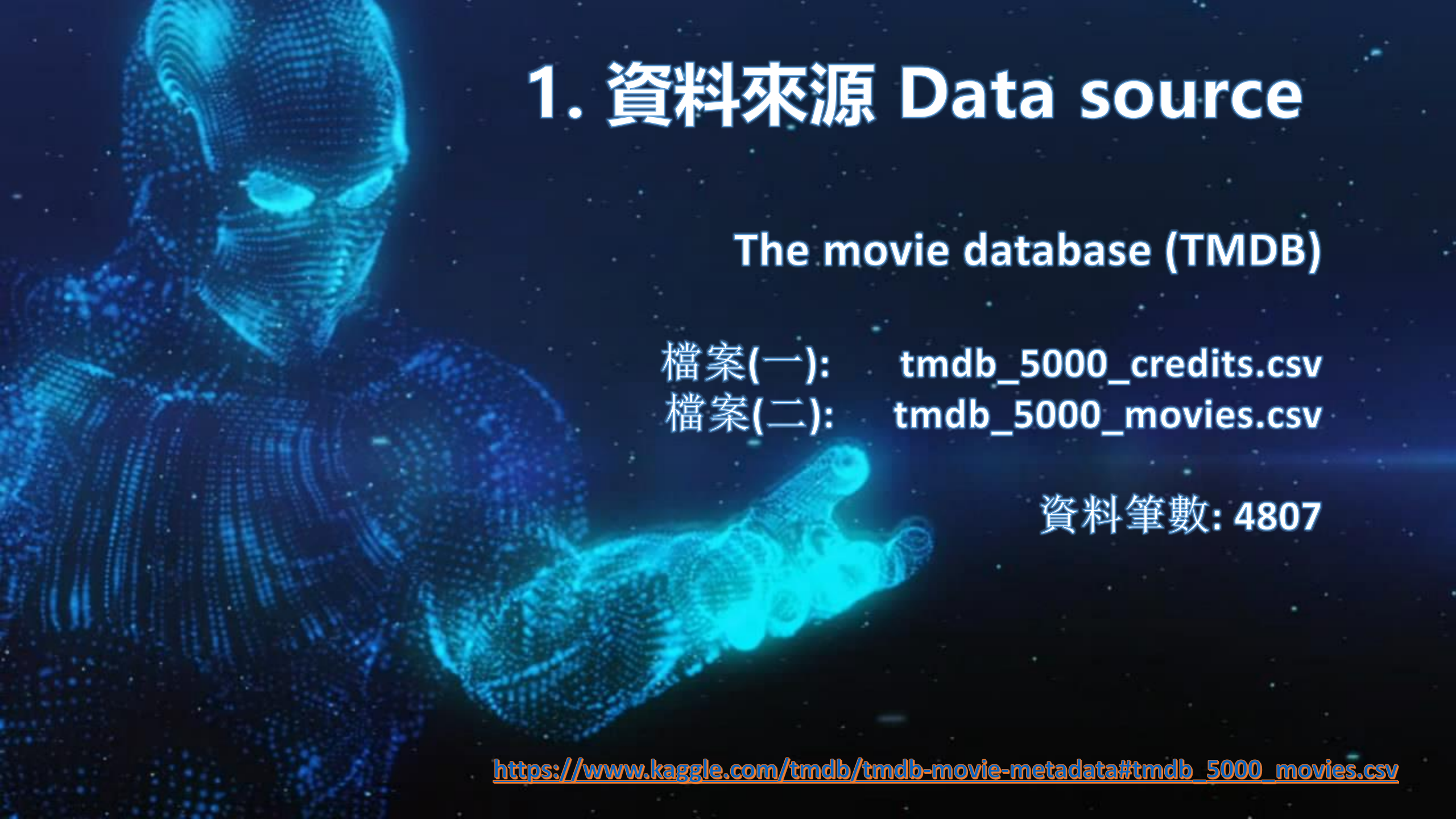
電影
票房預測
Box Office of Films Prediction

劉順勇
D0657457
經濟三乙



1. 資料來源 Data source

The movie database (TMDB)

檔案(一): tmdb_5000_credits.csv
檔案(二): tmdb_5000_movies.csv

資料筆數: 4807

https://www.kaggle.com/tmdb/tmdb-movie-metadata#tmdb_5000_movies.csv

欄位名稱	解釋	備注
'budget'	電影製作預算	
'genres'	電影種類	例子: Action, Adventure, Romance, Thriller
'keywords'	關鍵字	
'original_language'	電影語言(原本)	
'overview'	摘要	
'production_companies'	製作公司	
'production_countries'	製作國家	
'release_date'	上映日期	
‘revenue’	電影票房	
‘runtime’	電視時長	
'spoken_languages'	電影語言	
'status'	電影狀態	共有三個值: ‘Released’, ‘Post production’, ‘Rumored’
'tagline'	宣傳語	
‘title’	電影片名	
‘vote_average’	平均評價	
‘vote_count’	評價數量	
‘cast’	演員	
‘crew’	工作人員	

2. 資料前處理 Data Preprocessing

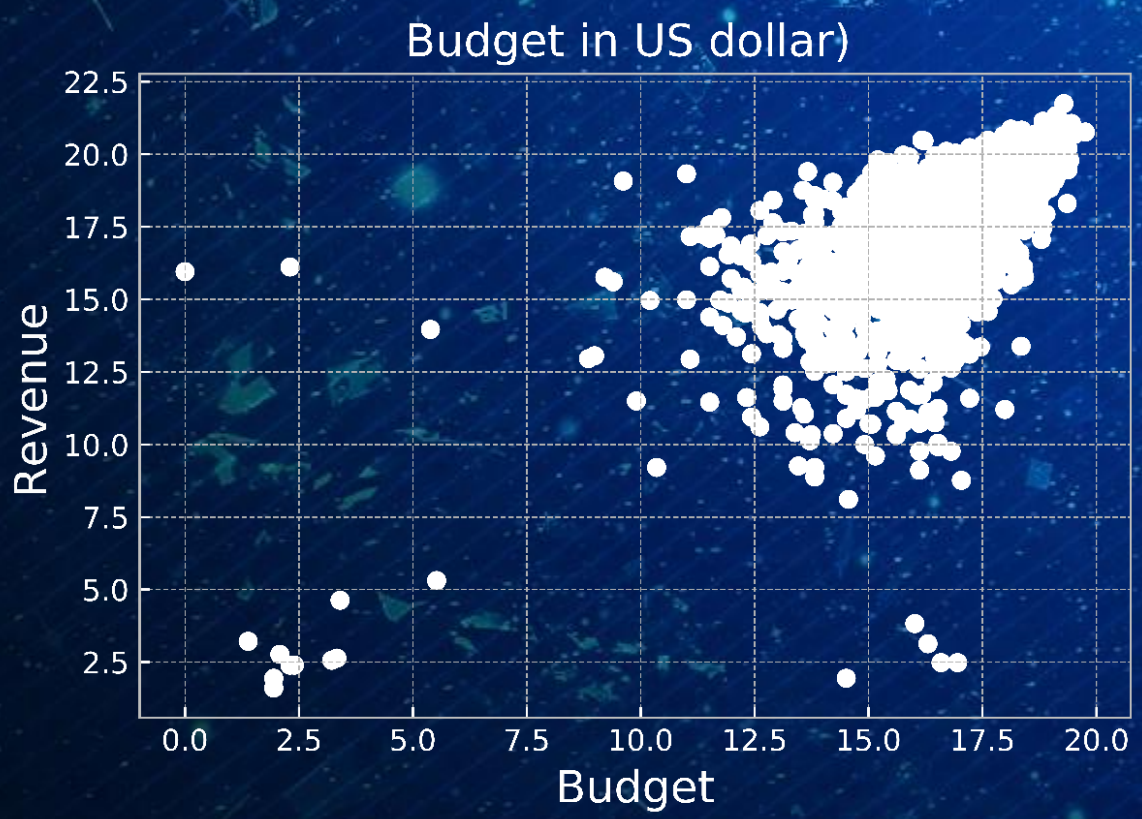
移除無助於預測的欄位
'homepage', 'id', 'original_title', 'popularity'....

