

김태언

깃허브 : <https://github.com/bjKim2>

포트폴리오 목차

1. 1-1 R을 이용한 분석 및 머신러닝 모델들을 이용한 예측
 1-2 1의 인공지능 이용한 앱 개발 (**REST API** 이용)
2. 가방 쇼핑몰 웹 개발

1. 1-1 R을 이용한 분석 및 머신러닝 모델들을 이용한 예측

개발 환경



SourceTree



3.9.7



8.0.28



개발 일정



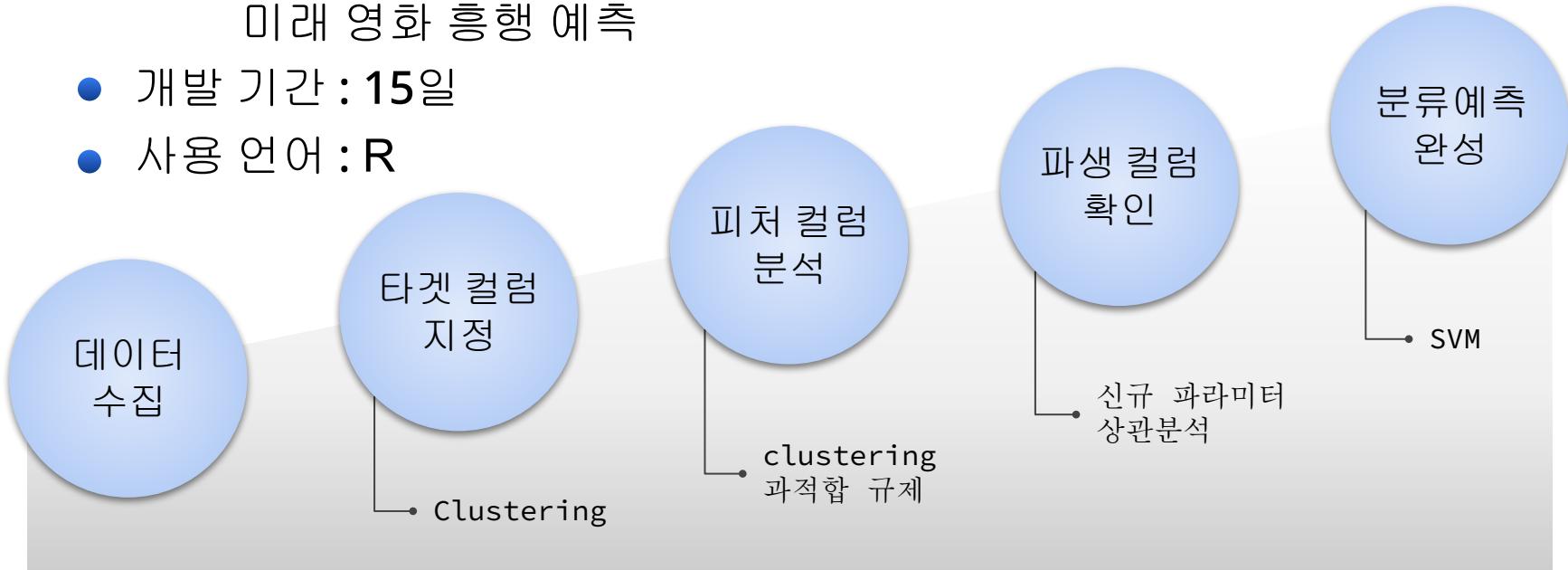
R 이용한 분석 및 예측 앱개발 깃허브 :

(R 분석) https://github.com/bjKim2/pj_rprac

(Flutter 앱) <https://github.com/RFlutterPythonteam1/mainProjectFlutter>

개발 개요

- 목적 : 국내 영화 흥행 예측
- 방법 : 2004~2019년의 국내 영화 흥행을 분석하고,
영화의 감독, 배우 등의 특징으로부터
미래 영화 흥행 예측
- 개발 기간 : 15일
- 사용 언어 : R



1-1. 데이터 수집

- 데이터 소스 다양화 (영화진흥위원회, 각종 영화 관련 사이트)
 - 각 소스별 추출 데이터 및 방법 소개

(영화진흥위원회) 2004년 ~ 2019년 영화 목록 2년 간격으로 API 추출 *주

데이터셋

(관련 사이트) 영화별 주연/조연 데이터 크롤링 *보조 데이터셋

```
4 # 조건 시작 및 끝 연도
5 openStartDt = "2000"
6 openEndDt = "2019"
7 itemPerPage = "100"
8 repNationCd = "22041011" # 한국 코드

1 # 총 개수를 가져오기 위한 변수 선언
2 temp_api_url = "http://www.kobis.or.kr/kobisopenapi/webservice/rest/movie/searchMovieList.json?key=%s&openStartDt=%s&openEndDt=%s&repNationCd=%s&itemPerPage=%s"
3 temp_url <- sprintf(temp_api_url, kobis_api_key, openStartDt, openEndDt, repNationCd)
4 temp_res <- GET(temp_url)
5 temp_resp_dat <- content(temp_res, as="parse", encoding="utf-8")
6
```

```

1 resp_df <- c()
2 for (curPage in 1:round(temp_resp$dat$movieListResult$totCnt / 100)) {
3   # 영화관별 API URL
4   kobis_api_url <- "http://www.kobis.or.kr/kobisopenapi/webservice/rest/movie/searchMovieList.json?key="tskipOpenSt
5 
6   # `tskip`를 지정한 부분에 변수값을 치환하여 현재 주소 결합
7   api_url <- sprintf(kobis_api_url, kobis_api_key, openStartDt, openEndDt, itemPerPage, curPage, repNationCd)
8   resp <- GET(api_url)
9 
10  resp_dat <- content(resp, as="parse", encoding="utf-8")
11  # 키워드에 따라 데이터가 제대로 변함
12  for (i in 1:length(resp$dat$movieListResult$movieList)) {
13    resp_df <- rbind(resp_df, resp$dat$movieListResult$movieList[[i]][1:3])
14  }
15

```

```

import requests
from bs4 import BeautifulSoup
import pandas as pd
data = pd.read_csv('../movies/movie2004_2019_ver2.csv')

url = [ ] + data['영화명'][194]
url = [ ] + ['[이스'
print(url)

response = requests.get(url)
temp = 0

if response.status_code == 200:
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')

    mName = soup.select('#content > div.culm2_area > div.culm2_1 > ul.mov_list > li > p.name > a')

    for k in range(0,len(mName)):

        if(mName[0].text == mName[len(mName)-1-k].text):
            temp = len(mName)-1-k
            print(temp)
            break

    for l in range(0,temp+1):
        title = soup.select('#content > div.culm2_area > div.culm2_1 > ul.mov_list > li:nth-child('+repr(l+1) + ')')
        for j in range(0,len(title)):
            print(title[j].text)

```

영화진흥위원회

<https://www.kofic.or.kr/kofic/business/main/main.do>

영화 관련 사이트 1

영화 관련 사이트

1-2. 데이터 정제

- 타겟 컬럼 확인 : 누적관객수 300만~1000만 범위에서 결측치,
이상치 無
- 피처 컬럼 확인 : 문자 데이터를 수치화 필요,
피처 컬럼 분석 단계에서 정제 진행
- 파생 컬럼 생성 : 주/조연 데이터를 크롤링으로 추가 수집
피처 컬럼 중 배우 컬럼을 바탕으로 주/조연 역할별
분리

2-1. 타겟 컬럼 지정

타겟 컬럼
지정

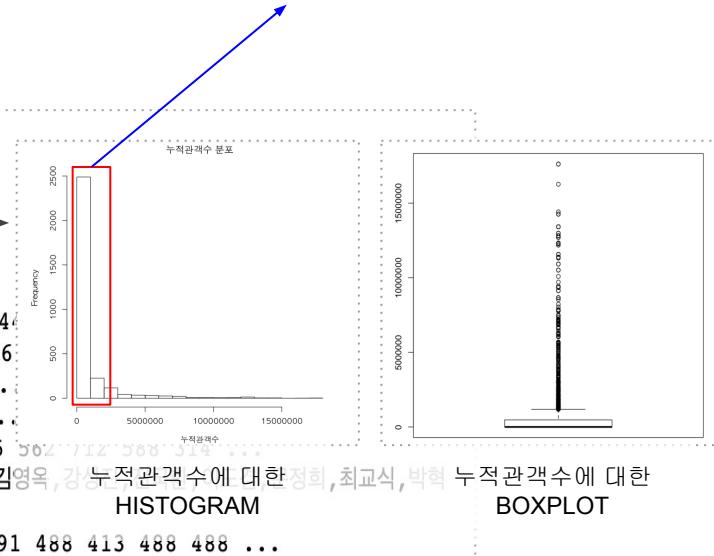
- 흥행 척도를 나타낼 컬럼 지정
 - 영화 흥행의 일반적 지표 : 매출액, 관객수
▶ 관객수를 기준으로 흥행 여부 판단 결정

- raw data

```
'data.frame': 2710 obs. of 12 variables:  
$ 영화명    : Factor w/ 2710 levels "...ing","0.0MHz",... : 1 2 3 4 5 7 6 8 9 10 ...  
$ 누적매출액: num 2.17e+08 1.09e+09 3.11e+09 1.90e+09 6.94e+07 ...  
$ 누적관객수: int 34300 137294 433866 288452 4078 32616 177 7232387 177 1 ...  
$ 스크린수   : int 1 446 252 340 35 64 20 1299 20 1 ...  
$ 상영횟수   : int 1 5998 12744 19036 436 1875 20 138226 20 1 ...  
$ 개봉일     : Factor w/ 1365 levels "2003-01-30","2003-04-25",... : 6 1219 277 249 5 ...  
$ 배급사     : Factor w/ 334 levels "(유)쏘아필름",... : 167 79 96 96 277 326 156 101 15 ...  
$ 등급       : Factor w/ 7 levels "12세관람가","12세이상관람가",... : 1 4 4 4 6 2 7 4 7 7 ...  
$ 장르       : Factor w/ 224 levels "가족,드라마",... : 24 9 136 108 77 19 54 19 54 54 ...  
$ 감독       : Factor w/ 930 levels "감수성","감수성,강수성",... : 601 499 784 471 744 705 ...  
$ 배우       : Factor w/ 1599 levels "", "감우성,김수로,이진영,이준원,최정우,김혁,최명진,조덕현,김민",... : 1185 1275 582 1111 440 1599 1 817 1 1 ...  
$ 제작사     : Factor w/ 916 levels "(유)내부자들 문화전문회사",... : 640 453 426 228 549 1 ...
```

