Project2 Investigate Dataset

May 26, 2022

1 Project 2: Investigate a Dataset

1.1 Table of Content

- Introduction
- Data Wrangling
- Explore the data
- Conclusions

1.2 Introduction

The movie Database was selected. It is popular databse for movies and TV shows.

Lets answer the following questions:

- What are the 10 most popular movies?
- Which 10 films had the biggest budget?
- Which 10 films had the highest revenue?
- which actors did the most amount of movies?
- What are the 10 most used genres?

[8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime

sns.set()

%matplotlib inline
```

1.3 Data Wrangling

1.3.1 Study the data

First, we load the data into a DataFrame and check the first five rows. This is crucial, because we need to get an idea of our data. How many columns it has, how many rows, and so on.

[9]:
```
df = pd.read_csv("tmdb-movies.csv")
df.head()
```

[9]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	…	overview	runtime	genres
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi…	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	…	Twenty-two years after the events of Jurassic …	124	Action|Adventure|Science Fiction|Thriller
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic…	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	…	An apocalyptic story set in the furthest reach…	120	Action|Adventure|Science Fiction|Thriller
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel…	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	…	Beatrice Prior must confront her inner demons …	119	Adventure|Science Fiction|Thriller
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D…	http://www.starwars.com/films/star-wars-episod…	J.J. Abrams	Every generation has a story.	…	Thirty years after defeating the Galactic Empi…	136	Action|Adventure|Science Fiction|Fantasy
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle …	http://www.furious7.com/	James Wan	Vengeance Hits Home	…	Deckard Shaw seeks revenge against Dominic Tor…	137	Action|Adventure|Science Fiction|Thriller

```
4                       Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda…       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ…      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago…      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights …       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

[10]:
```
print(f"There are {df.shape[0]} rows and {df.shape[1]} columns")
```

```
There are 10866 rows and 21 columns
```

Lets analyze a little bit more our data to answer our questions

[11]:
```
df.count()
```

[11]:
```
id                      10866
imdb_id                 10856
popularity              10866
budget                  10866
revenue                 10866
original_title          10866
cast                    10790
homepage                 2936
director                10822
tagline                  8042
keywords                 9373
overview                10862
runtime                 10866
genres                  10843
production_companies     9836
release_date            10866
vote_count              10866
vote_average            10866
release_year            10866
budget_adj              10866
revenue_adj             10866
dtype: int64
```

[12]:
```
df.isnull().sum()
```

[12]:
```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10866 non-null  int64
 1   imdb_id               10856 non-null  object
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64
 4   revenue               10866 non-null  int64
 5   original_title        10866 non-null  object
 6   cast                  10790 non-null  object
 7   homepage              2936 non-null   object
 8   director              10822 non-null  object
 9   tagline               8042 non-null   object
 10  keywords              9373 non-null   object
 11  overview              10862 non-null  object
 12  runtime               10866 non-null  int64
 13  genres                10843 non-null  object
 14  production_companies  9836 non-null   object
 15  release_date          10866 non-null  object
```

```
 16  vote_count            10866 non-null  int64
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

[14]:
```
sum(df.duplicated())
```

[14]: 1

[15]:
```
df.describe()
```

[15]:

	id	popularity	budget	revenue	runtime	\
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	

	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10866.000000	10866.000000	10866.000000	1.086600e+04	1.086600e+04
mean	217.389748	5.974922	2001.322658	1.755104e+07	5.136436e+07
std	575.619058	0.935142	12.812941	3.430616e+07	1.446325e+08
min	10.000000	1.500000	1960.000000	0.000000e+00	0.000000e+00
25%	17.000000	5.400000	1995.000000	0.000000e+00	0.000000e+00
50%	38.000000	6.000000	2006.000000	0.000000e+00	0.000000e+00
75%	145.750000	6.600000	2011.000000	2.085325e+07	3.369710e+07
max	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

[16]:
```
df.query('budget == 0')['budget'].count(), df.query('revenue == 0')['revenue'].
↪count()
```

[16]: (5696, 6016)

There are a total of 10866 entries, includind one duplicated row. In the columns we are interested in, there are 76 missing values in the cast column. Here we notice that there are a lot of zero in the columns budget and revenue columns.

1.4 Data Cleaning

First, remove the duplicated lines

[17]:
```
df_dropped = df.drop_duplicates()
```

[18]:
```
df_dropped = df[['id', 'original_title', 'budget', 'revenue', 'cast',
                 'genres', 'release_date', 'runtime', 'popularity',␣
 ↪'vote_average']].copy()
df_dropped.head()
```

[18]:

	id	original_title	budget	revenue	cast	genres	release_date	runtime	popularity	vote_average
0	135397	Jurassic World	150000000	1513528810	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi…	Action|Adventure|Science Fiction|Thriller	6/9/15	124	32.985763	6.5
1	76341	Mad Max: Fury Road	150000000	378436354	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic…	Action|Adventure|Science Fiction|Thriller	5/13/15	120	28.419936	7.1
2	262500	Insurgent	110000000	295238201	Shailene Woodley|Theo James|Kate Winslet|Ansel…	Adventure|Science Fiction|Thriller	3/18/15	119	13.112507	6.3
3	140607	Star Wars: The Force Awakens	200000000	2068178225	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D…	Action|Adventure|Science Fiction|Fantasy	12/15/15	136	11.173104	7.5
4	168259	Furious 7	190000000	1506249360	Vin Diesel|Paul Walker|Jason Statham|Michelle …	Action|Crime|Thriller	4/1/15	137	9.335014	7.3

[19]:
```
print(f"The shape of df_droppend is: {df_dropped.shape}")
```

