Underfiting: Modelo demasiado simple que lo que hace es obtener un error muy grande tanto en el entrenamiento como la vereficación

Overfiting: Modelo demasiado completo que lo que hace es tener un error muy pequeño en el entrenamiento, pero que al momento de testear el modelo se descubre grandes errores.

Lo que debemos de lograr en un error de train no muy alto y un error de test no muy alto, también.

Una manera de comprobar para verificar que estamos en una buena situación es que

Etrain - Etest <<1

**Evaluando el modelo**

En sklearn “tenemos metrics y model\_selection “ para poder sacar “scores”

El “ score R2 “ lo que nos permite saber es que si nuestro modelo es mucho mejor que una predicción, bastante inocente, que es sería tomar la media de todos los puntos y predecir la media para todos los puntos. Si obtenemos el valor de “ 0 ”, quiere decir que tenemos el mismo performance que aquel modelo simple; y, si es que obtenemos 1, quiere decir que nuestra predicción es 100% justa: que todos los puntos lo predecimos de forma perfecta.

Si obtenemos un valor negativo, tendremos un modelo muy malo. Mientras más nos acercamos a 1 tendremos una buena regresión.

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

El estimador aún no vio estos datos, por lo que ahora va a predecir los datos y un score sobre estos datos nos va a dar una buena idea sobre lo que pasaría si nos llegasen datos nuevos.

**0.8237343903680256**

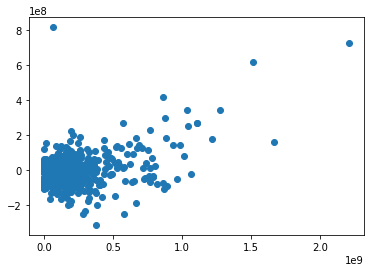
Obtuvimos este valor, por lo que podemos decir que hemos obtenido un buen estimador.Sin embargo, debemos de tener cuidado y no confiar en solo un número, así que debemos de graficar nuestros puntos para poder visualizar nuestro resultado.

**Vamos a usar los residuales:**

**Los residuales** son todas las diferencias que hay entre las predicciones y el valor real, por lo que debemos de esperar en nuestro “ plt.scatter( ) “ puntos muy cercanos a cero.En caso que no obtengamos un conjunto de puntos dejanos a cero, tendríamos un mal modelo .

residuals = Y\_test - predicted

**plt.scatter(** Y\_test , residuals **)**



Vemos que la mayoría de puntos están cera a cero, por lo que podemos decir que tenemos un modelo aceptable.

**Nota:** Lo anterior son los errores absolutos: en dinero

**Lo que haremos ahora** es ver esos errores, pero en porcentajes

ap\_residual = **np.abs**(residuals) / Y\_test #aplicamos abs( ) para obtener valores positivos para

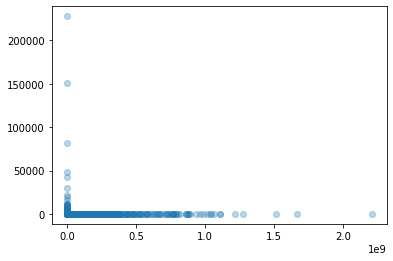
nuestro porcentaje

# lo dividimos entre Y\_test para normalizar nuestros

datos para obtener un porcentaje de error, mas no

el absoluto.

plt.scatter(Y\_test, ap\_residual, alpha =0.3)

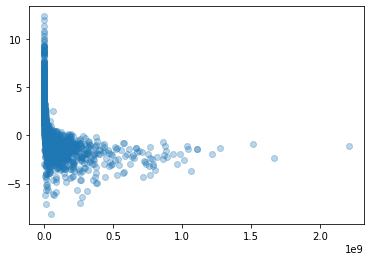


Sin embargo, esa escala no es muy buena para ver los puntos alejados.Entonces utilizaremos una escala logarítmica .

**Nota:** El logaritmo compacta los datos y permite ver las cosas en orden de magnitud y no de manera absoluta.

lap\_residuals =**np.log** (ap\_residual)

**plt.scatter**(Y\_test, lap\_residuals, alpha =0.3)

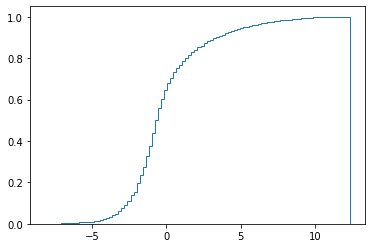


Con esto podemos decitr que tenemos una buena aproximación o un buen **estimador**

**Ahora,** haremos una evaluación numérica de nuestros errores. Para ello , graficaremos un diagrama que se hace llamar **“ La función cumulativa de la dist. de los errores“**

**plt.hist(** lap\_residuals, bins = 100 , normed =1, histtype = 'step', cumulative = True**) ;**

**#No olvides ese “ ; “**

****