Metodo automático de selección

Metodo Ridge o Regrresión Ridge

> La regresión Ridge es una variante de la regresión lineal que modifica la función de costo (o pérdida) de la regresión lineal agregando un término de penalización. La idea es ajustar el modelo de tal manera que los coeficientes del modelo no crezcan demasiado grandes, lo cual puede ayudar a evitar el **sobreajuste**.

La función de costo en la regresión Ridge se define matemáticamente como:

$$J(\beta) = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$
$$= ||y - X\beta||^2 + \lambda||\beta||^2$$

Donde:

- λ (lambda) es el parámetro de regularización que controla la fuerza de la penalización:
  El parametro de regularización: el valor de `alpha` en `Ridge(alpha=0.5)`).
- β son los coeficientes del modelo
- $||\beta||^2$ es la norma L2 al cuadrado de los coeficientes
- X es la matriz de características
- y es el vector de valores objetivo

Cuanto mayor sea el valor de λ, mayor será la penalización sobre los coeficientes grandes, lo que resulta en un modelo más simple y regularizado.

λ Parametro:

- **λ=0**, el modelo se comporta como una regresión lineal estándar, sin regularización.
- **Cuando λ es grande** (por ejemplo, $\lambda \to \infty$), el modelo tiende a minimizar los coeficientes de los parámetros, lo que puede hacer que el modelo sea muy simple (posiblemente subajustado).
- **Un valor intermedio de λ** (como 0.5 en tu ejemplo) equilibra la relación entre el ajuste del modelo y la complejidad del mismo, ayudando a reducir el sobreajuste sin hacer que el modelo sea demasiado simple.

Conclusión.

la regresión Ridge se utiliza para mejorar la generalización de los modelos lineales, especialmente en situaciones donde hay muchas variables o

correlaciones entre ellas

EN PYTHON

```
reg = linear_model.Ridge(alpha = .5)
reg.fit(X,y)
```

Coef_ e intercept_

coef_:

Este atributo contiene los coeficientes θ del modelo para cada una de las características (variables independientes) del conjunto de datos

En el contexto de la **regresión Ridge**, cuando los coeficientes son **más bajos**, esto indica una **menor influencia** o **importancia** de las características correspondientes sobre la variable dependiente, lo cual es un efecto directo de la regularización Ridge sobre los coeficientes.

intercept_

Este atributo contiene el **intercepto** θ (también conocido como término independiente) del modelo.

EN PYTHON

```
from sklearn.linear_model import Ridge

# Datos de ejemplo
X = [[1, 2], [3, 4], [5, 6]]
y = [7, 8, 9]
```

```
# Ajustando el modelo
model = Ridge(alpha=0.5)
model.fit(X, y)

# Resultados
print("Coeficientes:", model.coef_)
print("Intercepto:", model.intercept_)
```

SCORE

El score (coeficiente de determinación R^2) en scikit-learn se calcula usando la siguiente fórmula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Donde:

- y_i son los valores reales
- ŷ_i son los valores predichos
- ȳ es la media de los valores reales

El score varía entre 0 y 1, donde:

- 1 indica una predicción perfecta
- 0 indica que el modelo no es mejor que una línea horizontal en la media
- Valores negativos son posibles e indican un ajuste peor que una línea horizontal

```
# Calculando el score
score = model.score(X, y)
print("R² Score:", score)
```

Si es cercano a 1 el error es muy bajo, si es cercano a cero el error es muy alto

¿Qué determina el valor de α?

1. **El sesgo vs. la varianza**:
   - **Sesgo**: Cuando el valor de α es **muy pequeño** (cercano a cero), el modelo se ajusta de manera muy precisa a los datos de entrenamiento (bajo sesgo), pero puede tener un alto **sobreajuste** (alta varianza) y no generalizar bien a datos nuevos.

     α\alpha

   - **Varianza**: Cuando α es **muy grande**, la regularización es muy fuerte, y los coeficientes del modelo se reducen a valores cercanos a cero, lo que hace que el modelo sea **más simple** (bajo sobreajuste), pero puede tener un **alto sesgo** porque no ajusta bien los datos.

     α\alpha

   El objetivo es encontrar un valor de α\alphaα que logre un **compromiso** entre el sesgo y la varianza, minimizando el error tanto en los datos de entrenamiento como en los datos de prueba.

2. **La escala de las características**:
   - La regresión Ridge es sensible a la **escala de las características** (las unidades de medida de las variables), por lo que es muy importante **normalizar o estandarizar** los datos antes de aplicar la regresión Ridge. Si las variables tienen diferentes escalas, la regularización puede afectar más a las variables con mayores valores.
   - Si no se estandarizan las características, un valor de α podría ser menos efectivo o incluso contraproducente.

     α\alpha

¿Cómo se determina el valor de α\alphaα?