```
The shape of df_droppend is: (10866, 10)
```

Lets delete the missing values

[20]:
```
df_dropped.dropna(inplace=True)
```

[21]:
```
df_dropped.query('budget == 0')['id'].count()
```

[21]:
```
5610
```

[22]:
```
df_dropped.query('revenue == 0')['id'].count()
```

[22]:
```
5923
```

[23]:
```
df_dropped.query('runtime == 0')['id'].count()
```

[23]: 30

With the budget and revenue columns, more than half of the dataset would be dropped, we would be losing too much data. However we can't keep them in the data set like that as it could affect our future analysis. We decide to replace these zeros with null values

[24]:
```
# budget
df_dropped['budget'].replace(0, np.NaN, inplace=True)

# revenue
df_dropped['revenue'].replace(0, np.NaN, inplace=True)

# runtime
df_dropped.query('runtime != 0', inplace=True)
```

[25]:
```
print(f"The shape of df_droppend is: {df_dropped.shape}")

df_dropped.isnull().sum()
```

```
The shape of df_droppend is: (10738, 10)
```

[25]:
```
id                   0
original_title       0
budget            5583
revenue           5893
cast                 0
genres               0
release_date         0
runtime              0
popularity           0
vote_average         0
dtype: int64
```

[26]:
```
def set_date(date):
    dt = datetime.datetime.strptime(date, '%m/%d/%y')
    if dt.year > 2050:
        dt = dt.replace(year = dt.year - 100)
    return dt


df_dropped['release_date'] = df_dropped['release_date'].apply(lambda x:␣
 ↪set_date(x))
```

[27]:
```
# cast column
def split_cell(df, old_column, new_column, delimiter):
    return df[old_column] \
```

```
            .apply(lambda x: x.split(delimiter)) \
            .apply(pd.Series) \
            .merge(df, left_index = True, right_index = True) \
            .drop([old_column], axis=1) \
            .melt(id_vars=tmp_columns, value_name=new_column) \
            .drop('variable', axis=1) \
            .dropna(subset=[new_column])

tmp_columns = list(df_dropped.columns.values)
del tmp_columns[4]

df_cast_splitted = split_cell(df_dropped, 'cast', 'actor', '|')

df_cast_splitted.head(2)
```

[27]:

```
      id      original_title       budget       revenue  \
0  135397      Jurassic World  150000000.0  1.513529e+09
1   76341  Mad Max: Fury Road  150000000.0  3.784364e+08

