Open in Colab

What is the most successful movie?

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter
import seaborn as sns
import datetime as dt
import pprint
%pprint
%matplotlib inline
```

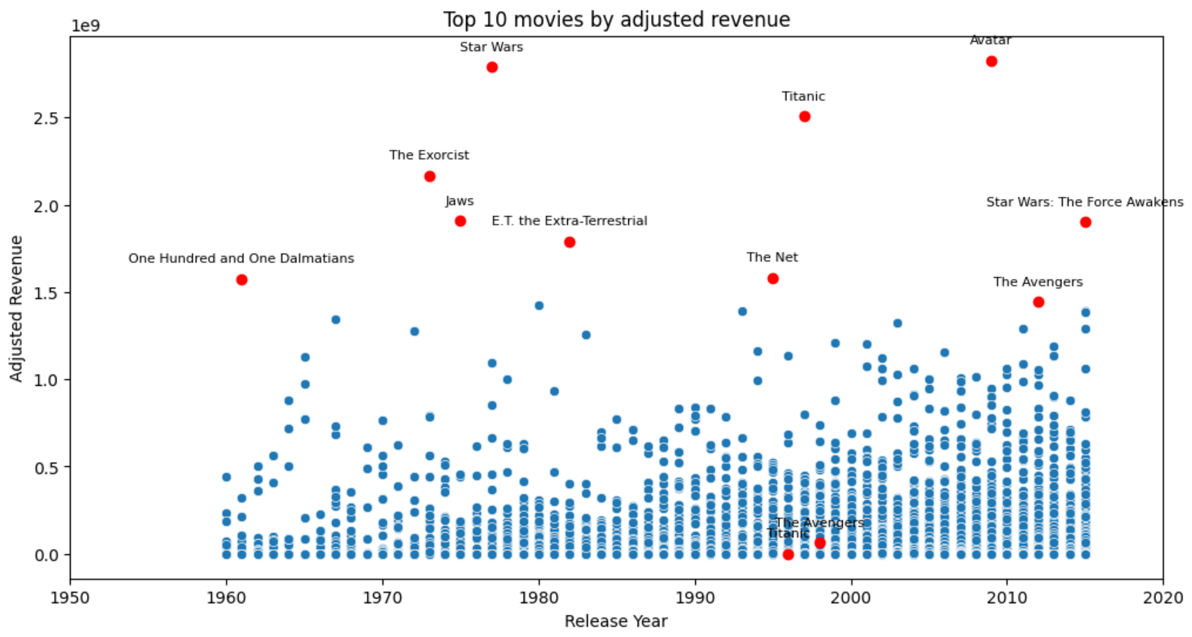
```
Pretty printing has been turned OFF
```

In [111…
```
df = pd.read_csv('../group_projects/tmdb_movies_data.csv')
df.columns
```

Out[111]:
```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_titl
e',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

Top 10 movies by adjusted Revenue

In [112…
```
fig = plt.figure(figsize=(12,6))
# create scatter plot
ax = sns.scatterplot(data=df, x='release_year', y='revenue_adj')

# set top 10 revenue movies to different color
top10_movies = df.sort_values('revenue_adj', ascending=False).head(10)['orig

for i, point in df.iterrows():
    if point['original_title'] in top10_movies:
        ax.scatter(point['release_year'], point['revenue_adj'], color='red')
        ax.annotate(point['original_title'], (point['release_year'], point['

# show every 10 years on x-axis
xticks = ax.get_xticks()
ax.set_xticks(xticks[::1])
ax.set(title='Top 10 movies by adjusted revenue', xlabel='Release Year', yla

plt.show()
```



