

# Sesión 3: Regresiones y métricas de Desempeño de Modelos Regresivos

En Machine Learning, los modelos de regresión se utilizan para predecir variables numéricas continuas a partir de una o más variables independientes. Son ampliamente utilizados en contextos reales como estimar precios, predecir demanda, calcular ingresos, entre otros.

**R** por Kibernetum Capacitación S.A. Kibernetum Capacitación S.A.

# ¿Qué es un modelo de regresión?

En Machine Learning, los modelos de regresión se utilizan para predecir variables numéricas continuas a partir de una o más variables independientes. Son ampliamente utilizados en contextos reales como estimar precios, predecir demanda, calcular ingresos, entre otros.

## Ejemplos comunes:

- Predecir el precio de una vivienda según su tamaño.
- Estimar el nivel de consumo energético de un edificio.
- Calcular la cantidad de productos que se venderán el próximo mes.

Esta imagen ilustra la fórmula básica de la regresión lineal:  $y = ax + b$ , donde:

- $y$  es la variable dependiente (lo que queremos predecir).
- $x$  es la variable independiente (la que usamos para hacer la predicción).
- $a$  representa la pendiente (cómo cambia  $y$  en función de  $x$ ).
- $b$  es la intersección o término independiente (valor de  $y$  cuando  $x$  es cero).

# ¿Cuándo usamos regresión?

Cuando la respuesta esperada es un número: precio, cantidad, temperatura, consumo, ingresos, etc.

## Ejemplos comunes:

- Predecir el precio de una vivienda según su tamaño.
- Estimar el nivel de consumo eléctrico de un edificio según el clima.
- Calcular la cantidad de productos vendidos en función del mes o la inversión en publicidad.

### Analicemos este ejemplo:

Problema: Queremos estimar cuánto cobra un plomero por su trabajo en función de las horas trabajadas.

Si observamos varios casos reales, podríamos obtener esta relación:

$$y = 25x + 10$$

### Donde:

- $y$  = precio total cobrado (en dólares) → variable dependiente
- $x$  = cantidad de horas trabajadas → variable independiente
- 25 = valor por hora (pendiente)
- 10 = tarifa base por desplazamiento o visita (intersección)

### Entonces:

Si el plomero trabaja 2 horas:

$$y = 20.000(2) + 5.000 = 45.000 \text{ CLP}$$

Si trabaja 5 horas:

$$y = 20.000(5) + 5.000 = 105.000 \text{ CLP}$$

Así, podemos predecir fácilmente el precio con base en las horas, gracias a este modelo de regresión lineal.

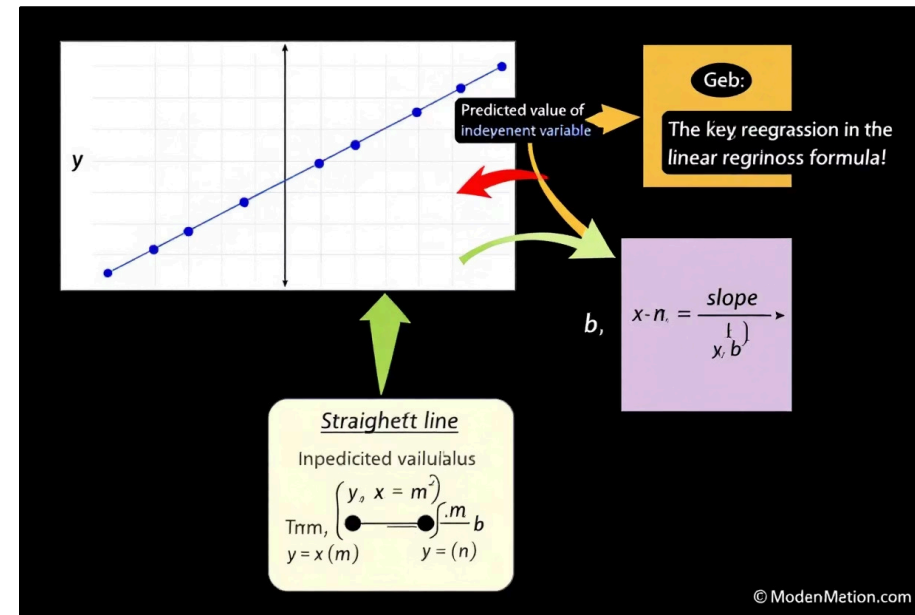
# Componentes de la Regresión Lineal

Veamos al detalle la siguiente imagen:

## La fórmula básica:

$$y = ax + b$$

- y: lo que queremos predecir (precio)
- x: lo que sabemos (horas trabajadas)
- a: lo que aporta cada unidad de x (valor por hora)
- b: lo que se cobra siempre (tarifa fija)



# Regresión lineal

La regresión lineal es el tipo más simple y clásico de regresión. Intenta establecer una relación lineal entre una variable independiente XXX y una variable dependiente YYY, ajustando una línea recta que represente la tendencia de los datos.

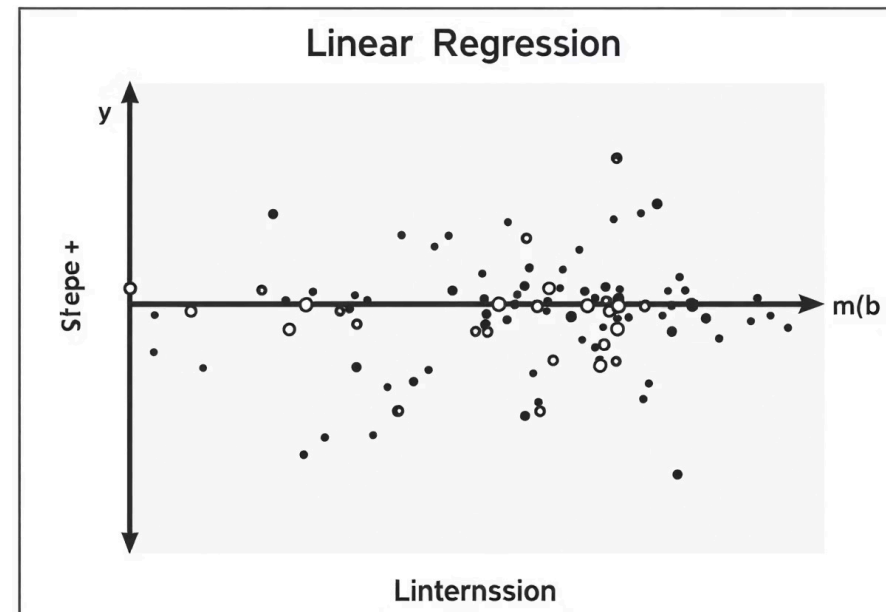
## Fórmula general:

The equation of a line:

$$Y_0 = + x_{11}X_{11} + \dots + x_{1n}X_n$$

..... + x\_{1n}X\_n

## Donde:



# Regresión no lineal

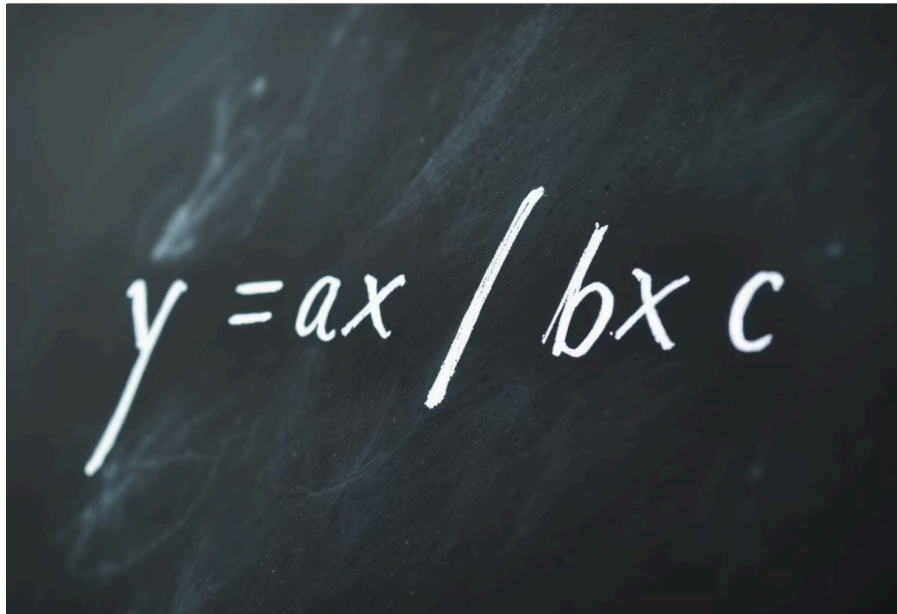
En muchos casos, la relación entre variables no puede representarse con una línea recta. Por ejemplo, el precio de un producto en función de la demanda, el crecimiento de una población o la velocidad de un auto con respecto al tiempo no siguen una tendencia lineal.