                                      genres release_date  runtime  \
0  Action|Adventure|Science Fiction|Thriller   2015-06-09      124
1  Action|Adventure|Science Fiction|Thriller   2015-05-13      120

   popularity  vote_average        actor
0   32.985763           6.5  Chris Pratt
1   28.419936           7.1    Tom Hardy
```

[28]:

```
# genres column
tmp_columns = list(df_dropped.columns.values)
del tmp_columns[5]

df_genres_splitted = split_cell(df_dropped, 'genres', 'genre', '|')

df_genres_splitted.head(2)
```

[28]:

```
      id      original_title       budget       revenue  \
0  135397      Jurassic World  150000000.0  1.513529e+09
1   76341  Mad Max: Fury Road  150000000.0  3.784364e+08

                                                cast release_date  runtime  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi…   2015-06-09      124
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic…   2015-05-13      120

   popularity  vote_average   genre
0   32.985763           6.5  Action
1   28.419936           7.1  Action
```

1.5 Data Analysis

1.5.1 What are the 10 most popular movies?

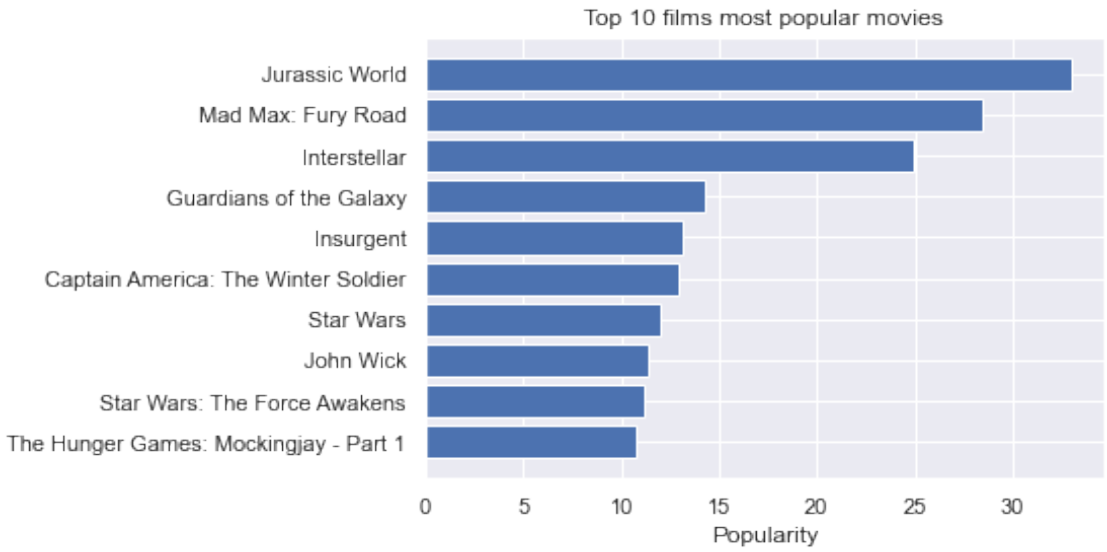
Using the column "popularity" we can see whioch movies are more popular

[29]:
```
def barh_chart_h(locations, heights, labels, title, xlabel):
    plt.barh(locations, heights, tick_label=labels)
    plt.gca().invert_yaxis()
    plt.title(title)
    plt.xlabel(xlabel);

def plot_year(x, y, title, xlabel, ylabel):
    plt.plot(x, y, 'y-')
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel);

res = df_dropped.sort_values(by='popularity', ascending=False)[['id',␣
 ↪'original_title', 'popularity']].head(10)
res
```

[29]:

	id	original_title	popularity
0	135397	Jurassic World	32.985763
1	76341	Mad Max: Fury Road	28.419936
629	157336	Interstellar	24.949134
630	118340	Guardians of the Galaxy	14.311205
2	262500	Insurgent	13.112507
631	100402	Captain America: The Winter Soldier	12.971027
1329	11	Star Wars	12.037933
632	245891	John Wick	11.422751
3	140607	Star Wars: The Force Awakens	11.173104
633	131631	The Hunger Games: Mockingjay - Part 1	10.739009

[30]:
```
locations = list(range(10))
heights = res['popularity']

