

Overview

The tmdb-5000-movies dataset is a collection of movie information available on The Movie Database (TMDB). It is often used in machine learning and data analysis to explore patterns and trends in the film industry. It can also be used for recommendation systems.

Dataset Details

The original dataset was generated from The Movie Database API. This product uses the TMDB API but is not endorsed or certified by TMDB. The data was collected from TMDB 5000 Movie Dataset, a popular movie database website known for its wealth of information on movies, TV shows, and celebrities. The dataset contains two main tables:

movies_metadata.csv:

Contains general information about the movies, such as title, language, release date, budget, revenue, popularity, and average votes. Each row corresponds to a movie.

credits.csv:

Includes details about the cast and crew of each movie. Each row corresponds to a crew member (actor, director, writer, etc.) for a specific movie.

Dataset Name: tmdb-5000-movies

Language: English Total Size: 4,803 examples Contents The dataset consists of a data frame with 22 columns as follows:

id
budget
genres
homepage
keywords

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
In [2]: df = pd.read_csv("tmdb-movies.csv")
df.head()
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt Bryce Dallas Howard Irrfan Khan Vi...	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy Charlize Theron Hugh Keays-Byrne Nic...	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley Theo James Kate Winslet Ansel...	http://www
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford Mark Hamill Carrie Fisher Adam D...	htt
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel Paul Walker Jason Statham Michelle ...	

5 rows × 21 columns



```
In [4]: df.shape
```

Out[4]: (10866, 21)

```
In [5]: df.dtypes
```

```
Out[5]: id                int64
imdb_id                object
popularity            float64
budget                int64
revenue               int64
original_title        object
cast                  object
homepage              object
director              object
tagline               object
keywords              object
overview              object
runtime               int64
genres                object
production_companies  object
release_date          object
vote_count            int64
vote_average          float64
release_year          int64
budget_adj             float64
revenue_adj            float64
dtype: object
```

```
In [6]: 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
```

```
In [7]: df.nunique()
```

```
Out[7]: id                10865
imdb_id                10855
popularity            10814
budget                 557
revenue               4702
original_title        10571
cast                 10719
homepage              2896
director              5067
tagline               7997
keywords              8804
overview             10847
runtime               247
genres                2039
production_companies  7445
release_date          5909
vote_count            1289
vote_average           72
release_year           56
budget_adj            2614
revenue_adj           4840
dtype: int64
```

```
In [8]: df.describe()
```

```
Out[8]:
```

	id	popularity	budget	revenue	runtime	vote_count	vote_average
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	7.269280
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	1.053301
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	0.000000
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.500000
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.500000
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	7.500000
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	10.000000

```
In [9]: # MISSING VALUES CHECKS
df.isnull().sum()
```

```
Out[9]: 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
```

```
In [10]: df.isnull().sum().any()
```

```
Out[10]: True
```

```
In [11]: ## deleting the columns

df.drop(["id", "imdb_id", "homepage", "overview"], axis=1, inplace=True)
```

```
In [13]: df.columns
```

```
Out[13]: Index(['popularity', 'budget', 'revenue', 'original_title', 'cast', 'director',
               'tagline', 'keywords', 'runtime', 'genres', 'production_companies',
               'release_date', 'vote_count', 'vote_average', 'release_year',
               'budget_adj', 'revenue_adj'],
              dtype='object')
```

```
In [14]: len(df.columns)
```

```
Out[14]: 17
```

```
In [15]: df.shape
```

```
Out[15]: (10866, 17)
```

```
In [17]: # Check duplicate records

df.drop_duplicates(inplace=True)
```

```
In [18]: df.shape
```

```
Out[18]: (10865, 17)
```

```
In [19]: df.isnull().sum()
```

```
Out[19]: popularity          0
budget          0
revenue          0
original_title    0
cast             76
director         44
tagline          2824
keywords         1493
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
```

```
In [20]: ## Filling the blank records
df['cast'].fillna('missing',inplace=True)
df['director'].fillna('missing',inplace=True)
df['tagline'].fillna('missing',inplace=True)
df['keywords'].fillna('missing',inplace=True)
df['production_companies'].fillna('missing',inplace=True)
df['genres'].fillna('missing',inplace=True)
```

```
In [21]: df.isnull().sum()
```

```
Out[21]: popularity      0  
budget      0  
revenue      0  
original_title      0  
cast      0  
director      0  
tagline      0  
keywords      0  
runtime      0  
genres      0  
production_companies      0  
release_date      0  
vote_count      0  
vote_average      0  
release_year      0  
budget_adj      0  
revenue_adj      0  
dtype: int64
```

```
In [22]: df
```

Out[22]:

	popularity	budget	revenue	original_title	cast	director	t
0	32.985763	150000000	1513528810	Jurassic World	Chris Pratt Bryce Dallas Howard Irrfan Khan Vi...	Colin Trevorrow	The
1	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy Charlize Theron Hugh Keays-Byrne Nic...	George Miller	\ Love
2	13.112507	110000000	295238201	Insurgent	Shailene Woodley Theo James Kate Winslet Ansel...	Robert Schwentke	One (Destr
3	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford Mark Hamill Carrie Fisher Adam D...	J.J. Abrams	genr has ε
4	9.335014	190000000	1506249360	Furious 7	Vin Diesel Paul Walker Jason Statham Michelle ...	James Wan	Veng Hits
...	
10861	0.080598	0	0	The Endless Summer	Michael Hynson Robert August Lord 'Tally Ho' B...	Bruce Brown	n
10862	0.065543	0	0	Grand Prix	James Garner Eva Marie Saint Yves Montand Tosh...	John Frankenheimer	Cin s YOL dri spei
10863	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy Oleg Efremov Georgi Z...	Eldar Ryazanov	n
10864	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi Akiko Wakabayashi Mie Hama Joh...	Woody Allen	W / ST
10865	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren Tom Neyman John Reynolds Dian...	Harold P. Warren	Shc It's E Imagi

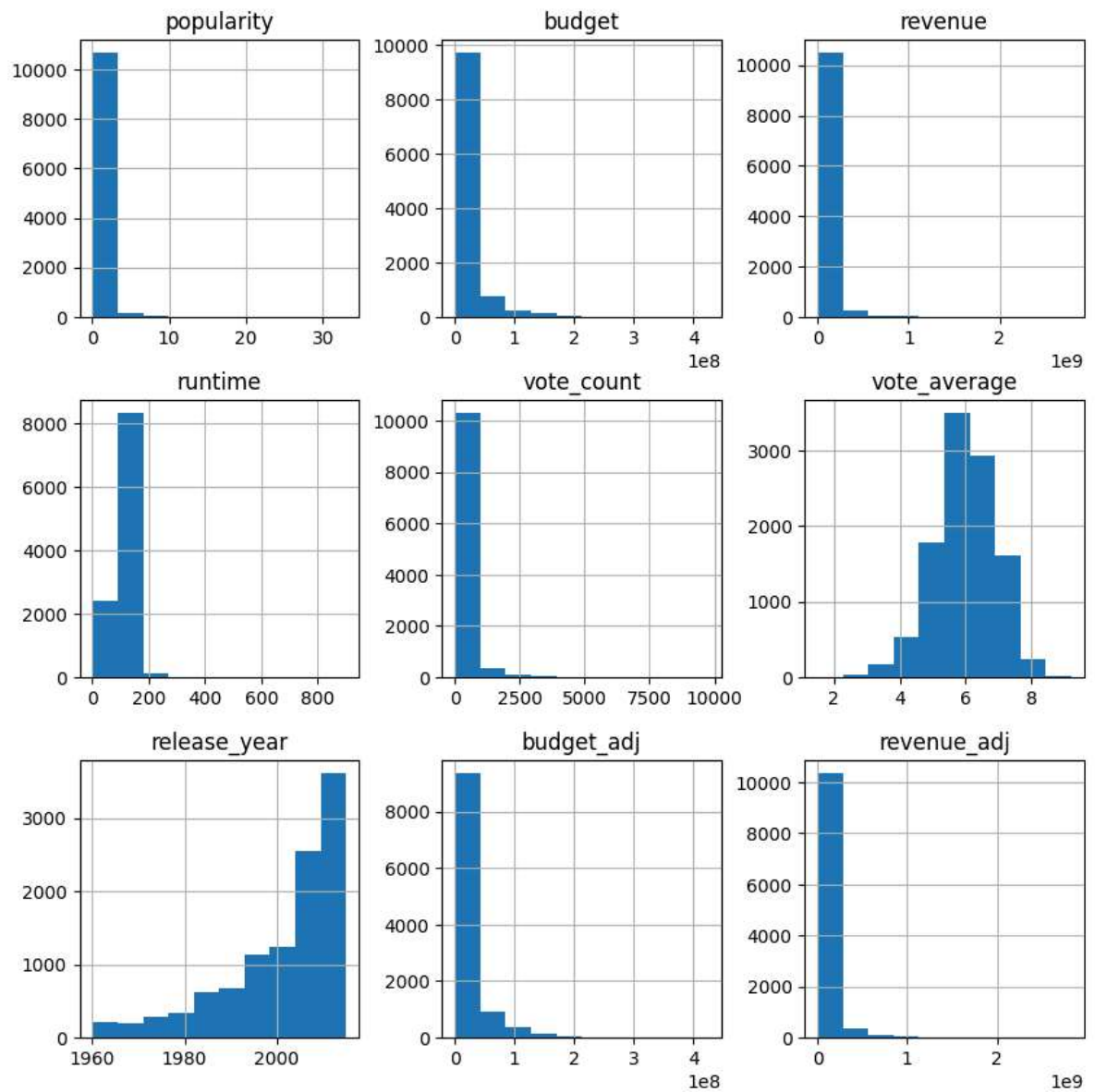
10865 rows × 17 columns


```
In [26]: # Check duplicate records
```

```
df.drop_duplicates().any()
```

```
Out[26]: popularity      True
budget      True
revenue      True
original_title      True
cast      True
director      True
tagline      True
keywords      True
runtime      True
genres      True
production_companies      True
release_date      True
vote_count      True
vote_average      True
release_year      True
budget_adj      True
revenue_adj      True
dtype: bool
```

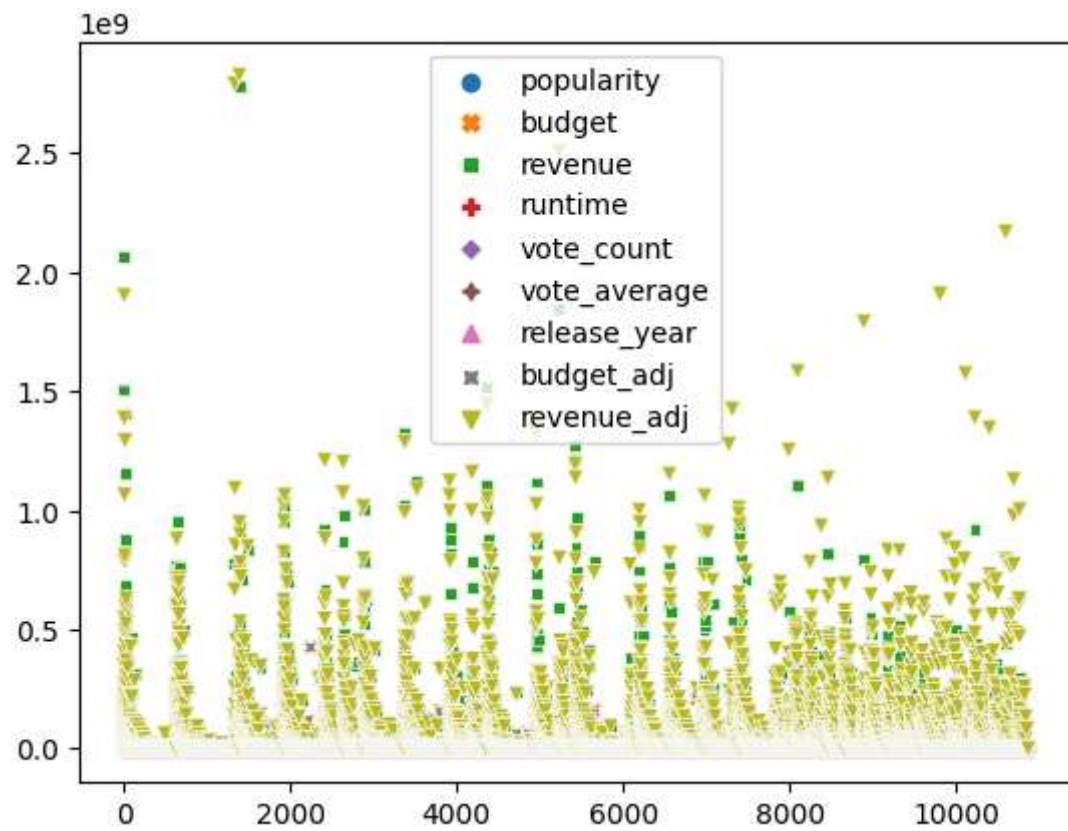
```
In [28]: ## Data variation  
df.hist(figsize=(10,10))  
plt.show()
```



```
In [29]: df.to_csv("tmdb_new.csv")
```

```
In [31]: sns.scatterplot(df)
```

```
Out[31]: <Axes: >
```



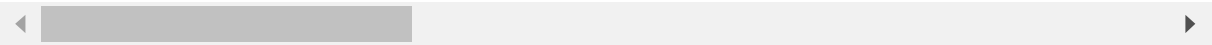
```
In [ ]: ## Q1 Does the higher budget means higher polularities? what is the relation?
```

```
In [35]: df.query('budget > 10000')
```

Out[35]:

	popularity	budget	revenue	original_title	cast	director	
0	32.985763	150000000	1513528810	Jurassic World	Chris Pratt Bryce Dallas Howard Irrfan Khan Vi...	Colin Trevorrow	The
1	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy Charlize Theron Hugh Keays-Byrne Nic...	George Miller	What ε
2	13.112507	110000000	295238201	Insurgent	Shailene Woodley Theo James Kate Winslet Ansel...	Robert Schwentke	One Can
3	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford Mark Hamill Carrie Fisher Adam D...	J.J. Abrams	genera
4	9.335014	190000000	1506249360	Furious 7	Vin Diesel Paul Walker Jason Statham Michelle ...	James Wan	Vengear
...	
10835	0.299911	12000000	20000000	The Sand Pebbles	Steve McQueen Richard Attenborough Richard Cre...	Robert Wise	Th heroic the mei
10841	0.264925	75000	0	The Shooting	Will Hutchins Millie Perkins Jack Nicholson Wa...	Monte Hellman	Susp desert p the "Higl
10848	0.207257	5115000	12000000	Fantastic Voyage	Stephen Boyd Raqueel Welch Edmond O'Brien Donal...	Richard Fleischer	A Fanta Spe V Thr
10855	0.141026	700000	0	The Ghost & Mr. Chicken	Don Knotts Joan Staley Liam Redmond Dick Sarge...	Alan Rafkin	GUARAI YOI S UNT
10865	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren Tom Neyman John Reynolds Dian...	Harold P. Warren	It's St It's Beyo Imag

5099 rows × 17 columns



```
In [36]: m = df['budget'].mean()  
m
```

```
Out[36]: 14624286.06433502
```

```
In [37]: low_bdg= df.query('budget < {}'.format(m))
high_bdg= df.query('budget >= {}'.format(m))
# print(low_bdg)
# print(high_bdg)
```

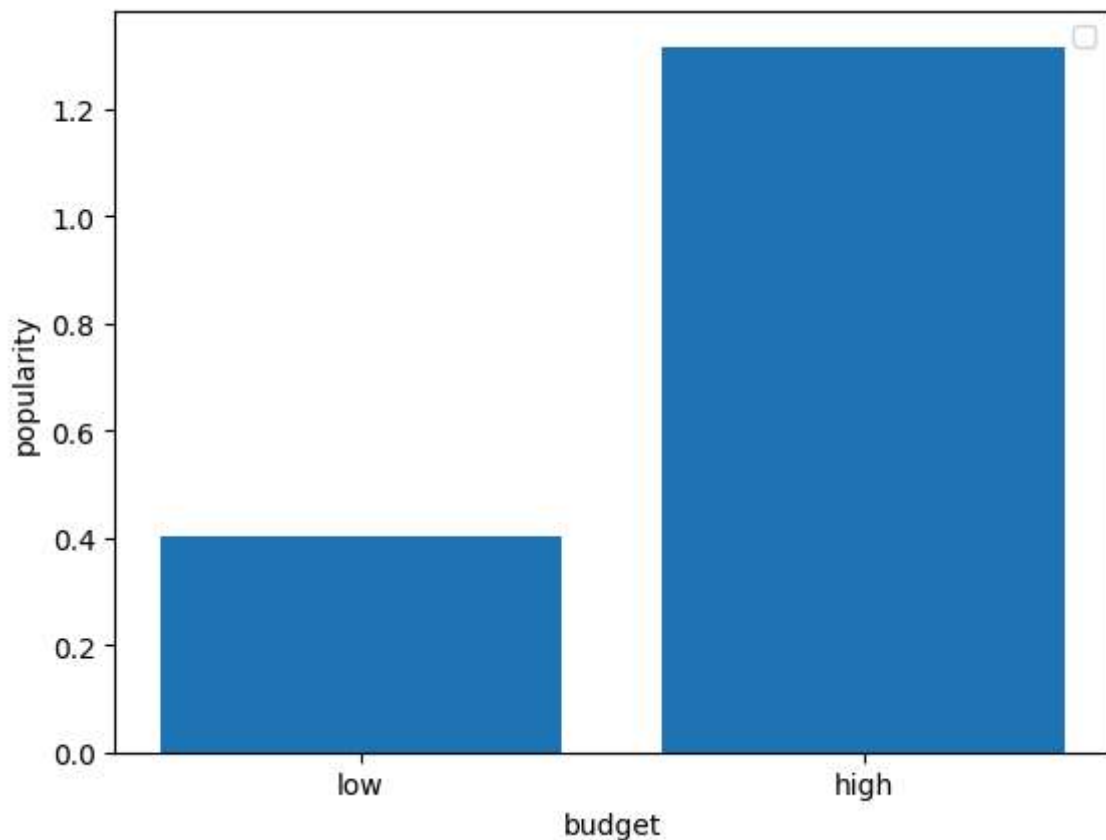
	popularity	budget	revenue	original_title \
30	3.927333	0	29355203	Mr. Holmes
35	3.557846	6000000	35401758	Room
36	3.358321	0	22354572	Solace
39	3.227329	11000000	62076141	Brooklyn
45	3.023253	10000000	14333790	Hardcore Henry
...
10861	0.080598	0	0	The Endless Summer
10862	0.065543	0	0	Grand Prix
10863	0.065141	0	0	Beregis Avtomobilya
10864	0.064317	0	0	What's Up, Tiger Lily?
10865	0.035919	19000	0	Manos: The Hands of Fate

	cast	direct
or \		
30	Ian McKellen Milo Parker Laura Linney Hattie M...	Bill Cond
on		
35	Brie Larson Jacob Tremblay Joan Allen Sean Bri...	Lenny Abrahams
on		
36

```
In [40]: m_popularity_low = low_bdg["popularity"].mean()
m_popularity_high = high_bdg["popularity"].mean()

labels = ["low", "high"]
location = [1, 2]
values = [m_popularity_low, m_popularity_high]
plt.bar(location, values, tick_label = labels)
plt.legend()
plt.xlabel("budget")
plt.ylabel("popularity")
plt.show()
```

No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



```
In [42]: p = (m_popularity_high - m_popularity_low) / m_popularity_high * 100
p
```

Out[42]: 69.2897746748735

In []:

