Aprendizaje Supervisado

Parte II: Regresión

Clase 06

Breve Introducción a Machine Learning

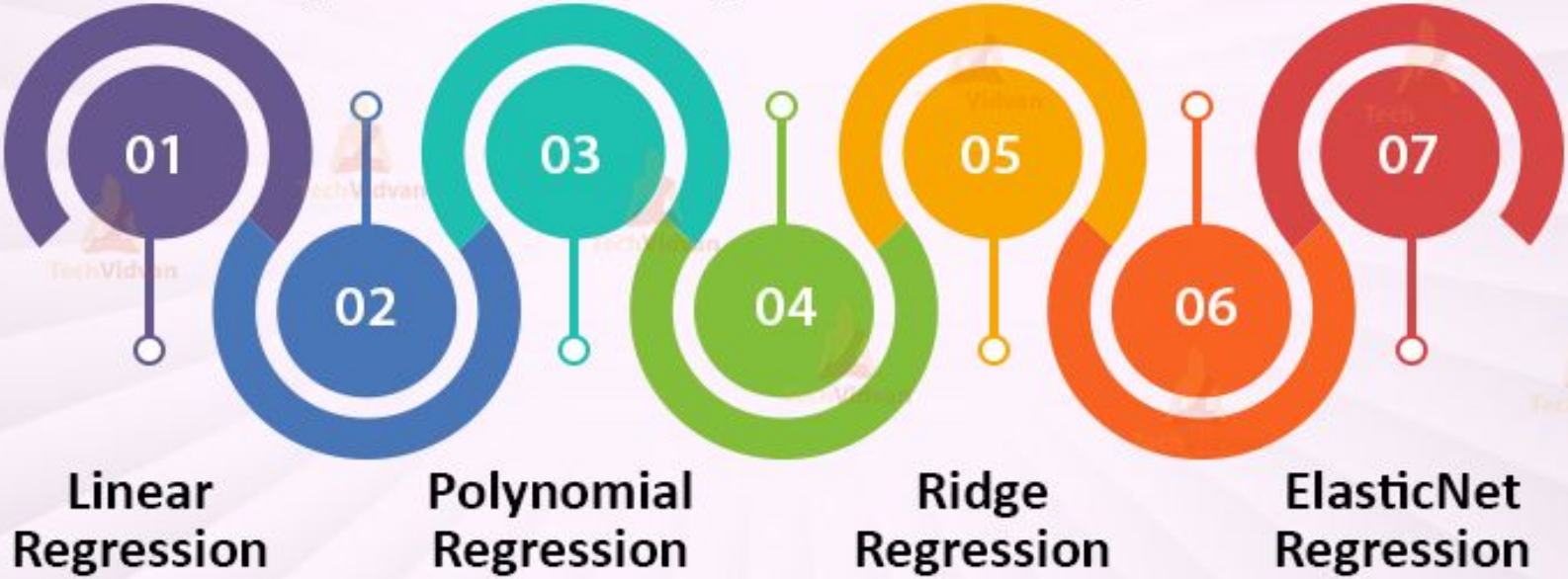
Dr. Ramón Caraballo
SECIU Red Académica Uruguaya
UDELAR

Objetivo

Vemos:

- **Reducción de Dimensionalidad: PCA**
- **Regresión: Lineal, Ridge, Lasso & Elastic-Net**
- **Características de cada uno**
- **Ajuste de Hiperparámetros: Grid Search**

Types of Regression

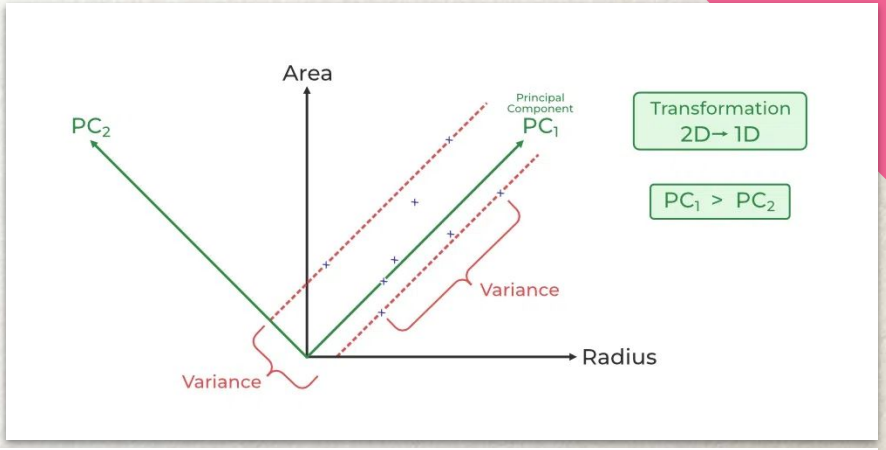


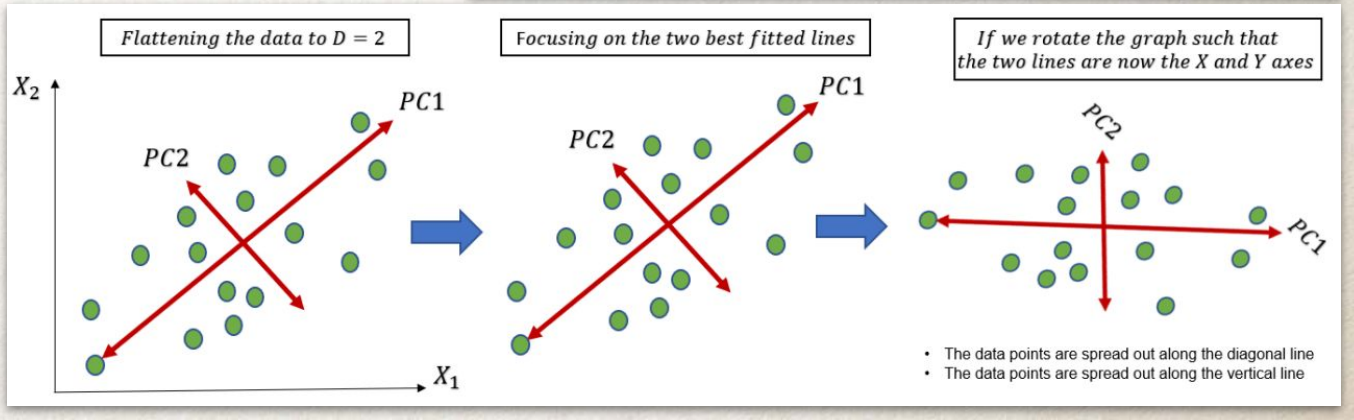
Reducción de Dimensionalidad

Principal Component Analysis (PCA)

NO todas las variables independientes contribuyen de igual forma a la varianza general de un modelo.
Solo algunas o combinaciones de algunas pueden explicar la variabilidad total de la variable objetivo.

El Análisis de Componentes Principales (PCA) es una técnica estadística utilizada para reducir la dimensionalidad de un conjunto de datos, manteniendo la mayor cantidad posible de información relevante





Reducción de dimensionalidad:

- Cuando se trabaja con datos que tienen muchas variables, puede ser difícil analizarlos y visualizarlos. El PCA transforma estas variables en un nuevo conjunto de variables, llamadas componentes principales, que son combinaciones lineales de las variables originales.
- Estos componentes principales se ordenan por la cantidad de varianza que explican, lo que permite seleccionar los primeros componentes que capturan la mayor parte de la información.

Visualización de datos:

- El PCA puede utilizarse para visualizar datos de alta dimensión en dos o tres dimensiones, lo que facilita la identificación de patrones y relaciones.

