from sklearn.linear\_model import Lasso

model = Lasso()

model.fit(X\_train, Y\_train)

Si queremos ver los coeficientes de nuestro modelo, corremos lo siguiente

**model.coef\_**

array([ 1.70983355e+04, 1.41630464e+00, 1.01314290e+06, -5.31928800e+06, 1.42504819e+05, -1.51646539e+02, 3.03478056e-03, 5.73593355e+06, 1.98219286e+00])

Nota: Es una convención en Scikit-Learn que los atributos de los estimadores se identifican con el guión abajo al final.

Como vemos, los coeficientes tienen una gran diferencia. Entonces, si queremos visualizarlos en un gráfico debemos de "estandarizarlo"

np.log(np.abs(model.coef\_))

#Sin embargo, es mejor obtener solo los valores enteros.

np.floor(np.log(np.abs(model.coef\_)))

array([ 9., 0., 13., 15., 11., 5., -6., 15., 0.])

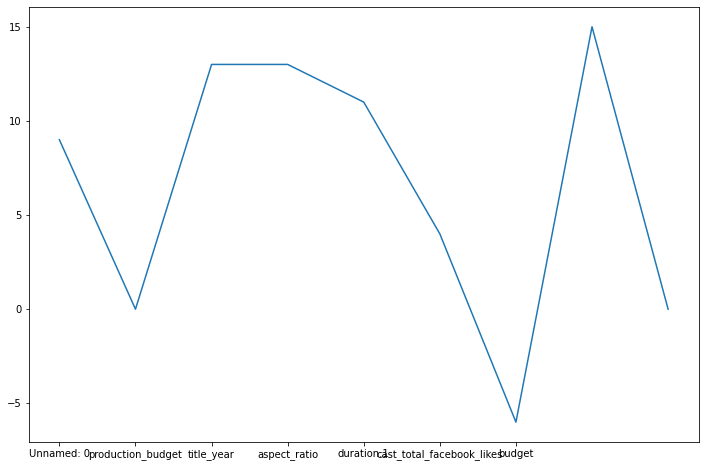
Dado lo anterior, ya somos capaces de visualizar mejor qué features son más relevantes y qué features se pueden suprimir para reducir la dimensionalidad

var = np.floor(np.log(np.abs(model.coef\_)))

plt.rcParams['figure.figsize']= [12, 8]

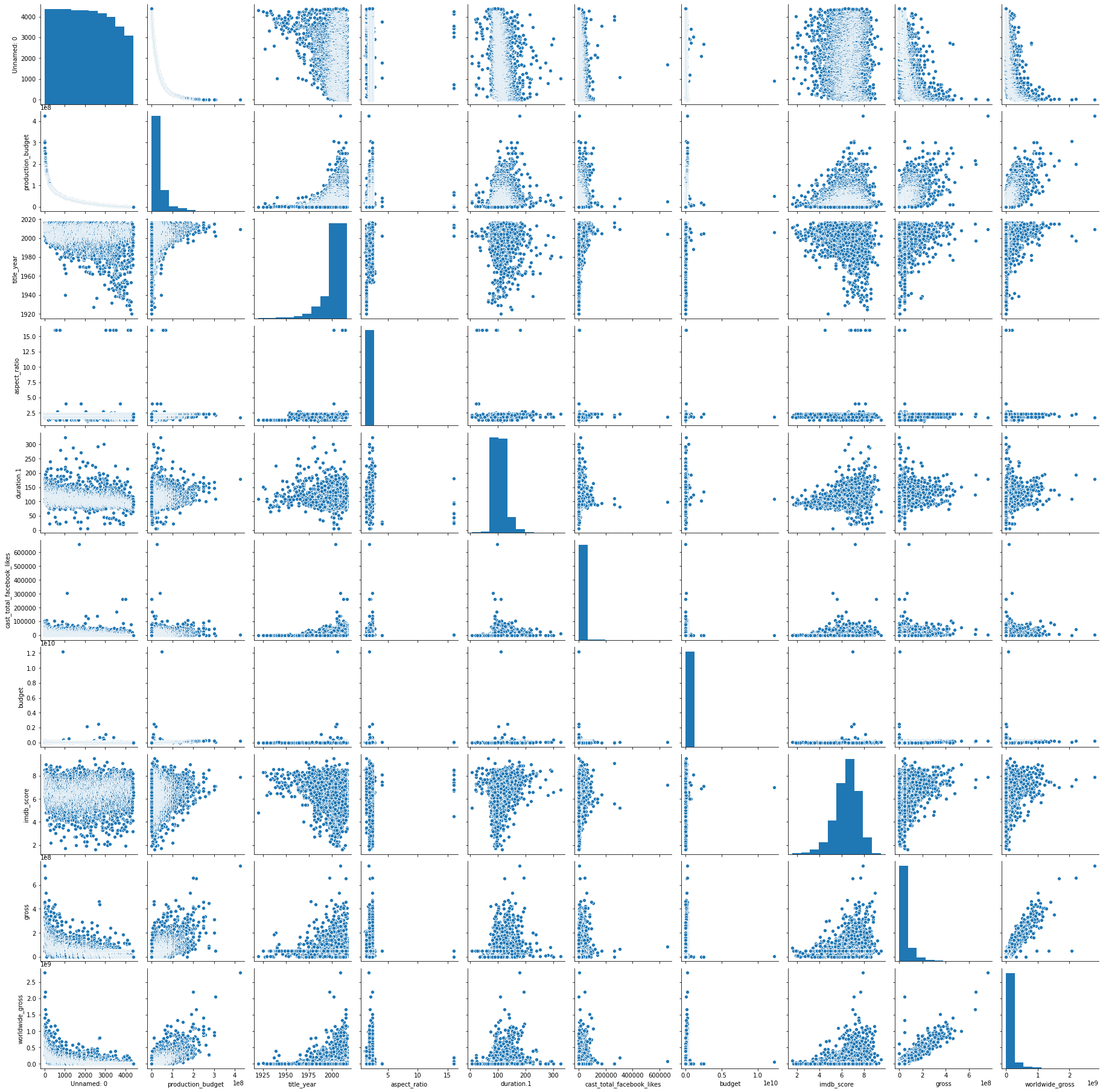
plt.plot(var)

plt.xticks(np.arange(7), list(X.columns));



También podemos usar técnicas de visualización, las cuales nos permiten entender la correlación entre las variables mismas y la correlación entre cada feature con la variable objetivo.

Si es que una feature tiene una gran correlación con nuestra variable objetivo quiere decir que es una feature informativa



En las gráfica de la diagonal, veremos la distribución de cada variable Ejm : “Cuantas películas hay de 10M$, cuantas de 20M$ ,etc. “Por otro lado, en las otras gráficas veremos las correlaciones entre variables.

Si en nuestro “scatterplot” obtuvieramos una diagonal perfecta, ello nos estaría mostrando que hay una correlación perfecta entre esas 2 variables de la gráfica.Donde nos interesa ver esto es en las gráficas en las que relaciona a los features con nuestra variable objetivo.Esto lo podemos ver en nuestra última línea,la cual es la variable ““worldwide\_gross “

Sin embargo, ver nuestros datos en un solo color no es una visualización óptima. Para ello, realizaremos lo siguiente:

Primero, partimos el rango de nuestros datos.

pd.cut(X['production\_budget'], 8)

#Lo que pasó es que partió todo el rango de valores de "production\_budget" en 8 pedazos de similar tamaño

**[ (-423898.9, 53125962.5] < (53125962.5, 106250825.0] < (106250825.0, 159375687.5] < (159375687.5, 212500550.0] < (212500550.0, 265625412.5] < (265625412.5, 318750275.0] < (318750275.0, 371875137.5] < (371875137.5, 425000000.0]]**

**OJO:**

Con este método no solo partió el rango en 8 intervalos iguales, sino también le asignó a cada elemento de la columna 'production\_budget' su intervalo respectivo.

pd.cut(X['production\_budget'], 8).cat.codes

#Al aplicarle "cad.codes" lo que estamos haciendo es categorizar los rangos nuevos que tenemos y representarlos con un número( del 0 hasta el 7).Luego, le asignamos a cada dato la categoría a la que pertenece

0 7

1 5

..

4099 0

4100 0

**Ahora** lo que haremos será cambiar el nombre de nuestra serie a “ class “ para poder reconocerlo y concatenarlo a nuestro dataset. Recuerda que lo que habíamos obtenido es una serie **“de números que señalan a qué intervalo pertenece el elemento”**

**clase = pd.cut(X['production\_budget'], 8).cat.codes.rename['class']**

#Lo estamos guardando en la variable clase para que pueda ser cncatenada por el método de pandas.

**pd.concat((X, clase), axis =1).head(2)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Unnamed: 0** | **production\_budget** | **title\_year** | **aspect\_ratio** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** | **imdb\_score** | **gross** | **class** |
| **0** | **0** | **425000000.0** | **2009.000000** | **1.780000** | **178.000000** | **4834.0** | **2.370000e+08** | **7.9** | **7.605058e+08** | **7** |

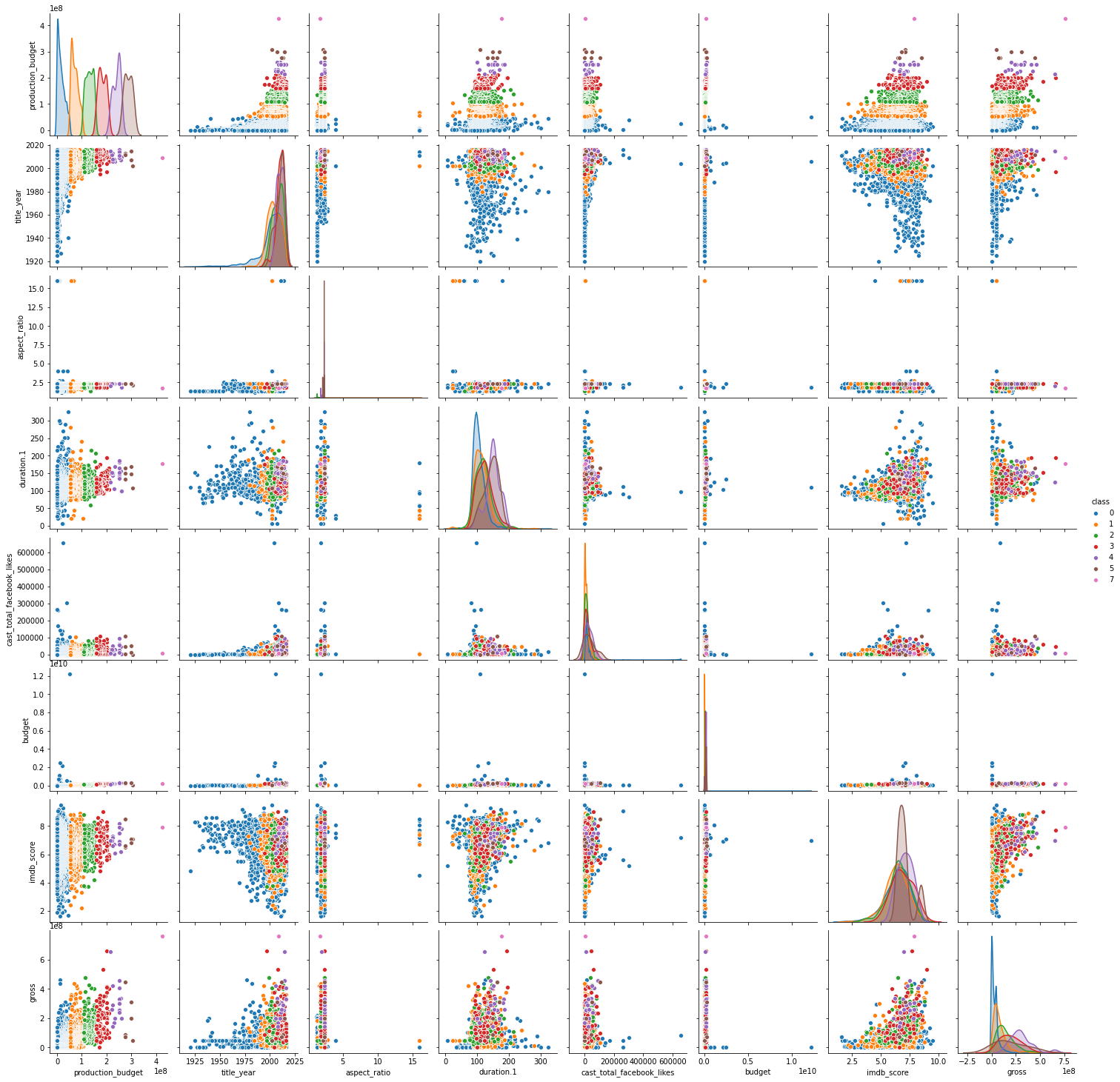
**Nota**: La **clase** nos indica es cuán costos es la película.Mientras más alto, más costosa fue la película

**Ahora** lo que haremos será graficar todas las correlaciones de nuevo.

z2 = pd.concat((X, clase), axis =1)

#Lo guardamos en z2 para poder introducirlo a nuestro metodo de seaborn

sns.pairplot(z2, hue = 'class') # con "hue" le decimos que nos pinte según los valores de "class"



**Pt. 2**

**Continuando con la exploración**

Hagamos un heatmap para visualizar la correlación entre nuestros features y nuestra variable objetivo. #No es necesario mostrarlo aquí, está en el colab

Por otro lado, veremos algunos métodos de Scikit-Learn para elegir los mejores features

from sklearn.feature\_selection import SelectKBest

#Lo que hace es un análisis de cada feature y se verá que tanta relación aporta; y , luego, se elegirán las K mejores

from sklearn.feature\_selection import mutual\_info\_regression

#es una métrica que nos dice qué tanta información nos da cada feature

selector = SelectKBest(mutual\_info\_regression, k=4)

selector.fit(X,Y)

scores = selector.scores\_

#Es un atributo que nos dice qué tanta información aporta cada feature

plt.rcParams["figure.figsize"] = [12, 8] #Hacemos esto para modificar el tamaño de la gráfica

plt.plot(scores)

plt.xticks(np.arange(7), list(X.columns)); #Para mostrar los labels del eje x

