

Taller:

¿Está funcionando mi fábrica?

Introducción a la Ingeniería Industrial con Python



Producción normal ➤ Análisis de ingeniería ➤ Defecto detectado ➤ Máquina en paro

Mtro. Augusto Ariel Aguilar Ayala.
Director Académico de Ingeniería Industrial
aaguilara@upemor.edu.mx

Dra. Jessica V. Briseño Ruiz
Profesora de Tiempo Completo de Ing. Industrial
jbriseno@upemor.edu.mx

Objetivo del Taller

- Analizar datos en un contexto industrial.
- Identificar factores que impactan el desempeño.
- Interpretar gráficos y relaciones entre variables.
- Desarrollar pensamiento analítico basado en evidencia

▼ Contexto del Problema



Contexto del Problema

Se analizarán datos de una planta de producción que fabrica **1000 unidades** diarias como meta. La empresa sospecha que está perdiendo eficiencia, pero no conoce la causa. El **equipo de Ingeniería Industrial** encargado de diagnosticar el problema.

Variables disponibles:

- | | |
|--------------------------|--------------------------|
| ✓ Producción planificada | ✓ Número de operadores |
| ✓ Producción real | ✓ Defectos de producción |
| ✓ Tiempo de paro | |

▼ Análisis

1. Bibliotecas de Python

Escribe las siguientes líneas de código para importar las bibliotecas:

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
pd.options.display.float_format = '{:.2f}'.format
```

```

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

pd.options.display.float_format = '{:.2f}'.format

```

Escribe línea de código **data.head(5)** para ver los primeros cinco renglones del dataset **data**

▼ 2. Obtención de Datos de Producción

[Mostrar código](#)

Escribe línea de código **data.head(5)** para ver los primeros cinco renglones del dataset **data**

`data.head(5)`

Dia	Produccion_Plan	Tiempo_Paro	Operadores	Produccion_Real	Defectos	grid icon
0	1	1000	56	4	802	34
1	2	1000	19	3	957	56
2	3	1000	65	5	750	33
3	4	1000	25	4	911	28
4	5	1000	28	3	916	29

Próximos pasos: [Generar código con data](#) [New interactive sheet](#)

Escribe línea de código **data.shape** para ver el tamaño del dataset **data**. (Renglones,Columnas)

`data.shape`

(20, 6)

Escribe línea de código **data["Produccion_Plan"]** para ver los datos de la columna **Produccion_Plan** del dataset **data**.

Haz doble clic (o ingresa) para editar

▼ 3. Cálculo de Indicadores (KPIs)

Escribe línea de código **data.columns** para el nombre de cada columna del dataset **data**.

¿Qué es un KPI?

Un **KPI** (Key Performance Indicator) es un **índicador clave** que permite medir el desempeño de un proceso y apoyar la **toma de decisiones**.



💡 Eficiencia (%)

¿Qué información proporciona?

- Indica qué tanto se cumple la producción planificada. Permite identificar si la planta produce **menos, igual o más** de lo esperado.

$$\text{Eficiencia}(\%) = \frac{\text{Producción Real}}{\text{Producción Plan}} \times 100$$

⚠️ Tasa de Defectos (%)

$$\text{Tasa de Defectos} (\%) = \frac{\text{Defectos}}{\text{Producción_Real}} \times 100$$

- Mide la **calidad** del proceso, indicando el porcentaje de unidades **defectuosas** respecto al total producido.

📊 Productividad

$$\text{Productividad} = \text{Producción Real} / \text{Operadores}$$

- Evalúa el desempeño del recurso humano, mostrando cuántas **unidades** produce **cada operador**.

Escribe las siguientes líneas de código calcular los tres KPIs en el dataset **data**.

- Eficiencia (%)
- Tasa_Defectos (%)
- Productividad

```

data["Eficiencia (%)"] = data["Produccion_Real"] / data["Produccion_Plan"] * 100
data["Tasa_Defectos (%)"] = data["Defectos"] / data["Produccion_Real"] * 100
data["Productividad"] = data["Produccion_Real"] / data["Operadores"]
data.head(5)

```

