

4조 포트폴리오

1. 프로젝트 소개

1-1. 프로젝트 구성원 소개 및 역할

Data Science

- **이재연 (조장)**: Jordan 신발 추천, 이미지 embedding 및 군집을 활용한 신발추천 프로그램 작성
- **김예은**: Jordan 인스타 알림 서비스, 객체 인식 & 이미지 분류 모델, 자동화 함수 작성
- **신혜빈**: Jordan 신제품 가격 예측, XGBoost 분석, 가격예측 프로그램 작성

Data Engineer

- **김흥민**: 데이터 파이프라인 & ERD 설계, 한국경제/ 빅카인즈 사이트 데이터 크롤링, 데이터 전처리 & 분석 & 시각화
- **박상권**: Lucky Draw & 매일경제 크롤링, 데이터 전처리 & 시각화, 인스타 새 게시물 크롤러, Airflow를 활용한 알림 서비스
- **박선재**: 데이터 파이프라인 & MongoDB & MySql 구축, KREAM 크롤링, 데이터 전처리, 이미지 색깔 추출, Airflow를 활용한 알림 서비스
- **최진호**: 데이터 파이프라인, KREAM 사이트 데이터 크롤링, 통합 데이터 전처리 & 분석 & 시각화, Slack 챗봇

1-2. 프로젝트 목적 및 서비스 소개

[1] Jordan 신발 추천 서비스

리셀 시장은 비교적 최근에 새로운 제테크 방식으로 부상해 용어 자체가 익숙하지 않은 사람이 많다.

본 프로젝트는 **리셀 시장**을 잘 모르는 사람도 이곳에 입문할 수 있도록 도움을 주기 위한 목적을 가지고 있다. **리셀 시장 입문자들**의 흥미를 유도하기 위한 신발 추천 서비스를 기획하고자 한다.

이번 프로젝트 동안 구현할 데이터는 국내 리셀 시장 중에 가장 큰 규모를 자랑하는 KREAM 사이트의 Jordan 신발 사진을 기반으로 한다. 화면 상에 여러 개의 Jordan 사진이 보이면, 자신의 취향에 맞는 신발을 여러 번 골라 그 신발과 비슷한 신발을 추천해주는 서비스로 입문자 에게도 친숙하게 다가갈 것을 기대한다.

[2] Jordan 신발 신제품 가격 예측 서비스

한정판 Jordan 신제품 구매 희망자를 위한 **가격 예측 서비스**를 만든다.

구매자는 한정판 Jordan 신제품의 예측된 가격을 알게되므로 '본인이 힘들게 구한 Jordan 신제품이 어느 정도의 가치가 있을지'를 확인할 수 있다.

최저가, 최고가, 가장 최근 체결된 가격 데이터를 활용해서 총 3가지 가격이 예측될 것이다.

이번 프로젝트 동안 구현할 데이터는 단순히 235, 270 사이즈에 한정되어 구현될 것이지만, '다른 사이즈의 데이터 체결 내역과 추가로 시간이 흘러 방대해 질 체결 내역 데이터로는 더 정교하고 섬세한 모델이 탄생하지 않을까.' 기대하며 서비스를 기획하게 되었다.

[3] 유명인 인스타그램 Jordan 신발 게시물 알림 서비스

유명인의 인스타그램에 **Jordan 신발**이 올라오게 되면 그 셀럽이 소유하고 있다는 사실만으로도 가격이 상승하게 된다. 이러한 원리에 따라, 스타들의 인스타그램에 **Jordan 신발**이 포함된 게시물이 올라오면 Jordan 신발 리셀로 제테크를 하는 사람들을 위해서 '어떤 신발 모델이 인스타그램에 포스팅 되었는지 모델명을 알려주는 알림 서비스'를 기획하게 되었다.

이번 프로젝트 동안 구현할 데이터는 단순히 **Jordan 신발의 3개의 모델**과 이들과는 완전히 결이 다른 **슬리퍼** 사진을 활용해서 총 4개의 클래스를 가지고 모델을 구성했다. 향후에 클래스를 Jordan 신발 모든 모델로 넓힐 수 있다는 확장성을 기대하고 있다.

2. 프로젝트 데이터 수집 및 전처리 (데이터 엔지니어링팀)

2-1 데이터 수집

- 위의 서비스를 구현하고자 'KREAM', '빅카인즈', '럭키드로우', '인스타그램' 사이트에서 데이터를 수집하였다.
- 데이터를 수집하기 전, **데이터에 대한 관계도와 파이프라인** 설계가 필요해서 각 테이블 별로 entity relationship diagram arrow types을 확인했다. 이어서 ERD를 구성하였고 데이터의 흐름을 만들기 위해 파이프라인을 설계했다.
- 크롤링한 데이터는 **공용 AWS** 상에 **MongoDB**를 구축하여 **각 수집처 및 특성에 맞게** 저장하여 **Data Lake**로 활용하였다.

- **첫 번째 수집처 - KREAM** : 리셀 플랫폼으로 Jordan 신발에 대한 정보를 제공하였고 고글 릭 및 무한 스크롤을 해야하기에 Selenium 기법을 사용하였다. KREAM과 KREAM 디테일로 나누어 **카테고리 페이지**에서는 '제품명', '가격', '좋아요', '링크', '총 거래량'을 **상품상세 페이지**에서는 '제품명', '최근 거래가', '색상', '발매일', '최근 구매가', '최근 판매가', '이미지url'을 수집하였다.
- **두 번째 수집처 - 뉴스 기사** : 리셀 시장의 확대로 연도 별 증가하는 기사 수를 그래프로 한눈에 보게 하기 위해 처음에는 '한국경제'와 '매일경제'에서 scrapy로 데이터를 수집하였으나, 프로젝트 시간 상의 문제로 빅카인즈라는 곳으로 대체하여 기사 데이터를 수집하였다. '빅카인즈' 사이트는 사이트 내에 어플 형태로 존재하였기 때문에 scrapy로는 불가하여 셀레니움으로 데이터를 수집하였다.
- **세 번째 수집처 - 럭키드로우** : 신제품을 응모하는 사이트로 신제품 정보를 얻기 위해 scrapy를 이용해 데이터를 수집하였다.
- **마지막 수집처 - Instagram** : 인스타그램은 Selenium으로 데이터를 수집하였다. '유명인 인스타그램 Jordan 신발 게시물 알림 서비스'를 위해 유명인 게시글을 수집하기 위해 진행하였다. 처음에는 scrapy를 사용해서 가져오고자 했으나, 발급받은 키로는 특정 계정 이외에 데이터를 수집할 수 없는 것을 확인하여 Selenium으로 진행하였다.

• 코드

(1) KREAM 체결내역 크롤러 (Scrapy - Spider)

```
import scrapy
import time
import pandas as pd

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as

from KreamTrade.items import KreamtradeItem

def get_chrome_driver():
    chrome_options = webdriver.ChromeOptions()
```

```

# chrome_options.headless = True

driver = webdriver.Chrome(
    service=Service(ChromeDriverManager().install()),
    options=chrome_options
)

return driver

class Spider(scrapy.Spider):
    name = 'KreamTrade'
    allow_domain = ['kream.co.kr']

    custom_settings = {
        'DOWNLOADER_MIDDLEWARES': {
            'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,
            'scrapy_fake_useragent.middleware.RandomUserAgentMiddleware': 800
        }
    }

    start_urls = ['https://kream.co.kr']

    def __init__(self):
        super().__init__()

        self.df = pd.read_csv('C:/Users/yhn03/MulticampusProject/data/kream.csv')
        self.id = 'yhn03955@gmail.com'
        self.pw = 'multi1234!'

    def parse(self, response):
        driver = get_chrome_driver()
        driver.implicitly_wait(3)
        driver.maximize_window()

        domain = 'https://kream.co.kr'
        driver.get(domain)

```

```

# Sign in
driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
time.sleep(2)

for idx in range(5, 6):
    all_trade = self.df.loc[idx, 'all_trade']
    link = self.df.loc[idx, 'link']
    try:
        '''
            Scroll down & Trade history scraping
        '''
        driver.get(link)

        model_no = driver.find_element(By.XPATH, '//*[@id="__layout"]/div')

        # Trade history window
        WebDriverWait(driver, 10).until(EC.element_to_be_clickable(
            driver.find_element(By.XPATH, '//*[@id="panel-trade-history"]')))
        time.sleep(1)

        # Select size
        driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
        size_cnt = len(driver.find_elements(By.XPATH, '//*[@id="__layout"]/div'))
        driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
        time.sleep(0.5)

        for i in range(2, size_cnt + 1):
            driver.find_element(By.XPATH, '//*[@id="__layout"]/div')
            time.sleep(0.5)
            size = driver.find_element(By.XPATH, f'//*[@id="__layout"]/div')

            if '235' not in size.text and '270' not in size.text:
                size.click()
                time.sleep(1)

```

```

        continue

    size.click()
    time.sleep(1)

    # Get end date
    try:
        driver.find_element(By.XPATH, '//*[@id="p
    except:
        continue
    time.sleep(1)
    end_date = driver.find_element(By.CSS_SELECTOR, 'p
    driver.find_element(By.XPATH, '//*[@id="p
    time.sleep(1)

    # Scroll down
    times = 0
    same_cnt = 0
    pass_product = False
    while True:
        if all_trade > 15000:
            if times > 500 and driver.find_element(By.XPATH, '//*[@id="p
                break
            elif all_trade > 8000:
                if times > 40 and driver.find_element(By.XPATH, '//*[@id="p
                    break
            else:
                if driver.find_elements(By.CSS_SELECTOR, 'p
                    break

        price_panel = driver.find_element(By.CSS_SELECTOR, 'p
        cur_height = driver.execute_script('return document.querySelector("p
        driver.execute_script('arguments[0].scrollTo(0,arguments[0].scrollHeight)')
        driver.execute_script('arguments[0].scrollTo(0,arguments[0].scrollHeight)')

        times += 1
        if times < 20:
            time.sleep(2)

```

```

        elif times < 40:
            time.sleep(3)
        elif times < 100:
            time.sleep(4)
        elif times < 150:
            time.sleep(5)
        else:
            time.sleep(6)

        next_height = driver.execute_script('

        if cur_height == next_height:
            same_cnt += 1
        else:
            same_cnt = 0

        if same_cnt >= 500:
            pass_product = True
            break

    if pass_product:
        break

    trade_history = [trade.text for trade in

    # Pass on to pipeline
    for trade in trade_history:
        item = KreamtradeItem()
        item['model_no'] = model_no
        item['trade'] = trade
        yield item

        del item

    except:
        continue

driver.quit()

```

(2) KREAM 체결내역 크롤러 (Scrapy - Pipeline)

```
import pymongo

class KreamtradePipeline:
    def process_item(self, item, spider):
        user = 'team04'
        pw = '1111'
        host = 'ec2-54-95-8-243.ap-northeast-1.compute.amazonaws.com'
        client = pymongo.MongoClient(f'mongodb://{user}:{pw}@{host}')
        db = client.resell

        data = {'model_no': item['model_no'], 'trade': item['trade']}
        db.trade.insert_one(data)

        return item
```

2-2 데이터 전처리

- 수집한 모든 데이터는 AWS에 연결된 **MongoDB**를 호출하여 데이터를 전처리하였다.
- 데이터 전처리는 **Spark, Pandas**를 이용하여 데이터 전처리를 하였고, 전처리한 모든 테이블들은 **MySQL**에 테이블 형태로 **DataWarehouse**를 구축하여 활용하였다.
- Science 팀에서 요청하는 자료가 있다면 MySQL에서 Join하여 데이터를 만들고, 전처리하여 제공하는 형태를 구성하였다.
- 'Product' 테이블의 Color 데이터를 수집하였으나 너무나도 다양한 색상컬러로 분류되어 있어, **K-means**를 통해 100개의 클러스터로 만들고 각 클러스터 별 평균 RGB값을 구해 그 평균 값의 색에 대한 레이블링 수작업을 진행하였다.

[1] Jordan 신발 추천 서비스

- 서비스를 위해 'Product', 'image_path' 테이블을 join하여 제품명과 이미지 url을 제공함으로써 모델 학습데이터로 활용하였다.

[2] Jordan 신발 신제품 가격 예측 서비스

- 'Product' 의 주요 요인들에서 Null값 제거, Datetime 변경 , ~원 등에 대한 전처리를 진행하였다.
- 제품명에서 뽑아낼수 있는 요인으로 Line, Type, Series, Collabo를 전처리한 후 One Hot Encoding 형태로 전처리하여 제공하였다.

