Elastic Net

¡Comencemos!

Competencias

- Conocer la mecánica de regularización en los métodos Elastic-net.
- Implementar los métodos con la librería sklearn.
- Visualizar el comportamiento de las variables del método.
- Analizar el comportamiento de las variables del método.

Introducción

En esta sección conocerás acerca de la mecánica de regulación en los métodos Elastic-net; como implementar los métodos de la librería sklearn. Además, visualizarás y analizaremos el comportamiento de las variables de método.

¡Vamos con todo!

Elastic Net

Elastic Net es el último modelo de regularización que veremos, este modelo combina ambas penalizaciones vistas, Lasso y Ridge bajo el argumento de formar una regularización que sea capaz de penalizar efectivamente los coeficientes de los atributos, como lo hace Ridge, y además sea capaz de realizar selección de atributos como lo hace Lasso. La función objetivo de Elastic Net se conforma de la siguiente manera:

$$\beta_{\text{ElasticNet}} = \underset{\beta}{\text{argmin}} \sum_{i}^{n}(y_i - \hat{y_i})^2 + \lambda_1 \sum_{j=1}^{p}|\beta_j| + \lambda_2 \sum_{j=1}^{p}\beta_j^2$$

Algo importante que debe tenerse en cuenta sobre este modelo es que, a diferencia de Lasso y Ridge, ahora tenemos que encontrar los valores para dos parámetros de regularización en lugar de uno.

Veamos Elastic Net en la práctica, el objeto `ElasticNet` de `sklearn` define el juego entre los parámetros de ambas penalizaciones de forma no tan intuitiva, por lo que trabajaremos con los valores predefinidos de penalización para cada modelo que define el objeto por defecto. Si se quiere modificar estos valores se puede consultar la documentación de `ElasticNet`:

```
from sklearn.linear_model import ElasticNet, ElasticNetCV

names_regressors = X.columns
alphas_ = np.logspace(0, -3, base = 10)
coefs_elastic_net = []
cv_err_elastic_net = []
model_elastic_net = ElasticNet(fit_intercept = True)


for a in alphas_:
   model_elastic_net.set_params(alpha = a)
   model_elastic_net.fit(X_train, y_train)
   coefs_elastic_net.append(model_elastic_net.coef_)
   dummy,cv_err_estimates = gfx.cv_error(X_train,y_train,k = 10, method =
'enet', alpha = a)
   cv_err_elastic_net.append(np.mean(cv_err_estimates))


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

plt.legend()
```

```
plt.xscale("log")

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

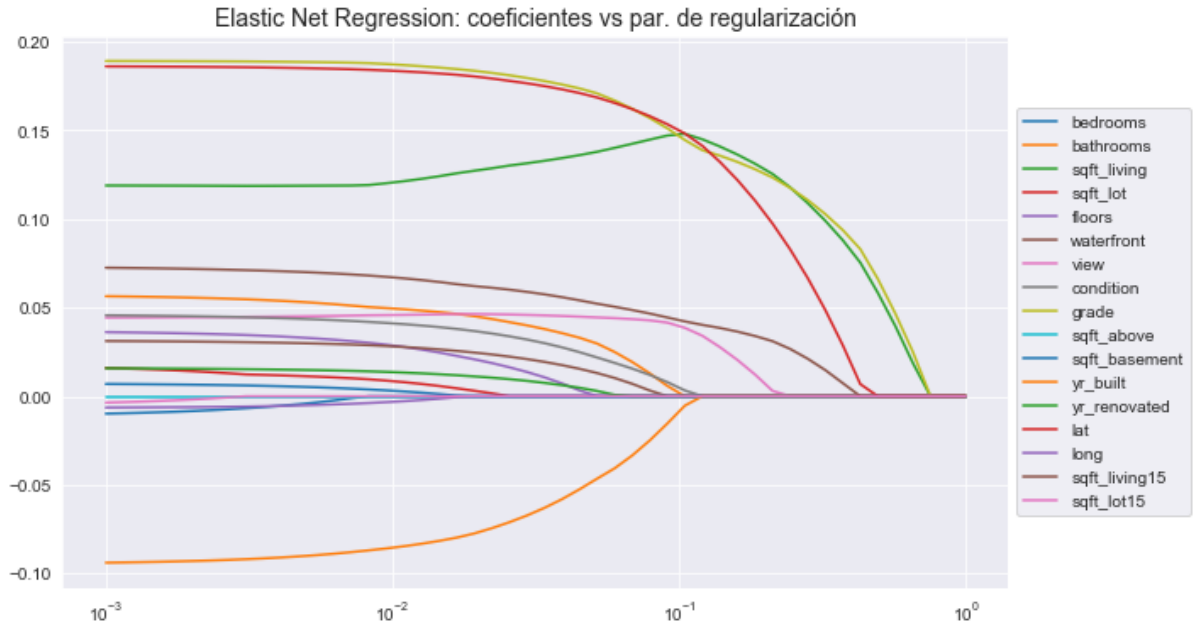

Imagen 1. Gráfico Elastic Net.
Fuente: Desafío Latam.

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

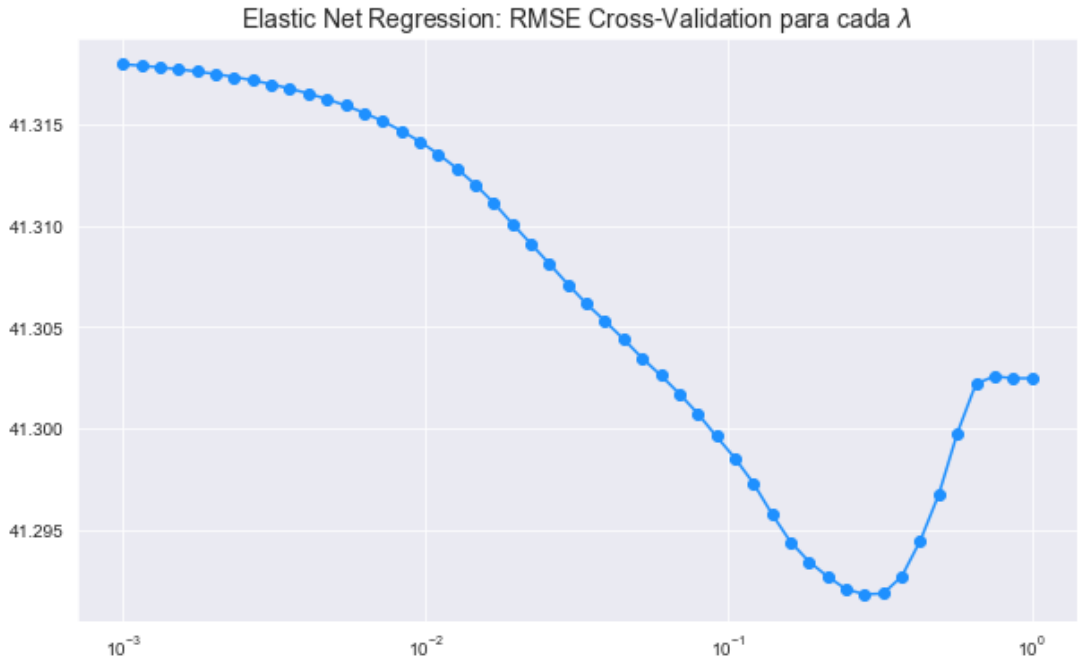

Imagen 2. Validación error de Elastic Net.
Fuente: Desafío Latam.

Podemos ver como Elastic Net hace en efecto decaer los pesos de los coeficientes de la misma forma en que lo hace Ridge, también podemos observar como ciertos atributos son restados del modelo al hacer sus coeficientes igual a cero.

Como mencionamos anteriormente la forma en la que la librería sklearn implementa el regularizador de `ElasticNet` es un poco más complicada de lo que desearíamos, de alguna forma trata de simplificar el uso del método, para configurar los parámetros de regularización por separado hay que modificar los parámetros `l1_ratio` y `alpha`, sin embargo, nosotros experimentaremos sólo con alpha y buscaremos el valor que nos sea conveniente:

```
alphas_ = np.logspace(0, -3, base = 10)
elastic_cv = ElasticNetCV(cv = 10)
model_elastic = elastic_cv.fit(X_train, y_train)

report_regularization(model_elastic, X_test, y_test)
```

```
Valor del parámetro de regularización: 0.0007507594702751691
Coeficientes finales:
[-0.01020259 0.05662165 0.11892443 0.01633775 0.03635761 0.03121637
0.04433303 0.04578218 0.18915541 0.         0.0070188 -0.09414242
0.01581814 0.18614231 -0.00645338 0.07265358 -0.00404842]
```

```
R-squared: 0.768536429557952
Mean Squared Error: 0.06231436384091725
```

Referencias

- Los ejemplos de la lectura se basan en James, G; Witten, D; Hastie, T; Tibshirani, R. 2013. An Introduction to Statistical Learning. Ch6: Linear Model Selection and Regularization.
- Aquellos que deseen profundizar en los aspectos teóricos de los métodos de regularización pueden consultar Efron, B; Hastie, T. 2016. Computer Age Statistical Inference. Algorithms, Evidence and Data Science. Ch 7. James-Stein Estimation and Ridge Regression. Ch 16. Sparse Modeling and the Lasso.