

Carga de archivo csv

Haz doble clic (o ingresa) para editar

```
# Define where you are running the code: colab or local
RunInColab = True # (False: no | True: yes)
```

```
# If running in colab:
```

```
if RunInColab:
```

```
    # Mount Google Drive in Google Colab
```

```
    from google.colab import drive
```

```
    drive.mount('/content/drive')
```

```
    # Define the correct path to the CSV file
```

```
    data_path = '/content/drive/My Drive/semana:tec/A01644090_X.csv'
```

```
else:
```

```
    # Define path for local files
```

```
    data_path = ''
```

```
# Import the packages that we will be using
```


```
import pandas as pd
```

```
# Read the .csv file and store it as a pandas Data Frame
```

```
df = pd.read_csv(data_path)
```

```
# Display the first few rows of the dataframe to understand its structure
```

```
print(df.head())
```

 Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)

```
   Unnamed: 0    x1    x2    x3    x4    x5    x6  \
0           0  5.304286  1.976987  4.003868 -1.497814 -0.153787 -8.479913
1           1  0.676356  6.684277  11.403086  3.383758 -3.400203  4.237569
2           2 -0.361257  6.198021  6.942344  2.222555 -1.270385 -0.704624
3           3  1.430333  0.672608  6.110127  0.205691  1.580989  8.666351
4           4 -2.445370  8.443515  9.291438 -1.281821 -2.047883  5.289597
```

```
   x7    x8    x9    x10
0  2.808287  6.283290  3.732846  5.661476
1 -2.016223 -7.519270  4.909410  7.900466
2 -4.088247 -6.163221  5.732328  8.604937
3 -6.448158  3.445698 -3.974647 -3.545221
4 -4.330002 -8.898600  2.831141  8.570586
```

Exploración del Dataset

```
# Importar las librerías necesarias
```

```
import pandas as pd
```

```
# Definir la ruta del dataset (ajusta la ruta según sea necesario)
```

```
data_path = '/content/drive/My Drive/semana:tec/A01644090_X.csv'
```

```
# Leer el archivo CSV y almacenarlo en un DataFrame de pandas
```

```
df = pd.read_csv(data_path)
```

```
# Mostrar las primeras filas del DataFrame para obtener una vista general
```

```
print("Primeras 5 filas del dataset:")
```

```
print(df.head())
```

```
# Mostrar información general del DataFrame
```

```
print("\nInformación general del dataset:")
```

```
print(df.info())
```

```
# Mostrar estadísticas descriptivas del DataFrame
```

```
print("\nEstadísticas descriptivas del dataset:")
```

```
print(df.describe())
```

```
# Mostrar nombres de las columnas del DataFrame
```

```
print("\nNombres de las columnas del dataset:")
```

```
print(df.columns)
```



```

3      3      1.4303333  0.012000  0.110121  0.203091  1.300909  0.000331
4      4      -2.445370  8.443515  9.291438  -1.281821  -2.047883  5.289597

```

```

      x7      x8      x9      x10
0  2.808287  6.283290  3.732846  5.661476
1  -2.016223 -7.519270  4.909410  7.900466
2  -4.088247 -6.163221  5.732328  8.604937
3  -6.448158  3.445698 -3.974647 -3.545221
4  -4.330002 -8.898600  2.831141  8.570586

