MINIPROYECTO I

Jose Manuel Gomez Garcia (2190257), Sergio Andrés León Zambrano (2181607), Joshep Peralta Carvajal (2200046).
jose_manuel.gomez@uao.edu.co , sergio.leon@uao.edu.co , joshep.peralta@uao.edu.co.
Universidad de Autónoma de Occidente - Cali

Resumen – En el presente documento, se evidencia una serie se análisis a unos datos los cuales tratan de la estabilidad local del sistema en estrella de 4 nodos por medio de gráficos se realiza el análisis para después aplicar los métodos de regresión lineal multiparamétrica y polinómica.

I. INTRODUCCIÓN

La regresión lineal multiparamétrica y polinómica es una técnica valiosa y ampliamente utilizada en el campo de la inteligencia artificial y el análisis de datos. En este informe, exploraremos cómo esta herramienta se aplica a problemas de regresión con la ayuda del lenguaje de programación Python. La regresión lineal multiparamétrica y polinómica nos permite modelar y comprender relaciones entre variables, lo que a su vez nos permite hacer predicciones útiles en una variedad de aplicaciones.

A lo largo de este informe, presentaremos conceptos clave de manera accesible y proporcionaremos ejemplos concretos para ilustrar cómo estas técnicas pueden ser beneficiosas en la toma de decisiones y la resolución de problemas en el ámbito de la inteligencia artificial. Exploraremos su aplicación en escenarios del mundo real y destacaremos su relevancia en la actualidad.

II. METODOLOGIA

Como trabajaremos con un dataset debemos conocer el tema que se escogió, las características o variables que allí se encuentran como también una breve información de lo que se planteo en su momento con los datos recolectados.

Datos simulados de estabilidad de la red eléctrica:

El análisis de estabilidad local del sistema en estrella de 4 nodos (el productor de electricidad está en el centro) implementando el concepto de control descentralizado de red inteligente.

Características del conjunto de datos	multivariado
Área temática	Ciencia física
# Instancias	10.000
Tareas asociadas	Clasificación, Regresión
# Atributos	14

Información de los atributos:

11 atributos predictivos, 1 no predictivo (p1), 2 campos de objetivos.

1. tau[x]: tiempo de reacción del participante (real desde el rango [0,5,10]s).

   Tau1: el valor para el productor de electricidad.

2. p[x]: potencia nominal consumida(negativa)/producida(positiva)(real).

   Para consumidores del rango [-0,5,-2]s^-2; p1 = abs (p2 + p3 + p4)

3. g[x]: coeficiente (gamma) proporcional a la elasticidad precio (real del rango [0.05,1]s^-1).

   g1 - el valor para el productor de electricidad.

4. stab: la parte real máxima de la raíz de la ecuación característica (si es positiva, el sistema es linealmente inestable) (real).
5. stabf: la etiqueta de estabilidad del sistema (categórica: estable/inestable).

Para descargar el comprimido del dataset, solo hacer clic en el siguiente enlace: Electrical Grid Stability Simulated Data

Después de conocer el tema se debe seguir los siguientes pasos:

AED:

1- Análisis del problema
2- Identificar relaciones
3- Particionar el dataset
4- Normalizar el dataset

Entrenamiento del modelo:

1- Entrenar los modelos que se desean implementar

Validación del modelo:

1- Por medio de las métricas determine cual fue el mejor modelo

Cabe resaltar que el lenguaje que usaremos será Python, usando la herramienta Google Colab y en el siguiente enlace encontraras el código desarrollado y las gráficas obtenidas: Archivo Google Colab Semestre 2023-03

III. DESARROLLO

En primer lugar, debemos cargar el dataset que se encuentra en la pagina web, para ello debemos utilizar las siguientes líneas de código donde obtenemos el zip, descomprimimos y luego eliminamos el zip para no generar conflictos durante el llamado del dataset:

```
!wget
"https://archive.ics.uci.edu/static/public/471/electrical+grid+stability+simulated+data.zip" #descargar archivos web
!unzip
/content/electrical+grid+stability+simulated+data.zip #descomprimir
!rm
/content/electrical+grid+stability+simulated+data.zip #eliminar archivo
```

Después hacemos la lectura del archivo que esta en formato csv y se asigna un nombre para almacenarlo ahí como se muestra a continuación:

```
df =
pd.read_csv('/content/Data_for_UCI_named.csv', encoding= "ISO-8859-1") #ISO-8859-1 codificaciòn de caracteres
```