Eliminación de redundancia:

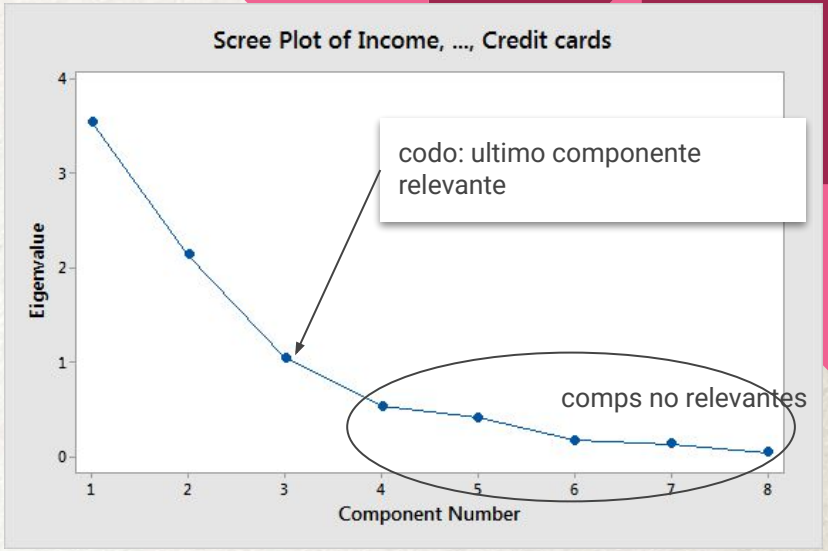
- En muchos conjuntos de datos, algunas variables están correlacionadas entre sí, lo que significa que contienen información redundante. El PCA puede eliminar esta redundancia al crear componentes principales que son independientes entre sí.

Mejora del rendimiento del modelo:

- Al reducir la dimensionalidad de los datos, el PCA puede mejorar el rendimiento de los modelos de aprendizaje automático, ya que reduce el riesgo de sobreajuste y acelera el tiempo de entrenamiento.

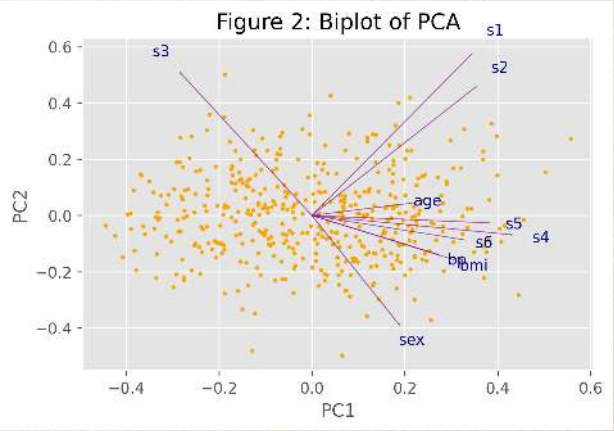
PCA: Scree Plot

¿Cuántos Componentes considerar?



¿Cómo funciona?:

- El PCA busca las direcciones en las que los datos varían más. Estas direcciones se llaman componentes principales.
- El primer componente principal explica la mayor cantidad de varianza en los datos, el segundo componente principal explica la segunda mayor cantidad de varianza, y así sucesivamente.
- Al seleccionar los primeros componentes principales, se puede reducir la dimensionalidad de los datos, manteniendo la mayor parte de la información relevante.



PCA: Ejemplo

```python
# import all libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

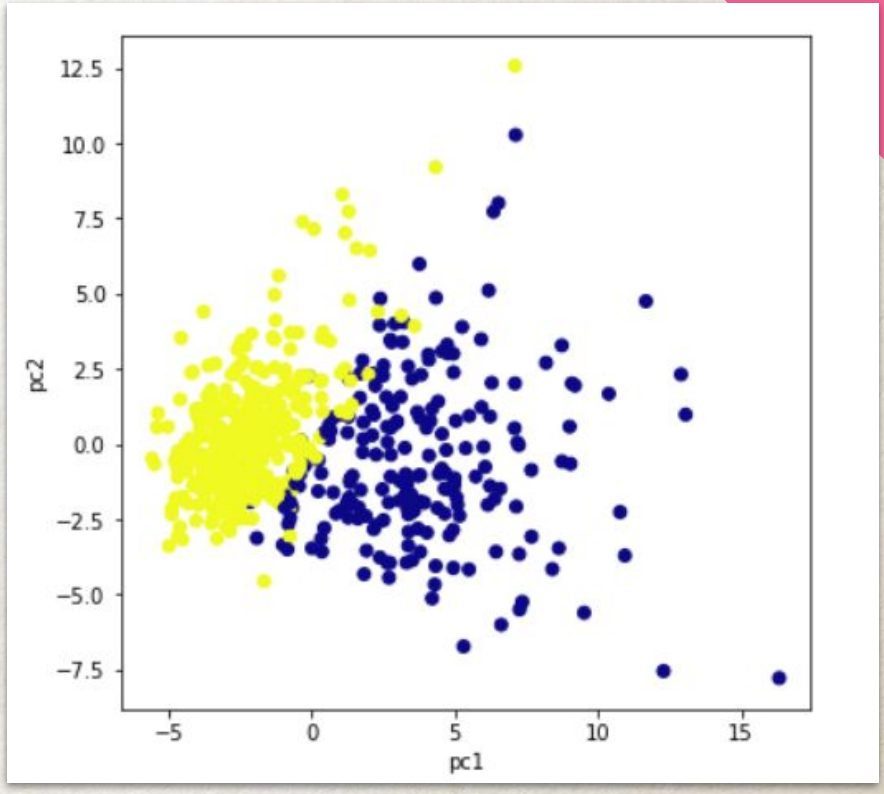
```python
#import the breast _cancer dataset
from sklearn.datasets import load_breast_cancer
data=load_breast_cancer()
data.keys()

# Check the output classes
print(data['target_names'])

# Check the input attributes
print(data['feature_names'])
```

```python
# construct a dataframe using pandas
df1=pd.DataFrame(data['data'],columns=data['feature_names'])

# Scale data before applying PCA
scaling=StandardScaler()

# Use fit and transform method
scaling.fit(df1)
Scaled_data=scaling.transform(df1)

# Set the n_components=3
principal=PCA(n_components=3)
principal.fit(Scaled_data)
x=principal.transform(Scaled_data)

# Check the dimensions of data after PCA
print(x.shape)
```

```python
# Check the values of eigen vectors
# prodeced by principal components
principal.components_
```

PCA

```
plt.figure(figsize=(10,10))
plt.scatter(x[:,0],x[:,1],c=data['target'],cmap='plasma')
plt.xlabel('pc1')
plt.ylabel('pc2')
```



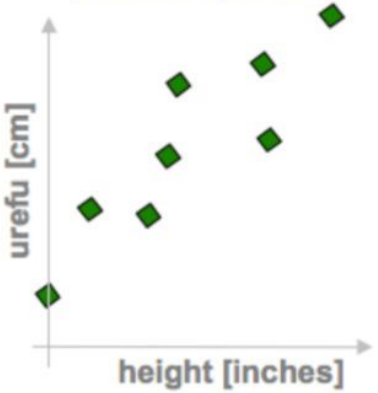

```
# check how much variance is explained by each principal component
print(principal.explained_variance_ratio_)
```

Output:

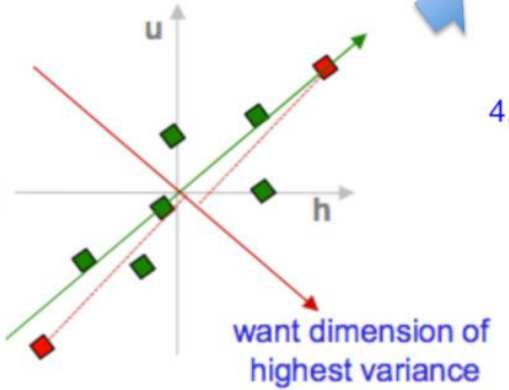
```
array([0.44272026, 0.18971182, 0.09393163])
```

PCA in a nutshell

1. correlated hi-d data
("urefu" means "height" in Swahili)



2. center the points



3. compute covariance matrix

$$\begin{array}{c} \\ h \\ u \end{array}\begin{array}{c} \begin{array}{cc} h & u \end{array} \\ \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

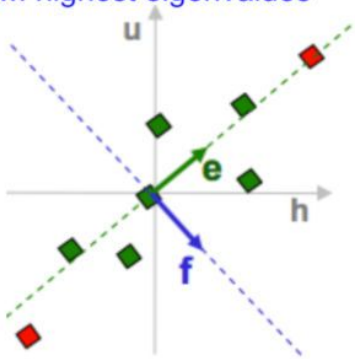
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

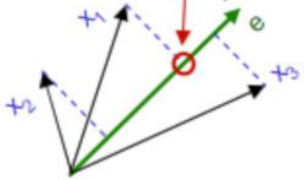
```
eig(cov(data))
```

5. pick m<d eigenvectors
w. highest eigenvalues

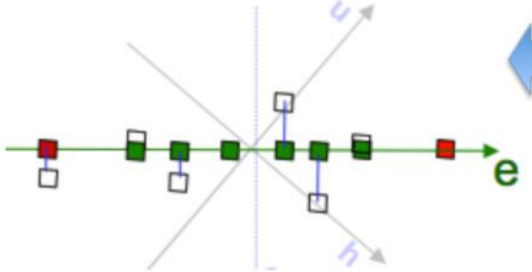


6. project data points to those eigenvectors

$$x_e' = x^T e = \sum_{j=1}^{a} x_{ij} e_j$$



7. uncorrelated low-d data



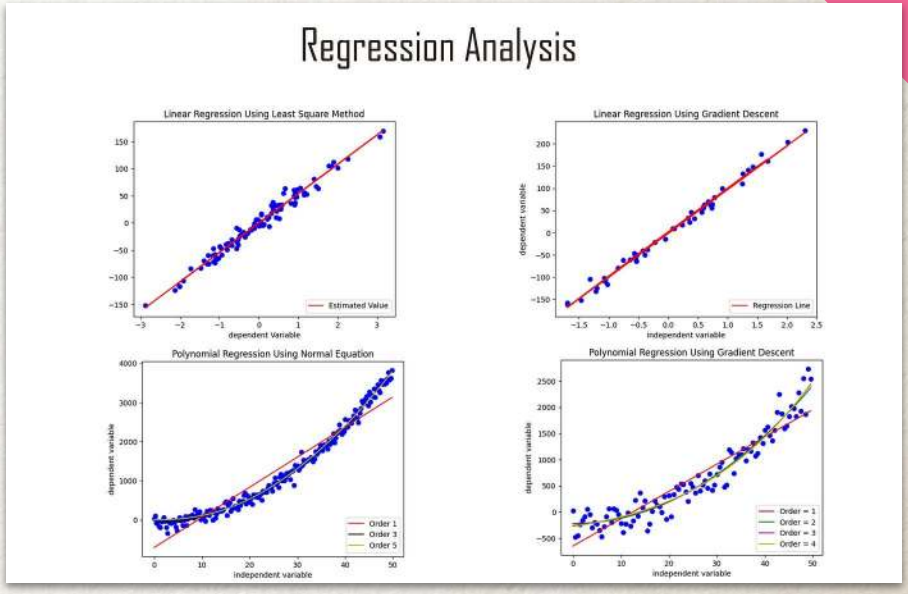
Técnicas de Regresión

Características

- Target variable continua
- Predicción de valores continuos (Tendencias)
- Muchos métodos de evaluación
- Ambiente controlado

Ejemplos

- Estimación de costos, precios, etc
- Estimaciones en Bolsa
- Análisis de satisfacción
- Evolución de ventas, etc.



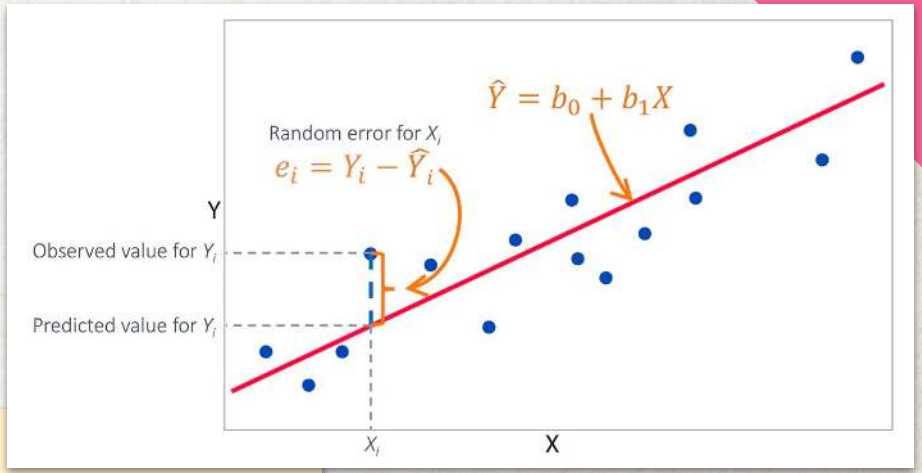
Regresión lineal:

Mínimos Cuadrados Ordinarios (MCO)

Modelo: $\hat{Y} = b_0 + b_1 X$

b_0 *Intercept o corte*

b_1 *Pendiente*



Concepto fundamental:

- Minimización de residuos: La idea fundamental es minimizar la suma de los cuadrados de los residuos $\boldsymbol{\varepsilon}$.
- Un residuo es la diferencia entre un valor observado y el valor predicho por un modelo.
- Al minimizar estas diferencias al cuadrado, buscamos el modelo que mejor represente los datos.

$$\varepsilon = \sum_i \left(y_i - \hat{y}_i \right)^2$$

y_i *Valor observado*

$\hat{y}_i$ *Valor predicho*

Regresión lineal: Múltiples Variables

Linear Regression: Single Variable

$$\hat{y} = \beta_0 + \beta_1 x + \epsilon$$

Predicted output — Coefficients — Input — Error

Linear Regression: Multiple Variables

$$\hat{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$$

Coeficiente de Determinación: R^2

El coeficiente de determinación (R^2): es la proporción de la variación en la variable dependiente que es predecible a partir de la(s) variable(s) independiente(s).

En el caso de regresión en múltiples variables se cumple que $R^2 = \rho^2$, el coeficiente de correlación de Pearson entre los valores observados y predichos.

$$R^2 = \rho^2(y, \hat{y}) = \frac{cov(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}}$$

En el caso de regresión simple en una dimensión: $\hat{Y} = b_0 + b_1 X$

$$R^2 = \rho^2(x, y)$$

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

$\bar{y} = media\ de\ los\ valores$

Ejemplo: Boston Housing Dataset

```
data_url = "http://lib.stat.cmu.edu/datasets/boston"
    raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
    data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
    target = raw_df.values[1::2, 2]
```

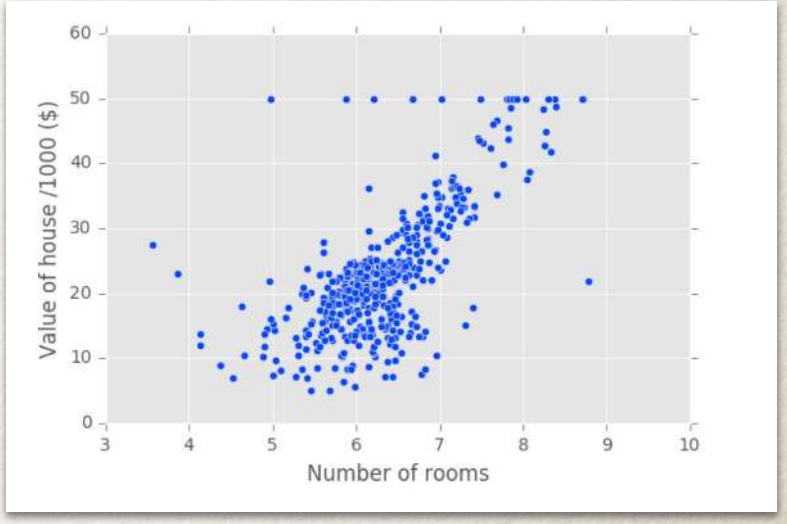
```
In [1]: boston = pd.read_csv('boston.csv')

In [2]: print(boston.head())
      CRIM    ZN  INDUS  CHAS    NX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

    PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

Regresión en una dimensión

```
In [3]: X = boston.drop('MEDV', axis=1).values

In [4]: y = boston['MEDV'].values
```