[3] 유명한 인스타그램 Jordan 신발 게시물 알림 서비스

- Instagram에서 수집한 게시물 이미지로 Fast R-CNN 객체 인식에 적합한 이미지를 선별하였다.
- 이미지에 신발이 두개 이상있는 데이터는 제거한 후 모델 학습을 위해 제공하였다.

• 코드

(1) 전처리 파이프라인

```
import findspark
import pyspark.pandas as ps
import numpy as np
import pymongo
import pymysql
import re
from urllib.request import urlretrieve
from tqdm.notebook import tqdm
from PIL import Image

findspark.init()

user = 'team04'
pw = '1111'
host = 'ec2-54-95-8-243.ap-northeast-1.compute.amazonaws.com'
db_name = 'resell'
client = pymongo.MongoClient(f'mongodb://{user}:{pw}@{host}:27017')
db = client.resell

kream = ps.DataFrame(db.kream.find())
```

```

kream_detail = ps.DataFrame(db.kream_detail.find())

def num_preprocessing(data):
    if data == '':
        data = '0'

    if '.' in data:
        data = data.replace('만', '000')
    else:
        data = data.replace('만', '0000')

    data = data.replace('거래 ', '')
    data = int(data.replace('.', '').replace(',', '', ''))

    return data

def price_preprocessing(data):
    data = data.replace('-', '0')
    data = int(data.replace('원', '').replace(',', '', ''))

    return data

def release_price_preprocessing(data):
    filtered = re.findall('\d[0-9,]*원', data)
    if filtered:
        data = filtered[0].replace('원', '').replace(',', '', '')
    else:
        data = '0'

    return int(data)

def date_preprocessing(data):
    data = data.strip()
    if data != '-':

```

```

        if int(data[:2]) < 50:
            data = '20' + data
        elif int(data[:2]) >= 50:
            data = '19' + data

    data = data.replace('-', '1900/01/01')

    return data

def extract_collabo(data):
    p = re.compile(r"\bx\s\w+")
    filtered = p.findall(data)

    collabo = [word[2:] for word in filtered]

    return ', '.join(collabo)

def kream_preprocessing(df):
    df = df.fillna('0')
    df['like'] = df['like'].apply(num_preprocessing)
    df['post'] = df['post'].apply(num_preprocessing)
    df['all_trade'] = df['all_trade'].apply(num_preprocessing)
    df['price'] = df['price'].apply(price_preprocessing)

    return df

def kream_detail_preprocessing(df):
    df['model_no'] = df['model_no'].apply(lambda x: x.strip())
    df['rescent_price'] = df['rescent_price'].apply(price_preprocessing)
    df['instant_buy_price'] = df['instant_buy_price'].apply(price_preprocessing)
    df['instant_sell_price'] = df['instant_sell_price'].apply(price_preprocessing)
    df['release_price'] = df['release_price'].apply(release_price_preprocessing)
    df['release_date'] = df['release_date'].apply(date_preprocessing)
    df['release_date'] = ps.to_datetime(df['release_date'], format='%Y-%m-%d')

```

```
return df
```

```
def collabo_one_hot_encoding(df):
```

```
    df['collabo'] = df['_id'].apply(extract_collabo)
```

```
    df['Clot'] = df['collabo'].apply(lambda x: 1 if 'Clot' in
```

```
df['Fragment'] = df['collabo'].apply(lambda x: 1 if 'Fragi
```

```
df['Union'] = df['collabo'].apply(lambda x: 1 if 'Union'
```

```
df['Jayson Tatum'] = df['collabo'].apply(lambda x: 1 if '
```

```
df['J Balvin'] = df['collabo'].apply(lambda x: 1 if 'J' i
```

```
df['League of Legends'] = df['collabo'].apply(lambda x: 1
```

```
df['Paris Saint Germain'] = df['collabo'].apply(lambda x:
```

```
df['A Ma Maniere'] = df['collabo'].apply(lambda x: 1 if '
```

```
df['Trophy Room'] = df['collabo'].apply(lambda x: 1 if 'T
```

```
df['Travis Scott'] = df['collabo'].apply(lambda x: 1 if '
```

```
df['Maison Chateau Rouge'] = df['collabo'].apply(lambda x
```

```
df['Dior'] = df['collabo'].apply(lambda x: 1 if 'Dior' in
```

```
df['Nina Chanel Abney'] = df['collabo'].apply(lambda x: 1
```

```
df['Billie Eilish'] = df['collabo'].apply(lambda x: 1 if
```

```
df['Off White'] = df['collabo'].apply(lambda x: 1 if 'Off
```

```
df['SoleFly'] = df['collabo'].apply(lambda x: 1 if 'SoleF
```

```
df['Russell Westbrook'] = df['collabo'].apply(lambda x: 1
```

```
df['Supreme'] = df['collabo'].apply(lambda x: 1 if 'Supre
```

```
df['Levis'] = df['collabo'].apply(lambda x: 1 if 'Levis'
```

```
df['Patta'] = df['collabo'].apply(lambda x: 1 if 'Patta'
```

```
df['Aleali May'] = df['collabo'].apply(lambda x: 1 if 'Al
```

```
df['Social Status'] = df['collabo'].apply(lambda x: 1 if
```

```
df['Ghetto Gastro'] = df['collabo'].apply(lambda x: 1 if
```

```
df['Blue The Great'] = df['collabo'].apply(lambda x: 1 if
```

```
df['Lance Mountain'] = df['collabo'].apply(lambda x: 1 if
```

```
df['SNS'] = df['collabo'].apply(lambda x: 1 if 'SNS' in x
```

```
df['Facetasm'] = df['collabo'].apply(lambda x: 1 if 'Face
```

```
df['DSM'] = df['collabo'].apply(lambda x: 1 if 'DSM' in x
```

```
df['Kaws'] = df['collabo'].apply(lambda x: 1 if 'Kaws' in
```

```
df['Olivia Kim'] = df['collabo'].apply(lambda x: 1 if 'Ol
```

```
df['Titan'] = df['collabo'].apply(lambda x: 1 if 'Titan'
```

```
df['Melody Ehsani'] = df['collabo'].apply(lambda x: 1 if
```

```
df['Anna Wintour'] = df['collabo'].apply(lambda x: 1 if '
```

```

df['Converse'] = df['collabo'].apply(lambda x: 1 if 'Converse' in x else 0)
df['Vogue'] = df['collabo'].apply(lambda x: 1 if 'Vogue' in x else 0)
df['Serena Williams'] = df['collabo'].apply(lambda x: 1 if 'Serena Williams' in x else 0)
df['Carmelo Anthony'] = df['collabo'].apply(lambda x: 1 if 'Carmelo Anthony' in x else 0)
df['Carhartt'] = df['collabo'].apply(lambda x: 1 if 'Carhartt' in x else 0)
df['Sheila Rashid'] = df['collabo'].apply(lambda x: 1 if 'Sheila Rashid' in x else 0)
df['Pendleton'] = df['collabo'].apply(lambda x: 1 if 'Pendleton' in x else 0)
df['Rox Brown'] = df['collabo'].apply(lambda x: 1 if 'Rox Brown' in x else 0)
df['Nike'] = df['collabo'].apply(lambda x: 1 if 'Nike' in x else 0)
df['Comme des Garçons Homme'] = df['collabo'].apply(lambda x: 1 if 'Comme des Garçons Homme' in x else 0)
df['PSNY'] = df['collabo'].apply(lambda x: 1 if 'PSNY' in x else 0)

return df

```

```

def series_one_hot_encoding(df):
    df['OG'] = df['_id'].str.contains(pat='OG').astype(int)
    df['Retro'] = df['_id'].str.contains(pat='Retro').astype(int)
    df['SP'] = df['_id'].str.contains(pat='SP').astype(int)
    df['QS'] = df['_id'].str.contains(pat='QS').astype(int)
    df['SB'] = df['_id'].str.contains(pat='SB').astype(int)
    df['SE'] = df['_id'].str.contains(pat='SE').astype(int)
    df['Chicago'] = df['_id'].str.contains(pat='Chicago').astype(int)

    return df

```

```

def download_image(df):
    print('Image downloading...')

    df['img_path'] = df['model_no'].apply(lambda x: f'image/{x}.jpg')

    for idx, row in tqdm(df.iterrows()):
        urlretrieve(row['img_url'], row['img_path'])

    return df

```

```

def calculate_color_ratio(df):
    print('Calculating the color ratio...')

    conn = pymysql.connect(user=user, passwd=pw, host=host, db=db)
    query = 'SELECT R, G, B, color FROM rgb_color'
    rgb_color_df = ps.read_sql(query, conn)
    rgb_color_df = rgb_color_df.groupby(['R', 'G', 'B'])['color'].agg('count').reset_index()

    color_lst = ['갈색', '검정', '고동', '남색', '노랑', '모카', '빨강', '빨강', '살구', '아이보리', '연두', '자몽', '주황', '하늘', '황갈색', '황토', '회색', '흰색']
    color_df = ps.DataFrame(None, columns=['model_no'] + color_lst)

    for idx, row in tqdm(df.iterrows()):
        img_name = row['model_no'].replace('/', ' ')
        img = Image.open(f'image/{img_name}.png')
        img = img.resize((128, 128))

        img_arr = np.array(img)
        img_arr = np.reshape(img_arr, (-1, 4))

        img_df = ps.DataFrame(img_arr, columns=['R', 'G', 'B', 'alpha'])
        img_df = img_df.drop('alpha', axis=1)
        img_df = img_df[~((img_df['R'] == 0) & (img_df['G'] == 0) & (img_df['B'] == 0))]
        img_df = img_df.reset_index(drop=True)

        img_df = img_df.merge(rgb_color_df, on=['R', 'G', 'B'])

        color_cnt = img_df[img_df['color'].notna()][['R', 'color']]
        color_cnt = color_cnt[color_cnt['R'] >= 80]
        color_cnt['model_no'] = row['model_no']
        color_cnt = color_cnt.pivot(index='model_no', columns='color', values='count')
        color_df = ps.concat([color_df, color_cnt])

    color_df = color_df.fillna(0)
    df = df.merge(color_df, on='model_no', how='left')
    df[color_lst] = df[color_lst].div(df[color_lst].sum(axis=1), axis=1)

```

```

return df

def pipeline(kream, kream_detail):
    kream = kream_preprocessing(kream)
    kream_detail = kream_detail_preprocessing(kream_detail)
    kream_detail = collabo_one_hot_encoding(kream_detail)
    kream_detail = series_one_hot_encoding(kream_detail)
    kream_detail = download_image(kream_detail)
    kream_detail = calculate_color_ratio(kream_detail)

    return kream.merge(kream_detail, on='_id', how='inner')

```

2-3 기술적인 요소

- **Scienium,Scrapy**
 - 데이터 Crawling 진행 시 활용
- **MongDB**
 - DataLake로 Crawling한 Data를 데이터베이스로 활용
- **Pandas,Spark**
 - 대용량 데이터로 전처리로 활용
- **MySQL**
 - DateWareHouse와 DataMart 데이터베이스로 활용
- **K-mean**
 - 제품 이미지에서 색상을 추출 후 군집화하여 별도로 분류하여 컬러를 활용하였다.
- **SeaBorn,matplotlib**
 - 데이터분석 및 시각화
- **Airflow**
 - **Jordan 신발 신제품 가격 예측서비스**
 - 새로운 신제품 데이터를 Luck_draw에서 신규등록 발매정보를 주기적으로 크롤링 하게끔 자동화하였다.
 - **유명인 인스타그램 Jordan 신발 게시물 서비스**

- 셀럽의 계정을 모니터링하는 프로그램이 실행되고 게시물이 업로드 되면 이를 탐지하여 크롤링이 진행된다. 이렇게 가져온 사진은 Fast R-CNN 객체 인식을 통해 신발 유무를 탐지하고 신발 제품명을 판별하여 Slack으로 알림주게 되는 과정을 **Airflow** 통해 자동화하였다.