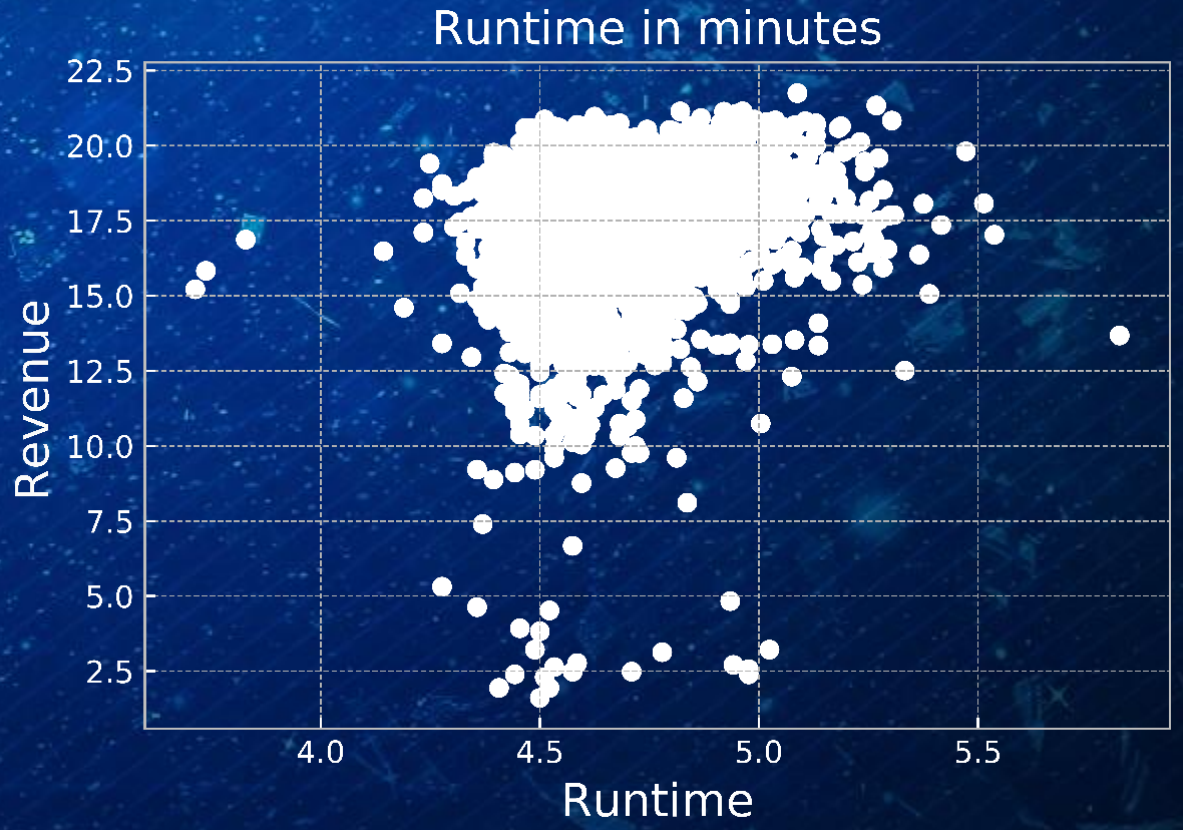
移除還在製作中/謠言中的電影
(status == 'post-production' or status == 'rumored')

移除票房爲零的電影 (revenue == 0)
- 嘗試爬蟲填補遺漏值, 但OMDB的值也為0

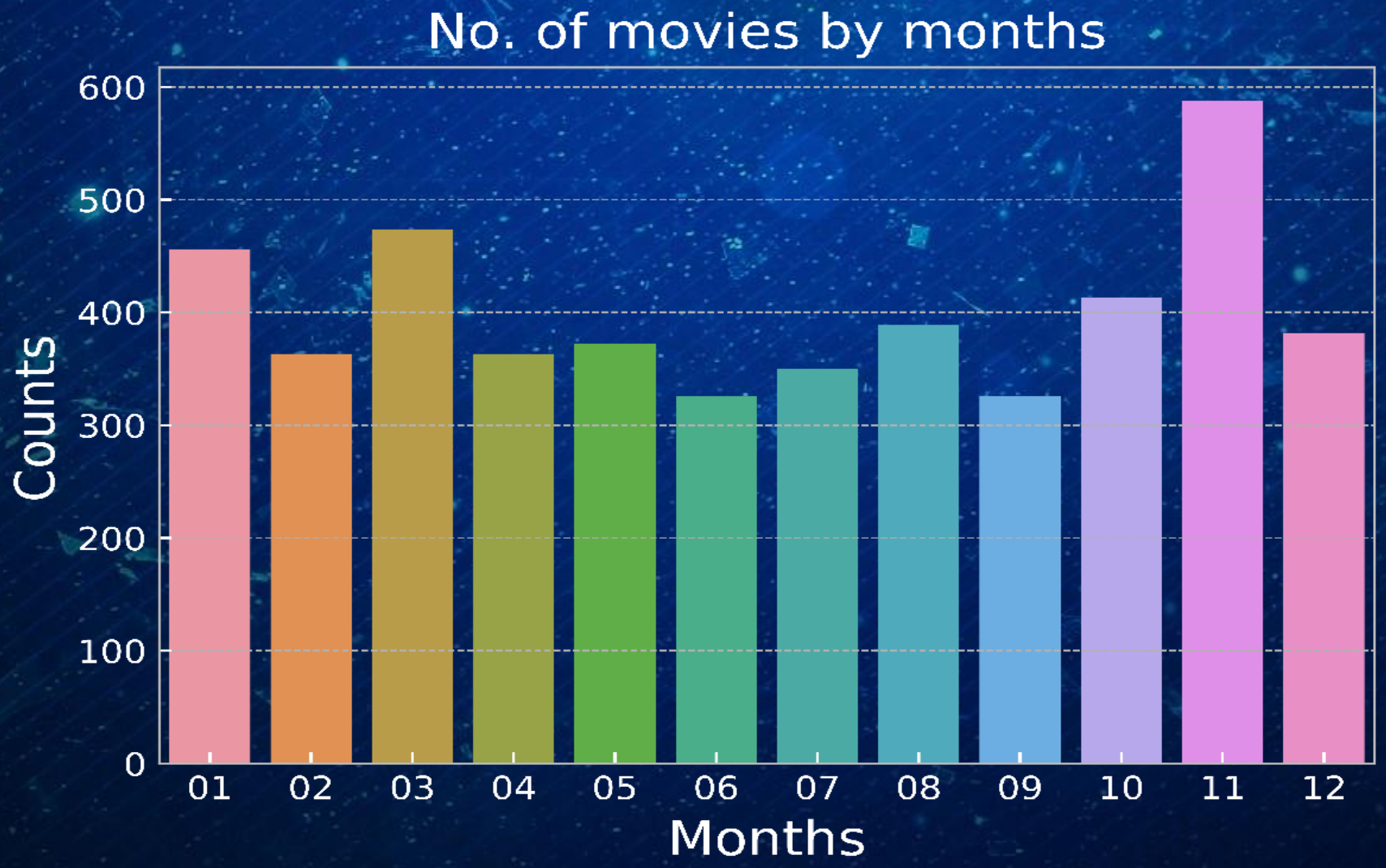
```
15  ''' web scrapping to recover missing data '''
16  ''' Website(1): omdb '''
17  # response = requests.get("http://www.omdbapi.com/?t=the+lover&apikey=e88b14b8")
18  # print(response.status_code)
19  # response.json()
```

3. 探索性分析 Explanatory Data Analysis



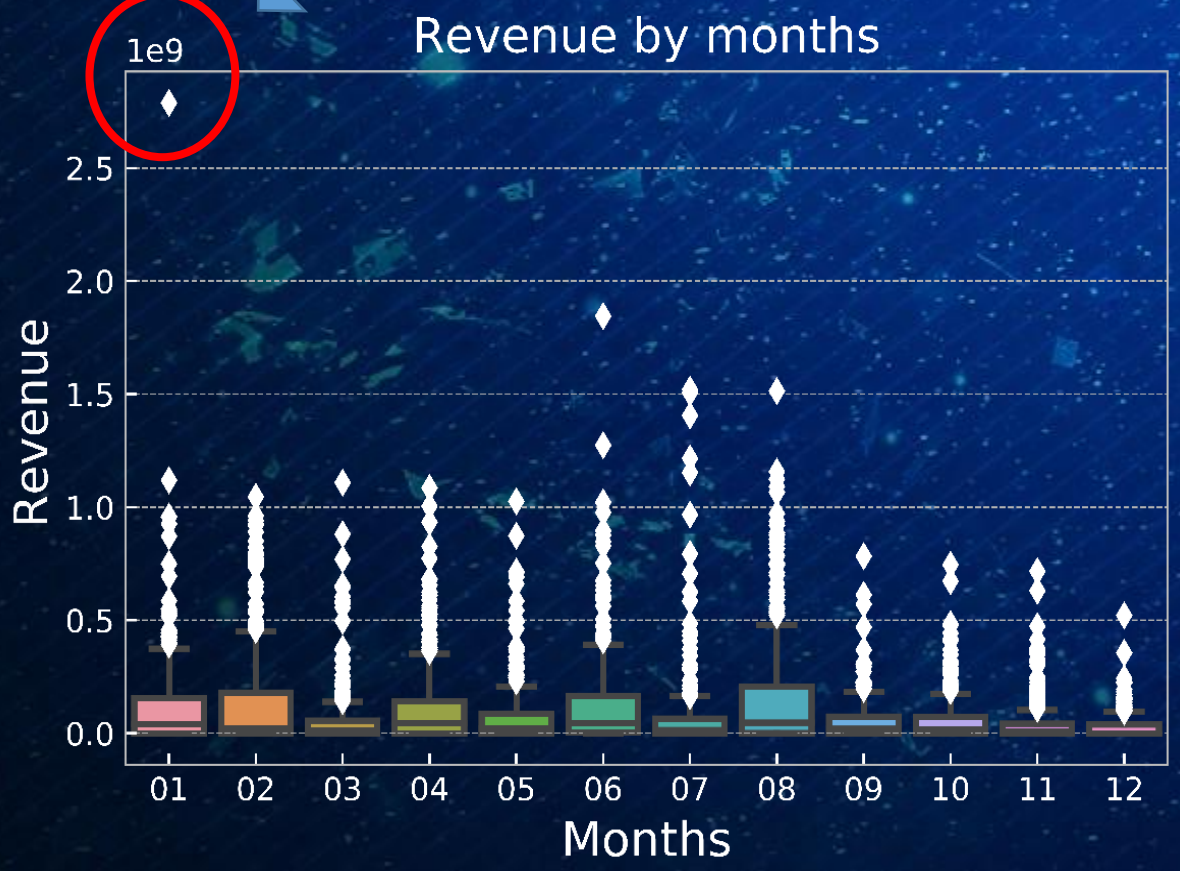


3. 探索性分析 Explanatory Data Analysis

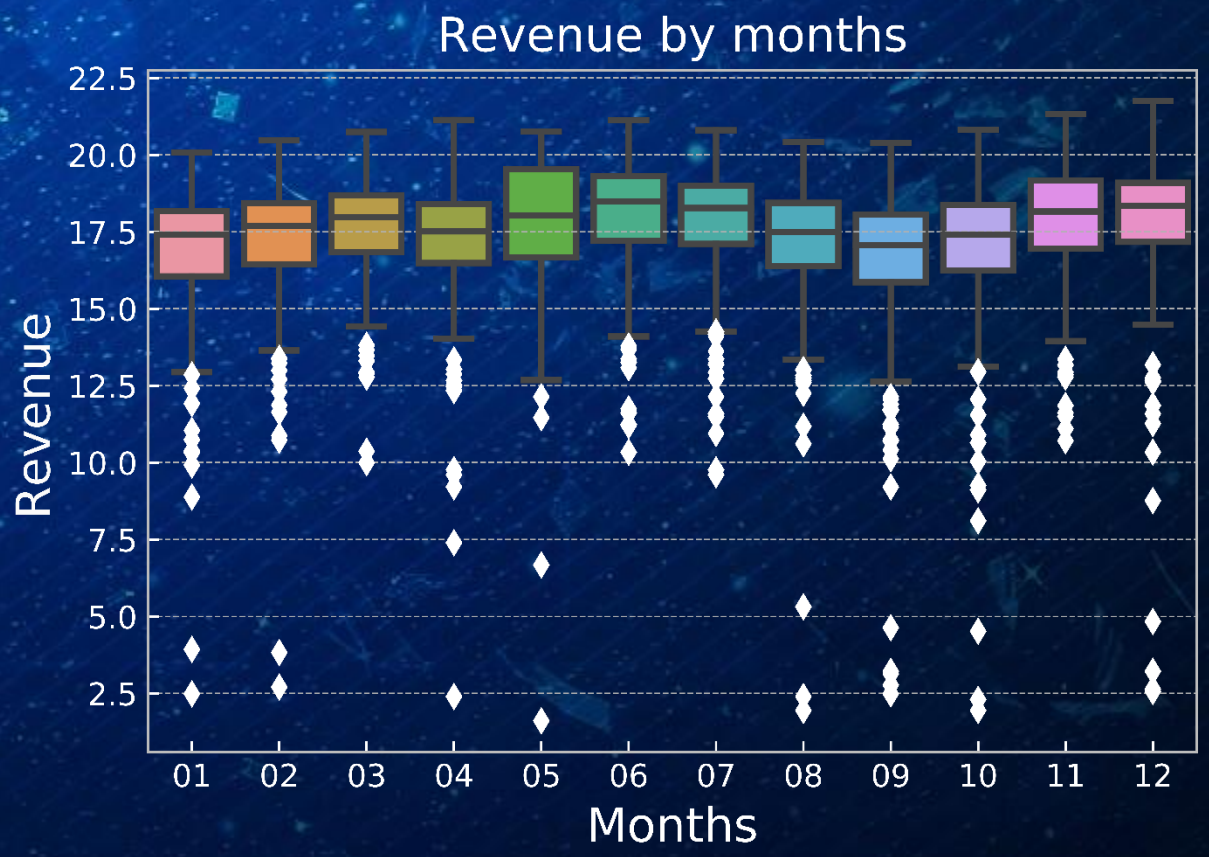


3. 探索性分析 Explanatory Data Analysis

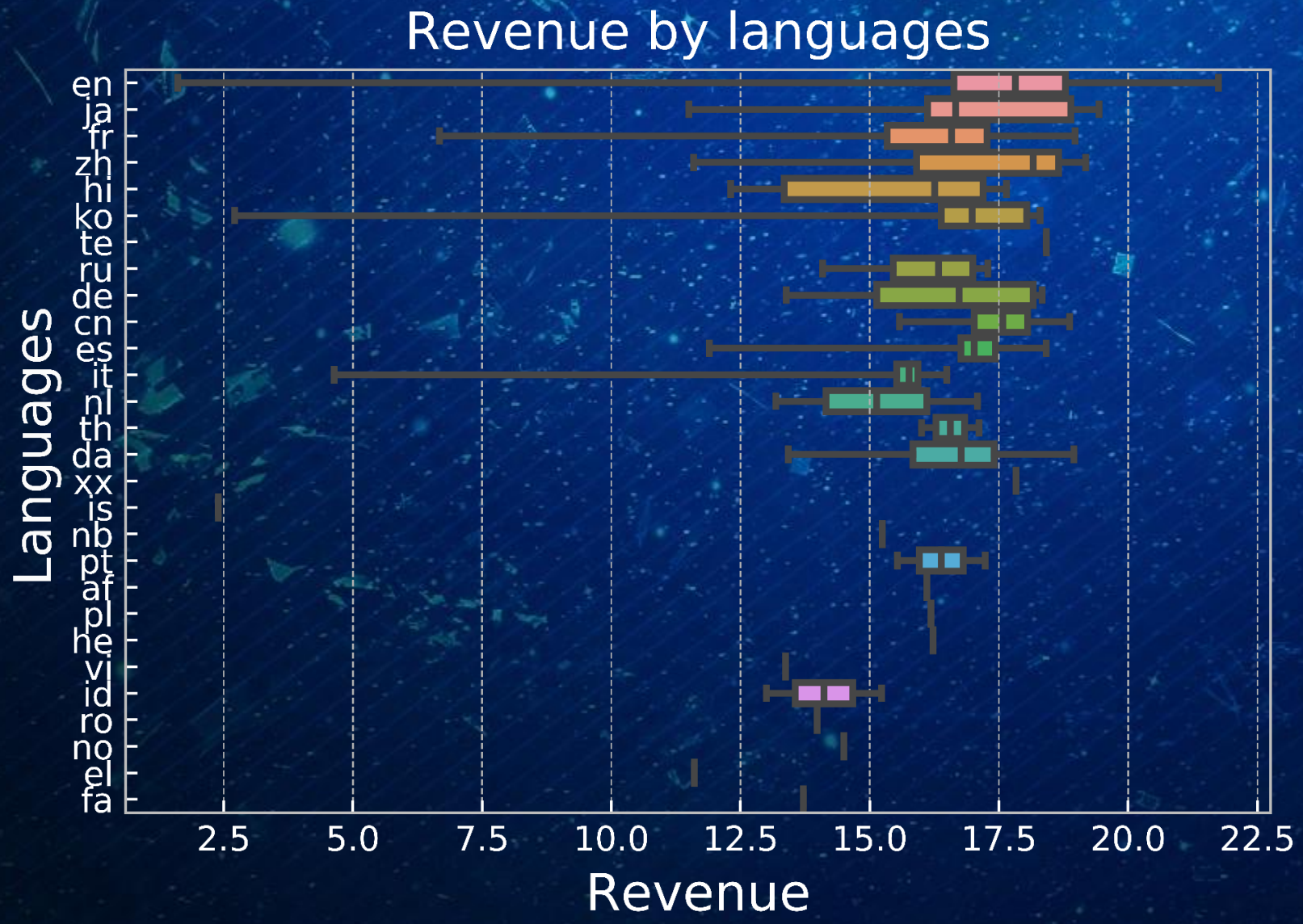
票房第一:阿凡達



移除阿凡達(極端值),
並對revenue取log, 得出下圖:



3. 探索性分析 Explanatory Data Analysis



- 英語片片酬整體較高，但全距很大
- 其餘片酬較高的語種：華語、韓語、泰盧固語

4. 特徵建制 Feature Engineering

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

json的形式

genres	original_language	overview	production_companies	production_countries
[{"id": 28, "name": "Action"}, {"id": 12, "nam...	en	In the 22nd century, a paraplegic Marine is di...	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...

```
{"id": 28, "name": "Action"},
{"id": 12, "name": "Adventure"},
{"id": 14, "name": "Fantasy"},
{"id": 878, "name": "Science Fiction"}]
```

{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"},
{"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]

	id	name	movie
0	28	Action	Avatar
1	12	Adventure	Avatar
2	14	Fantasy	Avatar
3	878	Science Fiction	Avatar

	id				name			
index	0	1	2	3	0	1	2	3
movie								
Avatar	28	12	14	878	Action	Adventure	Fantasy	Science Fiction

有兩部電影的名稱都是The Host, 分別更名為 The Host, The Host_1

```
28  ''' two rows with duplicate index name 'The Host' '''
29  df_genres.index = df_genres.index.where(~df_genres.index.duplicated(), df_genres.index + '_1')
```

Json column 處理的最終成果 (以genre為例) :

	Drama	Science Fiction	Thriller	Fantasy	Comedy	Romance	Crime	Animation	Mystery	Horror	Documentary	Music	Adventure	Family	Western	War
Avatar	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Pirates of the Caribbean: At World's End	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Spectre	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
The Dark Knight Rises	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
John Carter	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

```
''' original_language '''
df_lang = pd.get_dummies(df_movie.loc[:,'original_language'])
df_lang.head()
```

	af	cn	da	de	el	en	es	fa	fr	he	hi	id	is	it	ja	ko	nb	nl
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

對 Overview 進行情感分析 (Sentiment analysis)

Avatar:

'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.’

- 從網路上抓下 IMBD 5萬筆 review 作爲訓練資料，資料的情感由網民的評星決定
- 利用分類模型 Logistic Regression & Multinomial Naïve Bayes 為 overview 評分

https://github.com/aaronkub/machine-learning-examples/blob/master/imdb-sentiment-analysis/movie_data.tar.gz

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

```python
''' text cleaning '''
REPLACE_NO_SPACE = re.compile("[.;:!\'?,\"()\[\]]")
REPLACE_WITH_SPACE = re.compile("(<br\s*/><br\s*/>)|(\-)|(\/)")

def preprocess_reviews(reviews):
    reviews = [REPLACE_NO_SPACE.sub("", line.lower()) for line in reviews]
    reviews = [REPLACE_WITH_SPACE.sub(" ", line) for line in reviews]

    return reviews

reviews_train_clean = preprocess_reviews(reviews_train)
reviews_test_clean = preprocess_reviews(reviews_test)
```

```python
''' Vectorization: convert each review to a numeric representation '''
cv = CountVectorizer(binary=True)
cv.fit(reviews_train_clean)
X = cv.transform(reviews_train_clean)
X_test = cv.transform(reviews_test_clean)
train = scipy.sparse.vstack([X, X_test])
```

https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

情感分析模型的分類結果：

Cure: 'A wave of gruesome murders is sweeping Tokyo. The only connection is a bloody X carved into the neck of each of the victims. In each case, the murderer is found near the victim and remembers nothing of the crime. Detective Takabe and psychologist Sakuma are called in to figure out the connection, but their investigation goes nowhere...'

(rated as negative)

Primer: 'Friends/fledgling entrepreneurs invent a device in their garage that reduces the apparent mass of any object placed inside it, but they accidentally discover that it has some highly unexpected capabilities -- ones that could enable them to do and to have seemingly anything they want. Taking advantage of this unique opportunity is the first challenge they face. Dealing with the consequences is the next.'

(rated as positive)

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

Json to dataframe

```
''' production_countries '''
df_countries = json_to_df('production_countries')

''' two rows with duplicate index name 'The Host' '''
df_countries.index = df_countries.index.where(~df_countries.index.duplicated(), df_countries.index + '_1')
```

22 rows went missing

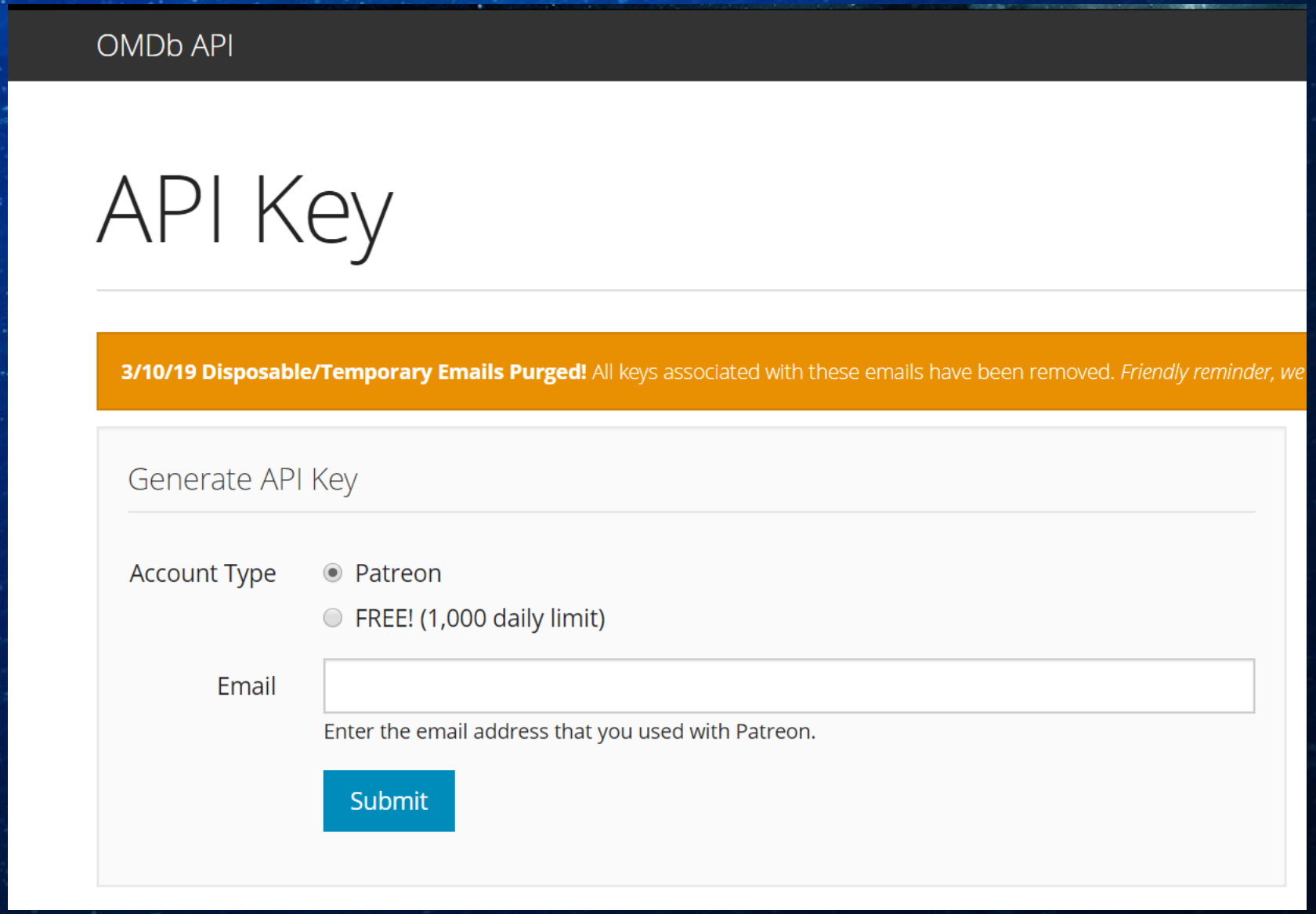
```
''' a movies list with missing countries data '''
missing_countries = list(set.difference(set(df_movie['title']), set(df_countries.index)))
```

Build a list for web scrapping

```
''' build a search list for OMDB api '''
search_list = []
for countries in missing_countries:
  temp = countries.split(' ')
  result = ''
  for words in temp:
    result += '+' + words
  search_list.append(result[1:])
```

OMDB database

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’



	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

Search Result

{'Actors': 'Noah Dalton Danby, Jacque Gray, Bryce Chamberlain, Mark Gollaher', 'Awards': 'N/A', 'BoxOffice': '$1,064,914', 'Country': 'USA', 'DVD': '04 Jul 2006', 'Director': 'Gary Rogers', 'Genre': 'Adventure', 'Language': 'English', 'Metascore': 'N/A', 'Plot': 'The story of Lehi and his wife Sariah and their four sons: Laman, Lemuel, Sam, and Nephi. Lehi leaves Jerusalem because he prophesied unto the people concerning the destruction of Jerusalem...', 'Poster': 'https://m.media-amazon.com/images/M/MV5BMGIzNmY5MjQtOTRmNy00YzNkLTkyMTUtMzk1NGMwZTA3ODRhXkEyXkFqcGdeQXVyNjExODE1MDc@._V1_SX300.jpg', 'Production': 'Mormon Movies', 'Rated': 'PG-13', 'Ratings': [{'Source': 'Internet Movie Database', 'Value': '3.0/10'}, {'Source': 'Rotten Tomatoes', 'Value': '17%'}], 'Released': '12 Jan 2004', 'Response': 'True', 'Runtime': '120 min', 'Title': 'The Book of Mormon Movie, Volume 1: The Journey', 'Type': 'movie', 'Website': 'N/A', 'Writer': 'Craig Clyde (screenplay), Gary Rogers', 'Year': '2003', 'imdbID': 'tt0349159', 'imdbRating': '3.0', 'imdbVotes': '448'}

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

Search Result Correction

```
''' correct data for a faulty search: Town & Country '''
df.drop('The Town That Dreaded Sundown', axis=0, inplace=True)  # remove the wron
t_c = pd.DataFrame(data=['USA'], index=['Town & Country'], columns=[('name',0)])
df = pd.concat([df,t_c])
```

Countries naming

```
countries_map = {'USA':'United States of America', 'UK':'United Kingdom',
                 'India':'India', 'China':'China', 'Australia':'Australia',
                 'South Korea':'South Korea'}
df[('name', 0)] = df[('name', 0)].map(countries_map)
df[('name', 1)] = df[('name', 1)].map(countries_map)
```

Months as a dummy variable

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

```
''' release date '''
df_movie['month'] = df_movie['release_date'].str[5:7]
df_month = pd.get_dummies(df_movie.loc[:, 'month'])
df_month
```

	01	02	03	04	05	06	07	08	09	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0	0

- Json to dataframe
- Replace missing data with data from OMDB

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

```
''' build a search list for OMDB api '''
search_list = []

for movie in missing_lang:
  temp = movie.split(' ')
  result = ''
  for words in temp:
    result += '+' + words
  search_list.append(result[1:])

''' get data from OMBD data '''
temp = []
for movies in search_list:
  url = "http://www.omdbapi.com/?t=" + movies + "&apikey=e88b14b8"
  response = requests.get(url)
  temp.append(response.json())
temp
```

Special case: Silent drama

```
''' The Big Parade is a silent film '''
df.loc['The Big Parade', ('name',0)] = 'No Language'
```

Select top 150 cast for further analysis

	欄位名稱
1	'budget'
2	'genres'
3	'original_language'
4	'overview'
5	'production_companies'
6	'production_countries'
7	'release_date'
8	‘runtime’
9	'spoken_languages'
10	‘cast’

```python
''' Find out all cast and find out top 150 cast'''
cast = []
for i in df_cast.columns[1120:1344]:
  cast.extend(df_cast[i])
cast

from collections import Counter
actor_count = Counter(cast)
sorted_actor = sorted(actor_count.items(), key=operator.itemgetter(1), reverse=True)
```

Part of selected actors list:

1	Samuel L. Jackson	54
2	Robert De Niro	49
3	Matt Damon	45
4	Morgan Freeman	45
5	Bruce Willis	44
6	Johnny Depp	38
7	Owen Wilson	37
8	Brad Pitt	37
9	Liam Neeson	37
10	Nicolas Cage	35

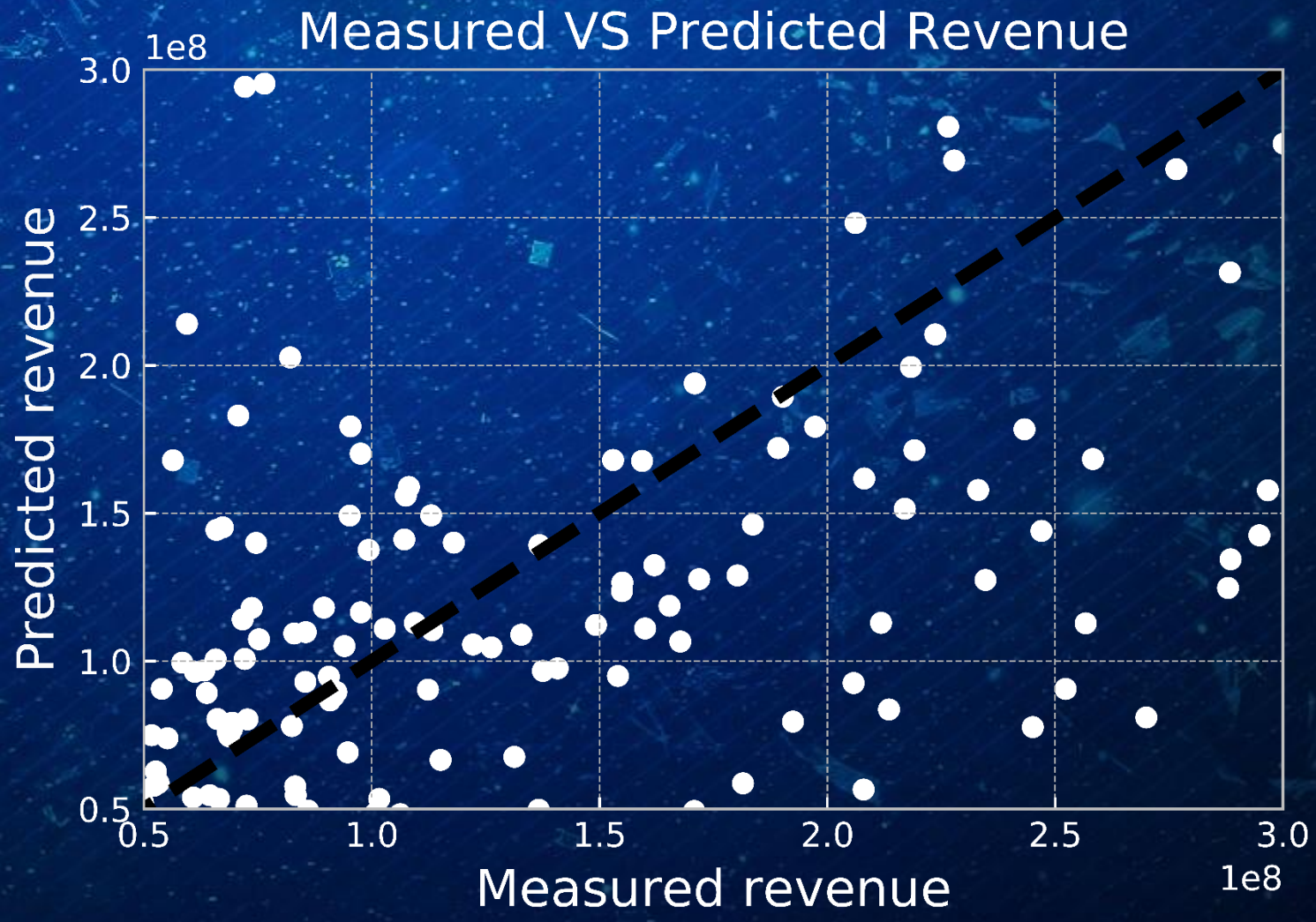
11	Willem Dafoe	35
12	Steve Buscemi	34
13	Stanley Tucci	34
14	Tom Cruise	33
15	Bill Murray	33
16	Ben Stiller	32
17	Alec Baldwin	32
18	Tom Hanks	31
19	Harrison Ford	31

5. 預測模型 Predictive Model

所選用模型: XG Boost

```
model = XGBRegressor
(colsample_bytree= 0.6,
gamma= 0.7, max_depth= 4,
min_child_weight= 5,
subsample = 0.8,
objective='reg:squarederror')
```

嘗試數次的結果:
R^2 介於 0.7-0.8之間



6. 結果分析 Result

1. New Zealand
0.056762554

2. United States of America
0.043951597

3. Australia
0.009809652

4. United Kingdom
0.008604818

5. Germany
0.00798585

6. France
0.004552861

7. Taiwan
0.0025611331

Top Countries
in terms of importance

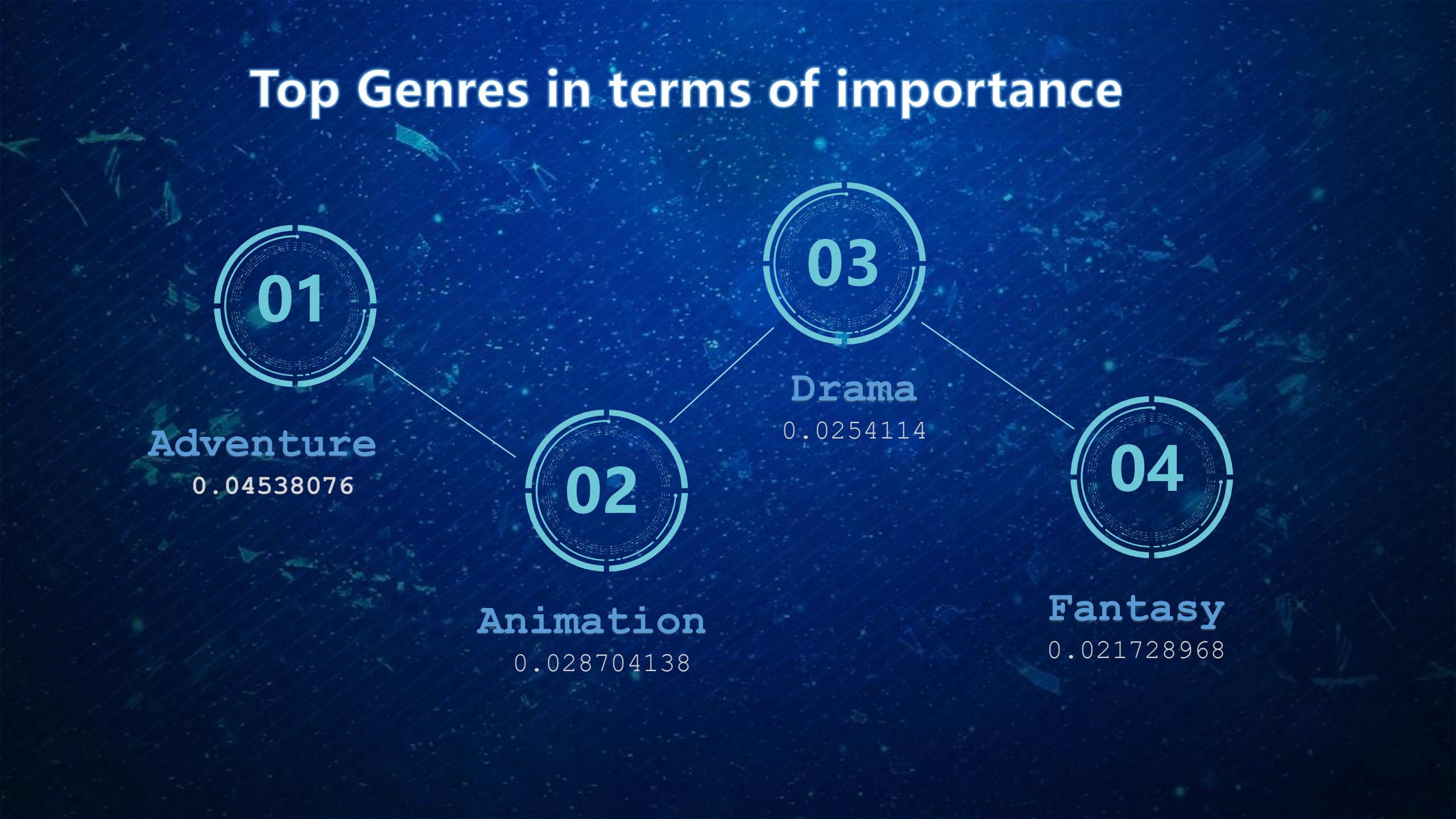

For original language,
only one language matters.
ENG - 0.020950291
For spoken languages
Deutsch 德語 0.016492184
Español 西語 0.0144747645
Svenska 瑞典語 0.008002615
Русский 俄語 0.007924542
Français 法語 0.006309815
Italiano 意大利語 0.0059972126

$ Budget

One of the most important features
Rank 3rd in importance after ‘New Zealand’ and ‘Stan Lee’

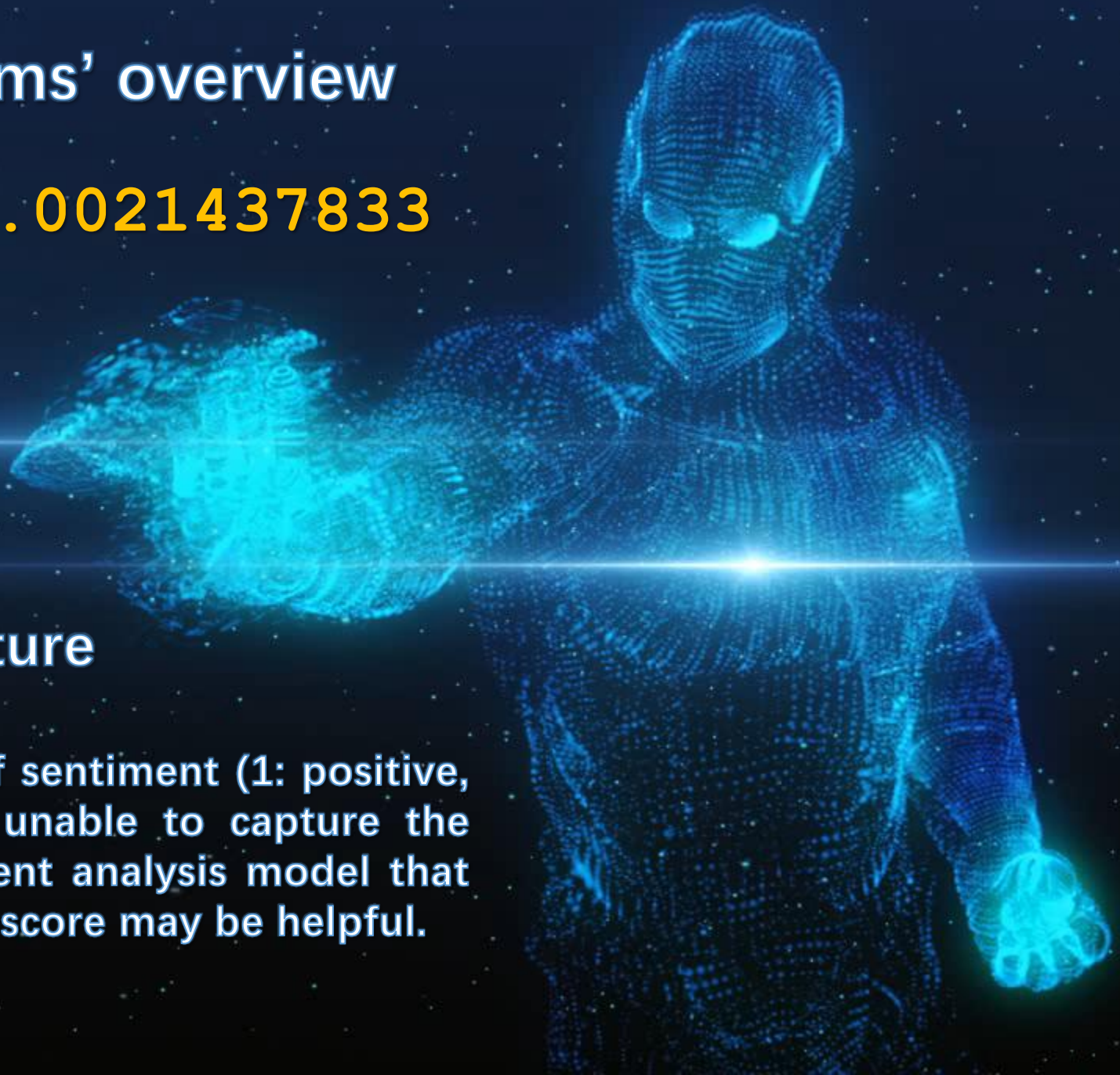
Positive relationship:
The higher the budget,
the better the quality,
the higher the box office

Top Genres in terms of importance



Sentiment score of films' overview

```
'sentiment', 0.0021437833
```

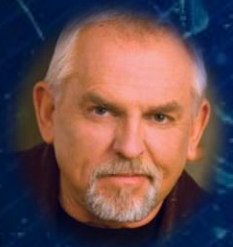


A rather unimportant feature

A simple binary classification of sentiment (1: positive, 0: negative) may have been unable to capture the sentiment accurately. A sentiment analysis model that produces continuous sentiment score may be helpful.

Top 10 Actors

Stan Lee
Marvel Comics' primary creative leader



John Ratzenberger
plays voice roles in every Pixar Animation Studios film

Bill Paxton
played in 'The Terminator', 'Titanic', 'Tombstone' & Marvel films

Christian Bale
Batman Begins (2005) The Dark Knight (2008) and The Dark Knight Rises (2012).

Paul Bettany
A Beautiful Mind (2001), The Da Vinci Code (2006), Margin Call (2011), and Solo: A Star Wars Story (2018).

Most of them have been involved with MARVEL films.

Cate Blanchett
Peter Jackson's The Lord of the Rings trilogy (2001–2003) and The Hobbit trilogy (2012–2014), Cinderella (2015), Thor: Ragnarok (2017)

Steve Carell
Crazy, Stupid, Love (2011), The Incredible Burt Wonderstone and The Way, Way Back (both 2013)

Dwayne Johnson
The Fast and the Furious Jumanji: Welcome to the Jungle (2017), Jumanji: The Next Level (2019).

Gary Oldman
Tinker Tailor Soldier Spy (2011), Dawn of the Planet of the Apes (2014), Darkest Hour (2017)

Alan Tudyk
DodgeBall: A True Underdog Story; I, Robot; A Knight's Tale; 42; Maze Runner: The Scorch Trials, and Rogue One

7. 總結 Conclusion

- 在整個分析過程中，找到適用的資料集其實花了蠻大功夫，因爲自己想到的題目不一定在網路上找到資料。
- 網路上現成的資料有很多需要進行前處理的地方，尤其是遺漏值的部分，這次的分析讓我深深感受到數據清理的重要性。
- 這次的分析也讓我發現我對於連續性的預測值的相關模型并不熟悉，這部分也是我需要加强的地方。


Thank
You!

執行方式

輸入:

原始資料	tmdb_5000_credits.csv, tmdb_5000_movies.csv
情感分析	full_test.txt，full_train.txt
Json處理後資料*	df_cast.csv, df_countries.csv, df_spoken_lang.csv, df_genre.csv

程式碼:　191224_final.ipynb

由於json欄位的處理過程需要較長時間，我將每個欄位處理後的結果輸出成 csv 檔，後續進行分析的時候就可以直接導入處理後結果，不用重新處理。處理json欄位的程式碼可以參考在程式碼中的附錄(1) & (2) 。

參考 Reference

https://www.kaggle.com/tmdb/tmdb-movie-metadata#tmdb_5000_movies.csv

https://github.com/aaronkub/machine-learning-examples/blob/master/imdb-sentiment-analysis/movie_data.tar.gz

https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184