# Bar char
barh_chart_h(locations, heights, res['original_title'],
             'Top 10 films most popular movies', 'Popularity')
```



The most popular film at the moment is Jurassic World with a score of 32.985763, followed by Mad Max: Fury Road with a score of 28.419936 and Interstellar with a score of 24.949134. Also we can see that the top 10 most popular movies at the moment ends with The Hunger Games: Mockingjay - Part 1 with a score of 10.739009.

1.5.2 Which 10 films had the biggest budget?

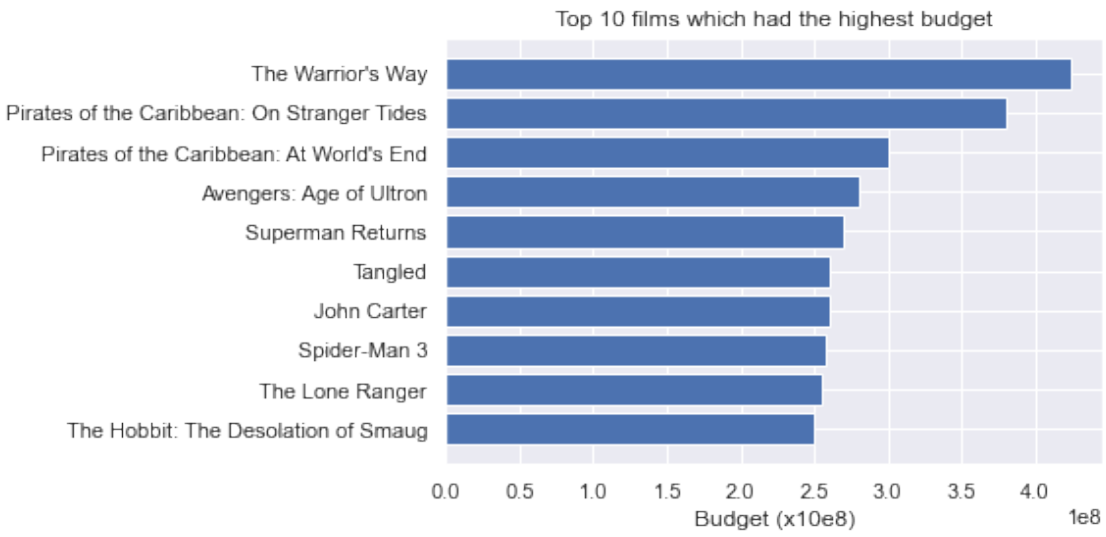
[31]:
```
res = df_dropped.sort_values(by='budget', ascending=False)[['id',
                                                            'original_title',
                                                            'budget']].head(10)
res
```

[31]:

	id	original_title	budget
2244	46528	The Warrior's Way	425000000.0
3375	1865	Pirates of the Caribbean: On Stranger Tides	380000000.0
7387	285	Pirates of the Caribbean: At World's End	300000000.0
14	99861	Avengers: Age of Ultron	280000000.0
6570	1452	Superman Returns	270000000.0
1929	38757	Tangled	260000000.0
4411	49529	John Carter	260000000.0
7394	559	Spider-Man 3	258000000.0
5508	57201	The Lone Ranger	255000000.0
5431	57158	The Hobbit: The Desolation of Smaug	250000000.0

[32]:
```
locations = list(range(10))
heights = res['budget']