```

Información general del dataset:

```
<class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 293 entries, 0 to 292

Data columns (total 11 columns):

#	Column	Non-Null Count	Dtype
0	Unnamed: 0	293 non-null	int64
1	x1	293 non-null	float64
2	x2	293 non-null	float64
3	x3	293 non-null	float64
4	x4	293 non-null	float64
5	x5	293 non-null	float64
6	x6	293 non-null	float64
7	x7	293 non-null	float64
8	x8	293 non-null	float64
9	x9	293 non-null	float64
10	x10	293 non-null	float64

dtypes: float64(10), int64(1)

memory usage: 25.3 KB

None

Estadísticas descriptivas del dataset:

	Unnamed: 0	x1	x2	x3	x4	x5 \
count	293.000000	293.000000	293.000000	293.000000	293.000000	293.000000
mean	146.000000	0.909694	-1.514259	4.377985	1.437095	-2.708286
std	84.726029	3.574770	6.850504	5.113507	4.231813	3.734142
min	0.000000	-7.622364	-13.357276	-8.836877	-9.742761	-12.566926
25%	73.000000	-1.753690	-8.526348	3.482412	-1.152837	-5.276104
50%	146.000000	0.581937	0.664927	5.712819	2.158886	-2.297168
75%	219.000000	3.079631	3.986220	7.509755	4.482817	0.081407
max	292.000000	9.965550	10.571876	12.829852	9.974987	6.003102

	x6	x7	x8	x9	x10
count	293.000000	293.000000	293.000000	293.000000	293.000000
mean	-2.675008	-3.328159	1.550753	2.636260	1.510719
std	7.211905	4.084773	5.232312	4.845762	5.864493
min	-15.234590	-12.665213	-10.895927	-9.226881	-9.698777
25%	-8.494261	-6.130856	-0.617468	-1.391626	-3.682991
50%	-5.590076	-3.914893	2.805098	4.353533	0.837139
75%	3.357675	-1.352790	5.359798	6.283956	7.188291
max	12.939601	8.102385	10.696987	11.880024	12.894341

Nombres de las columnas del dataset:

```
Index(['Unnamed: 0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9',
      'x10'],
      dtype='object')
```

✓ Sección nueva

Visualización de Datos:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Definir la ruta del archivo CSV
```

```
data_path = '/content/drive/My Drive/semana:tec/A01644090_X.csv'
```

```
# Leer el archivo CSV y almacenarlo en un DataFrame de pandas
```

```
df = pd.read_csv(data_path)
```

```
# Eliminar la columna 'Unnamed: 0' si no es relevante para el análisis
```

```
df = df.drop(columns=['Unnamed: 0'])
```

```
# 1. Histograma para cada columna
```

```
# Un histograma ayuda a visualizar la distribución de cada característica en el dataset.
```

```
# Utilizamos 30 bins para una visualización más granular y colores para mejorar la claridad.
```

```
plt.figure(figsize=(15, 10))
```

```
df.hist(bins=30, figsize=(15, 10), color='steelblue', edgecolor='black')
```

```
plt.suptitle('Distribución de las características') # Título general del gráfico
plt.show()

# 2. Matriz de Correlación
# La matriz de correlación muestra la relación lineal entre las características.
# Un mapa de calor (heatmap) facilita la visualización de las correlaciones entre variables.
# Los valores de correlación cercanos a 1 o -1 indican una fuerte relación positiva o negativa, respectivamente.
plt.figure(figsize=(12, 10))
correlation_matrix = df.corr() # Calcula la matriz de correlación
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Matriz de Correlación') # Título del gráfico
plt.show()

# 3. Boxplot para detectar valores atípicos
# Un boxplot muestra la distribución de los datos y ayuda a identificar valores atípicos.
# Los valores fuera de los "bigotes" del boxplot se consideran outliers.
plt.figure(figsize=(15, 10))