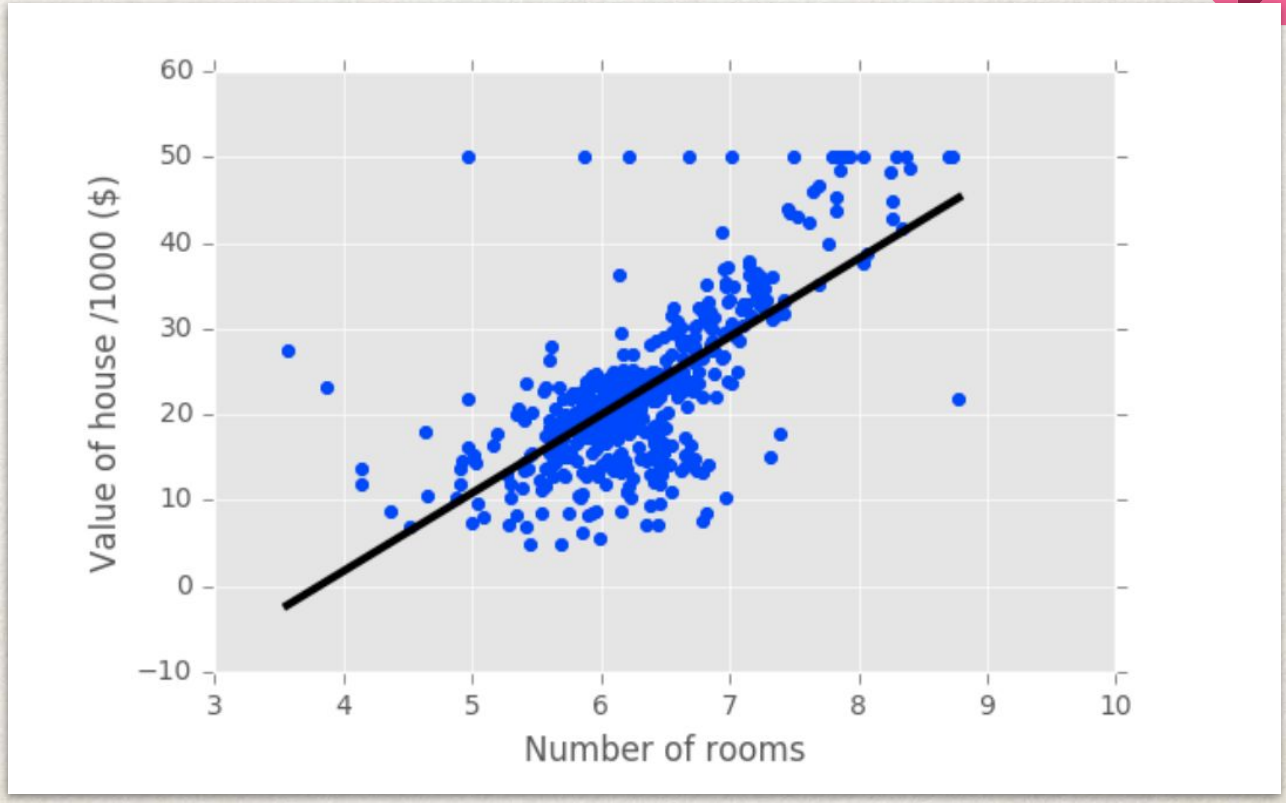
```
In [5]: X_rooms = X[:,5]

In [6]: type(X_rooms), type(y)
Out[6]: (numpy.ndarray, numpy.ndarray)

In [7]: y = y.reshape(-1, 1)

In [8]: X_rooms = X_rooms.reshape(-1, 1)
```

```
In [9]: plt.scatter(X_rooms, y)

In [10]: plt.ylabel('Value of house /1000 ($)')

In [11]: plt.xlabel('Number of rooms')

In [12]: plt.show();
```



Ajustando un Modelo de Regresión

```
In [13]: import numpy as np

In [14]: from sklearn import linear_model

In [15]: reg = linear_model.LinearRegression()

In [16]: reg.fit(X_rooms, y)

In [17]: prediction_space = np.linspace(min(X_rooms),
   ...:                                 max(X_rooms)).reshape(-1, 1)

In [18]: plt.scatter(X_rooms, y, color='blue')

In [19]: plt.plot(prediction_space, reg.predict(prediction_space),
   ...:           color='black', linewidth=3)

In [20]: plt.show()
```


Value of house /1000 ($)
Number of rooms
60
50
40
30
20
10
0
−10
3
4
5
6
7
8
9
10

Regresión Multivariada

Todas las características a la vez

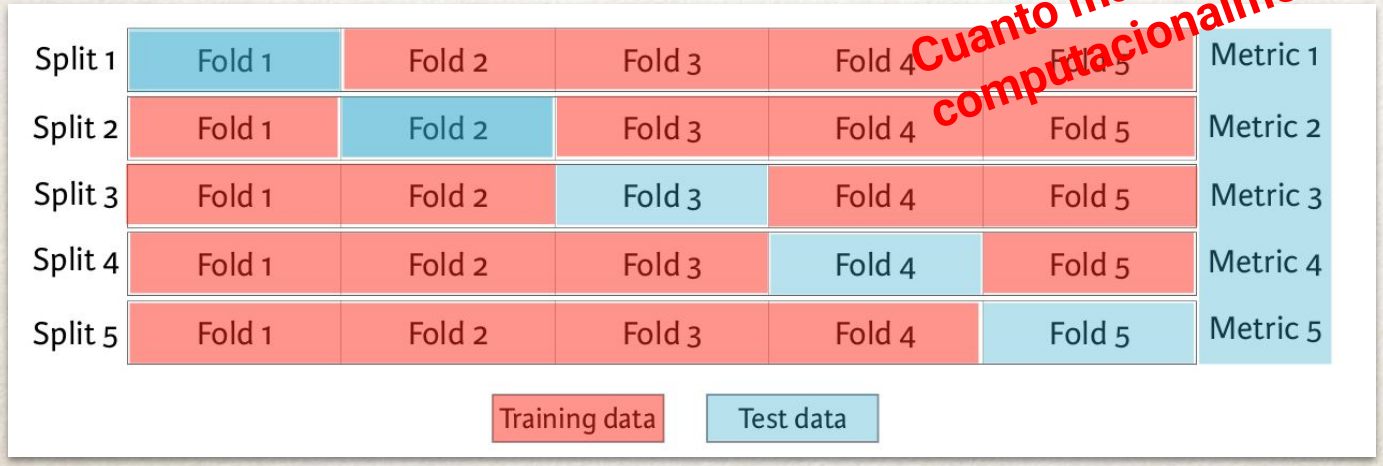
```
In [1]: from sklearn.model_selection import train_test_split

In [2]: X_train, X_test, y_train, y_test = train_test_split(X, y,
   ...: test_size = 0.3, random_state=42)

In [3]: reg_all = linear_model.LinearRegression()

In [4]: reg_all.fit(X_train, y_train)

In [5]: y_pred = reg_all.predict(X_test)

In [6]: reg_all.score(X_test, y_test)
Out[6]: 0.711226005748495 26
```

El Rol de la Validación Cruzada

(Cross-Validation)

- A menudo el rendimiento del modelo depende de cómo se dividen los datos.
- El tener un R^2 bueno en una única instancia NO es representativo de la capacidad del modelo para generalizar.
- Solución: ¡Validación cruzada!

Cuanto mas plegamientos k, más pesado computacionalmente !!!

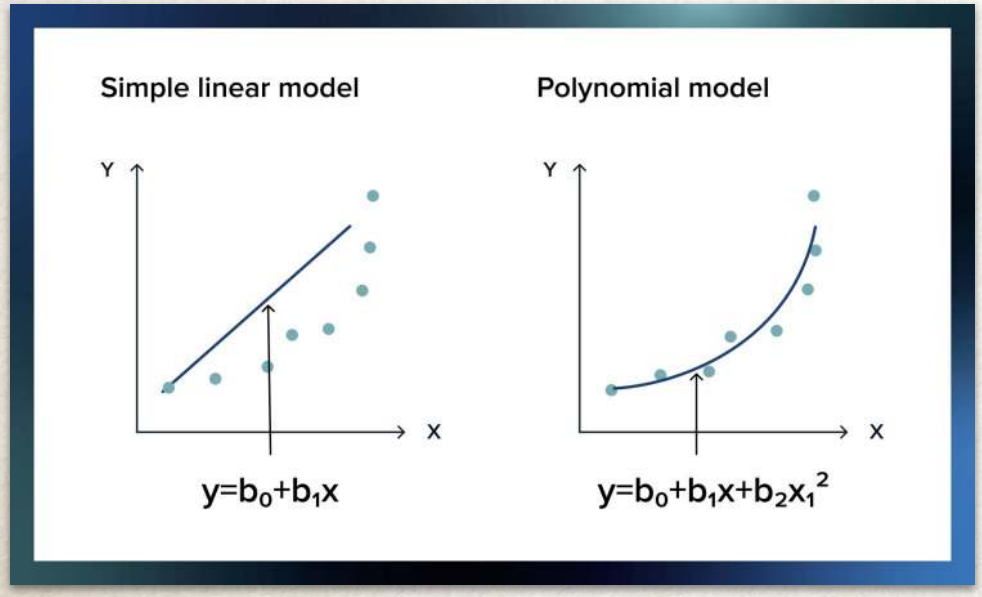


Cross-Validation en sklearn

```
In [1]: from sklearn.model_selection import cross_val_score

In [2]: reg = linear_model.LinearRegression()

In [3]: cv_results = cross_val_score(reg, X, y, cv=5)

In [4]: print(cv_results)
[ 0.63919994  0.71386698  0.58702344  0.07923081 -0.25294154]

In [5]: np.mean(cv_results)
Out[5]: 0.3532759243958705 8
```

Regresión Polinómica (RP)

La regresión polinómica es necesaria cuando no existe una correlación lineal que ajuste todas las variables.



