Importo pandas

In [1]:
```
import pandas as pd
```

El nombre del primer csv se asigna a la variable 'tmdb'

In [2]:
```
tmdb=pd.read_csv('C:/Users/Carina/Downloads/ml-latest-small/tmdb_5000_movies.csv')
```

Visualizo solo las primeras dos filas del csv que contiene los datos de 5000 peliculas

In [3]:
```
tmdb.head(2)
```

Out[3]:

	budget	genres	homepage	id	keywords	original_langua
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	

Veo la descripcion de mi csv que contiene datos de 5000 peliculas

In [4]:
```
tmdb.describe()
```

Out[4]:

	budget	id	popularity	revenue	runtime	vote_average	vote_co
count	4.803000e+03	4803.000000	4803.000000	4.803000e+03	4801.000000	4803.000000	4803.000(
mean	2.904504e+07	57165.484281	21.492301	8.226064e+07	106.875859	6.092172	690.217
std	4.072239e+07	88694.614033	31.816650	1.628571e+08	22.611935	1.194612	1234.585
min	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000(
25%	7.900000e+05	9014.500000	4.668070	0.000000e+00	94.000000	5.600000	54.000(
50%	1.500000e+07	14629.000000	12.921594	1.917000e+07	103.000000	6.200000	235.000(
75%	4.000000e+07	58610.500000	28.313505	9.291719e+07	118.000000	6.800000	737.000(
max	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	13752.000(

Grafico con la funcion 'seaborn' para una visualizacon estadistica

In [5]:
```
import seaborn as sb
```

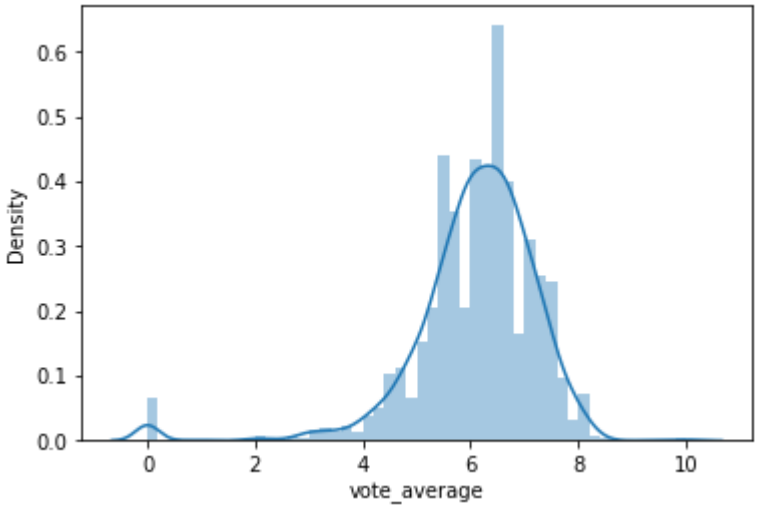
Grafico una distribución con ayuda de 'distplot'

'vote_average' es utilizada como entrada para crear un histograma o un gráfico de distribución de datos con la función distplot(). Esta función generará un gráfico que muestra la distribución de los valores en la columna 'vote_average' en el conjunto de datos 'tmdb'. El eje x del gráfico representará los valores de 'vote_average', mientras que el eje y mostrará la frecuencia o densidad de los valores en 'vote_average'.

In [6]:
```
sb.distplot(tmdb.vote_average)
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[6]:
```
<AxesSubplot:xlabel='vote_average', ylabel='Density'>
```



Visualizo la frecuencia con parámetros: 'kde' (estimación de densidad de kernel), 'norm_hist' (indica si se debe normalizar o no el histograma en el gráfico)

In [7]:
```
ax= sb.distplot(tmdb.vote_average, kde= False, norm_hist= False)
ax.set(xlabel = "Nota promedio", ylabel = "Frecuencia")
ax.set_title("Grafico de distribución del Promedio de notas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[7]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de TMDB')
```

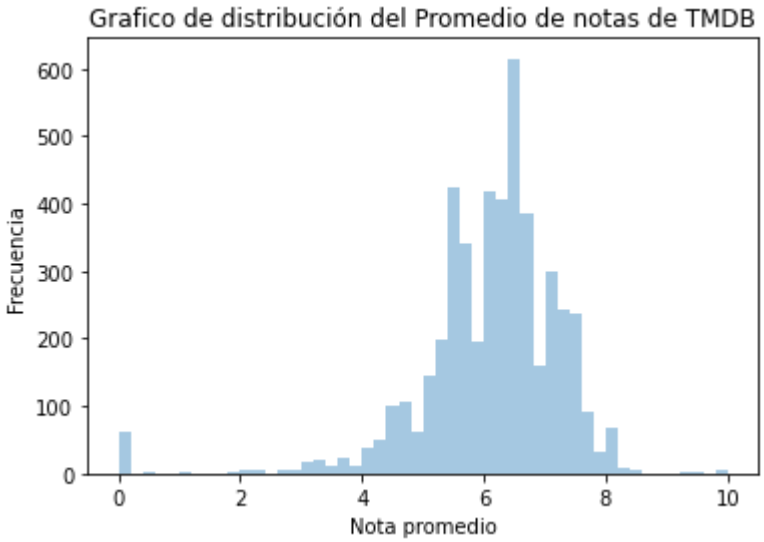


Grafico una gráfica con boxplot

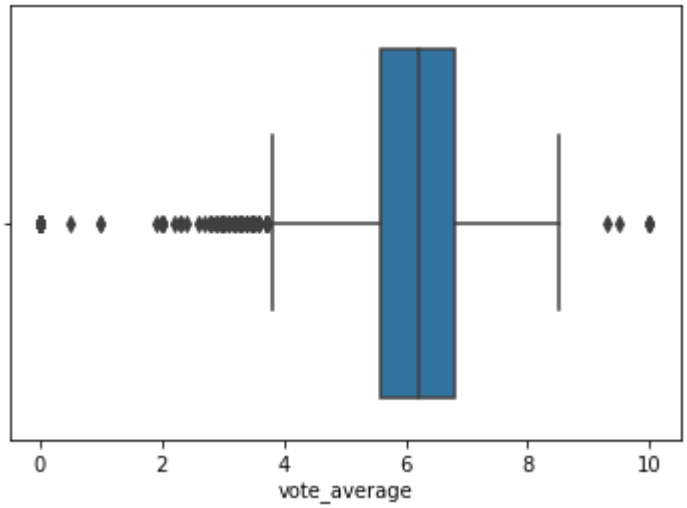
Grafico mi diagrama de caja para representar la distribución de datos en un conjunto de datos.

El resultado muestra la mediana en el tercer cuartil

In [8]:
```
sb.boxplot(tmdb.vote_average)
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]:
```
<AxesSubplot:xlabel='vote_average'>
```



Realizo una consulta para ver mejor el promedio de las notas, hay peliculas con nota 0 y 10, para ver más a detalle realizo una query

Utilizo 'query' para filtrar y seleccionar filas del DataFrame

In [9]:
```
tmdb.query("vote_average== 0")
```

Out[9]:

	budget	genres	homepage	id	keywords	original_languag
1464	0	[{"id": 18, "name": "Drama"}, {"id": 80, "name...	NaN	310706	[]	e
3669	0	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	http://www.romeothemovie.com/	113406	[]	e
3670	0	[{"id": 10751, "name": "Family"}]	NaN	447027	[]	e
3852	0	[{"id": 18, "name": "Drama"}]	NaN	395766	[{"id": 11162, "name": "miniseries"}]	e
3855	3500000	[{"id": 99, "name": "Documentary"}]	http://www.timetochoose.com	370662	[{"id": 2210, "name": "climate change"}, {"id"...	e
...	...	...	...	...	...	.
4769	0	[{"id": 28, "name": "Action"}, {"id": 37, "nam...	NaN	69382	[]	e
4771	0	[{"id": 27, "name": "Horror"}]	NaN	220490	[]	e
4780	0	[{"id": 53, "name": "Thriller"}, {"id": 80, "n...	NaN	366967	[]	e
4785	0	[{"id": 18, "name": "Drama"}]	NaN	287625	[]	e
4794	0	[{"id": 53, "name":	NaN	286939	[]	e

budget	genres	homepage	id	keywords	original_languag
	"Thriller"}, {"id": 27, "n...				

63 rows × 20 columns

Verifico anteriormente que existe un error de datos, pues no tiene nota porque no tuvo votos, exluyo esta información porque la distribución de las notas es afectada.

Realizo otra consulta dode el promedio de votos sea igual a 10

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

Out[10]:

	budget	genres	homepage	id	keywords	original_language	original_title	overv
3519	0	[{"id": 35, "name": "Comedy"}]	NaN	89861	[{"id": 131, "name": "italy"}, {"id": 8250, "n...	en	Stiff Upper Lips	Stiff U Lips b paroc Briti
4045	0	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	NaN	78373	[{"id": 1415, "name": "small town"}, {"id": 15...	en	Dancer, Texas Pop. 81	Four g frie have gr toget
4247	1	[{"id": 10749, "name": "Romance"}, {"id": 35, ...	NaN	361505	[]	en	Me You and Five Bucks	womani yet lov l Char
4662	0	[{"id": 35, "name": "Comedy"}]	NaN	40963	[{"id": 10183, "name": "independent film"}]	en	Little Big Top	An a out of cl return his sr

Se deben dejar número considerable de votos, para no afectar la distribución muestro cuantos votos tiene

Uso unicamente peliculas que tienen 10 o más votos

In [11]:
```
tmdb_10_o_mas_votos = tmdb.query("vote_count >= 10")
```

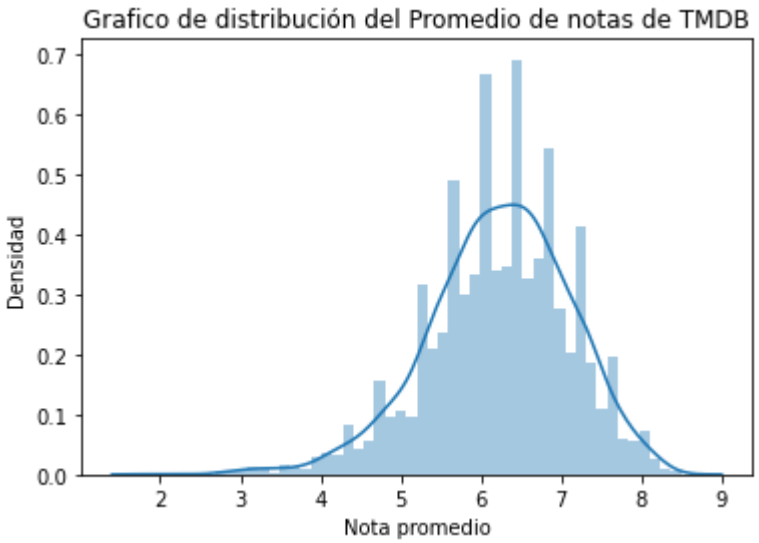
Realizo graficos que había echo anteriormente

Visualmente hay una leve tendencia del gráfico hacia la cola izquierda, o sea, la distribución es mayor ahí, los datos están más agrupados entre 6.3, más o menos, con aproximación visual, entre 6.3- 8, están agrupados los datos y antes de la media, antes del promedio, los datos están más distribuidos. Además la cola de la curva a la izquierda es mayor.

In [12]:
```
ax= sb.distplot(tmdb_10_o_mas_votos.vote_average)
ax.set(xlabel = "Nota promedio", ylabel = "Densidad")
ax.set_title("Grafico de distribución del Promedio de notas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de TMDB')
```



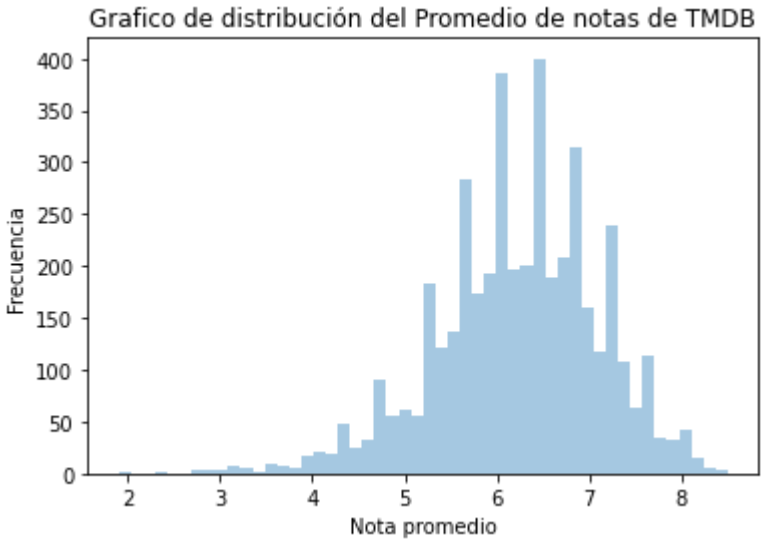
Cambio los parámetros que había definido anteriormente, que era kde, con minúscula, kde = False para eliminar la curva. Y hist_norm para graficar la frecuencia y no la curva normalizada. También false.

Desactivo esos dos parámetros que en distplot están por default como verdaderos.

In [13]:
```
ax= sb.distplot(tmdb_10_o_mas_votos.vote_average, kde= False, norm_hist= False)
ax.set(xlabel = "Nota promedio", ylabel = "Frecuencia")
ax.set_title("Grafico de distribución del Promedio de notas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de TMDB')
```



Nuevo gráfico con diagrama de caja

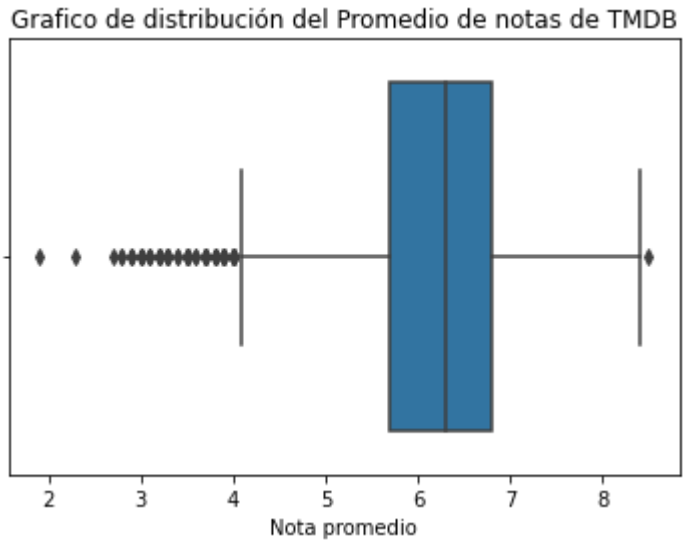
Ya no tengo valores 0. En el gráfico de boxplot también visualmente puedo destacar que la caja,el rectángulo a la derecha que vendría a ser entre la media y el tercer cuartil es más pequeño. El área es menor que el área del primero al segundo cuartil, que es la mediana.