El valor de α\alphaα se **determina empíricamente** a través de la validación del modelo. Algunas formas comunes de elegir α\alphaα son:

1. **Validación cruzada**:

- Una de las formas más efectivas de seleccionar el valor de α es utilizando **validación cruzada**. Esto consiste en dividir los datos en varios subconjuntos y entrenar el modelo con diferentes valores de α para encontrar el que minimice el error en un conjunto de validación.

  α\alpha

  α\alpha

- Puedes usar técnicas como **K-fold cross-validation** para probar diferentes valores de α y seleccionar el que mejor generalice a los datos no vistos.

  α\alpha

Ejemplo en Python con `RidgeCV` (que selecciona el mejor α\alphaα utilizando validación cruzada):

```
from sklearn.linear_model import RidgeCV
model = RidgeCV(alphas=[0.1, 1.0, 10.0, 100.0], store_cv_valu
model.fit(X_train, y_train)
print("Mejor valor de alpha:", model.alpha_)
```

Búsqueda de cuadrícula (Grid Search):

- Otra técnica popular es la **Grid Search** para buscar exhaustivamente en un rango de valores de α. Este método prueba sistemáticamente varios valores predefinidos de α y selecciona el que produce el mejor rendimiento según una métrica de validación, como el **error cuadrático medio (MSE)**.α\alphaα\alpha

Ejemplo con GridSearchCV:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
parameters = {'alpha': [0.1, 1.0, 10.0, 100.0]}
ridge = Ridge()
grid_search = GridSearchCV(ridge, parameters, cv=5)
```

```
grid_search.fit(X_train, y_train)
print("Mejor alpha encontrado:", grid_search.best_params_['alpha
```

- **Regresión Ridge y otros métodos de regularización**:
  - Aunque el ajuste de α en Ridge se realiza por validación cruzada, también puedes comparar la **regresión Ridge** con otras técnicas de regularización, como **Lasso** o **ElasticNet**. Estos métodos tienen diferentes mecanismos de regularización que te ayudarán a comprender mejor la relación entre el valor de α y el rendimiento del modelo.

    α
- **Experiencia previa y comprensión del problema**:
  - En algunos casos, puedes elegir un valor de α basándote en el conocimiento del dominio o en pruebas anteriores que hayan demostrado buenos resultados en problemas similares.

    α

Regularización

La regularización es una técnica fundamental en el aprendizaje automático que busca prevenir el sobreajuste (overfitting) del modelo. Su principio básico es:

- **Penalización de pesos grandes:** Añade un término a la función de costo que penaliza los coeficientes (pesos) de gran magnitud.

La regularización Ridge agrega un término de penalización a la función de error original, que se puede expresar matemáticamente como:

$$E_{total} = E_{original} + \lambda \sum_{i=1}^{n} w_i^2$$

Donde:

- E_{total} es el error total después de la regularización
- E_{original} es el error original del modelo
- λ (lambda) es el parámetro que controla la fuerza de la regularización
- w_i son los pesos del modelo (excluyendo el sesgo w_0)

Esta penalización cuadrática (L2) tiene el efecto de "encoger" los pesos hacia cero sin llegar a hacerlos exactamente cero, lo que ayuda a prevenir el sobreajuste mientras mantiene la contribución de todas las características.

- **Preferencia por soluciones simples:** Favorece modelos con coeficientes más pequeños, lo que generalmente resulta en soluciones más estables y generalizables.

Una ventaja al usar la sumatoria de los pesos elevados al cuadrado es que esa función sí es derivable, lo que se requiere para hacer la minimización del error total E. Al agregar la penalización, la única forma de minimizar el error total E es colocar pesos pequeños, idealmente menores a 1, porque de esa forma al elevarlos al cuadrado se hacen más pequeños.

Matemáticamente, esto se puede expresar de la siguiente manera:

$$w_i^2 < w_i \text{ cuando } |w_i| < 1$$

Por ejemplo:

- Si w = 0.5, entonces w^2 = 0.25 (menor que 0.5)
- Si w = 0.1, entonces w^2 = 0.01 (mucho menor que 0.1)

Por lo tanto, la función de error total:

$$E_{total} = E_{original} + \lambda \sum_{i=1}^{n} w_i^2$$

Se minimiza cuando los pesos w_i son pequeños, ya que sus cuadrados serán aún más pequeños, reduciendo así el término de penalización.

- **Balance complejidad-precisión:** Busca un equilibrio entre el ajuste a los datos de entrenamiento y la simplicidad del modelo.

Los beneficios principales de la regularización incluyen:

- Reducción del sobreajuste
- Mejor generalización a nuevos datos
- Mayor estabilidad del modelo

La regularización puede implementarse de diferentes formas, siendo las más comunes la regularización L1 (Lasso), L2 (Ridge) y la combinación de ambas (Elastic Net).