Maths

Polynomial Regression Equation

$$y_{pred} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n$$

Where :

$\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ – are the model's parameters

y_{pred} – is the prediction of a target

x – is the feature value

n – is the Polynomial Regression's degree

Regresión Polinómica

Pros & Cons

Pros

- Se pueden modelar relaciones no lineales entre variables.
- Existe una amplia gama de funciones que puedes usar para el ajuste.
- **Ideal para fines de exploración:** i.e.,comprobar la presencia de curvatura y sus inflexiones.
- En resumen, es una herramienta flexible que permite ajustar una gran variedad de distribuciones de datos.

Cons

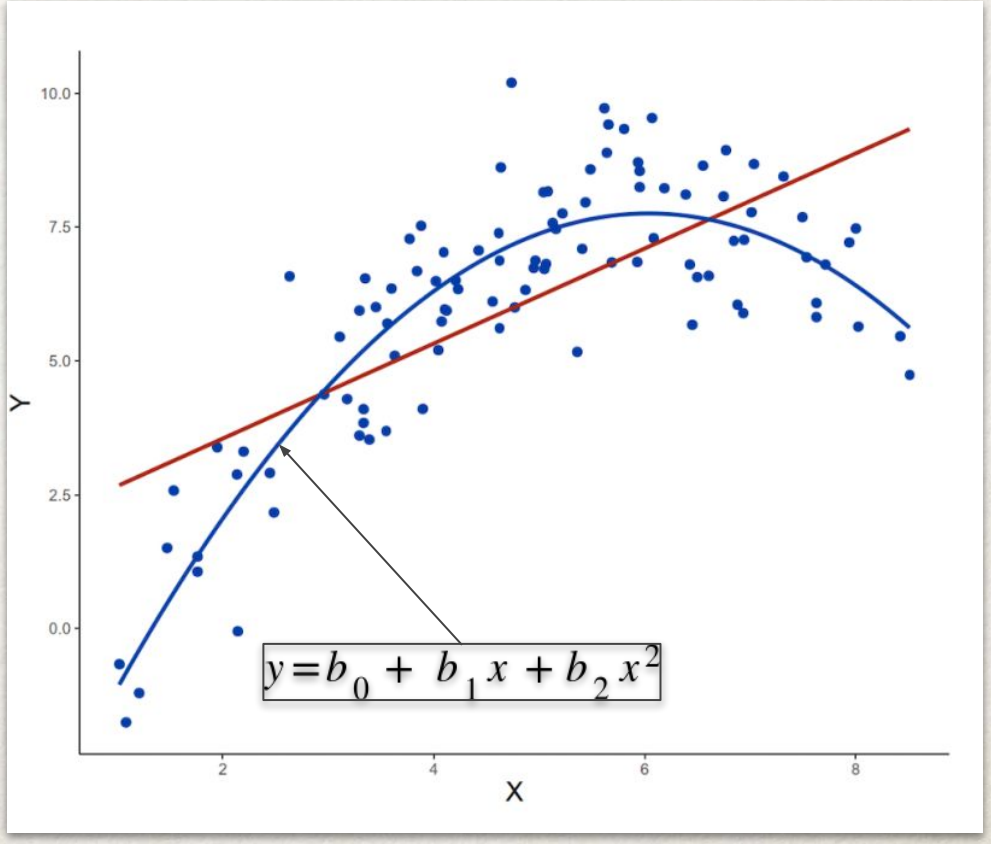
- Incluso un solo valor atípico puede alterar gravemente los resultados.
- Los modelos RP son propensos al sobreajuste. Si se utilizan suficientes parámetros, se puede ajustar cualquier parámetro.
- Como dijo John von Neumann: «Con cuatro parámetros puedo ajustar un elefante, con cinco puedo hacer que mueva la trompa».
- Como consecuencia de lo anterior, los modelos RP podrían no ser generalizables adecuadamente fuera de los datos utilizados.

Regresión Polinómica

```
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)

poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)
```

```
# Visualising the Polynomial Regression results
plt.scatter(X, y, color = 'blue')

plt.plot(X, lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')

plt.show()
```

Regresión Polinómica



Regularización

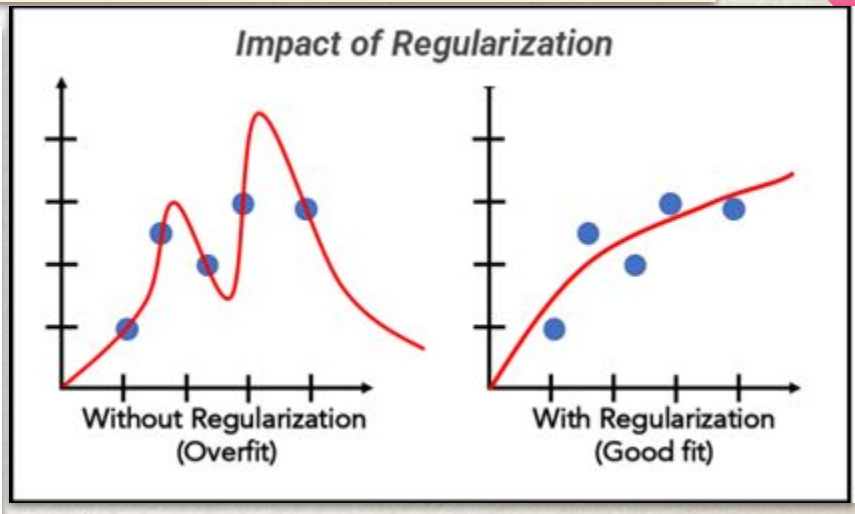
La regularización se utiliza para evitar el sobreajuste (overfitting), que ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento y su rendimiento es deficiente con datos nuevos e inéditos.

El problema del sobreajuste:

En la regresión estándar, el objetivo es minimizar la diferencia entre los valores predichos y los reales. Sin embargo, los modelos complejos pueden "memorizar" los datos de entrenamiento, incluyendo su ruido, lo que resulta en una generalización deficiente.

La solución: Regularización:

La regularización añade un término de penalización a la función de pérdida del modelo, lo que desalienta los modelos excesivamente complejos.
Este término de penalización limita el tamaño de los coeficientes de regresión, simplificando eficazmente el modelo.



Importancia

Por qué es Importante la Regularización:

Generalización mejorada: Mejora el rendimiento de los modelos con datos no analizados.

Reducción del sobreajuste: Evita que los modelos se vuelvan demasiado complejos y generen ruido de ajuste.

Estabilidad mejorada del modelo: Hace que los modelos sean menos sensibles a pequeños cambios en los datos de entrenamiento.