El área del primero al segundo cuartil es mayor, lo que demuestra que mis datos están menos densos al lado izquierdo de la media. Y tengo mis datos outliers conocidos que están fuera de la distribución de los datos.

In [14]:
```
ax= sb.boxplot( tmdb_10_o_mas_votos.vote_average)
ax.set(xlabel = "Nota promedio")
ax.set_title("Grafico de distribución del Promedio de notas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de TMDB')
```



Importo el archivo csv que contiene las calificaciones de cada usuario

Visualizo las primeros 5 filas del archivo

In [15]:
```
 notas=pd.read_csv('C:/Users/Carina/Downloads/ml-latest-small/ratings.csv')
notas.head()
```

Out[15]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Agrupo el conjunto de datos para obtener el promedio con la variable 'rating'

In [16]:
```
nota_promedio_por_pelicula= notas.groupby("movieId").mean()["rating"]
nota_promedio_por_pelicula.head()
```

Out[16]:
```
movieId
1    3.920930
2    3.431818
3    3.259615
4    2.357143
5    3.071429
Name: rating, dtype: float64
```

Visualizo la distribución de mis datos

Con 'nota_promedio_por_pelicula' se creó una serie (no es dataFrame), se tiene que pasar como parámetro el 'values' para decirle los valores que se obtuvieron de mi variable 'nota_promedio_por_pelicula

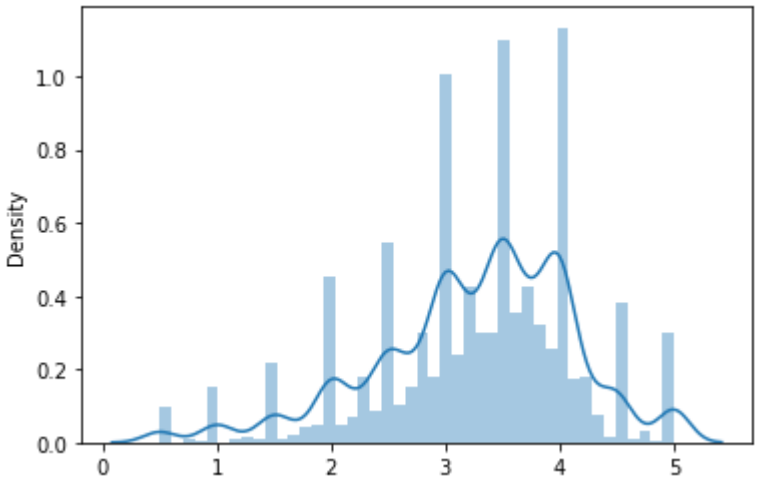
Se visualiza:

No tiene distribución normal, no hay nota cero, pero hay promedio con nota 5, probablemente hay datos que perjudican este análisis, se limpian los datos de películas con menos de 10 evaluaciones para que el gráfico se acerque mas a la realidad

In [17]:
```
sb.distplot(nota_promedio_por_pelicula.values)
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[17]: <AxesSubplot:ylabel='Density'>



Agrupo nuevamente por la variable 'movieId'

In [18]:
```
cantidad_de_votos_por_pelicula= notas.groupby("movieId").count()
cantidad_de_votos_por_pelicula
```

Out[18]:

movieId	userId	rating	timestamp
1	215	215	215
2	110	110	110
3	52	52	52
4	7	7	7
5	49	49	49
...	...	...	...
193581	1	1	1
193583	1	1	1
193585	1	1	1
193587	1	1	1
193609	1	1	1

9724 rows × 3 columns

Denoto los registros que son mayores a 10

In [19]:
```
peliculas_con_10_o_mas_votos= cantidad_de_votos_por_pelicula.query("rating >= 10")
peliculas_con_10_o_mas_votos
```

Out[19]:

movieId	userId	rating	timestamp
1	215	215	215
2	110	110	110
3	52	52	52
5	49	49	49
6	102	102	102
...	...	...	...
174055	13	13	13
176371	18	18	18
177765	13	13	13
179819	12	12	12
187593	12	12	12

2269 rows × 3 columns

Entonces aquí le paso el index para que sepa que solo quiero los índices que tienen mayor notas con más de 10 votos.

In [20]:
```
peliculas_con_10_o_mas_votos= cantidad_de_votos_por_pelicula.query("rating >= 10").inc
peliculas_con_10_o_mas_votos
```

Out[20]:
```
Int64Index([     1,      2,      3,      5,      6,      7,      9,     10,
                11,     12,
            ...
            166461, 166528, 166643, 168250, 168252, 174055, 176371, 177765,
            179819, 187593],
           dtype='int64', name='movieId', length=2269)
```

In [21]:
```
nota_promedio_por_pelicula_con_10_o_mas_votos= nota_promedio_por_pelicula.loc[pelicula
nota_promedio_por_pelicula_con_10_o_mas_votos
```

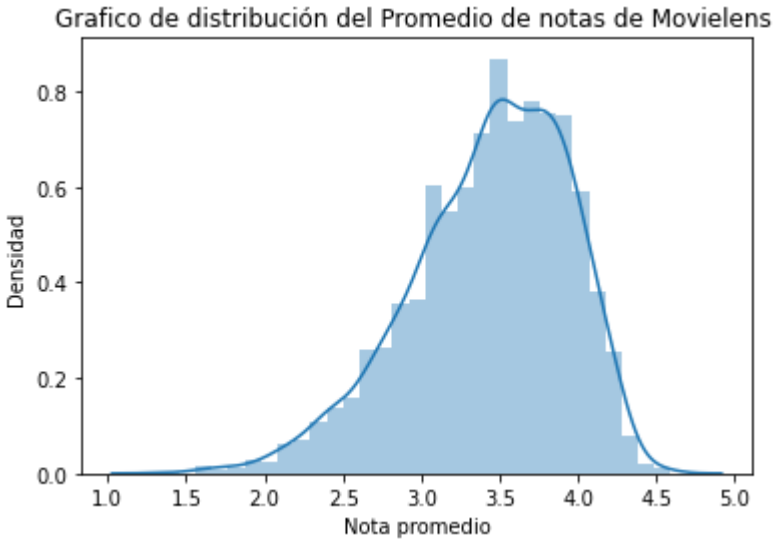
Out[21]:
```
movieId
1         3.920930
2         3.431818
3         3.259615
5         3.071429
6         3.946078
            ...
174055    3.423077
176371    3.805556
177765    3.538462
179819    3.125000
187593    3.875000
Name: rating, Length: 2269, dtype: float64
```

Hago limpieza de datos al graficarlos: Las distribución es similar a la que se obtuvo en 'TMDB'

In [22]:
```
ax=sb.distplot(nota_promedio_por_pelicula_con_10_o_mas_votos.values)
ax.set(xlabel = "Nota promedio", ylabel= "Densidad")
ax.set_title("Grafico de distribución del Promedio de notas de Movielens ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[22]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de Movielens ')
```



Obtengo la frecuencia real de las notas con 'kde'

In [23]:
```
ax=sb.distplot(nota_promedio_por_pelicula_con_10_o_mas_votos.values,kde= False, norm_h
ax.set(xlabel = "Nota promedio", ylabel= "Densidad")
ax.set_title("Grafico de distribución del Promedio de notas de Movielens ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[23]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de Movielens ')
```

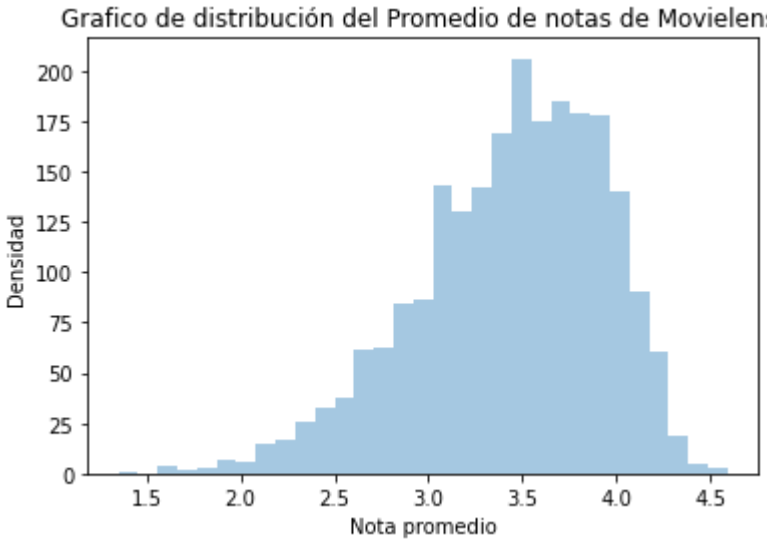


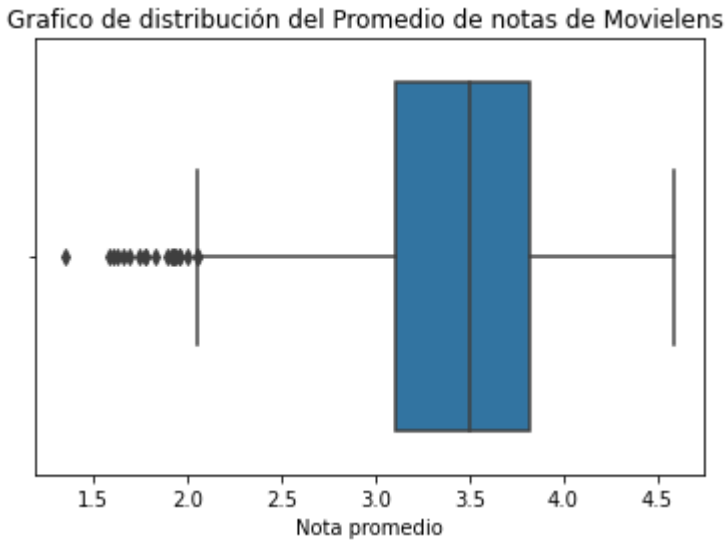
Grafico:

El área a la izquierda es un poco mayor que el área derecha, es decir: El área entre el primer y segundo cuartil que la mediana es mayor que el área entre el segundo y tercer cuartil; similar a lo que obtuve en TMDB.

In [24]:
```
ax=sb.boxplot(nota_promedio_por_pelicula_con_10_o_mas_votos.values)
ax.set(xlabel = "Nota promedio")
ax.set_title("Grafico de distribución del Promedio de notas de Movielens ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[24]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de Movielens ')
```



Conclusión: Por comparación de frecuencias, el comportamiento de los usuarios, personas es muy similar.

En TMDB. se analizó el promedio de las notas, con la diferencia de la escala, las bases eran diferentes, pero después de hacer todos los tratamientos correspondientes se llegó a bases que podrían ser comparables, los resultados son comparables. La escala es diferente en TMDB, la valoración era de 0 a 10. Mientras que en MovieLens la valoración era de 0 a 5.

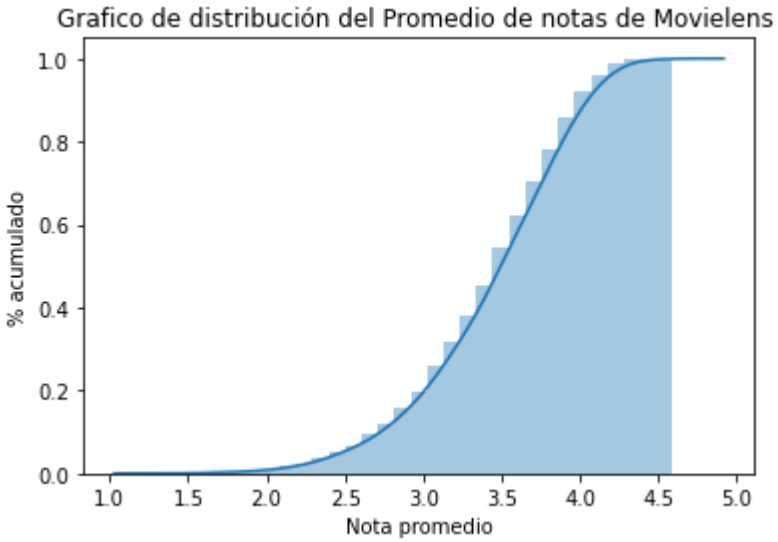
Pero si nos concentramos en el gráfico el comportamiento es muy similar.

Nuevo caso: Obtener un cierto porcentaje de las mejores películas, ahora se debe aplicar 'seaborn.displot' (distribución acumulada)

In [25]:
```
ax=sb.distplot(nota_promedio_por_pelicula_con_10_o_mas_votos.values,
               hist_kws= {"cumulative": True},
               kde_kws={"cumulative": True} )
ax.set(xlabel = "Nota promedio", ylabel= "% acumulado")
ax.set_title("Grafico de distribución del Promedio de notas de Movielens ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[25]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de Movielens ')
```



De la gráfica anterior:

El 3.9 que es el promedio, la nota promedio del primer filme de la primera película de nuestro conjunto de datos de MovieLens está encima del 80%. Entonces podemos deducir que ese filme, esa película está entre los 20% mejores.

Nueva gráfica con TMDB

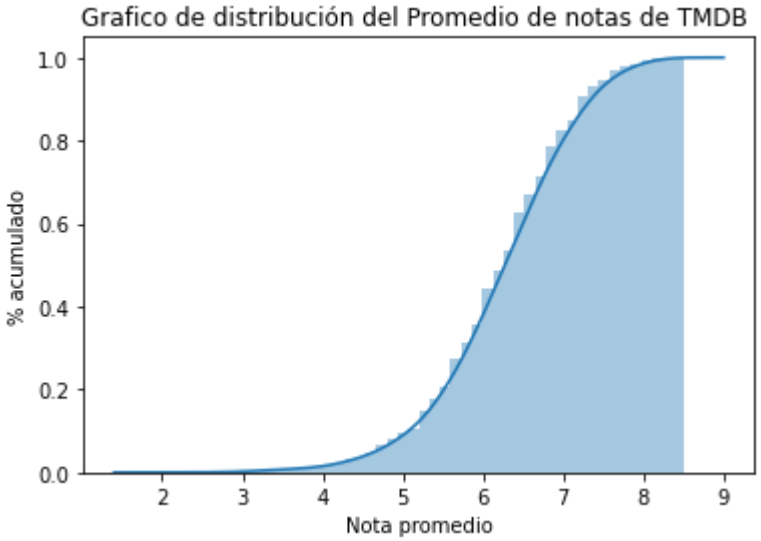
Se visualiza:

El 20% de las películas mejores, de las mejores películas, tienen 7 o más de nota. Aquí, inversamente, podemos decir que nota 5 será un 10%, 15%. Tienen 5 o menos de nota.

In [26]:
```
ax=sb.distplot(tmdb_10_o_mas_votos.vote_average,
               hist_kws= {"cumulative": True},
               kde_kws={"cumulative": True} )
ax.set(xlabel = "Nota promedio", ylabel= "% acumulado")
ax.set_title("Grafico de distribución del Promedio de notas de TMDB ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[26]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de TMDB ')
```



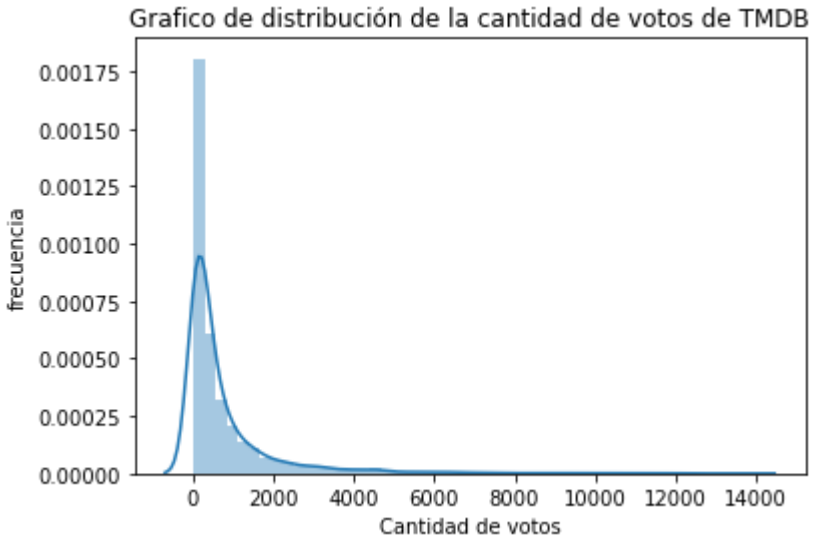
Analizando otras variables

Existen más películas con menos votos:

In [27]:
```
ax= sb.distplot(tmdb_10_o_mas_votos.vote_count)
ax.set(xlabel = "Cantidad de votos", ylabel = "frecuencia")
ax.set_title("Grafico de distribución de la cantidad de votos de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[27]:
```
Text(0.5, 1.0, 'Grafico de distribución de la cantidad de votos de TMDB')
```



Existen películas que tuvieron presupuesto 0:

In [28]:
```
tmdb.budget
```

Out[28]:
```
0       237000000
1       300000000
2       245000000
3       250000000
4       260000000
          ...
4798       220000
4799         9000
4800            0
4801            0
4802            0
Name: budget, Length: 4803, dtype: int64
```

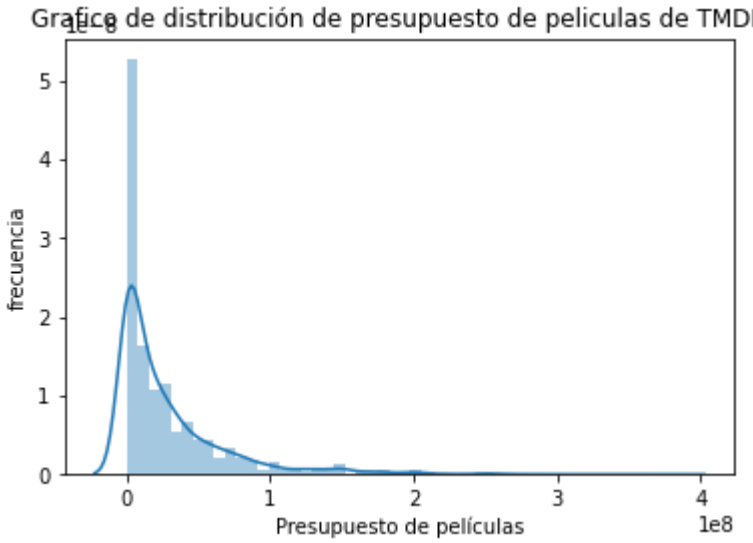
Obtengo los valores que son mayor que 10 para que el análisis sea coherente

Lo que podemos ver es que la mayor cantidad de películas no gastaron ni siquiera 100.000.000 para ser realizadas. No son pocas películas que gastaron un presupuesto mayor que 100.000.000.

In [29]:
```
ax= sb.distplot(tmdb.query("budget >= 0").budget)
ax.set(xlabel = "Presupuesto de películas", ylabel = "frecuencia")
ax.set_title("Grafico de distribución de presupuesto de peliculas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]:
```
Text(0.5, 1.0, 'Grafico de distribución de presupuesto de peliculas de TMDB')
```



Analizamos la popularidad:

Solo existe un registro de los 5000 registros que hay, no afecta el análisis, puede ser que tuvo 0 popularidad

In [30]:
```
tmdb.query("popularity == 0")
```

Out[30]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	pop
4553	0	[]	NaN	380097	[]	en	America Is Still the Place	1971 post civil rights San Francisco seemed li...	

Visualizo la distribución

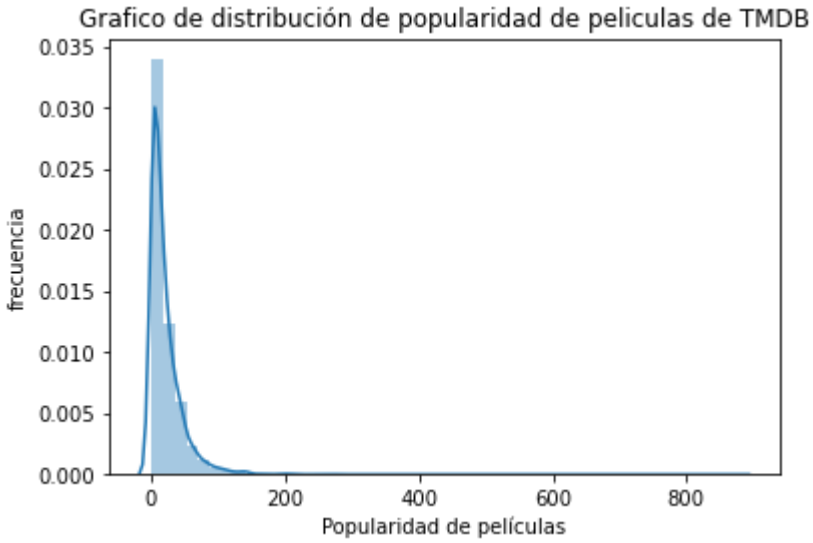
La mayor cantidad de películas tienen poca popularidad.

Son pocas las películas, nuestra escala está súper espaciada, de 0 a 800. Tenemos algunos valores que tienen 800. Pero la mayor cantidad de películas están distribuidas, están localizadas entre 0 y 50, 100,

In [31]:
```
ax= sb.distplot(tmdb.popularity)
ax.set(xlabel = "Popularidad de películas", ylabel = "frecuencia")
ax.set_title("Grafico de distribución de popularidad de peliculas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[31]:
```
Text(0.5, 1.0, 'Grafico de distribución de popularidad de peliculas de TMDB')
```

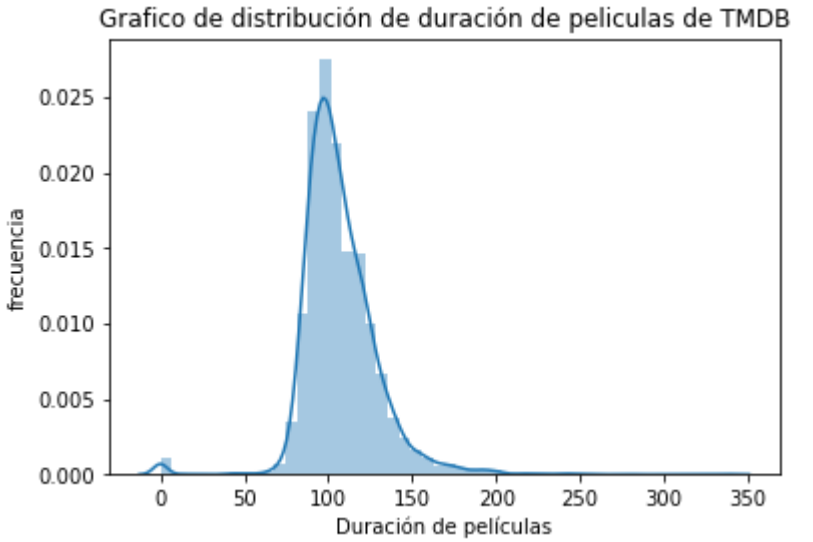


¿Cuál es la duración?

In [32]:
```
ax= sb.distplot(tmdb.runtime)
ax.set(xlabel = "Duración de películas", ylabel = "frecuencia")
ax.set_title("Grafico de distribución de duración de peliculas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[32]:
```
Text(0.5, 1.0, 'Grafico de distribución de duración de peliculas de TMDB')
```



Veo cuantos valores nulos tiene la columna

In [33]:
```
tmdb.runtime.isnull().sum()
```

Out[33]:
```
2
```

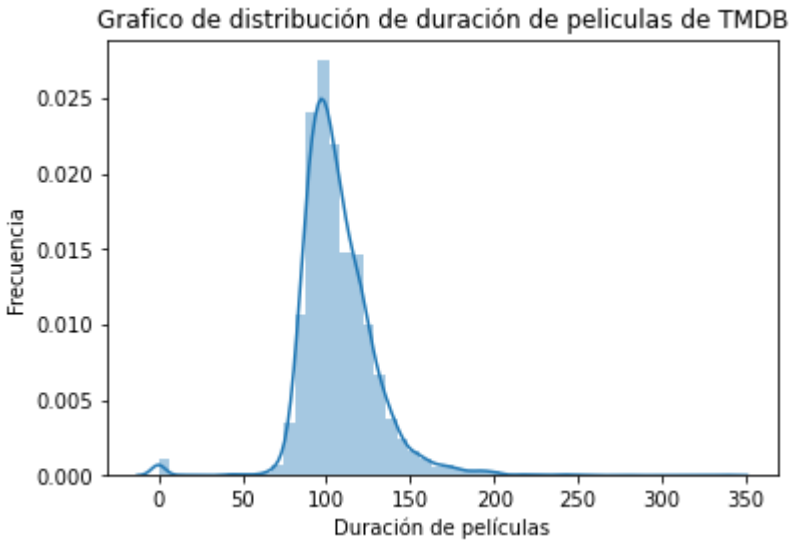
Elimino los valores nulos

Obtengo una campana de Gauss, parece ser una normal, pero la sospecha inicial de que sí existe películas con 0, y aquí vemos que hay un número considerable de películas con duración 0 que no tiene ningún sentido. No puede ser que una película no tenga duración, entonces lo que vamos a hacer es dar un query aquí antes de nuestra variable.

In [34]:
```
ax= sb.distplot(tmdb.runtime.dropna())
ax.set(xlabel = "Duración de películas", ylabel = "Frecuencia")
ax.set_title("Grafico de distribución de duración de peliculas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[34]:
```
Text(0.5, 1.0, 'Grafico de distribución de duración de peliculas de TMDB')
```



Podemos ver es que la mayoría de las películas, la media está próxima de 100. Entonces los valores están entre 100, 80 y 120 podemos decir así a ojo. Entre 80 y 120 la mayoría de las películas tienen esa duración, de 80 a 120 minutos.

In [35]:
```
ax= sb.distplot(tmdb.query("runtime > 0").runtime.dropna())
ax.set(xlabel = "Duración de películas", ylabel = "Frecuencia")
ax.set_title("Grafico de distribución de duración de peliculas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[35]:
```
Text(0.5, 1.0, 'Grafico de distribución de duración de peliculas de TMDB')
```

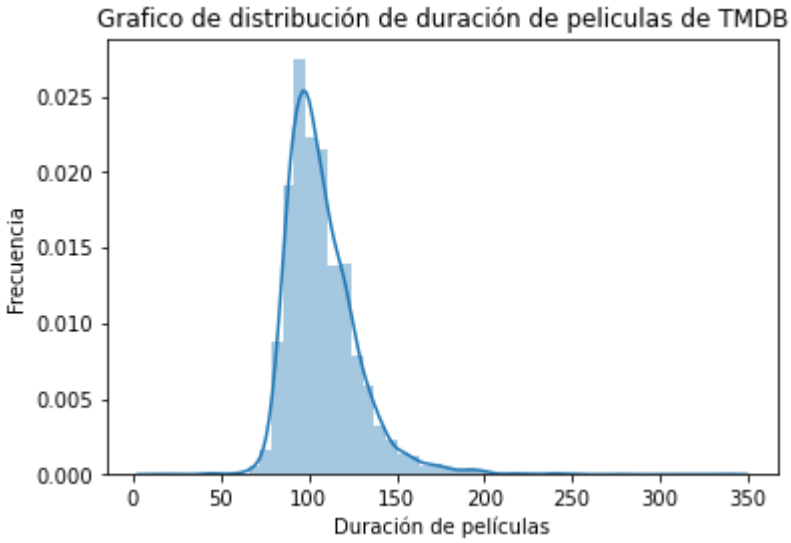


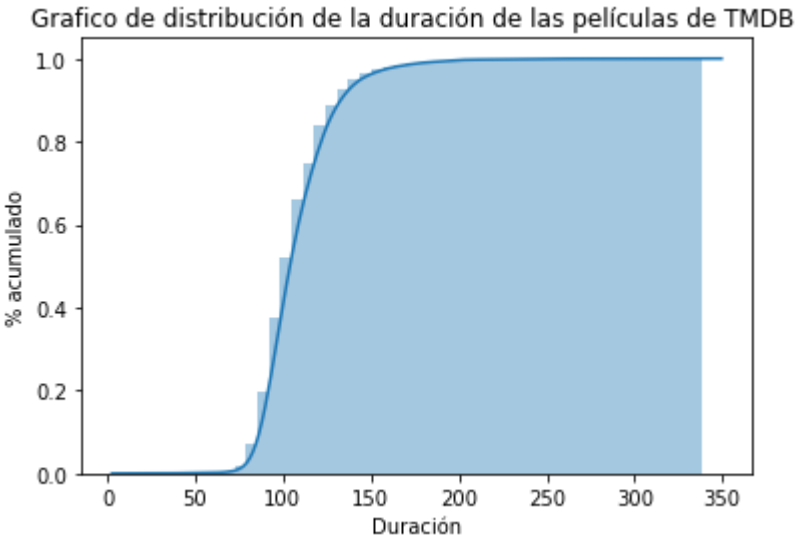
Grafico la distribución acumulada para visualizar mejo el análisis

Decimos que el 40% de las películas, tienen menos de 100 minutos.

In [36]:
```
ax= sb.distplot(tmdb.query("runtime > 0").runtime.dropna(),
                hist_kws= {"cumulative": True},
                kde_kws={"cumulative": True} )
ax.set(xlabel = "Duración", ylabel= "% acumulado")
ax.set_title("Grafico de distribución de la duración de las películas de TMDB ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[36]:
```
Text(0.5, 1.0, 'Grafico de distribución de la duración de las películas de TMDB ')
```



Existen en pandas en realidad una forma de pedir los cuantiles, en realidad, son los percentiles, decirle: "quiero que me digas el 80% de los valores de los registros, cuánto representan o qué

número está representado". En esta curva de acumulada que hicimos.

El 80% de las películas tienen 121 o menos de duración.

In [37]:
```
tmdb.query("runtime > 0").runtime.dropna().quantile(0.8)
```

Out[37]:
```
121.0
```

Efecto del tamaño de la muestra

Cuando se grupa el promedio de cada película y calculando el promedio de los promedios de esas películas. Nos da 3.43.