- **코드**

(1) 인스타그램 모니터

```
import pymongo
import re

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By

def get_chrome_driver():
    chrome_options = webdriver.ChromeOptions()
    chrome_options.headless = True

    driver = webdriver.Chrome(
        service=Service(ChromeDriverManager().install()),
        options=chrome_options
    )

    return driver

if __name__ == '__main__':
    user = 'team04'
    pw = '1111'
    host = 'ec2-54-95-8-243.ap-northeast-1.compute.amazonaws.'
    client = pymongo.MongoClient(f'mongodb://{user}:{pw}@{host}')
    db = client.resell

    db_img_url = list(db.insta.find({}, {'_id': 0, 'img_url':
    pattern = '/[\d_n]+\.'
    img_lst = [re.findall(pattern, dic['img_url'])[0] for dic
```



```

driver = get_chrome_driver()
driver.implicitly_wait(10)

driver.get('https://www.instagram.com/apfhda7/')

img_url = driver.find_element(By.CSS_SELECTOR, '._aagt').get_attribute('src')
img = re.findall(pattern, img_url)[0]

driver.quit()

if img not in img_lst:
    exec(open('insta_celeb_img.py', encoding='utf-8').read())

```

(2) 인스타그램 게시물 크롤러

```

import re
import time
import pymongo

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

def get_chrome_driver():
    options = webdriver.ChromeOptions()
    options.headless = True
    options.add_experimental_option('excludeSwitches', ['enable-logging'])
    driver = webdriver.Chrome(
        service=Service(ChromeDriverManager().install()),
        options=options
    )

```

```

        return driver

driver = get_chrome_driver()

try:
    url = 'https://www.instagram.com/'
    driver.get(url)
    driver.implicitly_wait(10)

    # 로그인
    driver.find_element(By.XPATH, '//*[@id="loginForm"]/div/d
    driver.find_element(By.XPATH, '//*[@id="loginForm"]/div/d
    driver.find_element(By.XPATH, '//*[@id="loginForm"]/div[1
    time.sleep(3)

    # 특정 셀럽 계정 접속
    user = 'apfhda7'
    url1 = 'https://www.instagram.com/apfhda7/'
    driver.get(url1)

    # 원하는 이미지 url 형태
    # ex) /294416712_395538335930284_918132169382424976_n.jpg
    pattern = '/[\d_n]+\.'

    # mongoDB 접속 정보
    client = pymongo.MongoClient('mongodb://team04:1111@ec2-5.
    db = client.resell
    db_img_url = list(db.insta.find({}, {'_id': 0, 'img_url':
    img_url_lst = [re.findall(pattern, dic['img_url'])[0] for

    t1 = driver.find_elements(By.CSS_SELECTOR, '._aabd._aa8k._
    url_lst = [t.get_attribute('href') for t in t1]

    stop = False
    for url in url_lst:
        driver.get(url)
        WebDriverWait(driver, 10).until(EC.presence_of_eleme

```

```

        bucket = set()
        while True:
            for el in driver.find_elements(By.CSS_SELECTOR, 'img'):
                url3 = el.get_attribute('src')

                img_name = re.findall(pattern, url3)[0]

                if img_name in img_url_lst:
                    stop = True
                    break

                if url3:
                    bucket.add(url3)

            if stop:
                break

            try:
                WebDriverWait(driver, 5).until(EC.element_to_be_clickable(
                    driver.find_element(By.CSS_SELECTOR, '._9zm2')))
                time.sleep(1)
            except:
                break

        if stop:
            break

        for img_url in bucket:
            data = {'user': user, 'img_url': img_url, 'new': 1}
            db.insta.insert_one(data)
    except Exception as e:
        print(e)
    finally:
        driver.quit()

```

(3) 슬랙 알림 봇

```

import os
from slack_sdk import WebClient
from slack_sdk.errors import SlackApiError

def send_slack():
    message_file = '/home/ubuntu/airflow/dags/resell/insta_ce
    if os.path.isfile(message_file):
        f = open(message_file, 'r', encoding='utf-8')
        message = f.read()
        f.close()
        os.remove(message_file)

    slack_token = 'xoxb-3199109467058-3851214835316-94m80
    client = WebClient(token=slack_token)

    try:
        response_msg = client.chat_postMessage(channel='4
        print(response_msg['ok'])
    except SlackApiError as e:
        print('Error: {}'.format(e.response['error']))

```

(4) Airflow dags

```

from datetime import datetime

from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator

from resell.insta_celeb_monitoring import slack

default_args = {
    "start_date": datetime(2022, 7, 26)
}

```

```

with DAG(
    dag_id='insta_alert_service',
    schedule_interval='@daily',
    default_args=default_args,
    tags=['insta', 'alert', 'slack'],
    catchup=False) as dag:

    insta_monitor = BashOperator(
        task_id='insta_monitor',
        bash_command='cd ~/airflow/dags/resell/insta_celeb_mo
        dag=dag
    )

    model_predict = BashOperator(
        task_id='model_predict',
        bash_command='cd ~/airflow/dags/resell/insta_celeb_mo
        dag=dag
    )

    slack_alert = PythonOperator(
        task_id='slack_alert',
        python_callable=slack.send_slack,
        dag=dag
    )

    insta_monitor >> model_predict >> slack_alert

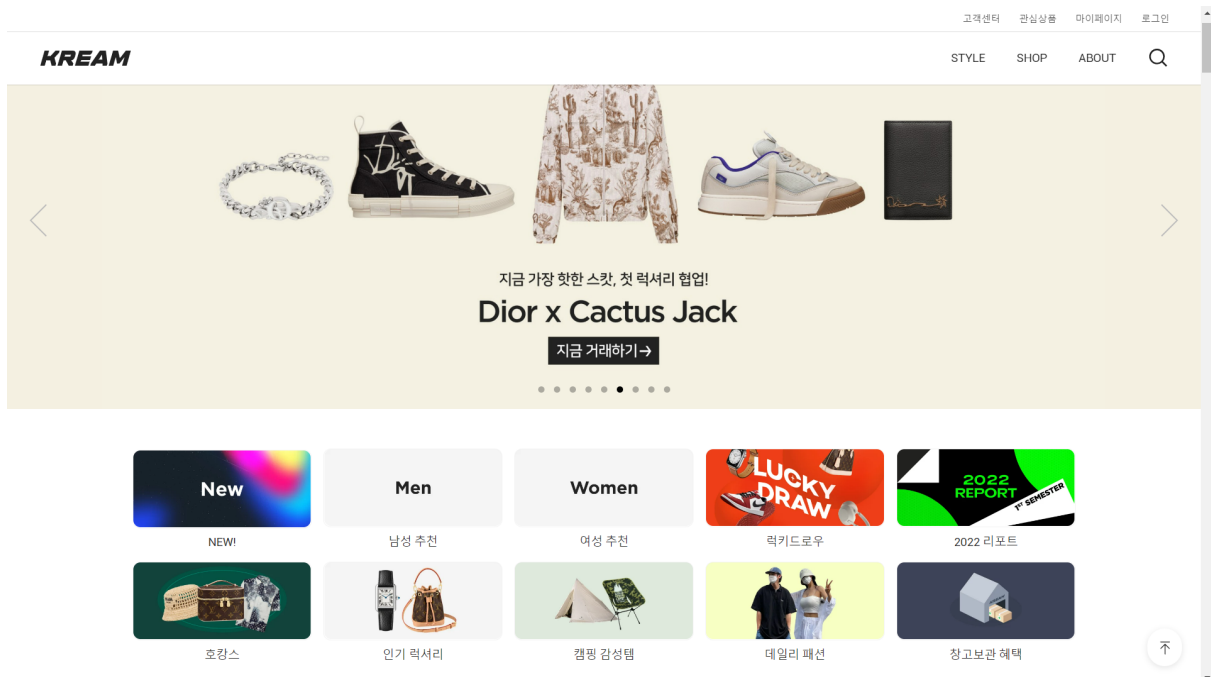
```

3. 프로젝트 모델 개발 (데이터 사이언스팀)

3-1 Jordan 신발 추천 서비스

- **서비스 타겟층**: 한정판 Jordan 신발을 처음 접하는 소비자

- **서비스 계획:** 현재는 **Test-Program** 형태로 구현된다. 추후에 KREAM 사이트 배너에 Jordan 운동화 추천 서비스 항목으로 위치하게 된다면, 해당 사이트를 방문하는 여러 소비자 중에서 Jordan 운동화에 관심은 있지만 처음 접하는 사람들에게 자신이 선호하는 디자인에 맞게 Jordan을 추천하는 서비스를 제공할 수 있다.



• 기술적인 설명:

◦ 입문자를 위한 Jordan 신발 추천 서비스에 Image Embedding 을 활용한 이유:

- Jordan을 처음 접하는 사람이 자신이 원하는 디자인의 신발을 찾기 위해 기존 필터링 방식의 조건식 나열을 하며 찾는다. 이 때 Jordan 운동화는 역사가 있는 만큼 세부적으로 많이 나뉘어 있기 때문에 Jordan에 대한 많은 시간과 정보를 필요로 한다.

따라서, 기존의 필터링 방식을 사용하지 않고 Data_Science적 사고로 접근해 본다.

- 각각의 Jordan 이미지의 지역적(디자인, 색깔 등)인 특성을 살린 동일한 512 차원의 벡터로 변환이 가능하다고 하였을 때, 3차원 이상의 벡터로 표현되어 시각적으로 표현할 수는 없지만 다차원 공간상에 각 차원들의 값으로 이루어진 점에 위치하게 되고 이로써 모든 신발을 512 차원 공간상에 점으로서 표현할 수 있게 된다.

그렇게 각 이미지가 나타내는 점들을 다차원 공간상의 유클리디안 거리가 가까운 것들끼리 군집을 진행하면, 모든 이미지로 학습된 같은 변환 층을 통과하며 벡터화된 것이므로 색깔과 디자인적으로 비슷한 부류의 신발들이 모이게 된다. 이를 통해 얻어진 각 군집에서, 각 군집을 대표하는 신발을 랜덤하게 샘플로 보

여주고 특정 군집을 선택하게 함으로써 많은 종류의 Jordan 신발 중 자신이 원하는 디자인에 가까운 군집에 속하는 신발들을 빠르게 얻을 수 있다.

이러한 작동 방식을 계속해서 적용해 나간다면 기존 필터링 방식으로 나열하여 원하는 디자인의 Jordan 신발을 찾는 것보다 편하고 빠르게 신발을 추천 받을 수 있게 된다.

이것이 Jordan 신발 추천 서비스에서 **Image Embedding** 을 활용하는 이유이다.

◦ 처리 프로세스

- KREAM에 존재하는 모든 Jordan 운동화 이미지를 가져온다
- Reference 모델로 vgg16 모델의 층 설계를 참고하여 fc 층을 Flatten과 Dense층 대신 GlobalAveragePooling2D 를 사용하여 2993개의 Jordan 이미지를 분류하는 모델을 설계 및 학습시킨다.
GlobalAveragePooling2D 층에 들어가는 input_shape이 (7, 7, 512) 이므로 층을 통과하여 나오는 output_shape은 (512)차원의 배열이 되므로 따로 Dense 층을 구성하지 않아도 초기 구상했던 512 차원의 벡터 형태로 얻을 수 있게 된다.
- 99.73% 이상의 정확도를 보이며 학습이 완료된 모델의 층을 활용하여 각각의 이미지를 Embedding(벡터화)한다.
(100%의 정확도를 갖지 못하는 이유를 확인하여 보니, KREAM 사이트 상에서 모델 번호는 다르지만 이미지가 같은 경우가 있었다 :
코드 부분에서 이미지가 겹치는 모델 확인)
- 변환된 Embedding Vector를 K-Means를 활용하여 여러 군집으로 분류한다.
이 때 실루엣 계수를 바탕으로 최적의 K(군집의 개수) 값을 구하고 추천 프로그램에 적용할 수 있도록 한다.
- 군집된 Jordan 모델들의 이미지를 확인하면서 사람이 인지하기에 명확하게 분류가 되었는지 판단한다.
이미지의 군집의 평가 지표로써 사용할 수 있는 것이 마땅치 않아 여러 사람의 주관적 판단의 평균으로 군집의 평가 지표를 대신한다.
- 분류가 잘 되지 않았다고 판단이 되면, 분류모델 단계로 넘어가서 층 구조 변경 등 이미지가 다른 방식으로 Embedding 될 수 있도록 수정한다.
- 분류가 잘 되었다고 판단이 되면, 변환된 Embedding Vector와 K-Means 군집을 활용하여 추천 프로그램을 작성한다.

