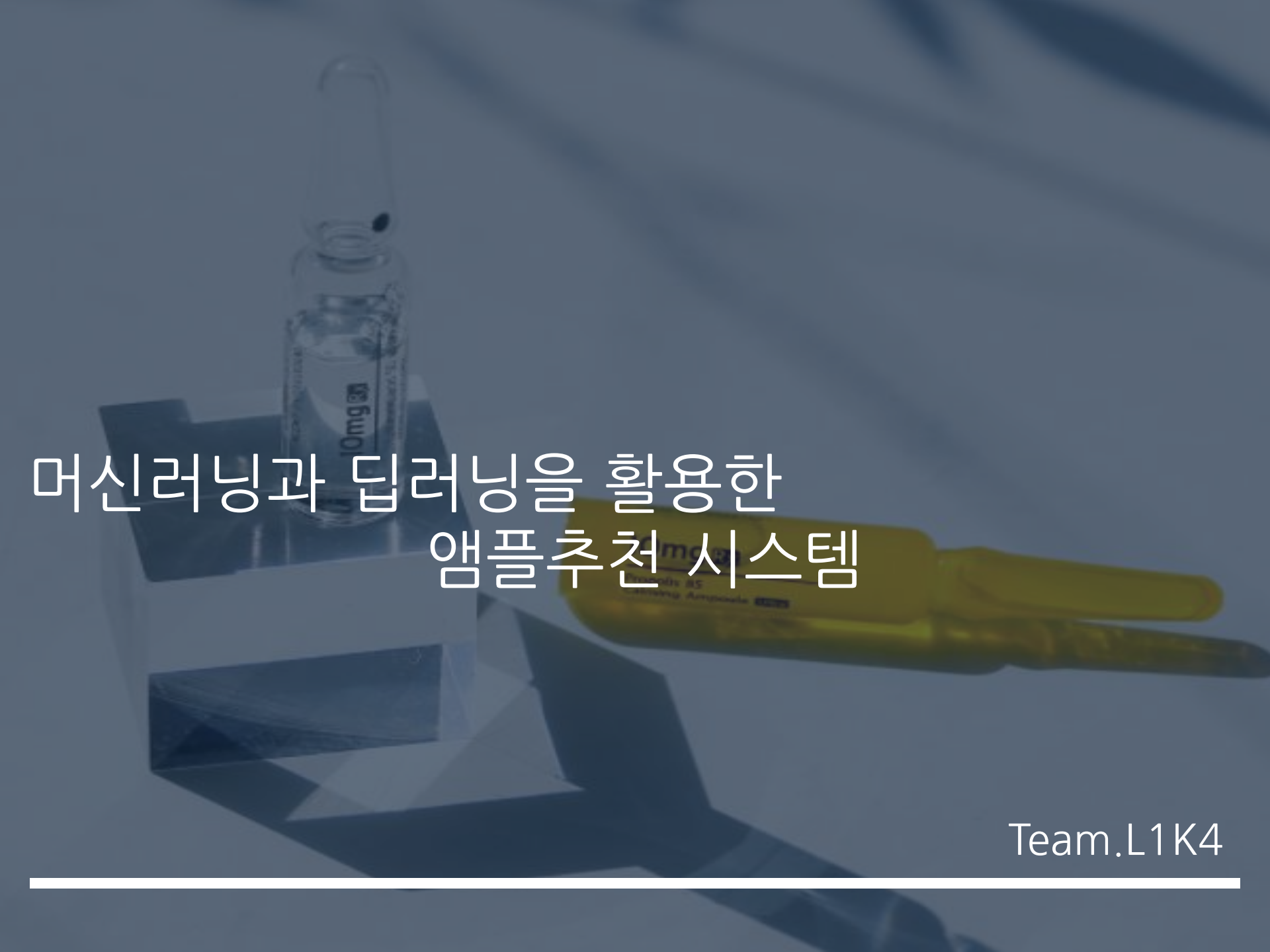


머신러닝과 딥러닝을 활용한 앰플추천 시스템

Team.L1K4

목차 CONTENTS

01.

주제 선정 및 기획 의도

사람마다 피부타입 및 고민이 각각 다르며 제품 선택이 상당히 까다롭다.

건조한 피부에 히알루론산원액 앰플 추천... 2021.03.19.
... 앰플 추천 좀 해주세요 피부속이 너무 건조하셔서 고민이시군요. 질문자님께 히알루론산앰플로 불릴만큼 효과가 좋은 토너를 하나 추천해드리고 싶어서 답변드렸습니다....
#기초화장품 #히알루론산앰플 #히알루론산토너 #킨베인히알루론산토너 #피부과화장품 #재생토너 #PDRN #베타글루칸
Q&A > 기초화장품 | 답변수 8 · 추천수 0

화농성 여드름 및 여드름 앰플 추천 2021.03.14.
모낭염 여드름입니다 앰플 추천 부탁드려요 안녕하세요~ 여드름성 피부 전용 앰플 추천드릴게요! 바버라고 독일 화장품으로 앰플로 유명한 브랜드입니다~! 그 중 여드름성...
#앰플추천 #여드름 #모낭염 #바버앰플 #티트리앰플 #여드름앰플
Q&A > 피부과 | 답변수 10 · 추천수 0

수부지 앰플 추천 토리든, 더랩바이블랑두 2021.02.16.
... 더랩바이블랑두 앰플 중에 더 유분 적고 수분 많은 화장품 추천해주세요 제발 둘 다 써보신 분만 알려주세요 다른 화장품 추천 말고 저 두 화장품 중에서만 알려주세여 광고...
#토리든다이브인저분자히알루론산세럼 #더랩바이블랑두앰플 #수부지앰플 #지성앰플 #속건조앰플 #수부지세럼 #지성세럼
Q&A > 기초화장품 | 답변수 2 · 추천수 0

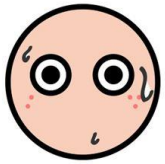
미백앰플 추천 2020.11.28.
미백앰플 효과 좋은것좀요 광고말고 내돈내산으로 효과 많이본거 알려주세여여 제가... 화이트닝 앰플이예요~ 한달 정도 사용하니 효과 있었구 주변에서도 피부 좋아졌다는 말...
Q&A > 피부미백 | 답변수 7 · 추천수 0

기획 의도

주제 선정

aida COSMETIC

겉은 번들번들하고
속당김은 심하고
오돌토돌 트러블 일어나는
예민피부



과다한 피지분비로
번들거리고
트러블,모공으로 고민하는
지성피부

뭘 발라도
각질이 심하고
나무껍질같이 거칠거칠한
건조피부

02.

쇼핑몰리뷰 및 인스타 크롤링

Python 이용

파이썬에서 제공하는 Selenium라이브러리와 BeautifulSoup 라이브러리를 이용하여 크롤링

NAVER

innisfree

danawa

OLIVE YOUNG

쇼핑몰 활용

코드 예시 - 크롤링

```
def crawling(url, name, n):
    driver = webdriver.Chrome("C:/Users/rhwns/Downloads/chromedriver.exe")
    driver.get(url)
    time.sleep(1)

    text = []
    point = []
    date = []
    star1 = []
    date_list1 = []

    k = 0
    for j in range(n):
        star1 = []
        date_list1 = []
        review = driver.find_elements_by_css_selector("div.atc")

        for num in range(10):
            # 별점 긁어오기
            star = driver.find_elements_by_xpath("""//*[@id="danawa-prodBlog-companyReview-content-wrap-{}"]/div[1]/span[1]/span""".format(num))
            star1.append(star)

        for num in range(10):
            # 날짜 긁어오기
            date_list = driver.find_elements_by_xpath("/html/body/div[2]/div[3]/div[2]/div[4]/div[4]/div[2]/div[2]/div[2]/div/div[3]/div[2]/ul/li[{}]/div[1]/span[2]".format(num+1))
            date_list1.append(date_list)

        for x, _ in enumerate(review, 1):
            text.append(_.text)
        for x, _ in enumerate(reduce(operator.add, star1), 1):
            point.append(_.text)
        for x, _ in enumerate(reduce(operator.add, date_list1), 1):
            date.append(_.text)
```

코드 예시 - 크롤링

```
    try:
        time.sleep(1)
        page_num = """/html/body/div[2]/div[3]/div[2]/div[4]/div[4]/div[2]/div[2]/div[2]/div/div[3]/div[2]/div[5]/div/div/a[{}]""".format(k+1)
        print(page_num)
        driver.find_element_by_xpath(page_num).click()
        time.sleep(1)
    except:
        # 다음버튼 누르기
        try:
            if(j < 10):
                driver.find_element_by_xpath("""/html/body/div[2]/div[3]/div[2]/div[4]/div[4]/div[2]/div[2]/div[2]/div/div[3]/div[2]/div[5]/div/a/span""").click()
                time.sleep(1)
                k=-1
            else:
                driver.find_element_by_xpath("""/html/body/div[2]/div[3]/div[2]/div[4]/div[4]/div[2]/div[2]/div[2]/div/div[3]/div[2]/div[5]/div/a[2]/span""").click()
                time.sleep(1)
                k=-1
        except:
            break

    k += 1
# 데이터프레임 저장
data = pd.DataFrame(text[:])
data1 = pd.DataFrame(point[:])
data2 = pd.DataFrame(date[:])
# csv로 저장
data.to_csv("danawha/review({}).csv.".format(name),encoding='utf-8-sig')
data1.to_csv("danawha/star({}).csv".format(name),encoding='utf-8-sig')
data2.to_csv("danawha/date({}).csv".format(name),encoding='utf-8-sig')

merge_save(name)
```

코드 예시 - 크롤링

```
def merge_save(name):
    a=pd.read_csv("danawha/review({}).csv".format(name))
    b=pd.read_csv("danawha/star({}).csv".format(name))
    c=pd.read_csv("danawha/date({}).csv".format(name))

    result = pd.concat([a, b, c],axis=1)

    result.columns = ['a', 'reviews', 'b', 'starpoint', 'c', 'date']
    del result['a']
    del result['b']
    del result['c']

    result.to_csv("다나와_완/{}.csv".format(name),encoding='utf-8-sig')
```

코드 - 형태소

```
def nouns(file):
    global text1, tokens, tokens1, stop_words, ingre_info
    df = pd.read_csv("data/{}.csv".format(file))

    # 평점 3이하 제거
    idx = df[df['point']<=3].index
    df = df.drop(idx)

    # 중복 제거
    df.dropna(inplace=True)
    df.drop_duplicates(['review'], keep = 'first', ignore_index=True, inplace=True)

    # date열 제거 (불필요하다고 판단)
    del df['date']

    # Unnamed: 0열 제거
    del df['Unnamed: 0']
    import nltk
    from konlpy.tag import Okt
    Okt=Okt()

    # 명사 추출 후 tokens에 저장
    df["noun"] = df["review"].apply(Okt.nouns)

    tokens = [take2 for take1 in df["noun"] for take2 in take1]
    text = nltk.Text(tokens, name="NMSC")
```

코드 - 형태소

```
# 불용어 처리 후 text1에 다시 저장
stop_words = ['피부', '좋아요', '앰플', '이니스프리', '제품', '구매', '사용', '것', '때', '거', '더', '용량', '일',
              '진짜', '저', '배송', '통째', '좀', '또',
              '임', '이', '항상', '보고', '아주', '정말', '정도', '구입', '통', '생각', '용', '형', '주문', '제', '템',
              '번', '전', '때문', '바로', '후', '요즘', '다음', '벌써', '날', '일단', '늘', '걸', '상품', '다른', '쟁',
              '끈', '감', '금방', '안', '중', '사서', '요']

tokens1 = [each_word for each_word in tokens
           if each_word not in stop_words]
text1 = nltk.Text(tokens1, name="NMSC")
count = Counter(text1.vocab())

# 빈도 저장 후 출력
ingre_info = dict()
for tags, counts in text1.vocab().most_common(30):
    if (len(str(tags)) > 1):
        ingre_info[tags] = counts
        print("%s:%d" %(tags, counts))
```

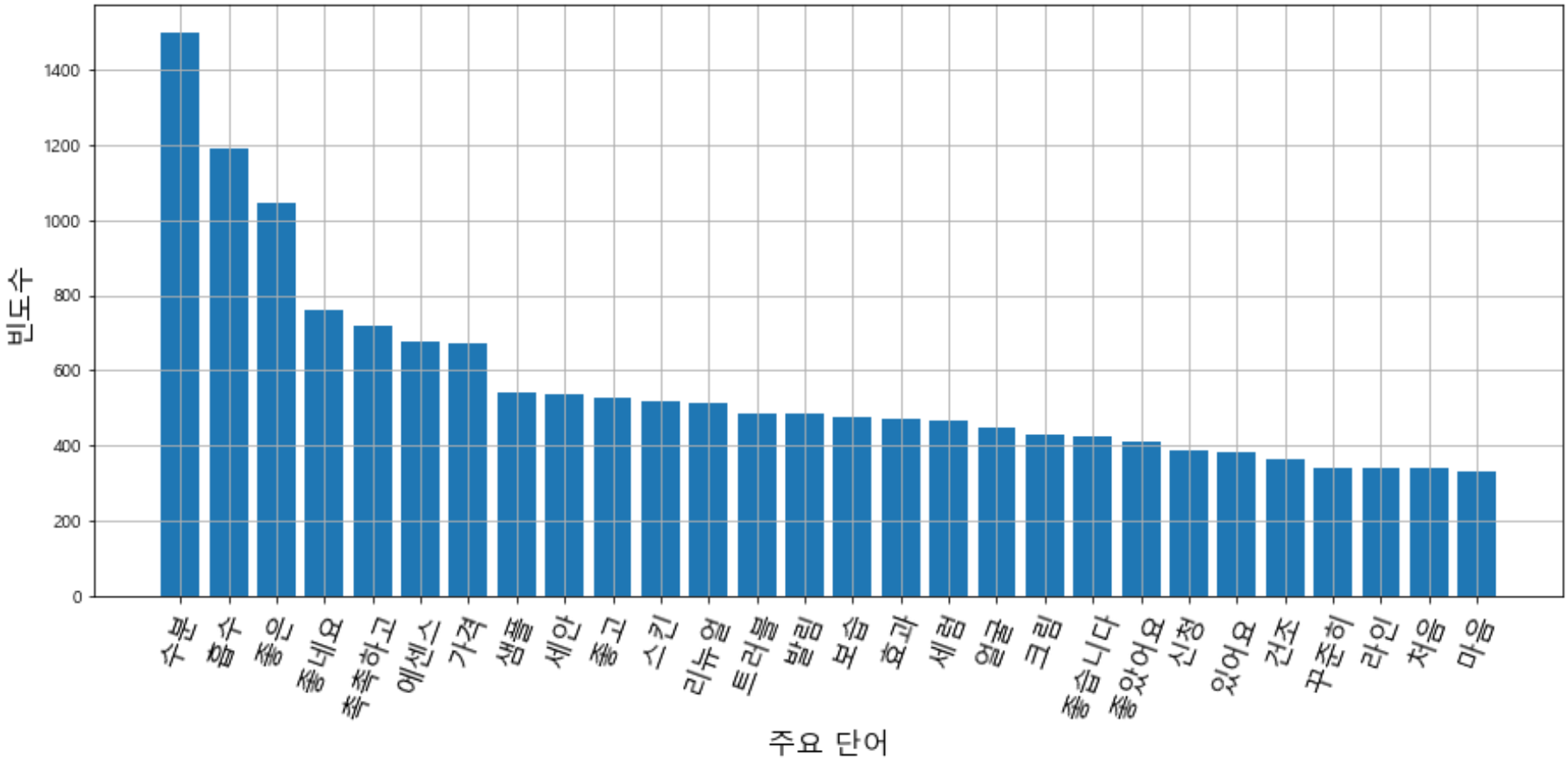
코드 - 그래프

```
def showGraph(wordInfo):

    font_location = "c:/Windows/Fonts/malgun.ttf"
    font_name = fm.FontProperties(fname=font_location).get_name()
    rc('font', family=font_name)
    plt.figure(figsize=(15, 6))
    plt.xticks(fontsize=16)
    plt.rc('axes', labelsize=16)
    plt.xlabel('주요 단어')
    plt.ylabel('빈도수')
    plt.grid(True)

    Sorted_Dict_Values = sorted(wordInfo.values(), reverse=True)
    Sorted_Dict_Keys = sorted(wordInfo, key=wordInfo.get, reverse=True)

    plt.bar(range(len(wordInfo)), Sorted_Dict_Values, align='center')
    plt.xticks(range(len(wordInfo)), list(Sorted_Dict_Keys), rotation='70')

    plt.show()
```