sns.boxplot(data=df, palette="Set2")
plt.title('Boxplot de las características') # Título del gráfico
plt.xticks(rotation=45) # Rote las etiquetas del eje x para mejor visibilidad
plt.show()

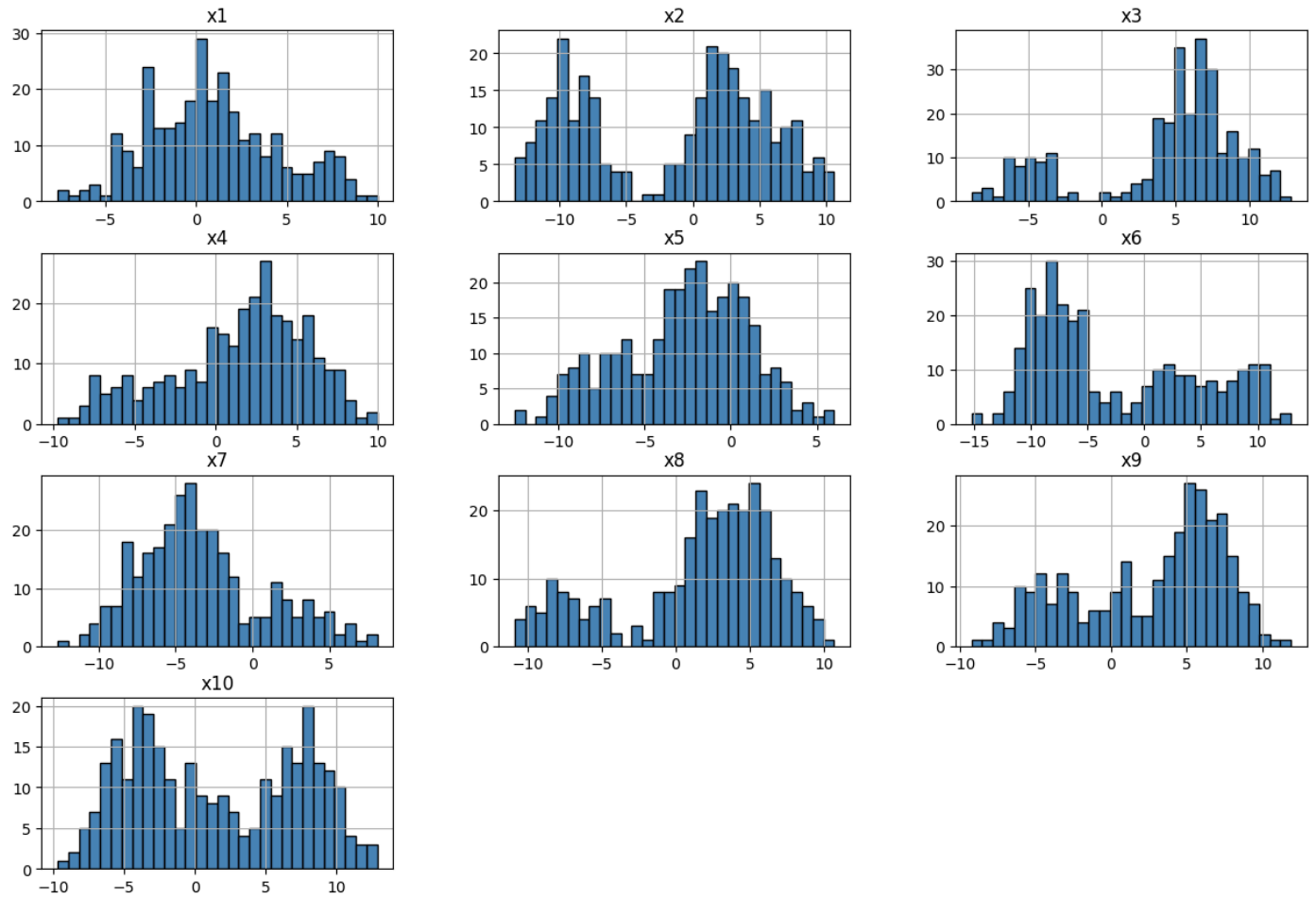
# 4. Pairplot para ver la relación entre pares de variables
# El pairplot muestra gráficos de dispersión para cada par de variables, facilitando la identificación de relaciones y patrones.
# Cada gráfico muestra cómo una variable se relaciona con otra en términos de dispersión.
# Nota: Si el dataset es grande, este gráfico puede tardar en generarse. Considera muestrear si es necesario.
plt.figure(figsize=(12, 12))
sns.pairplot(df)
plt.suptitle('Pairplot de las características', y=1.02) # Título del gráfico
plt.show()

# 5. Gráfico de distribución de cada variable
# Este gráfico muestra la distribución de cada variable en el dataset usando histogramas.
# Incluye una estimación de densidad (KDE) para visualizar la forma de la distribución.
# Cada subplot representa una variable diferente.
plt.figure(figsize=(15, 10))
for column in df.columns:
    plt.subplot(3, 4, list(df.columns).index(column) + 1) # Crea un subplot para cada columna
    sns.histplot(df[column], kde=True, color='skyblue') # Histograma con KDE
    plt.title(column) # Título del subplot
    plt.xlabel('') # Elimina la etiqueta del eje x para un aspecto más limpio
    plt.ylabel('') # Elimina la etiqueta del eje y para un aspecto más limpio

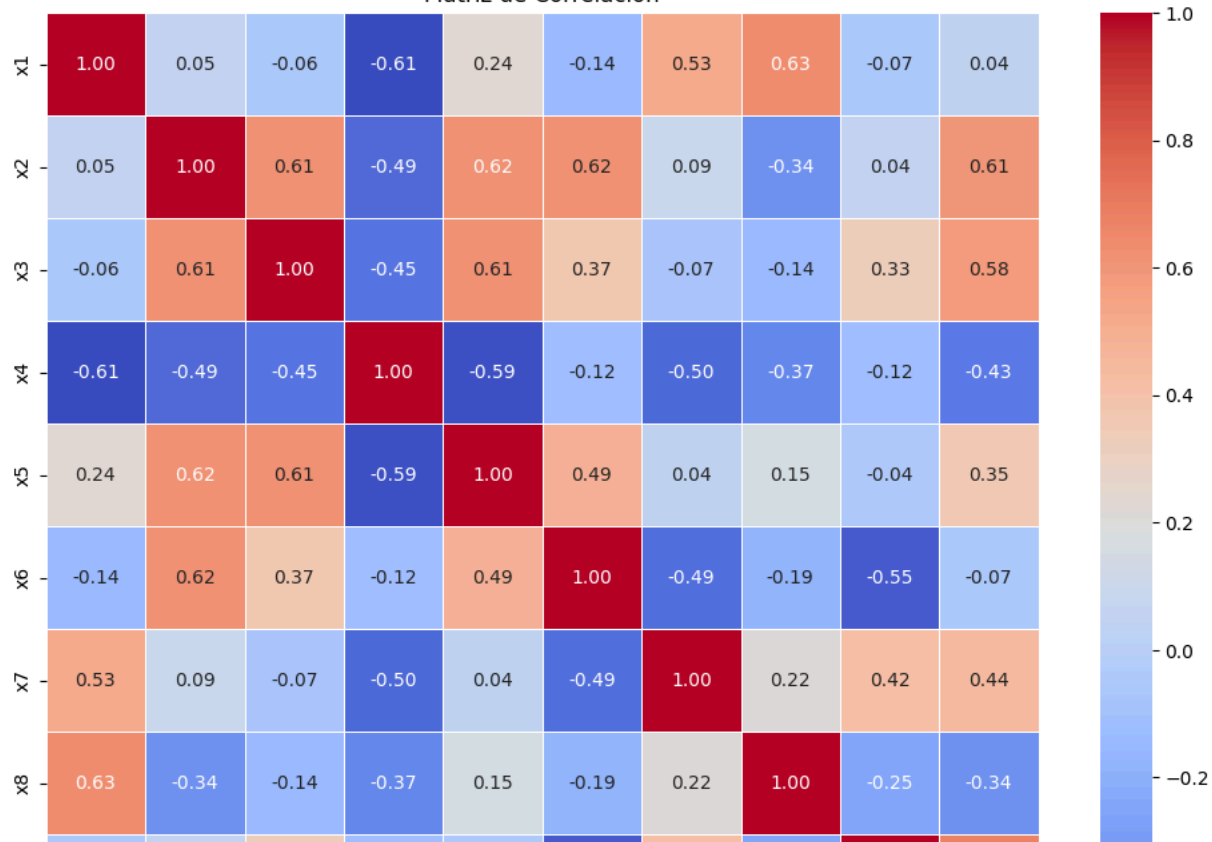
plt.tight_layout() # Ajusta el diseño para que no se superpongan los subplots
plt.suptitle('Distribución de Variables', y=1.02) # Título general para todos los subplots
plt.show()
```

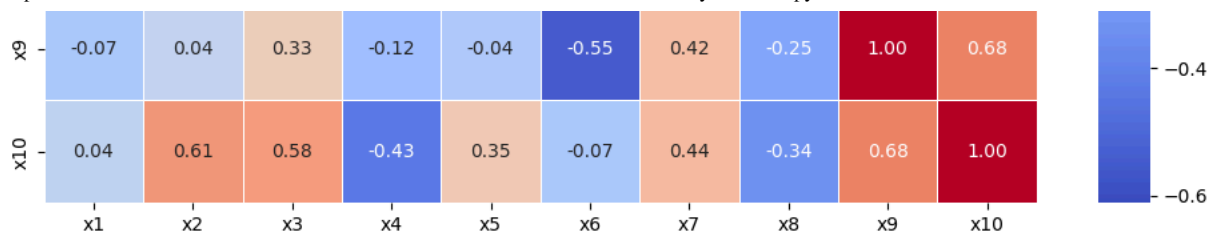
 <Figure size 1500x1000 with 0 Axes>

Distribución de las características

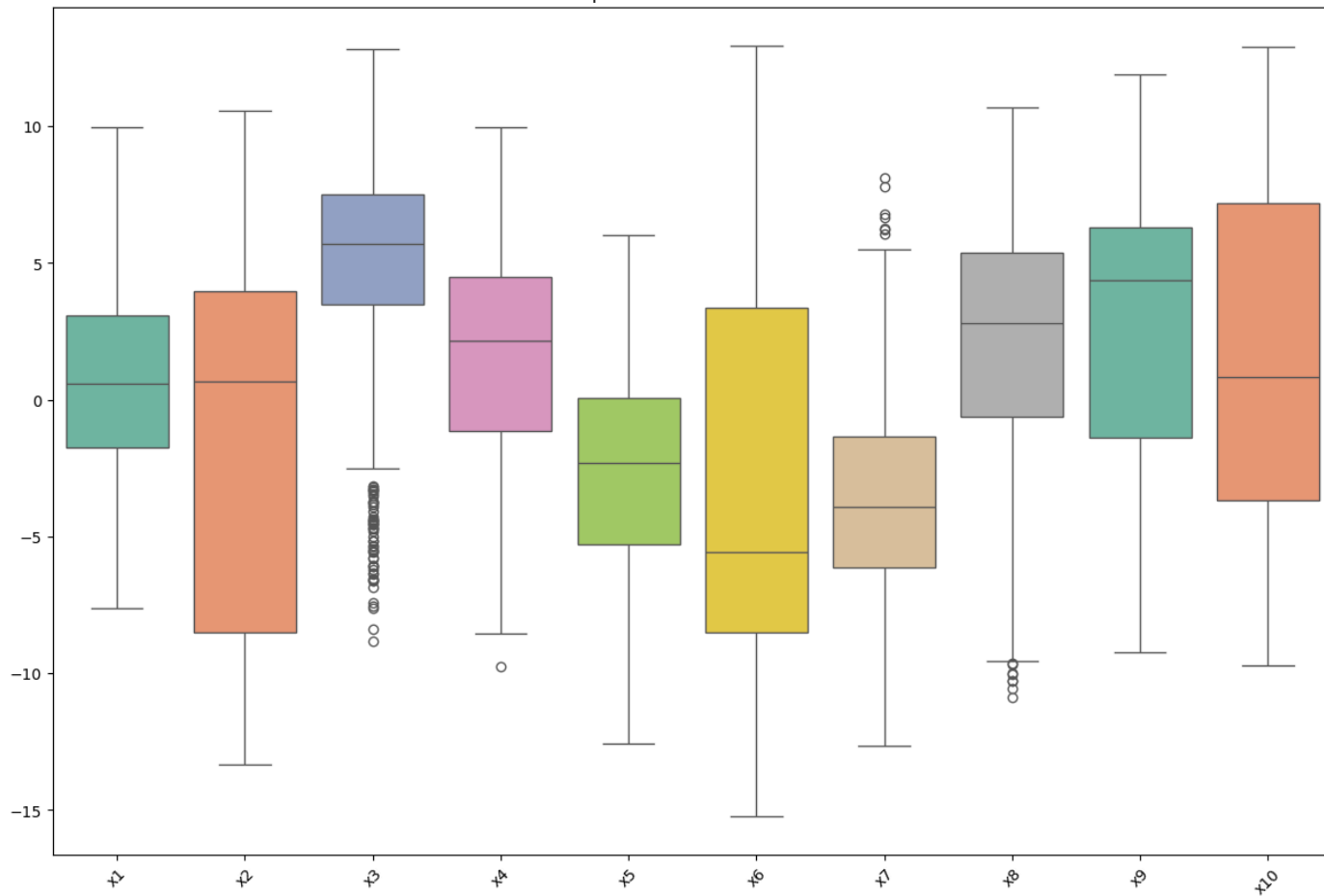


Matriz de Correlación



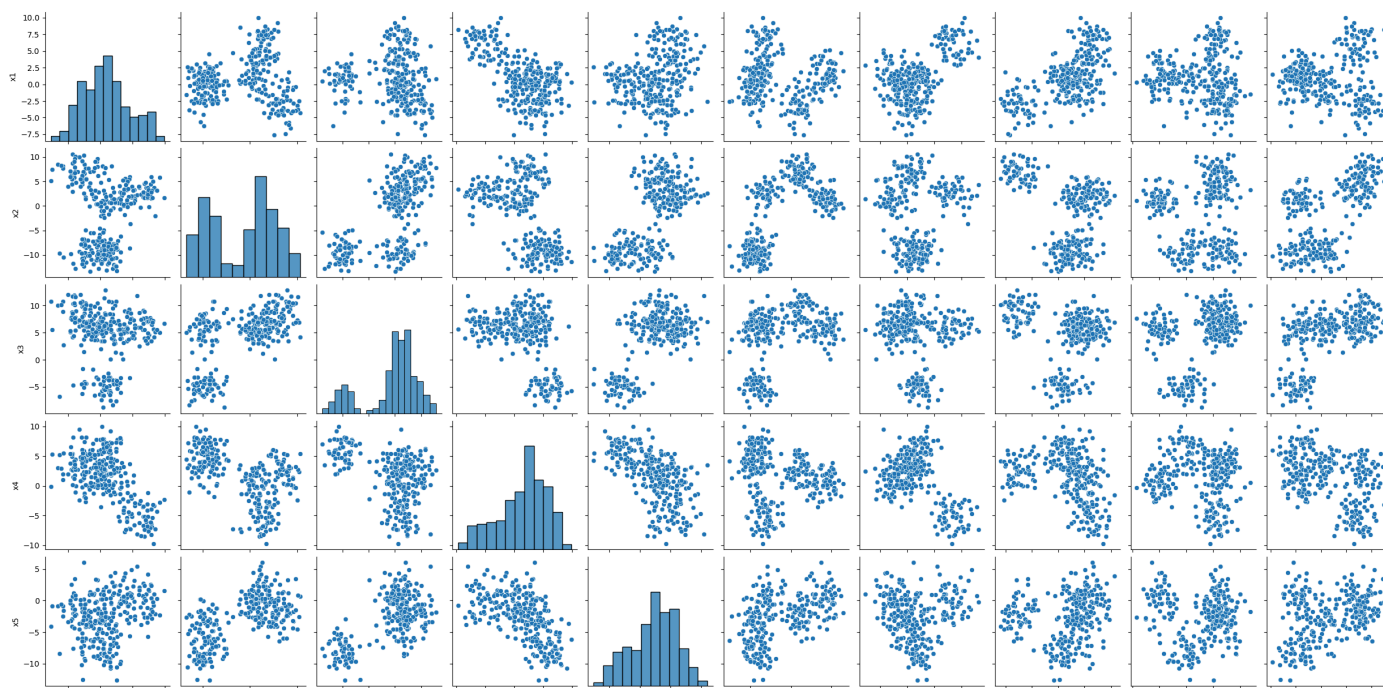


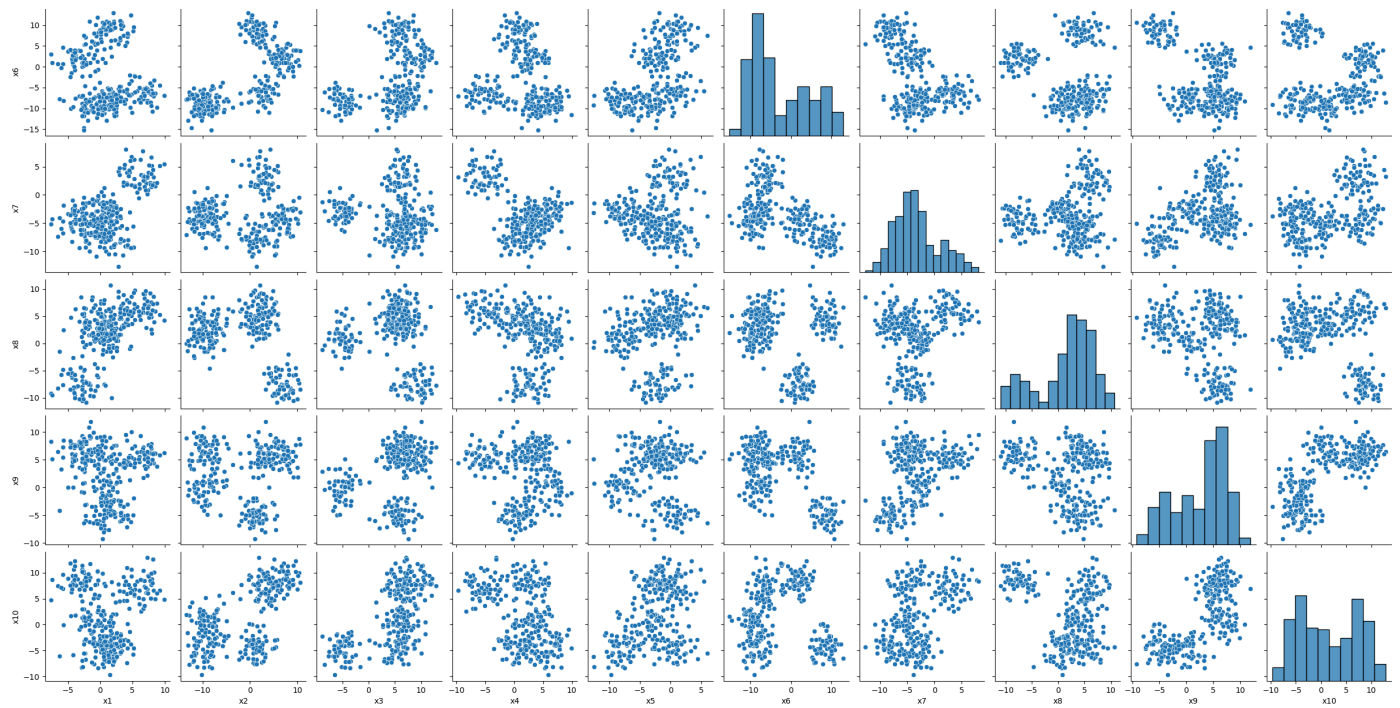
Boxplot de las características



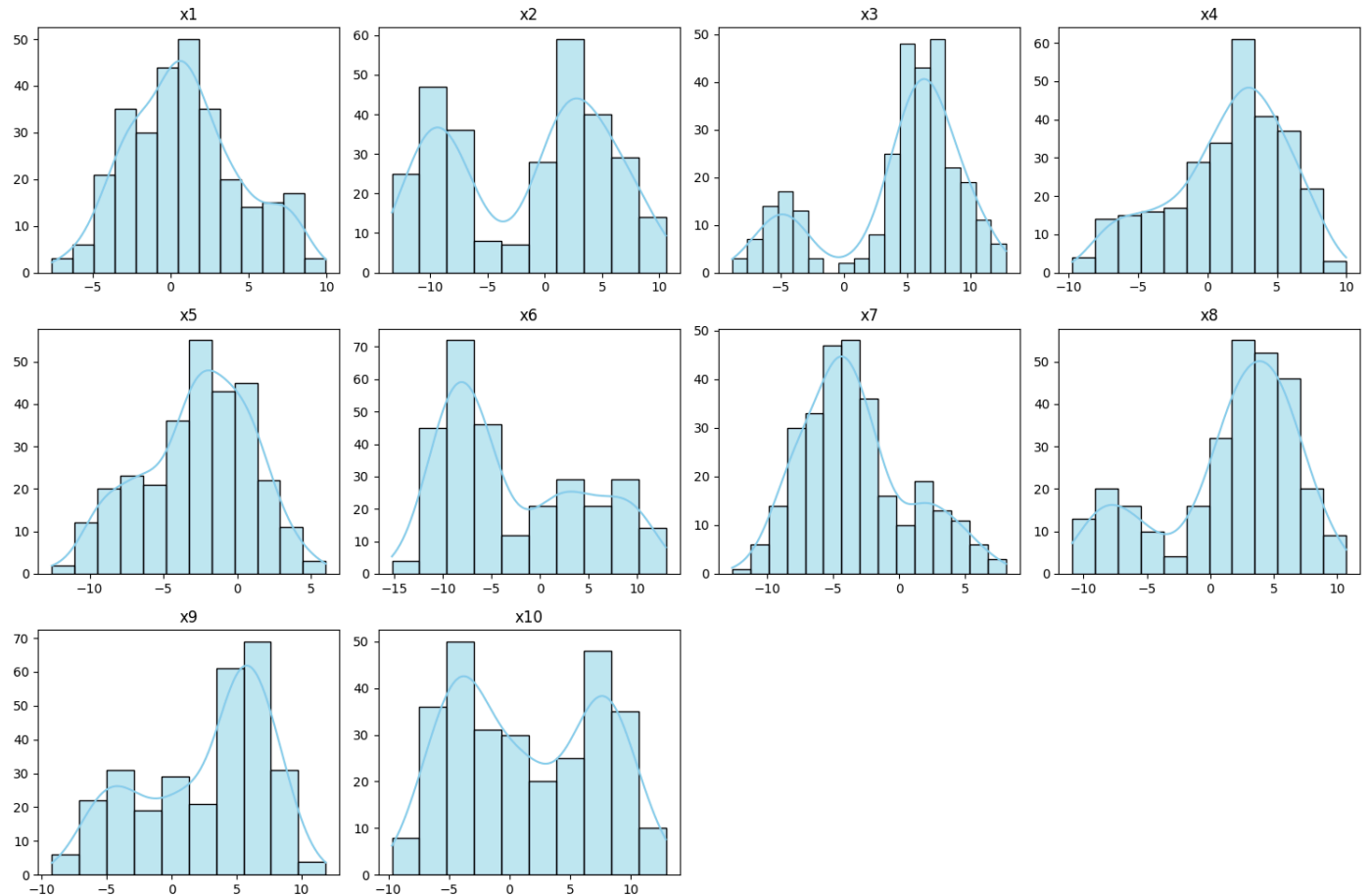
<Figure size 1200x1200 with 0 Axes>

Pairplot de las características





Distribución de Variables



Preparación de los Datos:

```
# Cargar el dataset desde la ruta proporcionada
data_path = '/content/drive/My Drive/semana:tec/A01644090_X.csv'
df = pd.read_csv(data_path)

# Eliminar la columna 'Unnamed: 0' si no es relevante para el análisis
df = df.drop(columns=['Unnamed: 0'])

# 1. Manejo de Valores Faltantes
# Verificar si hay valores faltantes en el dataset
print("Valores faltantes en cada columna:")
print(df.isnull().sum()) # Muestra la cantidad de valores faltantes por columna

# Si se encuentran valores faltantes, se pueden manejar de diferentes maneras:
# - Eliminando las filas o columnas con valores faltantes
# - Rellenando los valores faltantes con la media, mediana, o algún otro valor
# Ejemplo: rellenar los valores faltantes con la media de la columna (si hubiera valores faltantes)
# df.fillna(df.mean(), inplace=True)

# 2. Normalización de Datos
# La normalización ajusta los valores de las variables para que tengan una escala similar.
# Aquí utilizamos Min-Max Scaling para llevar los valores a un rango [0, 1].
from sklearn.preprocessing import MinMaxScaler

# Crear un objeto MinMaxScaler
scaler = MinMaxScaler()

# Aplicar la normalización a todas las columnas del dataset
df_normalized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Mostrar las primeras filas del dataset normalizado
print("\nPrimeras 5 filas del dataset normalizado:")
print(df_normalized.head())

# 3. Dividir el Dataset en Conjunto de Entrenamiento y Conjunto de Prueba
# Dividir el dataset en datos de entrenamiento y datos de prueba para evaluar el modelo
from sklearn.model_selection import train_test_split

# Definir las características (X) y la variable objetivo (y)
X = df_normalized.drop(columns=['x10']) # Suponiendo que 'x10' es la variable objetivo
y = df_normalized['x10']

# Dividir el dataset en entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Mostrar las dimensiones de los conjuntos de datos
print("\nDimensiones del conjunto de entrenamiento:")
print(X_train.shape, y_train.shape)

print("\nDimensiones del conjunto de prueba:")
print(X_test.shape, y_test.shape)

# 4. Verificar el Dataset Preparado
# Verificar las primeras filas del conjunto de entrenamiento
print("\nPrimeras 5 filas del conjunto de entrenamiento:")
print(X_train.head())

# Verificar las primeras filas del conjunto de prueba
print("\nPrimeras 5 filas del conjunto de prueba:")
print(X_test.head())
```

```
x5      0
x6      0
x7      0
x8      0
x9      0
x10     0
dtype: int64
```

Primeras 5 filas del dataset normalizado: