Regresión Lineal

Walter Andrés Fontecha

Lilibeth Ramirez Esquivel

Fundación Universitaria uninpahu

Ingeniería de datos

Bogotá, 2025

**Introducción**

En la actualidad, el análisis de datos se ha vuelto fundamental en diversos campos como la economía, la ciencia, la salud y la tecnología. Por eso, cada vez es más importante usar métodos que nos ayuden a entender la información y predecir lo que podría pasar. Uno de los métodos más comunes con este propósito es la regresión, una técnica estadística que permite identificar relaciones entre variables y construir modelos para hacer predicciones.

La regresión lineal es un método estadístico utilizado para el análisis predictivo. Es decir, es una herramienta que usamos para entender cómo cambia una cosa en función de otra. Es muy usada para hacer predicciones, ya que permite identificar patrones y tendencias en la información. Esto ayuda a anticipar lo que podría pasar en el futuro o a completar datos que faltan sin tener que hacerlo manualmente.

Una de las cosas más interesantes de la regresión lineal es que busca una conexión directa entre lo que queremos predecir (la variable dependiente) y los factores que creemos que afectan a eso (las variables independientes). Si usamos solo una variable para hacer la predicción, la relación entre las dos cosas se puede representar con una línea recta, y eso es lo que se llama "regresión lineal".

Esta relación se puede escribir con una fórmula matemática que también se puede dibujar en una gráfica. La forma más simple de esta fórmula es:

**Y = mX + b**

Donde:

* **Y** es lo que queremos saber o predecir.
* **X** es el dato que ya tenemos.
* **m** muestra cuánto cambia Y cuando X cambia.
* **b** es el punto donde la línea cruza el eje vertical.

Dependiendo de cómo sea la relación entre las variables, tenemos tres formas de usar la regresión:

**Regresión lineal simple**: Este es el modelo más básico, donde solo se usa una variable para predecir algo.

**Regresión lineal múltiple**: Este modelo es como el anterior, pero con más de una variable que puede influir en lo que estamos tratando de predecir.

**Regresión no lineal**: Este modelo es un poco más complicado porque no sigue una línea recta, sino que tiene una forma más curvada.

Este trabajo tiene como objetivo principal abordar en detalle y con mayor profundidad los diferentes tipos de regresión analizando sus principales características, usos, beneficios y posibles desventajas. Se incluirán ejemplos prácticos para ilustrar cada modelo y se explicarán los escenarios en los que resulta más conveniente aplicarlos.

**OBJETIVOS**

1. Comprender las relaciones entre las variables
2. Identificar relaciones lineales en grandes volúmenes
3. Evaluar el impacto en cada uno de los problemas propuestos
4. 4. Utilizar regresión lineal para predecir los resultados finales

1. **Regresión Lineal**

La regresión lineal es una pieza fundamental del análisis de datos permitiendo la integridad entre las variables. Siendo este modelo eficiente para interpretar datos y procesarlos en poco tiempo.

En Big Data, la regresión lineal los modelos de regresión lineal son relativamente simples y proporcionan una fórmula matemática fácil de interpretar para generar predicciones. La regresión lineal es una técnica estadística establecida y se aplica fácilmente al software y a la computación. Las empresas lo utilizan para convertir datos sin procesar de manera confiable y predecible en inteligencia empresarial y conocimiento práctico. Los científicos de muchos campos, incluidas la biología y las ciencias del comportamiento, ambientales y sociales, utilizan la regresión lineal para realizar análisis de datos preliminares y predecir tendencias futuras.

A continuación se presenta un ejemplo de regresión lineal con una sola variable

**1.2 Descripción del problema**

En esta parte del ejercicio, se implementará regresión lineal con una variable para predecir la nota obtenida de un estudiante según el tiempo de estudio invertido en Bogotá.

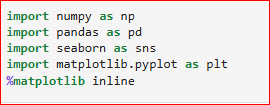
El archivo ex1data1.txt contiene el conjunto de datos para nuestro problema de regresión lineal.



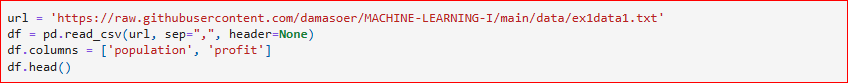
La primera columna representa las horas de estudio de un estudiante en Bogotá y la segunda columna representa la nota final obtenida en la ciudad de Bogotá, esta descripción se realizará a través de las diferentes codificaciones

**1.3 Trazar los datos o códigos**

Importación de las principales librerías usadas en análisis, visualización y manipulación de datos



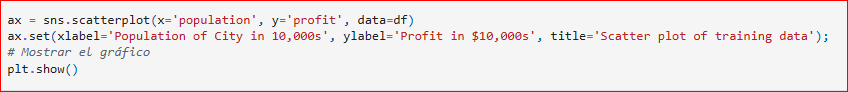
cargar y visualizar las primeras filas de un archivo CSV (valores separados por comas) que contiene datos sobre horas de estudio y notas finales. Vamos a ver qué hace línea por línea

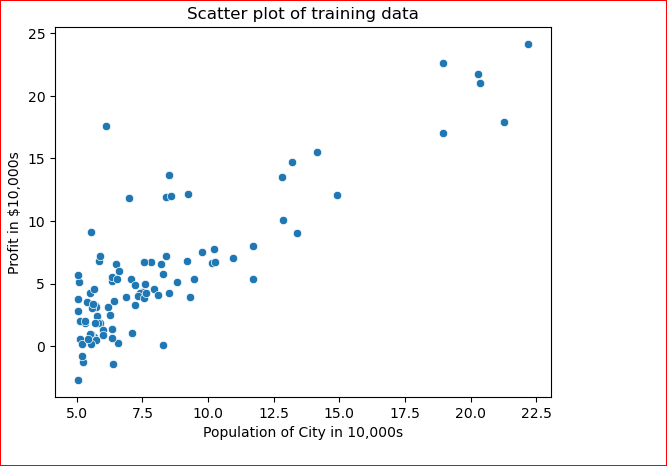


**1.3.1 Resultado**



**1.3.2 : Gráfico de dispersión** (scatter plot) para visualizar la relación entre las horas de estudio y la nota final usando la biblioteca Seaborn.





**La trama muestra que tienen una relación lineal.**

**1.4 Descenso por gradientes:**

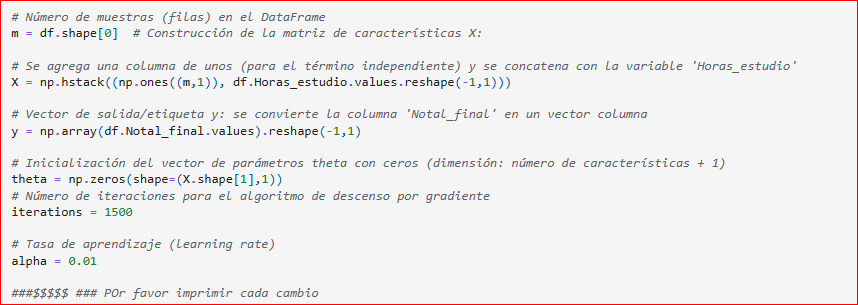
Ajustar los parámetros de la regresión linéa a los datos utilizando el descenso por gradientes

**1.4.1 Ecuaciones**

En la regresión lineal, las hipótesis principales son sobre la relación entre las variables dependiente e independiente. Se busca establecer si existe una relación lineal significativa y, en caso afirmativo, cómo de fuerte es esa relación. Se formulan dos hipótesis principales: la hipótesis nula (H0) y la hipótesis alternativa (H1).