• 코드

(1) 이미지 저장 코드

```
import pandas as pd
from tqdm import tqdm
import urllib
import os
import cv2

df = pd.read_csv('img_url3.csv')
df['model_no'] = df.model_no.str.strip()
df['model_no'] = df.model_no.str.replace('/', ' ')
df = df.sort_values(by='model_no').reset_index(drop=True)
df

file_path = "./image_224/"+df['model_no'].str[:]+"png"

df['file_path'] = file_path
df

# DataFrame을 csv 형태로 저장
df.to_csv('2993_dataframe.csv')

# url에서 이미지 저장하기

os.makedirs("./image", exist_ok=True)

for i, j in tqdm(zip(df.img_url, df.model_no)):
    if os.path.isfile(f'./image/{j}.png'):
        pass
    else:
        urllib.request.urlretrieve(i, f'./image/{j}.png')

# 저장된 이미지 224 resize 하기 PIL
from PIL import Image

os.makedirs("./image_224", exist_ok=True)

for i in tqdm(df.model_no):
```



```

img = Image.open(f'./image/{i}.png')
img = img.resize((224,224))
img.save(f'./image_224/{i}.png')

# cv2로 변환시 이미지에 오류 생김..
# cv2로 저장된 이미지 224 resize 하기

# os.makedirs("./image_224",exist_ok=True)

# for i in tqdm(df.model_no):
#     img = cv2.imread(f'./image/{i}.png')
#     img = cv2.resize(img, (224,224))
#     cv2.imwrite(f'./image_224/{i}.png', img)

```

(2) 이미지 분류 모델 학습 코드

```

from tensorflow.config import list_physical_devices
from tensorflow.config.experimental import set_memory_growth

physical_devices = list_physical_devices('GPU')
try:
    set_memory_growth(physical_devices[0], True)
except:
    # Invalid device or cannot modify virtual devices once in
    pass

import pandas as pd

df = pd.read_csv('2993_dataframe.csv')#, index_col="Unnamed: 0"
df = df.drop(columns='Unnamed: 0')

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1/255,)
test_datagen = ImageDataGenerator(rescale=1/255,)

```

```

train_generator = train_datagen.flow_from_dataframe(
    dataframe=df,
    x_col="file_path",
    y_col="model_no",
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='sparse',
    batch_size=64,
    shuffle=True,
    seed=42,
)

validation_generator = test_datagen.flow_from_dataframe(
    dataframe=df,
    x_col="file_path",
    y_col="model_no",
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='sparse',
    batch_size=64,
    shuffle=True,
    seed=24,
)

from tensorflow.keras.layers import Activation, BatchNormaliz
from tensorflow.keras import Model, Input
from tensorflow.keras.callbacks import ModelCheckpoint, Early
# from tensorflow.keras.optimizers import Adam
# from tensorflow.keras.metrics import Accuracy
# from tensorflow.keras.losses import SparseCategoricalCrosse

inputs = Input(shape=(224, 224, 3))

x = Conv2D(filters=32, kernel_size=(3,3), padding='same')(inp
x = BatchNormalization()(x)
x = Activation('relu')(x) # 224 224 32

```

```

x = Conv2D(filters=32, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPool2D(pool_size=(2, 2))(x) # 112 112 32

x = Conv2D(filters=64, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x) #112 112 64

x = Conv2D(filters=64, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPool2D(pool_size=(2, 2))(x) #56 56 64

x = Conv2D(filters=128, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x) # 56 56 128

x = Conv2D(filters=128, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPool2D(pool_size=(2, 2))(x) # 28 28 128

x = Conv2D(filters=256, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x) # 28 28 256

x = Conv2D(filters=256, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPool2D(pool_size=(2, 2))(x) # 14 14 256

x = Conv2D(filters=512, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x) # 14 14 512

x = Conv2D(filters=512, kernel_size=(3,3), padding='same')(x)
x = BatchNormalization()(x)

```

```

x = Activation('relu')(x)
x = MaxPool2D(pool_size=(2, 2))(x) # 7 7 512

x = GlobalAveragePooling2D()(x)

# GlobalAveragePooling2D 대신 아래의 모듈을 쓰기도 한다.
# x= tf.keras.layers.Flatten()(x)
# x = Dense(4096, activation='relu')(x)
# x = Dense(4096, activation='relu')(x)

outputs = Dense(2993, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

model.summary()

model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

# ModelCheckpoint Callback
model_checkpoint_cb = ModelCheckpoint(
    filepath = './model/2993_cls2.hdf5', # './model/weights.{
    monitor='val_loss',
    verbose=0,
    save_best_only=True,
    save_weights_only=False,
    mode='auto',
    save_freq='epoch'
)

# EarlyStopping Callback
early_stopping_cb = EarlyStopping(
    monitor='val_loss',
    min_delta=0,

```

```

        patience=10,
        verbose=0,
        mode='auto',
        restore_best_weights=True
    )

# Model fit
history = model.fit(
    x=train_generator,
    epochs=1000,
    verbose='auto',
    callbacks=[model_checkpoint_cb, early_stopping_cb],
    validation_data=validation_generator,
)

# 시각화

import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.plot(acc, label='Training acc')
plt.plot(val_acc, label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(loss, label='Training loss')
plt.plot(val_loss, label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()

```

```

# 정확도 측정

import numpy as np
from tqdm import tqdm
from keras.models import load_model
import cv2

# 모델 불러오기
model = load_model('./model/2993_cls2.hdf5')

dict1 = train_generator.class_indices

def index_predict(x):

    k = list(globals()['dict1'].keys())[list(globals()['dict1'].values()).index(x)]

    a = cv2.imread('./image_224/{0}.png'.format(k))
    a = cv2.cvtColor(a, cv2.COLOR_BGR2RGB)
    b = a.astype('float32')/255
    b = b.reshape((-1,224,224,3))
    preds = globals()['model'].predict(b,verbose=0)
    return preds.argmax()

# test
# index_predict(2146)

j = 0
for i in tqdm(range(2993)):
    k = index_predict(i)
    if i==k:
        j +=1
print("accuracy : " + str(j/2993*100) + "%")

from keras.preprocessing.image import ImageDataGenerator

test_datagen = ImageDataGenerator(rescale=1/255,)

test_generator = test_datagen.flow_from_dataframe(

```

```

        dataframe=df,
        x_col="file_path",
        y_col="model_no",
        target_size=(224, 224),
        color_mode='rgb',
        class_mode='sparse',
        batch_size=64,
        shuffle=False,
        seed=24,
    )

a = model.predict(test_generator)

# 잘못 분류한 이미지 확인 (KREAM 사이트 상에 다른 모델번호 동일한 이미지
for i in range(2993):
    if i != a[i].argmax():
        print(df.iloc[i,1])
        print(df.iloc[a[i].argmax(),1])
        print('\n')

```

(3) 이미지 분류 벡터화(임베딩) 코드

```

# load_model
from tensorflow.keras.models import load_model
model = load_model(
    filepath="./model/2993_cls2.hdf5", custom_objects=None, compile=True
)

# 모델의 층 개수 확인
try:
    for _ in range(50):
        model.get_layer(index=_)
except:
    print('모델의 층 개수는 '+str(_-1)+"개 입니다.")

model.get_layer(index = 37) # 마지막 2993 분류층 확인

```

```

# 분류층 제외하고 층 가져오기 ( GlobalAveragePooling2D 까지 output :
layer_lst = []

for i in range(37):
    layer_lst.append(model.get_layer(index=i))
# 레이어 개수 확인
len(layer_lst)

# 층을 통과하는 함수 만들기
def transform_to_vector(x):
    for i in layer_lst:
        x = i(x)
    return x

# cv2 테스트
import cv2

img = cv2.imread('./image_224/136027-089.png')
img = img.astype('float')/255
img = img.reshape(1, 224, 224, 3)
img.shape

# 현재 이미지가 저장되어있는 위치 확인 및 파일 확인
import os

model_lst = os.listdir('./image_224')
model_lst

# test img를 변환시켜 확인
a = transform_to_vector(img)
a.__array__()[0].shape

# 한 번 전처리가 진행된 엔지니어링 팀이 크롤링해온 csv를 DataFrame 형태로
import pandas as pd

df = pd.read_csv('2993_dataframe.csv')
df

```



```

# dict에 임베딩(벡터화된 array) 저장

from tqdm import tqdm

embedding_dict = dict()

for i, j in tqdm(zip(df.model_no, df.file_path)):

    img = cv2.imread(j)
    img = img.astype('float')/255
    img = img.reshape(1, 224, 224, 3)
    img = transform_to_vector(img)
    embedding_dict[i] = img.__array__()[0]

# dict에 임베딩 된 벡터 확인 테스트
embedding_dict.get('130207-061')

df2 = pd.DataFrame(embedding_dict).T
df2

# pickle을 활용하여 embedding_dict 객체를 pickle 파일로 저장
import pickle

with open('embedding_dict.pkl', 'wb') as f:
    pickle.dump(embedding_dict, f)

```

(4) Jordan 신발 추천 프로그램 "JordanRecommend"

```

class JordanRecommend:

    def __init__(self, k_num=8, opt=False, random_state=42):

        #         print('__init__ 실행')

        import pandas as pd

```

```

import pickle
from sklearn.cluster import KMeans

# 피클에서 df 불러오기
with open("jordan_embedding_dict_df.pkl", "rb") as f:
    self.__df = pickle.load(f)

# 변하지 않는 멤버 변수 초기화
self.df = self.__df.loc[:, 0:511] # 임베딩 벡터
self.df_ = self.__df.loc[:, ['model_code', 'file_path']]

# 변하는 멤버변수 초기화
self.dfm = self.df.copy()
self.df_m = self.df_.copy()

# k_num, opt, random_state 초기화
self.opt = opt
self.random_state = random_state

#         if self.opt == True:
#             self.silhouette()
#         else:
#             self.k_num = k_num
self.k_num = k_num

# kmean 초기화
self.kmeans = KMeans(n_clusters=self.k_num, random_state=self.random_state)
self.labels = self.kmeans.fit_predict(self.dfm)
self.df_m['cluster'] = self.labels # df_m 의 컬럼 ['inc

self.show_samples(self.dfm, self.df_m)

```

```

def show_samples(self, dfm, df_m):
#     print('show_samples 실행')
    import matplotlib.pyplot as plt
    from PIL import Image
    import random

    print('현재 k_num : '+str(self.k_num))
    print('현재 df_m 개수 : '+str(len(self.df_m)))

    self.dfm = dfm
    self.df_m = df_m

    # 현재 k_num 을 바탕으로 현재 df_m 에 존재하는 클러스터 별 샘플

    if self.k_num < len(self.df_m):

        fig, ax = plt.subplots(1, self.k_num)
        fig.set_size_inches(20, 2)

        for i in range(self.k_num): # 여기 위치에서는 k_num :

            df = self.return_cluster_num_df(i+1) # for 문

            img_loc = df.file_path.tolist()
            random.shuffle(img_loc)
            ax[i].axis('off')
            ax[i].set_title(f'{i+1}')
            img = Image.open(img_loc[0])
            ax[i].imshow(img)

        plt.show()

        self.recommend()

    else: # self.k_num >= len(self.df_m) 일 때
        print('*****here is my recomme

```

```

fig, ax = plt.subplots(1, len(self.df_m))
fig.set_size_inches(20, 2)

if len(self.df_m) == 1: # ax가 1개 이므로 ax[1] 라 표
    for j, k in zip(self.df_m['file_path'], self.d

        ax.axis('off')
        ax.set_title(k)
        img = Image.open(j)
        ax.imshow(img)

    plt.show()

else:

    for i, j, k in zip(range(len(self.df_m)), self.

        ax[i].axis('off')
        ax[i].set_title(k)
        img = Image.open(j)
        ax[i].imshow(img)

    plt.show()

print('\n*****')

return 0

```

```

def recommend(self):
#     print('recommend 실행')
    self.input1 = input('Enter the "number" of your most
    print('현재 k_num : ' + str(self.k_num))
    try:

        if int(self.input1) <= self.k_num:

```

```

        self.select(int(self.input1))

    else:
        print("\n\n*****")
        print("*****Please enter a")
        print("*****")

        self.recommend()

except:

    if self.input1 == 're':
        print('\n\n*****')
        print('*****reshuffling***')
        print('*****')
        self.show_samples(self.dfm, self.df_m)
        self.recommend()

    else:
        print('\n\n*****')
        print('*****Please enter i')
        print('*****')
        self.recommend()

def select(self, num):

    #         print('select 실행')
    import matplotlib.pyplot as plt
    from PIL import Image
    import random
    from sklearn.cluster import KMeans

    self.silhouette() # opt = True일 때 self.k_num 을 최적으

```

```

if self.k_num > len(self.df_m):
    self.show_samples(self.dfm, self.df_m)
else:

    self.df_m = self.return_cluster_num_df(num)
    self.dfm = self.df.loc[self.df_m.index]

    if self.k_num > len(self.df_m):
        self.show_samples(self.dfm, self.df_m)

    else:
        self.kmeans = KMeans(n_clusters=self.k_num, r
        self.labels = self.kmeans.fit_predict(self.dfm
        self.df_m = self.df_m.drop(columns=['cluster'
        self.df_m['cluster'] = self.labels
        self.dfm = self.df.loc[self.df_m.index]

        self.show_samples(self.dfm, self.df_m)

def return_cluster_num_df(self,num): # num에 해당하는 클러스터

    return self.df_m[self.df_m['cluster']==num-1] # -1 의미

def silhouette(self):

    import copy
    import numpy as np

    if self.opt==True:

```

```

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
import pandas as pd
import copy
average_score_dict = dict()

try:
    print('최적의 분류를 위해 silhouette 계수를 계산중입니다')
    for i in range(4,10):

        df0 = copy.deepcopy(self.dfm)
        kmeans = KMeans(n_clusters= i, random_state= 50)
        labels = kmeans.fit_predict(df0.loc[:,0:50])
        df0['cluster'] = labels

        # 모든 데이터의 평균 실루엣 계수값을 구함.
        average_score = silhouette_score(df0.loc[:,0:50])
        average_score_dict[f'{i}'] = average_score

    self.k_num = np.array(list(average_score_dict.items()))
    print('실루엣 후 k_num = ' +str(self.k_num))
except:
    pass

else:
    pass

```

- 테스트(구동 확인)

JordanRecommend(k_num=6,opt=True,random_state=50)



input : 1

1



2



3



4



5



input : 2

1



2



3



4



5



6



7



8



9



input : 3

378040-100



505835-106



384666-160



398614-140



537738-109



3-2 Jordan 신발 신제품 가격 예측 서비스

- **서비스 타겟층:** 한정판 Jordan 신제품 구매 희망자
 - **서비스 계획:** KREAM 사이트에서, Jordan - 235 & 270 size 신발의 현재까지 거래된 체결내역 데이터 중 '가장 최저가, 가장 최고가, 가장 최근에 체결된 가격' 정보를 크롤링해 온다. 이어 **XGBoost**를 활용하여 대략적인 가격을 예측해 결과를 반환할 수 있는 모델을 만든다.
 - **기술적인 설명:**
 - **XGBoost를 선택한 이유:**
 - (1) 크롤링한 데이터에는 결측치가 많지 않지만, 서비스가 확장 된다면 더 많은 결측치가 발생할 텐데 XGBoost에서는 이러한 결측치를 내부적으로 처리해 주는 기능이 있다는 점
 - (2) 과적합 규제가 가능하다는 점
 - (3) 성능과 자원 효율이 좋아서 구동이 빠르다는 점
- ⇒ 크게 3가지 이유로 XGBoost를 사용하기로 결정했다.
- **데이터셋 설명**

- **"kream_235_price (1).csv"** : KREAM 사이트의 Jordan 신발 체결내역 중
에서 신발 사이즈가 235인 데이터 (지금까지 거래 되었던 가장 최저가, 가장 최
고가, 가장 최근에 체결된 가격 정보 포함)
 - **데이터 사이즈**: 388 rows × 6 columns

 - **"kream_270_price (1).csv"** : KREAM 사이트의 Jordan 신발 체결내역 중
신발 사이즈가 270인 데이터 (지금까지 거래 되었던 가장 최저가, 가장 최고가,
가장 최근에 체결된 가격 정보 포함)
 - **데이터 사이즈**: 719 rows × 6 columns

 - **"kream_product2.csv"**: KREAM 사이트의 모든 Jordan 신발에 관한 정보
(모델명, 콜라보 one-hot encoding 상태, 색깔 등)
 - **데이터 사이즈**: 2993 rows × 116 columns
- **처리 프로세스**
- (1) 체결 내역 가격 데이터 상의 결측치를 제거한다.
 - (2) 각 데이터를 모델명으로 merge 한 후 새로운 데이터프레임을 생성한다.
 - (3) 종속변수를 가격으로, 독립변수를 시리즈, 콜라보, 색깔 등의 105개의 column
으로 설정한다.
 - (4) column 개수가 너무 많아서 다중공선성 문제가 발생할 수 있으므로 독립변수
들끼리 VIF를 구해 본다.
 - (5) VIF가 10 이상인 변수들("흰색", "Jordan 1", "검정", "밝은회색", "Retro",
"Jordan 4")을 제거해서 다중공선성 영향을 받지 않는 최종 모델을 완성시킨다.

• 코드

```
# 필요한 라이브러리 호출

import pandas as pd
import xgboost
import numpy as np
```

```

from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_infla

```

```

def whole_prediction(csv_path, pricewhat, model_no):
    df = pd.read_csv(csv_path)
    df1 = df.copy()
    df1_1= df1[df1[pricewhat] != 0].reset_index(drop=True)
    df3 = df1_1.copy()
    df_product = pd.read_csv("kream_product2.csv")
    df_product_C=df_product.copy()
    df_NEW=pd.merge(df3, df_product_C, on='model_no', how='in
    y_label = df_NEW[pricewhat]
    X_features = df_NEW.loc[:, "OG":]

    vif = pd.DataFrame()

    vif["VIF Factor"] = [variance_inflation_factor(X_features
    vif["features"] = X_features.columns
    vif1 = vif.sort_values(by="VIF Factor", ascending=False).

    del_list = vif1.loc[vif1['VIF Factor']>=10, "features"].t

    X_features1 = X_features.drop(del_list,axis=1)

    X_train, X_test, y_train, y_test = train_test_split(X_fea
    xgb_model = xgboost.XGBRegressor(n_estimators=100, learni
                                   colsample_bytree=1, max_depth=7)
    xgb_model.fit(X_train,y_train)

    if sum(df_product["model_no"]==model_no) >= 1:
        X_pred = df_product[df_product["model_no"]==model_no]

        print(np.unique(df_NEW["size"])[0], "사이즈 신발의", pri
        print(xgb_model.predict(X_pred))

    else:
        print("해당 모델에 대한 정보가 없어서 예측할 수 없습니다. 해당

```

- 테스트(구동 확인)

```
whole_prediction("kream_270_price (1).csv", "last price", "55
```

270 사이즈 신발의 last price 를 예측합니다
[348331.47]

- 결론

⚙️ XGBoost

	Min_Price	Max_Price	Lastest_Price
235	0.9966	0.9932	0.9919
270	0.9944	0.9939	0.9929

⚙️ VIF 후 XGBoost

	Min_Price	Max_Price	Lastest_Price
235	0.9921	0.9895	0.9881
270	0.9943	0.9886	0.9888

결론적으로, 미세한 수치를 보이며 예측력이 조금은 떨어지지만 다중공선성 문제가 해결되면서, 보다 개선된 모델이 완성된 것을 확인할 수 있다.

3-3 유명한 인스타그램 Jordan 신발 게시물 알림 서비스

- 서비스 타겟층: Jordan 신발을 재테크 목적으로 하는 리셀러
- 서비스 계획: 유명 인플루언서/연예인 인스타그램에 Jordan 신발을 포함한 사진이 올라오면 그 포스팅에서 신발 부분만 객체 인식을 하여 CNN 모델을 통해 학습된 모델을 통과하여 사진에 있는 신발이 Jordan의 어떤 모델인지 보여주며 Slack으로 모델명을 확인할 수 있는 서비스를 만든다.
- 기술적인 설명:
 - Object Detection Faster R-CNN을 선택한 이유:
 - (1) 기존의 R-CNN 기법에 비해서 시간적인 단점을 해결할 수 있다는 점

(2) 한 번의 forwarding만 수행하면 바로 어느 곳에 물체가 있는지 예측이 가능하다는 점

(3) Region Proposal을 위한 모든 연산을 GPU 상에서 가능하도록 한다는 점

⇒ 크게 3가지 이유로 **Faster R-CNN**을 사용하기로 결정했다.

◦ 데이터셋 설명

- **Jordan 신발 모델명 555088-105 사진**(Train data: 4901장, Validation data: 1226장)
- **Jordan 신발 모델명 555088-140 사진**(Train data: 4901장, Validation data: 1226장)
- **Jordan 신발 모델명 DM7866-140 사진**(Train data: 4901장, Validation data: 1226장)
- **Slipper 사진**(Train data: 4901장, Validation data: 1226장)

◦ 처리 프로세스

(1) Tensorflow 내에 있는 **객체 탐지 모델** 코드를 가져와서 사용 목적에 맞게 일부 함수를 변경한다.

(2) **객체 인식**을 통해 Footwear로 분류되는 부분 중 점수가 가장 높은 부분을 추출한다.

(3) 추출된 사진들을 (메모리 문제로 인하여) **100*100 사이즈**로 조정한다.

(4) 사진 데이터를 풍부하게 하기 위해서 **데이터 증강**을 실행하였고 원본 데이터를 (60, 120, 180, 240, 300, 360)도로 돌려 **데이터를 6배로 증강**했다.

(5) 증강된 데이터를 Random Shuffle 후 각각 train과 validation 폴더에 4901장, 1226장씩 넣어서 CNN 모델을 완성시킨다.

(6) 위의 과정을 **함수로 자동화**시켜 데이터 엔지니어가 **자동 알림 서비스**를 구축할 수 있도록 돕는다.

• 코드

(1) 자동화 함수 코드

```
# 텐서플로우 관련
```

```
import tensorflow as tf
import tensorflow_hub as hub
```

```
# 이미지 다운로드
```

```
import matplotlib.pyplot as plt
import tempfile
from six.moves.urllib.request import urlopen
from six import BytesIO
```

```
import pandas as pd
```

```
# 이미지에 그림 그리기
```

```
import numpy as np
from PIL import Image
from PIL import ImageColor
from PIL import ImageDraw
from PIL import ImageFont
from PIL import ImageOps
```

```
# 시간 관련
```

```
import time
```

```
# 이미지 보여 주기
```

```
def display_image(image):
    fig = plt.figure(figsize=(20, 15))
    plt.grid(False)
    plt.imshow(image)
```

```
# url
```

```
def download_and_resize_image(url, display=False):
    _, filename = tempfile.mkstemp(suffix=".jpg")
    response = urlopen(url)
    image_data = response.read()
```

```

image_data = BytesIO(image_data)
pil_image = Image.open(image_data)
pil_image = ImageOps.fit(pil_image, (pil_image.size[0], pil_image.size[1]))
pil_image_rgb = pil_image.convert("RGB")
pil_image_rgb.save(filename, format="JPEG", quality=90)
print("Image downloaded to %s." % filename)
if display:
    display_image(pil_image)
return filename

```

```

def draw_bounding_box_on_image(image,
                               ymin,
                               xmin,
                               ymax,
                               xmax,
                               color,
                               font,
                               thickness=4,
                               display_str_list=()):

    """Adds a bounding box to an image."""
    draw = ImageDraw.Draw(image)
    im_width, im_height = image.size
    (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
                                   ymin * im_height, ymax * im_height)
    draw.line([(left, top), (left, bottom), (right, bottom),
               (left, top)],
              width=thickness,
              fill=color)

    # If the total height of the display strings added to the top of the
    # box exceeds the top of the image, stack the strings below
    # instead of above.
    display_str_heights = [font.getsize(ds)[1] for ds in display_str_list]

```

```

# Each display_str has a top and bottom margin of 0.05x.
total_display_str_height = (1 + 2 * 0.05) * sum(display_s

if top > total_display_str_height:
    text_bottom = top
else:
    text_bottom = top + total_display_str_height
# Reverse list and print from bottom to top.

for display_str in display_str_list[::-1]:
    text_width, text_height = font.getsize(display_str)
    margin = np.ceil(0.05 * text_height)
    draw.rectangle([(left, text_bottom - text_height - 2
                    (left + text_width, text_bottom)],
                    fill=color)
    draw.text((left + margin, text_bottom - text_height -
              display_str,
              fill="black",
              font=font)
    text_bottom -= text_height - 2 * margin

def draw_boxes(image, boxes, class_names, scores, max_boxes=1
    """Overlay labeled boxes on an image with formatted score
    colors = list(ImageColor.colormap.values())

    try:
        font = ImageFont.truetype("/usr/share/fonts/truetype/
                                25)
    except IOError:
        print("Font not found, using default font.")
        font = ImageFont.load_default()

    for i in range(min(boxes.shape[0], max_boxes)):
        if scores[i] >= min_score:
            ymin, xmin, ymax, xmax = tuple(boxes[i])
            display_str = "{}: {}".format(class_names[i].dec

```

```

                                int(100 * scores[i]))
        color = colors[hash(class_names[i]) % len(colors)]
        image_pil = Image.fromarray(np.uint8(image)).convert('RGB')
        draw_bounding_box_on_image(
            image_pil,
            ymin,
            xmin,
            ymax,
            xmax,
            color,
            font,
            display_str_list=[display_str])
        np.copyto(image, np.array(image_pil))
    return image

```

```

def load_img(path):
    img = tf.io.read_file(path)
    img = tf.image.decode_jpeg(img, channels=3)
    return img

def run_detector(detector, path):

    img = load_img(path)

    converted_img = tf.image.convert_image_dtype(img, tf.float32)
    start_time = time.time()
    result = detector(converted_img)
    end_time = time.time()

    global result1

    result1 = {key:value.numpy() for key,value in result.items()}

    print("Found %d objects." % len(result["detection_scores"]))
    print("Inference time: ", end_time-start_time)