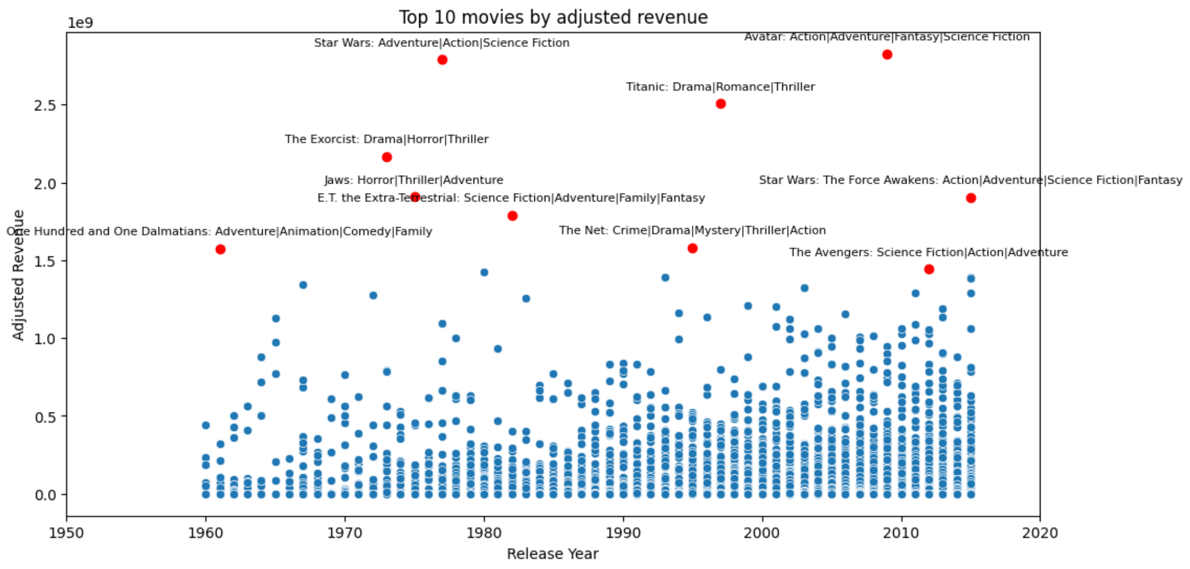
In [21]:
```
df[df['original_title']=='Titanic']
```

Out[21]:

	id	imdb_id	popularity	budget	revenue	original_title	cast
5231	597	tt0120338	4.355219	200000000	1845034188	Titanic	Kate Winslet|Leonardo DiCaprio|Frances Fisher|...
8630	2699	tt0115392	0.219364	13000000	0	Titanic	Peter Gallagher|George C. Scott|Catherine Zeta...

2 rows × 21 columns

In [113…
```
fig = plt.figure(figsize=(12,6))
# create scatter plot
ax = sns.scatterplot(data=df, x='release_year', y='revenue_adj')

# set top 10 revenue movies to different color
top10_movies = df.sort_values('revenue_adj', ascending=False).head(10)
top10_movies_title = [row['original_title'] for index, row in top10_movies.i
top10_movies_id = [row['imdb_id'] for index, row in top10_movies.iterrows()]

for i, point in df.iterrows():
    if point['imdb_id'] in top10_movies_id:
        ax.scatter(point['release_year'], point['revenue_adj'], color='red')
        ax.annotate(point['original_title']+': '+point['genres'], (point['re
# show every 10 years on x-axis
xticks = ax.get_xticks()
ax.set_xticks(xticks[::1])
ax.set(title='Top 10 movies by adjusted revenue', xlabel='Release Year', yla

plt.show()
```



Top 10 movies by ROI

ROI: Return on investment, the ratio of net profit over the total cost of the investment

ROI = (Revenue-Budget)/Budget

```
In [23]: len(df[df['budget_adj']==0])
```

```
Out[23]: 5696
```

```
In [24]: len(df[df['budget_adj']<1000])
```

```
Out[24]: 5754
```

```
In [25]: df['roi'] = df.apply(lambda row: (row['revenue_adj'] - row['budget_adj'])/ro
```

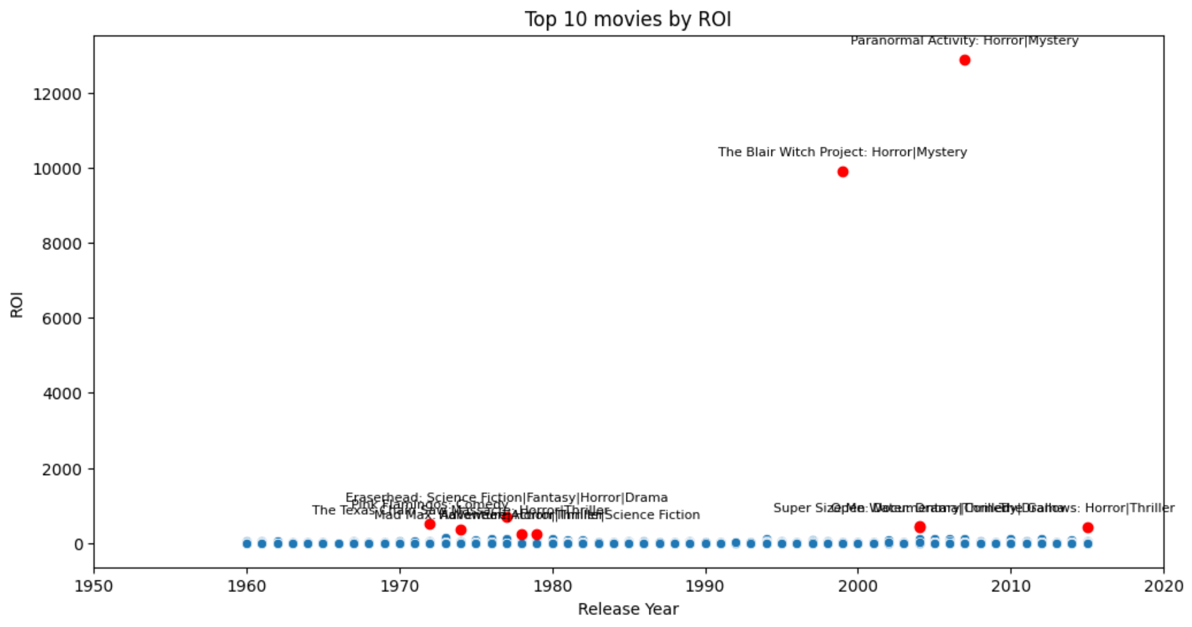
```
In [26]: len(df[df['roi']=='NA'])
```

```
Out[26]: 5754
```

```
In [27]: df_roi = df[df['roi']!='NA']
```

```
In [28]: fig = plt.figure(figsize=(12,6))
         # create scatter plot
         ax = sns.scatterplot(data=df_roi, x='release_year', y='roi')

         # set top 10 revenue movies to different color
```

```
top10_movies = df_roi.sort_values('roi', ascending=False).head(10)
top10_movies_title = [row['original_title'] for index, row in top10_movies.i
top10_movies_id = [row['imdb_id'] for index, row in top10_movies.iterrows()]

for i, point in df_roi.iterrows():
    if point['imdb_id'] in top10_movies_id:
        ax.scatter(point['release_year'], point['roi'], color='red')
        ax.annotate(point['original_title']+': '+point['genres'], (point['re

# show every 10 years on x-axis
xticks = ax.get_xticks()
ax.set_xticks(xticks[::1])
ax.set(title='Top 10 movies by ROI', xlabel='Release Year', ylabel='ROI')

plt.show()
```