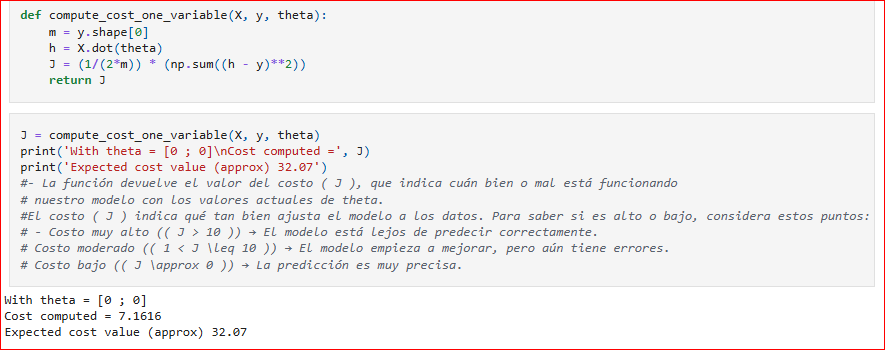
**1.4.2 Implementación**

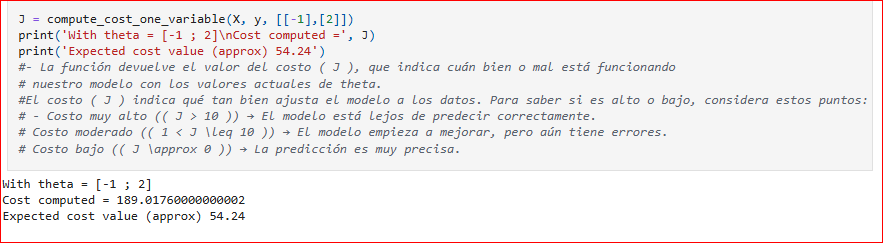
En programación, se crea el código para el descenso del gradiente



**1.4.3 Calculando la Función de Costo:**

Esa función compute\_cost\_one\_variable calcula el costo (o error promedio cuadrático) de un modelo de regresión lineal con una sola variable (aunque puede generalizarse a varias).



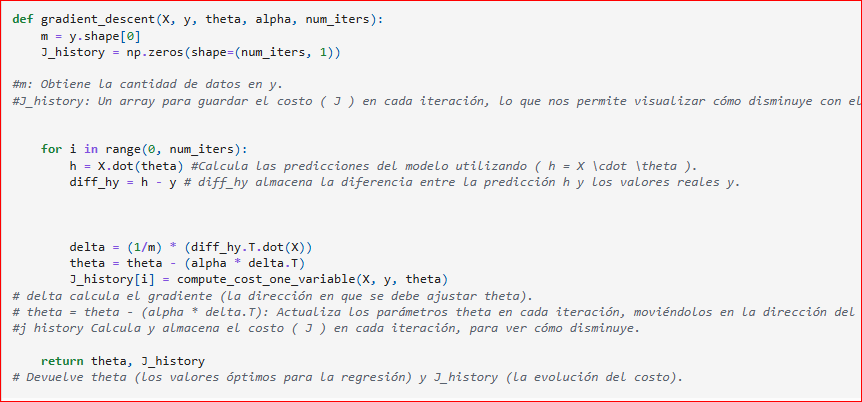


**1.4.4: Gradiente Descendiente**

es un algoritmo de optimización genérico que mide el gradiente local de la función de costo con respecto al parámetro y avanza en la dirección del gradiente descendente.

**Algoritmo:**

* Tasa de aprendizaje demasiado pequeña: descenso de gradiente lento
* Tasa de aprendizaje demasiado grande: el descenso del gradiente puede sobrepasar el mínimo y puede no converger



*#ejecuta el algoritmo de descenso por gradiente para encontrar los valores óptimos de theta en una regresión lineal.*

theta, \_ **=** gradient\_descent(X, y, theta, alpha, iterations)

print('Theta found by gradient descent:\n', theta)

*# Muestra los valores finales de theta, los cuales representan la mejor línea de ajuste para los datos.*

print('Expected theta values (approx)\n -3.6303\n 1.1664')

*# Presenta los valores aproximados esperados de theta, permitiendo verificar si la optimización fue exitosa*

Theta found by gradient descent:

[[1.74620969]

[0.1776025 ]]

Expected theta values (approx)

-3.6303

1.1664

*# Resultado de Theta Found*

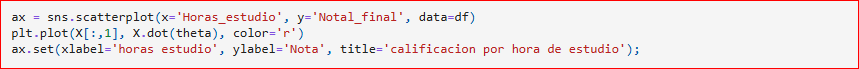
*#- Intercepto (theta[0] = 1.7462) → Representa el valor predicho cuando Horas\_estudio = 0. Es el punto de inicio de la recta de regresión.*

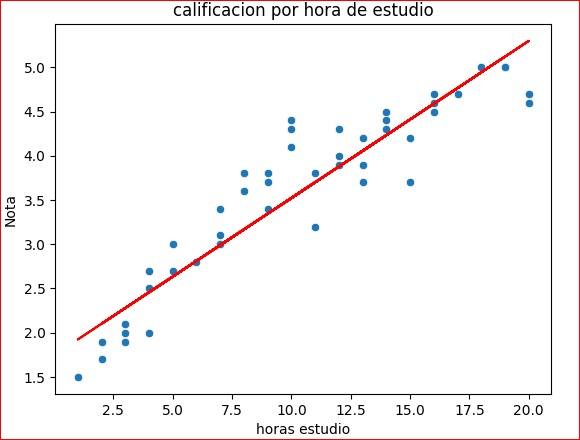
*# Pendiente (theta[1] = 0.1776) → Indica cuánto cambia la Nota\_final por cada hora adicional de estudio.*

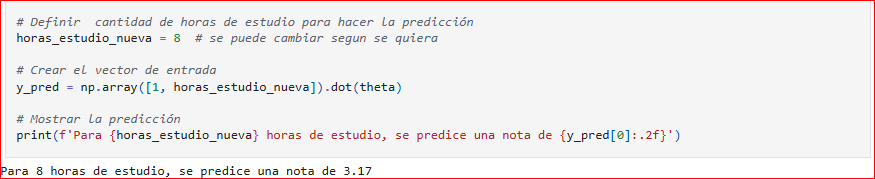
*# En este caso, cada hora de estudio incrementa la nota en aproximadamente 0.1776.*

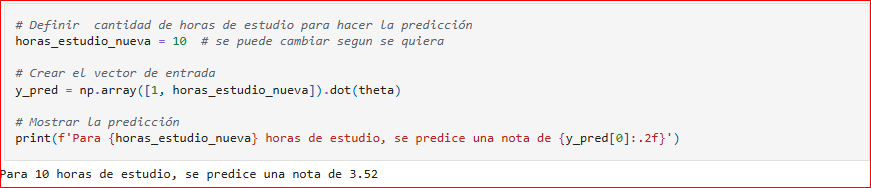
**1.4.5 Grafica el ajuste lineal**

Visualización de los datos y la línea de regresión lineal





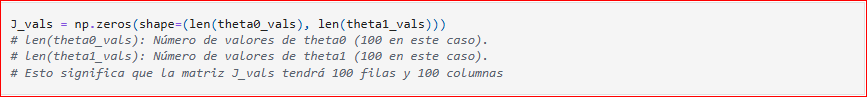




**1.4.6 Visualizante J(0)**

theta0\_vals **=** np**.**linspace(**-**10, 10, 100) *# Representa posibles valores para theta0, el término independiente en una regresión lineal.*

theta1\_vals **=** np**.**linspace(**-**1, 4, 100) *# - Representa posibles valores para theta1, el coeficiente de la variable predictora*