코드 - 워드클라우드

```
def wordcloud(Text):
    data = Text.vocab().most_common(100)

    wordcloud = WordCloud(font_path="C:/Windows/Fonts/malgun.ttf",
                          relative_scaling = 0.2,
                          #stopwords=STOPWORDS,
                          background_color='white',
                          ).generate_from_frequencies(dict(data))
    plt.figure(figsize=(12, 8))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
```

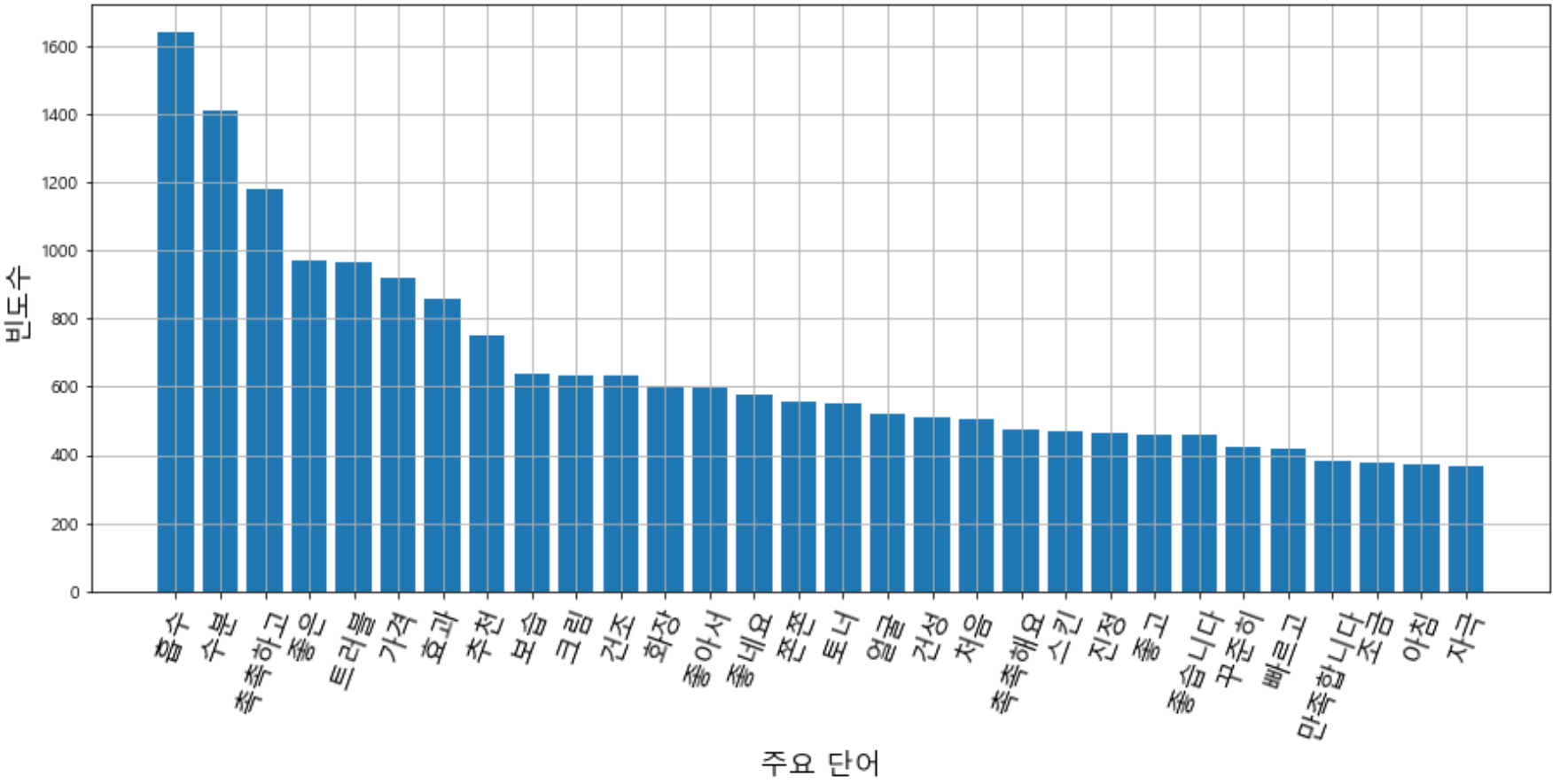
갈락토미세스


빈도수
0
200
400
600
800
1000
1200
1400
주요 단어
수분
좋
좋은
좋네요
촉촉하고
에센스
가격
샘플
세안
좋고
스킨
리뉴얼
트러블
발림
보습
효과
세럼
얼굴
크림
습니다
있어요
있어요
신청
있어요
건조
꾸준히
라인
처음
마음

갈락토미세스

세안
촉촉하고
좋네요
향
수분
보습
발림
리뉴얼
에센스
흡수
가격
좋은
스킨
크림
얼굴
마스크
아침
꾸준히
있어요
빠르고
좋습니다
추천
신청
원래
건성
유지
건조한
처음
약간
화장
타입
업그레이드
좋아서
이벤트
마음
하나
앰플
만족합니다
확실히
라인
기대
건조
조금
완전
촉촉함이
기분
새로
자극
케어
겨울
진정
최고
뜨겁

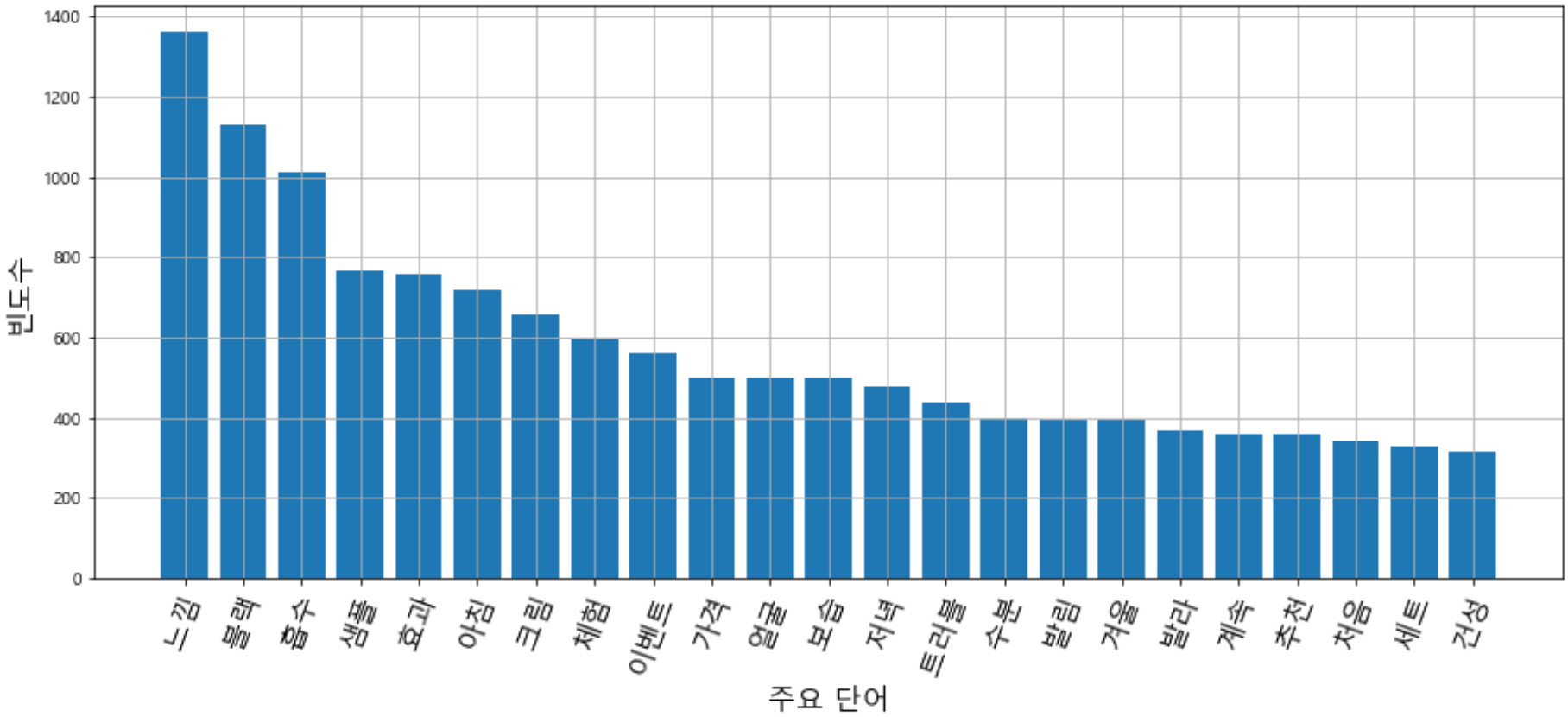
히알루론산


빈도수
0
200
400
600
800
1000
1200
1400
1600
좋아
수분
촉촉하고
좋은
트러블
가격
효과
추천
보습
크림
건조
화장
좋아서
좋네요
쫀쫀
토너
얼굴
건성
처음
촉촉해요
스킨
진정
좋고
좋습니다
꾸준히
빠르고
만족합니다
조금
아침
자극
주요 단어

히알루론산


수분
흡수
좋은
가격
보습
트러블
촉촉하고
효과
크림
토너
건성
화장
좋아서
얼굴
진정
추천
좋네요
쫀쫀
처음
촉촉해요
꾸준히
좋다고
만족합니다
히알루론산
파이
촉촉하니
촉촉해서
에센스
스킨
확실히
성분
좋고
샘플
케어
아침
자극
하나
조금
웰라쥬
최고
약간
겨울
무난
저렴하게
건조한
팩
지금
라인
개
디렉터
발림
후기
저녁
꼭
양
기대
자주
중요한
같은
할인
만족
완전

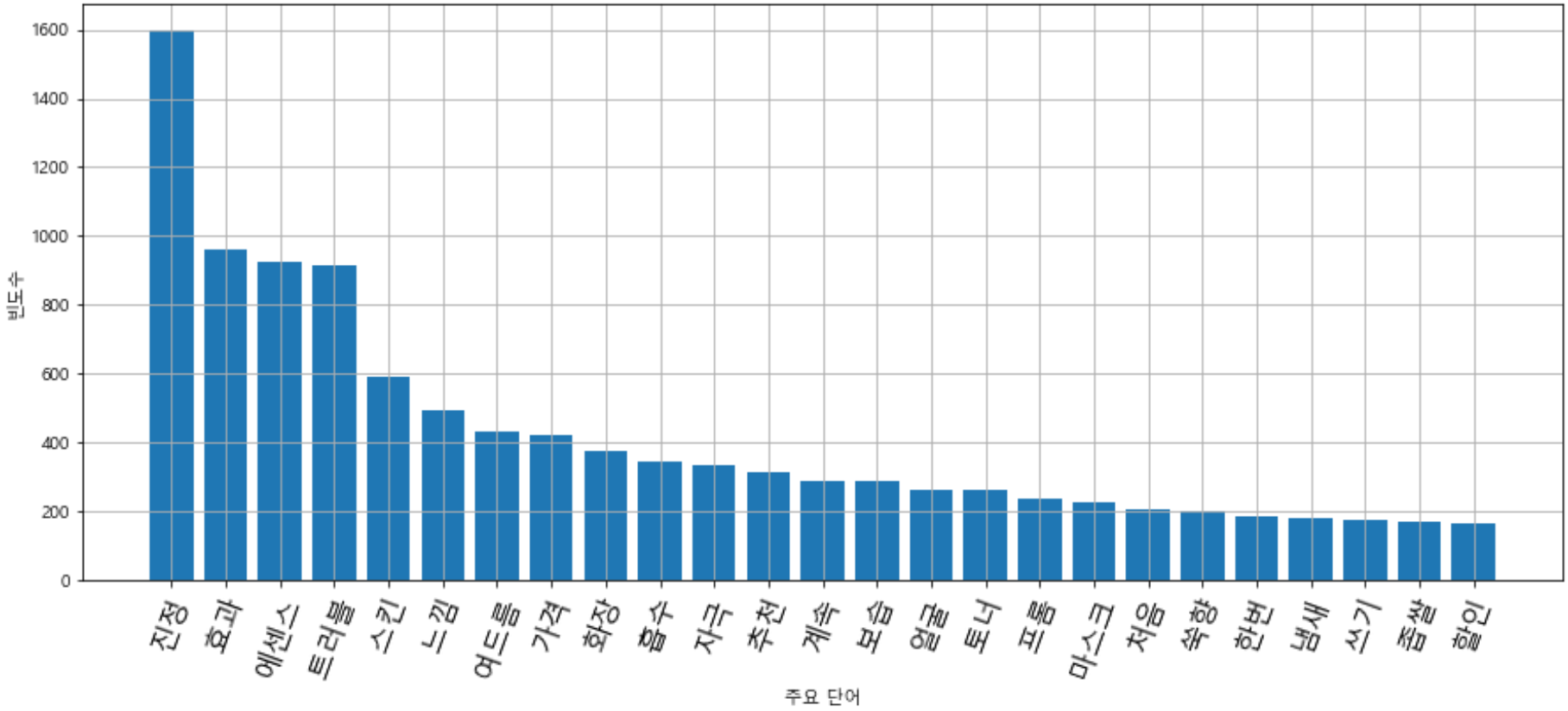
나이아신아마이드


1400
1200
1000
800
600
400
200
0
빈도수
느낌
블랙
흡수
샘플
효과
아침
크림
체험
이벤트
가격
얼굴
보습
저녁
트러블
수분
발림
겨울
발라
계속
추천
처음
세트
건성
주요 단어

