

Tarea1_Parra_Mena

April 30, 2025

TAREA 1/ MATEO PARRA MENA/ 2021429077/ 24-04-25

Preguntas y Respuestas:

1. Cargar la base de datos en el ambiente. Identifique los tipos de datos que se encuentran en la base, realice estadísticas descriptivas sobre las variables importantes (Hint: Revisar la distribuciones, datos faltantes, outliers, etc.) y limpie las variables cuando sea necesario.

R: Se cargó la base machine_failure_data.csv y se transformó la variable objetivo Failure_today a binaria (1 si hay fallo, 0 si no). Se identificaron los tipos de datos: principalmente variables numéricas (float64, int64), una variable de fecha (datetime64) y algunas categóricas (object), como las direcciones del viento. Se realizó un análisis exploratorio que reveló variables con altos niveles de valores faltantes, en particular Electricity (42.3%) y Evaporation (37.5%), y otras como Parameter5_9am con un porcentaje menor a 10%. Por ello, se decidió eliminar filas con NaN solo para variables con menos del 10% de datos faltantes. Adicionalmente:

-Se eliminaron directamente las columnas Parameter6_9am y Parameter6_3pm por tener más del 35% de datos faltantes.

-Las variables de dirección cardinal fueron transformadas a ángulos y luego agrupadas en las regiones N, E, S y W.

-Estas nuevas categorías fueron convertidas en dummies (variables ficticias) para usarlas en los modelos.

-Se generaron variables dummy para la variable Location.

-Se añadió información estacional a partir de la fecha, creando variables indicadoras para las estaciones del año (Summer, Autumn, Winter, Spring).

-La variable Leakage fue eliminada más adelante por ser un predictor perfecto del fallo, con un R^2 cercano a 1 que distorsiona los modelos predictivos

-Se realizaron estadísticas descriptivas y gráficos de distribución para variables como Evaporation, Electricity, Min_Temp, Max_Temp, y varios parámetros horarios. Los gráficos permitieron visualizar la distribución de los datos y la presencia de outliers, como fue el caso de Evaporation (distribución sesgada a la derecha) y Leakage, que mostró valores extremos. También se construyó una matriz de correlación excluyendo las dummies de Location y Season para evitar confusión visual. Se descartaron variables redundantes por correlación como Parameter7_9am, Parameter7_3pm y Parameter5_9am.

2. Ejecute un modelo de probabilidad lineal (MCO) que permita explicar la probabilidad de que un día se reporte fallo medido por sensor, a partir de la información disponible. Seleccione

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

R: -Para modelar la probabilidad de que en un día se reporte una falla, se utilizó un modelo de regresión lineal clásico (OLS) donde la variable dependiente es binaria (Failure_today: 1 si hubo falla, 0 en caso contrario). Se seleccionaron como variables explicativas tanto factores climáticos como operacionales, incluyendo temperatura mínima y máxima, presión atmosférica, velocidad del viento, y variables categóricas transformadas a dummies como estación del año y región del viento.

-El modelo alcanza un R^2 de 0.280, lo que implica que aproximadamente el 28% de la variabilidad en la ocurrencia de fallas puede explicarse linealmente por las variables independientes incluidas.

-Entre las variables incluidas, se destacaron Min_Temp, Max_Temp, Parameter1_Speed, Parameter4_9am, y Parameter5_3pm. Se encontraron asociaciones positivas de Min_Temp y Parameter1_Speed con la probabilidad de falla, y negativas para Max_Temp y Parameter5_3pm. Las dummies de dirección cardinal mostraron efectos significativos, especialmente las provenientes del sur y oeste, lo que podría vincularse a patrones climáticos adversos. Este modelo, sin embargo, tiene limitaciones: asume varianza constante (homocedasticidad), no restringe las predicciones al intervalo $[0, 1]$, y no captura relaciones no lineales inherentes a variables binarias.

-Finalmente, las dummies de estación como Season_Spring y Season_Summer fueron positivas y significativas, lo que respalda la presencia de estacionalidad en las fallas. Esto es consistente con la teoría vista en clases sobre la necesidad de incorporar efectos fijos temporales cuando hay patrones cíclicos.

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.

R:-El modelo Probit permitió estimar la probabilidad de falla diaria de manera no lineal, ajustándose mejor a las restricciones del modelo (predicciones entre 0 y 1). Para evitar problemas de matriz singular durante su estimación, se excluyeron las variables Electricity y Evaporation.

-Los resultados muestran que la temperatura mínima se relaciona positivamente con la probabilidad de falla, lo que sugiere que, a medida que aumentan las temperaturas mínimas, podrían generarse condiciones menos óptimas para el funcionamiento del equipo. En contraste, temperaturas máximas más altas reducen la probabilidad de falla, posiblemente por asociarse a ambientes más secos o estables. También se observa que velocidades más altas del parámetro operacional principal están relacionadas con un mayor riesgo de falla, lo que puede explicarse por una mayor exigencia sobre el sistema.

-En relación a los factores ambientales, las direcciones del viento provenientes del sur y del oeste aumentan la probabilidad de fallo, lo que puede indicar que ciertas corrientes de aire o condiciones específicas generan mayor estrés sobre los sensores o componentes de la máquina. Además, se evidencian efectos sistemáticos según la ubicación geográfica de los equipos: varias ubicaciones mostraron ser significativamente distintas en términos de riesgo, lo cual sugiere diferencias operativas o de mantenimiento entre sitios.

-Las estaciones del año también fueron significativas, especialmente primavera y verano, las cuales mostraron un aumento en la probabilidad de fallas. Esto puede vincularse a cambios estacionales en la carga operativa o condiciones térmicas más agresivas

-Para facilitar la interpretación se calcularon los efectos marginales (dy/dx), que representan el cambio en la probabilidad de falla por una variación unitaria en la variable independiente, manteniendo las demás constantes. Por ejemplo, Min_Temp tuvo un efecto marginal de aproximadamente

0.022, lo que significa que un aumento de un grado en la temperatura mínima eleva la probabilidad de falla en 2.2 puntos porcentuales. Estos efectos marginales permiten cuantificar de forma intuitiva el impacto de cada variable en términos de probabilidad, lo cual es particularmente útil en aplicaciones prácticas como mantenimiento predictivo o análisis de riesgo operacional.

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.

-Se estimó un modelo Logit para explicar la probabilidad de que un día ocurra una falla registrada por el sensor, utilizando como variable dependiente *Failure_today* y como variables independientes aquellas previamente seleccionadas tras el proceso de limpieza y transformación de la base. Las variables *Electricity* y *Evaporation* fueron excluidas por generar colinealidad y problemas de matriz singular en los modelos no lineales.

-Los resultados del modelo Logit fueron coherentes con los obtenidos previamente en los modelos MCO y Probit. La variable *Min_Temp* presentó un coeficiente positivo y altamente significativo, lo que indica que temperaturas mínimas más altas están asociadas con una mayor probabilidad de falla. Por el contrario, *Max_Temp* mostró un efecto negativo, lo que sugiere que condiciones más cálidas podrían favorecer el desempeño de los equipos al reducir humedad o condensación.

-Asimismo, *Parameter1_Speed* tuvo un impacto positivo significativo, lo que refuerza la hipótesis de que mayores velocidades operativas incrementan la exigencia sobre los sistemas mecánicos, elevando el riesgo de falla. También se observó un patrón horario en *Parameter3_9am* (positivo) y *Parameter3_3pm* (negativo), lo que puede reflejar condiciones distintas de operación o ambiente a lo largo del día.

-Respecto a las direcciones cardinales del viento, se observó que las provenientes del oeste (*Parameter1_Dir_region_W*, *Parameter2_9am_region_W*, *Parameter2_3pm_region_W*) y del sur (*Parameter2_9am_region_S*, *Parameter2_3pm_region_S*) aumentan la probabilidad de falla, y todos estos efectos fueron altamente significativos. Esto podría relacionarse con patrones climáticos regionales que afectan negativamente el funcionamiento de la maquinaria.

-Las variables dummy de *Location* mostraron una fuerte heterogeneidad entre estaciones, revelando diferencias significativas en el riesgo de falla según ubicación, posiblemente debido a variaciones locales en mantenimiento, exposición ambiental o carga operativa. Algunas locaciones como la 6, 13, 20 y 49 presentaron coeficientes negativos muy pronunciados, lo cual indica una menor probabilidad de falla respecto al grupo base.

-Las estaciones del año también fueron significativas. En particular, *Spring* y *Summer* mostraron coeficientes positivos y altamente significativos, lo que sugiere una mayor frecuencia de fallas en esos períodos. *Winter*, en cambio, tuvo un efecto negativo. A diferencia del modelo anterior, esta vez sí se reportaron errores estándar válidos, lo que permite interpretar estas relaciones con mayor confianza estadística.

-Finalmente, se calcularon los efectos marginales (dy/dx), los cuales cuantifican el cambio esperado en la probabilidad de falla ante una variación unitaria en cada variable independiente, manteniendo constantes las demás. Por ejemplo, el efecto marginal de *Min_Temp* fue de aproximadamente 0.022, lo que implica que un aumento de un grado en la temperatura mínima incrementa la probabilidad de falla en 2.2 puntos porcentuales. Esta interpretación directa es especialmente útil para fines prácticos y para comunicar hallazgos a tomadores de decisiones técnicos o no técnicos.

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?

-Los tres modelos estimados MCO, Probit y Logit arrojaron resultados en general consistentes en cuanto a la significancia estadística y el signo de los coeficientes, lo cual refuerza la robustez de ciertas variables explicativas clave. No obstante, existen diferencias relevantes en la forma en que cada modelo interpreta y modela la relación entre las variables independientes y la probabilidad de falla diaria (Failure_today), que es una variable binaria.

-El modelo MCO, aunque es simple e intuitivo, tiene limitaciones importantes en este contexto. Su principal fortaleza es que permite interpretar los coeficientes directamente como cambios en la probabilidad ante una unidad de cambio en cada predictor. Sin embargo, viola supuestos fundamentales del análisis de variables binarias: puede predecir probabilidades fuera del rango [0,1], asume homocedasticidad, y no captura no linealidades inherentes al fenómeno. Aun así, sirve como punto de partida para validar la dirección e importancia relativa de los efectos.

-El modelo Probit mejora esta situación modelando la probabilidad de manera no lineal, utilizando la distribución normal estándar acumulada. Esto asegura que las predicciones estén contenidas en el intervalo [0,1]. Los coeficientes no se interpretan directamente como cambios porcentuales, pero se pueden interpretar a través de sus efectos marginales (dy/dx). En este caso, por ejemplo, un aumento de 1 grado en Min_Temp eleva la probabilidad de falla en aproximadamente 2.2 puntos porcentuales. Los resultados de Probit confirmaron que variables como Min_Temp, Max_Temp, Parameter1_Speed y Parameter5_3pm son robustas y estadísticamente significativas.

-El modelo Logit, por su parte, utiliza la distribución logística como función de enlace. Aunque los coeficientes tienen una escala distinta a Probit, los resultados fueron muy similares en términos de significancia y dirección. Al igual que en el modelo Probit, se calcularon efectos marginales, lo que permitió cuantificar de forma clara el impacto de cada variable. Por ejemplo, Max_Temp presentó un efecto marginal negativo de -2.7%, consistente con las otras especificaciones. Logit también mostró un ajuste muy aceptable, con un Pseudo R^2 cercano a 0.33, lo cual es alto en modelos binarios.

-Un aspecto a destacar es que las variables estacionales (Season_Spring, Season_Summer, Season_Winter) mostraron significancia estadística en Probit y Logit, con errores estándar válidos. Esto indica que la colinealidad que antes se sospechaba fue probablemente corregida al eliminar una categoría base.

-En todos los modelos, las variables Min_Temp, Max_Temp, Parameter1_Speed, Parameter5_3pm, así como varias direcciones del viento (especialmente oeste y sur en horarios AM y PM), se mantuvieron significativas. Estas variables son, por tanto, robustas a la especificación del modelo.

-Conclusión: Aunque MCO es útil como primera aproximación, el modelo Logit es el más apropiado para responder la pregunta de investigación. Captura correctamente la naturaleza binaria de la variable dependiente, permite interpretar efectos marginales y mostró mayor poder explicativo. Además, mantiene la consistencia con los otros modelos, lo que aporta robustez a las conclusiones.

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.

R: Se transformaron los datos diarios a formato mensual, promediando las variables numéricas por mes y estación (Location) y sumando los días con fallos (Failure_today). Se eliminó cualquier tipo de promediación de variables categóricas. Se aplicó un modelo Poisson para modelar el número de fallas por mes. Se utilizó C(Location) para controlar heterogeneidad espacial. El modelo presentó buena significancia global y coeficientes esperables: Min_Temp y Parameter1_Speed tuvieron efectos positivos sobre los fallos, mientras que Max_Temp y Parameter5_3pm efectos negativos. Esto es coherente con la literatura del curso: condiciones climáticas extremas y sobreexigencia operativa incrementan la tasa de fallos.

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

R: Para evaluar sobre-dispersión se utilizó el método propuesto en clases: se estimó el valor esperado λ del modelo Poisson y se graficó su distribución. Luego se construyó la variable auxiliar $((y - \lambda)^2 - \lambda)/\lambda$ y se regresó sobre λ usando MCO. El coeficiente de esta regresión representa una estimación de la sobre-dispersión (α). En este caso, se obtuvo un valor pequeño y no significativo ($p=0.107$), lo que sugiere ausencia de sobre-dispersión fuerte. Esto implica que el modelo Poisson no estaría subestimando la varianza, y podría ser adecuado, aunque se validó en la siguiente pregunta.

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.

R: Se ajustó un modelo Binomial Negativa con los mismos predictores del modelo Poisson. Este modelo relaja la restricción de varianza igual a la media (característica del Poisson), introduciendo un parámetro de dispersión (α). El valor de α fue estimado automáticamente. Se confirmó que el modelo mejora el ajuste (mayor log-verosimilitud), aunque el valor de α fue bajo. Se graficó la relación entre fallas observadas y predichas (Failure_today vs ypred), mostrando un patrón lineal que respalda la capacidad predictiva del modelo. Los coeficientes fueron similares a los del Poisson, reafirmando la robustez de las variables seleccionadas.

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?

R: -Los modelos estimados en las preguntas 6, 7 y 8 :Poisson, test de sobre-dispersión y Binomial Negativa permitieron evaluar con mayor precisión el comportamiento de la variable dependiente Failure_today agregada a nivel mensual. La comparación entre estos enfoques es clave para determinar la especificación más adecuada en contextos de conteo de eventos.

-El modelo Poisson asumió, como es estándar, que la media y la varianza de la variable dependiente son iguales. Este modelo se ajustó razonablemente bien a la data, entregando coeficientes significativos y consistentes con las interpretaciones obtenidas en los modelos individuales (MCO, Probit, Logit). Sin embargo, esta especificación puede ser sensible a problemas de sobre-dispersión, es decir, cuando la varianza de los datos es mayor que su media, lo cual podría sesgar las inferencias.

-En la pregunta 7, se aplicó el test de sobre-dispersión propuesto en clases, que consiste en estimar la esperanza (λ) del modelo Poisson, construir la variable auxiliar $((y - \lambda)^2 - \lambda)/\lambda$ y regresarla sobre λ usando MCO. El coeficiente estimado en esta regresión (una aproximación de α) resultó pequeño y no estadísticamente significativo ($p = 0.107$), lo que

sugiere una baja sobre-dispersión en la muestra analizada. Esto indicaría que el modelo Poisson no subestima la varianza de forma importante y podría ser aceptable en este caso.

-Pese a lo anterior, en la pregunta 8 se estimó un modelo de Binomial Negativa, que relaja la restricción de varianza igual a la media al introducir un parámetro de dispersión (α) que se ajusta automáticamente. Este modelo entregó un ajuste superior (mayor log-verosimilitud y menor devianza), y reveló la existencia de heterogeneidad no explicada por el modelo Poisson, aunque leve. Esto se traduce en una mayor robustez del modelo Binomial frente a posibles errores de especificación.

-Ambos modelos coincidieron en la dirección y significancia de varias variables clave, lo que confirma la robustez de ciertas covariables. En particular, se destacaron nuevamente Min_Temp (relación positiva), Max_Temp (negativa), Parameter1_Speed, y Parameter5_3pm. También las dummies de ubicación geográfica (Location) mantuvieron su relevancia explicativa, indicando diferencias sistemáticas entre estaciones. Las variables estacionales (especialmente primavera y verano) también fueron significativas, mostrando una clara componente temporal en la ocurrencia de fallos.

-Conclusión: Aunque el modelo Poisson fue razonable en este contexto por la ausencia de fuerte sobre-dispersión, el modelo Binomial Negativa es preferible por su mayor flexibilidad estadística y mejor capacidad de ajuste frente a posibles heterogeneidades no capturadas por la media. Esto lo convierte en una opción más robusta para responder la pregunta de investigación cuando se modelan conteos de fallas agregadas por mes.

```
[1]: import numpy as np
import pandas as pd
from IPython.display import display, HTML
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 warnings
warnings.filterwarnings("ignore")
%matplotlib inline
print("Librerías cargadas correctamente.")
```

Librerías cargadas correctamente.

1. Cargar la base de datos en el ambiente. Identifique los tipos de datos que se encuentran en la base, realice estadísticas descriptivas sobre las variables importantes (Hint: Revisar la distribuciones, datos faltantes, outliers, etc.) y limpie las variables cuando sea necesario.

```
[2]: # Cargar los datos
df = pd.read_csv('machine_failure_data.csv')

# Convertir columna de fecha
```

```

df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

# Convertir variable objetivo a binaria
df['Failure_today'] = df['Failure_today'].map({'Yes': 1, 'No': 0})

# Identificar columnas por tipo de dato
float_cols = df.select_dtypes(include='float64').columns.tolist()
int_cols = df.select_dtypes(include='int64').columns.tolist()
obj_cols = df.select_dtypes(include='object').columns.tolist()
date_cols = df.select_dtypes(include='datetime64').columns.tolist()

# Crear DataFrame resumen
tipo_dato = ['float64'] * len(float_cols) + ['int64'] * len(int_cols) + \
            ['object'] * len(obj_cols) + ['datetime64'] * len(date_cols)
columnas = float_cols + int_cols + obj_cols + date_cols
resumen_tipos = pd.DataFrame({'Columna': columnas, 'Tipo de Dato': tipo_dato})

# Mostrar resumen en texto
print("Resumen de tipos de datos por columna:\n")
print(f"- Columnas tipo float64 ({len(float_cols)}): {' '.join(float_cols)}")
print(f"- Columnas tipo int64 ({len(int_cols)}): {' '.join(int_cols)}")
print(f"- Columnas tipo object ({len(obj_cols)}): {' '.join(obj_cols)}")
print(f"- Columnas tipo datetime64 ({len(date_cols)}): {' '.join(date_cols)}")
print("""\nObservación:\n
- La base contiene principalmente variables numéricas (float64 e int64).
- La columna 'Date' es de tipo datetime y no será utilizada directamente como
  variable explicativa.
- La variable 'Failure_today' se transforma a binaria""")

```

Resumen de tipos de datos por columna:

- Columnas tipo float64 (17): 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
- Columnas tipo int64 (1): Location
- Columnas tipo object (3): Parameter1_Dir, Parameter2_9am, Parameter2_3pm
- Columnas tipo datetime64 (1): Date

Observación:

- La base contiene principalmente variables numéricas (float64 e int64).
- La columna 'Date' es de tipo datetime y no será utilizada directamente como variable explicativa.
- La variable 'Failure_today' se transforma a binaria

```
[3]: print("Revisamos cómo es el DataFrame:")
df
```

Revisamos cómo es el DataFrame:

```
[3]:
```

	Date	Location	Min_Temp	Max_Temp	Leakage	Evaporation	\
0	2008-12-01	3	13.4	22.9	0.6	NaN	
1	2008-12-02	3	7.4	25.1	0.0	NaN	
2	2008-12-03	3	12.9	25.7	0.0	NaN	
3	2008-12-04	3	9.2	28.0	0.0	NaN	
4	2008-12-05	3	17.5	32.3	1.0	NaN	
...	
142188	2017-06-20	42	3.5	21.8	0.0	NaN	
142189	2017-06-21	42	2.8	23.4	0.0	NaN	
142190	2017-06-22	42	3.6	25.3	0.0	NaN	
142191	2017-06-23	42	5.4	26.9	0.0	NaN	
142192	2017-06-24	42	7.8	27.0	0.0	NaN	

	Electricity	Parameter1_Dir	Parameter1_Speed	Parameter2_9am	...	\
0	NaN	W	44.0	W	...	
1	NaN	WNW	44.0	NNW	...	
2	NaN	WSW	46.0	W	...	
3	NaN	NE	24.0	SE	...	
4	NaN	W	41.0	ENE	...	
...	
142188	NaN	E	31.0	ESE	...	
142189	NaN	E	31.0	SE	...	
142190	NaN	NNW	22.0	SE	...	
142191	NaN	N	37.0	SE	...	
142192	NaN	SE	28.0	SSE	...	

	Parameter3_3pm	Parameter4_9am	Parameter4_3pm	Parameter5_9am	\
0	24.0	71.0	22.0	1007.7	
1	22.0	44.0	25.0	1010.6	
2	26.0	38.0	30.0	1007.6	
3	9.0	45.0	16.0	1017.6	
4	20.0	82.0	33.0	1010.8	
...	
142188	13.0	59.0	27.0	1024.7	
142189	11.0	51.0	24.0	1024.6	
142190	9.0	56.0	21.0	1023.5	
142191	9.0	53.0	24.0	1021.0	
142192	7.0	51.0	24.0	1019.4	

	Parameter5_3pm	Parameter6_9am	Parameter6_3pm	Parameter7_9am	\
0	1007.1	8.0	NaN	16.9	
1	1007.8	NaN	NaN	17.2	
2	1008.7	NaN	2.0	21.0	

3	1012.8	NaN	NaN	18.1
4	1006.0	7.0	8.0	17.8
...
142188	1021.2	NaN	NaN	9.4
142189	1020.3	NaN	NaN	10.1
142190	1019.1	NaN	NaN	10.9
142191	1016.8	NaN	NaN	12.5
142192	1016.5	3.0	2.0	15.1

	Parameter7_3pm	Failure_today
0	21.8	0.0
1	24.3	0.0
2	23.2	0.0
3	26.5	0.0
4	29.7	0.0
...
142188	20.9	0.0
142189	22.4	0.0
142190	24.5	0.0
142191	26.1	0.0
142192	26.0	0.0

[142193 rows x 22 columns]

```
[4]: # Seleccionar columnas numéricas
numeric_vars = df.select_dtypes(include=['float64', 'int64']).columns.tolist()

# Estadísticas descriptivas de todas las variables numéricas
summary_stats = df[numeric_vars].describe().T.round(2)

# Valores faltantes en todas las columnas
missing_all = df.isnull().sum()
missing_df = pd.DataFrame({
    'Valores Faltantes': missing_all,
    'Porcentaje (%)': (missing_all / len(df) * 100).round(2)
}).sort_values(by='Porcentaje (%)', ascending=False)

# Mostrar ambas tablas lado a lado
html_summary = summary_stats.to_html(classes='table table-striped', border=1)
html_missing = missing_df.to_html(classes='table table-striped', border=1)

html_combined = f"""
<div style="display: flex; justify-content: space-between;">
    <div style="flex: 1; margin-right: 20px;">
        <h4>Estadísticas Descriptivas (Todas las variables numéricas)</h4>
        {html_summary}
    </div>
```

```

        <div style="flex: 1;">
            <h4>Valores Faltantes</h4>
            {html_missing}
        </div>
    </div>
    """

display(HTML(html_combined))

# Conclusiones
print("Conclusiones:")
print("- Algunas variables tienen un porcentaje muy alto de datos faltantes_
    ↳(>35%) como 'Electricity', 'Evaporation' y los parámetros 6.")
print("- Variables como 'Parameter5_9am' y 'Parameter5_3pm' tienen menos del_
    ↳10% de faltantes, por lo tanto es razonable hacer dropna() sobre esas.")

```

<IPython.core.display.HTML object>

Conclusiones:

- Algunas variables tienen un porcentaje muy alto de datos faltantes (>35%) como 'Electricity', 'Evaporation' y los parámetros 6.
- Variables como 'Parameter5_9am' y 'Parameter5_3pm' tienen menos del 10% de faltantes, por lo tanto es razonable hacer dropna() sobre esas.

```

[5]: # Eliminar columnas irrelevantes
df.drop(columns=['Parameter6_9am', 'Parameter6_3pm'], inplace=True)

# Diccionario que convierte direcciones cardinales a ángulos en grados
direccion_a_angulo = {
    'N': 0, 'NNE': 22.5, 'NE': 45, 'ENE': 67.5, 'E': 90,
    'ESE': 112.5, 'SE': 135, 'SSE': 157.5, 'S': 180,
    'SSW': 202.5, 'SW': 225, 'WSW': 247.5, 'W': 270,
    'WNW': 292.5, 'NW': 315, 'NNW': 337.5
}

# Convertir columnas de dirección a ángulos
df['Parameter1_Dir_angle'] = df['Parameter1_Dir'].map(direccion_a_angulo)
df['Parameter2_9am_angle'] = df['Parameter2_9am'].map(direccion_a_angulo)
df['Parameter2_3pm_angle'] = df['Parameter2_3pm'].map(direccion_a_angulo)

# Rellenar NaN con 0
df[['Parameter1_Dir_angle', 'Parameter2_9am_angle', 'Parameter2_3pm_angle']] =_
    ↳df[
        ['Parameter1_Dir_angle', 'Parameter2_9am_angle', 'Parameter2_3pm_angle']
    ].fillna(0)

# Función para agrupar en regiones cardinales
def agrupar_direccion(angle):

```

```

if (angle >= 315 or angle < 45):
    return 'N'
elif (angle >= 45 and angle < 135):
    return 'E'
elif (angle >= 135 and angle < 225):
    return 'S'
elif (angle >= 225 and angle < 315):
    return 'W'
else:
    return 'Unknown' # si se desconoce

# Crear columnas '_region' a partir de ángulos
columnas_angulos = ['Parameter1_Dir_angle', 'Parameter2_9am_angle',
                    ↪ 'Parameter2_3pm_angle']
for col in columnas_angulos:
    nueva_col = col.replace('_angle', '_region')
    df[nueva_col] = df[col].apply(agrupar_direccion)

# Eliminar columnas originales e intermedias
df.drop(columns=columnas_angulos + ['Parameter1_Dir', 'Parameter2_9am',
    ↪ 'Parameter2_3pm'], inplace=True)

# Crear variables dummy desde las regiones
region_cols = ['Parameter1_Dir_region', 'Parameter2_9am_region',
               ↪ 'Parameter2_3pm_region']
df_dummies = pd.get_dummies(df[region_cols], prefix=region_cols,
    ↪ drop_first=True)

# Concatenar dummies y eliminar originales
df = pd.concat([df, df_dummies], axis=1)
df.drop(columns=region_cols, inplace=True)

# Seleccionar columnas numéricas nuevamente por seguridad
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Calcular porcentaje de valores nulos por columna
nan_pct = df[numeric_cols].isnull().mean() * 100

# Filtrar las columnas con 10% de valores nulos
cols_con_menos_de_10pct = nan_pct[nan_pct <= 10].index.tolist()

# Eliminar filas con NaN solo en esas columnas
df.dropna(subset=cols_con_menos_de_10pct, inplace=True)

# Eliminar posibles filas restantes con NaN en Failure_today
df = df[df['Failure_today'].notna()]

```

```

# Transformar a dummies la variable Location
df = pd.get_dummies(df, columns=['Location'], drop_first=True)

# Copia del DataFrame limpio
dfcopy1 = df.copy()

# Asegurar que 'Date' esté en formato datetime
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Función para obtener estación según el mes
def get_season(month):
    if month in [12, 1, 2]:
        return 'Summer'
    elif month in [3, 4, 5]:
        return 'Autumn'
    elif month in [6, 7, 8]:
        return 'Winter'
    elif month in [9, 10, 11]:
        return 'Spring'
    else:
        return 'Unknown'

# Crear columna de estación
df['Season'] = df['Date'].dt.month.apply(get_season)

# Crear variables dummy para estaciones y eliminar una como pivote (categoría_
↳base)
season_dummies = pd.get_dummies(df['Season'], prefix='Season', drop_first=True).
↳astype(int)

# Concatenar al DataFrame original y eliminar columna 'Season'
df = pd.concat([df, season_dummies], axis=1)
df.drop(columns=['Season'], inplace=True)

# Eliminar columnas auxiliares
df.drop(columns=['Date'], inplace=True)

# Asegurar que las columnas booleanas sean enteros
df = df.astype({col: int for col in df.select_dtypes(include='bool').columns})
dfcopy1 = dfcopy1.astype({col: int for col in dfcopy1.
↳select_dtypes(include='bool').columns})

```

```
[6]: df
```

```

[6]:      Min_Temp  Max_Temp  Leakage  Evaporation  Electricity  \
0          13.4      22.9      0.6           NaN           NaN
1           7.4      25.1      0.0           NaN           NaN

```

2	12.9	25.7	0.0	NaN	NaN
3	9.2	28.0	0.0	NaN	NaN
4	17.5	32.3	1.0	NaN	NaN
...
142188	3.5	21.8	0.0	NaN	NaN
142189	2.8	23.4	0.0	NaN	NaN
142190	3.6	25.3	0.0	NaN	NaN
142191	5.4	26.9	0.0	NaN	NaN
142192	7.8	27.0	0.0	NaN	NaN

	Parameter1_Speed	Parameter3_9am	Parameter3_3pm	Parameter4_9am	\
0	44.0	20.0	24.0	71.0	
1	44.0	4.0	22.0	44.0	
2	46.0	19.0	26.0	38.0	
3	24.0	11.0	9.0	45.0	
4	41.0	7.0	20.0	82.0	
...	
142188	31.0	15.0	13.0	59.0	
142189	31.0	13.0	11.0	51.0	
142190	22.0	13.0	9.0	56.0	
142191	37.0	9.0	9.0	53.0	
142192	28.0	13.0	7.0	51.0	

	Parameter4_3pm	...	Location_43	Location_44	Location_45	\
0	22.0	...	0	0	0	
1	25.0	...	0	0	0	
2	30.0	...	0	0	0	
3	16.0	...	0	0	0	
4	33.0	...	0	0	0	
...	
142188	27.0	...	0	0	0	
142189	24.0	...	0	0	0	
142190	21.0	...	0	0	0	
142191	24.0	...	0	0	0	
142192	24.0	...	0	0	0	

	Location_46	Location_47	Location_48	Location_49	Season_Spring	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
...	
142188	0	0	0	0	0	
142189	0	0	0	0	0	
142190	0	0	0	0	0	
142191	0	0	0	0	0	

142192	0	0	0	0	0
--------	---	---	---	---	---

	Season_Summer	Season_Winter
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...
142188	0	1
142189	0	1
142190	0	1
142191	0	1
142192	0	1

[119590 rows x 70 columns]

```
[7]: # Recalcular valores faltantes para todas las columnas y mostrarlos en tabla
      ↪ sin los de 0
missing_all = df.isnull().sum()
missing_filtered = missing_all[missing_all > 0]
missing_df = pd.DataFrame({
    'Valores Faltantes': missing_filtered,
    'Porcentaje (%)': (missing_filtered / len(df) * 100).round(2)
}).sort_values(by='Porcentaje (%)', ascending=False)

display(missing_df)
print("Nos percatamos que nos quedan dos variables con NaN, posteriormente en
      ↪ los modelos veremos qué haremos con ellas.")
```

	Valores Faltantes	Porcentaje (%)
Electricity	50559	42.28
Evaporation	44826	37.48

Nos percatamos que nos quedan dos variables con NaN, posteriormente en los modelos veremos qué haremos con ellas.

```
[8]: # Filtrar columnas numéricas que no sean dummies de Location
numeric_cols_filtradas = [col for col in df.select_dtypes(include=['float64',
      ↪ 'int64']).columns if not col.startswith('Location_')]

# Crear matriz de correlación filtrada
corr_filtrada = df[numeric_cols_filtradas].corr()

# Crear la máscara para ocultar la mitad superior
mask = np.triu(np.ones_like(corr_filtrada, dtype=bool))

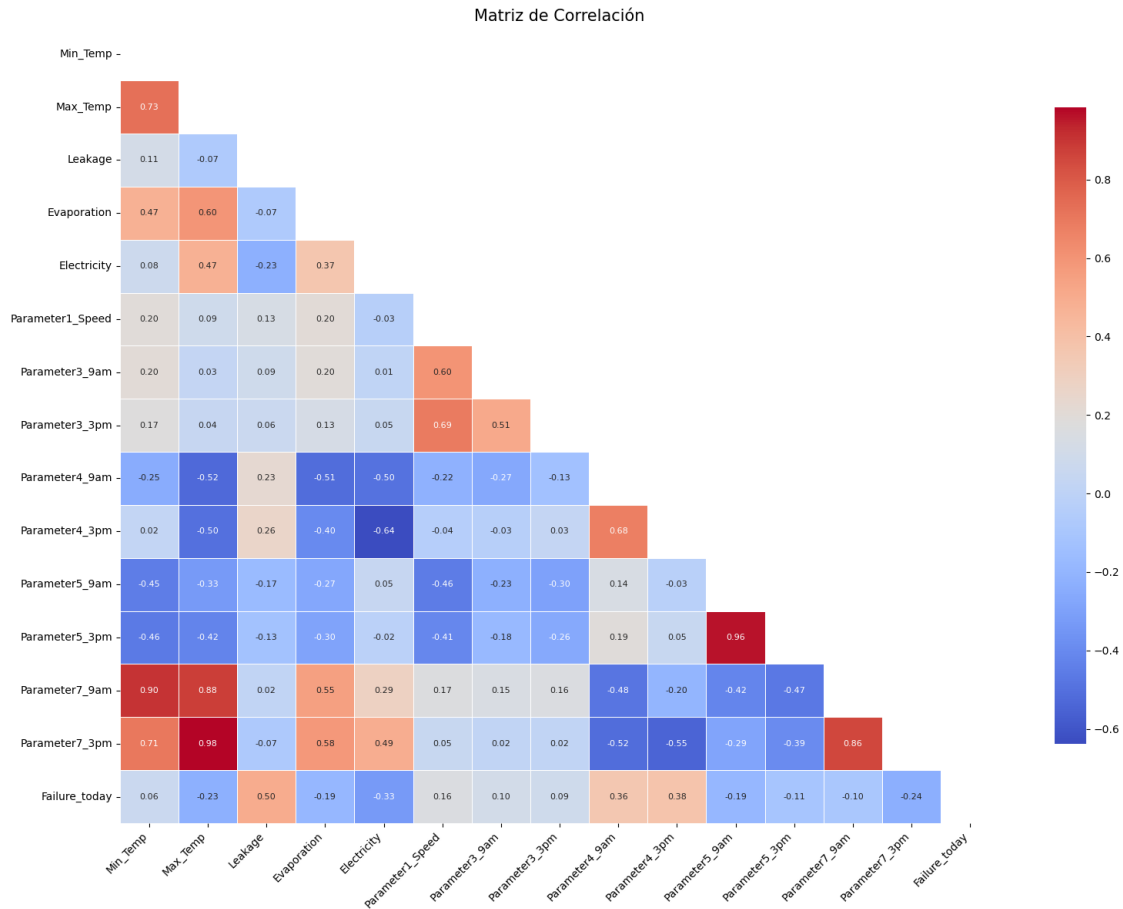
# Visualizar la matriz de correlación
```

```

plt.figure(figsize=(16, 12)) # Tamaño amplio para mejor lectura
sns.heatmap(
    corr_filtrada,
    annot=True,
    fmt=".2f",
    cmap='coolwarm',
    mask=mask,
    linewidths=0.5,
    annot_kws={'size': 8},
    cbar_kws={"shrink": .8}
)
plt.title('Matriz de Correlación', fontsize=15)
plt.xticks(rotation=45, ha='right') # Rota etiquetas para evitar encimamiento
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()

# Eliminación de variables altamente correlacionadas y redundantes
df.drop(columns=["Parameter7_9am", "Parameter7_3pm", "Parameter5_9am"],
        inplace=True)
dfcopy1.drop(columns=["Parameter7_9am", "Parameter7_3pm", "Parameter5_9am"],
             inplace=True)
print(" Variables eliminadas por alta correlación y redundancia:")
print("- 'Parameter7_9am' y 'Parameter7_3pm': Altamente correlacionadas con
        ↪Min_Temp y Max_Temp, respectivamente.")
print("- 'Parameter5_9am': Muy correlacionada con 'Parameter5_3pm', y esta
        ↪última es más representativa de la presión a lo largo del día.")

```



Variables eliminadas por alta correlación y redundancia:

- 'Parameter7_9am' y 'Parameter7_3pm': Altamente correlacionadas con Min_Temp y Max_Temp, respectivamente.
- 'Parameter5_9am': Muy correlacionada con 'Parameter5_3pm', y esta última es más representativa de la presión a lo largo del día.

```
[9]: fig, axes = plt.subplots(2, 3, figsize=(18, 8)) # 2 filas, 3 columnas

# Evaporation
sns.histplot(df['Evaporation'], bins=40, kde=True, ax=axes[0, 0],
             color='skyblue')
axes[0, 0].set_title('Distribución: Evaporation')

# Electricity
sns.histplot(df['Electricity'], bins=40, kde=True, ax=axes[0, 1],
             color='salmon')
axes[0, 1].set_title('Distribución: Electricity')

# Parameter5_3pm
```



```

sns.boxplot(data=df, x='Parameter5_3pm', ax=axes[0, 2], color='lightgreen')
axes[0, 2].set_title('Boxplot: Parameter5_3pm')

# Min y Max Temp
sns.kdeplot(df['Min_Temp'], ax=axes[1, 0], fill=True, label='Min_Temp')
sns.kdeplot(df['Max_Temp'], ax=axes[1, 0], fill=True, label='Max_Temp')
axes[1, 0].set_title('Distribución: Min y Max Temp')
axes[1, 0].legend()

# Parameter3_9am y 3pm
sns.boxplot(data=df[['Parameter3_9am', 'Parameter3_3pm']], ax=axes[1, 1])
axes[1, 1].set_title('Boxplot: Parameter3_9am & 3pm')

# Leakage
sns.boxplot(data=df, x='Leakage', ax=axes[1, 2], color='lightcoral')
axes[1, 2].set_title('Boxplot: Leakage')

plt.tight_layout()
plt.show()

print("""
Observaciones a partir de los gráficos:

- La variable 'Evaporation' presenta una distribución sesgada a la derecha con
  ↪ varios outliers visibles, lo que indica que hay días con una evaporación
  ↪ anormalmente alta.

- 'Electricity' muestra un patrón de distribución más disperso y algunos
  ↪ valores elevados por encima de 12, que pueden representar situaciones
  ↪ inusuales de consumo energético.

- El boxplot de 'Parameter5_3pm' revela outliers en ambos extremos, aunque el
  ↪ grueso de los datos se concentra en un rango estable.

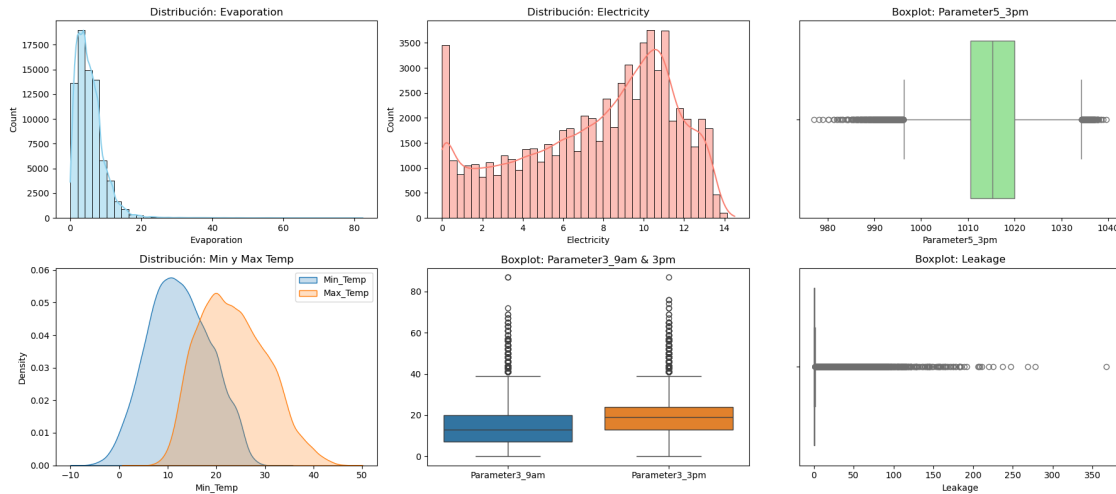
- La comparación entre 'Min_Temp' y 'Max_Temp' permite observar claramente la
  ↪ diferencia entre temperaturas mínimas y máximas diarias.

- 'Parameter3_9am' y 'Parameter3_3pm' muestran una cantidad significativa de
  ↪ outliers, lo cual podría indicar sensores sensibles o condiciones extremas
  ↪ ocasionales.

- Se eliminó la variable 'Leakage' porque actúa como un predictor casi perfecto
  ↪ del fallo de la máquina, generando una estimación con  $R^2$  cercano a 1. Esto
  ↪ indica que su inclusión contamina el modelo, ya que anticipa directamente la
  ↪ variable objetivo, distorsionando la interpretación y la contribución de las
  ↪ demás variables explicativas.""")

# Eliminar 'Leakage' después de graficar
df.drop(columns=['Leakage'], inplace=True)
dfcopy1.drop(columns=['Leakage'], inplace=True)

```



Observaciones a partir de los gráficos:

- La variable 'Evaporation' presenta una distribución sesgada a la derecha con varios outliers visibles, lo que indica que hay días con una evaporación anormalmente alta.
- 'Electricity' muestra un patrón de distribución más disperso y algunos valores elevados por encima de 12, que pueden representar situaciones inusuales de consumo energético.
- El boxplot de 'Parameter5_3pm' revela outliers en ambos extremos, aunque el grueso de los datos se concentra en un rango estable.
- La comparación entre 'Min_Temp' y 'Max_Temp' permite observar claramente la diferencia entre temperaturas mínimas y máximas diarias.
- 'Parameter3_9am' y 'Parameter3_3pm' muestran una cantidad significativa de outliers, lo cual podría indicar sensores sensibles o condiciones extremas ocasionales.
- Se eliminó la variable 'Leakage' porque actúa como un predictor casi perfecto del fallo de la máquina, generando una estimación con R^2 cercano a 1. Esto indica que su inclusión contamina el modelo, ya que anticipa directamente la variable objetivo, distorsionando la interpretación y la contribución de las demás variables explicativas.

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

```
[10]: # Crear el DataFrame modelo a partir del DataFrame limpio actual
df_model = df.copy()
# y: variable dependiente (lo que queremos predecir)
y = df_model['Failure_today']
```

```

# X: todas las demás variables (excepto la variable objetivo)
X = df_model.drop(columns=['Failure_today'])

# Agregamos constante (intercepto) al modelo
X = sm.add_constant(X)

# Eliminamos filas con posibles NaN (por seguridad)
Xy = pd.concat([y, X], axis=1).dropna()
y_clean = Xy['Failure_today']
X_clean = Xy.drop(columns=['Failure_today'])

# Ejecutamos el modelo
modelo = sm.OLS(y_clean, X_clean).fit()

# Mostramos el resumen
print(modelo.summary())

```

OLS Regression Results

```

=====
Dep. Variable:          Failure_today    R-squared:                0.280
Model:                  OLS              Adj. R-squared:          0.279
Method:                 Least Squares    F-statistic:              518.1
Date:                   Thu, 24 Apr 2025  Prob (F-statistic):       0.00
Time:                   23:02:41          Log-Likelihood:           -24513.
No. Observations:       65434            AIC:                     4.913e+04
Df Residuals:           65384            BIC:                     4.958e+04
Df Model:                49
Covariance Type:        nonrobust
=====

```

```

=====
                                coef    std err          t      P>|t|      [0.025
0.975]
-----
const                5.9395      0.287     20.723      0.000      5.378
6.501
Min_Temp              0.0166      0.001     28.202      0.000      0.015
0.018
Max_Temp             -0.0145      0.001    -24.980      0.000     -0.016
-0.013
Evaporation          -0.0133      0.001    -22.385      0.000     -0.014
-0.012
Electricity          -0.0049      0.001     -9.030      0.000     -0.006
-0.004
Parameter1_Speed      0.0058      0.000     32.462      0.000      0.005
0.006
Parameter3_9am        0.0034      0.000     14.595      0.000      0.003
0.004

```

Parameter3_3pm -0.003	-0.0035	0.000	-14.016	0.000	-0.004
Parameter4_9am 0.007	0.0069	0.000	55.631	0.000	0.007
Parameter4_3pm 0.001	0.0005	0.000	3.883	0.000	0.000
Parameter5_3pm -0.006	-0.0061	0.000	-22.049	0.000	-0.007
Parameter1_Dir_region_N -0.001	-0.0119	0.005	-2.235	0.025	-0.022
Parameter1_Dir_region_S 0.023	0.0130	0.005	2.662	0.008	0.003
Parameter1_Dir_region_W 0.037	0.0267	0.005	5.031	0.000	0.016
Parameter2_9am_region_N -0.000	-0.0091	0.004	-2.025	0.043	-0.018
Parameter2_9am_region_S 0.029	0.0199	0.005	4.268	0.000	0.011
Parameter2_9am_region_W 0.071	0.0609	0.005	12.090	0.000	0.051
Parameter2_3pm_region_N 0.012	0.0021	0.005	0.399	0.690	-0.008
Parameter2_3pm_region_S 0.056	0.0466	0.005	9.743	0.000	0.037
Parameter2_3pm_region_W 0.062	0.0520	0.005	9.818	0.000	0.042
Location_3 -3.15e-17	-6.346e-17	1.63e-17	-3.886	0.000	-9.55e-17
Location_4 0.137	0.1112	0.013	8.578	0.000	0.086
Location_5 9.62e-19	-3.506e-17	1.84e-17	-1.908	0.056	-7.11e-17
Location_6 2.8e-16	2.526e-16	1.41e-17	17.888	0.000	2.25e-16
Location_7 -5.46e-17	-7.341e-17	9.62e-18	-7.630	0.000	-9.23e-17
Location_8 0.016	-0.0092	0.013	-0.725	0.468	-0.034
Location_9 -0.030	-0.0567	0.014	-4.148	0.000	-0.083
Location_10 -0.056	-0.0849	0.014	-5.861	0.000	-0.113
Location_11 0.000	-0.0359	0.018	-1.954	0.051	-0.072
Location_12 -0.016	-0.0442	0.014	-3.062	0.002	-0.073
Location_13 -0.096	-0.1222	0.013	-9.307	0.000	-0.148

Location_14 -0.070	-0.0959	0.013	-7.250	0.000	-0.122
Location_15 3.05e-17	1.988e-17	5.41e-18	3.671	0.000	9.27e-18
Location_16 -0.116	-0.1402	0.013	-11.189	0.000	-0.165
Location_17 6.64e-17	5.787e-17	4.37e-18	13.241	0.000	4.93e-17
Location_18 3.88e-17	2.683e-17	6.11e-18	4.395	0.000	1.49e-17
Location_19 -0.092	-0.1175	0.013	-9.053	0.000	-0.143
Location_20 -0.134	-0.1588	0.013	-12.558	0.000	-0.184
Location_21 -0.060	-0.0844	0.013	-6.748	0.000	-0.109
Location_22 -0.007	-0.0329	0.013	-2.484	0.013	-0.059
Location_23 -0.067	-0.0927	0.013	-7.186	0.000	-0.118
Location_26 -6.67e-17	-7.41e-17	3.78e-18	-19.601	0.000	-8.15e-17
Location_27 2.24e-17	1.53e-17	3.6e-18	4.247	0.000	8.24e-18
Location_28 -0.134	-0.1593	0.013	-12.247	0.000	-0.185
Location_29 -0.052	-0.0769	0.013	-6.150	0.000	-0.101
Location_30 2.21e-17	1.549e-17	3.36e-18	4.613	0.000	8.91e-18
Location_32 0.008	-0.0156	0.012	-1.283	0.200	-0.039
Location_33 -0.008	-0.0324	0.012	-2.607	0.009	-0.057
Location_34 -0.075	-0.1001	0.013	-7.749	0.000	-0.125
Location_35 6.56e-19	5.317e-19	6.34e-20	8.391	0.000	4.08e-19
Location_36 -0.155	-0.1817	0.014	-13.134	0.000	-0.209
Location_38 -0.090	-0.1155	0.013	-8.813	0.000	-0.141
Location_39 -0.076	-0.1011	0.013	-8.004	0.000	-0.126
Location_40 -0.067	-0.0935	0.014	-6.918	0.000	-0.120
Location_41 0	0	0	nan	nan	0

Location_42	0	0	nan	nan	0
0					
Location_43	-0.0593	0.013	-4.674	0.000	-0.084
-0.034					
Location_44	0	0	nan	nan	0
0					
Location_45	-0.1300	0.012	-10.470	0.000	-0.154
-0.106					
Location_46	-0.0701	0.015	-4.763	0.000	-0.099
-0.041					
Location_47	0	0	nan	nan	0
0					
Location_48	0	0	nan	nan	0
0					
Location_49	-0.0605	0.013	-4.514	0.000	-0.087
-0.034					
Season_Spring	0.0555	0.004	13.589	0.000	0.048
0.064					
Season_Summer	0.0436	0.005	9.606	0.000	0.035
0.052					
Season_Winter	-0.0164	0.005	-3.591	0.000	-0.025
-0.007					

```
=====
Omnibus:                    5445.436    Durbin-Watson:                1.804
Prob(Omnibus):              0.000    Jarque-Bera (JB):             6873.761
Skew:                      0.789    Prob(JB):                     0.00
Kurtosis:                  2.827    Cond. No.                     1.03e+19
=====
```

Notes:

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

[2] The smallest eigenvalue is 6.4e-28. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

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.

```
[11]: # Crear el DataFrame modelo a partir del DataFrame limpio actual y eliminar
      ↪ columnas conflictivas
df_model2 = df.drop(columns=['Electricity', 'Evaporation'])

# y: variable dependiente (lo que queremos predecir)
y = df_model2['Failure_today']

# X: todas las demás variables (excepto la variable objetivo)
X = df_model2.drop(columns=['Failure_today'])
```

```

# Agregar constante (intercepto) al modelo
X = sm.add_constant(X)

# Unir y eliminar posibles filas con NaN por seguridad
Xy = pd.concat([y, X], axis=1).dropna()
y_clean2 = Xy['Failure_today']
X_clean2 = Xy.drop(columns=['Failure_today'])

# Ejecutar modelo Probit
modelo_probit = sm.Probit(y_clean2, X_clean2)
probit_result = modelo_probit.fit(cov_type="HCO")

# Mostrar resumen del modelo
print(probit_result.summary())

# Mostrar efectos marginales
mfx_probit = probit_result.get_margeff()
print(mfx_probit.summary())

```

Optimization terminated successfully.

Current function value: 0.356680

Iterations 7

Probit Regression Results

```

=====
Dep. Variable:          Failure_today    No. Observations:          119590
Model:                  Probit           Df Residuals:              119526
Method:                 MLE             Df Model:                  63
Date:                  Thu, 24 Apr 2025   Pseudo R-squ.:              0.3248
Time:                  23:02:42          Log-Likelihood:             -42655.
converged:              True             LL-Null:                   -63172.
Covariance Type:       HCO              LLR p-value:                0.000
=====

```

```

=====

```

	coef	std err	z	P> z	[0.025
0.975]					

const	21.3376	0.953	22.381	0.000	19.469
23.206					
Min_Temp	0.1097	0.002	53.764	0.000	0.106
0.114					
Max_Temp	-0.1299	0.002	-55.838	0.000	-0.134
-0.125					
Parameter1_Speed	0.0211	0.001	35.320	0.000	0.020
0.022					
Parameter3_9am	0.0119	0.001	14.330	0.000	0.010
0.014					
Parameter3_3pm	-0.0135	0.001	-15.865	0.000	-0.015

-0.012					
Parameter4_9am	0.0419	0.001	82.859	0.000	0.041
0.043					
Parameter4_3pm	-0.0027	0.000	-5.963	0.000	-0.004
-0.002					
Parameter5_3pm	-0.0239	0.001	-26.095	0.000	-0.026
-0.022					
Parameter1_Dir_region_N	-0.0444	0.020	-2.264	0.024	-0.083
-0.006					
Parameter1_Dir_region_S	0.0307	0.018	1.735	0.083	-0.004
0.065					
Parameter1_Dir_region_W	0.0887	0.019	4.554	0.000	0.051
0.127					
Parameter2_9am_region_N	-0.0085	0.017	-0.494	0.621	-0.042
0.025					
Parameter2_9am_region_S	0.1290	0.017	7.565	0.000	0.096
0.162					
Parameter2_9am_region_W	0.2710	0.018	15.124	0.000	0.236
0.306					
Parameter2_3pm_region_N	-0.0139	0.019	-0.721	0.471	-0.052
0.024					
Parameter2_3pm_region_S	0.1310	0.017	7.552	0.000	0.097
0.165					
Parameter2_3pm_region_W	0.1588	0.020	8.007	0.000	0.120
0.198					
Location_3	-0.1956	0.046	-4.253	0.000	-0.286
-0.105					
Location_4	0.2484	0.061	4.044	0.000	0.128
0.369					
Location_5	-0.2381	0.046	-5.210	0.000	-0.328
-0.149					
Location_6	-0.9238	0.047	-19.488	0.000	-1.017
-0.831					
Location_7	-0.3808	0.046	-8.267	0.000	-0.471
-0.291					
Location_8	0.3170	0.044	7.258	0.000	0.231
0.403					
Location_9	0.0628	0.045	1.400	0.162	-0.025
0.151					
Location_10	-0.1535	0.046	-3.351	0.001	-0.243
-0.064					
Location_11	-0.1270	0.053	-2.416	0.016	-0.230
-0.024					
Location_12	0.0671	0.045	1.492	0.136	-0.021
0.155					
Location_13	-0.5069	0.044	-11.522	0.000	-0.593
-0.421					
Location_14	-0.0776	0.047	-1.668	0.095	-0.169

0.014					
Location_15	-0.0631	0.045	-1.399	0.162	-0.151
0.025					
Location_16	-0.3587	0.045	-8.054	0.000	-0.446
-0.271					
Location_17	0.0277	0.079	0.350	0.726	-0.127
0.183					
Location_18	-0.3442	0.049	-6.981	0.000	-0.441
-0.248					
Location_19	-0.2927	0.047	-6.281	0.000	-0.384
-0.201					
Location_20	-0.5514	0.045	-12.193	0.000	-0.640
-0.463					
Location_21	-0.5649	0.051	-11.114	0.000	-0.664
-0.465					
Location_22	-0.0144	0.051	-0.282	0.778	-0.114
0.086					
Location_23	-0.3140	0.044	-7.204	0.000	-0.399
-0.229					
Location_26	-0.7974	0.058	-13.641	0.000	-0.912
-0.683					
Location_27	-0.5230	0.044	-11.838	0.000	-0.610
-0.436					
Location_28	-0.5060	0.043	-11.833	0.000	-0.590
-0.422					
Location_29	-0.4938	0.049	-10.054	0.000	-0.590
-0.398					
Location_30	0.1161	0.049	2.364	0.018	0.020
0.212					
Location_32	0.0874	0.043	2.025	0.043	0.003
0.172					
Location_33	0.0961	0.046	2.108	0.035	0.007
0.185					
Location_34	-0.4255	0.043	-9.915	0.000	-0.510
-0.341					
Location_35	-0.2180	0.045	-4.820	0.000	-0.307
-0.129					
Location_36	-0.6216	0.046	-13.486	0.000	-0.712
-0.531					
Location_38	-0.2526	0.046	-5.522	0.000	-0.342
-0.163					
Location_39	-0.2139	0.045	-4.707	0.000	-0.303
-0.125					
Location_40	-0.1297	0.047	-2.764	0.006	-0.222
-0.038					
Location_41	-0.0849	0.045	-1.871	0.061	-0.174
0.004					
Location_42	0.2088	0.077	2.698	0.007	0.057

0.361					
Location_43	-0.1769	0.049	-3.632	0.000	-0.272
-0.081					
Location_44	-0.2959	0.043	-6.860	0.000	-0.380
-0.211					
Location_45	-0.5283	0.044	-12.006	0.000	-0.615
-0.442					
Location_46	-0.0289	0.047	-0.611	0.541	-0.121
0.064					
Location_47	-0.0459	0.045	-1.028	0.304	-0.134
0.042					
Location_48	-0.6068	0.045	-13.571	0.000	-0.694
-0.519					
Location_49	-0.7232	0.060	-12.119	0.000	-0.840
-0.606					
Season_Spring	0.2007	0.015	13.817	0.000	0.172
0.229					
Season_Summer	0.1530	0.016	9.545	0.000	0.122
0.184					
Season_Winter	-0.1242	0.016	-7.897	0.000	-0.155
-0.093					

=====

=====

Probit Marginal Effects

=====

Dep. Variable:	Failure_today
Method:	dydx
At:	overall

=====

=====

	dy/dx	std err	z	P> z	[0.025
0.975]					

Min_Temp	0.0220	0.000	55.585	0.000	0.021
0.023					
Max_Temp	-0.0260	0.000	-59.057	0.000	-0.027
-0.025					
Parameter1_Speed	0.0042	0.000	36.045	0.000	0.004
0.004					
Parameter3_9am	0.0024	0.000	14.378	0.000	0.002
0.003					
Parameter3_3pm	-0.0027	0.000	-15.925	0.000	-0.003
-0.002					
Parameter4_9am	0.0084	8.78e-05	95.524	0.000	0.008
0.009					
Parameter4_3pm	-0.0005	9.14e-05	-5.971	0.000	-0.001
-0.000					

Parameter5_3pm -0.004	-0.0048	0.000	-26.346	0.000	-0.005
Parameter1_Dir_region_N -0.001	-0.0089	0.004	-2.265	0.024	-0.017
Parameter1_Dir_region_S 0.013	0.0062	0.004	1.735	0.083	-0.001
Parameter1_Dir_region_W 0.025	0.0178	0.004	4.552	0.000	0.010
Parameter2_9am_region_N 0.005	-0.0017	0.003	-0.494	0.621	-0.008
Parameter2_9am_region_S 0.033	0.0258	0.003	7.568	0.000	0.019
Parameter2_9am_region_W 0.061	0.0543	0.004	15.162	0.000	0.047
Parameter2_3pm_region_N 0.005	-0.0028	0.004	-0.721	0.471	-0.010
Parameter2_3pm_region_S 0.033	0.0262	0.003	7.558	0.000	0.019
Parameter2_3pm_region_W 0.040	0.0318	0.004	8.012	0.000	0.024
Location_3 -0.021	-0.0392	0.009	-4.257	0.000	-0.057
Location_4 0.074	0.0497	0.012	4.044	0.000	0.026
Location_5 -0.030	-0.0477	0.009	-5.213	0.000	-0.066
Location_6 -0.167	-0.1849	0.009	-19.697	0.000	-0.203
Location_7 -0.058	-0.0762	0.009	-8.284	0.000	-0.094
Location_8 0.081	0.0635	0.009	7.265	0.000	0.046
Location_9 0.030	0.0126	0.009	1.400	0.162	-0.005
Location_10 -0.013	-0.0307	0.009	-3.353	0.001	-0.049
Location_11 -0.005	-0.0254	0.011	-2.418	0.016	-0.046
Location_12 0.031	0.0134	0.009	1.492	0.136	-0.004
Location_13 -0.084	-0.1015	0.009	-11.561	0.000	-0.119
Location_14 0.003	-0.0155	0.009	-1.668	0.095	-0.034
Location_15 0.005	-0.0126	0.009	-1.399	0.162	-0.030
Location_16 -0.054	-0.0718	0.009	-8.074	0.000	-0.089

Location_17 0.037	0.0055	0.016	0.350	0.726	-0.025
Location_18 -0.050	-0.0689	0.010	-6.991	0.000	-0.088
Location_19 -0.040	-0.0586	0.009	-6.289	0.000	-0.077
Location_20 -0.093	-0.1104	0.009	-12.239	0.000	-0.128
Location_21 -0.093	-0.1131	0.010	-11.147	0.000	-0.133
Location_22 0.017	-0.0029	0.010	-0.282	0.778	-0.023
Location_23 -0.046	-0.0629	0.009	-7.216	0.000	-0.080
Location_26 -0.137	-0.1596	0.012	-13.697	0.000	-0.182
Location_27 -0.087	-0.1047	0.009	-11.869	0.000	-0.122
Location_28 -0.085	-0.1013	0.009	-11.864	0.000	-0.118
Location_29 -0.080	-0.0989	0.010	-10.091	0.000	-0.118
Location_30 0.042	0.0232	0.010	2.364	0.018	0.004
Location_32 0.034	0.0175	0.009	2.025	0.043	0.001
Location_33 0.037	0.0192	0.009	2.109	0.035	0.001
Location_34 -0.068	-0.0852	0.009	-9.941	0.000	-0.102
Location_35 -0.026	-0.0436	0.009	-4.823	0.000	-0.061
Location_36 -0.106	-0.1244	0.009	-13.555	0.000	-0.142
Location_38 -0.033	-0.0506	0.009	-5.526	0.000	-0.069
Location_39 -0.025	-0.0428	0.009	-4.709	0.000	-0.061
Location_40 -0.008	-0.0260	0.009	-2.763	0.006	-0.044
Location_41 0.001	-0.0170	0.009	-1.872	0.061	-0.035
Location_42 0.072	0.0418	0.015	2.698	0.007	0.011
Location_43 -0.016	-0.0354	0.010	-3.635	0.000	-0.054
Location_44 -0.042	-0.0592	0.009	-6.868	0.000	-0.076

Location_45	-0.1058	0.009	-12.054	0.000	-0.123
-0.089					
Location_46	-0.0058	0.009	-0.611	0.541	-0.024
0.013					
Location_47	-0.0092	0.009	-1.028	0.304	-0.027
0.008					
Location_48	-0.1215	0.009	-13.616	0.000	-0.139
-0.104					
Location_49	-0.1448	0.012	-12.178	0.000	-0.168
-0.121					
Season_Spring	0.0402	0.003	13.811	0.000	0.034
0.046					
Season_Summer	0.0306	0.003	9.548	0.000	0.024
0.037					
Season_Winter	-0.0249	0.003	-7.912	0.000	-0.031
-0.019					

=====

=====

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.

```
[12]: # Crear el DataFrame modelo a partir del DataFrame limpio actual
df_model3 = df.drop(columns=['Electricity', 'Evaporation'])

# y: variable dependiente (lo que queremos predecir)
y = df_model3['Failure_today']

# X: todas las demás variables (excepto la variable objetivo)
X = df_model3.drop(columns=['Failure_today'])

# Agregar constante (intercepto) al modelo
X = sm.add_constant(X)

# Unir y eliminar posibles filas con NaN por seguridad
Xy = pd.concat([y, X], axis=1).dropna()
y_clean3 = Xy['Failure_today']
X_clean3 = Xy.drop(columns=['Failure_today'])

# Ejecutar modelo Logit
modelo_logit = sm.Logit(y_clean3, X_clean3)
logit_result = modelo_logit.fit(cov_type="HCO")

# Mostrar resumen del modelo
print(logit_result.summary())

# Mostrar efectos marginales
mfx_logit = logit_result.get_margeff()
```

```
print(mfx_logit.summary())
```

Optimization terminated successfully.

Current function value: 0.355396

Iterations 8

Logit Regression Results

```
=====
Dep. Variable:          Failure_today    No. Observations:          119590
Model:                  Logit            Df Residuals:            119526
Method:                 MLE              Df Model:                 63
Date:                   Thu, 24 Apr 2025  Pseudo R-squ.:            0.3272
Time:                   23:02:48          Log-Likelihood:           -42502.
converged:              True              LL-Null:                -63172.
Covariance Type:        HCO              LLR p-value:             0.000
=====
```

```
=====
                                coef    std err          z      P>|z|      [0.025
0.975]
-----
-----
const                36.7051      1.675     21.911     0.000     33.422
39.988
Min_Temp              0.1967      0.004     54.530     0.000      0.190
0.204
Max_Temp             -0.2396      0.004    -58.012     0.000     -0.248
-0.232
Parameter1_Speed      0.0371      0.001     35.068     0.000      0.035
0.039
Parameter3_9am         0.0203      0.001     13.766     0.000      0.017
0.023
Parameter3_3pm        -0.0227      0.002    -15.034     0.000     -0.026
-0.020
Parameter4_9am         0.0761      0.001     85.370     0.000      0.074
0.078
Parameter4_3pm        -0.0058      0.001     -7.230     0.000     -0.007
-0.004
Parameter5_3pm        -0.0412      0.002    -25.533     0.000     -0.044
-0.038
Parameter1_Dir_region_N -0.0955      0.035     -2.752     0.006     -0.163
-0.027
Parameter1_Dir_region_S  0.0347      0.031      1.107     0.268     -0.027
0.096
Parameter1_Dir_region_W  0.1410      0.034      4.091     0.000      0.073
0.209
Parameter2_9am_region_N -0.0212      0.030     -0.699     0.484     -0.081
0.038
Parameter2_9am_region_S  0.2253      0.030      7.420     0.000      0.166
0.285
=====
```

Parameter2_9am_region_W 0.539	0.4770	0.032	15.055	0.000	0.415
Parameter2_3pm_region_N 0.034	-0.0327	0.034	-0.958	0.338	-0.100
Parameter2_3pm_region_S 0.279	0.2192	0.031	7.136	0.000	0.159
Parameter2_3pm_region_W 0.331	0.2618	0.035	7.469	0.000	0.193
Location_3 -0.251	-0.4099	0.081	-5.040	0.000	-0.569
Location_4 0.582	0.3676	0.109	3.362	0.001	0.153
Location_5 -0.247	-0.4069	0.082	-4.973	0.000	-0.567
Location_6 -1.565	-1.7276	0.083	-20.781	0.000	-1.891
Location_7 -0.560	-0.7195	0.081	-8.829	0.000	-0.879
Location_8 0.782	0.6299	0.078	8.102	0.000	0.478
Location_9 0.374	0.2190	0.079	2.761	0.006	0.064
Location_10 -0.140	-0.3012	0.082	-3.669	0.000	-0.462
Location_11 -0.130	-0.3160	0.095	-3.323	0.001	-0.502
Location_12 0.318	0.1613	0.080	2.020	0.043	0.005
Location_13 -0.775	-0.9266	0.077	-11.967	0.000	-1.078
Location_14 0.143	-0.0201	0.083	-0.242	0.809	-0.183
Location_15 0.117	-0.0402	0.080	-0.500	0.617	-0.198
Location_16 -0.525	-0.6818	0.080	-8.541	0.000	-0.838
Location_17 0.463	0.1855	0.142	1.311	0.190	-0.092
Location_18 -0.452	-0.6223	0.087	-7.157	0.000	-0.793
Location_19 -0.368	-0.5307	0.083	-6.398	0.000	-0.693
Location_20 -0.844	-1.0022	0.081	-12.424	0.000	-1.160
Location_21 -0.872	-1.0492	0.090	-11.604	0.000	-1.226
Location_22 0.150	-0.0334	0.094	-0.357	0.721	-0.217

Location_23 -0.430	-0.5814	0.077	-7.545	0.000	-0.732
Location_26 -1.242	-1.4462	0.104	-13.904	0.000	-1.650
Location_27 -0.765	-0.9191	0.079	-11.659	0.000	-1.074
Location_28 -0.729	-0.8777	0.076	-11.567	0.000	-1.026
Location_29 -0.787	-0.9575	0.087	-11.000	0.000	-1.128
Location_30 0.385	0.2137	0.087	2.450	0.014	0.043
Location_32 0.343	0.1934	0.076	2.533	0.011	0.044
Location_33 0.355	0.1964	0.081	2.421	0.015	0.037
Location_34 -0.625	-0.7737	0.076	-10.174	0.000	-0.923
Location_35 -0.208	-0.3658	0.081	-4.533	0.000	-0.524
Location_36 -0.980	-1.1410	0.082	-13.917	0.000	-1.302
Location_38 -0.265	-0.4240	0.081	-5.224	0.000	-0.583
Location_39 -0.217	-0.3786	0.082	-4.593	0.000	-0.540
Location_40 0.058	-0.1056	0.084	-1.264	0.206	-0.269
Location_41 -0.017	-0.1758	0.081	-2.171	0.030	-0.335
Location_42 0.591	0.3162	0.140	2.255	0.024	0.041
Location_43 -0.227	-0.3984	0.088	-4.544	0.000	-0.570
Location_44 -0.377	-0.5270	0.077	-6.877	0.000	-0.677
Location_45 -0.824	-0.9763	0.078	-12.542	0.000	-1.129
Location_46 0.121	-0.0443	0.084	-0.526	0.599	-0.209
Location_47 0.070	-0.0836	0.079	-1.064	0.287	-0.238
Location_48 -0.913	-1.0698	0.080	-13.343	0.000	-1.227
Location_49 -1.172	-1.3775	0.105	-13.157	0.000	-1.583
Season_Spring 0.372	0.3208	0.026	12.350	0.000	0.270

Season_Summer	0.2541	0.029	8.911	0.000	0.198
0.310					
Season_Winter	-0.2418	0.028	-8.782	0.000	-0.296
-0.188					

=====

=====

Logit Marginal Effects

Dep. Variable: Failure_today

Method: dydx

At: overall

=====

=====

	dy/dx	std err	z	P> z	[0.025
0.975]					

Min_Temp	0.0221	0.000	56.500	0.000	0.021
0.023					
Max_Temp	-0.0269	0.000	-61.152	0.000	-0.028
-0.026					
Parameter1_Speed	0.0042	0.000	35.960	0.000	0.004
0.004					
Parameter3_9am	0.0023	0.000	13.811	0.000	0.002
0.003					
Parameter3_3pm	-0.0026	0.000	-15.096	0.000	-0.003
-0.002					
Parameter4_9am	0.0086	8.6e-05	99.523	0.000	0.008
0.009					
Parameter4_3pm	-0.0007	9.01e-05	-7.238	0.000	-0.001
-0.000					
Parameter5_3pm	-0.0046	0.000	-25.779	0.000	-0.005
-0.004					
Parameter1_Dir_region_N	-0.0107	0.004	-2.753	0.006	-0.018
-0.003					
Parameter1_Dir_region_S	0.0039	0.004	1.107	0.268	-0.003
0.011					
Parameter1_Dir_region_W	0.0158	0.004	4.090	0.000	0.008
0.023					
Parameter2_9am_region_N	-0.0024	0.003	-0.700	0.484	-0.009
0.004					
Parameter2_9am_region_S	0.0253	0.003	7.420	0.000	0.019
0.032					
Parameter2_9am_region_W	0.0536	0.004	15.093	0.000	0.047
0.061					
Parameter2_3pm_region_N	-0.0037	0.004	-0.958	0.338	-0.011
0.004					
Parameter2_3pm_region_S	0.0246	0.003	7.142	0.000	0.018

0.031					
Parameter2_3pm_region_W	0.0294	0.004	7.475	0.000	0.022
0.037					
Location_3	-0.0461	0.009	-5.046	0.000	-0.064
-0.028					
Location_4	0.0413	0.012	3.362	0.001	0.017
0.065					
Location_5	-0.0457	0.009	-4.977	0.000	-0.064
-0.028					
Location_6	-0.1941	0.009	-21.013	0.000	-0.212
-0.176					
Location_7	-0.0809	0.009	-8.849	0.000	-0.099
-0.063					
Location_8	0.0708	0.009	8.109	0.000	0.054
0.088					
Location_9	0.0246	0.009	2.762	0.006	0.007
0.042					
Location_10	-0.0338	0.009	-3.672	0.000	-0.052
-0.016					
Location_11	-0.0355	0.011	-3.325	0.001	-0.056
-0.015					
Location_12	0.0181	0.009	2.020	0.043	0.001
0.036					
Location_13	-0.1041	0.009	-12.008	0.000	-0.121
-0.087					
Location_14	-0.0023	0.009	-0.242	0.809	-0.021
0.016					
Location_15	-0.0045	0.009	-0.500	0.617	-0.022
0.013					
Location_16	-0.0766	0.009	-8.568	0.000	-0.094
-0.059					
Location_17	0.0208	0.016	1.311	0.190	-0.010
0.052					
Location_18	-0.0699	0.010	-7.168	0.000	-0.089
-0.051					
Location_19	-0.0596	0.009	-6.408	0.000	-0.078
-0.041					
Location_20	-0.1126	0.009	-12.476	0.000	-0.130
-0.095					
Location_21	-0.1179	0.010	-11.641	0.000	-0.138
-0.098					
Location_22	-0.0038	0.011	-0.357	0.721	-0.024
0.017					
Location_23	-0.0653	0.009	-7.557	0.000	-0.082
-0.048					
Location_26	-0.1625	0.012	-13.963	0.000	-0.185
-0.140					
Location_27	-0.1033	0.009	-11.695	0.000	-0.121

-0.086					
Location_28	-0.0986	0.009	-11.602	0.000	-0.115
-0.082					
Location_29	-0.1076	0.010	-11.036	0.000	-0.127
-0.088					
Location_30	0.0240	0.010	2.450	0.014	0.005
0.043					
Location_32	0.0217	0.009	2.533	0.011	0.005
0.039					
Location_33	0.0221	0.009	2.422	0.015	0.004
0.040					
Location_34	-0.0869	0.009	-10.200	0.000	-0.104
-0.070					
Location_35	-0.0411	0.009	-4.536	0.000	-0.059
-0.023					
Location_36	-0.1282	0.009	-13.996	0.000	-0.146
-0.110					
Location_38	-0.0476	0.009	-5.229	0.000	-0.066
-0.030					
Location_39	-0.0425	0.009	-4.596	0.000	-0.061
-0.024					
Location_40	-0.0119	0.009	-1.264	0.206	-0.030
0.007					
Location_41	-0.0198	0.009	-2.171	0.030	-0.038
-0.002					
Location_42	0.0355	0.016	2.255	0.024	0.005
0.066					
Location_43	-0.0448	0.010	-4.549	0.000	-0.064
-0.025					
Location_44	-0.0592	0.009	-6.887	0.000	-0.076
-0.042					
Location_45	-0.1097	0.009	-12.599	0.000	-0.127
-0.093					
Location_46	-0.0050	0.009	-0.526	0.599	-0.024
0.014					
Location_47	-0.0094	0.009	-1.064	0.287	-0.027
0.008					
Location_48	-0.1202	0.009	-13.397	0.000	-0.138
-0.103					
Location_49	-0.1548	0.012	-13.208	0.000	-0.178
-0.132					
Season_Spring	0.0360	0.003	12.346	0.000	0.030
0.042					
Season_Summer	0.0286	0.003	8.907	0.000	0.022
0.035					
Season_Winter	-0.0272	0.003	-8.794	0.000	-0.033
-0.021					

=====

=====

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.

```
[13]: # Paso 1: Asegurar formato de fecha y extraer año y mes
dfcopy1['Year'] = dfcopy1['Date'].dt.year
dfcopy1['Month'] = dfcopy1['Date'].dt.month

# Paso 2: Reconstruir variable Location desde las dummies
location_cols = [col for col in dfcopy1.columns if col.startswith('Location_')]

if location_cols:
    dfcopy1['Location'] = dfcopy1[location_cols].idxmax(axis=1).str.
    ↪replace('Location_', '').astype(int)
    dfcopy1.drop(columns=location_cols, inplace=True)

# Paso 3: Eliminar columnas de dirección y otras no deseadas
cols_a_eliminar = [
    'Parameter1_Dir_region_W',
    'Parameter2_9am_region_N', 'Parameter2_9am_region_S',
    ↪'Parameter2_9am_region_W',
    'Parameter2_3pm_region_N', 'Parameter2_3pm_region_S',
    ↪'Parameter2_3pm_region_W',
    'Evaporation', 'Electricity',
    'Parameter1_Dir_region_N', 'Parameter1_Dir_region_S'
]
dfcopy1.drop(columns=[col for col in cols_a_eliminar if col in dfcopy1.
    ↪columns], inplace=True)

# Paso 4: Agrupar por Year, Month y Location
cols_promedio = ['Min_Temp', 'Max_Temp', 'Parameter1_Speed', 'Parameter3_9am',
    'Parameter3_3pm', 'Parameter4_9am', 'Parameter4_3pm',
    ↪'Parameter5_3pm']

df_mensual = dfcopy1.groupby(['Year', 'Month', 'Location']).agg({
    **{col: 'mean' for col in cols_promedio},
    'Failure_today': 'sum'
}).reset_index()

# Paso 5: Crear columna 'Mes' en formato 'YYYY-MM'
df_mensual['Mes'] = df_mensual['Year'].astype(str) + '-' + df_mensual['Month'].
    ↪astype(str).str.zfill(2)
```

```
# Paso 6: Ajustar modelo Poisson con fórmula (C(Location) como variable
↳categorica)
poisson_model = smf.poisson(
    formula="Failure_today ~ C(Location) + Min_Temp + Max_Temp +
↳Parameter1_Speed + Parameter3_9am + Parameter3_3pm + Parameter4_9am +
↳Parameter4_3pm + Parameter5_3pm",
    data=df_mensual
).fit()

# Paso 7: Mostrar resumen
print(poisson_model.summary())
```

Optimization terminated successfully.

Current function value: 2.243287

Iterations 8

Poisson Regression Results

```
=====
Dep. Variable:      Failure_today    No. Observations:      4037
Model:              Poisson          Df Residuals:           3986
Method:             MLE              Df Model:              50
Date:               Thu, 24 Apr 2025  Pseudo R-squ.:          0.3332
Time:               23:02:53          Log-Likelihood:         -9056.2
converged:          True              LL-Null:               -13582.
Covariance Type:    nonrobust         LLR p-value:           0.000
=====
```

```
=====
coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
Intercept      23.9085      2.867      8.340      0.000      18.290
29.527
C(Location) [T.4] -0.6129      0.075     -8.225      0.000     -0.759
-0.467
C(Location) [T.5] -0.8916      0.055    -16.195      0.000     -0.999
-0.784
C(Location) [T.6] -1.0855      0.061    -17.810      0.000     -1.205
-0.966
C(Location) [T.7] -0.8222      0.054    -15.287      0.000     -0.928
-0.717
C(Location) [T.8] -0.7212      0.053    -13.507      0.000     -0.826
-0.617
C(Location) [T.9] -0.7526      0.056    -13.477      0.000     -0.862
-0.643
C(Location) [T.10] -0.7323      0.057    -12.929      0.000     -0.843
-0.621
C(Location) [T.11] -0.7356      0.063    -11.685      0.000     -0.859
```

-0.612					
C(Location) [T.12]	-0.6866	0.055	-12.569	0.000	-0.794
-0.580					
C(Location) [T.13]	-0.9660	0.051	-18.779	0.000	-1.067
-0.865					
C(Location) [T.14]	-1.0670	0.055	-19.263	0.000	-1.176
-0.958					
C(Location) [T.15]	-0.7973	0.063	-12.690	0.000	-0.920
-0.674					
C(Location) [T.16]	-1.3110	0.051	-25.622	0.000	-1.411
-1.211					
C(Location) [T.17]	-1.3076	0.104	-12.626	0.000	-1.511
-1.105					
C(Location) [T.18]	-1.0274	0.058	-17.606	0.000	-1.142
-0.913					
C(Location) [T.19]	-1.0381	0.057	-18.255	0.000	-1.150
-0.927					
C(Location) [T.20]	-0.9559	0.057	-16.714	0.000	-1.068
-0.844					
C(Location) [T.21]	-0.8259	0.066	-12.530	0.000	-0.955
-0.697					
C(Location) [T.22]	-0.7530	0.068	-11.135	0.000	-0.886
-0.620					
C(Location) [T.23]	-0.7038	0.052	-13.588	0.000	-0.805
-0.602					
C(Location) [T.26]	-0.9686	0.076	-12.725	0.000	-1.118
-0.819					
C(Location) [T.27]	-1.3323	0.054	-24.862	0.000	-1.437
-1.227					
C(Location) [T.28]	-1.2867	0.061	-21.155	0.000	-1.406
-1.168					
C(Location) [T.29]	-0.8090	0.053	-15.324	0.000	-0.912
-0.705					
C(Location) [T.30]	-0.7335	0.058	-12.599	0.000	-0.848
-0.619					
C(Location) [T.32]	-0.5966	0.049	-12.104	0.000	-0.693
-0.500					
C(Location) [T.33]	-0.5252	0.055	-9.541	0.000	-0.633
-0.417					
C(Location) [T.34]	-0.9500	0.049	-19.474	0.000	-1.046
-0.854					
C(Location) [T.35]	-0.8952	0.055	-16.327	0.000	-1.003
-0.788					
C(Location) [T.36]	-0.9228	0.057	-16.312	0.000	-1.034
-0.812					
C(Location) [T.38]	-0.9837	0.055	-17.908	0.000	-1.091
-0.876					
C(Location) [T.39]	-0.8256	0.056	-14.636	0.000	-0.936

-0.715					
C(Location) [T.40]	-1.0694	0.064	-16.660	0.000	-1.195
-0.944					
C(Location) [T.41]	-0.7046	0.054	-13.070	0.000	-0.810
-0.599					
C(Location) [T.42]	-0.7962	0.102	-7.797	0.000	-0.996
-0.596					
C(Location) [T.43]	-0.6245	0.055	-11.453	0.000	-0.731
-0.518					
C(Location) [T.44]	-1.1755	0.049	-24.099	0.000	-1.271
-1.080					
C(Location) [T.45]	-1.0998	0.049	-22.663	0.000	-1.195
-1.005					
C(Location) [T.46]	-0.7620	0.056	-13.548	0.000	-0.872
-0.652					
C(Location) [T.47]	-0.7834	0.050	-15.758	0.000	-0.881
-0.686					
C(Location) [T.48]	-1.5235	0.058	-26.428	0.000	-1.636
-1.411					
C(Location) [T.49]	-1.1568	0.082	-14.104	0.000	-1.318
-0.996					
Min_Temp	0.1026	0.008	13.039	0.000	0.087
0.118					
Max_Temp	-0.1011	0.008	-12.959	0.000	-0.116
-0.086					
Parameter1_Speed	0.0562	0.003	21.125	0.000	0.051
0.061					
Parameter3_9am	-0.0096	0.004	-2.524	0.012	-0.017
-0.002					
Parameter3_3pm	-0.0578	0.004	-16.211	0.000	-0.065
-0.051					
Parameter4_9am	0.0166	0.002	9.896	0.000	0.013
0.020					
Parameter4_3pm	0.0169	0.002	8.509	0.000	0.013
0.021					
Parameter5_3pm	-0.0229	0.003	-8.305	0.000	-0.028
-0.017					

=====

=====

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

```
[14]: # Paso 1: Obtener el lambda estimado del modelo Poisson
df_mensual['plambda'] = poisson_model.predict(df_mensual)

# Paso 2: Graficar la distribución de los valores esperados ( )
plt.figure(figsize=(8, 5))
```

```

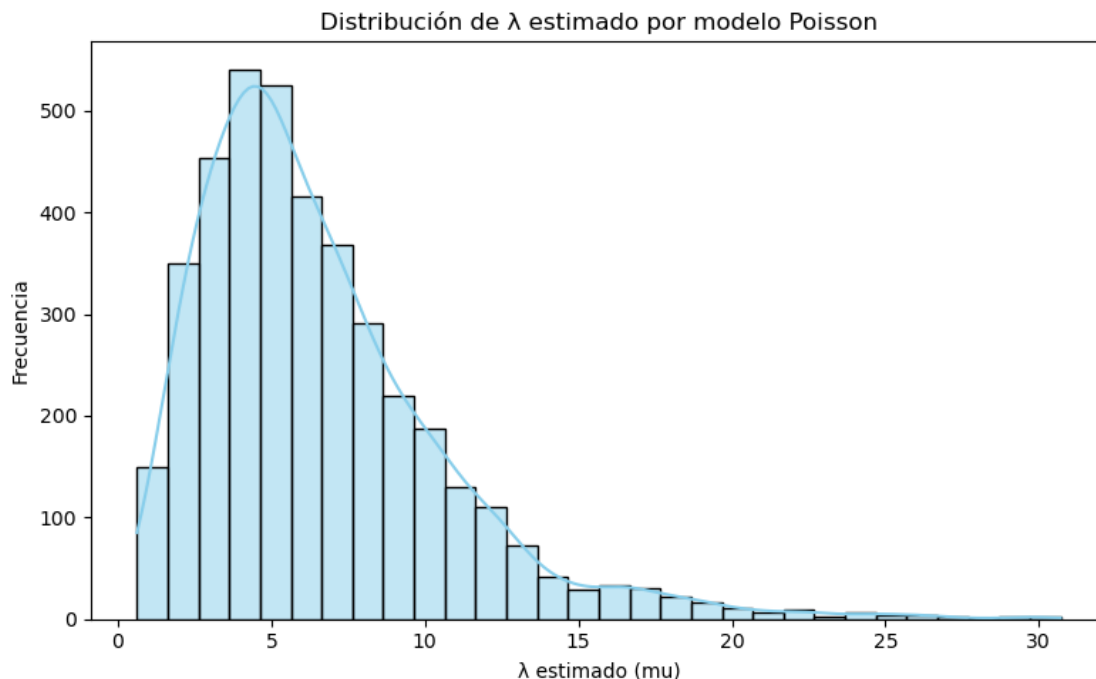
sns.histplot(data=df_mensual, x="plambda", bins=30, kde=True, color='skyblue')
plt.title("Distribución de estimado por modelo Poisson")
plt.xlabel(" estimado (mu)")
plt.ylabel("Frecuencia")
plt.tight_layout()
plt.show()

# Paso 3: Calcular variable auxiliar para test de sobre-dispersión
aux = ((df_mensual['Failure_today'] - df_mensual['plambda'])*2 -
      df_mensual['plambda']) / df_mensual['plambda']

# Paso 4: Regresión auxiliar (estima alpha implícitamente)
auxr = sm.OLS(aux, df_mensual['plambda']).fit()

# Paso 5: Resultado de la regresión auxiliar
print(auxr.summary())

```



OLS Regression Results

```

=====
=====
Dep. Variable:              y    R-squared (uncentered):
0.001
Model:                    OLS    Adj. R-squared (uncentered):
0.000
Method:                  Least Squares    F-statistic:

```



```

2.598
Date: Thu, 24 Apr 2025 Prob (F-statistic):
0.107
Time: 23:02:53 Log-Likelihood:
-7134.4
No. Observations: 4037 AIC:
1.427e+04
Df Residuals: 4036 BIC:
1.428e+04
Df Model: 1
Covariance Type: nonrobust
=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
plambda      -0.0047      0.003      -1.612      0.107      -0.010      0.001
=====
Omnibus: 3474.559 Durbin-Watson: 1.829
Prob(Omnibus): 0.000 Jarque-Bera (JB): 147695.752
Skew: 3.919 Prob(JB): 0.00
Kurtosis: 31.576 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.

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.

```

[15]: # Paso 1: Estimar modelo Binomial Negativa (con alpha ajustado automáticamente)
neg_bin_model = smf.glm(
    formula="Failure_today ~ C(Location) + Min_Temp + Max_Temp + \
↳Parameter1_Speed + \
    Parameter3_9am + Parameter3_3pm + Parameter4_9am + Parameter4_3pm \
↳+ Parameter5_3pm",
    data=df_mensual,
    family=sm.families.NegativeBinomial()
).fit()

# Paso 2: Mostrar resumen
print(neg_bin_model.summary())

# Paso 3: Generar predicciones y graficar
df_mensual['ypred'] = neg_bin_model.predict(df_mensual)

```

```
# Gráfico de valores predichos vs observados
sns.lmplot(data=df_mensual, x='Failure_today', y='ypred', height=5, aspect=1.5,
↳scatter_kws={'alpha':0.5})
plt.title("Predicción vs Observado - Modelo Binomial Negativa")
plt.xlabel("Fallas Observadas")
plt.ylabel("Fallas Predichas")
plt.tight_layout()
plt.show()
```

Generalized Linear Model Regression Results

```
=====
Dep. Variable:          Failure_today    No. Observations:          4037
Model:                  GLM              Df Residuals:            3986
Model Family:          NegativeBinomial  Df Model:                  50
Link Function:          Log              Scale:                    1.0000
Method:                 IRLS             Log-Likelihood:           -11238.
Date:                   Thu, 24 Apr 2025 Deviance:                  1058.5
Time:                   23:02:53          Pearson chi2:              741.
No. Iterations:         9                 Pseudo R-squ. (CS):        0.2858
Covariance Type:        nonrobust
=====
```

```
=====
coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
Intercept      25.3367      8.261      3.067      0.002      9.145
41.528
C(Location) [T.4] -0.6922      0.172     -4.016      0.000     -1.030
-0.354
C(Location) [T.5] -0.9538      0.161     -5.913      0.000     -1.270
-0.638
C(Location) [T.6] -1.1281      0.189     -5.966      0.000     -1.499
-0.757
C(Location) [T.7] -0.8022      0.160     -5.012      0.000     -1.116
-0.489
C(Location) [T.8] -0.8423      0.161     -5.232      0.000     -1.158
-0.527
C(Location) [T.9] -0.9796      0.177     -5.533      0.000     -1.327
-0.633
C(Location) [T.10] -0.7119      0.165     -4.318      0.000     -1.035
-0.389
C(Location) [T.11] -0.7103      0.165     -4.311      0.000     -1.033
-0.387
C(Location) [T.12] -0.8244      0.173     -4.764      0.000     -1.164
-0.485
C(Location) [T.13] -1.0489      0.170     -6.162      0.000     -1.383
-0.715
```

C(Location) [T.14]	-1.4833	0.172	-8.645	0.000	-1.820
-1.147					
C(Location) [T.15]	-0.9933	0.189	-5.256	0.000	-1.364
-0.623					
C(Location) [T.16]	-1.3900	0.160	-8.661	0.000	-1.705
-1.075					
C(Location) [T.17]	-1.7464	0.271	-6.451	0.000	-2.277
-1.216					
C(Location) [T.18]	-1.1114	0.179	-6.220	0.000	-1.462
-0.761					
C(Location) [T.19]	-1.1061	0.172	-6.424	0.000	-1.444
-0.769					
C(Location) [T.20]	-0.9822	0.175	-5.628	0.000	-1.324
-0.640					
C(Location) [T.21]	-0.7526	0.167	-4.501	0.000	-1.080
-0.425					
C(Location) [T.22]	-0.7485	0.177	-4.236	0.000	-1.095
-0.402					
C(Location) [T.23]	-0.7310	0.172	-4.257	0.000	-1.068
-0.394					
C(Location) [T.26]	-0.9643	0.208	-4.636	0.000	-1.372
-0.557					
C(Location) [T.27]	-1.5192	0.168	-9.050	0.000	-1.848
-1.190					
C(Location) [T.28]	-1.5181	0.188	-8.069	0.000	-1.887
-1.149					
C(Location) [T.29]	-0.7893	0.158	-4.990	0.000	-1.099
-0.479					
C(Location) [T.30]	-0.8331	0.166	-5.025	0.000	-1.158
-0.508					
C(Location) [T.32]	-0.7520	0.151	-4.973	0.000	-1.048
-0.456					
C(Location) [T.33]	-0.6771	0.165	-4.102	0.000	-1.001
-0.354					
C(Location) [T.34]	-1.1143	0.166	-6.731	0.000	-1.439
-0.790					
C(Location) [T.35]	-0.9400	0.160	-5.865	0.000	-1.254
-0.626					
C(Location) [T.36]	-0.9992	0.171	-5.836	0.000	-1.335
-0.664					
C(Location) [T.38]	-1.0869	0.171	-6.357	0.000	-1.422
-0.752					
C(Location) [T.39]	-0.9236	0.175	-5.282	0.000	-1.266
-0.581					
C(Location) [T.40]	-1.3916	0.184	-7.558	0.000	-1.753
-1.031					
C(Location) [T.41]	-0.6918	0.158	-4.365	0.000	-1.002
-0.381					

C(Location) [T.42]	-0.8857	0.217	-4.079	0.000	-1.311
-0.460					
C(Location) [T.43]	-0.5751	0.160	-3.601	0.000	-0.888
-0.262					
C(Location) [T.44]	-1.3576	0.163	-8.336	0.000	-1.677
-1.038					
C(Location) [T.45]	-1.1368	0.155	-7.339	0.000	-1.440
-0.833					
C(Location) [T.46]	-0.8168	0.169	-4.831	0.000	-1.148
-0.485					
C(Location) [T.47]	-0.9518	0.165	-5.768	0.000	-1.275
-0.628					
C(Location) [T.48]	-1.7404	0.175	-9.940	0.000	-2.084
-1.397					
C(Location) [T.49]	-1.1373	0.184	-6.171	0.000	-1.499
-0.776					
Min_Temp	0.1094	0.021	5.238	0.000	0.068
0.150					
Max_Temp	-0.1025	0.021	-4.872	0.000	-0.144
-0.061					
Parameter1_Speed	0.0640	0.008	8.328	0.000	0.049
0.079					
Parameter3_9am	-0.0077	0.010	-0.751	0.453	-0.028
0.012					
Parameter3_3pm	-0.0652	0.010	-6.629	0.000	-0.084
-0.046					
Parameter4_9am	0.0162	0.004	3.601	0.000	0.007
0.025					
Parameter4_3pm	0.0242	0.006	4.386	0.000	0.013
0.035					
Parameter5_3pm	-0.0248	0.008	-3.120	0.002	-0.040
-0.009					

=====

=====