```



```

print(converted_img.shape)

image_with_boxes = draw_boxes(
    img.numpy(), result1["detection_boxes"],
    result1["detection_class_entities"], result1["detection_

```

인스타그램 사진 불러오고 저장

```

def insta_picture(image_url, save_path):

    downloaded_image_path = download_and_resize_image(image_u

    module_handle = "https://tfhub.dev/google/faster_rcnn/ope
    detector = hub.load(module_handle).signatures['default']

    run_detector(detector, downloaded_image_path)

    result_final = pd.DataFrame(result1["detection_boxes"])
    result_final.columns = ["ymin", "xmin", "ymax", "xmax"]
    img_color = Image.open(downloaded_image_path)
    result_final["xmin"] = round(result_final["xmin"]*img_col
    result_final["xmax"] = round(result_final["xmax"]*img_col
    result_final["ymin"] = round(result_final["ymin"]*img_col
    result_final["ymax"] = round(result_final["ymax"]*img_col
    result_final_array = result_final.values

    empty_df = pd.DataFrame(columns = ["ymin", "xmin", "ymax"

    for i in range(min(result1["detection_boxes"].shape[0], 1
        if result1["detection_class_entities"][i] == b'Footwe
            if result1["detection_scores"][i] > 0.1:
                ymin, xmin, ymax, xmax = tuple(result_final_a
                empty_df.loc[i] = [ymin, xmin, ymax, xmax, re

    empty_df = empty_df.sort_values(by="score", ascending=Fal

```

```

empty_df = empty_df.astype({'ymin': 'int'})
empty_df = empty_df.astype({'xmin': 'int'})
empty_df = empty_df.astype({'ymax': 'int'})
empty_df = empty_df.astype({'xmax': 'int'})

try:
    xmin = empty_df.loc[0, "xmin"]
    ymin = empty_df.loc[0, "ymin"]
    xmax = empty_df.loc[0, "xmax"]
    ymax = empty_df.loc[0, "ymax"]

    croppedImage=img_color.crop((xmin, ymin, xmax, ymax))
    croppedImage.save(f"{save_path}.png", 'png')

except (ValueError, KeyError):
    pass

```

이미지 조정

```

def image_resize(save_path):
    im = Image.open(f'{save_path}.png')
    im = im.resize((100, 100))
    im.save(f'{save_path}.png')

```

모델 이름 반환

```

import cv2
from tensorflow.keras.models import load_model
import numpy as np

def model_predict(save_path):
    model = load_model('objection_detection_100_12.h5')
    image = cv2.imread(f"{save_path}.png")
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    test_data = image / 255.0
    pred = model.predict(test_data[np.newaxis, ...])

```

```
list = ["555088-105", "555088-134", "DM7866-140", "slippe"]
print(list[np.argmax(pred)])
```

(2) CNN 모델 코드

```
import os

base_dir = './insta_img_realfinal/'

# 훈련, 검증, 테스트 분할을 위한 디렉토리 설정
train_dir = os.path.join(base_dir, 'train')

train_555088105_dir = os.path.join(train_dir, '555088-105')
train_555088134_dir = os.path.join(train_dir, '555088-134')
train_DM7866140_dir = os.path.join(train_dir, 'DM7866-140')
train_slipper_dir = os.path.join(train_dir, 'slipper')

validation_dir = os.path.join(base_dir, 'validation')

validation_555088105_dir = os.path.join(validation_dir, '555088-105')
validation_555088134_dir = os.path.join(validation_dir, '555088-134')
validation_DM7866140_dir = os.path.join(validation_dir, 'DM7866-140')
validation_slipper_dir = os.path.join(validation_dir, 'slipper')
```

```
from keras import layers
from keras import models
from tensorflow.keras import optimizers
from keras.preprocessing.image import ImageDataGenerator

model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(3, 3), activation='relu'))
# model.add(layers.Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.MaxPooling2D((2, 2)))
```

```
# model.add(layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
# model.add(layers.Dropout(0.2))
# model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(4, activation='softmax'))
```

```
from keras import layers
from keras import models
from tensorflow.keras import optimizers
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(100, 100),
    color_mode = "rgb",
    batch_size=64,
    class_mode='sparse')

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(100, 100),
    color_mode = "rgb",
    batch_size=64,
    class_mode='sparse')
```

```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimizers.RMSprop(learning_rate=0.001),
              metrics=['acc'])
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# ModelCheckpoint Callback
model_checkpoint_cb = ModelCheckpoint(
    filepath = 'objection_detection_100_12.h5', # './model/weights.h5'
    monitor='val_acc',
    verbose=0,
    save_best_only=True,
    save_weights_only=False,
    mode='auto',
    save_freq='epoch'
)

# EarlyStopping Callback
early_stopping_cb = EarlyStopping(
    monitor='val_acc',
    min_delta=0,
    patience=10,
    verbose=0,
    mode='auto',
    restore_best_weights=True
)

# Model fit
history = model.fit(
    train_generator,
    epochs=100,
    validation_data=validation_generator,
    verbose='auto',
    callbacks=[model_checkpoint_cb, early_stopping_cb]
)
```

- **결론**

: Validation Accuracy 기준 90.72% 정도 분류되는 모델을 구현할 수 있었고, 이 모델을 데이터 엔지니어링 팀에게 전달해서 원활히 알림 서비스 구현이 될 수 있도록 도왔다.

3-4 번외- 기준치에 도달하지 못한 모델

- **Jordan 신발 추천 서비스**

- **기술적인 설명:**

- **군집화에 K-means를 사용한 이유:** DBSCAN을 사용하면 자동으로 클러스터링을 진행해 주지만 이미지를 Flatten 시켜 feature의 개수가 (100*100*4)의 array로 되어 있으면 feature 개수가 너무 많아 DBSCAN을 사용할 수 없다고 판단했다. 따라서, K-means로 분석하고 실루엣 스코어를 구해서 최적의 군집 개수를 파악하여 군집화를 진행하고자 했다.
 - **데이터셋 설명:** Kream 사이트에 있는 Jordan 신발 2993개의 측면 사진 1장씩 (.png 형태)
 - **처리 프로세스**
 - (1) 모든 사진의 크기를 (100*100)으로 재조정한다.
 - (2) 사진을 Numpy의 array 형태로 변환 후 저장한다.
 - (3) 저장된 array를 불러와 (100*100*4)의 array로 reshape 후 각각의 feature들을 K-means 모델에 넣는다.
 - (4) 실루엣 스코어를 계산해서 8개의 군집으로 나눈다.

- **코드**

```
# 파일 이름 가져오기

import os
import numpy as np
from PIL import Image

image_path = './image6/'

img_list = os.listdir(image_path) # 디렉토리 내 모든 파일 불러
```

```
img_list_jpg = [img for img in img_list if img.endswith(".
print("img_list_png: {}".format(img_list_png))
```

```
# 모든 사진을 numpy array 형태로 저장
```

```
image_path = './image6/'
```

```
img_list_np = []
```

```
for i in img_list_PNG:
    img = Image.open(image_path + i)
    img_array = np.array(img)
    img_list_np.append(img_array)
    print(i, " 추가 완료 - 구조:", img_array.shape) # 불러온
    print(img_array.T.shape) #축변경 (색X가로X세로)
```

```
img_np = np.array(img_list_np) # 리스트를 numpy로 변환
np.save('./image224size_test/test.npy', img_np) # x_save.
print(img_np.shape)
print("결과: 정상적 으로 저장 완료")
```

```
# K-means 활용하기
```

```
import numpy as np
```

```
sneakers_data = np.load('./image224size_test/test.npy')
```

```
sneakers_data_2d = sneakers_data.reshape(-1, 100 * 100 * 4)
```

```
from sklearn.cluster import KMeans
```

```
km = KMeans(n_clusters=8)
```

```
km.fit(sneakers_data_2d)
```

```

# 분류가 잘 되었는지 사진으로 확인

import matplotlib.pyplot as plt

def draw_sample_data(arr, ratio=1):
    n = len(arr)

    rows = int(np.ceil(n / 10))

    cols = n if rows < 2 else 10
    fig, axs = plt.subplots(rows, cols,
                             figsize=(cols * ratio, rows *
    for i in range(rows):
        for j in range(cols):
            if i * 10 + j < n:
                axs[i, j].imshow(arr[i * 10 + j], cmap='gr
            axs[i, j].axis('off')
    plt.show()

draw_sample_data(sneakers_data[km.labels_ == 0])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 1])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 2])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 3])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 4])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 5])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 6])
print("-----")
draw_sample_data(sneakers_data[km.labels_ == 7])

```

- 문제점

(100*100*4)의 array로 되어 있기 때문에 feature의 개수가 너무 많고 특징 픽셀을 잡아내기 어렵다. 군집화가 되긴 하지만 정밀한 군집이 어렵다는 걸 draw_sample_data를 돌린 후 육안으로 확인할 수 있었다.

- **대안**

Image Embedding을 활용하여 저차원 공간에서 유사한 데이터를 가깝게 배치해 정밀한 군집이 가능한 모델을 생성하였다.

⇒ [1] Jordan 신발 추천 서비스 최종 모델

- **Jordan 신발 기존 제품 가격 예측 서비스**

초기에 서비스를 기획할 때는 신제품 가격 예측과 기존에 있던 제품의 가격 예측을 동시에 진행하고자 했다.

체결내역 데이터는 시간에 걸쳐 순차적으로 기록된 데이터 형태를 보이므로 **시계열 데이터**라고 생각하였고 따라서, 시계열 데이터 모델 중에 유명한 ARIMA와 PROPHET 모델을 사용해서 신제품 가격을 예측해보자고 기획을 했었다.

- **데이터셋 설명:** Kream 사이트 내에서 거래 되는 체결내역 중 신발 사이즈 235, 270에 해당하는 데이터들 (261928 rows × 4 columns)
→ 분석의 편의상 체결내역 데이터 중 **전체 체결내역이 가장 많은 모델을 추출하여** 사이즈가 270인 데이터만 활용해서 **ARIMA와 PROPHET**을 적용했다.



(1) ARIMA 모델

- **처리 프로세스**

(1) 체결내역 데이터 중 전체 체결내역이 가장 많은 모델(555088-105)을 추출하여 사이즈가 270인 데이터만 뽑는다.

(2) 적절한 **ARIMA**의 **p, d, q** - 수를 결정하기 위해서 ACF와 PACF 그래프, 차분 그래프를 그리고 분석한다.

(3) 이 과정을 **자동화**하여 최적의 p, d, q를 찾은 후 그 모델을 기준으로 미래 가격을 예측해 본다.

- **코드**

(1) 사이즈가 270인 데이터만 정리

```
df = pd.read_csv('./forARIMA.csv')
df1= df.copy() #사본 만들고
df1.groupby("model_no")[["price"]].count().sort_values(by="
df_test = df1[df1["model_no"] == "555088-105"].reset_index(
cols = ["trade_date", "price"]
df_test = df_test[cols]
df_test_new = df_test.copy()
df_test_new = df_test_new.drop_duplicates(["trade_date"], k
```

(2) ACF와 PACF, 차분 그래프

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from math import sqrt

%matplotlib inline

np.set_printoptions(formatter={'float_kind': '{:f}'.format})

sns.set(rc={'figure.figsize':(8, 6)})

import itertools
import warnings

import datetime
warnings.filterwarnings('ignore')

from scalecast.Forecaster import Forecaster
import pandas_datareader as pdr

f = Forecaster(y=df_test_new['price'], current_dates=df_test
```

```
f.plot_acf()
f.plot_pacf()
plt.show()

fig, ax = plt.subplots(2,1, figsize=(8,8))
sm.graphics.tsa.plot_acf(df_test_new['price'].diff().dropna())
sm.graphics.tsa.plot_pacf(df_test_new['price'].diff().dropna())
plt.show()
```

(3) ARIMA 최적의 모델 찾기 및 예측

```
df_test_new_datetime = df_test_new.copy()
df_test_new_datetime["trade_date"] = pd.to_datetime(df_test_new["trade_date"])
df_test_new_datetime.set_index("trade_date", inplace=True)

from sklearn.model_selection import train_test_split

train_df, test_df = train_test_split(df_test_new_datetime,
df_final = train_df.resample("D").last())

model = sm.tsa.arima.ARIMA(df_final, order = (3,1,0), freq="D")
model_fit = model.fit()
model_fit.summary()
```

