

보안 프로그래밍 1분반 2조

# 영화데이터 추출 및 업무 자동화

2024.12.10

---

## 01 제작동기

### 선정 이유

- 수업에서 엑셀 파일을 읽어들이고 데이터를 추출 할 때 파이썬으로 업무 자동화
- 어떤 방식으로 데이터를 추출하고 다시 엑셀로 정리하는지 과정을 공부하고 싶어서 이러한 주제를 정함


### 소감



- 프로젝트를 진행하면서 AI학습에서 데이터 분석의 중요성을 실제 데이터 처리 프로젝트를 통해 체득할 수 있는 시간이었음




## 데이터 출처





### 문화 빅데이터


 문화 빅데이터 플랫폼

 로그인  회원가입

[데이터 상품](#) [데이터 상담소](#) [데이터 분석](#) [문화 서비스](#) [데이터 활용](#) [플랫폼 이용안내](#)

[홈](#) · [데이터 상품](#) · [전체 상품](#) 



    [URL 복사](#)



 KCISA  
한국문화정보원

[문화예술](#) [공지 및 관련자료](#) [여가](#) [CSV](#) [무료](#)

### KOBIS 박스오피스 영화정보

[한국문화정보원 >](#)

유형 CSV · 가격 무료 · 데이터 갱신주기 Monthly 2024.05.21 업데이트  10055  4547

평점 ★★★★★ 3.6 [평가하기](#) 5  [평가하기](#) 관심  31

### 영화진흥위원회

 영화진흥위원회  
Korean Film Council

[이용안내](#) | [OPEN API](#) | [키 발급/관리](#) [로그인](#)

# OPEN API 서비스

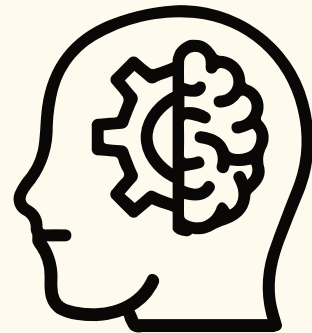
영화진흥위원회 영화관입장권통합전산망에서 제공하는 오픈API 서비스로  
더욱 풍요롭고 편안한 영화 서비스를 즐겨보세요.

## 오픈API 제공서비스



양승권

조건 별 검색 및 리스트 저장



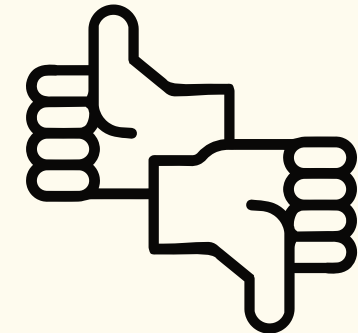
김태헌

장르별, 영화별  
분포도 그래프 추출 및  
전체적인 오류 보완



곽남호

장르 별 평균 관객 수 , 평균 매출  
영화 등급 별 총 매출, 총 영화 수



전성배

흥행 예측 (결정 트리)  
영화 추천  
데이터 합산



## 양승권

```
# PDF 저장 함수
def save_results_to_pdf(result):
    pdfmetrics.registerFont(TTFont('Malgun', 'C:\\Windows\\Fonts\\malgun.ttf'))
    pdf_filename = "search_results.pdf"
    pdf = SimpleDocTemplate(pdf_filename, pagesize=letter)
    elements = []

    # 필요 열 추출
    selected_columns = ['번호', '영화명', '감독', '제작사명', '날짜', '장르']
    result = result[selected_columns]

    # 표 스타일 설정
    styles = getSampleStyleSheet()
    styles['BodyText'].fontName = 'Malgun'
    cell_style = styles['BodyText']
    cell_style.wordWrap = 'CJK'

    # 데이터 준비
    data = [list(result.columns)] + [
        [
            Paragraph(str(cell), cell_style) if isinstance(cell, str) else cell
            for cell in row
        ]
        for row in result.values.tolist()
    ]

    # 열 너비 고정
    column_widths = [40, 100, 100, 100, 50, 50]
    table = Table(data, colWidths=column_widths)

    # 테이블 스타일 설정
    style = TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('FONTNAME', (0, 0), (-1, -1), 'Malgun'),
        ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),
    ])
    table.setStyle(style)
    elements.append(table)

    pdf.build(elements)
    print(f"검색 결과가 '{pdf_filename}'로 저장되었습니다.")
```

다량의 결과를 보기쉽게 출력하기 위해  
reportlab 라이브러리를 이용해 결과 리스트를  
표로 나타낸 pdf를 만들도록 작성했습니다

## PDF

| 번호  | 영화명             | 감독명       | 제작사명                               | 개봉년도 | 장르명 |
|-----|-----------------|-----------|------------------------------------|------|-----|
| 751 | 남산의 부장들         | 우민호       | (주)하이프미디어코프                        | 2020 | 드라마 |
| 757 | #살아있다           | 조일형       | 영화사 집.(주)퍼스팩티브픽처스                  | 2020 | 드라마 |
| 758 | 강철비2: 정상회담      | 양우석       | (주)스튜디오게니우스우정                      | 2020 | 드라마 |
| 759 | 담보              | 강대규       | (주)제이케이필름,(주)레드로버,(주)씨제이엔엠,(주)영화사연 | 2020 | 드라마 |
| 761 | 삼진그룹 영어토익반      | 이종필       | 더컬프(주)                             | 2020 | 드라마 |
| 768 | 결백              | 박상현       | (주)이디오플랜                           | 2020 | 드라마 |
| 774 | 1917            | 샘 멘데스     | nan                                | 2020 | 드라마 |
| 779 | 작은 아씨들          | 그레타 거윅    | nan                                | 2020 | 드라마 |
| 785 | 시동              | 최정열       | (주)외유내강                            | 2020 | 드라마 |
| 804 | 이웃사촌            | 이환경       | (주)시네마허브,한타지엔터테인먼트                 | 2020 | 드라마 |
| 820 | 내가 죽던 날         | 박지완       | 오스카10스튜디오                          | 2020 | 드라마 |
| 821 | 기생충             | 봉준호       | (주)바른손이엔케이                         | 2020 | 드라마 |
| 827 | 밤엘: 세상을 바꾼 복탄신언 | 제이 로치     | nan                                | 2020 | 드라마 |
| 831 | 타오르는 여인의 초상     | 셀린 시아마    | nan                                | 2020 | 드라마 |
| 837 | 다크 워터스          | 토드 헤인즈    | nan                                | 2020 | 드라마 |
| 861 | 저 산 너머          | 최종태       | 리온픽처스(주)                           | 2020 | 드라마 |
| 889 | 주디              | 루퍼트 굴드    | nan                                | 2020 | 드라마 |
| 900 | 에어로너즈           | 통 하퍼      | nan                                | 2020 | 드라마 |
| 953 | 피아니스트의 전설       | 쥬세페 토르나토레 | nan                                | 2020 | 드라마 |
| 956 | 아구소녀            | 최윤태       | 한국영화아카데미                           | 2020 | 드라마 |
| 962 | 통보이             | 셀린 시아마    | nan                                | 2020 | 드라마 |
| 964 | 미스비헤이버          | 필립파 로소프   | nan                                | 2020 | 드라마 |
| 970 | 찬실이는 복도 많지      | 김초희       | (주)지이프로덕션,(주)사이드미러                 | 2020 | 드라마 |
| 980 | 마리 퀴리           | 마르잔 사트라피  | nan                                | 2020 | 드라마 |
| 983 | 블루 아워           | 하코타 유코    | nan                                | 2020 | 드라마 |
| 986 | 테슬라             | 마이클 알메레이다 | nan                                | 2020 | 드라마 |

## 검색

```
# 검색 및 인터페이스 함수
def search_movie(df):
    print("\n조건별 영화 검색")
    print("1. 영화명으로 검색")
    print("2. 감독명으로 검색")
    print("3. 제작사명으로 검색")
    print("4. 개봉년도 범위별 검색")
    print("5. 장르별 검색")
    print("0. 종료")

    filters = []
    selected_columns = ['번호', '영화명', '감독', '제작사명', '날짜', '장르']

    while True:
        try:
            print("\n새로운 조건을 추가하려면 숫자를 입력하세요.")
            choice = int(input("원하는 검색 옵션을 선택하세요 (0~5): "))

            if choice == 0:
                print("검색 조건 입력을 종료합니다.")
                break
            elif choice == 1:
                keyword = input("영화명을 입력하세요: ")
                filters.append(('영화명', keyword))
            elif choice == 2:
                keyword = input("감독명을 입력하세요: ")
                filters.append(('감독', keyword))
            elif choice == 3:
                keyword = input("제작사명을 입력하세요: ")
                filters.append(('제작사명', keyword))
            elif choice == 4:
                start_year = int(input("검색할 시작 연도를 입력하세요 (예: 2010): "))
                end_year = int(input("검색할 종료 연도를 입력하세요 (예: 2020): "))
                filters.append(('개봉년도범위', (start_year, end_year)))
            elif choice == 5:
                keyword = input("장르를 입력하세요: ")
                filters.append(('장르', keyword))
            else:
                print("올바른 옵션을 선택하세요.")
                continue

            continue_choice = input("새로운 조건을 추가하려면 숫자를 입력하세요. (0~5): ")
            if continue_choice != 'y':
                print("조건 추가가 완료되었습니다.")
                break
        except Exception as e:
            print(f"오류 발생: {e}")
            continue

    result = df.copy()
    for filter_type, value in filters:
        if filter_type == '영화명':
            result = result[result['영화명'].str.contains(value, na=False, case=False)]
        elif filter_type == '감독명':
            result = result[result['감독'].str.contains(value, na=False, case=False)]
        elif filter_type == '제작사명':
            result = result[result['제작사명'].str.contains(value, na=False, case=False)]
        elif filter_type == '개봉년도범위':
            start_year, end_year = value
            result = result[(result['날짜'] >= start_year) & (result['날짜'] <= end_year)]
        elif filter_type == '장르':
            result = result[result['장르'].str.contains(value, na=False, case=False)]

    if result.empty:
        print("검색 결과가 없습니다.")
    else:
        print("검색 결과:")
        print(result[selected_columns].head(100).to_string(index=False))
        save_results_to_pdf(result.head(100))
```

Squeezed text (202 lines).  
검색 결과가 'search\_results.pdf'로 저장되었습니다.

xlsx 파일의 데이터를 읽고 이를 분류하여 사용하기 위해,  
pandas 라이브러리를 활용하여 코드를 작성하였습니다.

# 그래프 함수: 장르별 영화 분포

```
def plot_genre_distribution(df):  
    genre_counts = df['장르명'].value_counts()  
    plt.figure(figsize=(10, 6))  
    sns.barplot(x=genre_counts.values, y=genre_counts.index, palette='viridis')  
    plt.title("장르별 영화 분포", fontsize=16)  
    plt.xlabel("영화 수", fontsize=12)  
    plt.ylabel("장르", fontsize=12)  
    plt.tight_layout()  
    plt.show()
```

장르별 그래프

# 그래프 함수: 연도별 영화 수

```
def plot_movies_by_year(df):  
    year_counts = df['개봉년도'].value_counts().sort_index()  
    plt.figure(figsize=(10, 6))  
    sns.barplot(x=year_counts.index, y=year_counts.values, palette='coolwarm')  
    plt.title("연도별 영화 수", fontsize=16)  
    plt.xlabel("개봉 연도", fontsize=12)  
    plt.ylabel("영화 수", fontsize=12)  
    plt.xticks(rotation=45)  
    plt.tight_layout()  
    plt.show()
```

연도별 그래프

```
# 한글 폰트 설정 (모든 그래프에 적용)
```

```
font_path = "C:/Windows/Fonts/malgun.ttf"
```

```
font_prop = fm.FontProperties(fname=font_path)
```

```
rc('font', family=font_prop.get_name())
```

```
rc('axes', unicode_minus=False)
```

0. 종료

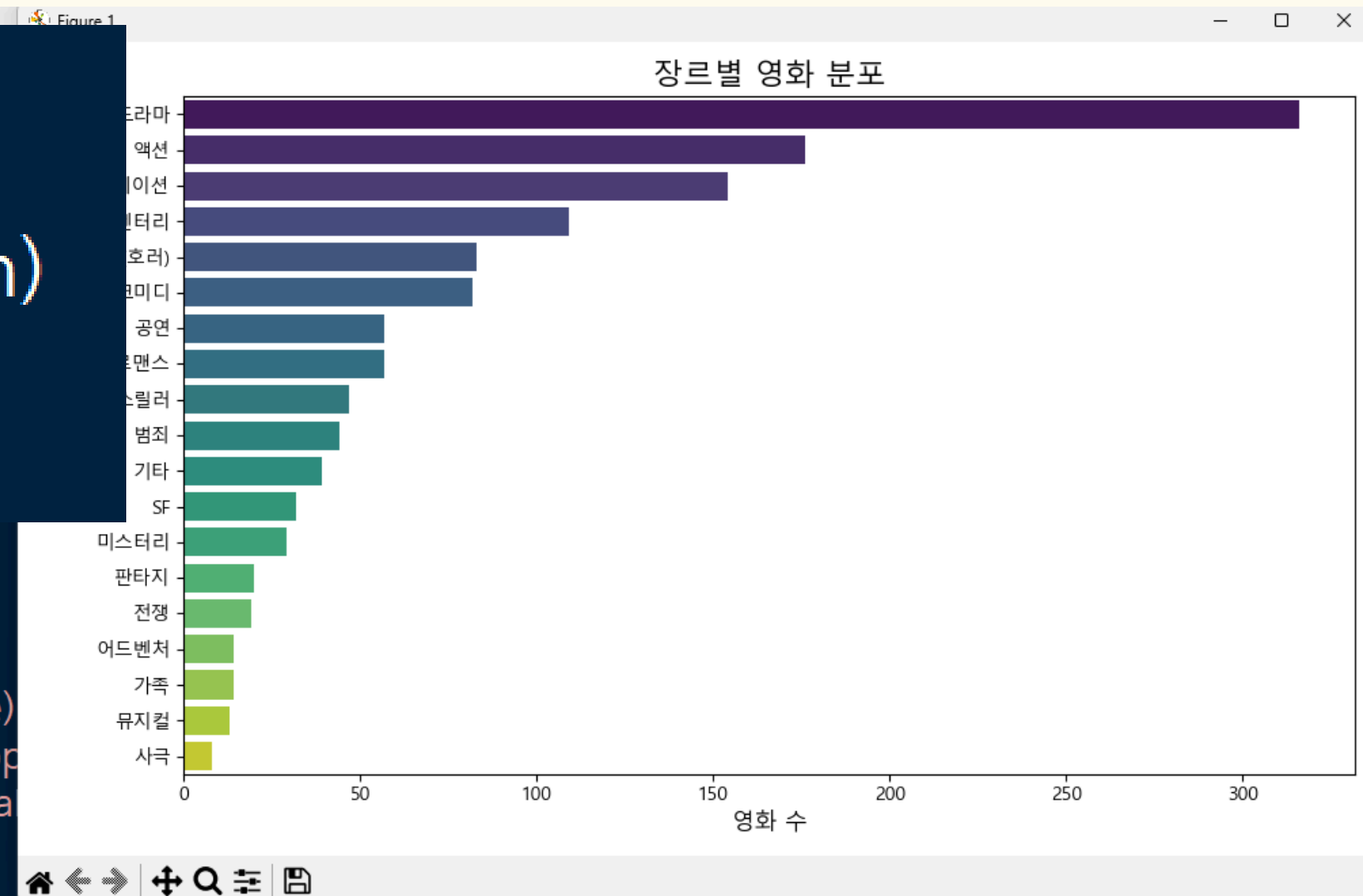
입력: 2

Warning (from warnings module)

File "C:\Users\gorde\Desktop\sns.barplot(x=genre\_counts.v

FutureWarning:

Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'y' variable to 'hue' and set 'legend=False' for the same effect.



폰트 설정

빨간줄 오류

```
elif a == 3:
    break
else:
    print("찾으시는 기능은 없는 기능입니다.")
except Exception as e:
    print(f"파일을 읽는 중 오류 발생: {e}")
    exit()
```

```
def movie_data(df):
    print("\n알고 싶은 기능을 선택하세요: ")
    print("1. 장르별 평균 매출과 평균 관람 인원")
    print("2. 관람 등급별 영화 수와 총 매출")
    print("3. 종료")
```

```
# 함수: 값 단위를 천만, 억, 십억 등으로 변환
def format_yaxis(value, tick_number):
    if value >= 1e8: # 1억 이상
        return f'{value / 1e8:.1f}억'
    elif value >= 1e7: # 1천만 이상
        return f'{value / 1e7:.1f}천만'
    elif value >= 1e6: # 1백만 이상
        return f'{value / 1e6:.1f}백만'
    elif value >= 1e4: # 1만 이상
        return f'{value / 1e4:.1f}만'
    else: # 1만 미만
        return str(value)
```

그래프  
단위



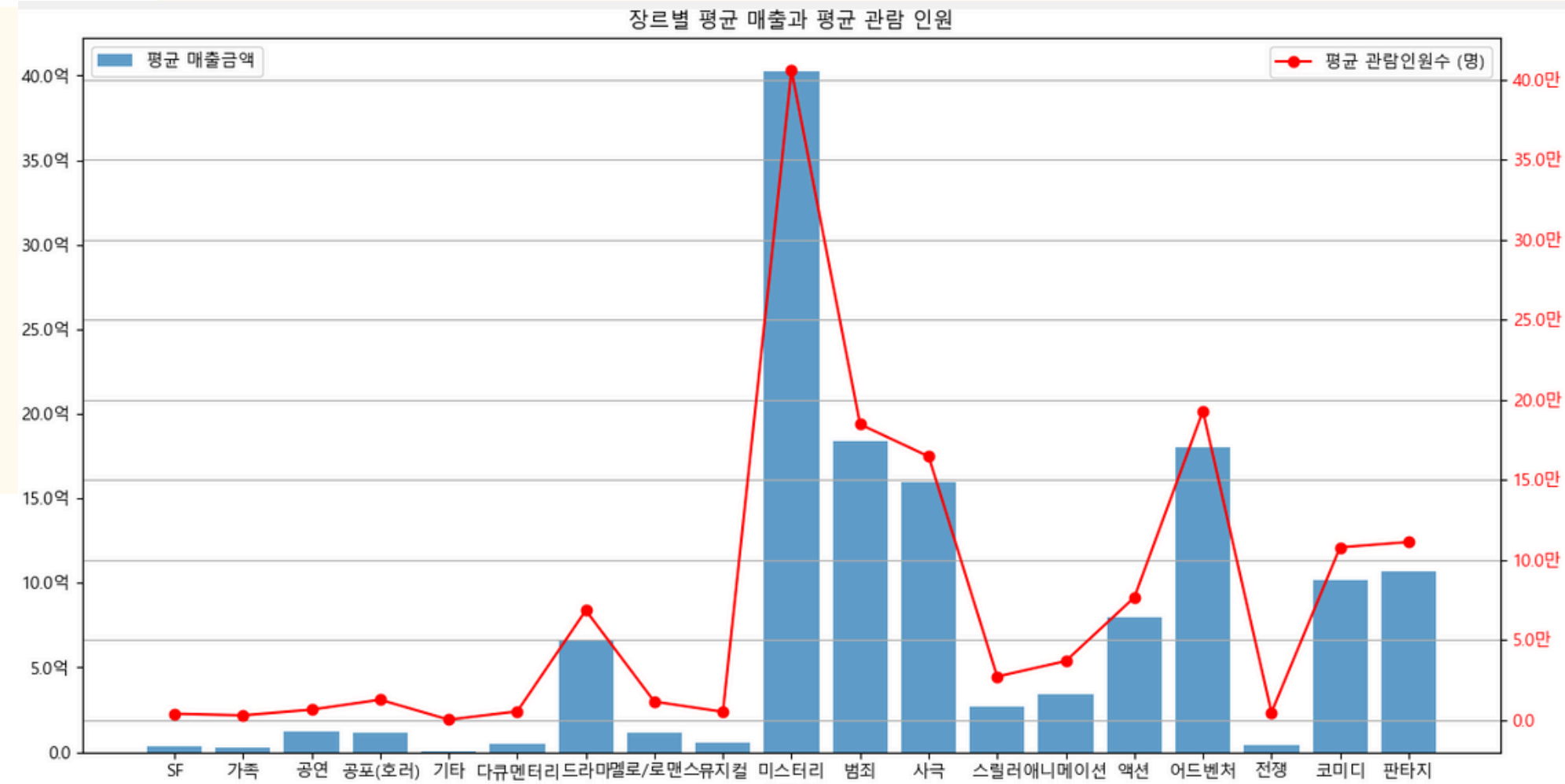
```
# 장르별 평균 매출과 평균 관람 인원 계산
genre_mean = df.groupby("장르명").agg({
    "매출금액": "mean",
    "관람인원수": "mean"
}).rename(columns={"매출금액": "평균 매출금액", "관람인원수": "평균 관람인원수"})
```

```
# 시각화: 장르별 평균 매출과 평균 관람 인원
fig, ax1 = plt.subplots(figsize=(20, 10))
```

```
# 평균 매출금액 그래프 (막대)
bar = ax1.bar(genre_mean.index, genre_mean["평균 매출금액"], label="평균 매출금액", alpha=0.7)
ax1.set_ylabel('평균 매출금액')
ax1.yaxis.set_major_formatter(FuncFormatter(format_yaxis)) # 단위 포맷터 적용
```

```
# 평균 관람 인원수 그래프 (꺾은선)
ax2 = ax1.twinx()
ax2.plot(genre_mean.index, genre_mean["평균 관람인원수"], color='red', marker='o', label="평균 관람인원수 (명)")
ax2.set_ylabel('평균 관람 인원수')
ax2.tick_params(axis='y', labelcolor='red')
ax2.yaxis.set_major_formatter(FuncFormatter(format_yaxis))
```

```
# 그래프 꾸미기
plt.title('장르별 평균 매출과 평균 관람 인원')
ax1.set_xlabel('장르명')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



# 곽남호

## 등급별 그래프

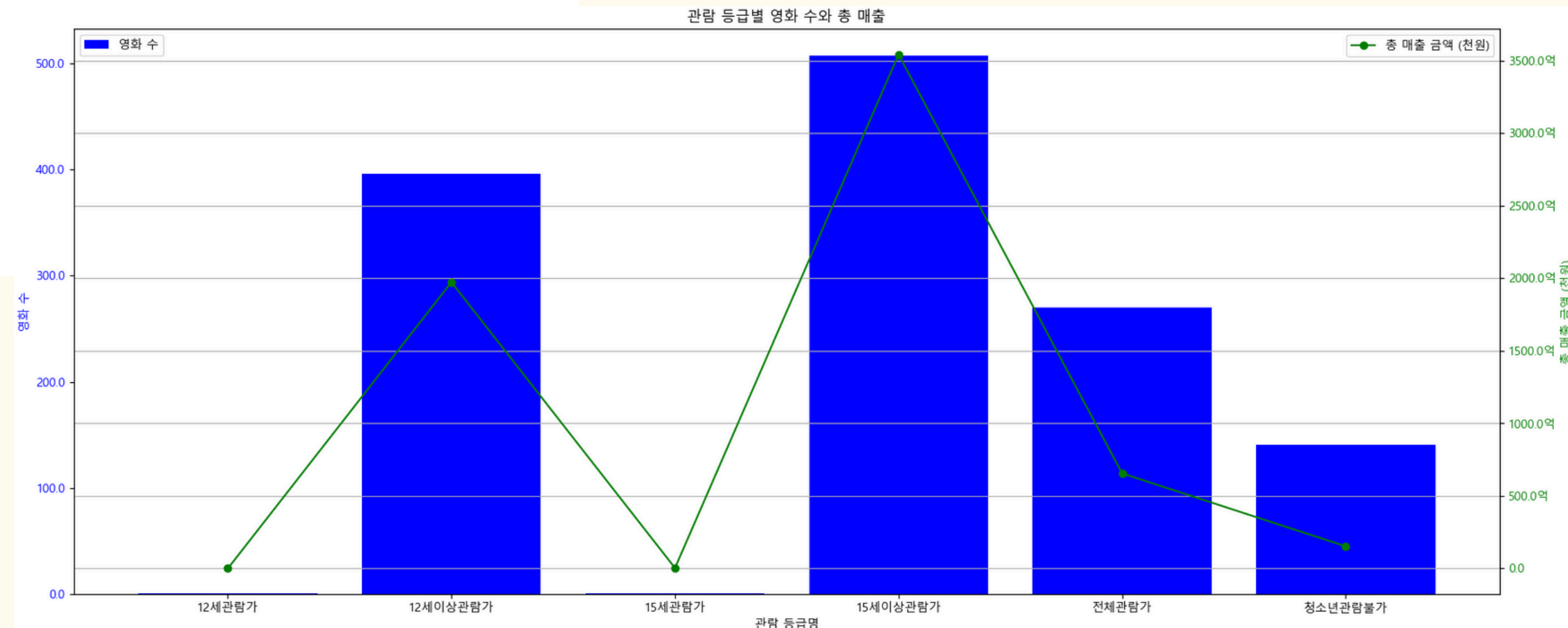
```
elif a == 2:
    ## 관람 등급별 영화 수와 총 매출 계산
    grade_total = df.groupby("등급명").agg({
        "영화명": "count",
        "매출금액": "sum"
    }).rename(columns={"영화명": "영화 수", "매출금액": "총 매출 금액"})

    # 시각화: 관람 등급별 영화 수와 총 매출
    fig, ax1 = plt.subplots(figsize=(20, 10))

    # 관람 등급 별 영화 수
    ax1.bar(grade_total.index, grade_total["영화 수"], color='blue', label='영화 수')
    ax1.set_xlabel('관람 등급명')
    ax1.set_ylabel('영화 수', color='blue')
    ax1.tick_params(axis='y', labelcolor='blue')
    ax1.yaxis.set_major_formatter(FuncFormatter(format_yaxis))

    #관람 등급 별 총 매출
    ax2 = ax1.twinx()
    ax2.plot(grade_total.index, grade_total["총 매출 금액"], color='green', marker='o', label='총 매출 금액 (천원)')
    ax2.set_ylabel('총 매출 금액 (천원)', color='green')
    ax2.tick_params(axis='y', labelcolor='green')
    ax2.yaxis.set_major_formatter(FuncFormatter(format_yaxis))

    # 그래프 꾸미기
    plt.title('관람 등급별 영화 수와 총 매출')
    ax1.legend(loc='upper left')
    ax2.legend(loc='upper right')
    plt.xticks(rotation=45)
    plt.grid(True)
    fig.tight_layout()
    plt.show()
```



# 전성배

## API

```
# 주간 박스오피스 데이터 요청 함수
def fetch_weekly_box_office_data(target_date, week_gbn="0"):

    base_url = "http://www.kobis.or.kr/kobisopenapi/webservice/rest/boxoffice/searchWeeklyBoxOfficeList.xml"
    params = {
        "key": API_KEY,
        "targetDt": target_date,
        "weekGb": week_gbn # 주간(0), 주말(1)
    }
    response = requests.get(base_url, params=params)

    if response.status_code == 200:
        root = ET.fromstring(response.content)
        movies = root.findall("./weeklyBoxOffice")
        data = []

        for movie in movies:
            data.append({
                "날짜": target_date,
                "순위": movie.find("rank").text,
                "영화명": movie.find("movieNm").text,
                "대표 코드": movie.find("movieCd").text,
                "누적 매출액": movie.find("salesAcc").text if movie.find("salesAcc") is not None else "0",
                "누적 관객수": movie.find("audiAcc").text if movie.find("audiAcc") is not None else "0",
                "스크린 수": movie.find("scrnCnt").text if movie.find("scrnCnt") is not None else "0"
            })

        return pd.DataFrame(data)
    else:
        print(f"API 요청 실패: 상태 코드 {response.status_code}")
        print(f"응답 내용: {response.text}")
        return pd.DataFrame()
```

### 3. 인터페이스

• 요청 인터페이스

| 요청 변수        | 값       | 설명   |
|--------------|---------|--|
| key          | 문자열(필수) | 발급받은키 값을 입력합니다.  |
| targetDt     | 문자열(필수) | 조회하고자 하는 날짜를 yyyyymmdd 형식으로 입력합니다.   |
| itemPerPage  | 문자열     | 결과 ROW 의 개수를 지정합니다.(default : "10", 최대 : "10")                                     |
| multiMovieYn | 문자열     | 다양성 영화/상업영화를 구분지어 조회할 수 있습니다.<br>"Y" : 다양성 영화 "N" : 상업영화 (default : 전체)            |
| repNationCd  | 문자열     | 한국/외국 영화별로 조회할 수 있습니다.<br>"K" : 한국영화 "F" : 외국영화 (default : 전체)                     |
| wideAreaCd   | 문자열     | 상영지역별로 조회할 수 있으며, 지역코드는 공통코드 조회 서비스에서 "0105000000" 로서 조회된 지역 코드입니다. (default : 전체) |

주간 박스오피스 데이터 수집  
특정 날짜와 주간/주말 구분하여 데이터 수집  
데이터를 XML로 받아 DataFrame 형식으로 반환

## 전성배

### API

```
# 영화 상세 정보 API 호출 함수
def fetch_movie_details(movie_cd):

    base_url = "http://www.kobis.or.kr/kobisopenapi/webservice/rest/movie/searchMovieInfo.xml"
    params = {
        "key": API_KEY,
        "movieCd": movie_cd
    }
    response = requests.get(base_url, params=params)

    if response.status_code == 200:
        root = ET.fromstring(response.content)
        movie_info = root.find("./movieInfo")
        if movie_info is not None:
            # 데이터 파싱
            return {
                "영화 코드": movie_cd,
                "영화 개봉일": movie_info.find("openDt").text if movie_info.find("openDt") is not None else "N/A",
                "제작 연도": movie_info.find("prdtYear").text if movie_info.find("prdtYear") is not None else "N/A",
                "장르": ", ".join(
                    [genre.find("genreNm").text for genre in movie_info.findall("./genres/genre")]
                ),
                "감독": ", ".join(
                    [director.find("peopleNm").text for director in movie_info.findall("./directors/director")]
                ),
                "제작사명": ", ".join(
                    [company.find("companyNm").text for company in movie_info.findall("./companys/company")]
                ),
                "영화 등급": ", ".join(
                    [audit.find("watchGradeNm").text for audit in movie_info.findall("./audits/audit")]
                )
            }
        else:
            return {"영화 코드": movie_cd, "에러": "영화 정보가 없습니다."}
    else:
        print(f"영화 상세 정보 요청 실패: 상태 코드 {response.status_code}, movieCd: {movie_cd}")
        return {"영화 코드": movie_cd, "에러": "API 요청 실패"}
```

영화의 고유 코드 (movieCd)를 이용해 영화  
의 상세 정보 요청  
필요한 정보를 추출하여 딕셔너리 형태로 반환

## 전성배

### API

```
# 2020년부터 2024년까지 주간 데이터와 상세 정보 수집
def main():
    # 결과를 저장할 데이터프레임
    all_results = pd.DataFrame()

    # 2020년부터 2024년까지 반복
    for year in range(2020, 2025):
        print(f"#{year}년 데이터 수집 중...")

        # 1월부터 12월까지 반복
        for month in range(1, 13):
            target_date = f"{year}{month:02}01" # 매월 1일 기준으로 데이터 요청
            print(f"{target_date} 주간 데이터 수집 중...")

            # 박스오피스 데이터 요청
            df = fetch_weekly_box_office_data(target_date)
            if not df.empty:
                # 영화 상세 정보 추가
                for idx, row in df.iterrows():
                    movie_details = fetch_movie_details(row["대표 코드"])
                    df.loc[idx, "영화 개봉일"] = movie_details.get("영화 개봉일", "N/A")
                    df.loc[idx, "제작 연도"] = movie_details.get("제작 연도", "N/A")
                    df.loc[idx, "장르"] = movie_details.get("장르", "N/A")
                    df.loc[idx, "감독"] = movie_details.get("감독", "N/A")
                    df.loc[idx, "제작사명"] = movie_details.get("제작사명", "N/A")
                    df.loc[idx, "영화 등급"] = movie_details.get("영화 등급", "N/A")
                    time.sleep(0.5) # API 호출 간격

                all_results = pd.concat([all_results, df], ignore_index=True)

    # 데이터 저장
    output_file = r"C:\Python38\work\movie.csv"
    all_results.to_csv(output_file, index=False, encoding="utf-8-sig")
    print(f"#{year}년 데이터 수집 완료. 결과가 '{output_file}'에 저장되었습니다.")
```

연도별 (2020 ~ 2024), 월별 (1~12) 데이터를  
수집하고, 각 영화의 상세 정보 추가  
모든 데이터 누적한 뒤 csv 파일로 저장



## 전성배

### 데이터 로드 및 전처리

```
csv_file_path = "movie.csv"
df = pd.read_csv(csv_file_path)

# CSV 데이터 처리
columns_to_keep = ['영화명', '개봉년도', '매출금액', '장르명', '등급명']
df = df[columns_to_keep]

# 데이터 전처리
df.fillna(0, inplace=True)
df['개봉 월'] = pd.to_datetime(df['개봉년도'], format='%Y%m%d', errors='coerce').dt.month.fillna(0).astype(int)
df[' 흥행'] = (df['매출금액'] > df['매출금액'].mean()).astype(int)
df_encoded = pd.get_dummies(df, columns=['장르명', '등급명'], drop_first=True)
```

### 모델 학습

```
x = df_encoded[['개봉년도', '개봉 월']] + [col for col in df_encoded.columns if col.startswith('장르명_') or col.startswith('등급명_')]
y = df_encoded[' 흥행']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

model = DecisionTreeClassifier(random_state=0)
model.fit(x_train, y_train)
```

### 흥행 예측 함수

```
def predict_success(genre, year, month, rating):
    new_data = pd.DataFrame(data=[[0] * len(x.columns)], columns=x.columns)
    new_data['개봉년도'] = year
    new_data['개봉 월'] = month
    if f'장르명_{genre}' in new_data.columns:
        new_data[f'장르명_{genre}'] = 1
    if f'등급명_{rating}' in new_data.columns:
        new_data[f'등급명_{rating}'] = 1
    prediction = model.predict(new_data)
    return " 흥행 " if prediction[0] == 1 else " 실패 "
```

## 전성배

### 데이터 로드 및 전처리

```
def reco(genre, rating, year):  
    recommendations = df[  
        (df['장르명'].str.contains(genre, na=False)) &  
        (df['등급명'] == rating) &  
        (df['개봉년도'].astype(str).str.startswith(str(year)))  
    ]  
    return recommendations if not recommendations.empty else "조건에 맞는 영화가 없습니다."
```

=== 영화 추천 및 흥행 예측 시스템 ===

#### 2. 관람객 추천

장르 입력: 액션

등급 입력: 15세이상관람가

개봉 연도 입력: 2021

추천 결과:

|   | 영화명  | 장르명 | 등급명      | 개봉년도     |
|---|------|-----|----------|----------|
| 0 | 영화 A | 액션  | 15세이상관람가 | 20210101 |

감사합니다

---