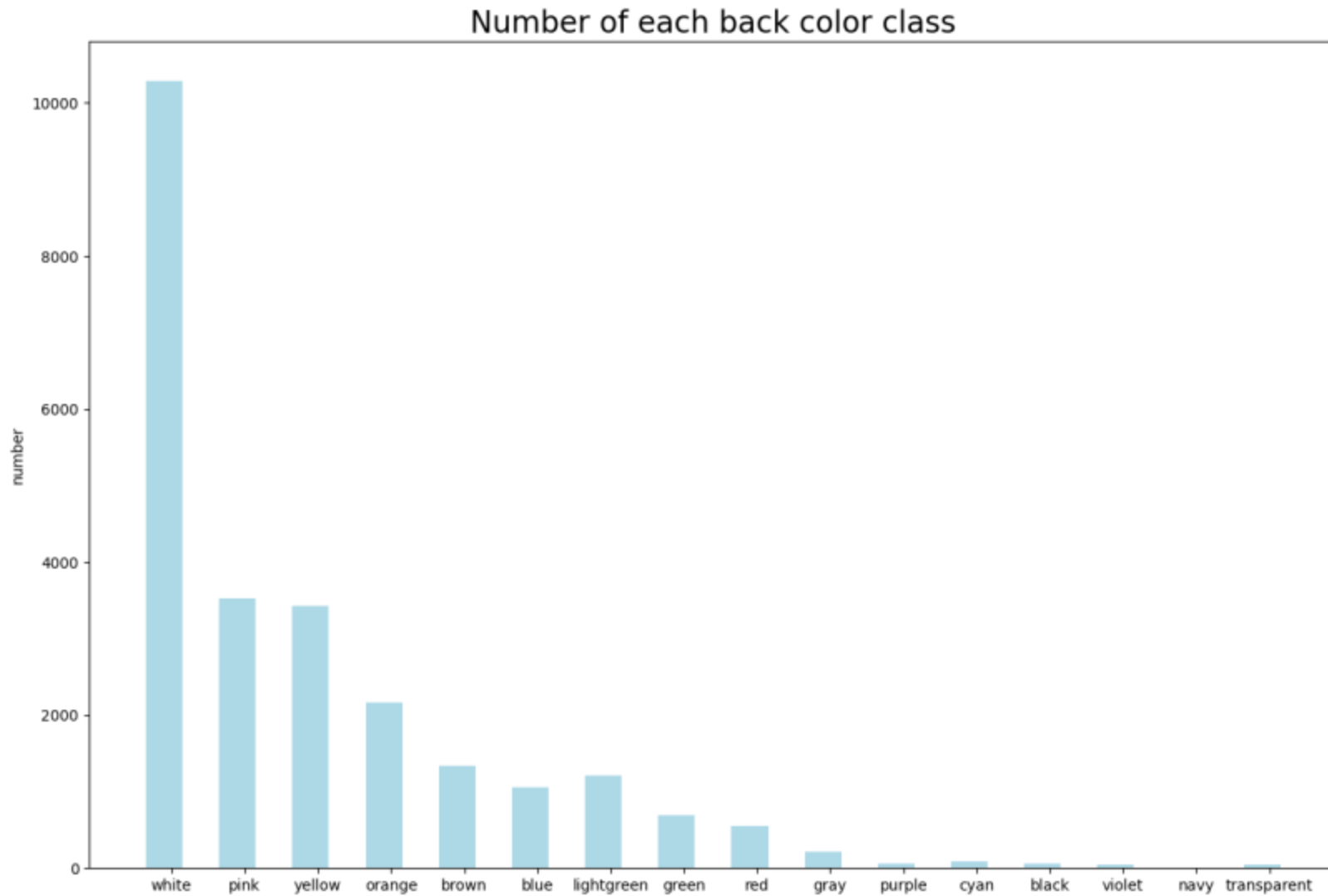
COLOR(BACK)

(소수점 셋째 자리에서 반올림)

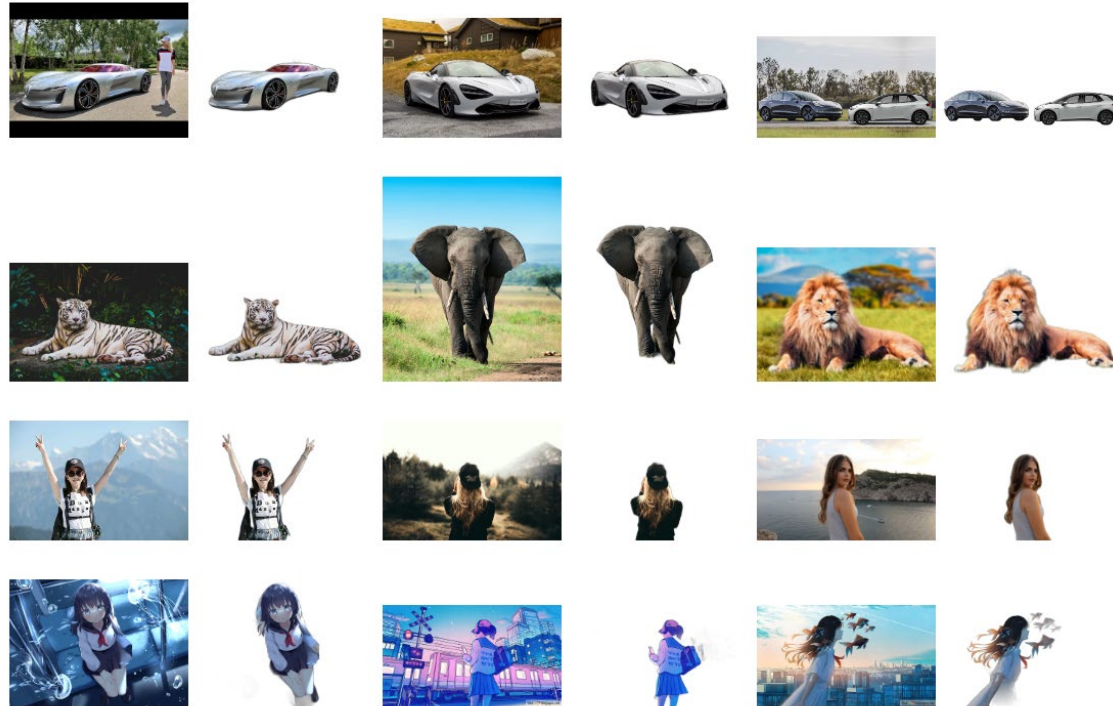
색상앞	개수	비율(%)
하양	10288	41.51
분홍	3528	14.23
노랑	3429	13.83
주황	2166	8.74
갈색	1332	5.37
파랑	1058	4.27
연두	1205	4.86
초록	696	2.81
빨강	553	2.23
회색	219	0.88
보라	62	0.25
청록	87	0.35
검정	65	0.26
자주	48	0.19
남색	5	0.02
투명	45	0.18



COLOR(BACK)