나이아신아마이드


블랙
흡수
아침
티
효과
느낌
샘플
이벤트
향
체험
보습
제품
얼굴
저녁
수분
트러블
계속
발림
에센스
자극
스킨
라인
마음
하나
화장품
이번
발라
인핸싱그린티
스포이드
한번
쫀쫀
할인
최고
기대
탄력
완전
케어
의사
선물
자고
마스크
유지
만족
가을
영향
날씨
주문
타입
예정

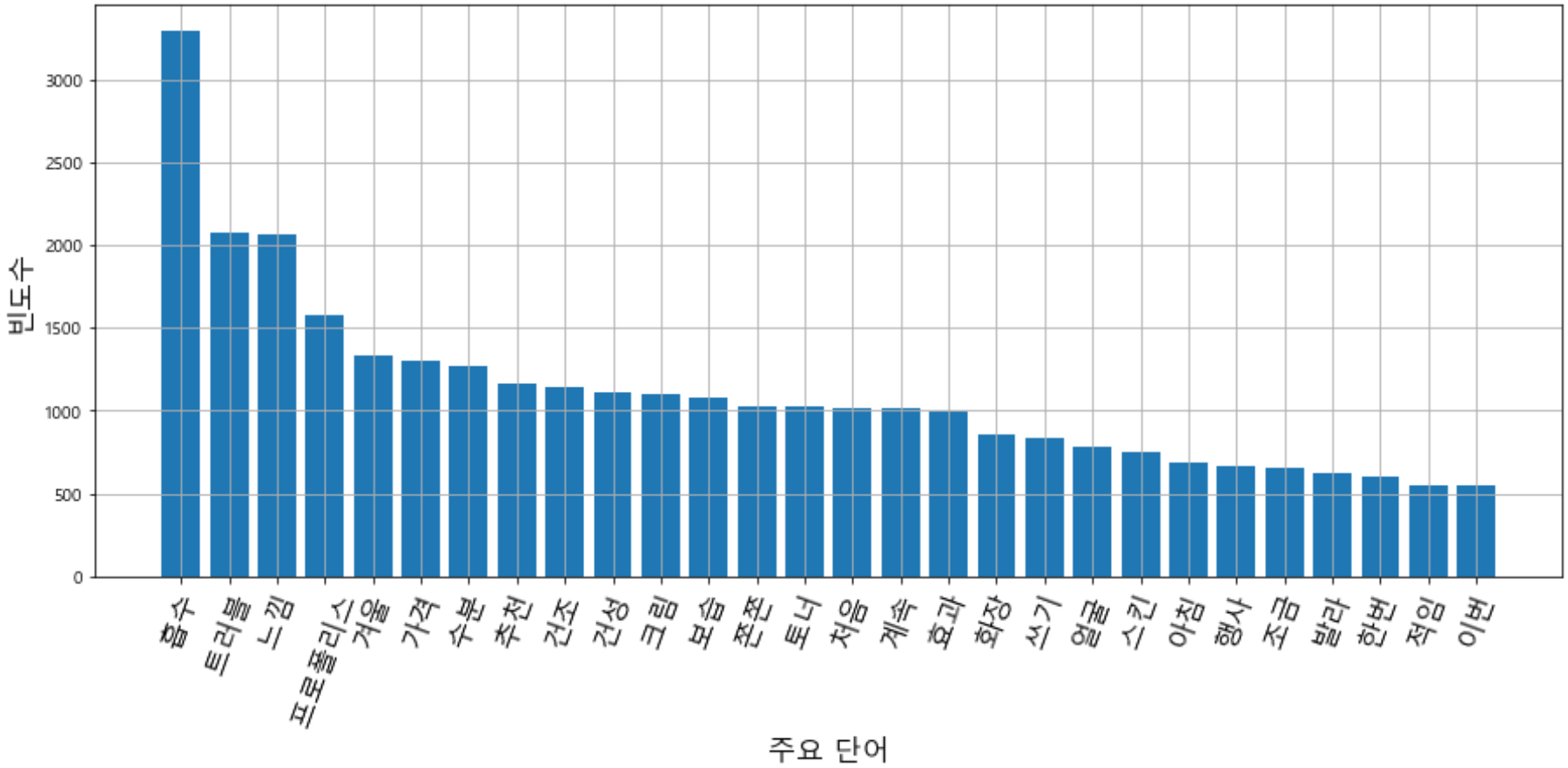
머그워트


빈도수
0
200
400
600
800
1000
1200
1400
1600
진정
효과
에센스
트러블
스킨
느낌
여드름
가격
화장
흡수
자극
추천
계속
보습
얼굴
토너
포름
마스크
처음
속향
한번
냄새
쓰기
좁쌀
할인
주요 단어

머그워트


진정
쑥
효과
에센스
팩
화장
계속
얼굴
추천
토너
향
흡수
피부
보습
처음
냄새
화장품
좁쌀
발라
할인
마스크
그냥
머그
병
아침
유리
쑥향
쓰기
후기
약간
완전
원래
여름
크림
친구
저녁
인생
한번
다시
워트
만족
도움
건성
자주
올리브영
기분
거의
타입
이건
매일
복합
살짝
부담
기간

프로폴리스


빈도수
3000
2500
2000
1500
1000
500
0
흡수
트러블
느낌
프로폴리스
겨울
가격
수분
추천
건조
건성
크림
보습
쫀쫀
토너
처음
계속
효과
화장
쓰기
얼굴
스킨
아침
행사
조금
발라
한번
적임
이번
주요 단어

프로폴리스


프로폴리스
화장
코스
만족
원래
보습
조금
아침
흡수
추천
가격
겨울
효과
스킨
행사
한번
그림
최고
약간
다시
부지
샘플
수분
무난
하나
거리
완전
성분
케어
저녁
자극
진정
발림
여름
역시
할인
대비
마음
의사
민감
이벤트
마무리
살짝

인스타 크롤링 핵심 코드

```
#스크롤링 작업
SCROLL_PAUSE_TIME = 1.0
reallink = []

while True:
    pageString = driver.page_source
    bsObj = BeautifulSoup(pageString, 'lxml')

    for link1 in bsObj.find_all(name='div', attrs={"class":"Nnq7C weEfm"}):
        for i in range(3):
            title = link1.select('a')[i]
            real = title.attrs['href']
            reallink.append(real)

    last_height = driver.execute_script('return document.body.scrollHeight')
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(SCROLL_PAUSE_TIME)
    new_height = driver.execute_script("return document.body.scrollHeight")

    if new_height == last_height:
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(SCROLL_PAUSE_TIME)
        new_height = driver.execute_script("return document.body.scrollHeight")

        if new_height == last_height:
            break
        else:
            last_height = new_height
            continue
    # 게시물의 개수가 계속 바뀌거나 전체 게시물을 가져오지 못한다면
    # 아래 time.sleep의 시간(초)을 늘려주세요.
    time.sleep(5)
```

인스타 크롤링 핵심 코드

```
insta_login("id","pw")
time.sleep(1)
```

```
word = "기능성앰플" #검색할 해시태그, 띄어쓰기를 사용하면 안됩니다
url = insta_searching(word)
driver.get(url)
time.sleep(2)
```

```
# 총 게시물 숫자 불러오기
pageString = driver.page_source #현재 페이지의 htmlcode전체를 변수로에 저장
bsObj = BeautifulSoup(pageString, 'lxml') #BeautifulSoup 사용
temp_data = bsObj.find_all(name='meta')[-1]
temp_data = str(temp_data)
start = temp_data.find('게시물') + 4
end = temp_data.find('개')
total_data = temp_data[start:end]
print("총 {0}개의 게시물이 검색되었습니다.".format(total_data))

"""태그 크롤링"""
print("게시물을 수집하는 중입니다.")
```

```
# 전체 데이터 및 데이터 배치 사이즈 나누기
num_of_data = len(set(reallink))
print("검색된 {0}개의 게시물 중 {1}개의 게시물을 가지고오는데 성공하였습니다.".format(total_data, num_of_data))
print('총 {0}개의 데이터를 수집합니다.'.format(num_of_data))
```

```
검색된 1,993개의 게시물 중 1098개의 게시물을 가지고오는데 성공하였습니다.
총 1098개의 데이터를 수집합니다.
```

인스타 크롤링 핵심 코드

```
#함수를 차례대로 실행합니다.
select_first(driver)
results = [ ]
target = 1095     # 크롤링할 게시글 수

for i in trange(target):
    # 게시글 수집에 오류 발생시 2초 대기후 다음글로 넘어갑니다.
    try:
        data = get_content(driver)    # 게시글 정보 가져오기
        results.append(data)
        move_next(driver)
    except:
        time.sleep(3.5)
        move_next(driver)
```

```
100%|██████████| 1095/1095 [59:12<00:00,  3.24s/it]
```

인스타 크롤링 핵심 코드(전처리)

```
sample = pd.concat([data_01, data_02, data_03, data_04, data_05, data_06, data_07, data_08,
                    data_09, data_10, data_11, data_12, data_13, data_14, data_15, data_16,
                    data_17, data_18, data_19], axis = 0)
```

```
print(len(sample))
```

```
26625
```

```
sample2 =sample.drop_duplicates(["content", "like"], keep=False)
print(len(sample2))
```

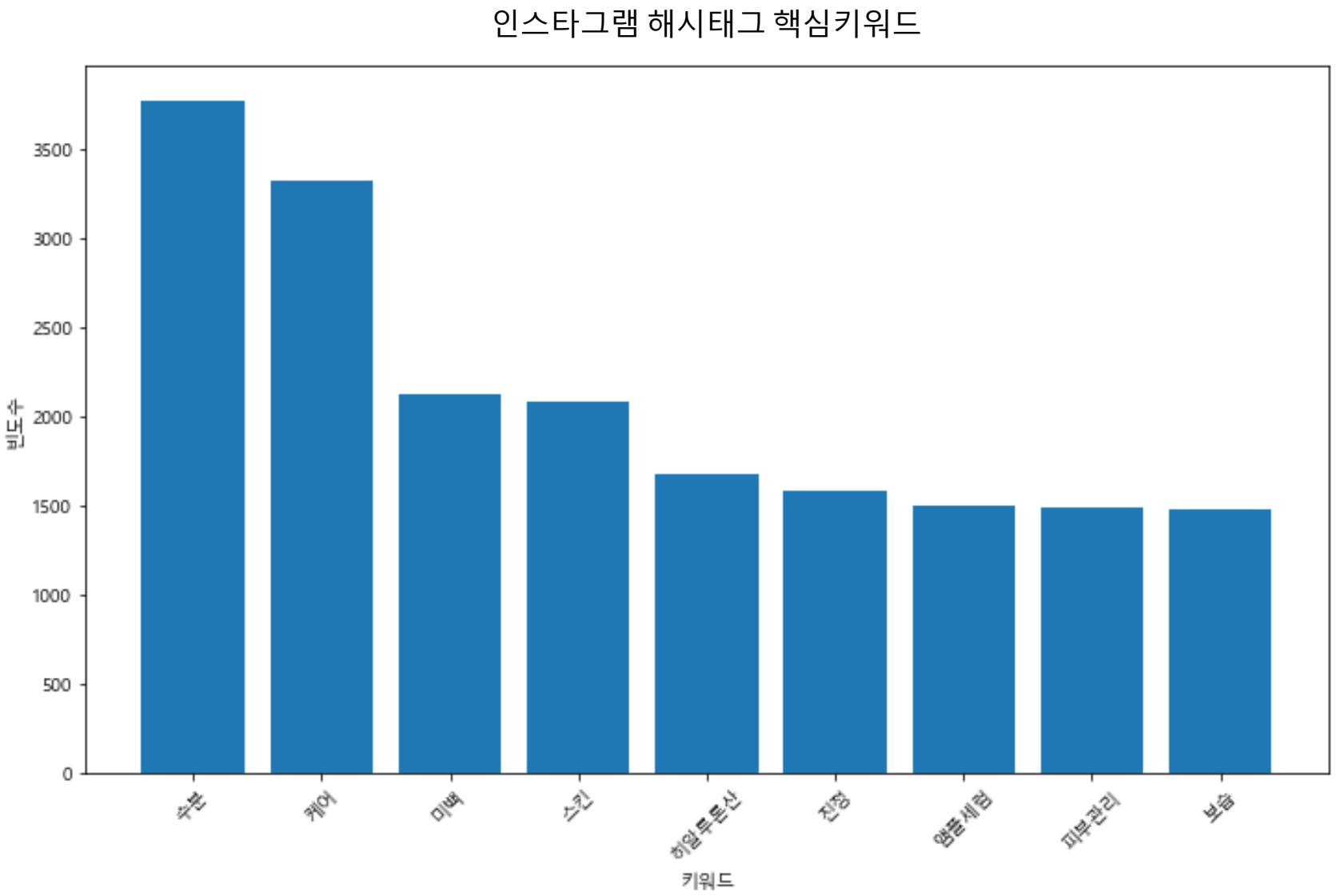
```
13358
```

```
sample2 =sample.drop_duplicates(["content"], keep=False)
print(len(sample2))
```