# Bar char
barh_chart_h(locations, heights, res['original_title'],
```

```
            'Top 10 films which had the highest budget', 'Budget (x10e8)')
```



The film which had the biggest budget was The Warrior's Way with a budget of 425,000,000, followed by Pirates of the Caribbean: On Stranger Tides with a budget of 380,000,000 and Pirates of the Caribbean: At World's End with a budget of 300,000,000. The top 10 films which had the biggest budget ends with The Hobbit: The Desolation of Smaug with a budget of 250,000,000.

1.5.3 Which 10 films had the highest revenue?

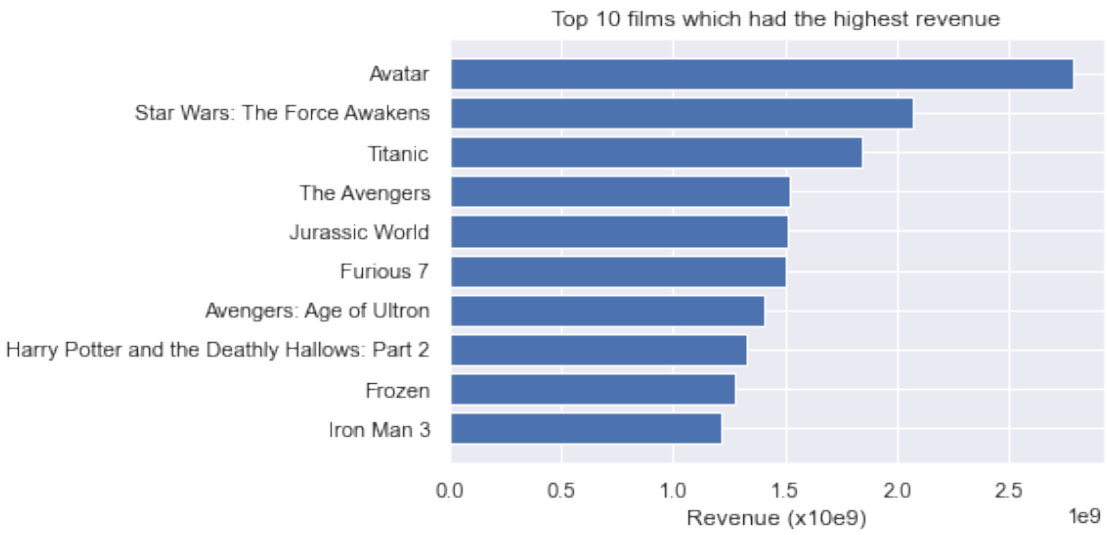
[33]:
```
res = df_dropped.sort_values(by='revenue', ascending=False)[['id',␣
 ↪'original_title', 'revenue']].head(10)
res
```

[33]:

	id	original_title	revenue
1386	19995	Avatar	2.781506e+09
3	140607	Star Wars: The Force Awakens	2.068178e+09
5231	597	Titanic	1.845034e+09
4361	24428	The Avengers	1.519558e+09
0	135397	Jurassic World	1.513529e+09
4	168259	Furious 7	1.506249e+09
14	99861	Avengers: Age of Ultron	1.405036e+09
3374	12445	Harry Potter and the Deathly Hallows: Part 2	1.327818e+09
5422	109445	Frozen	1.274219e+09
5425	68721	Iron Man 3	1.215440e+09

[34]:
```
locations = list(range(10))
heights = res['revenue']

