



ARISA Learning Material

Educational Profile and EQF level: DATA SCIENTIST – EQF 6

PLO: 1, 2, 3, 4, 5

Learning Unit (LU): MACHINE LEARNING: SUPERVISED

Topic: LINEAR MODELS AND LOGISTIC REGRESSION



Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

www.aiskills.eu

Copyright © 2024 by the Artificial Intelligence Skills Alliance

All learning materials (including Intellectual Property Rights) generated in the framework of the ARISA project are made freely available to the public under an open license Creative Commons Attribution–NonCommercial (CC BY-NC 4.0).

ARISA Learning Material 2024

This material is a draft version and is subject to change after review coordinated by the European Education and Culture Executive Agency (EACEA).

Authors: Universidad Internacional de La Rioja (UNIR)

Disclaimer: This learning material has been developed under the Erasmus+ project ARISA (Artificial Intelligence Skills Alliance) which aims to skill, upskill, and reskill individuals into high-demand software roles across the EU.



This project has been funded with support from the European Commission. The material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

• **About ARISA**

- The Artificial Intelligence Skills Alliance (ARISA) is a four-year transnational project funded under the EU's Erasmus+ programme. It delivers a strategic approach to sectoral cooperation on the development of Artificial Intelligence (AI) skills in Europe.
- ARISA fast-tracks the upskilling and reskilling of employees, job seekers, business leaders, and policymakers into AI-related professions to open Europe to new business opportunities.
- ARISA regroups leading ICT representative bodies, education and training providers, qualification regulatory bodies, and a broad selection of stakeholders and social partners across the industry.

[ARISA Partners & Associated Partners](#) | [LinkedIn](#) | [Twitter](#)

Modelos lineales

Un modelo lineal es una suma de variables ponderadas que predice un valor de salida objetivo dada una instancia de datos de entrada.

Ejemplo: predicción de los precios de la vivienda

- House features: taxes per year (X_{TAX}), age in years (X_{AGE})

$$\widehat{Y_{PRICE}} = 212000 + 109 X_{TAX} - 2000 X_{AGE}$$

- A house with feature values (X_{TAX} , X_{AGE}) of (10000, 75) would have a predicted selling price of:

$$\widehat{Y_{PRICE}} = 212000 + 109 \cdot \mathbf{10000} - 2000 \cdot \mathbf{75} = \mathbf{1,152,000}$$

La regresión lineal es un ejemplo de un modelo lineal

Input instance – feature vector: $\mathbf{x} = (x_0, x_1, \dots, x_n)$

Predicted output: $\hat{y} = \hat{w}_0 x_0 + \hat{w}_1 x_1 + \dots + \hat{w}_n x_n + \hat{b}$

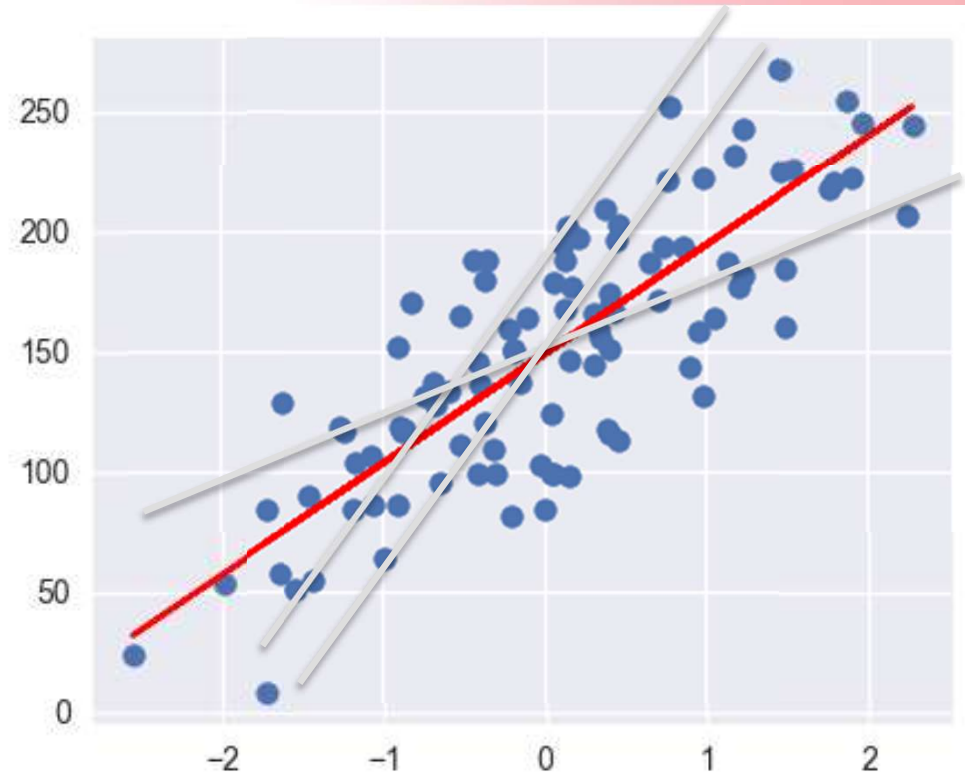
Parameters to estimate: $\left\{ \begin{array}{l} \hat{\mathbf{w}} = (\hat{w}_0, \dots, \hat{w}_n): \text{feature weights /} \\ \text{model coefficients} \\ \hat{b}: \text{constant bias term / intercept} \end{array} \right.$

Regresión lineal de una característica

Input instance: $\mathbf{x} = (x_0)$

Predicted output: $\hat{y} = \hat{w}_0 x_0 + \hat{b}$

Parameters to estimate: \hat{w}_0 (slope)
 \hat{b} (y-intercept)

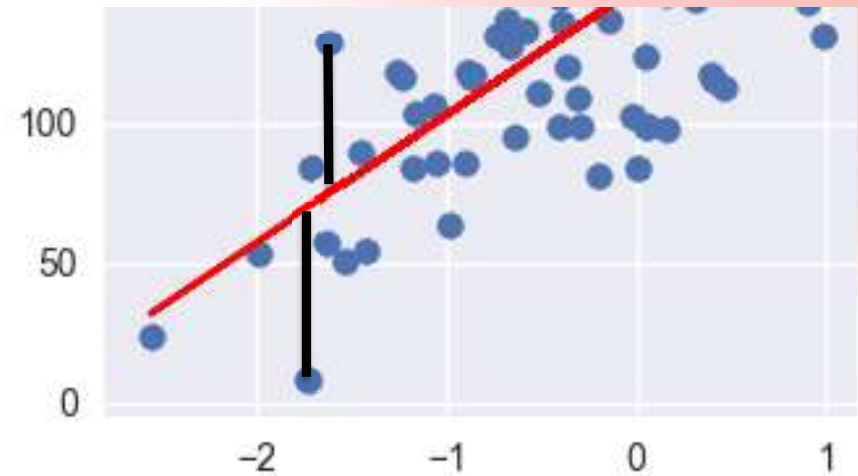


Parámetros

- Los parámetros se estiman a partir de los datos de entrenamiento.
- Hay muchas formas diferentes de estimar w y b :
 - Los diferentes métodos corresponden a diferentes criterios y objetivos de "ajuste" y formas de controlar la complejidad del modelo.
- El algoritmo de aprendizaje encuentra los parámetros que optimizan una función objetivo, normalmente para minimizar algún tipo de función de pérdida de los valores objetivo previstos frente a los valores objetivo reales.

Least-Squares Linear regression

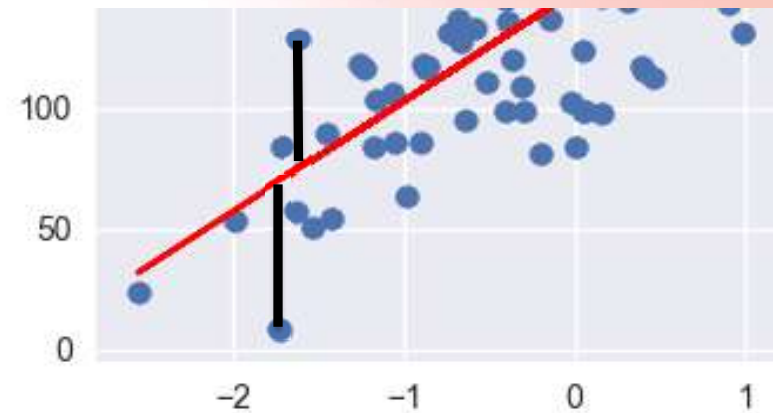
- Encuentra w y b que minimiza el error cuadrático medio del modelo lineal: la suma de las diferencias al cuadrado entre los valores objetivo predichos y los valores objetivo reales.
-
- No hay parámetros para controlar la complejidad del modelo.



Regresión lineal por mínimos cuadrados

minimiza la suma de las diferencias al cuadrado (RSS) en los datos de entrenamiento entre los valores objetivo previstos y los valores objetivo reales.

- No hay parámetros para controlar la complejidad del modelo.



$$RSS(\mathbf{w}, b) = \sum_{\{i=1\}}^N (y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2$$

Regresion lineal en Python

```
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test =
train_test_split(X_R1, y_R1, random_state = 0)

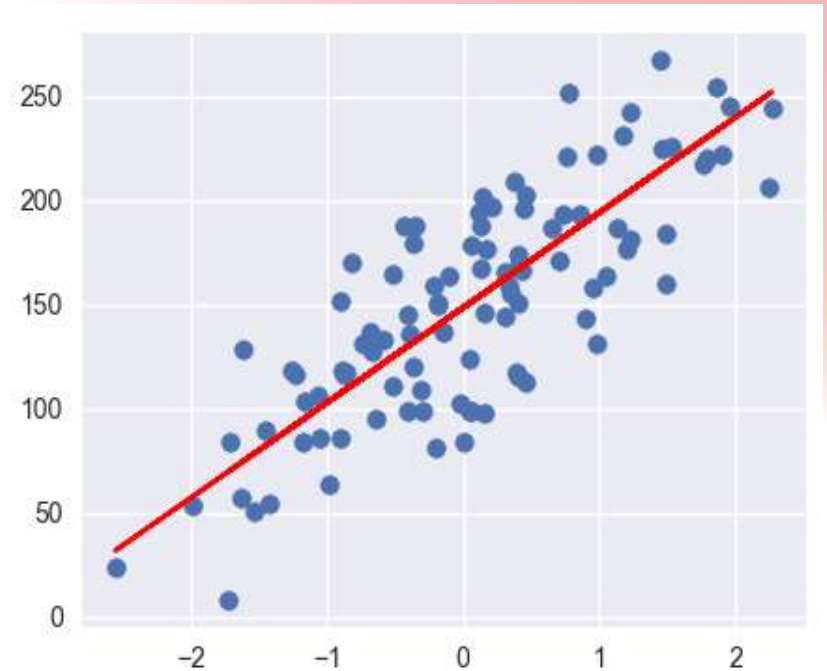
linreg = LinearRegression().fit(X_train, y_train)

print("linear model intercept (b): {}".format(linreg.intercept_))
print("linear model coeff (w): {}".format(linreg.coef_))
```

Underscore denotes a quantity derived from training data, as opposed to a user setting.

linreg.coef_
linreg.intercept_

$$\hat{y} = w_0 x_0 + b$$



Regresion Lineal en Python

```
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test =
train_test_split(X_R1, y_R1, random_state = 0)

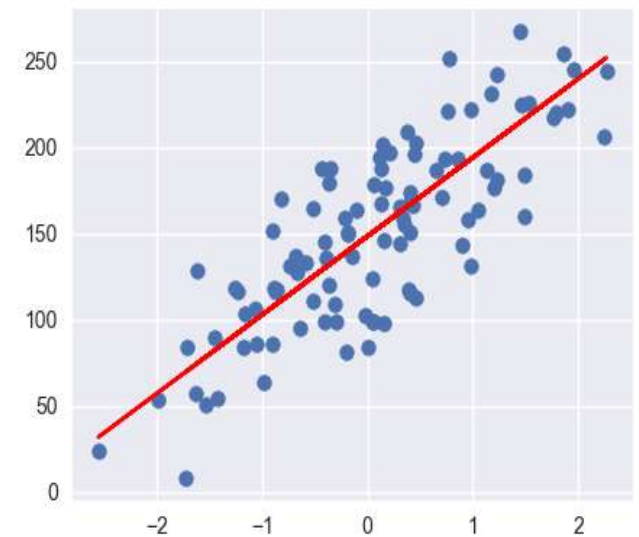
linreg = LinearRegression().fit(X_train, y_train)

print("linear model intercept (b): {}".format(linreg.intercept_))
print("linear model coeff (w): {}".format(linreg.coef_))
```

$$\hat{y} = \mathbf{w}_0 x_0 + b$$

Diagram illustrating the linear regression equation $\hat{y} = \mathbf{w}_0 x_0 + b$. Arrows point from the labels `linreg.coef_` and `linreg.intercept_` to the terms \mathbf{w}_0 and b respectively in the equation.

```
linear model coeff (w): [ 45.70870465]
linear model intercept (b): 148.44575345658873
R-squared score (training): 0.679
R-squared score (test): 0.492
```



Regresion Lineal en Python

Linear regression

```
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X_R1, y_R1,
                                                    random_state = 0)
linreg = LinearRegression().fit(X_train, y_train)

print('linear model coeff (w): {}'.format(linreg.coef_))
print('linear model intercept (b): {:.3f}'.format(linreg.intercept_))
print('R-squared score (training): {:.3f}'.format(linreg.score(X_train, y_train)))
print('R-squared score (test): {:.3f}'.format(linreg.score(X_test, y_test)))
```

```
linear model coeff (w): [ 45.71]
linear model intercept (b): 148.446
R-squared score (training): 0.679
R-squared score (test): 0.492
```

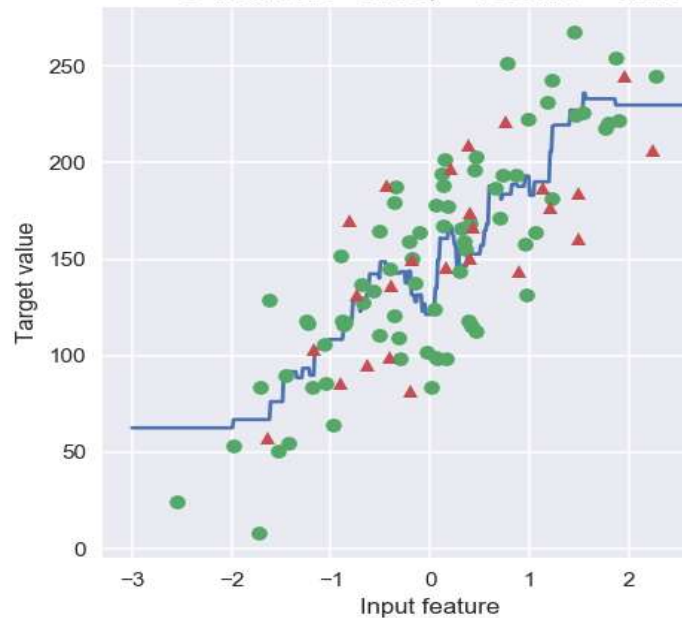
Métodos básicos de aprendizaje automático supervisado

- **Algoritmos simples pero potentes:**
 - *K-nearest neighbors*
 - *Linear model fit using least-squares*
- **Estos representan dos enfoques complementarios del aprendizaje supervisado:**
 - *K-vecinos más cercanos hace pocas suposiciones sobre la estructura de los datos y proporciona predicciones potencialmente precisas, pero a veces inestables (sensibles a pequeños cambios en los datos de entrenamiento).*
 - *Los modelos lineales hacen suposiciones sólidas sobre la estructura de los datos y dan predicciones estables pero potencialmente inexactas.*

KNN y Regresion Lineal

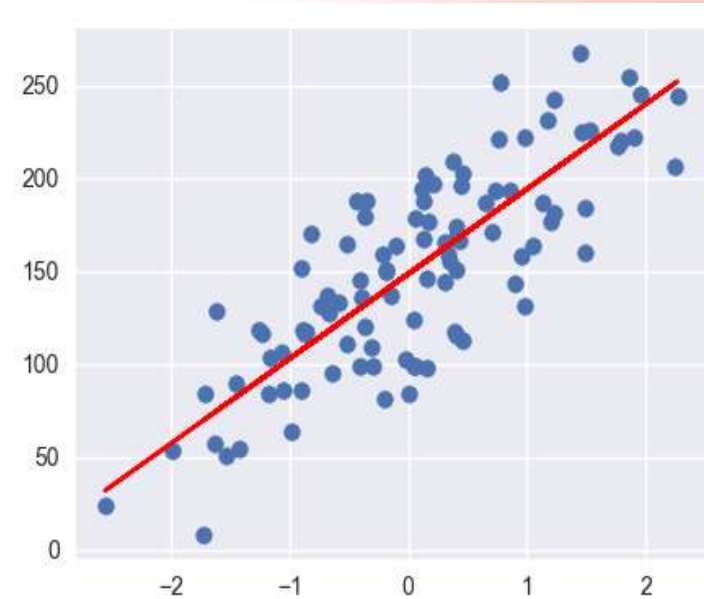
k-NN (k=7)

k-NN regression (K=7)
'o' Train $R^2 = 0.720$, '^' Test $R^2 = 0.471$



R-squared score (training): 0.720
R-squared score (test): 0.471

LS linear

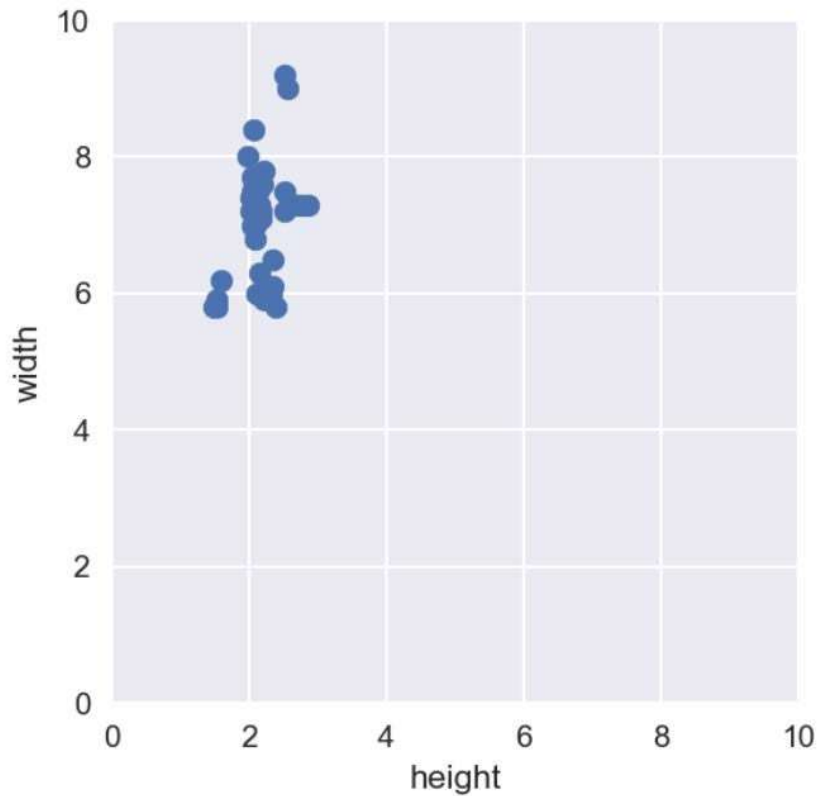


R-squared score (training): 0.679
R-squared score (test): 0.492

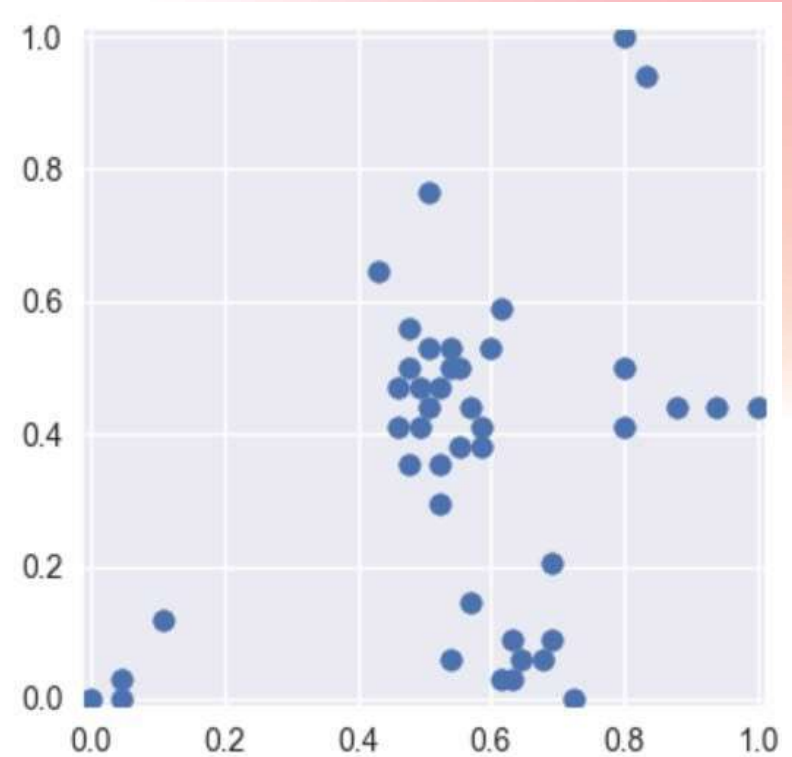
Normalización de características

- Es importante que algunos métodos de aprendizaje automático tengan todas las características en la misma escala (por ejemplo, una convergencia más rápida en el aprendizaje, una influencia más uniforme o "justa" para todos los pesos)
- –por ejemplo, regresión regularizada, k-NN, máquinas de vectores de soporte, redes neuronales, ...
- Hacemos el escalado MinMax de las características:
 - For each feature x_i : compute the min value x_i^{MIN} and the max value x_i^{MAX} achieved across all instances in the training set.
 - For each feature: transform a given feature x_i value to a scaled version x'_i using the formula
$$x'_i = (x_i - x_i^{MIN}) / (x_i^{MAX} - x_i^{MIN})$$

Escalador MinMax



Unnormalized data points



Normalized with MinMaxScaler

Fit Transform

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)  clf
= Ridge().fit(X_train_scaled, y_train)
r2_score = clf.score(X_test_scaled, y_test)
```

Tip: It can be more efficient to do fitting and transforming together on the training set using the `fit_transform` method.

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

Normalización de características: el conjunto de pruebas debe usar un escalado idéntico al del conjunto de entrenamiento

- Ajuste el escalador con el conjunto de entrenamiento y, a continuación, aplique el mismo escalador para transformar el conjunto de prueba.
- No escale los conjuntos de entrenamiento y prueba con diferentes escaladores: esto podría provocar un sesgo aleatorio en los datos.
- No coloque el escalador utilizando ninguna parte de los datos de prueba: hacer referencia a los datos de prueba puede provocar una forma de fuga de datos.

Regresión de cresta (Ridge)

- La regresión de Ridge aprende w, b utilizando el mismo criterio de mínimos cuadrados, pero agrega una penalización para grandes variaciones en los parámetros

$$RSS_{RIDGE}(w, b) = \sum_{\{i=1\}}^N (y_i - (w \cdot x_i + b))^2 + \alpha \sum_{\{j=1\}}^p w_j^2$$

- Una vez que se aprenden los parámetros, la fórmula de predicción de la regresión de cresta es la misma que la de los mínimos cuadrados ordinarios.
- La adición de una penalización de parámetro se denomina regularización. La regularización evita el sobreajuste restringiendo el modelo, normalmente para reducir su complejidad.
- Ridge utiliza la regularización L2: minimizar la suma de los cuadrados de las entradas w
- La influencia del término de regularización está controlada por el parámetro α . Un alfa más alto significa más regularización y modelos más simples.

Ridge regression

```
from sklearn.linear_model import Ridge
X_train, X_test, y_train, y_test = train_test_split(X_crime, y_crime,
                                                    random_state = 0)

linridge = Ridge(alpha=20.0).fit(X_train, y_train)
print('Crime dataset')
print('ridge regression linear model intercept: {}'.format(linridge.intercept_))
print('ridge regression linear model coeff:\n{}'.format(linridge.coef_))
print('R-squared score (training): {:.3f}'.format(linridge.score(X_train, y_train)))
print('R-squared score (test): {:.3f}'.format(linridge.score(X_test, y_test)))
print('Number of non-zero features: {}'.format(np.sum(linridge.coef_ != 0)))
```

Lasso Regresión

- La regresión de lazo es otra forma de regresión lineal regularizada que utiliza una penalización de regularización L1 para el entrenamiento (en lugar de la penalización L2 de cresta)
- **Penalización L1: Minimizar la suma de los valores absolutos de los coeficientes**

$$RSS_{LASSO}(\mathbf{w}, b) = \sum_{\{i=1\}}^N (y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2 + \alpha \sum_{\{j=1\}}^p |w_j|$$

Lasso Regresión

- **Esto tiene el efecto de establecer los pesos de los parámetros en w a cero para las variables menos influyentes. A esto se le llama una solución dispersa: un tipo de selección de características**
- Complejidad del modelo: - Alfa alto: Modelo más simple (más disperso) - Alfa bajo: Modelo más complejo (menos disperso) - Alfa = 0: Equivalente a la regresión lineal estándar
- Compensación entre sesgos y varianzas: - Alfa más alto: aumenta el sesgo, reduce la varianza - Alfa más bajo: reduce el sesgo, aumenta la varianza
- **El parámetro α controla la cantidad de regularización L1 (por defecto = 1,0).**
- **La fórmula de predicción es la misma que la de los mínimos cuadrados ordinarios.**
- **Cuándo usar la regresión de cresta frente a la regresión de lazo:**
 - *Muchos efectos de tamaño pequeño/mediano: use cresta.*
 - *Solo algunas variables con efecto medio/grande: use el lazo.*

Lasso regression

```
from sklearn.linear_model import Lasso
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X_train, X_test, y_train, y_test = train_test_split(X_crime, y_crime, random_state = 0)
#X_train, X_test, y_train, y_test = train_test_split(X_cancer, y_cancer, random_state = 0)

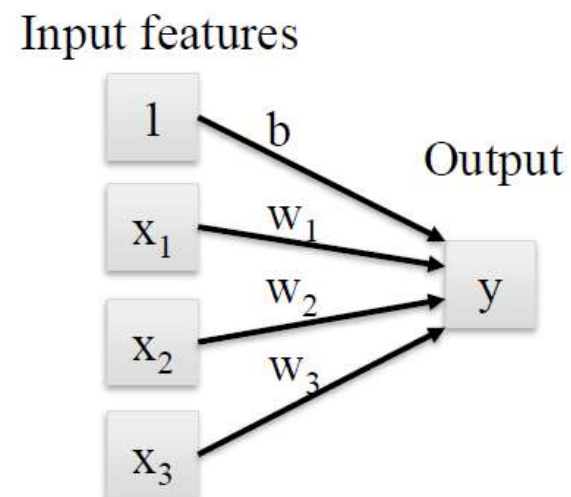
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

linlasso = Lasso(alpha=2.0, max_iter = 10000).fit(X_train_scaled, y_train)

print('Crime dataset')
print('lasso regression linear model intercept: {}'.format(linlasso.intercept_))
print('lasso regression linear model coeff:\n{}'.format(linlasso.coef_))
print('Non-zero features: {}'.format(np.sum(linlasso.coef_ != 0)))
print('R-squared score (training): {:.3f}'.format(linlasso.score(X_train_scaled, y_train)))
print('R-squared score (test): {:.3f}\n'.format(linlasso.score(X_test_scaled, y_test)))
print('Features with non-zero weight (sorted by absolute magnitude):')

for e in sorted(list(zip(list(X_crime), linlasso.coef_)),
                 key = lambda e: -abs(e[1])):
    if e[1] != 0:
        print('\t{0}, {:.3f}'.format(e[0], e[1]))
```

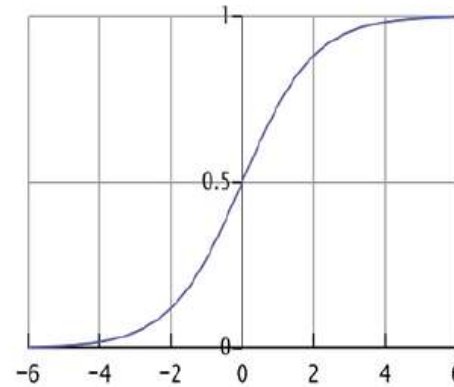
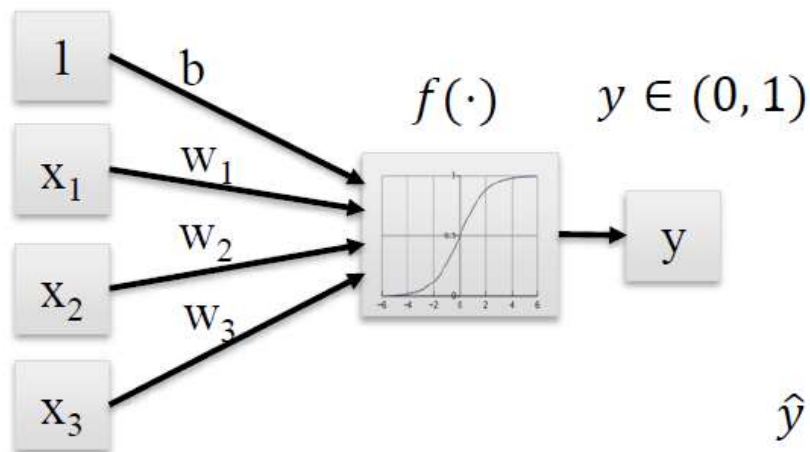
Regresión lineal



$$\hat{y} = \hat{b} + \hat{w}_1 \cdot x_1 + \dots \hat{w}_n \cdot x_n$$

Linear models for classification: Logistic Regression

Input features



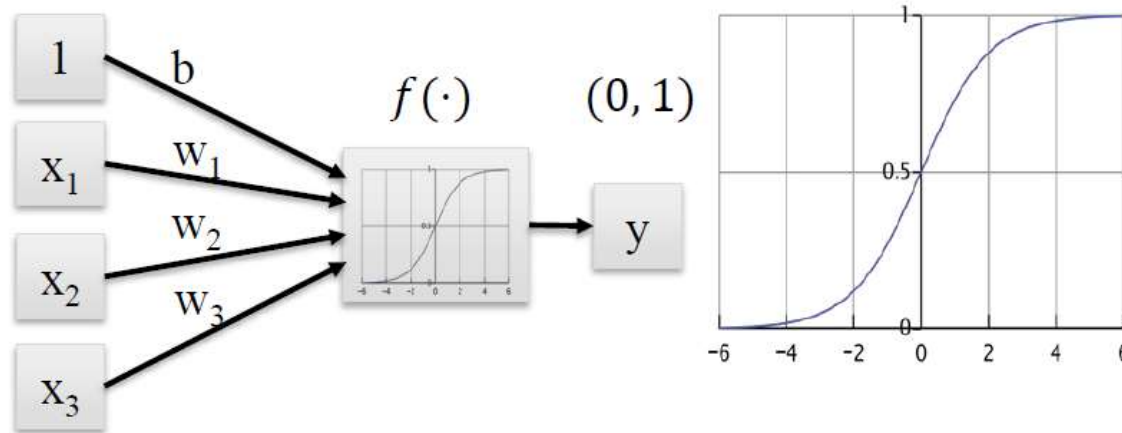
$$\hat{y} = \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)$$

$$= \frac{1}{1 + \exp[-(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)]}$$

● Title

Linear models for classification: Logistic Regression

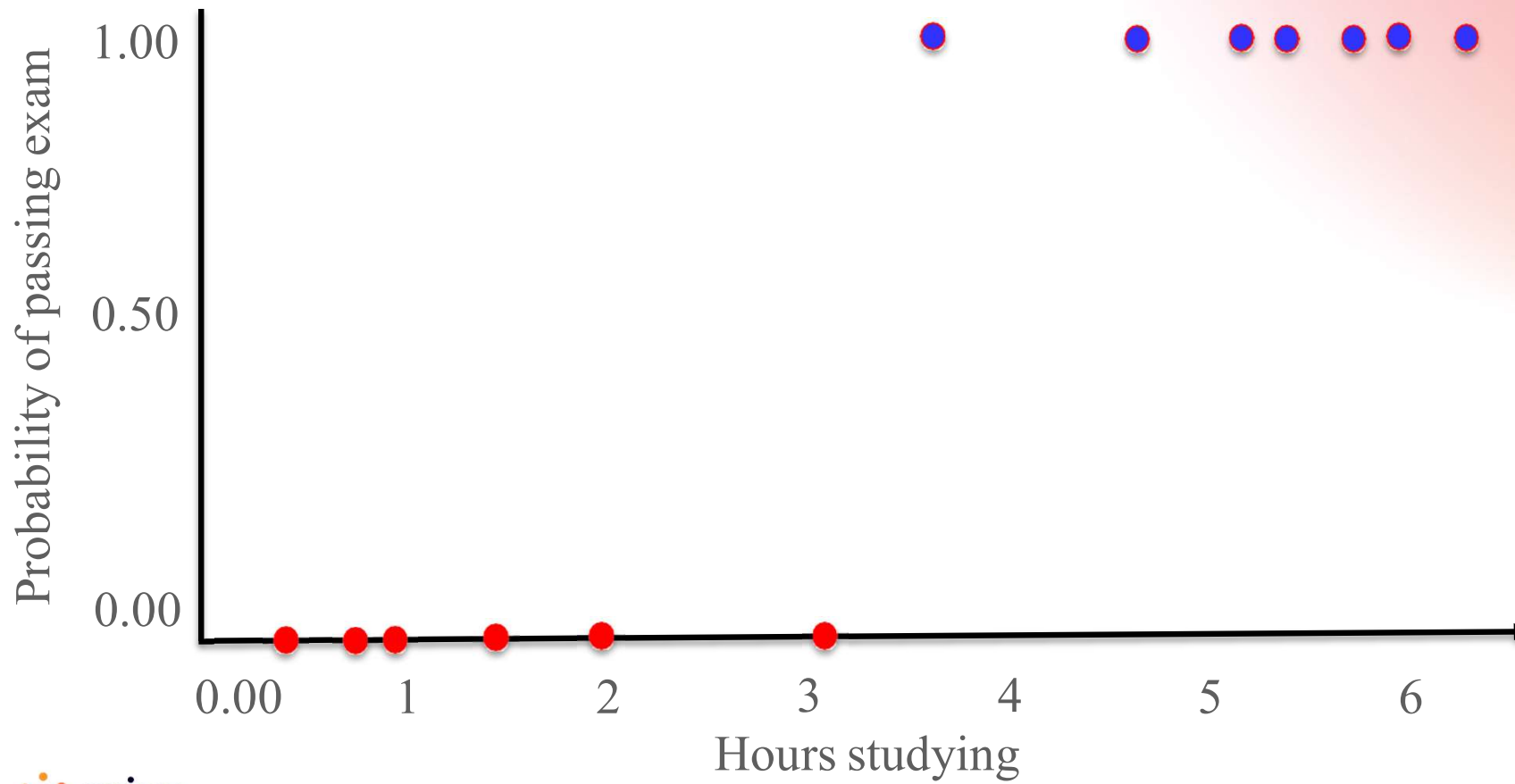
Input features



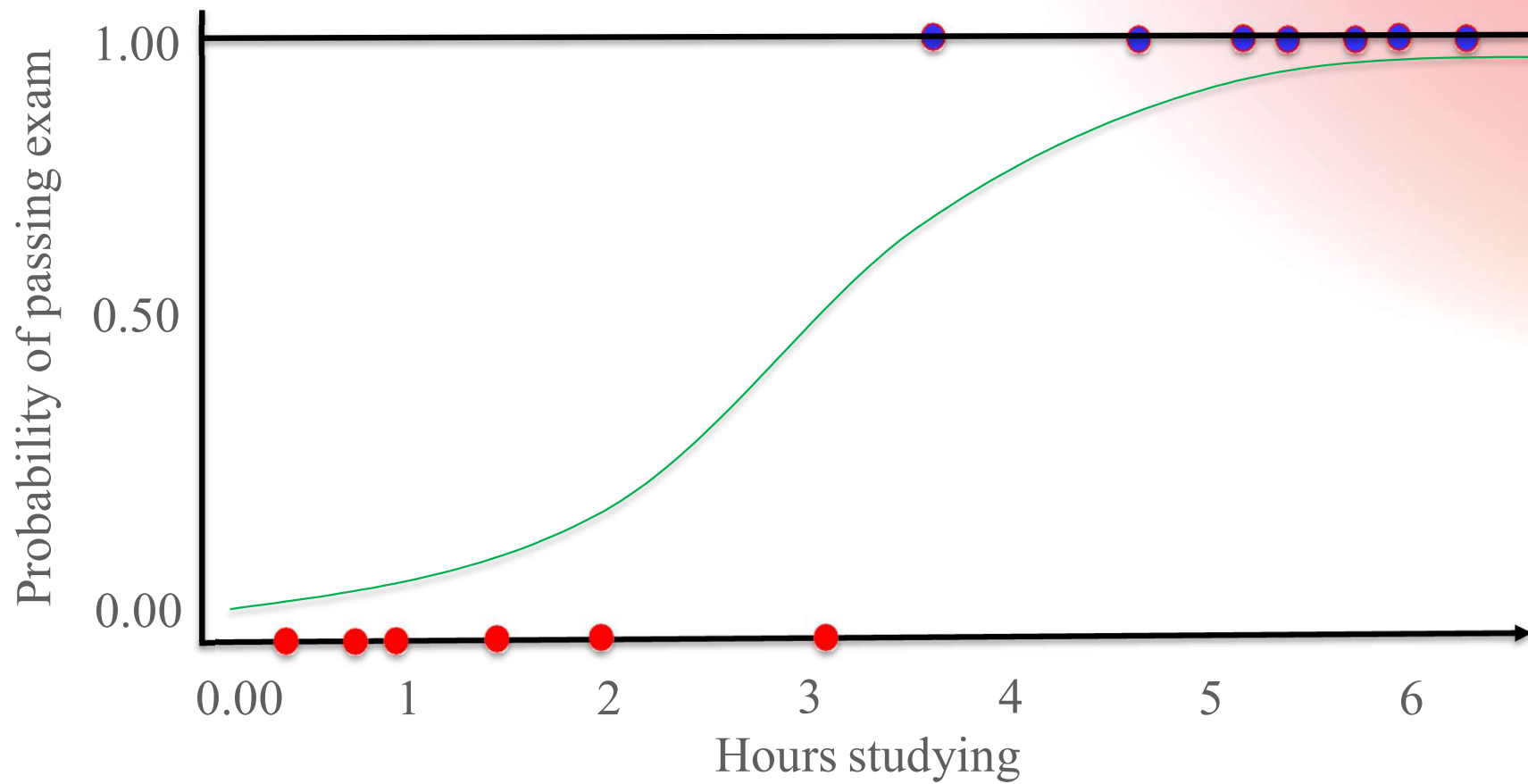
The logistic function transforms real-valued input to an output number y between 0 and 1, interpreted as the probability the input object belongs to the positive class, given its input features (x_0, x_1, \dots, x_n)

$$\begin{aligned}\hat{y} &= \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \dots \hat{w}_n \cdot x_n) \\ &= \frac{1}{1 + \exp[-(\hat{b} + \hat{w}_1 \cdot x_1 + \dots \hat{w}_n \cdot x_n)]}\end{aligned}$$

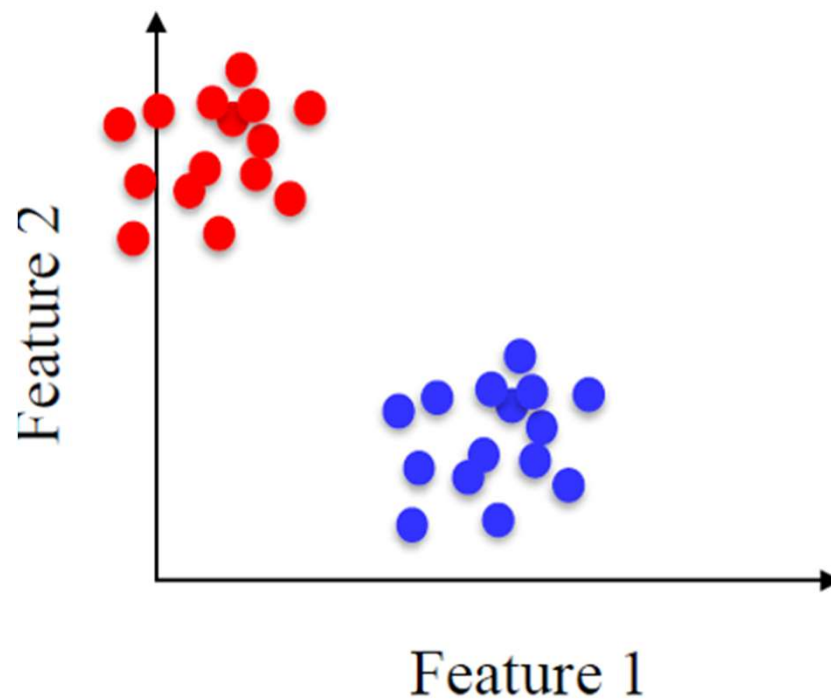
Regresión logística



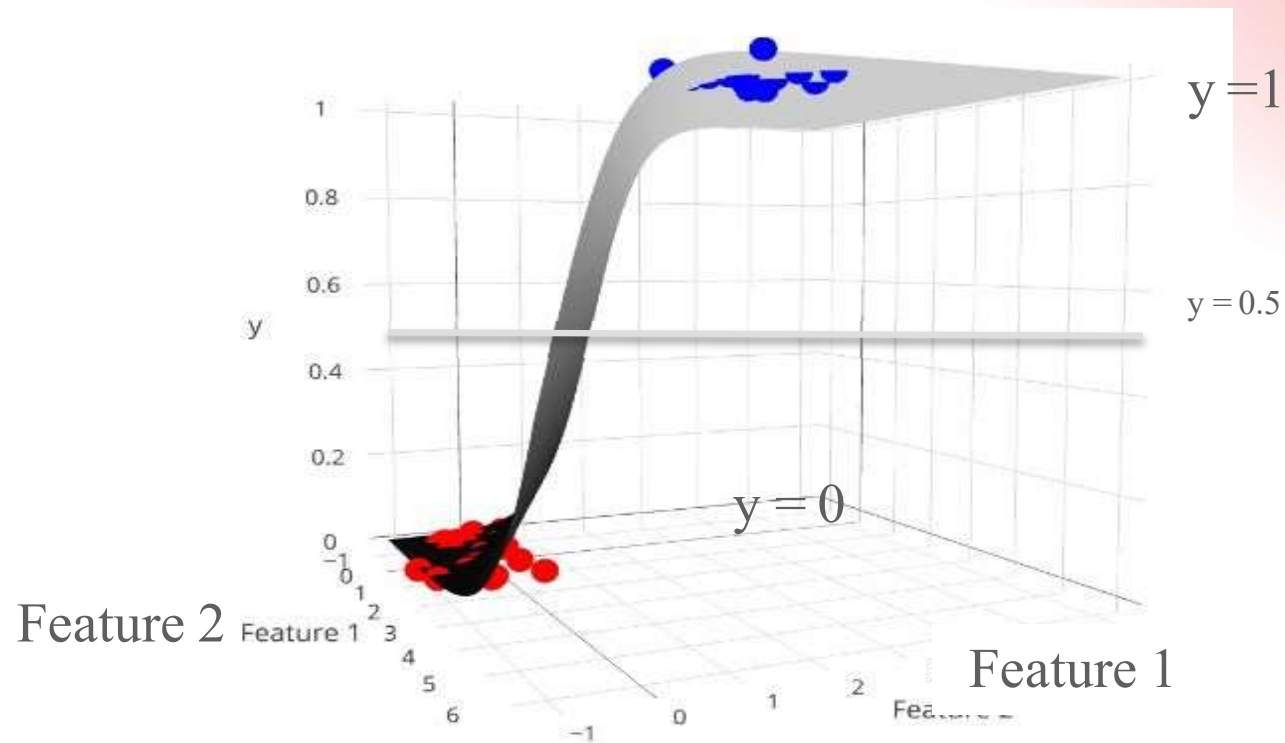
Regresión logística



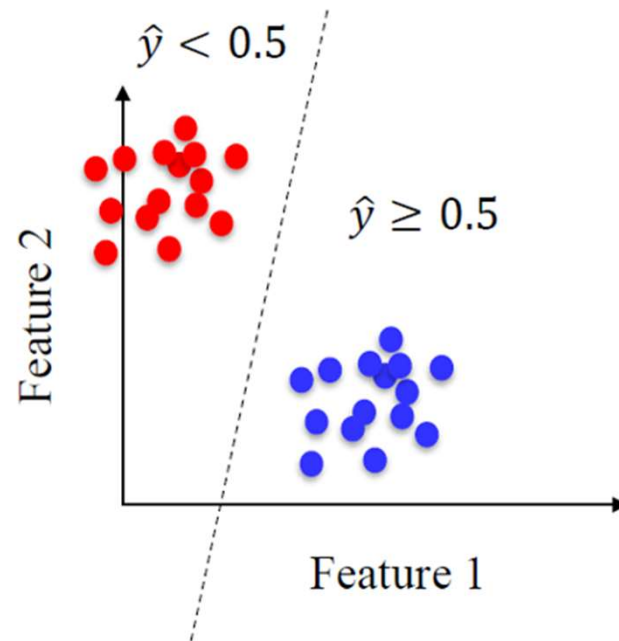
Regresión logística para la clasificación binaria



Regresión logística para la clasificación binaria

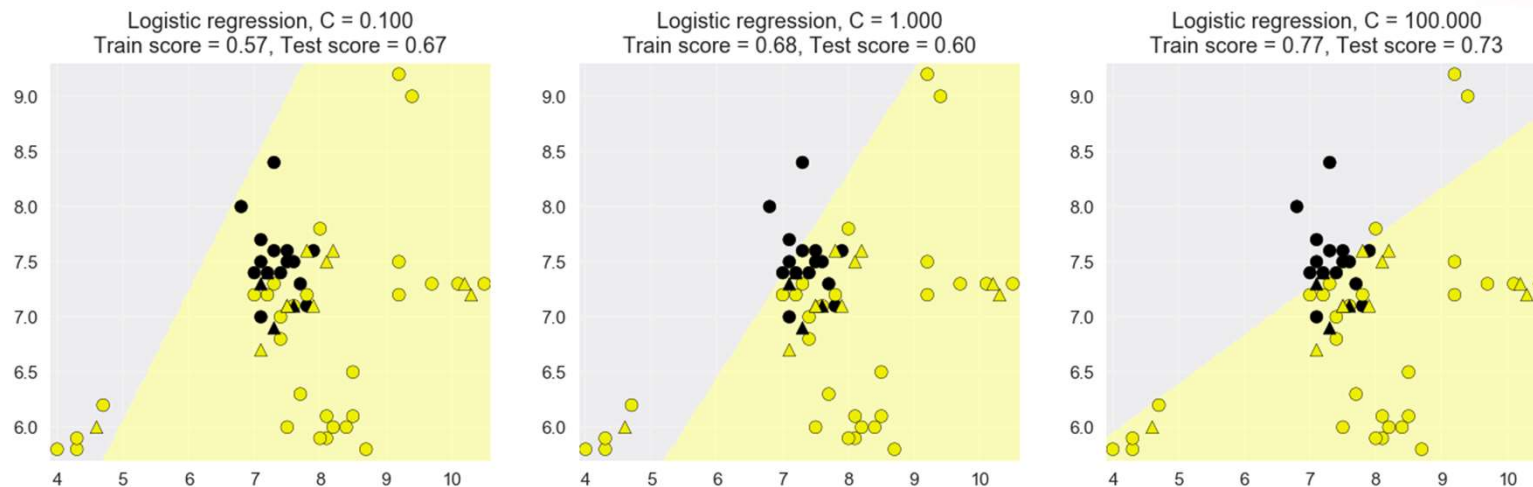


Regresión logística para la clasificación binaria



Regresión logística: Regularización



- La regularización L2 está 'activada' de forma predeterminada (como la regresión de cresta)
- El parámetro C controla la cantidad de regularización (por defecto 1.0)
- Al igual que con la regresión lineal regularizada, puede ser importante normalizar todas las entidades para que estén en la misma escala.



Regresión logística: Regularización

- Small C (Strong Regularization): More regularization - Simpler model - Higher bias, lower variance - Better for preventing overfitting
- Large C (Weak Regularization): - Less regularization - More complex model - Lower bias, higher variance - May lead to overfitting
- 4. When to Adjust C:
 - High variance → Decrease C
 - High bias → Increase C –
 - Small dataset → Consider smaller C
 - Large dataset → Can use larger C
- 5. Practical Tips:
 - Use grid search or random search for tuning
 - Common range: 0.001 to 1000 -



#AIskills4all |  @AIskillsEU |  [linkedin.com/AIskillsEU](https://www.linkedin.com/AIskillsEU)



www.aiskills.eu

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.