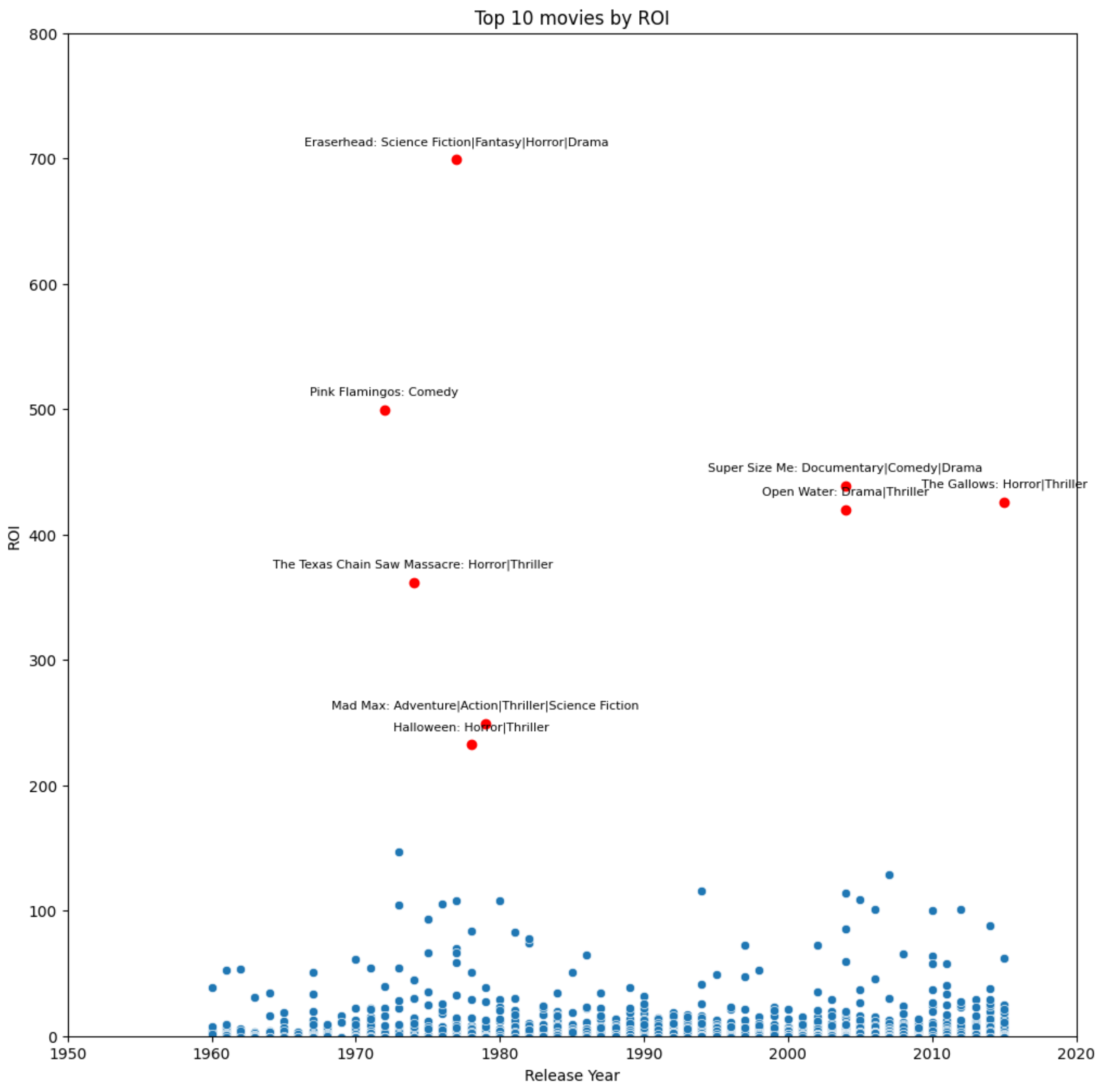


In [29]:
```
df_roi[df_roi['original_title'] == 'Paranormal Activity']['roi']
```

Out[29]:
```
7447    12889.386664
Name: roi, dtype: object
```

In [30]:
```
fig = plt.figure(figsize=(12,12))
# create scatter plot
ax = sns.scatterplot(data=df_roi, x='release_year', y='roi')

# set top 10 revenue movies to different color
top10_movies = df_roi.sort_values('roi', ascending=False).head(10)
top10_movies_title = [row['original_title'] for index, row in top10_movies.i
top10_movies_id = [row['imdb_id'] for index, row in top10_movies.iterrows()]

for i, point in df_roi.iterrows():
    if point['imdb_id'] in top10_movies_id:
        ax.scatter(point['release_year'], point['roi'], color='red')
        ax.annotate(point['original_title']+': '+point['genres'], (point['re

# show every 10 years on x-axis
xticks = ax.get_xticks()
```

```
ax.set_xticks(xticks[::1])

ax.set_ylim(0, 800)

ax.set(title='Top 10 movies by ROI', xlabel='Release Year', ylabel='ROI')

plt.show()
```

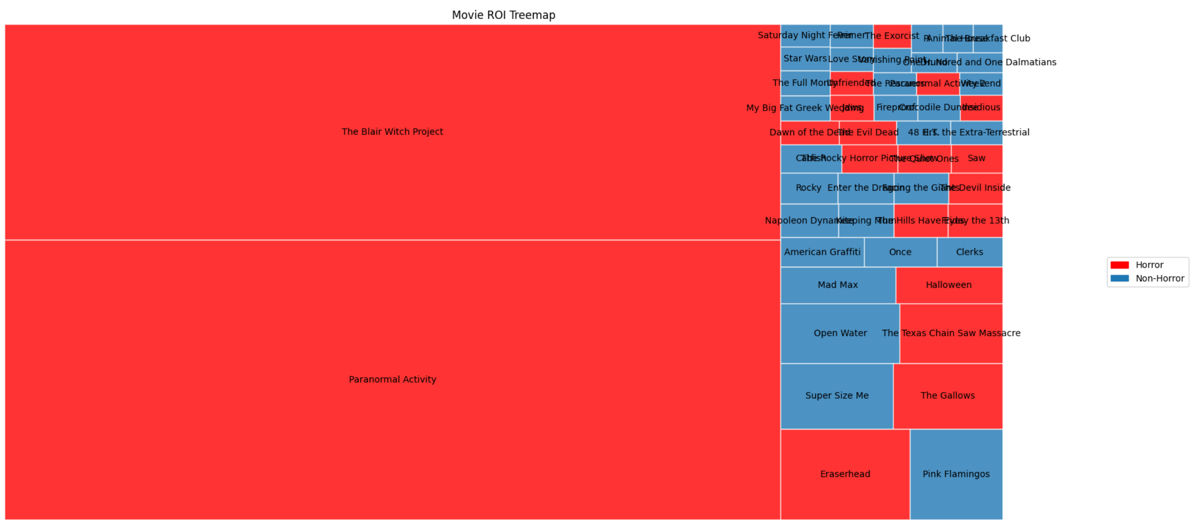


Visual ROI with part-to-whole

Part-to-Whole: charts show how much of a whole an individual part takes up.

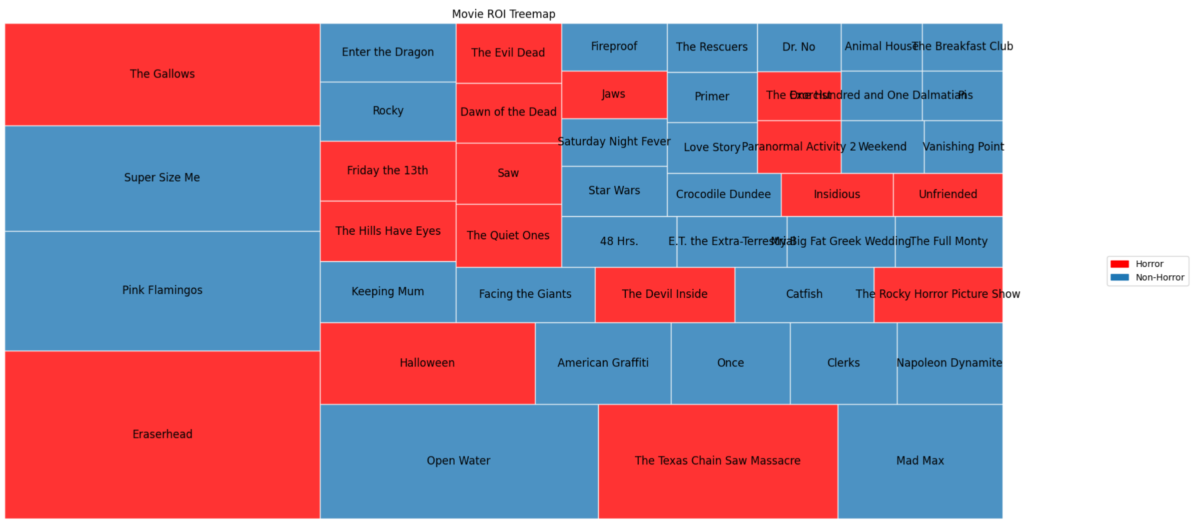
In [103…

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = data

# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Horror' in x else blue)

# calculate treemap sizes
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)


# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')

# add legend
horror_patch = mpatches.Patch(color='red', label='Horror')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Horror')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [104…
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = pd.DataFrame(data[2:])
# df = data

# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Horror' in x else blue)

# calculate treemap sizes
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)


# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')
```

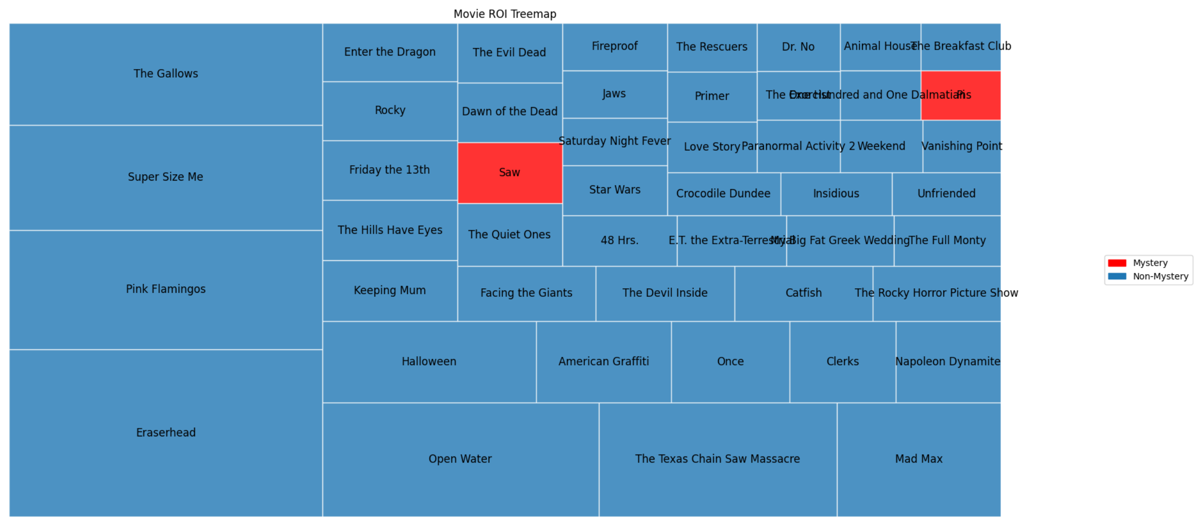
```
# add legend
horror_patch = mpatches.Patch(color='red', label='Horror')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Horror')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [105…

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = pd.DataFrame(data[2:])
# df = data

# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Mystery' in x else blue

# calculate treemap sizes
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)
```

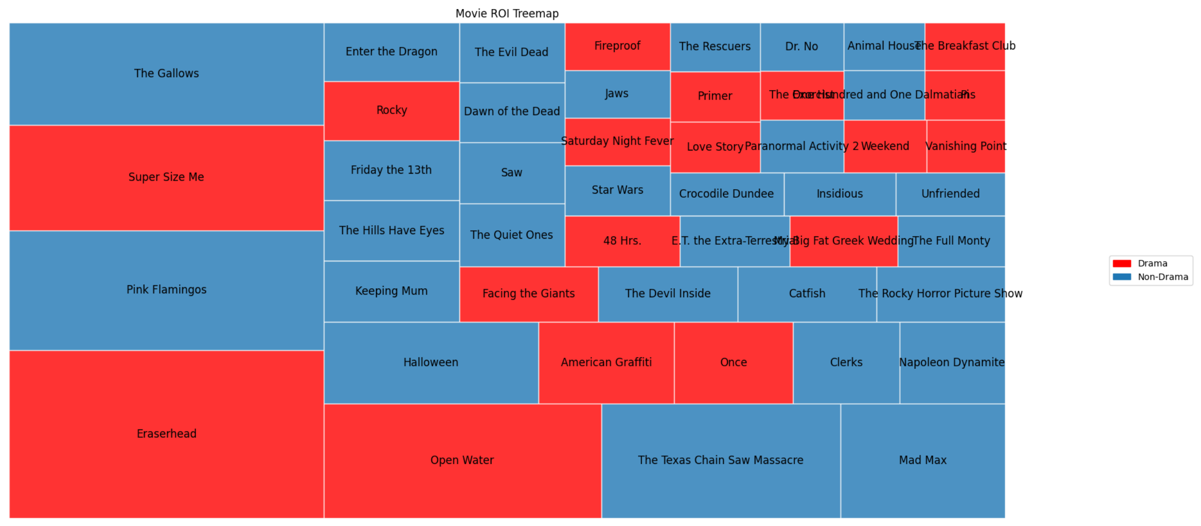
```
# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')

# add legend
horror_patch = mpatches.Patch(color='red', label='Mystery')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Mystery')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [106…

