Importing required Python libraries:

In [1]:

```
%matplotlib inline

from IPython.display import Image, HTML
import json
import datetime
import ast
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.dummy import DummyClassifier, DummyRegressor
from sklearn.model_selection import train_test_split
from wordcloud import WordCloud
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD
from sklearn import datasets, linear_model
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

sns.set_style('whitegrid')
sns.set(font_scale=1.25)
pd.set_option('display.max_colwidth', 50)
import warnings; warnings.simplefilter('ignore')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Apurva
[nltk_data]     Sarode\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Displaying the Dataset:

Once all the required libraries are imported, we then upload the dataset by reading Data from CSV file.

In [3]:

```
data = pd.read_csv('C:\\Users\\Apurva Sarode\\Desktop\\movie_stats\\movies_metadata.csv')
data.head().transpose()
```

Out[3]:

	0	1	2
adult	False	False	False
belongs_to_collection	{'id': 10194, 'name': 'Toy Story Collection', ...	NaN	{'id': 119050, 'name': 'Grumpy Old Men Collect...
budget	30000000	65000000	0
genres	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...

	0	1	2	
homepage	http://toystory.disney.com/toy-story	NaN	NaN	
id	862	8844	15602	
imdb_id	tt0114709	tt0113497	tt0113228	
original_language	en	en	en	
original_title	Toy Story	Jumanji	Grumpier Old Men	
overview	Led by Woody, Andy's toys live happily in his ...	When siblings Judy and Peter discover an encha...	A family wedding reignites the ancient feud be...	Chea
popularity	21.9469	17.0155	11.7129	
poster_path	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	/vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg	/16XC
production_companies	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[{
production_countries	[{'iso_3166_1': 'US', 'name': 'United States o...	[{'iso_3166_1': 'US', 'name': 'United States o...	[{'iso_3166_1': 'US', 'name': 'United States o...	
release_date	1995-10-30	1995-12-15	1995-12-22	
revenue	3.73554e+08	2.62797e+08	0	
runtime	81	104	101	
spoken_languages	[{'iso_639_1': 'en', 'name': 'English'}]	[{'iso_639_1': 'en', 'name': 'English'}, {'iso...	[{'iso_639_1': 'en', 'name': 'English'}]	
status	Released	Released	Released	
tagline	NaN	Roll the dice and unleash the excitement!	Still Yelling. Still Fighting. Still Ready for...	Fr
title	Toy Story	Jumanji	Grumpier Old Men	
video	False	False	False	
vote_average	7.7	6.9	6.5	
vote_count	5415	2413	92	

Understanding the Dataset

In [4]:

```
data.columns
```

Out[4]:

```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'video',
       'vote_average', 'vote_count'],
      dtype='object')
```

Features

- **adult:** The adult column indicates if the movie is X-Rated or Adult.
- **budget:** The budget of the movie in dollars.
- **genres:** A stringified list of dictionaries that list out all the genres associated with the movie.
- **id:** The ID of the move.
- **imdb_id:** The IMDB ID of the movie.
- **original_language:** The language in which the movie was originally shot in.
- **original_title:** The original title of the movie.
- **overview:** A brief blurb of the movie.
- **popularity:** The Popularity Score assigned by TMDB.
- **production_companies:** A stringified list of production companies involved with the making of the movie.
- **production_countries:** A stringified list of countries where the movie was shot/produced in.
- **release_date:** Theatrical Release Date of the movie.
- **revenue:** The total revenue of the movie in dollars.
- **runtime:** The runtime of the movie in minutes.
- **spoken_languages:** A stringified list of spoken languages in the film.

- **status:** The status of the movie (Released, To Be Released, Announced, etc.)
- **title:** The Official Title of the movie.
- **vote_average:** The average rating of the movie.
- **vote_count:** The number of votes by users, as counted by TMDB.

Using shape method we can view number of rows and columns our data has.

In [7]:

```
data.shape
```

Out[7]:

```
(45466, 24)
```

In [8]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
adult                    45466 non-null object
belongs_to_collection    4494 non-null object
budget                   45466 non-null object
genres                   45466 non-null object
homepage                 7782 non-null object
id                       45466 non-null object
imdb_id                  45449 non-null object
original_language        45455 non-null object
original_title           45466 non-null object
overview                 44512 non-null object
popularity               45461 non-null object
poster_path              45080 non-null object
production_companies     45463 non-null object
production_countries     45463 non-null object
release_date             45379 non-null object
revenue                  45460 non-null float64
runtime                  45203 non-null float64
spoken_languages         45460 non-null object
status                   45379 non-null object
tagline                  20412 non-null object
title                    45460 non-null object
video                    45460 non-null object
vote_average             45460 non-null float64
vote_count               45460 non-null float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

There are 45,466 films in total, with 24 scenes. Most functions have very few NaN values (except for homepage and tagline). In the next part, we'll try to clean this dataset to a type suitable for review.

Data Cleaning

1. What is data cleaning:

1) Data cleaning is the process of detecting and correcting or removing corrupt or inaccurate records from a record set which can be a table or database.

2) It refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty data.

3) After cleansing, a data set should be consistent with other similar data sets in the system.

4) The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities.

2. Why is Data Cleaning Important?

2. Why is Data Cleaning important?

Data cleaning step determines how easy your modelling is going to be. The better your data is, the less complex your learning algorithms need to be. Better structured data that provides the right input values will also determine the accuracy of your predictions. Data cleaning impacts efficiency of rest of your data modeling and decision-making process. This step is critical algorithm building step.

Efficiency gains of data cleaning results from:

1) Lesser processing time

2) More accurate predictions

3) Simpler algorithms needed

4) Better learning ability of the model

In [9]:

```
data = data.drop(['imdb_id'], axis=1)
```

In [10]:

```
data[data['original_title'] != data['title']][['title', 'original_title']].head()
```

Out[10]:

	title	original_title
28	The City of Lost Children	La Cité des Enfants Perdus
29	Shanghai Triad	摇啊摇，摇到外婆桥
32	Wings of Courage	Guillaumet, les ailes du courage
57	The Postman	Il postino
58	The Confessional	Le confessionnal

The original title refers to the film title in the native language in which the film was being made. As such, in this article, we would choose to use the translated, Anglicized name and will thus remove the original names entirely. By looking at the original language function, we would be able to deduce if the film is a foreign language film and no meaningful knowledge is lost in doing so.

In [11]:

```
data = data.drop('original_title', axis=1)
```

In [12]:

```
data[data['revenue'] == 0].shape
```

Out[12]:

```
(38052, 22)
```

We see most films have a reported revenue of **0**. This means we don't have the overall sales information for these movies. Although this forms the majority of the movies that are available to us, we can also use revenue as an incredibly important feature from the remaining 7000 moves.

In [13]:

```
data['revenue'] = data['revenue'].replace(0, np.nan)
```

The budget function has some unclean values that make it allocated to Pandas as a generic entity. We're turning this into a numeric attribute and replacing all non-numeric values with NaN. Finally, as with budget, we'll convert all the 0 values to NaN to show the lack of budget details.

In [15]:

```
data['budget'] = pd.to_numeric(data['budget'], errors='coerce')
data['budget'] = data['budget'].replace(0, np.nan)
data[data['budget'].isnull()].shape
```

Out[15]:

```
(36576, 22)
```

As we move forward trying to address some questions, we will need to create a variety of features appropriate for that specific query. We're going to create two very significant features for now:

year: the year the film was released in.

Return: The ratio of revenue to budget.

The return feature is particularly informative, because it gives us a more detailed image of a film's financial success.Now, our data won't be able to determine if a $200 million budget film that earned $100 million did better than a $50,000 budget film that took $200,000 in. This framework can capture the information. A return value > 1 indicates benefit while a return value < 1 indicates a loss.

In [16]:

```
data['return'] = data['revenue'] / data['budget']
data[data['return'].isnull()].shape
```

Out[16]:

```
(40085, 23)
```

We have nearly 5000 films for which we have the revenue and budget ratio info. That is almost 10 per cent of the entire dataset. Although this may seem tiny, this is adequate to conduct very valuable research and to uncover insightful insights about the world of films.

In [17]:

```
data['year'] = pd.to_datetime(data['release_date'], errors='coerce').apply(lambda x: str(x).split('
-')[0] if x != np.nan else np.nan)
```

In [18]:

```
data['adult'].value_counts()
```

Out[18]:

```
False
45454
True
9
 Avalanche Sharks tells the story of a bikini contest that turns into a horrifying affair when it
is hit by a shark avalanche.        1
 - Written by Ørnås
1
 Rune Balot goes to a casino connected to the October corporation to try to wrap up her case once
and for all.                        1
Name: adult, dtype: int64
```

This dataset includes close to 0 adult movies. Therefore the adult element isn't really useful for us and can be safely removed.

In [19]:

```
data = data.drop('adult', axis=1)
```

In [ ]:

```

```