```
p = range(0,4)
d = range(0,4)
q = range(0,4)
pdq = list(itertools.product(p, d,q))

aic = []
for i in pdq:
    model = sm.tsa.arima.ARIMA(df_final, order = (i))
    model_fit = model.fit()
    print(f'ARIMA: {i} >> AIC : {round(model_fit.aic,2)}')
    aic.append(round(model_fit.aic,2))
```

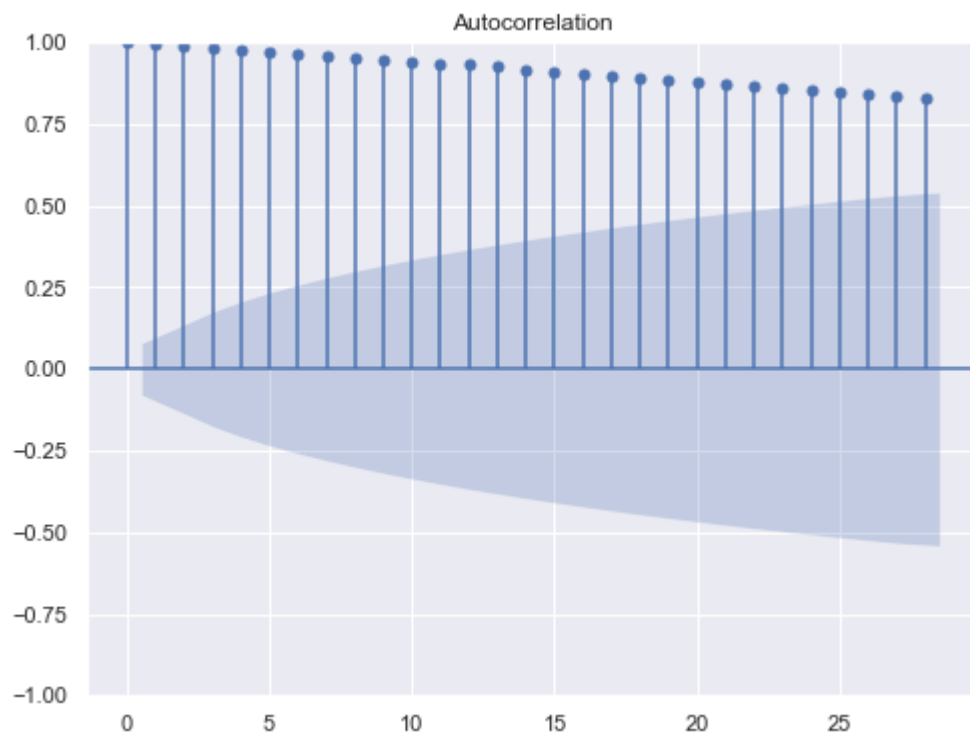
```
# Search optimal parameters
optimal = [(pdq[i],j) for i, j in enumerate(aic) if j == min(aic)]
optimal
```

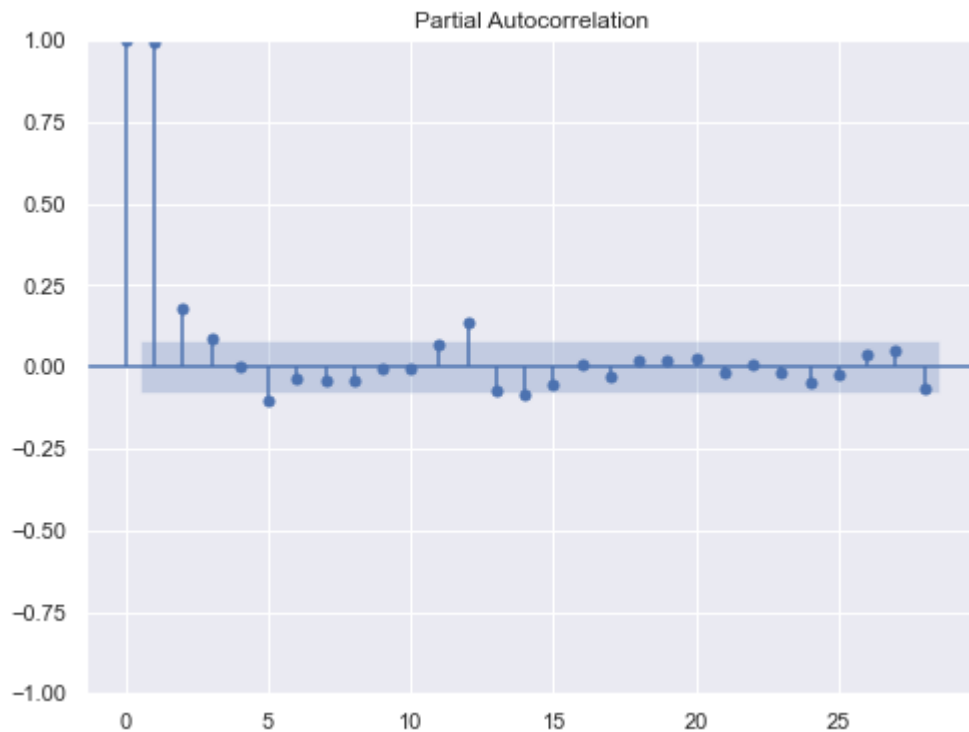
```
model = sm.tsa.arima.ARIMA(df_final, order = optimal[0][0],
model_fit = model.fit()
model_fit.summary()
```

```
model_fit.resid
```

- 해석

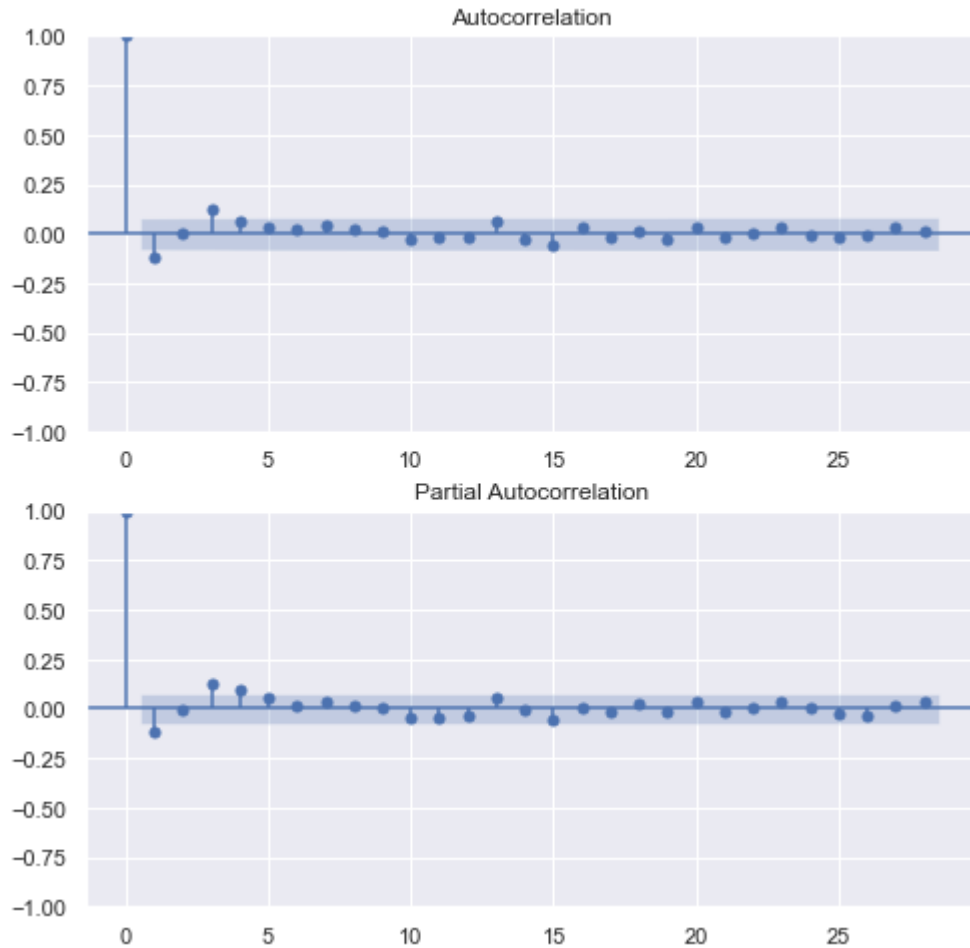
— <ARIMA의 p, q 결정>





ARIMA의 **p, q**를 확인하기 위하여 **ACF**와 **PACF 그래프**를 그렸다. 확인 결과 ACF 그래프는 점점 하락하는 형태를 보이고, PACF 그래프는 95%의 신뢰 구간으로 추정된 표준편차 안에 들어가지 않는 것을 확인했을 때 '**3**'의 시차를 보여서 **AR(3)** 모델을 사용하면 되겠다고 판단했다.

— <ARIMA의 d 결정>



ARIMA의 **d**를 확인하기 위해서 **1차 차분**을 진행했는데, 1차 차분 결과 변동성이 별로 없는 그래프로 확인되어 **정상성이 만족되었다**고 생각해서 **ARIMA(3, 1, 0)** 모델을 선정하였다.

— <ARIMA 모델 성능 검증>

ARIMA 모델의 성능을 검증하는 것에서 제일 중요한 요소는 **AIC**이다. AIC가 낮을수록 성능이 좋다고 판단할 수 있기에, AIC를 자동적으로 확인하는 코드를 작성한 후 p, d, q 각각에 0, 1, 2, 3을 넣어서 총 64번을 돌렸다. 확인 결과 **ARIMA(2, 2, 3)**이 제일 AIC가 낮은 것으로 확인하여 저 모델을 바탕으로 가격을 예측했다.

• 결론

해당 모델은 정상성 검증을 했더니 **잔차가 너무 크게 나오는 현상을 띄며** 기준치 미달의 결과를 보였다. 이는 **크게 2가지**로 해석할 수 있는데, **(1)** Kream 사이트에서 매일 해당 Jordan 제품이 체결되는 것이 아니기에 **중간중간 비어 있는 데이터**가 너무 많았다는 이

유와 **(2)** 같은 날에 이루어진 체결 내역을 **중복 제거**하여 모델에 사용된 체결 수가 609 개밖에 되지 않은 점 때문이다.

- **대안**

ARIMA 모델보다 비교적 최근에 개발된 시계열 모델인 Prophet 모델은 좀 더 간단한 알고리즘으로 작동하기 때문에 이 모델을 통해서 신제품 가격을 예측할 수 있는지 확인해 본다.



(2) PROPHET 모델

Prophet 모델은 트렌드와 주기적 특성뿐 아니라 예외적이고 이벤트와 같은 휴가철 상황까지도 모델링할 수 있게 되어 있다. 이 모델은 날짜 정보와 예측할 y만 있으면 되기 때문에 간편하면서 성능이 좋고, **1)** 시간별, 일별, 주별 기록 데이터 **2)** 계절성을 띄는 데이터 **3)** 불규칙한 이벤트 등을 예측하는 데에 유용하다.

따라서, 비교적 불규칙한 이벤트를 보이는 4조의 NIKE Jordan 데이터는 ARIMA보다는 **Prophet**을 통한 예측이 조금 더 적합했다.

- **처리 프로세스**

(1) 체결내역 데이터 중 전체 체결내역이 가장 많은 모델(555088-105)을 추출하여 사이즈가 270인 데이터만 뽑는다.

(2) 데이터 형태를 **ds**와 **y** - 두 개의 변수만 있도록 조정한다.

(3) 주기를 30일로 잡고 **PROPHET**을 사용해서 분석해 본다.

- **코드**

```
import pandas as pd
from prophet import Prophet

df = pd.read_csv('./forARIMA.csv')
df1= df.copy() #사본 만들고
df1.groupby("model_no")[["price"]].count().sort_values(by="price")
df_test = df1[df1["model_no"] == "555088-105"].reset_index(
    cols=["trade_date", "price"])
```

```
df_test = df_test[cols]
df_test.columns = ["ds", "y"]
```

```
m = Prophet()
m.fit(df_test)
```

```
future = m.make_future_dataframe(periods=30)
```

```
forecast = m.predict(future) #이후 30일 간의 데이터를 예측
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

```
fig1 = m.plot(forecast)
```

```
fig2 = m.plot_components(forecast)
```

```
df_test_final = df_test.copy()
df_test_final["ds"] = pd.to_datetime(df_test_final["ds"])
```

