



# Regresiones y Métricas de Desempeño de Modelos Regresivos

Bienvenidos a esta presentación sobre modelos de regresión en Machine Learning. Exploraremos qué son los modelos regresivos, sus tipos, implementación práctica y cómo evaluar su rendimiento mediante métricas específicas.

Los modelos de regresión son fundamentales para predecir variables numéricas continuas a partir de variables independientes, siendo ampliamente utilizados en contextos reales como estimación de precios, predicción de demanda y cálculo de ingresos.

 **por Kibernetum Capacitación S.A.**

# Preguntas de Activación de Contenido

## Influencia del Preprocesamiento

En la sesión anterior revisamos el preprocesamiento de datos, como la codificación de variables y el escalado. ¿Cómo crees que estos pasos podrían influir en la forma en que un modelo se ajusta o se sobreajusta?

## Recomendaciones "Perfectas"

¿Has tenido la sensación de que una aplicación o recomendación es "demasiado perfecta", como si supiera exactamente lo que quieres? ¿Podría eso relacionarse con el sobreajuste?

## Limitaciones de la Evaluación

¿Por qué crees que no es suficiente entrenar un modelo y evaluar su precisión solo con los datos usados en el entrenamiento?

# ¿Qué es un modelo de regresión?

## Regresión Lineal

$$y = ax + b$$

Variable  
dependiente

Pendiente

Variable  
independiente

Intersección

### Definición

Técnica de Machine Learning utilizada para predecir variables numéricas continuas a partir de una o más variables independientes.

### Aplicaciones

Predicción de precios de viviendas, estimación de consumo energético, cálculo de ventas futuras, entre otros casos de uso prácticos.

### Cuándo usarlo

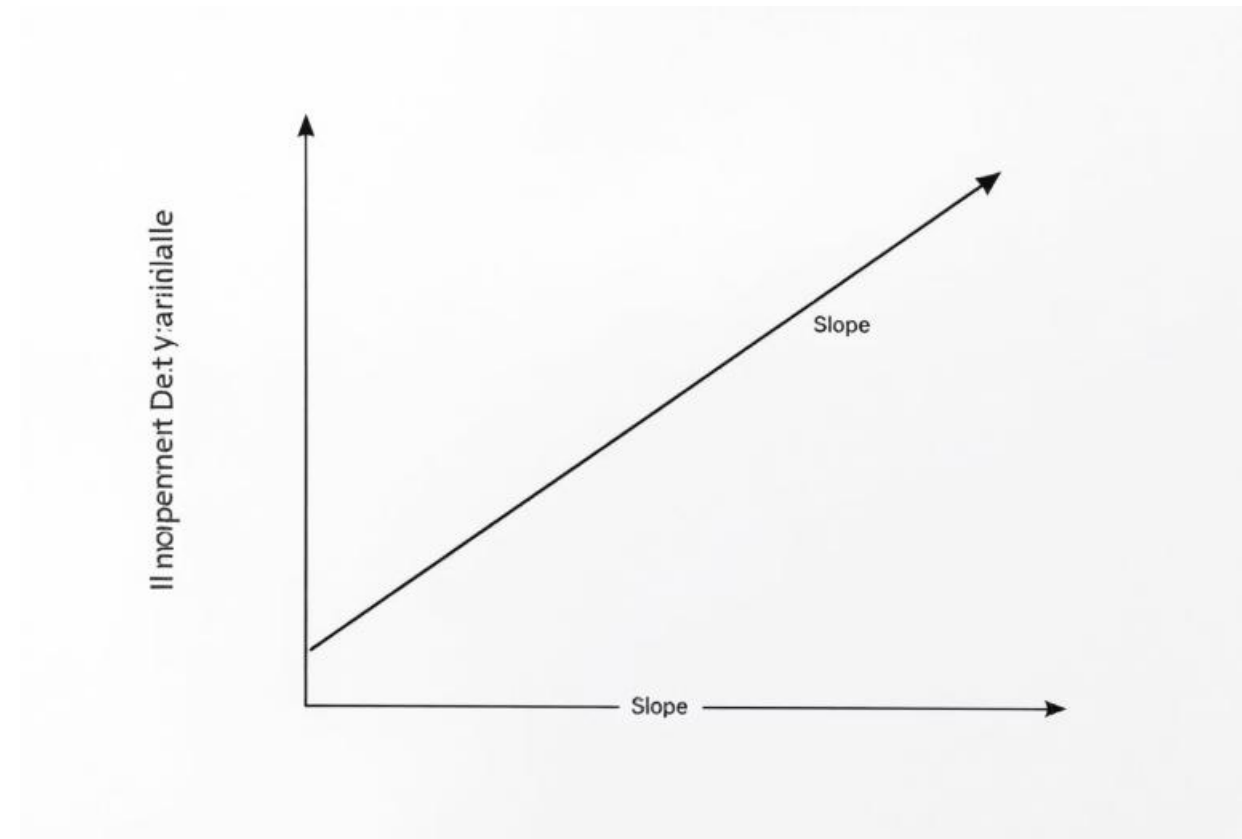
Cuando la respuesta esperada es un número: precio, cantidad, temperatura, consumo, ingresos, etc.

# Componentes de la Regresión Lineal

## Fórmula Básica

$$y = ax + b$$

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



La regresión lineal establece una relación directa entre variables mediante una línea recta, permitiendo hacer predicciones basadas en datos históricos.

# Ejemplo Práctico: El Plomero



## Planteamiento del Problema

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



## Modelo Matemático

$$y = 20.000x + 5.000$$

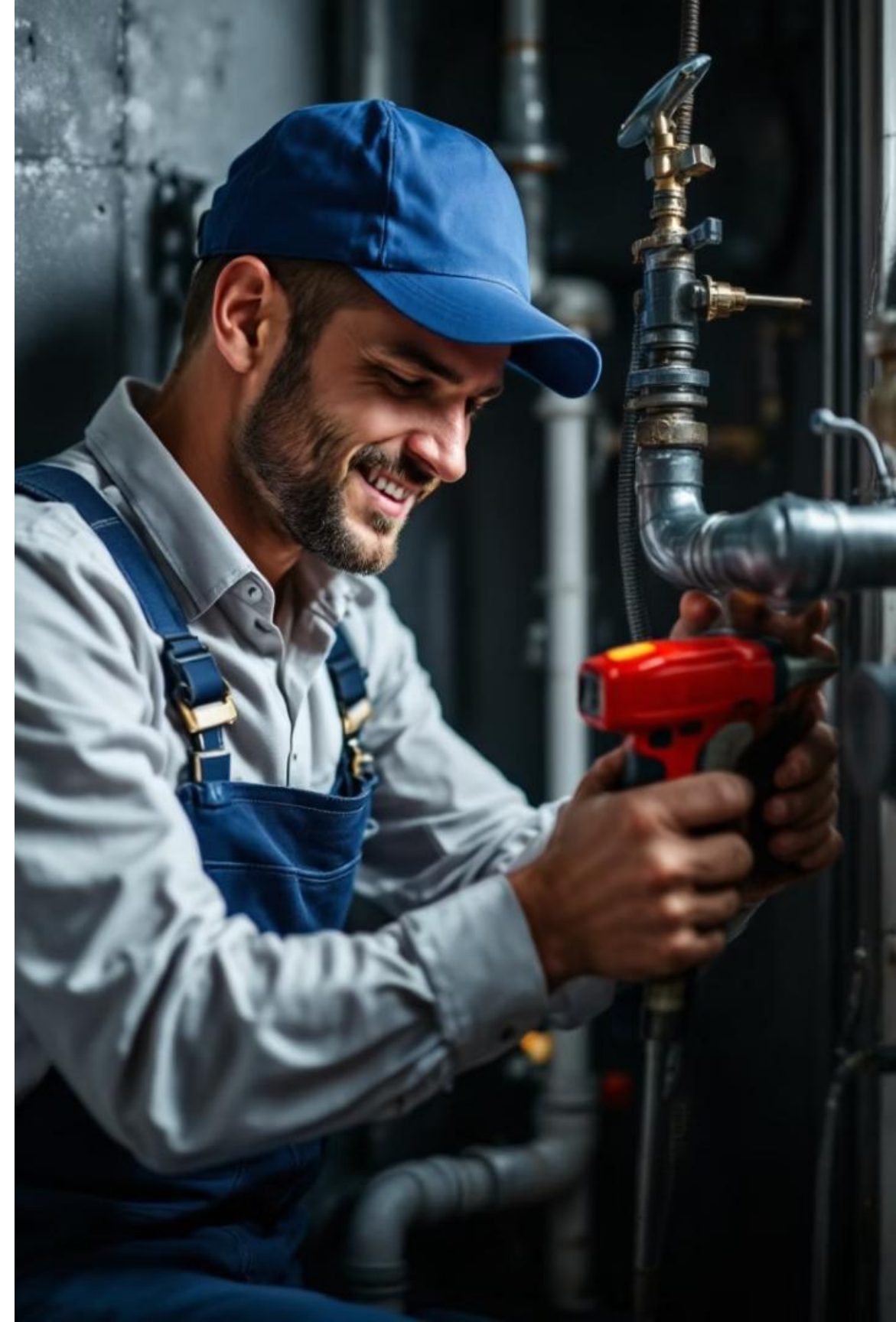
Donde  $y$  = precio total (CLP),  $x$  = horas trabajadas, 20.000 = valor por hora, 5.000 = tarifa base



## Predicciones

Si trabaja 2 horas:  $y = 20.000(2) + 5.000 = 45.000$  CLP

Si trabaja 5 horas:  $y = 20.000(5) + 5.000 = 105.000$  CLP





# Regresión Lineal en Detalle

1

## Definición

Tipo más simple y clásico de regresión que establece una relación lineal entre variables mediante una línea recta.

2

## Fórmula General

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \varepsilon$$

3

## Variables

Y: variable dependiente, X: variables independientes,  $\beta$ : coeficientes,  $\varepsilon$ : error aleatorio

4

## Aplicación

Ideal para relaciones que pueden representarse mediante una línea recta con tendencia constante.

$$y_0 + 70 +$$

$$y_0 + 212 \times 21 \dots = 23n \times n + 2n$$

# Regresión No Lineal

## Definición

Modelos que capturan relaciones complejas entre variables que no pueden representarse con una línea recta.

## Tipos

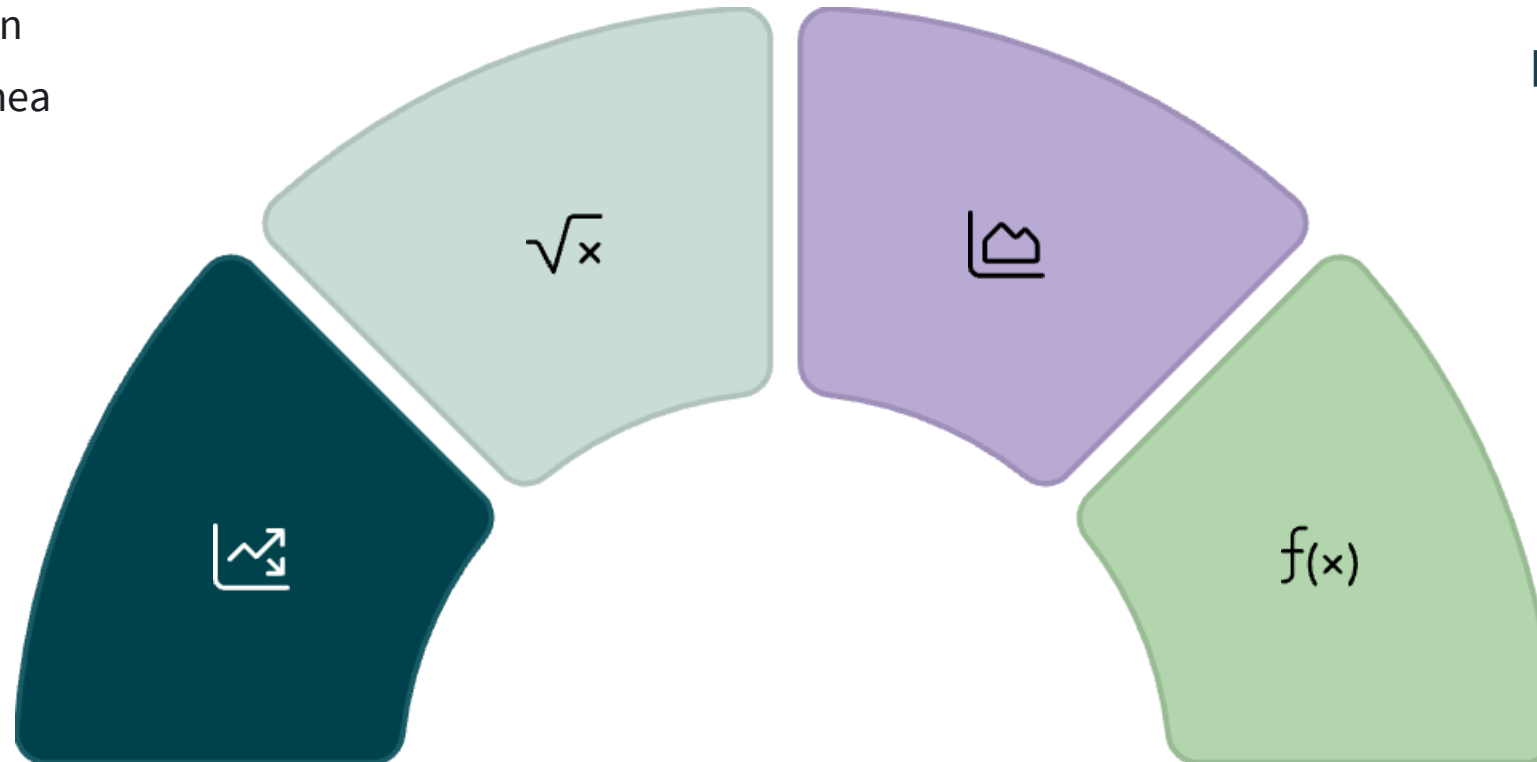
Cuadráticas (con potencias), exponenciales (crecimientos acelerados), logarítmicas (crecimientos que se frenan), entre otras.

## Aplicaciones

Precio en función de la demanda, crecimiento poblacional, velocidad respecto al tiempo, fenómenos naturales.

## Fórmula Cuadrática

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$$



# Ejemplo: Modelo de Velocidad



## Problema

Estimar la velocidad de un auto en función del tiempo, sabiendo que acelera al inicio y luego desacelera.

$\lambda$

## Modelo Propuesto

$$v = -0,5t^2 + 6t$$



## Comportamiento

El auto acelera al principio, alcanza velocidad máxima y luego desacelera debido a factores como resistencia o pendiente.



# Cálculos con el Modelo de Velocidad

# 5.5

## Velocidad a 1 hora

$$v = -0,5(1)^2 + 6(1) = -0,5 + 6 = 5,5$$

km/h

## Velocidad a 5 horas

$$v = -0,5(5)^2 + 6(5) = -12,5 + 30 = 17,5$$

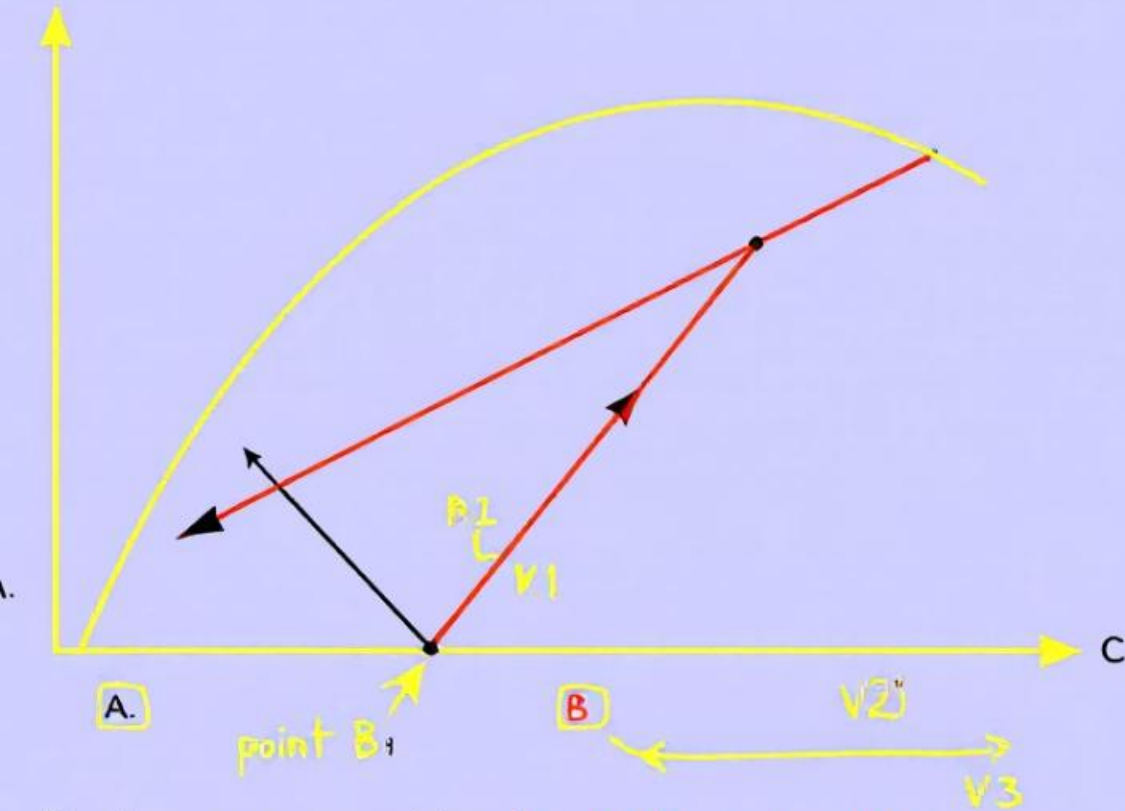
km/h

10

## Velocidad a 10 horas

$$v = -0,5(10)^2 + 6(10) = -50 + 60 = 10$$

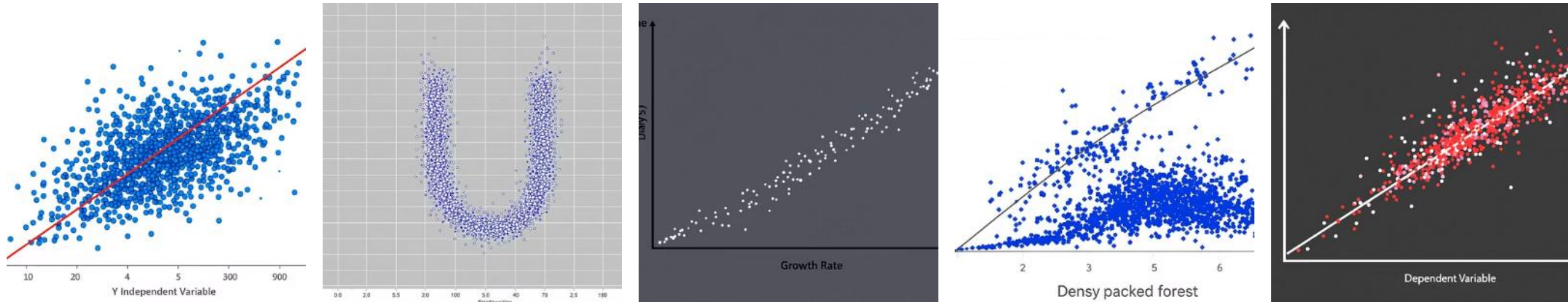
km/h



Parabolic curve rant for. (r a B, the vertex V)

How the **pod** **thine** in the how  
the velocity changes, at **each** **point** one  
how the east changes at each **point** **road**.  
with is the **vector** of at **when** the  
**works** of the of a velocity vector  $V_3$ .

# Patrones en los Datos



Antes de aplicar un modelo, es fundamental observar la distribución de los datos. Cuando la tendencia es recta, aplicamos regresión lineal; cuando sigue una curva, necesitamos modelos no lineales. También puede ocurrir que no exista relación entre las variables, lo cual es un resultado igualmente válido y valioso para el análisis.

# Implementación con Scikit-learn

En esta sección veremos cómo:

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

## 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:

```
✓ 15 s ▶ # Paso 1: Subir el archivo CSV desde tu computador
from google.colab import files
uploaded = files.upload() # Se abrirá un cuadro para seleccionar el archivo local (ej. precios_casas.csv)

# Paso 2: Importar librerías necesarias
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd

# Paso 3: Leer el archivo subido
df = pd.read_csv("precios_casas.csv") # Asegúrate de que el archivo tenga este nombre exacto

# Paso 4: Preparar variables
X = df[['metros_cuadrados']] # Variable independiente
y = df['precio']             # Variable dependiente

# Paso 5: Separar datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Paso 6: Crear y entrenar el modelo
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Paso 7: Confirmación
print("Modelo entrenado correctamente con los datos cargados.")
```

Elegir archivos precios\_casas.csv

- **precios\_casas.csv**(text/csv) - 164 bytes, last modified: 13/4/2025 - 100% done

Saving precios\_casas.csv to precios\_casas (1).csv

Modelo entrenado correctamente con los datos cargados.

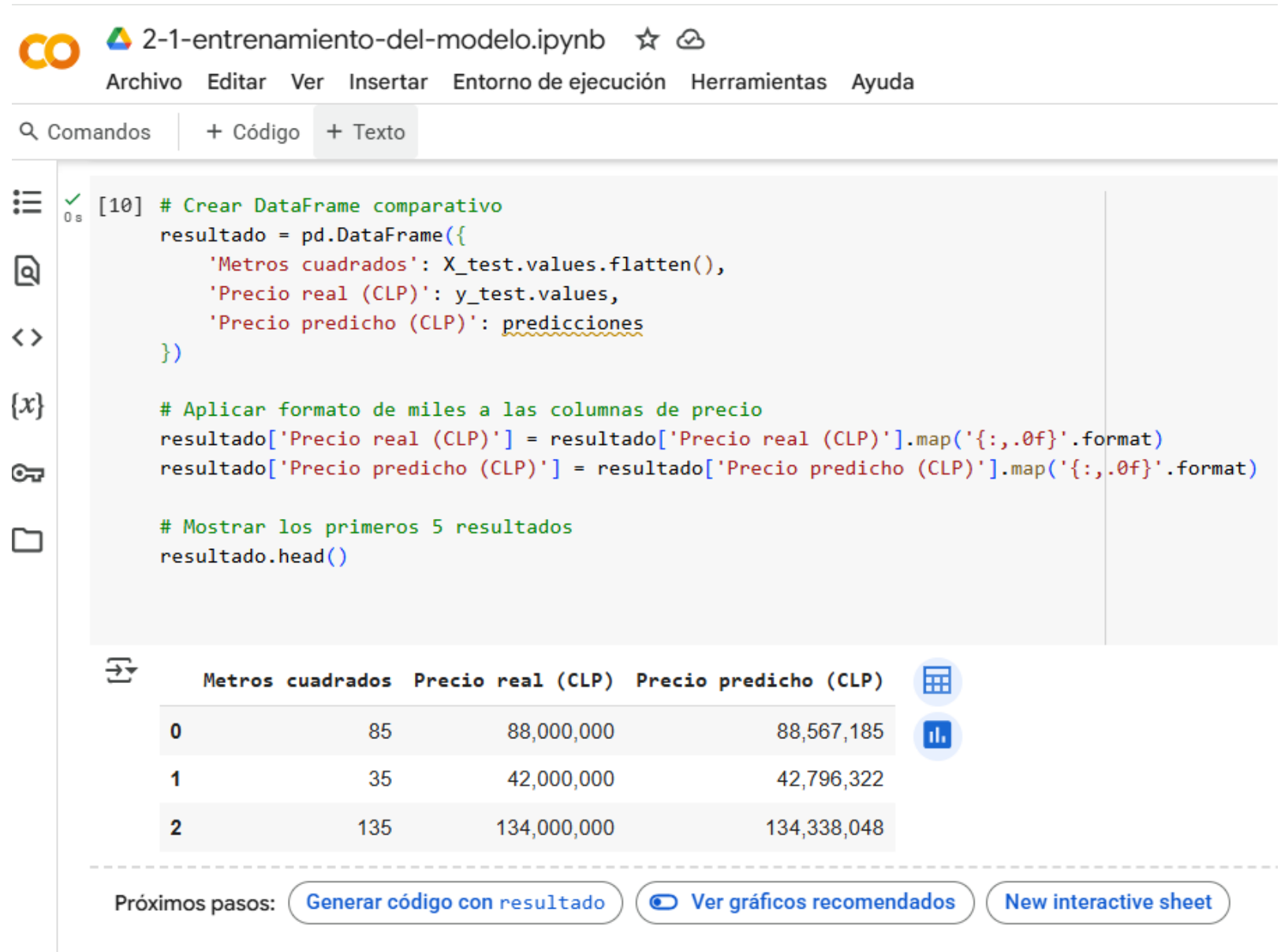
# Implementación con Scikit-learn

## 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.



The screenshot shows a Jupyter Notebook titled "2-1-entrenamiento-del-modelo.ipynb". The code cell [10] contains the following Python code:

```
[10] # Crear DataFrame comparativo
resultado = pd.DataFrame({
    'Metros cuadrados': X_test.values.flatten(),
    'Precio real (CLP)': y_test.values,
    'Precio predicho (CLP)': predicciones
})

# Aplicar formato de miles a las columnas de precio
resultado['Precio real (CLP)'] = resultado['Precio real (CLP)'].map('{:,.0f}'.format)
resultado['Precio predicho (CLP)'] = resultado['Precio predicho (CLP)'].map('{:,.0f}'.format)

# Mostrar los primeros 5 resultados
resultado.head()
```

The output of the code is a DataFrame with 4 columns: "Metros cuadrados", "Precio real (CLP)", and "Precio predicho (CLP)". The first 3 rows are displayed:

	Metros cuadrados	Precio real (CLP)	Precio predicho (CLP)
0	85	88,000,000	88,567,185
1	35	42,000,000	42,796,322
2	135	134,000,000	134,338,048

At the bottom of the notebook, there are three buttons: "Generar código con resultado", "Ver gráficos recomendados", and "New interactive sheet".

# Implementación con Scikit-learn

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.

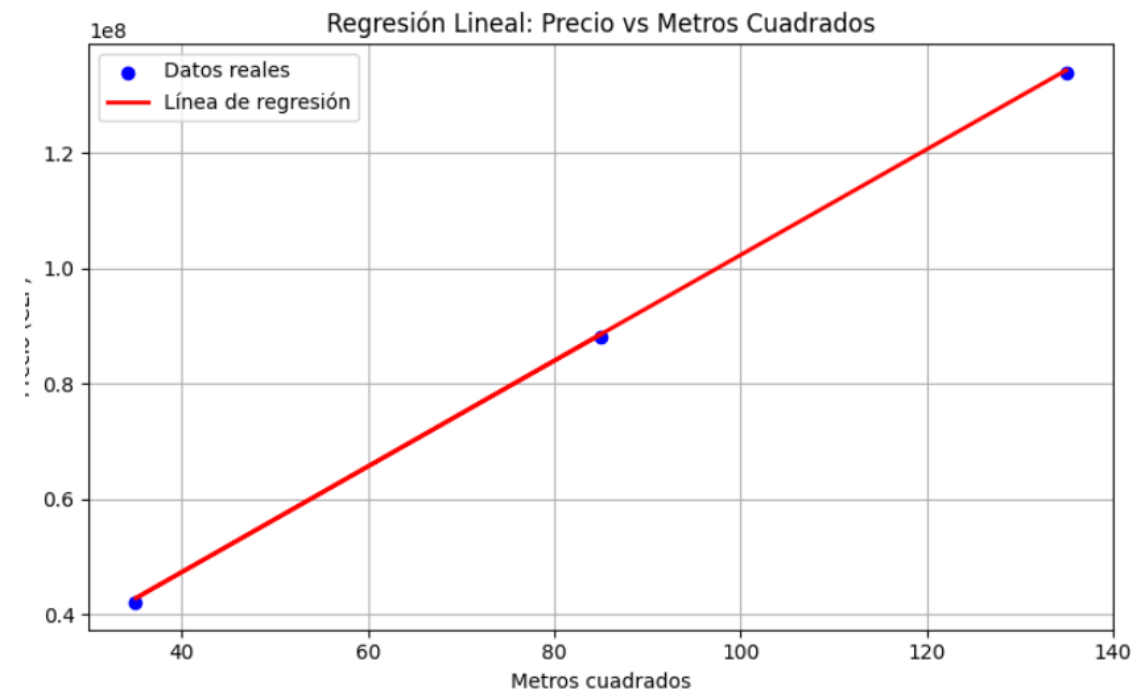
Si la línea roja atraviesa la nube de puntos, significa que el modelo ha aprendido correctamente la relación entre tamaño y precio.

```
✓ [19] import matplotlib.pyplot as plt
0s

# Asegurarse de tener predicciones listas
y_pred = modelo.predict(X_test)

# Crear gráfico
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, color='blue', label='Datos reales') # Puntos reales
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Línea de regresión') # Línea del modelo

plt.title("Regresión Lineal: Precio vs Metros Cuadrados")
plt.xlabel("Metros cuadrados")
plt.ylabel("Precio (CLP)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



# Implementación con Scikit-learn

## 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



# Implementación con Scikit-learn

 Generar

create a dataframe with 2 columns and 10 rows




```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Calcular métricas
mae = mean_absolute_error(y_test, predicciones)
mse = mean_squared_error(y_test, predicciones)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, predicciones)

# Mostrar resultados
print(" Evaluación del modelo:")
print(f"MAE (Error absoluto medio): {mae:,.0f} CLP")
print(f"MSE (Error cuadrático medio): {mse:,.0f}")
print(f"RMSE (Raíz del error cuadrático): {rmse:,.0f} CLP")
print(f"R² (Coeficiente de determinación): {r2:.3f}")
```

 Evaluación del modelo:  
MAE (Error absoluto medio): 567,185 CLP  
MSE (Error cuadrático medio): 356,701,723,921  
RMSE (Raíz del error cuadrático): 597,245 CLP  
R² (Coeficiente de determinación): 1.000

# Recursos Adicionales

- Video: [Statistics 101: Linear Regression, The Very Basics](#) 
- Este video proporciona una introducción clara y concisa a los conceptos fundamentales de la regresión lineal.
- Te compartimos un artículo que vale la pena leer:  
Artículo: [Regresión Lineal: teoría y ejemplos](#)
- Video: [¿Que es un modelo de regresión lineal? explicado con manzanitas](#)
- Video: [Cómo usar LinearRegression en Scikit-learn](#)
- Documentación oficial: [aquí](#)

# Validación Cruzada y Equilibrio de Modelos

## Técnicas de Validación Cruzada

1. ¿Qué técnica de validación cruzada te pareció más útil o realista para aplicar en proyectos con datos reales? ¿Por qué?

## Equilibrio del Modelo

2. ¿Qué aprendiste sobre el equilibrio entre un modelo que "sabe poco" y uno que "sabe demasiado"? ¿Cuál crees que es el mayor riesgo?

## Mantenimiento a Largo Plazo

3. ¿Cómo te aseguras en el futuro de que tu modelo no solo funcione, sino que funcione bien a lo largo del tiempo?