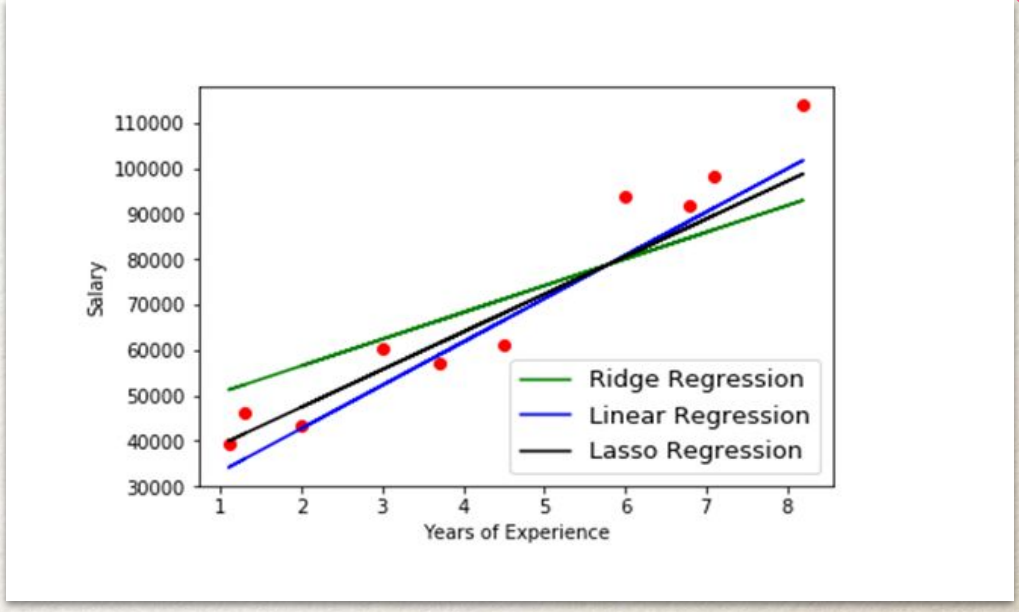
Selección de características: Lasso y Elastic net pueden utilizarse para reducir el número de características utilizadas en un modelo.

En esencia, la regularización es una técnica crucial para construir modelos de regresión robustos y fiables que se generalicen correctamente a nuevos datos.

Técnicas de Regularización

Con un n^{ro} grande de puntos, mínimos cuadrados es muy confiable, pero con pocos datos la varianza es alta haciendo que el sobreajuste sea considerable.

- Ridge Regression
- Lasso Regression
- Elastic-Net Regression



Ridge Regression

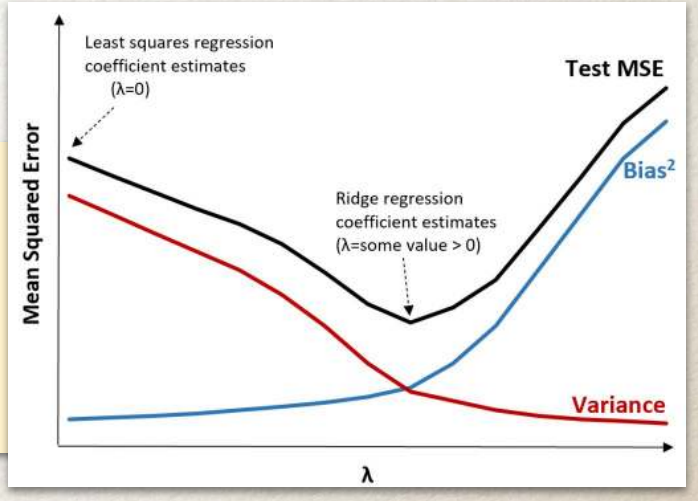
Ridge Regression (Regularización L2):

- Añade una penalización proporcional al cuadrado de los coeficientes de regresión.
- Reduce los coeficientes hacia cero, pero rara vez los establece exactamente en cero.
- Eficaz para reducir el impacto de la multicolinealidad (alta correlación entre las variables predictoras).

$$RSS(\alpha) = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \alpha \underbrace{\sum_{j=1}^{p} b_j^2}_{\text{penalty}}$$

$$con \qquad \hat{y}_i = b_0 + \sum_{i}^{p} b_j x_{ij}$$

- ***α***: Parámetro que debemos elegir
- Seleccionar ***α*** aquí es similar a seleccionar *k* en KNN
- Ajuste de hiperparámetros
- ***α*** controla la complejidad del modelo
- ***α*** = 0: Obtenemos MCO (puede provocar sobreajuste)
- ***α*** muy alto: puede provocar subajuste



Ridge Regression en sklearn

```
In [1]: from sklearn.linear_model import Ridge

In [2]: X_train, X_test, y_train, y_test = train_test_split(X, y,
   ...: test_size = 0.3, random_state=42)

In [3]: ridge = Ridge(alpha=0.1, normalize=True)

In [4]: ridge.fit(X_train, y_train)

In [5]: ridge_pred = ridge.predict(X_test)

In [6]: ridge.score(X_test, y_test)
Out[6]: 0.699693827512731 79
```

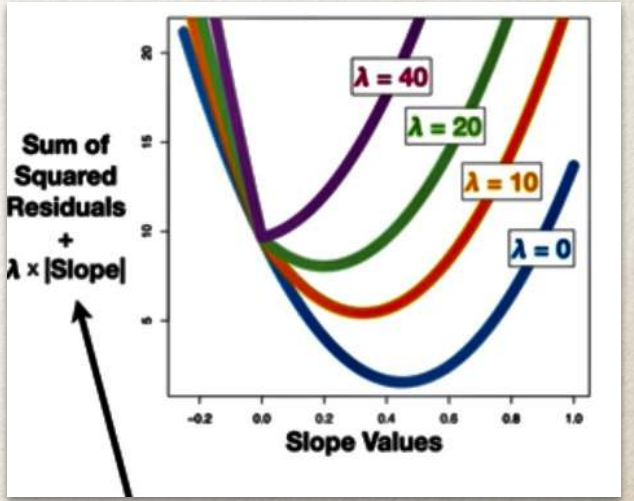
Lasso Regression

$$LSS(\alpha) = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \alpha \sum_{j=1}^{p} |b_j|$$

Lasso Regression (Regularización L1):

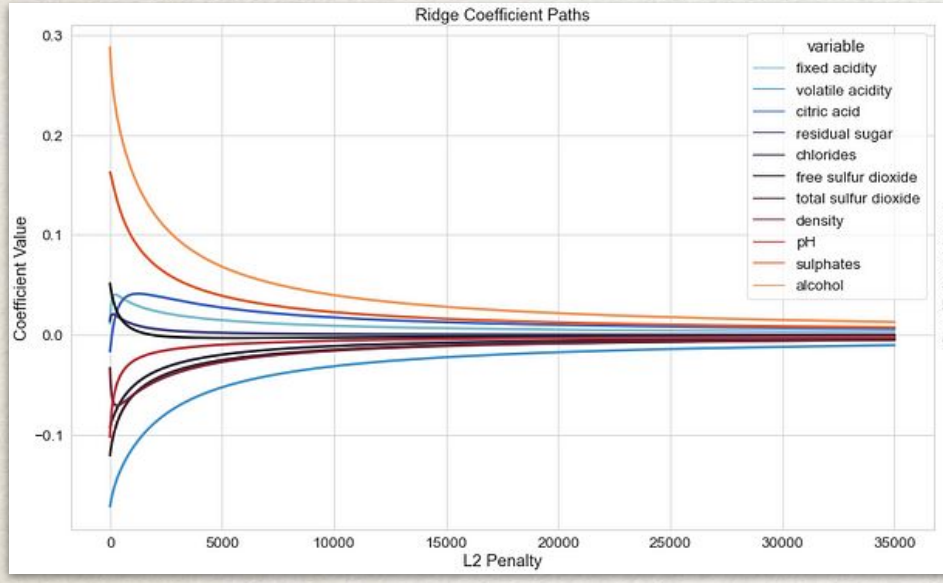
- Añade una penalización proporcional al valor absoluto de los coeficientes.
- Puede establecer algunos coeficientes exactamente a cero, lo que permite una selección de características eficaz.
- Es útil al trabajar con conjuntos de datos con muchas características, ya que permite identificar las más importantes.
- Anula los coeficientes de las características menos importantes