¿Ahora, qué significa ese 3.43? Si recordamos en MovieLens, la menor nota que teníamos, el range de notas era de 0.5 a 5. Aquí tenemos una diferencia de 4.5, es nuestro intervalo de notas. Si lo dividimos entre 2, dan 2.75.

¿qué significa? Que en realidad la mitad de nuestro intervalo es 2.75, sin embargo, el promedio de las notas, el promedio, la media de las notas es 3.43.

Entonces da a parecer que a las personas les gusta o aprecian normalmente las películas y la tendencia es a dar una nota alta.

In [38]:
```
nota_promedio_por_pelicula_con_10_o_mas_votos.mean()
```

Out[38]:
```
3.4320503405352603
```

Viendo el tamaño del array que consolidó el tamaño de las notas

In [39]:
```
len(nota_promedio_por_pelicula_con_10_o_mas_votos)
```

Out[39]:
```
2269
```

Visualizando solo el promedio de las primeras 5 películas

Las primeras 6 películas tienen un promedio mayor que el promedio de todas las películas

In [40]:
```
nota_promedio_por_pelicula_con_10_o_mas_votos[0:5].mean()
```

Out[40]:
```
3.5259741603585653
```

Calculando en forma acumulada los promedios, medias, etc...Para ver como se comporta el acumulado de los promedios a medida que se avanza en los datos

In [45]:
```
promedios = list()

for i in range(1,len(nota_promedio_por_pelicula_con_10_o_mas_votos)):
    promedio = sum(nota_promedio_por_pelicula_con_10_o_mas_votos[0:i]) / i
    promedios.append(promedio)

promedios
```

Out[45]:
```
[3.9209302325581397,
 3.676374207188161,
 3.5374545996639024,
 3.4209480926050695,
 3.5259741603585653,
 3.469175997829668,
 3.420007998139716,
 3.4295335135237663,
 3.4564107421798558,
 3.352874931119765,
 3.396552967684635,
 3.363506887044249,
 3.406839377909513,
 3.4332165225577675,
 3.451002087720583,
 3.405769002692592,
 3.3524884731224396,
 3.3603714480862994,
 3.3531004361987162,
 3.3416954143887807,
 3.331376585132172,
 3.3447231039898004,
 3.3514742733815486,
 3.3879658756270143,
 3.412973556391407,
 3.404174978311576,
 3.425614825316986,
 3.4337122869128076,
 3.4475781010027933,
 3.4424344719950075,
 3.4507430374145236,
 3.4223910131474935,
 3.4190609824460543,
 3.4086964437466603,
 3.4248871013004156,
 3.4171696491401096,
 3.4393473639752203,
 3.4400036363518125,
 3.4550555181639973,
 3.4495614831510744,
 3.455822178683975,
 3.433832556918074,
 3.441766683501375,
 3.4432967795375005,
 3.4437021502486327,
 3.4427521035040973,
 3.4283255881604555,
 3.4216342812642555,
 3.432757391170427,
 3.427227243347019,
 3.4237521993598223,
 3.433872349372133,
 3.4261353149275555,
 3.4349105868733414,
 3.4231070177613585,
 3.4295266905808375,
 3.4296998002091375,
 3.4271637466822935,
 3.425464752279595,
 3.435567765901939,
```

```
3.4465546751620586,
3.448117958802502,
3.4355486261230976,
3.4378577621732576,
3.430377072035604,
3.43143196488355,
3.432368592967551,
3.4309121920905774,
3.42821923335326,
3.4287684824005944,
3.434641759328454,
3.436180825802427,
3.429055860985416,
3.4350821331342614,
3.427462856207623,
3.4192916556378283,
3.412454843783534,
3.405592512228055,
3.4085080795428526,
3.4184752579603312,
3.4202299592275627,
3.419025064254404,
3.420670009932731,
3.416654097116071,
3.404693460679411,
3.402645530073505,
3.394330584705061,
3.3860922239551132,
3.378383322562359,
3.3850123967561108,
3.3951517364367287,
3.3952346281904355,
3.383078405116914,
3.379430534545761,
3.3735724938312446,
3.3744554192427247,
3.36514586411062,
3.3682226069938452,
3.3655132877312814,
3.3585248215206356,
3.357450318337263,
3.358179415038056,
3.351951135927654,
3.3526878695107674,
3.3485007800702924,
3.3550400808872394,
3.3547736372550676,
3.355456950137627,
3.3590766111455386,
3.363591998596663,
3.36583441302372,
3.3673757266437008,
3.3629178561746094,
3.3632065884303874,
3.3605713170174263,
3.3651599879298377,
3.3663124666654802,
3.359718592321854,
3.36239802792059,
3.3633595258360667,
```

```
3.363173985051546,
3.3601971491085005,
3.367781753166772,
3.3723042505490675,
3.3707375812505576,
3.37352714189319,
3.3741567578790153,
3.3748173738044156,
3.370582909112687,
3.3715784251964354,
3.3656885135537147,
3.372244477066271,
3.376069348524062,
3.380014815361658,
3.3841583644285635,
3.3842532207454816,
3.38101908388302,
3.3837512971601647,
3.380605090500225,
3.379672196996652,
3.3785658992202667,
3.3834839075139485,
3.3862967073615033,
3.3859578413381595,
3.3849077572749575,
3.389365336236871,
3.389612800441092,
3.3938640449819037,
3.3992546782632926,
3.400049770531661,
3.397090003839398,
3.4024956617088757,
3.40572009093518,
3.40974138904599,
3.4119366058908547,
3.4098020594968173,
3.4095803903280477,
3.408277373324096,
3.414697151366202,
3.4158552941701634,
3.4166884911007833,
3.4130270153024127,
3.411413352631846,
3.411304237789691,
3.4078013838232892,
3.4100435441617027,
3.412176419742973,
3.408783139049014,
3.409141746244594,
3.4083393081650804,
3.406187456484626,
3.4072981237272866,
3.4083156682914835,
3.409443425683748,
3.409309262750016,
3.407986315735727,
3.4091241049123617,
3.4032140979024206,
3.407465045615825,
3.4080870035608934,
```

```
3.4061573711070303,
3.406834593049881,
3.4097582316950525,
3.407078386232942,
3.4058764132035537,
3.406382453992782,
3.404531985402372,
3.4040266227647202,
3.4006291526014407,
3.401869860410715,
3.395577348052544,
3.3948191326981036,
3.395515612663823,
3.390899552804731,
3.3914446372069422,
3.393621867722121,
3.3924169343834296,
3.390916031155423,
3.3864390661747423,
3.381590204177202,
3.3801951806548796,
3.3807882738199546,
3.377870408128534,
3.373720308088688,
3.375274392889757,
3.37187500263301,
3.3687274856301976,
3.3679563598980016,
3.3654780998027958,
3.361555188216433,
3.363264510652479,
3.3650325801215484,
3.361654279489309,
3.360126238301726,
3.3557565112428107,
3.3566375117999225,
3.3592731519693766,
3.3580498466511344,
3.3564149158445082,
3.3572250910695174,
3.3528570675438267,
3.3557365639204666,
3.35369290219885,
3.3515186779330817,
3.3508654636069144,
3.3517465898741405,
3.3532492795348587,
3.357401694975495,
3.3575106523824862,
3.35921712780691,
3.358345602304898,
3.3565163637783724,
3.354400998970896,
3.3528864647872596,
3.3526363847118192,
3.3543201288443965,
3.355638046725503,
3.355732209472403,
3.357649453317172,
3.3577796176320542,
```

```
3.3536671157608287,
3.3537950011693183,
3.354864307187701,
3.3556270912015114,
3.355876232325859,
3.356885543034561,
3.355665583571083,
3.351831294543627,
3.350942151925415,
3.3514795597883014,
3.3549597209046826,
3.3570233728058545,
3.356600355522037,
3.35519641711447,
3.3533677601410363,
3.3543590691136997,
3.354017853539976,
3.3532916861489936,
3.3541798778369643,
3.3546318710499285,
3.351995989040797,
3.354854044241454,
3.3526900147413508,
3.3525551024534357,
3.3474511209347435,
3.348226731160027,
3.3479048897772823,
3.349649217111638,
3.3513233835907767,
3.352035068321439,
3.349467595744607,
3.3472126086687486,
3.345924978802462,
3.34624927672178,
3.3478714601765223,
3.350129107577539,
3.3518808647836074,
3.352156730120974,
3.355056850524055,
3.35599194433248,
3.357467232591815,
3.3577953629726953,
3.358245503687227,
3.359864997752349,
3.358602313549709,
3.3612506344794215,
3.3619829617858645,
3.3626200158958253,
3.362787805152626,
3.3637535614895375,
3.363128353999258,
3.3620404299973923,
3.3597188523585553,
3.3596095576460994,
3.3602110067287807,
3.3593059279638444,
3.359607859492036,
3.359379868912085,
3.359803719220443,
3.3576295307224124,
```

```
3.3562903809676716,
3.354716414080204,
3.3548260332894535,
3.353957885452676,
3.3529613022216838,
3.3549668753081052,
3.3532987654304147,
3.355699070064058,
3.354652335418272,
3.3551211988524066,
3.356038802289436,
3.355926941421143,
3.3536715837808204,
3.356201763922117,
3.35449376409437,
3.351844231231129,
3.353560285180978,
3.3528058074402716,
3.355674884022319,
3.3539055052729077,
3.3509794262731214,
3.3503797843804786,
3.352512403271791,
3.3527995395408765,
3.352728732447035,
3.3510502871191474,
3.351105324831467,
3.35036834213381,
3.3484879134862093,
3.3465480511221095,
3.3468802979136756,
3.3493231124887615,
3.34868359454244,
3.348648483546023,
3.349569362273178,
3.3462968344092694,
3.346011087126156,
3.3451987128193585,
3.3451769993238383,
3.3438157520939935,
3.343607281480015,
3.345282893717403,
3.343790329401414,
3.341458574567883,
3.340238273034221,
3.3397590254377385,
3.342494770321203,
3.3443841531651075,
3.342654515880224,
3.34321396451617,
3.3459693231004155,
3.3480389218561144,
3.3490595859112906,
3.3504668249969143,
3.3523669184476277,
3.3549218000303727,
3.35762910191104,
3.3599379885713887,
3.3616177238761007,
3.363418785754223,
```

```
3.3646494603302286,
3.367067555743681,
3.3692807430431597,
3.371131227658034,
3.3722148863950436,
3.3740877755242713,
3.3760657379887826,
3.377436306044815,
3.378159905335871,
3.380741455141269,
3.382527807717356,
3.3839040647203844,
3.3861701842966485,
3.3882570732869426,
3.390555054424844,
3.3927540870044264,
3.3923754289487116,
3.3950222059696333,
3.395629007536996,
3.397310201618795,
3.3991107872663395,
3.40032021219787,
3.4020046360444933,
3.402910873971461,
3.404385360624247,
3.406187471088951,
3.4065292645446936,
3.4063015649406654,
3.408295180922356,
3.405966218919991,
3.405034506510136,
3.402980847054753,
3.402485560760296,
3.4013371202507523,
3.4015868996931555,
3.4017611454901275,
3.4009533871414903,
3.398270924694736,
3.398496048287218,
3.3990382697049615,
3.3994895458403604,
3.398540211504581,
3.3997530743635327,
3.3997183248866785,
3.3981552343725547,
3.3983547699200773,
3.398297387193001,
3.398730481832234,
3.4000477667177296,
3.4011746476904112,
3.3996409705265256,
3.402532398003182,
3.4022207203416963,
3.402017727824402,
3.402196457338537,
3.402351433803268,
3.403118510034488,
3.403815462030684,
3.404456508166896,
3.403859816847085,
```

```
3.404976419609519,
3.4057940110322455,
3.4058018051174908,
3.4070205159810523,
3.4082439178072965,
3.4085571167482813,
3.410109508913809,
3.411721402584571,
3.411621929895674,
3.4134606930269302,
3.41478474474642,
3.4127774167438103,
3.412266358879973,
3.412250550231414,
3.412315436760712,
3.412898811141231,
3.4143567085985733,
3.4151604454342452,
3.413555356300658,
3.413300014171059,
3.4148571569960677,
3.4172556702155332,
3.417526057048087,
3.4180583095183534,
3.418663796463256,
3.4187919190084344,
3.4188588676207603,
3.420358979726314,
3.419679749516199,
3.4213288271970126,
3.4223901823473524,
3.423747094080465,
3.4235907896265796,
3.425216723362826,
3.4245818814133773,
3.4254782661763596,
3.427920691560365,
3.4283611040889004,
3.4289771171760948,
3.4301097756170162,
3.430450007170427,
3.430944900897528,
3.432173961586734,
3.4338352408048594,
3.43417303403089,
3.435850001362211,
3.437555663813678,
3.4392008440192043,
3.440776036466274,
3.4418898868446135,
3.443384458917838,
3.444879667282456,
3.446687532679322,
3.4478460681237397,
3.4492625809385786,
3.4508809918362378,
3.452672762410073,
3.45410598906202,
3.4550047560673507,
3.4566175933810994,
```

```
3.458267037053904,
3.459327018475352,
3.460527616131251,
3.4605152562932635,
3.4625210667613877,
3.4636269904923314,
3.464802591288678,
3.4655089872276847,
3.4671330791198174,
3.4684206426638307,
3.4696851336447305,
3.470763009389355,
3.4722101645961025,
3.473405083291252,
3.474847699284603,
3.475645767744054,
3.476426432926936,
3.4779044922985687,
3.478903806943261,
3.4805229224062972,
3.4820033601304807,
3.482829940219634,
3.4839575148911655,
3.48532016209836,
3.486277448906086,
3.4875023168056005,
3.488956947344445,
3.4898828240732582,
3.49100936300286,
3.492526415001586,
3.493659295933928,
3.494886957899335,
3.4960108206937393,
3.4974035319507157,
3.498029932859549,
3.498336560896643,
3.498928458895077,
3.499288027864749,
3.500411882931174,
3.5013831956668726,
3.5018283974666162,
3.5028672657608544,
3.504062370370765,
3.5046759562962957,
3.505512870558695,
3.5066641980122166,
3.508074702380315,
3.50877780018798,
3.509778242334896,
3.5097597928210567,
3.5109113037973105,
3.5119160843267414,
3.512350744100218,
3.5127567664271218,
3.5141757814890546,
3.51531537891165,
3.5162044631654243,
3.5173515428497515,
3.5178444321356044,
3.519127176364593,
```

```
3.520280096820217,
3.5206701839424546,
3.521444596993368,
3.522352146475276,
3.5229783560480485,
3.522661545872136,
3.523619072427084,
3.524397139813166,
3.5249598651201244,
3.5244186157785835,
3.524747951054741,
3.5250201467955837,
3.525446637606473,
3.5263354652332795,
3.5268424144989714,
3.5268262539384647,
3.527728561857257,
3.528473758676683,
3.529064724277282,
3.528385923287881,
3.5281803217815937,
3.52658806735375,
3.5267224983660395,
3.5266139784068433,
3.5247969625158575,
3.5252478512746634,
3.526022168493378,
3.5257073798555765,
3.524783465303985,
3.5247948101016973,
3.525308622725305,
3.5259271756211015,
3.5267545278451484,
3.527300993009033,
3.52614240191008,
3.525358245110074,
3.525148632492389,
3.524316964346977,
3.524007674417601,
3.522529500266306,
3.5229487765692182,
3.522500247334399,
3.5224217635242994,
3.5215700139292236,
3.5208493814267805,
3.5199605599567687,
3.5193127666388753,
3.516863165621498,
3.515738044231082,
3.5165681399281388,
3.515047122870432,
3.5143342447166206,
3.514468786710452,
3.5152716663798698,
3.515209463360013,
3.514236769413046,
3.5137104096652854,
3.5142448961597137,
3.5125516659490965,
3.513364079839181,
```

```
3.5132031856408905,
3.513305838156437,
3.5115735730848674,
3.5101057360433363,
3.510045535174713,
3.5106477702651837,
3.5099437926096124,
3.5085568236524693,
3.5081564115977906,
3.508537113571212,
3.5062913304661256,
3.5063438961654643,
3.505871339510491,
3.5056396864553214,
3.5049653277633763,
3.504200422659513,
3.5031131178145762,
3.5005610349798055,
3.500738108415259,
3.499874848952298,
3.497620622142391,
3.497680717605184,
3.497437496302694,
3.497835336300883,
3.498331107455109,
3.4970558181460754,
3.4958702926678256,
3.495134668483776,
3.492130103284111,
3.4901017788571296,
3.4900407822237995,
3.4900565404797748,
3.489867082864688,
3.4898639467527226,
3.490119970497034,
3.4890277292626903,
3.488665149650236,
3.4880167716727275,
3.487693193000313,
3.4868374223862504,
3.485507902826545,
3.484661099964068,
3.483069929453298,
3.482855267429623,
3.483458937126976,
3.4823830897606207,
3.4816375208429076,
3.4809328333107423,
3.481827937835846,
3.48174600474467,
3.4811832376335183,
3.4815208380379126,
3.4821346697425657,
3.4822796061449823,
3.4827019709774762,
3.48060848474123,
3.4805036996896943,
3.481222521380277,
3.4816973243023996,
3.4809201313867897,
```

```
3.4807598891305314,
3.4816604377525744,
3.4819143435779853,
3.481687439446091,
3.4817149771311344,
3.4818925822705773,
3.482387495940336,
3.4819980269677044,
3.4818007204998898,
3.4809655287114163,
3.4809938960307734,
3.480278131304537,
3.4811662978024454,
3.4806069511687125,
3.4797754665576623,
3.480914851962163,
3.4808164337381204,
3.4802055933737086,
3.479045213548869,
3.4793670833328654,
3.4800207708784656,
3.4797834703613155,
3.4795769201520304,
3.480078878845765,
3.4802448220883258,
3.4802736197237656,
3.477172784760558,
3.4763753990684223,
3.4752606790891263,
3.4751800735500735,
3.4741463390951783,
3.4744697066205705,
3.4738164157458438,
3.473743304084938,
3.4737810835035208,
3.4731235374528806,
3.473162097657396,
3.4716937792683007,
3.471018966994669,
3.471774654184677,
3.4716470911133386,
3.4723522472038706,
3.4717396076867475,
3.472134863925829,
3.471564129993822,
3.469415378328881,
3.469134498491546,
3.4698382510421912,
3.4693518783326818,
3.4690869461096776,
3.4682074286046007,
3.468017998695512,
3.4681563558735924,
3.4681175881349406,
3.4669165495501364,
3.4674646758775802,
3.4673938278870486,
3.4668185549040405,
3.4659585636572823,
3.4650799175040548,
```

```
3.4656152627360703,
3.4659069181563673,
3.4671297301644497,
3.4678029563030974,
3.4687956418806105,
3.4689804153442334,
3.4699609349679936,
3.470626573044336,
3.471176872493926,
3.471665842531605,
3.47224202762097,
3.4729630084575533,
3.473734493491665,
3.474172804189162,
3.474434700600696,
3.474730723687254,
3.4751819597021516,
3.475539613462423,
3.475388188225852,
3.475260572270077,
3.475727735561789,
3.4761361362712027,
3.474620475253341,
3.473870537338574,
3.473066684268321,
3.4734006729250955,
3.4723764863928444,
3.4723138599603587,
3.472753877226704,
3.4730213165015127,
3.47263227488677,
3.471917049266835,
3.4717923902113514,
3.471608757509922,
3.4717393094054008,
3.471247590742166,
3.4712508095265573,
3.47175990193766,
3.4718046808895693,
3.471669830232896,
3.4711751761790266,
3.471040328247862,
3.471979331749503,
3.472670458278627,
3.4726178595520305,
3.47246503670086,
3.4733435393854024,
3.472998474012939,
3.4720724286178863,
3.4702255813079934,
3.469300973473841,
3.469146438534108,
3.468539522054623,
3.468026458811059,
3.4685047597382956,
3.4688427294174593,
3.4697101868350138,
3.469333272183857,
3.469944721351674,
3.4704065725797424,
```

```
3.4703164233190766,
3.4702589510825157,
3.4692162782891103,
3.469557637762292,
3.4695095620222354,
3.469641446462038,
3.4684093734678045,
3.4684494630953835,
3.46809947352531,
3.4678315124909873,
3.4677572512983197,
3.4670403860820342,
3.466599789200245,
3.4664403436219073,
3.4664563515754225,
3.466335640543758,
3.466133908944024,
3.4658630644466006,
3.466179568746417,
3.466395455396595,
3.4659796475038815,
3.4662892560659535,
3.4666218140700225,
3.4653630126278276,
3.4644122511214577,
3.4641952057530814,
3.4643870922627404,
3.4635235768845276,
3.4633523487301585,
3.462920600600413,
3.463302606361359,
3.4620722899918075,
3.462177513555925,
3.4628996542026624,
3.4629159619597027,
3.4630154738286967,
3.4626527457905545,
3.461720407470517,
3.4612505694600806,
3.4614222648830886,
3.4611555194233334,
3.4608606221977576,
3.461385526840114,
3.4599981436985385,
3.46086102473648,
3.461006950068321,
3.460887315262334,
3.4609013734232823,
3.4614310460729527,
3.462055827945154,
3.4619059412689266,
3.4624787244386295,
3.46295727472275,
3.4629128726258718,
3.4628717451479623,
3.462562742408007,
3.462010098749216,
3.462560297823593,
3.4622256826674485,
3.4609313663785586,
```

```
3.4613220231179542,
3.461085185515563,
3.4611589347782417,
3.459901637462154,
3.4604619116584514,
3.4610374512676643,
3.4613195794243725,
3.4610638304444437,
3.4594606928349685,
3.4596764533644064,
3.4602757724200264,
3.460469110715308,
3.4602372733808315,
3.4603871539671744,
3.460823348329006,
3.4614875871331847,
3.462261813986005,
3.4622866913093047,
3.4630846430920963,
3.4626803407342974,
3.463556500923416,
3.462795637590202,
3.462466293199682,
3.4627110232347267,
3.4610842539079303,
3.4606095607740874,
3.4602880020901705,
3.4607254583089606,
3.4607620657371814,
3.460825907440732,
3.4610695940258482,
3.462213245103036,
3.4613974223709594,
3.4616017731462785,
3.460747697650438,
3.4598412124552507,
3.459411899024097,
3.4588072971215316,
3.4585291155710927,
3.458033886204225,
3.4568991523190142,
3.4555874752755686,
3.4548176843284453,
3.4547980941878023,
3.455183016731606,
3.4548402764774115,
3.454891189356242,
3.4549922610542,
3.4553546701115554,
3.4560327250096456,
3.4551369472451476,
3.4552432959589985,
3.453520365803016,
3.4528000117132183,
3.4528010214249396,
3.4525941473613684,
3.452303972941019,
3.4514986975442765,
3.450670441542252,
3.44899322601897,
```

```
3.4489265175426875,
3.44883196083104,
3.4482310948722015,
3.4483611373266396,
3.447618044433423,
3.445744293832503,
3.445877615081493,
3.4455187190296415,
3.445358632430049,
3.444295887635692,
3.4442147399767835,
3.4437775029213866,
3.4435835151489282,
3.4421750539726164,
3.4421121472301492,
3.442048401360064,
3.44185994156494,
3.441728752708582,
3.442064194762218,
3.441040211941824,
3.4407518279945384,
3.439708893980083,
3.439774214788338,
3.440109956980126,
3.439779692577569,
3.439137195702439,
3.439118948697606,
3.438762889110835,
3.439097912911577,
3.4397982651900754,
3.4392661987875557,
3.438633938917612,
3.4381102155104117,
3.437694679947767,
3.43767219009399,
3.437449427782636,
3.4378401663106923,
3.437136475064922,
3.4364490737780224,
3.435984766252727,
3.4356231933670713,
3.4353996018666817,
3.4360537261552326,
3.436827680541368,
3.4363375816312987,
3.4371368506181215,
3.4372325643743618,
3.4377856296667293,
3.4372804112301294,
3.4377411686919923,
3.438069516569288,
3.438440942147121,
3.4378129873284986,
3.4380466364882323,
3.438492280181581,
3.4380214431657254,
3.43711093660721,
3.43653421891202,
3.436204871087474,
3.43589253646797,
```

```
 3.435550445230379,
 3.4357282071137663,
 3.4352969298096663,
 3.4340673686791585,
 3.43225745430747,
 3.4327510707202897,
 3.4326564675874556,
 3.432381684735265,
 3.4327293102094605,
 3.432752637871984,
 3.432264049504797,
 3.432023394214605,
 3.432634177765745,
 3.4326605629357325,
 3.4321484324438325,
 3.432422870525345,
 3.4330038092453803,
 3.4327634338439705,
 3.4321286609337287,
 3.4325671440401875,
 3.4313183978890875,
 3.4312865054268786,
 3.431232949243025,
 3.4310718662384363,
 3.4304280115519,
 3.430460299725725,
 3.4296864459941556,
 3.4300359536399103,
 3.4296011346776862,
 3.4293692143396277,
 3.4280432964799354,
 3.4266997428496513,
 3.426044317196193,
 3.4257059864844703,
 3.4251106370843183,
 3.4252630592126705,
 3.425839525552477,
 3.4256633336431057,
 3.4249557126885084,
 3.425447423642486,
 ...]
```

