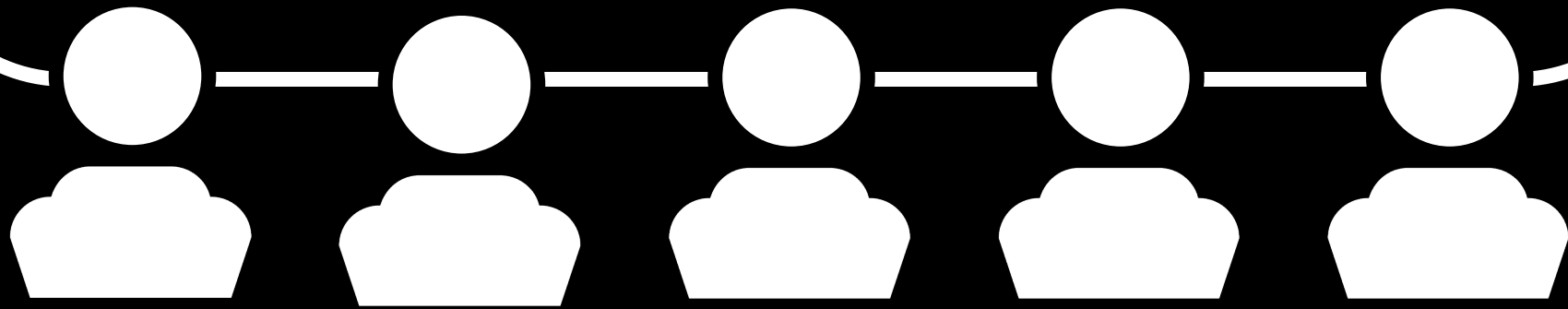
IM in Love with Movies

GenEdu 5010 Final Project

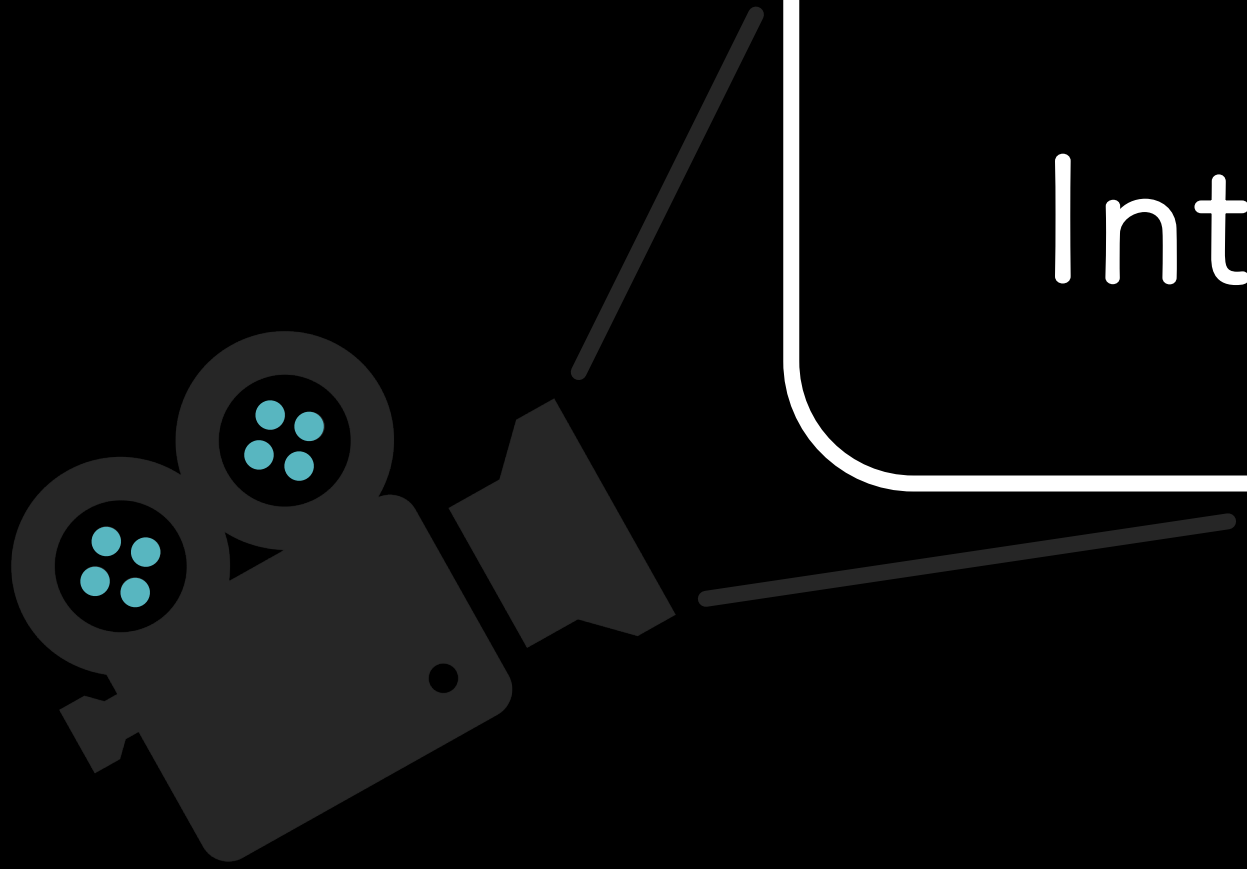
B07705031 陳立軒 資管二 / B08705003 楊佳芊 資管二

B08705027 林暐倫 資管二 / B08705028 葉柏辰 資管二



Agenda

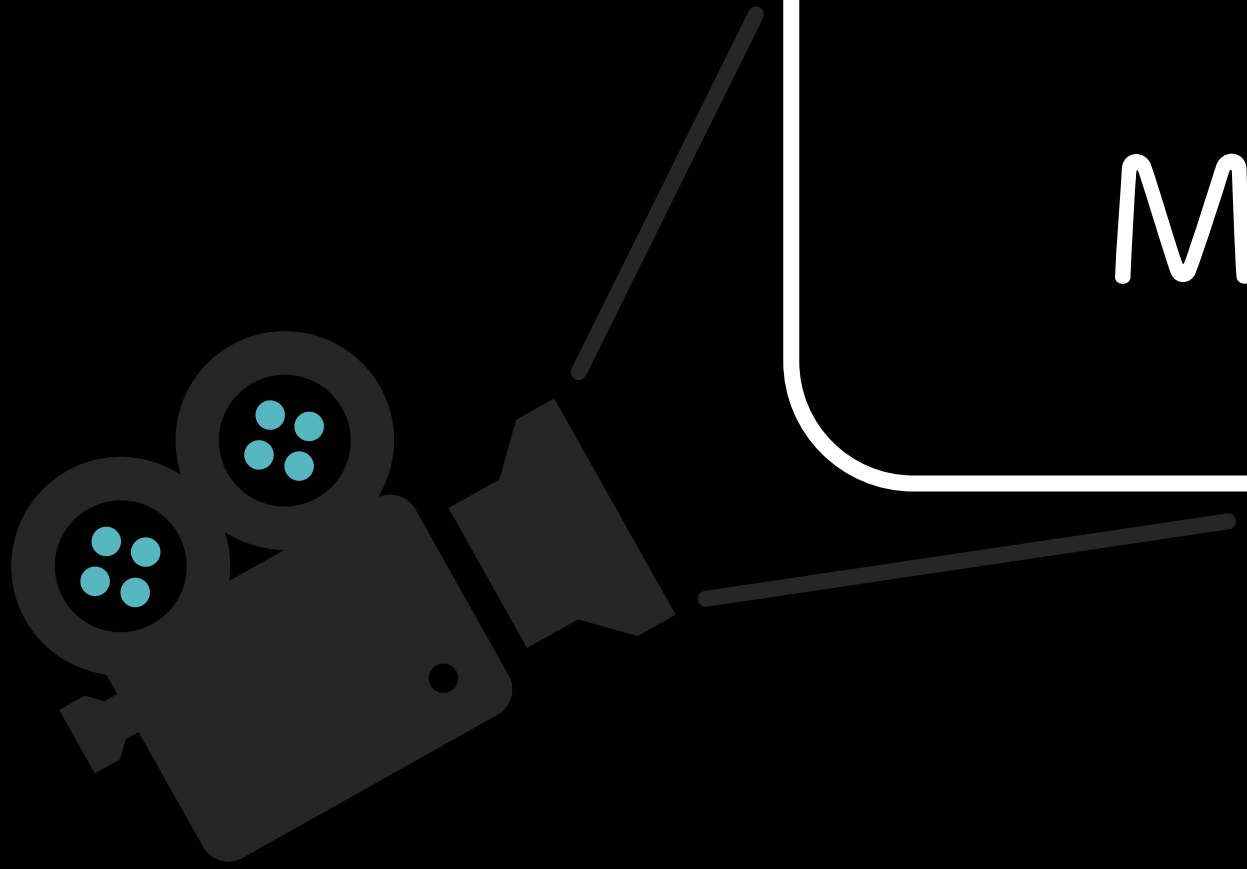
01
Introduction



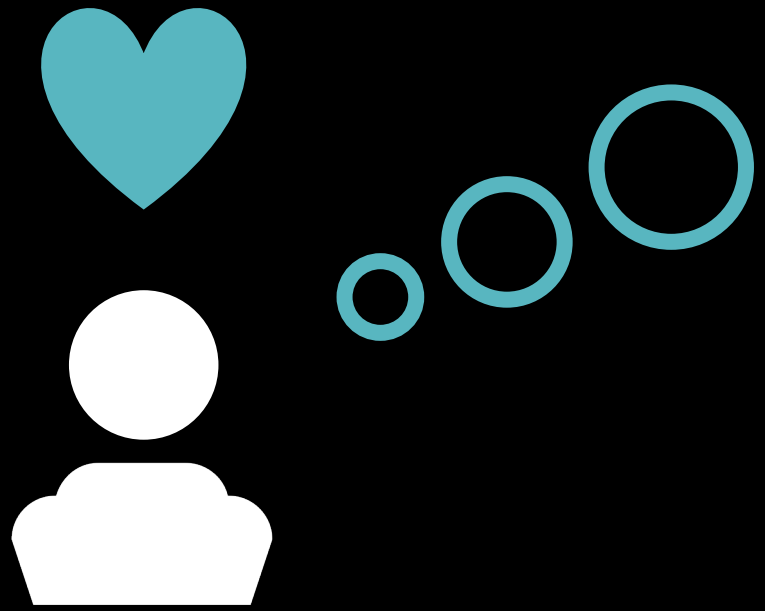
	Step 1	Data Collection	Web Crawler
+)	Step 2	Data Analysis	Statistics

What factors influence the revenue of movies?

02
Motivation



IM in Love with Movies!



Budget

Cast

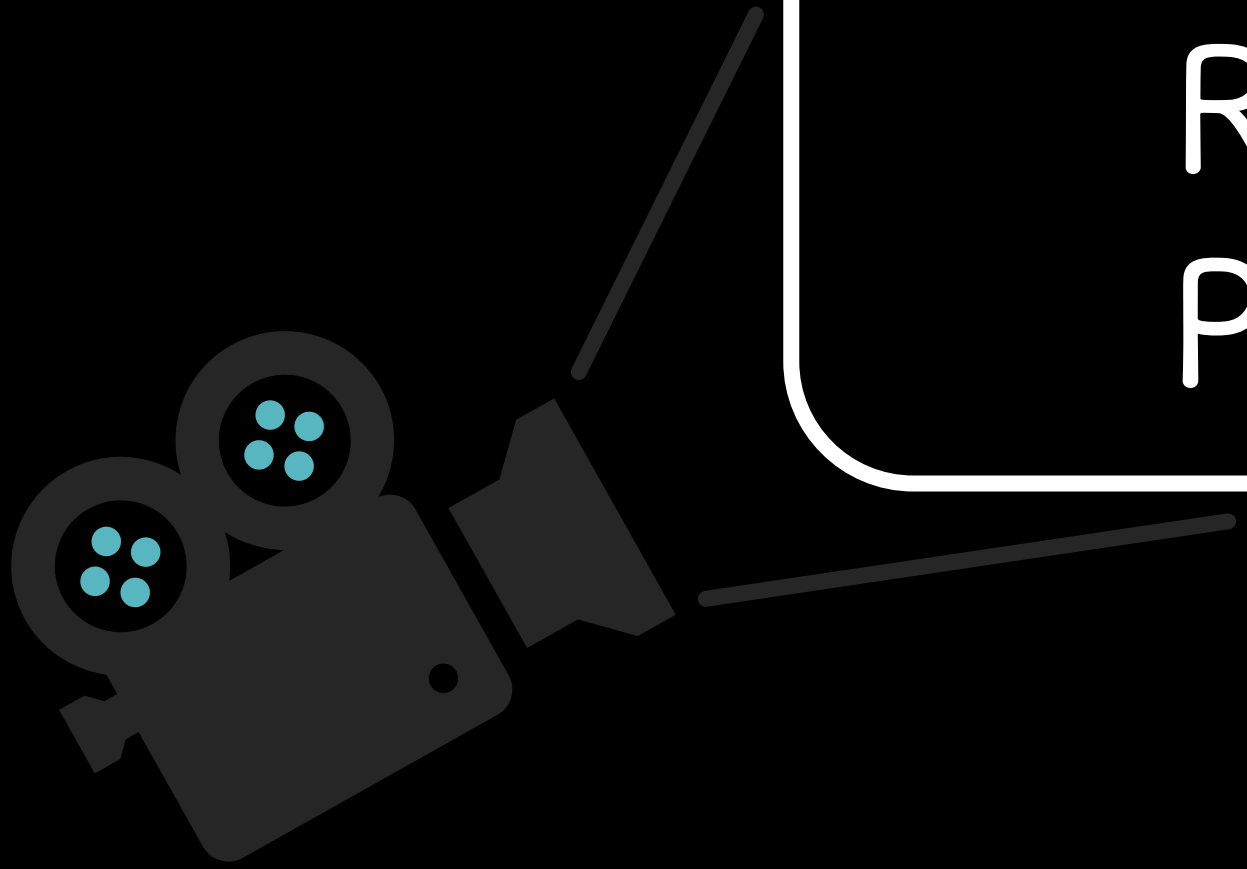
Language

Date

⋮

Revenue

03
Required Packages



For Data Collection

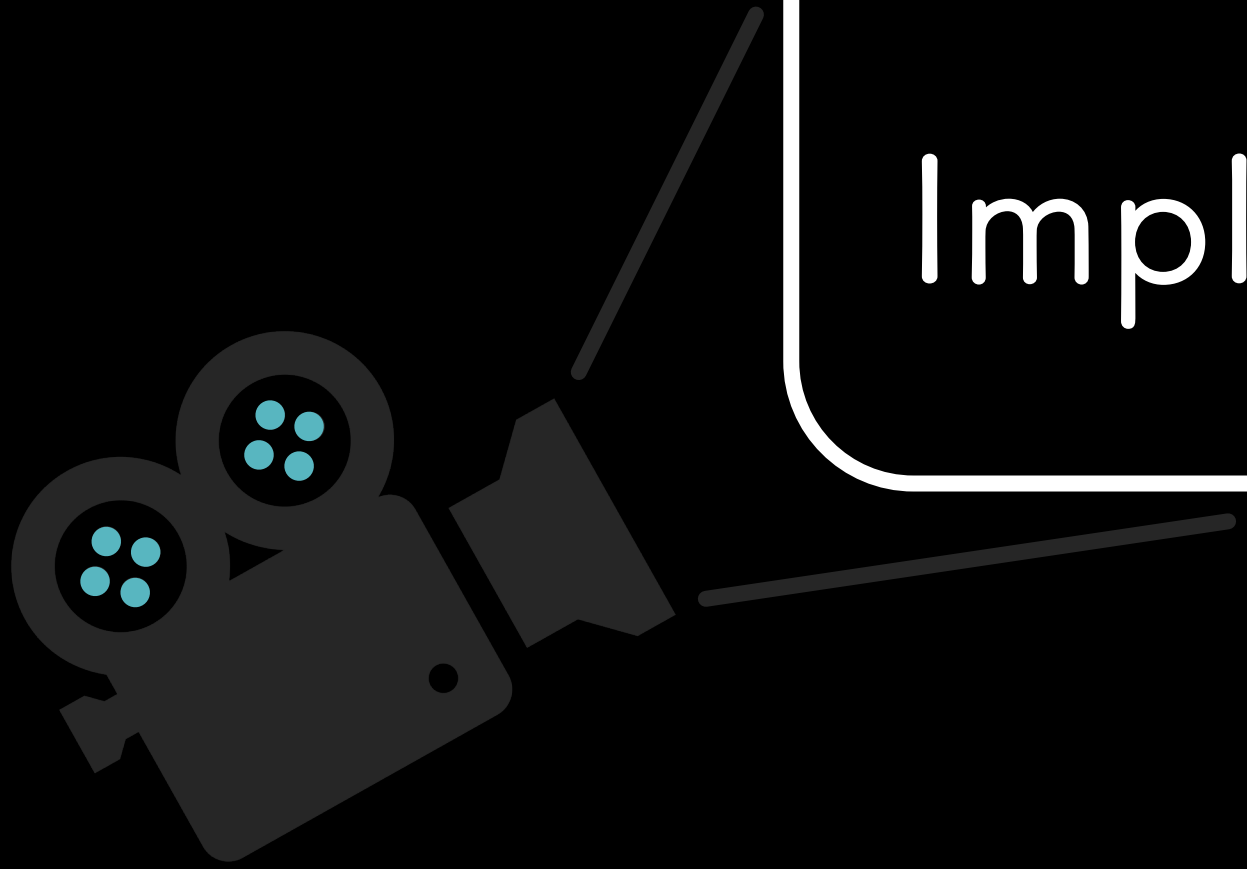
- csv
- datetime
- pandas
- numpy
- requests
- tmdbv3api: The main source of data, using API key to access the data we need.
- BeautifulSoup: Since Rotten Tomatoes' API key has to be requested and gone through a series of application process, we handcrafted to crawler with some help from parm 83.

For Data Analysis

- matplotlib: visualization
- pandas
- numpy
- random
- math
- itertools
- scipy
- statsmodels

} main statistics packages

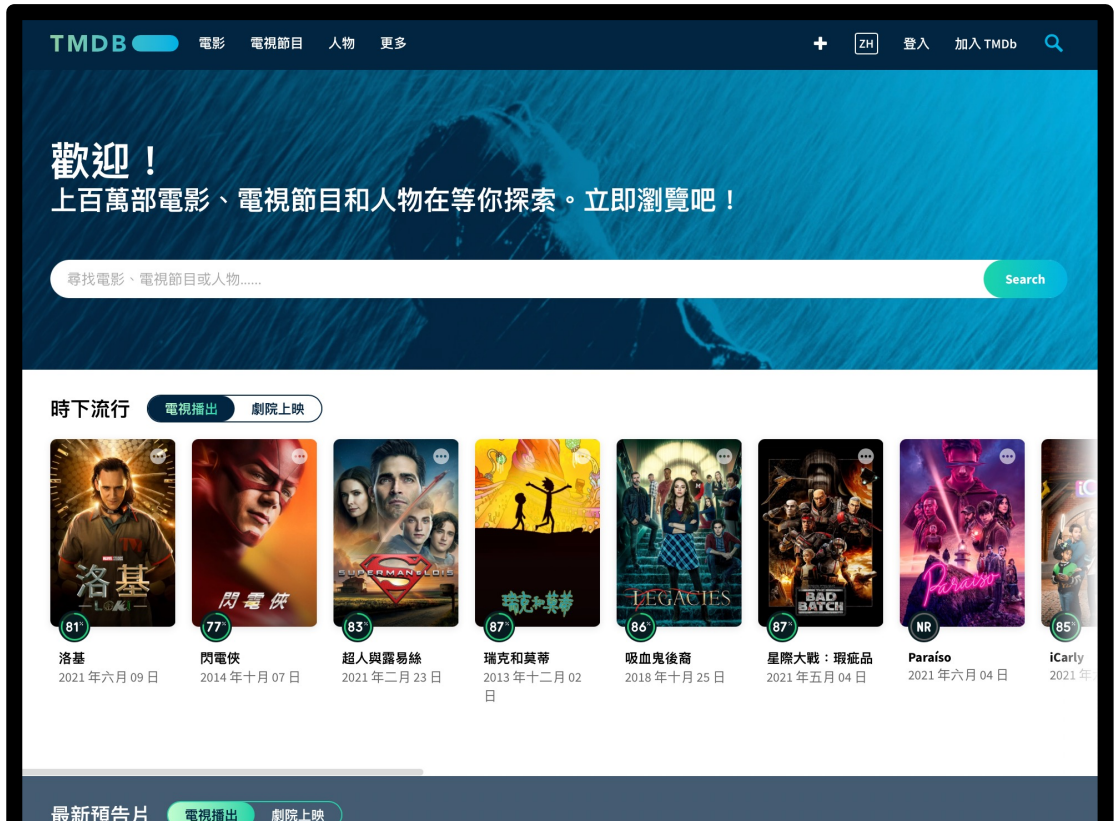
04 Implementation



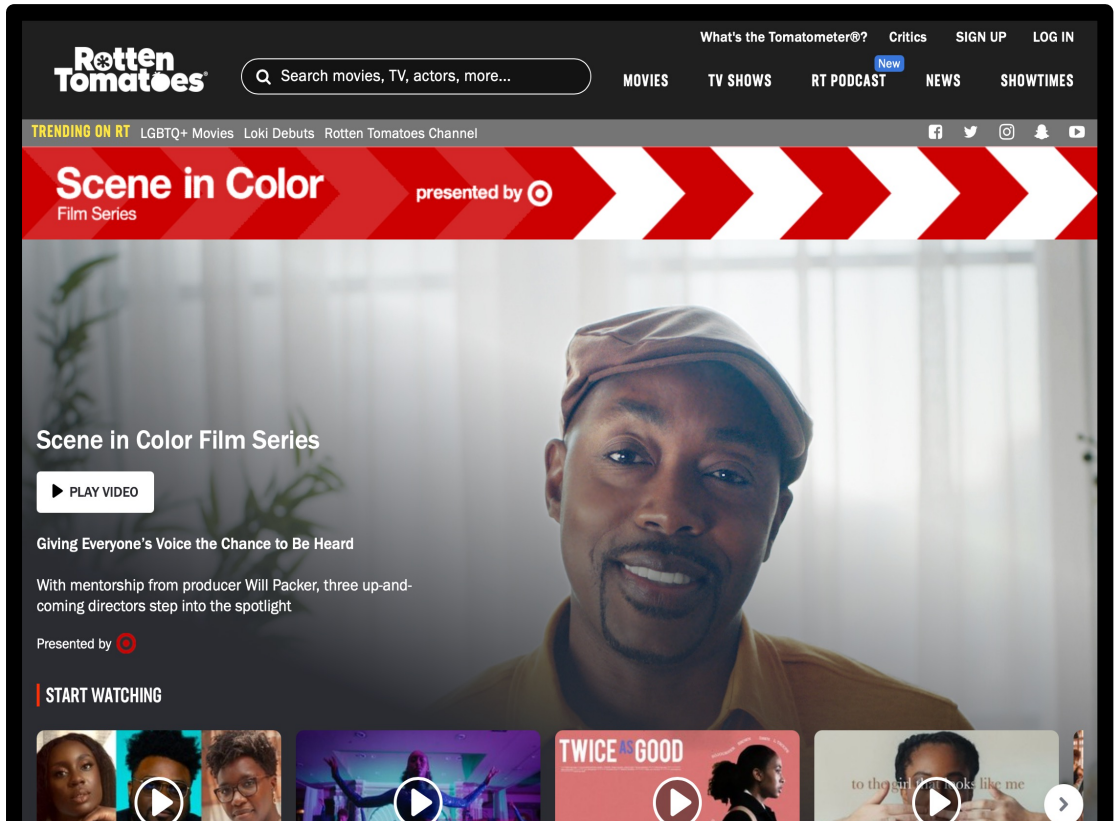
04-1 Data Collection

Data Source

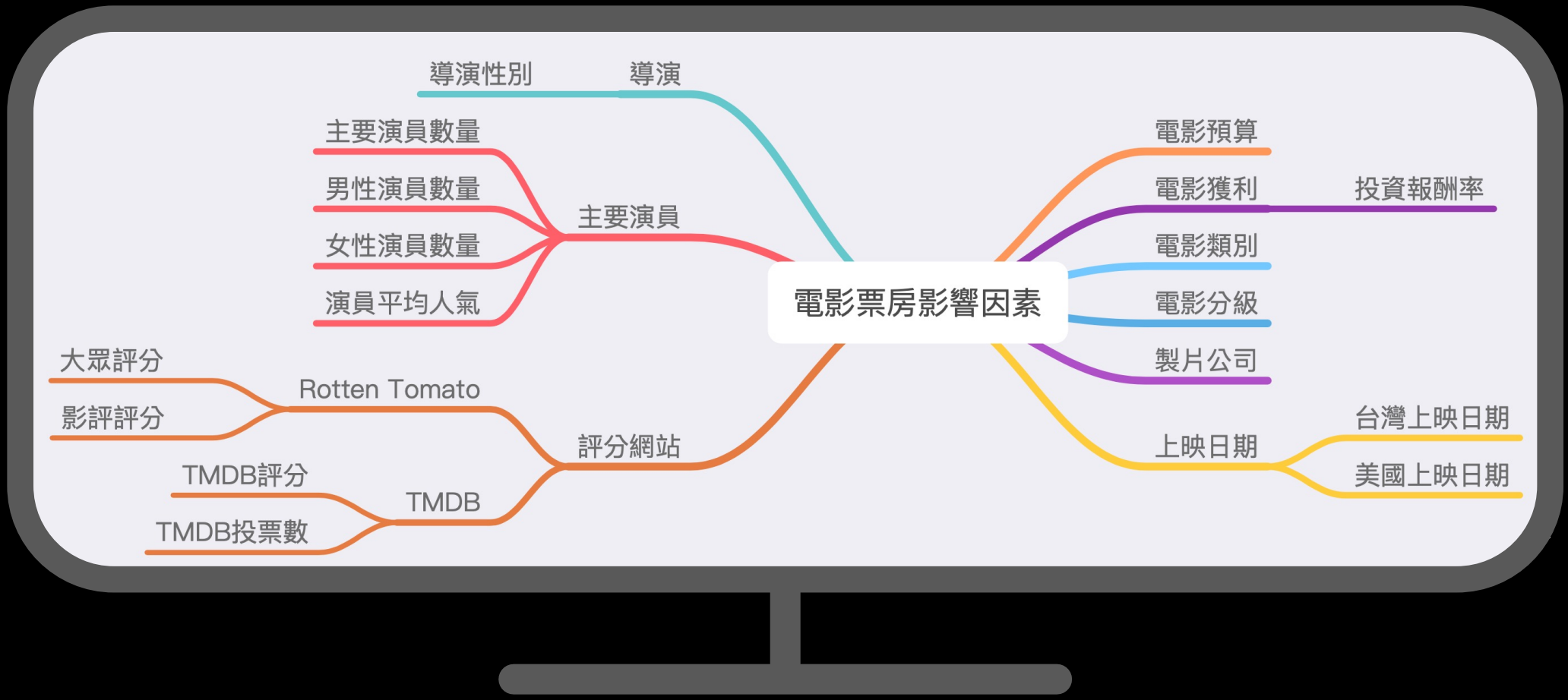
TMDb



Rotten Tomatoes



Mind Map


電影票房影響因素
導演
導演性別
主要演員
主要演員數量
男性演員數量
女性演員數量
演員平均人氣
評分網站
Rotten Tomato
大眾評分
影評評分
TMDB
TMDB評分
TMDB投票數
電影預算
電影獲利
投資報酬率
電影類別
電影分級
製片公司
上映日期
台灣上映日期
美國上映日期

Get ID

```python
from tmdbv3api import TMDb
from tmdbv3api import Movie, Discover
import pandas as pd
import numpy as np
import json
import csv
from tqdm import tqdm  # 進度條
import configparser

config = configparser.ConfigParser()
config.read('.config')
api_key = config['TMDb']['key']

tmdb = TMDb()
tmdb.api_key = api_key
tmdb.language = 'en'
tmdb.debug = True
```

```python
discover = Discover()
vote_bound = 400  # 只抓 400 票觀眾投票以上的
movie = []
for year in tqdm(range(2000, 2022)):
    page = 1
    try:
        while True and page <= 1000:  # 1000 是頁數上限
            new_movie = discover.discover_movies({
                'primary_release_year': year,
                'vote_count.gte': vote_bound,
                'sort_by': 'primary_release_date.desc',
                'page': page
            })
            if len(new_movie) > 0:
                movie += new_movie
                page += 1
            else:
                break
    except:
        # 萬一不小心超出該年的 page 上限
        continue


movie_id = [m.id for m in movie]
len(movie_id)


with open('data/movie_id.csv', 'w', newline='') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
    wr.writerow(movie_id)
```

Get Data

Import packages

```
import csv
from datetime import datetime
from tmdbv3api import TMDb
from tmdbv3api import Movie, Discover
import pandas as pd
import numpy as np
import json
from rt_scraper import MovieScraper
from tqdm import tqdm
import configparser
```

```
config = configparser.ConfigParser()
config.read('.config')
api_key = config['TMDb']['key']

tmdb = TMDb()
tmdb.api_key = api_key
tmdb.language = 'en'
tmdb.debug = True
```

◀Column Names

```
col_names = ['id',
             'title',
             'budget',
             'genres',
             'original_language',
             'production_companies',
             'release_date',
             'TW_release_date',
             'revenue',
             'runtime',
             'cast',
             'cast_cnt',
             'crew_cnt',
             'female_cast_cnt',
             'male_cast_cnt',
             'cast_popularity_ave',
             'director',
             'direcotr_gender',
             'TMDB_score',
             'TMDB_vote_count',
             'profit',
             'ROI',
             'rotten_score',
             'rating',
             'rotten_aud_score',
             'zh_title',
             'belongs_to_collection',
             'has_hompage']
```

（方便大家抓取）

▼Open the previous file

```
with open('data/movie_id.csv', newline='') as f:
    reader = csv.reader(f)
    movie_id = list(reader)[0]
```

Start getting data▶

```
if __name__ == '__main__':

    movie_df = pd.DataFrame(columns=col_names)

    for ID in tqdm(movie_id):
        movie_obj = Movie()
        m = movie_obj.details(ID)
        data_row = dict()
        data_row['id'] = m.id
        data_row['title'] = m.title
        data_row['budget'] = m.budget

        gen = [g['name'] for g in m.genres]
        data_row['genres'] = gen

        data_row['original_language'] = m.original_language

        pcs = [pc['name'] for pc in m.production_companies]
        data_row['production_companies'] = pcs

        data_row['release_date'] = m.release_date

        if m['homepage'] != "":
            data_row['has_homepage'] = 1
        else:
            data_row['has_homepage'] = 0
```

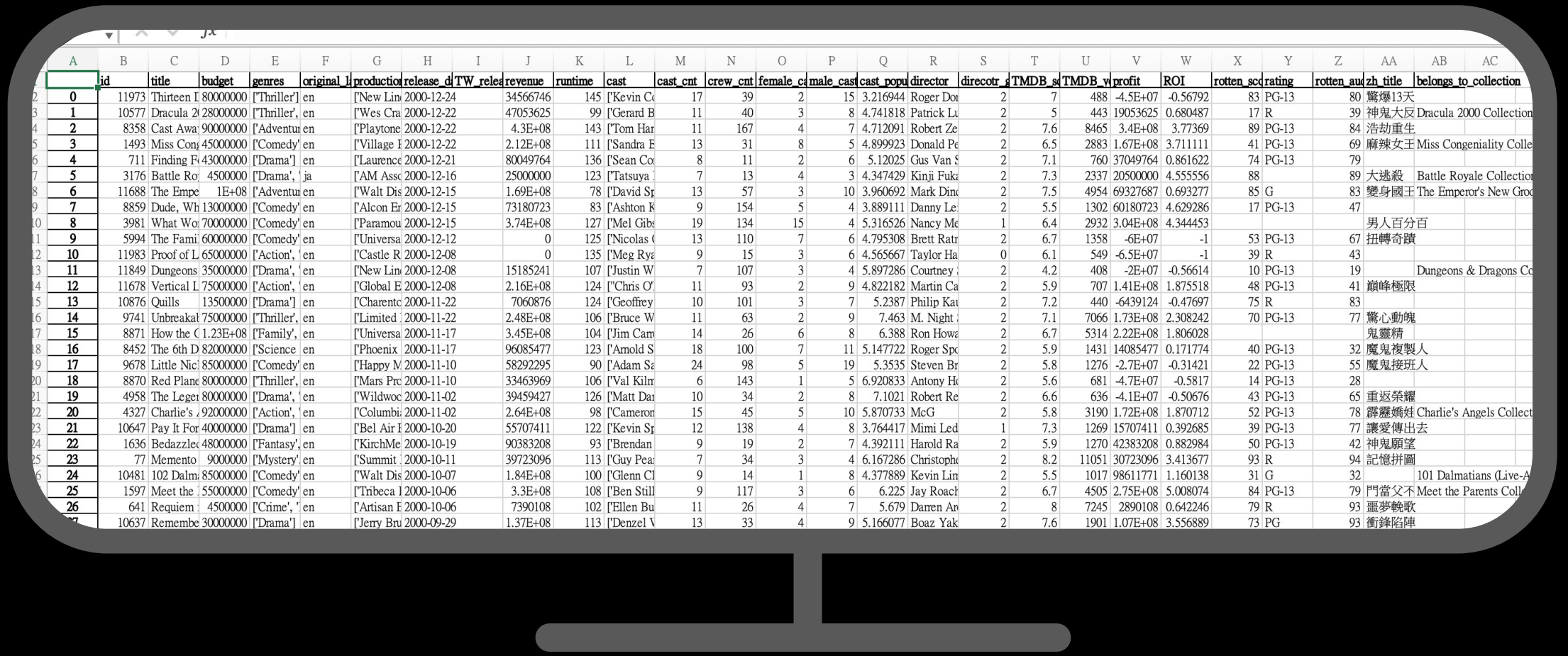
```
        TW_release_date_flag = False
        for loc in m.release_dates['results']:
            for key, value in loc.items():
                if value == 'TW':
                    TW_release_date_flag = True
                    data_row['TW_release_date'] = loc['release_dates'][0]['release_date']
                else:
                    continue
        if not TW_release_date_flag:
            data_row['TW_release_date'] = np.nan

        try:
            belongs_to_collection = m['belongs_to_collection']['name']
            data_row['belongs_to_collection'] = belongs_to_collection
        except:
            data_row['belongs_to_collection'] = np.nan

        data_row['revenue'] = m.revenue
        data_row['runtime'] = m.runtime
```

```
pop_bound = 2
cn = [c['name']
      for c in m.casts['cast'] if c['popularity'] >= pop_bound]
data_row['cast'] = cn
data_row['cast_cnt'], data_row['crew_cnt'] = len([c for c in movie_obj.credits(ID)
    ['cast'] if c['popularity'] >= pop_bound and c['gender'] != 0 and c['gender'] != 3]), len(movie_obj.credits(ID)['crew'])
data_row['female_cast_cnt'] = len(
    [c['gender'] for c in m.casts['cast'] if c['gender'] == 1 and c['popularity'] >= pop_bound])
data_row['male_cast_cnt'] = len(
    [c['gender'] for c in m.casts['cast'] if c['gender'] == 2 and c['popularity'] >= pop_bound])
data_row['cast_popularity_ave'] = np.mean(
    [c['popularity'] for c in m.casts['cast'] if c['popularity'] >= pop_bound])
try:
    data_row['director'] = [c['name']
                            for c in movie_obj.credits(ID)['crew'] if c['job'] == 'Director'][0]
    data_row['direcotr_gender'] = [c['gender']
                                   for c in movie_obj.credits(ID)['crew'] if c['job'] == 'Director'][0]
except:
    data_row['director'] = np.nan
    data_row['direcotr_gender'] = np.nan

data_row['TMDB_score'] = m.vote_average
data_row['TMDB_vote_count'] = m.vote_count

data_row['profit'] = data_row['revenue'] - data_row['budget']
```

- 本想用 MYSQL 來儲存資料，但考量到簡易性，最終輸出成 excel 檔，方便大家之後分析使用。

```
        try:
            data_row['ROI'] = data_row['profit'] / data_row['budget']
        except:
            continue

        try:
            # 爛番茄爬蟲
            year = datetime.strptime(data_row['release_date'], '%Y-%m-%d').year
            movie_scraper = MovieScraper(movie_title=m.title, year=year)
            movie_scraper.extract_metadata()
            data_row['rotten_score'] = float(
                movie_scraper.metadata['Score_Rotten'])
            data_row['rotten_aud_score'] = float(
                movie_scraper.metadata['Score_Audience'])
            data_row['rating'] = movie_scraper.metadata['Rating']
        except:
            data_row['rotten_score'] = np.nan
            data_row['rating'] = np.nan
            data_row['rotten_aud_score'] = np.nan

        # zh_title
        zh_title_flag = False
        for loc in m.translations['translations']:
            for key, value in loc.items():
                if value == 'TW':
                    data_row['zh_title'] = loc['data']['title']
                    zh_title_flag = True
                    break
                else:
                    continue
        if not zh_title_flag:
            data_row['zh_title'] = np.nan

        movie_df = movie_df.append(data_row, ignore_index=True)

    movie_df.to_excel('data/movie_data_set.xlsx')
```

Data File



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
		id	title	budget	genres	original_l	productio	release_d	TW_relea	revenue	runtime	cast	cast_cnt	crew_cnt	female_ca	male_cast	cast_popu	director	directr_g	TMDB_sc	TMDB_v	profit	ROI	rotten_sc	rating	rotten_au	zh_title	belongs_to_collection	
2	0	11973	Thirteen I	80000000	['Thriller']	en	['New Lin	2000-12-24		34566746	145	['Kevin C	17	39	2	15	3.216944	Roger Do	2	7	488	-4.5E+07	-0.56792	83	PG-13	80	驚爆13天		
3	1	10577	Dracula 2	28000000	['Thriller',	en	['Wes Cra	2000-12-22		47053625	99	['Gerard B	11	40	3	8	4.741818	Patrick Lu	2	5	443	19053625	0.680487	17	R	39	神鬼大反	Dracula 2000 Collection	
4	2	8358	Cast Awa	90000000	['Adventu	en	['Playtone	2000-12-22		4.3E+08	143	['Tom Har	11	167	4	7	4.712091	Robert Ze	2	7.6	8465	3.4E+08	3.77369	89	PG-13	84	浩劫重生		
5	3	1493	Miss Con	45000000	['Comedy'	en	['Village I	2000-12-22		2.12E+08	111	['Sandra B	13	31	8	5	4.899923	Donald Pe	2	6.5	2883	1.67E+08	3.711111	41	PG-13	69	麻辣女王	Miss Congeniality Colle	
6	4	711	Finding F	43000000	['Drama']	en	['Laurence	2000-12-21		80049764	136	['Sean Co	8	11	2	6	5.12025	Gus Van S	2	7.1	760	37049764	0.861622	74	PG-13	79			
7	5	3176	Battle Ro	4500000	['Drama', '	ja	['AM Assc	2000-12-16		25000000	123	['Tatsuya	7	13	4	3	4.347429	Kinji Fuka	2	7.3	2337	20500000	4.555556	88		89	大逃殺	Battle Royale Collection	
8	6	11688	The Empe	1E+08	['Adventu	en	['Walt Dis	2000-12-15		1.69E+08	78	['David Sp	13	57	3	10	3.960692	Mark Din	2	7.5	4954	69327687	0.693277	85	G	83	變身國王	The Emperor's New Groc	
9	7	8859	Dude, Wh	13000000	['Comedy'	en	['Alcon Er	2000-12-15		73180723	83	['Ashton K	9	154	5	4	3.889111	Danny Le	2	5.5	1302	60180723	4.629286	17	PG-13	47			
10	8	3981	What Wo	70000000	['Comedy'	en	['Paramou	2000-12-15		3.74E+08	127	['Mel Gibs	19	134	15	4	5.316526	Nancy Me	1	6.4	2932	3.04E+08	4.344453				男人百分百		
11	9	5994	The Fami	60000000	['Comedy'	en	['Universa	2000-12-12		0	125	['Nicolas C	13	110	7	6	4.795308	Brett Ratn	2	6.7	1358	-6E+07	-1	53	PG-13	67	扭轉奇蹟		
12	10	11983	Proof of L	65000000	['Action', '	en	['Castle R	2000-12-08		0	135	['Meg Rya	9	15	3	6	4.565667	Taylor Ha	0	6.1	549	-6.5E+07	-1	39	R	43			
13	11	11849	Dungeons	35000000	['Drama', '	en	['New Lin	2000-12-08		15185241	107	['Justin W	7	107	3	4	5.897286	Courtney	2	4.2	408	-2E+07	-0.56614	10	PG-13	19		Dungeons & Dragons Co	
14	12	11678	Vertical L	75000000	['Action', '	en	['Global E	2000-12-08		2.16E+08	124	["Chris O'	11	93	2	9	4.822182	Martin Ca	2	5.9	707	1.41E+08	1.875518	48	PG-13	41	巔峰極限		
15	13	10876	Quills	13500000	['Drama']	en	['Charento	2000-11-22		7060876	124	['Geoffrey	10	101	3	7	5.2387	Philip Kau	2	7.2	440	-6439124	-0.47697	75	R	83			
16	14	9741	Unbreakal	75000000	['Thriller',	en	['Limited	2000-11-22		2.48E+08	106	['Bruce W	11	63	2	9	7.463	M. Night	2	7.1	7066	1.73E+08	2.308242	70	PG-13	77	驚心動魄		
17	15	8871	How the C	1.23E+08	['Family', '	en	['Universa	2000-11-17		3.45E+08	104	['Jim Carr	14	26	6	8	6.388	Ron Howa	2	6.7	5314	2.22E+08	1.806028				鬼靈精		
18	16	8452	The 6th D	82000000	['Science	en	['Phoenix	2000-11-17		96085477	123	['Arnold S	18	100	7	11	5.147722	Roger Spo	2	5.9	1431	14085477	0.171774	40	PG-13	32	魔鬼複製人		
19	17	9678	Little Nic	85000000	['Comedy'	en	['Happy M	2000-11-10		58292295	90	['Adam Sa	24	98	5	19	5.3535	Steven Br	2	5.8	1276	-2.7E+07	-0.31421	22	PG-13	55	魔鬼接班人		
20	18	8870	Red Plane	80000000	['Thriller',	en	['Mars Prc	2000-11-10		33463969	106	['Val Kilm	6	143	1	5	6.920833	Antony Ho	2	5.6	681	-4.7E+07	-0.5817	14	PG-13	28			
21	19	4958	The Leger	80000000	['Drama', '	en	['Wildwoc	2000-11-02		39459427	126	['Matt Dar	10	34	2	8	7.1021	Robert Re	2	6.6	636	-4.1E+07	-0.50676	43	PG-13	65	重返榮耀		
22	20	4327	Charlie's	92000000	['Action', '	en	['Columbi	2000-11-02		2.64E+08	98	['Cameron	15	45	5	10	5.870733	McG	2	5.8	3190	1.72E+08	1.870712	52	PG-13	78	霹靂嬌娃	Charlie's Angels Collect	
23	21	10647	Pay It For	40000000	['Drama']	en	['Bel Air F	2000-10-20		55707411	122	['Kevin Sp	12	138	4	8	3.764417	Mimi Led	1	7.3	1269	15707411	0.392685	39	PG-13	77	讓愛傳出去		
24	22	1636	Bedazzle	48000000	['Fantasy',	en	['KirchMe	2000-10-19		90383208	93	['Brendan	9	19	2	7	4.392111	Harold Ra	2	5.9	1270	42383208	0.882984	50	PG-13	42	神鬼願望		
25	23	77	Memento	9000000	['Mystery'	en	['Summit	2000-10-11		39723096	113	['Guy Pea	7	34	3	4	6.167286	Christoph	2	8.2	11051	30723096	3.413677	93	R	94	記憶拼圖		
26	24	10481	102 Dalm	85000000	['Comedy'	en	['Walt Dis	2000-10-07		1.84E+08	100	['Glenn Cl	9	14	1	8	4.377889	Kevin Lin	2	5.5	1017	98611771	1.160138	31	G	32		101 Dalmatians (Live-A	
	25	1597	Meet the	55000000	['Comedy'	en	['Tribeca I	2000-10-06		3.3E+08	108	['Ben Still	9	117	3	6	6.225	Jay Roach	2	6.7	4505	2.75E+08	5.008074	84	PG-13	79	門當父不	Meet the Parents Coll	
	26	641	Requiem	4500000	['Crime', '	en	['Artisan E	2000-10-06		7390108	102	['Ellen Bu	11	26	4	7	5.679	Darren Ar	2	8	7245	2890108	0.642246	79	R	93	噩夢輓歌		
		10637	Remembe	30000000	['Drama']	en	['Jerry Bru	2000-09-29		1.37E+08	113	['Denzel V	13	33	4	9	5.166077	Boaz Yak	2	7.6	1901	1.07E+08	3.556889	73	PG	93	衝鋒陷陣		

Columns

Name	Meaning
id	電影編號
title	電影名稱
budget	電影預算
genres	類別 e.g. 喜劇、恐怖……
original_language	原始語言
production_companies	製片公司

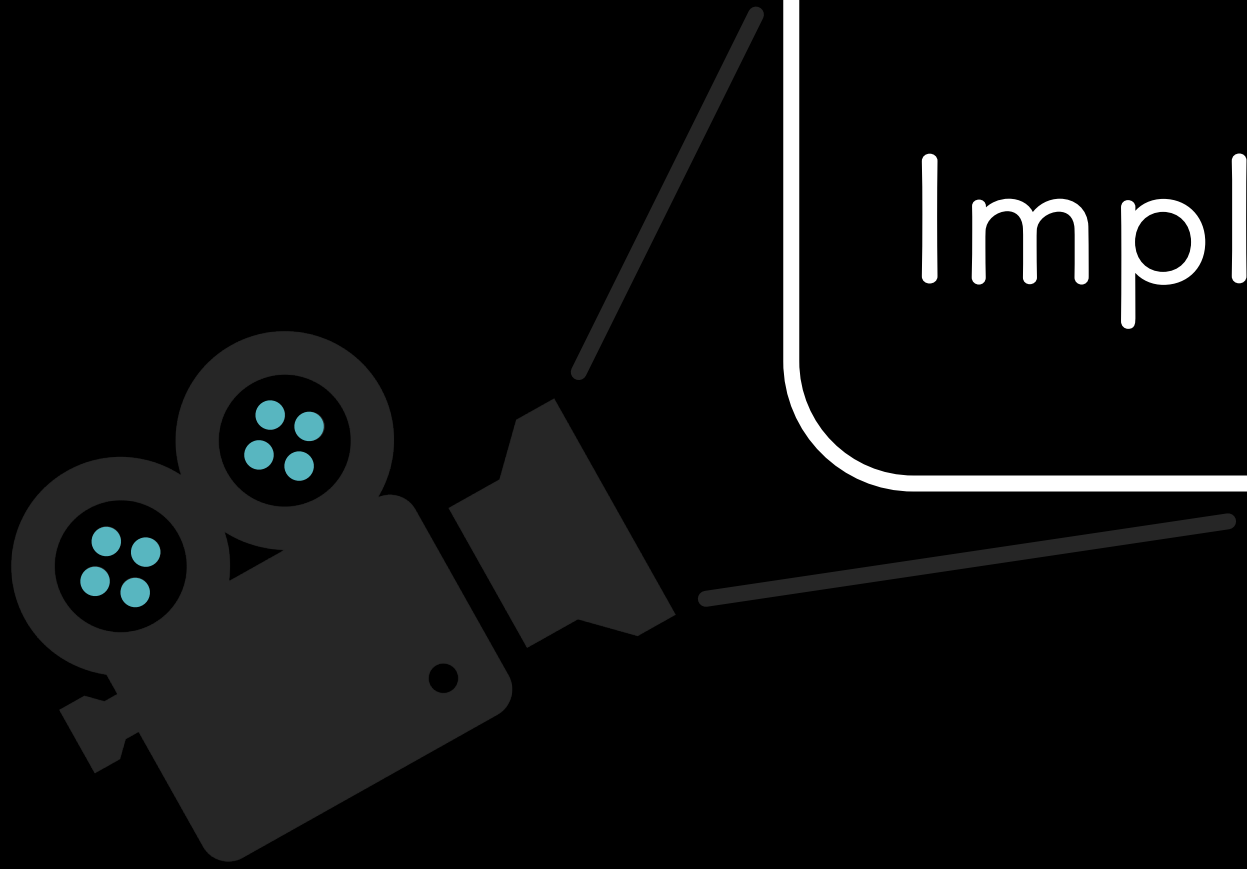
Name	Meaning
release_date	美國上映日期
TW_release_date	臺灣上映日期
revenue	全球票房收入
runtime	電影時長
cast	主要演員
cast_cnt	主要演員數
crew_cnt	工作人員數

Columns

Name	Meaning
female_cast_cnt	女性演員數
male_cast_cnt	男性演員數
cast_popularity_ave	演員平均人氣
director	導演
direcotr_gender	導演性別
TMDB_score	TMDB 評分
TMDB_vote_count	TMDB 投票數

Name	Meaning
profit	電影獲利
ROI	ROI
rotten_score	影評評分
rating	電影分級
rotten_aud_score	觀眾評分
zh_title	電影中文名稱
belongs_to_collection	屬於什麼系列 e.g. 哈利波特
has_homepage	是否有官方網頁

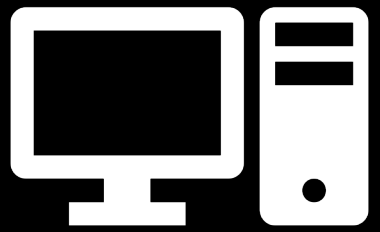
04 Implementation

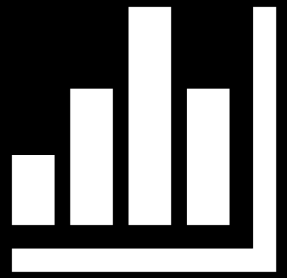


04-2 Data Analysis

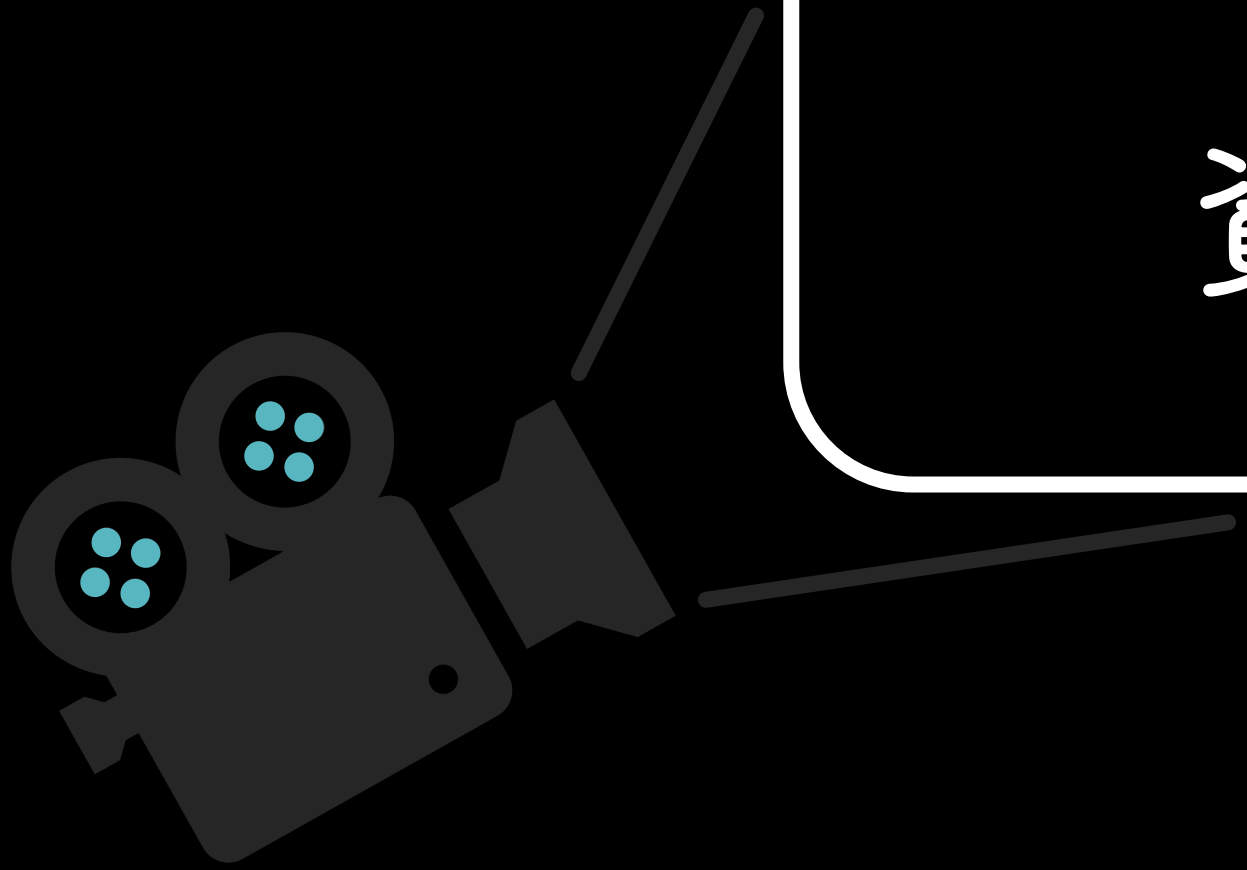
Steps

I. 資料處理 → II. 主題分析 → III. 票房預測



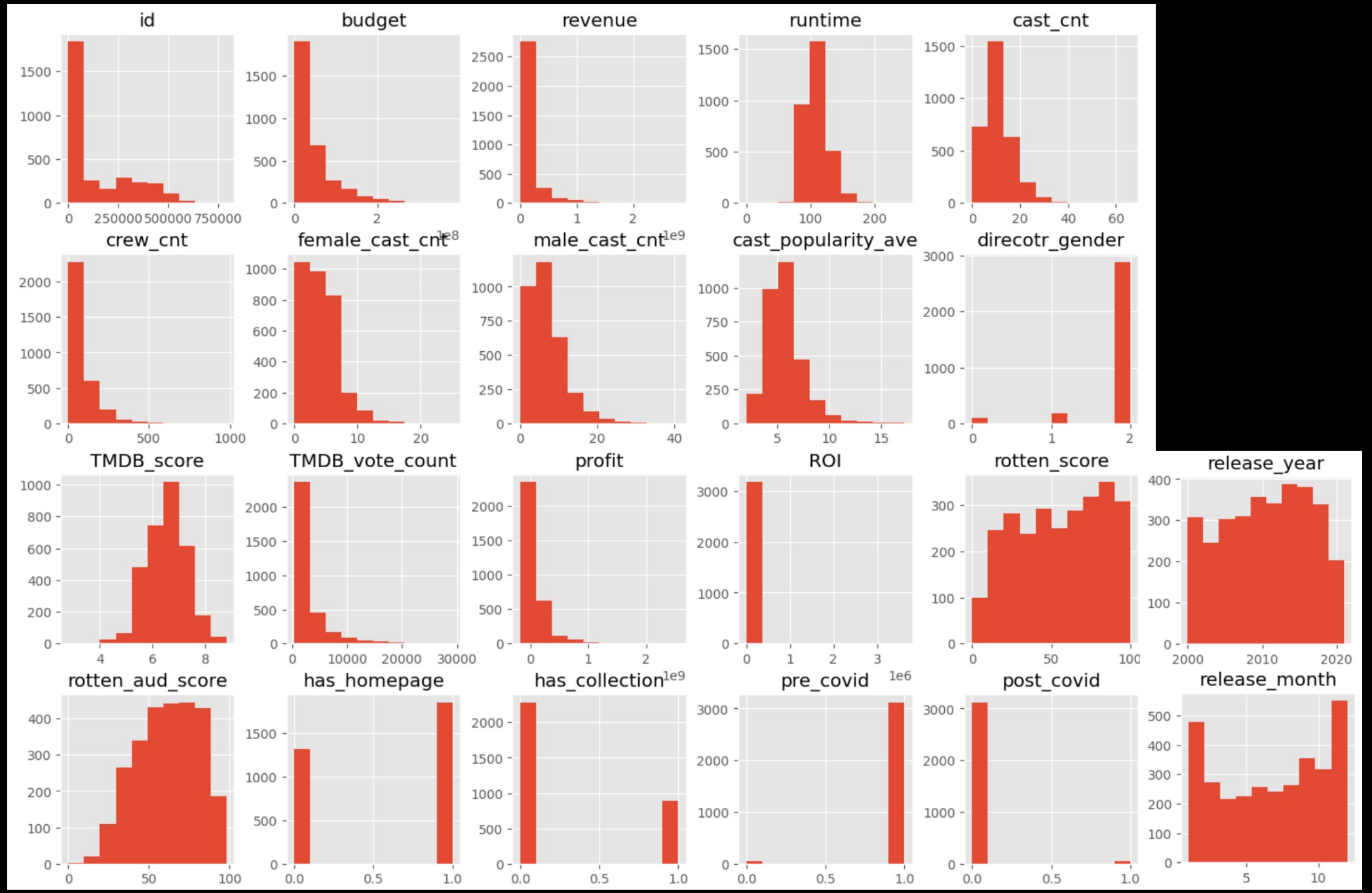


1.
資料處理

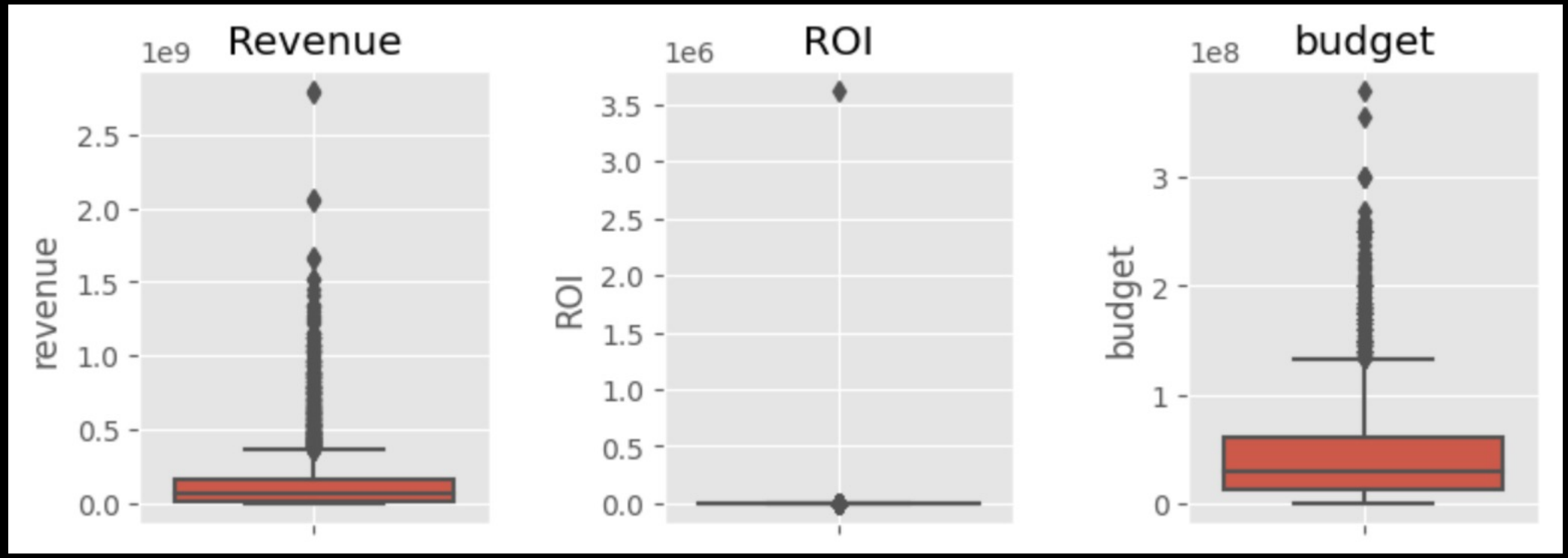


Data Overview


```
movie_df = pd.read_excel('../data/sorted_all_movie.xlsx', index_col=0)
covid_date = datetime(2020, 3, 1)
movie_df['has_collection'] = movie_df['belongs_to_collection'].isna().replace({True: 0, False: 1})
movie_df['pre_covid'] = (movie_df['release_date'] < covid_date).replace({True: 1, False: 0})
movie_df['post_covid'] = (movie_df['release_date'] >= covid_date).replace({True: 1, False: 0})
movie_df['release_year'] = movie_df['release_date'].apply(lambda x: x.year)
movie_df['release_month'] = movie_df['release_date'].apply(lambda x: x.month)

#display(movie_df.head())
#display(movie_df.tail())

movie_df.hist(figsize=(15, 15))
plt.show()
```


id
budget
revenue
runtime
cast_cnt
crew_cnt
female_cast_cnt
1e8
male_cast_cnt
1e9
cast_popularity_ave
direcotr_gender
TMDB_score
TMDB_vote_count
profit
ROI
rotten_score
release_year
rotten_aud_score
has_homepage
has_collection
1e9
pre_covid
1e6
post_covid
release_month

```python
# Plot
fig = plt.figure(figsize=(9, 3))
row, col = 1, 3
fig.subplots_adjust(hspace=0.2, wspace=.5)
ax = fig.add_subplot(row, col, 1)
ax = sns.boxplot(y=movie_df['revenue']) # orient='h' results in horizontal boxplot
plt.title('Revenue')
ax.grid(True)
ax = fig.add_subplot(row, col, 2)
ax = sns.boxplot(y=movie_df['ROI']) # orient='h' results in horizontal boxplot
plt.title('ROI')
ax.grid(True)

ax = fig.add_subplot(row, col, 3)
ax = sns.boxplot(y=movie_df['budget']) # orient='h' results in horizontal boxplot
plt.title('budget')
ax.grid(True)


plt.show()
```



- 有需多outliers
- 不是常態分佈
  →用`np.log1p()`做data transformation
  →做預測的話用`np.exp1p()`還原

Movie with the Highest Revenue

```
movie_df.sort_values(by='revenue', ascending=False).head(20)[['release_date', 'zh_title', 'revenue']]
```

票房最高的電影是
2019 年上映的
《復仇者聯盟：終局之戰》

	release_date	zh_title	revenue
3005	2019-04-24	復仇者聯盟：終局之戰	2797800564
1339	2009-12-10	阿凡達	2787965087
2418	2015-12-15	STAR WARS：原力覺醒	2068223624
2850	2018-04-25	復仇者聯盟3：無限之戰	2046239637
2315	2015-06-06	侏羅紀世界	1671713208
3032	2019-07-12	獅子王	1656943394
1756	2012-04-25	復仇者聯盟	1518815515
2287	2015-04-01	玩命關頭7	1515047671
3086	2019-11-20	冰雪奇緣2	1450026933
2298	2015-04-22	復仇者聯盟2：奧創紀元	1405403694
2822	2018-02-13	黑豹	1346739107
1600	2011-07-07	哈利波特：死神的聖物 II	1341511219
2795	2017-12-13	STAR WARS：最後的絕地武士	1332539889
2866	2018-06-06	侏羅紀世界：殞落國度	1303459585
2035	2013-11-20	冰雪奇緣	1274219009
2669	2017-03-16	美女與野獸	1263521126
2871	2018-06-14	超人特攻隊2	1242805359
2681	2017-04-12	玩命關頭8	1238764765
1915	2013-04-18	鋼鐵人 3	1214811252
2318	2015-06-17	小小兵	1156730962

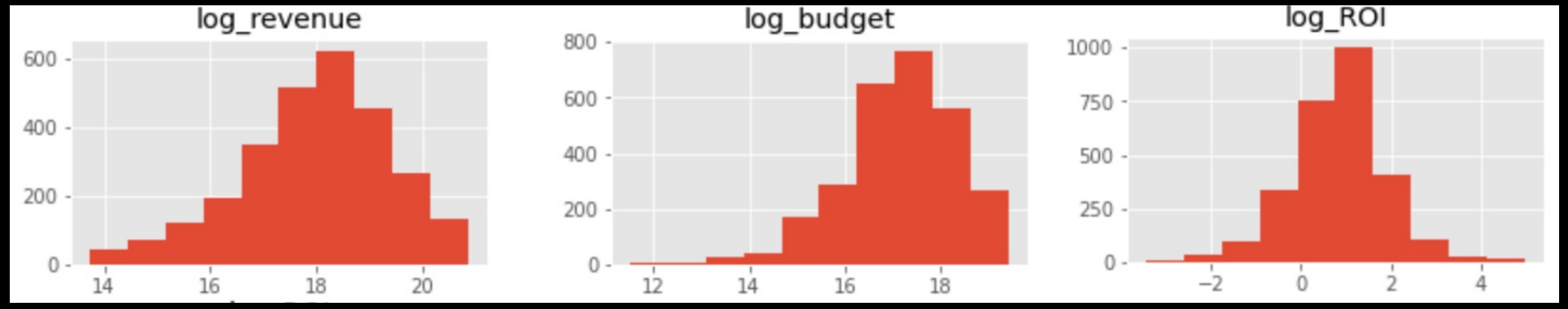
Remove Outliers

- 若去除掉所有 outliers：No.1 博物館驚魂夜3 No.2 玩命關頭4。
- 前十名幾乎都是一些老牌電影（2012前上映）。
- 對新電影的公平性→只刪去前20名票房的電影。

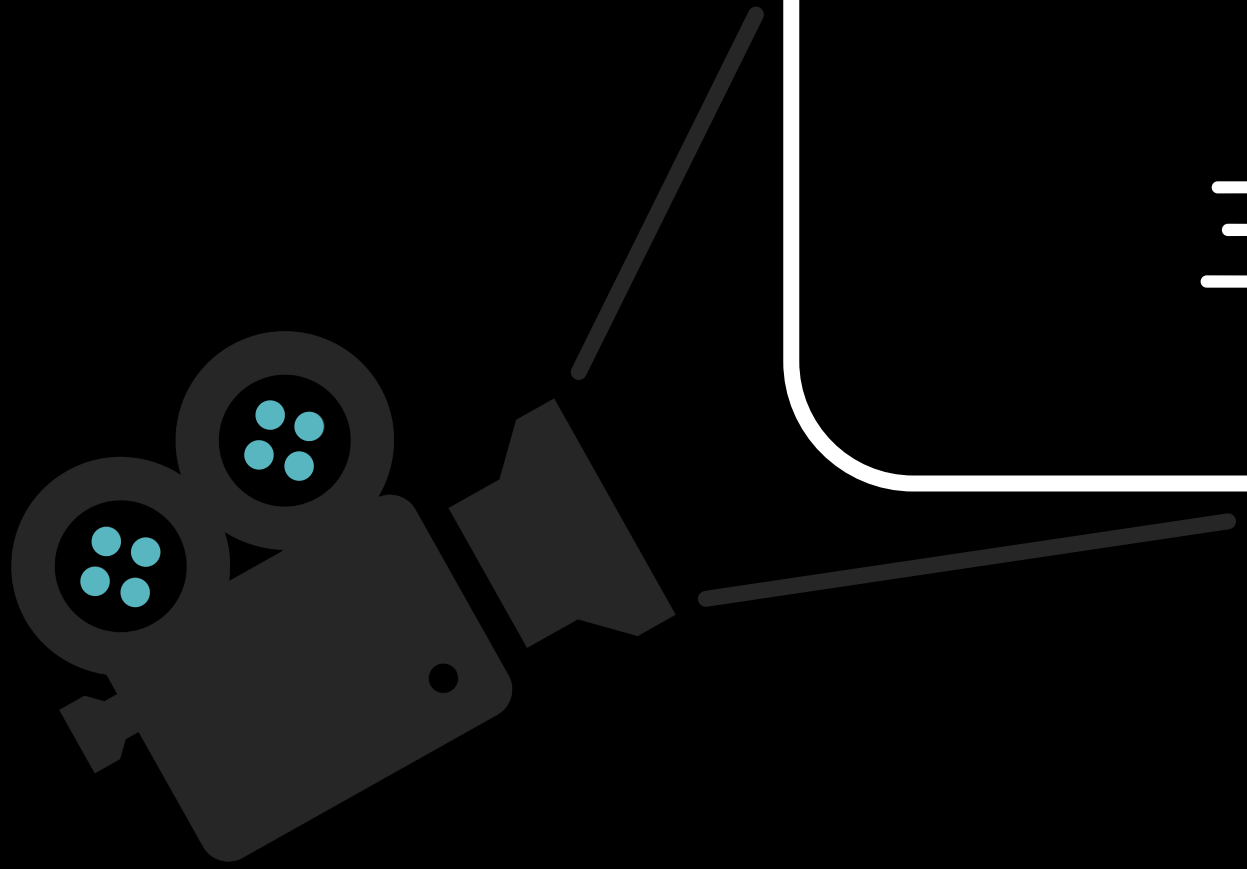
```
rev_outlier = sorted(mgt2001.des.outlier(movie_df['revenue'].dropna(), show=False)[0], reverse=True)
[:20]
roi_outlier = sorted(mgt2001.des.outlier(movie_df['ROI'].dropna(), show=False)[0], reverse=True)[:5]
budget_outlier = sorted(mgt2001.des.outlier(movie_df['budget'].dropna(), show=False)[0], reverse=True)
[:5]

def filter_rows_by_values(df, col, values):
    return df[df[col].isin(values) == False]

rev_df = movie_df[movie_df['revenue'] >= 1e5 * 9] # 排除 90 萬以下票房的資料點（這些點沒有被移除）
rev_df = filter_rows_by_values(rev_df, 'revenue', rev_outlier).reset_index(drop=True)
roi_df = filter_rows_by_values(movie_df, 'ROI', roi_outlier).reset_index(drop=True)
budget_df = filter_rows_by_values(movie_df, 'budget', budget_outlier).reset_index(drop=True)

u_movie_df = movie_df[movie_df['revenue'] >= 1e5 * 9] # 排除 90 萬以下票房的資料點（這些點沒有被移除）
u_movie_df = filter_rows_by_values(u_movie_df, 'revenue', rev_outlier).reset_index(drop=True)
u_movie_df = filter_rows_by_values(u_movie_df, 'ROI', roi_outlier).reset_index(drop=True)
u_movie_df = filter_rows_by_values(u_movie_df, 'budget', budget_outlier).reset_index(drop=True)
```

Data Transformation


```
u_movie_df['log_revenue'] = np.log1p(u_movie_df['revenue'])
u_movie_df['log_budget'] = np.log1p(u_movie_df['budget'])
u_movie_df['log_ROI'] = np.log1p(u_movie_df['ROI'])

# 年份以及月份
u_movie_df['release_month'] = u_movie_df['release_date'].apply(lambda x: x.month)
u_movie_df['release_year'] = u_movie_df['release_date'].apply(lambda x: x.year)

u_movie_df[['log_revenue', 'log_budget', 'log_ROI']].hist(figsize = (9,5))
plt.show()
```



II.
主題分析



Topics

觀眾喜好分析

1. 觀眾評分與電影類別的關係
2. 寒暑假比較多電影？
3. 過年大片是否真的比較夯？
4. 夏天的電影比較熱門？

電影票房分析

1. 在疫情前後的票房差異
2. 票房與投入成本的關係
3. 票房與男女演員人數的關係
4. 評分差異
5. 票房與觀眾評分的關係
6. 票房與原始語言的關係
7. 票房與電影級數的關係
8. 票房與上映時間的關係
9. 票房與 ROI 的關係

Topics

電影票房

1. 在疫情前後的
2. 票房與投入成
3. 票房與男女演
4. 評分差異
5. 票房與觀眾評
6. 票房與原始語
7. 票房與電影級
8. 票房與上映時間的關係

電影投資報酬率分析

ROI的關係

I的變數

喜好分析

與電影類別的關係

較多電影？

是否真的比較夯？

影比較熱門？

過年大片真的比較夯？ Wilcoxon Rank Sum Test

聖誕假期（12/24~1/3） vs. 剩餘年度▼

```
n1 = y_xmas.shape[0]
n2 = n_xmas.shape[0]

if np.sum(np.array([n1, n2]) > 10) == 2:
    print('Both datasets have sizes larger than 10.\n')
    updated_df, result_dict = non.ranksum_z_test(df=xmas_df, to_compute='no', alternative='less')
```

```
Both datasets have sizes larger than 10.

======= z-test =======
T (sum of ranks) = 3447837.0
(n1, n2) = (2532, 225)
mu_t = 3491628.0
sigma_t = 11442.729132510303
z statistic value (observed) = -3.8270
p-value = 0.0001 (Overwhelming Evidence)
Reject H_0 (less) → True
```

YES!!

夏天的電影比較熱門？ Wilcoxon Rank Sum Test

夏天 vs. 冬天▶

```
n1 = short_df.loc[:, 2].dropna().shape[0]
n2 = short_df.loc[:, 4].dropna().shape[0]

if np.sum(np.array([n1, n2]) > 10) == 2:
    print('Both datasets have sizes larger than 10.\n')
    updated_df, result_dict = non.ranksum_z_test(df=short_df[[2, 4]], to_compute=2,
alternative='greater')
```

夏天 vs. 秋天▶

```
n1 = short_df.loc[:, 2].dropna().shape[0]
n2 = short_df.loc[:, 3].dropna().shape[0]

if np.sum(np.array([n1, n2]) > 10) == 2:
    print('Both datasets have sizes larger than 10.\n')
    updated_df, result_dict = non.ranksum_z_test(df=short_df[[2, 3]], to_compute=2,
alternative='greater')
```

夏天 vs. 春天▶

```
n1 = short_df.loc[:, 2].dropna().shape[0]
n2 = short_df.loc[:, 1].dropna().shape[0]

if np.sum(np.array([n1, n2]) > 10) == 2:
    print('Both datasets have sizes larger than 10.\n')
    updated_df, result_dict = non.ranksum_z_test(df=short_df[[2, 1]], to_compute=2,
alternative='greater')
```

夏天的電影比較熱門？

夏天 vs. 冬天▼

```
Both datasets have sizes larger than 10.

======= z-test =======
T (sum of ranks) = 467979
(n1, n2) = (668, 674)
mu_t = 448562.0
sigma_t = 7098.483265224105
z statistic value (observed) = 2.7354
p-value = 0.0031 (Overwhelming Evidence)
Reject H_0 (greater) → True
```

夏天 vs. 秋天▼

```
Both datasets have sizes larger than 10.

======= z-test =======
T (sum of ranks) = 537890.0
(n1, n2) = (668, 792)
mu_t = 487974.0
sigma_t = 8025.744077654109
z statistic value (observed) = 6.2195
p-value = 0.0000 (Overwhelming Evidence)
Reject H_0 (greater) → True
```

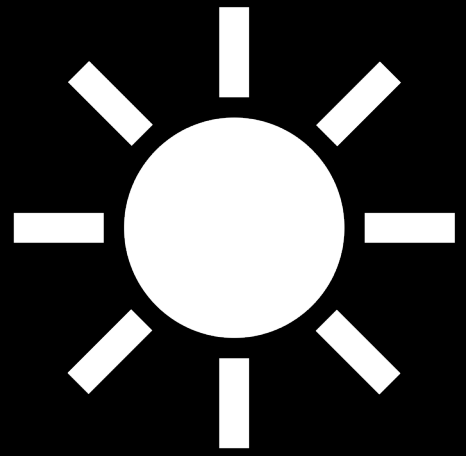
夏天 vs. 春天▼

```
Both datasets have sizes larger than 10.

======= z-test =======
T (sum of ranks) = 449577
(n1, n2) = (668, 623)
mu_t = 431528.0
sigma_t = 6693.802407202252
z statistic value (observed) = 2.6964
p-value = 0.0035 (Overwhelming Evidence)
Reject H_0 (greater) → True
```

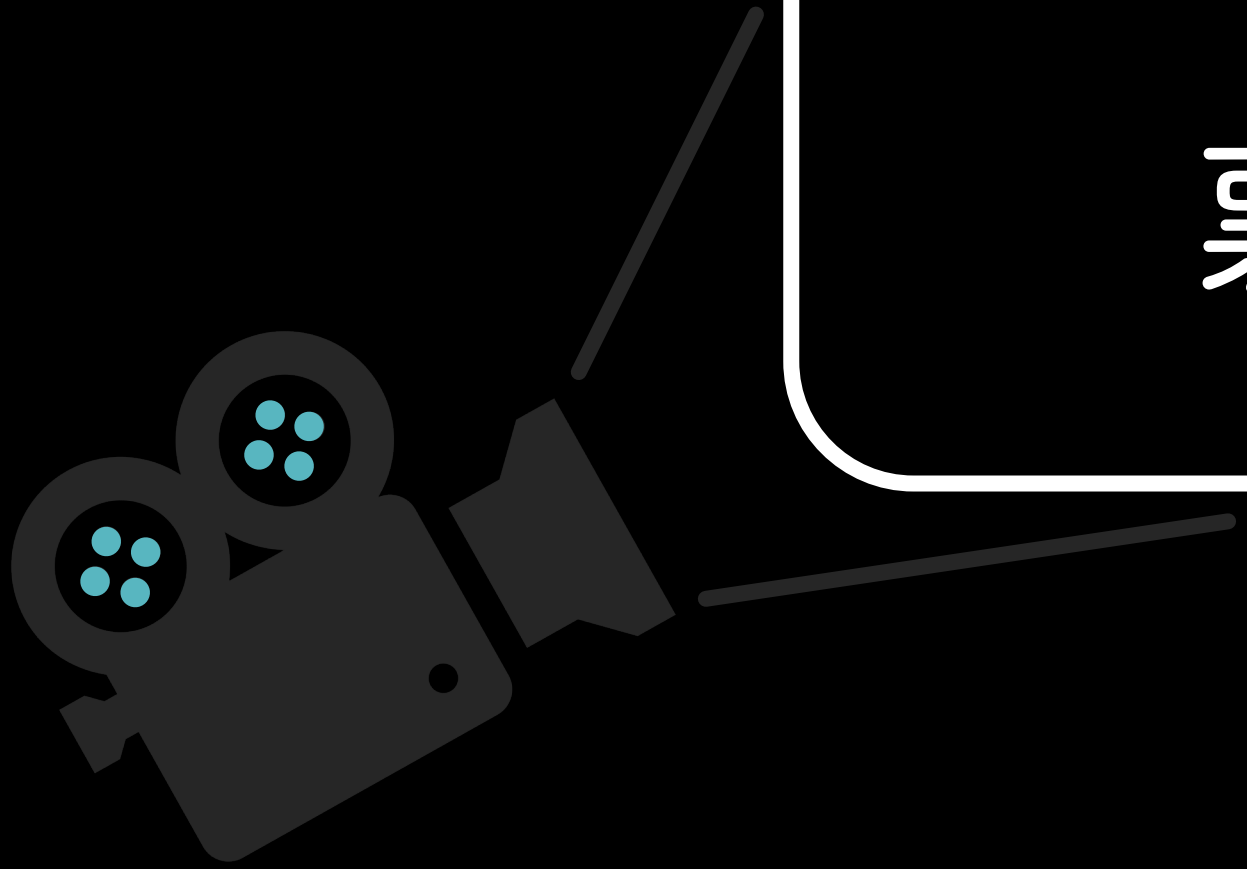
夏天的電影票房確實比其他季節來得高！

Interesting Findings

vs.



III.
票房預測



Multiple Regression

```
res_dict, assessment = mgt2001.model.MultipleRegression(x_names=new_x_names,
                                                         y_name=y_name,
                                                         df=regression_df,
                                                         assessment=False,
                                                         t_test_c=0,
                                                         t_test_option='two-tail')
```

Multiple Regression

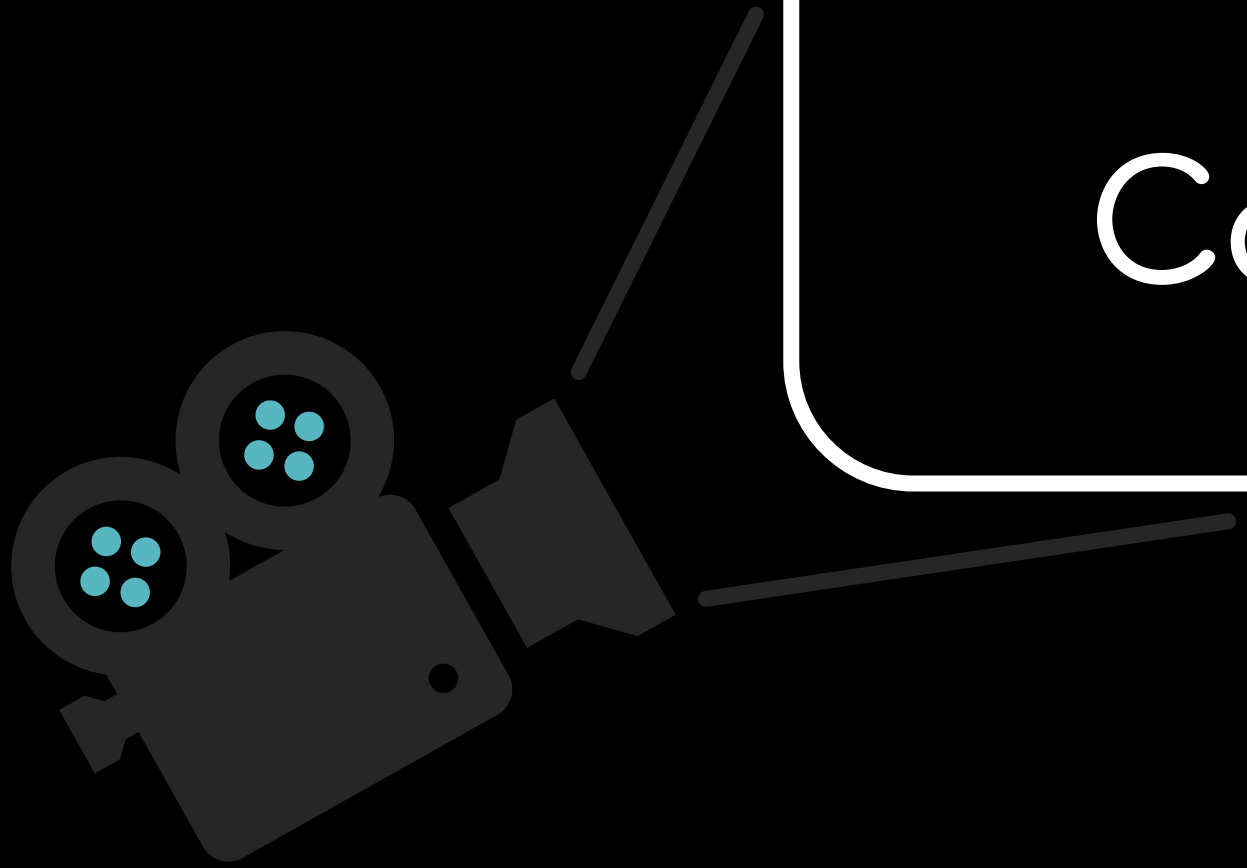
OLS Regression Results

Dep. Variable:	revenue	R-squared:	0.761
Model:	OLS	Adj. R-squared:	0.760
Method:	Least Squares	F-statistic:	981.7
Date:	Sun, 20 Jun 2021	Prob (F-statistic):	0.00
Time:	15:51:42	Log-Likelihood:	-30770.
No. Observations:	1550	AIC:	6.155e+04
Df Residuals:	1544	BIC:	6.158e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-9.306e+07	1.07e+07	-8.730	0.000	-1.14e+08	-7.21e+07
budget	1.7223	0.065	26.509	0.000	1.595	1.850
TMDB_vote_count	2.444e+04	983.570	24.850	0.000	2.25e+04	2.64e+04
has_collection	8.655e+07	6.2e+06	13.969	0.000	7.44e+07	9.87e+07
rotten_aud_score	6.553e+05	1.63e+05	4.014	0.000	3.35e+05	9.75e+05
cast_cnt	1.441e+06	4.85e+05	2.969	0.003	4.89e+05	2.39e+06

Omnibus:	572.065	Durbin-Watson:	1.960
Prob(Omnibus):	0.000	Jarque-Bera (JB):	5493.980
Skew:	1.444	Prob(JB):	0.00
Kurtosis:	11.760	Cond. No.	2.95e+08

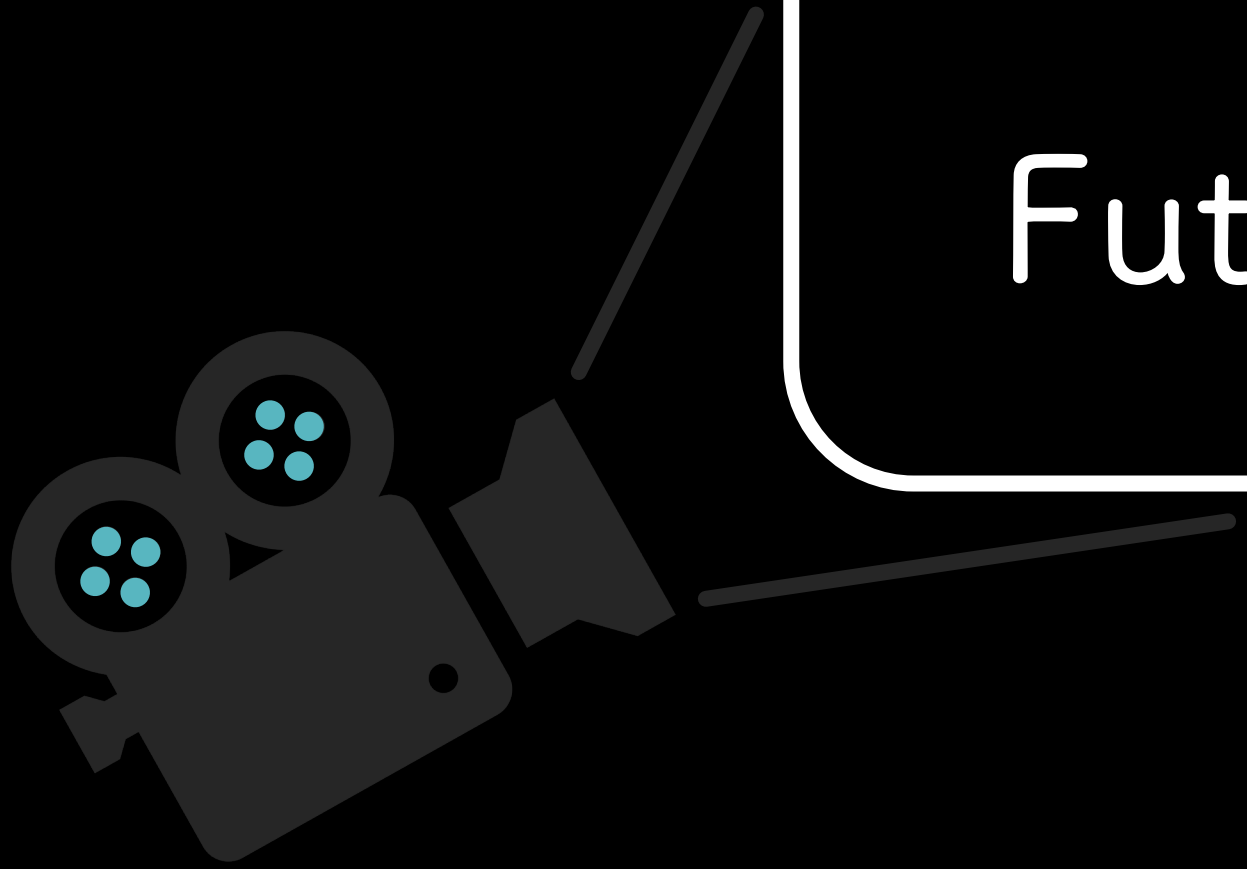
05
Conclusion



	−93056430.8499	
	1.7223	budget
	24441.7030	TMDB_vote_count
	86546747.0422	has_collection
	655289.0341	rotten_aud_score
+)	1441269.2973	cast_cnt

Average Revenue

06
Future Works



- 更多分析面向
- ploty 的互動式網頁
- 電影資訊整合系統
- 電影推薦系統