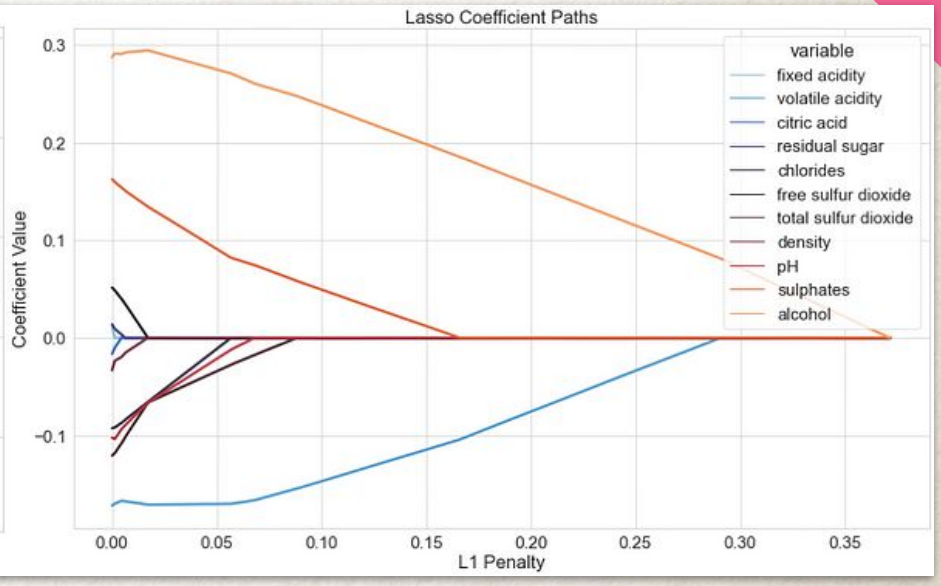
$$con \qquad \hat{y}_i = b_0 + \sum_{j}^{p} b_j x_{ij}$$



Ridge vs Lasso Coefficients

En función del valor de penalty



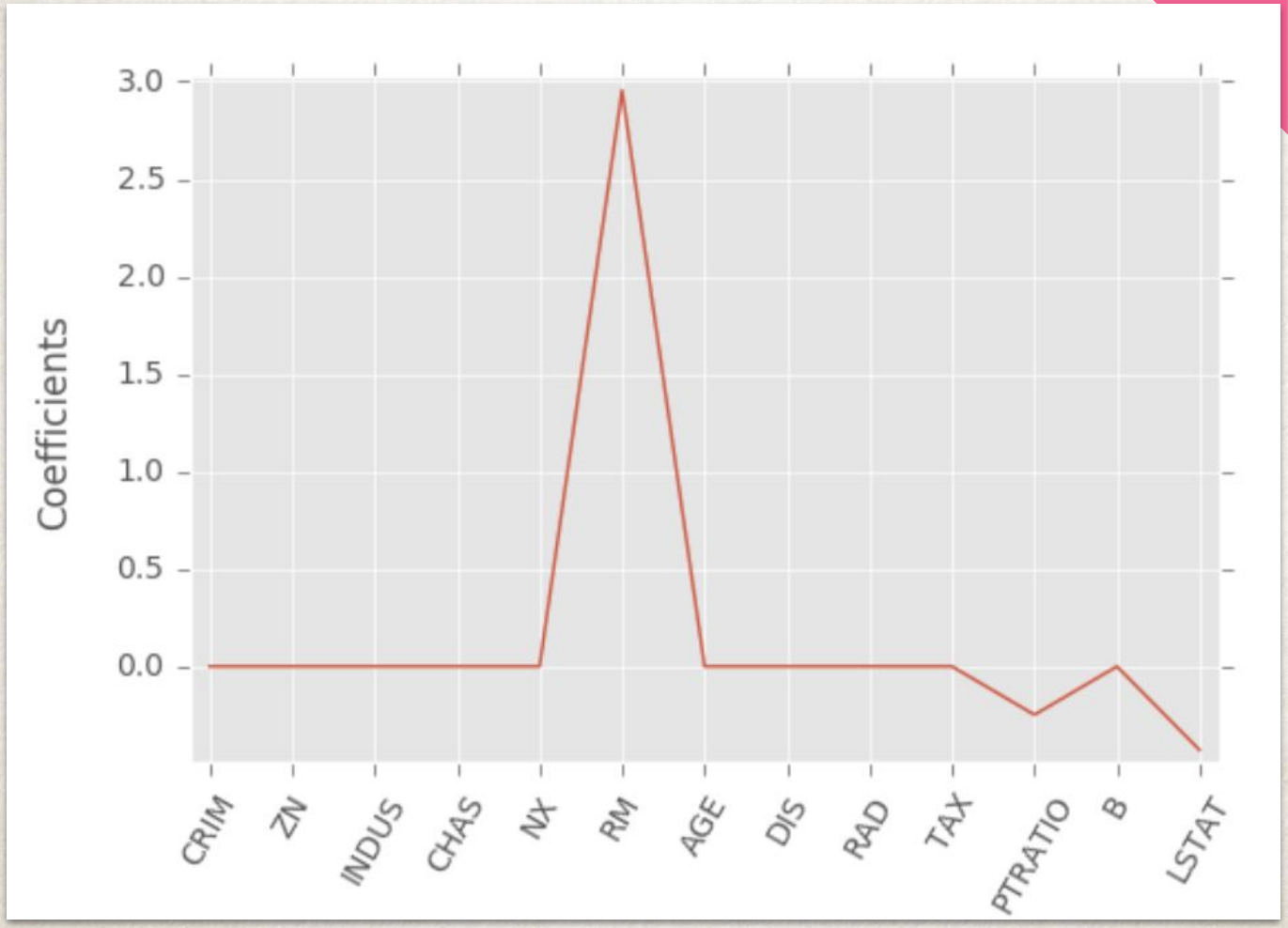


Lasso Regression en sklearn

```
In [1]: from sklearn.linear_model import Lasso

In [2]: X_train, X_test, y_train, y_test = train_test_split(X, y,
   ...: test_size = 0.3, random_state=42)

In [3]: lasso = Lasso(alpha=0.1, normalize=True)

In [4]: lasso.fit(X_train, y_train)

In [5]: lasso_pred = lasso.predict(X_test)

In [6]: lasso.score(X_test, y_test)
Out[6]: 0.59502295353285506
```

Lasso: selección de características en sklearn

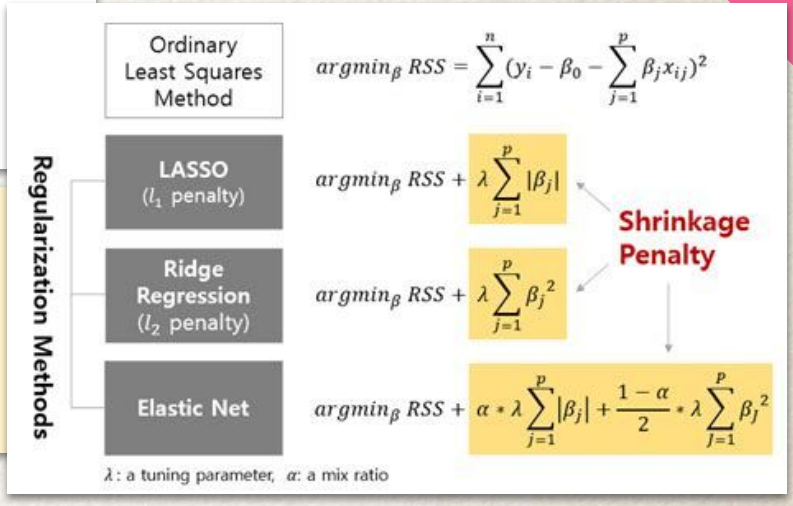
```
In [1]: from sklearn.linear_model import Lasso

In [2]: names = boston.drop('MEDV', axis=1).columns

In [3]: lasso = Lasso(alpha=0.1)

In [4]: lasso_coef = lasso.fit(X, y).coef_

In [5]: _ = plt.plot(range(len(names)), lasso_coef)

In [6]: _ = plt.xticks(range(len(names)), names, rotation=60)

In [7]: _ = plt.ylabel('Coefficients')

In [8]: plt.show()
```


Coefficients
3.0
2.5
2.0
1.5
1.0
0.5
0.0
CRIM
ZN
INDUS
CHAS
NX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT

Elastic-Net Regression

- Combina la regularización L1 y L2
- Proporciona un equilibrio entre la selección de características y la reducción de coeficientes.
- Resulta útil al trabajar con conjuntos de datos con alta multicolinealidad y numerosas características.

Combina las fortalezas de ambos enfoques, mitigando sus limitaciones individuales.
Enfoque híbrido: mayor flexibilidad y adaptabilidad, lo que permite adaptar la estrategia de regularización a las características específicas del conjunto de datos.



$$ENSS(\alpha,\beta) = \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 + \alpha \sum_{j=1}^{p}\left|b_j\right| + \beta \sum_{j=1}^{p} b_j^2$$

Características

- Elastic Net, con penalizaciones L1 y L2, gestiona eficazmente la multicolinealidad al promover la dispersión de coeficientes (como Lasso) y estabilizar las estimaciones de coeficientes (como Ridge).

- Elastic Net puede realizar la selección automática de características al llevar algunos coeficientes a cero (como Lasso), lo que facilita la interpretabilidad del modelo y reduce el sobreajuste.

- El parámetro de mezcla de Elastic Net permite a los profesionales controlar el equilibrio entre las penalizaciones de Lasso y Ridge, lo que proporciona flexibilidad en la regularización según los requisitos específicos del problema.

- Elastic Net es más robusto en conjuntos de datos de alta dimensión, especialmente cuando el número de características es mayor que el número de observaciones, lo que lo hace adecuado para escenarios con datos amplios.

Elastic-Net Regression

$$\text{Elastic Net Loss} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\left(\alpha\sum_{j=1}^{p}|\beta_j| + (1-\alpha)\sum_{j=1}^{p}\beta_j^2\right)$$

Here:

1. $\frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$: Mean squared error (MSE), the regression loss term measuring the difference between predicted and actual values.
2. λ: Regularization parameter, controlling the overall strength of the regularization.
3. α: Mixing parameter, determining the balance between L1 (Lasso) and L2 (Ridge) penalties.
4. $\sum_{j=1}^{p}|\beta_j|$: L1 penalty, inducing sparsity.
5. $\sum_{j=1}^{p}\beta_j^2$: L2 penalty, encouraging small coefficients.

Elastic-Net Regression

```
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
```

```
# Load your dataset
X, y = load_your_dataset()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create an Elastic Net model
model = ElasticNet(alpha=0.1, l1_ratio=0.5)  # Adjust alpha and
l1_ratio as needed

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Parámetros de ElasticNet()

alpha (α):

Definición: α es el parámetro de intensidad de regularización en Elastic Net.
Función: Controla la intensidad general de la regularización aplicada al modelo.

Valores:

α = 0: No se aplica regularización y Elastic Net equivale a la regresión de Mínimos Cuadrados Ordinarios (MCO).

α = 1: Se aplican regularizaciones L1 y L2, combinando las penalizaciones de Lasso y Ridge.

0 < α < 1: Elastic Net aplica una combinación de regularización L1 y L2, lo que permite una combinación flexible de penalizaciones.

Ajuste de Hiperparámetros

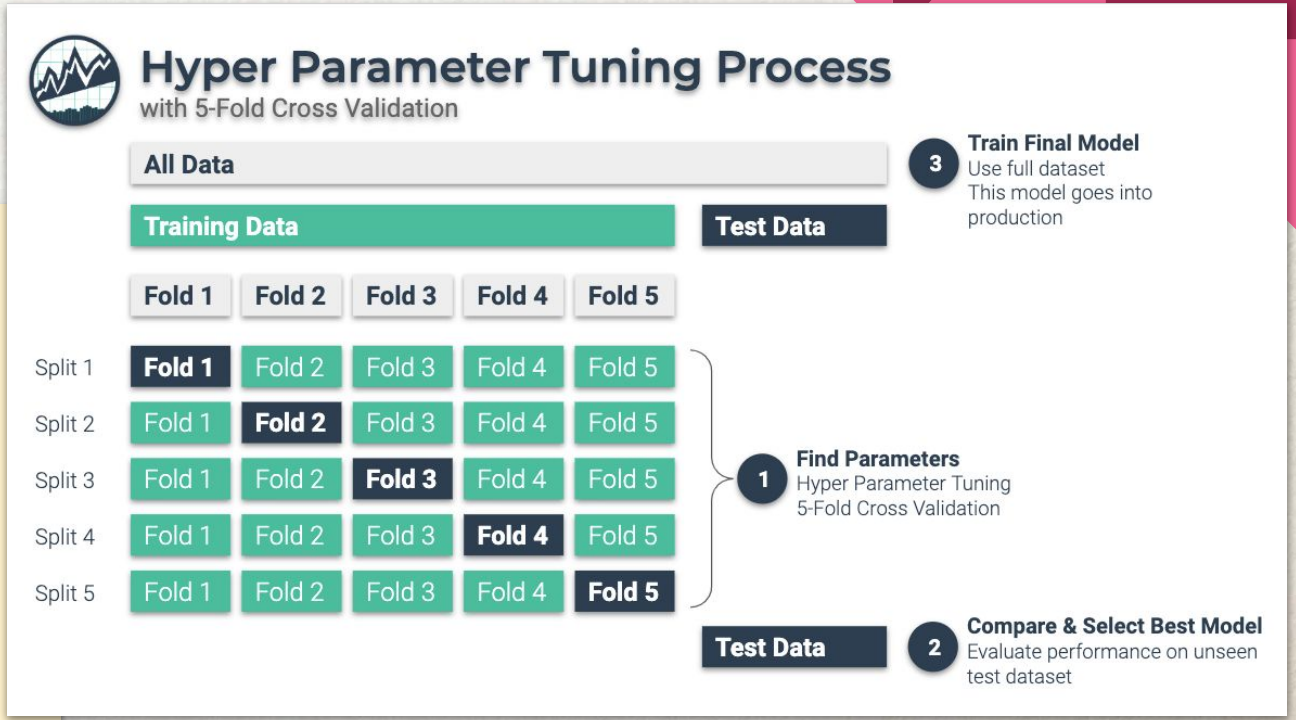
Regresión lineal:
Selección de parámetros

Regresión Ridge/Lasso:
Selección de α

KNN: Selección de k vecinos

Parámetros como alfa y k: Hiperparámetros

Los hiperparámetros no se pueden aprender mediante la corrección del modelo



Ajuste de Hiperparámetros

Elección del hiperparámetro correcto
- Pruebe con varios valores de hiperparámetro diferentes
- Ajústelos por separado
- Observe el rendimiento de cada uno
- Elija el que tenga el mejor rendimiento
- Es fundamental utilizar la validación cruzada

Grid search cross-validation

C \ Alpha	0.1	0.2	0.3	0.4
0.5	0.701	0.703	0.697	0.696
0.4	0.699	0.702	0.698	0.702
0.3	0.721	0.726	0.713	0.703
0.2	0.706	0.705	0.704	0.701
0.1	0.698	0.692	0.688	0.675

GridSearchCV in scikit-learn

```
In [1]: from sklearn.model_selection import GridSearchCV

In [2]: param_grid = {'n_neighbors': np.arange(1, 50)}

In [3]: knn = KNeighborsClassifier()

In [4]: knn_cv = GridSearchCV(knn, param_grid, cv=5)

In [5]: knn_cv.fit(X, y)

In [6]: knn_cv.best_params_
Out[6]: {'n_neighbors': 12}

In [7]: knn_cv.best_score_
Out[7]: 0.933216168717
```

Razonamiento

Razonamiento del conjunto de reserva

- ¿Qué tan bien puede funcionar el modelo con datos nunca antes vistos?
- Usar TODOS los datos para la validación cruzada no es ideal.
- Dividir los datos en el conjunto de entrenamiento y el conjunto de reserva al principio.
- Realizar una validación cruzada de búsqueda en cuadrícula en el conjunto de entrenamiento.
- Elegir los mejores hiperparámetros y evaluarlos en el conjunto de reserva.

Sumario

- Principal Component Analysis (PCA), es una técnica muy útil para reducir la dimensionalidad en conjuntos de datos con muchas características (variables).
- El análisis de regresión se usa cuando tenemos variables continuas y existen muchas técnicas de regresión.
- La regresión es muy buena con grandes conjuntos de datos, pero en caso de pocos datos el sobre ajuste puede hacer que los modelos de regresión pierdan generalidad.
- Las técnicas de regularización permiten resolver el sobreajuste y la colinealidad de las variables, adicionando diferentes términos de penalización a la función de perdida del modelo.
- Estas penalizaciones limitan el tamaño de los coeficientes de regresión, robustecen y simplifican el modelo.
- Para elegir los mejores parámetros del modelo debo realizar un grid-search en el conjunto de entrenamiento y seleccionar la combinación de estos que proporcione la mejor predicción.

Sumario