Técnicas Multivariantes

Técnicas de regresión avanzadas II

Índice

Esquema

Regresión Avanzada II

Mínimos Cuadrados Recortados

Método robusto	Outliers	Coste computacional

Regresión Penalizada

Selección de variables	Reducir varianza del modelo	Orden de la penalización

Red Elástica

Combinación de regresión LASSO y de regresión Ridge

Es necesario ajustar hiperparámetros α y r

r (LASSO ratio)	Hiperparámetro de encogimiento α

- r define la proporción de regresión LASSO respecto al total de la Red Elástica
- r = 0 → regresión ridge
- r = 1 → regresión LASSO
- r se puede prefijar de antemano. α se suele ajustar con técnicas de validación

Ideas clave

6.1 Introducción y objetivos

En el tema anterior se trataron las bases de la regresión lineal múltiple. En cambio, en este tema se explicarán distintas estrategias de regresión que permiten resolver problemas donde existan outliers o donde las variables predictoras presenten colinealidad. En concreto, se estudiarán:

- La regresión por **Mínimos cuadrados recortados**, que es un método de regresión robusta, y por tanto, adecuado cuando existen outliers.

- La **regresión penalizada**. Se explicarán las diferencias entre dos tipos de regresiones penalizadas: la regresión LASSO y la regresión Ridge. Además, se explicará el desarrollo de una Red Elástica que consiste en una combinación de las regresiones anteriores, LASSO y Ridge.

Por último, se expondrá un caso de aplicación donde se profundizará en la aplicación de las técnicas de regresión avanzadas.

6.2 Mínimos cuadrados recortados

Los mínimos cuadrados recortados o *least trimmed squares* es un método robusto de estimación de los parámetros de una regresión lineal múltiple. Consiste en escoger un subcojunto de observaciones h de los n puntos orginales cuyo ajuste por mínimos cuadrados posea la menor suma del cuadrado de los residuos. El número de puntos

escogido (h) debe establecerse con un valor entre $n/2$ y n (Rousseeuw and Van Driessen 2006). El inconveniente que presenta este método, es que el tiempo de computación crece demasiado a medida que aumenta el número de observaciones del conjunto de datos. Para poder emplear esta técnica en conjunto de datos muy grandes existen algunos algoritmos como FAST-LTS o librerías específicas en Python como lts-fit (https://pypi.org/project/ltsfit/).

Aplicación de los mínimos cuadrados recortados

Para la aplicación de la técnica con la librería *Scikit-Learn* podemos definir un objeto de tipo LTS, el cual va a realizar el siguiente algoritmo:

- Escoger de manera aleatoria el h/n por ciento de los puntos y ajustar por mínimos cuadrados.
- Repetir *rep* veces el proceso y escoger el subconjunto h^* que obtiene la mejor regresión.
- Reemplazar un punto del subconjunto h^* por otro punto de n que no estuviese incluido en h^*. Repetir este proceso *rep2* veces.

A continuación se muestra el código con el que se podría implementar este ajuste:

```
def LTS(x, y, porcentaje_n, rep, rep2):
  from sklearn import linear_model
  lm = linear_model.LinearRegression()
  max_score = 0

  n = int(porcentaje_n*len(x))
  newx = np.ones(n)
  newy = np.ones(n)
  bestx = np.ones(n)
```

```
besty = np.ones(n)

#find starting point randomly
for i in range(rep):

  newx = np.array(sorted(random.sample(range(len(x)), n)))
  j = 0
  for val in newx:
    newy[j] = y[val]
    j = j + 1

  lm.fit(newx.reshape(-1, 1), newy)
  score = lm.score(newx.reshape(-1, 1), newy)
  if(score > max_score):
    max_score = score
    np.copyto(bestx, newx)
    np.copyto(besty, newy)


#alter starting point to get iteratively better
for i in range(rep2):
  np.copyto(newx, bestx.astype(int))
  np.copyto(newy, besty.astype(int))
  flag = True
  while flag:
    xloc = random.randint(0, len(x) - 1)
    if int(x[xloc]) not in newx:
      flag = False

  newxloc = random.randint(0, len(newx) - 1)
  newx[newxloc] = xloc
  newx = np.array(sorted(newx))
```

```
    j = 0
    for val in newx:
      newy[j] = y[val]
      j = j + 1

    lm.fit(newx.reshape(-1, 1), newy)
    score = lm.score(np.array(newx).reshape(-1, 1), newy)
    if(score > max_score):
      max_score = score
      np.copyto(bestx, newx)
      np.copyto(besty, newy)
      mejor_lm = lm

  return mejor_lm
```

Una vez que se ha definido el método, pasamos a aplicarlo a diferentes conjuntos de datos lineales con distintos tipos de *outliers* para comprobar como se diferencia con respecto a la regresión mediante mínimos cuadrados ordinarios. Se va a emplear un porcentaje del 80 % (*porcentaje_n = 0.8*) de los puntos del total, con 20 repeticiones (*rep = 20*) para obtener el conjunto de datos inicial y 10000 permutaciones entre los puntos (*rep2 = 10000*).

En particular, se va a comprobar la diferencia entre las dos regresiones para cuatro tipos de datos:

- Sin outliers.
- Con los outliers en contra la dirección del ajuste.
- Con los outliers distribuidos de forma aleatoria.
- Con un único outlier (muy grande).

```
import numpy as np
import random
from matplotlib import pyplot as plt
from sklearn import linear_model

# inicializar valores lineales----------------------------------
n_samples = 100
slope = 1
intercept = 0
x = np.ones(n_samples)
y = np.ones(n_samples)
# crear datos lineales con ruido--------------------------------
for i in range(n_samples):
    x[i] = i
    y[i] = slope * i + intercept + random.gauss(0, n_samples/5000)

xr = x.reshape(-1, 1)
title = ['sin outliers', 'outliers en un nivel',
'outliers aleatorios', 'un gran outlier']

#introducir outliers-------------------------------------------
# inicializar datos outliers
y1 = y.copy()
y2 = y.copy()
y3 = y.copy()
y4 = y.copy()

for i in range(n_samples):
        # uno de cada 10 puntos outlier contra
        if i%10 == 0:
          y2[i] = -y[i]
        # uno de cada 10 puntos outlier aleatorio
```

```
        if i%10 == 0:
          y3[i] = random.gauss(0, 100)
        # un outlier muy grande
        if i == 99:
          y4[i] = -5000

# ajuste y1----------------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(xr, y1)
```

```
## LinearRegression()
```

```
lts1 = LTS(x, y1, porcentaje_n=0.8, rep=20, rep2=10000)

# ajuste y2----------------------------------------------------
lm2 = linear_model.LinearRegression()
lm2.fit(xr, y2)
```

```
## LinearRegression()
```

```
lts2 = LTS(x, y2, porcentaje_n=0.8, rep=20, rep2=10000)

# ajuste y3----------------------------------------------------
lm3 = linear_model.LinearRegression()
lm3.fit(xr, y3)
```

```
## LinearRegression()
```

```
lts3 = LTS(x, y3, porcentaje_n=0.8, rep=20, rep2=10000)

# ajuste y4----------------------------------------------------
lm4 = linear_model.LinearRegression()
lm4.fit(xr, y4)
```

```
## LinearRegression()
```

```
lts4 = LTS(x, y4, porcentaje_n=0.8, rep=20, rep2=10000)
```

En la Figura 1 se representan las regresiones por mínimos cuadrados ordinarios (OLS) y la regresión por mínimos cuadrados recortados (LTS) para cada uno de los cuatro conjuntos de datos generados. Se observa que la regresión por mínimos cuadrados recortados es capaz de ajustar los datos sin verse afectada por los outliers, de ahí su robustez. Además, en ausencia de ellos, ajusta del mismo modo que la regresión por mínimos cuadrados ordinarios (OLS).

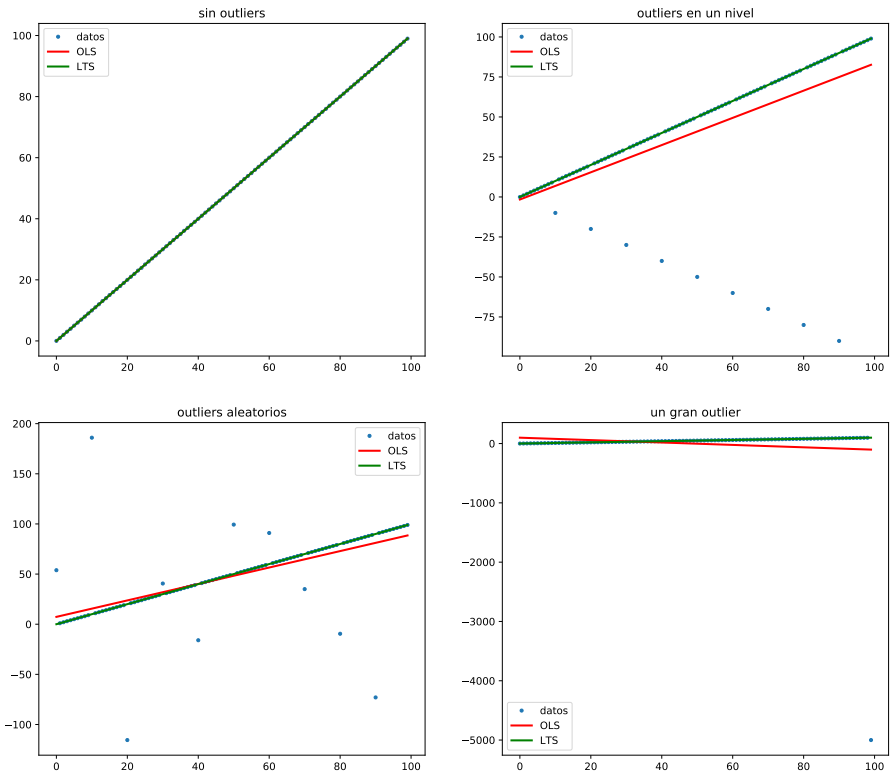

Figura 1: Ejemplos de regresión por mínimos cuadrados recortados

Así pues, la regresión por mínimos cuadrados recortados es muy útil frente a la presencia de outliers, aunque, como se ha comentado anteriormente, si el conjunto de datos es muy grande, con muchas observaciones y muchas variables, puede ser muy costoso computacionalmente. En estos casos, otra opción es aplicar algún método de selección

de variables.

6.3 La regresión penalizada

En el tema anterior se describieron dos estrategias (mejor subconjunto y selección por etapas) para realizar la selección de variables. Estos métodos, en algunas ocasiones, no son lo suficientemente consistentes para poder realizar la selección de variables y se han desarrollado diversas alternativas. Una de ellas es ajustar el modelo mediante una regresión penalizada. Este método es muy útil cuando se tienen variables predictoras que presentan problemas de colinealidad o cuando tenemos muchas variables predictoras en comparación con el número de observaciones.

La regresión penalizada es similar a la regresión por mínimos cuadrados, pero añadiendo una restricción sobre el tamaño de los coeficientes de regresión. Dentro de la regresión penalizada, se pueden distinguir dos tipos distintos atendiendo a la manera en la que están definidas estas restricciones: la regresión de tipo Ridge y la regresión de tipo LASSO. Por último, se pueden combinar ambas restricciones dando lugar a lo que se conoce como la Red Elástica.

Regresión Ridge

La regresión Ridge consiste en añadir una penalización de tipo cuadrático a la estimación de los coeficientes β, ya sea cuando se obtienen por mínimos cuadrados o por otros métodos, como el de máxima verosimilitud. Aunque se tiene conocimiento de la regresión Ridge desde los años 70 (Kennard and Hoerl 1970), es a partir de los últimos años cuando ha sufrido una gran expansión en su aplicación debido al aumento en las capacidades de cálculo computacionales.

En la regresión Ridge se busca minimizar el error cuadrático medio, pero con la restric-

ción de que $\sum_{i=1}^{p} \beta_i^2 < s$ donde s es conocido como el parámetro de ajuste.

Es interesante apuntar que en la regresión Ridge no se llega a producir una selección de variables, sino que los distintos coeficientes se van encogiendo a medida que el valor del parámetro de ajuste s disminuye y, en consecuencia, la restricción aumenta.

Aunque no realiza selección de variables, la regresión Ridge presenta dos grandes ventajas frente a la regresión por mínimos cuadrados ordinarios.

- La primera de las ventajas consiste en que se reduce la varianza de los modelos para distintas posibles muestras de la población, a costa de aumentar el sesgo. Pero, en el caso en el que se tienen muchas variables predictoras, suele mejorar los resultados en el cómputo global. En la Figura 2 se muestran los distintos escenarios que se pueden dar en cuanto al sesgo y la varianza de un modelo. En esta figura se representan los parámetros reales de los parámetros del modelo como una diana, y la estimación de los mismos para diferentes muestras de la población como aspas. Con la regresión de tipo Ridge se trata de pasar de un escenario con una alta varianza a un escenario con baja varianza, aunque sea a costa de aumentar ligeramente el sesgo. Esto es muy útil para que la estimación de los parámetros del modelo (los coeficientes β) no varíen drásticamente frente a cambios en la muestra obtenida de la población de estudio.

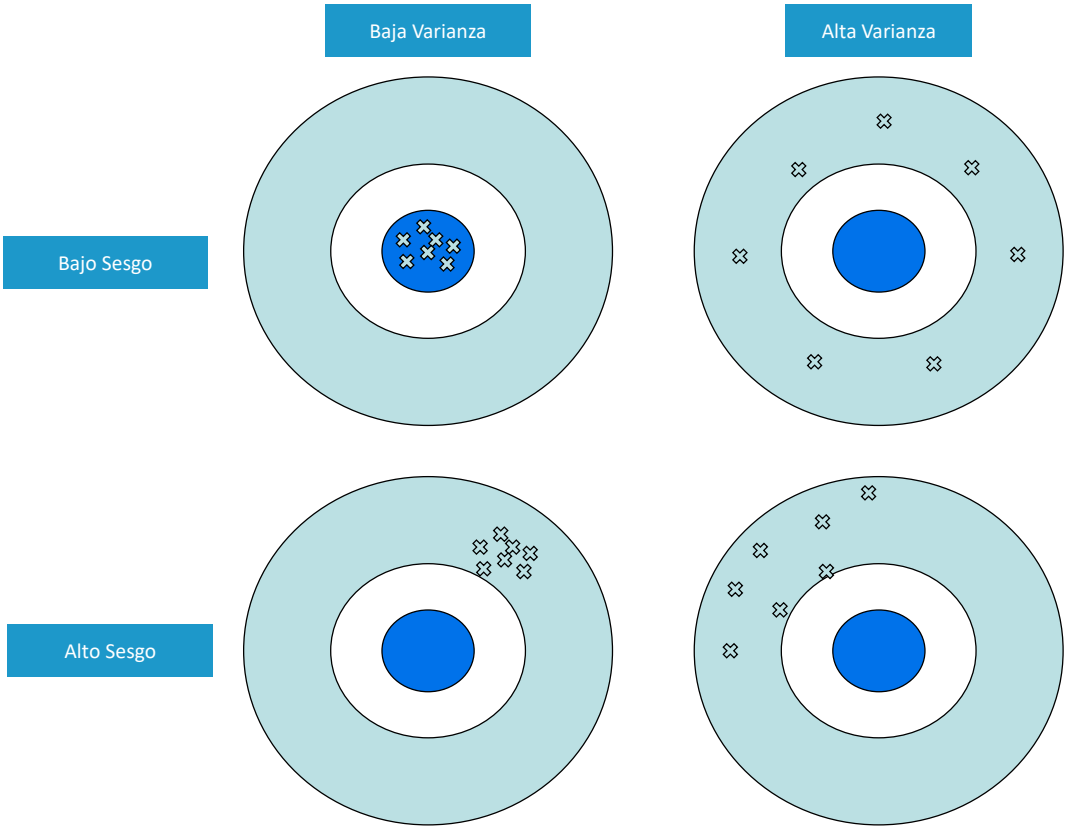

Figura 2: Posibles escenarios sesgo-varianza

- La segunda ventaja la encontramos cuando las variables predictoras están correlacionadas entre sí. Supongamos que existen dos variables predictoras fuertemente correlacionadas. En los modelos sin penalizar es habitual que el valor en la estimación de sus coeficientes sea para una de las variables muy elevado y para la otra muy reducido, pudiendo ocasionar problemas computacionales e, incluso, predicciones con un gran error. En cambio, en una regresión con una penalización de tipo Ridge, cuando existen dos variables muy correlacionadas, el peso de las mismas en la regresión se divide, y por tanto sus coeficientes son similares.

En este punto es interesante hacer notar que cuando se aplica una penalización, al contrario de lo que sucede cuando no se penaliza, la escala en la que están definidas las variables influye en la estimación de los coeficientes y, por lo tanto, es muy recomendable normalizar las variables (es suficiente con estandarizar) antes de estimar los

coeficientes.

Regresión LASSO

La regresión de tipo LASSO (siglas en inglés de *Least Absolute Shrinkage and Selection Operator*) es similar a la regresión Ridge, pero con la restricción sobre los coeficientes β en norma L^1, es decir, que la restricción tiene la forma: $\sum_{i=1}^{p} \mid \beta_i \mid < s$ (Tibshirani 1996).

La regresión LASSO, tiene la ventaja de que, al contrario de lo que ocurre con la regresión Ridge, con un s suficientemente pequeño se consigue que los coeficientes β se encojan hasta 0 y, por lo tanto, es posible hacer selección de variables. En cambio, el mayor inconveniente de este tipo de regresión lo encontramos cuando entre las variables predictoras algunas de ellas están fuertemente correlacionadas, ya que en estos casos la regresión de tipo LASSO tiende a seleccionar únicamente a una de estas variables y a descartar el resto, aumentando la varianza del modelo. En la Figura 3, extraída de (James et al. 2013), se muestran los dos tipos de restricciones que emplea cada una de las regresiones penalizadas descritas.

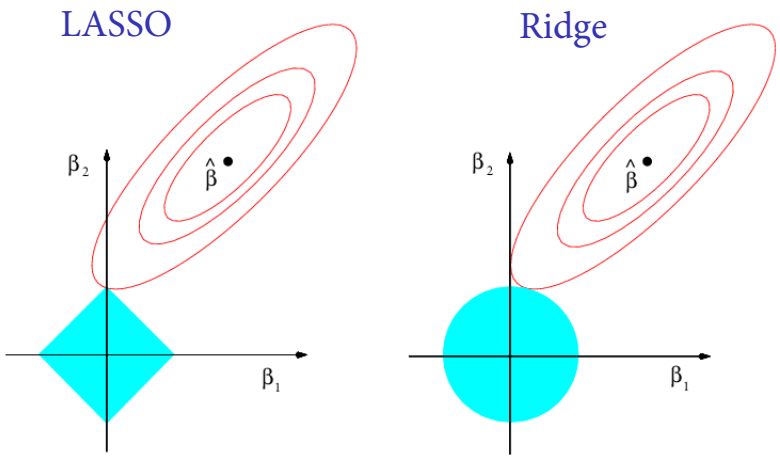

Figura 3: Representacion gráfica de las regresiones LASSO y Ridge

Las elipses representas diferentes estimaciones de los coeficientes de regresión del

modelo donde cada elipse proporcionaría el mismo valor de la condición de búsqueda (minimimos cuadrados o máxima verosimilitud). Por otro lado, las restricciones, con forma de rombo en el caso de la regresión LASSO y con forma de circumferencia para la regresión Ridge, marcan el espacio permitido de valores que pueden tomar la estimación de los coeficientes de regresión del modelo. Se observa, que la regresión LASSO favorece que el encogimiento de los coeficientes de la regresión venga acompañado con la selección de variables (encogimiento de algunos coeficientes hasta 0).

Red Elástica

Una vez que ya se han definido las regresiones LASSO y Ridge, se puede definir la regresión de tipo Red Elástica. Ésta se obtiene como la regresión en la que intervienen al mismo tiempo las penalizaciones de tipo LASSO y de tipo Ridge, siendo determinado el peso de cada una de ellas mediante el *hiperparámetro* de ajuste r, tal y como se define en la Ecuación 1 (Zou and Hastie 2005).

$$(1-r)\sum_{i=1}^{p}\beta_i^2 + r\sum_{i=1}^{p} \mid \beta_i \mid < s, \tag{1}$$

donde r tiene un valor entre 0 y 1 (Con $r = 1$ se obtiene la regresión de LASSO y con $r = 0$ se obtiene la regresión de Ridge). La elección del hiperparámetro r no es trivial y debe escogerse cuidadosamente según las características del problema y según el tipo de modelo que se desea obtener. En algunas ocasiones se podrá prefijar de antemano este hiperparámetro y en otras ocasiones será más adecuado ajustarlo mediante técnicas de validación.

A continuación, vamos a ver con un ejemplo cómo realizar el ajuste de los hiperparámetros del modelo r y α mediante el uso de una rejilla de posibles valores y de la validación cruzada con repetición (que vimos en el Tema 3).

Ejemplo red elástica: predicción nivel colesterol en sangre

Se quiere medir la relación entre el nivel de colesterol en sangre con diferentes variables predictoras. Se selecciona un primer número de posibles variables predictoras ($m = 48$) que proceden de distintos indicadores sociodemográficos, sanitarios y de las medidas de metabolitos obtenidas a través de un análisis de orina para una muestra de $n = 990$ individuos.

Se tiene un número de m bastante grande en comparación a n (48 frente a 990). Además, se ha realizado el análisis del FIV (ver Tema 5) y para algunas de las variables predictoras se obtiene un valor superior a 100, lo que nos indica que podemos tener problemas de colinealidad. Por lo tanto, se decide ajustar la regresión mediante la Red Elástica.

Para determinar la red, se deben escoger los hiperparámetros α y r. Como ya se ha comentado, r determina qué parte de la penalización de tipo Ridge y qué parte de la penalización de tipo LASSO se va a aplicar en la red y el hiperparámetro α (con comportamiento inverso al de la restricción s) indica cómo de restrictiva va a ser esa penalización.

Para seleccionar estos parámetros de ajuste, se crea una rejilla de posibles valores de r, entre 0 y 1, y para cada uno de ellos se obtiene el α que minimiza el error por validación cruzada (CV). En este caso, el error hace referencia al error cuadrático medio de los valores predichos frente a los reales de la partición que no estamos considerando para el entrenamiento.

A continuación, se muestra este procedimiento con más detalle.

- Paso 1. Para un valor de r predeterminado se realiza la estimación mediante CV del error para diferentes valores de α. En la Figura 4 se muestran estos errores $\pm$ 1 desviación estándar para $r = 1$. Se marca con líneas punteadas azules el valor de α que minimiza el error de la validación cruzada (α_{min}). También se señala el máximo

α posible que se encuentra a menos de una desviación típica del α_{min}. Este valor de α_{1se} se puede emplear como criterio alternativo de selección del valor de α óptimo, aunque en este ejemplo, se escogerá únicamente el α_{min}.

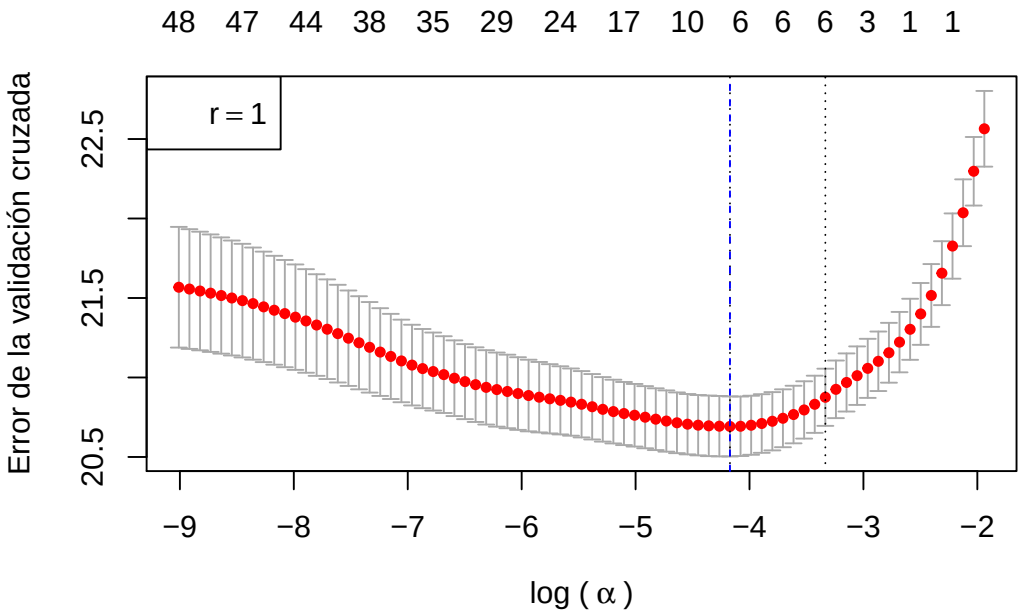

Figura 4: Error de la validación cruzada para la Red Elástica con $\alpha = 1$

- Paso 2. Se repite 200 veces el proceso de estimación del α_{min} por CV y se calculan las medianas de los α_{min} así como de los errores estimados. Al hacer 200 repeticiones del paso 1 se consigue reducir la varianza en el muestreo de la CV en los α_{min} y en el error. En la Figura 5 se muestra el kernel de la función de densidad de α_{min} y del error para $r = 1$.

- Paso 3. Por último, se crea una rejilla de valores de r, que en este caso consiste de 41 valores, obtenidos de dividir el espacio [0, 1] en 40 partes iguales. Para cada uno de los valores de r de la rejilla se repiten los pasos 1 y 2 y se escoge aquel valor de r que minimiza la mediana del error mínimo por CV, como se muestra en la Figura 6. Se ajusta la red con $r = A$, que es la que consigue minimizar el error por CV.

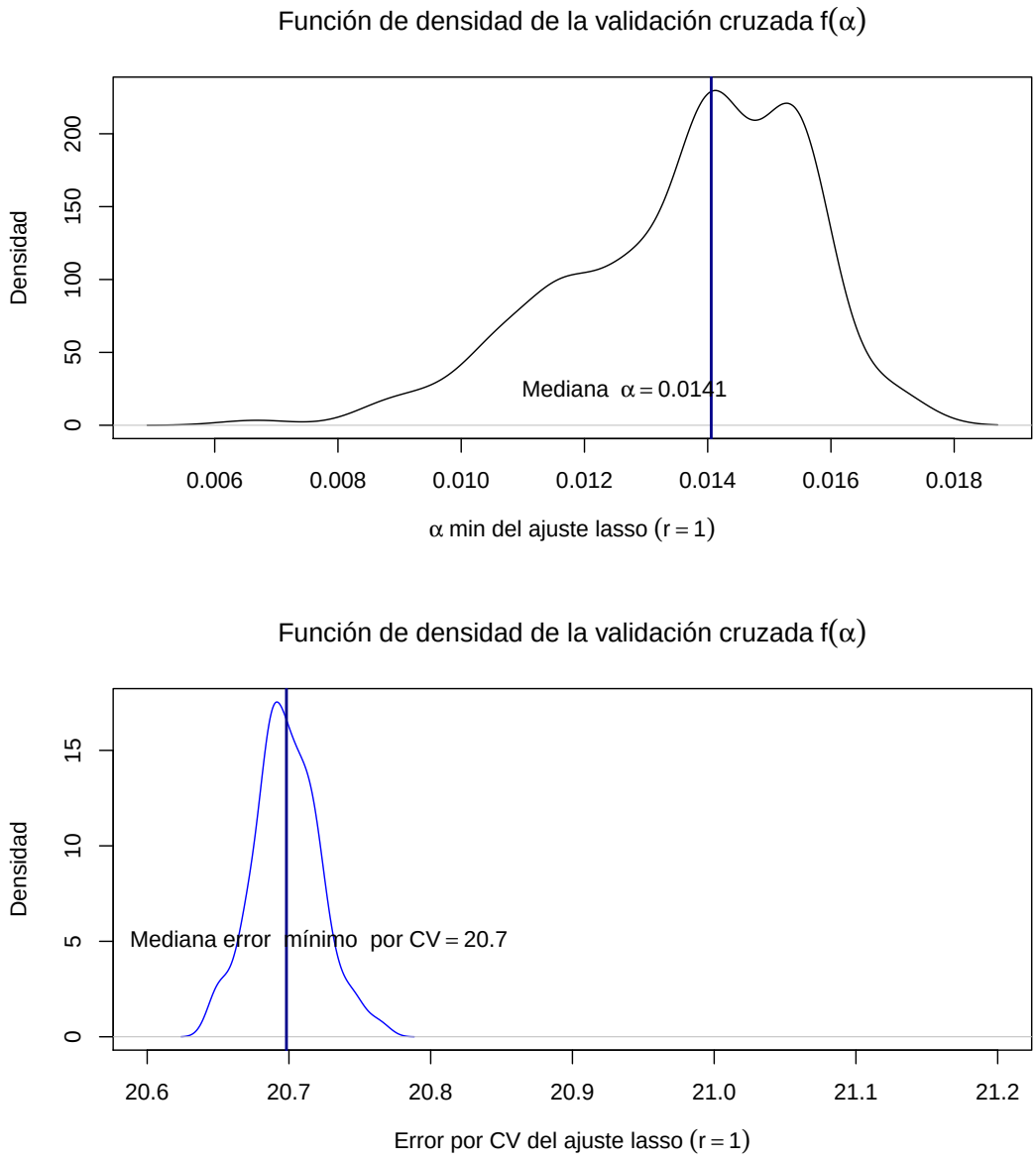

Figura 5: Funciones de densidad estimadas de α_{min} y de los errores de la validación cruzada para la Red Elástica con $r = 1$

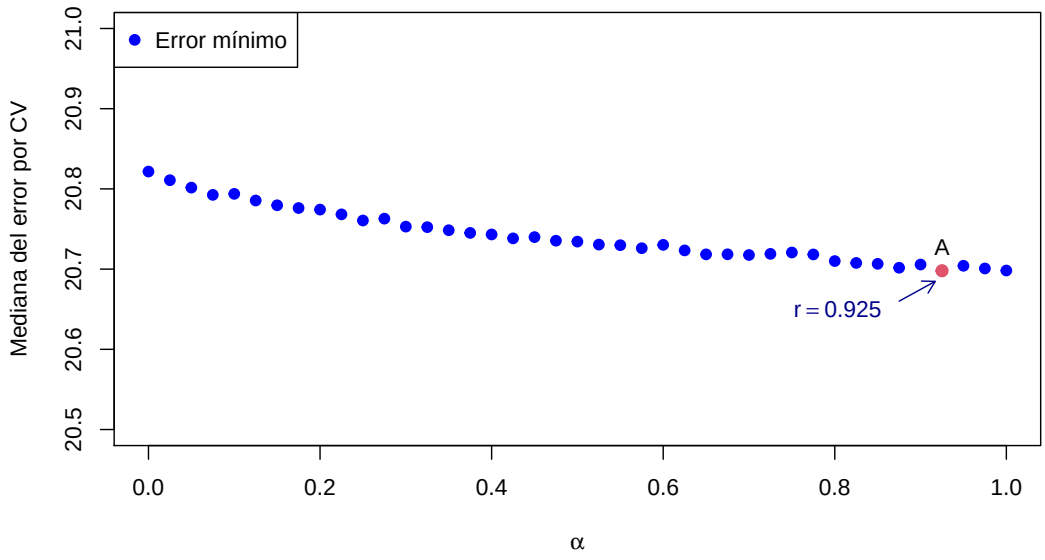

Figura 6: Medianas del error mínimo de la validación cruzada para cada valor de α

Red Elástica A $(r = 0.925)$ **ajustando con** α_{min}

Esta red es la que proporciona un menor error mínimo por CV. Penaliza de forma muy parecida al LASSO, ya que el r seleccionado está muy cercano a 1. En la Figura 7 se puede observar cómo la red ajusta los coeficientes dependiendo del α seleccionado. Se ha marcado con una línea roja el valor del $\log(\alpha_{min})$ que se ha calculado para minimizar el error. Esta penalización encoge bastante el valor de los coeficientes y, además, hace selección de variables ya que muchos de ellos se encogen hasta valer 0. De hecho, se observa que son únicamente 6 de los 48 posibles parámetros los que tienen una estimación de sus coeficientes con un valor distinto de 0. El α óptimo obtenido es: $\alpha = 0.015$ y la estimación de los coeficientes de regresión de las variables seleccionadas se muestran en la Tabla 1.

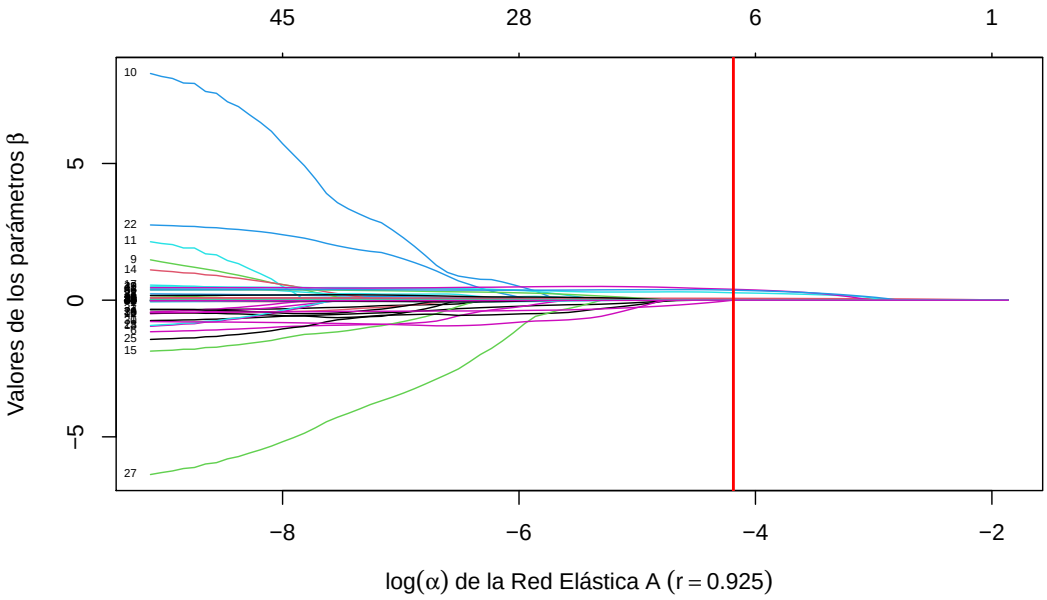

Figura 7: Valor de los coeficientes β según el valor de la penalización α para la Red Elástica A

Tabla 1: Resultados de la Red Elástica A

Variable	β
edad	0.067
cigarrillos fumados	0.012
diabetes	0.269
tratamiento diabetes	0.397
SBP	0.005
tratamiento hipertensión	0.365
α	0.015

Con la regresión mediante la Red Elástica se ha conseguido reducir el número de variables predictoras de las 48 que se tenían inicialmente a únicamente 7.

Material audiovisual

Accede al vídeo: Regresión Avanzada

6.4 Referencias bibliográficas

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2 edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning : with applications in R*. New York : Springer, [2013] ©2013.

Kennard, R. W. and Hoerl, A. E. (1970). Ridge regression: Biased estimation for no-northogonal problems. *Technometrics*, 12(1):55–67.

Rousseeuw, P. J. and Van Driessen, K. (2006). Computing lts regression for large data sets. *Data Mining and Knowledge Discovery*, 12(1):29–45.

Tibshirani, R. (1996). Regression Selection and Shrinkage via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Zou, H. and Hastie, T. (2005). Erratum: Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67(5):301–320.

6.5 Ejercicios resueltos

Ejercicio 1.

En este ejercicio profundizarás en los modelos de regresión penalizada. Carga el dataset **California housing**, y realiza un modelo de regresión *Ridge* de la mediana de los valores de las casas (variable respuesta Y) con las variables restantes (numéricas y categóricas) del dataset:

a) Con el hiperparámetro de encogimiento $\alpha = 1$.
b) Con el hiperparámetro de encogimiento $\alpha = 10^4$.
c) Compara ambos modelos con el modelo regresión lineal múltiple. ¿Qué se observa?

Solución

a) El primer paso, como viene siendo habitual es importar las librerías que se van a emplear en la resolución de los ejercicios:

```
# cargar librerias-----------------------------------------------
import pandas as pd
import numpy as np
from pandas.core.common import flatten
from plotnine import *
from array import *
import scipy.stats as stats
import math
import matplotlib as mpl
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model
```

```
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
```

Una vez que se han importado las librerías, se carga el dataset **California housing** y se prepara la matriz de diseño y la variable respuesta del modo que se hizo en los ejercicios resultados del tema anterior.

```
# path que se va a crear en nuestro sistema----------------------
HOUSING_PATH = os.path.join("datasets", "housing")

# definir una funcion que cargue el csv en un dataframe----------
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
# cargar base
housing = load_housing_data()
# separar variable respuesta del dataset-------------------------
respuesta = housing["median_house_value"].copy()
housing = housing.drop("median_house_value", axis=1)
# definir var numerica-------------------------------------------
housing_num = housing.drop("ocean_proximity", axis=1)
# importar el "imputador"----------------------------------------
from sklearn.impute import SimpleImputer
# importar el "estandarizador"-----------------------------------
from sklearn.preprocessing import StandardScaler
# importar la clase pipeline"------------------------------------
from sklearn.pipeline import Pipeline
# definir el pipeline--------------------------------------------
num_pipeline = Pipeline([
        ("imputador", SimpleImputer(strategy="median")),
        ("std_scaler", StandardScaler()),
```

```
    ])
# aplicar el pipeline-----------------------------------------------
housing_num_tr = num_pipeline.fit_transform(housing_num)
# importar clases---------------------------------------------------
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
# separar dataset en variables numericas y variable categorica---
num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]
# definir full pipeline---------------------------------------------
full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attribs),
        ("cat", OneHotEncoder(drop = "first"), cat_attribs),
    ])
housing_prepared = full_pipeline.fit_transform(housing)
```

Una vez que se ha obtenido la matriz de diseño X, y la variable respuesta por separado, se puede realizar la regresión penalizada. Para ello emplearemos una clase de tipo *Ridge* de la librería *sklearn*.

```
# importar clase----------------------------------------------------
from sklearn.linear_model import Ridge
# ajustar el modelo-------------------------------------------------
ridge_reg = Ridge(alpha = 1, solver = "auto")
ridge_reg.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo-----------------------------------
# intercepto
```

```
## Ridge(alpha=1)
```

```
print(ridge_reg.intercept_)
# coeficientes de regresion
```

```
## 219251.8547047686
```

```
print(ridge_reg.coef_)
```

```
## [-52925.7575442  -53744.69781116  13319.85812079 -10293.66412867
##   29937.78985234 -44479.43818497  29692.21439608  73624.96037269
##  -39789.87191589 130031.63921427  -3697.62935336   4750.43033774]
```

El valor de $\alpha = 1$ no aplica apenas penalización y los resultados obtenidos son prácticamente los mismos que se obtenían en el modelo de regresión lineal (ver ejercicios resueltos del Tema 5).

b) Si se aumenta el valor de la penalización hasta $\alpha = 10^4$ se observa cómo los valores de las estimaciones de los coeficientes de regresión ya disminuyen.

```
# importar clase-----------------------------------------------
from sklearn.linear_model import Ridge
# ajustar el modelo--------------------------------------------
ridge_reg_b = Ridge(alpha = 1e4, solver = "auto")
ridge_reg_b.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo------------------------------
# intercepto
```

```
## Ridge(alpha=10000.0)
```

```
print(ridge_reg_b.intercept_)
# coeficientes de regresion
```

```
## 213248.3244996107
```

```
print(ridge_reg_b.coef_)
```

```
## [ -9760.89761151 -12228.34267466  11199.36514172   5640.33695302
##    4851.91951441  -9762.34854409   5734.63871066  50971.33381983
```

```
##   -24316.41748776      89.44198348    5253.33235919    5765.58163202]
```

Se puede calcular (como se hizo en el Tema 5) cuál era la estimación de los coeficientes de regresión para el modelo de regresión lineal general (sin penalizar), y comparar las 3 series de coeficientes obtenidas mediante el gráfico de barras que se muestra en la Figura 8. Se observa que los valores de la estimación de los coeficientes de la regresión penalizada para un $\alpha = 10^4$ son mucho menores.

```
# importar clase-----------------------------------------------
from sklearn.linear_model import LinearRegression
# ajustar el modelo--------------------------------------------
lm1 = LinearRegression()
lm1.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo------------------------------
# intercepto
```

```
## LinearRegression()
```

```
print(lm1.intercept_)
# coeficientes de regresion
```

```
## 219237.0006433122
```

```
print(lm1.coef_)
```

```
## [-52952.95152846 -53767.62485624  13312.88334575 -10320.06092603
##   29920.76507621 -44490.47744263  29746.22226671  73636.15586366
##  -39766.3987444  156065.71982235  -3697.40166109   4758.75361226]
```

```
fig = plt.figure(figsize=(8, 8))
# ancho de barra-----------------------------------------------
barWidth = 0.25
# definir posicion barras series------------------------------
r1 = np.arange(len(lm1.coef_))
```

```
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
# pintar las barras-----------------------------------------------
plt.bar(r1, lm1.coef_, color = "blue",
width = barWidth, edgecolor = "white", label = "Lineal");
plt.bar(r2, ridge_reg.coef_, color = "red",
width = barWidth, edgecolor = "white", label = "Ridge a");
plt.bar(r3, ridge_reg_b.coef_, color = "green",
width = barWidth, edgecolor = "white", label = "Ridge b");
plt.xticks([r + barWidth for r in range(len(lm1.coef_))],
['B1', 'B2', 'B3', 'B4',
 'B5', 'B6', 'B7', 'B8',
 'B9', 'B10', 'B11', 'B12']);
plt.legend();
plt.xlabel("Variable");
plt.ylabel("Beta");
plt.show(fig)
```

Ejercicio 2

Sobre el dataset **California housing**, realiza un modelo de regresión *LASSO* de la mediana de los valores de las casas (variable respuesta Y) con las variables restantes (numéricas y categóricas) del dataset:

a) Con el hiperparámetro de encogimiento $\alpha = 1$.

b) Con el hiperparámetro de encogimiento $\alpha = 10^4$.

c) Compara ambos modelos con los modelos del ejercicio 1. ¿Qué se observa?

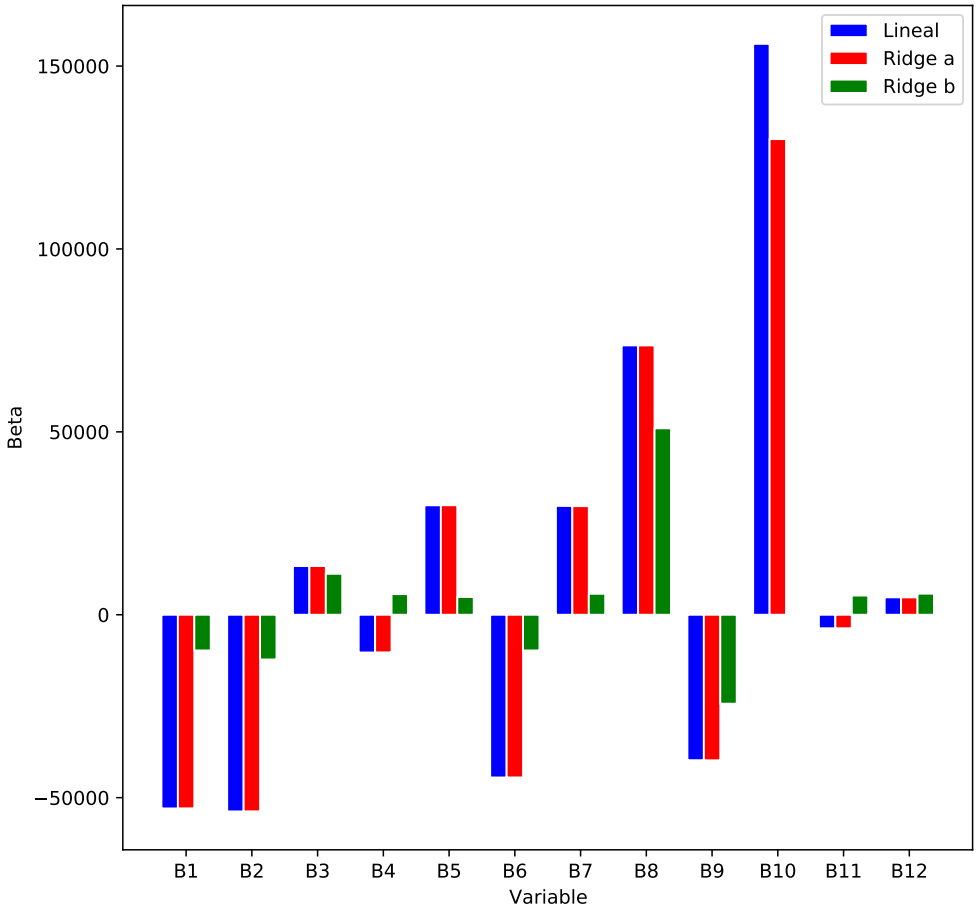

Figura 8: Gráfico de barras de los coeficientes de regresión obtenidos para los diferentes modelos

Solución

Para realizar la regresión de tipo LASSO se utiliza la clase de tipo *LASSO* de la librería *sklearn*.

```
# importar clase----------------------------------------------
from sklearn.linear_model import Lasso
# ajustar el modelo-------------------------------------------
lasso_reg = Lasso(alpha = 1)
lasso_reg.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo-----------------------------
# intercepto
```

```
## Lasso(alpha=1)
```

```
print(lasso_reg.intercept_)
# coeficientes de regresion
```

```
## 219239.371611401
```

```
print(lasso_reg.coef_)
```

```
## [-52936.10528879 -53753.78630134  13312.79044653 -10299.69365603
##   29909.69564186 -44483.79617817  29730.65349082  73630.76579416
##  -39774.84408823 151938.98956871  -3679.84442713   4753.79358651]
```

El valor de $\alpha = 1$ no aplica apenas penalización y los resultados obtenidos son prácticamente los mismos que se obtenían en el modelo de regresión lineal (ver ejercicios resueltos del Tema 5).

b) Si se aumenta el valor de la penalización hasta $\alpha = 10^4$ se observa como los valores de las estimaciones de los coeficientes de regresión ya disminuyen, y en este caso, además, al hacerse 0 algunos de los coeficientes se está produciendo selección de

variables.

```
# importar clase----------------------------------------------
from sklearn.linear_model import Lasso
# ajustar el modelo-------------------------------------------
lasso_reg_b = Lasso(alpha = 1e4)
lasso_reg_b.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo-----------------------------
# intercepto
```

```
## Lasso(alpha=10000.0)
```

```
print(lasso_reg_b.intercept_)
# coeficientes de regresion
```

```
## 218717.123954823
```

```
print(lasso_reg_b.coef_)
```

```
## [    -0.             -0.           5925.06123247      0.
##       0.             -0.              0.          65973.15122729
##  -37370.99334873      0.              0.              0.        ]
```

Se pueden comparar los valores de la regresión lineal con las regresiones Ridge y LASSO con $\alpha = 10^4$. En la Figura 9 Se observa que aunque en la regresión de tipo Lasso desaparezcan algunas variables, en otras se estima un coeficiente de regresión más alto que con los otros modelos. Esto es debido a que cuando dos variables están correlacionadas tiende a descartar una y a ajustar por la otra.

```
fig = plt.figure(figsize=(8, 8))
# ancho de barra----------------------------------------------
barWidth = 0.25
# definir posicion barras series-----------------------------
r1 = np.arange(len(lm1.coef_))
```

```
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
# pintar las barras-----------------------------------------------
plt.bar(r1, lm1.coef_, color = "blue",
width = barWidth, edgecolor = "white", label = "Lineal");
plt.bar(r2, ridge_reg_b.coef_, color = "red",
width = barWidth, edgecolor = "white", label = "Ridge b");
plt.bar(r3, lasso_reg_b.coef_, color = "green",
width = barWidth, edgecolor = "white", label = "Lasso b");
plt.xticks([r + barWidth for r in range(len(lm1.coef_))],
['B1', 'B2', 'B3', 'B4',
 'B5', 'B6', 'B7', 'B8',
 'B9', 'B10', 'B11', 'B12']);
plt.legend();
plt.xlabel("Variable");
plt.ylabel("Beta");
plt.show()
```

Ejercicio 3

Sobre el dataset **California housing**, realiza un modelo de regresión de Red Elástica de la mediana de los valores de las casas (variable respuesta Y) con las variables restantes (numéricas y categóricas) del dataset

a) Fijando $r = 0.1$.

b) Fijando $r = 0.95$.

En ambos casos, ajusta α por validación cruzada (Usa $k = 10$ particiones).

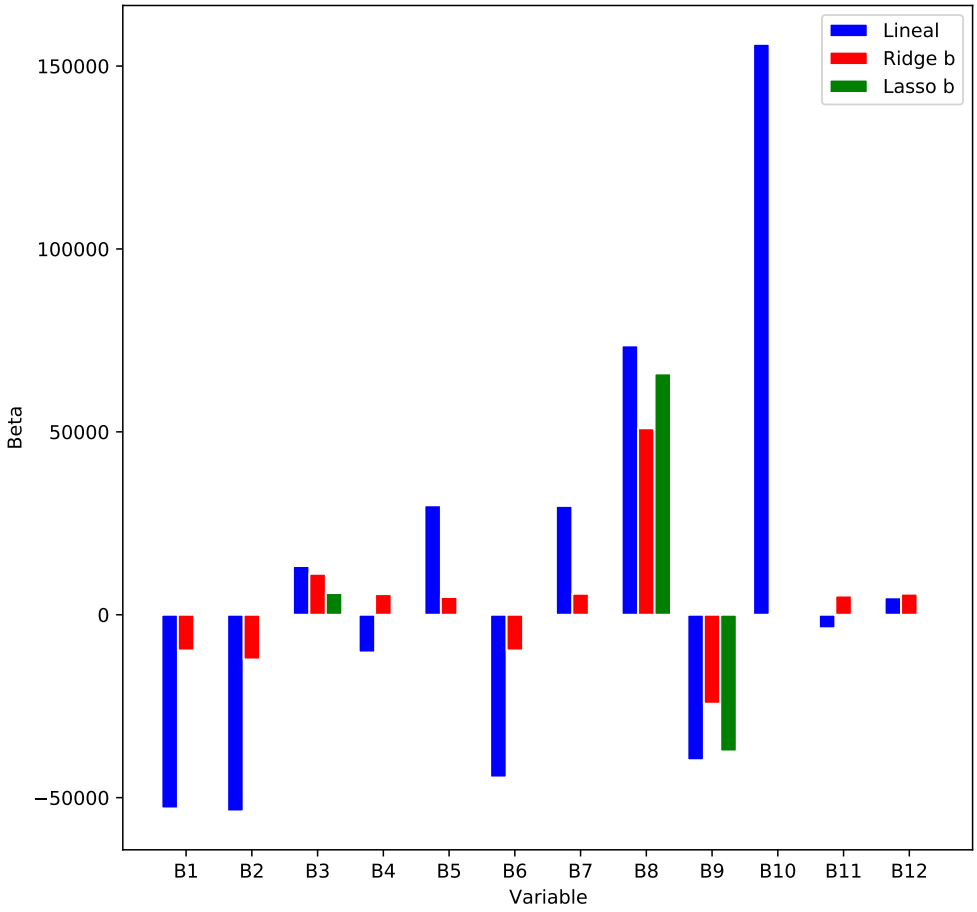

Figura 9: Gráfico de barras de los coeficientes de regresión obtenidos para RLM, Ridge y LASSO

Solución

a) En la primera Red, se fija el hiperparámetro r en 0.1 y se ajusta el modelo de tipo red elástica realizando validación cruzada utilizando la clase *ElasticNetCV* del paquete *sklearn.linear_model*. En los argumentos de la clase definiremos las particiones de la CV con el argumento *cv* y *r* con el argumento *l1_ratio: (cv = 10 y l1_ratio = 0.1)*.

```
# semilla para que los resultados sean los mismos----------------
np.random.seed(3)
# importar clase-------------------------------------------------
from sklearn.linear_model import ElasticNetCV
# ajustar el modelo----------------------------------------------
e_net = ElasticNetCV(cv = 10, l1_ratio = 0.1)
e_net.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo--------------------------------
# intercepto
```

```
## ElasticNetCV(cv=10, l1_ratio=0.1)
```

```
print(e_net.intercept_)
# coeficientes de regresion
```

```
## 206865.4749645959
```

```
print(e_net.coef_)
```

```
## [ -7.3285382  -23.12318484  16.94967114  21.48093501   7.83412633
##   -3.90892666  10.4757704  110.83304095 -36.29600822   0.
##    8.01288948   7.55581778]
```

Se observa que la red A no realiza apenas selección de variables ya que sólo uno de los parámetros es encogido hasta 0.

b) En esta Red, al aumentar el valor de r se espera un comportamiento diferente. Veamos:

```
# semilla para que los resultados sean los mismos----------------
np.random.seed(3)
# importar clase-------------------------------------------------
from sklearn.linear_model import ElasticNetCV
# ajustar el modelo----------------------------------------------
e_net_b = ElasticNetCV(cv = 10, l1_ratio = 0.95)
e_net_b.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo--------------------------------
# intercepto
```

```
## ElasticNetCV(cv=10, l1_ratio=0.95)
```

```
print(e_net_b.intercept_)
# coeficientes de regresion
```

```
## 208267.51670197048
```

```
print(e_net_b.coef_)
```

```
## [-1469.95946607 -3059.90966939  2747.34800787  2439.66788593
##    801.03210661 -1171.1003709   1151.60483666 15113.62768328
##  -5336.05497542     0.          1215.79217006  1141.77891116]
```

En la Figura 10 se muestran los coeficientes de las dos Redes Elásticas ajustadas.

```
fig = plt.figure(figsize=(8, 8))
# ancho de barra-------------------------------------------------
barWidth = 0.25
# definir posicion barras series--------------------------------
r1 = np.arange(len(e_net.coef_))
r2 = [x + barWidth for x in r1]
```

```
r3 = [x + barWidth for x in r2]
# pintar las barras-----------------------------------------------
plt.bar(r2, e_net.coef_, color = "red",
width = barWidth, edgecolor = "white", label = "Red El. a");
plt.bar(r3, e_net_b.coef_, color = "green",
width = barWidth, edgecolor = "white", label = "Red El. b");
plt.xticks([r + barWidth for r in range(len(lm1.coef_))],
['B1', 'B2', 'B3', 'B4',
 'B5', 'B6', 'B7', 'B8',
 'B9', 'B10', 'B11', 'B12']);
plt.legend();
plt.xlabel("Variable");
plt.ylabel("Beta");
plt.show()
```

Se obtienen unos valores muy distintos de los coeficientes. En la primera Red, más cercana a la regresión Ridge, los coeficientes se han encogido demasiado. En cambio, en la segunda red se obtienen unos valores más grandes de los coeficientes. Si tenemos dudas entre ambas Redes, podemos poner la lista de posibles valores de r en el argumento *l1_ratio* y ver con cuál de los dos modelos se obtiene un menor error de test.

```
# semilla para que los resultados sean los mismos----------------
np.random.seed(3)
# importar clase-------------------------------------------------
from sklearn.linear_model import ElasticNetCV
# ajustar el modelo----------------------------------------------
e_net_c = ElasticNetCV(cv = 10, l1_ratio = [0.1, 0.95])
e_net_c.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo--------------------------------
# intercepto
# print(e_net_c.intercept_)
```

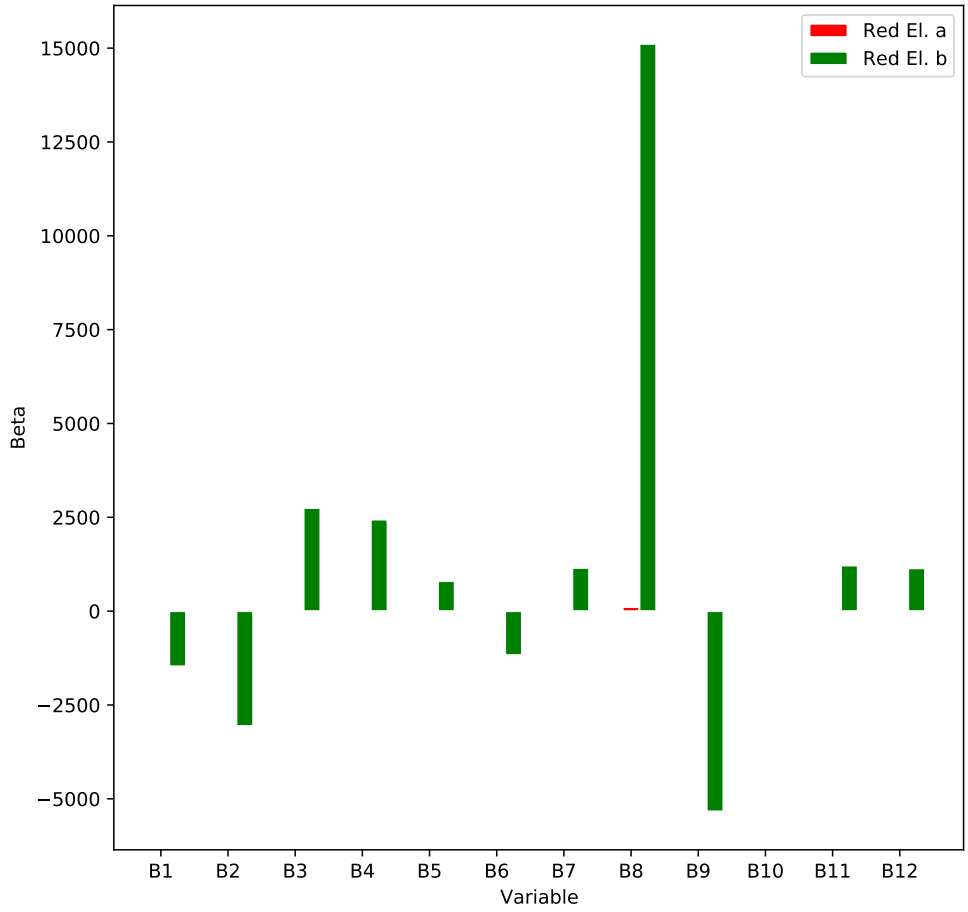

Figura 10: Gráfico de barras de los coeficientes de regresión obtenidos para RLM, RE A y RE B

```
# coeficientes de regresion
# print(e_net_c.coef_)
```

```
## ElasticNetCV(cv=10, l1_ratio=[0.1, 0.95])
```

```
print(e_net_c.l1_ratio_)
# se puede comprobar que el error de test es mayor con r = 0.1---
# error = e_net_c.mse_path_
# np.mean(error, axis = 2)
```

```
## 0.95
```

Se observa que se ha escogido el valor de 0.95, por lo que es el valor que minimiza el error de test por CV (el cual se puede analizar en el atributo *.mse_path_*).