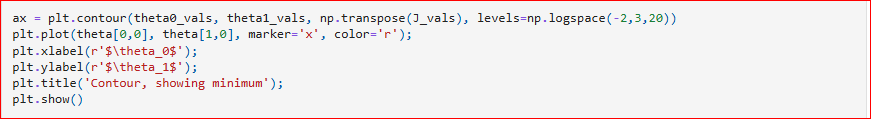


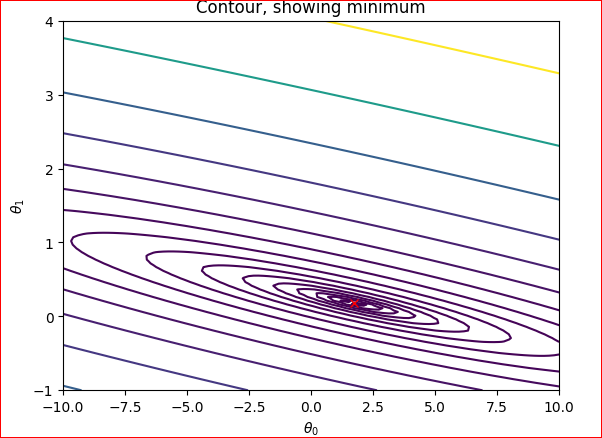
**for** i **in** range(0, len(theta0\_vals)): *# Recorre todos los valores de theta0*

**for** j **in** range(0, len(theta1\_vals)): *# Recorre todos los valores de theta1*

J\_vals[i,j] **=** compute\_cost\_one\_variable(X, y, [[theta0\_vals[i]], [theta1\_vals[j]]]) *# Calcula el costo para cada combinación de theta0 y theta1*

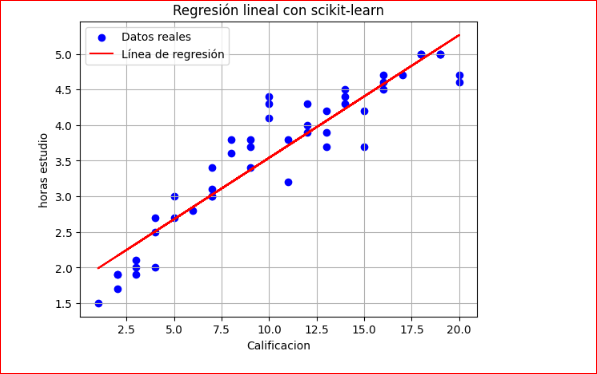
*#grafica de contorno que muestra cómo varía la función de costo ( J(\theta) ) en función de los valores de ( \theta\_0 ) y ( \theta\_1 )*





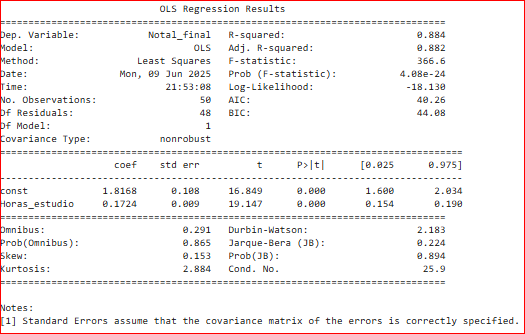
### 1.4.7 Usando sklearn

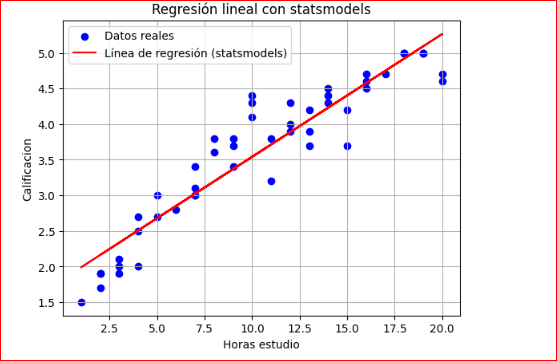




### 1.4.8. Usando statsmodels

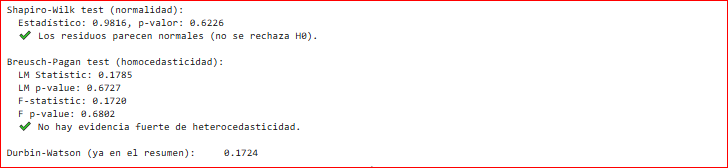


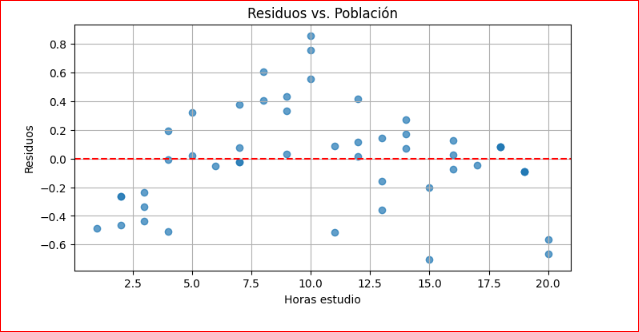




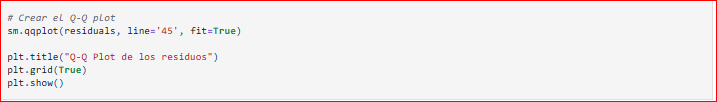


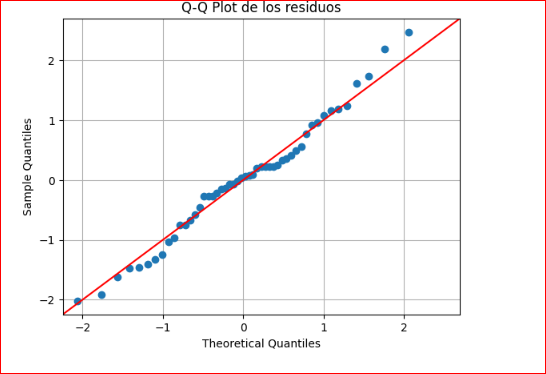






## 1.4.9 Crear el Q-Q plot

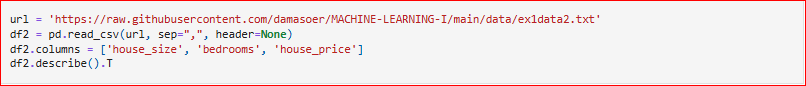


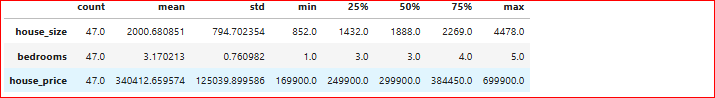


**1.5 Regresión lineal con múltiples variables**

En esta parte, implementará una regresión lineal con múltiples variables para predecir los precios de las viviendas. Suponga que está vendiendo su casa y desea saber cuál sería un buen precio de mercado. Una forma de hacerlo es recopilar información sobre las casas vendidas recientemente y crear un modelo de precios de vivienda. El archivo ex1data2.txt contiene un conjunto de entrenamiento de precios de vivienda en Portland, Oregón. La primera columna representa el tamaño de la casa (en pies cuadrados), la segunda el número de habitaciones y la tercera el precio de la vivienda.

**1.5.1 Normalización de funciones**



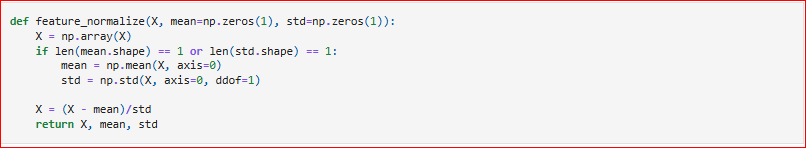


Al observar los valores, note que los tamaños de las casas son aproximadamente 1000 veces mayores que el número de habitaciones. Cuando las características difieren por órdenes de magnitud, se realiza escalado d(escalado de características) previamente puede hacer que el descenso por gradiente converja mucho más rápido

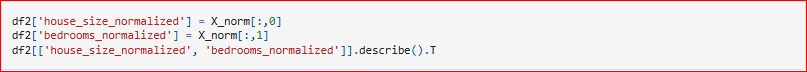
Podemos acelerar el descenso por gradiente si hacemos que cada uno de nuestros valores de entrada esté, idealmente, en un rango similar 

* Escala de características: implica dividir los valores de entrada por el rango (es decir, el valor máximo menos el valor mínimo) de la variable de entrada.
* Normalización de media: implica restar el valor promedio de una variable de entrada de los valores de esa variable de entrada

 donde  es el promedio de todos los valores de las características (i) y  es el rango de valores (máximo-mín.), la desviación estándar.