Para estos casos se utilizan modelos de regresión no lineales, que permiten capturar relaciones más complejas entre las variables. Pueden adoptar formas:

- Cuadráticas (con potencias),
- Exponenciales (crecimientos acelerados),
- Logarítmicas (crecimientos que se frenan),
- Entre otras.

## ¿CÓMO SE VE UNA REGRESIÓN NO LINEAL?

Una regresión cuadrática tiene esta forma:



A photograph of a chalkboard with the equation  $y = ax^2 + bx + c$  written in white chalk. The equation is written in a slightly cursive, handwritten style.

Donde:

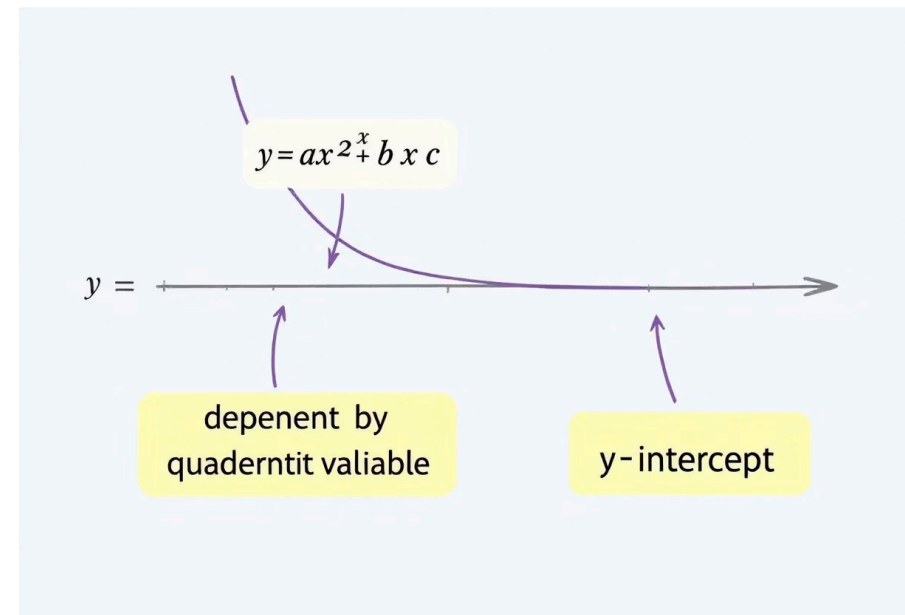


Gráfico de dispersión con una curva suave (en forma de U o de campana) ajustada sobre los datos lo que nos confirma que cuando la relación entre las variables tiene forma de parábola o curva, una regresión lineal no es suficiente y se necesita un modelo más flexible.

# Ejemplo de regresión no lineal: Modelo de velocidad

Puedes descargar el siguiente ejercicio desde el siguiente enlace: [regresion no lineal modelo de velocidad.ipynb](#)

Queremos estimar la velocidad de un auto en función del tiempo transcurrido, sabiendo que acelera al inicio, pero luego comienza a desacelerar.

Esto puede parecer algo complejo al principio...

pero con paciencia y trabajo todo se ira despejando.

Lo importante es que comprendas cómo funciona el comportamiento de los datos y cómo una fórmula matemática puede ayudarte a representarlo.

## MODELO PROPUESTO

BASÁNDONOS EN OBSERVACIONES REALES, PODEMOS MODELAR ESTA RELACIÓN ASÍ:

## ¿QUÉ NOS ESTÁ DICIENDO ESTA ECUACIÓN?

- $v$ : es la velocidad del auto en kilómetros por hora (km/h)  $\rightarrow$  es lo que queremos predecir.
- $t$ : es el tiempo transcurrido en horas (h)  $\rightarrow$  es lo que conocemos.
- El coeficiente  $-0,5$  delante de  $t^2$  nos indica que a medida que el tiempo avanza, la velocidad dejará de crecer y comenzará a disminuir. Esto es típico en situaciones donde hay resistencia o fatiga (como pendiente o fricción).
- El coeficiente  $+6$  delante de  $t$  representa que al comienzo hay una aceleración fuerte: por cada hora adicional, el auto inicialmente gana velocidad.

# Visualización del modelo de velocidad

¡Mejor veámoslo en un gráfico!

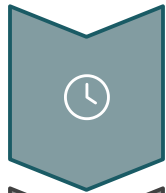
El gráfico muestra cómo la velocidad aumenta al inicio, alcanza un punto máximo alrededor de las 6 horas, y luego comienza a disminuir. Los puntos rojos destacan tres momentos clave: 1 h, 5 h y 10 h, con sus respectivas velocidades anotadas.

## EN RESUMEN:

El auto acelera al principio, alcanza una velocidad máxima y luego desacelera.

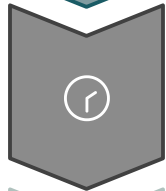
## CÁLCULO CONCRETO

Vamos a probar el modelo con valores reales de tiempo.



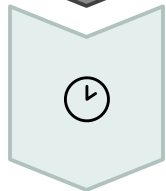
**SI HAN PASADO 1 HORA:**

$$v = -0,5(1)^2 + 6(1) = -0,5 + 6 = 5,5 \text{ km/h}$$



**SI HAN PASADO 5 HORAS:**

$$v = -0,5(5)^2 + 6(5) = -12,5 + 30 = 17,5 \text{ km/h}$$



**SI HAN PASADO 10 HORAS:**

$$v = -0,5(10)^2 + 6(10) = -50 + 60 = 10 \text{ km/h}$$



# Interpretación de los resultados

## ¿QUÉ NOS DICEN ESTOS RESULTADOS?

El modelo nos indica que:

- A la hora 1, el auto aún está en fase de aceleración.
- A la hora 5, alcanza su velocidad máxima.
- A la hora 10, ya está en desaceleración.

## ¿EN QUÉ SE DIFERENCIA DE UNA REGRESIÓN LINEAL?

En este caso, la relación entre tiempo y velocidad no sigue una línea recta. Si usáramos una regresión lineal para este problema, subestimaríamos o sobreestimaríamos muchas predicciones, especialmente en los extremos.

Este comportamiento es típico en la física del movimiento: no siempre más tiempo significa más velocidad, porque hay otros factores en juego (como pendientes, fatiga del motor, viento, etc.).

Esta imagen resume las principales formas en que las variables pueden relacionarse entre sí. Cuando la tendencia es recta, hablamos de una regresión lineal; cuando sigue una curva, entramos al mundo no lineal.

Pero también puede pasar que no haya relación... y eso también es un resultado válido y valioso.

# Implementación con Scikit-learn

Ya conocimos qué es un modelo de regresión y cómo puede ser lineal o no lineal. Ahora daremos un paso más: entrenar un modelo real utilizando la biblioteca Scikit-learn, una de las herramientas más utilizadas en la ciencia de datos con Python.

En esta sección veremos cómo:

- Cargar datos reales,
- Entrenar un modelo de regresión lineal,
- Hacer predicciones,
- Visualizar resultados,
- Y reflexionar sobre su interpretación.

Puedes descargar EL Siguiente ejercicio desde el siguiente enlace:

`entrenamiento_del_modelo.ipynb`

# Entrenamiento del modelo

Usaremos un conjunto de datos muy común: precios de casas según sus metros cuadrados.

Veamos cómo entrenar un modelo de regresión lineal con estos datos:

```
# Importar bibliotecas necesarias
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Cargar datos (simulados para este ejemplo)
np.random.seed(42)
X = 2 * np.random.rand(100, 1) # Metros cuadrados (normalizado)
y = 4 + 3 * X + np.random.randn(100, 1) # Precio = base + (precio/m²) + ruido

# Dividir en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear y entrenar el modelo
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Mostrar coeficientes aprendidos
print(f"Intercepto (precio base): {modelo.intercept_[0]:.2f}")
print(f"Coeficiente (precio por m²): {modelo.coef_[0][0]:.2f}")
```

Este modelo ha aprendido a predecir el precio de una casa en base a su tamaño.

# Predicción con el modelo

Una vez que el modelo ha sido entrenado, ya está listo para hacer lo más importante: predecir precios de viviendas a partir de su tamaño, incluso si nunca ha visto esos datos antes.

En esta etapa, usaremos el conjunto de prueba ( $X_{\text{test}}$ ) que dejamos apartado intencionalmente. ¿Por qué?

Porque queremos ver qué tan bien generaliza el modelo a datos nuevos, no usados durante el entrenamiento. Esta es una parte fundamental del aprendizaje automático.

```
# Hacer predicciones con el conjunto de prueba
y_pred = modelo.predict(X_test)

# Mostrar algunas predicciones vs valores reales
print("\nComparación de algunas predicciones:")
print("Metros² | Precio Real | Precio Predicho | Error")
print("-" * 50)

for i in range(5): # Mostrar solo 5 ejemplos
    m2 = X_test[i][0]
    real = y_test[i][0]
    pred = y_pred[i][0]
    error = real - pred

    print(f"{m2*100:6.1f} | ${real*100000:10.2f} | ${pred*100000:14.2f} | ${error*100000:8.2f}")

# Calcular métricas de evaluación (veremos esto en detalle más adelante)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\nMétricas de evaluación:")
print(f"MAE: ${mae*100000:.2f}")
print(f"MSE: ${mse*100000**2:.2f}")
print(f"RMSE: ${rmse*100000:.2f}")
print(f"R²: {r2:.4f}")
```

# Visualización del modelo

Ahora que hemos entrenado y probado el modelo, es hora de ver gráficamente cómo se ajusta a los datos reales.

Visualizar los resultados te permite:

- Entender si el modelo está siguiendo bien la tendencia de los datos.
- Detectar posibles desviaciones o errores sistemáticos.
- Comunicar resultados de forma clara a otros equipos.

En este gráfico veremos:

- Puntos azules: los datos reales de prueba (precio real de las casas).
- Línea roja: la predicción que realiza el modelo lineal.

Si la línea roja atraviesa la nube de puntos, significa que el modelo ha aprendido correctamente la relación entre tamaño y precio.

```
# Visualizar el modelo y las predicciones
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Datos reales')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicciones')
plt.xlabel('Metros cuadrados (normalizado)')
plt.ylabel('Precio (normalizado)')
plt.title('Regresión Lineal: Precio vs Metros Cuadrados')
plt.legend()
plt.grid(True)
plt.show()
```

# Métricas de evaluación

Ya entrenamos y visualizamos el modelo. Ahora es momento de cuantificar su rendimiento con métricas específicas para regresión.

Estas métricas te ayudarán a responder:

- ¿Qué tan buenos son los resultados del modelo?
- ¿Cuánto se equivocó, en promedio?
- ¿Es mejor que una predicción aleatoria?

A continuación, evaluaremos el modelo con 4 métricas clave:



## MAE Mean Absolute Error (Error absoluto medio)

Promedia la magnitud de los errores, sin importar si fueron positivos o negativos.

Penaliza todos los errores por igual.

Útil cuando queremos interpretabilidad directa en las mismas unidades que la predicción.



## MSE Mean Squared Error (Error cuadrático medio)

Promedia los errores al cuadrado. Penaliza más los errores grandes.

Útil cuando los errores grandes deben evitarse especialmente.

No está en las mismas unidades que la variable objetivo.



## RMSE –Root Mean Squared Error (Raíz del MSE)

Es simplemente la raíz cuadrada del MSE.

Se interpreta en las mismas unidades que el precio → más comprensible.



## $R^2$ Coeficiente de determinación

Mide cuánta varianza de los datos explica el modelo.

$R^2 = 1$  → Ajuste perfecto

$R^2 = 0$  → No explica nada

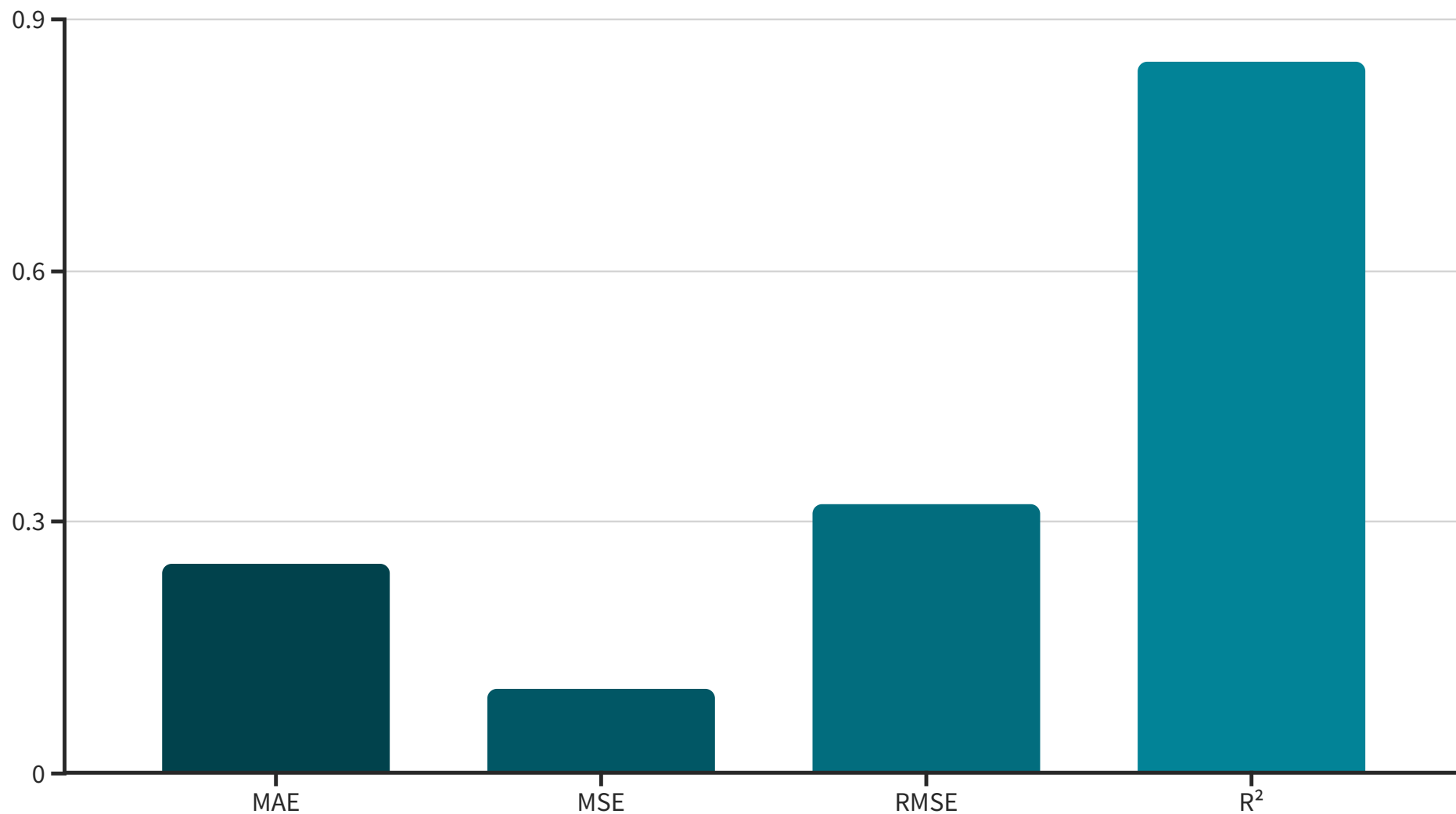
$R^2 < 0$  → Peor que una línea horizontal constante

# Cálculo de métricas en Python

Veamos cómo calcular estas métricas utilizando Scikit-learn:

```
# Calcular métricas de evaluación
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\nMétricas de evaluación:")
print(f"MAE: ${mae*100000:.2f}")
print(f"MSE: ${mse*100000**2:.2f}")
print(f"RMSE: ${rmse*100000:.2f}")
print(f"R²: {r2:.4f}")
```



Estas métricas nos permiten evaluar objetivamente el rendimiento de nuestro modelo de regresión y compararlo con otros modelos alternativos. Un buen modelo tendrá valores bajos de MAE, MSE y RMSE, y un valor de  $R^2$  cercano a 1.