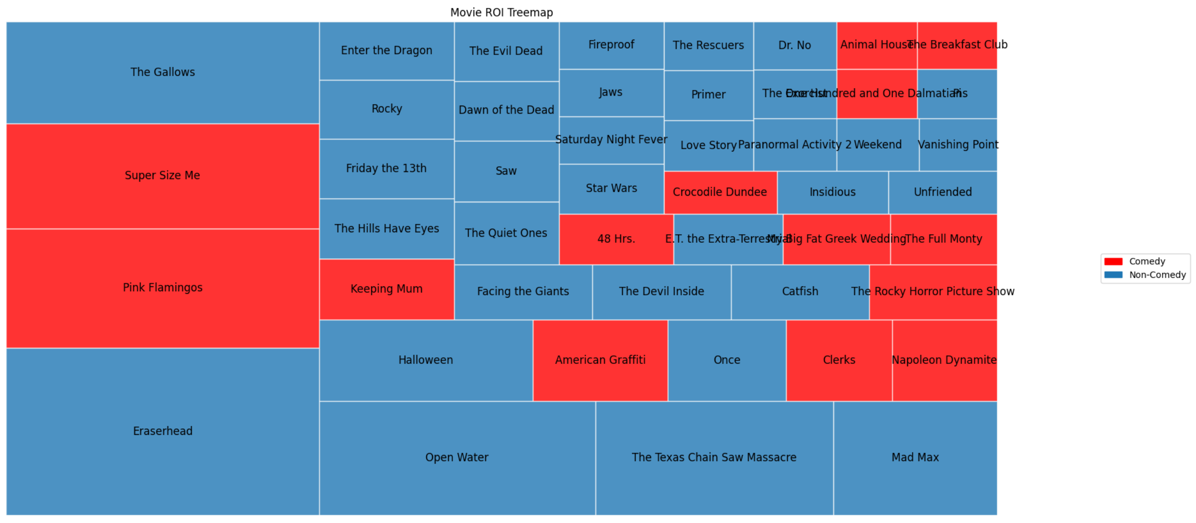
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = pd.DataFrame(data[2:])
# df = data

# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Drama' in x else blue)

# calculate treemap sizes
```

```
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)


# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')

# add legend
horror_patch = mpatches.Patch(color='red', label='Drama')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Drama')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [107…

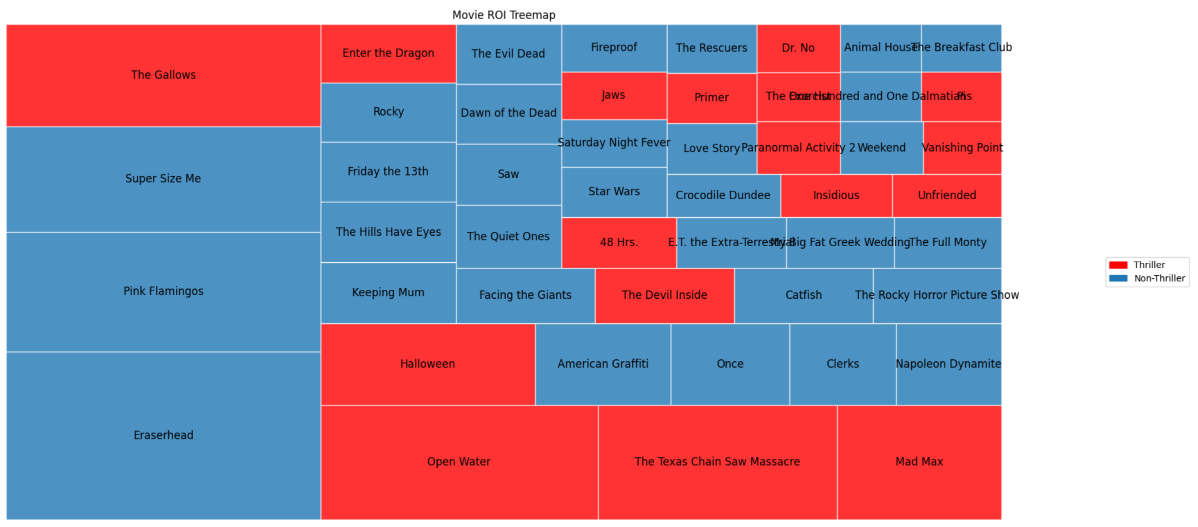
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = pd.DataFrame(data[2:])
# df = data
# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Comedy' in x else blue)

# calculate treemap sizes
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)


# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')

# add legend
horror_patch = mpatches.Patch(color='red', label='Comedy')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Comedy')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [108…

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
```

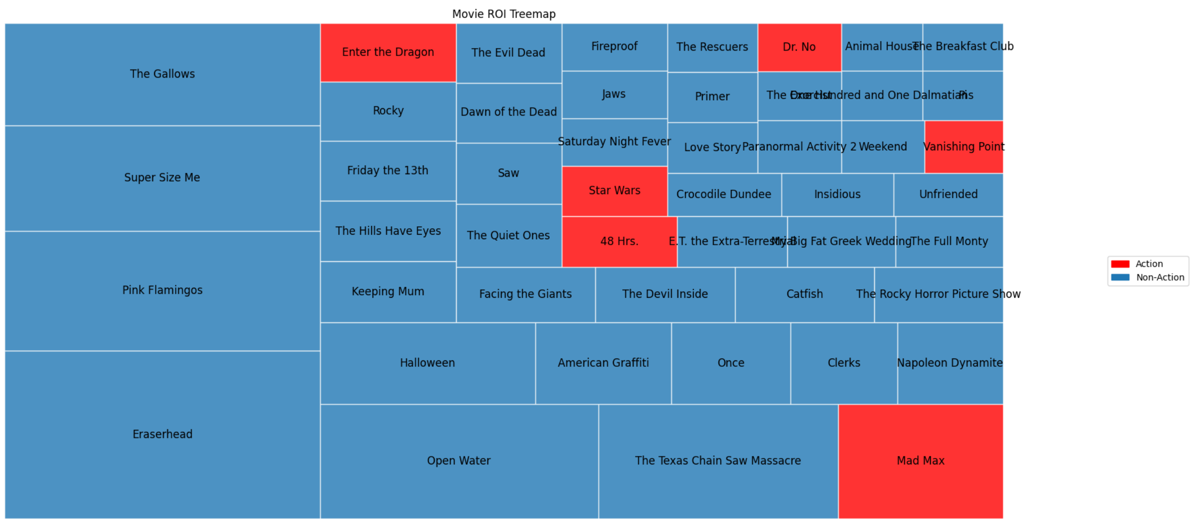
```
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = pd.DataFrame(data[2:])
# df = data

# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Thriller' in x else blu

# calculate treemap sizes
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)


# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')

# add legend
horror_patch = mpatches.Patch(color='red', label='Thriller')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Thriller')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [109…

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import squarify

# create sample data
# set top 10 revenue movies to different color
top50_movies = df_roi.sort_values('roi', ascending=False).head(50)
top50_movies_title = [row['original_title'] for index, row in top50_movies.i
top50_movies_id = [row['imdb_id'] for index, row in top50_movies.iterrows()]

data = top50_movies

df = pd.DataFrame(data[2:])
# df = data

# define color palette
blue = '#1f77b4'

# add color column based on genre
df['color'] = df['genres'].apply(lambda x: 'red' if 'Action' in x else blue)

# calculate treemap sizes
sizes = df['roi'].values
labels = df['original_title'].values
colors = df['color'].values

# define function to map square size to font size
def adjust_font_size(size):
    return int(0.05*size)


# create treemap
plt.figure(figsize=(20, 10))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.8, edgecolor=

# set title and axis labels
plt.title('Movie ROI Treemap')
plt.axis('off')
```

```
# add legend
horror_patch = mpatches.Patch(color='red', label='Action')
non_horror_patch = mpatches.Patch(color=blue, label='Non-Action')
plt.legend(handles=[horror_patch, non_horror_patch], loc='center left', bbox

# show plot
plt.show()
```



In [69]:
```
import sys
sys.path.insert(0, 'src')
import pandas as pd
import matplotlib.pyplot as plt

import mpl_extra.treemap as tr
```

In [ ]:
```
# f = pd.DataFrame({'title':list('ABCDEFG'),
#                    'counts':[100, 30, 25, 2, 2, 2, 2]})
# df['labels'] = [f'{a} - {b}' for a,b in zip(df['title'], df['counts'])]
# plt.figure(figsize=(20, 10))
fig, ax = plt.subplots(figsize=(20,10), dpi=100, subplot_kw=dict(aspect=1.15
df['color'] = df['genres'].apply(lambda x: 'red' if 'Horror' in x else blue)

tr.treemap(ax, df, area='roi', labels='original_title',
           cmap='Set2', fill='color',
           rectprops=dict(ec='w'),
           textprops=dict(c='w'))

ax.axis('off')
```