타겟 컬럼
확인

- 관객수 낮은 구간 분포 多
 - 관객수 높은 구간 예측 위해
관객수 낮은 구간 예측 제외 결정



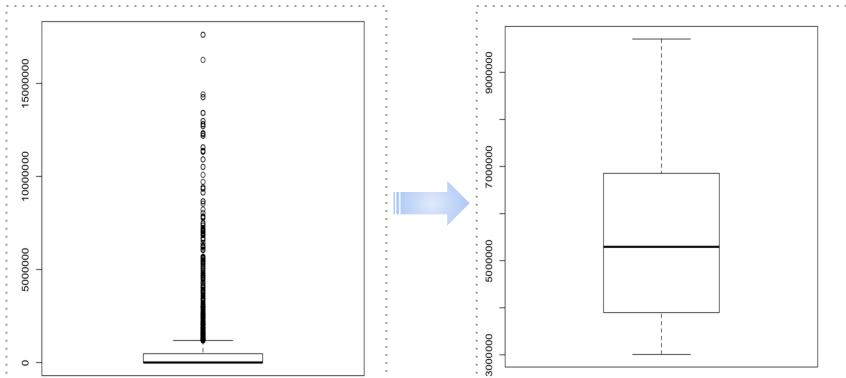
2-2. 타겟 컬럼 분석

타겟 범위 지정

- 분포가 의미있는 구간 찾기 위해 Boxplot 이용
- 목표 : 예측 구간의 분포가 관객수별 편차 및 이상치 無
- 이상치 無

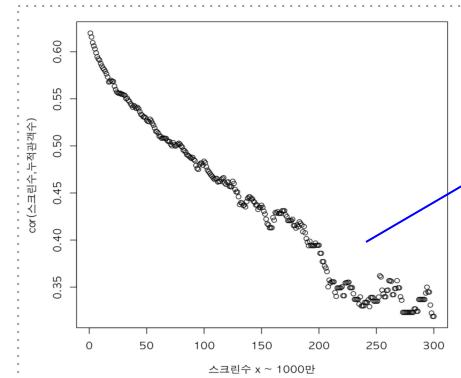
Boxplot 이용 최적 구간 설정

▶ 관객수 300~1000만 구간 선택



타겟 범위에 대한 피처 컬럼과의 관계

- 스크린수 - 타겟 컬럼 (관객수) 상관관계 파악
- 타겟 컬럼의 범위에 따라 상관관계 변화 大
- 범위 300~1000만에서 스크린수와 상관성 최
▶ 다른 피처 컬럼과의 관계가 중요도 高 추론



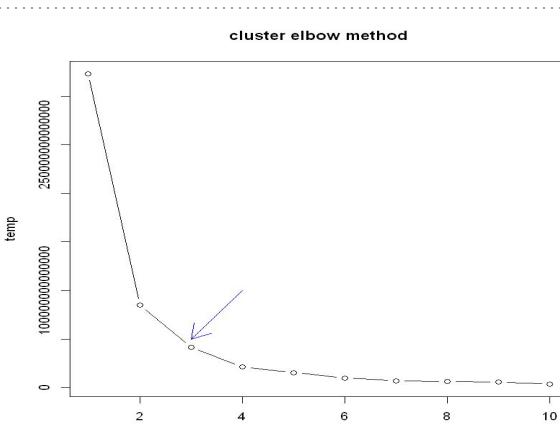
▲ 타겟 컬럼 범위의 최솟값의 변화에 따른 스크린수-관객수 상관관계

타겟 컬럼 범위의
최솟값 키울수록
스크린수와 관객수
상관관계 감소

2-2. 타겟 컬럼 분석

클러스터링

- 연속된 누적관객수 컬럼을 구간별로 군집
 - **K-mean clustering** 사용
 - 군집수 선택 elbow method 이용



○ 클러스터링 수행 결과

: K-means clustering with 3 clusters of sizes 43, 37, 21

: Cluster means:

```
[,1]  
1 3575761  
2 5273662  
3 7651660
```

: Clustering vector:

```
[1] 3 1 1  
[28] 1 2 2
```

```
Within cluster sum of squares by cluster:  
[1] 6639411436057 12126481346736 22651575244285  
(between_SS / total_SS =  87.2 %)
```

: Available components:

```
[1] "cluster"      "centers"       "totss"        "withinss"      "tot.withinss"
[6] "betweenss"    "size"          "iter"         "ifault"
```

○ 최종 타겟 컬럼

3-1. 피처 컬럼 분석 - 배우

배우 컬럼 분석

- 원 데이터 : 주연, 조연, 단역이 ','(콤마)로 분리되어 한 데이터에 삽입 되어 있음 (datatype : character)
 - ▶ 이 자체로 데이터 분석 불가
→ 양을 줄이고 배우별 분리 필요
 - 처리 방식 : 역할별 영화 출연 비중이 다르기 때문에 주연, 조연만 처리하되 (단역 제외), 배우 1명씩 분리하여 처리 필요
 - 주·조연 분리 기준 : 영화 관련 사이트에서 크롤링
 - raw data
- 

배우
임은정,김래원,이미숙,김자영,김현수,박혜연,장미,이성경,박소연,김순애,최덕문,김인문,양주호
정은지,이성열,최윤경,신주환,정원창,정원상,박영신,남관호,박대희,노혜영,서진석,김대현,신승민,오규택,오세규,오제현,김성우,김수진 이정수,이상찬,허현준,이성훈,김정연,홍주영,김호영,김정현,김효은,조승구,조용한,김승희,김시아,황지원,조지영,강성민,강태웅,이경환,구원희,최현복,김진록,김난희,김하나,김선경,김상체,한소연,이채린,김진호,이병수,조원원,노희성,한정원,이혜미,조창순,조희원,주민준,나소이,이정희,한시랑,김원숙,정낙안,유희찬,정설리,정승필,윤수빈,김채민,윤영식,최하림,이태준,김진아,최원,양시열,서우린,임창록,서한별,홍의현,한소영,황연희,한정윤,한태규,신지혜,박선영

- 1차 가공 후 : 배우를 주연, 조연으로 분리했지만
 - ▶ 아직 한 데이터에 여러 배우가 쌓여있음
→ 한 데이터에 한 값이 들어 가도록 처리 필요
- 접근 방식 (2가지)
 - 배우 한명 당 1개 row로 적용하여 한 영화당 여러 행을 만드는 방법
 - 한 영화에 있는 배우들 수치화, 배우별 점수 부여

○ 1차 가공 결과물

주연	조연
김윤석,하정우,유해진,김태리,박희순,이희준	유승목,현봉식,박지현,이용직,박지홍,김경덕,김의성,최광일,김수진,김혜진,조우진,임형렬,이창훈,서현우,이현균,김승훈,박경혜,이화정
김하늘,김지환	류승룡,장영남,강신일,도마뱀코 바딤,엘리자베스 수진 포드,코디 헌터,장남영,김정필,유승목,김형종,기형범,박성민
정유미,공유	김미경,공민정,박성원,이봉련,김성월,이얼,김미경,송상천,강애심,류이영,김정영
신현준,김원희,김수미	탁재훈,공형진,임현준,정준하,김해곤,정호빈
장혁,수예,박민하	유해진,이희준,김기현,이상업,박효주,박정민,보리스 스타웃,김문수,최병모,장경업,엔드류 월리엄 브랜드
설경구,정우성,한효주	김병록,진경,정호,손민석,김대진,이태형,강신하,이동희,이화정,박지훈,김지훈,성일,변요한,장현형,박지연,최현,남정현,이동진,이도윤,정재호,김준우,이현균

3-1. 피처 컬럼 분석 - 배우

1번째 접근

- 한 영화당 배우 한명을 값으로 넣어, 여러 행을 만드는 방법으로 시도

- 수행 결과

영화명	주연
1987	김윤석
1987	하정우
1987	유해진
1987	김태리
1987	박희순
1987	이희준
7급 공무원	김하늘
7급 공무원	강지환
82년생 김지영	정유미
82년생 김지영	공유

- 영화마다 배우수가 다르기 때문에 영화마다 행의 수가 달라지는 단점 발생
- 7급 공무원 주연이 2명인데 비해, 1987의 주연이 6명이기 때문에 1987의 학습이 더 많이 일어나는 불균형 발생
- factor의 level 수가 많으면 분류기 사용 어려움
 - randomForest는 53 종류가 넘으면 쓸 수 없고,
 - 인공신경망의 경우도 weight 가 커져 쓸 수가 없게 됨

3-1. 피처 컬럼 분석 - 배우

2번째 접근

- 한 영화에 있는 배우들을 수치화하여 배우점수를 주는 방법

- 수행 결과

주연 횟수	점수	평균점수
하정우	15	50 3.333333
송강호	11	42 3.818182
오달수	11	40 3.636364
유해진	13	39 3.000000
황정민	13	38 2.923077
김윤석	12	30 2.500000

- 알고리즘 : 영화를 관객수로 군집 (5개) 하여 점수화 후 출연횟수로 나누어,
평균적으로 배우의 출연 영화의 흥행도를 점수화 ['평균점수' 컬럼]
- ※ ['점수' 컬럼] : K-mean clustering 이용,
출연한 영화당 군집에 따라 점수(1~5점)를 주어 합산
타겟 컬럼 범위인 관객수 300~1000만에서의 영화만 대상시 과적합 발생
可
- ▶ 타겟 컬럼의 범위 + 위, 아래 패딩 적용하여 과적합에 대한 규제 적용
(총 200만~1761만 관객수의 영화를 대상으로 배우 점수를 매기는 방식)

K-mean clustering - 5개 클러스터로 군집

- 기 군집된 (300~1000만) 3개 클러스터

```
[1]
267844
2 3459965
3 5273662
4 7735577
5 12934724
```

3-1. 피처 컬럼 분석 - 배우

2번째 접근

- 한 영화에 있는 배우들을 수치화하여 배우점수를 주는 방법
 - 같은 방식으로 주연/조연에 분리하여 각각 점수화

○ 수행 결과 (주연)

주연	횟수	점수	평균점수
하정우	15	50	3.333333
송강호	11	42	3.818182
오달수	11	40	3.636364
유해진	13	39	3.000000
황정민	13	38	2.923077
김윤석	12	30	2.500000

○ 수행 결과 (조연)

조연	횟수	점수	평균점수
송영창	18	47	2.611111
이경영	18	46	2.555556
김의성	12	38	3.166667
장광	9	31	3.444444
주진모	12	31	2.583333
오달수	10	30	3.000000

3-1. 피처 컬럼 분석 - 배우

2번째 접근

- 한 영화에 있는 배우들을 수치화하여 배우점수를 주는 방법
 - ▶ 주/조연 배우 점수를 영화 점수로 반영
(가공한 주/조연 배우 점수를 영화데이터 컬럼에 추가)
- 주연 점수 반영 방식 : 출연한 주연 중 가장 높은 점수로 주연 점수를
- 조연 점수 반영 방식 : 조연 중 조연 점수 상위 3명의 평균으로 결정
(조연은 주연에 비해 상대적으로 수가 많기 때문)

○ 수행 결과

cluster	누적관객수	스크린점유율	배급사점수	장르_관객점수부여	감독_배우시너지	감독점수	배우점수	조연점수
1	4039891	23.79760	2	10	3	3.000000	4.000000	2.600000
1	3678156	48.26242	2	10	8	2.000000	8.250000	3.250000
1	3117859	36.90476	1	10	0	2.333333	4.000000	3.000000
1	4313101	35.27944	3	10	3	3.571429	6.777778	3.000000
1	3024666	21.64329	3	10	0	3.500000	7.666667	3.500000
1	4111237	28.49592	2	20	6	3.500000	6.500000	3.166667

```
df$주연점수[i] <- max(x)

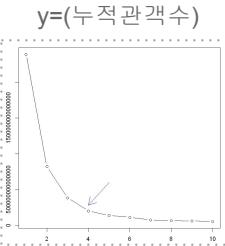
if(length(x) >= 3){
  df$조연점수2[i] <- sum(sort(x,decreasing = T)[1:3])/3
} else if(length(x) >= 2){
  df$조연점수2[i] <- sum(sort(x,decreasing = T)[1:2])/2
} else{
  df$조연점수2[i] <- max(x)
}
```

3-2. 피처 컬럼 분석 - 기타

기타 피처 컬럼

● 감독 점수

- 정의 : 감독별 스크린수 대비 관객수
- K-mean clustering,
영화별로 4개 군집 점수화 (1~4점)
- 감독 점수 = $\text{mean}(\text{영화별 점수}^2)$



● 장르 점수

- 평균 누적관객수로 3개 그룹 분할 점수화

장르	영화개수	누적관객수합	평균누적관객수
공포(호러)	2	7525189	3762594
드라마	33	180733979	4870727
미스터리	3	15511995	5170665
범죄	16	82285716	5142857
스릴러	3	13012318	4337439
코미디	17	79566549	4680385
멜로/로맨스	3	16847559	5615853
사극	11	58756875	5341534
시부극(웹스ери)	1	6685845	6685845
액션	14	84409628	6029259
어드벤처	2	14956710	7478355
판타지	2	13486983	6743492
SF	1	9350323	9350323

각 그룹당
장르 점수로
10/20/30점
부여

● 개봉일 스크린 수

- 상영 기간 동안 일자별 최대 스크린 수
- 흥행 가능성에 따라 증감
- 흥행 관련 지표 컬럼으로 분류기에 적용

● 배급사 점수

- 평균 누적관객수로 3개 그룹 분할 점수화

배급사	횟수	누적관객수합	배급사 평균누적관객수
기타	9	41673129	36328355
(주)네스트엔터테인먼트(NEW)	14	65733469	4695248
(주)롯데엔터테인먼트	14	72799502	5199964
(주)쇼박스	29	156775745	5406060
CJ ENM	42	226147824	5384472

각 그룹당
배급사 점수로
1/2/3점 부여

4. 파생 컬럼 확인

감독-배우 점수 컬럼 생성

- 개념 : 감독이 어떤 배우랑 영화를 찍을 때 시너지가 나는지 점수화
- 과적합의 규제를 위해 200~1700만의 범위에서 점수 부여 (타겟 컬럼 범위 + 상/하단 패딩)

① 감독별 영화들의 누적 관객수로 ② 감독_배우 매칭
영화등급을 상, 중, 하로 분류
(최대/최소관객수 범위를 3등분)

감독	누적관객수	등급
강우석	4313101	상
강우석	974262	하
강우석	3134421	중
강우석	1887733	하
강우석	1719165	하
강우석	3350303	상
강우석	1744585	하
강우석	3331593	상
강현철	8223266	상
강현철	1473125	하

감독	누적관객수	등급	배우	감독_배우
강우석	4313101	상	정재영	강우석_정재영
강우석	4313101	상	강신일	강우석_강신일
강우석	4313101	상	유해진	강우석_유해진
강우석	4313101	상	탁트인	강우석_탁트인
강우석	4313101	상	이상현	강우석_이상현
강우석	4313101	상	김윤성	강우석_김윤성

③ 감독_배우 점수 생성

- 상 등급 하나당 +1, 중 0, 하 -1점 → 총합
- 점수가 양수일 때 감독+배우 시너지가 양의 관계

감독_배우	점수
강우석_정재영	0
강우석_강신일	1
강우석_유해진	2
강우석_탁트인	1
강우석_이상현	1
강우석_김윤성	1
강우석_이재원	1
강우석_조한영	1
강우석_박창용	1
강우석_김율호	-1

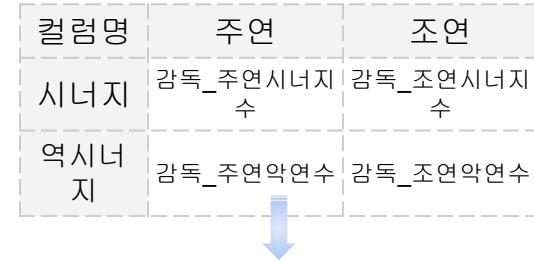
4. 파생 컬럼 확인

주/조연별 시너지 점수 추가

- 목적 : 주/조연별 시너지 점수 및 악연 점수 확인하여 상관관계 도출
- 한 영화 당 감독-배우 일대다 관계이므로 배우별 시너지 점수 확인 필요
 - ▶ 시너지 점수, 주/조연 여부 매트릭스화

○ 수행 결과

영화명	감독	주연	제작사	cluster	조연	감독_주연 시너지 점수			
						감독_주연 시너지 수	감독_주연 악연 수	감독_조연 시너지 수	감독_조연 악연 수
1987 장준환	김윤석, 하정우, 유해진, 김태리, 박희순, 이희준	(주)우정필름	4	유승목, 현봉식, 박지환, 이윤직, 박지홍, 김경덕, 김성, 최광일, 김수진, 김종수, 김혜정, 조우진, 임철현, 이창훈, 서현우, 이현규, 김승훈, 박경혜, 이화룡		6	0	0	0
1번가의 운제군	임창정, 하지원	(주)두사부필름	1	주현, 정두홍, 이훈, 강예원, 박창익, 박유선, 고태호		0	0	0	0
26년 조근현	진구, 한혜진, 임슬옹	영화사청어람(주)	2	배수빈, 이경영, 장광, 이미도, 조덕제, 김의성, 안석환, 민복기, 구성환, 김민재, 최귀화, 김태수, 장광		0	0	0	0
7광구 김지훈	하지원, 안성기, 오지호, 박철민, 송새벽	(주)제이케이필름, (주)씨제이이엔엠	1	박정학, 이한위, 박영수, 차예련, 오민석		2	0	1	0
7급 공무원 신태라	김하늘, 김자한	(주)하리마오피처스, (주)영화사수작, (주)디씨플러스	2	류승룡, 장영남, 강신일, 도아천코 바님, 엘리자베스 수진 포드, 코디 헌터, 장남열, 김정필, 유승목, 김형종, 김형범, 박성민		1	0	2	0
7번방의 선물 이환경	류승룡, 박신혜, 갈소원, 오달수, 박원상, 김정태, 정만식, 김기현	(주)화인웍스, (주)씨엔터테인먼트	5	박길수, 조재윤, 조덕현		0	0	0	0



누적 관객수와 각 컬럼간의
상관관계 분석

4. 파생 컬럼 확인

상관관계 분석

- 누적관객수와 감독+배우 시너지 컬럼 상관성
파악
○ 수행 결과

```
1 cor(df$누적관객수, df$감독_주연시너지수) → 0.485
2 cor(df$누적관객수, df$감독_주연악연수)
3 cor(df$누적관객수, df$감독_조연시너지수) → 0.460
4 cor(df$누적관객수, df$감독_조연악연수)
5
6 cor(df$누적관객수, df$감독_주연시너지수 - df$감독_주연악연수)
7 cor(df$누적관객수, df$감독_조연시너지수 - df$감독_조연악연수)
8
9 cor(df$누적관객수, df$감독_주연시너지수 + df$감독_조연시너지수) → 0.560
10 cor(df$누적관객수, df$감독_주연시너지수 - df$감독_주연악연수 + df$감독_조연악연수)
```

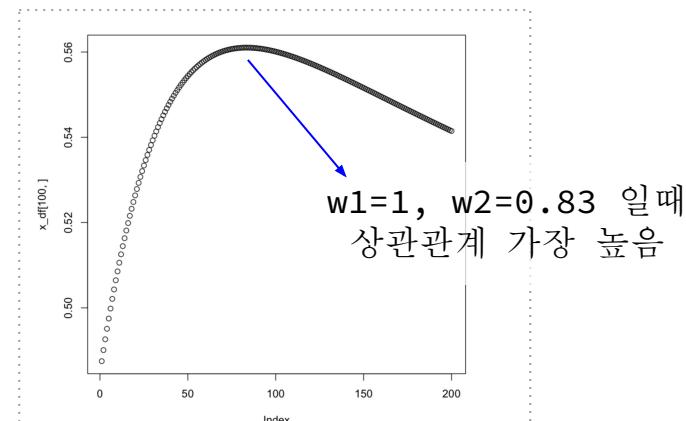
유의
미



0.484830381781054
-0.131253036386317
0.460139424425956
-0.0389317925544824
0.477959148726438
0.436601057882286
0.560090637108292
0.535599443139518

가중치 부여

- 주/조연 비중 차이로, 감독_주연시너지수와
감독_조연 시너지수에 각각 가중치 부여
- $(w1) * \text{감독_주연시너지수} + (w2) * \text{감독_조연시너지수}$
 - 가중치 조정에 따른 시각화
 $(0 \leq w1, w2 \leq 2)$



5-1. 최종 data 점검

● 최종 dataset 구조

```
'data.frame': 108 obs. of 8 variables:  
 $ cluster      : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 ...  
 $ 배급사점수   : int 2 2 1 3 3 2 1 3 1 3 ...  
 $ 장르점수     : int 10 10 10 10 10 20 10 10 10 10 ...  
 $ 감독_배우시너지점수: int 3 8 0 3 0 6 1 5 7 4 ...  
 $ 주연점수     : num 4 8.25 4 6.78 7.67 ...  
 $ 감독점수     : num 2 4 3 2.71 2.5 ...  
 $ 개봉일스크린수 : int 339 1954 671 575 344 549 264 1180 513 825 ...  
 $ 조연점수     : num 2.6 3.25 3 3 3.5 ...
```

● 피처-타겟 컬럼의 상관관계

배급사점수	장르점수	감독_배우시너지점수	주연점수	감독점수	개봉일스크린수	조연점수
0.1195046	0.293891	0.222731	0.6464251	0.4231618	0.2367386	0.7575004

● 피처 컬럼 간의 상관관계

배급사점수	장르점수	감독_배우시너지점수	주연점수	감독점수	개봉일스크린수	조연점수
1.000000000	0.1424967	-0.05560081	0.2279586	0.07741333	-0.008884294	0.06128511
장르점수	0.142496688	1.0000000	0.16640569	0.1380842	0.20950164	0.288948774
감독_배우시너지점수	-0.055600808	0.1664057	1.0000000	0.1543465	0.31772470	0.355112344
주연점수	0.227958623	0.1380842	0.15434646	1.0000000	0.23063766	0.269348352
감독점수	0.077413333	0.2095016	0.31772470	0.2306377	1.000000000	0.087373655
개봉일스크린수	-0.008884294	0.2889488	0.35511234	0.2693484	0.087373766	1.000000000
조연점수	0.061285114	0.3674948	0.33701061	0.5110963	0.36348825	0.239786020
						1.000000000

◎ 참고

- cluster : 타겟 컬럼 (누적관객수 clustering 결과)
- 감독_배우시너지점수 : 피처 컬럼 (기존 피처에서 파생)
- 그 외 : 피처 컬럼

◎ 각 피처 컬럼과 타겟 컬럼 모두 양의 상관관계

◎ 피처 컬럼간 가장 높은 상관관계가 0.51이며, (주연점수-조연점수)

대부분 0.06~0.37 구간으로 피처 컬럼간의 상관성 낮음

5-2. 머신러닝 적용

● 피처 1개씩 추가하여 분류기 적용

- 인공신경망, 의사결정나무, randomForest, SVM, 다항로지스틱회귀분석 적용
- 1000번의 머신러닝 예측력 평균으로 정확도 확인
- SVM의 정확도가 가장 높았으며, train data과 test data의 정확도 차이가 3.2%로 적정 수준으로 판단

○ 최종 dataset

cluster	배급사점수	장르점수	감독_배우시너지점수	주연점수	감독점수	개봉일스크린수	조연점수
1	2	10	3	4.000000	2.000000	339	2.600000
1	2	10	8	8.250000	4.000000	1354	3.250000
1	1	10	0	4.000000	3.000000	671	3.000000
1	3	10	3	6.777778	2.714286	575	3.000000
1	3	10	0	7.666667	2.500000	344	3.500000
1	2	20	6	6.500000	4.000000	549	3.166667

○ ML 적용 결과 정확도 확인

	인공신경망	의사결정나무	randomForest	SVM	다항로지스틱회귀분석
트레이닝	0.4545455	0.4025974	0.4545455	0.4545455	0.3636364
테스트	0.4193548	0.3870968	0.4193548	0.4193548	0.3225806
	인공신경망	의사결정나무	randomFore	SVM	다항로지스틱회귀분석
트레이닝	0.4025974	0.4025974	0.7190083	0.5064935	0.3766234
테스트	0.3812317	0.3870968	0.4926686	0.4516129	0.3870968
	인공신경망	의사결정나무	randomFore	SVM	다항로지스틱회귀분석
트레이닝	0.4073200	0.4025974	0.6776860	0.5714286	0.4415584
테스트	0.3841642	0.3870968	0.4428152	0.4838710	0.4193548
	인공신경망	의사결정나무	randomFore	SVM	다항로지스틱회귀분석
트레이닝	0.4250295	0.5454545	0.9905545	0.6623377	0.4805195
테스트	0.3900293	0.6129032	0.4926686	0.5483871	0.4193548
	인공신경망	의사결정나무	randomFore	SVM	다항로지스틱회귀분석
트레이닝	0.4262102	0.6753247	1.000000	0.7272727	0.6233766
테스트	0.4046921	0.5806452	0.6363636	0.7096774	0.6129032
	인공신경망	의사결정나무	randomFore	SVM	다항로지스틱회귀분석
트레이닝	0.4923259	0.6623377	1.000000	0.7792208	0.6883117
테스트	0.3900293	0.5806452	0.6803519	0.7419355	0.7419355

SVM

트레이닝 0.8701299

테스트 0.8387097

5-3. 결론

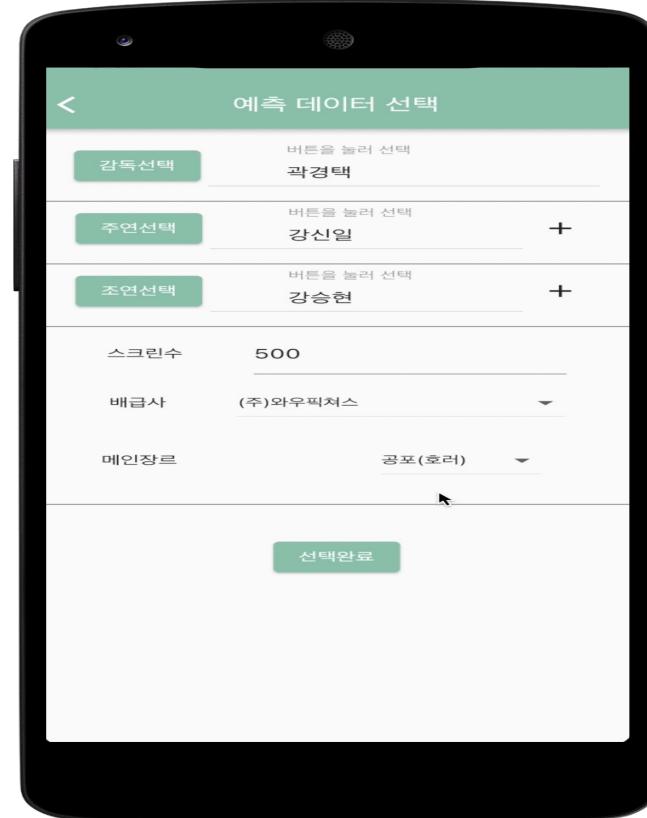
- 영화 산업구조상 일반적으로 스크린수와 관객수 상관성이 클 것으로 예상하였으나, 실질적으로 유의미한 타겟 범위 (누적관객수 300~1000만)에서는 스크린 수와 관객수 상관성이 적었음
- 누적관객수 300만 이하의 구간에서는 스크린수-관객수 상관성이 보였으나, 유의미한 예측을 위해서는 다양한 컬럼 추가가 필요함
- 대부분의 컬럼은 문자형 데이터였으므로, 수치화가 필요
→ 누적관객수와의 분포에 따라 구간을 나누어 가중치 부여
- 수치화된 컬럼 추가 시마다 예측력 향상 확인
SVM의 test data 예측력이 84% 수준
- 특히 배우 점수 (주연점수, 감독_주연시너지점수, 조연점수) 반영 시 예측력이 유의미하게 증가하였음 (test data에서 29% 상승)
→ 피처 컬럼의 점수화 알고리즘에 따라 성공적인 예측이 가능함을 확인

1. 1-2 1의 인공지능 이용한 앱 개발 (REST API 이용)

머신러닝 결과를 이용한 App 만들기

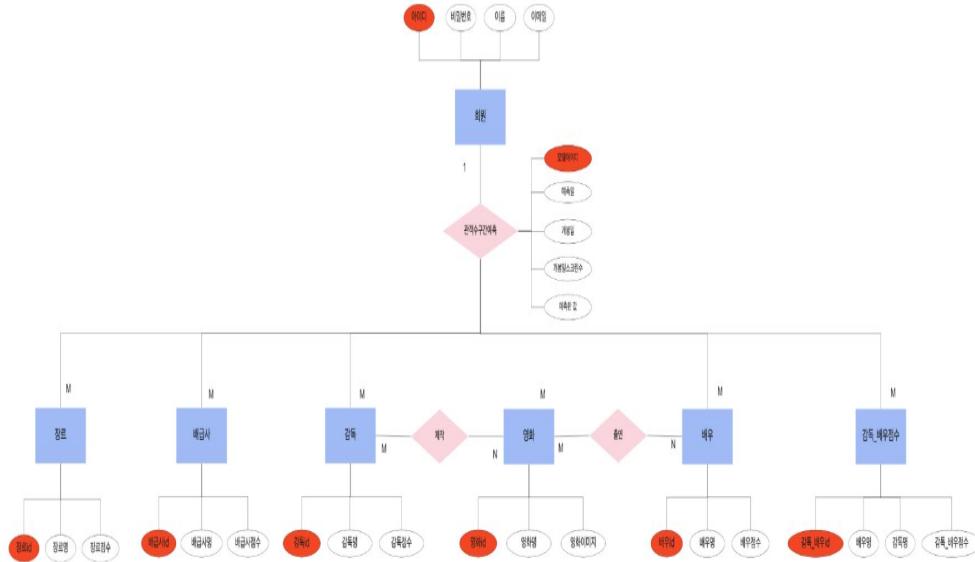
App 개발환경
Flutter 프레임워크
Dart 언어

서버 연결방식 - REST API(Tomcat server을 이용한 MYSQL, Rserver 연결)
DBMS - MYSQL

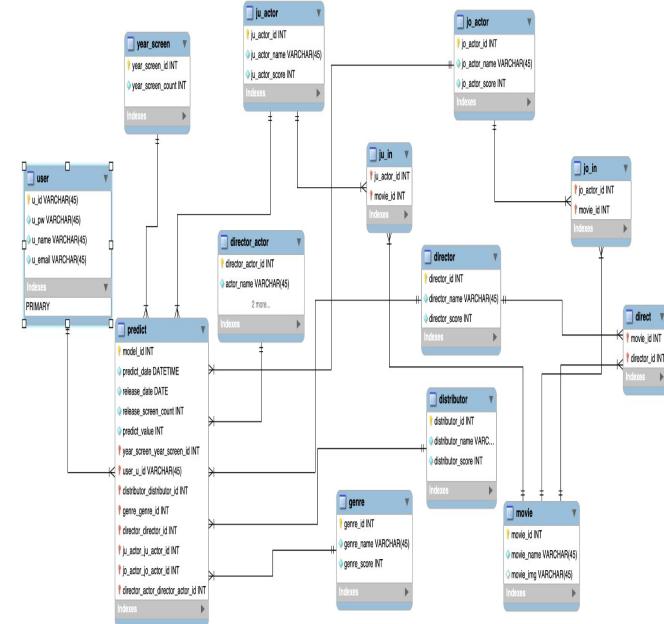


APP 개발 diagram

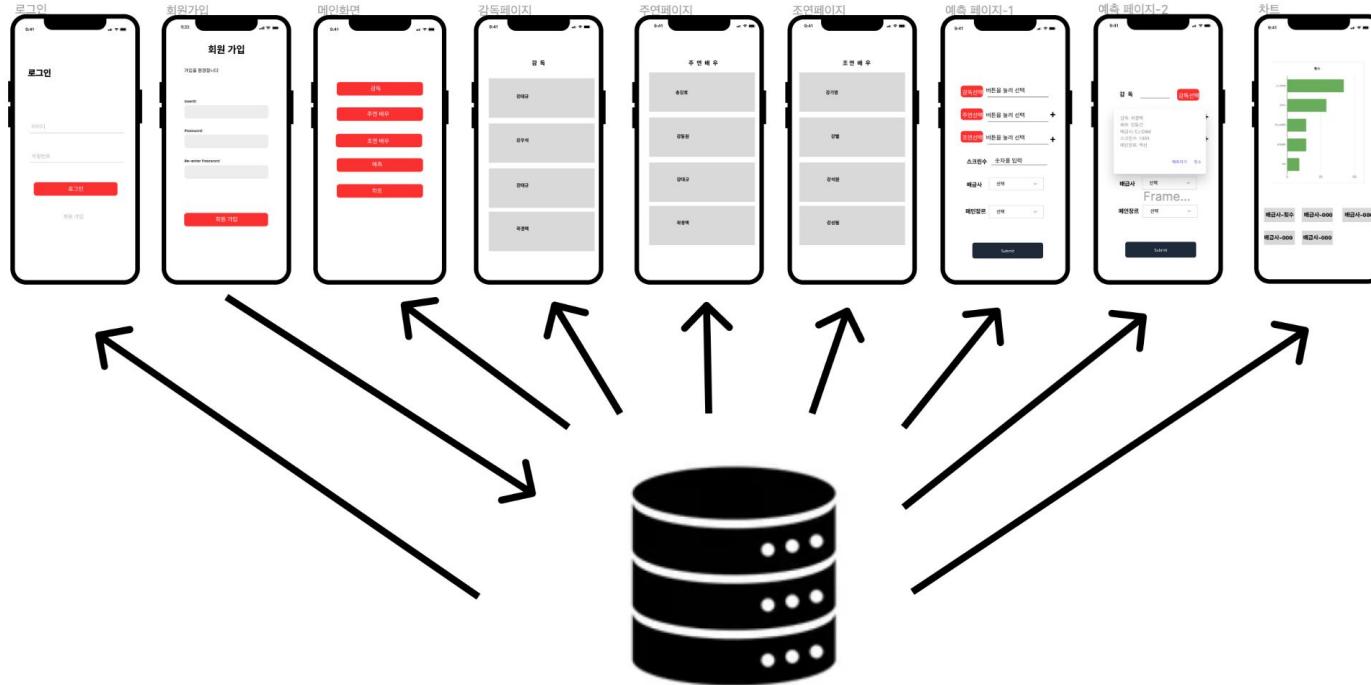
ERD



EER



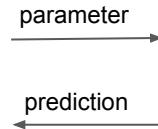
APP 화면 구성



REST API

APP

```
mainproject > lib > inputData.dart > _FirstState > predict
1395 }
1396
1397
1398 // ignore: prefer_interpolation_to_compose_strings
1399 var url = Uri.parse("http://localhost:8080/Rserve/movie_predict.jsp?dis=" +
1400   dis.toString() +
1401   "&genre=" +
1402   genre.toString() +
1403   "&direct=" +
1404   direct.toString() +
1405   "&ju=" +
1406   ju.toString() +
1407   "&dir=" +
1408   dir.toString() +
1409   "&screen=" +
1410   screen.toString() +
1411   "&jo=" +
1412   jo.toString());
1413
1414 var response = await http.get(url);
1415
1416 var jsondata = json.decode(utf8.decode(response.bodyBytes));
1417
```



WEB Server(jsp)

```
usr > local > Tomcat > webapps > ROOT > Rserve > movie_predict.jsp > ? > ? > ? > ?
1 <%@page import="org.rosuda.REngine.Rserve.RConnection"%>
2 <%@ page language="java" contentType="text/html; charset=UTF-8"%>
3 <% pageEncoding="UTF-8"%>
4
5 <%
6 request.setCharacterEncoding("UTF-8");
7
8 double dis = Double.parseDouble(request.getParameter("dis"));
9 double genre = Double.parseDouble(request.getParameter("genre"));
10 double direct = Double.parseDouble(request.getParameter("direct"));
11 double ju = Double.parseDouble(request.getParameter("ju"));
12 double dir = Double.parseDouble(request.getParameter("dir"));
13 double screen = Double.parseDouble(request.getParameter("screen"));
14 double jo = Double.parseDouble(request.getParameter("jo"));
15
16 RConnection conn = new RConnection();
17
18 conn voidEval("library(e1071)");
19 conn voidEval("rf <- readRDS(url('http://localhost:8888/Rserve/SVM_Model_2.rds','rb'))");
20
21 conn voidEval("result <- as.character(predict(rf, (data.frame(배급사점수=" + dis + ", 강로점수=" +
22 + "감독_배우사내지점수=" + direct + ", 주연점수=" + ju + ", 감독점수=" + dir + ", 개봉일스크린수=" + s
23
24 String result = conn eval("result")asString();
25 // out.println(result);
26
27 %>
28 {"result":"<%=result%>"}
29
```

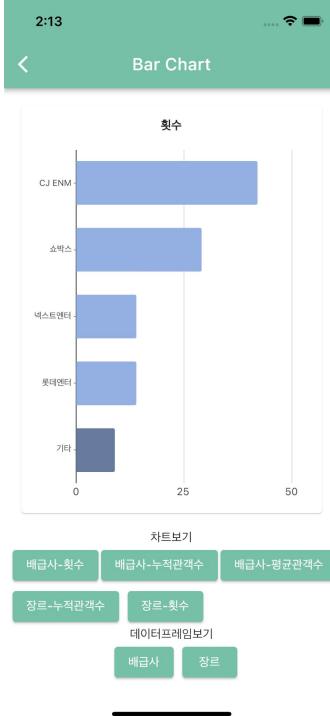


R
server

App에서 Tomcat으로
Parameter get방식으로 전달
후 리턴 값을 받음

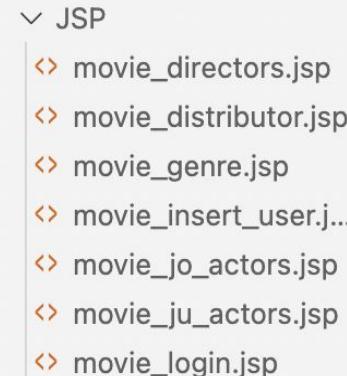
Tomcat이 Rserver에 예측할
데이터 전달 후 rds 파일을 읽어와
예측하고 예측 값을 리턴 해주는
부분

APP 화면 - 차트 구현



```
mainproject > lib > showchart.dart > _MyHomePageState > _loadCSVDF
183     ); // SCATTERD
184   }
185
186   void _loadCSVChart(String path, int colNum) async {
187     final rawData = await rootBundle.loadString(path);
188     List<List<dynamic>> listData = const CsvToListConverter().convert(rawData);
189     setState(() {
190       ischart = true;
191       csvData = listData;
192       Cdata.title = csvData[0][colNum];
193       csvData.removeAt(0);
194       data.clear();
195       for (int i = 0; i < csvData.length; i++) {
196         data.add(DeveloperSeries(
197           distributor: csvData[i][0],
198           cnt: csvData[i][colNum],
199           barColor: charts.ColorUtil.fromDartColor(Color.fromRGBO(255, 148, 176, 227)), Prefer co
200         )); // DeveloperSeries
201       }
202     });
203   }
204
205   void _loadCSVDF(String path) async {
206     final rawData = await rootBundle.loadString(path); Avoid leading underscores for local identifiers.
207     List<List<dynamic>> listData = Avoid leading underscores for local identifiers.
208     const CsvToListConverter().convert(_rawData);|
209     setState(() {
210       ischart = false;
211       csvData = _listData;
212     });
213   }
214 }
```

APP 산출물



movie_predict.jsp
SVM_Model_2.rds

dependencies:

```
flutter:  
|  sdk: flutter
```

```
# The following adds the  
# Use with the Cupertino  
cupertino_icons: ^1.0.2  
http: ^0.13.4  
animation_list: ^2.1.0  
charts_flutter: ^0.12.0  
csv: ^5.0.1  
blinking_text: ^1.0.2
```



- dart : 17 개
- jsp : 8 개
- csv : 2 개
- jpeg : 113 개
- dependencies : 5 개

향후 계획

- 관리자(영화감독, 배우 등록) 구축하기
- **300만** 미만 혹은 **1000만** 이상에 대한 예측 및 분석 해보기
- 배우 선택관련 기능 보강 및 배우 또는 감독 선택 시 관련 정보 표시기능 구현
- 다른 컬럼 두 개를 합친 파생컬럼 더 만들어 보기
- **factor**형 컬럼을 수치화하지 않고 **factor**의 레벨을 줄인 뒤 원 핫 인코딩을 써서 머신러닝 해보기
- 수치화할 때 가중치를 분포에 따라 가중치 주는 것을 좀 더 다양하게 머신러닝 돌려보기
- 수치화할 때 특정 점수가 아닌 분포에 따른 확률 값을 점수로 사용해보기
- 수치화 된 컬럼들을 특성공학을 이용해 추가해보기
- 영화에 관련된 새로 컬럼 찾아보기
- 영화를 **10년치, 5년치**만 써서 예측해보기
- 타겟 컬럼을 좀 더 넓은 범위로 예측해보기
- **kmeans**가 아닌 군집화 방법(**gmm** 등) 사용해보기
- 외부 데이터를 이용한 컬럼 추가해보기
- **Visibility** 위젯의 효율적인 사용

2. 가방 쇼핑몰 웹 개발

mvc패턴

팀 웹개발 프로젝트

(JSP ver, SPRING ver)

- 프로젝트명 : elapid
- 캐리어 쇼핑몰 웹 프로젝트
- JSP version
- SPRING version
(DBCP2, JDBC, mybatis
3가지 방식 DB연결)



github 주소

(jsp버전) https://github.com/team2Joe/ELAPID_JSP_PROJECT

(spring버전)

https://github.com/team2Joe/ELAPID_SPRING_PROJECT

팀원 구성 및 역할

김태언

관리자 페이지(경영정보테이블)
회원가입(정규식/도로명 주소 api)
구글 api 로그인
로그인 체크
로그인 로그
상품 구매/카트 선택구매/카트 전체 구매
구매내역(페이징 처리)
이미지 업로드(cos.jar)
주소록(기본주소지 설정)
에러페이지관리
ERD&ProcessDiagram&DataFlow Diagram

정OO

장바구니 기능
ERD&ProcessDiagram& DataFlowDiagram
영상 촬영

정OO

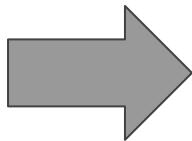
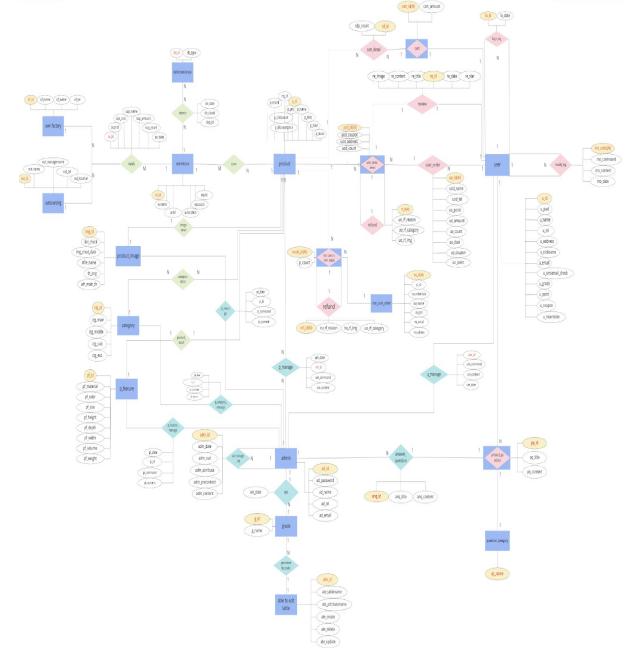
메인 페이지
상품 리스트 및 상세페이지
상품 필터링 기능
상품 검색 기능
상품 리스트 페이징
상품 조회수 순위 페이지
상품관련
ERD&ProcessDiagram &DataFlowDiagram

최OO

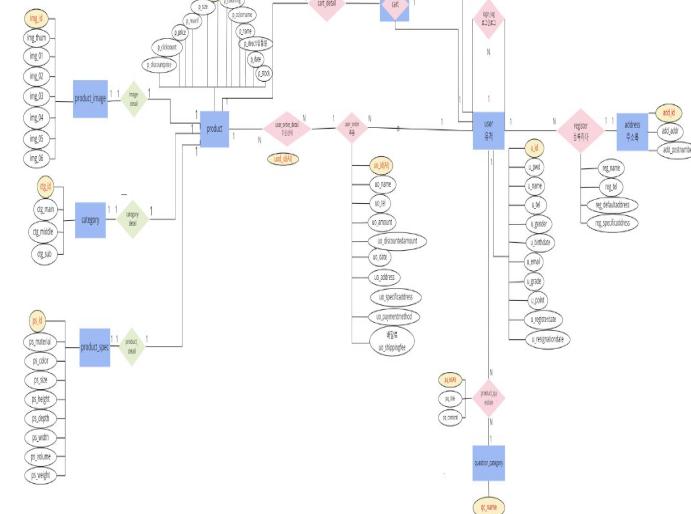
1:1 문의 게시판
ERD&ProcessDiagram& DataFlowDiagram
영상 촬영/편집

ER-Diagram 소개

기획 단계의 ER-Diagram

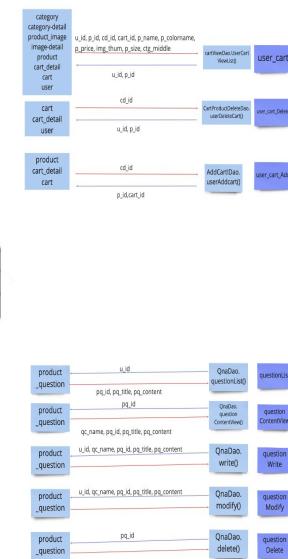
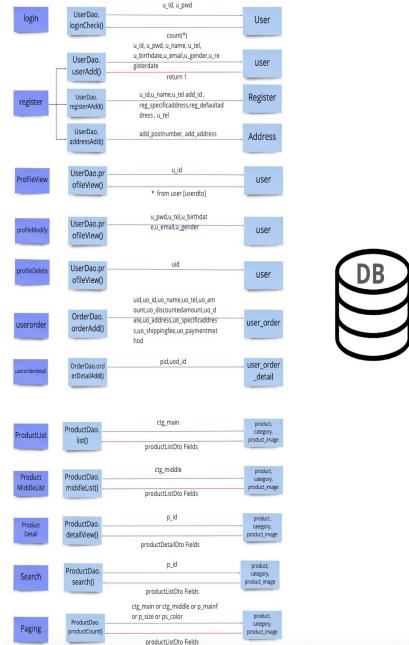


실제 작업한 ER-Diagram

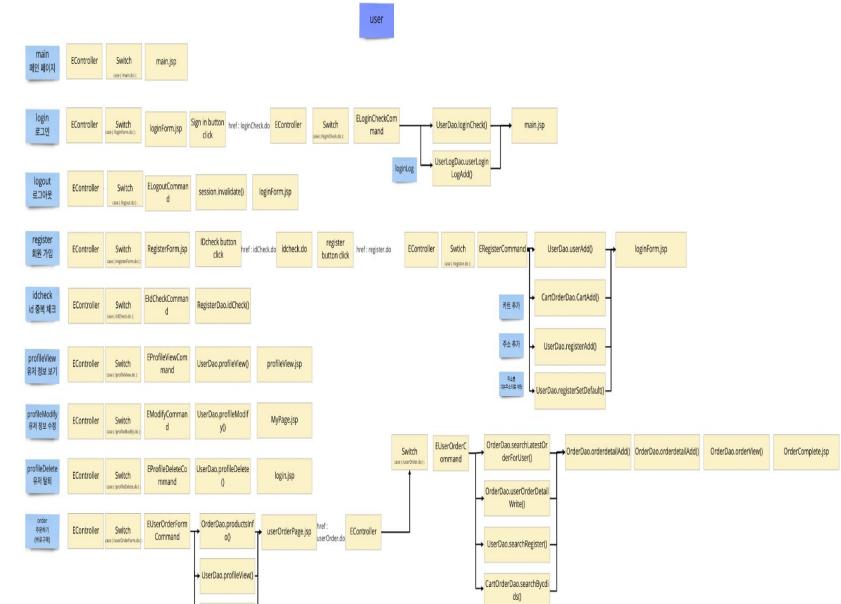


Flow diagram 소개

기획 단계 DFD



Process Flow Diagram



IP로 순방문자수 구하기

동일 ip는 1시간 이내 다시 방문시 방문경로에 revisit을 입력하고, 방문자는 기간 내 총 튜플의 수로 구하고, 순방문자수는 방문경로에 revisit을 제외한 것으로 구함

```
<insert id="visitorCount">
    INSERT INTO VISITOR(IP,FUNNEL,DATE)
    (SELECT #{param1} AS IP, 'revisit' AS FUNNEL,NOW() AS DATE
    FROM VISITOR
    WHERE EXISTS (SELECT * FROM VISITOR WHERE DATE_BETWEEN DATE_SUB(NOW(), INTERVAL 1 HOUR) AND NOW() AND IP = #{param1}) AND IP = #{param1}
    ORDER BY DATE DESC
    LIMIT 1)
    UNION ALL
    (SELECT #{param1} AS IP, #{param2} AS FUNNEL, NOW() AS DATE
    FROM VISITOR
    WHERE NOT EXISTS (SELECT * FROM VISITOR WHERE DATE_BETWEEN DATE_SUB(NOW(), INTERVAL 1 HOUR) AND NOW() AND IP = #{param1}) AND IP = #{param1}
    ORDER BY DATE DESC
    LIMIT 1
    );
</insert>
```

sql의 exists문과 not exists

2개를 union all 해서 자바에서

if 문을 쓰지 않고도 sql에서

if문 효과를 낼 수 있다.

	visitor_id	ip	funnel	date
▶	6	127.0.0.1	homepage	2022-04-20 17:34:47
	9	127.0.0.1	homepage	2022-04-21 10:20:05
	10	127.0.0.1	homepage	2022-04-10 10:21:24
	11	127.0.0.1	homepage	2022-04-11 10:26:43
	13	127.0.0.1	homepage	2022-04-12 10:30:33
	14	192.168.150.246	homepage	2022-04-13 11:30:31
	15	192.168.150.246	homepage	2022-04-14 10:44:04
	26	192.168.150.246	homepage	2022-04-21 12:26:47
	27	192.168.150.246	homepage	2022-04-21 16:34:36
	28	127.0.0.1	homepage	2022-04-21 20:13:23
	29	127.0.0.1	revisit	2022-04-21 20:13:40
	30	127.0.0.1	revisit	2022-04-21 20:16:22
	31	127.0.0.1	revisit	2022-04-21 20:28:33
	32	127.0.0.1	revisit	2022-04-21 20:29:34
	33	127.0.0.1	revisit	2022-04-21 20:32:18
	34	127.0.0.1	revisit	2022-04-21 20:59:21
	35	127.0.0.1	revisit	2022-04-21 21:00:22
	36	127.0.0.1	revisit	2022-04-21 21:01:21
	37	127.0.0.1	revisit	2022-04-21 21:07:45
	38	127.0.0.1	revisit	2022-04-21 21:18:15
	39	127.0.0.1	revisit	2022-04-21 21:28:13
	40	127.0.0.1	revisit	2022-04-21 21:30:25
	41	127.0.0.1	revisit	2022-04-21 21:38:26
	42	127.0.0.1	homepage	2022-04-22 11:47:06
	43	127.0.0.1	googleSe...	2022-04-22 14:39:15
	44	127.0.0.1	googleSe...	2022-05-01 17:02:02
	45	127.0.0.1	revisit	2022-05-01 17:07:50

관리자 페이지



관리자 페이지

시작일: [] ~ [] 종료일: [] ~ [] 시간 간격: [] 일 [] []

날짜	방문자수	순방문자수	회원가입수	방문자대비 가입수	구매전환율 (방문자수대비)	구매전환율 (순방문자수대비)	최다 구매 상품	최다 구매 카테고리	최다 유입경로
2022-05-01	2	1	0	0.0%	0.0%	0.0%	>>	googleSearch	
2022-04-22	2	2	0	0.0%	0.0%	0.0%	>>	homepage	
2022-04-21	17	4	0	0.0%	0.0%	0.0%	>>	homepage	
2022-04-20	1	1	0	0.0%	0.0%	0.0%	>>	homepage	
2022-04-14	1	1	0	0.0%	0.0%	0.0%	>>	homepage	
2022-04-13	1	1	1	100.0%	2000.0%	2000.0%	CUTE	luggage >small > 소분류값을	homepage
2022-04-12	1	1	0	0.0%	0.0%	0.0%	>>	homepage	
2022-04-11	1	1	1	100.0%	1800.0%	1800.0%	CUTE	luggage >small > 소분류값을	homepage
2022-04-10	1	1	0	0.0%	500.0%	500.0%	CUTE	luggage >small > 소분류값을	homepage

```
7<select id="customerTrendTable" resultType="com.elapid.spring01.dto.customerTrendTableDto">
8    SELECT * FROM (
9        SELECT DATE_FORMAT(UO_DATE, '#param3') AS AA, COUNT(UO_ID) AS ORDERCOUNT, MAX(P_ID) AS P_ID
10       FROM USER_ORDER_DETAIL UOD,USER_ORDER UO
11      WHERE UO.UO_ID = UOD.UO_ID
12        AND DATE(UO_DATE) >= :STR_TO_DATE('#param1', '%Y-%m-%d')
13        AND DATE(UO_DATE) <= :STR_TO_DATE('#param2', '%Y-%m-%d')
14        GROUP BY AA
15    ) B,BB
16   RIGHT JOIN (
17     SELECT DATE_FORMAT(DATE, '#param3') AS BB, COUNT(VISITOR_ID) AS PV
18       FROM VISITOR
19      WHERE DATE(DATE) >= :STR_TO_DATE('#param1', '%Y-%m-%d')
20        AND DATE(DATE) <= :STR_TO_DATE('#param2', '%Y-%m-%d')
21      GROUP BY BB
22   ) A,AA
23 ON B.BB = A.AA
24 LEFT JOIN (
25   SELECT DATE_FORMAT(DATE, '#param3') AS CC, COUNT(VISITOR_ID) AS UV
26     FROM VISITOR
27    WHERE FUNNEL NOT IN ('REVISIT')
28      AND DATE(DATE) >= :STR_TO_DATE('#param1', '%Y-%m-%d')
29        AND DATE(DATE) <= :STR_TO_DATE('#param2', '%Y-%m-%d')
30      GROUP BY CC
31 ) ON B.BB = CC
32 LEFT JOIN (
33   SELECT DATE_FORMAT(U_REGISTERDATE, '#param3') AS DD, COUNT(U_ID) AS SIGNUPCOUNT
34     FROM USER
35    WHERE
36      DATE(U_REGISTERDATE) >= :STR_TO_DATE('#param1', '%Y-%m-%d')
37        AND DATE(U_REGISTERDATE) <= :STR_TO_DATE('#param2', '%Y-%m-%d')
38      GROUP BY DD
39 ) ON B.BB = DD
40 LEFT JOIN (
41   SELECT SPID,DATE,SPID_P_ID,PRO_P_NAME,CTG_MAIN,CTG_MIDDLE,CTG_SUB
42     FROM CATEGORY_CAT,CATEGORY_DETAIL CATD,PRODUCT PRO,
43      (SELECT DATE, MAX(P_ID) AS P_ID
44        FROM (
45          SELECT AA.AA AS DATE , MAX(ACONT) AS MAX_A
46            FROM (
47              (SELECT COUNT(P_ID) AS ACNT, P_ID, DATE_FORMAT(UO_DATE, '#param3') AS AA
48                FROM USER_ORDER_DETAIL UOD,USER_ORDER UO
49               WHERE UO.UO_ID = UOD.UO_ID
50                 AND DATE(UO_DATE) >= :STR_TO_DATE('#param1', '%Y-%m-%d')
51                   AND DATE(UO_DATE) <= :STR_TO_DATE('#param2', '%Y-%m-%d')
52                 GROUP BY AA.P_ID) AS A
53           GROUP BY AA.AA) DESC
54         WHERE CAT.CTG_ID = CATD.CTG_ID AND SPID.P_ID = CATD.P_ID AND PRO.P_ID = SPID.P_ID ) AS E
55   ON E.DATE = B.BB
56 LEFT JOIN (
57   (SELECT DATE, MAX(FUNNEL) AS FUNNEL
58     FROM (
59       SELECT BB AS DATE,MAX(PV) AS MAXPV, FUNNEL
60         FROM (
61           (SELECT DATE_FORMAT(DATE, '#param3') AS BB, COUNT(VISITOR_ID) AS PV , FUNNEL
62             FROM VISITOR
63            WHERE DATE(DATE) >= :STR_TO_DATE('#param1', '%Y-%m-%d')
64              AND DATE(DATE) <= :STR_TO_DATE('#param2', '%Y-%m-%d')
65              GROUP BY BB,FUNNEL) AS TMP1
66                GROUP BY TMP1.BB) AS TMP2
67                  GROUP BY DATE) AS F
68 ON F.DATE = B.BB
69 ORDER BY DATE DESC
70 )</select>
```

관리자 페이지

mysql query

SPRING version(DBCP2 방식)

- dao 인스턴스 생성시
Tomcat server의
context.xml에 저장된
Resource를 불러와
dataSource에 입력
- 메소드 하나 연결 할 때도
커넥션 쿼리준비문, 그리고
메모리 관리차원의 인스턴스
정리 (매번 같은 형식의
반복이 잦다.)

```
1 package com.elapid.spring01.dao;
2
3 *import java.sql.Connection;□
4
5 public class CartOrderDao {
6     //field
7     DataSource dataSource = null;
8
9     //constructor
10    public CartOrderDao() {
11        try {
12            Context context = new InitialContext();
13            dataSource = (DataSource)context.lookup("java:comp/env/jdbc/elapid");
14        } catch (Exception e) {
15            e.printStackTrace();
16        }
17
18        //회원가입 시 회원의 장바구니를 초기 생성
19        public void CartAdd(String uid) {
20            Connection connection = null;
21            PreparedStatement preparedStatement = null;
22
23            try {
24                connection = dataSource.getConnection();
25                String query = "insert into cart (u_id) values (?)";
26                preparedStatement = connection.prepareStatement(query);
27                preparedStatement.setString(1, uid);
28
29                preparedStatement.executeUpdate();
30            } catch (Exception e) {
31                e.printStackTrace();
32            } finally {
33                try {
34                    if(preparedStatement != null) preparedStatement.close();
35                    if(connection != null) connection.close();
36                } catch (Exception e) {
37                    e.printStackTrace();
38                }
39            }
40        }
41
42        // cdid으로 상품번호들을 검색
43        public ArrayList<Integer> searchBycdids(ArrayList<Integer> cd_ids) {
44            Connection conn = null;
45            PreparedStatement stmt = null;
46            ResultSet rs = null;
47
48            ArrayList<Integer> p_ids = new ArrayList<Integer>();
49
50            try {
51                conn = dataSource.getConnection();
52
53                for(int i = 0 ; i < cd_ids.size() ; i++) {
54
55                    String query = "select p_id from cart_detail where cd_id = ?";
56                    stmt = conn.prepareStatement(query);
57                    stmt.setInt(1, cd_ids.get(i));
58                    rs = stmt.executeQuery();
59
60                    while(rs.next()) {
61                        p_ids.add(rs.getInt("p_id"));
62                    }
63                }
64            } catch (Exception e) {
65                e.printStackTrace();
66            } finally {
67                try {
68                    if(stmt != null) stmt.close();
69                    if(rs != null) rs.close();
70                    if(conn != null) conn.close();
71                } catch (Exception e) {
72                    e.printStackTrace();
73                }
74            }
75        }
76    }
77}
```

SPRING version(jdbc 방식)

servlet-context.xml

```
<context:component-scan base-packages="com.elapid.spring01" />

<beans:bean name="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <beans:property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
    <beans:property name="url" value="jdbc:mysql://localhost:3306/elapid?serverTimezone=Asia/Seoul&characterEncoding=utf8&useSSL=false"/>
    <beans:property name="username" value="root"/>
    <beans:property name="password" value="quer1234"/>
</beans:bean>

<beans:bean name="template" class="org.springframework.jdbc.core.JdbcTemplate">
    <beans:property name="dataSource" ref="dataSource" />
</beans:bean>
```

ktecontroller.java(DI)

```
private JdbcTemplate template;

@Autowired
public void setTemplate(JdbcTemplate template) {
    this.template = template;
    Constant.template = this.template;
}
```

constant.java

```
package com.elapid.spring01.util;
import org.springframework.jdbc.core.JdbcTemplate;
public class Constant {
    public static JdbcTemplate template;
}
```

UserLogDao.java(jdbc template)

```
1 package com.elapid.spring01.dao;
2
3 import java.sql.Connection;
4
5 public class UserLogDao {
6     // DataSource dataSource = null;
7     JdbcTemplate template;
8
9     public UserLogDao() {
10        // TODO Auto-generated constructor stub
11
12        this.template = Constant.template;
13    }
14    public void userLoginLogAdd(final String uid) {
15        this.template.update(new PreparedStatementCreator() {
16
17            @Override
18            public PreparedStatement createPreparedStatement(Connection con) throws SQLException {
19                // TODO Auto-generated method stub
20                String query = "Insert into login_log (u_id,lo_date) values (?,now())";
21                PreparedStatement ps = con.prepareStatement(query);
22                ps.setString(1, uid);
23
24                return ps;
25            }
26        });
27    }
28}
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

pom.xml

```
<!-- JDBC -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>4.1.4.RELEASE</version>
</dependency>
```

SPRING version(mybatis 방식)

servlet-context.xml

```
<context:component-scan base-package="com.elapid.spring01" />

<beans:bean name="dataSource" class = "org.springframework.jdbc.datasource.DriverManagerDataSource">
    <beans:property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
    <beans:property name="url" value="jdbc:mysql://localhost:3306/elapid?serverTimezone=Asia/Seoul&characterEncoding=utf8&useSSL=false"/>
    <beans:property name="username" value="root"/>
    <beans:property name="password" value="qwer1234"/>
</beans:bean>

<beans:bean name="sqlSessionFactory" class = "org.mybatis.spring.SqlSessionFactoryBean">
    <beans:property name="dataSource" ref = "dataSource" />
    <beans:property name="mapperLocations" value="classpath:com/elapid/spring01/dao/mapper/*.xml" />
</beans:bean>

<beans:bean name="sqlSession" class = "org.mybatis.spring.SqlSessionTemplate">
    <beans:constructor-arg index="0" ref="sqlSessionFactory" />
</beans:bean>
```

ktecontroller.java(DI)

```
44    @Autowired
45    private SqlSession sqlSession;
46
47    @Autowired
48    public void Auto(
49        ECommand eLoginCheck, ECommand eGoogleLogin, ECommand eIdCheck, ECommand eRegisterCheck, ECommand eMyPage,
50        ECommand eProfileModify, ECommand eProfileDelete, ECommand eAdminDashboardTable
51    ) {
52    }
53
54    this.eLoginCheckCommand = eLoginCheck;
55    this.eGoogleLoginCommand = eGoogleLogin;
56    this.eIdCheckCommand = eIdCheck;
57    this.eRegisterCheckCommand = eRegisterCheck;
58    this.eMyPageCommand = eMyPage;
59    this.eProfileModifyCommand = eProfileModify;
60    this.eProfileDeleteCommand = eProfileDelete;
61    this.eAdminDashboardTableCommand = eAdminDashboardTable;
62
63
64    //로그인 체크
65    @RequestMapping("loginCheck")
66    public String loginCheck(HttpServletRequest request, Model model) {
67        model.addAttribute("request", request);
68        eLoginCheckCommand.execute_session(sqlSession, model, session, request);
69
70        return (String)request.getAttribute("loginviewparam");
71    }
72
```

UserDao.java

```
1 package com.elapid.spring01.dao;
2
3 import com.elapid.spring01.dto.UserDto;
4
5 public interface UserDao {
6
7     public int loginCheck(String uid, String upassword);
8     public String nameReturn(String uid);
9     public void userAdd(String uid, String upwd, String uname, String uemail, String utel, String ugender, String ubirthdate);
10    public UserDto profileView(String uid);
11    public void profileModify(String uemail, String upwd, String utel, String ugender, String ubirthdate, String uid);
12    public void profileDelete(String uid);
13
14 }
```

UserDao.xml(mapper)

```
6 <mapper namespace="com.elapid.spring01.dao.UserDao" ><!-- 저번 파일을 바로 --->
7     <select id="loginCheck" resultType="int">
8         SELECT COUNT(*) FROM USER WHERE U_ID = #{param1} AND U_PWD = #{param2};
9     </select>
10    <select id="nameReturn" resultType="string">
11        SELECT U_NAME FROM USER WHERE U_ID = #{param1};
12    </select>
13    <insert id="userAdd" >
14        INSERT INTO USER (U_ID, U_PWD, U_NAME, U_EMAIL, U_TEL, U_GENDER, U_BIRTHDATE, U_POINT, U_GRADE, U_REGISTERDATE ) VALUES (#{param1},#{param2},#{param3},#{param4},#{param5},#{param6});
15    </insert>
16    <select id="profileView" resultType="com.elapid.spring01.dto.UserDto">
17        SELECT * FROM USER WHERE U_ID = #{param1};
18    </select>
19    <update id="profileModify" >
20        UPDATE USER SET U_EMAIL = #{param1}, U_PWD = #{param2}, U_TEL = #{param3}, U_GENDER = #{param4}, U_BIRTHDATE = #{param5} WHERE U_ID = #{param6};
21    </update>
22    <delete id="profileDelete" >
23        DELETE FROM USER WHERE U_ID = #{param1};
24    </delete>
25 </mapper>
```

pom.xml

```
<!-- mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.2.8</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.2.2</version>
</dependency>
```

도로명 API, 구글 API

주소정보연계 | 도로명주소 안내시스템
juso.go.kr/addrlink/addrLinkUrlSearch.do

호텔신라

변동된 주소정보 포함 예시 : 도로명(반포대로 58), 건물명(독립기념관), 지번(삼성동 25)

도로명주소 검색 결과 (9건) 정확도순 도로명 포함 지번 포함

No	도로명주소	우편번호
1	제주특별자치도 제주시 노연로 69(연동) (지번) 제주특별자치도 제주시 연동 252-18 (주)호텔신라 제주면세점	<input type="button" value="상세보기"/> 63131
2	서울특별시 중구 동화로 249(장충동2가) (지번) 서울특별시 중구 장충동2가 202 신라호텔	<input type="button" value="상세보기"/> 04605
3	경상남도 양산시 하북면 통도8길 11 (지번) 경상남도 양산시 하북면 치산리 17-44 신라호텔	<input type="button" value="상세보기"/> 50502
4	경상남도 청녕군 부곡면 온천중앙로 19 (지번) 경상남도 청녕군 부곡면 거문리 215-15 신라호텔	<input type="button" value="상세보기"/> 50365
5	제주특별자치도 서귀포시 중문관광로72번길 75(색달동) (지번) 제주특별자치도 서귀포시 색달동 3039-3 신라호텔	<input type="button" value="상세보기"/> 63535

1 2 

ELAPID 카리어 백팩 주요기능 트렌드 프레스 심플문의 회원 접수/나니 상품등 Search Search

sign in

아이디:
비밀번호:

로그인

계정 선택
태스트용 구글로그인(1)로 이용

Tue Eon Kim  로그인됨
TMH DMB  로그인됨
 다른 계정 사용

계속 진행하기 위해 Google에서 나의 이름, 이메일 주소, 언어 환경 설정, 프로필 사진을 테스트용 구글로그인(1)에 공유합니다.

ELAPID's Work copyright. 회원 ▾ 도움말 개인정보처리방침 약관