```

data["Eficiencia (%)"] = data["Produccion_Real"] / data["Produccion_Plan"] * 100
data["Tasa_Defectos (%)"] = data["Defectos"] / data["Produccion_Real"] * 100
data["Productividad"] = data["Produccion_Real"] / data["Operadores"]
data.head(5)

```

	Dia	Produccion_Plan	Tiempo_Paro	Operadores	Produccion_Real	Defectos	Eficiencia (%)	Tasa_Defectos (%)	Productividad	grid icon
0	1	1000	56	4	802	34	80.20	4.24	200.50	
1	2	1000	19	3	957	56	95.70	5.85	319.00	
2	3	1000	65	5	750	33	75.00	4.40	150.00	
3	4	1000	25	4	911	28	91.10	3.07	227.75	
-	-	1000	66	4	810	66	81.00	6.17	200.00	

Próximos pasos: [Generar código con data](#) [New interactive sheet](#)

▼ 4. Cálculo de Estadísticos

Escribe las siguientes líneas de código para imprimir un texto y calcular los principales estadísticos del dataset **data**.

```

print("\n Resumen estadístico")
data.describe().T

```

```

print("\n Resumen estadístico")
data.describe().T

```

	count	mean	std	min	25%	50%	75%	max	grid icon
Dia	20.00	10.50	5.92	1.00	5.75	10.50	15.25	20.00	
Produccion_Plan	20.00	1000.00	0.00	1000.00	1000.00	1000.00	1000.00	1000.00	
Tiempo_Paro	20.00	38.45	21.20	6.00	25.00	35.50	58.25	68.00	
Operadores	20.00	4.20	0.77	3.00	4.00	4.00	5.00	5.00	
Produccion_Real	20.00	872.10	92.44	750.00	789.50	900.00	937.50	1000.00	
Defectos	20.00	34.35	10.69	17.00	28.00	33.50	38.75	56.00	
Eficiencia (%)	20.00	87.21	9.24	75.00	78.95	90.00	93.75	100.00	
Tasa_Defectos (%)	20.00	3.97	1.21	1.97	3.09	4.01	4.86	5.85	
Productividad	20.00	215.15	48.51	150.00	178.95	212.50	250.00	319.00	

¿Cuál fue el **tiempo de paro promedio** del proceso?



¿Tiempo de paro **mínimo** y **máximo**?

¿Diferencia entre el tiempo de paro mínimo y el máximo?

¿El proceso alcanza un **desempeño $\geq 95\%$** ?

¿La **calidad del proceso** es estable o muestra variaciones significativas?



¿La **productividad** se mantiene **constante** o varía entre días?

▼ 5. Visualización de datos

▼ 5.1 Gráfica de Producción Real vs. Producción Planeada

Ejecuta las siguientes líneas de código para generar la gráfica de **Producción Planeada vs. Producción Real**

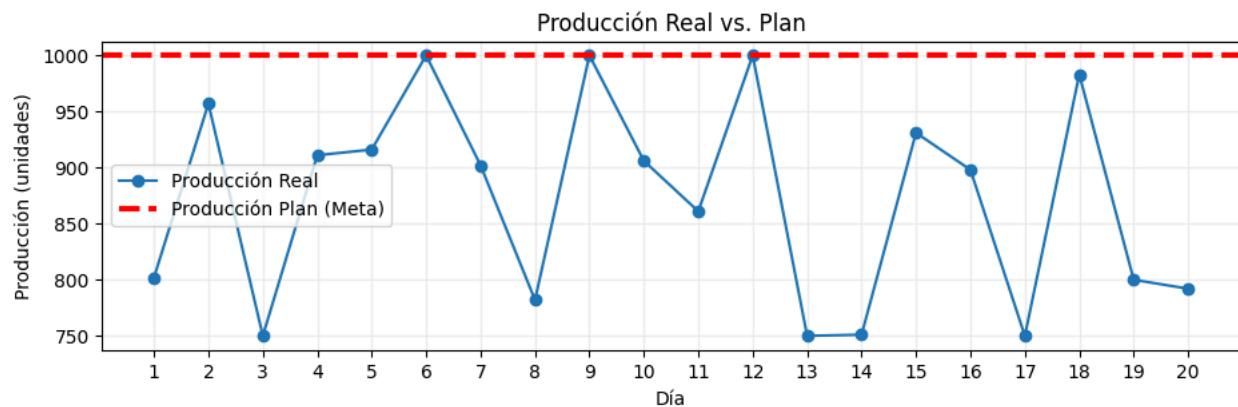
El código incluye comentarios explicativos para facilitar su comprensión.

```
# Crea la figura con tamaño (ancho, alto en pulgadas)
plt.figure(figsize=(11, 3))

# Gráfica: Eje X, Eje Y, 'o' agrega puntos visibles, etiqueta
plt.plot(data["Dia"], data["Produccion_Real"], marker='o', label="Producción Real")

# Línea horizontal que representa la producción planeada (meta):
plt.axhline(
    y=1000,                      # y=1000 indica el valor objetivo de producción
    linestyle='--',               # linestyle='--' define una línea discontinua
    linewidth=3,                  # linewidth=3 hace la línea más gruesa
    color='red',                  # color='red' la diferencia visualmente de la producción real
    label="Producción Plan (Meta)") #etiqueta

plt.title("Producción Real vs. Plan")      # Título de la gráfica
plt.xlabel("Día")                         # Etiqueta del eje X
plt.ylabel("Producción (unidades)")        # Etiqueta del eje Y
plt.xticks(range(1, 21))                  # Etiquetas Eje X del 1 al 20
plt.legend()                             # leyenda identificar cada línea
plt.grid(alpha=0.2)                      # Activa Grid
```



¿Cuántos días se alcanza la meta de **1000** unidades?



⚠️ ¿Cuántos días están por **debajo de la meta**?



⚠️ ¿Cuál fue el **día con menor producción**?

▼ 5.2 Gráfica de Defectos por día

Ejecuta las siguientes líneas de código para generar la gráfica de **Defectos por día**

El código incluye comentarios explicativos para facilitar su comprensión.

```
# Código: Creación de la gráfica "Defectos por Día"

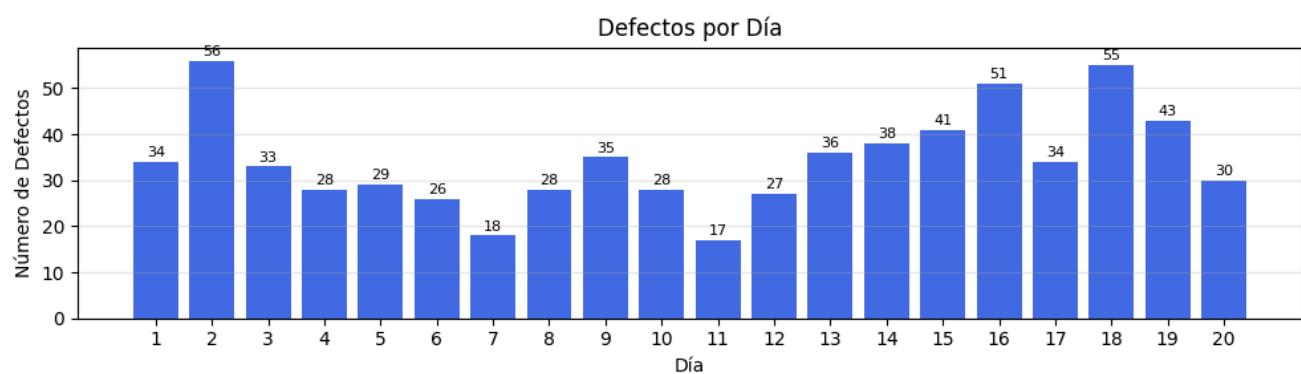
# Crea la figura con tamaño (ancho, alto en pulgadas)
plt.figure(figsize=(10, 3))

# Gráfica de barras: Eje X, Eje Y, color azul
bars = plt.bar(data["Dia"], data["Defectos"], color="royalblue")

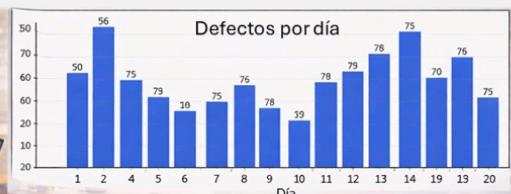
plt.title("Defectos por Día", fontsize=12)      # Título de la gráfica
plt.xlabel("Día")                                # Etiqueta del eje X
plt.ylabel("Número de Defectos")                 # Etiqueta del eje Y
plt.xticks(range(1, len(data) + 1))              # Etiquetas del eje X según número de días
plt.grid(axis='y', alpha=0.3)                     # Activa la cuadrícula solo en eje Y

# Agrega el valor numérico encima de cada barra
for bar in bars:
    height = bar.get_height()                    # Obtiene la altura de la barra
    plt.text(
        bar.get_x() + bar.get_width() / 2,       # Posición horizontal centrada
        height + 0.5,                           # Posición vertical ligeramente arriba
        f'{int(height)}',                      # Valor del número de defectos
        ha='center', va='bottom', fontsize=8   # Alineación y tamaño del texto
    )

plt.tight_layout()                             # Ajusta automáticamente los espacios
```



¿Cuál fue el **día con mayor número de defectos** y cuántos se registraron?



¿Se observa una **tendencia creciente, decreciente o variable** en el número de defectos a lo largo de 20 días con mayor número de defectos?

5.3 Gráfica de Eficiencia(%) vs Tiempo de Paro (min)

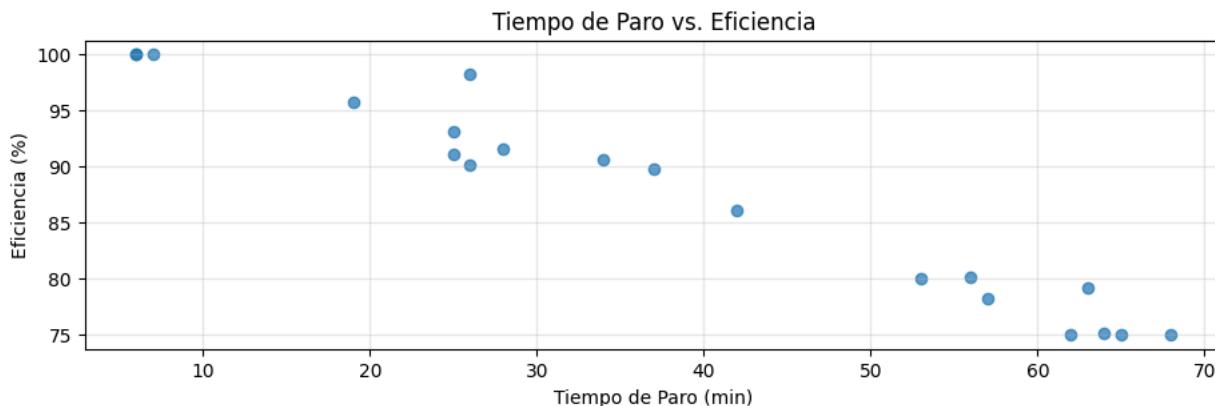
Ejecuta las siguientes líneas de código para generar la gráfica de Defectos por día

El código incluye comentarios explicativos para facilitar su comprensión.

```
plt.figure(figsize=(11, 3)) # Crea la figura con tamaño (ancho, alto en pulgadas)

# Gráfica de dispersión: Eje X, Eje Y, alpha controla la transparencia
plt.scatter(data["Tiempo_Paro"], data["Eficiencia (%)"], alpha=0.7)

plt.title("Tiempo de Paro vs. Eficiencia") # Título de la gráfica
plt.xlabel("Tiempo de Paro (min)") # Etiqueta del eje X
plt.ylabel("Eficiencia (%)") # Etiqueta del eje Y
plt.grid(alpha=0.3) # Activa la cuadricula con transparencia
```



¿La relación parece positiva, negativa o no hay patrón claro?



6. Análisis de Tendencia y Predicción

Ejecuta las siguientes líneas de código para la creación del:

- **Modelo lineal para estimación de la Eficiencia(%)**

El código incluye comentarios explicativos para facilitar su comprensión.

```
# Bloque 1: Cálculo del modelo lineal Tiempo de Paro vs. Eficiencia

# Regresión lineal: m = pendiente , b = intercepto
m, b = np.polyfit(
    data["Tiempo_Paro"], # Variable independiente (X)
    data["Eficiencia (%)"], # Variable dependiente (Y)
    1) # Grado 1: modelo lineal

# Calcular correlación y coeficiente de determinación R²
correlacion = data["Tiempo_Paro"].corr(data["Eficiencia (%)"])
r2 = correlacion**2

# Imprimir la ecuación del modelo y R²
print("Modelo lineal estimado:")
```

```

print("Modelo lineal estimado: ")
print(f"Eficiencia = {m:.2f} * Tiempo_Paro + {b:.2f}")
print(f"R2 = {r2:.3f}")

```

```

Modelo lineal estimado:
Eficiencia = -0.43 * Tiempo_Paro + 103.63
R2 = 0.959

```

Ejecuta las siguientes líneas de código para la creación de la **Gráfica de "Tiempo de Paro vs Eficiencia (%) con el modelo"**.

El código incluye comentarios explicativos para facilitar su comprensión.

```

plt.figure(figsize=(11, 3.5))      # Crea la figura con tamaño (ancho, alto en pulgadas)

# Gráfica de dispersión: Eje X, Eje Y, etiqueta
plt.scatter(data["Tiempo_Paro"],data["Eficiencia (%)"],alpha=0.7,
            label="Datos reales")

# Genera valores ordenados de X para una línea suave
x_vals = np.linspace(data["Tiempo_Paro"].min(),data["Tiempo_Paro"].max(),100)

# Calcula los valores estimados usando el modelo
y_vals = m * x_vals + b

# Dibuja la línea de tendencia
plt.plot(x_vals,y_vals,linewidth=3,label="Línea de tendencia")

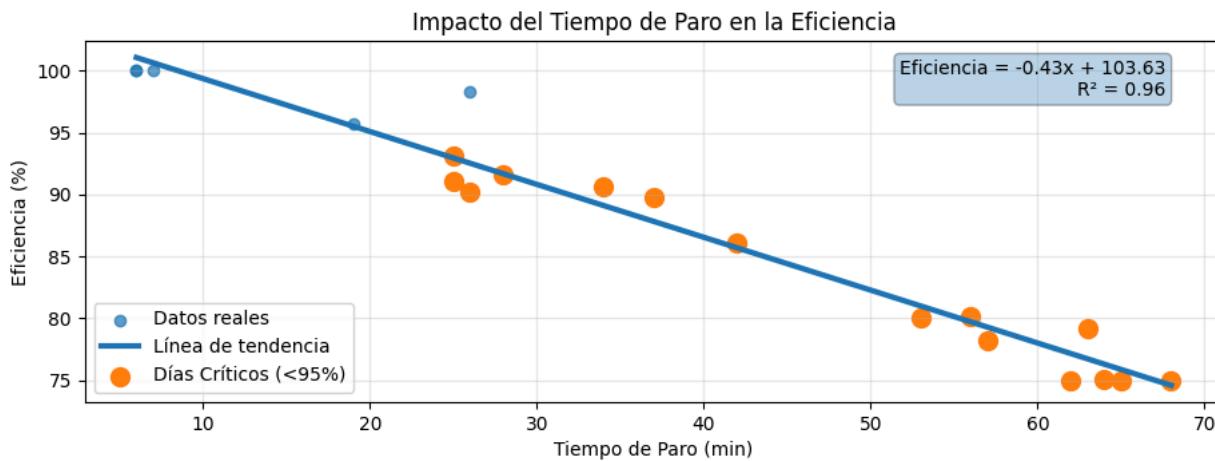
# Identifica días críticos con eficiencia menor al 95%
criticos = data[data["Eficiencia (%)" ] < 95]

# Resalta los días críticos: Eje X, Eje Y, etiqueta
plt.scatter(criticos["Tiempo_Paro"],criticos["Eficiencia (%)" ],s=100,
            label="Días Críticos (<95%)")

# Escribe la ecuación y R2 dentro de la gráfica
ecuacion = f"Eficiencia = {m:.2f}x + {b:.2f}\nR2 = {r2:.2f}"
plt.text(0.95, 0.95,ecuacion,
        transform=plt.gca().transAxes,
        verticalalignment='top',
        horizontalalignment='right',
        bbox=dict(boxstyle="round", alpha=0.3))

plt.title("Impacto del Tiempo de Paro en la Eficiencia") # Título
plt.xlabel("Tiempo de Paro (min)")                         # Eje X
plt.ylabel("Eficiencia (%)" )                            # Eje Y
plt.legend()                                              # Leyenda
plt.grid(alpha=0.3)                                       # Cuadrícula

```



Ejecuta las siguientes líneas de código para la creación de una **Función para estimar la Eficiencia del Proceso en función de Tiempo de Paro**. El código incluye comentarios explicativos para facilitar su comprensión.

```

def estimar_eficiencia(tiempo_paro):
    # Esta función recibe como entrada el tiempo de paro (en minutos)
    # y utiliza el modelo lineal (pendiente m y ordenada b)
    # para estimar la eficiencia del proceso.

```

```

eficiencia = m * tiempo_paro + b # Fórmula de la recta: y = mx + b

# Mostrar el resultado en pantalla
print(f"Si el tiempo de paro es {tiempo_paro} minutos,")
# print(f"La eficiencia estimada es {eficiencia:.2f}")

# Regresa el valor calculado por si se quiere usar después
return eficiencia

```

Ejecuta las siguientes líneas de código para la **Estimar la Eficiencia del Proceso en función de Tiempo de Paro**. El código incluye comentarios explicativos para facilitar su comprensión.

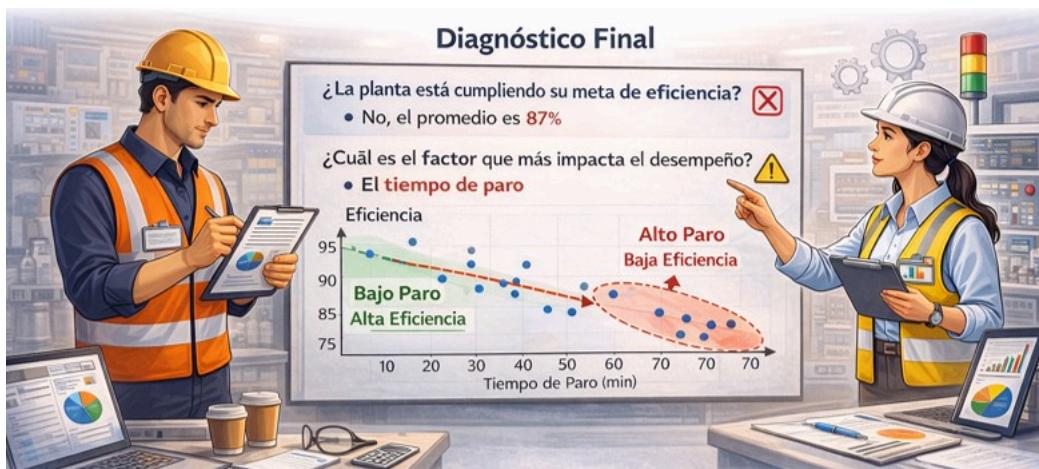
```

# Guardamos el resultado en una variable para usarlo después
ef_Estimada = estimar_eficiencia(30)

# Mostramos el valor almacenado
print(f"Eficiencia estimada es: {ef_Estimada:.2f}%")

Si el tiempo de paro es 30 minutos,
Eficiencia estimada es: 90.82%

```



▼ 7. Diagnóstico

Ejecuta las siguientes líneas de código para la **calcular y visualizar algunos indicadores importantes para el diagnóstico**. El código incluye comentarios explicativos para facilitar su comprensión.

```

print("\n====")
print("  TALLER: ¿ESTÁ FUNCIONANDO MI FÁBRICA?")
print("====")

print("\n⭐ DIAGNÓSTICO INTEGRAL")

# Calcular promedios de los principales indicadores
ef_prom = data["Eficiencia (%)].mean()
paro_prom = data["Tiempo_Paro"].mean()
TasDef_prom = data["Tasa_Defectos (%)].mean()

# Mostrar resultados promedio
print(f"\nEficiencia promedio: {ef_prom:.2f}%")
print(f"Tiempo de paro promedio: {paro_prom:.2f} min")
print(f"Tasa de defectos promedio: {TasDef_prom:.2f}%")

# Evaluar si se cumple la meta de eficiencia (95%)
if ef_prom < 95:
    print("\n⚠ La planta NO cumple la meta de eficiencia (95%).")
else:
    print("\n✅ La planta cumple la meta establecida.")

# Mostrar indicadores del modelo estadístico
print("\n📊 Análisis de relación estadística:")
print(f"R² del modelo: {r2:.2f}")           # Qué tan fuerte es la relación
print(f"Pendiente del modelo: {m:.2f}")      # Dirección de la relación

```

```

# Interpretar la pendiente
if m < 0:
    print("Existe una relación negativa: mayor tiempo de paro reduce la eficiencia.")

# Conclusión principal
print("\n⌚ Causa principal identificada:")
print("El tiempo de paro impacta directamente el desempeño del sistema.")

# Recomendación estratégica final
print("\n✖ Recomendación estratégica:")
print("Reducir el tiempo de paro debe ser la prioridad operativa.")

```

=====
 TALLER: ¿ESTÁ FUNCIONANDO MI FÁBRICA?
=====

❖ DIAGNÓSTICO INTEGRAL

Eficiencia promedio: 87.21%
 Tiempo de paro promedio: 38.45 min
 Tasa de defectos promedio: 3.97%

⚠ La planta NO cumple la meta de eficiencia (95%).

📊 Análisis de relación estadística:
 R^2 del modelo: 0.96
 Pendiente del modelo: -0.43
 Existe una relación negativa: mayor tiempo de paro reduce la eficiencia.

⌚ Causa principal identificada:
 El tiempo de paro impacta directamente el desempeño del sistema.

✖ Recomendación estratégica:
 Reducir el tiempo de paro debe ser la prioridad operativa.



<https://www.upemor.edu.mx>