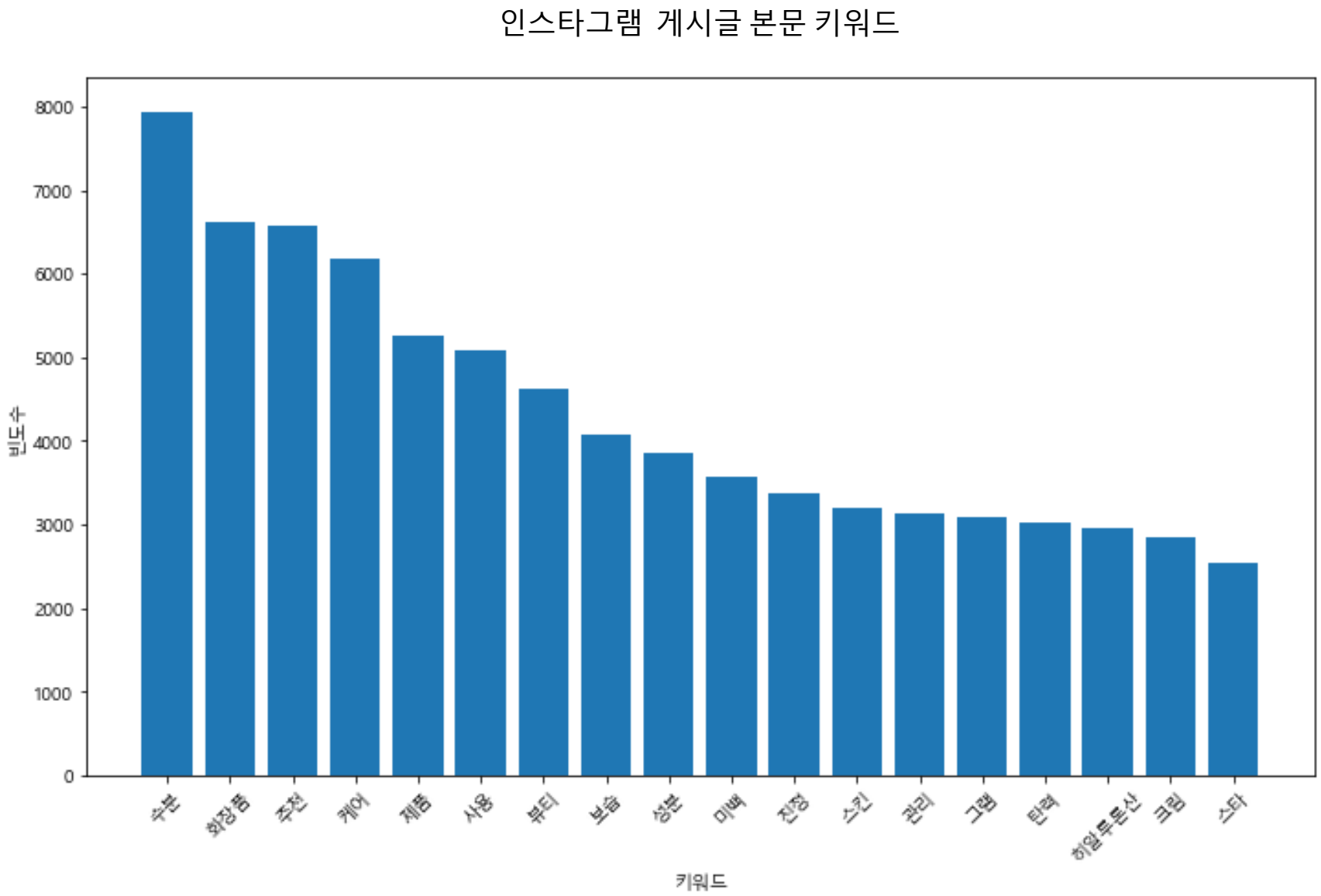
```
10251
```

```
#content 특수문문자 전처리
df_sorted_by_values["content"] = df_sorted_by_values["content"].str.replace(pat=r'[^ㅟw]', repl=r" ", regex=True)
df_sorted_by_values["like"] = df_sorted_by_values["like"].str.replace(pat=r'[^ㅟw]', repl=r'', regex=True)
df_sorted_by_values["tags"] = df_sorted_by_values["tags"].str.replace(pat=r'[^ㅟw]', repl=r" ", regex=True)

#like 컬럼에 숫자를 제외한 모든데이터 전처리
df_sorted_by_values["like"] = df_sorted_by_values["like"].str.replace(pat=r'[ㅟD]', repl=r'', regex=True)

#content 컬럼에 자음으로된 모든데이터 전처리
df_sorted_by_values['content'] = df_sorted_by_values['content'].str.replace(pat=r'[ㄱ-ㅎ]', repl=r"", regex=True)
```

인스타 크롤링 핵심 코드(시각화)



인스타 크롤링 핵심 코드(시각화)



인스타 크롤링 핵심 코드(시각화)

해시태그

게시글 본문


03.
AIDA CRM 데이터 시각화

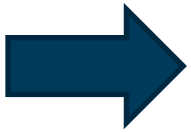
DATA 전처리

데이터시각화

고객정보
총 결제금액
휴대전화
나이
성별
주문상품명
주문상품명(옵션)
수량
회원등급
최종접속일
최종주문일
총방문횟수
추가사항1(유입경로)
추가사항2(피부타입)
추가사항3(피부고민)
추가사항4(선호제품)

제품정보
상품명
품목 구성방식
옵션 표시방식
옵션입력
판매상태
상품 간략설명
공급가
상품가
자체분류코드

주문정보
주문상품명
주문상품명(옵션포함)
수량
판매가
수령인
수령인휴대전화
수령인우편번호
결제수단



CRM 정보
나이
성별
건조함
과잉피지
여드름
좁쌀
홍조&예민
기미&잡티
모름
복합성
지성
미백앰플
수분앰플
쑥앰플
진정앰플

	아이디	나이	성별	피부타입	피부고민	선호제품	주문상품명	옵션입력	앰플
0	thsk486	4.0	0	건성	건조함,좁쌀여드름,홍조&예민	진정앰플	[투비리본X아이다] 앰플&크림 2차 공구!	필수 앰플 쑥앰플 진정앰플 수분라이트 앰플 쑥앰플 진...	[' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진...
1	thsk486	4.0	0	건성	건조함,좁쌀여드름,홍조&예민	진정앰플	[투비리본X아이다] 앰플&크림 2차 공구!	필수 앰플 쑥앰플 진정앰플 수분라이트 앰플 쑥앰플 진...	[' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진...
2	sjh3460	3.0	0	지성	여드름	진정앰플	[투비리본X아이다] 앰플&크림 2차 공구!	필수 앰플 쑥앰플 진정앰플 수분라이트 앰플 쑥앰플 진...	[' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진...
3	kaeng2da	3.0	0	복합성	좁쌀여드름	수분앰플	[투비리본X아이다] 앰플&크림 2차 공구!	필수 앰플 쑥앰플 진정앰플 수분라이트 앰플 쑥앰플 진...	[' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진...
4	gmlwlss625	2.0	0	복합성	과잉피지,여드름,좁쌀여드름,홍조&예민	진정앰플	[투비리본X아이다] 앰플&크림 2차 공구!	필수 앰플 쑥앰플 진정앰플 수분라이트 앰플 쑥앰플 진...	[' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진정앰플', ' 쑥앰플', ' 진...

전처리 내용

데이터 병합

- 주문정보, 데이터, 회원정보 데이터, 상품정보 데이터를 병합
- 주문정보 데이터와 회원정보 데이터는 휴대전화로 병합
- 병합 후 제품정보데이터를 상품명으로 병합

컬럼명 변경

- 추가사항2을 피부타입으로 변경
- 추가사항3을 피부고민으로 변경
- 추가사항4을 선호제품으로 변경

칼럼별 수정 사항

- 아이디 나이 성별 피부타입 피부고민 선호제품 주문상품명 옵션입력 앰플 Freature 열만 따로 추출
- 나이 결측값은 평균인 3.7을 반올림하여 4로 대체 함
- 20이하 :0, 20-25 : 1, 25-30 : 2, 30-35 : 3, 35-40 : 4, 40-45 : 5. 45-50 : 6, 50-55 : 7, 55-60 : 8
- 성별: 여자 0, 남자 1 로 대체
- 앰플 행은 옵션입력 행에서 '앰플'이라는 단어가 들어간 것 정규표현식으로 추출함

One-hot Encoding

	나이	성별	건성	모름	복합성	지성	건조함	과잉피지	여드름	좁쌀	홍조&예민	기미&잡티	미백앰플	수분라이트	수분앰플	쑥앰플	진정앰플
0	3.0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0
1	3.0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1
2	4.0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1
3	1.0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	1
4	1.0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1

전처리 내용

피부타입

- 건성 모름 복합성 지성으로 나눈 뒤 0, 1 값 부여

피부고민

- 6개(건조함 과잉피지 여드름 좁쌀여드름 홍조&예민 기미&잡티)의 피쳐로 나누어서 0과 1값 부여

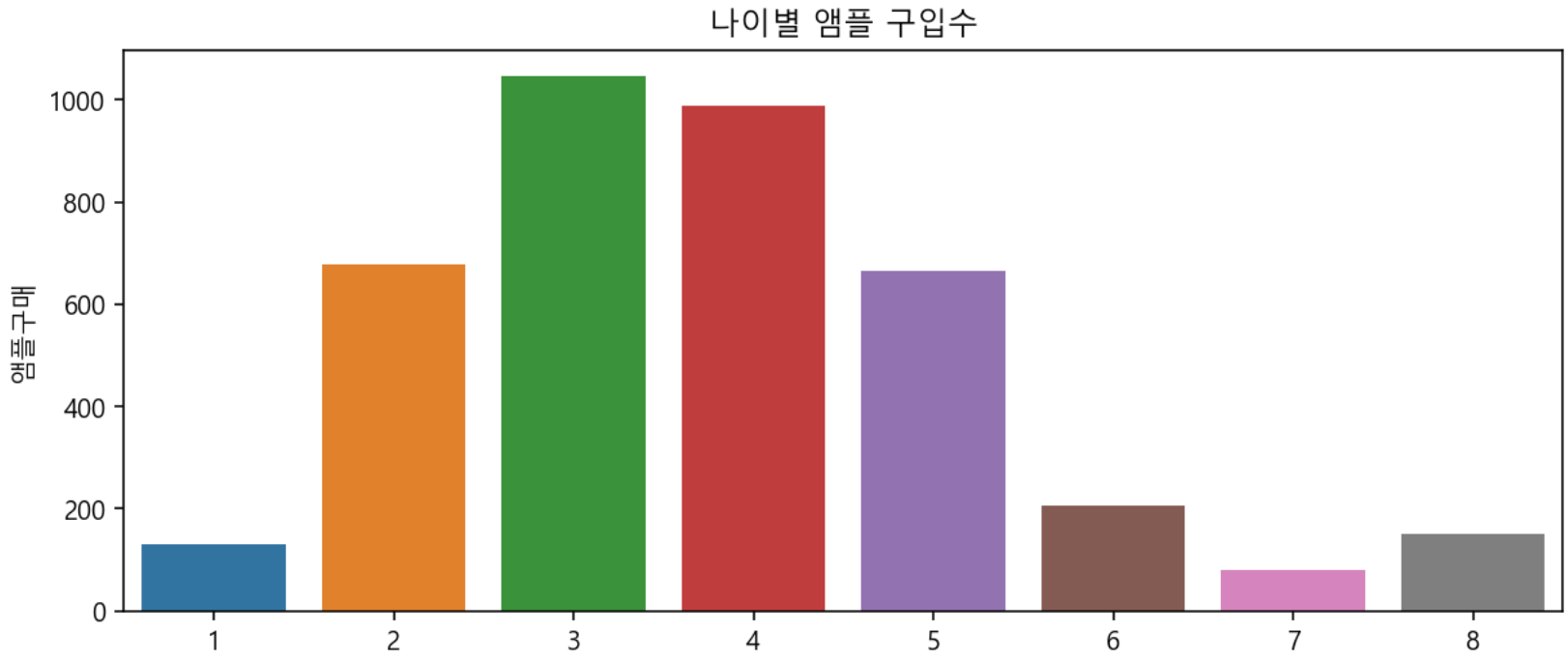
코드(그래프) - 나이에 따른 앰플 구입 수

```
plt.figure(figsize=(10, 4))
ax = sns.barplot(x=crm_age.index, y="앰플구매", data=crm_age, palette="tab10")
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.title("나이별 앰플 구입수")
```

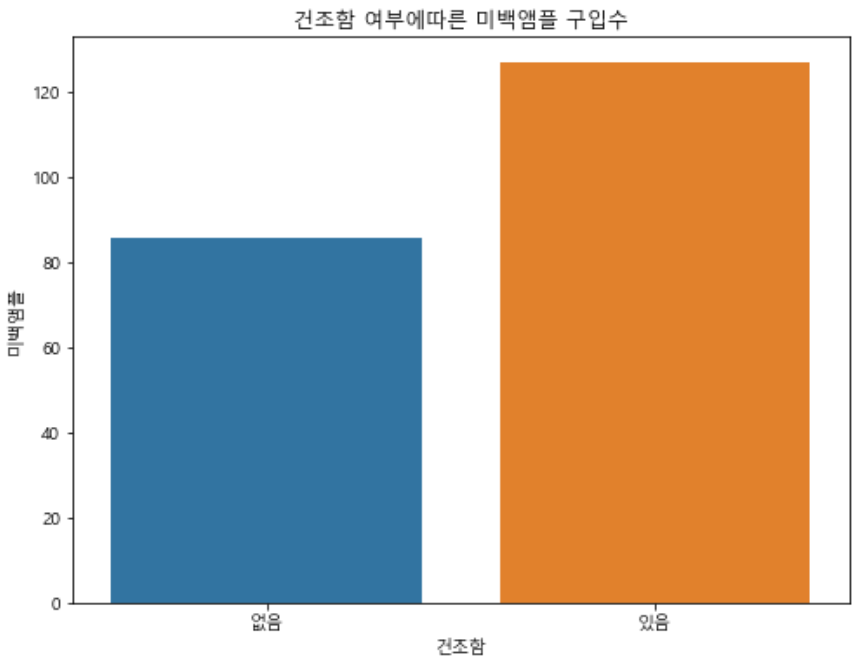
코드(그래프) – 각 피부고민 여부에 따른 앰플 구입 수

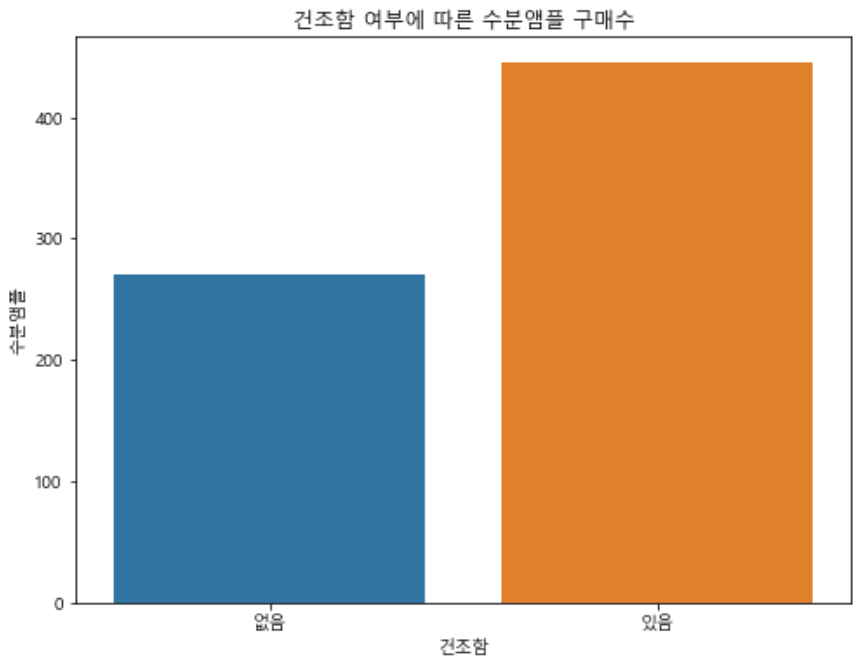
```
def problem_graph(xlabel, ylabel, title):
    plt.figure(figsize=(8, 6))
    ax = sns.barplot(x=xlabel.index, y=ylabel, data=xlabel, palette="tab10")
    plt.rcParams['font.family'] = 'Malgun Gothic'
    plt.title(title)
```

나이에 따른 앰플 구입 수

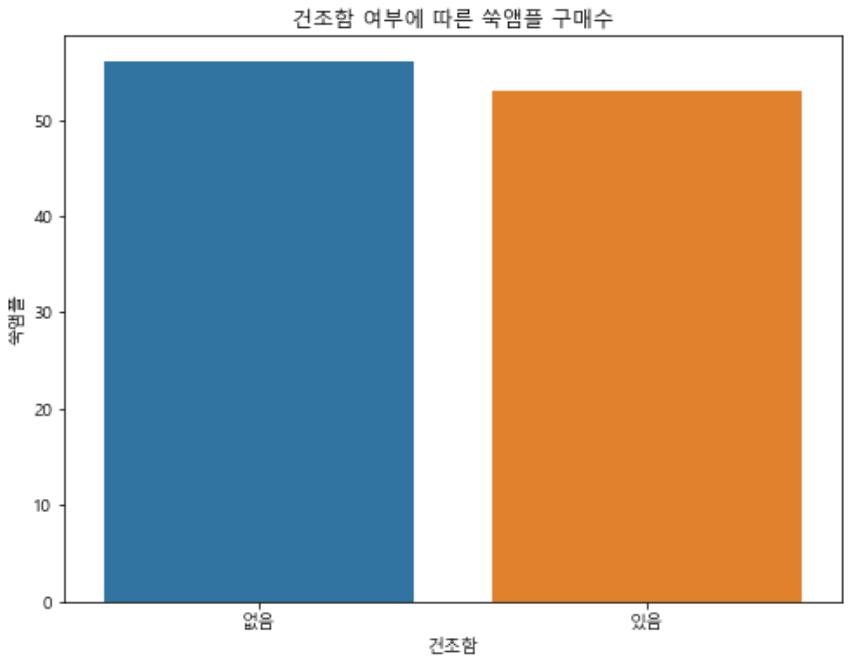


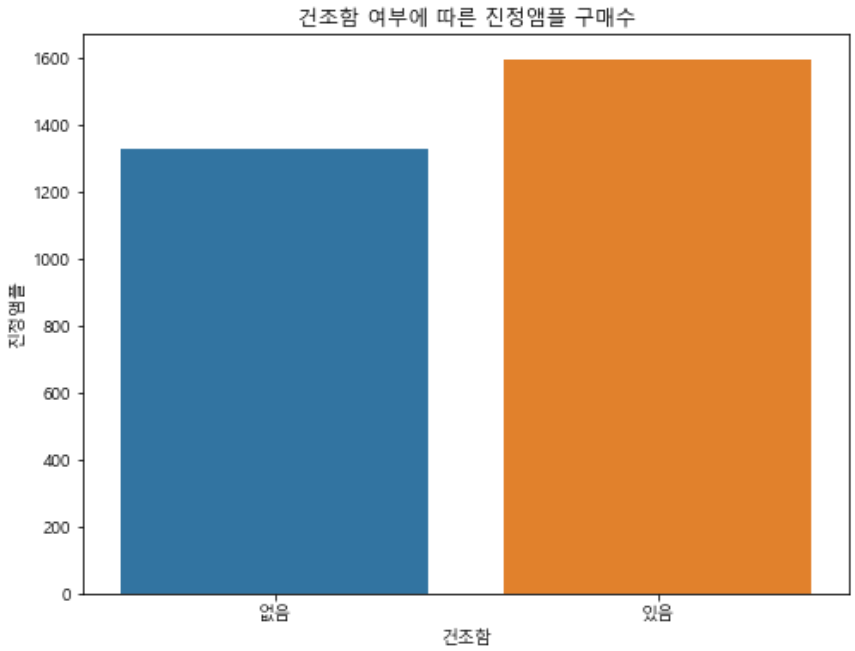
건조함 여부에 따른 앰플 구입 수



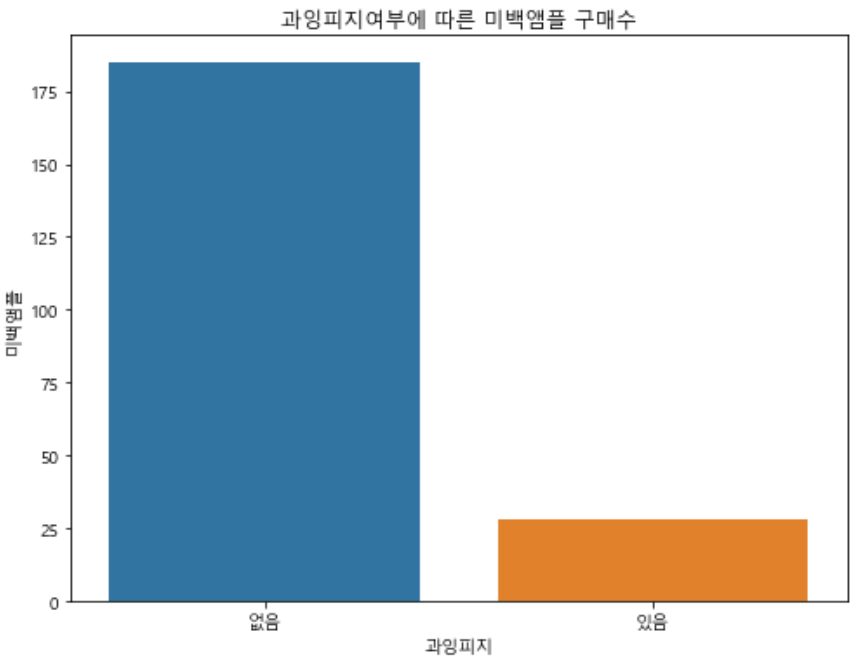


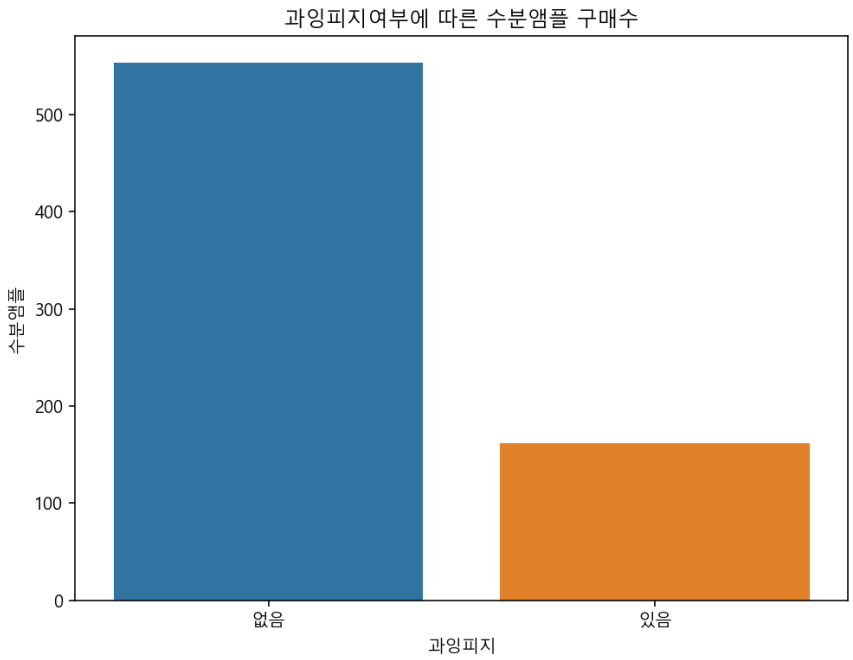
건조함 여부에 따른 앰플 구입 수



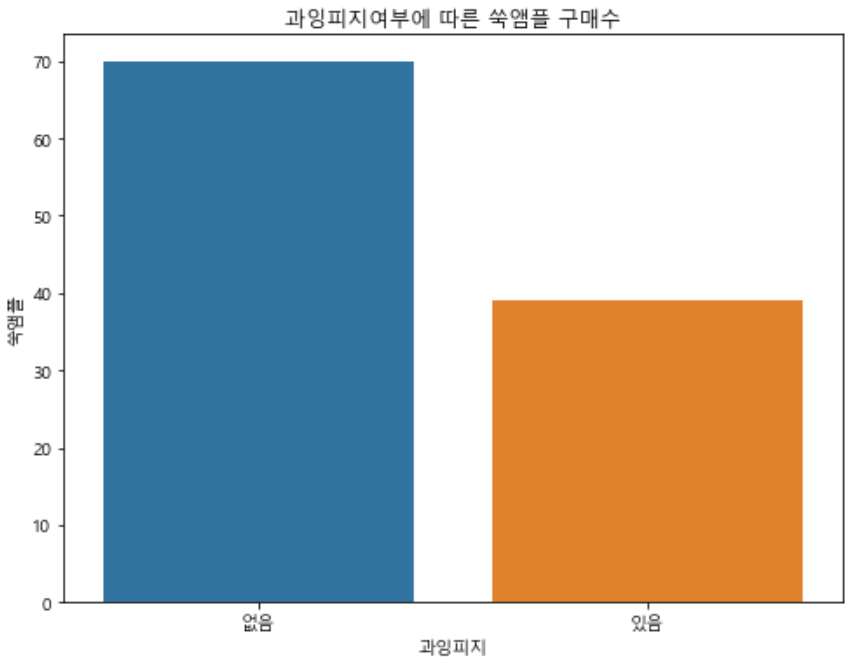


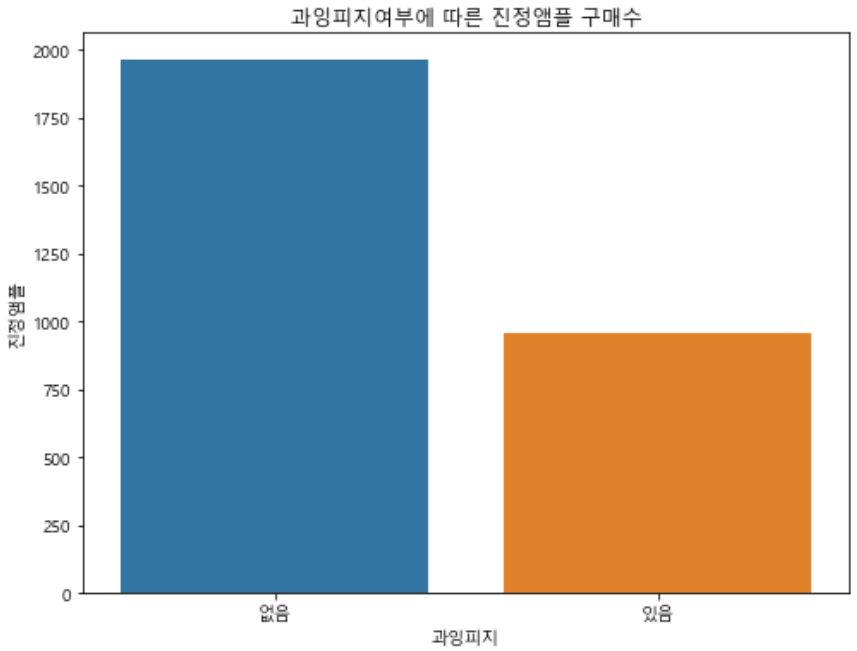
과잉 피지 여부에 따른 앰플 구입 수



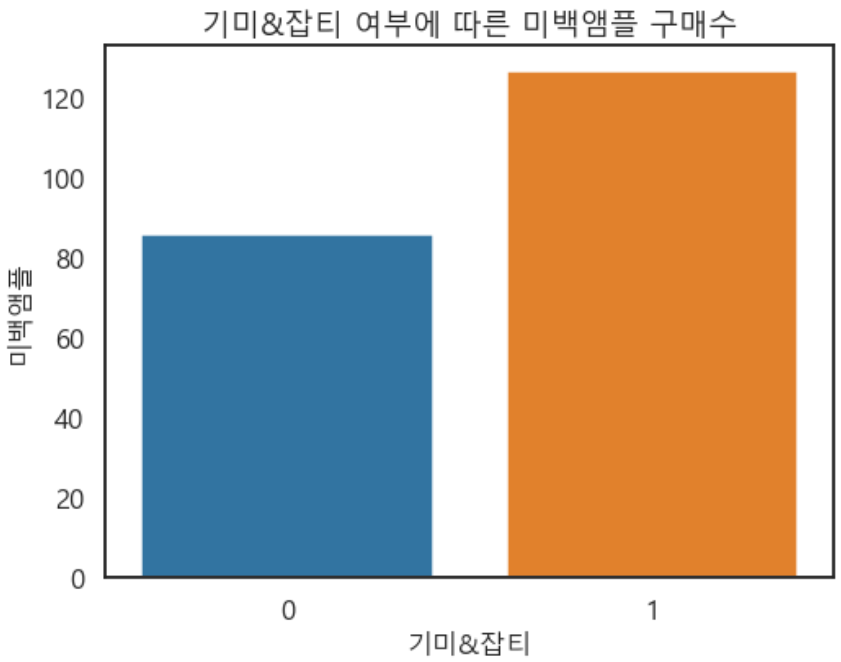


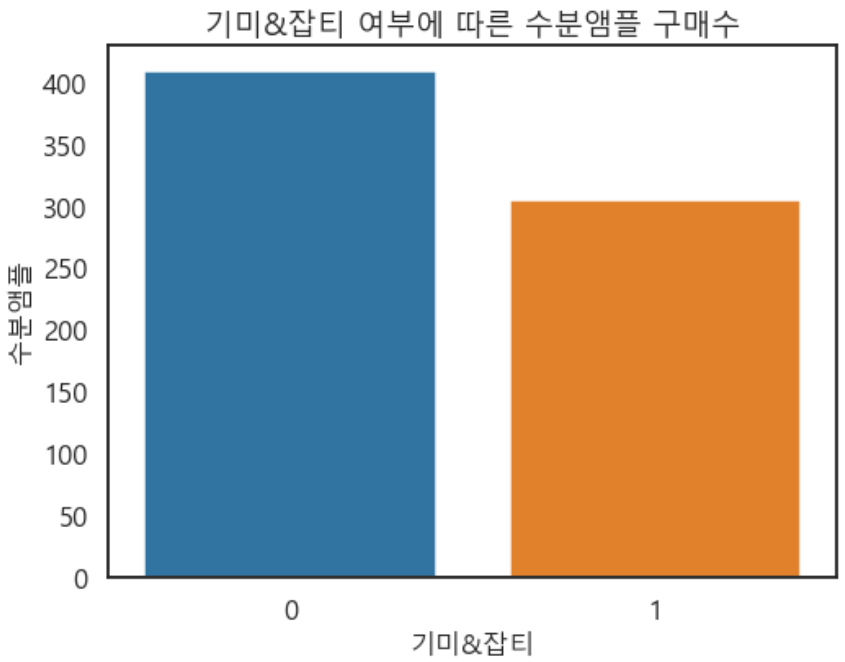
과잉 피지 여부에 따른 앰플 구입 수



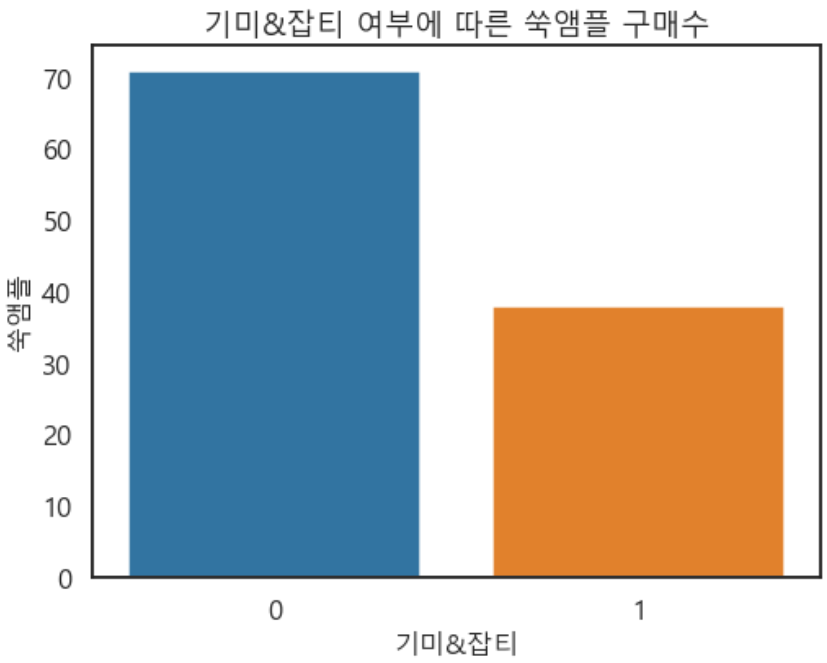


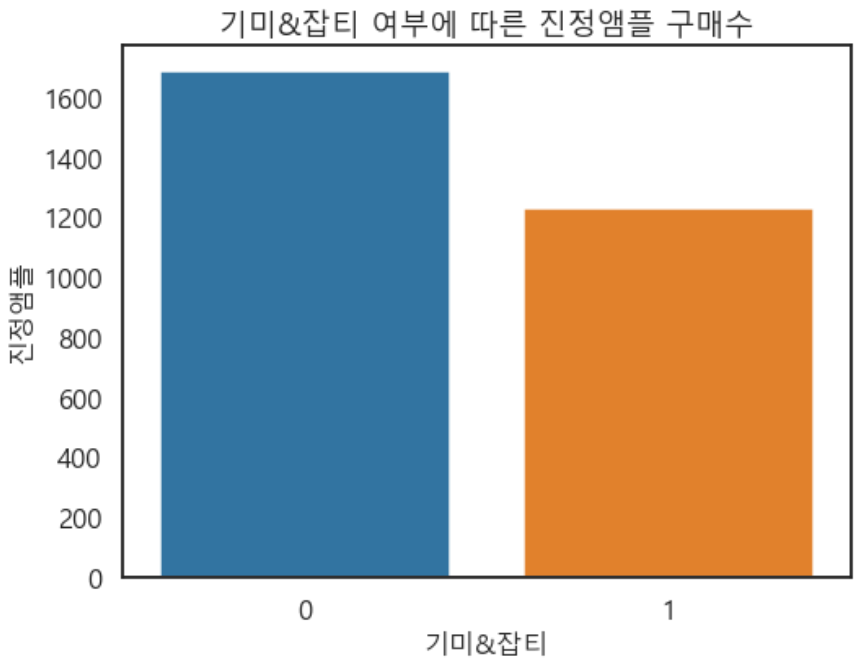
기미&잡티 여부에 따른 앰플 구입 수



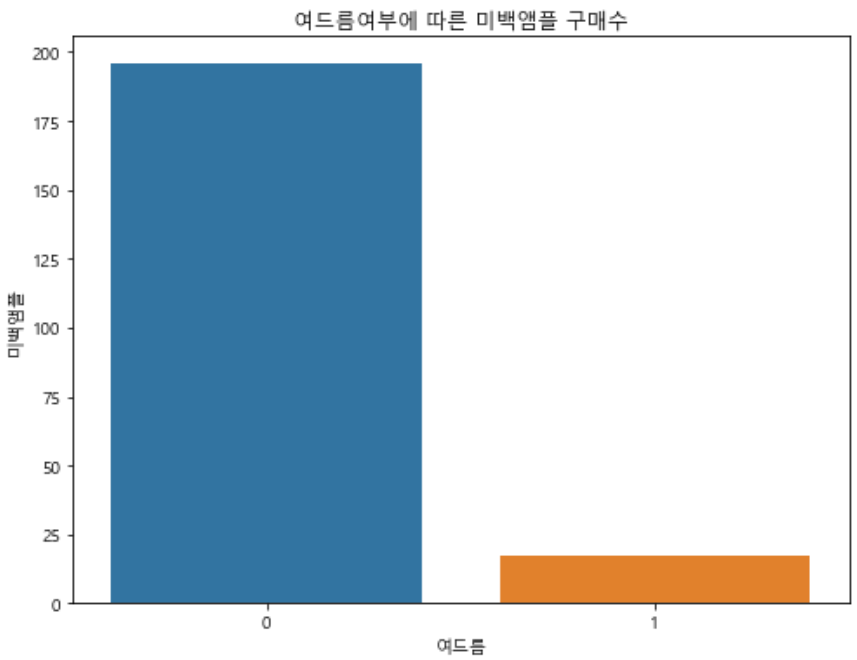


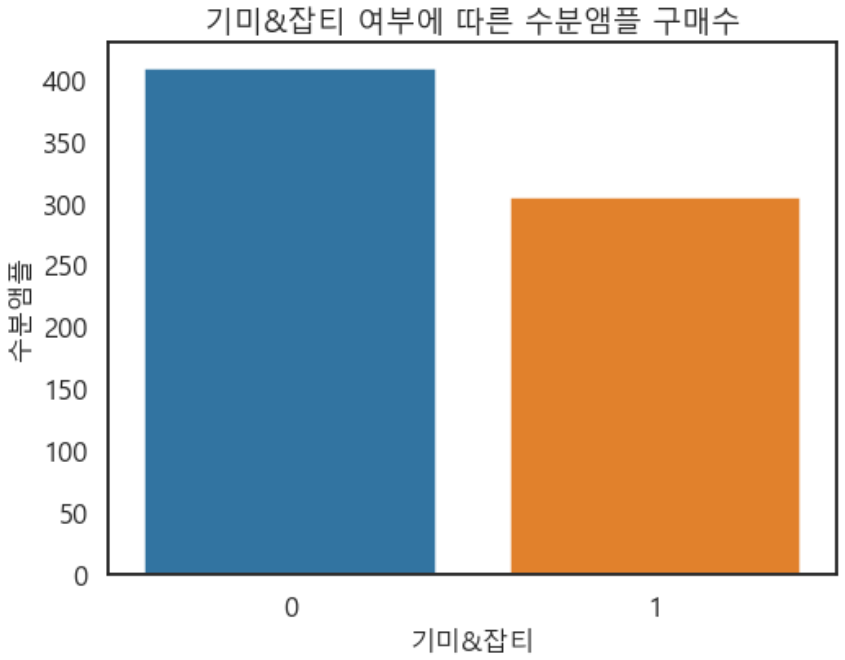
기미&잡티 여부에 따른 앰플 구입 수



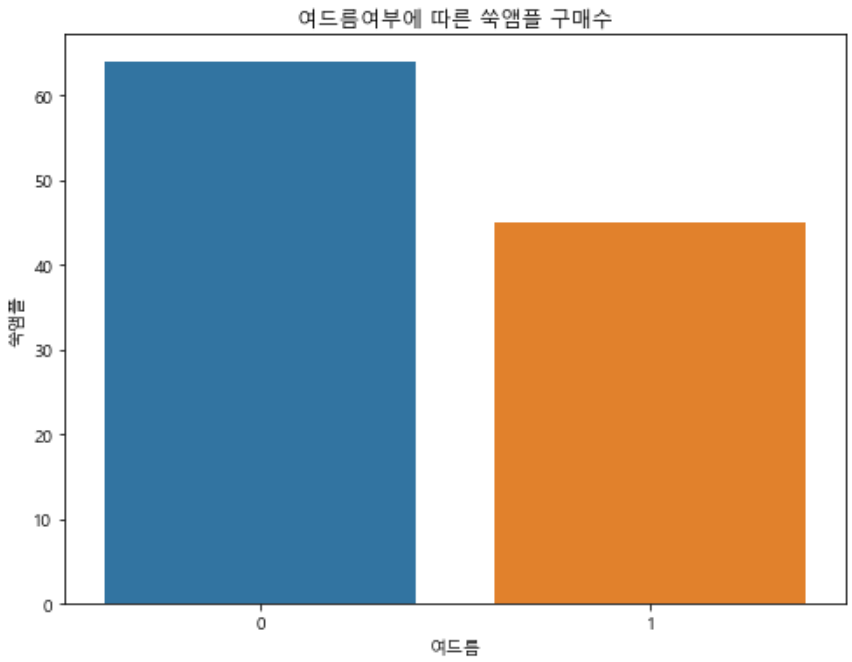


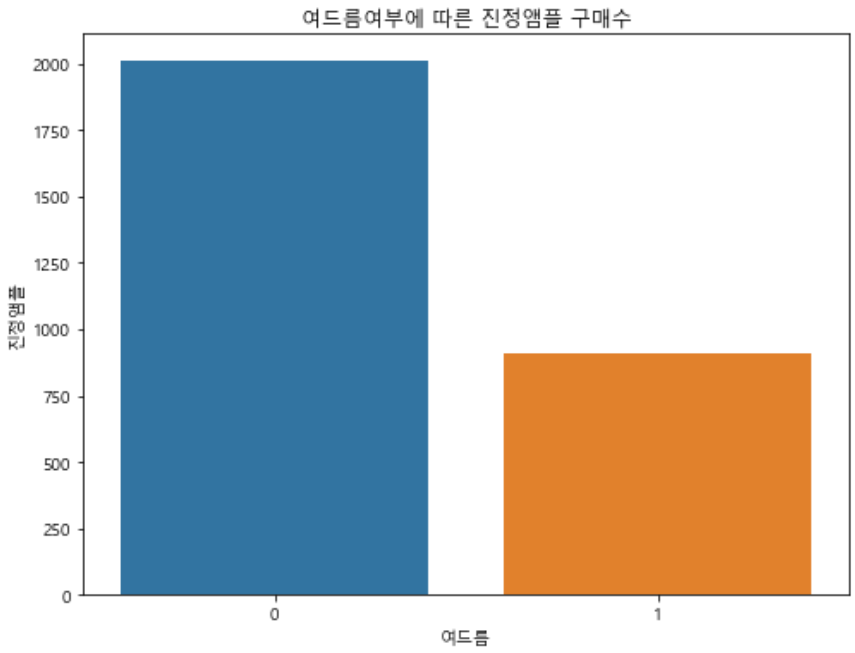
여드름 여부에 따른 앰플 구입 수



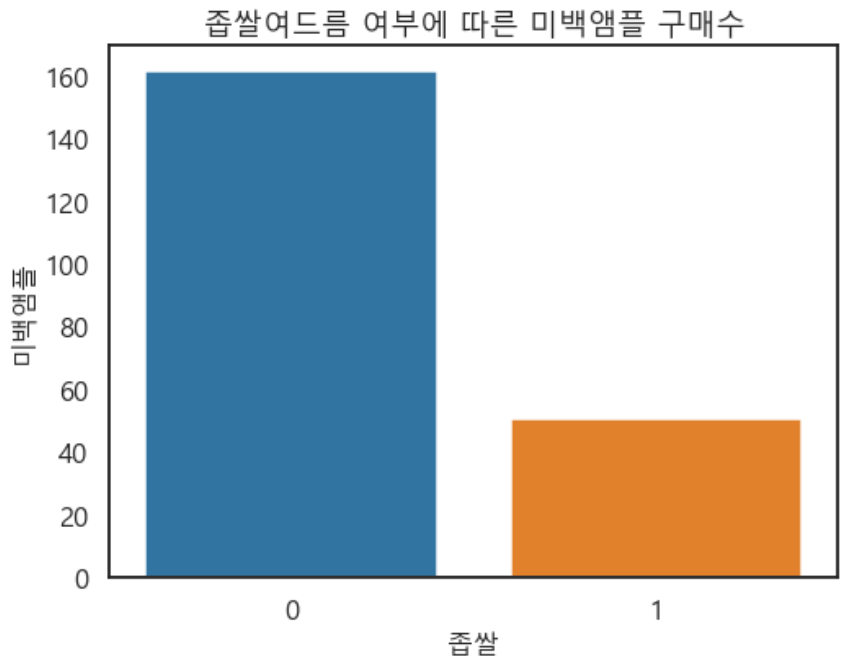


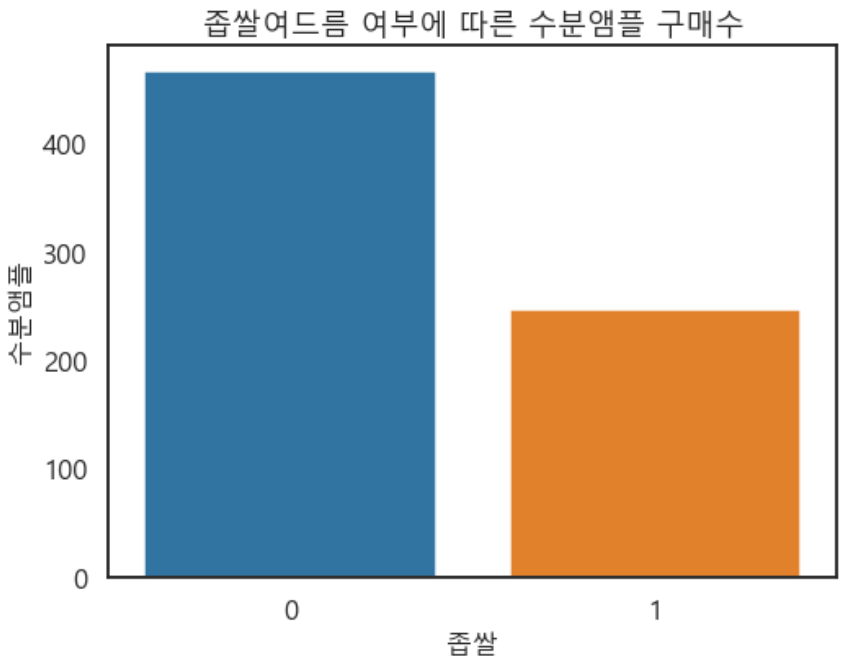
여드름 여부에 따른 앰플 구입 수



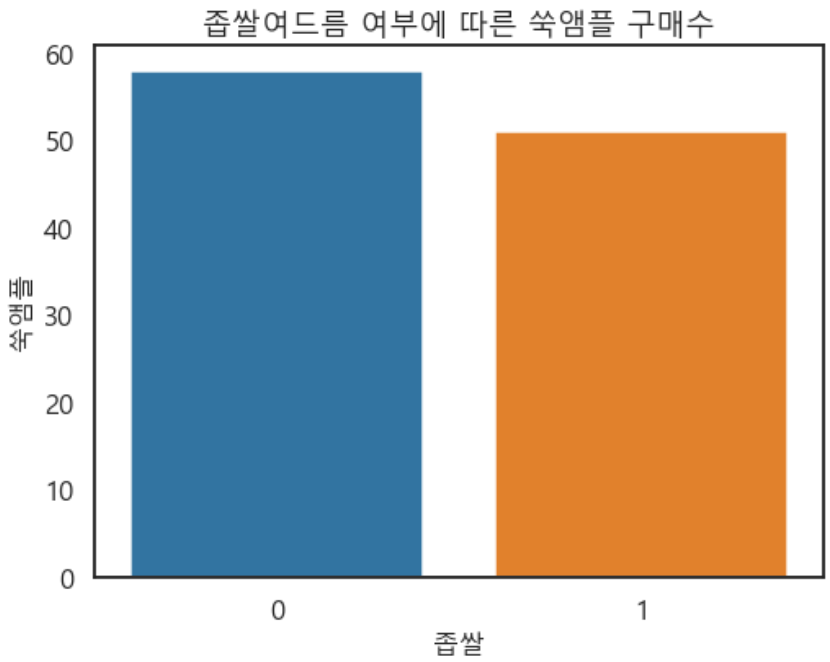


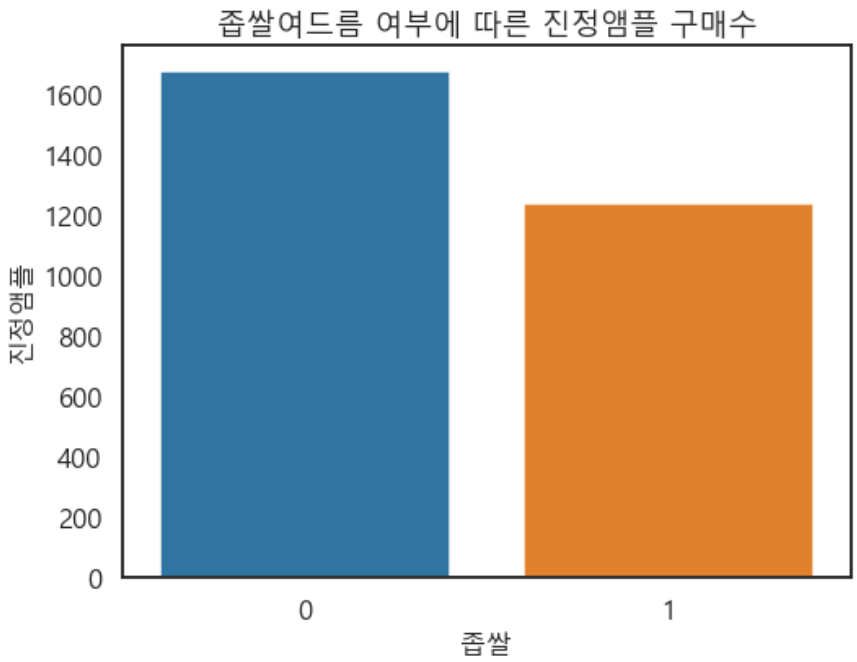
좁쌀여드름 여부에 따른 앰플 구입 수



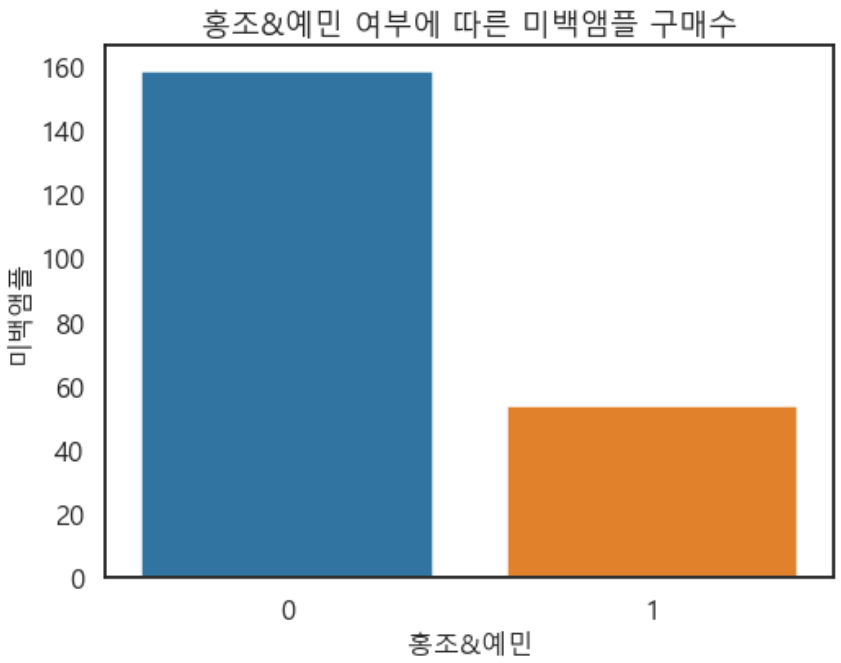


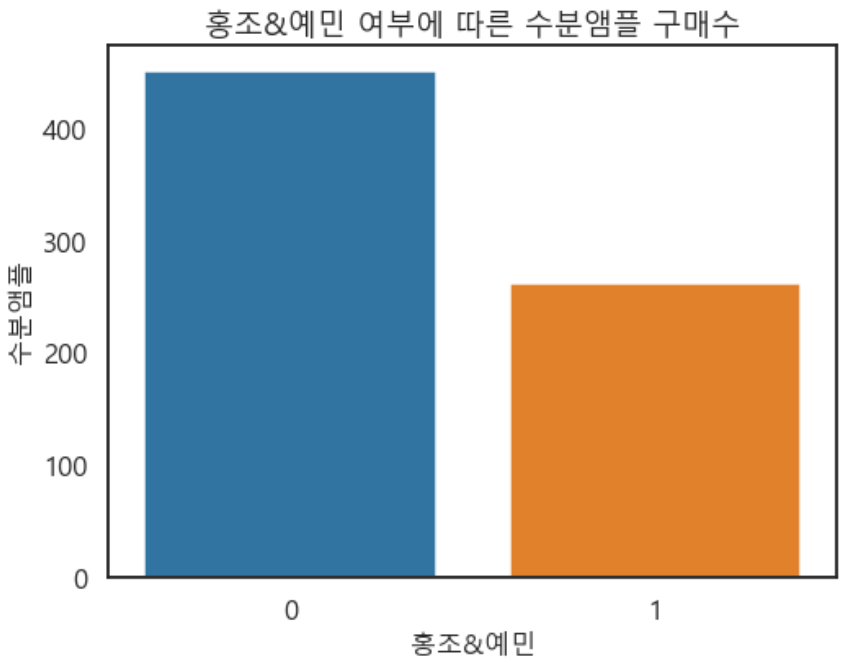
좁쌀여드름 여부에 따른 앰플 구입 수



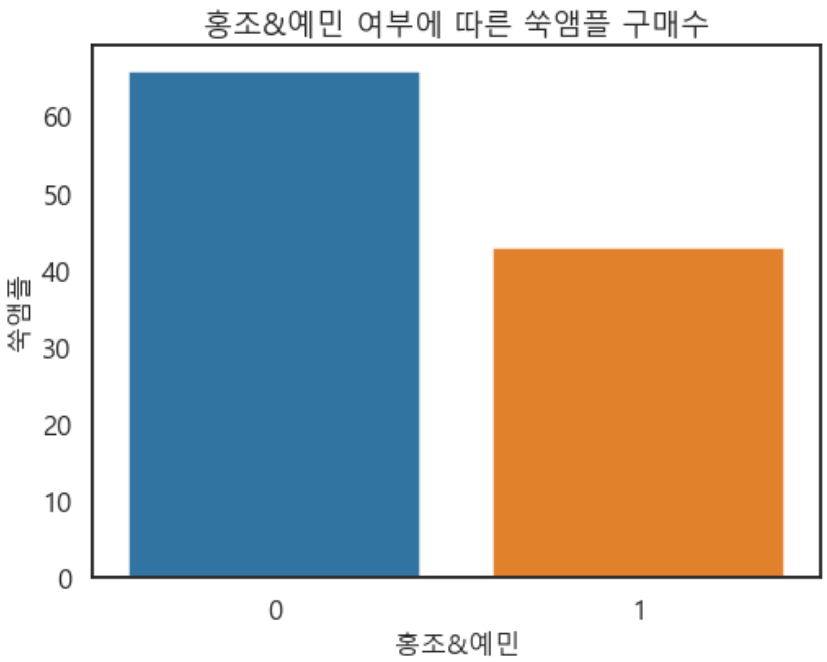


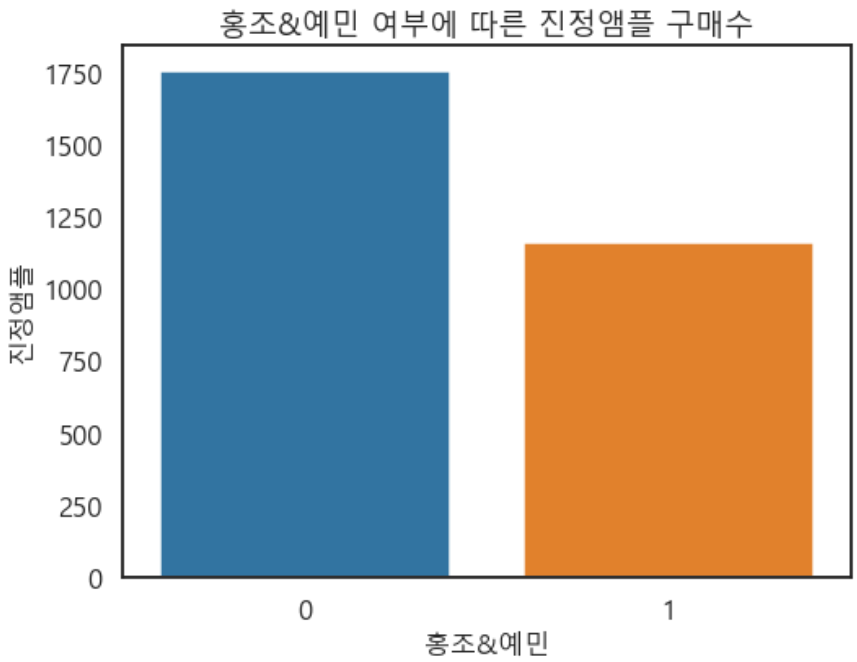
홍조&예민 여부에 따른 앰플 구입 수



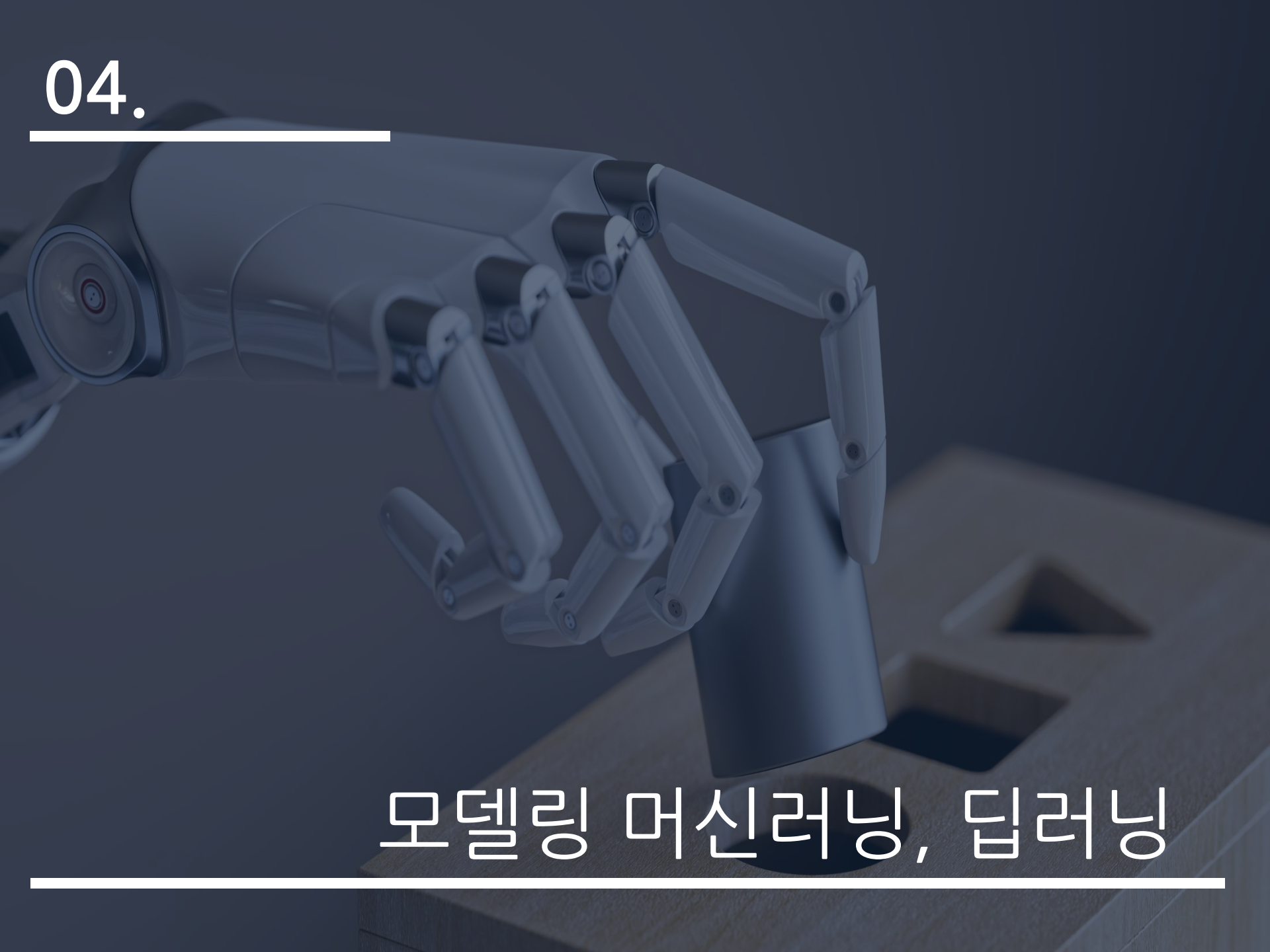


홍조&예민 여부에 따른 앰플 구입 수





04.

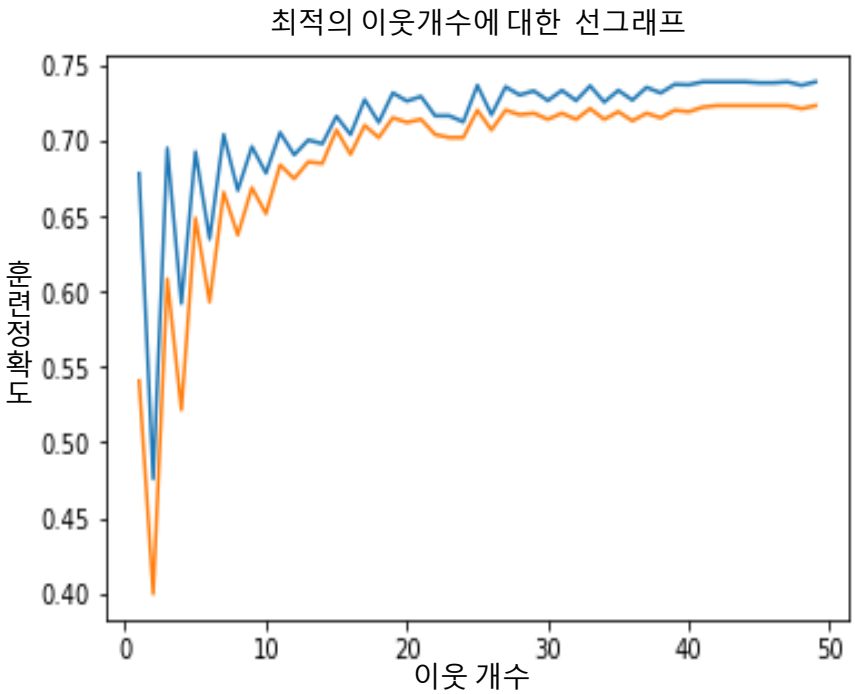


모델링 머신러닝, 딥러닝

Python 이용

파이썬에서 제공하는 sklearn, Google의 TensorFlow, Keras를 활용하여 분류 알고리즘 모델링

KNN 모델링




```
clf = KNeighborsClassifier(n_neighbors=25).fit(X_train,y_train)
```

```
print(round(clf.score(X_train,y_train)*100,2))
print(round(clf.score(X_test,y_test)*100,2))
```

최적의 이웃의 개수는 25개

훈련 정확도 : 73.83%
테스트 정확도 : 72.31%

SVM 외 3개 모델링

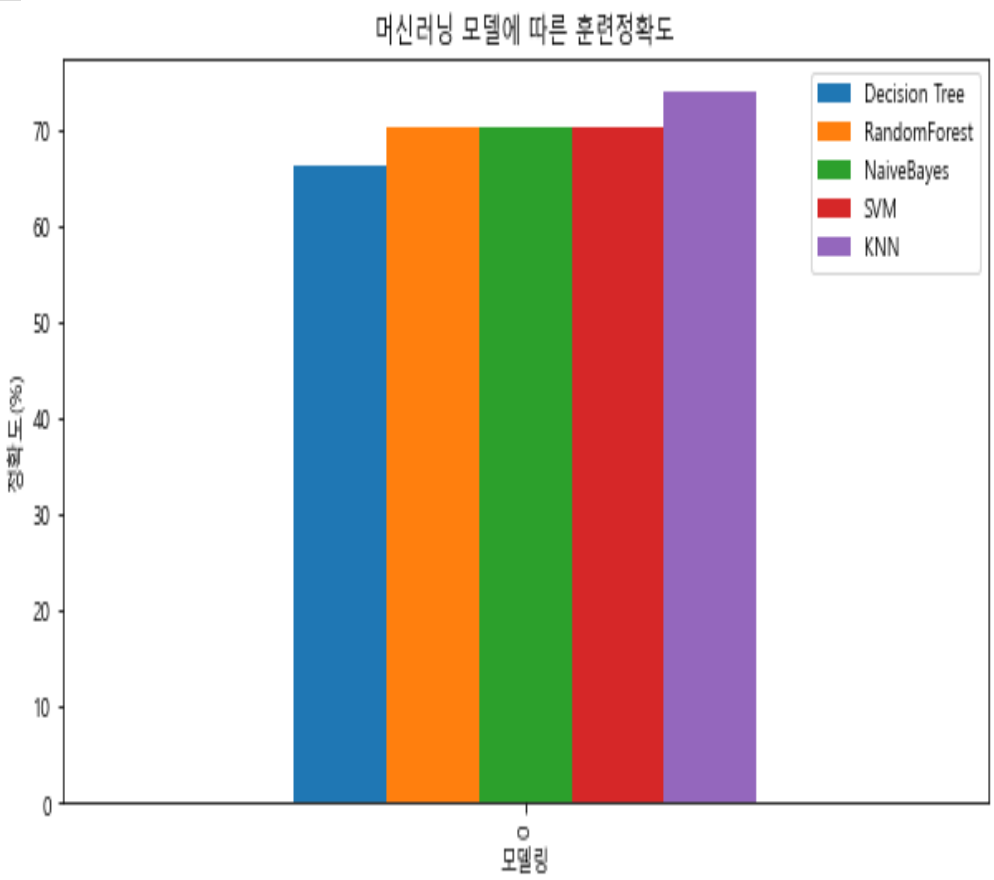
```
kfold = KFold(n_splits=50, shuffle=True, random_state=0)
```

```
# Decision Tree

dtclf = DecisionTreeClassifier()
score = cross_val_score(dtclf, X,np.ravel(y), cv=kfold, scoring="accuracy"
print(score)
round(np.mean(score)*100,2)
```

```
# RandomForest
rfclf = RandomForestClassifier(n_estimators=300)
score = cross_val_score(rfclf, X,np.ravel(y), cv=kfold, scoring="accuracy")
print(score)
round(np.mean(score)*100,2)
```

```
# NaiveBayes
nb = GaussianNB()
score = cross_val_score(rfclf, X,np.ravel(y), cv=kfold, scoring="accuracy")
print(score)
round(np.mean(score)*100,2)
```

```
# SVM
svcclf = SVC()
score = cross_val_score(rfclf, X,np.ravel(y), cv=kfold, scoring="accuracy"
listall.append(score)
listavg.append(round(np.mean(score)*100,2))
print(score)
round(np.mean(score)*100,2)
```



코드

```
x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.2, random_state=20)
```

모듈

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from keras import optimizers
```

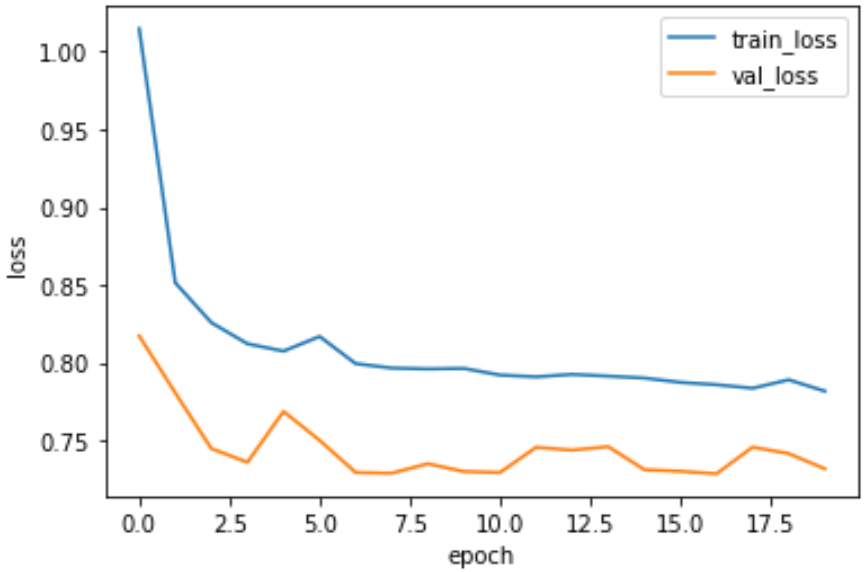
```
model = Sequential()

model.add(layers.Dense(256, activation='relu', input_shape=(11,)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#adam = optimizers.Adam(lr=0.0001)
```

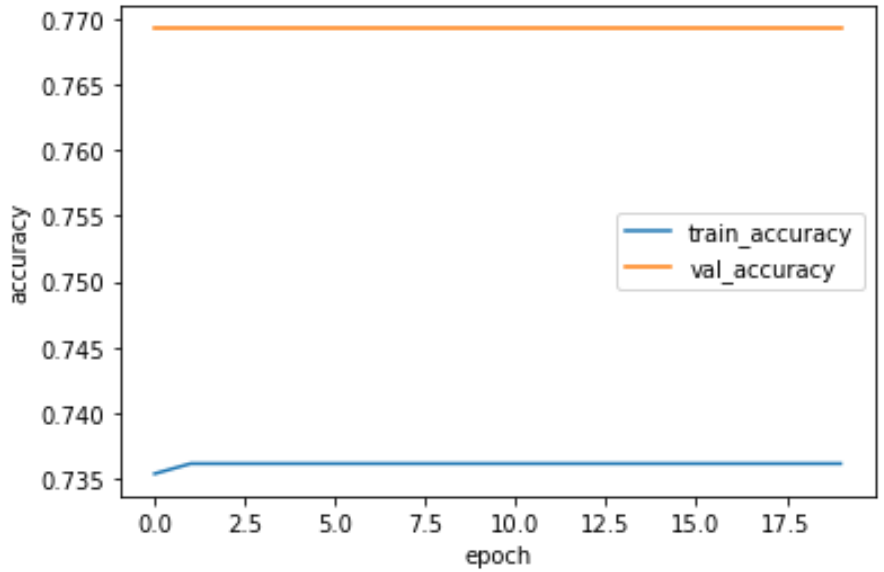
레이어에 대한 내용

```
model = Sequential()

model.add(layers.Dense(256, activation='relu', input_shape=(11,)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#adam = optimizers.Adam(lr=0.0001)
```

하이퍼파라미터

```
model.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=20, batch_size=64, validation_data=(x_test, y_test))
```


train_loss
val_loss
loss
epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


train_accuracy
val_accuracy
accuracy
epoch
0.770
0.765
0.760
0.755
0.750
0.745
0.740
0.735
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Adam(Ir=0.001)

Dense Layer1	Dense Layer2	Epoch	Bath Size	Dropout	Accuracy	Val Accuracy
64	64	20	64	-	72.85%	74%
64	32	20	64	-	74.20%	74%
32	16	20	64	-	73.11%	74%
64	64	20	64	0.5	73.35%	74%
32	16	20	64	0.5	72.24%	74%
16	8	20	64	0.5	73.26%	74%
64	64	20	64	0.3	73.26%	74%

SGD(Ir=0.01)

Dense Layer1	Dense Layer2	Epoch	Bath Size	Dropout	Accuracy	Val Accuracy
64	64	20	64	-	72.74%	74%
64	32	20	64	-	72.46%	74%
32	16	20	64	-	71.93%	74%
64	64	20	64	0.5	72.45%	74%
32	16	20	64	0.5	71.79%	74%
16	8	20	64	0.5	73.41%	74%
64	64	20	64	0.3	73.33%	74%

05.

결과 및 느낀점

결과 및 느낀점

제품추천 예시

목표

제품 추천 시스템을 제작하여 웹 페이지 통해
실제 소비자에 적용해보고 발생한 데이터를 재분석하여
매출에 영향이 있는지 인사이드 도출

현실

- 웹사이트 구축하기엔 짧은 시간
- 충분하다고 생각했지만 그렇지 못한 자사 데이터
- 머신러닝, 딥러닝 성능 부족

느낀점

- 개인의 피부타입에 따라 구매를 하지 않고 유명제품을 구매하는 경우가 많음
- 시장에서 진정 앰플이 주로 판매됨
- AIDA 코스메틱의 고객 약 84%가 진정 앰플구매

개선점

- CRM정보 전처리에 대한 다양한 방법 모색
- 화장품 시장에 대한 이해
- 독립변수 개선
- AI학습능력개선