**import** warnings

warnings**.**simplefilter("ignore")

**%matplotlib** inline

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

La regresión Lasso es interpretable:

* mientras más grande el coeficiente para una feature, más relevante es esta para la regresión.
* la regresión Lasso trata de seleccionar un pequeño número de features relevantes.

X **=** pd**.**read\_csv('../vol/intermediate\_results/X.csv')

**#TARGET**

y **=** X['worldwide\_gross']

**#SIN TARGET**

X **=** X**.**drop('worldwide\_gross',axis**=**1)

**from** sklearn.linear\_model **import** Lasso

model **=** Lasso()

#Separa datos en dos set, default 75% train 25% test

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y)

#Para conocer % de toda la data en train

len(X\_train)**/**len(X)

# output 0.75

#Entrena

model**.**fit(X\_train,y\_train)

OUTPUT

Lasso(alpha=1.0, copy\_X=True, fit\_intercept=True, max\_iter=1000,

normalize=False, positive=False, precompute=False, random\_state=None,

selection='cyclic', tol=0.0001, warm\_start=False)

model**.**score(X\_test,y\_test)

OUTPUT

0.53670649813230509

model**.**coef\_

array([ 2.89526507e+00, -1.40301472e+00, 1.66308214e-02,

3.33819964e+00, 2.15878265e+02, -8.00752044e-03,

2.53750354e+07])

In [15]:

var **=** np**.**floor(np**.**log10(np**.**abs(model**.**coef\_)))

In [17]:

plt**.**rcParams["figure.figsize"] **=** [12,8]

plt**.**plot(var)

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

Chart, line chart

Description automatically generated

Esto nos guía a guardar únicamente:

* production\_budget
* title\_year
* duration
* cast\_total\_facebook\_likes
* imdb\_score

**Correlación entre variables**

**import** seaborn **as** sns

Z **=** pd**.**concat([X,y],axis**=**1)

sns**.**pairplot(Z)

A picture containing table

Description automatically generated

In [28]:

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

Z2 **=** pd**.**concat([X,clase],axis**=**1)

In [29]:

sns**.**pairplot(Z2,hue**=**'class')

Out[29]:

<seaborn.axisgrid.PairGrid at 0x7faf37ebc748>

Diagram

Description automatically generated with low confidence

In [31]:

Z3 **=** pd**.**concat([X,y],axis**=**1)

sns**.**heatmap(Z3**.**corr())

Out[31]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7faf2191fe10>

Chart

Description automatically generated with medium confidence

De esto concluimos, sin sorpresa, que son muy importantes:

* production\_budget
* imdb\_score

**Metodos de selección automatica de features**

Sklearn posee una serie de métodos para seleccionar las mejores features.

Estos métodos los puedes encontrar en sklearn.feature\_selection

**from** sklearn.feature\_selection **import** SelectKBest

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

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

selector**.**fit(X,y)

SelectKBest(k=4,

score\_func=<function mutual\_info\_regression at 0x7faf38ed7730>)

scores **=** selector**.**scores\_

plt**.**rcParams["figure.figsize"] **=** [12,8]

plt**.**plot(scores)

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

Chart, line chart

Description automatically generated

Del analisis univariante obtenemos que las mejores features son:

* production\_budget
* cast\_total\_facebook\_likes
* budget

**Guardaremos las 5 features entregadas por la interpretación de nuestra regresión Lasso**

X2 **=** X[['production\_budget','title\_year','duration.1','cast\_total\_facebook\_likes','imdb\_score']]

X3 **=** X[['production\_budget','cast\_total\_facebook\_likes','imdb\_score']]

**Veamos los resultados del modelo con estas features**

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y)

cols2 **=** ['production\_budget','title\_year','duration.1','cast\_total\_facebook\_likes','imdb\_score']

X2\_train, X2\_test, y2\_train, y2\_test **=** X\_train[cols2], X\_test[cols2], y\_train, y\_test

cols3 **=** ['production\_budget','cast\_total\_facebook\_likes','imdb\_score']

X3\_train, X3\_test, y3\_train, y3\_test **=** X\_train[cols3], X\_test[cols3], y\_train, y\_test

**from** sklearn.linear\_model **import** Lasso

model1 **=** Lasso()

model2 **=** Lasso()

model3 **=** Lasso()

model1**.**fit(X\_train,y\_train)

model2**.**fit(X2\_train,y2\_train)

model3**.**fit(X3\_train,y3\_train)

Lasso(alpha=1.0, copy\_X=True, fit\_intercept=True, max\_iter=1000,

normalize=False, positive=False, precompute=False, random\_state=None,

selection='cyclic', tol=0.0001, warm\_start=False)

print(model1**.**score(X\_test,y\_test))

print(model2**.**score(X2\_test,y2\_test))

print(model3**.**score(X3\_test,y3\_test))

0.588827166128

0.5887559341

0.564313646779