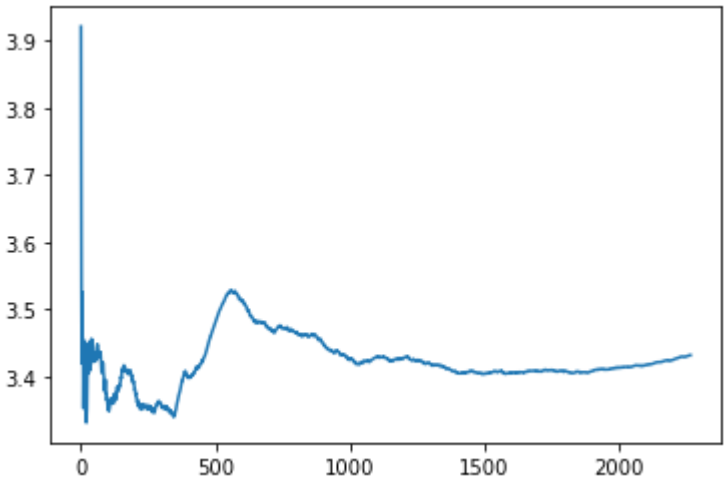
Notamos que existe una oscilación del promedio, entonces graficando el comportamiento:

Lo que da la conclusión que con el tamaño, la muestra es pequeña, siempre tenemos la probabilidad de tomar conclusiones erradas, porque a medida que van creciendo los datos a partir del número 500, í baja un poco, a partir del número 1000, se comienza a estabilizar.

In [46]:
```
import matplotlib.pyplot as plt

promedios = list()

for i in range(1,len(nota_promedio_por_pelicula_con_10_o_mas_votos)):
    promedio = sum(nota_promedio_por_pelicula_con_10_o_mas_votos[0:i]) / i
    promedios.append(promedio)
```

```
promedios

plt.plot(promedios)
```

Out[46]:
```
[<matplotlib.lines.Line2D at 0x21f13eba310>]
```



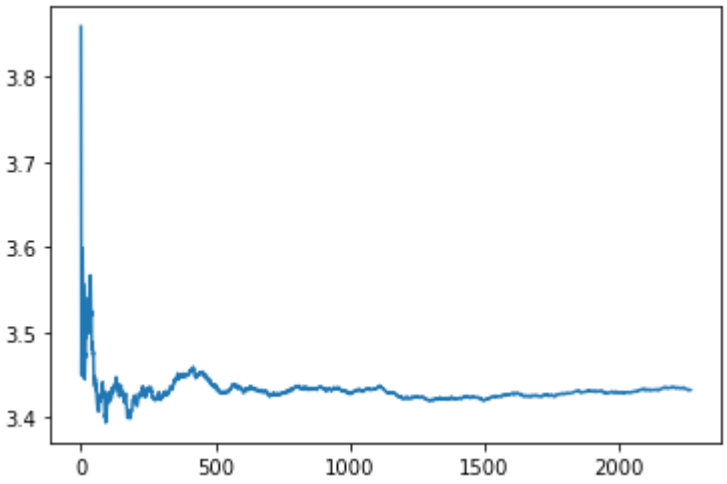
Ordenando aleatoriamente para ver si el comportamiento es el mismo cuando aleatorizamos los datos

A partir del número 500, ya está estabilizado, a diferencia del otro, que es a partir del número 1000. Entonces esto ayuda incluso a ver la importancia del tamaño de los datos, de la gráfica anterior a partir del número 1000 se estabilizó y en este, a partir del número 500.

Aparte del número de muestras, el orden también puede influenciarlo, porque, si con mis datos yo recibí ordenados de menor a mayor y capturo una muestra que no sea aleatoria, la probabilidad de que yo tome decisiones erradas es muy grande. Entonces, aparte de tomar datos aleatorios es importante tomar en cuenta estos detalles como el orden de los datos, por ejemplo, para analizar esa tendencia y consistencia en los datos.

In [49]:
```
import matplotlib.pyplot as plt
import numpy as np

##numpy ayuda a aleatorizar las variables

np.random.seed(75243)

## La función seed() en NumPy permite establecer la semilla del
## generador de números aleatorios para asegurar que los resultados sean reproducibles
temp= nota_promedio_por_pelicula_con_10_o_mas_votos.sample(frac=1)
promedios = list()

for i in range(1,len(temp)):
    promedio = sum(temp[0:i]) / i
    promedios.append(promedio)

promedios

plt.plot(promedios)
```

Out[49]:
```
[<matplotlib.lines.Line2D at 0x21f14f8d100>]
```



Sacando intervalos de confianza para sacar el rango de valores y verificar qué el universo se encuentra en el intervalo de valores

Teniendo como base este promedio:

In [51]:
```
nota_promedio_por_pelicula_con_10_o_mas_votos.mean()
```

Out[51]:
```
3.4320503405352603
```

Analizamos Tests Estadísticos Z test - T Test

Se importa la función zconfint() del módulo para calcular elintervalo de confianza para la media de la muestra, utilizando el método de distribución normal z

el promedio era 3.43 y nos dice que el universo seleccionar aleatoriamente un valor de universo para ver el promedio de las notas,entonces existe un 95% de confianza de que si se selecciona cualquier dato va a estar entre 3.411 y 3.452.

In [53]:
```
from statsmodels.stats.weightstats import zconfint
zconfint(nota_promedio_por_pelicula_con_10_o_mas_votos)
```

Out[53]:
```
(3.4112459477469557, 3.452854733323563)
```

Calculando el T test

In [59]:
```
from statsmodels.stats.weightstats import DescrStatsW
desc_notas_con_mas_de_10_votos= DescrStatsW(nota_promedio_por_pelicula_con_10_o_mas_vo
desc_notas_con_mas_de_10_votos.tconfint_mean()
```

Out[59]:
```
(3.41123483922938, 3.4528658418411386)
```

Se observa una similitud de intervalos del T y Z tests

In [61]:
```
desc_notas_con_mas_de_10_votos.tconfint_mean()
```

Out[61]:
```
(3.41123483922938, 3.4528658418411386)
```

In [62]:
```
desc_notas_con_mas_de_10_votos.zconfint_mean()
```

Out[62]:
```
(3.4112459477469557, 3.452854733323563)
```

Resumiendo:

media= 3.43

int(3.41, 3.45)

Importando para extraer el nuevo dataset de películas registradas y se analiza una película específica: Toy Story

In [63]:
```
peliculas =  pd.read_csv('C:/Users/Carina/Downloads/ml-latest-small/movies.csv')
```

In [64]:
```
peliculas.head()
```

Out[64]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [65]:
```
notas.head()
```

Out[65]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Notas de todos los usuarios que vieron la película de Toy Story

In [67]:
```
notas1 = notas.query("movieId == 1")
notas1
```

Out[67]:

	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
...	...	...	...	...
97364	606	1	2.5	1349082950
98479	607	1	4.0	964744033
98666	608	1	2.5	1117408267
99497	609	1	3.0	847221025
99534	610	1	5.0	1479542900

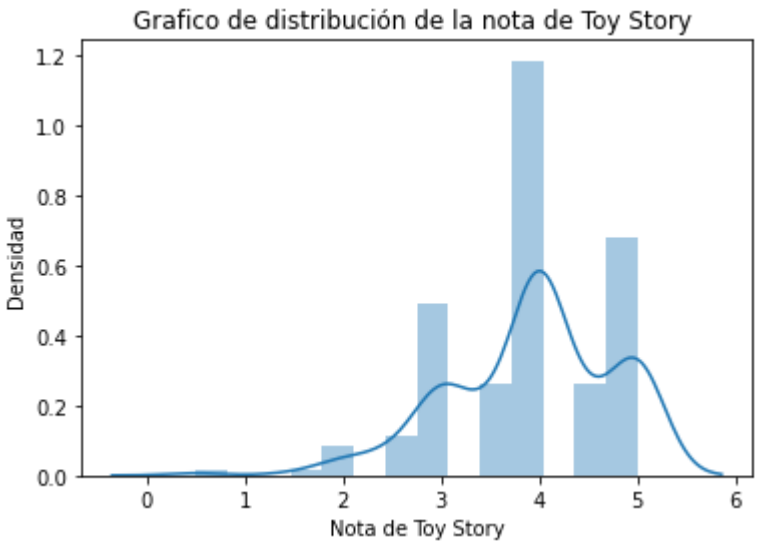
215 rows × 4 columns

Graficando un histograma

In [71]:
```
ax=sb.distplot(notas1.rating)
ax.set(xlabel = "Nota de Toy Story", ylabel= "Densidad")
ax.set_title("Grafico de distribución de la nota de Toy Story ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[71]:
```
Text(0.5, 1.0, 'Grafico de distribución de la nota de Toy Story ')
```



Graficando con boxplot

Se visualiza una mediana de 4, tiene una mediana mayor al de movielens (3.5) vs Toy Story (4)

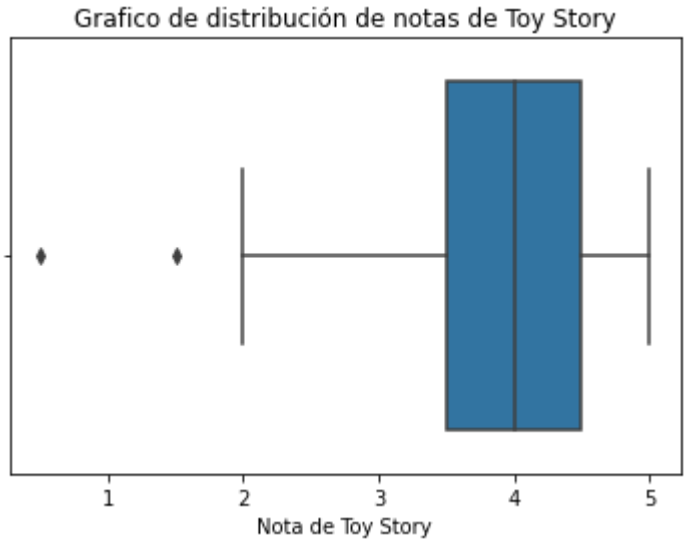
La mediana a la derecha están mas concentrados (mayores votaciones que jalan hacia los votos máximos)

Podemos decir que la nota de Toy Story es mejor que todas las notas (movielens)

In [73]:
```
ax=sb.boxplot(notas1.rating)
ax.set(xlabel = "Nota de Toy Story")
ax.set_title("Grafico de distribución de notas de Toy Story")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[73]:
```
Text(0.5, 1.0, 'Grafico de distribución de notas de Toy Story')
```



Podria exisitir un sesgo, quizá por el tipo de muestra que se hizo, quizá se seleccionaron datos aleatorios y se seleccionaron los datos 'buenos' entonces se realiza un test para comparar y concluir si este resultado es real

In [74]:
```
zconfint(notas1.rating)
```

Out[74]:
```
(3.8093359183563402, 4.032524546759939)
```

Las medias de los promedios de Toy Story con un nivel de confianza del 95% con seleccion aleatoria de las votaciones oscila entre 3.80 a 4.03

Obteniendo resultados enteros

Comparando las notas que fueron dadas a Toy Story con el promedio de todas las notas que ya se había calculado anteriormente con un alpha de 0.05.

ie si la media de notas de Toy Story es igual a la media de todas las películas analizadas

In [77]:
```
from statsmodels.stats.weightstats import ztest
ztest(notas1.rating,nota_promedio_por_pelicula_con_10_o_mas_votos)
```

Out[77]:
```
(12.64176977802645, 1.2422593409480636e-36)
```

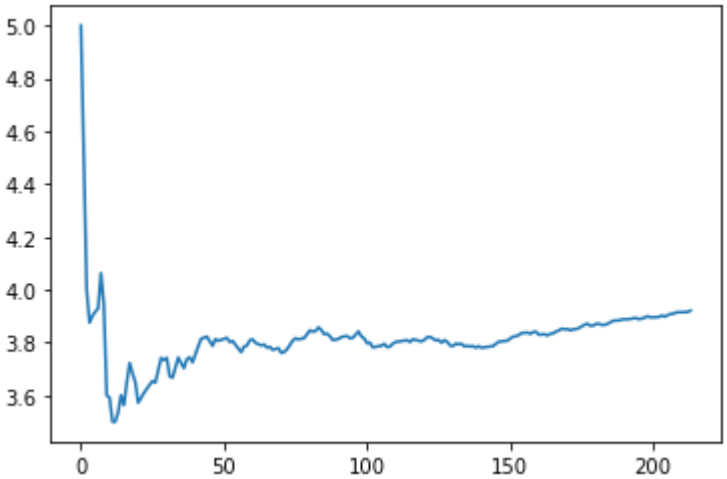
1.2422593409480636e-36 es mucho menor que 0.0.5 (del 5% de error), usando el pvalue menor que el valor de alpha, se verifica que podemos rechazar la hiótesis nula: Decir que este eran iguales.

Se considera entonces que la media de Toy Stroy es mayor que la media de todas las películas en general.

Capturando los datos para ver la gráfica de muestras

In [81]:
```
np.random.seed(75241)

## La función seed() en NumPy permite establecer la semilla del
## generador de números aleatorios para asegurar que los resultados sean reproducibles
temp= notas1.sample(frac=1).rating
promedios = list()

for i in range(1,len(temp)):
    promedio = sum(temp[0:i]) / i
    promedios.append(promedio)

promedios
##graficando:
plt.plot(promedios)
```

Out[81]:
```
[<matplotlib.lines.Line2D at 0x21f1617aeb0>]
```



Comparando usando Z Tests con cada uno de los promedios con diferentes muestras.

Nota: El Test-Z es ideal cuando tenemos una gran cantidad de datos.

Se obtiene una lista con 2 valores, p value y media respectivamente.

In [84]:
```
np.random.seed(75241)

## La función seed() en NumPy permite establecer la semilla del
## generador de números aleatorios para asegurar que los resultados sean reproducibles
temp= notas1.sample(frac=1).rating
promedios = list()

def calcula_test(i):
    media = temp[0:i].mean()
    stat,p = ztest(temp[0:i], value=3.4320503405352603)
    #i con cantidad de registros
    return(i, p, media)
#Calculando desde el 2do porque el primer valor nos devuelve nulo porque no puede calc
for i in range(2,len(temp)):
    promedios.append(calcula_test(i))

promedios
```

Out[84]:
```
[(2, 0.032687621135204896, 4.5),
 (3, 0.3252543510880489, 4.0),
 (4, 0.29952196979944745, 3.875),
 (5, 0.15826781784711086, 3.9),
 (6, 0.07406936464331344, 3.9166666666666665),
 (7, 0.03058372185045264, 3.9285714285714284),
 (8, 0.008546846816661634, 4.0625),
 (9, 0.034351369792154834, 3.9444444444444446),
 (10, 0.6797757440816464, 3.6),
 (11, 0.6661040559180447, 3.590909090909091),
 (12, 0.8452594429449549, 3.5),
 (13, 0.8319722700425317, 3.5),
 (14, 0.7285036535608292, 3.5357142857142856),
 (15, 0.5561564945645507, 3.6),
 (16, 0.6284166359766319, 3.5625),
 (17, 0.42056196093078513, 3.6470588235294117),
 (18, 0.2692718087435868, 3.7222222222222223),
 (19, 0.3157395629293568, 3.6842105263157894),
 (20, 0.36558117183067695, 3.65),
 (21, 0.5650170364597873, 3.5714285714285716),
 (22, 0.4930885087384419, 3.590909090909091),
 (23, 0.4265824656611449, 3.608695652173913),
 (24, 0.36578337526912086, 3.625),
 (25, 0.3108287352909964, 3.64),
 (26, 0.26171758950670954, 3.6538461538461537),
 (27, 0.2559999054486449, 3.6481481481481484),
 (28, 0.16314137427171593, 3.6964285714285716),
 (29, 0.10054986793464257, 3.7413793103448274),
 (30, 0.09812718831667805, 3.7333333333333334),
 (31, 0.07894877889458934, 3.7419354838709675),
 (32, 0.19389049416761317, 3.671875),
 (33, 0.18994435269346233, 3.6666666666666665),
 (34, 0.12400478766945784, 3.7058823529411766),
 (35, 0.07870821336871098, 3.742857142857143),
 (36, 0.09353576061527746, 3.7222222222222223),
 (37, 0.11013590254134484, 3.7027027027027026),
 (38, 0.070293980214127, 3.736842105263158),
 (39, 0.057719600728433226, 3.7435897435897436),
 (40, 0.06895669959313765, 3.725),
 (41, 0.043027696867534854, 3.7560975609756095),
 (42, 0.026192682251694282, 3.7857142857142856),
 (43, 0.015559932099710492, 3.813953488372093),
 (44, 0.01233272538941352, 3.8181818181818183),
 (45, 0.00971325069982623, 3.8222222222222224),
 (46, 0.012260404094830826, 3.8043478260869565),
 (47, 0.015299059283941084, 3.7872340425531914),
 (48, 0.008966188879749698, 3.8125),
 (49, 0.008766884002523942, 3.806122448979592),
 (50, 0.006897537294395224, 3.81),
 (51, 0.005394516156167565, 3.8137254901960786),
 (52, 0.004193844231736312, 3.8173076923076925),
 (53, 0.005385212289644282, 3.80188679245283),
 (54, 0.004195763368449785, 3.8055555555555554),
 (55, 0.005368352947838337, 3.790909090909091),
 (56, 0.006794205663451532, 3.7767857142857144),
 (57, 0.008512220201001898, 3.763157894736842),
 (58, 0.004975558567317606, 3.7844827586206895),
 (59, 0.003904050433428437, 3.788135593220339),
 (60, 0.0022169155153936113, 3.808333333333333),
 (61, 0.0017134856636438817, 3.8114754098360657),
```

```
(62, 0.0022185468321933607, 3.7983870967741935),
(63, 0.0021667091142656882, 3.7936507936507935),
(64, 0.002115742986059037, 3.7890625),
(65, 0.00163816968831084, 3.792307692307692),
(66, 0.002112904717074153, 3.7803030303030303),
(67, 0.0016391796508841768, 3.783582089552239),
(68, 0.002107877006211813, 3.7720588235294117),
(69, 0.00163820800693732, 3.7753623188405796),
(70, 0.0012667046775475992, 3.7785714285714285),
(71, 0.0022426047709492537, 3.76056338028169),
(72, 0.0017525084760790894, 3.763888888888889),
(73, 0.0011365250751412, 3.7739726027397262),
(74, 0.0006361085905667348, 3.7905405405405403),
(75, 0.00035042445221225517, 3.8066666666666666),
(76, 0.00021841519783205381, 3.8157894736842106),
(77, 0.0002133389727594162, 3.811688311688312),
(78, 0.0001607423772756106, 3.8141025641025643),
(79, 0.00012053616176043722, 3.8164556962025316),
(80, 6.355135417245393e-05, 3.83125),
(81, 3.300722928041108e-05, 3.845679012345679),
(82, 3.2304003745802037e-05, 3.841463414634146),
(83, 2.374556884407959e-05, 3.8433734939759034),
(84, 1.2077073301628071e-05, 3.857142857142857),
(85, 1.7088673225760976e-05, 3.847058823529412),
(86, 3.608185615666001e-05, 3.8313953488372094),
(87, 2.681257612846113e-05, 3.8333333333333335),
(88, 3.681562512354262e-05, 3.8238636363636362),
(89, 7.313870608516293e-05, 3.808988764044944),
(90, 5.525606847923858e-05, 3.811111111111111),
(91, 4.1572170968967595e-05, 3.8131868131868134),
(92, 2.540508532221419e-05, 3.8206521739130435),
(93, 1.891161316381205e-05, 3.8225806451612905),
(94, 1.4019284811900051e-05, 3.824468085106383),
(95, 1.9300245184706242e-05, 3.8157894736842106),
(96, 1.4337154349835086e-05, 3.8177083333333335),
(97, 7.470440358184904e-06, 3.829896907216495),
(98, 3.843679820388642e-06, 3.8418367346938775),
(99, 1.3101881021708747e-05, 3.823232323232323),
(100, 1.773810320525232e-05, 3.815),
(101, 5.156319239792069e-05, 3.797029702970297),
(102, 3.956916550250529e-05, 3.799019607843137),
(103, 0.00010532490098882295, 3.7815533980582523),
(104, 8.207340957567451e-05, 3.7836538461538463),
(105, 6.373666797007717e-05, 3.7857142857142856),
(106, 4.9327620681370155e-05, 3.7877358490566038),
(107, 3.158510492524062e-05, 3.794392523364486),
(108, 5.741195652105531e-05, 3.7824074074074074),
(109, 4.452063805181686e-05, 3.7844036697247705),
(110, 2.5034997999585643e-05, 3.7954545454545454),
(111, 1.593383648900579e-05, 3.8018018018018016),
(112, 1.2181676067708934e-05, 3.8035714285714284),
(113, 9.282385148662153e-06, 3.8053097345132745),
(114, 7.049800097807919e-06, 3.807017543859649),
(115, 5.336512954125956e-06, 3.8086956521739133),
(116, 7.132420343988585e-06, 3.8017241379310347),
(117, 3.8777239631651985e-06, 3.8119658119658117),
(118, 3.789024541840527e-06, 3.809322033898305),
(119, 3.701652394675697e-06, 3.80672268907563),
(120, 3.6156244484077226e-06, 3.8041666666666667),
(121, 2.234409015059164e-06, 3.809917355371901),
```

```
(122, 1.1935921381583365e-06, 3.819672131147541),
(123, 8.907771906449635e-07, 3.821138211382114),
(124, 1.2165528009735344e-06, 3.814516129032258),
(125, 1.6492029231809994e-06, 3.808),
(126, 1.2383902023580329e-06, 3.8095238095238093),
(127, 2.4447090501870245e-06, 3.7992125984251968),
(128, 1.3259626346058774e-06, 3.80859375),
(129, 1.7799716841358229e-06, 3.802325581395349),
(130, 5.106740278151876e-06, 3.7884615384615383),
(131, 4.986913728968981e-06, 3.786259541984733),
(132, 2.779940274691362 7e-06, 3.7954545454545454),
(133, 2.715247214619515e-06, 3.793233082706767),
(134, 2.075872794199074e-06, 3.794776119402985),
(135, 3.869818273690176e-06, 3.785185185185185),
(136, 2.9771501112494567e-06, 3.786764705882353),
(137, 2.906855295169462e-06, 3.7846715328467155),
(138, 2.229978257172786e-06, 3.786231884057971),
(139, 2.923020543496903e-06, 3.7805755395683454),
(140, 1.861736987479408e-06, 3.7857142857142856),
(141, 2.441850783063681e-06, 3.780141843971631),
(142, 1.8743496554091788e-06, 3.7816901408450705),
(143, 1.4348129269825041e-06, 3.7832167832167833),
(144, 1.0953489095524478e-06, 3.7847222222222223),
(145, 8.33914593413763e-07, 3.786206896551724),
(146, 4.565410010534 6796e-07, 3.7945205479452055),
(147, 2.4781068087348057e-07, 3.802721088435374),
(148, 1.8647631467675183e-07, 3.804054054054054),
(149, 1.3994497318116365e-07, 3.8053691275167787),
(150, 1.0474187906210276e-07, 3.8066666666666666),
(151, 5.5638738167742754e-08, 3.814569536423841),
(152, 2.9310557354650833e-08, 3.8223684210526314),
(153, 2.169106984430368e-08, 3.823529411764706),
(154, 1.296533237512879e-08, 3.8279220779220777),
(155, 6.69721358751862e-09, 3.835483870967742),
(156, 4.90242789582791e-09, 3.8365384615384617),
(157, 3.579198670373318e-09, 3.837579617834395),
(158, 5.065038279328382e-09, 3.8322784810126582),
(159, 2.593122661980238e-09, 3.839622641509434),
(160, 1.888254008877338e-09, 3.840625),
(161, 6.89862648428305e-09, 3.829192546583851),
(162, 5.088654193995244e-09, 3.830246913580247),
(163, 3.744165106538347e-09, 3.831288343558282),
(164, 5.23409616911476e-09, 3.826219512195122),
(165, 2.715045381204715e-09, 3.8333333333333335),
(166, 1.9924856311040464e-09, 3.834337349397 5905),
(167, 1.021658248002295e-09, 3.841317365269461),
(168, 5.986014433616568e-10, 3.8452380952380953),
(169, 3.026046512497991e-10, 3.8520710059171597),
(170, 2.9667891226340393e-10, 3.85),
(171, 2.1473424779915278e-10, 3.8508771929824563),
(172, 3.087411923363227e-10, 3.8459302325581395),
(173, 1.7917893544489147e-10, 3.8497109826589595),
(174, 1.2939702905988413e-10, 3.8505747126436782),
(175, 7.437375301814674e-11, 3.854285714285714),
(176, 3.68536489501502e-11, 3.8607954545454546),
(177, 1.8128994549484078e-11, 3.867231638418079),
(178, 1.0226884113071301e-11, 3.8707865168539324),
(179, 2.476407211840968e-11, 3.863128491620112),
(180, 1.7738569534234892e-11, 3.863888888888889),
(181, 8.709233016873075e-12, 3.8701657458563536),
```

```
 (182, 6.201886620364462e-12, 3.870879120879121),
 (183, 9.186029818083115e-12, 3.866120218579235),
 (184, 6.5513651601619266e-12, 3.8668478260869565),
 (185, 3.688482306426001e-12, 3.8702702702702703),
 (186, 1.787257905280176e-12, 3.8763440860215055),
 (187, 8.600037830948384e-13, 3.8823529411764706),
 (188, 6.056150219694947e-13, 3.882978723404255),
 (189, 4.2551292144889585e-13, 3.8835978835978837),
 (190, 2.3370424038676717e-13, 3.886842105263158),
 (191, 1.6326985261947053e-13, 3.887434554973822),
 (192, 1.1380629973295943e-13, 3.8880208333333335),
 (193, 7.91493424125946e-14, 3.88860103626943),
 (194, 4.278258438283099e-14, 3.8917525773195876),
 (195, 2.958555465053198e-14, 3.8923076923076922),
 (196, 4.623476152843946e-14, 3.8877551020408165),
 (197, 2.493280074064741e-14, 3.8908629441624365),
 (198, 1.3370684043107038e-14, 3.893939393939394),
 (199, 6.160613753635314e-15, 3.899497487437186),
 (200, 9.749074156510373e-15, 3.895),
 (201, 6.7127681104016146e-15, 3.8955223880597014),
 (202, 4.6119520070165714e-15, 3.896039603960396),
 (203, 3.1616414709692205e-15, 3.896551724137931),
 (204, 1.443848995257856e-15, 3.9019607843137254),
 (205, 2.309845735394206e-15, 3.897560975609756),
 (206, 1.0555998801123747e-15, 3.9029126213592233),
 (207, 4.793088683620974e-16, 3.9082125603864735),
 (208, 3.2609910418318076e-16, 3.9086538461538463),
 (209, 1.466694033719233e-16, 3.9138755980861246),
 (210, 9.927030087378508e-17, 3.914285714285714),
 (211, 6.704625749290763e-17, 3.914691943127962),
 (212, 4.518611521397103e-17, 3.9150943396226414),
 (213, 3.038859398683116e-17, 3.915492957746479),
 (214, 1.3418325209316871e-17, 3.9205607476635516)]
```

In [86]:
```
valores = np.array(promedios)
```

Graficando se atribuyen Arrays

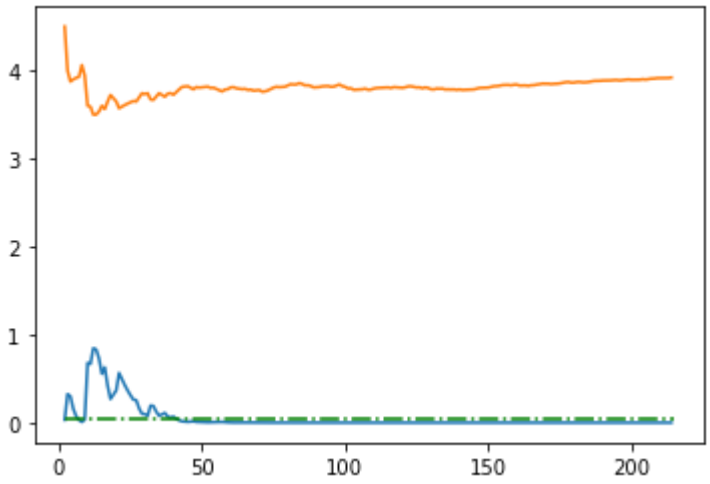
Amarillo: promedio, azul: p-value

Promedio: En los primeros 50 registros, existe oscilación. Esa oscilación es bastante fuerte de valores de los promedios de las medias, de las notas de Toy Story.

P-value: también muestra la oscilación en el P value, en los primeros 50 valores.

In [92]:
```
##graficando:
#Todos las lineas(i), y 1: la clumna del p-value
#posicion 0 de la primer columna
plt.plot(valores[:,0],valores[:,1])
#Eje horizontal:
#columna 2 en el eje vertical y empieza desde la posicion 0
plt.plot(valores[:,0],valores[:,2])
## Graficando una línea constante con plt.hlines.
## Para ver los parámetros, se grafica una línea constante en el valor de 0.05, que es
plt.hlines(y=0.05, xmin=2, xmax=len(temp), colors="g", linestyles="dashdot")
```

Out[92]: <matplotlib.collections.LineCollection at 0x21f19e90460>



Con la línea verde, para diferenciarla se hizo ese estilo y lo que se estaba diciendo, antes se podría decir que en los primeros 40 valores tenemos p values encima de nuestro alpha.

¿Cuál es el riesgo de si se tuviera una muestra pequeña? Hacer el Z test, se iba a aceptar en este caso la hipótesis nula: el valor de la media es igual, el valor de la media de Toy Story, las notas de Toy Story, es igual al valor de las medias en general. Entonces ese es el riesgo de tener muestras pequeñas donde podríamos tomar decisiones erradas.

Y en el caso de que dudar de los valores y no presenten una consistencia como en este caso, se rechaza la hipótesis nula, ya sea por falta de datos o por tener una muestra, en realidad por inconsistencia en los resultados o por tener una muestra pequeña de datos.

Comparando la distribución de un conjunto de datos con la distribución de otro conjunto de datos: notas de Toy Story con el de todos

Hay un intervalo de confianza de 0.2799 a 0.5588 de que las notas 1, en este caso las notas de Toy Story son mayores en ese rango que las notas en general.

In [93]:
```
zconfint(notas1.rating, notas.rating)
```

Out[93]: (0.2799245129192442, 0.5588219849631111)

Aplicando Z test para ver el P value

Hipótesis nula: La media de la distribución es igual a la media de la otra distribución.

Se rechaza la hipótesis nula porque el P value (3.76) es menor que el valor de Alpha (0.05)

In [94]:
```
ztest(notas1.rating, notas.rating)
```

Out[94]:
```
(5.894327101501841, 3.762112778881965e-09)
```

Comparando notas generales vs Toy Story.

Da valores negativos porque dice que la media de esos valores de esta distribución es menor, muestra negativo, entonces es menor que Toy Story en este rango de -0,56 a -0,28.

In [95]:
```
ztest(notas.rating, notas1.rating)
```

Out[95]:
```
(-5.894327101501841, 3.762112778881965e-09)
```

Visualizando el test P con el modulo de scipy

Tenemos un valor parecido al de ztest donde descartamos la hipótesis de que las medias de las distribuciones no son iguales.

In [96]:
```
from scipy.stats import ttest_ind
ttest_ind(notas1.rating, notas.rating)
```

Out[96]:
```
Ttest_indResult(statistic=5.894327101501841, pvalue=3.774003138720876e-09)
```

Importando la biblioteca de statsmodels y usar el DescrStatsW.

In [100…
```
from statsmodels.stats.weightstats import DescrStatsW

desc_todos_las_notas = DescrStatsW(notas.rating)

desc_toy_story = DescrStatsW(notas1.rating)
# comparar esos dos conjuntos de datos.
comparacion= desc_todos_las_notas.get_compare(desc_toy_story)
# con parametro para que no arroje el Z test
comparacion.summary(use_t=True)
```

Out[100]:

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

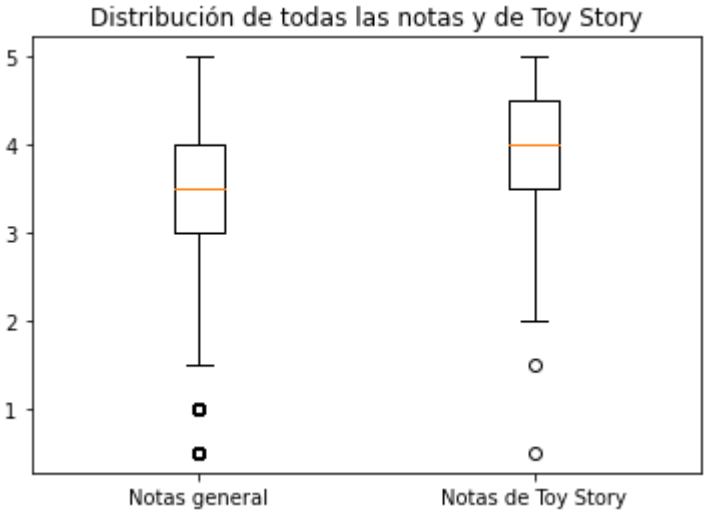
El P value se rechaza por ser menor que el alpha.