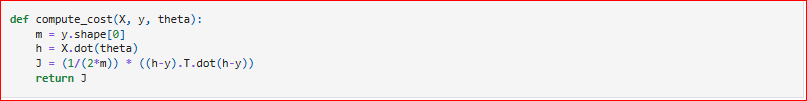


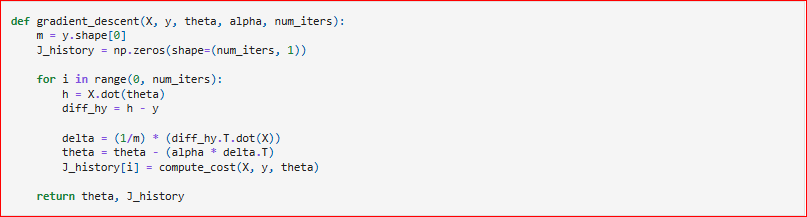
**1.5.2 Descenso de gradiente**

La única diferencia con el problema de regresión univariada es que ahora hay una característica más en la matriz X. La función de hipótesis y la regla de actualización del descenso por gradiente por lotes permanecen sin cambios.

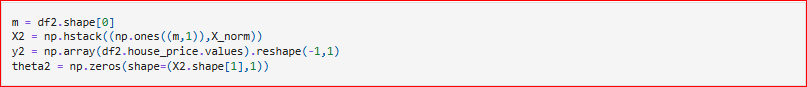
Nota: En el caso multivariable, la función de costo también puede escribirse en la siguiente forma vectorizada:

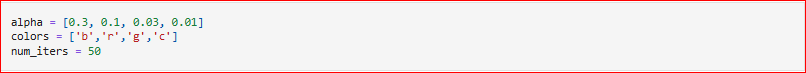


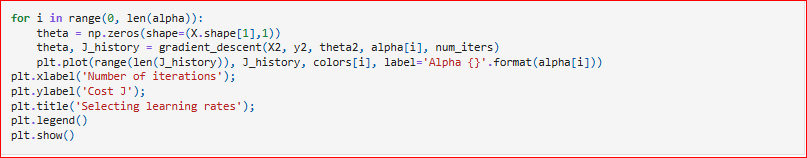


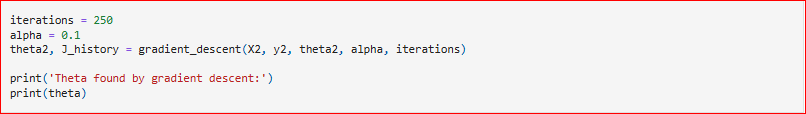


**1.5.3 Seleccionando Tasa de aprendizaje**

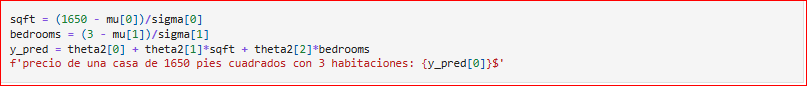








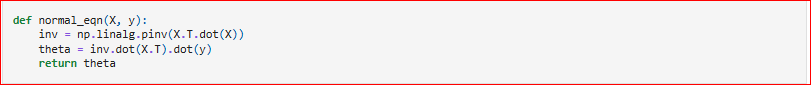
##### Estima el precio de una casa de 1650 pies cuadrados con 3 habitaciones.

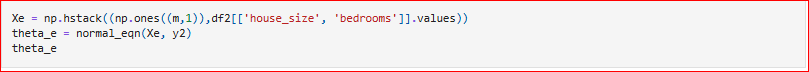


**1.5.4 Ecuaciones normales**

Una solución de forma cerrada para encontrar sin iteración.

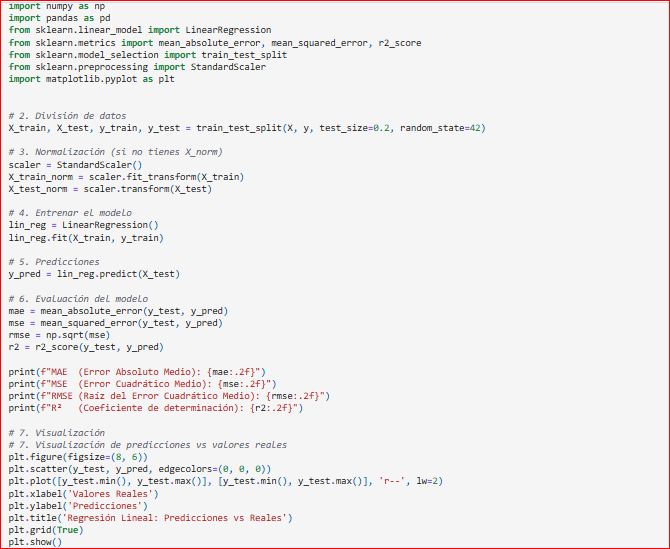








**1.5.5 Código equivalente usando Scikit-Learn**







1. **Programming Exercise 2: Regresión Logística**

**2.1 Descripción del problema**

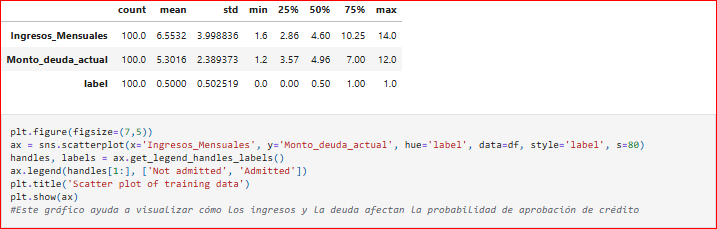
En esta parte del ejercicio, construirás un modelo de regresión logística para predecir si un estudiante será admitido en una universidad. Supón que eres el administrador de un departamento universitario y deseas determinar la probabilidad de admisión de cada solicitante en función de sus resultados en dos exámenes. Cuentas con datos históricos de solicitantes anteriores que pueden utilizar como conjunto de entrenamiento para la regresión logística. Para cada ejemplo de entrenamiento, tienes las calificaciones del solicitante en los dos solicitudes y la decisión Tu tarea es construir un modelo de clasificación que estime la probabilidad de admisión de un solicitante obteniendo en las calificaciones de esos dos exámenes.

