In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.stats.weightstats import zconfint
from statsmodels.stats.weightstats import DescrStatsW
from statsmodels.stats.weightstats import ztest
from scipy.stats import ttest_ind
from scipy.stats import normaltest
from scipy.stats import ranksums
```

In [2]:

```
tmdb = pd.read_csv('tmdb_5000_movies.csv')
```

In [3]:

```
tmdb.head()
```

Out[3]:

	budget	genres	homepage	id	keywords	original_language	orig
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pira Cari W
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	Kn
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	Jc

In [4]:

```
tmdb.describe()
```

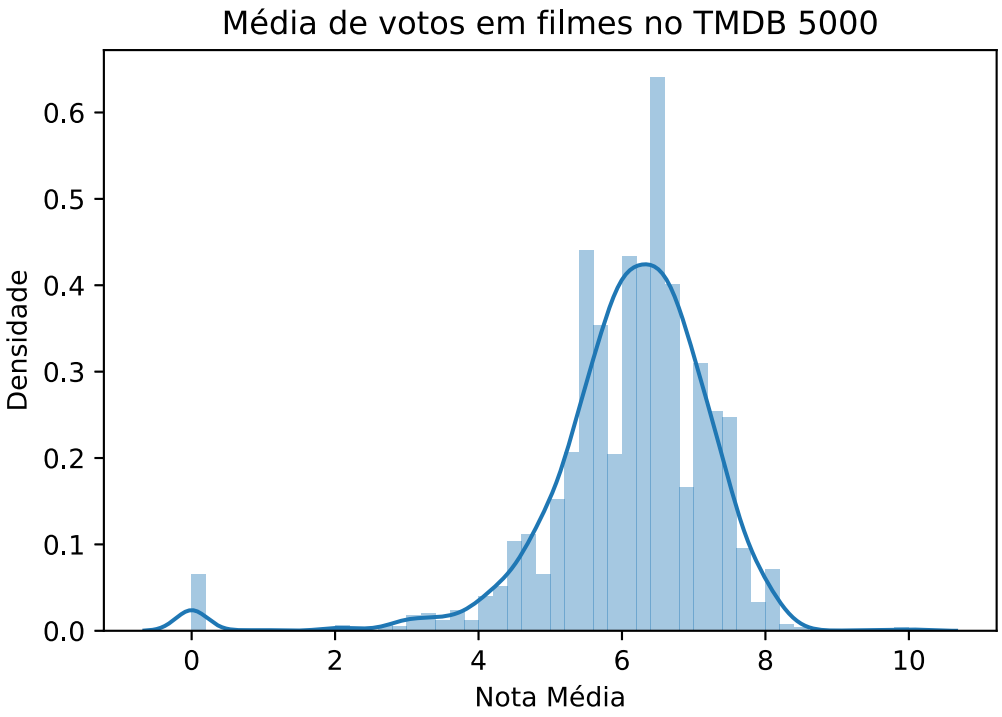
Out[4]:

	budget	id	popularity	revenue	runtime	vote_average	vote_count
count	4.803000e+03	4803.000000	4803.000000	4.803000e+03	4801.000000	4803.000000	4803.000000
mean	2.904504e+07	57165.484281	21.492301	8.226064e+07	106.875859	6.092172	690.217989
std	4.072239e+07	88694.614033	31.816650	1.628571e+08	22.611935	1.194612	1234.585891

	budget	id	popularity	revenue	runtime	vote_average	vote_count
min	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	7.900000e+05	9014.500000	4.668070	0.000000e+00	94.000000	5.600000	54.000000
50%	1.500000e+07	14629.000000	12.921594	1.917000e+07	103.000000	6.200000	235.000000
75%	4.000000e+07	58610.500000	28.313505	9.291719e+07	118.000000	6.800000	737.000000
max	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	13752.000000

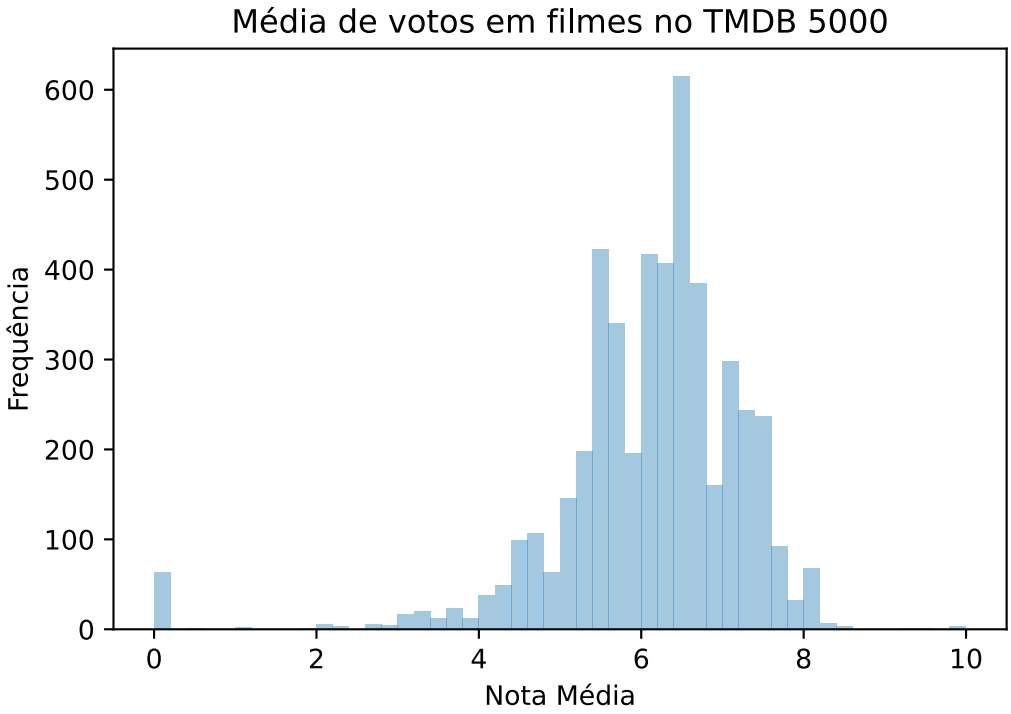
In [5]:
```
ax = sns.distplot(tmdb['vote_average'])
ax.set(xlabel='Nota Média', ylabel='Densidade')
ax.set_title('Média de votos em filmes no TMDB 5000')
```

Out[5]:
```
Text(0.5, 1.0, 'Média de votos em filmes no TMDB 5000')
```



In [6]:
```
ax = sns.distplot(tmdb['vote_average'], norm_hist=False, kde=False)
ax.set(xlabel='Nota Média', ylabel='Frequência')
ax.set_title('Média de votos em filmes no TMDB 5000')
```

Out[6]:
```
Text(0.5, 1.0, 'Média de votos em filmes no TMDB 5000')
```

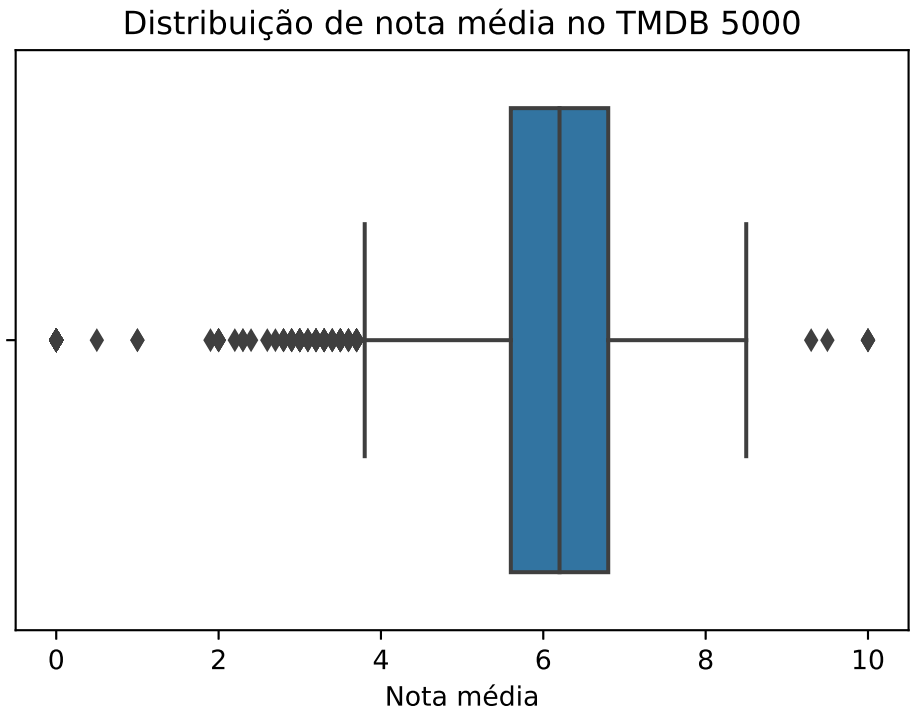


In [7]:

```
ax = sns.boxplot(tmdb['vote_average'])
ax.set_title('Distribuição de nota média no TMDB 5000')
ax.set(xlabel='Nota média')
```

Out[7]:

```
[Text(0.5, 0, 'Nota média')]
```



Tem algo de estranho com os dados. Não faz sentido filmes com nota média 0 ou 10.

In [8]:

```
tmdb.query('vote_average == 0')
tmdb.head()
```

Out[8]:

	budget	genres	homepage	id	keywords	original_language	orig
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	

	budget	genres	homepage	id	keywords	original_language	orig
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pira Cari W
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	Kn
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	Jc

In [9]:
```
tmdb.query('vote_average == 10')
```

Out[9]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popu
3519	0	[{"id": 35, "name": "Comedy"}]	NaN	89861	[{"id": 131, "name": "italy"}, {"id": 8250, "n...	en	Stiff Upper Lips	Stiff Upper Lips is a broad parody of British ...	0.3
4045	0	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	NaN	78373	[{"id": 1415, "name": "small town"}, {"id": 15...	en	Dancer, Texas Pop. 81	Four guys, best friends, have grown up togethe...	0.3
4247	1	[{"id": 10749, "name": "Romance"}, {"id": 35, ...	NaN	361505	[]	en	Me You and Five Bucks	A womanizing yet lovable loser, Charlie, a wai...	0.0
4662	0	[{"id": 35, "name": "Comedy"}]	NaN	40963	[{"id": 10183, "name": "independent film"}]	en	Little Big Top	An aging out of work clown returns to his smal...	0.0

Detectamos que alguns filmes tiveram poucos ou nenhum votos.

Esses filmes não devem fazer parte da análise.

In [10]:
```
tmdb_com_mais_de_10_votos = tmdb.query('vote_count >= 10')
```

```
tmdb_com_mais_de_10_votos.describe()
```

Out[10]:

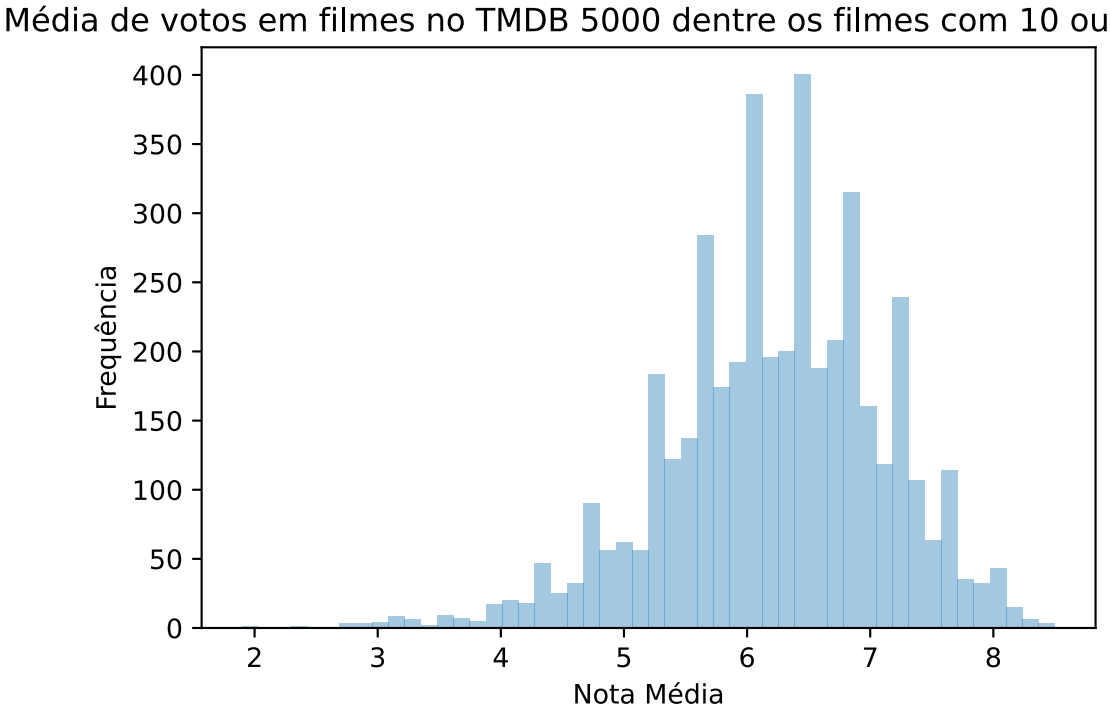
	budget	id	popularity	revenue	runtime	vote_average	vote_count
count	4.392000e+03	4392.000000	4392.000000	4.392000e+03	4391.000000	4392.000000	4392.000000
mean	3.164545e+07	49204.119991	23.448815	8.990969e+07	108.430881	6.226935	754.441712
std	4.162736e+07	80136.249777	32.592158	1.682870e+08	21.014719	0.893215	1272.263761
min	0.000000e+00	5.000000	0.011697	0.000000e+00	0.000000	1.900000	10.000000
25%	2.895962e+06	8403.500000	6.512166	1.365700e+04	95.000000	5.700000	83.750000
50%	1.700000e+07	13084.500000	14.827784	2.685837e+07	105.000000	6.300000	288.500000
75%	4.200000e+07	46831.250000	30.258282	1.022818e+08	118.500000	6.800000	831.000000
max	3.800000e+08	417859.000000	875.581305	2.787965e+09	338.000000	8.500000	13752.000000

In [11]:

```
ax = sns.distplot(tmdb_com_mais_de_10_votos['vote_average'], norm_hist=False, kde=False)
ax.set(xlabel='Nota Média', ylabel='Frequência')
ax.set_title('Média de votos em filmes no TMDB 5000 dentre os filmes com 10 ou mais votos')
```

Out[11]:

```
Text(0.5, 1.0, 'Média de votos em filmes no TMDB 5000 dentre os filmes com 10 ou mais votos')
```

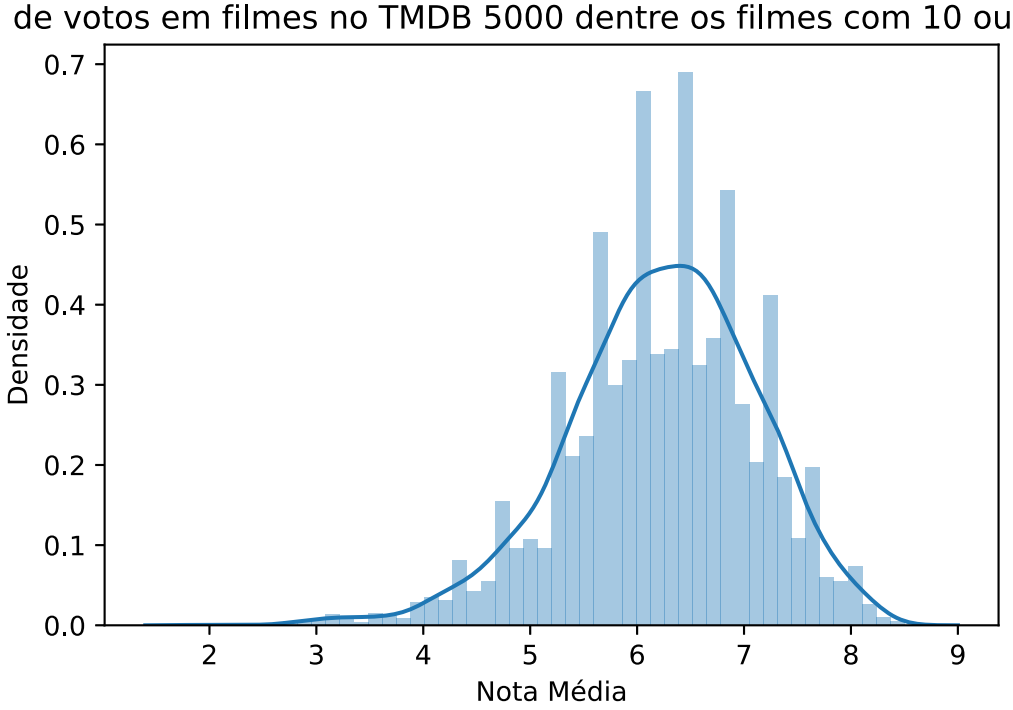


In [12]:

```
ax = sns.distplot(tmdb_com_mais_de_10_votos['vote_average'])
ax.set(xlabel='Nota Média', ylabel='Densidade')
ax.set_title('Média de votos em filmes no TMDB 5000 dentre os filmes com 10 ou mais votos')
```

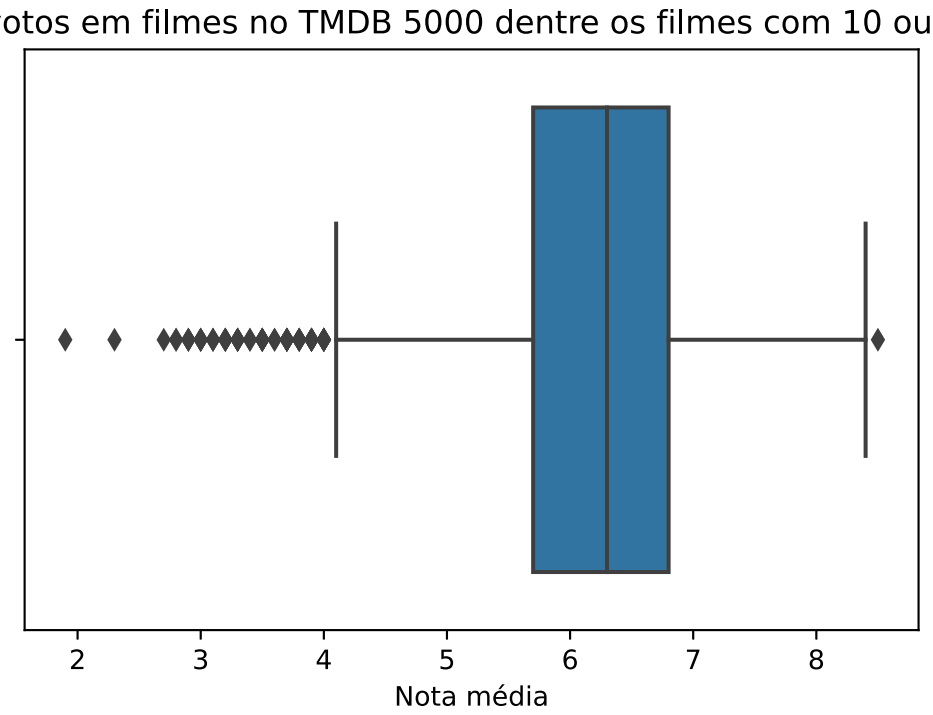
Out[12]:

```
Text(0.5, 1.0, 'Média de votos em filmes no TMDB 5000 dentre os filmes com 10 ou mais votos')
```



In [13]:
```
ax = sns.boxplot(tmdb_com_mais_de_10_votos['vote_average'])
ax.set_title('Média de votos em filmes no TMDB 5000 dentre os filmes com 10 ou mais votos')
ax.set(xlabel='Nota média')
```

Out[13]:
```
[Text(0.5, 0, 'Nota média')]
```



Analizaremos também o MovieLens

In [14]:
```
notas = pd.read_csv('ratings.csv')
notas.head()
```

Out[14]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224

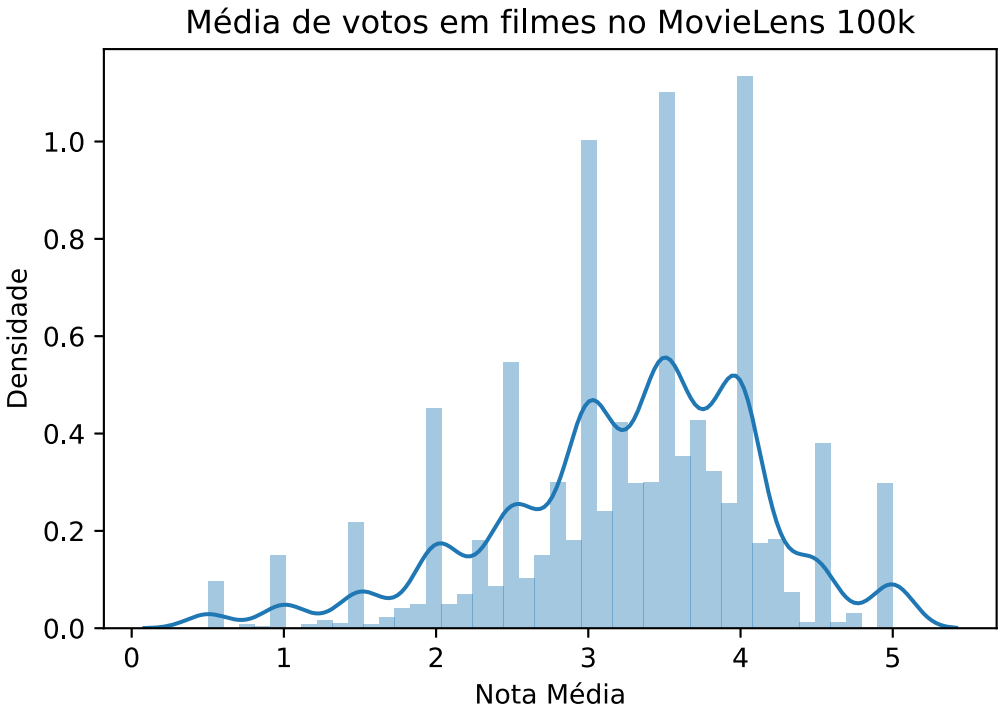
	userId	movieId	rating	timestamp
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [15]:
```
nota_media_por_filme = notas.groupby('movieId')['rating'].mean()
nota_media_por_filme.head()
```

Out[15]:
```
movieId
1    3.920930
2    3.431818
3    3.259615
4    2.357143
5    3.071429
Name: rating, dtype: float64
```

In [16]:
```
ax = sns.distplot(nota_media_por_filme.values)
ax.set(xlabel='Nota Média', ylabel='Densidade')
ax.set_title('Média de votos em filmes no MovieLens 100k')
```

Out[16]:
```
Text(0.5, 1.0, 'Média de votos em filmes no MovieLens 100k')
```



In [17]:
```
quantidade_de_votos_por_filme = notas.groupby('movieId').count()
filmes_com_pelo_menos_10_votos = quantidade_de_votos_por_filme.query('userId >= 10').index
filmes_com_pelo_menos_10_votos.values
```

Out[17]:
```
array([     1,      2,      3, ..., 177765, 179819, 187593], dtype=int64)
```

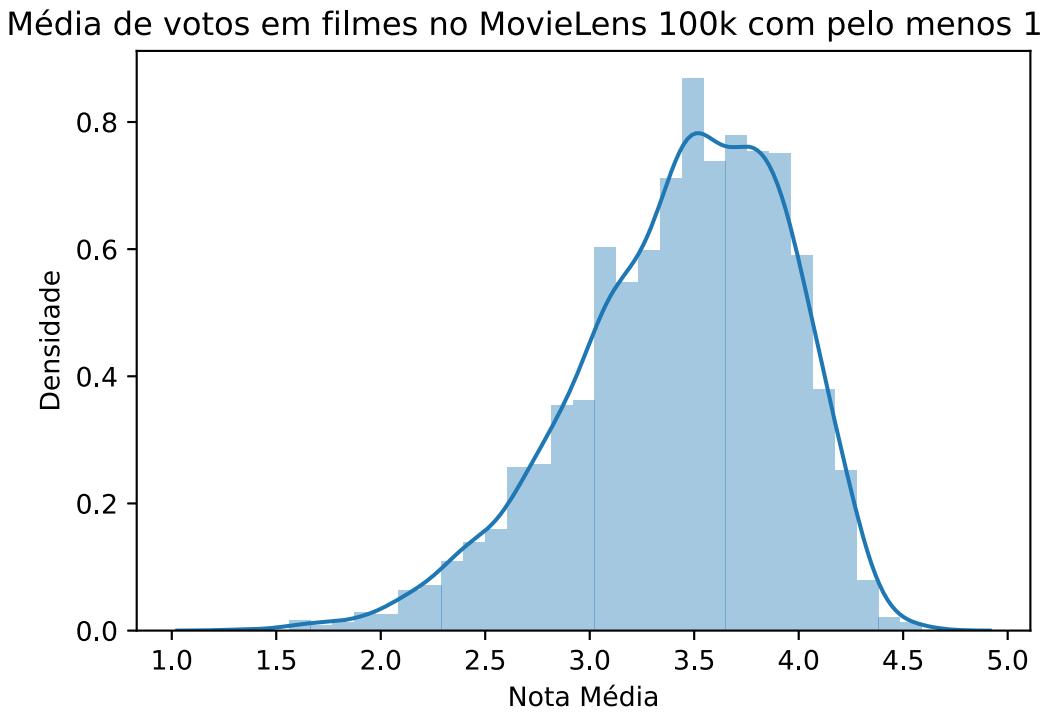
In [18]:
```
nota_media_dos_filmes_com_pelo_menos_10_votos = nota_media_por_filme.loc[filmes_com_pelo_menos_
nota_media_dos_filmes_com_pelo_menos_10_votos.head()
```

Out[18]:
```
movieId
1    3.920930
2    3.431818
3    3.259615
5    3.071429
6    3.946078
Name: rating, dtype: float64
```

In [19]:
```
ax = sns.distplot(nota_media_dos_filmes_com_pelo_menos_10_votos.values)
```

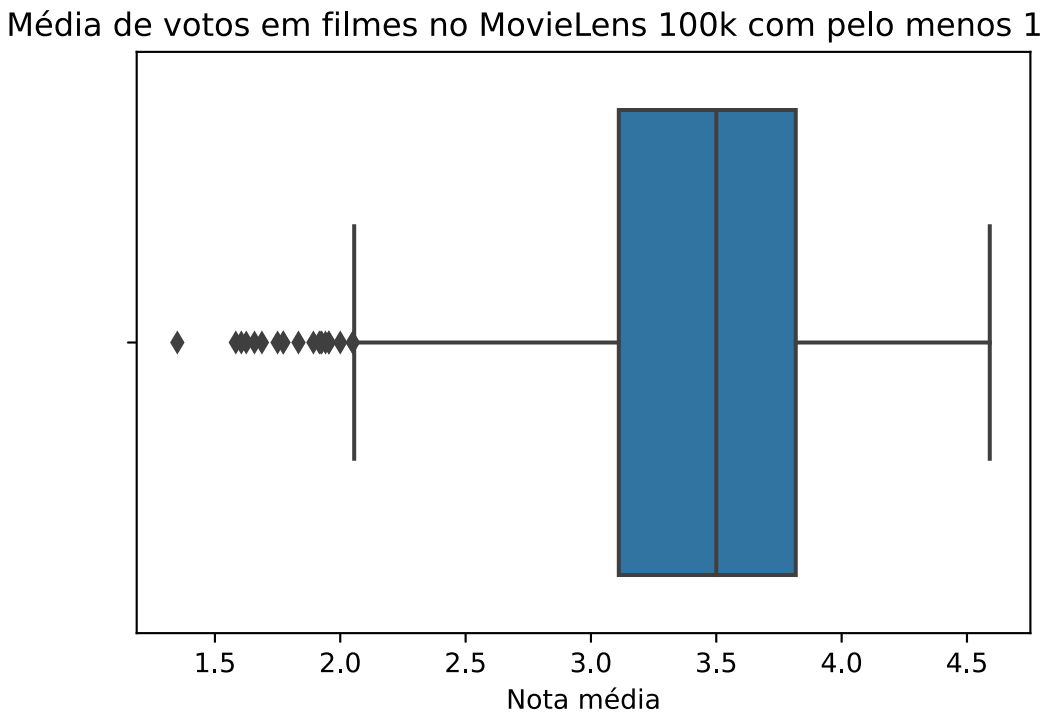
```
ax.set(xlabel='Nota Média', ylabel='Densidade')
ax.set_title('Média de votos em filmes no MovieLens 100k com pelo menos 10 votos')
```

Out[19]:
```
Text(0.5, 1.0, 'Média de votos em filmes no MovieLens 100k com pelo menos 10 votos')
```



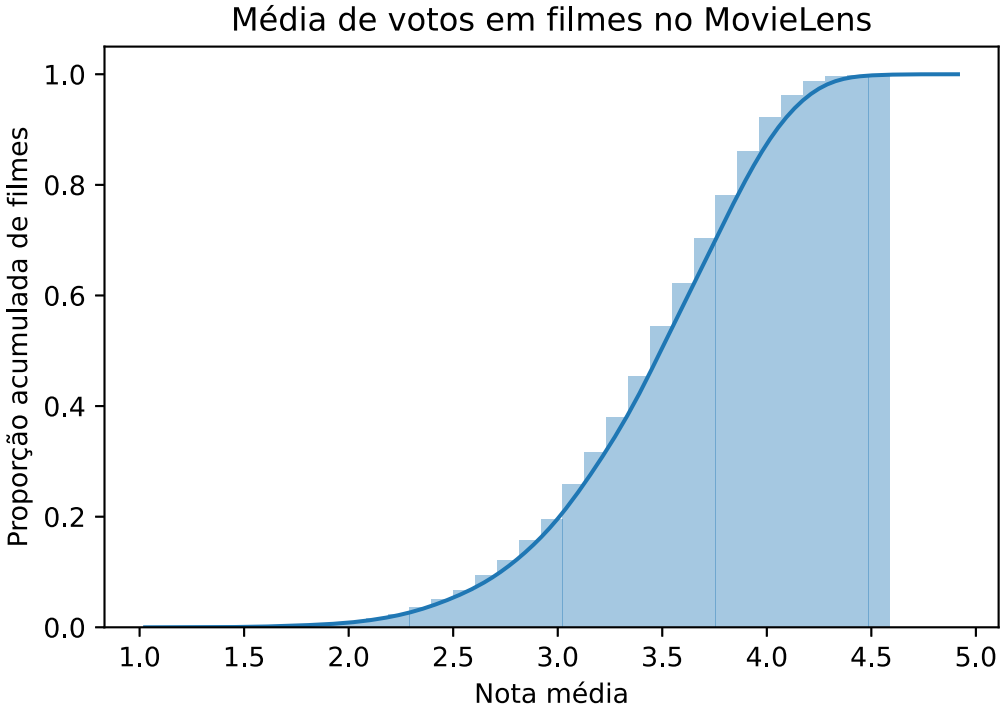
In [20]:
```
ax = sns.boxplot(nota_media_dos_filmes_com_pelo_menos_10_votos.values)
ax.set_title('Média de votos em filmes no MovieLens 100k com pelo menos 10 votos')
ax.set(xlabel='Nota média')
```

Out[20]:
```
[Text(0.5, 0, 'Nota média')]
```



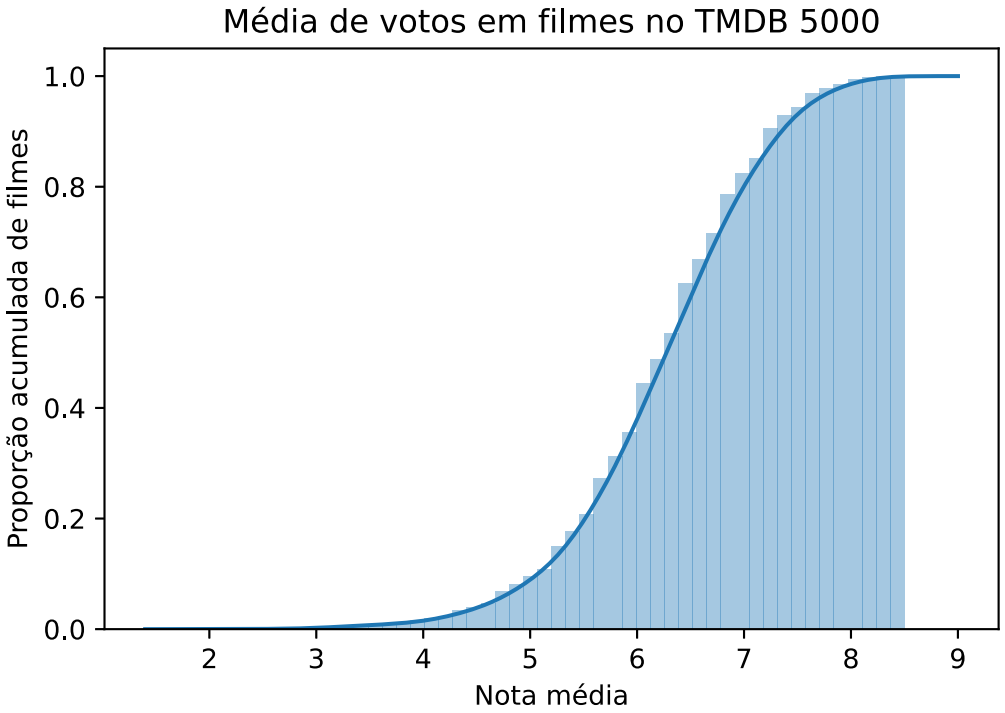
In [21]:
```
ax = sns.distplot(nota_media_dos_filmes_com_pelo_menos_10_votos,
                  hist_kws = {'cumulative':True},
                  kde_kws = {'cumulative':True})
ax.set(xlabel='Nota média', ylabel='Proporção acumulada de filmes')
ax.set_title('Média de votos em filmes no MovieLens')
```

Out[21]:
```
Text(0.5, 1.0, 'Média de votos em filmes no MovieLens')
```



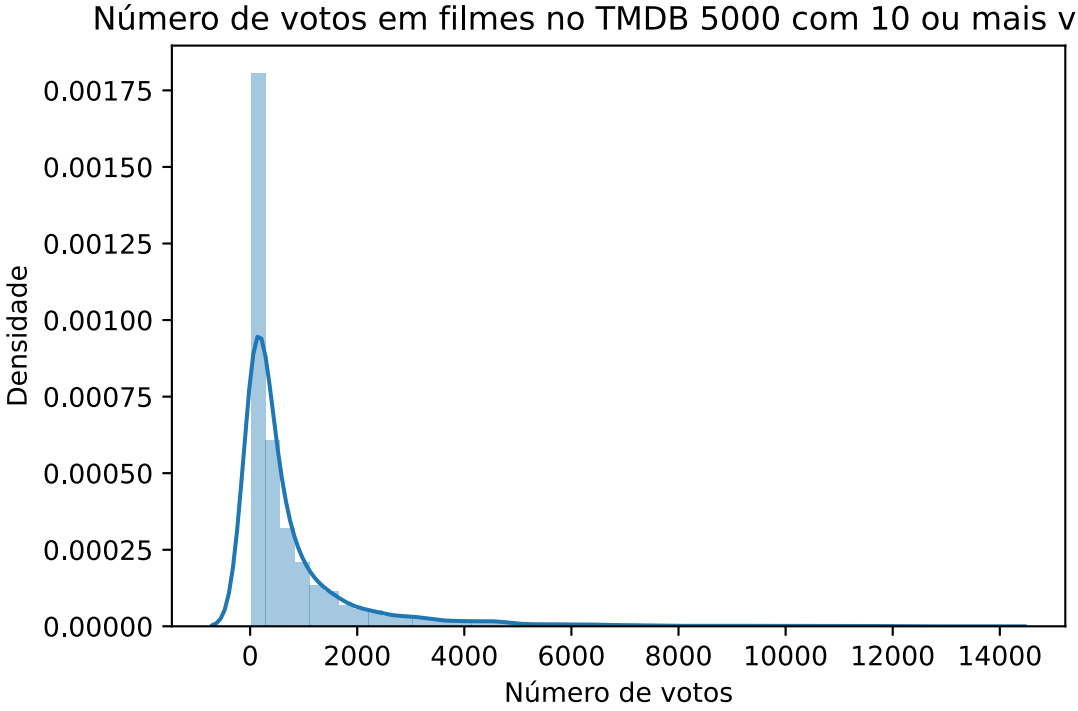
In [22]:
```
ax = sns.distplot(tmdb_com_mais_de_10_votos.vote_average,
                  hist_kws = {'cumulative':True},
                  kde_kws = {'cumulative':True})
ax.set(xlabel='Nota média', ylabel='Proporção acumulada de filmes')
ax.set_title('Média de votos em filmes no TMDB 5000')
```

Out[22]:
```
Text(0.5, 1.0, 'Média de votos em filmes no TMDB 5000')
```



In [23]:
```
ax = sns.distplot(tmdb_com_mais_de_10_votos['vote_count'])
ax.set(xlabel='Número de votos', ylabel='Densidade')
ax.set_title('Número de votos em filmes no TMDB 5000 com 10 ou mais votos')
```

Out[23]:
```
Text(0.5, 1.0, 'Número de votos em filmes no TMDB 5000 com 10 ou mais votos')
```

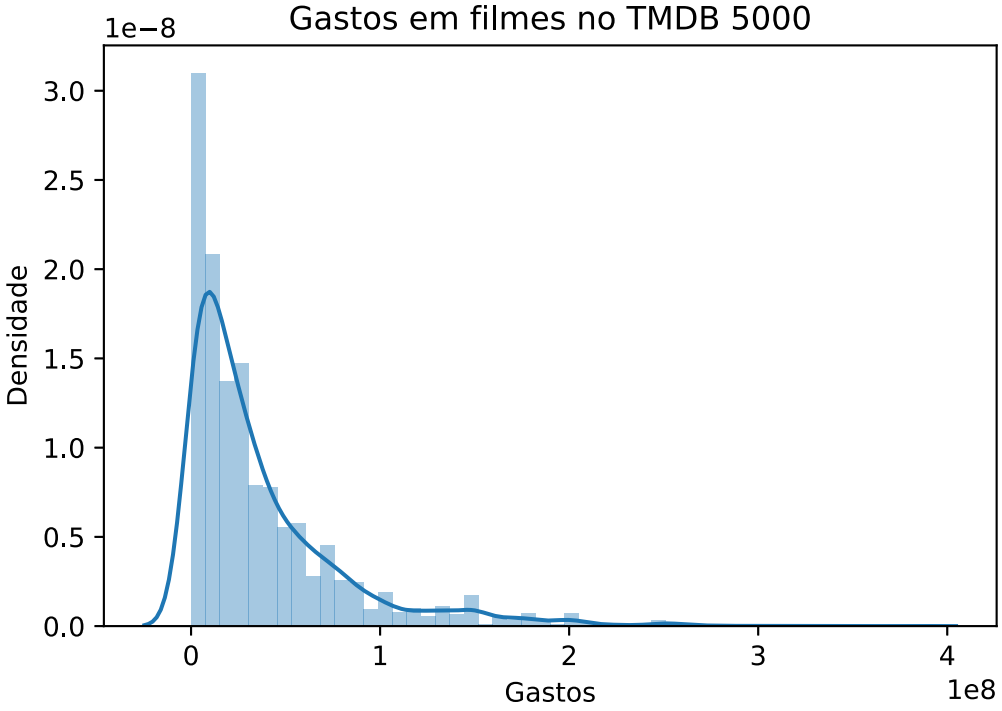


In [24]:

```
ax = sns.distplot(tmdb.query("budget > 0")['budget'])
ax.set(xlabel='Gastos', ylabel='Densidade')
ax.set_title('Gastos em filmes no TMDB 5000')
```

Out[24]:

```
Text(0.5, 1.0, 'Gastos em filmes no TMDB 5000')
```

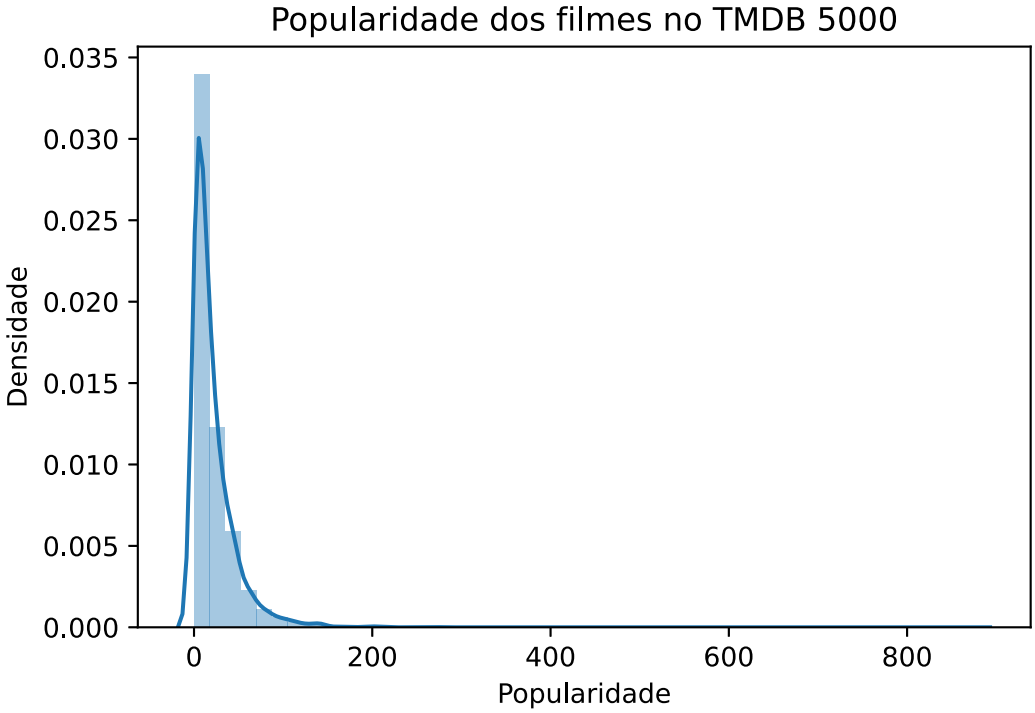


In [25]:

```
ax = sns.distplot(tmdb['popularity'])
ax.set(xlabel='Popularidade', ylabel='Densidade')
ax.set_title('Popularidade dos filmes no TMDB 5000')
```

Out[25]:

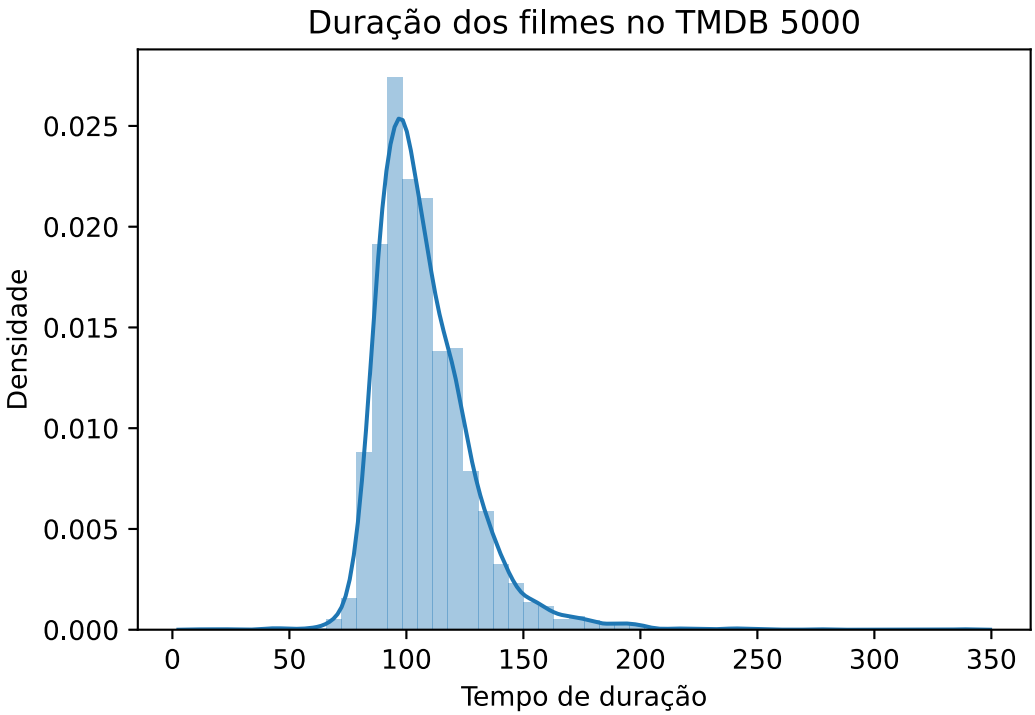
```
Text(0.5, 1.0, 'Popularidade dos filmes no TMDB 5000')
```



In [26]:

```
ax = sns.distplot(tmdb.query("runtime > 0")['runtime'].dropna())
ax.set(xlabel='Tempo de duração', ylabel='Densidade')
ax.set_title('Duração dos filmes no TMDB 5000')
```

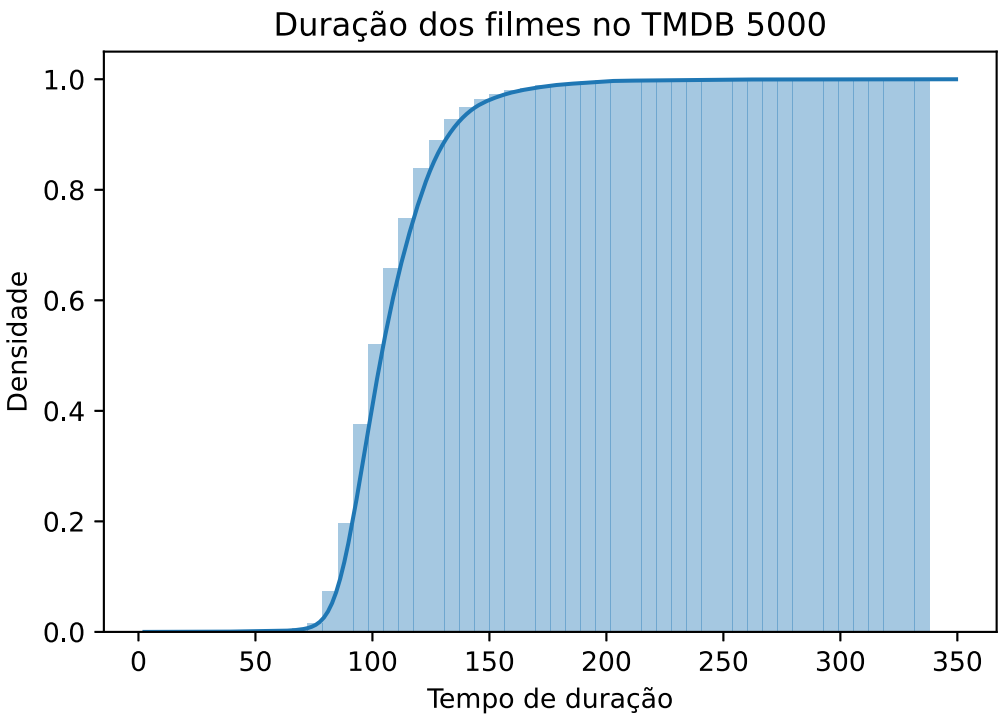
Out[26]: Text(0.5, 1.0, 'Duração dos filmes no TMDB 5000')



In [27]:

```
ax = sns.distplot(tmdb.query("runtime > 0")['runtime'].dropna(),
                  hist_kws={'cumulative':True},
                  kde_kws={'cumulative':True})
ax.set(xlabel='Tempo de duração', ylabel='Densidade')
ax.set_title('Duração dos filmes no TMDB 5000')
```

Out[27]: Text(0.5, 1.0, 'Duração dos filmes no TMDB 5000')



In [28]:
```
tmdb.query("runtime>0").runtime.dropna().quantile(q=0.8)
```

Out[28]:
```
121.0
```

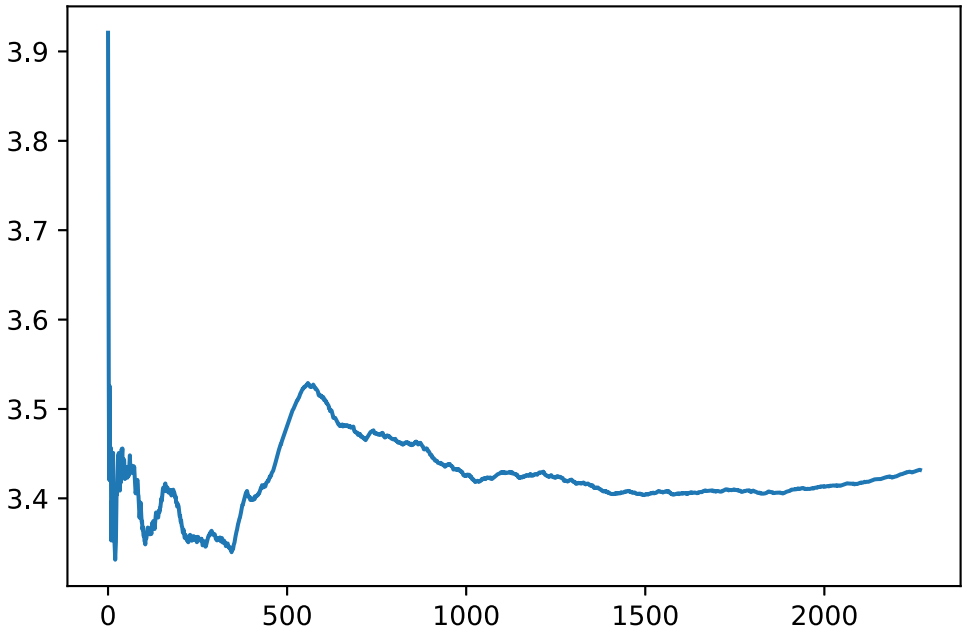
MovieLens: média dos filmes com pelo menos 10 votos

In [29]:
```
print("Média dos filmes com pelo menos 10 votos", nota_media_dos_filmes_com_pelo_menos_10_votos
```

```
Média dos filmes com pelo menos 10 votos 3.4320503405352594
```

In [30]:
```
medias = [nota_media_dos_filmes_com_pelo_menos_10_votos[0:i].mean() for i in range(1, len(nota_
plt.plot(medias)
```

Out[30]:
```
[<matplotlib.lines.Line2D at 0xf19eca0>]
```

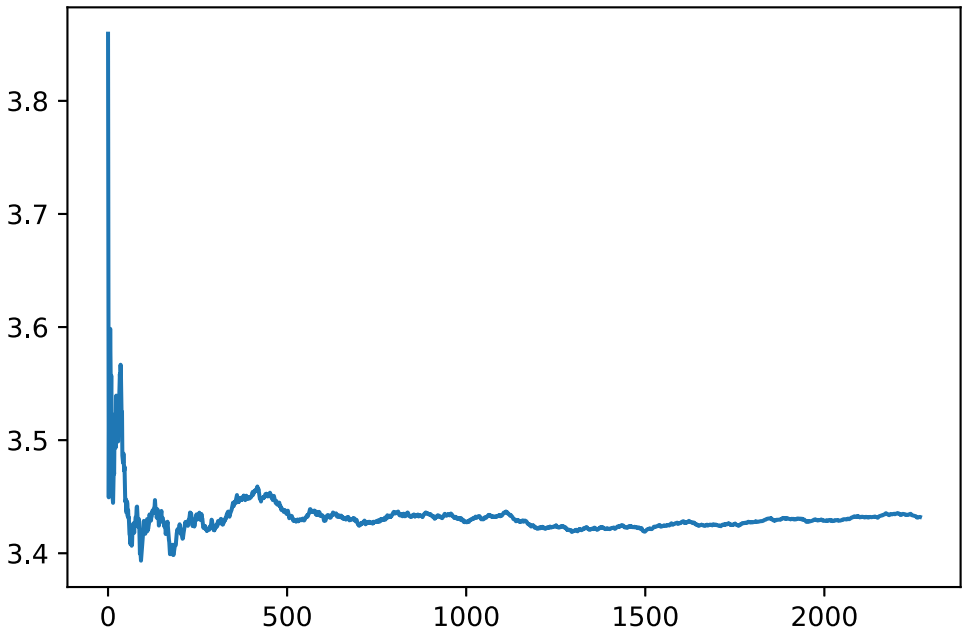


In [31]:
```
np.random.seed(75243)
tmp = nota_media_dos_filmes_com_pelo_menos_10_votos.sample(frac=1)
```

```
medias = [tmp[0:i].mean() for i in range(1, len(tmp))]

plt.plot(medias)
```

Out[31]: [<matplotlib.lines.Line2D at 0xd9c56a0>]



Intervalo de confiança das médias

In [32]:
```
zconfint(nota_media_dos_filmes_com_pelo_menos_10_votos)
```

Out[32]: (3.4112459477469557, 3.452854733323563)

In [33]:
```
descr_todos_com_10_votos = DescrStatsW(nota_media_dos_filmes_com_pelo_menos_10_votos)
descr_todos_com_10_votos.tconfint_mean()
```

Out[33]: (3.41123483922938, 3.4528658418411386)

Vamos ver o filme 1...

In [34]:
```
filmes = pd.read_csv('movies.csv')
filmes.query('movieId == 1')
```

Out[34]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

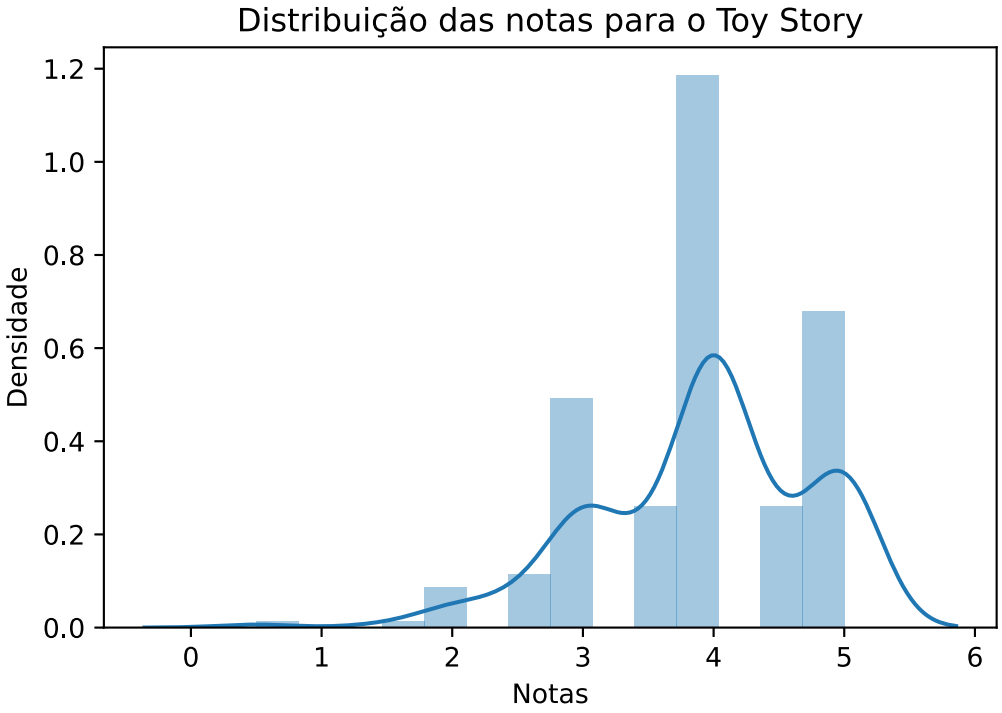
In [35]:
```
notas1 = notas.query('movieId == 1')
notas1.head()
```

Out[35]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
516	5	1	4.0	847434962
874	7	1	4.5	1106635946
1434	15	1	2.5	1510577970
1667	17	1	4.5	1305696483

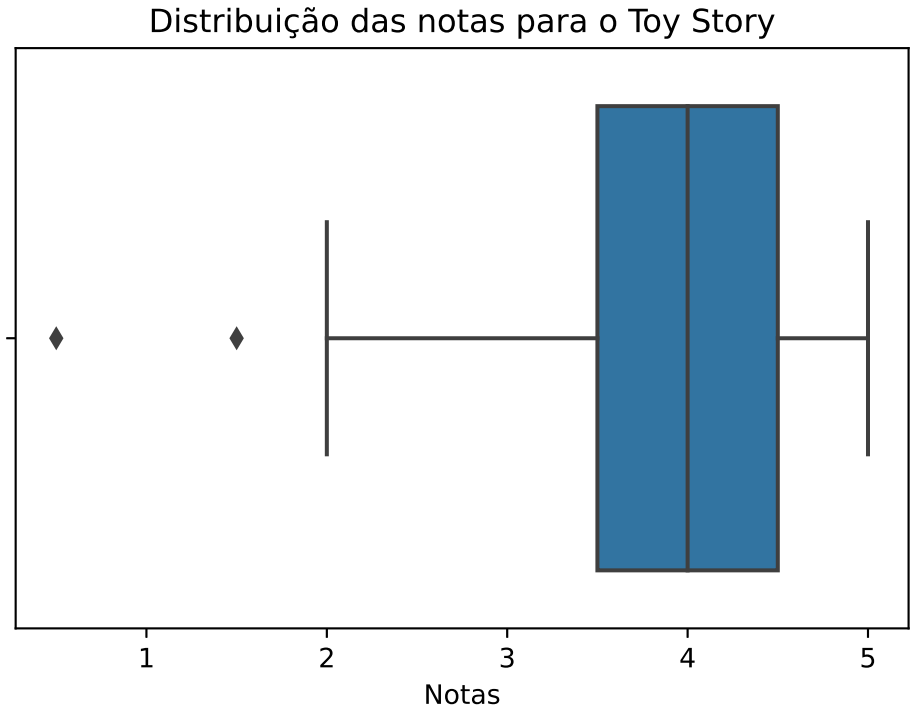
In [36]:
```
ax = sns.distplot(notas1['rating'])
ax.set(xlabel='Notas', ylabel='Densidade')
ax.set_title('Distribuição das notas para o Toy Story')
```

Out[36]:
```
Text(0.5, 1.0, 'Distribuição das notas para o Toy Story')
```



In [37]:
```
ax = sns.boxplot(notas1['rating'])
ax.set(xlabel='Notas')
ax.set_title('Distribuição das notas para o Toy Story')
```

Out[37]:
```
Text(0.5, 1.0, 'Distribuição das notas para o Toy Story')
```



In [38]:
```
notas1['rating'].mean()
```

Out[38]:
```
3.9209302325581397
```

Intervalo de confiança das notas do filme 1

In [39]:
```
zconfint(notas1['rating'])
```

Out[39]: (3.8093359183563402, 4.032524546759939)

P-test da média das notas do filme 1 ser a média das notas de todos os filmes

In [40]:
```
ztest(notas1['rating'], value=3.4320503405352603)
```

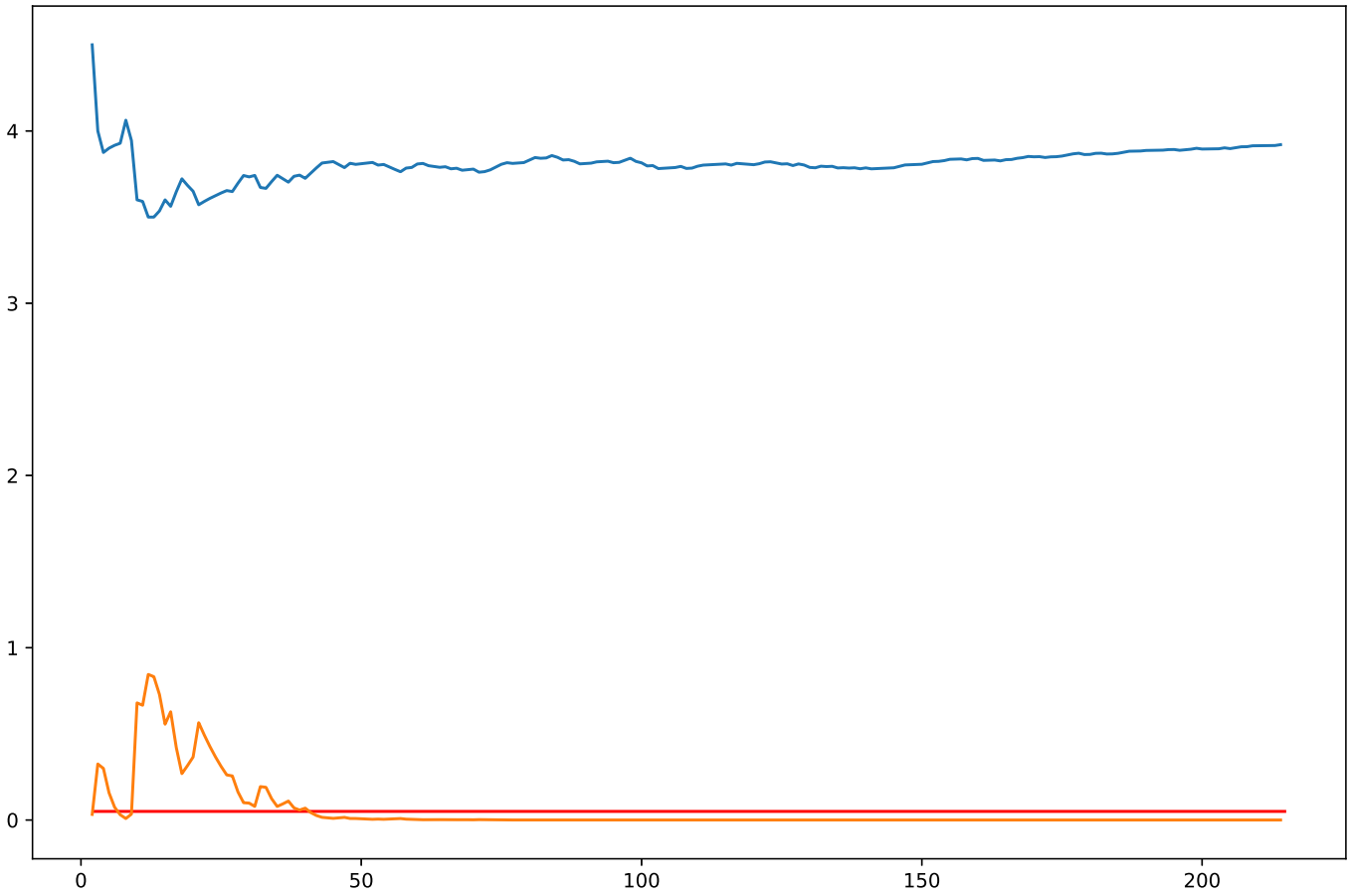
Out[40]: (8.586342305916716, 8.978190401886942e-18)

In [41]:
```
np.random.seed(75241)
tmp = notas1.sample(frac=1)['rating']

def calcula_teste(i):
    media = tmp[0:i].mean()
    stat, p = ztest(tmp[0:i], value=3.4320503405352603)
    return (i, media, p)

valores = np.array([calcula_teste(i) for i in range(2, len(tmp))])
plt.figure(figsize=(12,8))
plt.plot(valores[:,0], valores[:,1])
plt.plot(valores[:,0], valores[:,2])
plt.hlines(y=0.05, xmin=2, xmax=len(tmp), colors='r')
```

Out[41]: <matplotlib.collections.LineCollection at 0xf3286d0>



Comparação de dois conjuntos de amostras

Intervalo de confiança entre as médias do filme 1 e a média

de todos os filmes

In [42]:
```
print(ztest(notas1['rating'], notas['rating']))
zconfint(notas1['rating'], notas['rating'])
```

```
(5.894327101501841, 3.762112778881965e-09)
```

Out[42]:
```
(0.2799245129192442, 0.5588219849631111)
```

In [43]:
```
ttest_ind(notas1['rating'], notas['rating'])
```

Out[43]:
```
Ttest_indResult(statistic=5.894327101501841, pvalue=3.774003138720876e-09)
```

P-Value menor do que 0.05, existe uma diferença significante

A média do filme 1 está entre 0.27 e 0.55 mais alta do que as médias de todos os filmes

In [44]:
```
descr_todas_as_notas = DescrStatsW(notas['rating'])
descr_toystory = DescrStatsW(notas1['rating'])
comparacao = descr_todas_as_notas.get_compare(descr_toystory)
comparacao.summary()
```

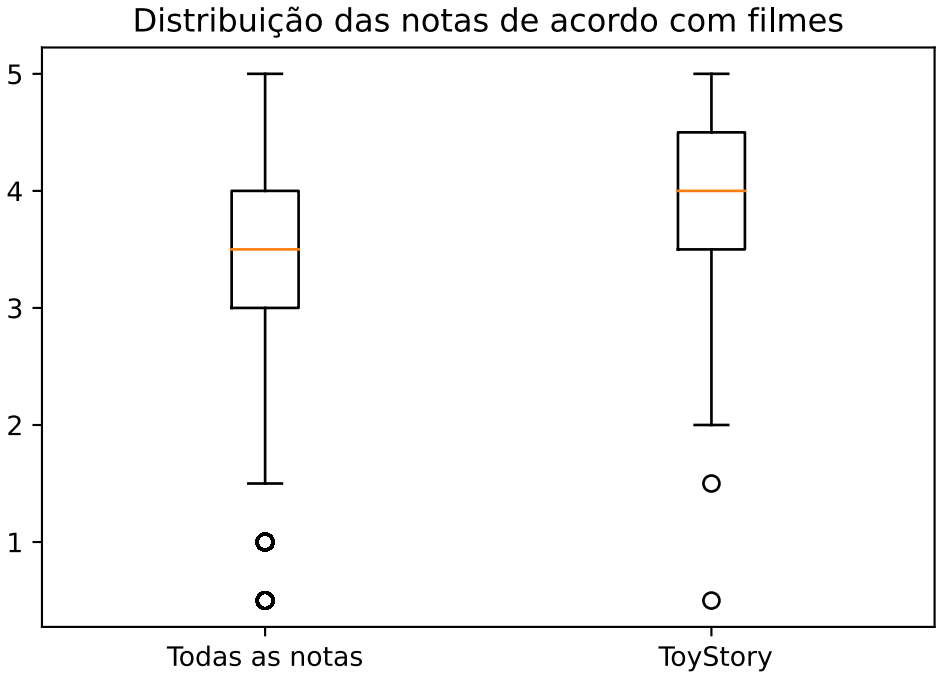
Out[44]:

Test for equality of means

	coef	std err	t	P>|t|	[0.025	0.975]
subset #1	-0.4194	0.071	-5.894	0.000	-0.559	-0.280

In [45]:
```
plt.boxplot([notas['rating'], notas1['rating']], labels=['Todas as notas',' ToyStory'])
plt.title('Distribuição das notas de acordo com filmes')
```

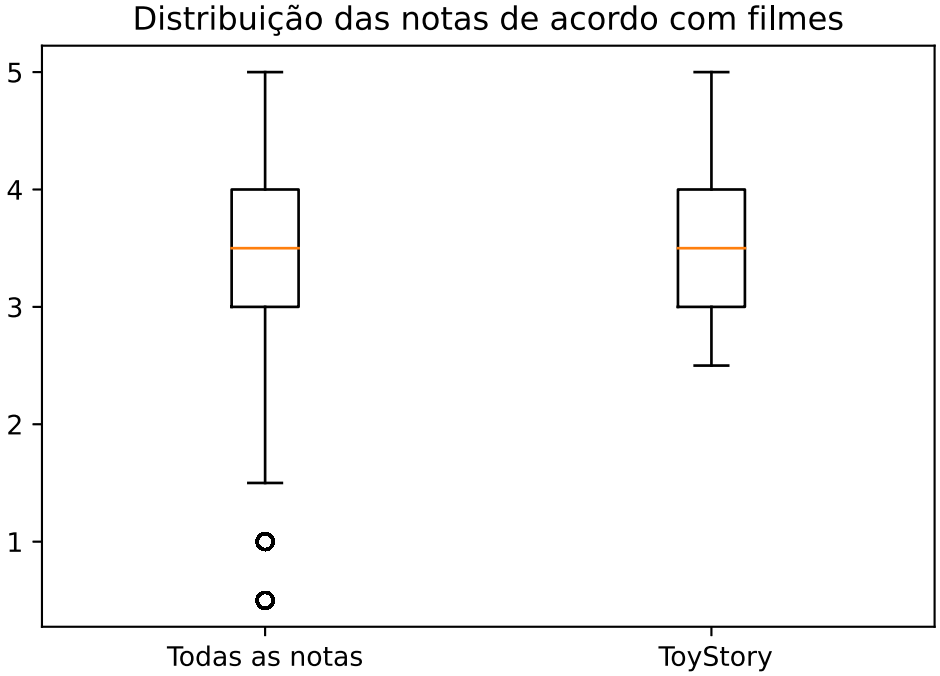
Out[45]:
```
Text(0.5, 1.0, 'Distribuição das notas de acordo com filmes')
```



Teste utilizando uma quantidade pequena de amostras

In [46]:
```
plt.boxplot([notas['rating'], notas1[3:12]['rating']], labels=['Todas as notas',' ToyStory'])
plt.title('Distribuição das notas de acordo com filmes')
```

Out[46]:
```
Text(0.5, 1.0, 'Distribuição das notas de acordo com filmes')
```



In [47]:
```
descr_todas_as_notas = DescrStatsW(notas['rating'])
descr_toystory = DescrStatsW(notas1[3:12]['rating'])
comparacao = descr_todas_as_notas.get_compare(descr_toystory)
comparacao.summary(use_t=True)
```

Out[47]:

Test for equality of means

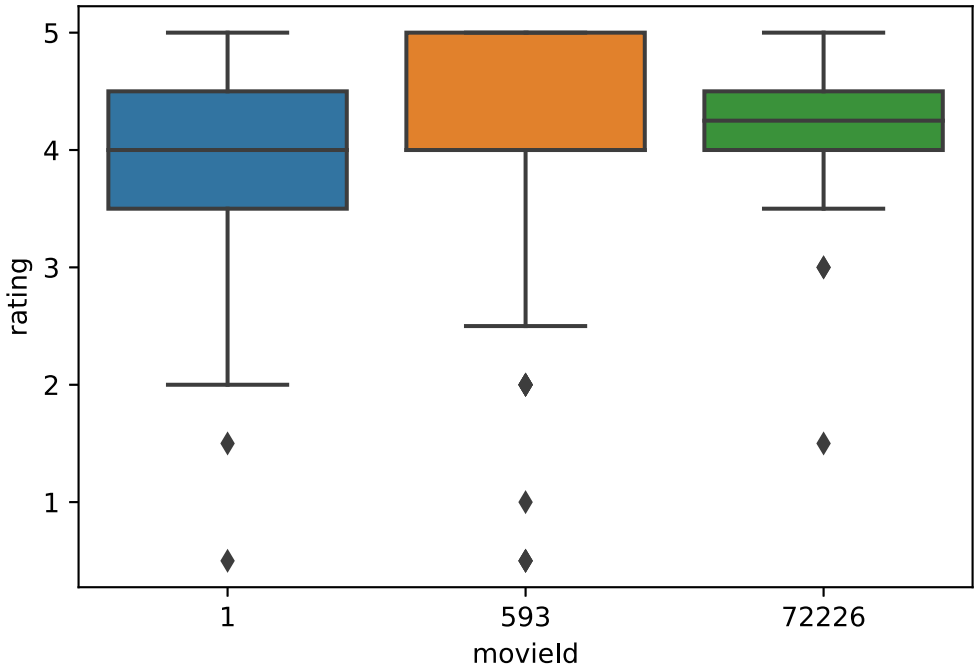
	coef	std err	t	P>|t|	[0.025	0.975]
subset #1	-0.0540	0.348	-0.155	0.877	-0.735	0.627

Comparar a média de dois filmes

In [48]:
```
filmes.query('movieId in [1, 593, 72226]')
notas593 =  notas.query("movieId == 593")
notas72226 =  notas.query("movieId == 72226")
```

In [49]:
```
sns.boxplot(x = "movieId", y = "rating", data = notas.query("movieId in (1, 593, 72226)"))
```

Out[49]:
```
<AxesSubplot:xlabel='movieId', ylabel='rating'>
```



In [50]:
```
notas.query("movieId in (1, 593, 72226)").groupby("movieId").count()
```

Out[50]:

movieId	userId	rating	timestamp
1	215	215	215
593	279	279	279
72226	18	18	18

In [51]:
```
descr_1 = DescrStatsW(notas1['rating'])
descr_593 = DescrStatsW(notas593['rating'])
comparacao = descr_1.get_compare(descr_593)
comparacao.summary()
```

Out[51]:

Test for equality of means

	coef	std err	t	P>|t|	[0.025	0.975]
subset #1	-0.2404	0.077	-3.132	0.002	-0.391	-0.090

In [52]:
```
descr_72226 = DescrStatsW(notas72226['rating'])
comparacao = descr_72226.get_compare(descr_593)
comparacao.summary(use_t=True)
```

Out[52]:

Test for equality of means

	coef	std err	t	P>|t|	[0.025	0.975]
subset #1	-0.0780	0.208	-0.374	0.708	-0.488	0.332

In [53]:
```
comparacao = descr_1.get_compare(descr_72226)
comparacao.summary(use_t=True)
```

Out[53]:

Test for equality of means

	coef	std err	t	P>|t|	[0.025	0.975]
subset #1	-0.1624	0.206	-0.788	0.431	-0.568	0.243

As notas do filme 1 não seguem uma distribuição normal

p < 0.05

In [54]:
```
_, p = normaltest(notas1['rating'])
p
```

Out[54]:
```
0.00011053430732728716
```

Testes não-paramétricos

In [55]:
```
_, p = ranksums(notas1['rating'], notas593['rating'])
p
```

Out[55]:
```
0.0003267718756440693
```