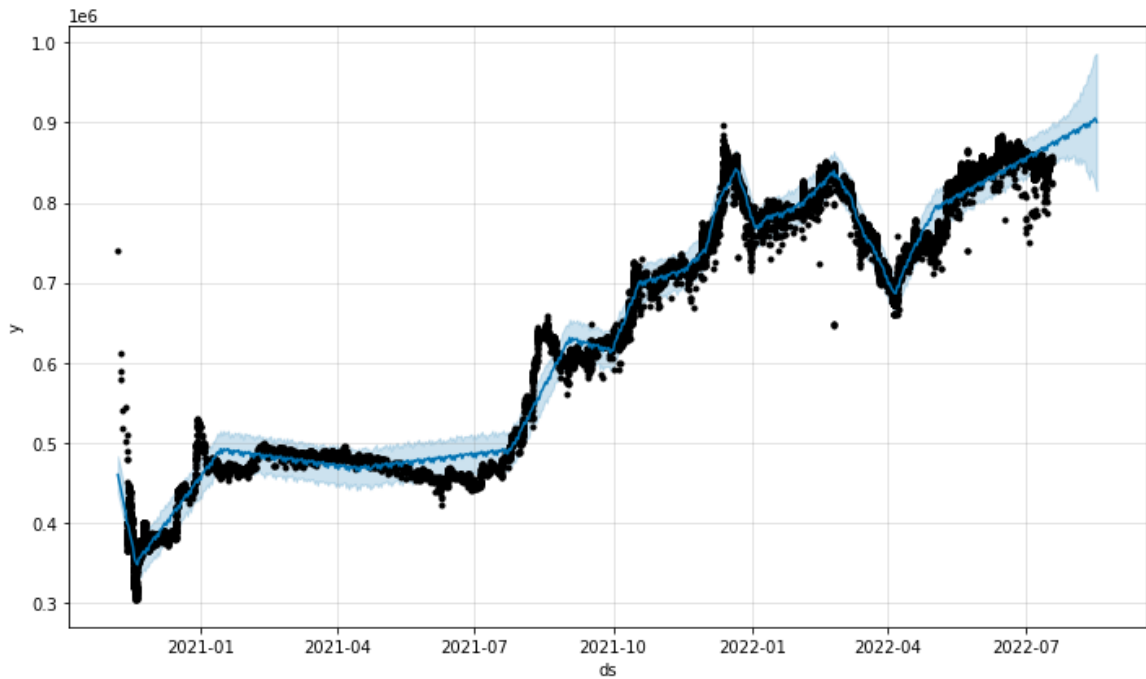
```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12,6))
plt.plot(df_test_final["ds"], df_test_final["y"], label="real")
plt.plot(forecast["ds"], forecast["yhat"], label="forecast")
plt.grid(True)
plt.legend()
plt.show()
```

```
# 실제 데이터와 예측 데이터의 그래프를 한번에 볼 수 있도록 함께 나열한 그래프
```

- **해석**

— <마지막 체결 데이터부터 이후 30일 동안의 가격 예측>



마지막 체결 가격 즈음부터 시작되는 **파란색 줄과 음영으로 나타난 부분이 예측된 미래 가격으로, 상승세**를 보이고 있다.

• 결론

ARIMA 모델로는 실패했던 가격 예측이 비교적 더 잘 나오는 모습을 살펴 볼 수 있다. 하지만 ARIMA 모델과 비슷하게 Prophet 모델도 장기간의 과거 데이터가 필요한 모델이기 때문에 2020년에 만들어진 KREAM 사이트의 데이터로는 부족했다. 결과적으로, 시간이 흘러 KREAM 사이트에 **최소 10년 이상의 데이터**가 더 쌓인다면 지금보다 좋은 결과를 낼 수 있을 것으로 보인다.

4. 프로젝트 향후 개선점

[데이터 엔지니어링팀]

- 현재 조던 신발에 한해서만 알림 서비스를 구현하고 있으나 다른 신발 종류에 대해서도 서비스 확대가 가능하다.
- 현재 인스타그램 셀럽이 한 명만 지정되어 있으나, 여러 인원에게 대해서도 지정이 가능하여 서비스 확대가 가능하다.

[데이터 사이언스팀]

- **Jordan 신발 추천 서비스**

(1) 신발 추천 프로그램 중간 군집에서 원하는 신발이 나왔을 때 작동을 멈추고 마지막으로 보여줬던 샘플들의 모델명을 표시하는 기능을 추가하면 이상적일 것 같습니다.

(2) 샘플사진 모델 중 하나 말고 두 개 이상을 선택할 수 있는 기능을 추가하면 좋을 것 같습니다.

- **Jordan 신발 신제품 가격 예측 서비스**

KREAM 사이트가 개설된 지 얼마 되지 않았기에 데이터양이 한참 부족해서 모델 성능 (R-squared)이 평균적으로 0.98 정도로 나온 부분이 아쉬웠다. 향후 시간이 흘러 데이터가 더 많이 쌓인다면 단순히 R-squared가 높은 모델이 아닌 좀 더 일반화된 모델이 만들어지지 않을까 예상한다.

- **유명인 인스타그램 Jordan 신발 게시물 알림 서비스**

(1) 전이학습을 이용하여 직접 만든 레이어 층보다 성능이 더 좋은 모델을 만들 수 있을 것 같다.

(2) 현재는 **메모리 문제, 프로젝트 기간 상의 문제로** (100*100 size)의 이미지를 4가지 클래스로만 학습을 시켰다. 하지만 여유가 된다면 (224*224 size)의 이미지를 KREAM 사이트 내에 있는 Jordan 신발의 개수만큼 더 큰 클래스로 확장시켜 좀 더 많은 신발을 분류할 수 있는 알림 서비스로 발전할 것을 기대한다.

5. 개발 후기

김예은

처음에는 한 달이라는 시간이 길어서 한 달 동안 많은 것을 할 수 있을 거라고 생각했는데 생각보다 너무나도 촉박한 일정이라서 추가적으로 작업하는 경우가 많아서 스트레스를 많이 받았던 것 같습니다. 전체적으로 정리하는 시기에 오니까 이렇게 하면 좋지 않았을까 하는 것들이 눈에 보여서 조금 아쉽지만 초반에 우려했던 부분이 생각보다는 잘 풀려서 다행스러운 부분도 있었습니다.

모두 고생 많으셨고, 다들 열심히 해 주셔서 감사합니다!!

안녕히 계세요 여러분! 전 이 세상의 모든 굴레와 속박을 벗어 던지고 제 행복을 찾아 떠납니다!
다! 여러분도 행복하세요~~!



 **김홍민**

길지도 짧지도 않은 프로젝트 기간 중 건강으로든, 심적으로든 이런 저런 힘든 점들이 많았을텐데 다들 너무 고생하셨습니다. 그래도 7월 29일이라는 날이 오니 신기하기도, 후련하기도 하네요.

다른 모든 분들도 모두 고생하셨고 오늘 이 과정이 마무리되는 순간 다들 하루의 마무리가 좋은 순간이길 바라요

고생하셨습니다.

 **박상권**

초반에 주제 선정에 어려움이 있었지만 결과론적으로 이는 데이터 엔지니어와 사이언스가 더 많은 것을 보여주기 위한 조원들의 고민과 열정이 넘쳤기 때문인 것 같다.

팀간에 협업을 통해 소통에 대한 것도 많이 배운것 같다. 끝까지 최선을 다해준 조원들과 조장에게 감사를 표합니다.

박선재

이론과 실전은 다르다는 것을 느낄 수 있었습니다.

힘들기도 했지만 조원분들 덕분에 재밌었고 무사히 끝마칠 수 있었던 것 같습니다.


마지막까지 열심히 달려주신 조원분들께 감사드립니다.

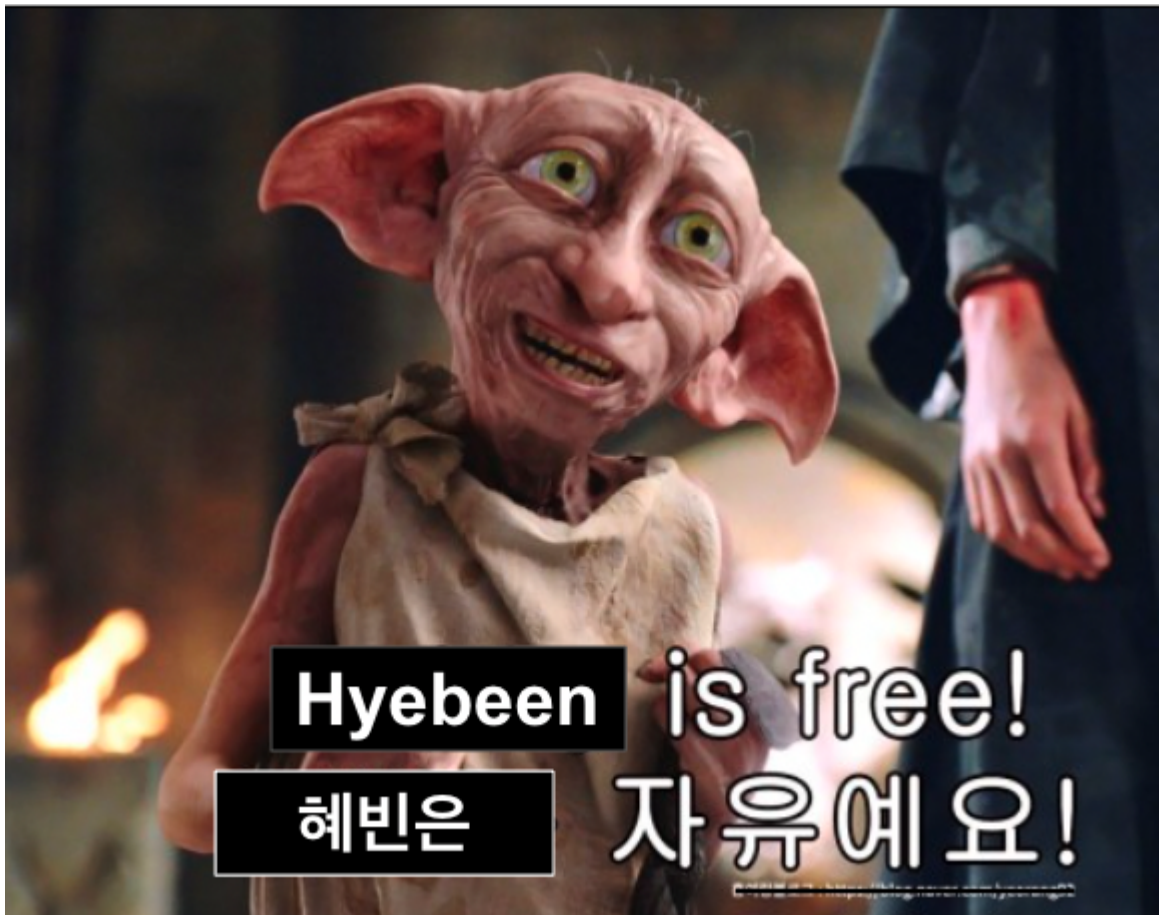
신혜빈

진짜 어떻게 버티다보니 끝이 오는구나...

이번 프로젝트는 현 과정의 대미를 장식할 최종 프로젝트이다 보니 모두가 부담감도 크고 스트레스를 받았을텐데, 그런 상황에서도 많이 부족한 저를 잘 이끌어주고 이해해주신 4조 4ir Jordan 팀에게 감사합니다! (심지어 초중반에 컨디션 난조로 자리를 비우는 상황에도 배려해주셔서 정말 감동이었습니다.)

이번 최종 프로젝트의 기간이 한 달정도로 제일 긴 만큼 공부가 많이 되었습니다. 맨땅에 헤딩 수준으로 새로 해 본 모델도 있고... 다양한 시도와 도전을 해 본 마지막 프로젝트였습니다.

 팀원분들 정말 고생 많으셨고 맛있는 거 드세요, 수료를 축하드려요👏!!! 건강이 최고... 건강 챙기시고 앞으로 여러분 취준 또는 학업도 응원할게여:)! 🍀🌹



이재연

프로젝트 조장을 맡게 되어 부담감이 적지 않았지만 나름대로 스스로 많이 성장한 것 같아서 과정을 잘 마친 것에 감사한 마음이다. 코드를 조금 더 효율적으로 짜는 방법, 오류를 읽고 해결하는 방법, api 를 활용하여 다른 모델들을 활용하는 방법 등을 터득할 수 있는 시간들이었다.

함께하여준 조원들에게 감사를 전합니다 ㅎㅎ

최진호

우여곡절 끝에 프로젝트가 완성되었습니다. 계획대로 진행이 되지는 않았지만 그 과정에서도 배우는 점들이 있었고 사이언스팀과 협업을 하면서 배우는 것도 많았고 상상만 하던 아이디어들이 모델로 구축되는 과정이 재미있었습니다. 시간과 데이터가 충분했다면 더욱 좋은 결과가 나올 수 있을 것 같아서 더 아쉽기도 하네요..

모두 고생하셨습니다~~🥹