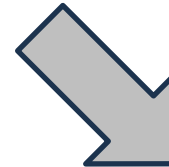
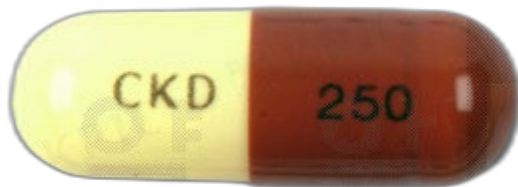
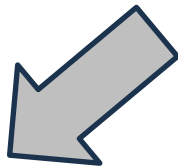
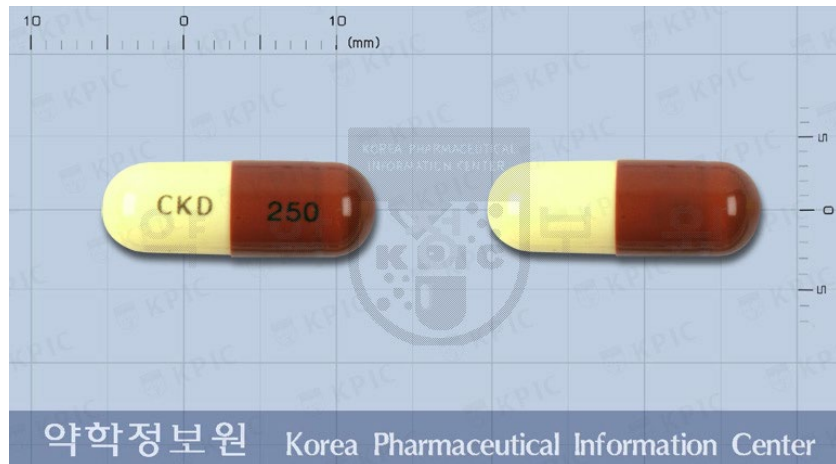


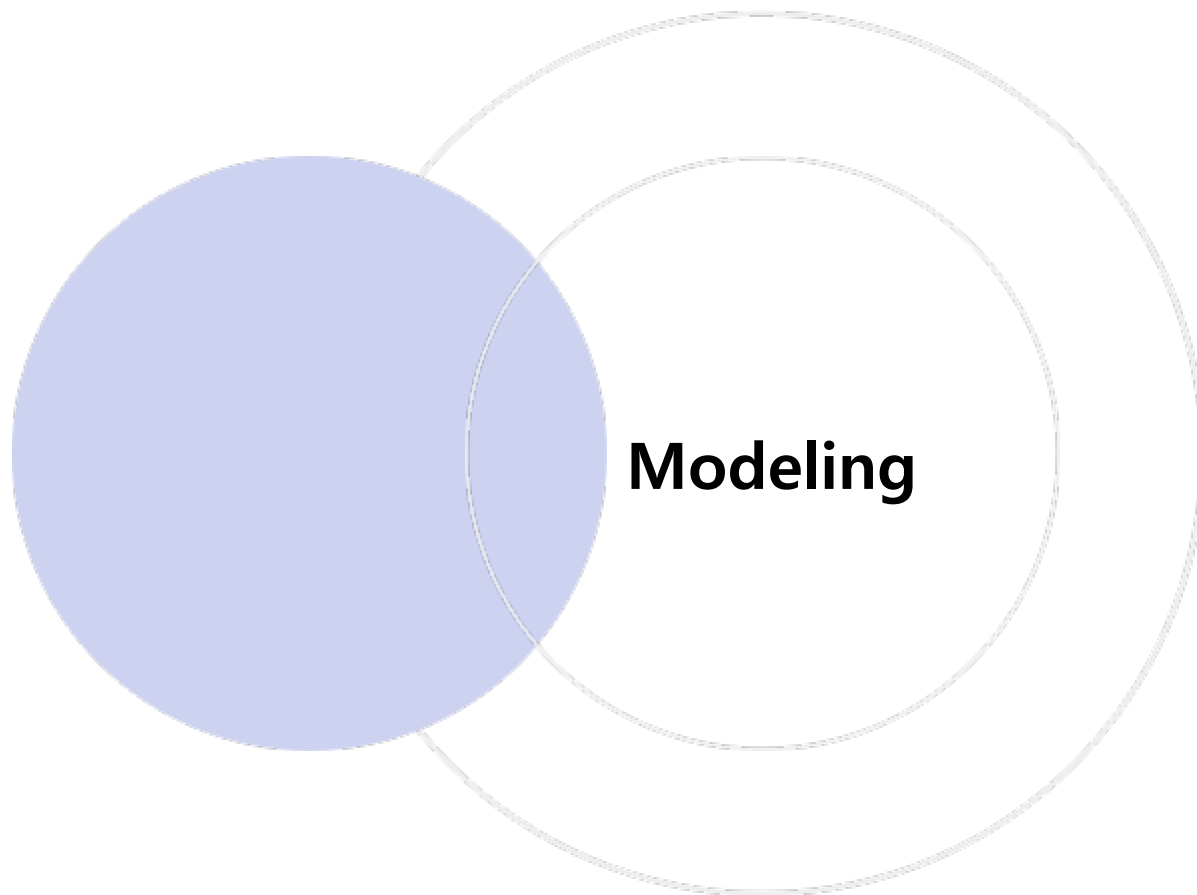
Rembg

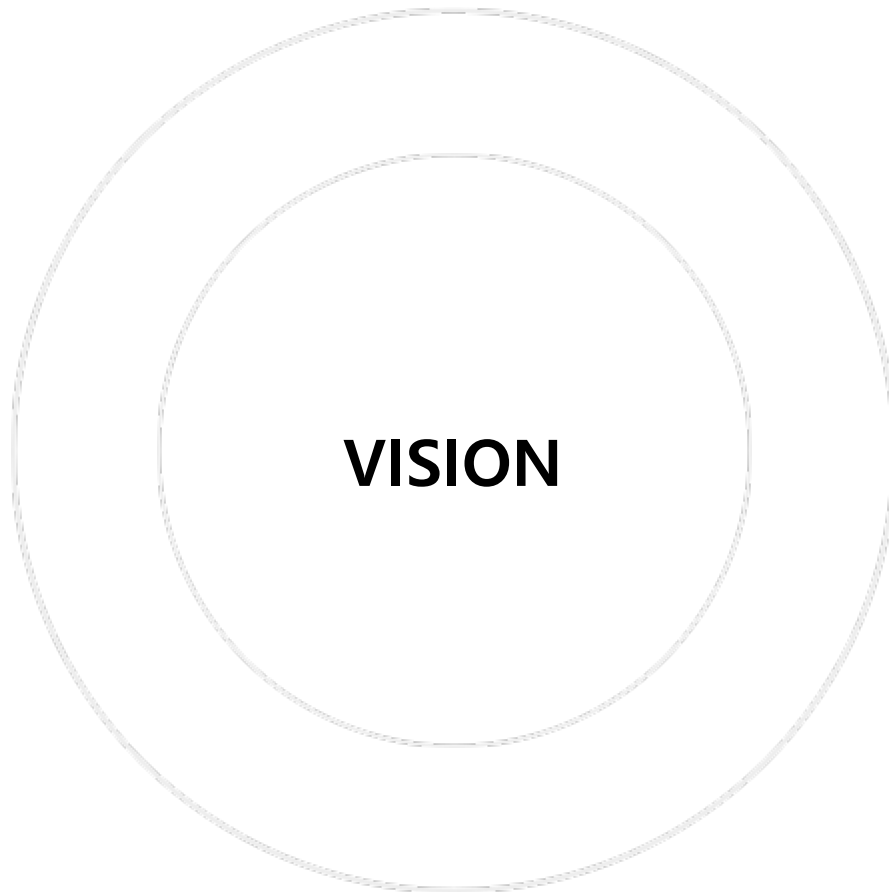


Rembg는 사물을 인식하고 배경을 제거하는 패키지로,
간단하게 파이썬에서 배경을 지울 수 있습니다.