De manera opcional se puede obtener información estadística de los datos y general del dataset para contextualizar parámetros no conocidos.

```
df.describe(include='all') #
informaciòn estadistica y general del datset
```

	tau1	tau2	tau3	tau4
count	10000.000000	10000.000000	10000.000000	10000.000000
unique	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN
mean	5.250000	5.250001	5.250004	5.249997
std	2.742548	2.742549	2.742549	2.742556
min	0.500793	0.500141	0.500788	0.500473
25%	2.874892	2.875140	2.875522	2.874950
50%	5.250004	5.249981	5.249979	5.249734
75%	7.624690	7.624893	7.624948	7.624838
max	9.999469	9.999837	9.999450	9.999443

Luego debemos rectificar que no existan valores NAN en nuestra tabla verificando cada característica y su fila:

```
df.isna().sum()#cantidad de valores NAN (Not a Numbe)
```

En nuestro caso no se encontró ninguna característica con valores no definidos o inexistentes. Entonces la última validación que debemos realizar es que no hallan valores duplicados ya que en muchos casos esto afecta al realizar el análisis de datos.

```
df.duplicated() #Cantidad de valores duplicados
```

Ahora verificamos los tipos de datos que tenemos por medio de la siguiente línea de código:

```
df.info()#infomración del dataset
```

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   tau1    10000 non-null  float64
 1   tau2    10000 non-null  float64
 2   tau3    10000 non-null  float64
 3   tau4    10000 non-null  float64
 4   p1      10000 non-null  float64
 5   p2      10000 non-null  float64
 6   p3      10000 non-null  float64
 7   p4      10000 non-null  float64
 8   g1      10000 non-null  float64
 9   g2      10000 non-null  float64
 10  g3      10000 non-null  float64
 11  g4      10000 non-null  float64
 12  stab    10000 non-null  float64
 13  stabf   10000 non-null  object
dtypes: float64(13), object(1)
```

Para poder realizar algunos gráficos separamos las variables numéricas y categóricas, en nuestro caso solo son 13 variables numéricas y una categórica.

```
numeric_features=
df.select_dtypes(exclude='object') #
Creamos un subdataset con los datos
numericos
categorical_features=df.select_dtypes(i
nclude='object')# Creamos un subdataset
con los datos categoricos
```

Lo primero que debemos ilustrar o mapear es la matriz de comparación entre variables la cual se muestra a continuación, la cual nos ayuda a profundizar un análisis entre cada variable con la Y deseada que en nuestro caso es la variable llamada stab, cuando el valor es positivo la red eléctrica es inestable, para el caso contrario si es negativo la red es estable.

La variable llamada stabf no se tiene en cuenta ya que solo tiene dos estados que son estable e inestable donde se le puede asignar un valor binario 1 o 0 por lo cual ya sería un problema de clasificación el cual no nos interesa por el momento.

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g1	g2	g3	g4	stab	stabf
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.650456	0.859578	0.887445	0.958034	0.055347	unstable
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.413441	0.862414	0.562139	0.781760	-0.005957	stable
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.163041	0.766689	0.839444	0.109853	0.003471	unstable
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.446209	0.976744	0.929381	0.362718	0.028871	unstable
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.797110	0.455450	0.656947	0.820923	0.049860	unstable
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	2.930406	9.487627	2.376523	6.187797	3.343416	-0.658054	-1.449106	-1.236256	0.601709	0.779642	0.813512	0.608385	0.023892	unstable
9996	3.392299	1.274827	2.954947	6.894759	4.349512	-1.663661	-0.952437	-1.733414	0.502079	0.567242	0.285880	0.366120	-0.025803	stable
9997	2.364034	2.842030	8.776391	1.008906	4.299976	-1.380719	-0.943884	-1.975373	0.487838	0.986505	0.149286	0.145984	-0.031810	stable
9998	9.631511	3.994398	2.757071	7.821347	2.514755	-0.966330	-0.649915	-0.898510	0.365246	0.587558	0.889118	0.818391	0.037789	unstable
9999	6.530527	6.781790	4.349695	8.673138	3.492807	-1.390285	-1.532193	-0.570329	0.073056	0.505441	0.378761	0.942631	0.045263	unstable

Fig. 1 Tabla extraída de la base de datos.

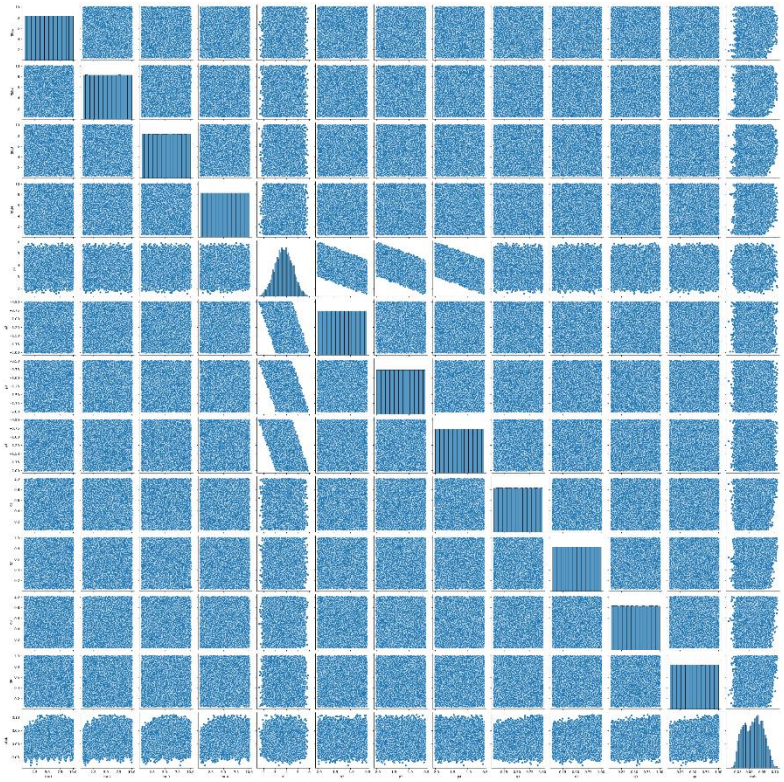

Fig. 2 Matriz de correlación Tau(x), P(x) y g(x).

```
#Informacion Completa para
Interpretacion Inicial
sns.pairplot(df)

corr = df.corr()
print(corr)

plt.figure(figsize=(15,10))
sns.heatmap(corr, annot=True)
plt.show()
```

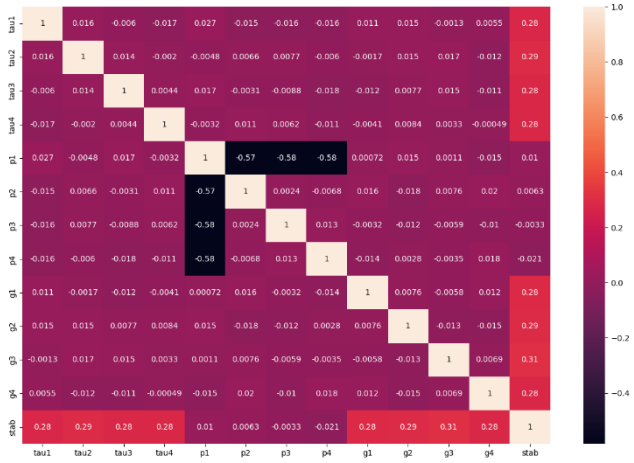

Fig. 3 Mapa de calor

Ahora a partir del mapa de calor y de la matriz, podemos apreciar que las variables que nos interesan son tau[x] y g[x] ya que tienen un valor entre 0.28 y 0.31 con relación a la variable stab, esto se puede presentar ya que en la ecuación matemática donde se calcula el valor del stab integra dichas variables pero el factor de potencia no, cabe resaltar que el valor de correlación es muy bajo pero a comparación de las tras es el más significativo y que es necesario emplear, también se observa que p1 tiene correlación con p2,p3 y p4 pero esto solo se debe a que este parámetro es la suma en magnitud de las demás.

Después como tenemos una variable categórica donde la respuesta es binaria (estable, inestable), por medio de un diagrama de torta revisamos la estabilidad de la red eléctrica:

```
# diagrama de torta V.Categorica
n=1
plt.figure(figsize=(20,15))
for i in categorical_features.columns:
  plt.subplot(2,2,n)
  n=n+1
  plt.pie(df[i].value_counts(),labels =
df[i].value_counts().keys().tolist(),au
topct='%.0f%%')
  plt.title(i)
  plt.tight_layout()
```

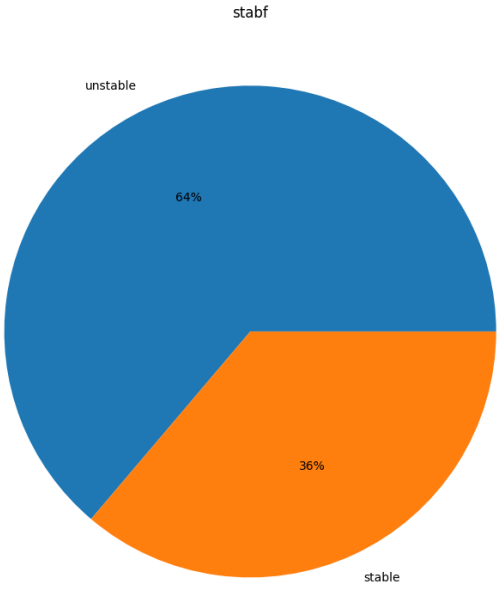

Fig. 4 Diagrama de torta para estabilidad de la red eléctrica.

Del grafico anterior se puede concluir que a partir de los datos proporcionados el sistema de red eléctrica presenta una inestabilidad del 64% por lo que presenta fallas constantes y solo tiene una estabilidad del 36%.

Ahora si queremos analizar de manera mas detallada cada una de las variables lo podemos hacer por medio de diagramas de cajas como se muestra a continuación:

```
#Diagrama de cajas para cada variable
numerica
n = 1
plt.figure(figsize=(20,15))

for i in numeric_features.columns:
  plt.subplot(4,4,n)
  n=n+1
  sns.boxplot(df[i])
  plt.title(i)
  plt.tight_layout()
```

Este tipo de gráficos nos sirve para identificar la media de cada variable, sus cuartiles y datos atípicos que se presenten, pero el caso mas necesario es cuando en una columna de datos hay valores NAN y las demás columnas tienen datos correctos, a ese valor NAN se le asigna la media para no crear incertidumbre en los demás datos.

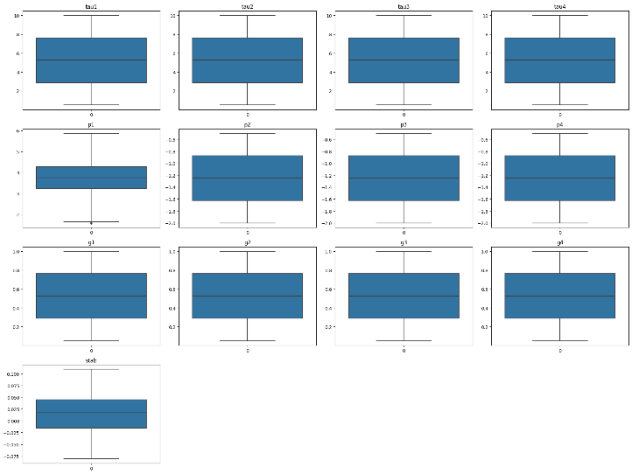

Fig. 5 Diagrama de cajas

Después de terminar estas fases de análisis, sabemos que nuestras variables de interés son:

- Tau1, Tau2, Tau3 y Tau4
- g1, g2, g3 y g4

Donde stab será nuestra salida o como se le llama comúnmente nuestra Y deseada. Por ello ahora debemos normalizar nuestro dataset como se muestra a continuación:

```
#Normalizacion de todos los datos de la tabla
def normalizar(x,xmax,xmin,ymax,ymin):
    m = (ymax-ymin)/(xmax-xmin)
    b = ymin - m*xmin
    y = m*x + b

    return y

max = np.max(df).values
min = np.min(df).values

df2 = normalizar(df,max,min,1,0)
df2.head(10)
```

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g1	g2	g3	g4	stab	stabf
0	0.258801	0.271561	0.829616	0.976978	0.509244	0.811642	0.495075	0.184572	0.632098	0.852229	0.881531	0.955894	0.715740	0.0
1	0.926793	0.463424	0.268117	0.091471	0.813957	0.039894	0.084809	0.496642	0.382588	0.855215	0.539078	0.770323	0.393359	1.0
2	0.891799	0.878795	0.268005	0.075171	0.425652	0.528367	0.481864	0.719670	0.118990	0.754440	0.831000	0.062980	0.442940	0.0
3	0.022700	0.754704	0.419623	0.193715	0.556118	0.648372	0.040670	0.668412	0.417084	0.975576	0.925677	0.329181	0.576507	0.0
4	0.277230	0.748301	0.467747	0.985065	0.453830	0.582991	0.102655	0.963811	0.786481	0.426782	0.638683	0.811553	0.686885	0.0
5	0.684139	0.906251	0.345657	0.396602	0.664921	0.095182	0.886441	0.065200	0.222947	0.029347	0.518808	0.442048	0.333266	1.0
6	0.653709	0.343702	0.676782	0.875683	0.190299	0.923668	0.527457	0.950677	0.134621	0.366277	0.370546	0.343827	0.455995	0.0
7	0.679328	0.092528	0.549405	0.775856	0.383459	0.834050	0.542331	0.473995	0.338316	0.613913	0.718672	0.347948	0.512161	0.0
8	0.441015	0.369234	0.102939	0.340386	0.574220	0.393088	0.507870	0.404810	0.231280	0.210878	0.120943	0.455217	0.221301	1.0
9	0.983369	0.096180	0.975829	0.751781	0.734501	0.005686	0.761603	0.060893	0.343549	0.520441	0.781097	0.069726	0.489807	0.0

Fig. 6 Dataset normalizado.

Una vez tenemos nuestro dataset normalizado particionamos nuestro dataset dejando un 80% para datos de entrenamiento y un 20% para validación.

```
X =
df2.drop(columns=['tau1','tau2','tau3',
'tau4','p1','p2','p3','p4','stabf','sta
b']) # Solo g(x)
#X =
df2.drop(columns=['p1','p2','p3','p4','
g1','g2','g3','g4','stabf','stab']) #
Solo tau(x)
#X =
df2.drop(columns=['p1','p2','p3','p4','
stabf','stab']) # Solo tau(x) y g(x)
y = df2['stab']
```

El código anterior elimina las columnas que no queremos analizar, es decir que en nuestro caso en la primera asignación dejamos solo las variables g[x] → 1,2,3 y 4 respectivamente eliminando todas las demás columnas, los otros dos casos que están comentados son para realizar más análisis dejando solo tau[x] y después dejar tau[x] y g[x] es decir que en este último caso analizamos 8 variables en total y en los dos primeros casos analizamos 4 para así realizar la regresión lineal multiparamétrica donde solo tenemos mínimo 4 características. En la última línea de código tenemos nuestra Y que es la característica llamada “stab”.

Después realizamos la partición del dataset donde en X quedan almacenadas aquellas variables que queremos analizar, en este caso tenemos un 20% para datos de validación si queremos cambiar dicho valor solo modificamos el test_size.

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,
test_size=0.20,random_state=42)
```

Ahora lo último que debemos realizar es la implementación del modelo por medio de Regresión lineal Multiparamétrica y polinómica:

```
#Regresión lineal Multiparametrica
from sklearn.linear_model import
LinearRegression

Linear = LinearRegression()
Linear.fit(X_train,y_train)

Score_training =
Linear.score(X_train,y_train)
print('El score para el modelo de
regresión lineal y los datos de
```

```
entrenamiento es: ' +
str(Score_training))
Score_test= Linear.score(X_test,y_test)
print('El score para el modelo de
regresión lineal y los datos de
validación es: ' + str(Score_test))
#Metricas
y_pred = Linear.predict(X_test)
```

Como se observa hacemos uso de la librería para realizar la regresión, también calculamos el score tanto para los datos de entrenamiento y validación.

```
#Regresión lineal Polinomica
from sklearn.preprocessing import
PolynomialFeatures

poly = PolynomialFeatures(2) #creating
variable with degree 2
poly_X_train =
poly.fit_transform(X_train) # fit the
train set
poly_X_test = poly.transform(X_test)
#transform the test set

linear_poly = LinearRegression()
linear_poly.fit(poly_X_train,y_train)

Score_training =
linear_poly.score(poly_X_train,y_train)
print('El score para el modelo de
regresión lineal y los datos de
entrenamiento es: ' +
str(Score_training))
Score_test=
linear_poly.score(poly_X_test,y_test)
print('El score para el modelo de
regresión lineal y los datos de
validación es: ' + str(Score_test))
#Metricas
y_pred =
linear_poly.predict(poly_X_test)
```

Del anterior código se utiliza igualmente una librería para realizar la regresión polinómica, para ambos casos se utilizan tres métricas para medir el desempeño de los modelos y se grafica la predicción y la deseada que por temas de longitud de código no se anexa de manera completa, pero en el Colab se puede visualizar de manera más detallada; Las métricas seleccionadas son MAE, MSE y RMSE. También se realiza la medición del tiempo tanto de entrenamiento como de predicción para los dos modelos implementados que estaremos explicando en la fase de análisis y resultados obtenidos.

IV. ANALISIS Y RESULTADOS

Como ya se menciono anteriormente, se realizan tres análisis con nuestras características de entrada que son tau[x] y g[x] donde se aplican los dos modelos para cada uno con el fin de mirar cual de esas características se ajustan mejor al modelo y se estime una predicción alta en cuanto a nuestra salida (stab). El ultimo análisis es cuando esas dos características las usamos como se muestra a continuación:

- Solo g[x]:

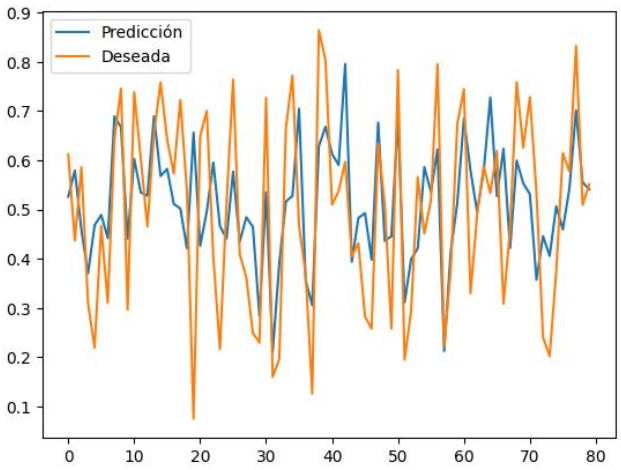

Fig. 7 Grafica regresión Multiparamétrica.

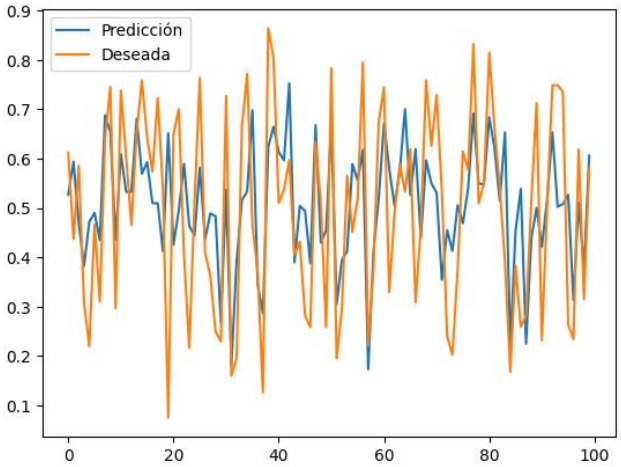

Fig. 8 Grafica regresión Polinómica.

RESUMEN	Multipara métrica	Polinómica
Score para el modelo de regresión lineal y los datos de entrenamiento	0.3439	0.3470

Score para el modelo de regresión lineal y los datos de validación	0.3260	0.3285
MAE	0.1337	0.1332
MSE	0.0252	0.0251
RMSE	0.1589	0.1587
TIEMPO		
Entrenamiento	0.0065 s	0.0045 s
Predicción	0.0075 s	0.0016 s

Como se observa en la anterior tabla, tenemos un score muy bajo, es decir que aun haciendo el análisis inicial donde comparamos con el mapa de calor, g[x] no tiene una correlación directa con stab; el error cuadrático medio y el error absoluto medio es bajo y entre ambos modelos no hay una diferencia significativa, al realizar la regresión polinómica el tiempo de entrenamiento y reducción es menor que la multiparamétrica.

- Solo tau[x]:

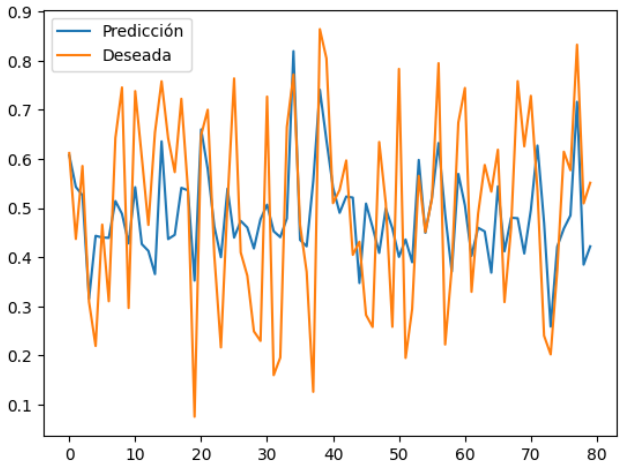

Fig. 9 Grafica regresión Multiparamétrica.

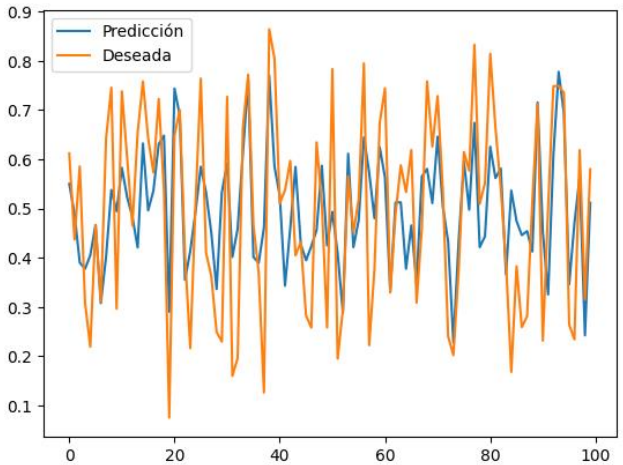

Fig. 10 Grafica regresión Polinómica.

RESUMEN	Multipara métrica	Polinómica
Score para el modelo de regresión lineal y los datos de entrenamiento	0.3160	0.4698
Score para el modelo de regresión lineal y los datos de validación	0.3120	0.4709
MAE	0.1312	0.1140
MSE	0.0258	0.0198
RMSE	0.1606	0.1408
TIEMPO		
Entrenamiento	0.0063 s	0.0065 s
Predicción	0.0017 s	0.0023 s

Ahora podemos observar que el score es mas alto que cuando solo utilizamos la característica g[x], la regresión polinómica es la que mas se ajusta ya que el score tanto de entrenamiento como de validación esta mas o menos en un valor de 0.47, en cuanto a las métricas de desempeño tenemos que la polinómica es la que tiene un error cuadrático medio y error absoluto medio, menor donde el tiempo de entrenamiento es levemente mayor que la multiparamétrica.

- Solo tau[x] y g[x]:

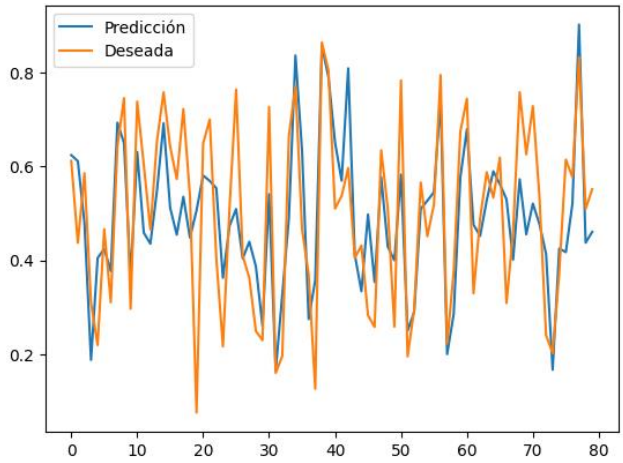

Fig. 11 Grafica regresión Multiparamétrica.

En este último caso donde nuestra X será respectivamente las características tau[x] y g[x], observamos que en la gráfica de regresión multiparamétrica tiene muy buena predicción y se ajusta a lo que realmente deseamos y esto se refleja en los valores que mas adelante se explica de manera mas detallada.

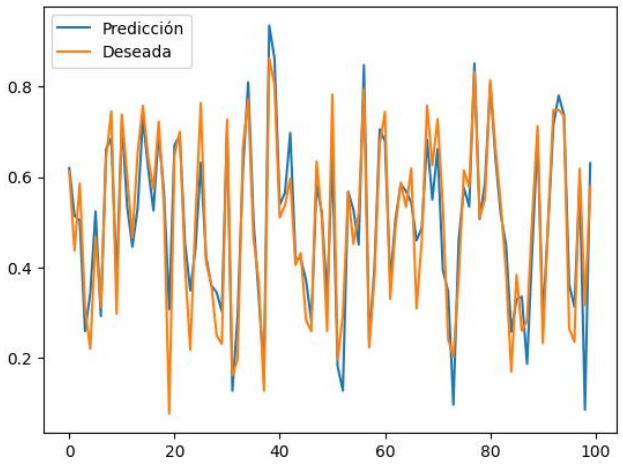

Fig. 12 Grafica regresión Polinómica.

RESUMEN	Multipara métrica	Polinómica
Score para el modelo de regresión lineal y los datos de entrenamiento	0.6470	0.8929
Score para el modelo de regresión lineal y los datos de validación	0.6452	0.8816
MAE	0.0923	0.0501
MSE	0.0133	0.0044
RMSE	0.1153	0.0666
TIEMPO		
Entrenamiento	0.0125 s	0.0103 s
Predicción	0.0061 s	0.0040 s

Cuando usamos estas dos características tenemos el score más alto tanto en entrenamiento como en validación para ambos modelos donde la regresión polinómica tiene el score más alto aproximando a 0.89, en métricas de desempeño tenemos un error cuadrático medio y error absoluto medio considerablemente bajo, igualmente en el tiempo observamos que la polinómica tiene un tiempo de predicción bajo a comparación de la multiparamétrica pero el tiempo de entrenamiento no tiene mucho margen de diferencia para ambos modelos.

V. CONCLUSIONES

En necesario hacer el análisis de datos donde verificamos la inestabilidad de la red eléctrica, las características que tienen correlación con nuestra salida que es el stab, pero el planteamiento inicial al cual se llegó donde nuestra característica de interés seria g[x] se evidencio que esta no era la que mejor se ajustaba ya que se obtuvo un score de 0.34 para datos de entrenamiento y un 0.32 para los de validación siendo estos los más bajos porque a comparación de la característica tau[x] en la cual se obtuvo un score mas alto de 0.47 aproximadamente tanto para entrenamiento como validación siento la característica que más importa y que mas peso tiene ya que es la única que tiene más correlación con stab.

Por otro lado, el regresor lineal polinómico es el mejor modelo y eso se refleja en las tres ultimas tablas las cuales se crearon con el fin de realizar la comparación entre ambos modelos y mirar su comportamiento. Cabe resaltar que el tiempo de entrenamiento de este modelo fue levemente mayor para un solo caso, pero esto no quiere decir que se descarte dicho método como el más adecuado. En cuanto a las métricas son proporcionales ya que, si el polinómico es el que mejor se ajusta el MAE, MSE y RMSE serán bajos y en algunos casos es considerable la diferencia entre ambos modelos.

Para finalizar, el mejor score se obtiene cuando las dos características se juntan y se hace el análisis donde en total tenemos 8 variables de entrada ya que observamos que tuvo un comportamiento muy superior a los otros casos donde el ajuste en cuento a la predicción y lo deseado fueron optimas, entonces lo que podemos predecir de ese comportamiento es que el valor del stab se saca de una ecuación la cual usa ambas características tau[x] y g[x]. Pero la potencia nominal p[x] no tiene mucha influencia en la ecuación entonces dicha característica se pudo haber creado para realizar un calculo aparte o hacer un análisis mas profundo y que dicha información no la tenemos disponible para saber con mas certeza cual era el fin de censar o anotar dichos valores de aquella característica.

VI. REFERENCIAS

[1] G. A. CORRALES GALLEGO. “Regresión Lineal Múltiple_y_Regresión Polinómica_Ejemplo Práctico”. Colab. Accedido el 28 de agosto de 2023. [En línea]. Disponible: https://colab.research.google.com/drive/18qt6-45vV5dYwdO02hFtE_C6BFEFqZsH?usp=sharing

[2] UC Irvine. “Electrical Grid Stability Simulated Data”. UCI Machine Learning Repository. Accedido el 28 de agosto de 2023. [En línea]. Disponible: https://archive.ics.uci.edu/dataset/471/electrical+grid+stability+simulated+data