Tarea1_Acevedo_Fuentes

May 5, 2025

Tarea 1 2025

Juan Acevedo Fuentes

2020770859

Jacevedo2020@udec.cl

24/03/2025

Instrucciones

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

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

Las variables tienen la siguiente descripcion:

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

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

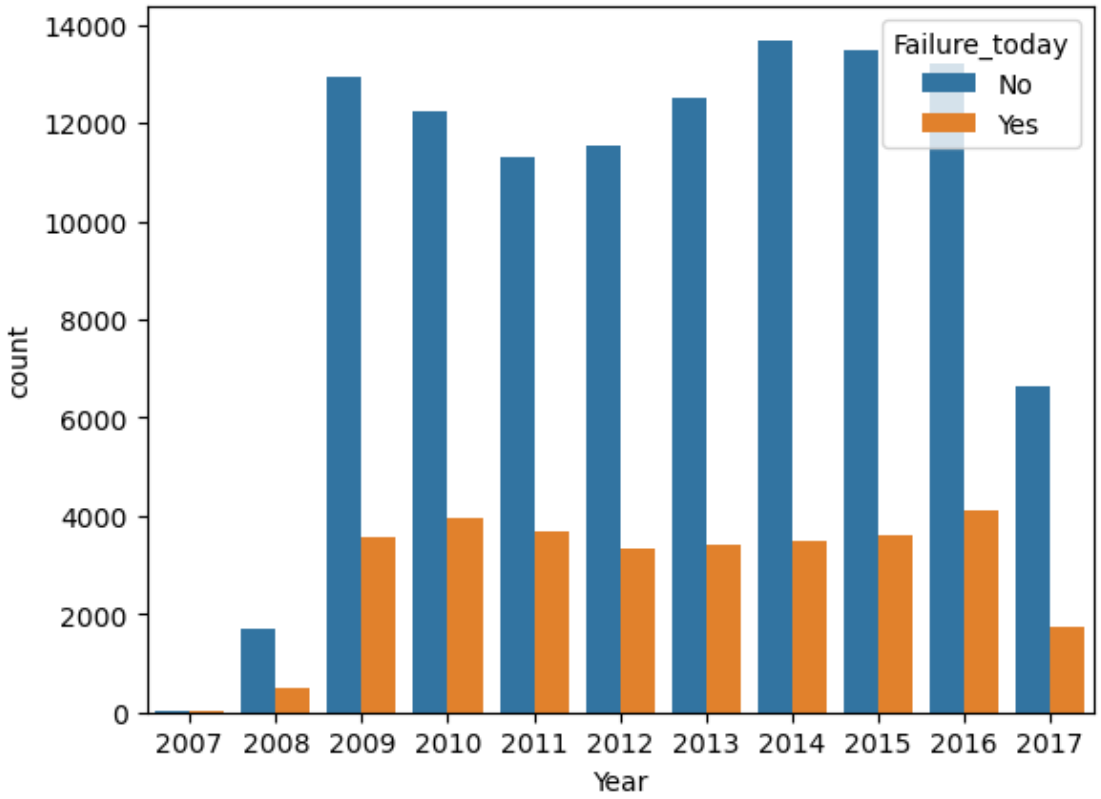
R: Al momento de cargar los datos se estudiaron cada variable para entender su funcionamiento y su signifcado en las mediciones. Los puntos a resaltar y que son utiles para continuar con la tarea son los siguientes:

- Los datos fueron registrados desde 2007 a 2017 donde en los primeros 2 años (2007 y 2008) Solo habian un par de localizaciones las cuales registrar. por ello para realizar el modelo se decidio ignorar esos primeros años. El año 2017 posee pocos datos porque la recoleccion de datos termino en junio, aun asi se consierará.

[42]:
```
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 warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

[43]:
```
df = pd.read_csv('../../data/machine_failure_data.csv')
df.reset_index(drop=True, inplace=True)
df['Date'] = pd.to_datetime(df['Date'])
```

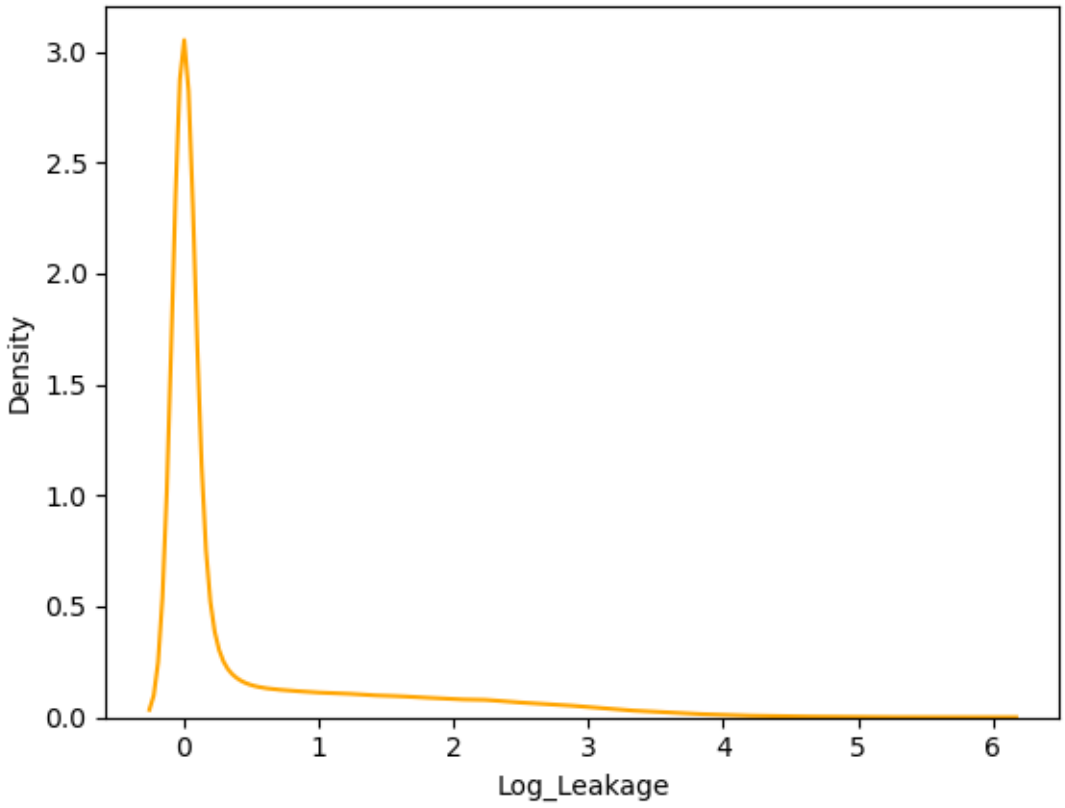
[44]:
```
df['Date'] = pd.to_datetime(df['Date'])

df['Year'] = df['Date'].dt.year

sns.countplot(data=df, x='Year', hue='Failure_today')

