Lasso

¡Comencemos!

Competencias

- Conocer la mecánica de regularización en los métodos Lasso.
- Conocer las normas L1.

Introducción

En esta sección conocerás las técnicas de regulación en los métodos Lasso y las normas L1.

¡Vamos con todo!

Lasso: Norma L1

Otro método de contracción para la regresión lineal que estudiaremos es el llamado Lasso (acrónimo de Least Absolute Shrinkage and Selection Operator), este método opera de una forma muy similar a Ridge. Sin embargo, sí permite hacer selección de atributos (descartar atributos poco relevantes para el modelo) a diferencia de Ridge, que simplemente los encogía cada vez más. Lasso opera penalizando la función objetivo de la siguiente forma:

$$\beta_{\text{Lasso}} = \underset{\beta}{\text{argmin}} \sum_{i}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

A diferencia de Ridge, Lasso penaliza por la norma

$$\ell_1 = |\beta|$$

Esto se traduce en que finalmente el modelo será capaz de eliminar ciertos atributos de la regresión, los cuales considerará que no son suficientemente relevantes.

Veamos cómo caen los pesos de los coeficientes al aplicar Lasso a los mismos datos anteriores:

```
from sklearn.linear_model import Lasso, LassoCV

names_regressors = X.columns
alphas_ = np.logspace(0, -3, base = 10)
coefs_lasso = []
cv_err_lasso = []
model_lasso = Lasso(fit_intercept = True)

for a in alphas_:
   model_lasso.set_params(alpha = a)
   model_lasso.fit(X_train, y_train)
   coefs_lasso.append(model_lasso.coef_)
   dummy,cv_err_estimates = gfx.cv_error(X_train,y_train,k = 10, method
= 'lasso', alpha = a)
   cv_err_lasso.append(np.mean(cv_err_estimates))

for y_arr, label in zip(np.squeeze(coefs_lasso).T, names_regressors):
   plt.plot(alphas_, y_arr, label = label)

plt.legend()
```

```
plt.xscale('log')

plt.title("Lasso Regression: coeficientes vs par. de regularización",
fontsize = 14)
plt.axis("tight")
plt.legend(loc = "center left", bbox_to_anchor=(1, .5));
```

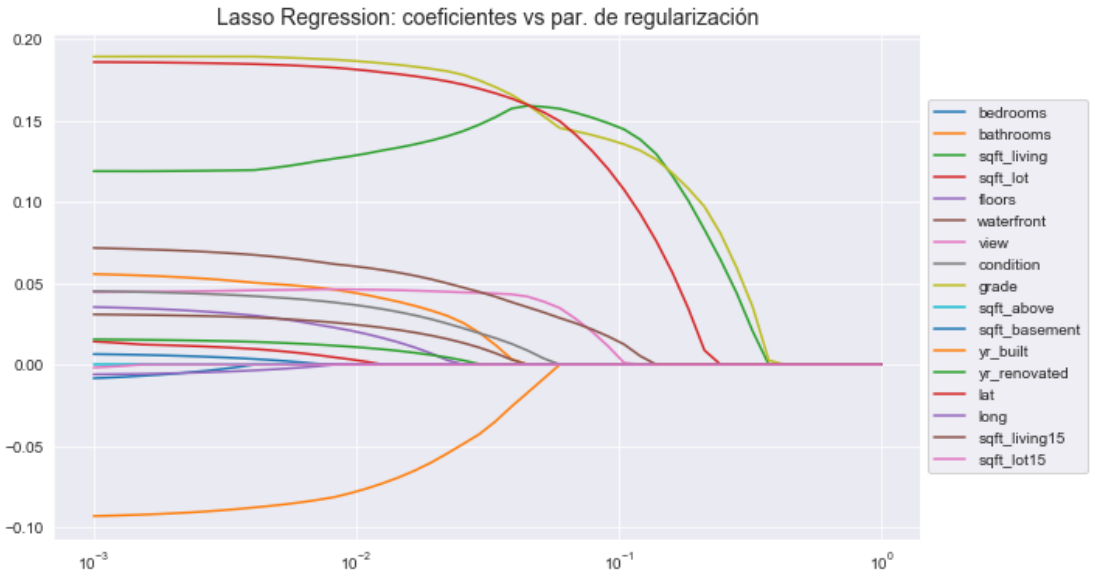

Imagen 1. Gráfica de Lasso.
Fuente: Desafío Latam.

```
plt.plot(alphas_, np.sqrt(cv_err_lasso),'o-',color='dodgerblue')
plt.xscale("log")
plt.title("Lasso Regression: RMSE Cross-Validation para cada $\lambda$",
fontsize = 14);
```

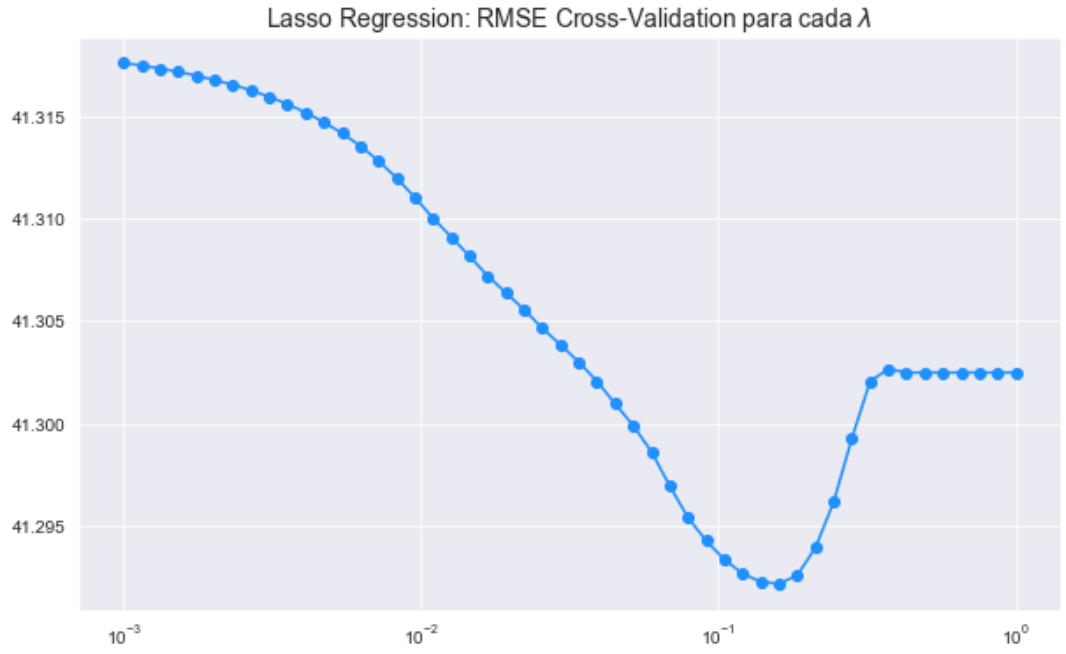

Imagen 2. Validación de error para Lasso.
Fuente: Desafío Latam.

- Los pesos caen de forma mucho más abrupta que en Ridge, producto de la naturaleza de Lasso.
- Ciertos atributos son restados del modelo de forma mucho más temprana que otros, reduciendo rápidamente el número de atributos del modelo, o dicho de otra forma, seleccionando atributos más relevantes para explicar los datos.

Nuevamente veremos el error de validación cruzada para Lasso, nuestro objetivo será encontrar ya sea un punto en el que observemos convergencia, o uno en el que el error sea un mínimo local:

```
alphas_ = np.logspace(0, -3, base = 10)
lasso_cv = LassoCV()
model_lasso = lasso_cv.fit(X_train, y_train)
report_regularization(model_lasso, X_test, y_test)
```

```
Valor del parámetro de regularización: 0.0003753797351375845
Coeficientes finales:
[-0.01024032 0.05663014 0.11897972 0.01634357 0.03634553 0.03122292
0.04432799 0.04579056 0.18927954 0.         0.00698097 -0.09423091
0.01580353 0.18619272 -0.00642786 0.07259197 -0.0040628 ]
R-squared: 0.768539353019913
Mean Squared Error: 0.06231357678975419
```

Ahora que ya hemos visto el efecto que produce Lasso sobre los coeficientes, al igual que en Ridge, nos interesa buscar el valor de α relativamente previo al valor en el que se dispara el error de validación:

El problema de Ridge es que no es capaz de hacer una selección de atributos afectiva debido a que nunca hace exactamente 0 los coeficientes, el problema de Lasso es que muchas veces es muy agresivo en la reducción de atributos, el modelo que veremos a continuación es una mezcla de ambos y trata de quedarse con lo mejor de ambos modelos de regularización.