Comparando con gráficos para visualizar y mejorar su análisis: graficando un gráfico de boxplot con las dos distribuciones

Se ve fácilmente que la media de las notas de Toy Story es mayor que la media de las notas generales, pero existe una debilidad: no se pueden ver las notas generales de muestras

In [102…
```
import matplotlib.pyplot as plt
plt.boxplot([notas.rating, notas1.rating], labels= ["Notas general", "Notas de Toy Sto
plt.title ("Distribución de todas las notas y de Toy Story")
```

Out[102]:
```
Text(0.5, 1.0, 'Distribución de todas las notas y de Toy Story')
```

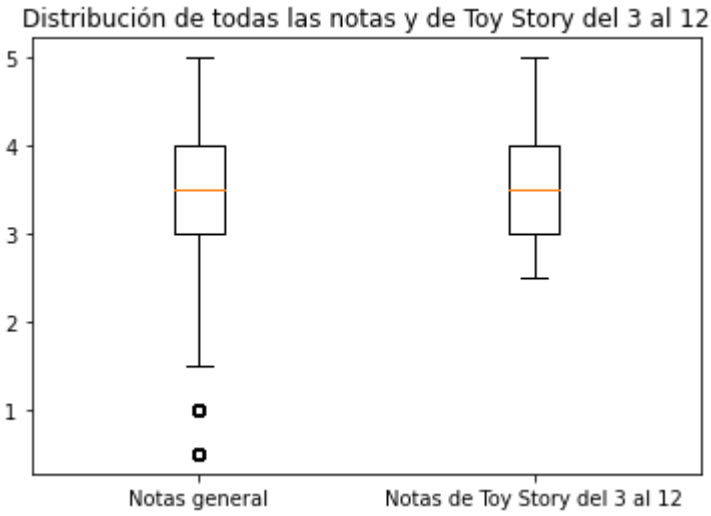


Suponiendo que existieran pocos datos deToy Story

Existen distribuciones parecidas con medias promedio bien parecidos: El promedio general de Toy Story es igual al general (Conclusión incorrecta)

In [105…
```
import matplotlib.pyplot as plt
plt.boxplot([notas.rating, notas1[3:12].rating], labels= ["Notas general", "Notas de T
plt.title ("Distribución de todas las notas y de Toy Story del 3 al 12")
```

Out[105]:
```
Text(0.5, 1.0, 'Distribución de todas las notas y de Toy Story del 3 al 12')
```



Comparando los dos conjuntos de datos

El p value es mayor que el alpha (0.05), se rechaza la hipótesis nula: de que los valores son iguales.

El intervalo de confianza es inconsistente, el primer valor es negativo (-0.735) y el segundo positivo (0.627)

Podemos decir que no existe una conclusión acertiva por falta de datos

In [108…
```
desc_todos_las_notas = DescrStatsW(notas.rating)

desc_toy_story_3_12 = DescrStatsW(notas1[3:12].rating)
# comparar esos dos conjuntos de datos.
comparacion= desc_todos_las_notas.get_compare(desc_toy_story_3_12)
# con parametro para que no arroje el Z test
# con estadístico T que aplica a pocas muestras, es recomendable
comparacion.summary(use_t=True)
```

Out[108]:

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

Si fuera el caso:

Con la hipótesis: El p value es menor que 0.05

Entonces se aceptaría la hipótesis nula y el intervalo de confianza sería de 0.01 y 0.02

Analizando 3 películas específicas:

In [109…
```
peliculas.query("movieId in (1,593, 72226)")
```

Out[109]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
510	593	Silence of the Lambs, The (1991)	Crime|Horror|Thriller
7180	72226	Fantastic Mr. Fox (2009)	Adventure|Animation|Children|Comedy|Crime

Asignando a cada una su propia variable

In [111…
```
notas1= notas.query("movieId ==1")
notas593= notas.query("movieId ==593")
notas72226= notas.query("movieId ==72226")
```

In [112…
```
notas1.head()
```

Out[112]:

	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

Usando boxplot para entender mejor los datos

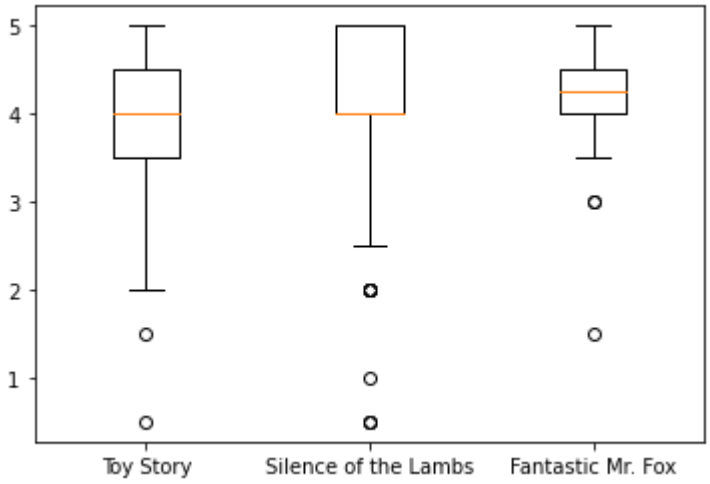
La mediana para the Silence of the Lambs y Toy Story es 4 para ambos; Fantastic Mr. Fox parecer tener mejor valoracion

In [115…

```
import matplotlib.pyplot as plt

plt.boxplot([notas1.rating, notas593.rating, notas72226.rating],
            labels= ["Toy Story", "Silence of the Lambs", "Fantastic Mr. Fox"])
```

Out[115]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x21f1a1bd130>,
  <matplotlib.lines.Line2D at 0x21f1a1bd400>,
  <matplotlib.lines.Line2D at 0x21f1a1ca550>,
  <matplotlib.lines.Line2D at 0x21f1a1ca820>,
  <matplotlib.lines.Line2D at 0x21f1a1d6940>,
  <matplotlib.lines.Line2D at 0x21f1a1d6c10>],
 'caps': [<matplotlib.lines.Line2D at 0x21f1a1bd700>,
  <matplotlib.lines.Line2D at 0x21f1a1bd9d0>,
  <matplotlib.lines.Line2D at 0x21f1a1caaf0>,
  <matplotlib.lines.Line2D at 0x21f1a1cadc0>,
  <matplotlib.lines.Line2D at 0x21f1a1d6ee0>,
  <matplotlib.lines.Line2D at 0x21f1a1e41f0>],
 'boxes': [<matplotlib.lines.Line2D at 0x21f1a1ace20>,
  <matplotlib.lines.Line2D at 0x21f1a1ca280>,
  <matplotlib.lines.Line2D at 0x21f1a1d6670>],
 'medians': [<matplotlib.lines.Line2D at 0x21f1a1bdca0>,
  <matplotlib.lines.Line2D at 0x21f1a1d60d0>,
  <matplotlib.lines.Line2D at 0x21f1a1e44c0>],
 'fliers': [<matplotlib.lines.Line2D at 0x21f1a1bdf70>,
  <matplotlib.lines.Line2D at 0x21f1a1d63a0>,
  <matplotlib.lines.Line2D at 0x21f1a1e4790>],
 'means': []}
```

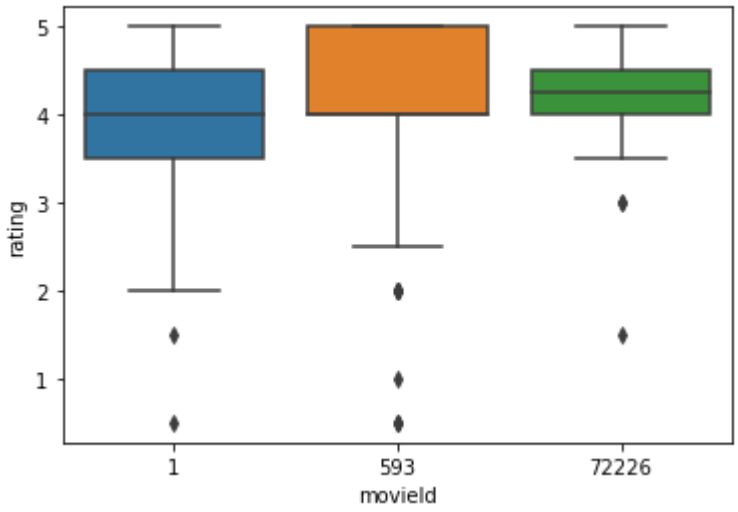


Aplicando query o describe para visualizar mejor los datos

La película 1 en este caso es Toy Story, la película 593, que es The Silence of Lambs, tienen la misma mediana. Parece que la Silencie of Lambs está un poco encima, pero la que tiene una gran diferencia en la mediana y una distribución más cerrada es Fantastic Mr. Fox.

In [117…
```
import seaborn as sb
#seaborn da una gráfica un poco más formateado
sb.boxplot(x="movieId", y= "rating",data= notas.query("movieId in(1,593,72226)") )
```

Out[117]:
```
<AxesSubplot:xlabel='movieId', ylabel='rating'>
```



Usando una query para saber la cantidad

Hay 215 muestras en "Toy Story", 279 en Silence of the Lambs", y 18 en "Fantastic Mr. Fox"

Son pocas muestras comparando entre ellas, no podemos concluir que "Fantastic Mr. Fox" es mejor que las demás

In [118…
```
notas.query("movieId in(1,593,72226)").groupby("movieId").count()
```

Out[118]:

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

Comparando películas entre si para ver cual es mejor

Se descarta la hipótesis nula: Que son iguales

Toy Story estaba evaluada, tiene notas menores en un rango de -0.39 a -0.09. Podemos redondearlo de -0.4 a -0.1.

Exactamente, es -0,4 a -0,1. Es un valor estadístico representativo entonces podemos concluir de que en este caso, Silence of Lambs está mejor evaluada que Toy Story.

In [120…

```
#1 vs 593

from statsmodels.stats.weightstats import DescrStatsW


desc_1 = DescrStatsW(notas1.rating)
desc_593= DescrStatsW(notas593.rating)

# comparar esos dos conjuntos de datos.
comparacion= desc_1.get_compare(desc_593)
# sin parametro para que arroje el Z test, default
comparacion.summary()
```

Out[120]:

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

Repitiendo el ejercicio, ahora se va a comparar la primera con la tercera, que es 72226.

Se rechaza la hipótesis nula, el P value es mayor que 0-05; además el rango nuevamente es inconsistente, uno negativo y positivo respectivamente.

Los datos no son suficientes para concluir.

In [123…

```
#1 vs 72226

from statsmodels.stats.weightstats import DescrStatsW


desc_1 = DescrStatsW(notas1.rating)
desc_72226= DescrStatsW(notas72226.rating)
```

```
# comparar esos dos conjuntos de datos.
comparacion= desc_1.get_compare(desc_72226)
# con parametro para que no arroje el Z test
comparacion.summary(use_t=True)
```

Out[123]:

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

Repitiendo el ejercicio, ahora se va a comparar 593 con la tercera, que es 72226.

Se rechaza la hipótesis nula, el P value es mayor que 0-05; además el rango nuevamente es inconsistente, uno negativo y positivo respectivamente.

In [125…

```
#593 vs 72226

# comparar esos dos conjuntos de datos.
comparacion= desc_593.get_compare(desc_72226)
# con parametro para que no arroje el Z test
comparacion.summary(use_t=True)
```

Out[125]:

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	-0.332	0.488

Para el caso que los datos no tengan distribución normal:

El gráfico tiene una cola más inclinada hacia la izquierda. O sea el máximo, la nota máxima está más próxima de la media que la nota mínima.

Concluimos que las personas normalmente dan notas buenas a las películas, buenas apreciaciones,

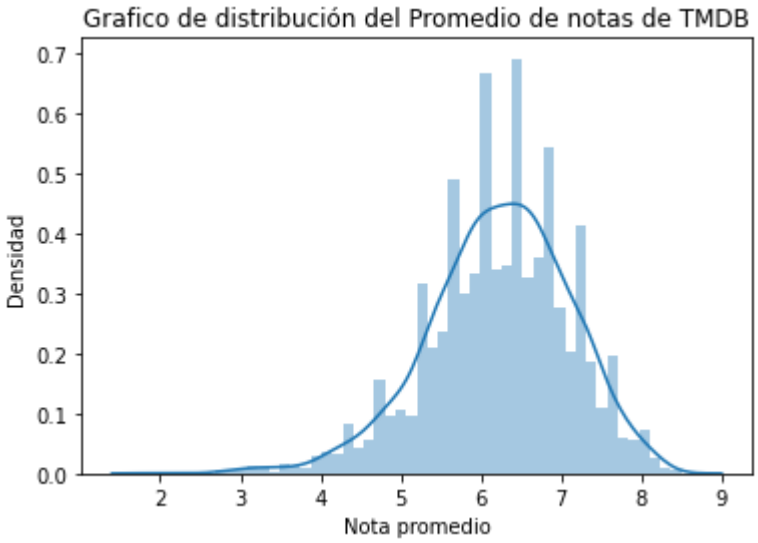
In [126…

```
ax= sb.distplot(tmdb_10_o_mas_votos.vote_average)
ax.set(xlabel = "Nota promedio", ylabel = "Densidad")
ax.set_title("Grafico de distribución del Promedio de notas de TMDB")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[126]:

```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de TMDB')
```

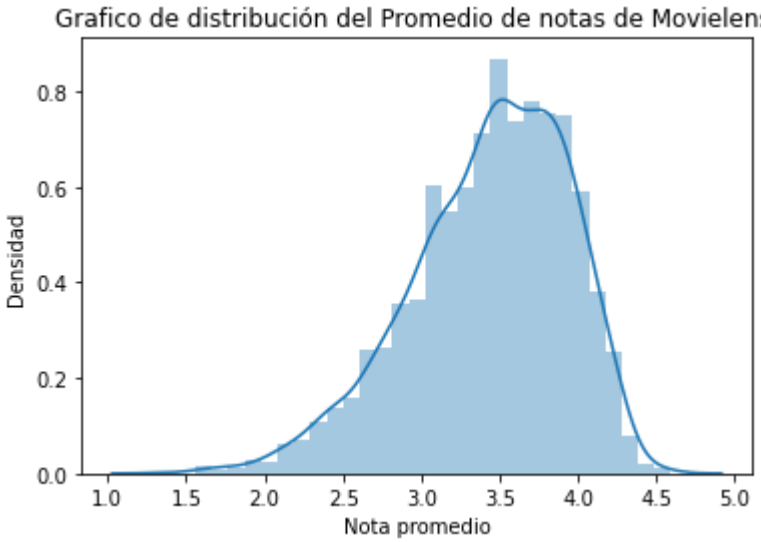


La media parece estar en 3.5, la nota máxima está cerca de la curva centuada hacia la izquierda; la nota máxima está mas cerca de la mediana que de la nota minima.

In [127…
```
ax=sb.distplot(nota_promedio_por_pelicula_con_10_o_mas_votos.values)
ax.set(xlabel = "Nota promedio", ylabel= "Densidad")
ax.set_title("Grafico de distribución del Promedio de notas de Movielens ")
```

```
C:\Users\Carina\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[127]:
```
Text(0.5, 1.0, 'Grafico de distribución del Promedio de notas de Movielens ')
```



Existe un test para saber si son normales

In [129…
```
from scipy.stats import normaltest
```

Se rechaza la hipótesis nula: La distribución es normal

ie: se acepta la hipótesis no nula

In [131…
```
#devuelve dos variables: valor de p y valor de estadítico
stat,p= normaltest(notas1.rating)
p
```

Out[131]:
```
0.00011053430732728716
```

Test para datos que no son normales, se analizan los datos, se comparan las distribuciones: test de Wilcoxon

In [132…
```
from scipy.stats import ranksums
```

El valor de P es menor que 0.05 entonces se rechaza la hipótesis nula.

La hipótesis nula: Las dos muestras tienen la misma distribución

La hipótesis alternativa: Los valores de una muestra tiende a ser mayores que la segunda u otra

In [133…
```
stat,p= ranksums(notas1.rating, notas593.rating)
p
```

Out[133]:
```
0.0003267718756440693
```

In [ ]:
```

```