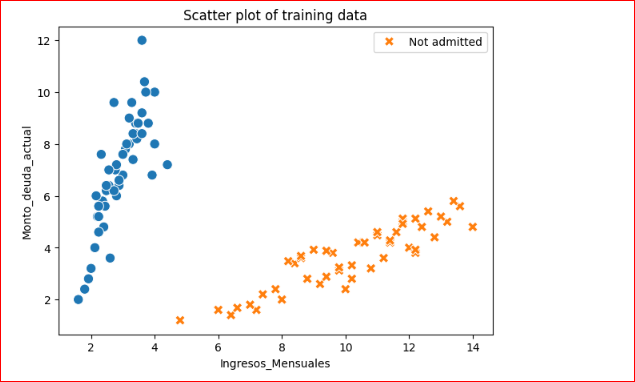
* 1. **Visualizando los datos**







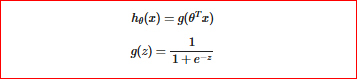




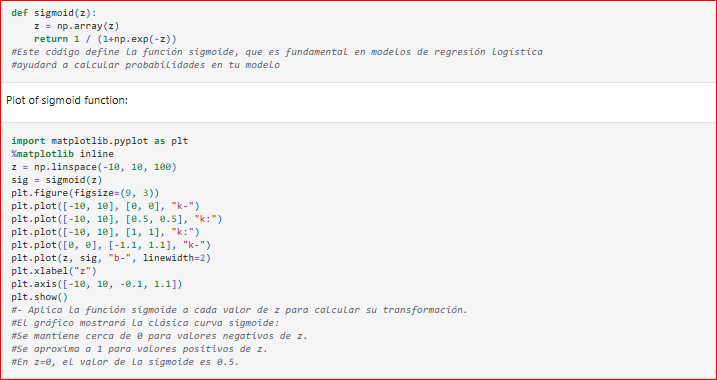
### Implementacion

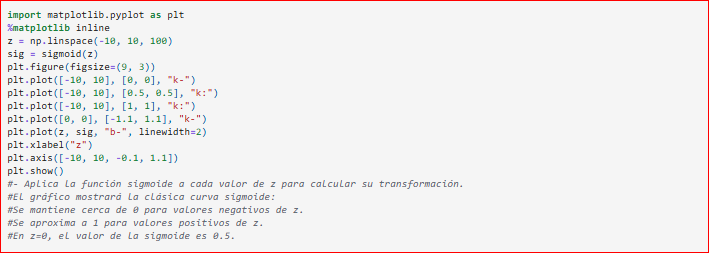
### Función sigmoidea

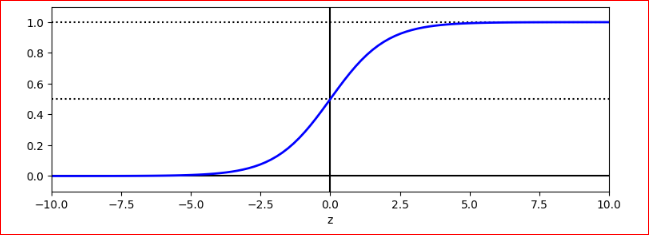
Logistic regression hypothesis:



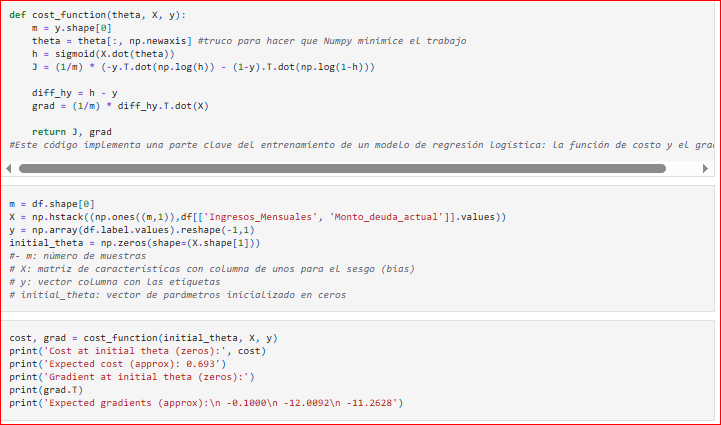
2.3.2 Gráfico de la función sigmoidea:

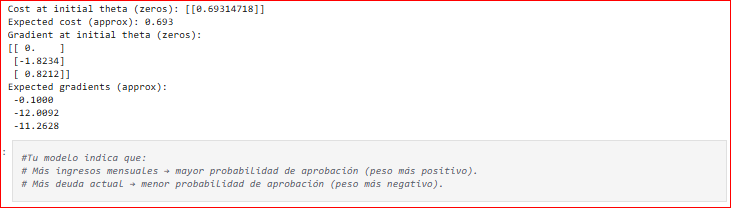


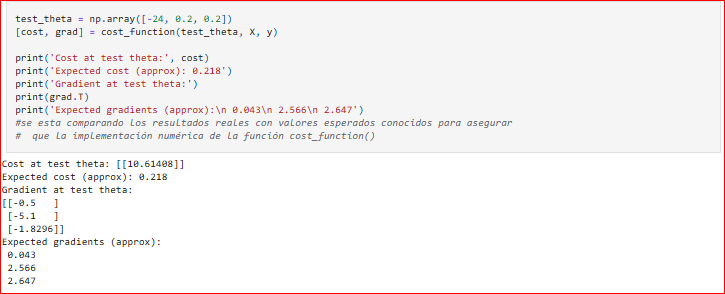




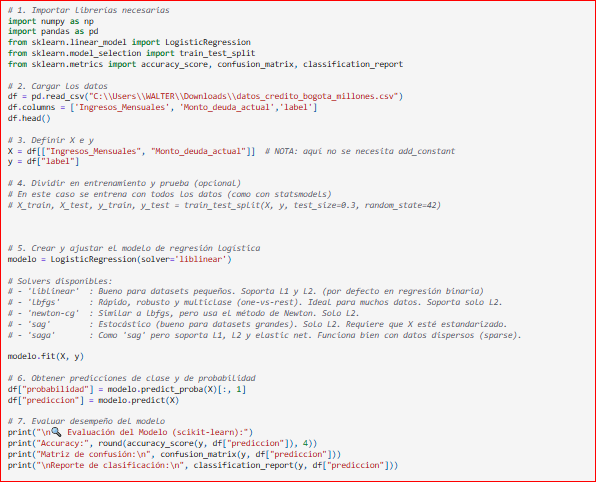
**2.4 Función de costo y gradiente**







* 1. **Codigo Equivalente usando Scikit-Learn:**



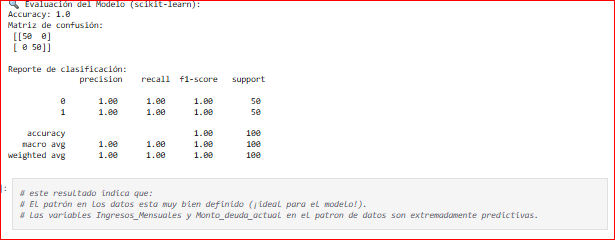
*NOTAS IMPORTANTES*

*#LogisticRegression() usa regularización L2 por defecto. Puedes cambiarla con penalty='l1' y usar solver='liblinear' para compatibilidad.*

*#Puedes ajustar la fuerza de regularización con el parámetro C. Por ejemplo: LogisticRegression(C=0.1) (a menor C, mayor regularización).*

*#predict\_proba() devuelve las probabilidades de pertenecer a cada clase (útil para umbrales distintos de 0.5).*

**2.5.1 Evaluación del Modelo (scikit-learn):**



**2.6 Regresión Logística (Logit clásico) con statsmodels**

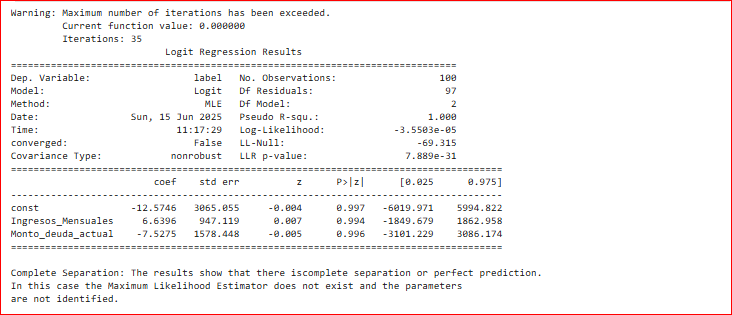


*#¿Qué ventajas ofrece statsmodels?*

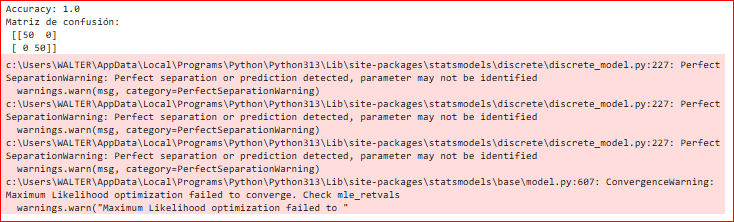
*#Permite analizar la significancia estadística de cada variable (p-value).*

*#Muestra intervalos de confianza y errores estándar.*

*#Es ideal para interpretación, informes técnicos y papers*



**2.5.1 Evaluación del Modelo:**



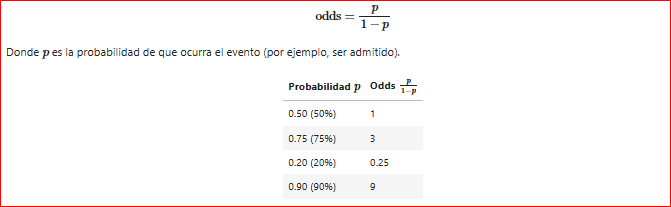
*# Accuracy: 1.0 y matriz de confusión perfecta*

*# El modelo clasificó todos los ejemplos correctamente.*

**2.7 Interpretación de los Coeficientes en Regresión Logística**

**2.7.1 ¿Qué son los odds?**

La razón de probabilidades (odds) se calcula así:



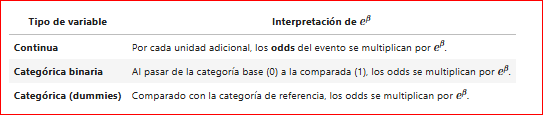
**2.8 Modelo Logístico**

La regresión logística modela el **logit** de la probabilidad como una combinación lineal de variables:



* El lado izquierdo es el **logit** (logaritmo de los odds).
* Cada coeficiente βi representa el cambio en el log-odds.
* Usamos eβi para interpretar directamente sobre los odds.

**2.8.1 Interpretación de (Razón de odds)**



## Ejemplo 1: Variable Continua

Supón que:

β=0.2⇒e0.2≈1.22

**Interpretación:**  
Por cada unidad adicional en la variable (ej. puntaje de examen), los **odds del evento aumentan un 22%**.  
La probabilidad también aumenta, aunque no de forma lineal.

## Ejemplo 2: Variable Categórica Binaria

Supón que:

β=−0.4⇒e−0.4≈0.67

**Interpretación:**  
Pasar de la categoría base a la comparada **reduce los odds del evento en 33%**.  
Esto implica que la **probabilidad del evento disminuye**.

## Notas clave

* β afecta el **logaritmo de los odds**.
* eβ afecta directamente los **odds** (razón de probabilidades).
* La **probabilidad** no cambia de forma lineal respecto a β.

Resumen

“Cada coeficiente

nos dice cómo cambia la razón de que ocurra frente a que no ocurra el evento.

Si tomamos

, eso nos dice cuántas veces más o menos probable es que ocurra el evento cuando cambia esa variable.”

Calculo de la Curva ROC

* 1. **¿Cómo surge la curva ROC?**

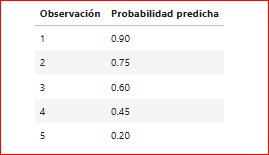
La curva ROC (Receiver Operating Characteristic) muestra gráficamente la capacidad de un modelo de clasificación binaria para distinguir entre clases a medida que varía el umbral de decisión.

Se construye comparando los valores predichos como probabilidades con los valores reales (0 o 1).

**2.9.1 ¿Cómo se construye paso a paso?**

1. El modelo predice probabilidades

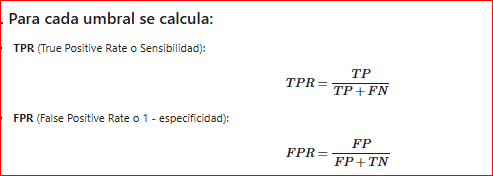
Por ejemplo:



**2.9.2 Se elige un umbral**

Un modelo de clasificación convierte las probabilidades en clases con un **umbral**.

* Umbral típico: **0.5**
* Si cambiamos el umbral, cambian las predicciones y la matriz de confusión.



### 2.9.3 Se repite para muchos umbrales

Por ejemplo: desde 1.0 hasta 0.0, en pasos.

Para cada umbral:

* Se generan predicciones binarias
* Se calcula la matriz de confusión
* Se obtienen TPR y FPR
* Se guarda el punto ( (FPR, TPR) )

### 2.9.4 . Se grafica la curva

* Eje **X**: FPR (Tasa de Falsos Positivos)
* Eje **Y**: TPR (Tasa de Verdaderos Positivos)

Curva ROC={(FPR,TPR) para todos los umbrales}

* Un modelo **perfecto** alcanza el punto (0, 1)
* Un modelo **aleatorio** genera una línea diagonal de (0, 0) a (1, 1)

**2.9.5 ¿Qué representa la curva ROC?**

Muestra el **compromiso entre sensibilidad y especificidad**.

Cuanto más arriba y a la izquierda esté la curva, **mejor es el modelo**.

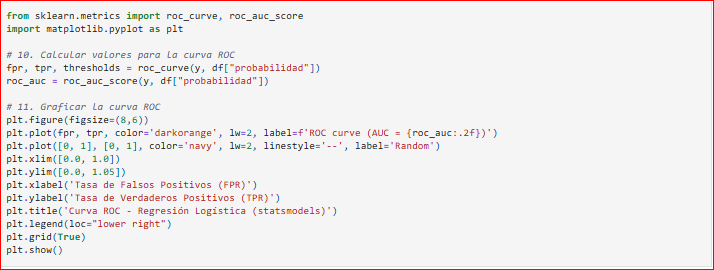
Se puede resumir con el **AUC (Área bajo la curva ROC)**.

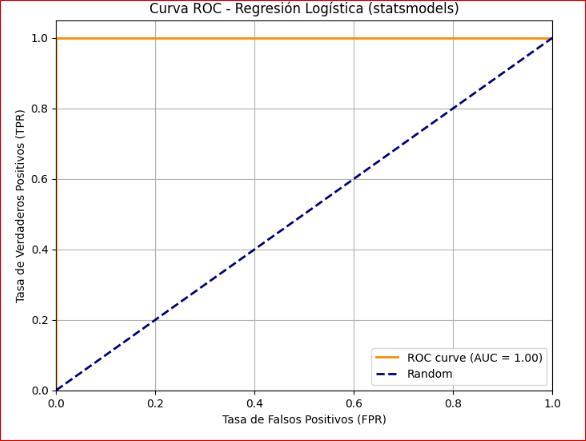


## 2.9.6 Resumen

1. El modelo predice probabilidades.
2. Se prueban muchos umbrales.
3. Para cada uno, se calcula TPR y FPR.
4. Se grafican los puntos ( (FPR, TPR) ).
5. Se analiza el **AUC** para medir la calidad general.

La curva ROC no depende de un solo umbral y es muy útil cuando las clases están desbalanceadas.



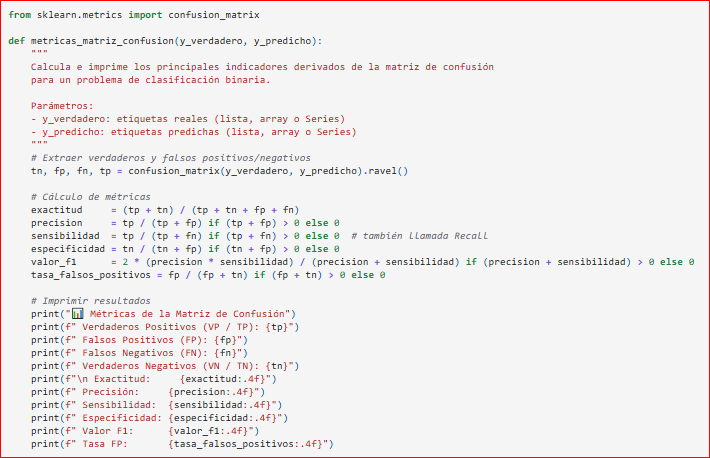


*# - FPR = 0 y TPR = 1: señal de que el modelo no comete errores al clasificar positivos y negativos.*

*# La línea punteada azul representa un clasificador aleatorio para comparación.*

*# El área bajo la curva (AUC) igual a 1 respalda lo que ya veías en la matriz de confusión: precisión total.*

* 1. **Mas Metricas de la matriz de confusion**





Métricas de la Matriz de Confusión

Verdaderos Positivos (VP / TP): 50

Falsos Positivos (FP): 0

Falsos Negativos (FN): 0

Verdaderos Negativos (VN / TN): 50

Exactitud: 1.0000

Precisión: 1.0000

Sensibilidad: 1.0000

Especificidad: 1.0000

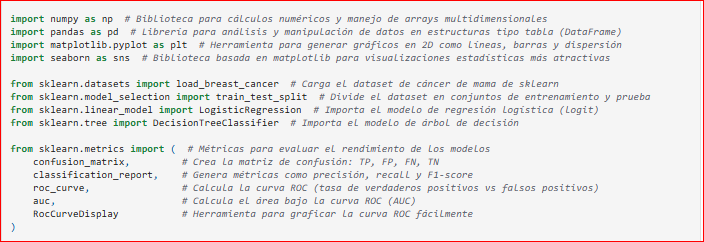
Valor F1: 1.0000

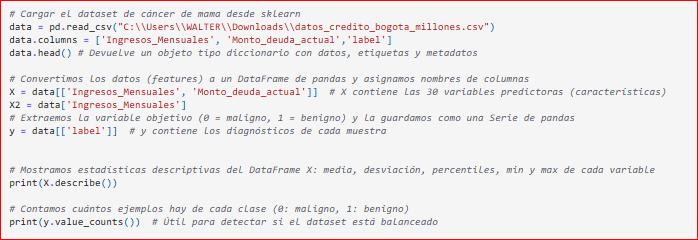
Tasa FP: 0.0000

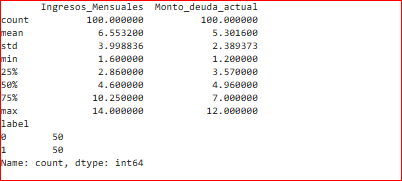
**3. Comparación de Modelos: Regresión Logística vs Árbol de Decisión (Python)**

En esta clase realizaremos una comparación entre dos modelos de clasificación binaria: **Regresión Logística (Logit)** y **Árbol de Decisión**, usando el dataset de créditos

**3.1 Importar Librerías**







**3.2 Dataset de Evaluación Crediticia**

Este conjunto de datos simula evaluaciones de crédito en base a ingresos y deudas mensuales. Es ideal para practicar modelos de clasificación binaria

Información general

Número de muestras: 100

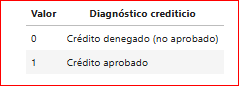
Número de características (variables): 2

Variable objetivo (target): Aprobación del crédito

0: Crédito no aprobado

1: Crédito aprobado

**3.3 Variable Objetivo: target**

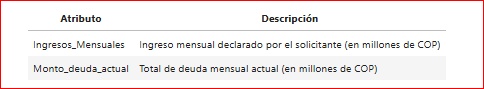


**3.4 Descripción de las Variables Predictoras**

Cada variable se calcula a partir de una imagen digitalizada de una muestra de tejido.  
Para 10 atributos básicos, se generan 3 tipos de métricas:

* **Media (mean)**
* **Error estándar (error)**
* **Peor valor (worst)**

**3.4.1 Atributos Básicos**



**3.4.2 Lista completa de variables**

**Variables tipo mean (promedios)**

* mean radius
* mean texture
* mean perimeter
* mean area
* mean smoothness
* mean compactness
* mean concavity
* mean concave points
* mean symmetry
* mean fractal dimension

**2.4.2 Variables tipo error (error estándar)**

* radius error
* texture error
* perimeter error
* area error
* smoothness error
* compactness error
* concavity error
* concave points error
* symmetry error
* fractal dimension error

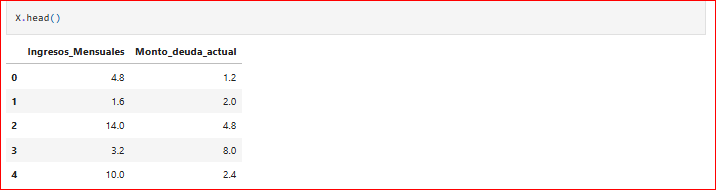
**3.4.3 Variables tipo worst (valores más extremos)**

* worst radius
* worst texture
* worst perimeter
* worst area
* worst smoothness
* worst compactness
* worst concavity
* worst concave points
* worst symmetry
* worst fractal dimension

**3.5 Aplicaciones del dataset**

**Este dataset es ideal para:**

* Modelos de clasificación binaria
* Comparar algoritmos como:
* Regresión Logística
* Árboles de Decisión



**3.6 Particionar los Datos**

La partición de los datos en conjuntos de entrenamiento y prueba es fundamental en machine learning porque permite evaluar de forma objetiva la capacidad del modelo para generalizar a datos nuevos. Si un modelo se entrena y evalúa sobre el mismo conjunto, corre el riesgo de memorizar los datos (sobreajuste) en lugar de aprender patrones reales. Al separar un subconjunto exclusivo para prueba, se simula cómo se comportaría el modelo en un entorno real, garantizando que las métricas obtenidas reflejen su verdadero desempeño. Además, usar una semilla aleatoria asegura que esta división sea reproducible, lo que es esencial para comparar modelos de manera justa y transparente.



**3.7 Entrenar los Modelos**

Por qué es importante usar semillas

Asegura que la partición sea reproducible. Si tú o un estudiante corre el código varias veces, obtendrán el mismo resulatdo Ideal para clases, investigación o pruebas controladas.

*# 🔹 Modelo de Regresión Logística (Logit)*

log\_model **=** LogisticRegression(max\_iter**=**10000)

*# Creamos el modelo logístico. Se establece max\_iter=10000 para asegurar que el algoritmo converge,*

*# especialmente cuando hay muchas variables o los datos requieren más iteraciones.*

log\_model**.**fit(X\_train, y\_train) *# Entrenamos el modelo usando los datos de entrenamiento. Aquí el modelo "aprende" la relación entre las variables X e y.*

y\_pred\_log **=** log\_model**.**predict(X\_test) *# Generamos predicciones de clase (0 o 1) sobre el conjunto de prueba. Esto es lo que el modelo cree que es la clase verdadera.*

y\_proba\_log **=** log\_model**.**predict\_proba(X\_test)[:, 1] *# Obtenemos las probabilidades predichas de que la clase sea "1" (benigno).*

*# Esto se usa para calcular curvas ROC, métricas de umbral, etc.*

*# 🔸 Modelo de Árbol de Decisión*

tree\_model **=** DecisionTreeClassifier(random\_state**=**42) *# Creamos el árbol de decisión. Usamos random\_state para que los resultados sean reproducibles.*

tree\_model**.**fit(X\_train, y\_train) *# Entrenamos el árbol con los datos de entrenamiento. El árbol genera reglas basadas en divisiones de los datos.*

y\_pred\_tree **=** tree\_model**.**predict(X\_test)

*# Realizamos predicciones de clase (0 o 1) sobre los datos de prueba usando el árbol entrenado.*

y\_proba\_tree **=** tree\_model**.**predict\_proba(X\_test)[:, 1]

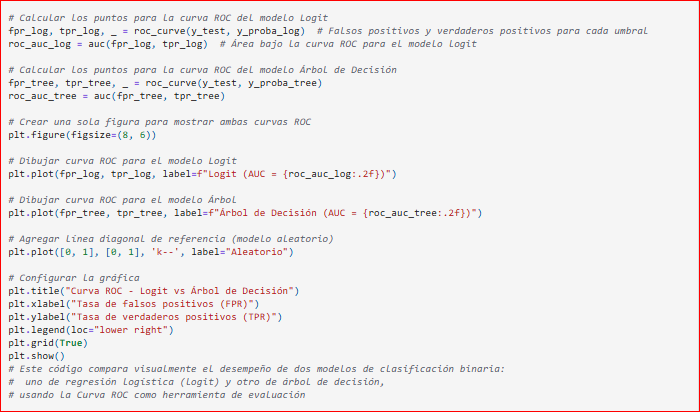
*# Extraemos la probabilidad predicha de que la clase sea "1".*

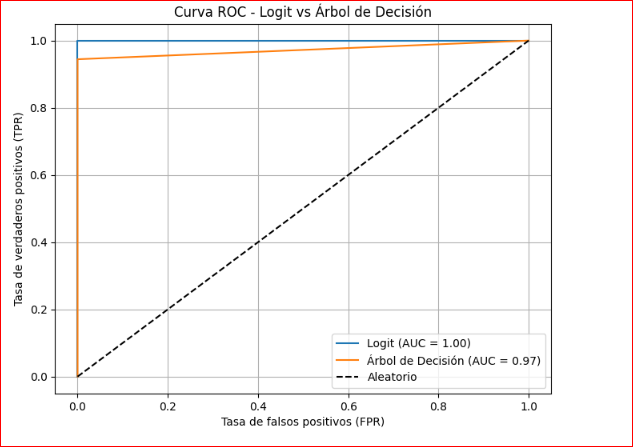
*# Igual que en el logit, esto sirve para métricas como AUC o análisis de umbrales de decisión.*

c:\Users\WALTER\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\utils\validation.py:1406: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

**3.4 Curvas ROC y AUC**





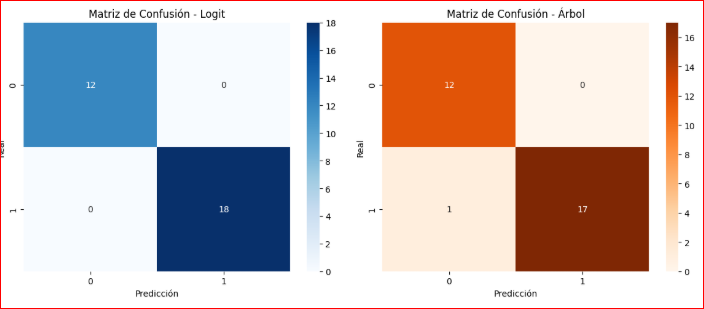
*# La curva ROC muestra claramente que ambos modelos —el logit y el árbol de decisión— tienen un desempeño sobresaliente:*

*# Logit (AUC = 1.00): separación perfecta entre clases. El modelo es capaz de distinguir sin errores entre créditos aprobados y no aprobados, sin importar el umbral.*

*# Árbol de Decisión (AUC = 0.97): también excelente. Tiene una ligera curva más alejada del ángulo superior izquierdo en comparación al logit, pero sigue siendo un modelo altamente preciso*

**3.5 Matriz de Confusión y Métricas**





*# Matriz de Confusión – Logit*

*#TP = 18, TN = 12, FP = 0, FN = 0*

*# El modelo de regresión logística clasificó todos los casos correctamente. No cometió errores, ni rechazó a clientes aprobables ni aprobó a clientes no elegibles.*

*# Esto confirma lo que ya habías visto con el AUC de 1.00: separación perfecta entre clases en el conjunto de prueba.*

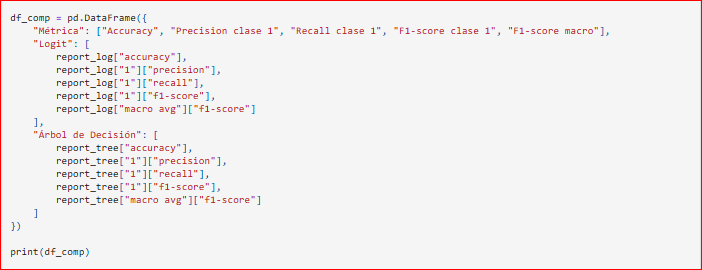
*# Matriz de Confusión – Árbol de Decisión*

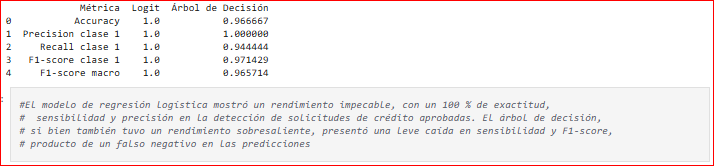
*# TP = 17, TN = 12, FP = 0, FN = 1*

*# El árbol cometió un único error: rechazó por error a un cliente que sí debía ser aprobado.*

*# Aunque su rendimiento sigue siendo muy alto, esto se ve reflejado en el AUC ligeramente menor (0.97).*

**3.6 Comparación de Métricas en Tabla**





***#El modelo de regresión logística mostró un rendimiento impecable, con un 100 % de exactitud***

***# sensibilidad y precisión en la detección de solicitudes de crédito aprobadas. El árbol de decisión,***

***# si bien también tuvo un rendimiento sobresaliente, presentó una leve caída en sensibilidad y F1-score,***

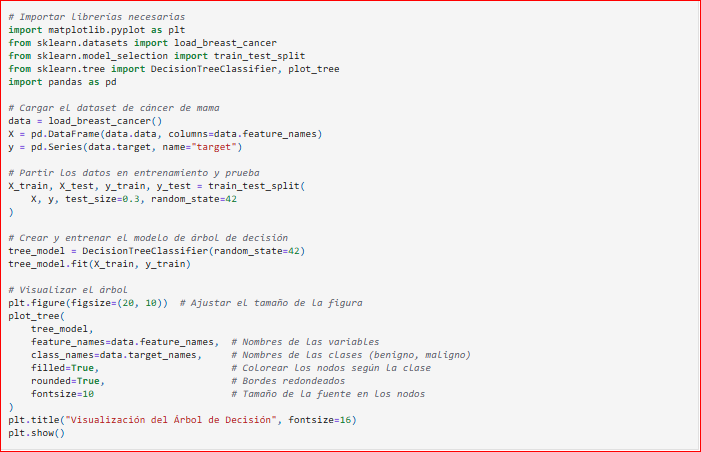
***# producto de un falso negativo en las predicciones***

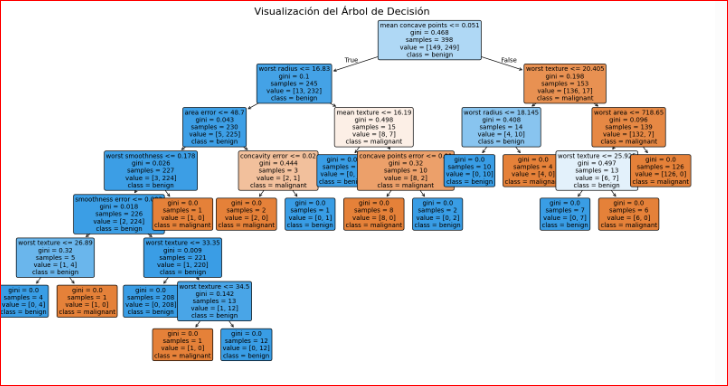
**3.7 Conclusión**

El modelo logit es más estable y con menor tasa de falsos negativos.

El árbol de decisión puede capturar relaciones no lineales pero es más propenso al sobreajuste.

Ambos modelos son útiles dependiendo del contexto.





## Un árbol de decisión trabaja dividiendo los datos en ramas basadas en preguntas binarias sobre las variables predictoras, con el objetivo de separar las clases de la forma más pura posible en cada nodo. Comienza en un nodo raíz y, en cada paso, elige la variable y el umbral que mejor separan las clases según una métrica de impureza (como Gini o entropía). A medida que se crean divisiones, se forman nodos internos que representan decisiones y hojas finales que contienen la predicción de clase. Este proceso continúa hasta que se cumplen ciertos criterios de parada

References

Insert your alphabetized list of references here. Be sure to use a hanging indent to differentiate between citations.