# Bar char
```

```
barh_chart_h(locations, heights, res['original_title'],
             'Top 10 films which had the highest revenue', 'Revenue (x10e9)')
```



The film which had the highest revenue was Avatar with a revenue of 2,781,506,000, followed by Star Wars: The Force Awakens with a revenue of 2,068,178,000 and Titanic with a revenue of 1,845,034,000. The top 10 films which had the highest revenue ends with Iron Man 3 with a budget of 1,215,440,000.

1.5.4 Which actors did the most amount of movies?

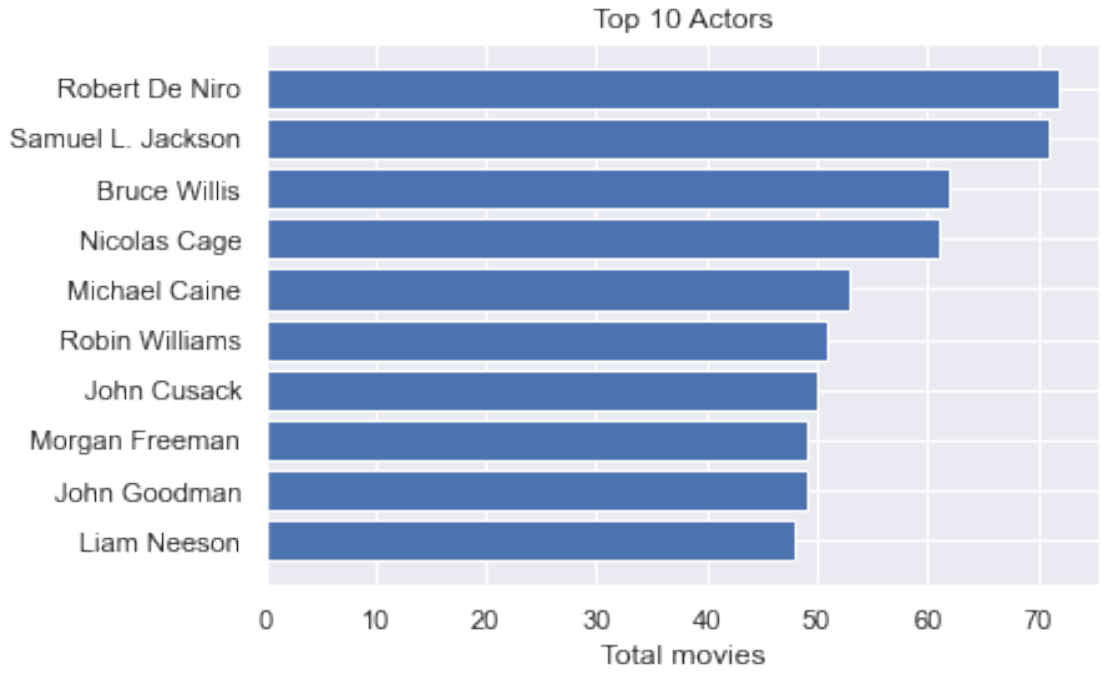
[35]:
```
res = df_cast_splitted.groupby(['actor'], as_index=False)['id'].count().
 ↪sort_values(by='id', ascending=False).head(10)
res = res.rename(columns={'id':'total_movies'})
res
```

[35]:

	actor	total_movies
15259	Robert De Niro	72
15995	Samuel L. Jackson	71
2496	Bruce Willis	62
13550	Nicolas Cage	61
12519	Michael Caine	53
15438	Robin Williams	51
8903	John Cusack	50
13127	Morgan Freeman	49
8944	John Goodman	49
10965	Liam Neeson	48

[36]:
```
locations = list(range(10))
heights = res['total_movies']
```

