

Tarea 1 2025

Instrucciones

Su notebook con las respuestas a la tarea se deben entregar a mas tardar el dia 21/04/25 hasta las 21:00, subiendolo al repositorio en la carpeta tareas/2025.

Es importante considerar que el código debe poder ejecutarse en cualquier computadora con la data original del repositorio. Recordar la convencion para el nombre de archivo ademas de incluir en su documento titulos y encabezados por seccion. La data a utilizar es **machine_failure_data.csv**.

Las variables tienen la siguiente descripcion:

- Date: data medida en frecuencia diaria
- Location: ubicacion del medidor
- Min_Temp: temperatura minima observada
- Max_Temp: temperatura maxima observada
- Leakage: Filtracion medida en el area
- Evaporation: Tasa de evaporacion
- Electricity: Consumo electrico KW
- Parameter#: Diferentes sensores de reportando direccion y velocidad de viento en distintos momentos del dia, asi como otras metricas relevantes.
- Failure today: El sensor reporta fallo (o no)

Preguntas (todas tienen el mismo puntaje):

1. Cargar la base de datos en el ambiente. Identifique los tipos de datos que se encuentran en la base, realice estadisticas descriptivas sobre las variables importantes (Hint: Revisar la distribuciones, datos faltantes, outliers, etc.) y limpie las variables cuando sea necesario.
2. Ejecute un modelo de probabilidad lineal (*MCO*) que permita explicar la probabilidad de que un dia se reporte fallo medido por sensor, a partir de las informacion disponible. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.
3. Ejecute un modelo *probit* para responder a la pregunta 2. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.
4. Ejecute un modelo *logit* para responder a la pregunta 2. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.
5. Comente los resultados obtenidos en 2, 3 y 4. ¿Cuáles y por qué existen las diferencias entre los resultados?. En su opinion, ¿Cuál sería el más adecuado para responder la

pregunta de investigación y por qué? ¿Qué variables resultaron ser robustas a la especificación?

6. Agregue la data a nivel mensual, usando la data promedio de las variables (ignorando aquellas categoricas, como la direccion del viento). En particular, genere una variable que cuente la cantidad de fallos observados en un mes, utilice un valor de 0 si en ese mes no se reporto fallos en ningun dia. Use un modelo Poisson para explicar el numero de fallas por mes. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.
7. Determine sobre dispersion en la data y posible valor optimo de alpha para un modelo Binomial Negativa.
8. Usando la informacion anterior, ejecute un modelo Binomial Negativa para responder a la pregunta 6. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.
9. Comente los resultados obtenidos en 6, 7 y 8. ¿Cuáles y por qué existen las diferencias entre los resultados?. En su opinión, ¿Cuál sería el más adecuado para responder la pregunta de investigación y por qué? ¿Qué variables resultaron ser robustas a la especificación?

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn
import scipy
from scipy.stats import nbinom
import seaborn as sns
from statsmodels.iolib.summary2 import summary_col
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Cargar la base de datos machine_failure_data.csv en el ambiente. Identifique los tipos de datos que se encuentran en la base

```
In [2]: import pandas as pd

df = pd.read_csv('data/machine_failure_data.csv')
df
```

Out[2]:

	Date	Location	Min_Temp	Max_Temp	Leakage	Evaporation	Electricity	Par
0	12/1/2008	3	13.4	22.9	0.6	NaN	NaN	NaN
1	12/2/2008	3	7.4	25.1	0.0	NaN	NaN	NaN
2	12/3/2008	3	12.9	25.7	0.0	NaN	NaN	NaN
3	12/4/2008	3	9.2	28.0	0.0	NaN	NaN	NaN
4	12/5/2008	3	17.5	32.3	1.0	NaN	NaN	NaN
...
142188	6/20/2017	42	3.5	21.8	0.0	NaN	NaN	NaN
142189	6/21/2017	42	2.8	23.4	0.0	NaN	NaN	NaN
142190	6/22/2017	42	3.6	25.3	0.0	NaN	NaN	NaN
142191	6/23/2017	42	5.4	26.9	0.0	NaN	NaN	NaN
142192	6/24/2017	42	7.8	27.0	0.0	NaN	NaN	NaN

142193 rows × 22 columns

In [3]: `df.describe()`

Out[3]:

	Location	Min_Temp	Max_Temp	Leakage	Evaporation	Electri
count	142193.000000	141556.000000	141871.000000	140787.000000	81350.000000	74377.00
mean	24.740655	12.186400	23.226784	2.349974	5.469824	7.62
std	14.237503	6.403283	7.117618	8.465173	4.188537	3.78
min	1.000000	-8.500000	-4.800000	0.000000	0.000000	0.00
25%	12.000000	7.600000	17.900000	0.000000	2.600000	4.90
50%	25.000000	12.000000	22.600000	0.000000	4.800000	8.50
75%	37.000000	16.800000	28.200000	0.800000	7.400000	10.60
max	49.000000	33.900000	48.100000	371.000000	145.000000	14.50

In [4]: `print(df.columns.tolist())`

```
[ 'Date', 'Location', 'Min_Temp', 'Max_Temp', 'Leakage', 'Evaporation', 'Electricity',
  'Parameter1_Dir', 'Parameter1_Speed', 'Parameter2_9am', 'Parameter2_3pm', 'Parameter3_9am',
  'Parameter3_3pm', 'Parameter4_9am', 'Parameter4_3pm', 'Parameter5_9am', 'Parameter5_3pm',
  'Parameter6_9am', 'Parameter6_3pm', 'Parameter7_9am', 'Parameter7_3pm', 'Failure_today']
```

In [5]: `# Ver los valores únicos de la columna 'Failure_today'`
`print("Valores únicos en df_l['Failure_today']:")`
`print(df['Failure_today'].unique())`

Valores únicos en df_1['Failure_today']:
['No' 'Yes' nan]

In [6]: # Identificar columnas con valores NaN y contar la cantidad de NaNs en cada columna
nan_columns = df.isna().sum()

Filtrar solo las columnas que tienen al menos un NaN
nan_columns = nan_columns[nan_columns > 0]

Mostrar las columnas y la cantidad de NaNs
print("Columnas con valores NaN y su cantidad:")
print(nan_columns)

Columnas con valores NaN y su cantidad:

Min_Temp	637
Max_Temp	322
Leakage	1406
Evaporation	60843
Electricity	67816
Parameter1_Dir	9330
Parameter1_Speed	9270
Parameter2_9am	10013
Parameter2_3pm	3778
Parameter3_9am	1348
Parameter3_3pm	2630
Parameter4_9am	1774
Parameter4_3pm	3610
Parameter5_9am	14014
Parameter5_3pm	13981
Parameter6_9am	53657
Parameter6_3pm	57094
Parameter7_9am	904
Parameter7_3pm	2726
Failure_today	1406
	dtype: int64

In [7]: print("Valores únicos y su frecuencia en 'Evaporation':")
print(df['Evaporation'].value_counts(dropna=False))

Valores únicos y su frecuencia en 'Evaporation':

NaN	60843
4.0	3282
8.0	2574
2.2	2057
2.0	1996
...	
44.4	1
44.0	1
50.4	1
35.8	1
39.6	1

Name: Evaporation, Length: 357, dtype: int64

In [8]: # Ver los valores únicos de la columna 'Leakage' y su frecuencia
print("Valores únicos y su frecuencia en 'Electricity':")
print(df['Electricity'].value_counts(dropna=False))

```
Valores únicos y su frecuencia en 'Electricity':  
NaN      67816  
0.0      2308  
10.7     1087  
11.0     1078  
10.8     1058  
...  
14.0      15  
14.1       6  
14.3       4  
14.2       2  
14.5       1  
Name: Electricity, Length: 146, dtype: int64
```

Tratamiento de valores NaN en el dataset

1)Electricity y Evaporation: Identificamos que estos parámetros requerían un enfoque especial. Decidimos crear una columna binaria para cada uno, donde:

- 1 indica la presencia de un valor NaN
- 0 indica que hay un valor numérico registrado.

2)Parámetro 6: Debido a la alta cantidad de valores NaN y a la falta de un método claro para imputarlos, optamos por excluir esta variable de los análisis para las preguntas 2, 3 y 4.

3)Otras variables categóricas: Como el número de valores NaN en estas variables era bajo, procedimos a eliminarlos. Esta estrategia nos permitió mantener la mayor integridad posible de los datos para el resto del análisis.

Este enfoque busca equilibrar el manejo de datos faltantes con la preservación de la calidad y utilidad del dataset.

```
In [9]: import pandas as pd

# Suponiendo que tu DataFrame se llama df
df_1 = df.copy()

# 2. Eliminar las filas con NaN en las otras columnas (excepto Evaporation y Electrical)
columnas_a_limpiar = [
    'Min_Temp', 'Max_Temp', 'Leakage',
    'Parameter1_Dir', 'Parameter1_Speed',
    'Parameter2_9am', 'Parameter2_3pm',
    'Parameter3_9am', 'Parameter3_3pm',
    'Parameter4_9am', 'Parameter4_3pm',
    'Parameter5_9am', 'Parameter5_3pm',
    'Parameter7_9am', 'Parameter7_3pm',
    'Failure_today'
]

df_1 = df_1.dropna(subset=columnas_a_limpiar)
```

```
In [10]: import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Configuración inicial
plt.style.use('ggplot')
figsize = (12, 6)

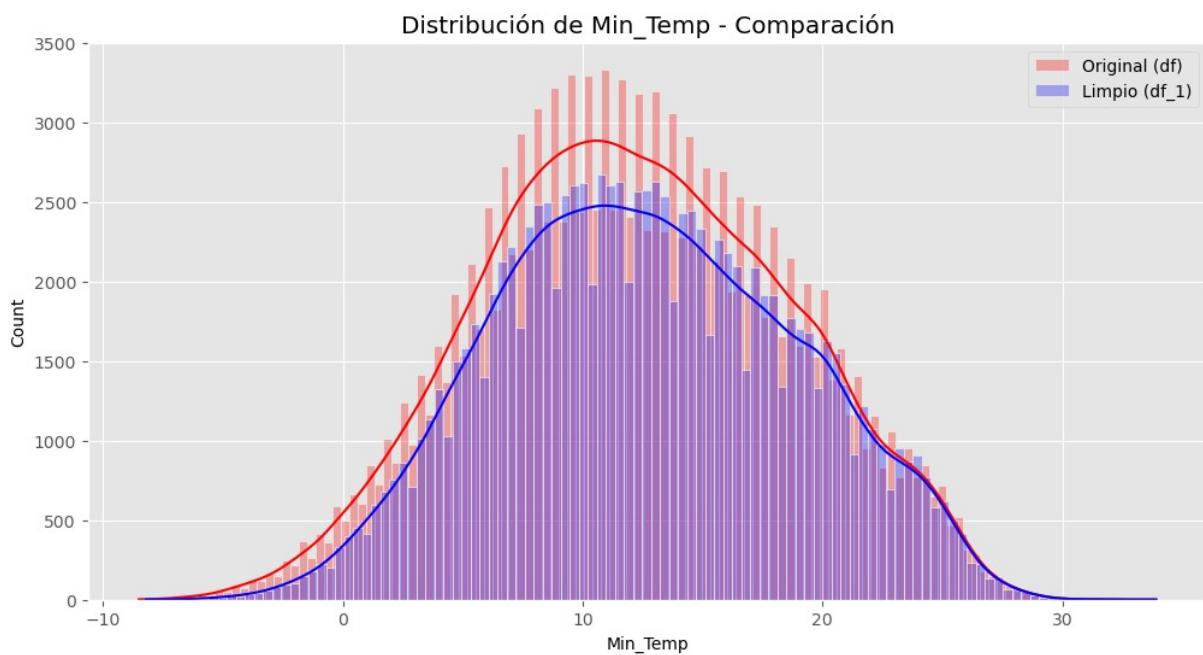
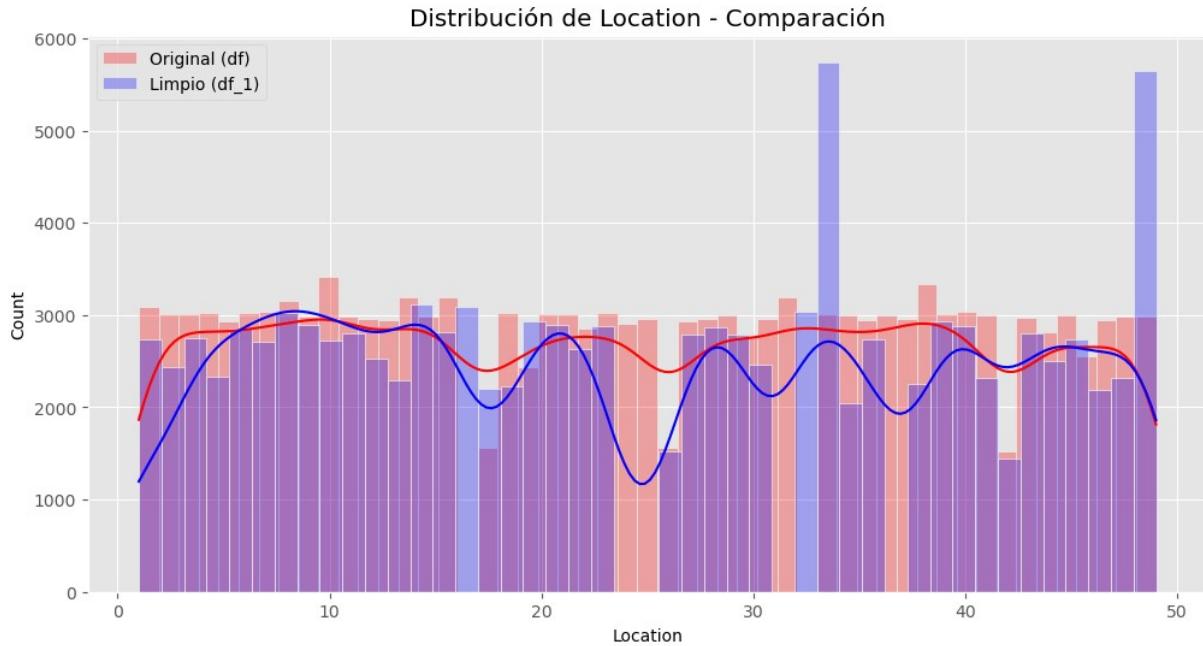
# Seleccionar solo columnas numéricas para la comparación
numeric_cols = df.select_dtypes(include=np.number).columns.tolist()

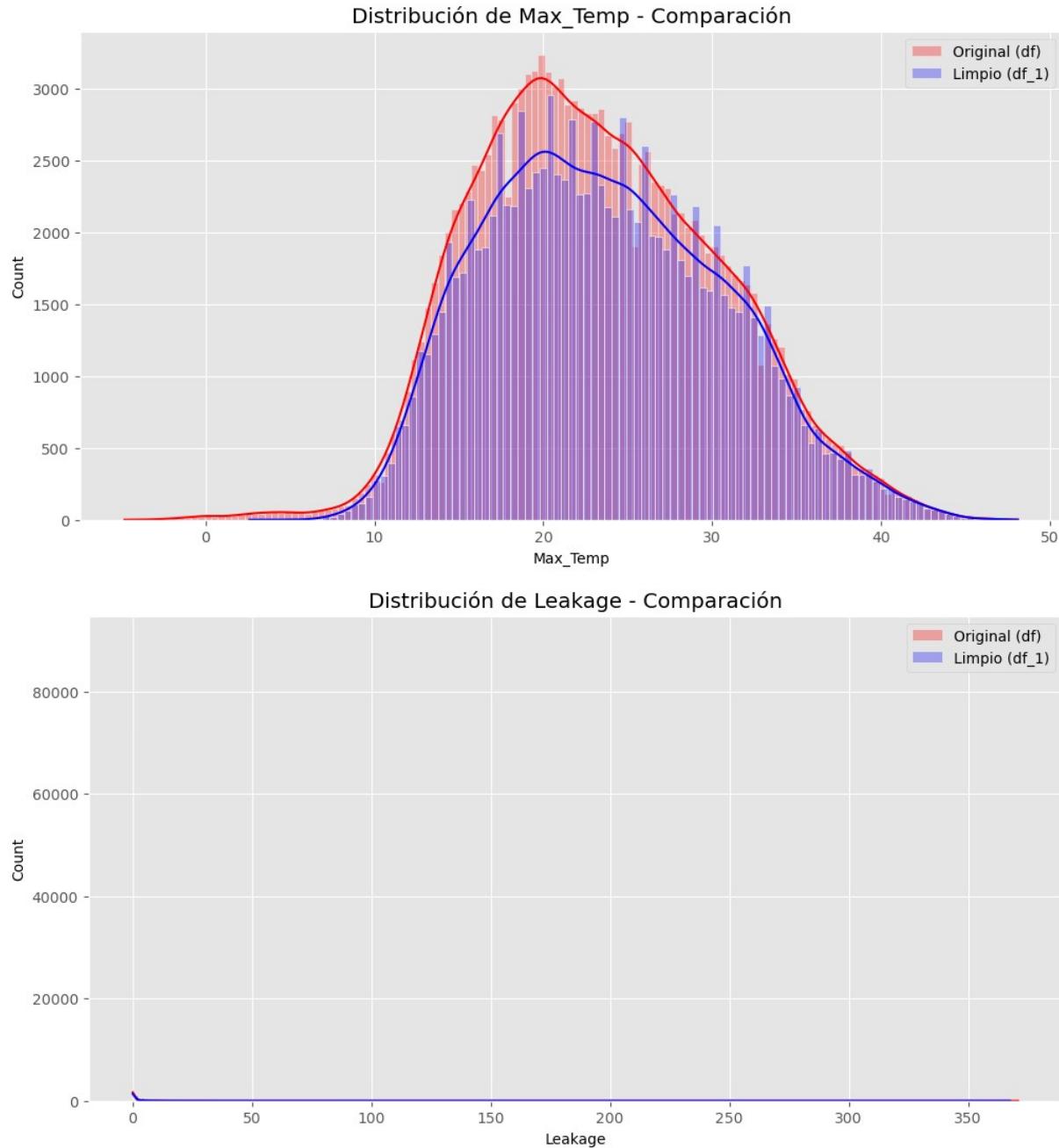
# Gráficos de distribución para cada variable numérica
for col in numeric_cols:
    plt.figure(figsize=figsize)

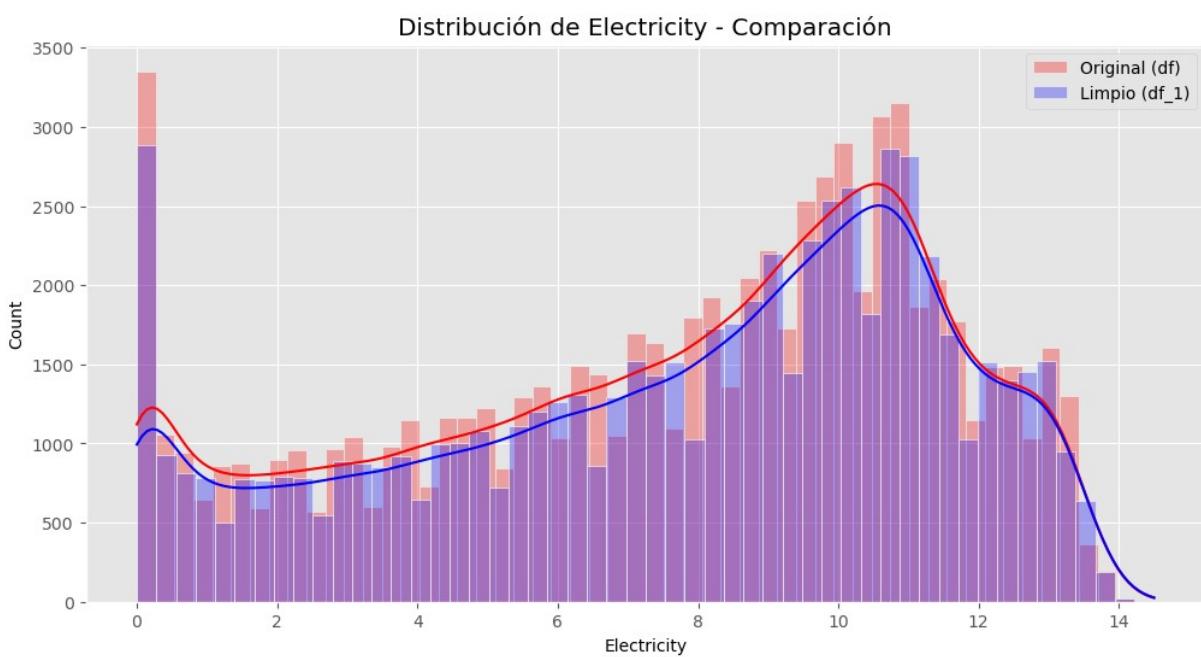
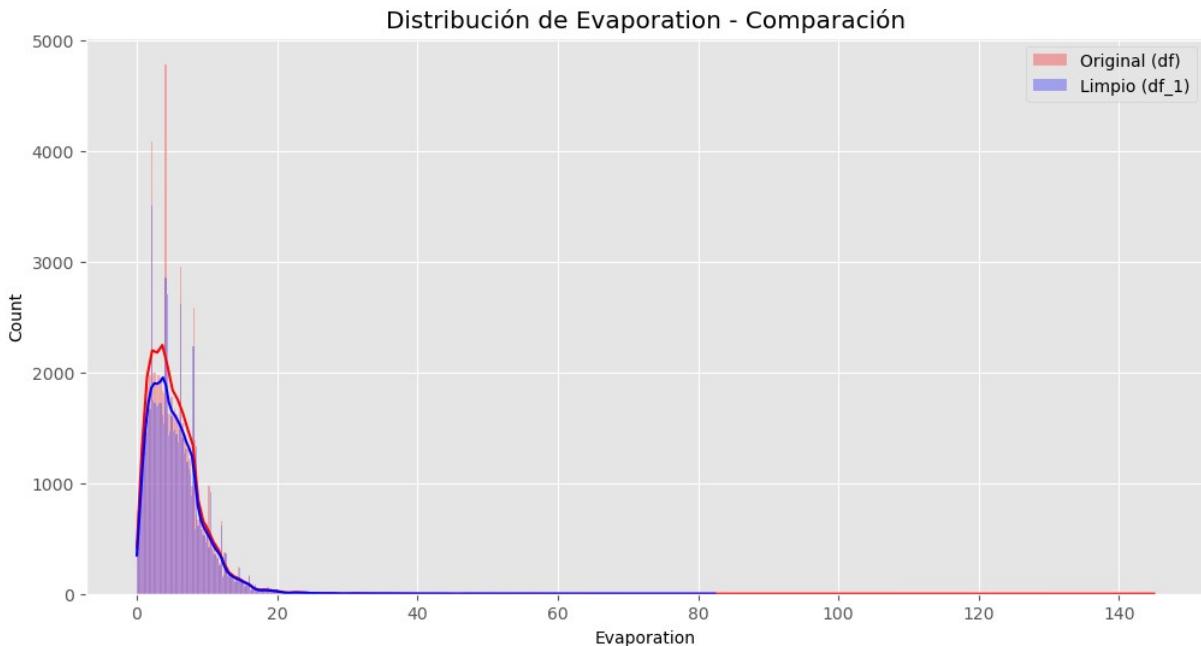
    # Histograma y KDE para df (no limpio)
sns.histplot(df[col], kde=True, color='red', alpha=0.3, label='Original (df)')

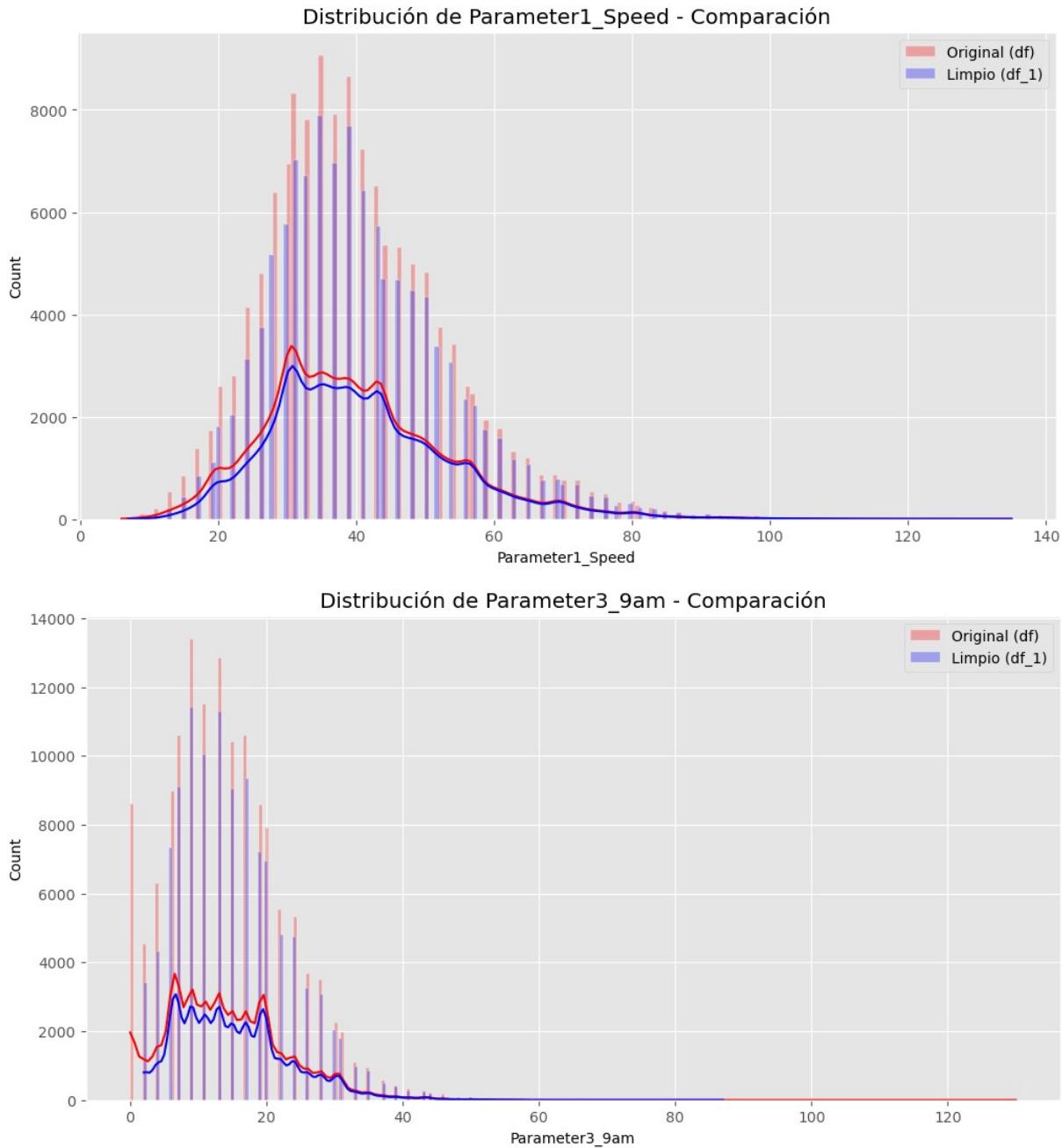
    # Histograma y KDE para df_1 (limpio)
sns.histplot(df_1[col], kde=True, color='blue', alpha=0.3, label='Limpio (df_1)'

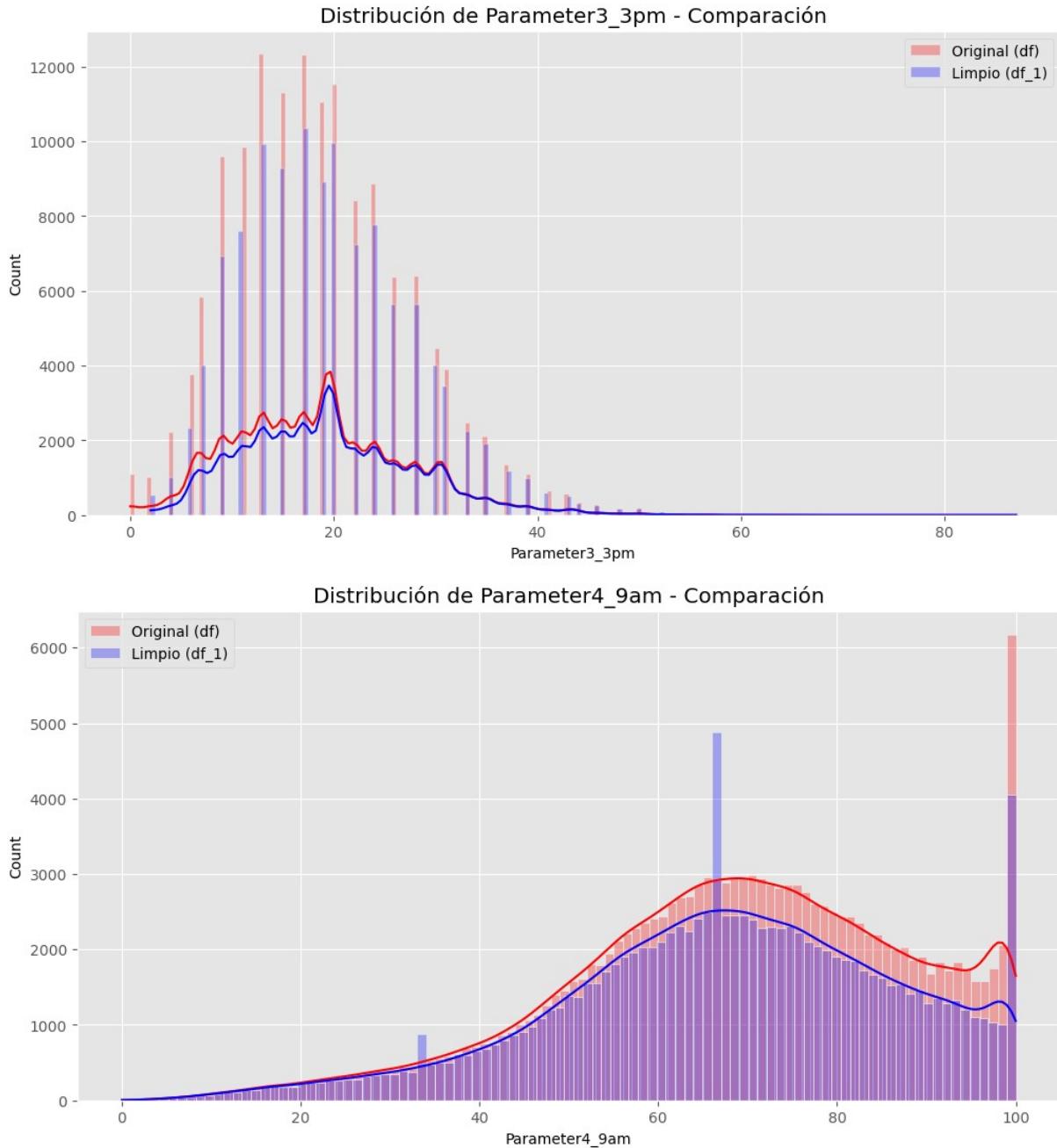
    plt.title(f'Distribución de {col} - Comparación')
    plt.legend()
    plt.show()
```

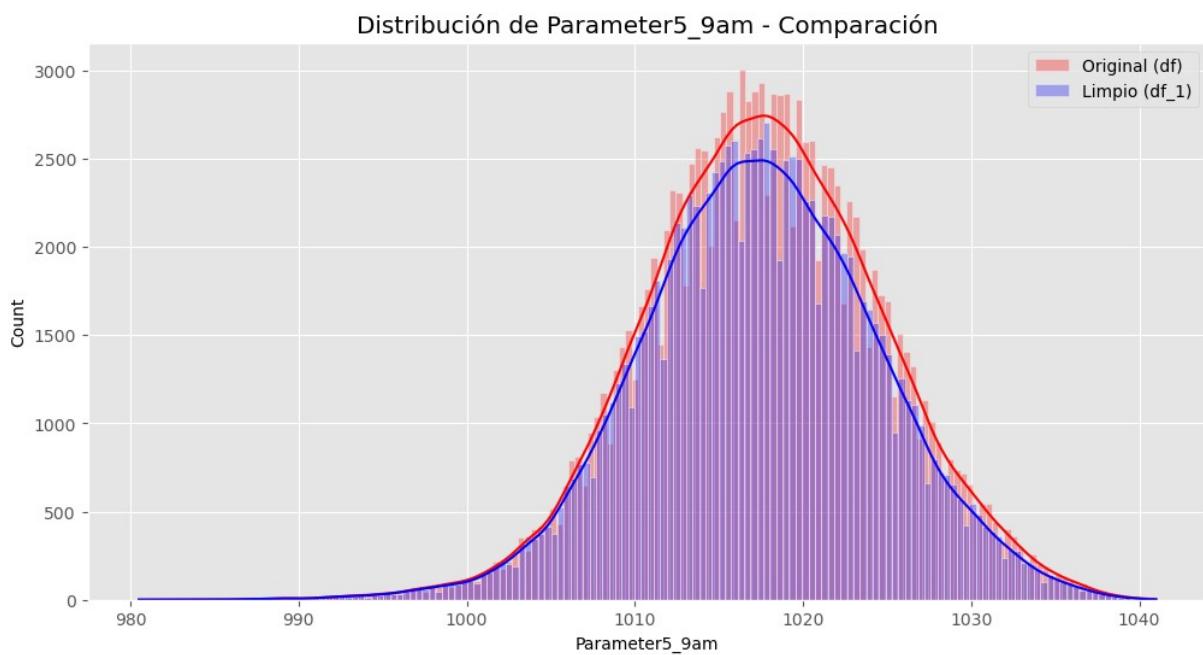
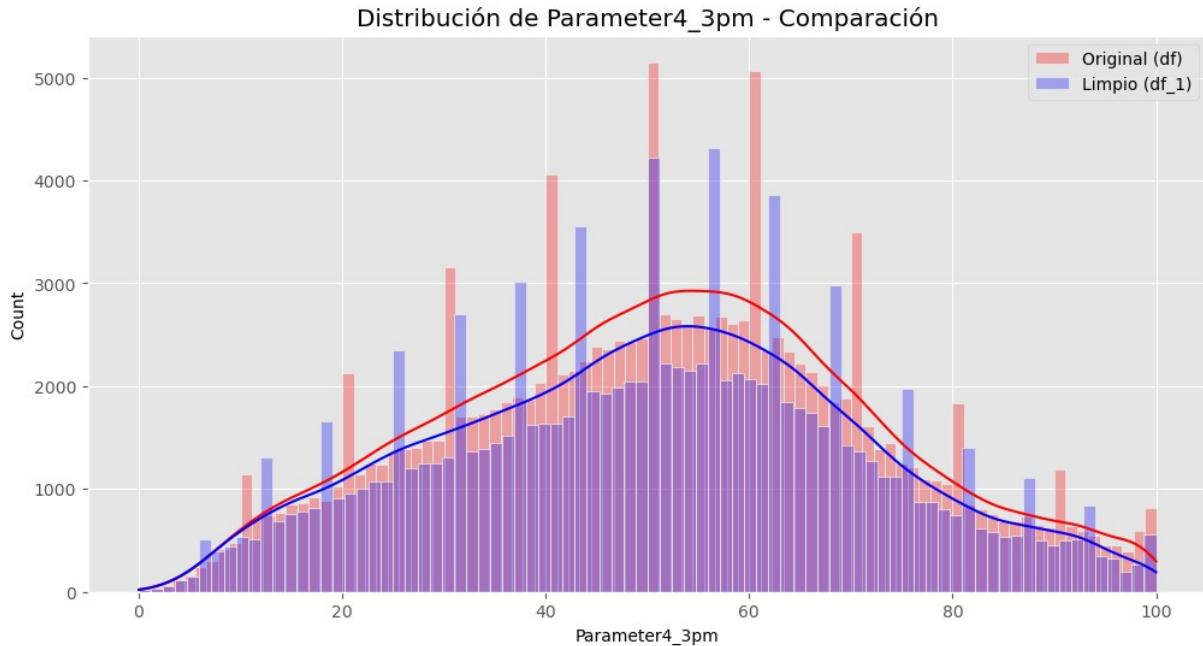


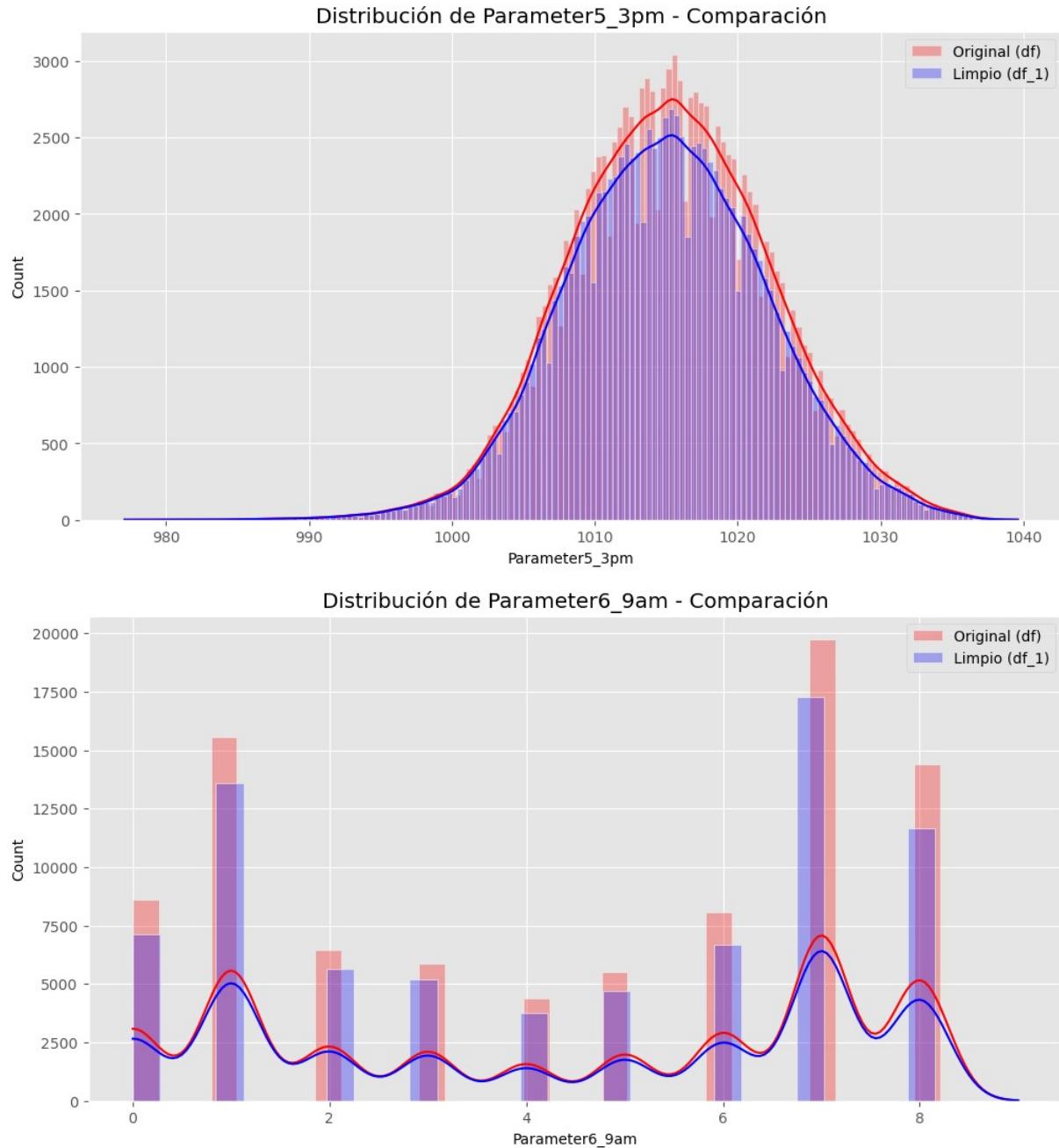


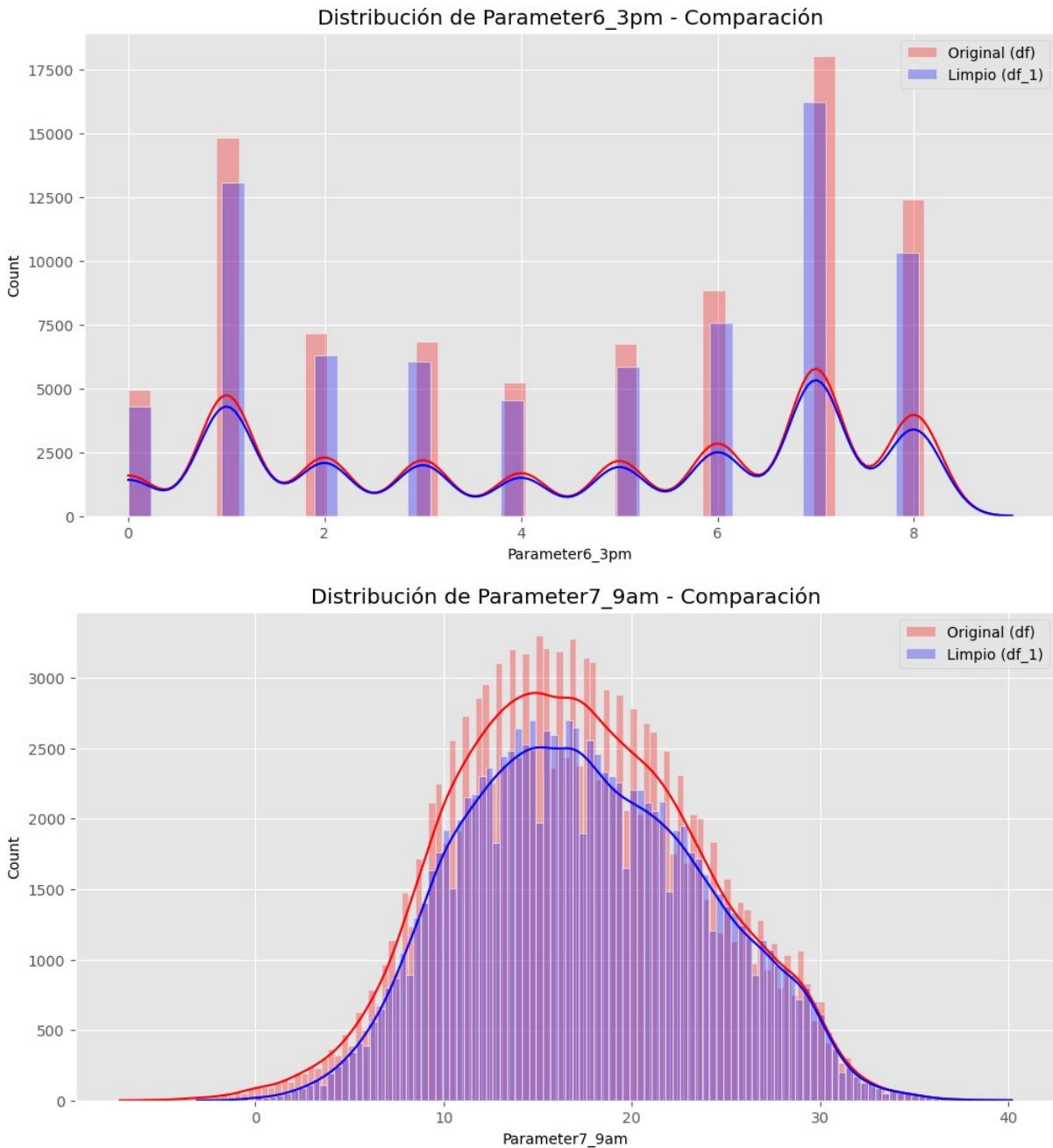


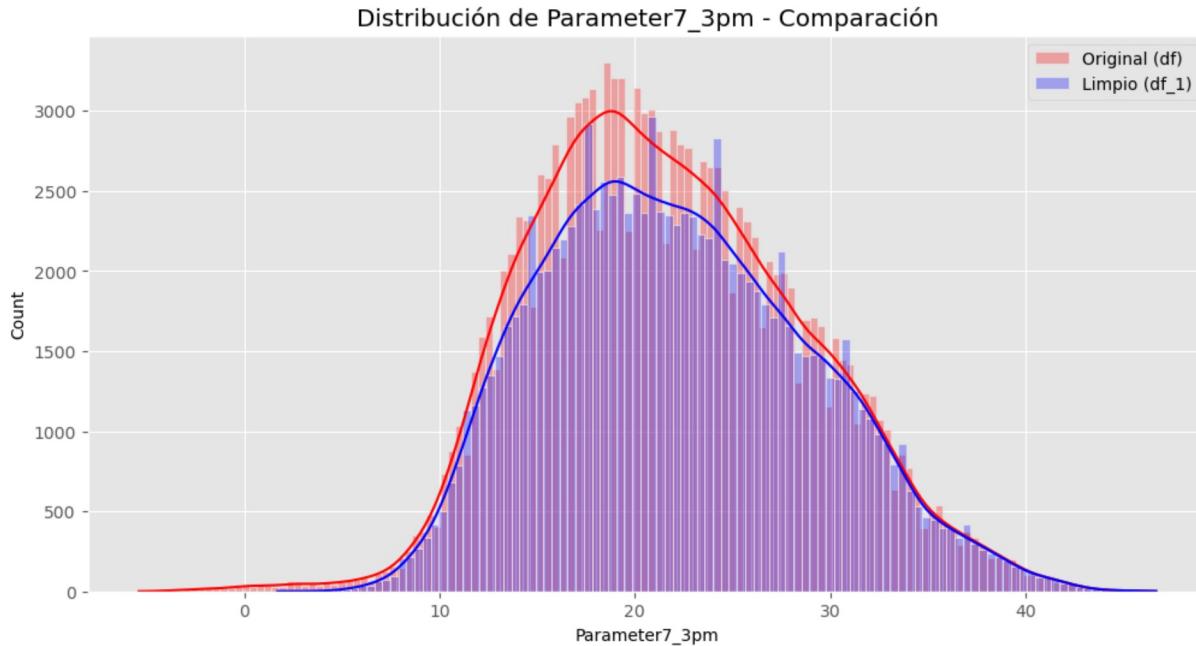








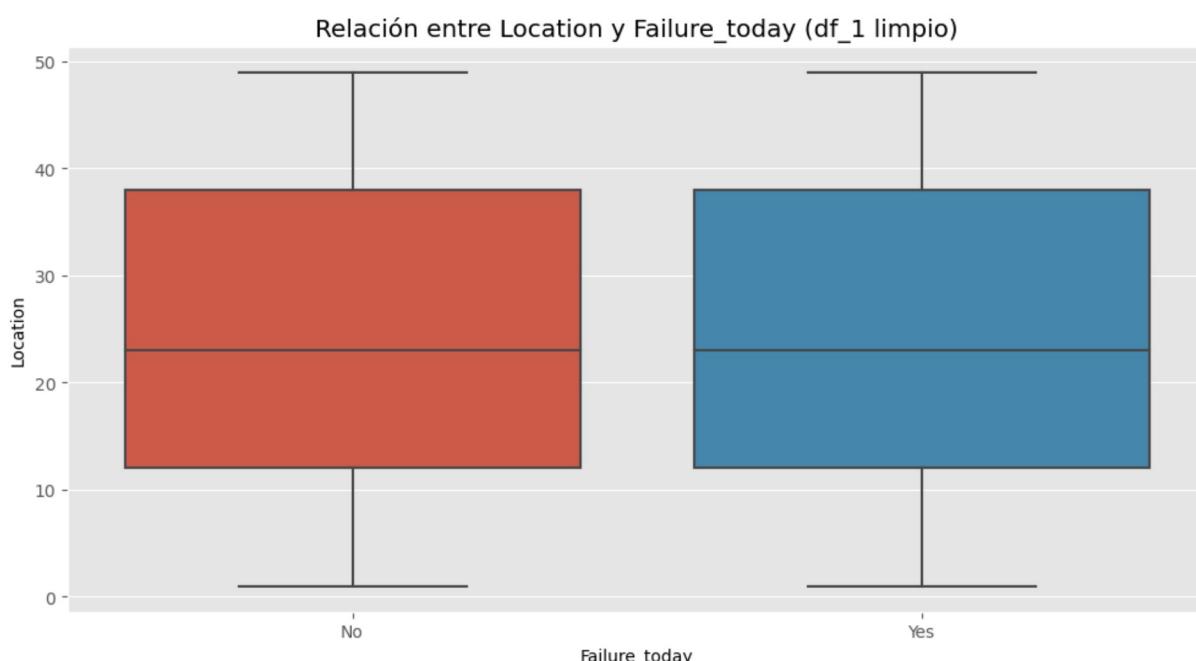


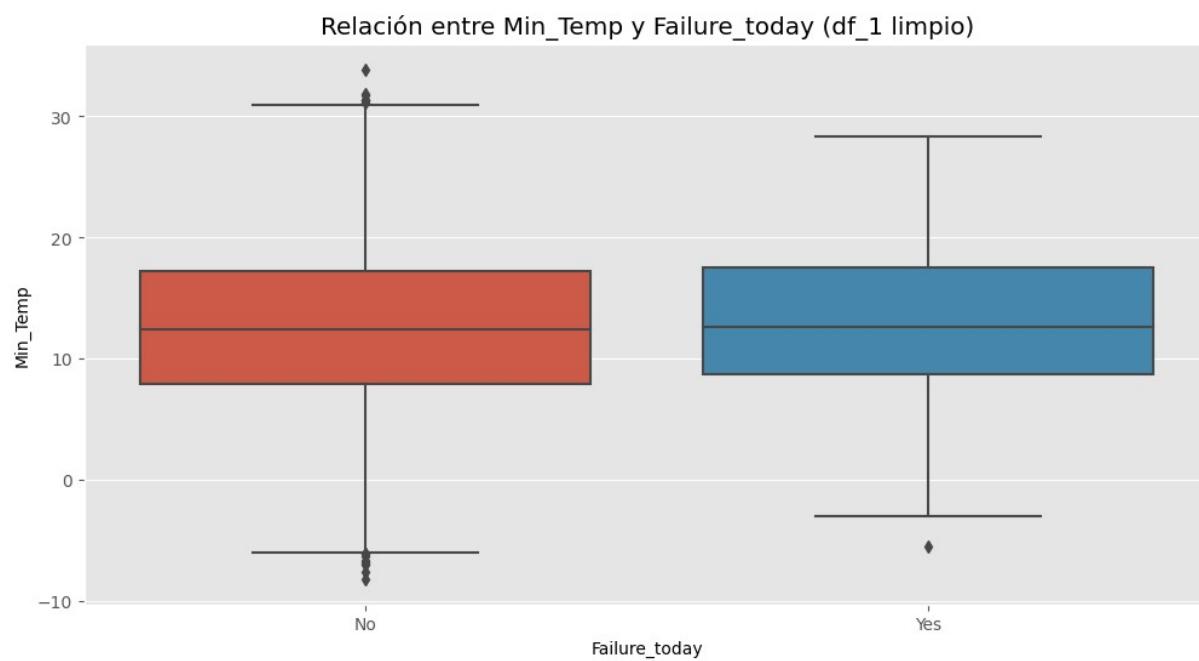
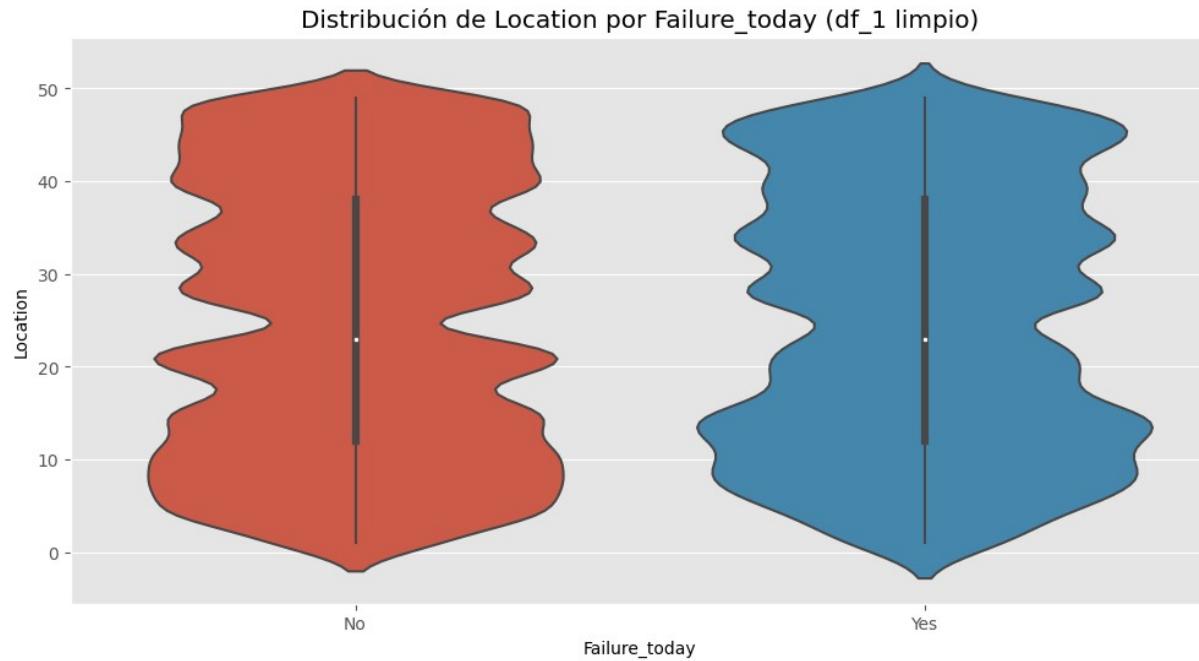


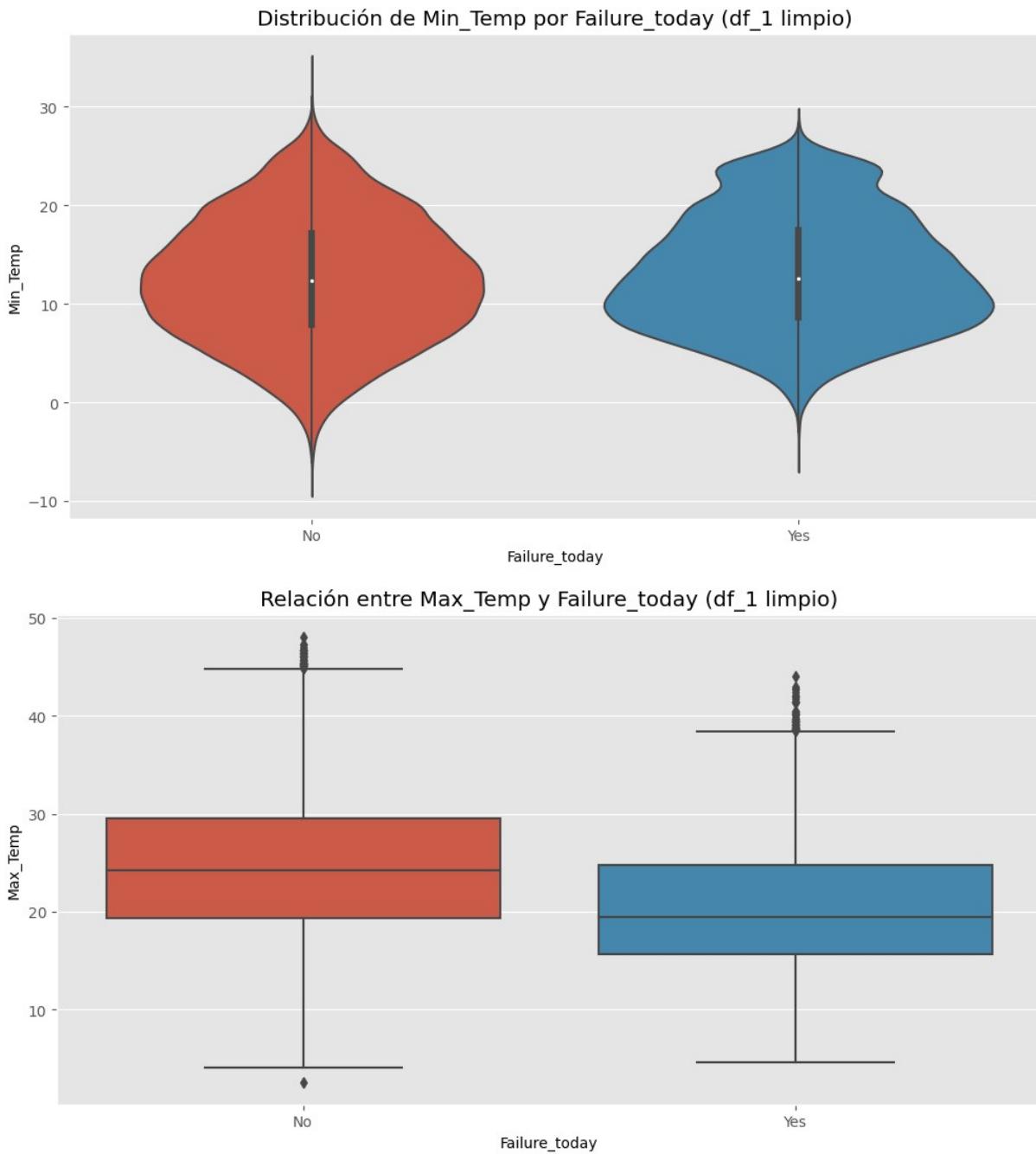
```
In [11]: # Gráficos de relación para variables numéricas vs Failure_today
for col in numeric_cols:
    plt.figure(figsize=figsize)

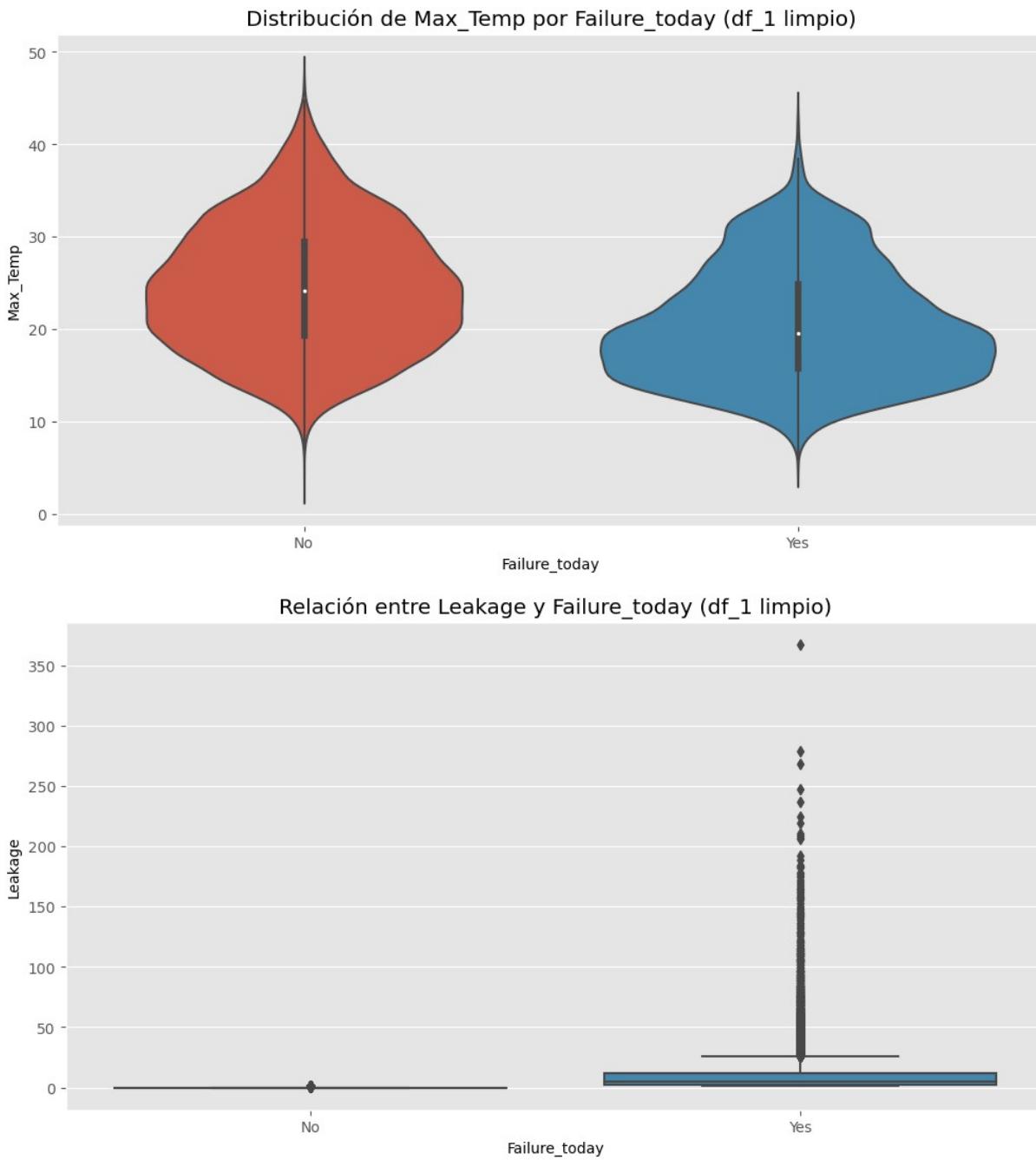
    # Boxplot comparativo
    sns.boxplot(x='Failure_today', y=col, data=df_1)
    plt.title(f'Relación entre {col} y Failure_today (df_1 limpio)')
    plt.show()

    # Violin plot para ver mejor la distribución
    plt.figure(figsize=figsize)
    sns.violinplot(x='Failure_today', y=col, data=df_1)
    plt.title(f'Distribución de {col} por Failure_today (df_1 limpio)')
    plt.show()
```

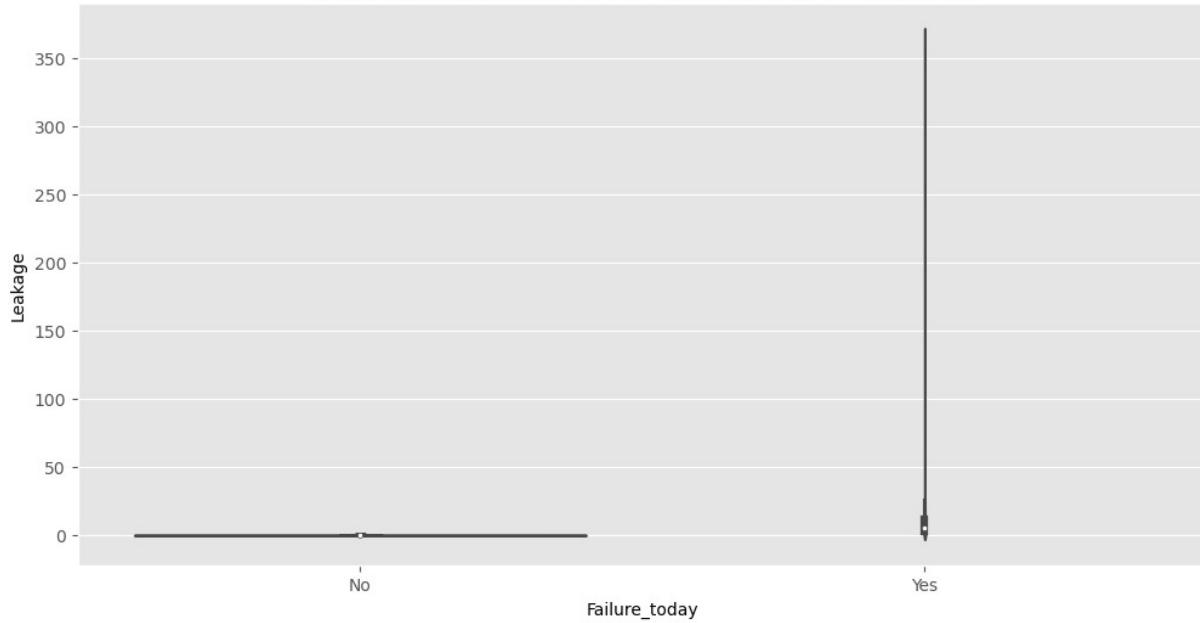




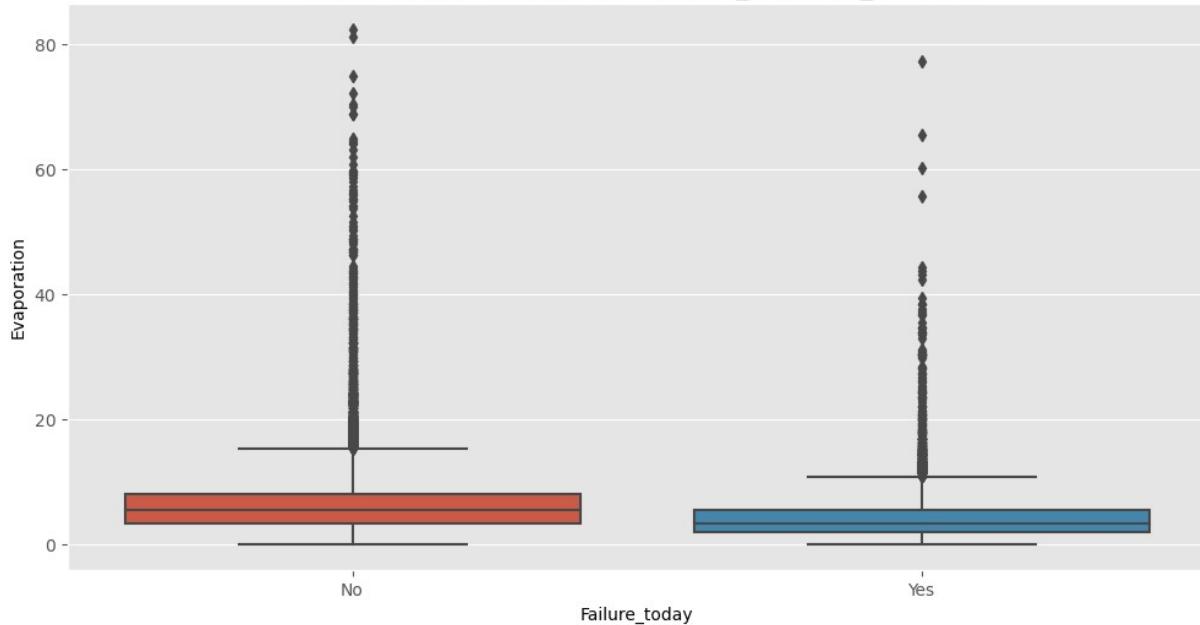


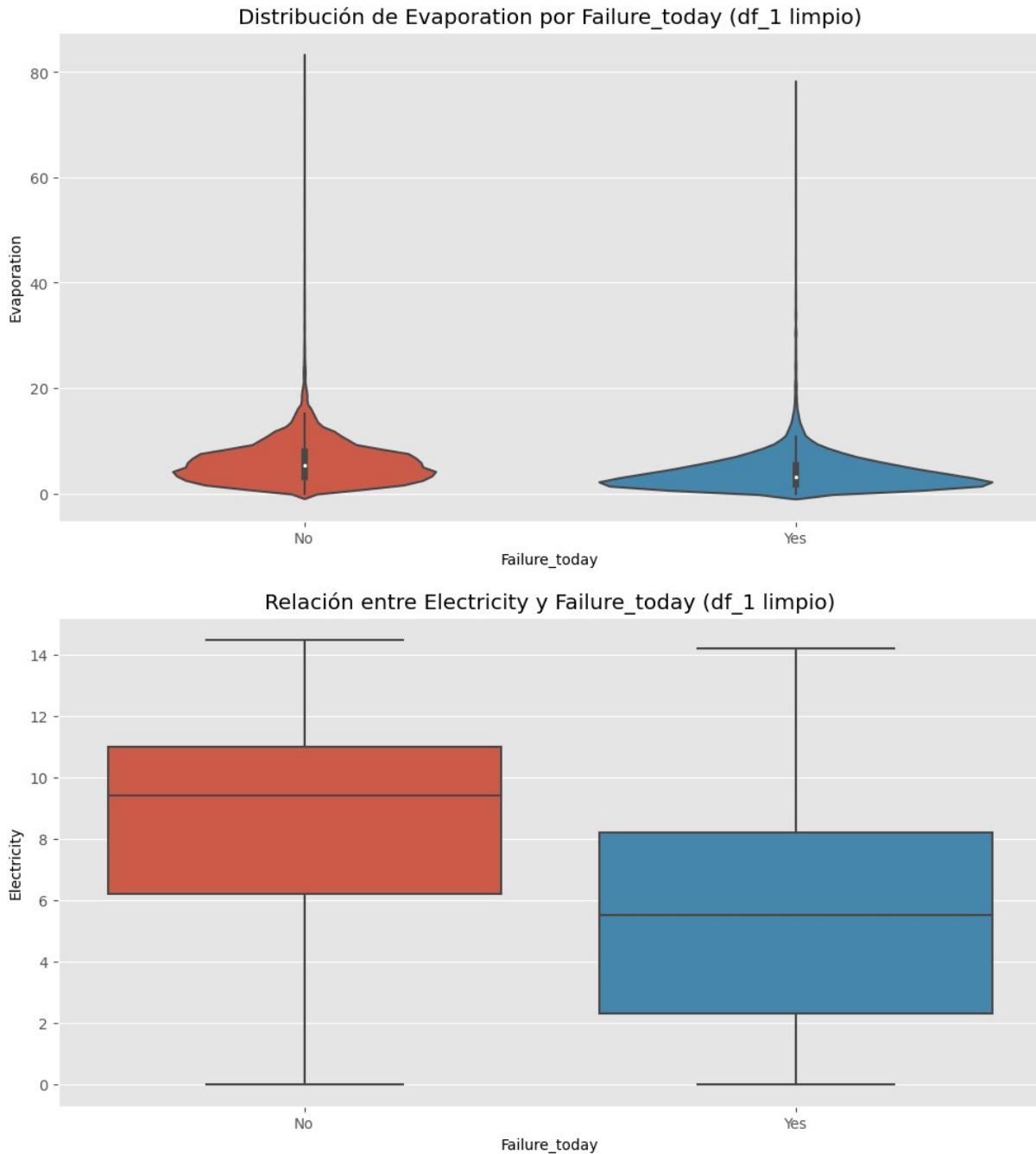


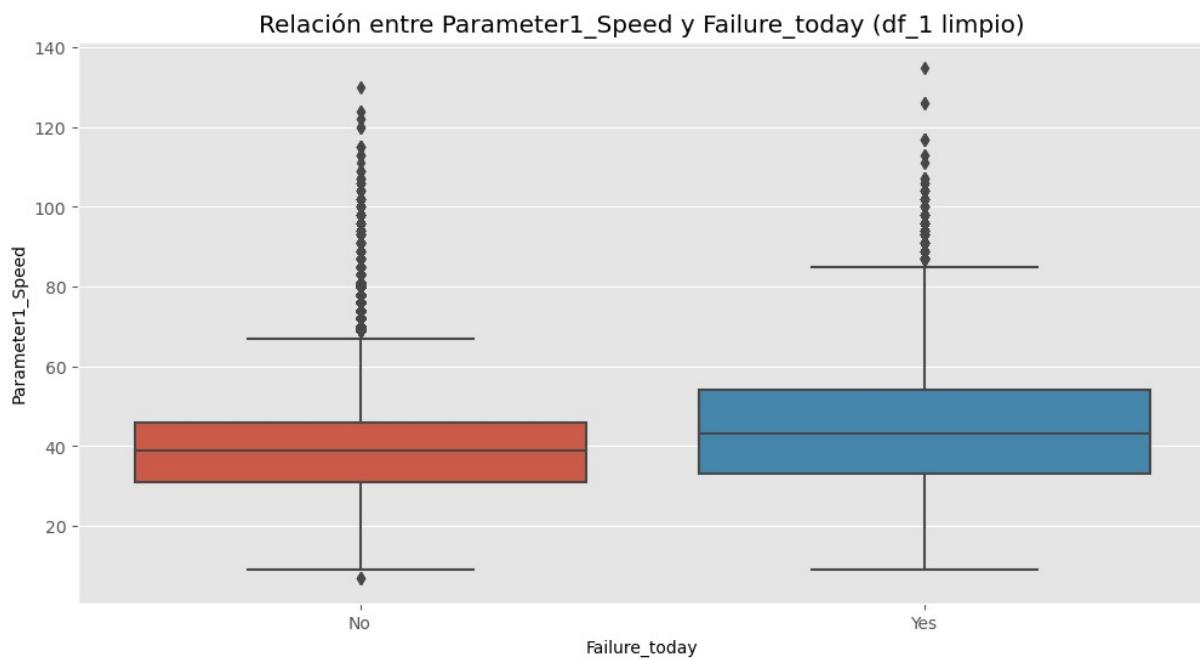
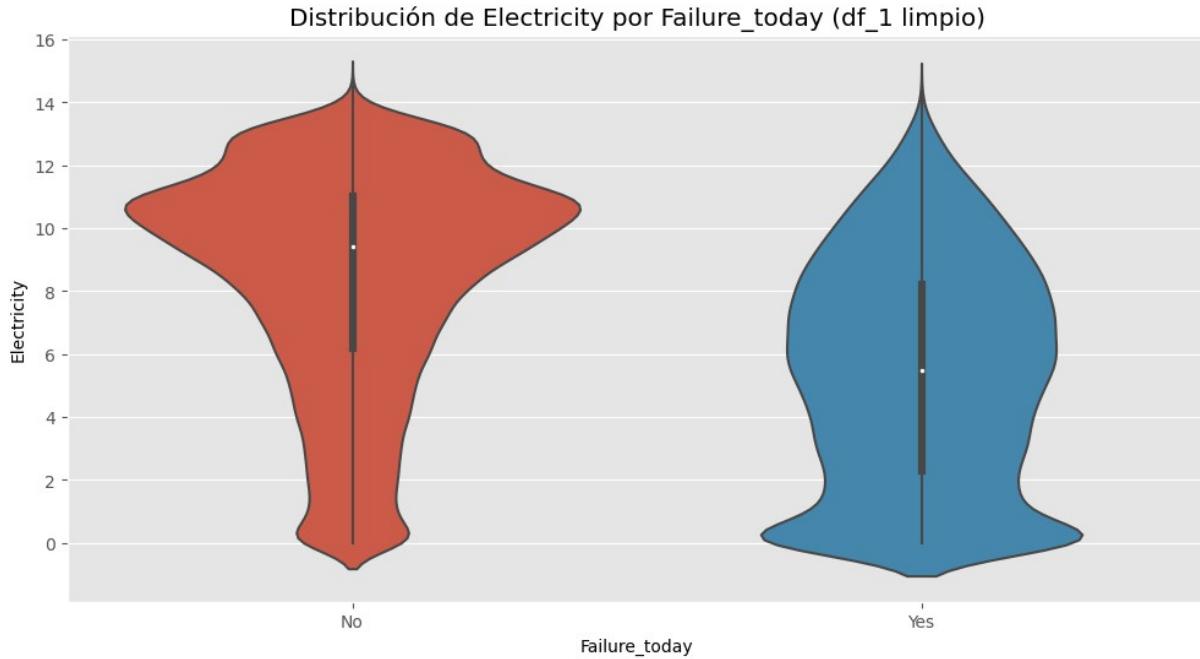
Distribución de Leakage por Failure_today (df_1 limpio)

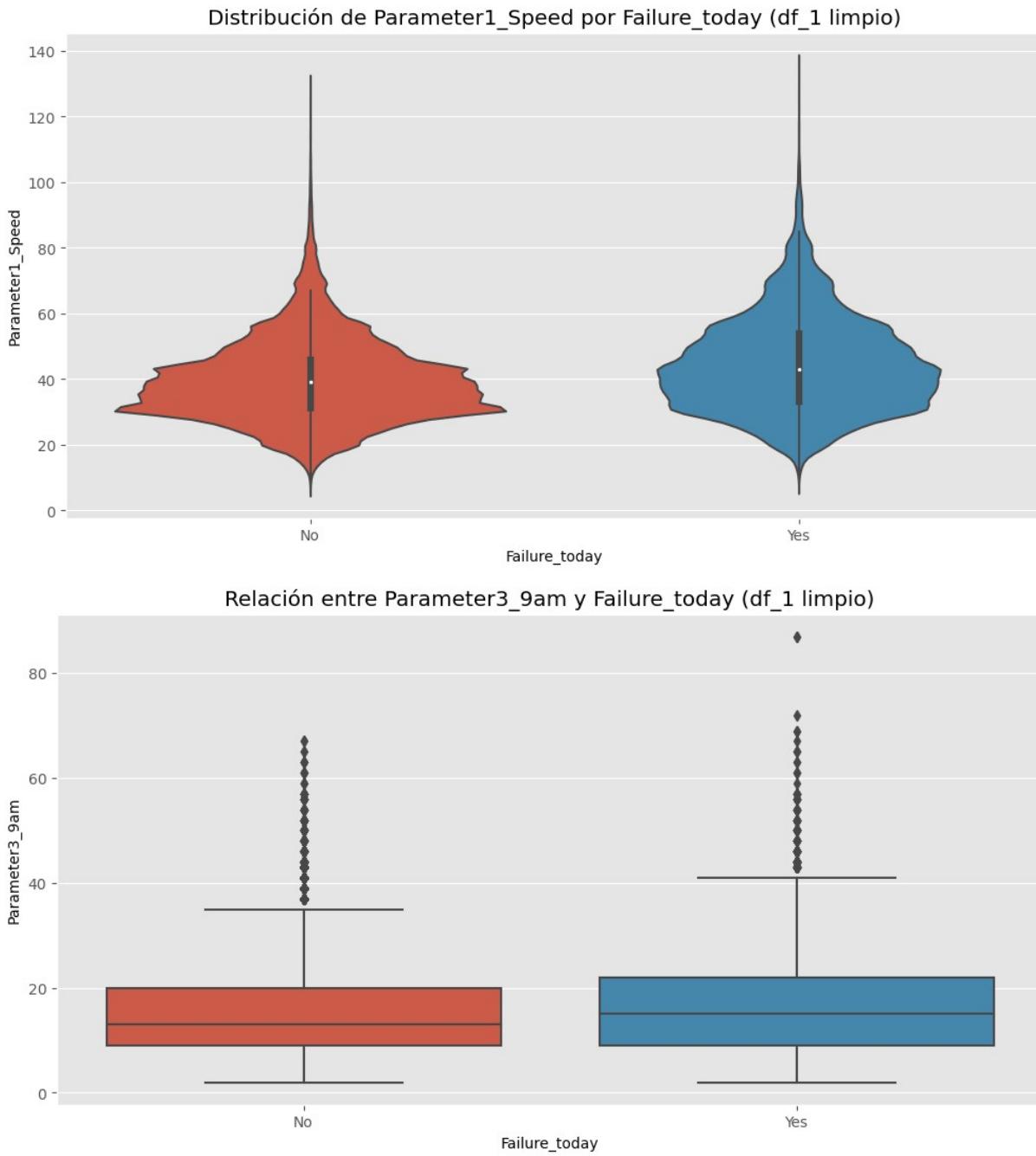


Relación entre Evaporation y Failure_today (df_1 limpio)

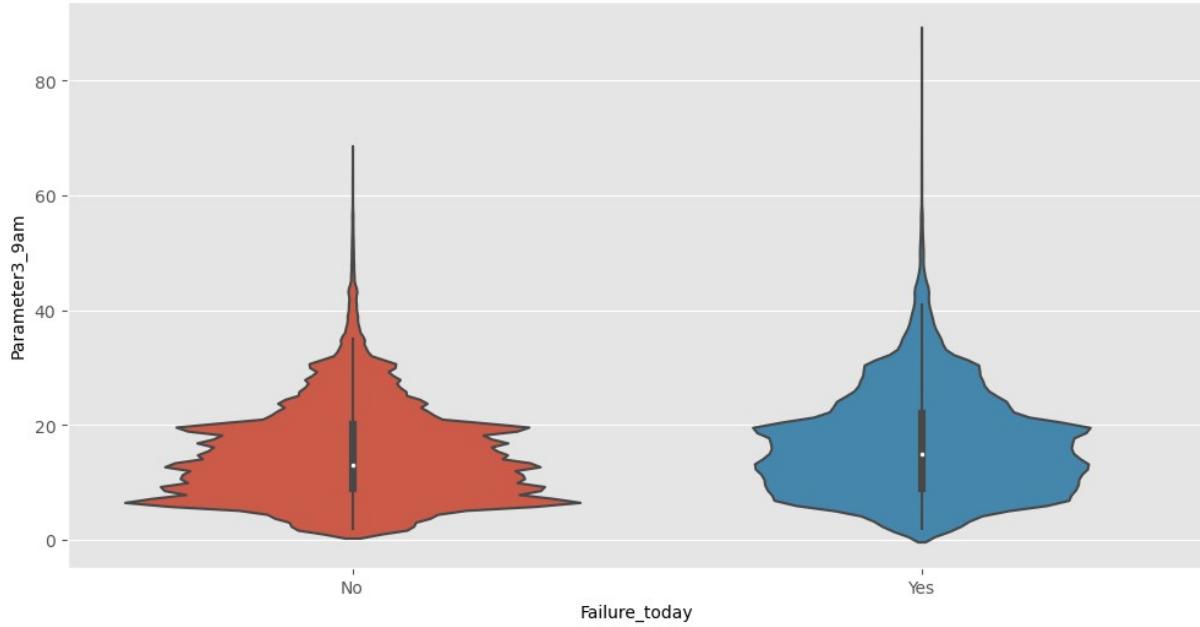




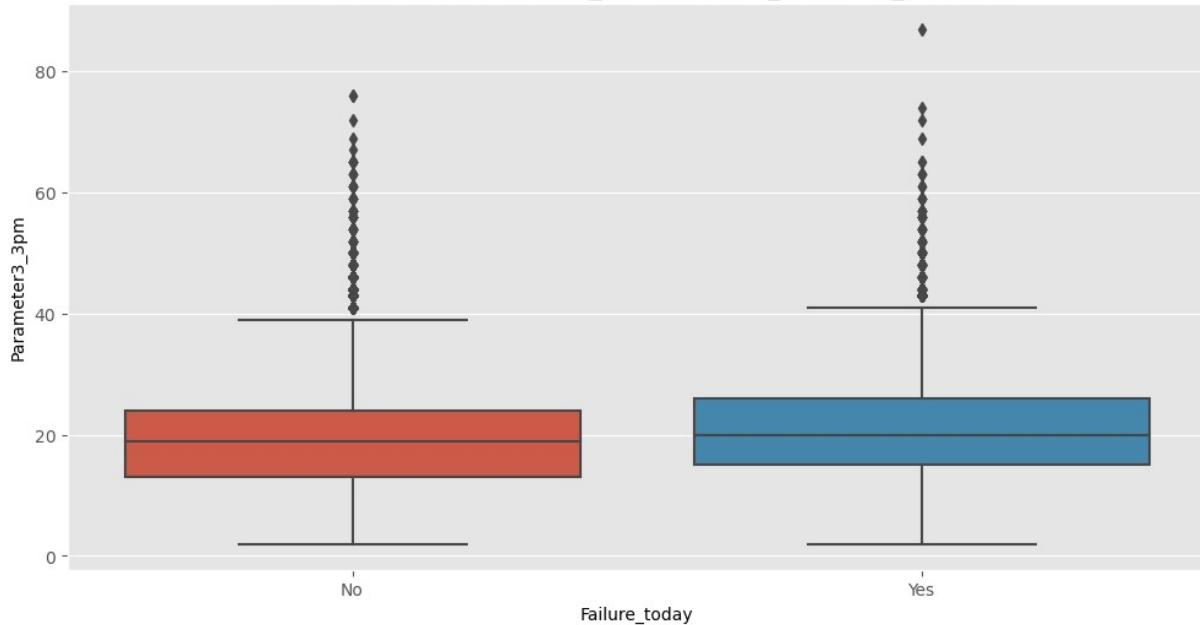




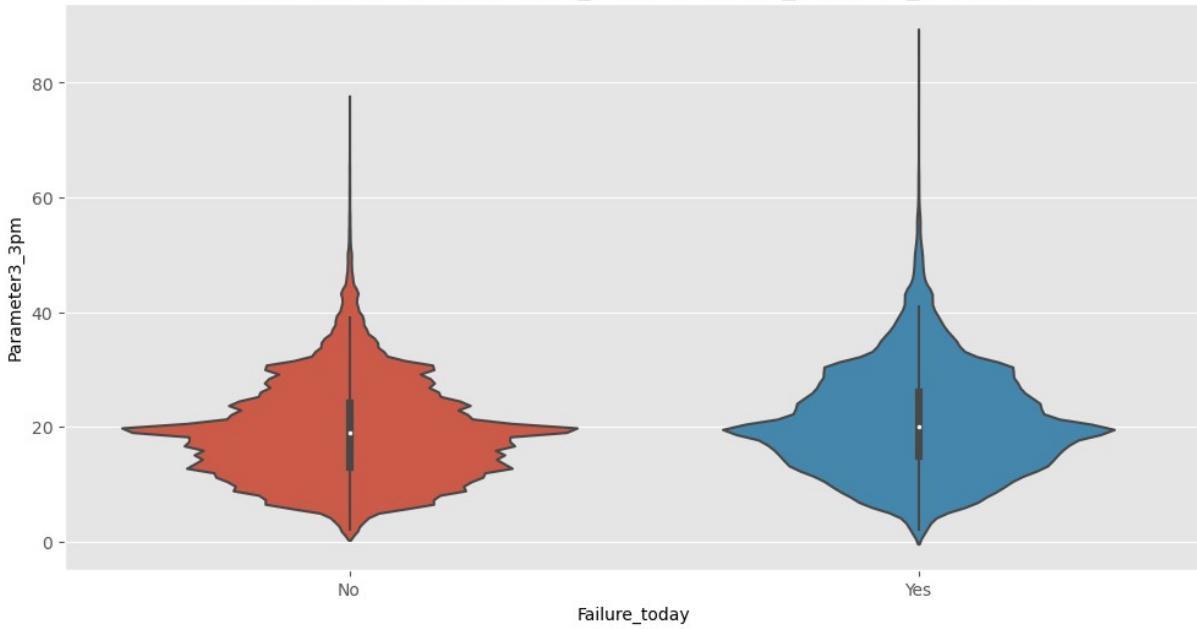
Distribución de Parameter3_9am por Failure_today (df_1 limpio)



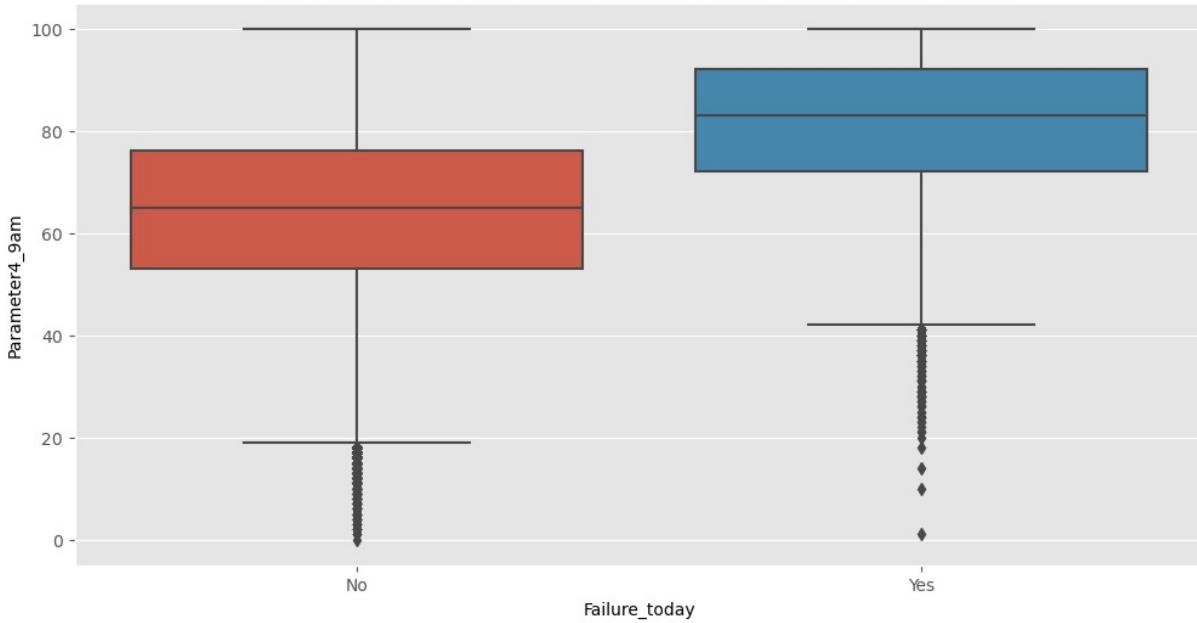
Relación entre Parameter3_3pm y Failure_today (df_1 limpio)

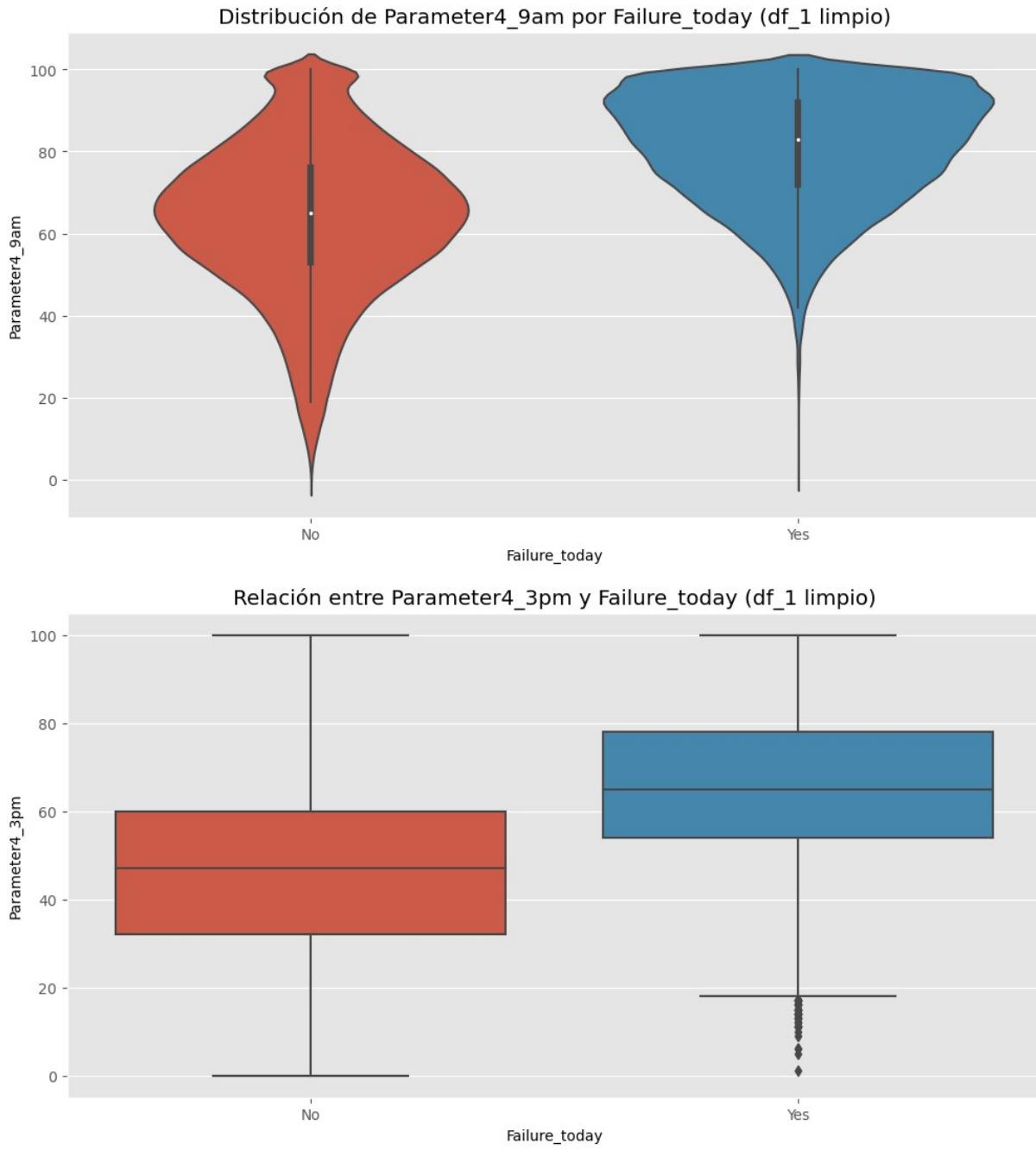


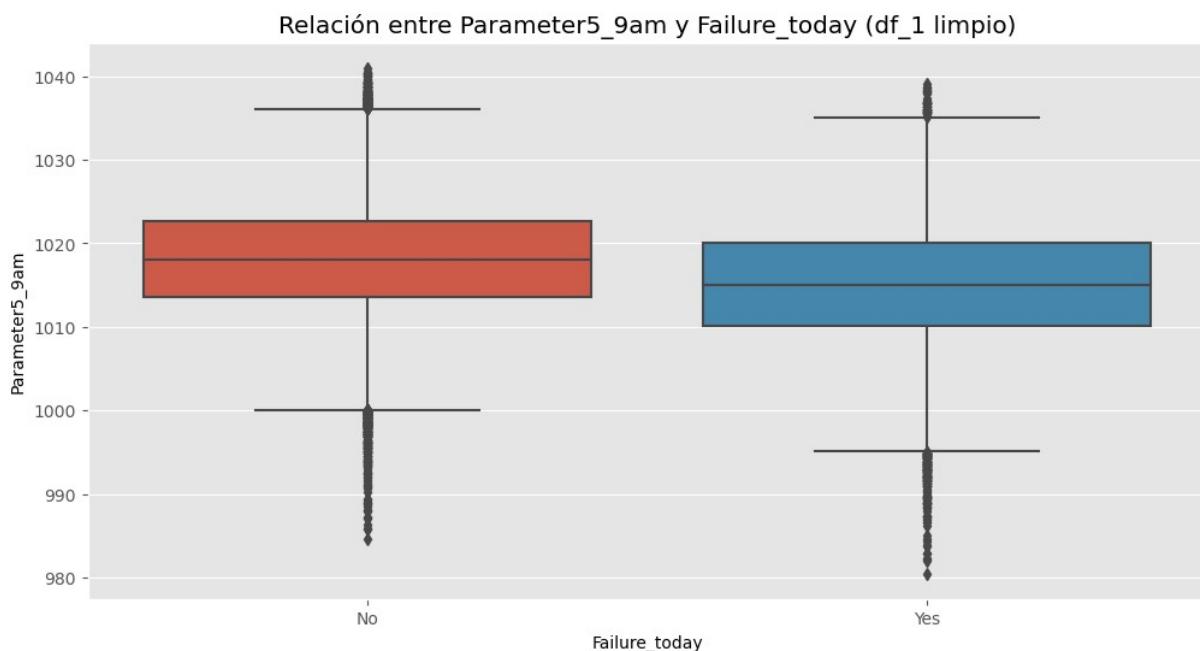
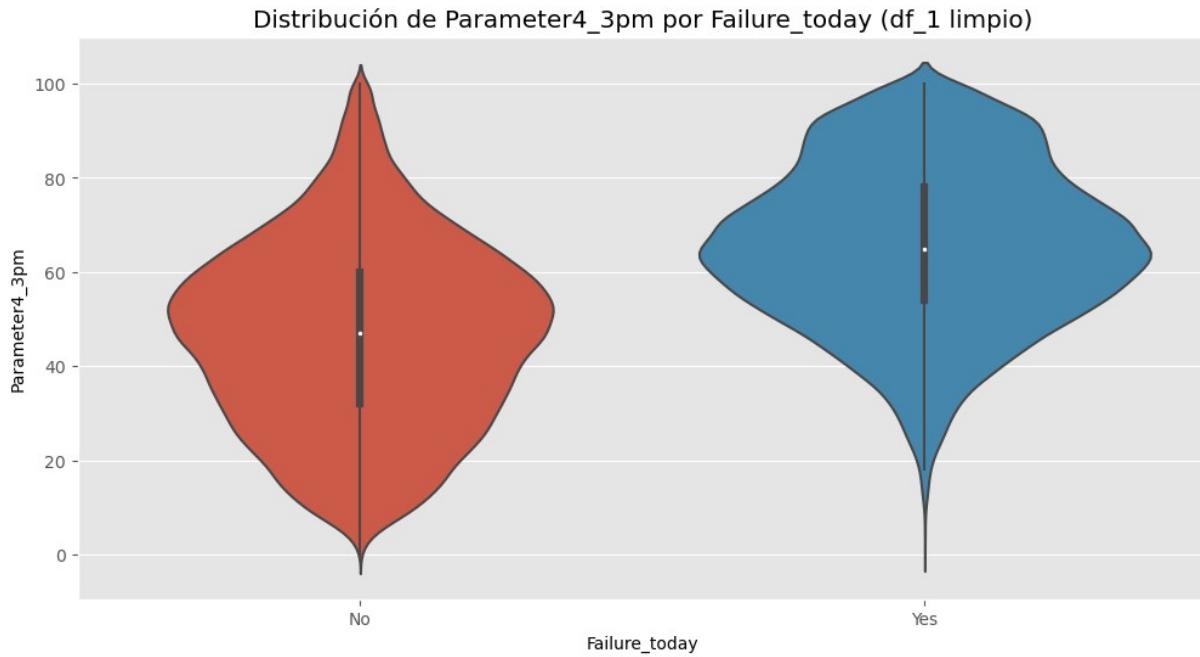
Distribución de Parameter3_3pm por Failure_today (df_1 limpio)

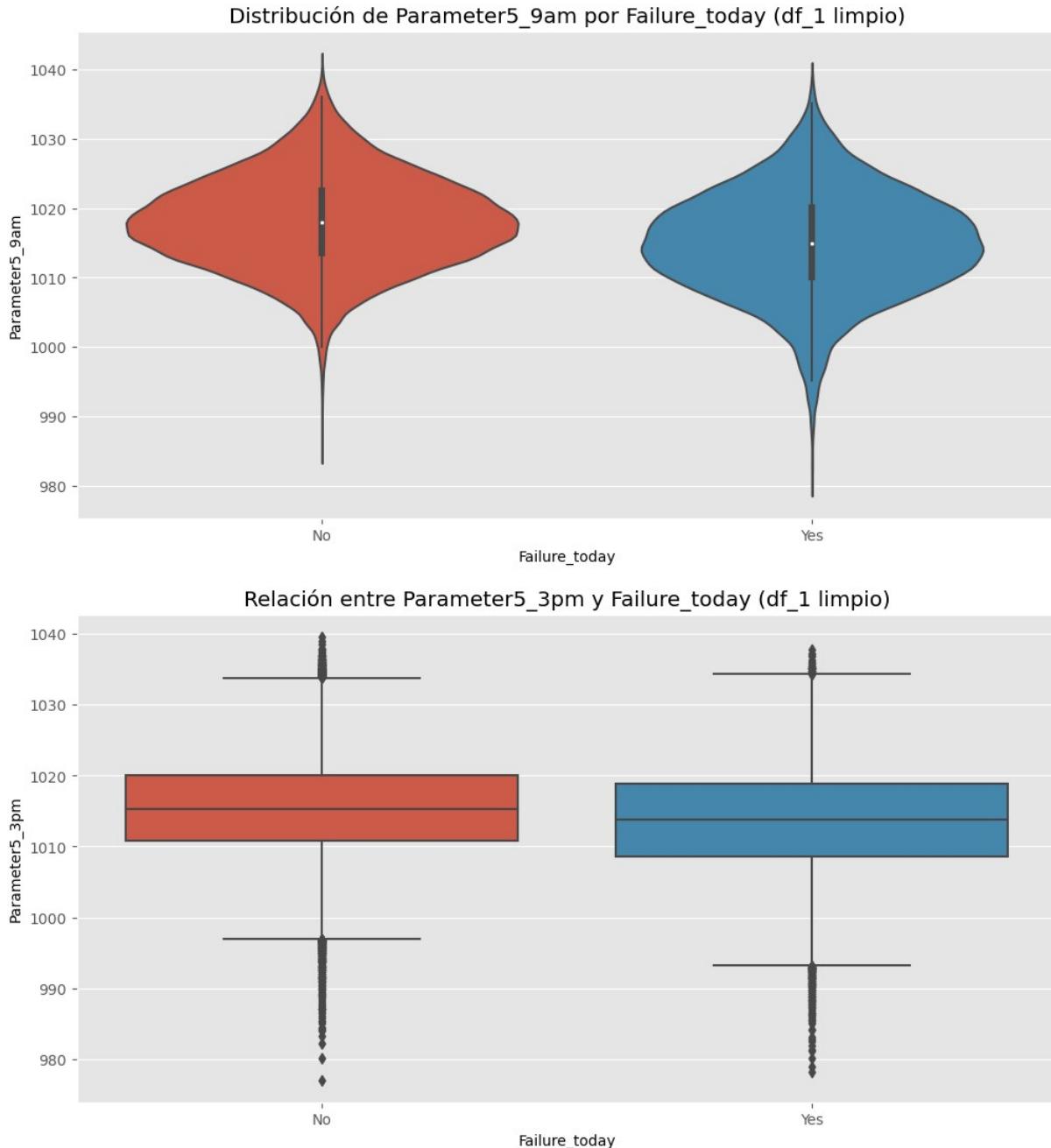


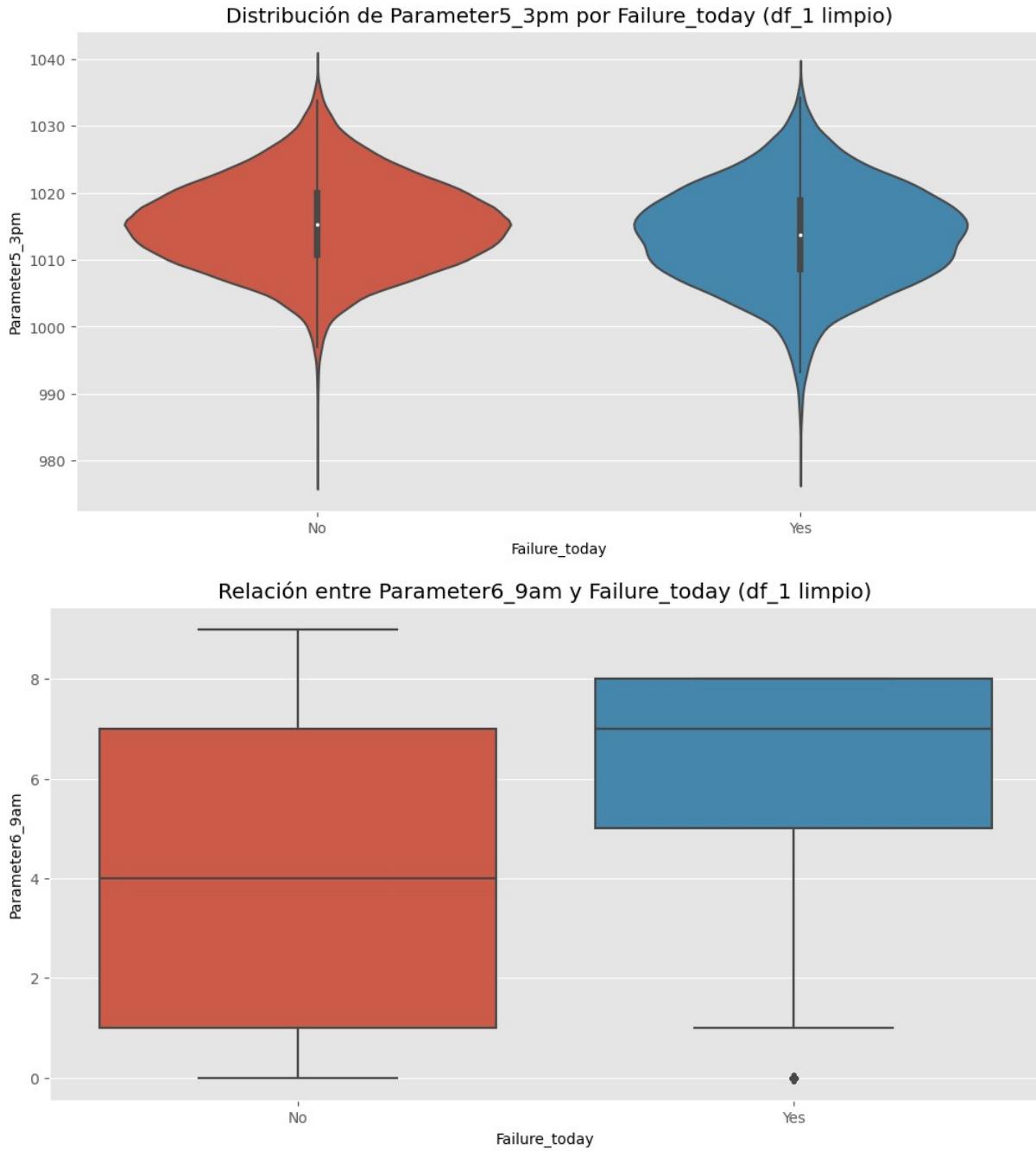
Relación entre Parameter4_9am y Failure_today (df_1 limpio)

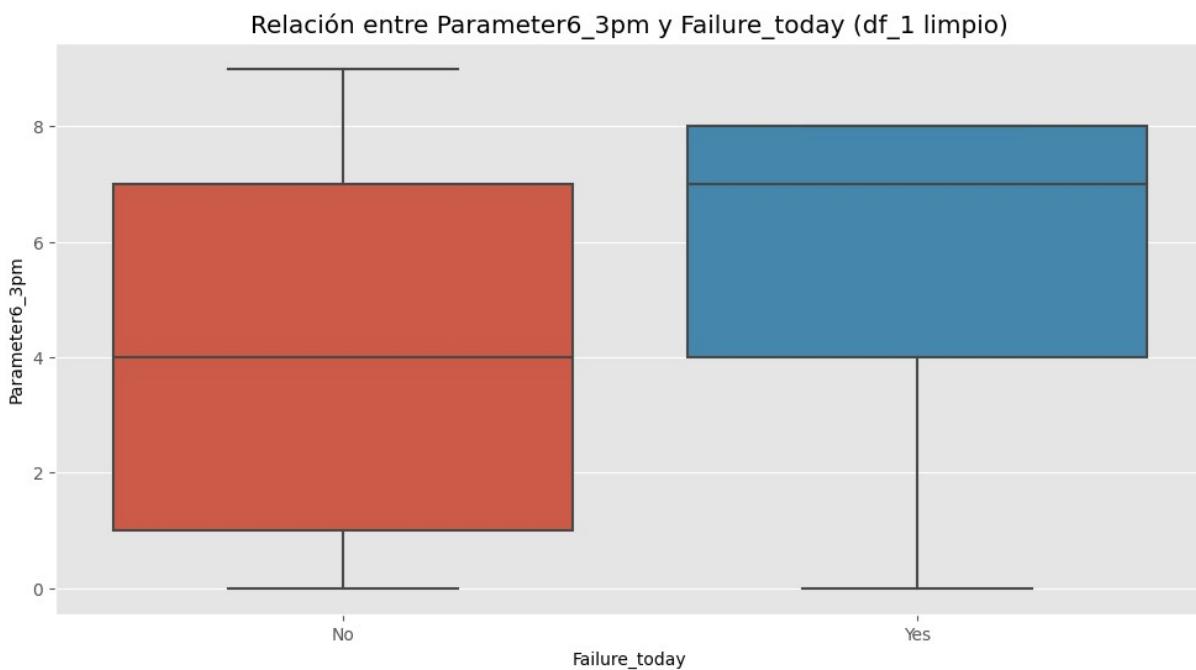
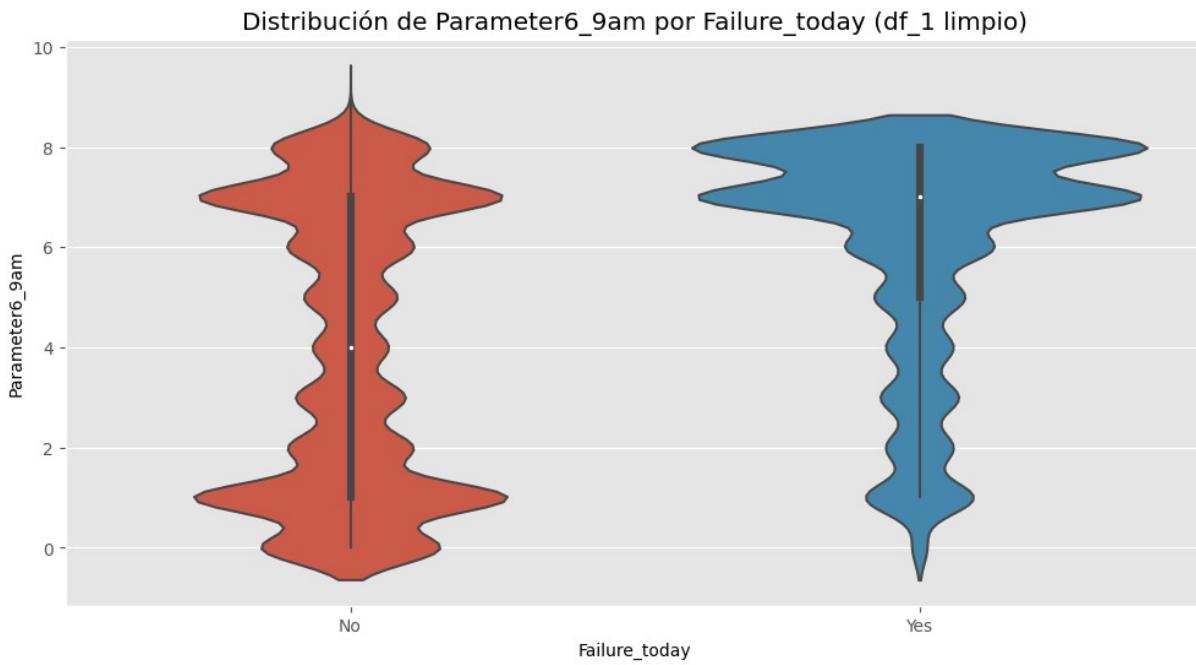


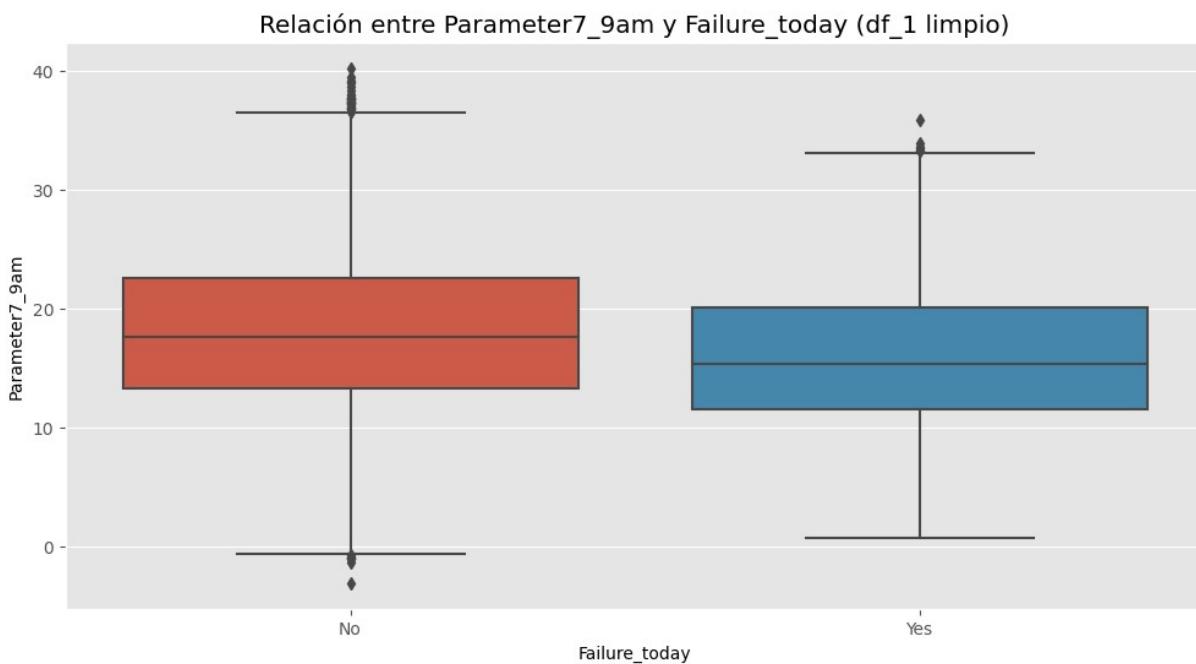
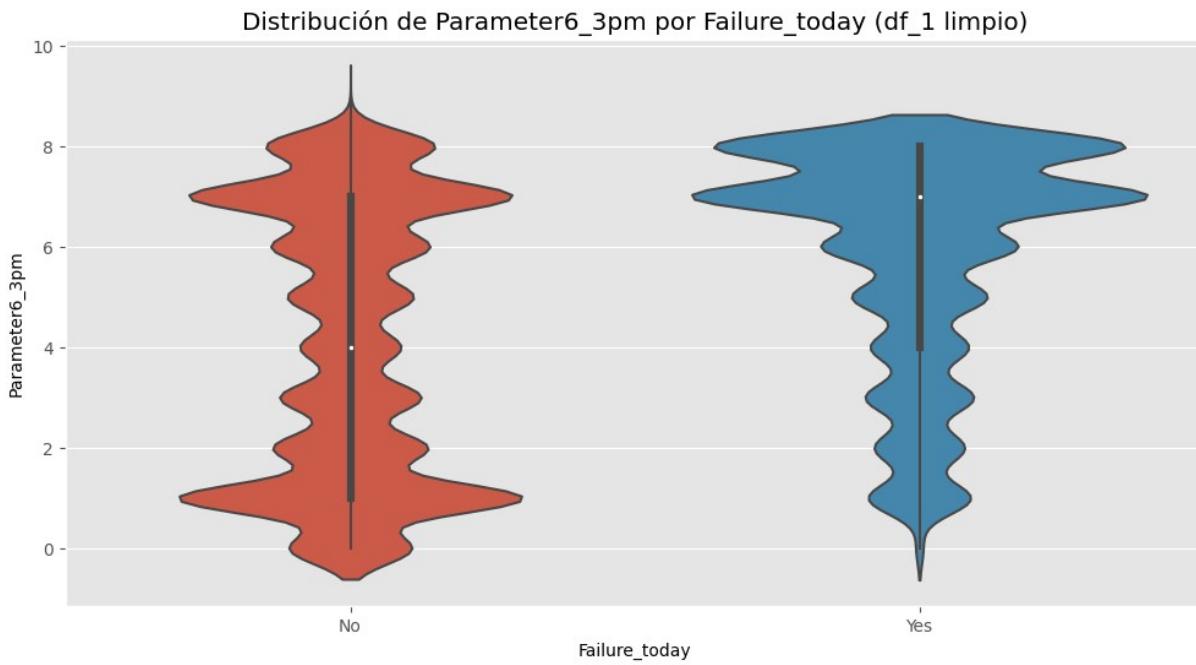








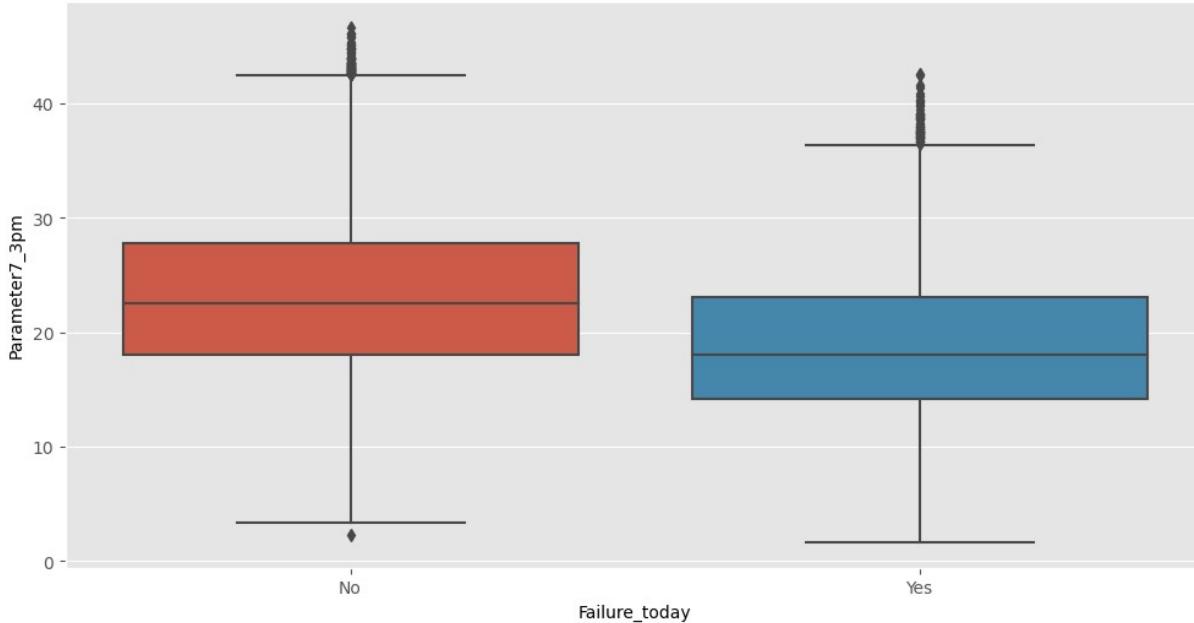


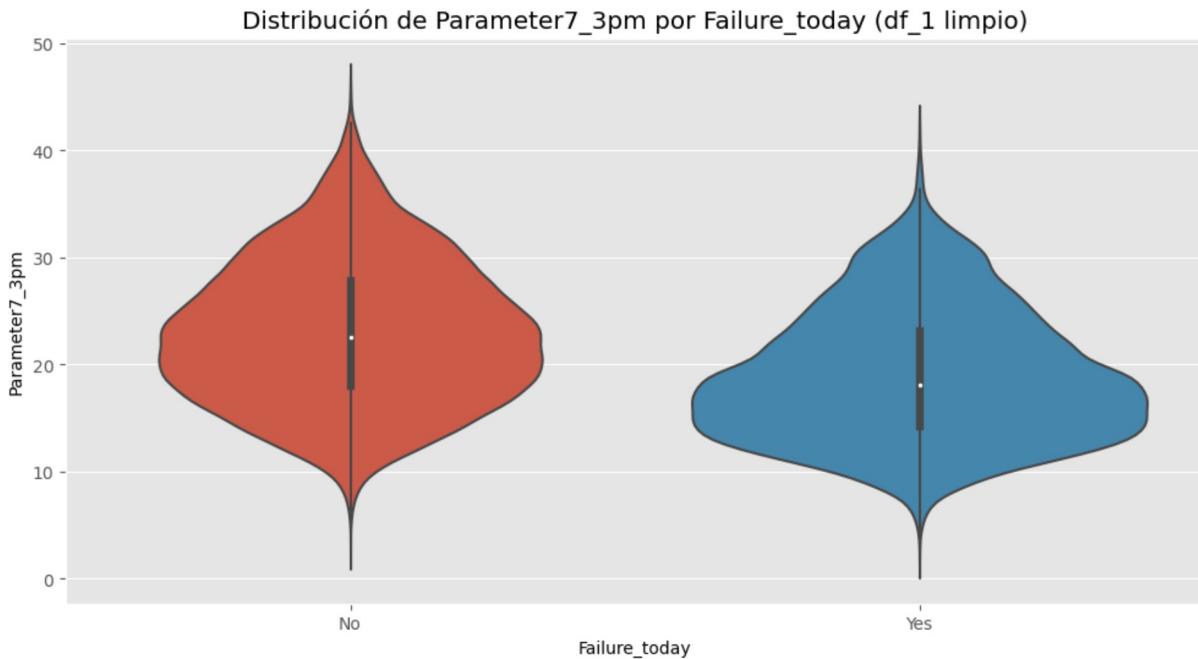


Distribución de Parameter7_9am por Failure_today (df_1 limpio)



Relación entre Parameter7_3pm y Failure_today (df_1 limpio)





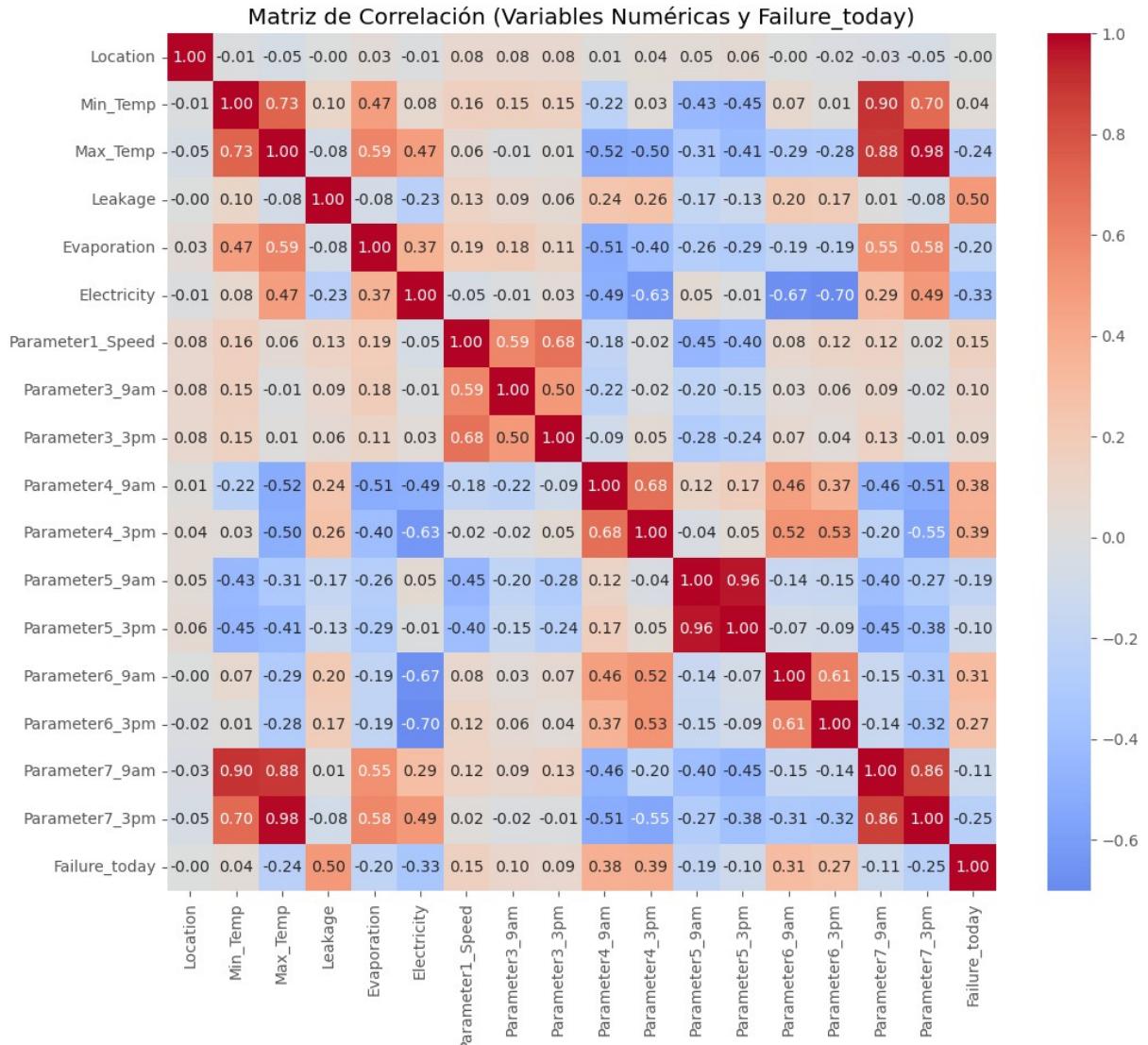
```
In [12]: df_1['Failure_today'] = df_1['Failure_today'].map({'Yes': 1, 'No': 0})
# Crear una copia del dataframe para no modificar el original
df_corr = df_1.copy()

# Convertir Failure_today a numérico (0 y 1)
df_corr['Failure_today'] = df_corr['Failure_today'].astype(int)

# Calcular matriz de correlación
corr_matrix = df_corr[numeric_cols + ['Failure_today']].corr()

# Visualizar matriz de correlación
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0, fmt='.2f')
plt.title('Matriz de Correlación (Variables Numéricas y Failure_today)')
plt.show()

# Mostrar solo las correlaciones con Failure_today
failure_corr = corr_matrix['Failure_today'].sort_values(ascending=False)
print("Correlación con Failure_today:")
print(failure_corr)
```



Correlación con Failure_today:

```

Failure_today      1.000000
Leakage           0.497554
Parameter4_3pm    0.386619
Parameter4_9am    0.378306
Parameter6_9am    0.307084
Parameter6_3pm    0.273361
Parameter1_Speed   0.154943
Parameter3_9am    0.099773
Parameter3_3pm    0.087702
Min_Temp          0.044605
Location          -0.000760
Parameter5_3pm    -0.104902
Parameter7_9am    -0.113986
Parameter5_9am    -0.188677
Evaporation       -0.197144
Max_Temp          -0.242983
Parameter7_3pm    -0.248650
Electricity        -0.332737
Name: Failure_today, dtype: float64

```

Selección de Variables Numéricas

Análisis Basado en la Matriz de Correlación

1. Variables de Temperatura

- Los 7 parámetros relacionados con temperatura (máxima y mínima) están altamente correlacionados.
- Para evitar **multicolinealidad**, se selecciona únicamente `Min_temp` (correlación positiva con `failure_today`).
- Se descartan `Max_temp` y variables similares.

2. Parámetro 6

- Relevante pero con **excesivos valores nulos** (no usable en preguntas 2-4).
- Reservado para análisis posteriores (preguntas 6-8).

3. Variables Categóricas

- `Location` es categórica (no numérica) con demasiadas categorías.
- Excluida por complejidad al generar *dummies*.

4. Otras Variables con Missing Values

- `Electricity` y `Evaporation` tienen muchos **NaN** (no se incluyen en esta etapa).

Variables Finales Seleccionadas

Variable	Razón de Selección
Parametro_1	Baja correlación con otras variables
Parametro_3	Contribución significativa al modelo
Parametro_5	Relación relevante con la variable objetivo
Max_temp	Mayor correlación con <code>failure_today</code>

Nota: - Esta selección optimiza el balance entre información predictiva y evitación de redundancias. -igualmente en el código se dejan comentarios para mejor compresión.

```
In [13]: # 1. Crear las columnas dummy
df_1['Evaporation_is_nan'] = df_1['Evaporation'].isna().astype(int)
df_1['Electricity_is_nan'] = df_1['Electricity'].isna().astype(int)
```

Creación de variables dummy para las variables categóricas

Comenzamos a generar variables dummy para las columnas categóricas del dataset, con el fin de prepararlas para el análisis. La selección final de qué variables se usarán específicamente para responder las preguntas 2, 3 y 4 se determinará más adelante, según su relevancia y comportamiento en los modelos.

Este paso nos permite trabajar con datos estructurados y listos para su implementación en los diferentes enfoques de análisis.

In [14]:

```
import pandas as pd

# Convertir la columna 'Date' a tipo datetime
df_1['Date'] = pd.to_datetime(df_1['Date'], errors='coerce')

# Verificar que no haya NaT (errores en conversión)
if df_1['Date'].isnull().any():
    print("Hay valores que no se pudieron convertir a fecha.")
    print(df_1[df_1['Date'].isnull()])

# Extraer fecha
df_1['Month'] = df_1['Date'].dt.month
df_1['DayOfWeek'] = df_1['Date'].dt.dayofweek # 0=Lunes, 6=Domingo
df_1['Quarter'] = df_1['Date'].dt.quarter

# Crear dummies excluyendo ciertas categorías
month_dummies = pd.get_dummies(df_1['Month'], prefix='Month').drop(columns=['Month'])
day_dummies = pd.get_dummies(df_1['DayOfWeek'], prefix='Day').drop(columns=['Day_6'])
quarter_dummies = pd.get_dummies(df_1['Quarter'], prefix='Q').drop(columns=['Q_1'])

# Concatenar dummies al DataFrame original
df_1 = pd.concat([df_1, month_dummies, day_dummies, quarter_dummies], axis=1)

# Resetear el índice
df_1.reset_index(drop=True, inplace=True)

# Mostrar resultado
df_1
```

Out[14]:

	Date	Location	Min_Temp	Max_Temp	Leakage	Evaporation	Electricity	Pa
0	2008-12-01	3	13.4	22.9	0.6	NaN	NaN	NaN
1	2008-12-02	3	7.4	25.1	0.0	NaN	NaN	NaN
2	2008-12-03	3	12.9	25.7	0.0	NaN	NaN	NaN
3	2008-12-04	3	9.2	28.0	0.0	NaN	NaN	NaN
4	2008-12-05	3	17.5	32.3	1.0	NaN	NaN	NaN
...
112920	2017-06-20	42	3.5	21.8	0.0	NaN	NaN	NaN
112921	2017-06-21	42	2.8	23.4	0.0	NaN	NaN	NaN
112922	2017-06-22	42	3.6	25.3	0.0	NaN	NaN	NaN
112923	2017-06-23	42	5.4	26.9	0.0	NaN	NaN	NaN
112924	2017-06-24	42	7.8	27.0	0.0	NaN	NaN	NaN

112925 rows × 47 columns

In [15]:

```
# Columnas de dirección a revisar
columnas_direccion = [
    'Parameter1_Dir',
    'Parameter2_9am',
    'Parameter2_3pm'
]

# Mostrar frecuencia de cada valor en cada columna
for col in columnas_direccion:
    print(f"Frecuencias en {col}:")
    print(df[col].value_counts(dropna=False).sort_index()) # Incluye NaN y orden a
    print("-----")
```

Frecuencias en Parameter1_Dir:

E	9071
ENE	7992
ESE	7305
N	9033
NE	7060
NNE	6433
NNW	6561
NW	8003
S	8949
SE	9309
SSE	8993
SSW	8610
SW	8797
W	9780
WNW	8066
WSW	8901
NaN	9330

Name: Parameter1_Dir, dtype: int64

Frecuencias en Parameter2_9am:

E	9024
ENE	7735
ESE	7558
N	11393
NE	7527
NNE	7948
NNW	7840
NW	8552
S	8493
SE	9162
SSE	8966
SSW	7448
SW	8237
W	8260
WNW	7194
WSW	6843
NaN	10013

Name: Parameter2_9am, dtype: int64

Frecuencias en Parameter2_3pm:

E	8342
ENE	7724
ESE	8382
N	8667
NE	8164
NNE	6444
NNW	7733
NW	8468
S	9598
SE	10663
SSE	9142
SSW	8010
SW	9182
W	9911
WNW	8656

```
WSW      9329  
NaN      3778  
Name: Parameter2_3pm, dtype: int64
```

Manejo de variables categóricas direccionales (ej. NNE, ESE, SSW):

Para las variables relacionadas con direcciones (como viento o ubicación), generar variables dummy resultaba inviable debido a la alta cardinalidad (muchas categorías distintas como "NNE", "ESE", "SW", etc.), lo que habría generado una matriz de datos excesivamente grande y difícil de manejar.

Solución implementada:

Agrupación por dirección principal:

Simplificamos las direcciones intermedias (ej. "NNE", "ENE") a sus puntos cardinales base (Norte, Sur, Este, Oeste).

Por ejemplo:

"NNE" (Norte-Noreste) → Norte

"ESE" (Este-Sureste) → Este

"SSW" (Sur-Suroeste) → Sur

Diccionario de conversión:

Creamos un diccionario para estandarizar automáticamente estas categorías, asegurando consistencia.

Justificación:

Esta aproximación reduce la dimensionalidad sin perder información crítica.

Mantiene la interpretabilidad de los datos (ej. vientos dominantes).

Permite un análisis más eficiente sin comprometer la calidad.

Nota: Se evaluó la opción de usar codificación circular (seno/coseno) para direcciones, pero se priorizó la simplicidad dado el alcance de la tarea. ademas aqui se hizo la categorizacion de la varibale de estudio que seria Failure_today

```
In [16]: import pandas as pd
```

```
# Diccionario de agrupación por puntos cardinales  
direction_groups = {
```

```

'Norte': ['N', 'NNE', 'NNW', 'NE', 'NW'],
'Sur': ['S', 'SSE', 'SSW', 'SE', 'SW'],
'Este': ['E', 'ENE', 'ESE'],
'Oeste': ['W', 'WNW', 'WSW']
}

# Función para agrupar direcciones
def agrupar_direccion(val):
    for grupo, valores in direction_groups.items():
        if val in valores:
            return grupo
    return val # En caso de que aparezca algo inesperado

# Columnas de dirección a transformar
columnas_direccion = ['Parameter1_Dir', 'Parameter2_9am', 'Parameter2_3pm']

# Aplicar la agrupación a todas las columnas de dirección
for col in columnas_direccion:
    df_1[col] = df_1[col].map(agrupar_direccion)

# Crear dummies para las columnas agrupadas (y eliminar una por columna para evitar
dummies = pd.get_dummies(df_1[columnas_direccion], prefix=columnas_direccion, drop_)

# Unir las dummies al dataframe original
df_1 = pd.concat([df_1.drop(columns=columnas_direccion), dummies], axis=1)

# Mostrar el DataFrame resultante
df_1

```

Out[16]:

	Date	Location	Min_Temp	Max_Temp	Leakage	Evaporation	Electricity	Pa
0	2008-12-01	3	13.4	22.9	0.6	NaN	NaN	NaN
1	2008-12-02	3	7.4	25.1	0.0	NaN	NaN	NaN
2	2008-12-03	3	12.9	25.7	0.0	NaN	NaN	NaN
3	2008-12-04	3	9.2	28.0	0.0	NaN	NaN	NaN
4	2008-12-05	3	17.5	32.3	1.0	NaN	NaN	NaN
...
112920	2017-06-20	42	3.5	21.8	0.0	NaN	NaN	NaN
112921	2017-06-21	42	2.8	23.4	0.0	NaN	NaN	NaN
112922	2017-06-22	42	3.6	25.3	0.0	NaN	NaN	NaN
112923	2017-06-23	42	5.4	26.9	0.0	NaN	NaN	NaN
112924	2017-06-24	42	7.8	27.0	0.0	NaN	NaN	NaN

112925 rows × 53 columns

Selección de variables categóricas significativas para el modelo binario

Para optimizar la inclusión de variables dummy en el modelo sin depender únicamente de un proceso iterativo manual, exploramos herramientas estadísticas que permiten evaluar su relevancia de manera más eficiente:

Métodos utilizados: Prueba de Chi-cuadrado (χ^2)

Evaluamos la independencia entre cada variable categórica y la variable objetivo binaria.

Variables con un p-valor < 0.05 se consideraron significativas.

Odds Ratio (OR)(visto en clases)

Calculamos la razón de momios para identificar categorías con mayor asociación con el evento de interés.

Variables con un OR significativamente distinto de 1 (y con intervalos de confianza estrechos) se priorizaron.

Resultados y ajustes: Aunque estos métodos redujeron el número de variables candidatas, 3-4 variables no significativas (según los criterios) se incluyeron inicialmente debido a:

Su relevancia teórica en el contexto del problema.

Posibles interacciones no capturadas por las pruebas univariadas.

```
In [17]: from scipy.stats import chi2_contingency
import statsmodels.api as sm
import numpy as np
import pandas as pd

# Identificar todas las columnas dummy creadas
dummy_columns = []
dummy_columns.extend([col for col in df_1.columns if col.startswith('Month_')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Day_')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Q_')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Parameter1_Dir')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Parameter2_9am')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Parameter2_3pm')])

# Crear un DataFrame para almacenar los resultados
results = pd.DataFrame(columns=['Variable', 'Chi2 p-value', 'Odds Ratio'])

# Aplicar los tests a cada dummy
for dummy in dummy_columns:
    # Test Chi-cuadrado
    tabla = pd.crosstab(df_1[dummy], df_1['Failure_today'])
    chi2, p_valor, _, _ = chi2_contingency(tabla)
```

```
# Odds Ratio (solo si hay suficientes datos)
try:
    modelo_logit = sm.Logit(df_1['Failure_today'], sm.add_constant(df_1[[dummy]])
    odds_ratio = np.exp(modelo_logit.params[dummy]))
except:
    odds_ratio = np.nan

# Almacenar resultados
results.loc[len(results)] = [dummy, p_valor, odds_ratio]

# Ordenar por significancia estadística (p-value)
results = results.sort_values(by='Chi2 p-value')

# Mostrar resultados
print("Resultados de los tests para todas las variables dummy:")
pd.set_option('display.max_rows', None)
display(results)
pd.reset_option('display.max_rows')
```

Resultados de los tests para todas las variables dummy:

	Variable	Chi2 p-value	Odds Ratio
24	Parameter2_9am_Oeste	0.000000e+00	1.956964
21	Parameter1_Dir_Oeste	1.087271e-209	1.666845
26	Parameter2_3pm_Norte	3.732449e-179	0.615252
27	Parameter2_3pm_Oeste	8.071763e-162	1.571575
20	Parameter1_Dir_Norte	2.989143e-138	0.653426
28	Parameter2_3pm_Sur	2.435070e-98	1.362444
22	Parameter1_Dir_Sur	2.330060e-82	1.327251
25	Parameter2_9am_Sur	6.809336e-81	1.327365
23	Parameter2_9am_Norte	9.231717e-59	0.775897
18	Q_3	2.609538e-44	1.253384
5	Month_7	2.828689e-38	1.376350
4	Month_6	4.619783e-26	1.296866
19	Q_4	3.513964e-23	0.845700
6	Month_8	2.091357e-18	1.243861
8	Month_10	6.716885e-13	0.825566
17	Q_2	1.125977e-11	1.116952
0	Month_2	1.108608e-07	0.861802
9	Month_11	1.169636e-05	0.890614
10	Month_12	9.705972e-05	0.899484
11	Day_0	1.900688e-01	1.026925
3	Month_5	2.314419e-01	1.030927
2	Month_4	2.708752e-01	0.970871
1	Month_3	3.529954e-01	0.976790
12	Day_1	4.677568e-01	1.014984
14	Day_3	5.380242e-01	1.012811
13	Day_2	6.564999e-01	0.990798
15	Day_4	7.093167e-01	1.007871
7	Month_9	7.350432e-01	1.009178
16	Day_5	7.753310e-01	0.993931

```
In [18]: # Clasificar Las variables dummy
results['Incluir en modelo'] = results['Chi2 p-value'] < 0.05
results['Interpretación OR'] = np.where(
    results['Odds Ratio'] > 1,
    'Mayor riesgo de fallo',
    np.where(
        results['Odds Ratio'] < 1,
        'Menor riesgo de fallo',
        'Sin cambio en el riesgo'
    )
)

# Crear tabla resumen
tabla_resumen = results[['Variable', 'Chi2 p-value', 'Odds Ratio', 'Incluir en mode

# Redondear p-value y Odds Ratio, manejando NaN
tabla_resumen['Chi2 p-value'] = tabla_resumen['Chi2 p-value'].round(4)
tabla_resumen['Odds Ratio'] = tabla_resumen['Odds Ratio'].apply(lambda x: np.nan if

# Separar en dos tablas: significativas y no significativas
tabla_significativas = tabla_resumen[tabla_resumen['Incluir en modelo']].sort_value
tabla_no_significativas = tabla_resumen[~tabla_resumen['Incluir en modelo']].sort_v

# Mostrar resultados
print("*"*80)
print("VARIABLES DUMMY QUE DEBEN INCLUIRSE EN EL MODELO (p < 0.05)")
print("*"*80)
display(tabla_significativas)

print("\n" + "*"*80)
print("VARIABLES DUMMY QUE NO DEBEN INCLUIRSE EN EL MODELO (p ≥ 0.05)")
print("*"*80)
display(tabla_no_significativas)

=====
VARIABLES DUMMY QUE DEBEN INCLUIRSE EN EL MODELO (p < 0.05)
=====
```

	Variable	Chi2 p-value	Odds Ratio	Incluir en modelo	Interpretación OR
24	Parameter2_9am_Oeste	0.0000	1.96	True	Mayor riesgo de fallo
0	Month_2	0.0000	0.86	True	Menor riesgo de fallo
17	Q_2	0.0000	1.12	True	Mayor riesgo de fallo
8	Month_10	0.0000	0.83	True	Menor riesgo de fallo
6	Month_8	0.0000	1.24	True	Mayor riesgo de fallo
19	Q_4	0.0000	0.85	True	Menor riesgo de fallo
4	Month_6	0.0000	1.30	True	Mayor riesgo de fallo
5	Month_7	0.0000	1.38	True	Mayor riesgo de fallo
9	Month_11	0.0000	0.89	True	Menor riesgo de fallo
18	Q_3	0.0000	1.25	True	Mayor riesgo de fallo
25	Parameter2_9am_Sur	0.0000	1.33	True	Mayor riesgo de fallo
22	Parameter1_Dir_Sur	0.0000	1.33	True	Mayor riesgo de fallo
28	Parameter2_3pm_Sur	0.0000	1.36	True	Mayor riesgo de fallo
20	Parameter1_Dir_Norte	0.0000	0.65	True	Menor riesgo de fallo
27	Parameter2_3pm_Oeste	0.0000	1.57	True	Mayor riesgo de fallo
26	Parameter2_3pm_Norte	0.0000	0.62	True	Menor riesgo de fallo
21	Parameter1_Dir_Oeste	0.0000	1.67	True	Mayor riesgo de fallo
23	Parameter2_9am_Norte	0.0000	0.78	True	Menor riesgo de fallo
10	Month_12	0.0001	0.90	True	Menor riesgo de fallo

=====
VARIABLES DUMMY QUE NO DEBEN INCLUIRSE EN EL MODELO ($p \geq 0.05$)
=====

	Variable	Chi2	p-value	Odds Ratio	Incluir en modelo	Interpretación OR
11	Day_0	0.1901		1.03	False	Mayor riesgo de fallo
3	Month_5	0.2314		1.03	False	Mayor riesgo de fallo
2	Month_4	0.2709		0.97	False	Menor riesgo de fallo
1	Month_3	0.3530		0.98	False	Menor riesgo de fallo
12	Day_1	0.4678		1.01	False	Mayor riesgo de fallo
14	Day_3	0.5380		1.01	False	Mayor riesgo de fallo
13	Day_2	0.6565		0.99	False	Menor riesgo de fallo
15	Day_4	0.7093		1.01	False	Mayor riesgo de fallo
7	Month_9	0.7350		1.01	False	Mayor riesgo de fallo
16	Day_5	0.7753		0.99	False	Menor riesgo de fallo

In [19]: `print(df_1.columns)`

```
Index(['Date', 'Location', 'Min_Temp', 'Max_Temp', 'Leakage', 'Evaporation',
       'Electricity', 'Parameter1_Speed', 'Parameter3_9am', 'Parameter3_3pm',
       'Parameter4_9am', 'Parameter4_3pm', 'Parameter5_9am', 'Parameter5_3pm',
       'Parameter6_9am', 'Parameter6_3pm', 'Parameter7_9am', 'Parameter7_3pm',
       'Failure_today', 'Evaporation_is_nan', 'Electricity_is_nan', 'Month',
       'DayOfWeek', 'Quarter', 'Month_2', 'Month_3', 'Month_4', 'Month_5',
       'Month_6', 'Month_7', 'Month_8', 'Month_9', 'Month_10', 'Month_11',
       'Month_12', 'Day_0', 'Day_1', 'Day_2', 'Day_3', 'Day_4', 'Day_5', 'Q_2',
       'Q_3', 'Q_4', 'Parameter1_Dir_Norte', 'Parameter1_Dir_Oeste',
       'Parameter1_Dir_Sur', 'Parameter2_9am_Norte', 'Parameter2_9am_Oeste',
       'Parameter2_9am_Sur', 'Parameter2_3pm_Norte', 'Parameter2_3pm_Oeste',
       'Parameter2_3pm_Sur'],
      dtype='object')
```

In [20]: `print(df_1['Evaporation_is_nan'].value_counts())
print(df_1['Electricity_is_nan'].value_counts())`

```
0    71781
1    41144
Name: Evaporation_is_nan, dtype: int64
0    66646
1    46279
Name: Electricity_is_nan, dtype: int64
```

In [21]: `# Comprobar si hay NaNs en las variables
print(df_1.isna().sum()) # Mostrar la cantidad de NaNs por columna`

```
Date          0  
Location      0  
Min_Temp      0  
Max_Temp      0  
Leakage        0  
Evaporation   41144  
Electricity    46279  
Parameter1_Speed 0  
Parameter3_9am 0  
Parameter3_3pm 0  
Parameter4_9am 0  
Parameter4_3pm 0  
Parameter5_9am 0  
Parameter5_3pm 0  
Parameter6_9am 37310  
Parameter6_3pm 38646  
Parameter7_9am 0  
Parameter7_3pm 0  
Failure_today  0  
Evaporation_is_nan 0  
Electricity_is_nan 0  
Month          0  
DayOfWeek      0  
Quarter         0  
Month_2         0  
Month_3         0  
Month_4         0  
Month_5         0  
Month_6         0  
Month_7         0  
Month_8         0  
Month_9         0  
Month_10        0  
Month_11        0  
Month_12        0  
Day_0           0  
Day_1           0  
Day_2           0  
Day_3           0  
Day_4           0  
Day_5           0  
Q_2             0  
Q_3             0  
Q_4             0  
Parameter1_Dir_Norte 0  
Parameter1_Dir_Oeste 0  
Parameter1_Dir_Sur  0  
Parameter2_9am_Norte 0  
Parameter2_9am_Oeste 0  
Parameter2_9am_Sur  0  
Parameter2_3pm_Norte 0  
Parameter2_3pm_Oeste 0  
Parameter2_3pm_Sur  0  
dtype: int64
```

2. Ejecute un modelo de probabilidad lineal (MCO) que permita explicar la probabilidad de que un dia se reporte fallo medido por sensor, a partir de las informacion disponible. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.

```
In [22]: import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler

# Definir Las variables a excluir
variables_a_excluir = [
    "Location",
    "Parameter7_3pm", # Redundante con Min_Temp
    "Parameter7_9am", # Redundante con Min_Temp
    "Parameter5_3pm", # Redundante con Parameter5_9am
    "Max_Temp", # Redundante con Min_Temp
    "Electricity", # Redundante con Electricity_is_nan
    "Evaporation", #redundante con Evaporation_is_nan
    "DayOfWeek",
    "Quarter",
    "Month",
    "Leakage", #explica toda la data
    "Parameter6_3pm",#muchos valores nulos no manejables
    "Parameter6_9am",#muchos valores nulos no manejables
]

# Definir La variable dependiente (y) y las independientes (X)
y = df_1['Failure_today']
X = df_1.drop(columns=variables_a_excluir + ['Failure_today'], axis=1)

# Seleccionar solo columnas numéricas y las dummies significativas
# Primero identificamos las columnas numéricas
numeric_cols = X.select_dtypes(include=["float64", "int64"]).columns

# Definir Las dummies significativas que quieres incluir
dummies_significativas = [
    # Meses
    'Month_6', 'Month_7', 'Month_10', 'Month_11', 'Month_2',

    # Parámetros de dirección del viento a las 9am (Parameter2_9am)
    'Parameter2_9am_Oeste', 'Parameter2_9am_Sur', 'Parameter2_9am_Norte',

    # Parámetros de dirección del viento a las 3pm (Parameter2_3pm)
    'Parameter2_3pm_Sur', 'Parameter2_3pm_Oeste', 'Parameter2_3pm_Norte',

    # Parámetros de dirección (Parameter1_Dir)
```

```
'Parameter1_Dir_Norte', 'Parameter1_Dir_Oeste', 'Parameter1_Dir_Sur',  
  
# Variables de preguntas (Q)  
'Q_2', 'Q_3', 'Q_4',  
  
# Indicadores de valores faltantes  
'Electricity_is_nan',  
'Evaporation_is_nan',  
]  
  
# Filtrar las columnas: numéricas + dummies significativas  
columnas_a_incluir = list(numeric_cols) + [col for col in dummies_significativas if  
X = X[columnas_a_incluir]]  
  
# Estandarizar las variables continuas (numéricas)  
scaler = StandardScaler()  
X[numeric_cols] = scaler.fit_transform(X[numeric_cols])  
  
# Agregar una constante para el modelo  
X = sm.add_constant(X)  
  
# Ajustar el modelo de regresión lineal  
model = sm.OLS(y, X)  
results = model.fit(cov_type='HC0') # Usar errores estándar robustos  
  
# Mostrar el resumen de los resultados  
print(results.summary())
```

OLS Regression Results

```
=====
Dep. Variable: Failure_today R-squared: 0.264
Model: OLS Adj. R-squared: 0.263
Method: Least Squares F-statistic: 1609.
Date: Thu, 24 Apr 2025 Prob (F-statistic): 0.00
Time: 23:14:26 Log-Likelihood: -44282.
No. Observations: 112925 AIC: 8.862e+04
Df Residuals: 112898 BIC: 8.888e+04
Df Model: 26
Covariance Type: HC0
=====
```

====

	coef	std err	z	P> z	[0.025
0.975]					

const	0.1458	0.004	37.199	0.000	0.138
0.153					
Min_Temp	0.0312	0.002	20.437	0.000	0.028
0.034					
Parameter1_Speed	0.0649	0.002	35.083	0.000	0.061
0.069					
Parameter3_9am	0.0347	0.001	25.030	0.000	0.032
0.037					
Parameter3_3pm	-0.0408	0.002	-26.764	0.000	-0.044
-0.038					
Parameter4_9am	0.1422	0.002	86.380	0.000	0.139
0.145					
Parameter4_3pm	0.0590	0.002	35.601	0.000	0.056
0.062					
Parameter5_9am	-0.0653	0.002	-43.163	0.000	-0.068
-0.062					
Month_6	-0.0017	0.005	-0.351	0.726	-0.011
0.008					
Month_7	-0.0140	0.005	-2.894	0.004	-0.024
-0.005					
Month_10	0.0495	0.005	9.223	0.000	0.039
0.060					
Month_11	0.0239	0.005	4.532	0.000	0.014
0.034					
Month_2	-0.0147	0.004	-3.277	0.001	-0.023
-0.006					
Parameter2_9am_Oeste	0.0528	0.004	13.316	0.000	0.045
0.061					
Parameter2_9am_Sur	0.0381	0.003	12.517	0.000	0.032
0.044					
Parameter2_9am_Norte	-0.0116	0.003	-3.795	0.000	-0.018
-0.006					
Parameter2_3pm_Sur	0.0424	0.003	12.216	0.000	0.036
0.049					
Parameter2_3pm_Oeste	0.0487	0.004	11.686	0.000	0.041
0.057					
Parameter2_3pm_Norte	-0.0183	0.004	-5.121	0.000	-0.025
-0.011					
Parameter1_Dir_Norte	-0.0246	0.004	-6.865	0.000	-0.032

-0.018						
Parameter1_Dir_Oeste	0.0098	0.004	2.304	0.021	0.001	
0.018						
Parameter1_Dir_Sur	0.0062	0.003	1.801	0.072	-0.001	
0.013						
Q_2	0.0556	0.004	14.537	0.000	0.048	
0.063						
Q_3	0.1019	0.004	24.835	0.000	0.094	
0.110						
Q_4	0.0327	0.005	7.247	0.000	0.024	
0.042						
Electricity_is_nan	-0.0124	0.003	-3.767	0.000	-0.019	
-0.006						
Evaporation_is_nan	-0.0008	0.003	-0.236	0.813	-0.007	
0.006						
<hr/>						
Omnibus:	9737.590	Durbin-Watson:	1.769			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	12352.226			
Skew:	0.804	Prob(JB):	0.00			
Kurtosis:	2.803	Cond. No.	11.5			
<hr/>						

Notes:

[1] Standard Errors are heteroscedasticity robust (HC0)

R:

Interpretación del Modelo de Regresión OLS: Coeficientes Marginales

Visión General del Modelo

- **Variable Dependiente:** Failure_today
 - **R²:** 0.264 (26.4% de la variabilidad explicada)
 - **Método:** Mínimos Cuadrados Ordinarios (OLS) con errores robustos (HC0)
 - **Nº Observaciones:** 112,925
-

Análisis de Coeficientes Significativos

1. Variables Climáticas

Variable	Coef.	Interpretación
Min_Temp	0.0312	11°C en temp. mínima → 13.12% probabilidad de falla
Parameter1_Speed	0.0649	11 unidad en velocidad → 16.49% probabilidad de falla
Parameter3_9am	0.0347	11 unidad a las 9am → 13.47% probabilidad

Variable	Coef.	Interpretación
Parameter3_3pm	-0.0408	↓1 unidad a las 3pm → ↓4.08% probabilidad
Parameter4_9am	0.1422	↑1 unidad a las 9am → ↑14.22% (efecto mayor)

2. Efectos Estacionales

Variable	Coef.	Interpretación
Month_10	0.0495	Octubre → ↑4.95% vs meses base
Q_3	0.1019	3er trimestre → ↑10.19% probabilidad

3. Dirección del Viento

Variable	Coef.	Interpretación
Parameter1_Dir_Norte	-0.0246	Viento norte → ↓2.46% probabilidad
Parameter1_Dir_Oeste	0.0098	Viento oeste → ↑0.98% probabilidad

4. Calidad de Datos

Variable	Coef.	Interpretación
Electricity_is_nan	-0.0124	Datos faltantes → ↓1.24% probabilidad

Conclusiones Clave

1. Mayores impactos positivos:

- Parameter4_9am (+14.22%)
- Q_3 (+10.19%)
- Parameter1_Speed (+6.49%)

2. **Efectos horarios:** Variables como Parameter3 y Parameter4 muestran comportamientos opuestos en 9am vs 3pm.

3. **Robustez:** Todos los coeficientes mencionados son significativos ($p\text{-value} < 0.05$), excepto Evaporation_is_nan ($p=0.813$), Parameter1_Dir_Sur ($p=0.072$) y Month_6 ($p=0.726$)

3. Ejecute un modelo *probit* para responder a la pregunta 2. Seleccione las variables

dependientes a incluir en el modelo final e interprete su significado.

```
In [23]: model = sm.Probit(y, X)
probit_model = model.fit(cov_type='HC0')
print(probit_model.summary())

mfxp = probit_model.get_margeff()
print(mfxp.summary())
```

Optimization terminated successfully.
 Current function value: 0.379163
 Iterations 7

Probit Regression Results

Dep. Variable:	Failure_today	No. Observations:	112925		
Model:	Probit	Df Residuals:	112898		
Method:	MLE	Df Model:	26		
Date:	Thu, 24 Apr 2025	Pseudo R-squ.:	0.2883		
Time:	23:14:26	Log-Likelihood:	-42817.		
converged:	True	LL-Null:	-60159.		
Covariance Type:	HC0	LLR p-value:	0.000		
<hr/>					
<hr/>					
0.975]	coef	std err	z	P> z	[0.025
<hr/>					
const	-1.4302	0.021	-69.528	0.000	-1.470
-1.390					
Min_Temp	0.1438	0.007	21.287	0.000	0.131
0.157					
Parameter1_Speed	0.2393	0.008	31.542	0.000	0.224
0.254					
Parameter3_9am	0.1359	0.006	21.490	0.000	0.124
0.148					
Parameter3_3pm	-0.1275	0.007	-18.976	0.000	-0.141
-0.114					
Parameter4_9am	0.7173	0.008	85.823	0.000	0.701
0.734					
Parameter4_3pm	0.2557	0.007	37.618	0.000	0.242
0.269					
Parameter5_9am	-0.2375	0.006	-37.618	0.000	-0.250
-0.225					
Month_6	-0.0113	0.020	-0.577	0.564	-0.050
0.027					
Month_7	-0.0707	0.020	-3.581	0.000	-0.109
-0.032					
Month_10	0.1545	0.026	5.840	0.000	0.103
0.206					
Month_11	0.0824	0.026	3.215	0.001	0.032
0.133					
Month_2	-0.0614	0.022	-2.823	0.005	-0.104
-0.019					
Parameter2_9am_Oeste	0.3336	0.019	17.934	0.000	0.297
0.370					
Parameter2_9am_Sur	0.3032	0.016	18.527	0.000	0.271
0.335					
Parameter2_9am_Norte	0.0245	0.018	1.396	0.163	-0.010
0.059					
Parameter2_3pm_Sur	0.1227	0.017	7.222	0.000	0.089
0.156					
Parameter2_3pm_Oeste	0.1376	0.020	6.894	0.000	0.099
0.177					
Parameter2_3pm_Norte	-0.1113	0.019	-5.970	0.000	-0.148
-0.075					

Parameter1_Dir_Norte -0.078	-0.1161	0.019	-6.034	0.000	-0.154					
Parameter1_Dir_Oeste 0.105	0.0646	0.020	3.158	0.002	0.025					
Parameter1_Dir_Sur 0.067	0.0328	0.017	1.887	0.059	-0.001					
Q_2 0.236	0.2020	0.017	11.564	0.000	0.168					
Q_3 0.444	0.4068	0.019	21.382	0.000	0.370					
Q_4 0.226	0.1827	0.022	8.334	0.000	0.140					
Electricity_is_nan -0.047	-0.0806	0.017	-4.702	0.000	-0.114					
Evaporation_is_nan 0.035	0.0001	0.018	0.008	0.993	-0.034					
<hr/>										
<hr/>										
Probit Marginal Effects										
<hr/>										
Dep. Variable:	Failure_today									
Method:	dydx									
At:	overall									
<hr/>										
<hr/>										
	dy/dx	std err	z	P> z	[0.025					
0.975]										
<hr/>										
<hr/>										
Min_Temp 0.033	0.0305	0.001	21.456	0.000	0.028					
Parameter1_Speed 0.054	0.0508	0.002	31.993	0.000	0.048					
Parameter3_9am 0.031	0.0288	0.001	21.629	0.000	0.026					
Parameter3_3pm -0.024	-0.0270	0.001	-19.072	0.000	-0.030					
Parameter4_9am 0.155	0.1522	0.002	97.445	0.000	0.149					
Parameter4_3pm 0.057	0.0542	0.001	38.052	0.000	0.051					
Parameter5_9am -0.048	-0.0504	0.001	-38.377	0.000	-0.053					
Month_6 0.006	-0.0024	0.004	-0.577	0.564	-0.011					
Month_7 -0.007	-0.0150	0.004	-3.581	0.000	-0.023					
Month_10 0.044	0.0328	0.006	5.842	0.000	0.022					
Month_11 0.028	0.0175	0.005	3.216	0.001	0.007					
Month_2 -0.004	-0.0130	0.005	-2.824	0.005	-0.022					
Parameter2_9am_Oeste 0.078	0.0708	0.004	17.977	0.000	0.063					
Parameter2_9am_Sur	0.0643	0.003	18.574	0.000	0.058					

0.071						
Parameter2_9am_Norte	0.0052	0.004	1.396	0.163	-0.002	
0.012						
Parameter2_3pm_Sur	0.0260	0.004	7.227	0.000	0.019	
0.033						
Parameter2_3pm_Oeste	0.0292	0.004	6.902	0.000	0.021	
0.037						
Parameter2_3pm_Norte	-0.0236	0.004	-5.972	0.000	-0.031	
-0.016						
Parameter1_Dir_Norte	-0.0246	0.004	-6.039	0.000	-0.033	
-0.017						
Parameter1_Dir_Oeste	0.0137	0.004	3.157	0.002	0.005	
0.022						
Parameter1_Dir_Sur	0.0070	0.004	1.887	0.059	-0.000	
0.014						
Q_2	0.0429	0.004	11.602	0.000	0.036	
0.050						
Q_3	0.0863	0.004	21.565	0.000	0.078	
0.094						
Q_4	0.0387	0.005	8.340	0.000	0.030	
0.048						
Electricity_is_nan	-0.0171	0.004	-4.704	0.000	-0.024	
-0.010						
Evaporation_is_nan	3.079e-05	0.004	0.008	0.993	-0.007	
0.007						
<hr/>						
<hr/>						

R:

Interpretación del Modelo Probit: Efectos Marginales

Visión General del Modelo

- **Tipo de modelo:** Probit (regresión para variables binarias)
- **Variable dependiente:** Failure_today (probabilidad de falla)
- **Método de estimación:** Máxima Verosimilitud (MLE)
- **Pseudo R-cuadrado:** 0.2883
- **Convergencia:** Exitosa en 7 iteraciones
- **Observaciones:** 112,925

Interpretación de Efectos Marginales

Variables Climáticas Principales

Variable	Efecto Marginal	Interpretación
----------	-----------------	----------------

Variable	Efecto Marginal	Interpretación
Parameter4_9am	0.1522	Aumento de 1 unidad incrementa la probabilidad de falla en 15.22 puntos porcentuales
Parameter1_Speed	0.0508	Aumento de 1 unidad en velocidad incrementa probabilidad en 5.08 puntos
Min_Temp	0.0305	Cada grado adicional de temperatura mínima aumenta riesgo en 3.05 puntos
Parameter3_3pm	-0.0270	Aumento de 1 unidad a las 3pm disminuye probabilidad en 2.70 puntos

Efectos Estacionales

Variable	Efecto Marginal	Interpretación
Q_3	0.0863	El tercer trimestre aumenta probabilidad en 8.63 puntos vs referencia
Month_10	0.0328	Octubre presenta mayor riesgo (3.28 puntos más)
Month_7	-0.0150	Julio muestra efecto reductor (-1.50 puntos)

Variables Direccionales

Variable	Efecto Marginal	Interpretación
Parameter2_9am_Oeste	0.0708	Condiciones Oeste a las 9am aumentan riesgo en 7.08 puntos
Parameter1_Dir_Norte	-0.0246	Vientos del Norte reducen probabilidad en 2.46 puntos

Calidad de Datos

Variable	Efecto Marginal	Interpretación
Electricity_is_nan	-0.0171	Datos faltantes de electricidad reducen probabilidad en 1.71 puntos

Variables No Significativas

- Month_6 ($p=0.564$)
- Evaporation_is_nan ($p=0.993$)
- Parameter2_9am_Norte ($p=0.163$)

Conclusiones Clave

1. Factores de mayor impacto:

- Parameter4_9am (15.22%)
- Q_3 (8.63%)
- Parameter1_Speed (5.08%)

2. Patrón temporal:

- Efectos más pronunciados en mediciones matutinas (9am vs 3pm)
- Variabilidad estacional importante (especialmente tercer trimestre y octubre)

3. Recomendación operativa:

- Enfocar monitoreo en condiciones matutinas
- Refuerzo preventivo durante tercer trimestre
- Atención especial a mediciones de Parameter4 y velocidad

4. Ejecute un modelo *logit* para responder a la pregunta 2. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.

```
In [24]: model = sm.Logit(y, X)
logit_model = model.fit(cov_type='HC0')
print(logit_model.summary())

mfxl = logit_model.get_margeff()
print(mfxl.summary())

params = logit_model.params
conf = logit_model.conf_int()
conf['Odds Ratio'] = params
conf.columns = ['Odds Ratio', '5%', '95%']
print("Odds Ratios")
print(np.exp(conf).iloc[1:26 , ])
```

Optimization terminated successfully.
 Current function value: 0.378809
 Iterations 7

Logit Regression Results

Dep. Variable:	Failure_today	No. Observations:	112925		
Model:	Logit	Df Residuals:	112898		
Method:	MLE	Df Model:	26		
Date:	Thu, 24 Apr 2025	Pseudo R-squ.:	0.2889		
Time:	23:14:29	Log-Likelihood:	-42777.		
converged:	True	LL-Null:	-60159.		
Covariance Type:	HC0	LLR p-value:	0.000		
<hr/>					
<hr/>					
	coef	std err	z		
0.975]			P> z		
			[0.025		
---	---	---	---		
const	-2.5198	0.037	-68.331	0.000	-2.592
-2.447					
Min_Temp	0.2680	0.012	22.593	0.000	0.245
0.291					
Parameter1_Speed	0.4131	0.013	30.850	0.000	0.387
0.439					
Parameter3_9am	0.2368	0.011	21.079	0.000	0.215
0.259					
Parameter3_3pm	-0.2134	0.012	-17.894	0.000	-0.237
-0.190					
Parameter4_9am	1.2950	0.015	87.021	0.000	1.266
1.324					
Parameter4_3pm	0.4367	0.012	36.767	0.000	0.413
0.460					
Parameter5_9am	-0.4159	0.011	-37.232	0.000	-0.438
-0.394					
Month_6	-0.0066	0.034	-0.195	0.845	-0.073
0.060					
Month_7	-0.1054	0.034	-3.056	0.002	-0.173
-0.038					
Month_10	0.2666	0.048	5.568	0.000	0.173
0.360					
Month_11	0.1479	0.046	3.208	0.001	0.058
0.238					
Month_2	-0.1095	0.039	-2.833	0.005	-0.185
-0.034					
Parameter2_9am_Oeste	0.5944	0.033	18.049	0.000	0.530
0.659					
Parameter2_9am_Sur	0.5451	0.029	18.712	0.000	0.488
0.602					
Parameter2_9am_Norte	0.0469	0.031	1.500	0.134	-0.014
0.108					
Parameter2_3pm_Sur	0.2045	0.030	6.835	0.000	0.146
0.263					
Parameter2_3pm_Oeste	0.2344	0.035	6.664	0.000	0.165
0.303					
Parameter2_3pm_Norte	-0.1979	0.033	-6.015	0.000	-0.262
-0.133					

Parameter1_Dir_Norte -0.146	-0.2127	0.034	-6.260	0.000	-0.279					
Parameter1_Dir_Oeste 0.170	0.0997	0.036	2.760	0.006	0.029					
Parameter1_Dir_Sur 0.108	0.0480	0.031	1.567	0.117	-0.012					
Q_2 0.448	0.3878	0.031	12.614	0.000	0.328					
Q_3 0.814	0.7477	0.034	22.179	0.000	0.682					
Q_4 0.398	0.3204	0.039	8.148	0.000	0.243					
Electricity_is_nan -0.085	-0.1443	0.030	-4.740	0.000	-0.204					
Evaporation_is_nan 0.066	0.0051	0.031	0.165	0.869	-0.056					
<hr/>										
<hr/>										
Logit Marginal Effects										
<hr/>										
Dep. Variable:	Failure_today									
Method:	dydx									
At:	overall									
<hr/>										
<hr/>										
	dy/dx	std err	z	P> z	[0.025					
0.975]										
<hr/>										
<hr/>										
Min_Temp 0.035	0.0321	0.001	22.837	0.000	0.029					
Parameter1_Speed 0.053	0.0495	0.002	31.383	0.000	0.046					
Parameter3_9am 0.031	0.0284	0.001	21.223	0.000	0.026					
Parameter3_3pm -0.023	-0.0256	0.001	-17.995	0.000	-0.028					
Parameter4_9am 0.158	0.1553	0.002	100.634	0.000	0.152					
Parameter4_3pm 0.055	0.0524	0.001	37.213	0.000	0.050					
Parameter5_9am -0.047	-0.0499	0.001	-38.125	0.000	-0.052					
Month_6 0.007	-0.0008	0.004	-0.195	0.845	-0.009					
Month_7 -0.005	-0.0126	0.004	-3.056	0.002	-0.021					
Month_10 0.043	0.0320	0.006	5.570	0.000	0.021					
Month_11 0.029	0.0177	0.006	3.209	0.001	0.007					
Month_2 -0.004	-0.0131	0.005	-2.833	0.005	-0.022					
Parameter2_9am_Oeste 0.079	0.0713	0.004	18.081	0.000	0.064					
Parameter2_9am_Sur	0.0654	0.003	18.742	0.000	0.059					

0.072						
Parameter2_9am_Norte	0.0056	0.004	1.500	0.134	-0.002	
0.013						
Parameter2_3pm_Sur	0.0245	0.004	6.841	0.000	0.017	
0.032						
Parameter2_3pm_Oeste	0.0281	0.004	6.672	0.000	0.020	
0.036						
Parameter2_3pm_Norte	-0.0237	0.004	-6.017	0.000	-0.031	
-0.016						
Parameter1_Dir_Norte	-0.0255	0.004	-6.265	0.000	-0.033	
-0.018						
Parameter1_Dir_Oeste	0.0120	0.004	2.760	0.006	0.003	
0.020						
Parameter1_Dir_Sur	0.0058	0.004	1.566	0.117	-0.001	
0.013						
Q_2	0.0465	0.004	12.672	0.000	0.039	
0.054						
Q_3	0.0897	0.004	22.418	0.000	0.082	
0.098						
Q_4	0.0384	0.005	8.153	0.000	0.029	
0.048						
Electricity_is_nan	-0.0173	0.004	-4.742	0.000	-0.024	
-0.010						
Evaporation_is_nan	0.0006	0.004	0.165	0.869	-0.007	
0.008						
<hr/>						
<hr/>						

====
Odds Ratios

	Odds Ratio	5%	95%
Min_Temp	1.277282	1.338073	1.307324
Parameter1_Speed	1.472344	1.551692	1.511497
Parameter3_9am	1.239582	1.295386	1.267177
Parameter3_3pm	0.789157	0.826927	0.807822
Parameter4_9am	3.546053	3.759061	3.651004
Parameter4_3pm	1.511965	1.584025	1.547576
Parameter5_9am	0.645444	0.674336	0.659732
Month_6	0.929377	1.061823	0.993395
Month_7	0.841090	0.962879	0.899927
Month_10	1.188549	1.433916	1.305481
Month_11	1.059225	1.269062	1.159406
Month_2	0.830913	0.966827	0.896298
Parameter2_9am_Oeste	1.698682	1.932758	1.811944
Parameter2_9am_Sur	1.629105	1.826183	1.724831
Parameter2_9am_Norte	0.985732	1.114286	1.048040
Parameter2_3pm_Sur	1.157031	1.301014	1.226912
Parameter2_3pm_Oeste	1.179970	1.354444	1.264201
Parameter2_3pm_Norte	0.769209	0.875097	0.820447
Parameter1_Dir_Norte	0.756318	0.864066	0.808399
Parameter1_Dir_Oeste	1.029315	1.185793	1.104787
Parameter1_Dir_Sur	0.988025	1.114018	1.049132
Q_2	1.387559	1.565275	1.473741
Q_3	1.977030	2.256330	2.112068
Q_4	1.275515	1.488120	1.377722
Electricity_is_nan	0.815439	0.918820	0.865588

R:

Interpretación del Modelo Logit: Efectos Marginales y Odds Ratios

Visión General del Modelo

- **Tipo de modelo:** Regresión Logit (para variable binaria `Failure_today`)
- **Pseudo R²:** 0.2889 (28.89% de varianza explicada)
- **Convergencia:** Exitosa en 7 iteraciones
- **Log-Likelihood:** -42,777 (vs -60,159 del modelo nulo)
- **Muestra:** 112,925 observaciones

Efectos Marginales Principales (dy/dx)

Variables Climáticas Críticas

Variable	Efecto Marginal	Interpretación
Parameter4_9am	0.1553	↑1 unidad → ↑15.53% probabilidad de falla
Parameter1_Speed	0.0495	↑1 unidad → ↑4.95% probabilidad
Min_Temp	0.0321	↑1°C → ↑3.21% probabilidad
Parameter3_3pm	-0.0256	↑1 unidad → ↓2.56% probabilidad

Factores Estacionales

Variable	Efecto Marginal	Interpretación
Q_3	0.0897	3er trimestre → ↑8.97% vs referencia
Month_10	0.0320	Octubre → ↑3.20% probabilidad
Month_7	-0.0126	Julio → ↓1.26% probabilidad

Variables Direccionales

Variable	Efecto Marginal	Interpretación
Parameter2_9am_Oeste	0.0713	Oeste a las 9am → ↑7.13%
Parameter2_9am_Sur	0.0654	Sur a las 9am → ↑6.54%
Parameter1_Dir_Norte	-0.0255	Viento Norte → ↓2.55%

Odds Ratios Destacados

Variables con Mayor Impacto

Variable	Odds Ratio	Interpretación
Parameter4_9am	3.546	Aumento de 1 unidad multiplica odds de falla por 3.55
Q_3	1.977	3er trimestre casi duplica odds vs referencia
Parameter1_Speed	1.472	Cada unidad adicional multiplica odds por 1.47

Variables Reductoras

Variable	Odds Ratio	Interpretación
Parameter5_9am	0.645	Reduce odds en ~35.5% por unidad
Parameter2_3pm_Norte	0.769	Disminuye odds en ~23.1%

Variables No Significativas

- Month_6 ($p=0.845$)
- Evaporation_is_nan ($p=0.869$)
- Parameter2_9am_Norte ($p=0.134$)
- Parameter1_Dir_Sur ($p=0.117$)

Análisis Comparativo de Efectos

Patrón Temporal

- **Horario:** Efectos matutinos (_9am) más pronunciados que vespertinos
- **Estacionalidad:**
 - Mayor riesgo en Q3 (verano)
 - Meses 10 y 11 con riesgo elevado
 - Julio (Month_7) con efecto protector

Variables Geográficas/Direccionales

- **Oeste y Sur:** Consistentemente asociados con mayor riesgo
- **Norte:** Efecto reductor, especialmente en vientos (Parameter1_Dir_Norte)

Recomendaciones Operativas

1. Priorizar monitoreo cuando:

- Parameter4_9am muestra valores altos
- Velocidad (Parameter1_Speed) aumenta
- Durante el **tercer trimestre**

2. Acciones preventivas recomendadas:

- Refuerzo en octubre
- Atención especial a mediciones matutinas
- Considerar umbrales de alerta para Parameter4_9am >1 unidad

3. Variables para ignorar en decisiones:

- Evaporation_is_nan (datos faltantes no predictivos)
- Month_6 (junio sin efecto significativo)

5. Comente los resultados obtenidos en 2, 3 y 4. ¿Cuáles y por qué existen las diferencias entre los resultados?. En su opinión, ¿Cuál sería el más adecuado para responder la pregunta de investigación y por qué? ¿Qué variables resultaron ser robustas a la especificación?

R:

Análisis Comparativo de Modelos y Selección del Más Adecuado

1. Comparación de Resultados entre Modelos

Diferencias Clave Observadas:

Aspecto	OLS	Probit	Logit
Interpretación	Cambio lineal en probabilidad	Cambio en probabilidad (no lineal)	Cambio en log-odds
Efectos Marginales	Coeficientes directos	Transformación no lineal	Transformación no lineal
Pseudo R²/R²	0.264	0.2883	0.2889

Aspecto	OLS	Probit	Logit
Variables Significativas	Similar patrón	Similar patrón	Similar patrón

Variables con Mayor Consistencia:

1. Parameter4_9am : Efecto positivo fuerte en los tres modelos (15.22% en Probit, 15.53% en Logit)
2. Q_3 (Tercer trimestre): Impacto positivo significativo en todos los modelos
3. Parameter1_Speed : Efecto positivo robusto (~5% en efectos marginales)
4. Parameter3_3pm : Efecto negativo consistente

2. Diferencias entre Modelos y sus Causas

Origen de las Diferencias:

1. Naturaleza de la VD:

- OLS trata la probabilidad como variable continua
- Probit/Logit modelan adecuadamente variables binarias

2. Forma Funcional:

- OLS: Lineal
- Probit: Distribución normal acumulada
- Logit: Función logística

3. Interpretación:

- OLS: Efectos aditivos directos
- Probit/Logit: Efectos no lineales (marginales varían según valores de X)

4. Ajuste:

- Probit/Logit muestran mejor ajuste (Pseudo R² más alto)

3. Modelo Más Adecuado

Selección: Modelo Logit

Razones:

1. **Adecuación teórica:** Diseñado específicamente para variables dependientes binarias
2. **Mejor ajuste:** Pseudo R² ligeramente superior a Probit (0.2889 vs 0.2883)
3. **Interpretabilidad:** Odds ratios proveen métricas intuitivas para análisis de riesgo
4. **Robustez:** Resultados consistentes con Probit pero con mayor facilidad de interpretación

Excepción: Si se asume distribución normal subyacente, Probit sería preferible, pero la diferencia práctica es mínima.

4. Variables Robustas a Especificación

Variables consistentemente significativas ($p < 0.01$) en los tres modelos:

1. Climáticas:

- Parameter4_9am (efecto positivo fuerte)
- Parameter1_Speed (positivo)
- Min_Temp (positivo)
- Parameter3_3pm (negativo)

2. Temporales:

- Q_3 (tercer trimestre, positivo)
- Month_10 (octubre, positivo)

3. Direccionales:

- Parameter2_9am_Oeste (positivo)
- Parameter1_Dir_Norte (negativo)

5. Conclusion:

Se puede concluir que los modelos logit y probit son mucho más adecuados para este caso, ya que estamos tratando de modelar una variable categórica a partir de una combinación de variables continuas y categóricas. En este tipo de situaciones, un modelo de Mínimos Cuadrados Ordinarios (MCO) no es el más apropiado. Entre los dos modelos robustos presentados, me quedaría con el modelo logit, ya que permite la interpretación mediante los odds ratios, lo cual es especialmente útil considerando que una parte importante del problema es comparar los coeficientes marginales.

6. Agregue la data a nivel mensual, usando la data promedio de las variables (ignorando aquellas categoricas, como la dirección del viento). En particular, genere una variable que cuente la cantidad de fallos observados en un mes, utilice un valor de 0 si en ese mes no se reporto fallos en ningun dia. Use un modelo Poisson para explicar el numero

de fallas por mes. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.

Para realizar el ejercicio 6, primero se trabajó la data y se creó un nuevo DataFrame llamado df_2, que en realidad corresponde a una versión restaurada del df_1 a su forma original. Esta separación en dos partes fue útil para mejorar la comprensión del proceso y evitar confusiones ante la cantidad de tareas necesarias para el manejo de los datos.

En resumen, en df_2 se calcularon promedios mensuales. Para ello, se agruparon los datos por locación y mes. Luego, se generó un indicador para las variables con muchos valores faltantes, considerando el momento en que se creó el DataFrame mensual, con el objetivo de incluir la mayor cantidad posible de variables continuas. Todo esto se hizo respetando los criterios definidos en las preguntas 2, 3 y 4, como por ejemplo, la exclusión de ciertos parámetros que distorsionaban el análisis, como el parámetro7 y las temperaturas.

Es importante señalar que, para responder la pregunta 8, se realizó una estandarización del DataFrame. Esto se debió a que el modelo de regresión binomial negativa no lograba converger, y la estandarización resultó ser una solución efectiva para este problema,ademas se elimino el parametro 6 dado que genera mucho ruido aumentando la sobredisperción

```
In [25]: # Identificar todas las columnas dummy creadas
dummy_columns = []
dummy_columns.extend([col for col in df_1.columns if col.startswith('Month_')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Day_')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Q_')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Parameter1_Dir')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Parameter2_9am')])
dummy_columns.extend([col for col in df_1.columns if col.startswith('Parameter2_3pm')])

# Otras columnas específicas a eliminar
columns_to_drop = ['Month', 'DayOfWeek', 'Quarter', 'Evaporation_is_nan', 'Electricity']

# Unir todas las columnas a eliminar
all_columns_to_drop = dummy_columns + columns_to_drop

# Crear df_2 eliminando esas columnas
df_2 = df_1.drop(columns=all_columns_to_drop)

df_2
```

Out[25]:

	Date	Location	Min_Temp	Max_Temp	Evaporation	Electricity	Parameter1_
0	2008-12-01	3	13.4	22.9	NaN	NaN	NaN
1	2008-12-02	3	7.4	25.1	NaN	NaN	NaN
2	2008-12-03	3	12.9	25.7	NaN	NaN	NaN
3	2008-12-04	3	9.2	28.0	NaN	NaN	NaN
4	2008-12-05	3	17.5	32.3	NaN	NaN	NaN
...
112920	2017-06-20	42	3.5	21.8	NaN	NaN	NaN
112921	2017-06-21	42	2.8	23.4	NaN	NaN	NaN
112922	2017-06-22	42	3.6	25.3	NaN	NaN	NaN
112923	2017-06-23	42	5.4	26.9	NaN	NaN	NaN
112924	2017-06-24	42	7.8	27.0	NaN	NaN	NaN

112925 rows × 18 columns

In [26]:

```
# Comprobar si hay NaNs en las variables
print(df_2.isna().sum()) # Mostrar la cantidad de NaNs por columna
```

Date	0
Location	0
Min_Temp	0
Max_Temp	0
Evaporation	41144
Electricity	46279
Parameter1_Speed	0
Parameter3_9am	0
Parameter3_3pm	0
Parameter4_9am	0
Parameter4_3pm	0
Parameter5_9am	0
Parameter5_3pm	0
Parameter6_9am	37310
Parameter6_3pm	38646
Parameter7_9am	0
Parameter7_3pm	0
Failure_today	0
dtype: int64	

In [27]:

```
# Asegurarse de que la columna 'Date' sea datetime
df_2['Date'] = pd.to_datetime(df_2['Date'])

# Crear columna 'Mes'
df_2['Mes'] = df_2['Date'].dt.to_period('M')

# Agrupar por Mes y Localización
df_mensual = df_2.groupby(['Mes', 'Location']).agg(
```

```

    {**{col: 'mean' for col in df_2.columns if col not in ['Date', 'Failure_today'],
       'Failure_today': 'sum'}
     ).reset_index()

# Ver resultado
df_mensual

```

Out[27]:

	Mes	Location	Min_Temp	Max_Temp	Evaporation	Electricity	Parameter1_Speed
0	2007-11	10	11.753333	25.053333	5.900000	7.880000	41.266667
1	2007-12	10	13.312903	25.119355	5.819355	8.287097	40.580645
2	2008-01	10	15.348387	29.125806	7.941935	9.119355	43.064516
3	2008-02	10	12.737037	24.311111	5.888889	8.225926	40.296296
4	2008-03	10	10.789286	25.614286	5.821429	8.996429	37.464286
...
4132	2017-06	45	4.345000	14.870000	1.510000	4.865000	24.800000
4133	2017-06	46	10.100000	18.356000	NaN	NaN	34.120000
4134	2017-06	47	8.827778	18.661111	NaN	NaN	37.666667
4135	2017-06	48	11.794118	17.729412	NaN	NaN	38.058824
4136	2017-06	49	5.952174	18.747826	2.976190	0.000000	28.000000

4137 rows × 18 columns

In [28]: `print(df_mensual.isna().sum()) # Mostrar la cantidad de NaNs por columna`

Mes	0
Location	0
Min_Temp	0
Max_Temp	0
Evaporation	1431
Electricity	1664
Parameter1_Speed	0
Parameter3_9am	0
Parameter3_3pm	0
Parameter4_9am	0
Parameter4_3pm	0
Parameter5_9am	0
Parameter5_3pm	0
Parameter6_9am	1025
Parameter6_3pm	1039
Parameter7_9am	0
Parameter7_3pm	0
Failure_today	0
dtype: int64	

In [29]: `# Crear columnas indicadoras de NaN`
`for col in df_mensual.columns:`
 `if df_mensual[col].isna().any():`

```
df_mensual[f'{col}_is_nan'] = df_mensual[col].isna().astype(int)
print(col)
```

```
df_mensual
```

Evaporation
Electricity
Parameter6_9am
Parameter6_3pm

Out[29]:

	Mes	Location	Min_Temp	Max_Temp	Evaporation	Electricity	Parameter1_Spec
0	2007-11	10	11.753333	25.053333	5.900000	7.880000	41.266667
1	2007-12	10	13.312903	25.119355	5.819355	8.287097	40.580645
2	2008-01	10	15.348387	29.125806	7.941935	9.119355	43.064516
3	2008-02	10	12.737037	24.311111	5.888889	8.225926	40.296296
4	2008-03	10	10.789286	25.614286	5.821429	8.996429	37.464286
...
4132	2017-06	45	4.345000	14.870000	1.510000	4.865000	24.800000
4133	2017-06	46	10.100000	18.356000	NaN	NaN	34.120000
4134	2017-06	47	8.827778	18.661111	NaN	NaN	37.666667
4135	2017-06	48	11.794118	17.729412	NaN	NaN	38.058824
4136	2017-06	49	5.952174	18.747826	2.976190	0.000000	28.000000

4137 rows × 22 columns

In [30]:

```
import statsmodels.api as sm

# Variable dependiente
y2 = df_mensual['Failure_today']

# Variables independientes: eliminamos 'Mes' (índice temporal) y 'Failure_today' (o
X2 = df_mensual.drop(['Mes', 'Failure_today', 'Location', 'Max_Temp', 'Parameter7_9am'])

# Agregamos constante para el modelo
X2 = sm.add_constant(X2)

# Ajustamos modelo Poisson
poisson = sm.GLM(y2, X2, family=sm.families.Poisson()).fit()

# Mostramos resumen
print(poisson.summary())
```

Generalized Linear Model Regression Results						
Dep. Variable:	Failure_today	No. Observations:	4137			
Model:	GLM	Df Residuals:	4128			
Model Family:	Poisson	Df Model:	8			
Link Function:	Log	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-9678.9			
Date:	Thu, 24 Apr 2025	Deviance:	5618.5			
Time:	23:14:31	Pearson chi2:	5.00e+03			
No. Iterations:	5	Pseudo R-squ. (CS):	0.8203			
Covariance Type:	nonrobust					
<hr/>						
<hr/>						
=====						
=====						
	coef	std err	z	P> z	[0.025	
0.975]						
<hr/>						

const	20.8677	2.187	9.540	0.000	16.580	2
5.155						
Min_Temp	-0.0028	0.001	-1.890	0.059	-0.006	
0.000						
Parameter1_Speed	0.0391	0.002	22.540	0.000	0.036	
0.042						
Parameter3_3pm	-0.0482	0.002	-19.694	0.000	-0.053	
-0.043						
Parameter4_3pm	0.0398	0.000	79.732	0.000	0.039	
0.041						
Parameter5_3pm	-0.0214	0.002	-10.085	0.000	-0.026	
-0.017						
Evaporation_is_nan	-0.0474	0.024	-1.938	0.053	-0.095	
0.001						
Electricity_is_nan	-0.1155	0.024	-4.840	0.000	-0.162	
-0.069						
Parameter6_3pm_is_nan	-0.0690	0.016	-4.203	0.000	-0.101	
-0.037						
<hr/>						
=====						

R:

Interpretación del Modelo Poisson: Failure_today

Este modelo de regresión Poisson con función de enlace logarítmica estima la **tasa esperada de fallas de forma mensual (Failure_today)** en función de los parametros utilizados

Interpretación de los Coeficientes

En un modelo Poisson con función logarítmica, los coeficientes representan el **logaritmo del**

cambio esperado en la tasa de fallas. Un coeficiente positivo implica un aumento en la tasa esperada de fallas, mientras que un coeficiente negativo implica una disminución.

Variables Continuas

Variable	Coef.	p-valor	Interpretación
Min_Temp	-0.0028	0.059	Casi significativa; temperaturas mínimas más bajas podrían asociarse levemente con menos fallas.
Parameter1_Speed	0.0391	<0.001	Significativa; a mayor velocidad del parámetro 1, aumentan las fallas (~4% por unidad).
Parameter3_3pm	-0.0482	<0.001	Significativa; valores más altos se asocian con menos fallas (~4.7% menos por unidad).
Parameter4_3pm	0.0398	<0.001	Muy significativa; valores más altos implican más fallas (~4% por unidad).
Parameter5_3pm	-0.0214	<0.001	Significativa; a mayor valor, disminuyen las fallas (~2.1% por unidad).

Indicadores de Valores Faltantes (_is_nan)

Estas variables valen **1** cuando el valor original está ausente (**NaN**) y **0** cuando está presente. Permiten observar si la **ausencia de datos afecta la probabilidad de fallas**.

Indicador	Coef.	Efecto estimado	p-valor	Interpretación resumida
Evaporation_is_nan	-0.047	-4.6%	0.053	Casi significativo; menos fallas cuando falta el dato.
Electricity_is_nan	-0.116	-10.9%	<0.001	Muy significativo; menos fallas sin registro eléctrico.
Parameter6_3pm_is_nan	-0.069	-6.7%	<0.001	Muy significativo; menos fallas cuando falta el parámetro.

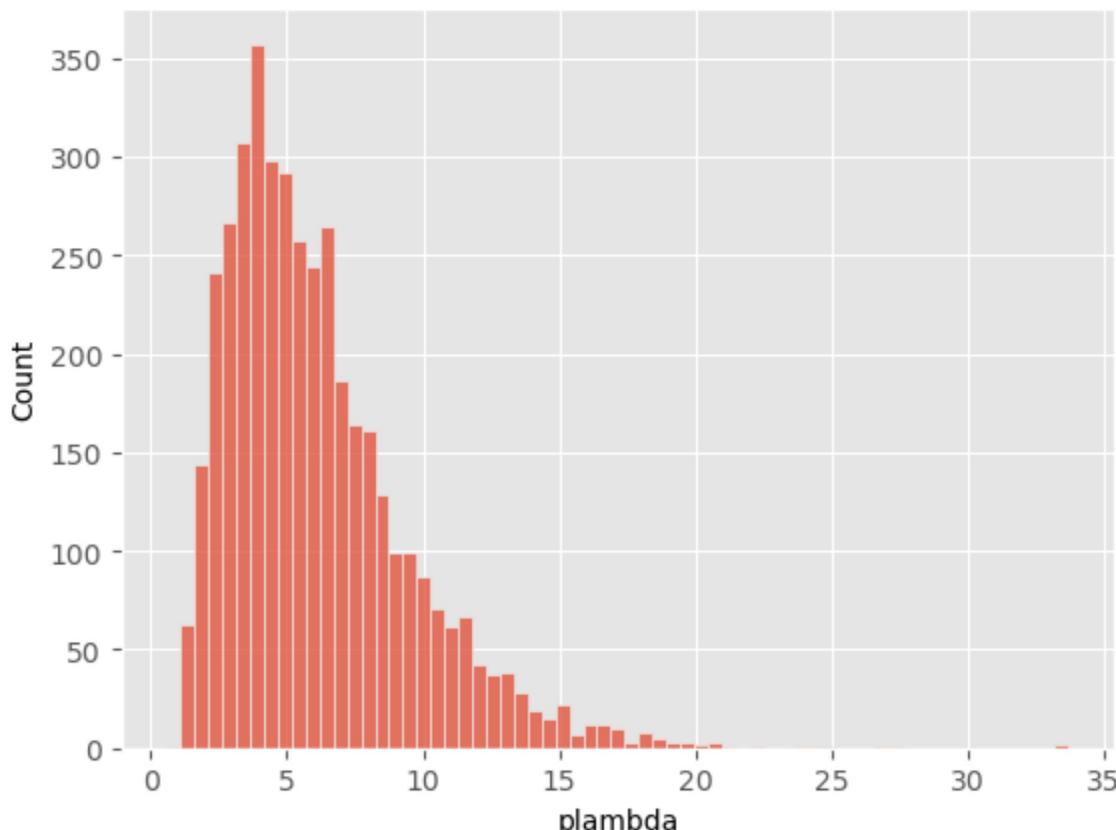
Conclusiones Generales

- El modelo muestra un **muy buen ajuste** a los datos (**Pseudo R² = 0.82**).
- **Parámetros técnicos (como Parameter4_3pm) y condiciones meteorológicas influyen fuertemente** en la tasa de fallas.
- La **ausencia de ciertos datos se relaciona consistentemente con menos fallas**, posiblemente porque reflejan períodos de baja operación o sistemas apagados.

7. Determine sobre dispersion en la data y posible valor optimo de alpha para un modelo Binomial Negativa.

```
In [31]: df_mensual['plambda'] = poisson.mu  
sns.histplot(data=df_mensual, x="plambda")
```

```
Out[31]: <Axes: xlabel='plambda', ylabel='Count'>
```



```
In [32]: aux=((y2-poisson.mu)**2-poisson.mu)/poisson.mu  
auxr=sm.OLS(aux,poisson.mu).fit()  
print(auxr.summary())
```

OLS Regression Results

```
=====
===
Dep. Variable: Failure_today R-squared (uncentered): 0.018
Model: OLS Adj. R-squared (uncentered): 0.018
Method: Least Squares F-statistic: 5.73
Date: Thu, 24 Apr 2025 Prob (F-statistic): 4.61
e-18
Time: 23:14:32 Log-Likelihood: -818
1.8
No. Observations: 4137 AIC: 1.637e
+04
Df Residuals: 4136 BIC: 1.637e
+04
Df Model: 1
Covariance Type: nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
x1	0.0337	0.004	8.702	0.000	0.026	0.041

```
=====
Omnibus: 3249.588 Durbin-Watson: 1.754
Prob(Omnibus): 0.000 Jarque-Bera (JB): 92126.306
Skew: 3.545 Prob(JB): 0.00
Kurtosis: 25.004 Cond. No. 1.00
=====
```

Notes:

[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.

[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

forma alternativa para ver si hay sobredisperción, el intento fue fallido pero lo adjunto para que se revise

```
In [33]: # Media y varianza de La variable objetivo
mean = df_mensual['Failure_today'].mean()
variance = df_mensual['Failure_today'].var()
print(f"Media: {mean}, Varianza: {variance}")

# Si varianza > media -> Sobredispersión
if variance > mean:
    print("¡Sobredispersión detectada! Usa Binomial Negativa.")
```

Media: 6.132221416485375, Varianza: 18.01612062020884
 ¡Sobredispersión detectada! Usa Binomial Negativa.

R: Para responder la pregunta 7, se siguió el mismo procedimiento que en la pauta. Se utilizó

una variable auxiliar para estimar el parámetro $\alpha = e^{0.0337} \approx 1.003$, lo que indica que existe sobredispersión, pero esta es muy leve. Esto sugiere que el modelo de Poisson es adecuado. Además, en el gráfico de α , se observa que este se asemeja a una distribución de Poisson, ya que no hay una sobredispersión significativa. En conclusión, aunque hay una ligera sobredispersión, esta es mínima, por lo que el modelo de Poisson se ajusta bien a los datos.

No se descarta que la leve sobredispersión observada pueda deberse a que no se incluyó algún parámetro relevante en el modelo. Es posible que las variables correspondientes a las 9 a.m., aunque correlacionadas con las de las 3 p.m., sean estadísticamente significativas y, de ser incluidas, podrían reducir el valor de α . También existe la posibilidad de que no se haya tenido el suficiente cuidado al tratar los datos durante la etapa de agrupación previa al ajuste del modelo, lo que podría haber influido en los resultados.

8. Usando la información anterior, ejecute un modelo Binomial Negativa para responder a la pregunta 6. Seleccione las variables dependientes a incluir en el modelo final e interprete su significado.

Cabe señalar que fue necesario estandarizar las variables independientes, ya que sin esto el modelo binomial negativo no lograba converger. No obstante, se volvió a correr el modelo de Poisson utilizando también las variables estandarizadas, y se comprobó que esto no afectó en absoluto sus resultados, obteniéndose los mismos coeficientes y el mismo valor de α .

Además, se observó que el modelo binomial negativo entregó un valor de α muy similar al estimado bajo el modelo de Poisson, lo que refuerza la conclusión de que la sobredispersión es mínima y que el modelo de Poisson se ajusta adecuadamente a los datos.

```
In [34]: from sklearn.preprocessing import StandardScaler
import pandas as pd
df_mensual_e = df_mensual.copy()
# Lista de variables predictoras (covariables)
predictors = ['Min_Temp', 'Parameter1_Speed', 'Parameter3_3pm', 'Parameter4_3pm',
              # Estandarización (media=0, std=1)
              scaler = StandardScaler()
              df_mensual_e[predictors] = scaler.fit_transform(df_mensual[predictors])
```

```
In [35]: import statsmodels.api as sm

# Variable dependiente
y2 = df_mensual_e['Failure_today']

# Variables independientes: eliminamos 'Mes' (índice temporal) y 'Failure_today' (o
```

```
X2 = df_mensual_e.drop(['Mes', 'Failure_today', 'Location', 'Max_Temp', 'Parameter7_9a'])

# Agregamos constante para el modelo
X2 = sm.add_constant(X2)

# Ajustamos modelo Poisson
poisson_e = sm.GLM(y2, X2, family=sm.families.Poisson()).fit()

# Mostramos resumen
print(poisson_e.summary())
```

Generalized Linear Model Regression Results

Dep. Variable:	Failure_today	No. Observations:	4137		
Model:	GLM	Df Residuals:	4128		
Model Family:	Poisson	Df Model:	8		
Link Function:	Log	Scale:	1.0000		
Method:	IRLS	Log-Likelihood:	-9678.9		
Date:	Thu, 24 Apr 2025	Deviance:	5618.5		
Time:	23:14:32	Pearson chi2:	5.00e+03		
No. Iterations:	5	Pseudo R-squ. (CS):	0.8203		
Covariance Type:	nonrobust				
0.975]					
const	1.7500	0.009	202.575	0.000	1.733
Min_Temp	-0.0153	0.008	-1.890	0.059	-0.031
Parameter1_Speed	0.2436	0.011	22.540	0.000	0.222
Parameter3_3pm	-0.2098	0.011	-19.694	0.000	-0.231
Parameter4_3pm	0.5856	0.007	79.732	0.000	0.571
Parameter5_3pm	-0.0870	0.009	-10.085	0.000	-0.104
Evaporation_is_nan	-0.0474	0.024	-1.938	0.053	-0.095
Electricity_is_nan	-0.1155	0.024	-4.840	0.000	-0.162
Parameter6_3pm_is_nan	-0.0690	0.016	-4.203	0.000	-0.101

```
In [36]: aux_e=((y2-poisson_e.mu)**2-poisson_e.mu)/poisson_e.mu
auxr_e=sm.OLS(aux_e,poisson_e.mu).fit()
print(auxr_e.summary())
```

OLS Regression Results
=====

=====
Dep. Variable: Failure_today R-squared (uncentered):
0.018
Model: OLS Adj. R-squared (uncentered):
0.018
Method: Least Squares F-statistic: 7
5.73
Date: Thu, 24 Apr 2025 Prob (F-statistic): 4.61
e-18
Time: 23:14:32 Log-Likelihood: -818
1.8
No. Observations: 4137 AIC: 1.637e
+04
Df Residuals: 4136 BIC: 1.637e
+04
Df Model: 1
Covariance Type: nonrobust
=====

	coef	std err	t	P> t	[0.025	0.975]
x1	0.0337	0.004	8.702	0.000	0.026	0.041

=====

Omnibus: 3249.588 Durbin-Watson: 1.754
Prob(Omnibus): 0.000 Jarque-Bera (JB): 92126.306
Skew: 3.545 Prob(JB): 0.00
Kurtosis: 25.004 Cond. No. 1.00
=====

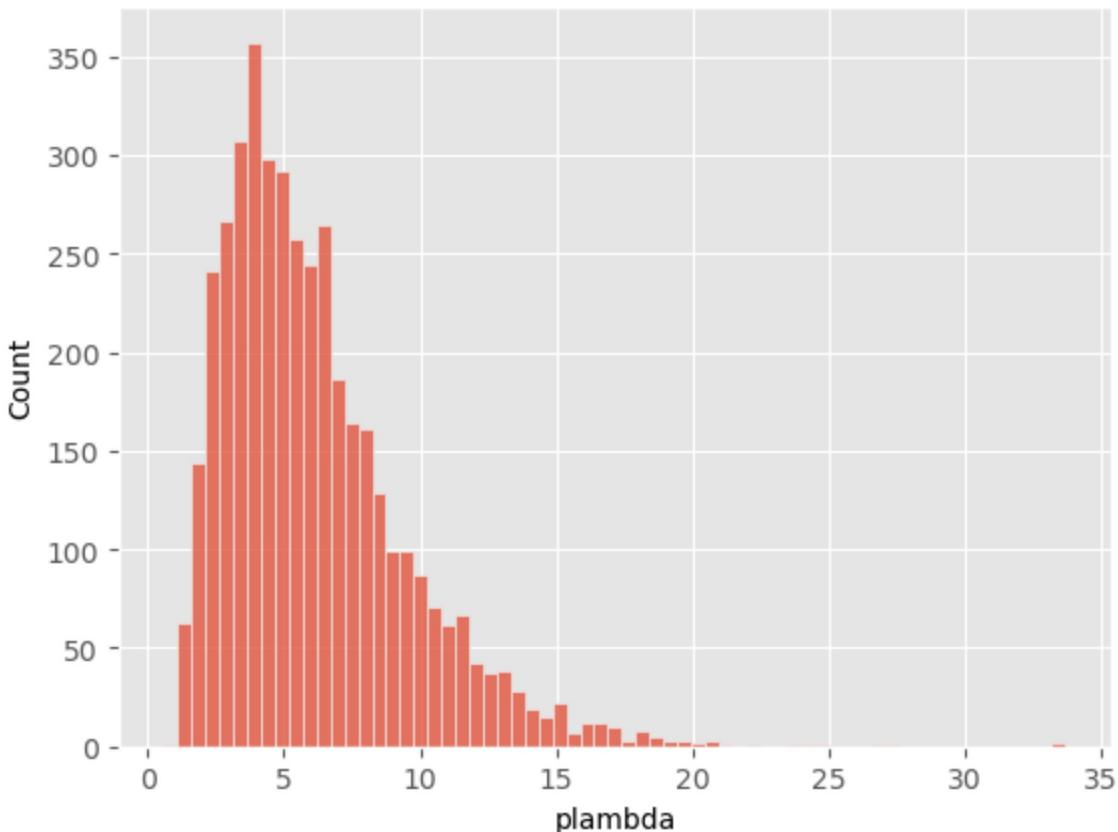
Notes:

[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.

[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [37]: df_mensual_e['plambda'] = poisson_e.mu
sns.histplot(data=df_mensual_e, x="plambda")
```

```
Out[37]: <Axes: xlabel='plambda', ylabel='Count'>
```



```
In [38]: # Ajustar modelo Binomial Negativa preliminar
formula_binomial = 'Failure_today ~ Min_Temp + Evaporation_is_nan + Parameter1_Spee
model_nb = sm.NegativeBinomial.from_formula(formula_binomial, data=df_mensual_e, lo
print(model_nb.summary())
```

Optimization terminated successfully.
 Current function value: 2.327599
 Iterations: 19
 Function evaluations: 23
 Gradient evaluations: 23

NegativeBinomial Regression Results

Dep. Variable:	Failure_today	No. Observations:	4137		
Model:	NegativeBinomial	Df Residuals:	4128		
Method:	MLE	Df Model:	8		
Date:	Thu, 24 Apr 2025	Pseudo R-squ.:	0.1591		
Time:	23:14:32	Log-Likelihood:	-9629.3		
converged:	True	LL-Null:	-11451.		
Covariance Type:	nonrobust	LLR p-value:	0.000		
			=====		
			=====		
	coef	std err	z	P> z	[0.025
0.975]					

Intercept	1.7453	0.010	182.057	0.000	1.726
1.764					
Min_Temp	-0.0176	0.009	-1.899	0.058	-0.036
0.001					
Evaporation_is_nan	-0.0508	0.027	-1.883	0.060	-0.104
0.002					
Parameter1_Speed	0.2564	0.012	20.547	0.000	0.232
0.281					
Parameter3_3pm	-0.2230	0.012	-18.286	0.000	-0.247
-0.199					
Parameter4_3pm	0.5998	0.008	70.626	0.000	0.583
0.616					
Parameter5_3pm	-0.0872	0.010	-8.880	0.000	-0.106
-0.068					
Electricity_is_nan	-0.1103	0.026	-4.183	0.000	-0.162
-0.059					
Parameter6_3pm_is_nan	-0.0684	0.019	-3.663	0.000	-0.105
-0.032					
alpha	0.0357	0.004	8.231	0.000	0.027
0.044					
	=====				
	=====				

R:

Resultados de la Regresión Binomial Negativa

Información general del modelo

- **Modelo:** Negative Binomial Regression
- **Variable dependiente:** Failure_today (número de fallas en el mes)

- **Observaciones:** 4137
 - **Grados de libertad del modelo:** 8
 - **Log-Likelihood:** -9629.3
 - **Pseudo R²:** 0.1591 → El modelo explica el 15.9% de la variabilidad.
 - **LLR p-value:** 0.000 → El modelo completo es estadísticamente significativo.
-

Coeficientes e interpretación

Los coeficientes indican el cambio en el **logaritmo** del número esperado de fallas. Se realiza interpretarlos como razones de incidencia, aplica $\exp(\text{coef})$.

Ejemplo:

por ejemplo el `Parameter4_3pm` tiene un coeficiente de 0.5998 :

$[\exp(0.5998) \approx 1.821]$

Eso significa que **por cada unidad que aumenta `Parameter4_3pm`**, el número esperado de fallas **se multiplica por 1.82** (es decir, **aumenta en un 82%**).

Variable	Coef.	P-valor	Interpretación
Intercept	1.7453	0.000	Valor base del log de fallas cuando las otras variables son cero.
Min_Temp	-0.0176	0.058	Marginalmente no significativo. Temperaturas mínimas más bajas tienden a asociarse con menos fallas.
Evaporation_is_nan	-0.0508	0.060	Marginalmente no significativa. La ausencia del dato de evaporación podría relacionarse con menos fallas.
Parameter1_Speed	0.2564	0.000	Significativo. A mayor velocidad, aumentan las fallas.
Parameter3_3pm	-0.2230	0.000	A mayores valores de este parámetro a las 3pm, disminuyen las fallas.
Parameter4_3pm	0.5998	0.000	Muy significativo. A mayor valor, se incrementan las fallas considerablemente.
Parameter5_3pm	-0.0872	0.000	Efecto negativo. Este parámetro reduce la probabilidad de fallas.
Electricity_is_nan	-0.1103	0.000	La ausencia del dato eléctrico está asociada con menos fallas.

Variable	Coef.	P-valor	Interpretación
Parameter6_3pm_is_nan	-0.0684	0.000	La ausencia de este parámetro también está asociada con menos fallas.
alpha (dispersión)	0.0357	0.000	El valor del parámetro alpha indica una sobredispersión leve pero muy reducida , lo cual sugiere que el uso del modelo binomial negativo podría no ser el más adecuado. En este caso, un modelo de Poisson es mejor ya que se ajusta mejor a los datos. Además, el valor óptimo alpha para la binomial negativa es muy cercano estimado para el modelo de Poisson (alpha_poisson): $\alpha = e^{0.0337} \approx 1.003$ y alpha_binomial negativa: $\alpha = e^{0.0357} \approx 1.003$, lo que valida el procedimiento realizado y refuerza la consistencia de los resultados..

9. Comente los resultados obtenidos en 6, 7 y 8. ¿Cuáles y por qué existen las diferencias entre los resultados?. En su opinión, ¿Cuál sería el más adecuado para responder la pregunta de investigación y por qué? ¿Qué variables resultaron ser robustas a la especificación?

Los resultados obtenidos en las preguntas 6, 7 y 8 fueron fundamentales para comprender mejor las distribuciones involucradas. Los resultados fueron muy satisfactorios, ya que se pudo validar con el valor de **alpha** obtenido en la regresión binomial negativa que el procedimiento realizado es sólido, aunque con posibilidad de mejoras, quizás no se agregaron todas las variables necesarias o se trató con poco cuidado al momento de agrupar, dado que son muchas variables y muchas combinaciones de factores es plausible que se haya podido trabajar mejor, haciendo que los modelos de la pregunta 6 y 8 tuvieron menor sobredispersión y quizás haber llegado a un **alpha** igual a 1. En base a esto, se concluyó que las diferencias observadas en los **Pseudo R²** de los dos modelos es debido a que, al no existir una **sobredispersión significativa** (solo leve), los datos se ajustan mejor a un modelo de **Poisson**.

Por lo tanto, se concluye que, para la pregunta de investigación, es más adecuado utilizar un modelo de **Poisson**, como se explicó previamente. Además, la mayoría de las variables seleccionadas resultaron robustas. Sin embargo, algunas, como **Min_Temp**, podrían cambiar dependiendo del caso, siendo el único modelo donde no resultó robusta el de la **regresión**

binomial negativa.

Notas adicionales:

Reflexión sobre el trabajo

Parte 1

1. Variables categóricas en las preguntas 2, 3 y 4

- Es cierto, como se señaló, que al realizar la categorización por trimestre, ya estaban incluidos los meses, por lo tanto estaba en limbo de lo perjudicial. No se alcanzó a corregir este punto por falta de tiempo, pero tratando de verlo positivamente, esa desagregación permitió especificar el análisis de manera más acotada, como a nivel mensual.
- Además, a partir de esa conversación, concluí que ninguna variable a nivel diario resultó significativa, ya que las demás variables categóricas temporales ya explicaban suficientemente esa dimensión temporal.

Parte 2

2. Manejo de datos y agrupación

- Reconozco que el manejo de los datos en esta parte no fue del todo correcto. Si bien el ramo es una instancia para aprender, admito que el proceso de agrupar y calcular promedios por columna podría haberse hecho de una mejor manera.
- A pesar de eso, lo realizado fue un avance significativo en comparación con lo que se hacía inicialmente, que consistía en eliminar datos, algo que no era apropiado ya que no se podía hacer de forma aleatoria, como se discutió en la reunión.
- Por el tiempo limitado, no se logró aplicar un enfoque más óptimo, pero probablemente habría sido mejor primero reemplazar los valores *Nan* por 0, y luego, si había un valor igual a 1, agrupar a partir de eso.

3. Análisis y comprensión de modelos estadísticos

- Siento que mis análisis en esta segunda parte fueron más simples, ya que comprendí los modelos de Poisson y binomial negativa más desde un enfoque estadístico general, pero no con suficiente profundidad técnica.
- Por ello, me comprometo a estudiar más sobre estos modelos durante el semestre para poder aplicarlos con mayor solidez en el futuro.