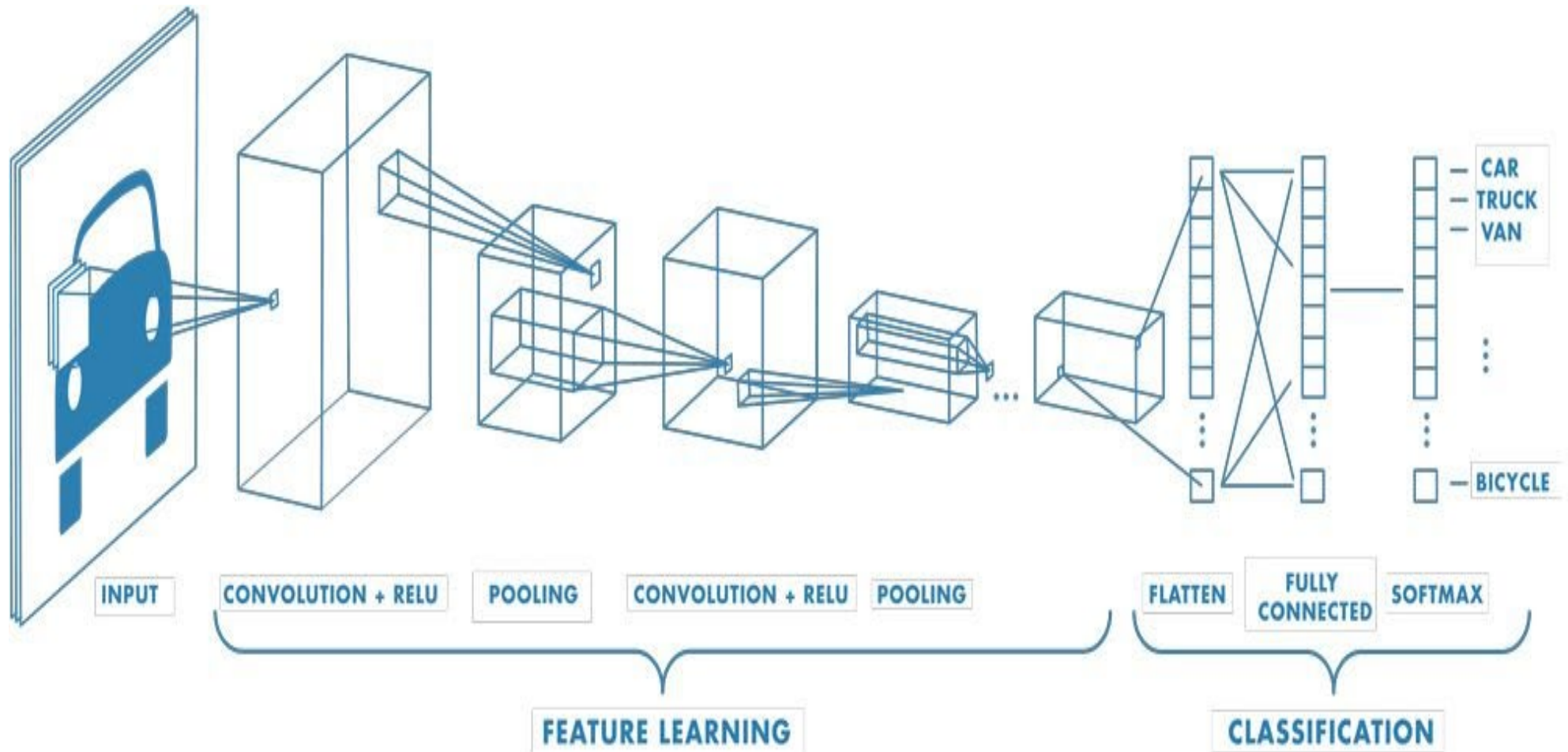
Back and forth separation







CNN



Hyper Parameter

cnn_shape

```
# Build model
def build(input_shape, classes):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Conv2D(128, (5, 5), activation='relu'))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(512, activation='relu'))
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dense(classes, activation='softmax'))
    return model
```

Hyper Parameter

cnn_shape

Filter 크기	Filter 개수
5 x 5	64
3 x 3	128
5 x 5	128

Test loss	Test accuracy
0.144975	0.968535
0.155883	0.968132
0.166936	0.970552

Hyper Parameter

cnn_color

	1번 모델	2번 모델	3번 모델
구조	Conv2D(32) → MaxPooling2D → Conv2D(64) → MaxPooling2D → Flatten → Dense(512) → Dense(256) → Dense(classes)	Conv2D(32) → MaxPooling2D → Conv2D(128) → MaxPooling2D → Flatten → Dense(512) → Dense(256) → Dense(classes)	Conv2D(32) → MaxPooling2D → Conv2D(64) → Dropout → MaxPooling2D → Flatten → Dense(512) → Dense(256) → Dense(classes)
특징	두 개의 Conv2D 층과 두 개의 완전 연결된 층	1번 모델보다 두번째 Conv2D층의 필터수가 많음	첫 번째 Conv2D층과 두 번째 Conv2D층 사이에 Dropout 추가 → 과적합 방지
성능	Test loss : 0.348390 Test accuracy : 0.894513	Test loss : 0.775313 Test accuracy : 0.810004	Test loss : 0.451353 Test accuracy : 0.852964

Hyper Parameter

cnn_color

4번 모델

Conv2D(32) →
MaxPooling2D →
Conv2D(64) →
MaxPooling2D →
Flatten →
Dense(512) →
Dense(256) →
Dense(128) →
Dense(classes)

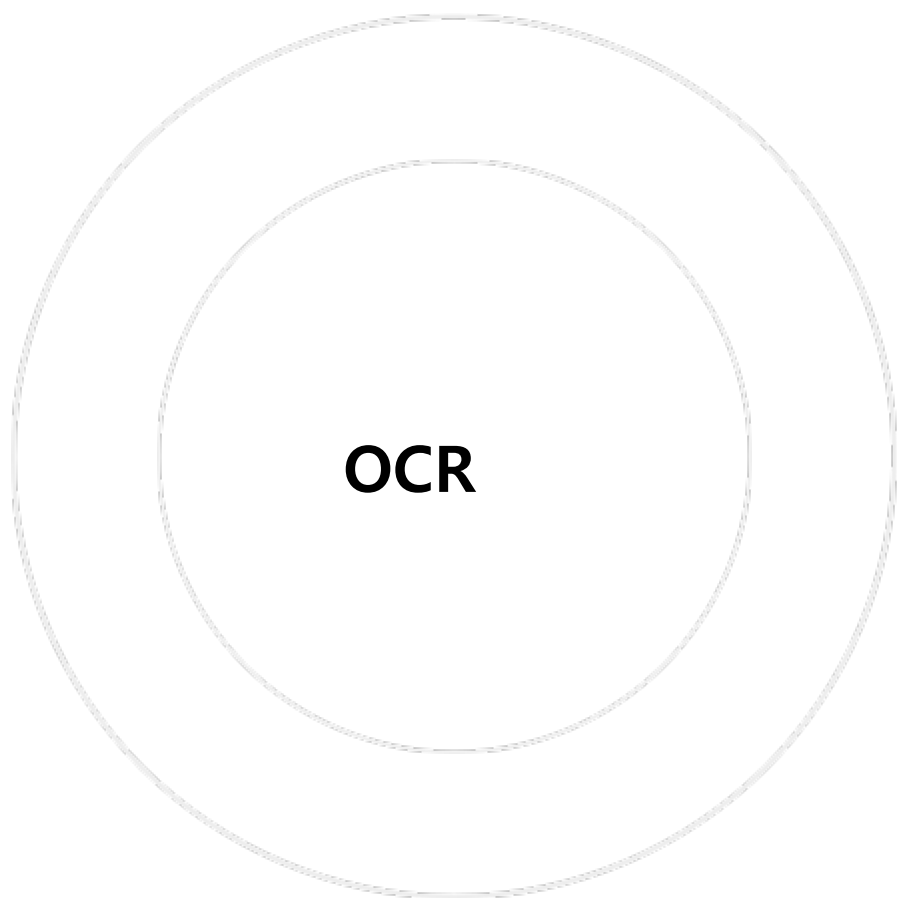
세 번째 모델과 비슷한
구조를 가지고 있지만,
하나의 완전 연결된 층
추가

	Test loss	Test accuracy
색상 앞	0.350228	0.893303
색상 뒤	0.374071	0.896127

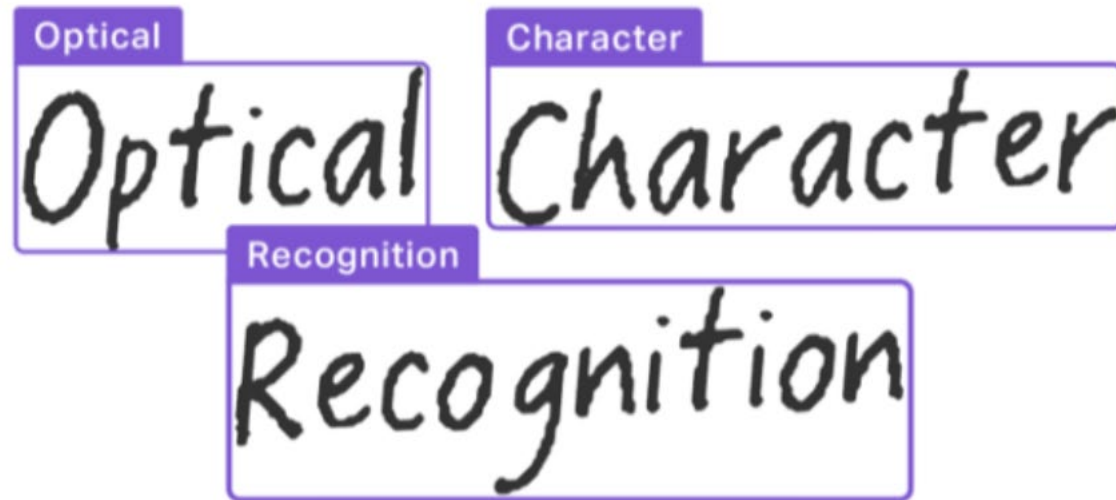
Hyper Parameter

cnn_color

```
# Build model
def build(input_shape, classes):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Conv2D(64, (5, 5), activation='relu'))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(512, activation='relu'))
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(classes, activation='softmax'))
    return model
```



OCR



OCR(Optical Character Recognition:광학 문자 인식)은 스캔한 이미지나 사진에서 문자를 인식하여 컴퓨터에서 사용 가능한 텍스트로 변환하는 기술입니다

NN Model

CNN < CRNN
CNN + RNN!

Open Source OCR

EasyOCR

인식률

중간

속도

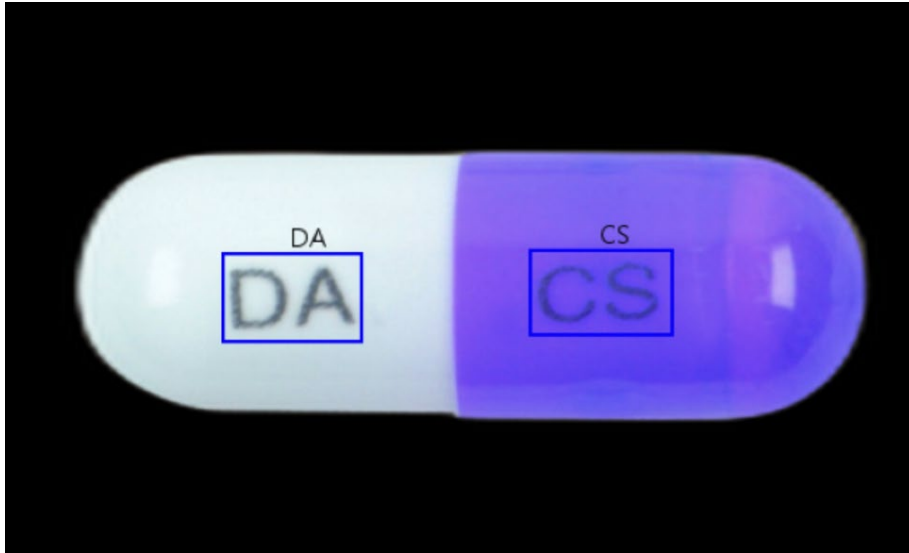
느림

Tesseract

낮음

빠름

Limitation



알약에 음각 되어있는 문자는
인식하지 못할 확률이 높음

오픈 소스 모델은
선명한 글자는 인식하지만

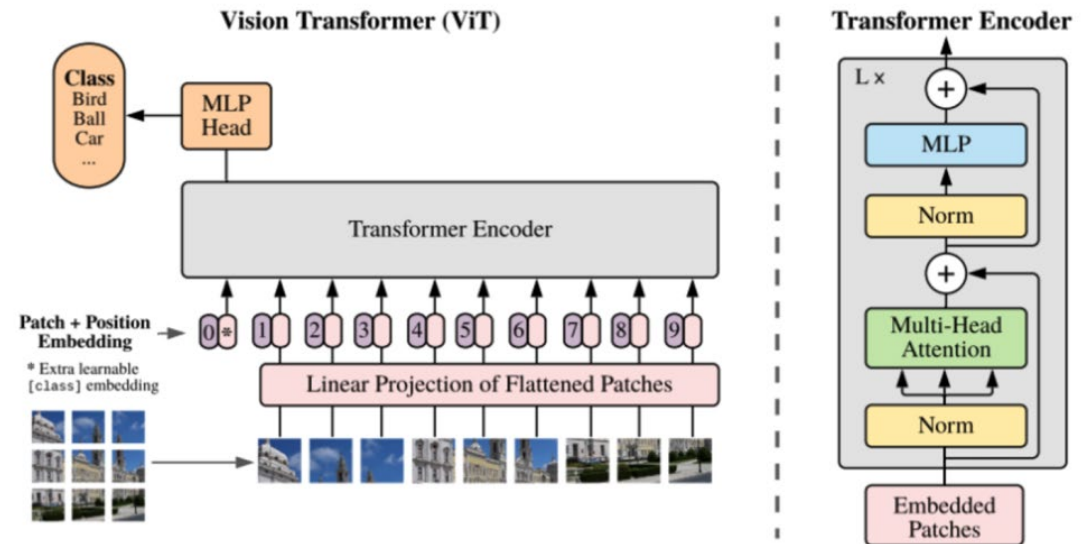


Vision Transformer

기존의 트랜스포머 모델을
문자를 넘어 이미지로 확대시킨 모델

이미지 작업에서 CNN이나 CRNN을
사용하지 않고, 셀프 어텐션 구조를
사용하여 이미지를 학습 가능

토큰이 아닌 Patch로 이미지를 나눠서
벡터화 시킴.



Vision Transformer

나



Tell me the letters on this pill



ChatGPT

The letters on the pill are "ASCOUGH KVP".

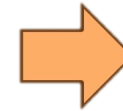
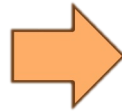
Vision Transformer (ViT)를 탑재한 ChatGPT 에게 이미지를 보여주면 기존에 있던 OCR보다 인식률이 높은것을 볼 수 있다.

종합 모델

Function



UPLOAD



Model Structure

```
def resize_image(image_path, size=(70, 130)):
    image = cv2.imread(image_path)
    resized_image = cv2.resize(image, size)
    return resized_image

def remove_background(image_path):
    input_image = cv2.imread(image_path)
    output_image = remove(input_image)
    cv2.imwrite("uploads/no_bg.png", output_image)
    return "uploads/no_bg.png"
```

이미지 사이즈를 조정하고, 배경을 제거함

```
def predict_shape(image_path):
    model = tf.keras.models.load_model(SHAPE_MODEL_PATH)
    image = resize_image(image_path)
    image = image / 255.0
    predictions = model.predict(tf.expand_dims(image, 0))
    return predictions.argmax(axis=1)[0]
```

앞면과 뒷면의 형태를 각각 학습 모델로 예측함

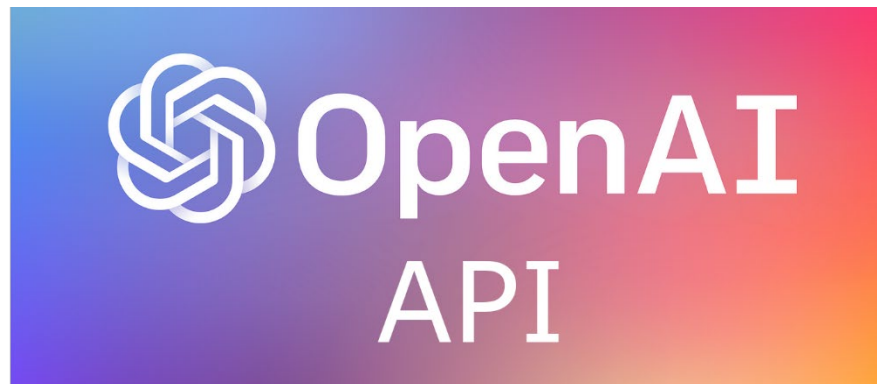
```
def predict_color(image_path):
    model = tf.keras.models.load_model(COLOR_MODEL_PATH)
    image = resize_image(image_path)
    image = image / 255.0
    predictions = model.predict(tf.expand_dims(image, 0))
    return predictions.argsort(axis=1)[0][::-1][:3]
```

색깔을 학습 모델로 예측하고, 예측률이 높은 3개를 뽑음

Model Structure

```
def compare_shapes_and_colors(shape1, shape2, colors1, colors2):  
    if shape1 != shape2:  
        return False, "Shape mismatch."  
    if colors1[0] != colors2[0]:  
        if colors1[0] not in colors2[:3] and colors2[0] not in colors1[:3]:  
            return False, "Color mismatch."  
    return True, ""
```

앞면과 뒷면의 형태를 예측한 결과가 서로 맞는지 여부와
두 면의 색상을 예측한 결과가 둘 다 3위 내에 있는지 여부를 검사함



형태와 색상이 앞면이 일치하면,
OpenAI API의 GPT-4-Turbo 모델을 사용하여 문자를 인식함

Model Structure

Method	Language	Precision ↑
GPT-4V	Arabic	16.44%
	English	86.57%
	French	83.0%
	Chinese	1.2%
	German	73.65%
	Korean	10.83%
	Japanese	11.9%
	Italian	62.7%
	Bangla	2.53%
	Hindi	7.29%

하지만 GPT-4도 100%에 가까운
OCR 정확도를 보여주진 못함

Model Structure

“존재성과 유일성 가정”

1. 국내에 유통되는 약은 데이터베이스 안에 존재함
2. 색깔과, 형태, 글자가 일치하는 약은 두개 이상 존재하지 않음



형태가 같은 약의 수



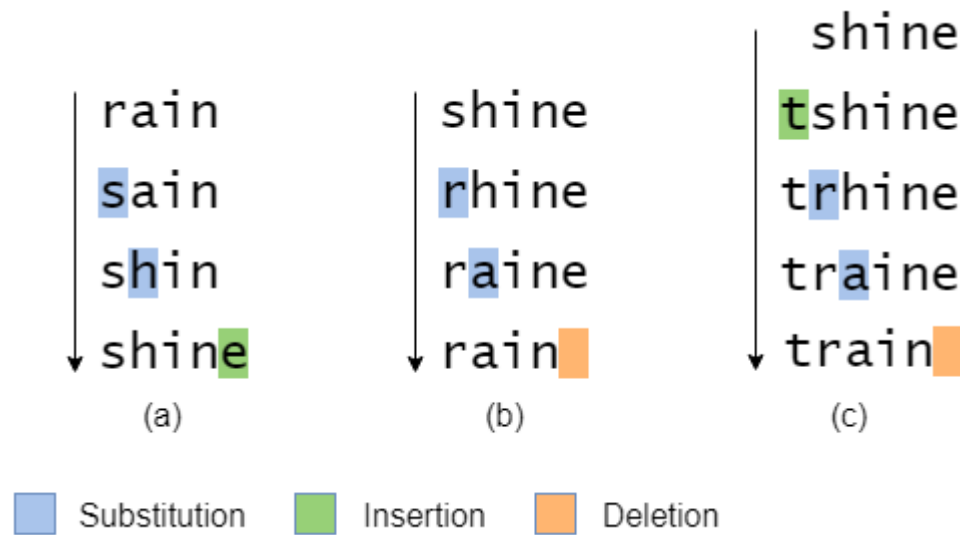
형태와 색깔이 같은 약의 수



형태, 색깔,
글자가 같은 약의 수

Model Structure

레벤슈타인 거리 (Levenshtein Distance)



문자열의 유사도를 가리기 위한 알고리즘 중의 하나

Model Structure

```
def identify_pills(front_image_path, back_image_path):
    process_info = {}
    for label, image_path in [('front', front_image_path), ('back', back_image_path)]:
        no_bg = remove_background(image_path)
        shape = predict_shape(no_bg)
        colors = predict_color(no_bg)
        text = extract_text(no_bg)
        process_info[label] = (shape, colors, text if text is not None else "")

    text_matches = []
    for front_label, back_label in [('front', 'back'), ('back', 'front')]:
        text_front, text_back = process_info[front_label][2], process_info[back_label][2]
        df_filtered = df.assign(
            front_similarity=df.apply(lambda row: calculate_similarity(text_front, row['front']), axis=1),
            back_similarity=df.apply(lambda row: calculate_similarity(text_back, row['back']), axis=1)
        )
        text_matches.extend(df_filtered[
            (df_filtered['front_similarity'] >= 0.9) & (df_filtered['back_similarity'] >= 0.9)
        ][['id']].tolist())

    if text_matches:
        return text_matches[:5]
    else:
        all_matches = []
        selected_ids = set()
        for front_label, back_label in [('front', 'back'), ('back', 'front')]:
            shape_front, colors_front, _ = process_info[front_label]
            shape_back, colors_back, _ = process_info[back_label]
            text_front, text_back = process_info[front_label][2], process_info[back_label][2]

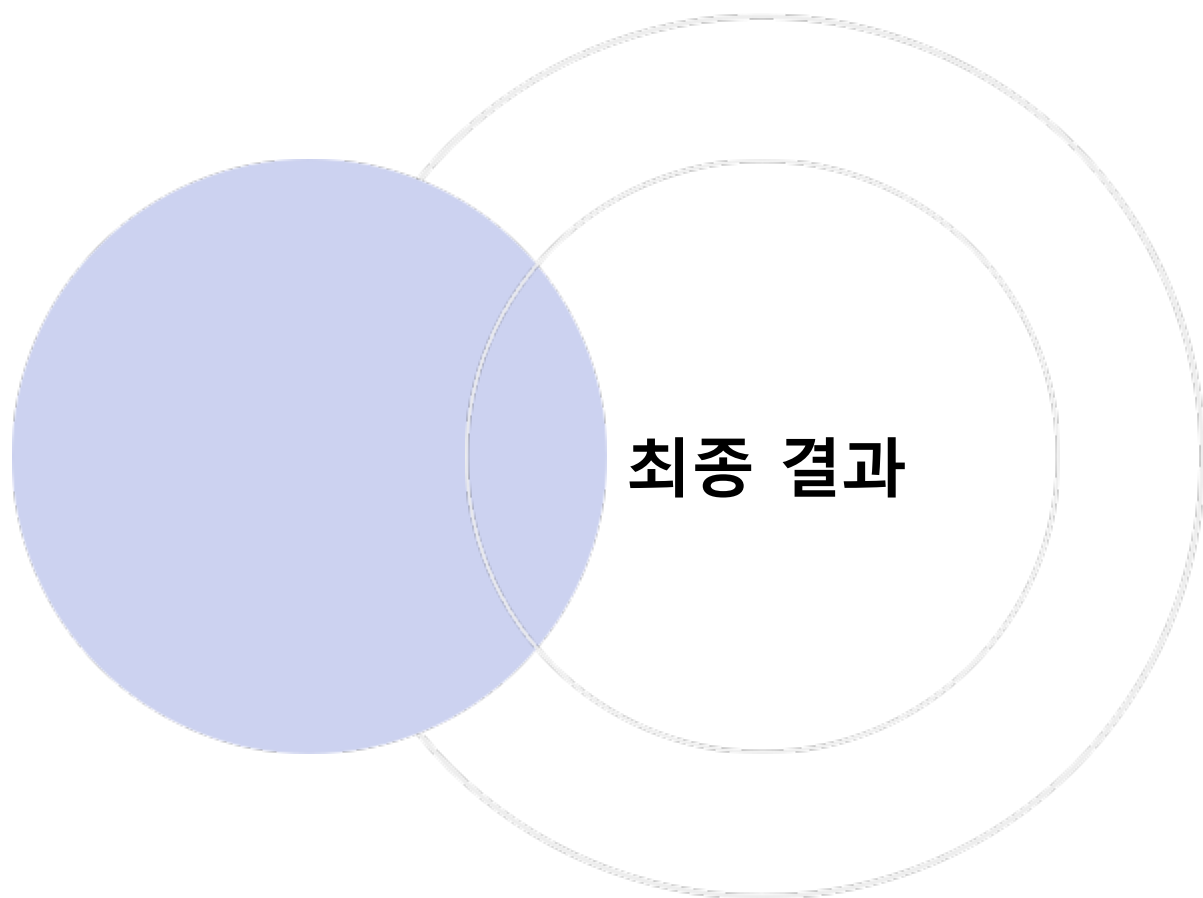
            df_filtered = df.assign(
                front_similarity=df.apply(lambda row: calculate_similarity(text_front, row['front']), axis=1),
                back_similarity=df.apply(lambda row: calculate_similarity(text_back, row['back']), axis=1),
                color_similarity=df.apply(lambda row: 1 if any(color in [str(row['color'])] for color in colors_front) else 0, axis=1),
                shape_similarity=df.apply(lambda row: 1 if row['shape'] == shape_front else 0, axis=1)
            )

            df_filtered['text_similarity'] = (df_filtered['front_similarity'] + df_filtered['back_similarity']) / 2
            df_filtered['color_weight'] = 1 - (df_filtered['text_similarity'] - 0.5) / 0.4
            df_filtered['color_weight'] = df_filtered['color_weight'].clip(0, 1)
            df_filtered['shape_weight'] = 1 - (df_filtered['text_similarity'] - 0.5) / 0.4
            df_filtered['shape_weight'] = df_filtered['shape_weight'].clip(0, 1)

            df_filtered['total_similarity'] = (
                0.5 * df_filtered['text_similarity'] +
                0.1 * df_filtered['color_weight'] * df_filtered['color_similarity'] +
                0.4 * df_filtered['shape_weight'] * df_filtered['shape_similarity']
            )
```

정답데이터의 존재성을 가정하면,
OCR이 내놓은 오답도 정답데이터와 유사도를 구할 수 있다.

색상과 형태를 결정짓고, 남은 선택지중에서 가장 유사한 답을 골라냄.



최종 결과

모의 서비스 구축

모바일에 최적화된 형태로
구현하고자 하였음.

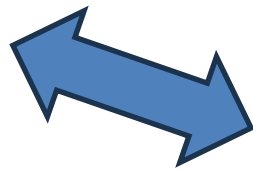
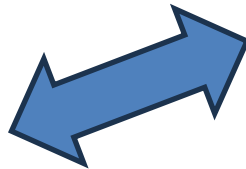


모의 서비스 구축



Flask

유저



비전 인식 모델
OCR API

모의 서비스 구축


촬영가이드

1. 앞면과 뒷면을 찍어주세요


2. 순서는 상관 없습니다.

3. 빛이 반사되지 않아야 잘 인식합니다.

<앞면>



<뒷면>



약품 앞면 사진을 찍거나 저장소에서 선택해주세요!

파일 선택

선택한 파일 없음


촬영가이드

1. 앞면과 뒷면을 찍어주세요


2. 순서는 상관 없습니다.

3. 빛이 반사되지 않아야 잘 인식합니다.

<앞면>



<뒷면>



약품 뒷면 사진을 찍거나 저장소에서 선택해주세요!

파일 선택

선택한 파일 없음

알약 식별

약품 앞면 사진을 찍거나 저장소에서 선택해주세요!

파일 선택


선택한 파일 없음

사진 보관함


사진 찍기

파일 선택

<앞면>



<뒷면>



이미지를 업로드하면, 그걸 서버로 보냄

모의 서비스 구축



인식이 불명확할 경우,
유사도가 높은 5개의 선택지를 출력하여
유저가 선택할 수 있게 함

모의 서비스 구축

사진과 함께

효능, 용법, 주의사항 제공



약품 확인: 이자벨타정150밀리그램(이르베사르탄)

효능 효과: 1. 본태고혈압 2. 고혈압 치료요법으로서, 고혈압을 가진 제 2형 당뇨병 환자의 신장병

용법 용량: 이 약은 식사와 관계없이 복용할 수 있다. 1. 본태고혈압 2. 고혈압을 가진 제 2형 당뇨병 환자의 신장병

주의 사항: 1. 경고 2. 다음 환자에는 투여하지 말 것. 3. 다음 환자에는 신중히 투여할 것. 4. 이상반응 5. 일반적 주의 6. 상호작용 7. 임부 및 수유부에 대한 투여 8. 소아에 대한 투여 9. 고령자에 대한 투여 10. 과량 투여시의 처치 11. 보관 및 취급

크롤링

의약품상세정보



니박스정(프레드니솔론)(수출용)



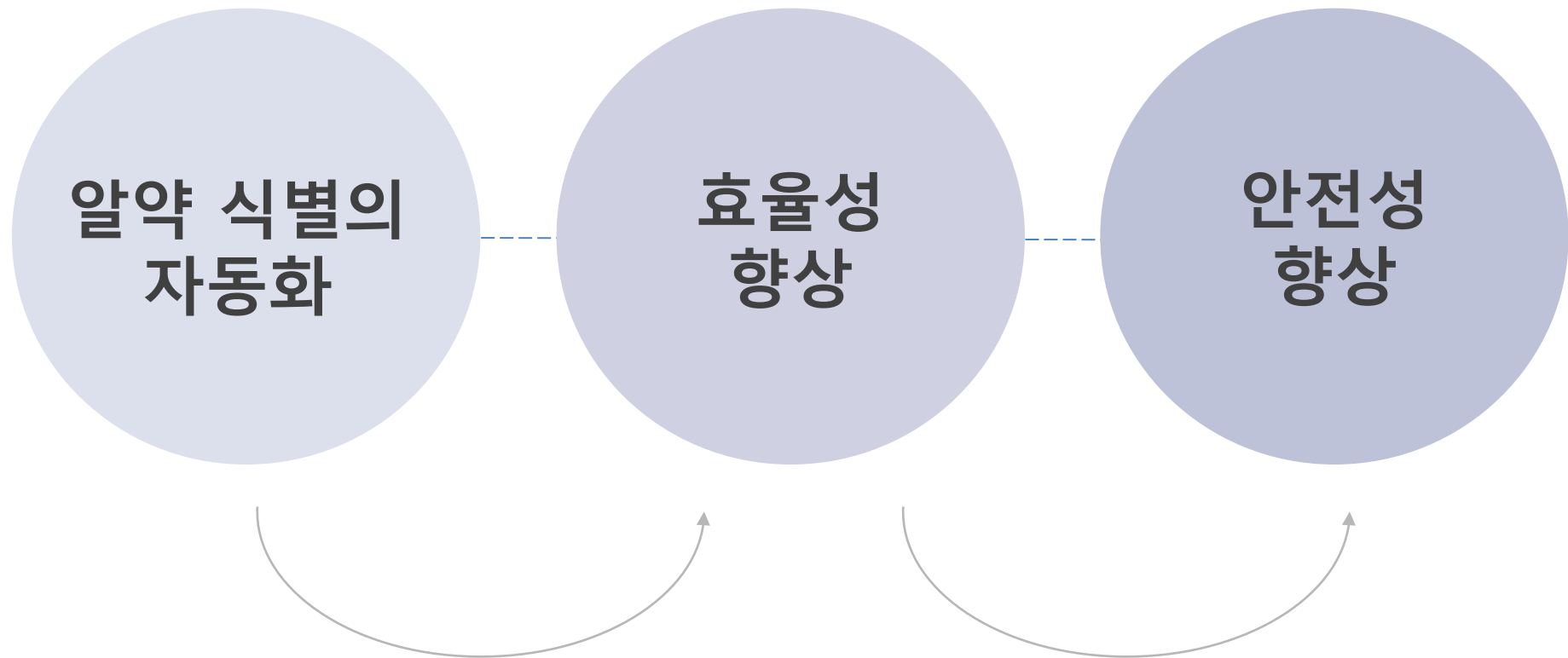
기본정보

제품명	니박스정(프레드니솔론)(수출용)
성상	백색의 원형정제
모양	원형
업체명	국제약품(주)
위탁제조업체	
전문/일반	전문의약품
허가일	1963-03-11
품목기준코드	196300019
취소/취하구분	취하
취소/취하일자	2015-01-22
표준코드	8806437009300, 8806437009317, 8806437009324, 8806437009331, 8806437009348, 8806437009355
기타식별표시	식별표시 : KJ040024 장축크기 : 8.6mm 단축크기 : 8.6mm 두께 : 2.6mm 분할선(앞) : -
첨부분서	첨부분서다운로드 *첨부분서는 수집된 자료로 허가사항과 다를 수 있습니다.
제조업체/제조소	국제약품(주) 안산 제 1공장 목록품목



효능, 용법, 주의사항은 의약품 안전나라에서 크롤링 하였음

의의 및 기대 효과

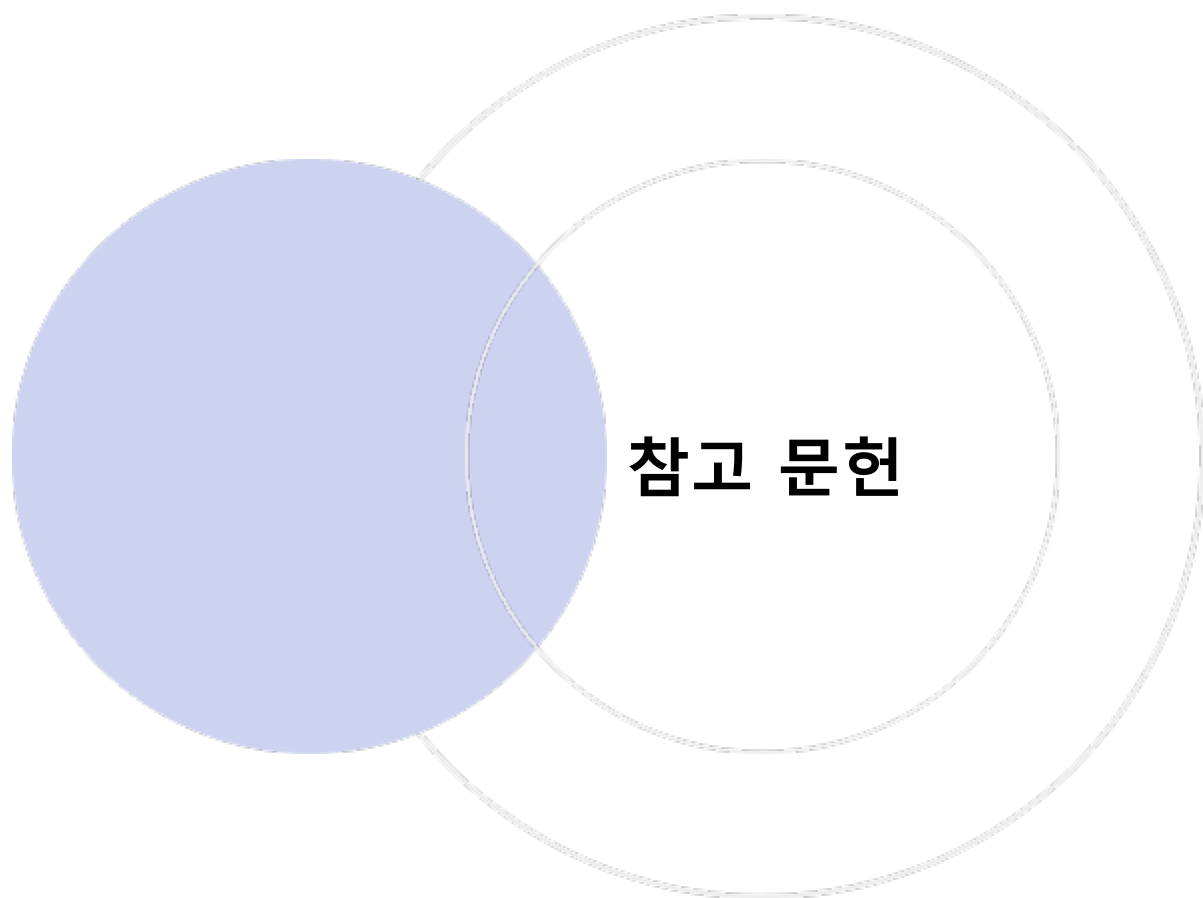


고찰 및 개선점

아쉽게도 시간적 제약으로 인해 직접 OCR을 구축하지 못했다. 이미지 전처리 작업을 시도했지만 실패했는데, 효과적인 전처리 작업은 OCR의 정확도를 높일 수 있었을 것이다.

의료 관련 데이터는 상용화를 위해 거의 100%에 가까운 정확도가 필요하지만, 이번 프로젝트의 결과는 그에 미치지 못했다. 향후 보완이 필요한 부분이 있는 것으로 보인다.

정답데이터의 유일성을 가정하면, 정답율이 낮은 OCR로 학습하여 앙상블 모델을 구축한다면 유사도를 통해 정답율을 올릴 수 있을 것으로 판단되었으나, 실행하지는 못했다.



VSCode 원격 접속
[VSCode SSH 원격 접속하기 \(velog.io\)](https://velog.io)

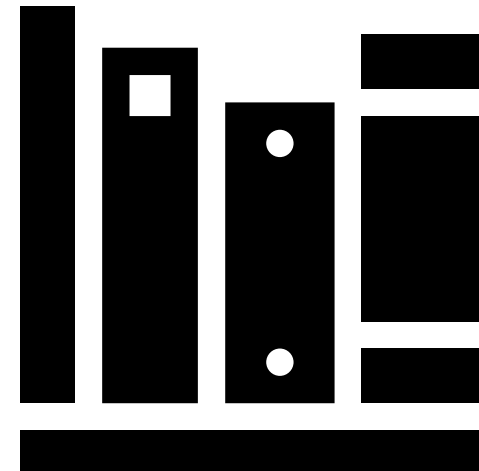
matplotlib 색상
<https://wikidocs.net/92085>

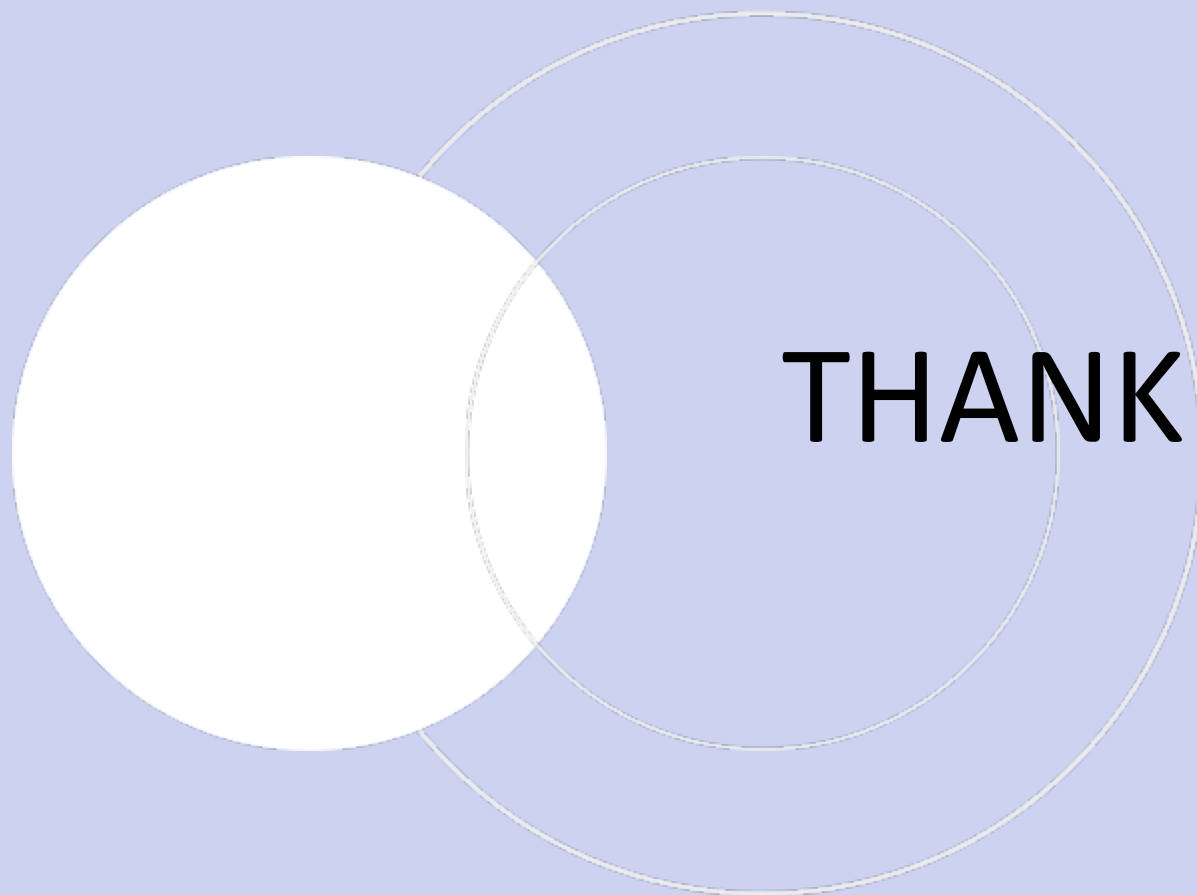
이미지 파일 복사 코드
<https://appia.tistory.com/696>
<https://seong6496.tistory.com/100>

CNN 하이퍼파라미터
<https://sonsnotation.blogspot.com/2020/11/9-2-cnn-best-architecture.html>

파이썬 파일 복사
[파이썬\[Python\] 파일 복사하기\(Copy file\) \(tistory.com\)](https://tistory.com)

폴더 안 파일 이름 목록 가져오기
[\[Python\]폴더 안에 파일 이름 목록 가져오기 \(tistory.com\)](https://tistory.com)





THANK YOU! —————