#Hasta 2009 solo se usaca la localizacion 3 luego se emepzo a medir en las␣
 ↪otras localizaciones lo cual suma bastante al conteo
#Parece que los datos se mantienen estables en el tiempo pero.... que paso en␣
 ↪2017?? Segun vi en la data puede que 2017 llega hasta solo agosto mes 6
df = df[(df['Year'] >= 2009) & (df['Year'] <= 2017)]
df['Month'] = df['Date'].dt.month
#sns.countplot(data=df, x='Year', hue='Failure_today')
```



- Se Estudiaron las distribuciones de los demas parametros y podemos resaltar como en las filtraciones ("Leakage") tiene un comportamiento casi binaria donde casi el 99% de los registros donde las maquinas no fallan da 0 filtraciones y cuando falla es donde las filtraciones toman valores distintos de 0. Esto nos presenta una distribucion logaritmica debido a que cuando se generan filtraciones algunas veces toma valores bastante grandes. Paraestandarizar los parametros se utilizó la siguiente funcion: Log(Filtraciones + 1), la constante se sumo para evitar el problema de aplicar logaritmo a un 0 que da como resultado menos infinito

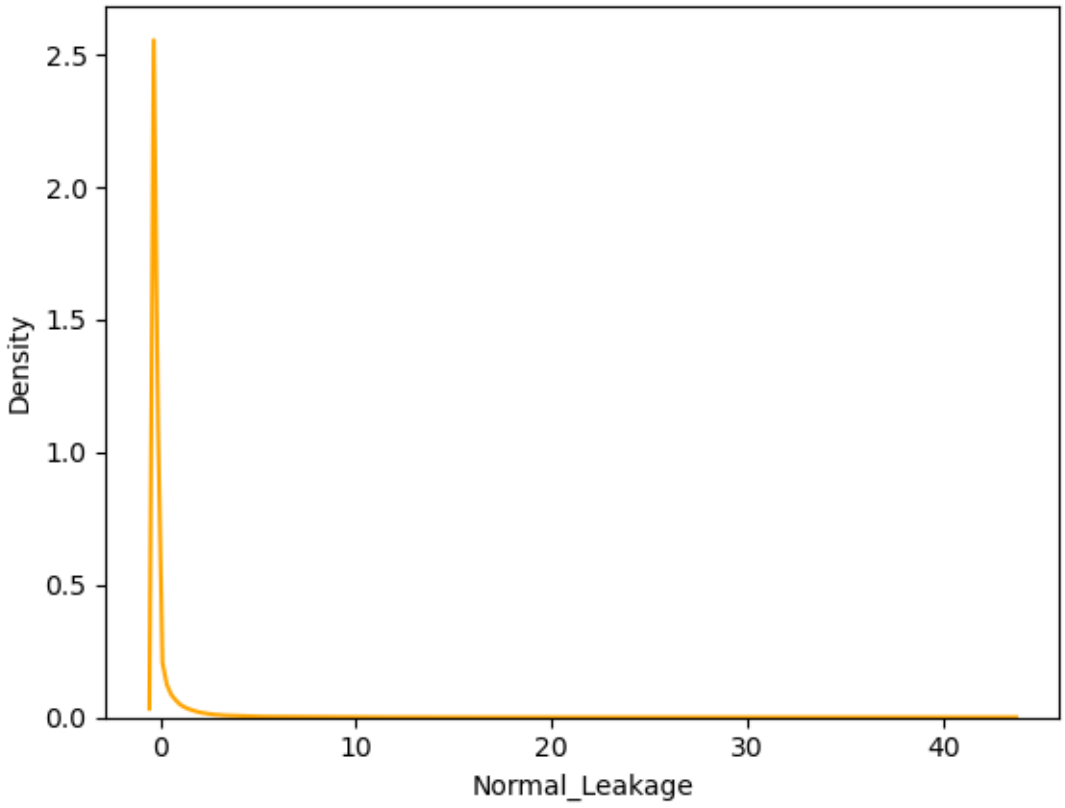
[45]:
```
 #Estandarizado por log de leakage
filtro_failure = df['Failure_today'] == 'Yes'
epsilon = 1
df['Log_Leakage'] = np.log(df['Leakage'] + epsilon)
leak_failure = pd.DataFrame(df['Log_Leakage'][filtro_failure])


sns.kdeplot(data=df, x='Log_Leakage', color='orange')
```

[45]:
```
<Axes: xlabel='Log_Leakage', ylabel='Density'>
```



[46]:
```
#Estandarizado comun de leakage
media = (df['Leakage']+epsilon).mean()
varianza = (df['Leakage']+epsilon).var() #En varianza no deberia afectar en␣
 ↪nada pero por si acaso

df['Normal_Leakage'] = (df['Leakage']+epsilon - media)/np.sqrt(varianza)

#print('Sinceramente comparando ambos metodos de estandarizacion, me parece una␣
 ↪mejor opcion la logaritima con un epsilon bajo')
sns.kdeplot(data=df, x='Normal_Leakage', color='orange')
```

[46]:
```
<Axes: xlabel='Normal_Leakage', ylabel='Density'>
```



Debido a su alta relación con la variable dependiente se decidio omitir las filtraciones a los modelos.

- Se analizó la proporcion de los registros faltantes por cada variable para así escoger los parametros mas completos que correalaciones lo suficiente para reemplazar estas variables.
  - Electricidad: 47%
  - Evaporacion: 42%
  - Parametro 6: 37%
- Se crearon variables categoricas para las variables de direccion (Parameter1_Dir, Parameter2_9am y 3pm) separandolas por direccion del viento (norte, sur, este oeste). ademas de categorizar los meses en estaciones del año. Como resultado se obtuvo que la direccion del viento no fue significativa para nuestro modelo

[47]:
```
#categorizar direcciones del viento
df['Is_P1Dir_North'] = df['Parameter1_Dir'].map({'NNW':1, 'N':1,'NNE':1,'NE':1,
                                                 'ENE':0,'E':0,'ESE':0,'SE':0,
                                                 'SSE':0,'S':0,'SSW':0,'SW':0,
                                                 'WSW':0,'W':0, 'WNW':0,'NW':0})

#categorizar direcciones del viento
df['Is_P1Dir_West'] = df['Parameter1_Dir'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
```

```
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':1,'W':1, 'WNW':1,'NW':1})

#categorizar direcciones del viento
df['Is_P1Dir_South'] = df['Parameter1_Dir'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
                                                'SSE':1,'S':1,'SSW':1,'SW':1,
                                                'WSW':0,'W':0, 'WNW':0,'NW':0})

df['Is_P1Dir_East'] = df['Parameter1_Dir'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':1,'E':1,'ESE':1,'SE':1,
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':0,'W':0, 'WNW':0,'NW':0})

#Cargorizar direcciones del parametro 2 de 9am
df['Is_P29am_North'] = df['Parameter2_9am'].map({'NNW':1, 'N':1,'NNE':1,'NE':1,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':0,'W':0, 'WNW':0,'NW':0})

df['Is_P29am_West'] = df['Parameter2_9am'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':1,'W':1, 'WNW':1,'NW':1})

df['Is_P29am_South'] = df['Parameter2_9am'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
                                                'SSE':1,'S':1,'SSW':1,'SW':1,
                                                'WSW':0,'W':0, 'WNW':0,'NW':0})

df['Is_P29am_East'] = df['Parameter2_9am'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':1,'E':1,'ESE':1,'SE':1,
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':0,'W':0, 'WNW':0,'NW':0})

#Cargorizar direcciones del parametro 2 de 3pm
df['Is_P23pm_North'] = df['Parameter2_3pm'].map({'NNW':1, 'N':1,'NNE':1,'NE':1,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':0,'W':0, 'WNW':0,'NW':0})

df['Is_P23pm_West'] = df['Parameter2_3pm'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                'ENE':0,'E':0,'ESE':0,'SE':0,
                                                'SSE':0,'S':0,'SSW':0,'SW':0,
                                                'WSW':1,'W':1, 'WNW':1,'NW':1})

df['Is_P23pm_South'] = df['Parameter2_3pm'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
```

```
                                                    'ENE':0,'E':0,'ESE':0,'SE':0,
                                                    'SSE':1,'S':1,'SSW':1,'SW':1,
                                                    'WSW':0,'W':0, 'WNW':0,'NW':0})

df['Is_P23pm_East'] = df['Parameter2_3pm'].map({'NNW':0, 'N':0,'NNE':0,'NE':0,
                                                    'ENE':1,'E':1,'ESE':1,'SE':1,
                                                    'SSE':0,'S':0,'SSW':0,'SW':0,
                                                    'WSW':0,'W':0, 'WNW':0,'NW':0})
```

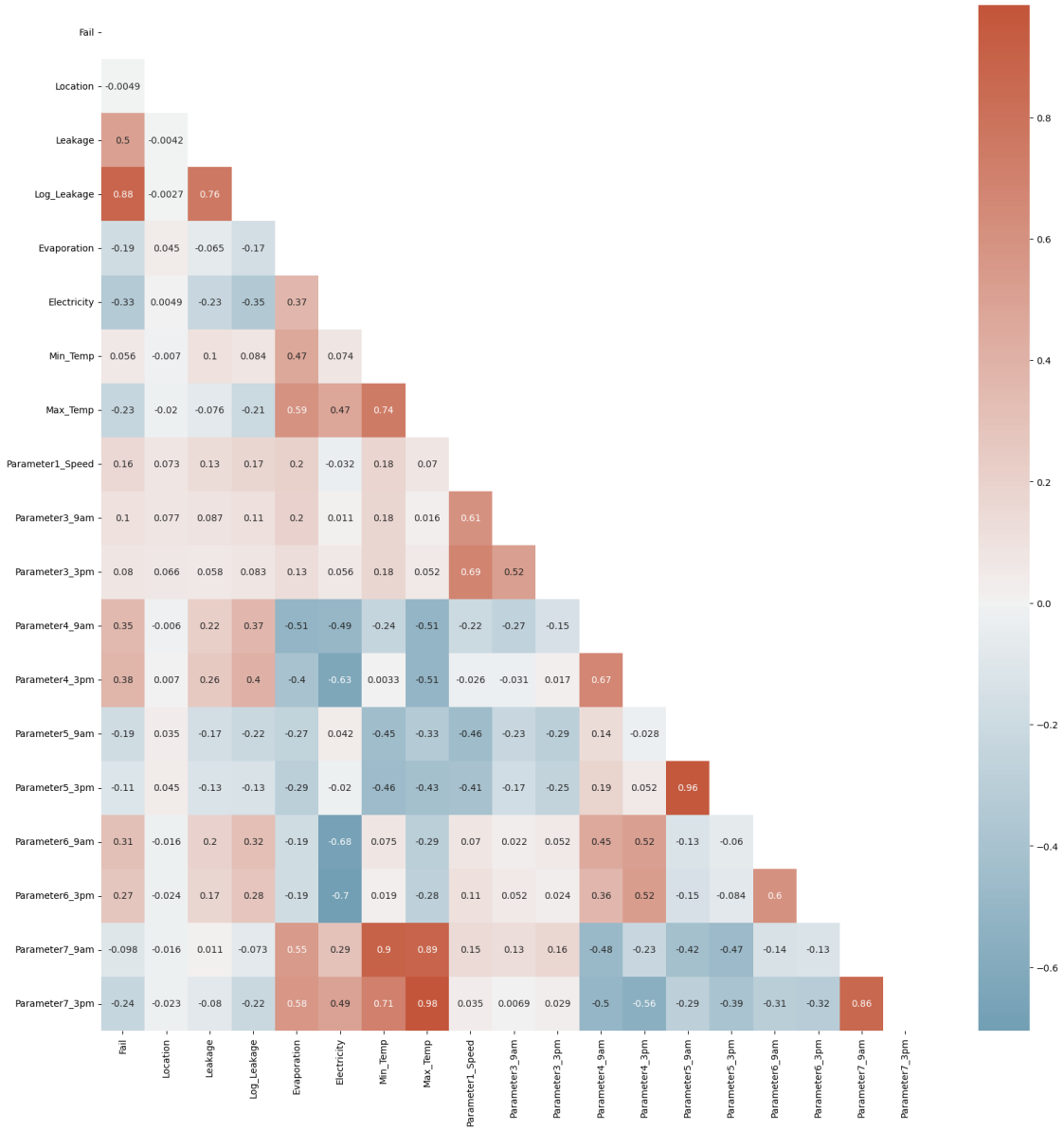
[48]:
```
#Categorizar meses en estaciones
# inv: diciembre-marzo, primavera: marzo-junio, verano: junio-septiembre, otoño:
 ↪ septiembre-diciembre
df['Is_Summer'] = df['Month'].map({ 12:0, 1:0, 2:0,
                                    3:0, 4:0, 5:0,
                                    6:1, 7:1, 8:1,
                                    9:0, 10:0, 11:0})

df['Is_Winter'] = df['Month'].map({12:1, 1:1, 2:1,
                                    3:0, 4:0, 5:0,
                                    6:0, 7:0, 8:0,
                                    9:0, 10:0, 11:0})

df['Is_Fall'] = df['Month'].map({12:0, 1:0, 2:0,
                                    3:0, 4:0, 5:0,
                                    6:0, 7:0, 8:0,
                                    9:1, 10:1, 11:1})

df['Is_Spring'] = df['Month'].map({12:0, 1:0, 2:0,
                                    3:1, 4:1, 5:1,
                                    6:0, 7:0, 8:0,
                                    9:0, 10:0, 11:0})
```

- Se realizó una martiz de calor presentando la correlacion entre las variables para eliminar las que posean alta correlación entre si. además de buscar reemplazos para las que posean data faltante

[49]:
```
#Hay que convertir en binaria las variable de Failure, Yes = 1 No = 0 y␣
 ↪eliminar las NaN
#Hacer matriz de covarianza para descartar algunas que se describen con otras

df = df[df['Failure_today'].notna()]
df['Fail'] = df['Failure_today'].apply(lambda x: 1 if x == 'Yes' else 0)


DF = df[['Fail','Location','Leakage','Log_Leakage', 'Evaporation',␣
 ↪'Electricity','Min_Temp', 'Max_Temp','Parameter1_Speed',
```

```
␣
↪'Parameter3_9am','Parameter3_3pm','Parameter4_9am','Parameter4_3pm','Parameter5_9am','Param
        'Parameter6_9am','Parameter6_3pm','Parameter7_9am','Parameter7_3pm']]
corr = DF.corr()

mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(20, 20))
cmap = sns.diverging_palette(230, 20, as_cmap=True)
#sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.6, center=0,
#            square=True, linewidths=.5, cbar_kws={"shrink": .5})
sns.heatmap(corr, annot=True, cmap=cmap, center=0,mask=mask, )
```

[49]: <Axes: >



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

Variable dependiente: Fail (Sí falla la maquina)

se seleccionaron las siguientes variables para los modelos:

- Location
- Min_Temp
- Max_Temp
- Parameter1_Speed
- Parameter3_9am
- Parameter3_3pm
- Parameter4_9am
- Parameter4_3pm
- Parameter5_9am
- Is_Summer
- Is_Fall
- Is_Spring

Para las estaciones se escogió la estacion de invierno como la variable categorica base para evaluar como varia el estar en otra estacion en comparacion a estar en invierno.

[50]:
```
#Regresion excluyendo variables de alta correlacion




funcion = df.drop(['Date','Evaporation','Electricity',
                   'Parameter1_Dir','Parameter2_9am','Parameter2_3pm',␣
 ↪'Is_P1Dir_West', 'Is_P1Dir_South', 'Is_P1Dir_East', 'Is_P1Dir_North',
                   'Is_P29am_North', 'Is_P29am_West', 'Is_P29am_South',␣
 ↪'Is_P29am_East',
```

```
                    'Is_P23pm_North', 'Is_P23pm_West', 'Is_P23pm_South',␣
 ↪'Is_P23pm_East',
                    'Failure_today', 'Year', 'Month', 'Is_Winter',
                     'Parameter5_3pm',␣
 ↪'Parameter6_9am','Parameter6_3pm','Parameter7_9am','Parameter7_3pm',␣
 ↪'Normal_Leakage','Leakage','Log_Leakage'], axis=1)
#funcion = funcion[~funcion.isna().any(axis=1)]
funcion.dropna(inplace=True)



y=funcion['Fail']
X=funcion.drop(['Fail'], axis=1)

X=sm.add_constant(X)
```

[51]:
```
model = sm.OLS(y, X)
results = model.fit(cov_type='HC0')
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Fail   R-squared:                       0.263
Model:                            OLS   Adj. R-squared:                  0.262
Method:                 Least Squares   F-statistic:                     3561.
Date:                Thu, 24 Apr 2025   Prob (F-statistic):               0.00
Time:                        22:20:11   Log-Likelihood:                -45616.
No. Observations:              117829   AIC:                         9.126e+04
Df Residuals:                  117816   BIC:                         9.138e+04
Df Model:                          12
Covariance Type:                  HC0
==============================================================================
====
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----
const                9.6130      0.209     46.040      0.000       9.204
10.022
Location            -0.0005    7.1e-05     -7.620      0.000      -0.001
-0.000
Min_Temp             0.0191      0.000     57.491      0.000       0.018
0.020
Max_Temp            -0.0203      0.000    -56.655      0.000      -0.021
-0.020
Parameter1_Speed     0.0046      0.000     33.821      0.000       0.004
0.005
Parameter3_9am       0.0028      0.000     17.176      0.000       0.002
```

```
0.003
Parameter3_3pm      -0.0042      0.000    -24.478      0.000      -0.005
-0.004
Parameter4_9am       0.0068   8.22e-05     83.058      0.000       0.007
0.007
Parameter4_3pm     1.24e-05   9.79e-05      0.127      0.899      -0.000
0.000
Parameter5_9am      -0.0096      0.000    -47.537      0.000      -0.010
-0.009
Is_Summer            0.0082      0.004      2.193      0.028       0.001
0.015
Is_Fall              0.0417      0.003     13.181      0.000       0.035
0.048
Is_Spring            0.0038      0.003      1.218      0.223      -0.002
0.010
==============================================================================
Omnibus:                    10236.244   Durbin-Watson:                   1.730
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            12810.160
Skew:                           0.797   Prob(JB):                         0.00
Kurtosis:                       2.736   Cond. No.                     1.90e+05
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)
[2] The condition number is large, 1.9e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

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.

Para el modelo de Probit y Logit se mantuvieron las mismas variables

[52]:
```
model = sm.Probit(y, X)
probit_model = model.fit()
print(probit_model.summary())

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

```
Optimization terminated successfully.
         Current function value: 0.368592
         Iterations 7
                          Probit Regression Results
==============================================================================
Dep. Variable:                   Fail   No. Observations:               117829
Model:                         Probit   Df Residuals:                   117816
Method:                           MLE   Df Model:                           12
Date:                Thu, 24 Apr 2025   Pseudo R-squ.:                  0.3023
Time:                        22:20:12   Log-Likelihood:                -43431.
```

```
converged:                       True   LL-Null:                       -62246.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
====
                       coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
----
const                29.9610      0.851     35.200      0.000      28.293
31.629
Location             -0.0018      0.000     -5.268      0.000      -0.002
-0.001
Min_Temp              0.1213      0.002     66.659      0.000       0.118
0.125
Max_Temp             -0.1273      0.002    -67.722      0.000      -0.131
-0.124
Parameter1_Speed      0.0166      0.001     29.551      0.000       0.015
0.018
Parameter3_9am        0.0087      0.001     11.485      0.000       0.007
0.010
Parameter3_3pm       -0.0131      0.001    -16.744      0.000      -0.015
-0.012
Parameter4_9am        0.0371      0.000     84.643      0.000       0.036
0.038
Parameter4_3pm       -0.0042      0.000    -10.174      0.000      -0.005
-0.003
Parameter5_9am       -0.0319      0.001    -38.561      0.000      -0.033
-0.030
Is_Summer            -0.0758      0.017     -4.329      0.000      -0.110
-0.041
Is_Fall               0.1549      0.016      9.945      0.000       0.124
0.185
Is_Spring            -0.0415      0.015     -2.788      0.005      -0.071
-0.012
================================================================================
====
       Probit Marginal Effects
=====================================
Dep. Variable:                   Fail
Method:                          dydx
At:                           overall
================================================================================
====
                      dy/dx    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Location             -0.0004   7.08e-05     -5.270      0.000      -0.001
```

```
-0.000
Min_Temp             0.0251      0.000     70.021      0.000       0.024
0.026
Max_Temp            -0.0263      0.000    -71.451      0.000      -0.027
-0.026
Parameter1_Speed     0.0034      0.000     29.899      0.000       0.003
0.004
Parameter3_9am       0.0018      0.000     11.508      0.000       0.001
0.002
Parameter3_3pm      -0.0027      0.000    -16.817      0.000      -0.003
-0.002
Parameter4_9am       0.0077    8.3e-05     92.272      0.000       0.007
0.008
Parameter4_3pm      -0.0009   8.62e-05    -10.183      0.000      -0.001
-0.001
Parameter5_9am      -0.0066      0.000    -39.372      0.000      -0.007
-0.006
Is_Summer           -0.0156      0.004     -4.331      0.000      -0.023
-0.009
Is_Fall              0.0320      0.003      9.955      0.000       0.026
0.038
Is_Spring           -0.0086      0.003     -2.789      0.005      -0.015
-0.003
==============================================================================
====
```

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.

[53]:
```
#Los modelos logit y probit tienen cte muuy grandes porque a mas grande la␣
 ↪ecuacion del logit// dara 0
#Osea esta forzando que de 0

model = sm.Logit(y, X)
logit_model = model.fit()
print(logit_model.summary())

mfx = logit_model.get_margeff()
print(mfx.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.368095
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                   Fail   No. Observations:               117829
Model:                          Logit   Df Residuals:                   117816
Method:                           MLE   Df Model:                           12
```

```
Date:                Thu, 24 Apr 2025   Pseudo R-squ.:                  0.3032
Time:                        22:20:13   Log-Likelihood:                -43372.
converged:                       True   LL-Null:                       -62246.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
====
                       coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----
const               50.6764      1.494     33.923      0.000      47.748
53.604
Location            -0.0032      0.001     -5.333      0.000      -0.004
-0.002
Min_Temp             0.2200      0.003     66.869      0.000       0.214
0.226
Max_Temp            -0.2281      0.003    -66.851      0.000      -0.235
-0.221
Parameter1_Speed     0.0287      0.001     29.155      0.000       0.027
0.031
Parameter3_9am       0.0145      0.001     10.882      0.000       0.012
0.017
Parameter3_3pm      -0.0215      0.001    -15.551      0.000      -0.024
-0.019
Parameter4_9am       0.0671      0.001     84.002      0.000       0.066
0.069
Parameter4_3pm      -0.0080      0.001    -10.971      0.000      -0.009
-0.007
Parameter5_9am      -0.0542      0.001    -37.311      0.000      -0.057
-0.051
Is_Summer           -0.0813      0.031     -2.628      0.009      -0.142
-0.021
Is_Fall              0.2849      0.028     10.205      0.000       0.230
0.340
Is_Spring           -0.0315      0.026     -1.195      0.232      -0.083
0.020
==============================================================================
====
        Logit Marginal Effects
=====================================
Dep. Variable:                   Fail
Method:                          dydx
At:                           overall
==============================================================================
====
                      dy/dx    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
```

```
----
Location            -0.0004   7.09e-05     -5.335      0.000      -0.001
-0.000
Min_Temp             0.0257      0.000     71.210      0.000       0.025
0.026
Max_Temp            -0.0266      0.000    -71.271      0.000      -0.027
-0.026
Parameter1_Speed     0.0033      0.000     29.596      0.000       0.003
0.004
Parameter3_9am       0.0017      0.000     10.908      0.000       0.001
0.002
Parameter3_3pm      -0.0025      0.000    -15.627      0.000      -0.003
-0.002
Parameter4_9am       0.0078   8.29e-05     94.475      0.000       0.008
0.008
Parameter4_3pm      -0.0009   8.54e-05    -10.986      0.000      -0.001
-0.001
Parameter5_9am      -0.0063      0.000    -38.231      0.000      -0.007
-0.006
Is_Summer           -0.0095      0.004     -2.628      0.009      -0.017
-0.002
Is_Fall              0.0333      0.003     10.221      0.000       0.027
0.040
Is_Spring           -0.0037      0.003     -1.195      0.232      -0.010
0.002
==============================================================================
====
```

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 investgación y por qué? ¿Qué variables resultaron ser robustas a la especificación?

R: Segun los resultados obtenidos en los modelos de minimos cuadrados, probit y logit podemos concluir a travez de el valor de R cuadrado los modelos probit y logit explican mas la variable dependiente que minimos cuadrados (aun asi es un valor bastante bajo), eso significa que es mejor usar los ultimos dos modelos para explicar las variables no lineales. Aun asi presentamos dificultades en la estimacion debido a falta de variables mas representativas (se explico previamente porqué no se uso "Leakage").

Para los resultados obtenidos entre probit y logit comparamos los coeficientes marginales y no descubrimos diferencia significativa entre las variables para encontrar un modelo mejor que el otro, podriamos decir que para las variables escogidas es irrelevante cual modelo se usa.

Variables robustas a la especificacion:

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.

Estudiando la data en excel, a travez de tablas dinamicas se pudo ver facilmmente como hay localizaciones los cuales jamas en los 10 años de estudio se midieron el consumo electrico o la evaporacion, eso explica los datos faltantes en esas columnas (al igual que en el parammetro 6). Por lo que se presentan casi la mitad de la data faltante en estas variables por lo que otra vez no se considerarán para los estudios siguientes.

Para correr el codigo se agruparon las localizaciones en 15 variables dummies.

[54]:
```
df_mensual = df.drop(['Date','Failure_today',
                      'Parameter1_Dir','Parameter2_9am','Parameter2_3pm',␣
 ↪'Is_P1Dir_West', 'Is_P1Dir_South', 'Is_P1Dir_East', 'Is_P1Dir_North',
                      'Is_P29am_North', 'Is_P29am_West', 'Is_P29am_South',␣
 ↪'Is_P29am_East',
                      'Is_P23pm_North', 'Is_P23pm_West', 'Is_P23pm_South',␣
 ↪'Is_P23pm_East',

                       ], axis=1)


df_mensual.dropna(inplace=True)


# Creamos un diccionario con funciones agregadas
agg_dict = {col: 'mean' for col in df_mensual.columns if col not in ['Year',␣
 ↪'Month', 'Location', 'Fail']}
agg_dict['Fail'] = 'sum'

# Aplicamos el groupby con agregación personalizada
df_mensual = df_mensual.groupby(['Year', 'Month', 'Location']).agg(agg_dict).
 ↪reset_index()

#df_mensual['Has_Failed'] = df_mensual['Fail'].apply(lambda x: 1 if x != 0.0␣
 ↪else 0)
#df_mensual
```

0.0.1 Poisson

[56]:
```
df_mensual['Location_group'] = pd.cut(df_mensual['Location'], bins=15,␣
 ↪labels=False)
```

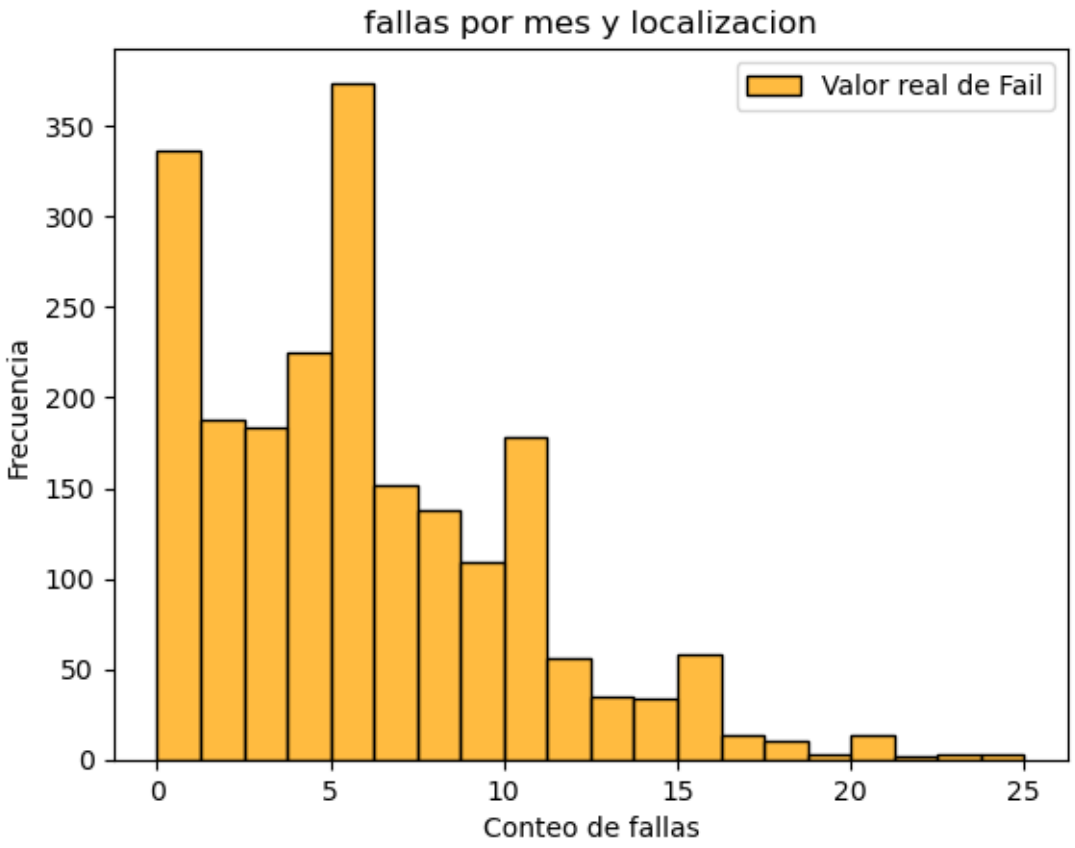
[57]:
```
poisson = smf.poisson("Fail ~ C(Location_group) + Min_Temp + Max_Temp +␣
 ↪Parameter1_Speed + Parameter3_9am + Parameter3_3pm + Parameter4_9am +␣
 ↪Parameter4_3pm + Parameter5_9am + Is_Summer + Is_Fall + Is_Spring +␣
 ↪C(Year)", data=df_mensual).fit()
print(poisson.summary())
```

```
Optimization terminated successfully.
         Current function value: 2.272270
```

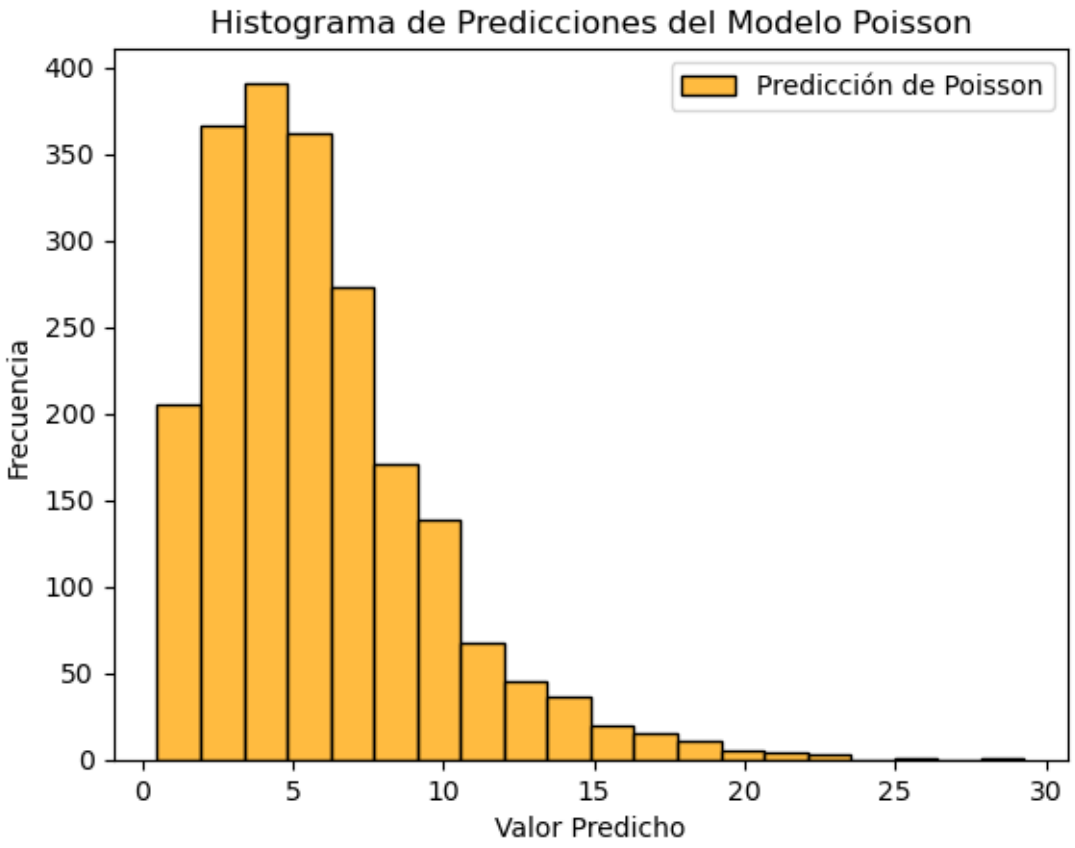
```
         Iterations 8
                          Poisson Regression Results
==============================================================================
Dep. Variable:                   Fail   No. Observations:                 2116
Model:                        Poisson   Df Residuals:                     2082
Method:                           MLE   Df Model:                           33
Date:                Thu, 24 Apr 2025   Pseudo R-squ.:                  0.3201
Time:                        22:20:14   Log-Likelihood:                -4808.1
converged:                       True   LL-Null:                       -7071.7
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
===========
                               coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
Intercept                   20.8299      4.284      4.862      0.000      12.434
29.226
C(Location_group)[T.1]     -0.2142      0.084     -2.539      0.011      -0.380
-0.049
C(Location_group)[T.2]     -0.1675      0.092     -1.813      0.070      -0.349
0.014
C(Location_group)[T.3]     -0.5213      0.088     -5.953      0.000      -0.693
-0.350
C(Location_group)[T.4]     -0.4167      0.094     -4.423      0.000      -0.601
-0.232
C(Location_group)[T.5]     -0.2452      0.087     -2.825      0.005      -0.415
-0.075
C(Location_group)[T.6]      0.0527      0.092      0.573      0.567      -0.128
0.233
C(Location_group)[T.7]     -0.5013      0.097     -5.177      0.000      -0.691
-0.312
C(Location_group)[T.8]     -0.2984      0.095     -3.151      0.002      -0.484
-0.113
C(Location_group)[T.9]     -0.1622      0.084     -1.931      0.053      -0.327
0.002
C(Location_group)[T.10]    -0.2098      0.100     -2.099      0.036      -0.406
-0.014
C(Location_group)[T.11]    -0.2466      0.087     -2.849      0.004      -0.416
-0.077
C(Location_group)[T.12]    -0.0004      0.094     -0.005      0.996      -0.185
0.185
C(Location_group)[T.13]    -0.5173      0.089     -5.824      0.000      -0.691
-0.343
C(Location_group)[T.14]    -0.8197      0.120     -6.809      0.000      -1.056
-0.584
C(Year)[T.2010]             0.0622      0.034      1.842      0.066      -0.004
0.128
```

```
C(Year)[T.2011]          -0.0311      0.035     -0.879      0.379      -0.101
0.038
C(Year)[T.2012]          -0.0580      0.036     -1.606      0.108      -0.129
0.013
C(Year)[T.2013]          -0.1038      0.036     -2.866      0.004      -0.175
-0.033
C(Year)[T.2014]          -0.1561      0.037     -4.209      0.000      -0.229
-0.083
C(Year)[T.2015]          -0.0603      0.038     -1.573      0.116      -0.136
0.015
C(Year)[T.2016]          -0.0430      0.040     -1.079      0.280      -0.121
0.035
C(Year)[T.2017]          -0.1245      0.060     -2.077      0.038      -0.242
-0.007
Min_Temp                  0.0984      0.011      8.924      0.000       0.077
0.120
Max_Temp                 -0.0772      0.011     -7.012      0.000      -0.099
-0.056
Parameter1_Speed          0.0588      0.003     17.578      0.000       0.052
0.065
Parameter3_9am           -0.0145      0.005     -3.152      0.002      -0.023
-0.005
Parameter3_3pm           -0.0697      0.004    -16.150      0.000      -0.078
-0.061
Parameter4_9am            0.0242      0.002     10.485      0.000       0.020
0.029
Parameter4_3pm            0.0143      0.003      5.016      0.000       0.009
0.020
Parameter5_9am           -0.0213      0.004     -5.141      0.000      -0.029
-0.013
Is_Summer                 0.1445      0.039      3.702      0.000       0.068
0.221
Is_Fall                   0.3166      0.032      9.776      0.000       0.253
0.380
Is_Spring                 0.1414      0.032      4.486      0.000       0.080
0.203
==============================================================================
===========
```

[58]:
```
sns.histplot(df_mensual['Fail'], bins=20, kde=False, color='Orange',␣
 ↪label='Valor real de Fail')
plt.xlabel('Conteo de fallas')
plt.ylabel('Frecuencia')
plt.title('fallas por mes y localizacion')
plt.legend()
plt.show()
```



[59]:
```
predictions = poisson.predict(df_mensual)
#print(predictions)
sns.histplot(predictions, bins=20, kde=False, color='Orange', label='Predicción␣
 ↪de Poisson')
plt.xlabel('Valor Predicho')
plt.ylabel('Frecuencia')
plt.title('Histograma de Predicciones del Modelo Poisson')
plt.legend()
plt.show()
```



Segun el entrenamiento, usando poisson.predict nos presenta que su estimacion se acerca bastante poco al caso real aunque nunca presenta un valor estrictamente 0, debe ser por su caracteristica de ser un estimador. Ademas de acuerdo a los resultados del modelo se concluye lo siguiente: - Con el R cuadrado vemos que el modelo explica alrededor de un 30% de la data real - Si la maquina trabaja en las estaciones de otoño y primavera aumenta la cantidad de fallas en 0.21 y 0.11 respectivamente - En general como en Poisson las variables de los años no son significativos para el modelo - las variables que favorecerian a reducir la cantidad de fallas en un mes son: los parametros 5, 3, temperatura maxima - Muchos de los grupos de localizaciones son significativas para el modelo y con un coeficiente negativo donde indica en su mayoria los grupos de localizaciones suelen tener menos de 29 fallas.

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

R: Para estudiar la sobredispersion de la data debemos realizar el test de sobredispersion. utilizando las predicciones del modelo de poisson se obtuvo los siguientes resultados.

[60]:
```
aux=((df_mensual['Fail']-predictions)**2-predictions)/predictions
auxr=sm.OLS(aux,predictions).fit()
print(auxr.summary())
```

```
                            OLS Regression Results
```

```
==============================================================================
=======
Dep. Variable:                      y   R-squared (uncentered):
0.009
Model:                            OLS   Adj. R-squared (uncentered):
0.009
Method:                 Least Squares   F-statistic:
19.88
Date:                Thu, 24 Apr 2025   Prob (F-statistic):
8.66e-06
Time:                        22:20:14   Log-Likelihood:
-4283.3
No. Observations:                2116   AIC:
8569.
Df Residuals:                    2115   BIC:
8574.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.0255      0.006      4.459      0.000       0.014       0.037
==============================================================================
Omnibus:                     2235.596   Durbin-Watson:                   1.732
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           236706.455
Skew:                           5.019   Prob(JB):                         0.00
Kurtosis:                      53.833   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² 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.
```

[63]:
```
print('Alpha:', np.exp(0.0255))
```

```
Alpha: 1.0258279062704445
```

Calculando la exponencial del X1 del resultado obtenemos el alpha el cual nos indica que hay una sobredispersion aunque es mayor a 1 por muy poco.

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.

[61]:
```

```

```
nbin = smf.negativebinomial("Fail ~  Min_Temp + Max_Temp + Parameter1_Speed +␣
 ↪Parameter3_9am + Parameter3_3pm + Parameter4_9am + Parameter4_3pm +␣
 ↪Parameter5_9am + Is_Summer + Is_Fall + Is_Spring +␣
 ↪C(Year)+C(Location_group)", data=df_mensual).fit()
print(nbin.summary())
```

```
         Current function value: 2.264007
         Iterations: 35
         Function evaluations: 49
         Gradient evaluations: 49
                     NegativeBinomial Regression Results
==============================================================================
Dep. Variable:                   Fail   No. Observations:                 2116
Model:               NegativeBinomial   Df Residuals:                     2082
Method:                           MLE   Df Model:                           33
Date:                Thu, 24 Apr 2025   Pseudo R-squ.:                  0.1858
Time:                        22:20:14   Log-Likelihood:                -4790.6
converged:                      False   LL-Null:                       -5884.0
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
Intercept                    20.8325      4.802      4.338      0.000      11.420
30.245
C(Year)[T.2010]               0.0642      0.038      1.697      0.090      -0.010
0.138
C(Year)[T.2011]              -0.0334      0.040     -0.844      0.399      -0.111
0.044
C(Year)[T.2012]              -0.0620      0.040     -1.542      0.123      -0.141
0.017
C(Year)[T.2013]              -0.0972      0.041     -2.400      0.016      -0.177
-0.018
C(Year)[T.2014]              -0.1500      0.041     -3.636      0.000      -0.231
-0.069
C(Year)[T.2015]              -0.0621      0.043     -1.459      0.145      -0.145
0.021
C(Year)[T.2016]              -0.0409      0.044     -0.919      0.358      -0.128
0.046
C(Year)[T.2017]              -0.1270      0.066     -1.917      0.055      -0.257
0.003
C(Location_group)[T.1]       -0.2261      0.090     -2.511      0.012      -0.403
-0.050
C(Location_group)[T.2]       -0.1806      0.099     -1.823      0.068      -0.375
0.014
C(Location_group)[T.3]       -0.5543      0.094     -5.887      0.000      -0.739
```

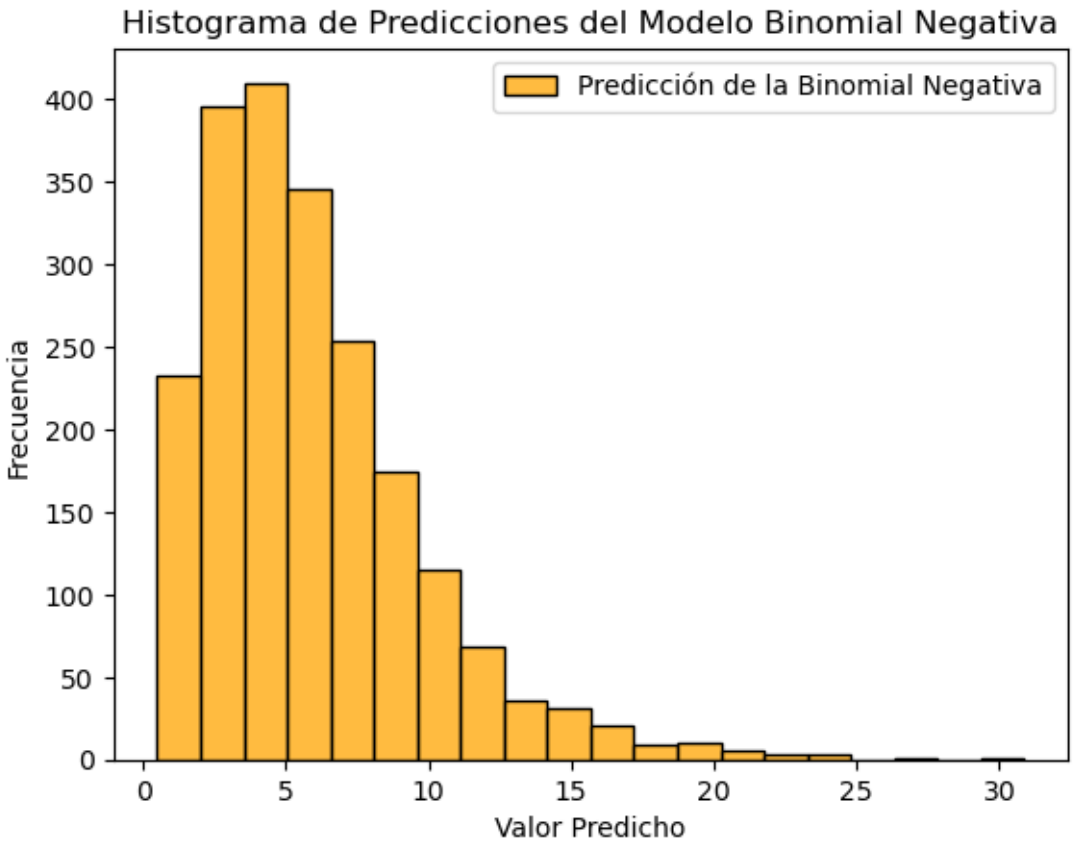
```
-0.370
C(Location_group)[T.4]     -0.4381      0.101     -4.335      0.000      -0.636
-0.240
C(Location_group)[T.5]     -0.2570      0.092     -2.780      0.005      -0.438
-0.076
C(Location_group)[T.6]      0.0410      0.100      0.411      0.681      -0.154
0.236
C(Location_group)[T.7]     -0.5306      0.105     -5.061      0.000      -0.736
-0.325
C(Location_group)[T.8]     -0.3100      0.101     -3.066      0.002      -0.508
-0.112
C(Location_group)[T.9]     -0.1672      0.089     -1.877      0.061      -0.342
0.007
C(Location_group)[T.10]    -0.2255      0.108     -2.094      0.036      -0.436
-0.014
C(Location_group)[T.11]    -0.2532      0.092     -2.741      0.006      -0.434
-0.072
C(Location_group)[T.12]    -0.0026      0.101     -0.026      0.979      -0.201
0.195
C(Location_group)[T.13]    -0.5295      0.095     -5.577      0.000      -0.716
-0.343
C(Location_group)[T.14]    -0.8255      0.126     -6.562      0.000      -1.072
-0.579
Min_Temp                    0.0999      0.012      8.110      0.000       0.076
0.124
Max_Temp                   -0.0801      0.012     -6.492      0.000      -0.104
-0.056
Parameter1_Speed            0.0602      0.004     15.956      0.000       0.053
0.068
Parameter3_9am             -0.0134      0.005     -2.614      0.009      -0.023
-0.003
Parameter3_3pm             -0.0719      0.005    -14.997      0.000      -0.081
-0.063
Parameter4_9am              0.0246      0.003      9.625      0.000       0.020
0.030
Parameter4_3pm              0.0146      0.003      4.589      0.000       0.008
0.021
Parameter5_9am             -0.0213      0.005     -4.587      0.000      -0.030
-0.012
Is_Summer                   0.1237      0.044      2.806      0.005       0.037
0.210
Is_Fall                     0.3200      0.036      8.882      0.000       0.249
0.391
Is_Spring                   0.1357      0.035      3.826      0.000       0.066
0.205
alpha                       0.0318      0.006      5.033      0.000       0.019
0.044
==============================================================================
```

==========

De acuerdo a los resultados entregados podemos destacar lo siguiente. - Segun el R cuadrado el modelo explica un 18% de la variable dependiente es un resultado bastante bajo y debe ser porque faltan variables relevantes que no se midieron

los coeficientes se mentienen similares a los del modelo de poisson y el histograma se mantiene similar al poisson

[62]:
```
predictions=nbin.predict(df_mensual)

sns.histplot(predictions, bins=20, kde=False, color='Orange', label='Predicción␣
 ↪de la Binomial Negativa')
plt.xlabel('Valor Predicho')
plt.ylabel('Frecuencia')
plt.title('Histograma de Predicciones del Modelo Binomial Negativa')
plt.legend()
plt.show()
```



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 investgación y por qué? ¿Qué variables resultaron ser robustas a la especificación?

R: Segun lo analizado no se encuentra mucha diferencia entre los modelos poisson y binomial, aunque el primero al parecer explica mejor la data, aunque haya sobredispersion no parece ser demasiada como para ser extrictamente necesario usar el modelo de la Binomial negativa ya que su alpha es cercano a 0 (aunque la varianza es muy grande). Se deberan analizar mas parametros no estudiados para estimar mejor la data ya que como se vio previamente en el mapa de calor muchos de las variables independientes no poseen una alta correlacion con la dependiente (excepto las filtraciones).