```
# bar char
barh_chart_h(locations, heights, res['actor'], 'Top 10 Actors', 'Total movies')
```



The actor with the most amount of movies is Robert De Niro with 72 movies, followed by Samuel L. Jackson with 71 movies and Bruce Willis with 62 movies. The top 10 actors with the most amount of movies ends with Liam Neeson with 48 movies.

1.5.5 What are the 10 most used genres?

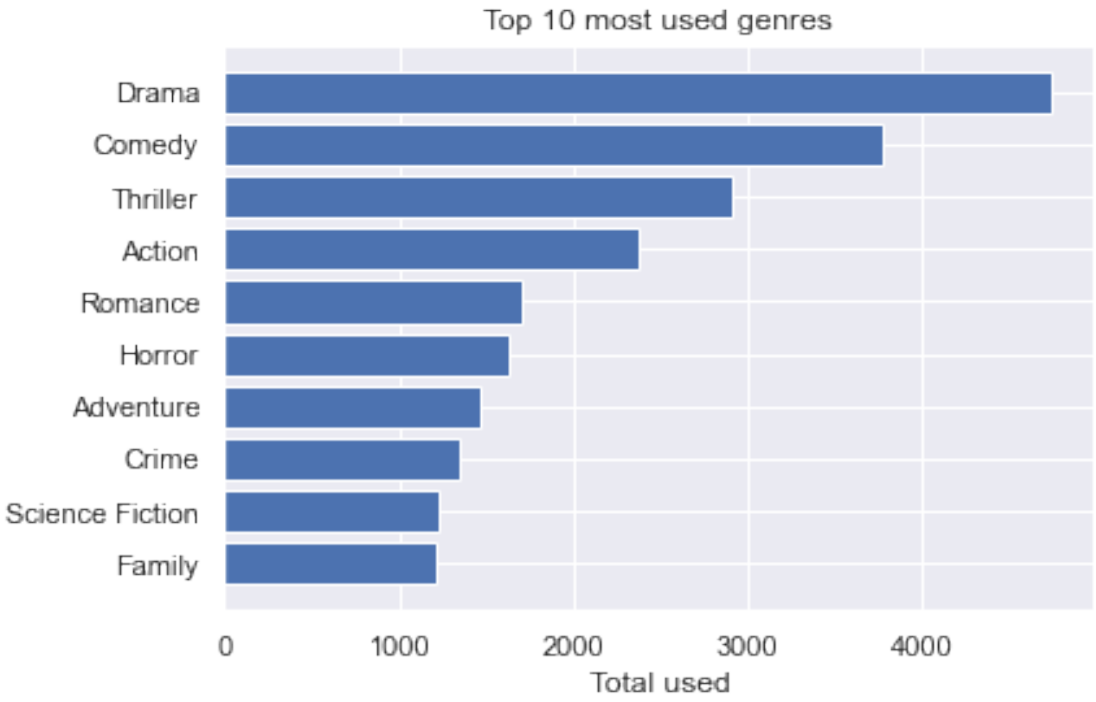
[37]:
```
res = df_genres_splitted.groupby(['genre'], as_index=False)['id'].count().
↪sort_values(by='id', ascending=False).head(10)
res = res.rename(columns={'id':'total_used'})
res
```

[37]:

	genre	total_used
6	Drama	4744
3	Comedy	3774
17	Thriller	2904
0	Action	2380
14	Romance	1705
11	Horror	1629
1	Adventure	1468
4	Crime	1354
15	Science Fiction	1227

```
7            Family        1217
```

[38]:
```
locations = list(range(10))
heights = res['total_used']

barh_chart_h(locations, heights, res['genre'], 'Top 10 most used genres',␣
 ↪'Total used')
```

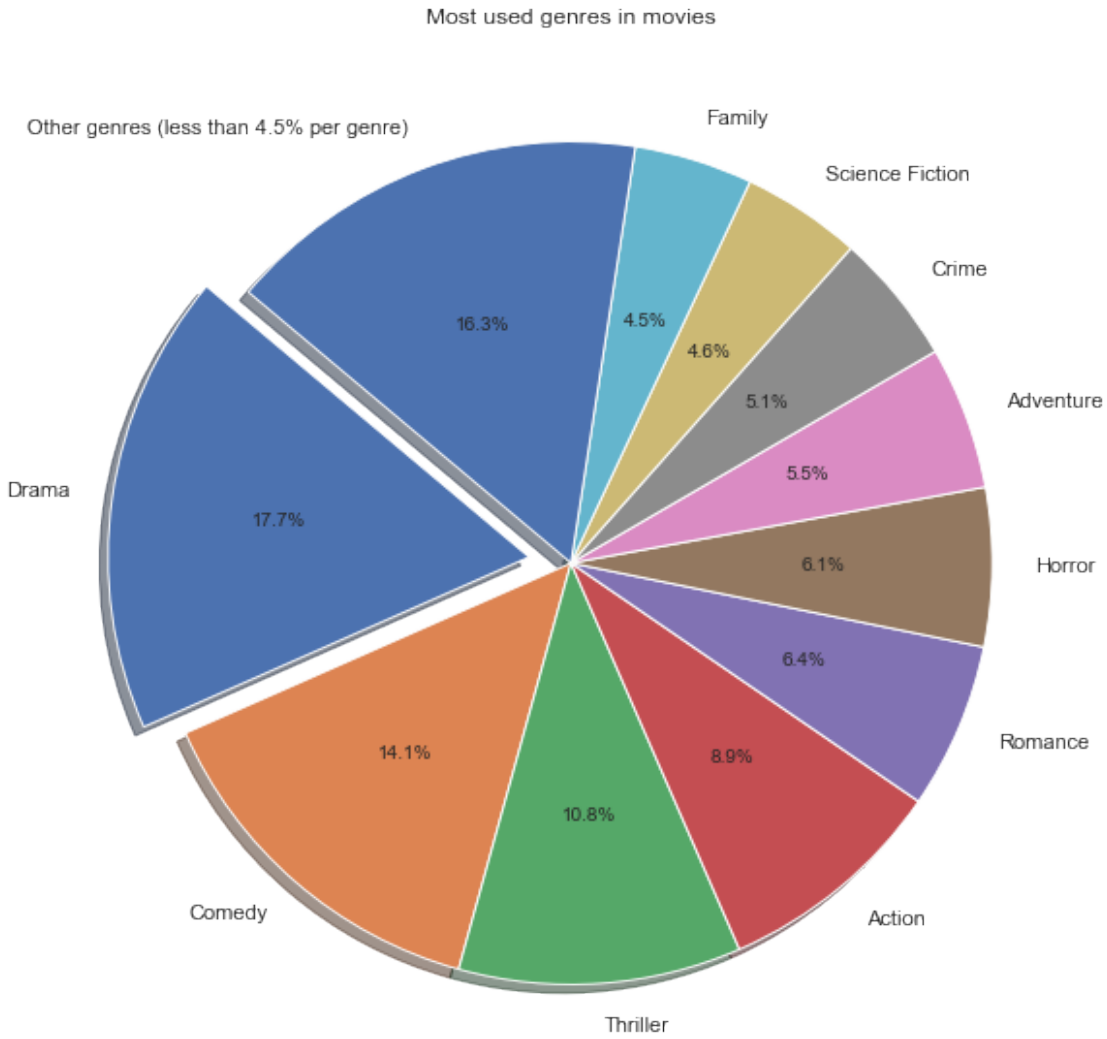


The genre the most used is Drama with 4744 uses, followed by Comedy with 3774 uses and Thriller with 2904 uses. The top 10 most used genres ends with Family with 1217 uses. Lets use a pie chart to gain more insights

[39]:
```
# compute the number of occurrences of the other genres
top_10_total_genre_occurence = res['total_used'].sum()
total_genre_occurence = df_genres_splitted.groupby(['genre'], as_index=False).
 ↪count()['id'].sum()
other_total_genre_occurence = total_genre_occurence -␣
 ↪top_10_total_genre_occurence

# pie
locations = list(range(11))
heights = res['total_used'].append(pd.Series([other_total_genre_occurence]))
labels = res['genre'].append(pd.Series(['Other genres (less than 4.5% per␣
 ↪genre)']))
```

```
explode = (0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

plt.figure(figsize=(20,10))
plt.pie(heights, labels=labels, explode=explode, shadow=True, startangle=140,␣
  ↪autopct='%1.1f%%');
plt.title('Most used genres in movies');
```

```
/var/folders/ly/5rsqm9ms11164z6svjbpk0340000gn/T/ipykernel_12711/1738503482.py:8
: FutureWarning: The series.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  heights = res['total_used'].append(pd.Series([other_total_genre_occurence]))
/var/folders/ly/5rsqm9ms11164z6svjbpk0340000gn/T/ipykernel_12711/1738503482.py:9
: FutureWarning: The series.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  labels = res['genre'].append(pd.Series(['Other genres (less than 4.5% per
genre)']))
```



From this pie chart it is seem that the most used genre is Drama with 17.7 % uses, followed by Comedy with 14.1 % uses and Thriller with 10.8% uses. The top 10 most used genres ends with Family with 4.5% uses.

1.6 Conclusions

In conclusion, we were able to get an overview of some of the top 10 movies with biggest budgets, revenue and so on. Thus, we could see that the most used genres in the films were Drama, Comedy and Thriller, that the films with the most income were Avatar, Star Wars and Titanic. Then, we showed that the actors who made the most films were Robert de Niro, Samuel L. Jackson and Bruce Willis. However, the data set may not be representative of the reality as it contains only a little bit more than 10,000 entries and it is not up to date.

[ ]: