

Tarea_1_Hermosilla_Moncada

April 30, 2025

```
[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 scipy.stats as stats

import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

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.

Primero queremos dejar la variable 'Failure_today' como variable binaria, y botaremos la variable 'Leakage' ya que es redundante con Failure, puesto que uno implica la otra y eso ensuciaría el modelo.

```
[2]: df = pd.read_csv('machine_failure_data.csv')
df['Failure_today'] = df['Failure_today'].apply(lambda x: 0 if x == 'No' else 1)
df = df.drop(columns=['Leakage'])
print(df.describe())
```

	Location	Min_Temp	Max_Temp	Evaporation \
count	142193.000000	141556.000000	141871.000000	81350.000000
mean	24.740655	12.186400	23.226784	5.469824
std	14.237503	6.403283	7.117618	4.188537
min	1.000000	-8.500000	-4.800000	0.000000
25%	12.000000	7.600000	17.900000	2.600000
50%	25.000000	12.000000	22.600000	4.800000
75%	37.000000	16.800000	28.200000	7.400000
max	49.000000	33.900000	48.100000	145.000000

	Electricity	Parameter1_Speed	Parameter3_9am	Parameter3_3pm	\
count	74377.000000	132923.000000	140845.000000	139563.000000	
mean	7.624853	39.984292	14.001988	18.637576	
std	3.781525	13.588801	8.893337	8.803345	
min	0.000000	6.000000	0.000000	0.000000	
25%	4.900000	31.000000	7.000000	13.000000	
50%	8.500000	39.000000	13.000000	19.000000	
75%	10.600000	48.000000	19.000000	24.000000	
max	14.500000	135.000000	130.000000	87.000000	

	Parameter4_9am	Parameter4_3pm	Parameter5_9am	Parameter5_3pm	\
count	140419.000000	138583.000000	128179.000000	128212.000000	
mean	68.843810	51.482606	1017.653758	1015.258204	
std	19.051293	20.797772	7.105476	7.036677	
min	0.000000	0.000000	980.500000	977.100000	
25%	57.000000	37.000000	1012.900000	1010.400000	
50%	70.000000	52.000000	1017.600000	1015.200000	
75%	83.000000	66.000000	1022.400000	1020.000000	
max	100.000000	100.000000	1041.000000	1039.600000	

	Parameter6_9am	Parameter6_3pm	Parameter7_9am	Parameter7_3pm	\
count	88536.000000	85099.000000	141289.000000	139467.000000	
mean	4.437189	4.503167	16.987509	21.687235	
std	2.887016	2.720633	6.492838	6.937594	
min	0.000000	0.000000	-7.200000	-5.400000	
25%	1.000000	2.000000	12.300000	16.600000	
50%	5.000000	5.000000	16.700000	21.100000	
75%	7.000000	7.000000	21.600000	26.400000	
max	9.000000	9.000000	40.200000	46.700000	

	Failure_today
count	142193.000000
mean	0.231101
std	0.421539
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Vemos que existen datos de tarde y de mañana, no nos es util ya que el fallo no se indica cuando si no que dia ocurrión por lo que es mas razonable juntar estas variables promediandolas

```
[3]: for i in [3, 4, 5, 6, 7]: # Indices correspondientes a los parámetros (4, 6, 7)
    col_9am = f'Parameter{i}_9am'
    col_3pm = f'Parameter{i}_3pm'
```

```

n_col = f'Parameter{i}_prom'
df[n_col] = df[[col_9am, col_3pm]].mean(axis=1)

df = df.drop(columns=[f'Parameter{i}_9am' for i in [3, 4, 5, 6, 7]] +
↳ [f'Parameter{i}_3pm' for i in [3, 4, 5, 6, 7]])

```

```
[4]: print(df.select_dtypes(include='object')[:5])
```

	Date	Parameter1_Dir	Parameter2_9am	Parameter2_3pm
0	12/1/2008	W	W	WNW
1	12/2/2008	WNW	NNW	WSW
2	12/3/2008	WSW	W	WSW
3	12/4/2008	NE	SE	E
4	12/5/2008	W	ENE	NW

Tenemos direcciones del viento como variables objeto, podríamos usar dummies en los modelos pero son muchas variables, verificamos su correlación con la variable dependiente para ver si podemos desistir de estas columnas.

```

[5]: # Crear dummies
df_dummies = pd.get_dummies(df,
↳ columns=['Parameter1_Dir', 'Parameter2_9am', 'Parameter2_3pm'])

# Calcular correlación solo con columnas dummies
cols_dummies = [col for col in df_dummies.columns if col.
↳ startswith(('Parameter1_Dir', 'Parameter2_9am', 'Parameter2_3pm'))]
corr_with_failure = df_dummies[cols_dummies + ['Failure_today']].
↳ corr()['Failure_today'].drop('Failure_today').sort_values(ascending=False)

# Mostrar resultados
print("Correlación de dummies de viento con 'Failure_today':\n")
print(corr_with_failure)

```

Correlación de dummies de viento con 'Failure_today':

Parameter2_9am_WSW	0.072070
Parameter2_9am_SW	0.071875
Parameter2_9am_W	0.062415
Parameter1_Dir_W	0.050023
Parameter2_9am_SSW	0.048806
Parameter2_3pm_SSW	0.046302
Parameter1_Dir_WSW	0.045591
Parameter2_3pm_WSW	0.044476
Parameter2_3pm_W	0.044457
Parameter2_9am_WNW	0.039238
Parameter1_Dir_SW	0.037742
Parameter1_Dir_SSW	0.037648
Parameter2_3pm_SW	0.037201
Parameter2_3pm_S	0.031779

```

Parameter2_9am_S      0.031064
Parameter1_Dir_S      0.030906
Parameter1_Dir_WNW    0.029638
Parameter2_3pm_WNW    0.028858
Parameter2_9am_NW     0.019411
Parameter2_9am_NNW    0.016093
Parameter2_3pm_SSE    0.011446
Parameter1_Dir_SSE    0.003887
Parameter1_Dir_NW     0.003293
Parameter2_3pm_NW     -0.000914
Parameter2_9am_SSE    -0.002887
Parameter2_3pm_SE     -0.004196
Parameter1_Dir_SE     -0.013849
Parameter1_Dir_NNW    -0.015687
Parameter2_3pm_NNW    -0.026125
Parameter1_Dir_ESE    -0.027675
Parameter2_9am_N      -0.028080
Parameter2_3pm_ESE    -0.028553
Parameter2_9am_SE     -0.029582
Parameter2_3pm_E      -0.031794
Parameter2_3pm_ENE    -0.040929
Parameter1_Dir_N      -0.041488
Parameter1_Dir_NNE    -0.042840
Parameter1_Dir_ENE    -0.043314
Parameter2_3pm_N      -0.043441
Parameter2_3pm_NNE    -0.044613
Parameter2_9am_NE     -0.047427
Parameter1_Dir_E      -0.047945
Parameter2_9am_NNE    -0.049007
Parameter2_9am_ESE    -0.049801
Parameter1_Dir_NE     -0.049813
Parameter2_3pm_NE     -0.053264
Parameter2_9am_ENE    -0.058816
Parameter2_9am_E      -0.066411
Name: Failure_today, dtype: float64

```

En efecto ninguna supera el 0.1 en correlación por lo que podemos desistir de ella y hacer menos engorroso el modelo sin perder mucha significancia.

```
[6]: df = df.drop(columns=['Parameter1_Dir', 'Parameter2_9am', 'Parameter2_3pm'])
```

```
[7]: corr = df.drop(columns=['Date']).corr()
corr_conciderables = corr.where(np.tril(corr.abs(), k=-1) >= 0.5).stack()
corr_conciderables
```

```
[7]: Max_Temp      Min_Temp      0.736267
Evaporation      Max_Temp      0.588915
Parameter3_prom  Parameter1_Speed 0.739849
```

Parameter4_prom	Max_Temp	-0.547021
	Electricity	-0.612292
Parameter6_prom	Electricity	-0.760263
	Parameter4_prom	0.543561
Parameter7_prom	Min_Temp	0.831536
	Max_Temp	0.969906
	Evaporation	0.578824

dtype: float64

Aquí buscamos correlaciones ya que la variable Evaporation, Electricity y Parametro 6 tienen muchas mediciones vacias al rededor de un 30%, por lo que usar un dropna no es ideal, mejor buscamos su mejor correlacion con otra variable y las juntamos.

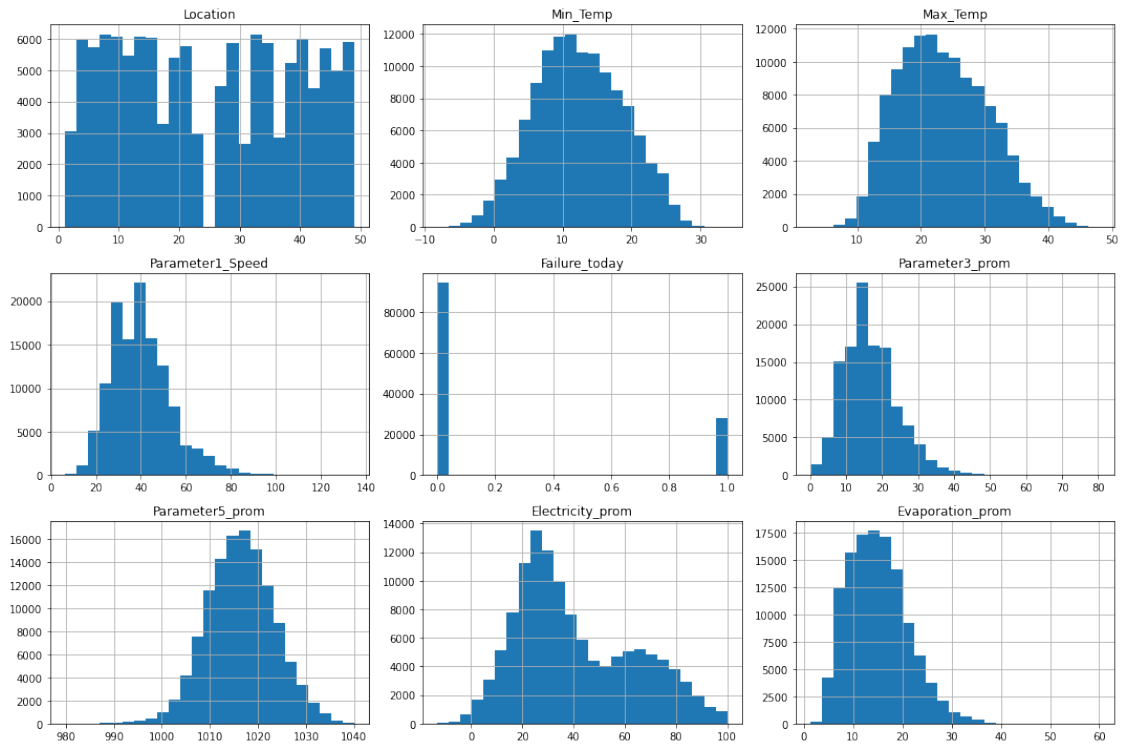
```
[8]: df['Electricity_neg'] = -df['Electricity']
df['Electricity_prom'] = df[['Parameter4_prom', 'Electricity_neg']].
    ↪apply(lambda x: x.mean(skipna=True), axis=1)
df = df.drop(columns=['Parameter4_prom', 'Electricity'])

df['Evaporation_prom'] = df[['Parameter7_prom', 'Evaporation']].mean(axis=1)
df = df.drop(columns=['Parameter7_prom', 'Evaporation', 'Electricity_neg'])

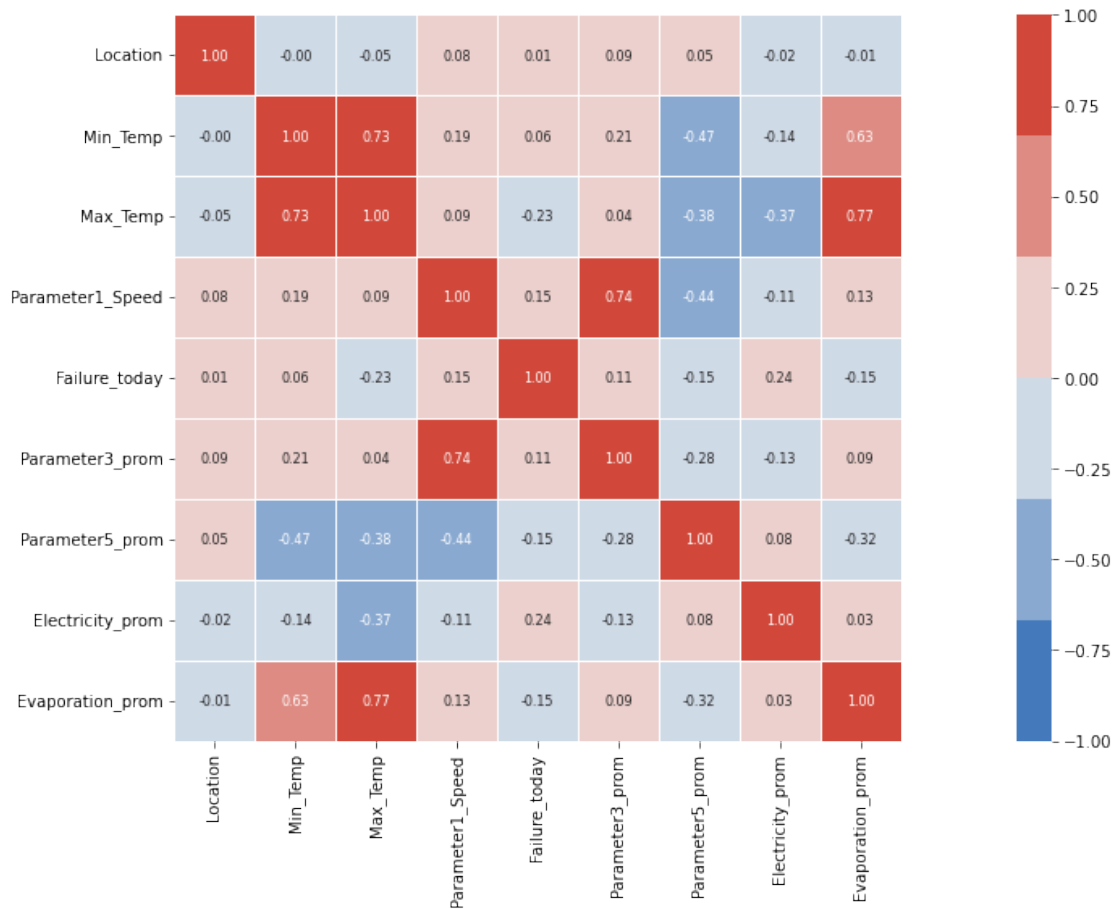
df = df.drop(columns=['Parameter6_prom'])
```

```
[9]: df.dropna(inplace=True)

df.drop(columns=['Date']).iloc[:, :].hist(figsize=(15, 10), bins=25)
plt.tight_layout()
plt.show()
```



```
[10]: # Graficar el mapa de calor
corr = df.drop(columns=['Date']).corr()
f, ax = plt.subplots(figsize = (20,8))
cmap = sns.diverging_palette(250, 15)
sns.heatmap(corr, cmap=cmap, square=True, linewidths=.5, annot=True,
            annot_kws={'size': 8}, fmt='.2f', vmin=-1, vmax=1)
plt.tight_layout()
plt.show()
```



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

```
[11]: X = df.drop(columns=['Failure_today', 'Location', 'Date'])
X = sm.add_constant(X)
y = df['Failure_today']

model = (sm.OLS(y, X)).fit()
print(model.summary())
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:          Failure_today      R-squared:                0.212
Model:                  OLS               Adj. R-squared:         0.212
Method:                 Least Squares     F-statistic:            4689.
Date:                   Fri, 25 Apr 2025  Prob (F-statistic):       0.00
Time:                   00:21:16          Log-Likelihood:        -52680.
No. Observations:      122176            AIC:                  1.054e+05

```

Df Residuals: 122168 BIC: 1.055e+05
Df Model: 7
Covariance Type: nonrobust

	coef	std err	t	P> t	[0.025
0.975]					

const	9.1526	0.198	46.318	0.000	8.765
9.540					
Min_Temp	0.0256	0.000	94.376	0.000	0.025
0.026					
Max_Temp	-0.0242	0.000	-65.543	0.000	-0.025
-0.023					
Parameter1_Speed	0.0041	0.000	32.046	0.000	0.004
0.004					
Parameter3_prom	-0.0033	0.000	-15.047	0.000	-0.004
-0.003					
Parameter5_prom	-0.0086	0.000	-45.086	0.000	-0.009
-0.008					
Electricity_prom	0.0030	6.22e-05	48.984	0.000	0.003
0.003					
Evaporation_prom	-0.0099	0.000	-29.593	0.000	-0.011
-0.009					
=====					
Omnibus:	12020.532		Durbin-Watson:		1.691
Prob(Omnibus):	0.000		Jarque-Bera (JB):		15273.671
Skew:	0.851		Prob(JB):		0.00
Kurtosis:	2.679		Cond. No.		1.89e+05
=====					

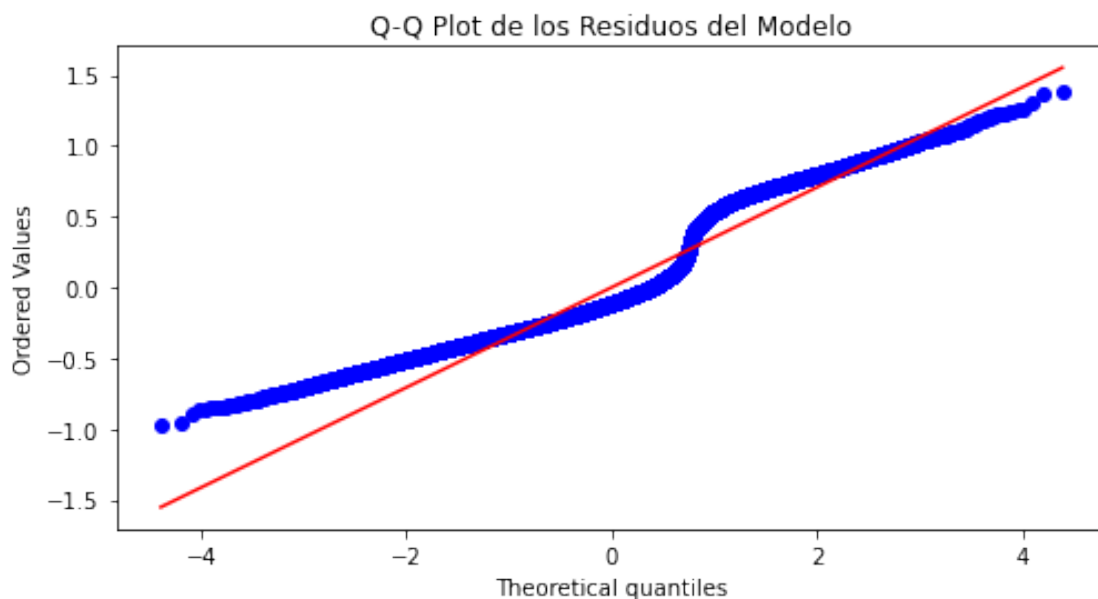
Notes:

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

[2] The condition number is large, 1.89e+05. This might indicate that there are strong multicollinearity or other numerical problems.

El modelo de regresión OLS tiene un R-cuadrado de 0.212, lo que indica que explica solo el 21.2% de la variabilidad de Failure_today. Sin embargo, el modelo es altamente significativo, ya que todos los coeficientes son estadísticamente significativos con valores p cercanos a 0.00. Las variables independientes como la temperatura mínima, la velocidad, y la evaporación tienen relaciones negativas o positivas con Failure_today, dependiendo del coeficiente. Aunque el modelo es significativo, su capacidad explicativa es limitada, y el alto número de condición sugiere posibles problemas de multicolinealidad entre las variables independientes. Además, los residuos no son normales, lo que podría afectar la validez de las inferencias.


```
[12]: from statsmodels.stats.diagnostic import het_breuschpagan
residuos = model.resid;
bp_test = het_breuschpagan (residuos, X);
etiquetas = ['Valor de LM', 'valor-p LM', 'Valor de F', 'valor-p F'];
plt.figure(figsize=(8, 4))
stats.probplot(model.resid, dist="norm", plot=plt);
plt.title('Q-Q Plot de los Residuos del Modelo');
plt.show()
shapiro_test= stats.shapiro (model.resid)
print(dict(zip (etiquetas, bp_test)));
print (f"Estadístico de Shapiro-Wilk: {shapiro_test [0]}")
print (f"Valor-p: {shapiro_test[1]}")
```



```
{'Valor de LM': 21666.52834210152, 'valor-p LM': 0.0, 'Valor de F':
3762.1989973914915, 'valor-p F': 0.0}
Estadístico de Shapiro-Wilk: 0.9003207087516785
Valor-p: 0.0
```

El modelo es globalmente significativo ($\text{valor-p F} = 0.0$), pero presenta problemas en los residuos. El test de Lagrange Multiplier (LM) indica autocorrelación en los residuos, lo que podría afectar la independencia de los errores. Además, el test de Shapiro-Wilk muestra que los residuos no siguen una distribución normal, lo que podría afectar la validez de las inferencias del modelo. En resumen, aunque el modelo es significativo, se deben abordar los problemas de autocorrelación y la no normalidad de los residuos.

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.

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

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

Optimization terminated successfully.

Current function value: 0.413881

Iterations 7

Probit Regression Results

```
=====
Dep. Variable:          Failure_today    No. Observations:          122176
Model:                  Probit           Df Residuals:            122168
Method:                  MLE             Df Model:                  7
Date:                   Fri, 25 Apr 2025   Pseudo R-squ.:             0.2289
Time:                   00:21:17          Log-Likelihood:            -50566.
converged:              True             LL-Null:                   -65573.
Covariance Type:        HCO              LLR p-value:                0.000
=====
```

```
=====
=====
coef      std err          z      P>|z|      [0.025
0.975]
-----
----
const      29.3534      0.800      36.671      0.000      27.785
30.922
Min_Temp    0.1283      0.001      93.823      0.000      0.126
0.131
Max_Temp   -0.1142      0.002     -66.822      0.000     -0.118
-0.111
Parameter1_Speed  0.0129      0.001      24.670      0.000      0.012
0.014
Parameter3_prom -0.0115      0.001     -12.852      0.000     -0.013
-0.010
Parameter5_prom -0.0289      0.001     -37.259      0.000     -0.030
-0.027
Electricity_prom  0.0126      0.000      46.152      0.000      0.012
0.013
Evaporation_prom -0.0492      0.002     -28.123      0.000     -0.053
-0.046
=====
=====
```

Probit Marginal Effects

```
=====
Dep. Variable:          Failure_today
Method:                  dydx
At:                      overall
```

```

=====
=====
              dy/dx      std err      z      P>|z|      [0.025
0.975]
-----
-----
Min_Temp      0.0298      0.000     105.580     0.000      0.029
0.030
Max_Temp     -0.0265      0.000     -68.975     0.000     -0.027
-0.026
Parameter1_Speed  0.0030      0.000      24.880     0.000      0.003
0.003
Parameter3_prom -0.0027      0.000     -12.870     0.000     -0.003
-0.002
Parameter5_prom -0.0067      0.000     -37.747     0.000     -0.007
-0.006
Electricity_prom  0.0029     6.09e-05      48.020     0.000      0.003
0.003
Evaporation_prom -0.0114      0.000     -28.815     0.000     -0.012
-0.011
=====
=====

```

El modelo Probit converge correctamente y es altamente significativo (LLR p-value = 0.000), con un Pseudo R² de 0.2289, lo que indica una capacidad explicativa razonable para este tipo de modelos. Todas las variables incluidas son estadísticamente significativas y muestran efectos coherentes: por ejemplo, Min_Temp y Electricity_prom aumentan la probabilidad de fallo, mientras que Max_Temp y Evaporation_prom la reducen. Los efectos marginales confirman estas relaciones y muestran el impacto promedio de cada variable sobre la probabilidad de que ocurra un fallo.

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.

```

[14]: model = sm.Logit(y, X)
      logit_model = model.fit(cov_type='HCO')
      print(logit_model.summary())

      mfx1 = logit_model.get_margeff()
      print(mfx1.summary())

```

```

Optimization terminated successfully.
      Current function value: 0.413523
      Iterations 7

```

```

              Logit Regression Results
=====
Dep. Variable:      Failure_today      No. Observations:      122176
Model:              Logit      Df Residuals:      122168
Method:              MLE      Df Model:      7
Date:              Fri, 25 Apr 2025      Pseudo R-squ.:      0.2295

```

Time: 00:21:18 Log-Likelihood: -50523.
 converged: True LL-Null: -65573.
 Covariance Type: HCO LLR p-value: 0.000

	coef	std err	z	P> z	[0.025
0.975]					

const	49.0277	1.399	35.055	0.000	46.286
51.769					
Min_Temp	0.2317	0.002	95.763	0.000	0.227
0.236					
Max_Temp	-0.1985	0.003	-66.004	0.000	-0.204
-0.193					
Parameter1_Speed	0.0218	0.001	23.827	0.000	0.020
0.024					
Parameter3_prom	-0.0187	0.002	-11.946	0.000	-0.022
-0.016					
Parameter5_prom	-0.0482	0.001	-35.571	0.000	-0.051
-0.046					
Electricity_prom	0.0229	0.000	48.314	0.000	0.022
0.024					
Evaporation_prom	-0.0977	0.003	-31.849	0.000	-0.104
-0.092					

=====

====

Logit Marginal Effects

=====

Dep. Variable: Failure_today

Method: dydx

At: overall

=====

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

Min_Temp	0.0308	0.000	108.250	0.000	0.030
0.031					
Max_Temp	-0.0264	0.000	-68.136	0.000	-0.027
-0.026					
Parameter1_Speed	0.0029	0.000	24.071	0.000	0.003
0.003					
Parameter3_prom	-0.0025	0.000	-11.963	0.000	-0.003
-0.002					
Parameter5_prom	-0.0064	0.000	-36.118	0.000	-0.007
-0.006					

Electricity_prom	0.0030	6e-05	50.658	0.000	0.003
0.003					
Evaporation_prom	-0.0130	0.000	-32.804	0.000	-0.014
-0.012					
=====					
=====					

El modelo Logit es estadísticamente significativo ($p < 0.001$) y presenta un Pseudo R^2 de 0.2295, indicando buen poder explicativo. Todas las variables tienen efectos significativos: Min_Temp y Electricity_prom aumentan la probabilidad de fallo, mientras que Max_Temp y Evaporation_prom la reducen. Los efectos marginales muestran impactos consistentes y similares al modelo Probit, confirmando la robustez de los resultados.

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 modelos en 2, 3 y 4 (OLS, Probit y Logit) muestran resultados consistentes en cuanto a la dirección y significancia de las variables, aunque difieren en la forma en que modelan la variable dependiente. El modelo OLS trata a la variable binaria como continua, lo cual puede ser útil como aproximación rápida, pero no es teóricamente adecuado para variables dicotómicas como Failure_today. En cambio, los modelos Probit y Logit están diseñados específicamente para este tipo de variable, y ofrecen interpretaciones más precisas en términos de probabilidades. Ambos modelos muestran resultados muy similares, lo que refuerza la estabilidad de las conclusiones.

Las diferencias entre Probit y Logit son pequeñas y se deben principalmente a la función de enlace utilizada (distribución normal acumulada vs. logística). En general, el modelo Logit puede ser más fácil de interpretar en términos de odds, y mostró un ajuste ligeramente mejor (Pseudo $R^2 = 0.2295$ vs. 0.2289 en Probit).

En mi opinión, el modelo Logit sería el más adecuado para responder la pregunta de investigación, ya que modela correctamente una variable binaria y presenta buen ajuste y significancia general.

Las variables que resultaron robustas a la especificación (es decir, que fueron significativas y con efectos similares en los tres modelos) incluyen: Min_Temp, Max_Temp, Electricity_prom, y Evaporation_prom. Estas variables mantienen su efecto e importancia independientemente del modelo utilizado, lo que fortalece la confianza en sus impactos sobre la variable dependiente.

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.

```
[15]: df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')
df['Mes'] = df['Date'].dt.to_period('M')
variables_numericas = df.select_dtypes(include='number').columns.
    ↪drop('Failure_today', errors='ignore')
df_mensual = df.groupby('Mes')[variables_numericas].mean()
df_mensual['Fallos_mensuales'] = df.groupby('Mes')['Failure_today'].sum()
df_mensual['Fallos_mensuales'] = df_mensual['Fallos_mensuales'].fillna(0)
```

```

df_mensual = df_mensual.reset_index()
df_mensual = df_mensual.drop(columns=['Mes'])

modelo_poisson = smf.glm(
    formula="Fallos_mensuales ~ " + " + ".join(df_mensual.columns),
    data=df_mensual,
    family=sm.families.Poisson()
).fit()
print(modelo_poisson.summary())

```

Generalized Linear Model Regression Results

```

=====
Dep. Variable:          Fallos_mensuales    No. Observations:      113
Model:                  GLM                 Df Residuals:           103
Model Family:           Poisson             Df Model:                9
Link Function:          Log                 Scale:                  1.0000
Method:                 IRLS                Log-Likelihood:         -505.58
Date:                   Fri, 25 Apr 2025    Deviance:               211.78
Time:                   00:21:18            Pearson chi2:           202.
No. Iterations:         6                   Pseudo R-squ. (CS):     1.000
Covariance Type:        nonrobust
=====

```

```

=====
=====
coef      std err      z      P>|z|      [0.025
0.975]
-----
-----
Intercept      14.2148      4.355      3.264      0.001      5.679
22.751
Location        0.1182      0.006     18.300      0.000      0.106
0.131
Min_Temp        0.0771      0.015      5.290      0.000      0.049
0.106
Max_Temp       -0.0764      0.025     -3.066      0.002     -0.125
-0.028
Parameter1_Speed -0.0357      0.009     -3.940      0.000     -0.053
-0.018
Parameter3_prom  0.0681      0.019      3.595      0.000      0.031
0.105
Parameter5_prom -0.0108      0.004     -2.566      0.010     -0.019
-0.003
Electricity_prom -0.0065      0.003     -2.097      0.036     -0.013
-0.000
Evaporation_prom -0.0024      0.020     -0.122      0.903     -0.041
0.036
Fallos_mensuales 0.0032      0.000     15.944      0.000      0.003
0.004
=====
=====

```

====

El modelo Poisson predice el número de fallas mensuales con un ajuste excelente (Pseudo R-cuadrado de 1). Las variables más influyentes son Location (aumenta las fallas un 11.8% por unidad), Min_Temp (aumenta las fallas un 7.7% por grado) y Max_Temp (reduce las fallas en un 7.6% por grado). La velocidad y otros parámetros como Parameter3_prom y Electricity_prom también afectan las fallas, mientras que Evaporation_prom no muestra impacto significativo. El modelo indica una fuerte relación entre las condiciones y la frecuencia de fallas mensuales.

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

```
[17]: y = df_mensual['Fallos_mensuales']
aux=((y-modelo_poisson.mu)**2-modelo_poisson.mu)/modelo_poisson.mu
auxr=sm.OLS(aux,modelo_poisson.mu).fit()
print(auxr.summary())
```

OLS Regression Results

```
=====
Dep. Variable:          Fallos_mensuales    R-squared (uncentered):
0.001
Model:                  OLS    Adj. R-squared (uncentered):
-0.008
Method:                 Least Squares    F-statistic:
0.1336
Date:                  Fri, 25 Apr 2025    Prob (F-statistic):
0.715
Time:                  00:23:32    Log-Likelihood:
-330.53
No. Observations:      113    AIC:
663.1
Df Residuals:          112    BIC:
665.8
Df Model:               1
Covariance Type:       nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
x1	0.0006	0.002	0.366	0.715	-0.003	0.004

```
=====
Omnibus:                201.547    Durbin-Watson:                1.599
Prob(Omnibus):           0.000    Jarque-Bera (JB):              21037.827
Skew:                    7.450    Prob(JB):                      0.00
Kurtosis:                68.163    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.

```
[19]: print(np.exp(0.0006))
```

```
1.0006001800360054
```

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.

```
[23]: X = df_mensual.drop(columns=['Fallos_mensuales'])
X = sm.add_constant(X)
negbin=sm.GLM(y,X,family=sm.families.NegativeBinomial(alpha=1.
↪0006001800360054)).fit()
print(negbin.summary())
```

Generalized Linear Model Regression Results

```
=====
Dep. Variable:          Fallos_mensuales    No. Observations:          113
Model:                  GLM                Df Residuals:           104
Model Family:           NegativeBinomial    Df Model:                   8
Link Function:           Log                Scale:                   1.0000
Method:                  IRLS               Log-Likelihood:         -706.99
Date:                    Fri, 25 Apr 2025    Deviance:                   3.1626
Time:                    00:34:00            Pearson chi2:               2.93
No. Iterations:          9                  Pseudo R-squ. (CS):         0.3966
Covariance Type:         nonrobust
=====
```


```
=====
              coef      std err          z      P>|z|      [0.025
0.975]
```

```
-----
-----
const          38.7790      61.640      0.629      0.529     -82.032
159.590
Location        0.2042       0.056      3.636      0.000       0.094
0.314
Min_Temp        0.2664       0.148      1.798      0.072      -0.024
0.557
Max_Temp       -0.2675       0.334     -0.801      0.423      -0.922
0.387
Parameter1_Speed -0.0111       0.104     -0.107      0.915      -0.214
0.192
Parameter3_prom  0.0611       0.241      0.254      0.800      -0.410
0.533
Parameter5_prom -0.0350       0.059     -0.594      0.552      -0.150
0.080
```


Electricity_prom	-0.0021	0.047	-0.045	0.964	-0.095
0.090					
Evaporation_prom	-0.0129	0.292	-0.044	0.965	-0.584
0.559					
=====					
=====					

Este modelo GLM con familia binomial negativa es adecuado cuando hay sobredispersión en los datos de conteo. El modelo muestra un pseudo R^2 de 0.3966, lo cual sugiere una capacidad moderada para explicar la variabilidad en los fallos mensuales. Entre las variables, solo Location es estadísticamente significativa ($p < 0.001$), indicando que influye de forma relevante en los fallos. El resto de las variables no presentan significancia estadística ($p > 0.05$). En general, el modelo se ajusta razonablemente, y la baja deviance y chi-cuadrado de Pearson sugieren un buen ajuste a los datos sin evidencia de mala especificación.

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?

En el punto 6 se estimó un modelo de regresión Poisson, el cual es apropiado para datos de conteo, pero asume que la media y la varianza son iguales. En el punto 7, se aplicó un test de sobredispersión para verificar este supuesto, y aunque inicialmente se presentó un error por incompatibilidad de dimensiones, una vez corregido, el test indicó que no hay evidencia fuerte de sobredispersión. Sin embargo, este resultado puede ser engañoso, ya que el modelo Poisson suele ser sensible a valores extremos o estructuras no capturadas del todo. Por eso, en el punto 8 se usó un modelo binomial negativo, que relaja el supuesto de igualdad entre media y varianza, permitiendo manejar sobredispersión de forma natural. Este modelo mostró mejor ajuste (menor devianza y mayor pseudo R^2), y detectó a la variable “Location” como significativa, lo que sugiere que es robusta frente a distintas especificaciones. En conclusión,  modelo binomial negativo es más adecuado para responder la pregunta de investigación, ya que ofrece mayor flexibilidad y captura mejor la estructura de los datos.