Análisis Inteligente de Datos:
Tarea 2

Profesor: Ricardo Ñanculef
Integrantes: Felipe Carmona y Bastien Got
Fecha: 23 de Junio del 2016

Parte 1: Regresión Lineal Ordinaria

a) **df = df.drop('Unnamed: 0', axis=1) :** se elimina la columna “unnamed: 0 (int)” porque no tiene sentido

istrain_str = df['train']
istrain = np.asarray([True if s == 'T' else False for s in istrain_str]) : se construya un array de booleanos (True si el data es un data de entrenamiento y False si es un data de test)

istest = np.logical_not(istrain) : se construya un array de booleanos (True si el data es un data de test y False si es un data de entrenamiento)

df = df.drop('train', axis=1) : se elimina la columna “train (object)”

b) El dataset contiene 97 datos. Cada persona está representada por:
- lcavol (float) : log (volumen de cáncer)
- lweight (float) : log (peso de la prostate)
- age (int)
- lbph (float) : log (cantidad de hyperplasia de la prostate benigno)
- svi (int) : invasión de vesículas seminales
- lcp (float) : log (penetración capsular)
- gleason (int) : puntuación de Gleason
- pgg45 (int) : porcentaje de Gleason con puntuación 4 o 5
- lpsa (float) : log (prostate-specific antigen) - la variable que queremos modelizar

c) Si $Y = X.\beta + e$, normalizar los datos permite de luego hacer hipótesis sobre los coeficientes β (supongamos que son nulos) y ver si los $(\hat{\beta} - \beta) / \sigma = \beta / \sigma$ siguen una ley de T(0,1) con $\sigma = error\ standard\ de\ la\ variable$.
Donde Y = el vector conteniendo el parámetro *lpsa* por el cual queremos modelizar el
comportamiento en función de los otros parámetros
X = la matriz de los datos sin el parámetro *lpsa*
β = el vector de los coeficientes de la regresión lineal
e = el vector de los residuos

d) Para hacer la regresión lineal tenemos que trabajar con los datos de entrenamiento que luego validaremos con los datos de prueba, así que tenemos que separar los datos de entrenamiento y los datos de prueba. Al fin, la regresión lineal se hace

sobre los datos de entrenamiento y por lo tanto los argumentos son Xtrain y ytrain. La introducción de una nueva variable "intercept" es importante porque hay que añadir al modelo la **constante β_0**.

e) Los pesos son los siguientes : [0.68, 0.26, -0.14, 0.21, 0.3 , -0.29, -0.02, 0.27, 2.46] y por lo tanto el modelo es:

lpsa = 0.68*lcavol + 0.26*lweight - 0.14*age + 0.21*lbph + 0.3*svi - 0.29*lcp - 0.02*gleason + 0.27*pgg45 + 2.46

Variable	Weight	Standard error	z-score
lcavol	0.68	0.13	5.22
lweight	0.26	0.14	1.92
age	-0.14	0.12	-1.14
lbph	0.21	0.12	1.69
svi	0.3	0.12	2.44
lcp	-0.29	0.12	-2.33
gleason	-0.02	0.12	-0.18
pgg45	0.27	0.13	2.08
Intercept	2.46	0	-

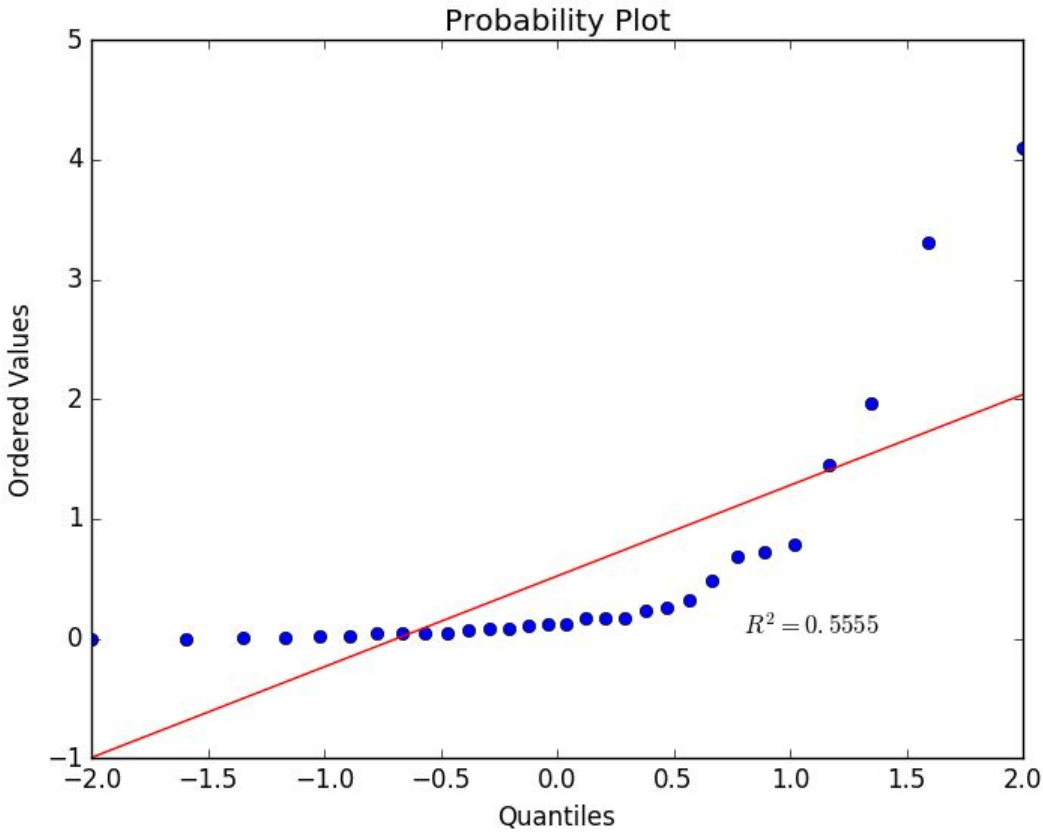
Mirando la distribución de student vemos que T(n-d, 5%) = T(58, 5%) ~= 2 y por lo tanto todas las variables que tienen un z-score superior a 2 en valor absoluto son relevantes. Las variables más significativas son, en orden de importancia descendiente **lcavol**, **svi**, **lcp** y **pgg45**.

f)

Cross-val 5-fold	Cross-val 10-fold	Error de prueba (RSS)
0.96	0.76	0.52

El error por la validación cruzada con 5 "fold" es bastante buena pero vemos que cuando aumentamos el número de fold el error aumenta también. Y al final cuando probamos el modelo sobre los datos de prueba nos damos cuenta que el error es mucho más grande así que parece que el modelo podría ser mejor.

g)



El gráfico es un quantile-quantile plot que muestra si los residuos siguen una distribución normal o no.

R^2 es bastante diferente de 1 (0.56) y también se puede constatar mirando el gráfico que no es razonable decir que los residuos siguen una distribución normal.

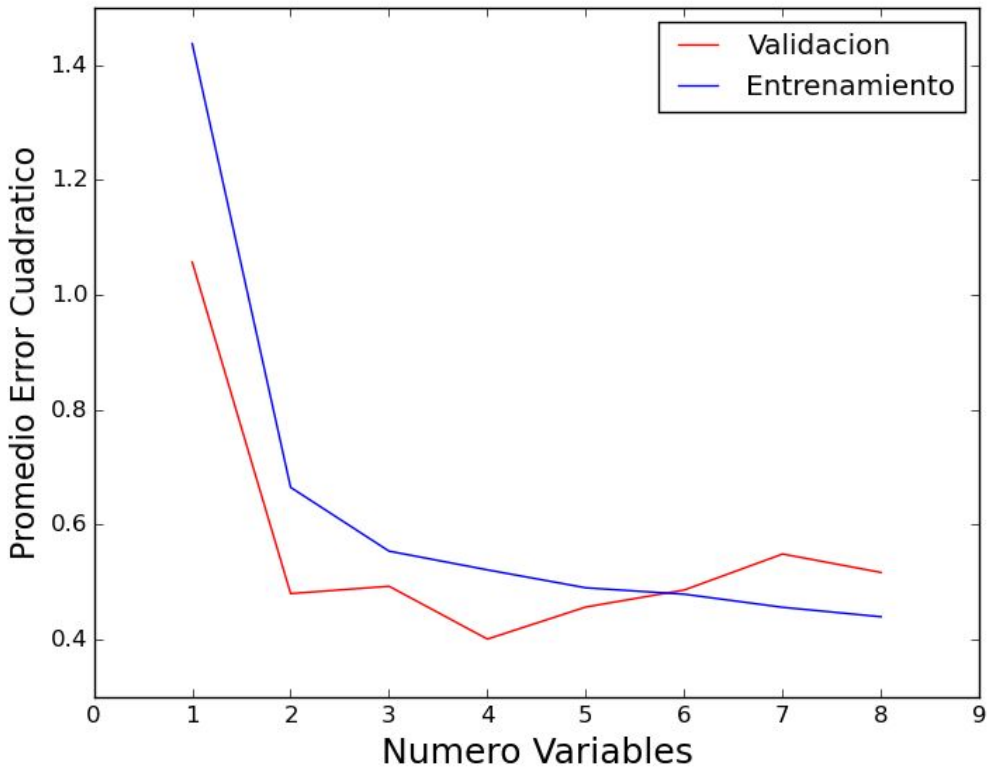
Parte 2: Selección de Atributos.

a) Lo primero que se realizó en este ítem fue interpretar el código y cómo funciona funciona *Forward Step-wise Selection* para la selección de atributos:

Lo que hace este algoritmo es comenzar con una lista vacía de “variables” que será denominada “variables seleccionadas”, luego modela una regresión lineal con cada una de las variables que no están en la lista “variables seleccionadas” y calcula el indicador: **promedio del error cuadrático.** La variable que entregue menor valor del indicador entrará a la lista de seleccionados. El Algoritmo continuará hasta llenar la lista de “variables seleccionadas” hasta un cierto número de variables requeridas.

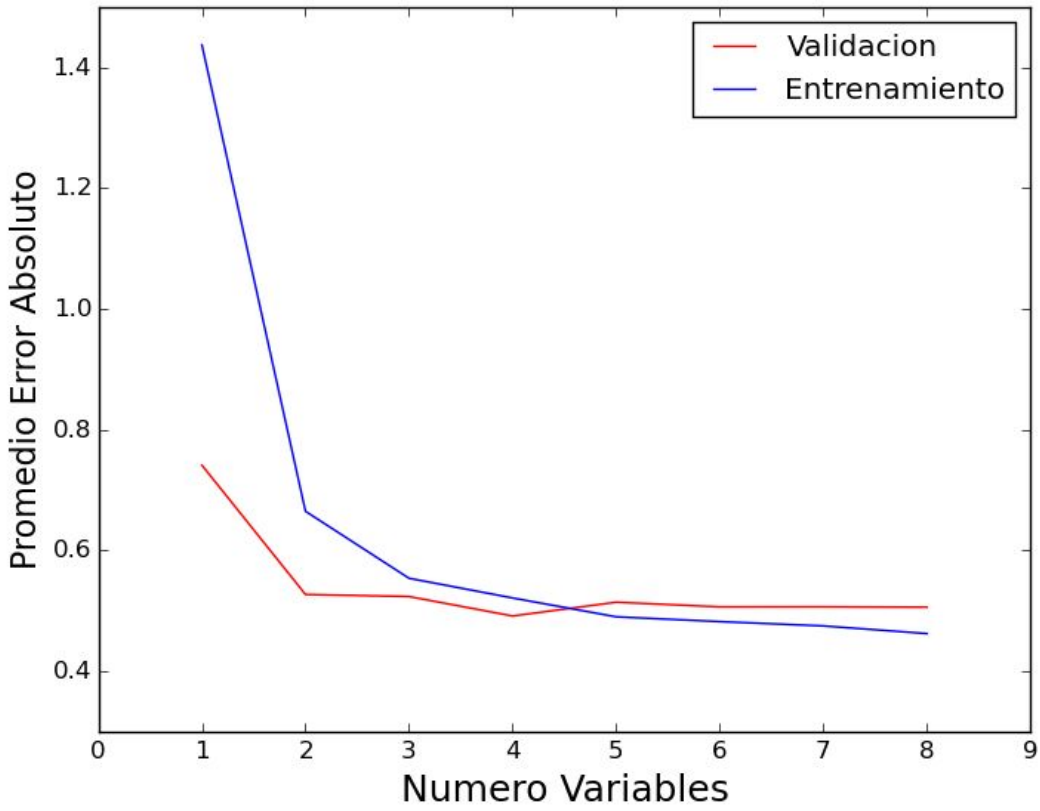
A continuación se construye el siguiente gráfico donde muestra el promedio del error cuadrático de predicción en función de la cantidad de variables con el cual se construye el modelo. Esta construcción está sujeta a las siguientes consideraciones:

- La lista de seleccionados comienza vacía, es decir, se probará construir el modelo con todas las variables.
- La medida de comparación es el **promedio del error cuadrático.**



Luego se procede a cambiar el indicador por uno distinto al utilizado en el ejemplo, a continuación se muestran las consideraciones:

- La lista de seleccionados comienza vacía, es decir, se probará construir el modelo con todas las variables.
- La medida de comparación es el **promedio del error absoluto.**



Observaciones: Se evidencia que en ambos casos la cantidad de variables necesarias para modelar el problema de manera óptima son 4 variables. Se decidió esto ya que el mínimo en ambas curvas rojas se encuentra en el número de variables 4.

En ambos modelos las variables seleccionadas fueron: **Intercepto, Lcavol, Lweight, Svi.**

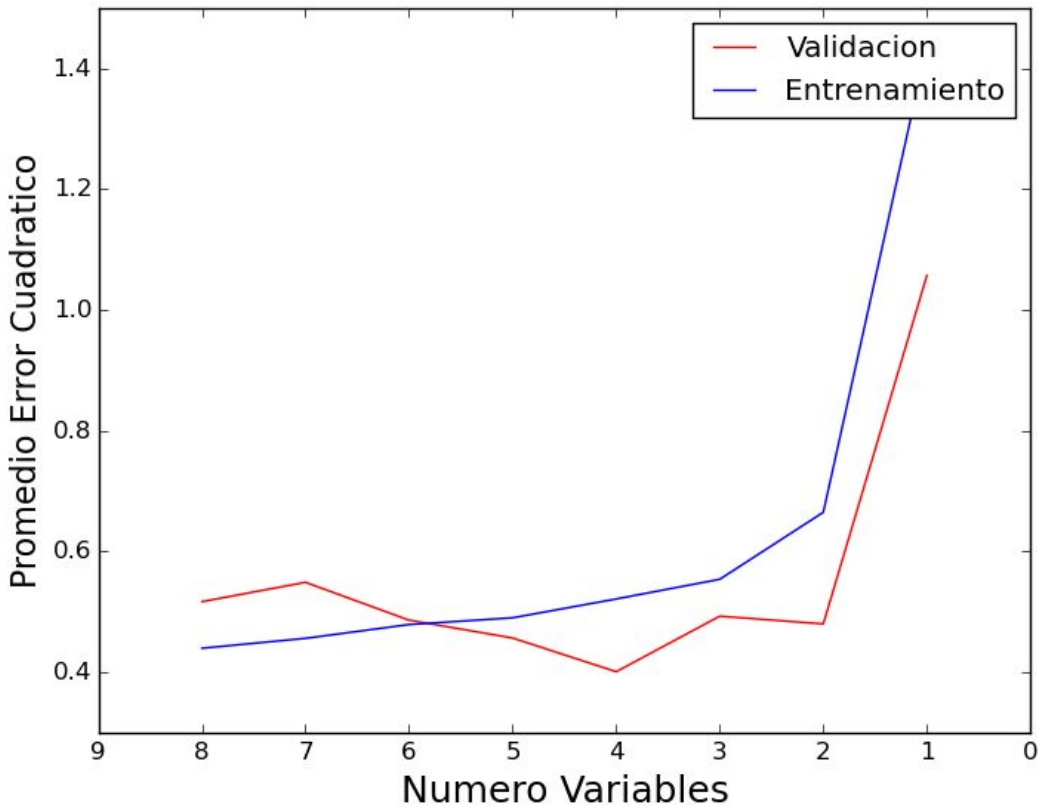
b) A continuación se explicará cómo se realizó el algoritmo *Backward Step-wise Selection* para la selección de atributos:

Se inicia una lista llamada “variables seleccionadas” la cual contendrá todas las variables posibles para predecir el modelo. Luego para un elemento de la lista se genera un modelo con todas las variables a excepción del elemento seleccionado. Luego se calcula el error de predicción y se asocia a esa variable. La variable que tenga un mayor error asociado será la variable que se irá guardando como predictor.

La lógica es: si sacando esta variable se produce un mayor error en el modelo, entonces esta variable es más importante que las demás. Este proceso se repite hasta tener las variables requeridas.

Luego se procede a generar un gráfico donde se muestra el error de entrenamiento y a la vez el error de validación, para esto se tienen las siguientes consideraciones:

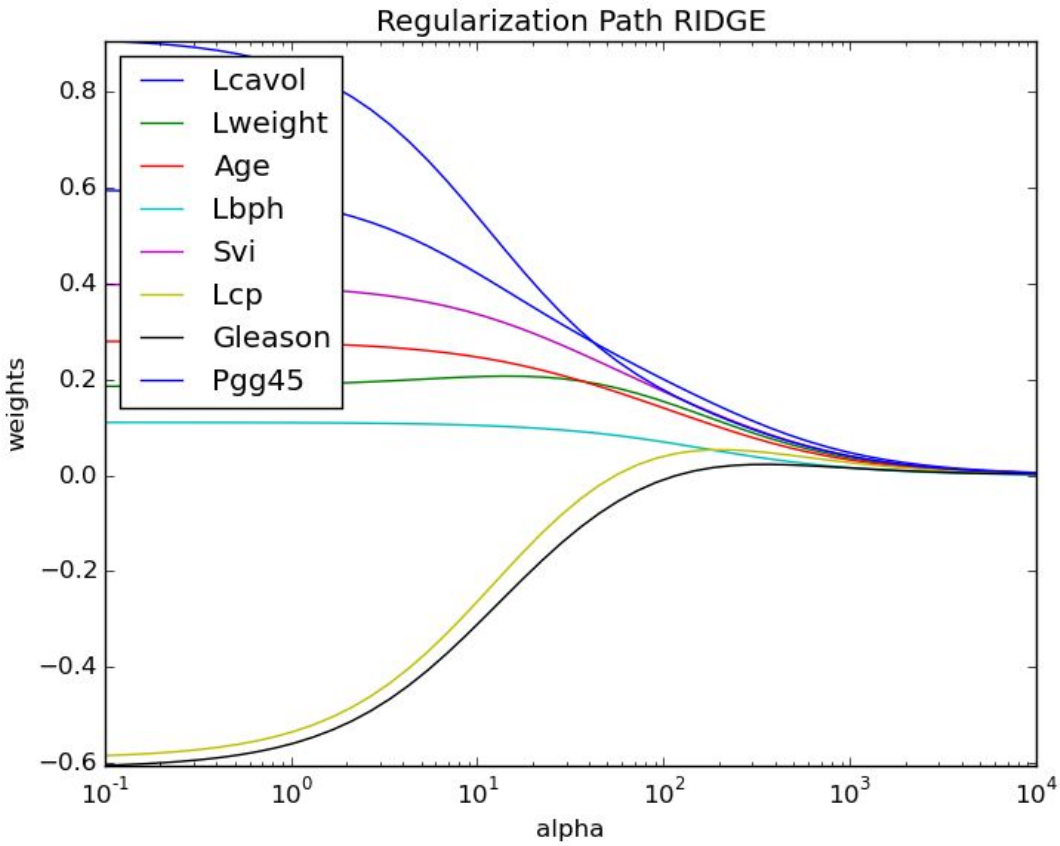
- La lista de seleccionados comienza vacía, es decir, se probará construir el modelo con todas las variables.
- La medida de comparación es el **promedio del error cuadrático.**



Al igual que el método anterior los atributos óptimos seleccionados fueron: **Intercepto, Lcavol, Lweight, Svi.**

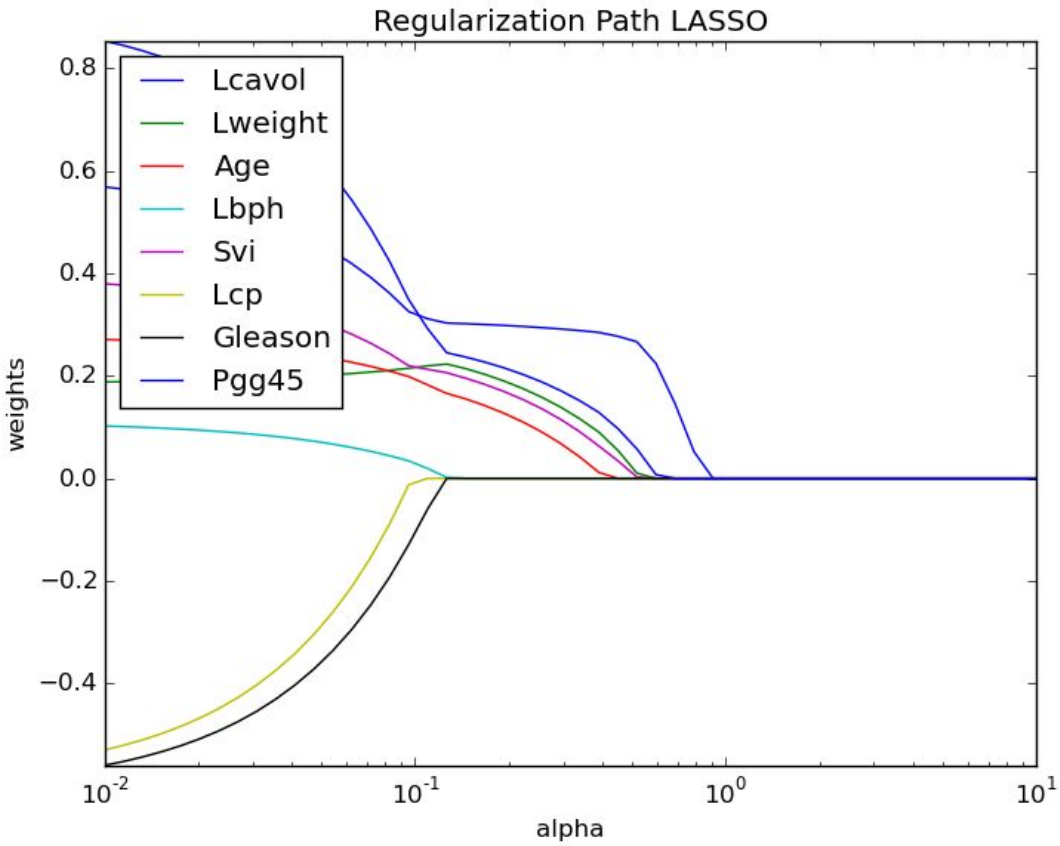
Parte 3: Regularización.

a) A continuación se muestra un gráfico donde se puede observar el peso que tienen las distintas variables (o predictores) en función de la regularización con el método **Ridge**.



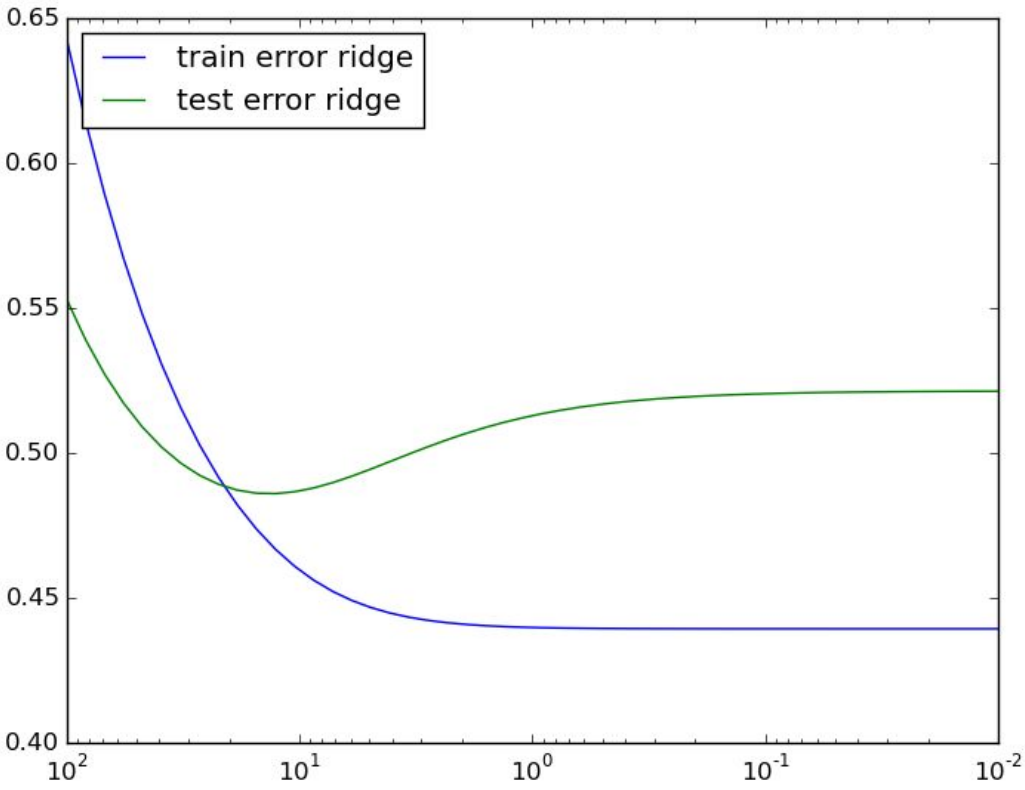
Se puede observar que a medida que más se penaliza (más aumenta el alpha) menos peso van teniendo las variables. Si bien las variables llegan a tener peso 0, cuando alpha es muy grande, se observa que las variables siguen teniendo algún peso en el modelo cuando alpha está entre 100 y 1000.

b) A continuación se realiza el mismo gráfico pero la diferencia es que el método de regularización será **Lasso**:



En este gráfico se puede observar que cuando alpha va creciendo, los pesos de algunas variables se hacen 0. Esto quiere decir, que a medida que más se va regularizando, más variables tienen un peso 0, lo que significa que solo toma en cuenta algunas variables y no parte de ellas (como lo hace **Ridge**). En este caso se aprecia que el método que sirve más para seleccionar variables es **Lasso**

c) Luego se hará un gráfico donde se muestre el error de entrenamiento y el error de validación de los distintos modelos de regularización que irán cambiando respecto a alpha. El método de regularización será **Ridge**:

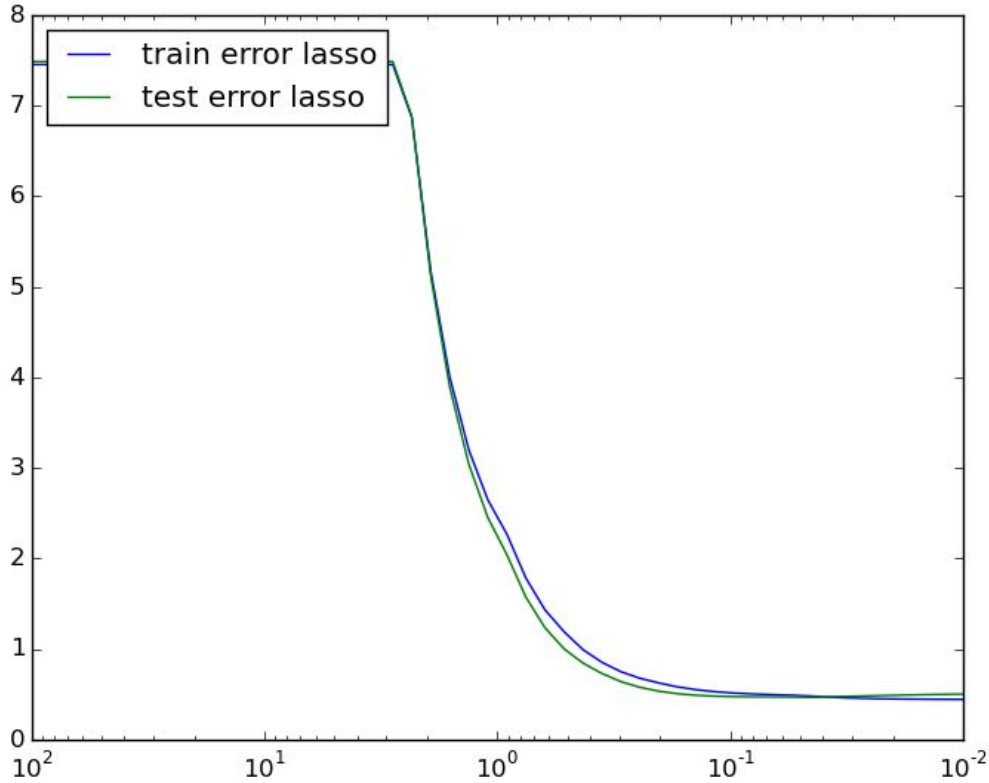


Se observa que el **error de entrenamiento** disminuye a medida que los valores de alpha van disminuyendo. Por otro lado se puede observar que la curva de error de test (o validación) presenta un mínimo que representa el menor error dado cierto valor de alpha. En este caso se tienen los siguientes valores aproximados:

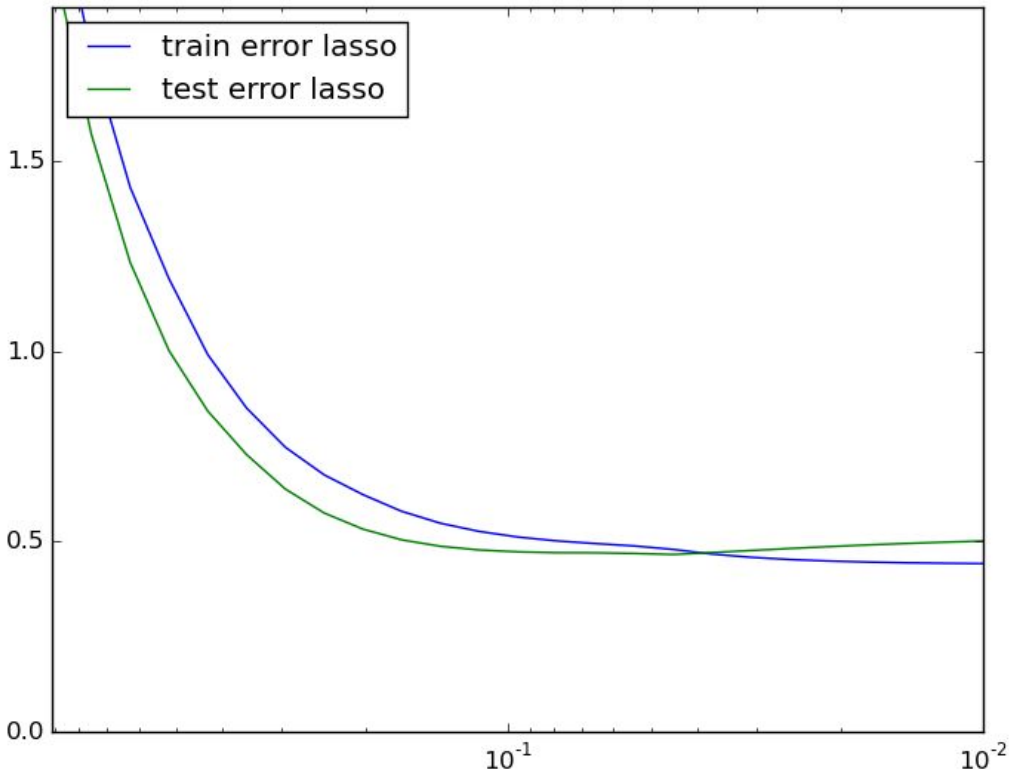
Curva	Alpha	Error
Entrenamiento	2,170	0,4425
Validación	11,608	0,4859

También se puede observar que después del alpha que otorga el mínimo local, el error de validación comienza a ascender nuevamente, comprobando así el fenómeno de *over-fiting* al modelar con más variables de las realmente necesarias.

d) A continuación se realizará el mismo análisis anterior pero regularizando con el método Lasso:



Se puede observar que ambas curvas se comportan muy similares, pero a pesar de esto se puede observar una gran diferencia en los alphas que minimizan el error en ambas curvas. A continuación una imagen que sirve para evidenciar la separación:



Luego se presenta la siguiente tabla que presenta los resultados de los distintos alphas.

Curva	Alpha	Error
Entrenamiento	0,01	0,4460
Validación	0,1	0,47

Al igual que **Ridge**, también se puede observar que después del alpha que otorga el mínimo local, el error de validación comienza a ascender nuevamente, pero de manera muy *suave.*

e) A continuación se estimará el parámetro alpha utilizando validación cruzada. Para esto se calculará el alpha para los **datos de entrenamiento**. También no está demás mencionar que la medida para calificar los alphas será el **promedio del error cuadrático entre cada modelo**. A continuación una tabla que resume los resultados:

Método	Entrenamiento	Error
Lasso	0.010	0.759
Ridge	2.12	0.752

Se puede observar que el método de validación cruzada da valores muy similares de los alphas óptimos que se obtuvieron al graficar el error de entrenamiento en función de los distintos alphas. La diferencia está en el mínimo error encontrado por ambos métodos, en validación cruzada se puede detectar un error más grande que realizando una comparación entre los distintos modelos y el error de predicción (ítems c) y d) de esta sección).

Parte 4: Predicción de Utilidades de Películas

No se realizó Parte 4 del laboratorio.