ExploratoryDataAnalysis

January 28, 2025

1 Análisis Exploratorio de Datos (EDA) de Base de Datos de EMOTHAW

Proyecto Integrador TC5035.10

Profesor Asesor: - Dra. Ludivina Facundo (ITESM)

Profesores Investigadores/Tutores: - Dr. Juan Arturo Nolazco (ITESM) - Dr. Marcos Faunez Zaunuy (TecnoCampus Barcelona)

Equipo 11: - Francisco José Arellano Montes (A01794283) - Armando Bringas Corpus (A01200230) - Moisés Díaz Malagón (A01208580)

1.1 Importación de librerías

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pywt
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.colors
```

[2]:
```
sns.set(style="whitegrid")
```

1.1.1 Descripción de librerías

- **pandas (`import pandas as pd`)**: Una librería para la manipulación y análisis de datos tabulares. Proporciona estructuras como DataFrames y Series, útiles para gestionar datos estructurados de manera eficiente.
- **numpy (`import numpy as np`)**: Una librería fundamental para el cálculo numérico en Python. Ofrece soporte para matrices y operaciones matemáticas de alto rendimiento.
- **matplotlib.pyplot (`import matplotlib.pyplot as plt`)**: Un módulo de Matplotlib que permite crear visualizaciones gráficas 2D, como gráficos de líneas, barras, dispersión, entre otros.
- **seaborn (`import seaborn as sns`)**: Una librería de visualización de datos basada en Matplotlib, que facilita la creación de gráficos estadísticos más atractivos y con estilo.

- **`pywt (import pywt)`**: Una librería para realizar transformadas wavelet, ampliamente utilizada en el análisis de señales y datos en aplicaciones como procesamiento de imágenes y series temporales.

1.2 Carga del Conjunto de Datos

Acerca del set de datos:

Los datos para usar son el conjunto de datos Emothaw desarrollado por (Likforman-Sulem et al., 2017), es un set de datos explícitamente diseñado al estudio de emociones usando análisis y patrones en la escritura.

Incluye muestras escritas a mano recopiladas de participantes en diferentes estados emocionales.

Estos estados emocionales a menudo se inducen mediante métodos psicológicos validados, como videos o estímulos diseñados para provocar emociones (por ejemplo, felicidad, tristeza, ira, etc.).

1.2.1 Carga de conjunto de datos

Los datos se encuentran en formato .csv, estos datos fueron conglomerados anteriormente junto con sus propias etiquetas y guardados en un documento. parquet, el cual usamos para cargar la información de manera mas sencilla.

El proceso de conglomerado de datos raw a un .parquet sucede en el notebook notebooks/DataPreparation.ipynb

[3]:
```
file_path = '../data/raw_binary/labeled_data_timeseries.parquet'
df = pd.read_parquet(file_path)
```

[4]:
```
df.head()
```

[4]:

Subject	homework	pen_status	x	y
1	1	0	[48331, 48318, 48305, 48305, 48305, 48305, 483…	[31876, 31963, 32053, 32159, 32159, 32159, 321…
1	1	1	[47944, 47949, 47949, 47950, 47950, 47950, 479…	
1	2	0	[41647, 41714, 41787, 41896, 41896, 41616, 416…	
1	2	1	[45074, 45128, 45144, 45158, 45164, 45169, 451…	
1	3	0	[33431, 33563, 33626, 33685, 33747, 33812, 338…	

```
1        [33492, 33506, 33512, 33515, 33519, 33524, 335…
1        [14655, 14657, 14675, 14677, 14677, 15475, 154…
1        [14676, 14676, 14679, 14689, 14696, 14701, 147…
1        [35956, 35956, 35959, 35976, 35991, 36002, 360…

                                                 timestamp  \
Subject
1        [672620, 672628, 672635, 672643, 672650, 67282…
1        [671854, 671861, 671869, 671876, 671884, 67189…
1        [692915, 692922, 692930, 692937, 692945, 69356…
1        [692434, 692441, 692449, 692456, 692464, 69247…
1        [724897, 724905, 724912, 724920, 724927, 72493…

                                                  azimuth  \
Subject
1        [1830, 1830, 1830, 1830, 1830, 1830, 1830, 234…
1        [1800, 1800, 1800, 1800, 1800, 1810, 1810, 181…
1        [1830, 1830, 1830, 1840, 1840, 2530, 2530, 253…
1        [1930, 1940, 1940, 1940, 1940, 1940, 1940, 194…
1        [1760, 1760, 1760, 1760, 1760, 1760, 1770, 177…

                                                  altitude  \
Subject
1        [530, 530, 530, 530, 530, 530, 530, 350, 360, …
1        [490, 500, 500, 500, 500, 500, 500, 500, 500, …
1        [530, 530, 530, 530, 530, 450, 450, 450, 460, …
1        [510, 510, 510, 510, 510, 510, 510, 510, 510, …
1        [620, 610, 610, 610, 610, 610, 600, 600, 600, …

                                                  pressure  depression  \
Subject
1        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …           2
1        [67, 148, 193, 228, 270, 306, 341, 365, 381, 3…           2
1        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …           2
1        [50, 207, 282, 304, 377, 418, 426, 438, 447, 4…           2
1        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …           2

         anxiety  stress
Subject
1              8      13
1              8      13
1              8      13
1              8      13
1              8      13
```

Como se puede observar, el dataframe continene las siguientes columnas:

- **Subject** el cual identifica al usuario al que se le hicieron las pruebas.

- **Homework** representa el tipo de dibujo realizado por el usuario. Se tienen 8 tipos de tareas:
    0. Clasificado erróneamente
    1. Pentágonos
    2. Casa
    3. Palabras copiadas a mano
    4. Círculos concéntricos dibujados con mano izquierda
    5. Círculos concéntricos dibujados con mano derecha
    6. Reloj
    7. Texto copiado en cursivas
- **pen_status** este nos muestra si la pluma se encuentra arriba (0) o abajo (1).
- **X y Y** son series de tiempo que representan las coordenadas de los trazos en x y y.
- **Azimuth** son series de tiempo que representan el ángulo entre la orientación del lápiz y una dirección de referencia en el plano de la superficie de la tableta.
- **Altitude** es una serie de tiempo que contiene ángulo entre el lápiz y la superficie de la tableta.
- **Pressure** Presión al escribir en la pluma.
- Nuestras etiquetas serian los valores booleanos designados a las emociones como **depresión, ansiedad y estrés**. De acuerdo con el trabajo desarrollado por (Nolazco-Flores, et al, 2021) se consideraron las etiquetas de acuerdo con lo siguiente:

Label	Depression	Anxiety	Stress
Normal (0)	0-9	0-7	0-14
Above Normal (1)	10-28+	8-20+	15-34+

En este notebook ya se cuentan con las etiquetas en forma booleana, para ver los valores raw se puede consultar el notebook notebooks/DataPreparation.ipynb

1.3 Estructura de los Datos

1.3.1 Forma del conjunto de datos

[5]:
```
print("Forma del conjunto de datos:", df.shape)
```

```
Forma del conjunto de datos: (1588, 11)
```

Podemos observar que nuestros datos actuales tienen la dimensión de (1588, 11), en otras palabras: 1588 ejemplos.

1.3.2 Tipos de datos y conteo de valores no nulos

[6]:
```
print("\nTipos de datos y conteo de valores no nulos:")
df.info()
```

```
Tipos de datos y conteo de valores no nulos:
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 1588 entries, 1 to 129
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   homework    1588 non-null   int64
 1   pen_status  1588 non-null   int64
 2   x           1588 non-null   object
 3   y           1588 non-null   object
 4   timestamp   1588 non-null   object
 5   azimuth     1588 non-null   object
 6   altitude    1588 non-null   object
 7   pressure    1588 non-null   object
 8   depression  1588 non-null   int64
 9   anxiety     1588 non-null   int64
 10  stress      1588 non-null   int64
dtypes: int64(5), object(6)
memory usage: 148.9+ KB
```

En este caso el tipo de dato no indica si la información es categórica o numérica, sin embargo de acuerdo a la descripción de las variables en el paper de la base de datos EMOTHAW (Likforman-Sulem et al., 2017), se determina que:

Variables categóricas (5): - homework - pen_status - depression (label) - anxiety (label) - stress (label)

Variables numéricas (5): - x - y - azimuth - altitude - pressure

En este caso las variables numéricas ya se encuentran en formato de series de tiempo no como valores individuales.

Se considera anular la variable de timestamp dado que las señales ya están ordenadas de manera cronológica en el tiempo y no aporta información adicional para los métodos de clasificación que serán utilizados.

[7]:
```
categorical_columns = ['homework', 'pen_status', 'depression', 'anxiety',␣
 ↪'stress']
numerical_columns = ['x', 'y', 'azimuth', 'altitude', 'pressure']
```

Podemos adicionalmente determinar el tipo de las variables en las series de tiempo

[8]:
```
for numerical_col in numerical_columns:
    print(f"Tipo de datos para {numerical_col}: {type(df[numerical_col].
 ↪iloc[0][0])}")
```

```
Tipo de datos para x: <class 'numpy.int64'>
Tipo de datos para y: <class 'numpy.int64'>
Tipo de datos para azimuth: <class 'numpy.int64'>
Tipo de datos para altitude: <class 'numpy.int64'>
Tipo de datos para pressure: <class 'numpy.int64'>
```

Podemos observar que todos los datos en la serie de tiempo son enteros.

1.3.3 Resumen estadístico para columnas numéricas

Para el análisis estadístico descriptivo de las variables numéricas se utilizará el dataframe que contiene un sample por fila.

[9]:
```
file_path = '../data/raw_binary/labeled_data.parquet'
df_raw = pd.read_parquet(file_path)
```

[10]:
```
print("\nEstadísticas descriptivas para columnas numéricas:")
df_raw[numerical_columns].describe()
```

```
Estadísticas descriptivas para columnas numéricas:
```

[10]:

	x	y	azimuth	altitude	pressure
count	2.680608e+06	2.680608e+06	2.680608e+06	2.680608e+06	2.680608e+06
mean	3.204618e+04	1.988526e+04	1.915995e+03	5.893773e+02	2.983005e+02
std	1.359955e+04	1.007838e+04	5.267162e+02	7.944353e+01	3.006754e+02
min	2.215000e+03	1.000000e+01	0.000000e+00	2.200000e+02	0.000000e+00
25%	2.138500e+04	1.169200e+04	1.750000e+03	5.400000e+02	0.000000e+00
50%	3.353400e+04	1.498200e+04	1.890000e+03	5.800000e+02	2.690000e+02
75%	4.572400e+04	3.054200e+04	2.060000e+03	6.300000e+02	5.480000e+02
max	6.150400e+04	4.063000e+04	3.590000e+03	9.000000e+02	1.023000e+03

1.3.4 Descripción de las variables numéricas y estadísticas descriptivas

- **`X` y `Y`**:
  - Representan las coordenadas de los trazos en el plano de la tableta. Estas series de tiempo capturan cómo varían las posiciones en los ejes `X` e `Y` durante el movimiento del lápiz.

  - **Estadísticas descriptivas (`X`)**:
    - **Media (`mean`)**: 32,046.18

    - **Desviación estándar (`std`)**: 13,599.55

    - **Valor mínimo (`min`)**: 2,215

    - **Valor máximo (`max`)**: 61,504

    - **Cuartiles**:
      - 25%: 21,385

      - 50%: 33,534

      - 75%: 45,724
  - **Estadísticas descriptivas (`Y`)**:
    - **Media (`mean`)**: 19,885.26

    * **Desviación estándar (`std`)**: 10,078.38
    * **Valor mínimo (`min`)**: 1,000
    * **Valor máximo (`max`)**: 40,630
    * **Cuartiles**:
        · 25%: 11,692
        · 50%: 14,982
        · 75%: 30,542

- `Azimuth`:
  - Representa el ángulo entre la orientación del lápiz y una dirección de referencia en el plano de la superficie de la tableta. Este valor varía dependiendo de la inclinación horizontal del lápiz.
  - **Estadísticas descriptivas**:
    * **Media (`mean`)**: 1,915.99
    * **Desviación estándar (`std`)**: 526.72
    * **Valor mínimo (`min`)**: 0.00
    * **Valor máximo (`max`)**: 3,590
    * **Cuartiles**:
        · 25%: 1,750
        · 50%: 1,890
        · 75%: 2,060

- `Altitude`:
  - Captura el ángulo entre el lápiz y la superficie de la tableta. Este valor describe cómo de inclinado está el lápiz verticalmente.
  - **Estadísticas descriptivas**:
    * **Media (`mean`)**: 589.38
    * **Desviación estándar (`std`)**: 79.44
    * **Valor mínimo (`min`)**: 220
    * **Valor máximo (`max`)**: 900
    * **Cuartiles**:
        · 25%: 540

        - 50%: 580
        - 75%: 630
- **Pressure**:
    - Indica la presión ejercida por el lápiz al escribir sobre la superficie de la tableta.
    - **Estadísticas descriptivas**:
        - **Media (mean)**: 298.30
        - **Desviación estándar (std)**: 300.68
        - **Valor mínimo (min)**: 0.00
        - **Valor máximo (max)**: 1,023
        - **Cuartiles**:
            - 25%: 0.00
            - 50%: 269
            - 75%: 548

Estas estadísticas muestran las características principales de cada variable, ayudando a entender el comportamiento del trazo, la orientación del lápiz, la inclinación y la presión ejercida.

1.3.5 Frecuencia de valores únicos para columnas categóricas

[11]:
```
for column in categorical_columns:
    print("Para {} se tienen {} valores únicos".format(column, df[column].
 ↪nunique()))
    print("Que son los siguientes: \n{}\n".format(df[column].value_counts()))
```

```
Para homework se tienen 8 valores únicos
Que son los siguientes:
homework
1    258
2    258
3    258
6    258
7    258
4    157
5    135
0      6
Name: count, dtype: int64

Para pen_status se tienen 2 valores únicos
Que son los siguientes:
pen_status
```

```
1    906
0    682
Name: count, dtype: int64

Para depression se tienen 25 valores únicos
Que son los siguientes:
depression
4     196
2     147
3     132
7     125
5     122
8     113
6     110
1      97
9      74
13     63
11     61
10     50
0      49
18     37
16     28
25     27
17     24
20     24
21     24
15     24
14     13
24     12
23     12
22     12
19     12
Name: count, dtype: int64

Para anxiety se tienen 25 valores únicos
Que son los siguientes:
anxiety
2     157
6     148
7     147
5     112
8      98
4      97
3      97
9      86
1      85
13     76
0      64
```

```
11     61
10     60
16     51
12     50
15     39
14     37
20     24
22     24
17     14
25     13
19     12
31     12
18     12
27     12
Name: count, dtype: int64

Para stress se tienen 33 valores únicos
Que son los siguientes:
stress
16    112
14    110
8     108
15    100
6      96
13     85
11     76
17     76
9      75
19     65
5      60
18     60
4      60
20     50
7      50
3      48
12     38
27     38
2      37
10     36
28     36
21     26
22     24
23     13
0      13
1      12
31     12
32     12
30     12
```

```
39     12
35     12
25     12
24     12
Name: count, dtype: int64
```

1.3.6 Verificación de valores nulos

[12]:
```
print("\nValores faltantes en el conjunto de datos:")
df_raw.isnull().sum()
```

```
Valores faltantes en el conjunto de datos:
```

[12]:
```
x             0
y             0
timestamp     0
pen_status    0
azimuth       0
altitude      0
pressure      0
homework      0
Subject       0
depression    0
anxiety       0
stress        0
dtype: int64
```

Por la naturaleza del set de datos, al ser datos recolectados por expertos que requerían realizar ejercicios manuales y repetirse en caso de que no se completar el ejercicio, nuestro set de datos no tiene ningún valor nulo.

Esto significa que no es necesario implementar algún método o algoritmo que nos ayude a llenar los datos nulos.

Sin embargo observamos que existen 6 casos de tareas que no están clasificadas dentro de una categoría dada. Por el momento no se eliminarán ya que representan el 0.37% de los datos. Al ser mínimo determinaremos si causan un impacto al modelo de clasificación más adelante.

[13]:
```
float(df['homework'].value_counts()[0]/df['homework'].shape[0]*100)
```

[13]:
```
0.3778337531486146
```

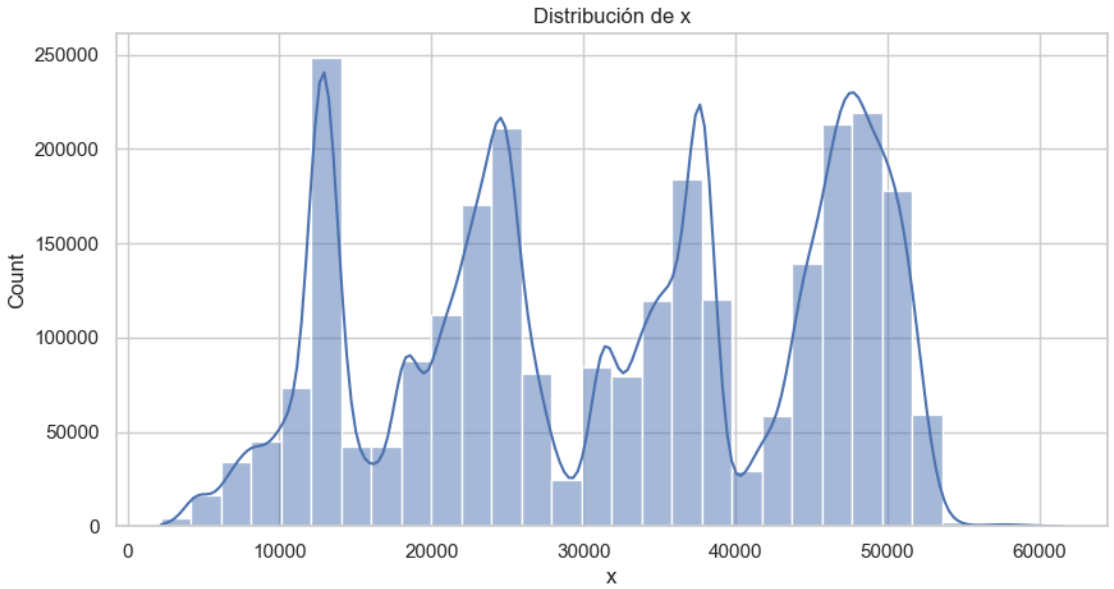
1.4 Análisis Univariante

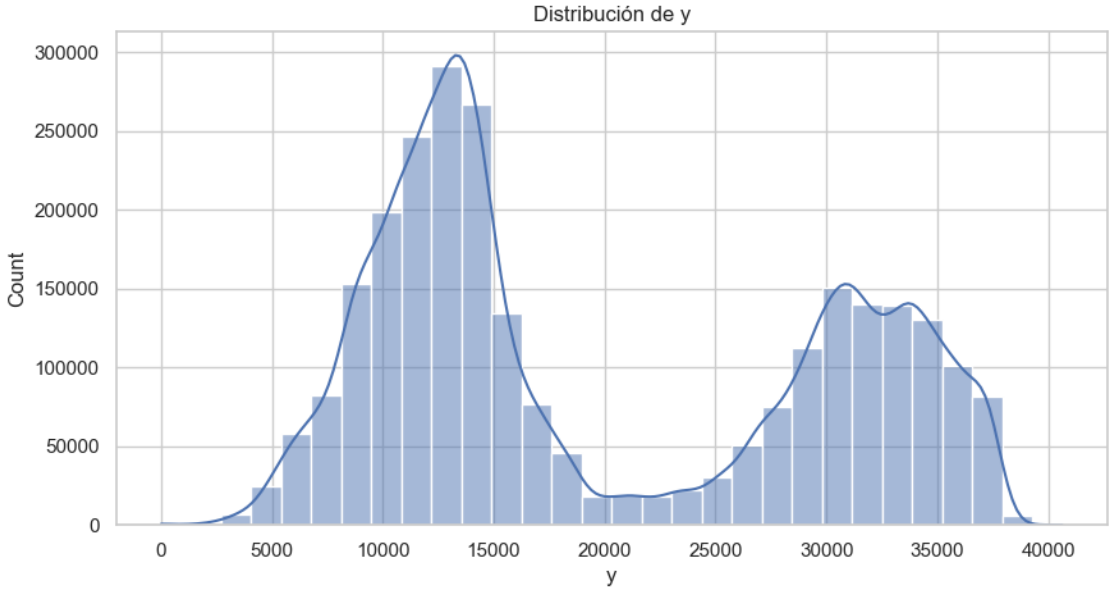
1.4.1 Visualizaciones para Datos Numéricos

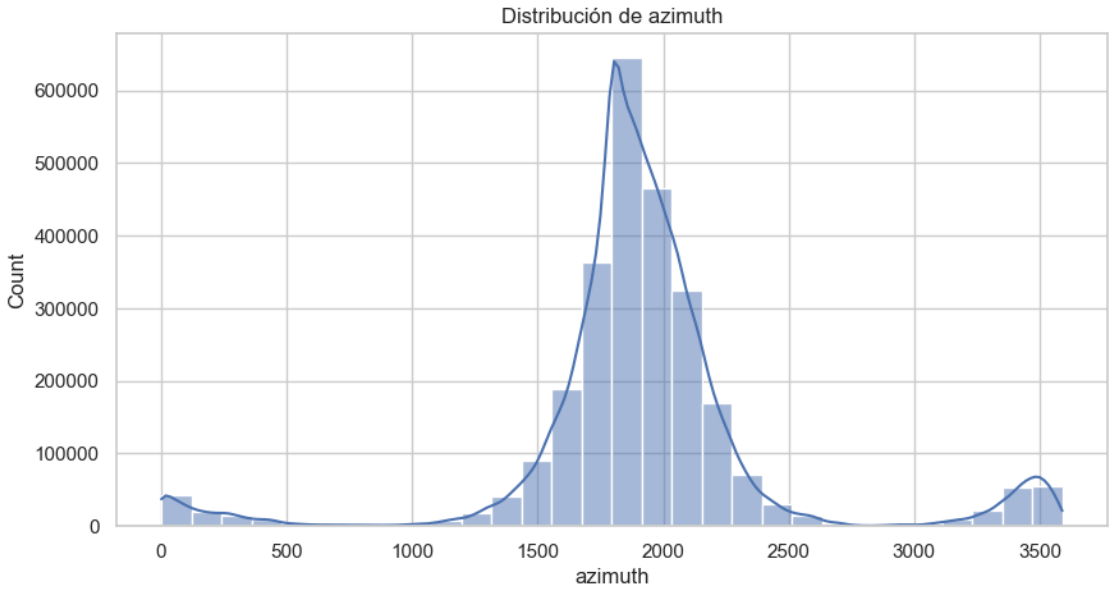
[14]:
```
def make_countplot(df, column_names, bins=10, fig_size=(10,20)):
  plt.figure(figsize=fig_size, constrained_layout=True)
  for idx,column_name in enumerate(column_names):
    ax = plt.subplot(len(column_names)//2+1, 3,idx+1)
    sns.countplot(df, x=column_name)
    ax.set_title(column_name)
  plt.show()

def make_hist_plot(df, column_names, bins=10, fig_size=(10,20)):
  plt.figure(figsize=fig_size, constrained_layout=True)
  for idx,column_name in enumerate(column_names):
    ax = plt.subplot(len(column_names)//2+1, 3,idx+1)
    df[column_name].hist(bins=bins)
    ax.set_title(column_name)
  plt.show()
```

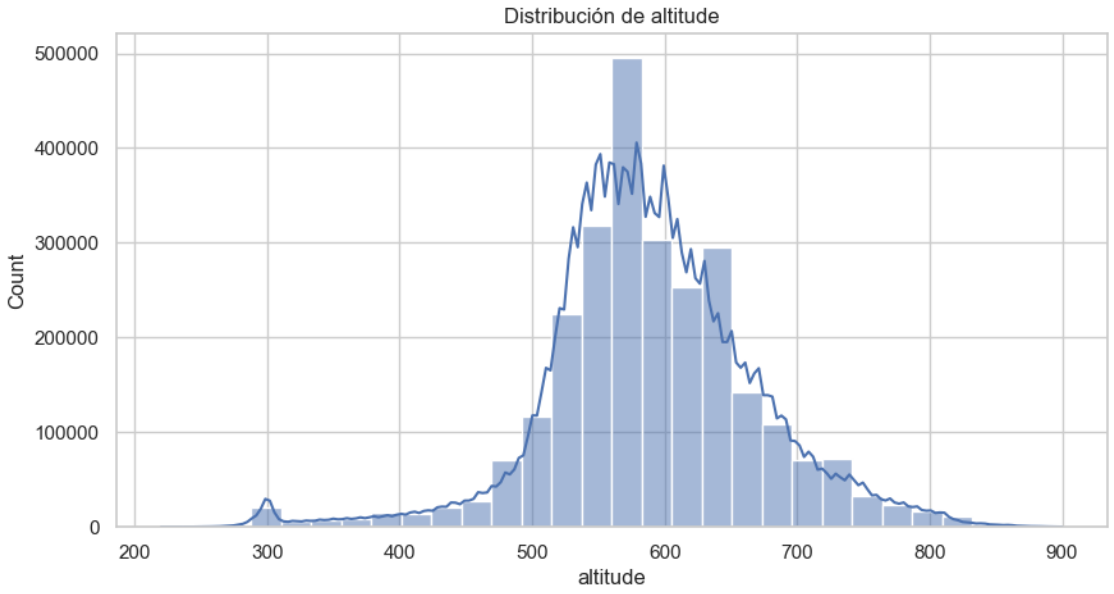
Histogramas

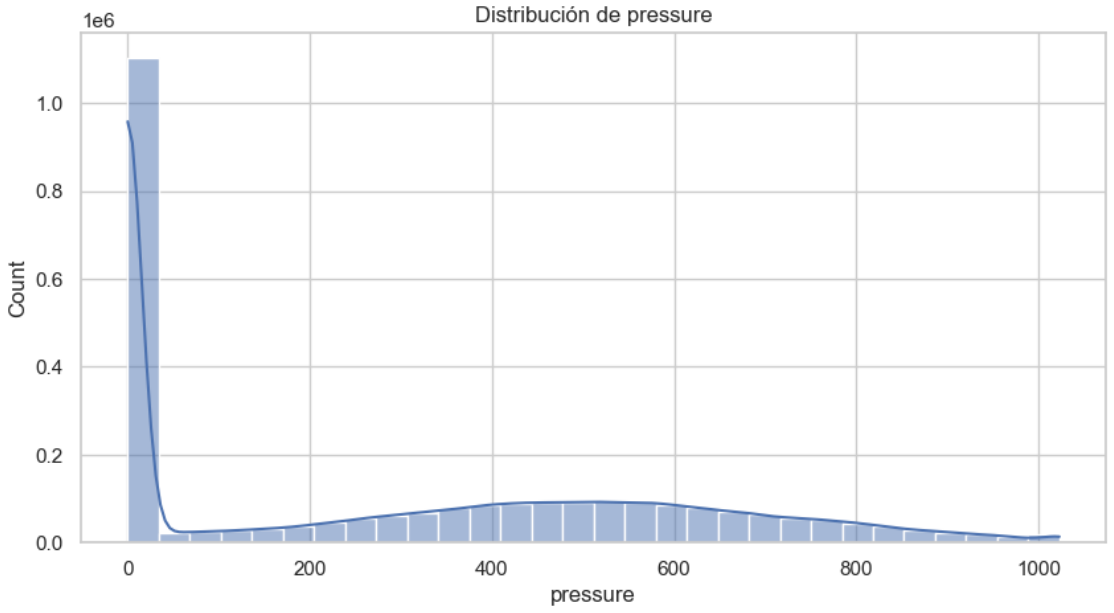
[15]:
```
for col in numerical_columns:
    plt.figure(figsize=(10, 5))
    sns.histplot(df_raw[col], kde=True, bins=30)
    plt.title(f"Distribución de {col}")
    plt.show()
```




Distribución de y
Count
y
0
50000
100000
150000
200000
250000
300000
0
5000
10000
15000
20000
25000
30000
35000
40000


Distribución de azimuth
Count
azimuth
0
100000
200000
300000
400000
500000
600000
0
500
1000
1500
2000
2500
3000
3500





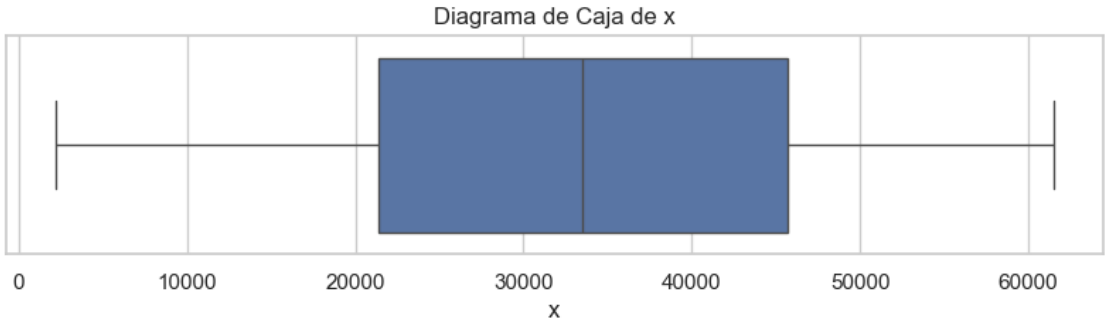
Descripción de los histogramas:

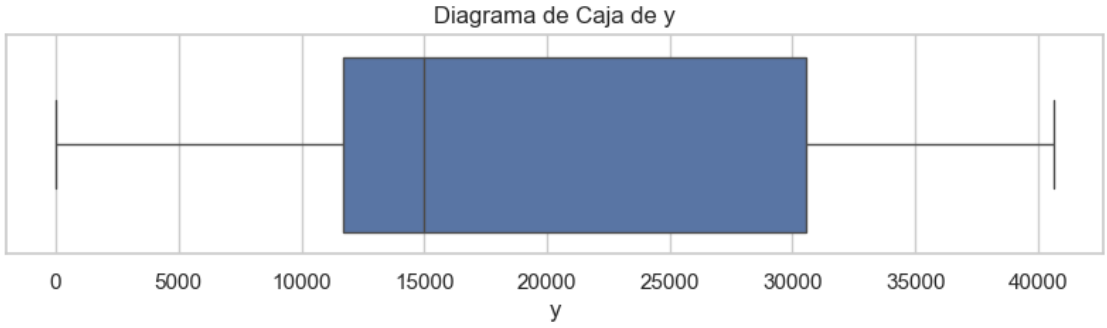
- **Distribución de la variable ´x´**:
  - **Multimodalidad:** La distribución muestra múltiples picos, indicando la presencia de varias subpoblaciones o grupos distintos.
  - **Sesgo:** Algunos picos son asimétricos, aunque la distribución general parece balanceada entre grupos.
  - **Curtosis:** Moderada, con picos definidos pero sin colas excesivamente largas o planas.

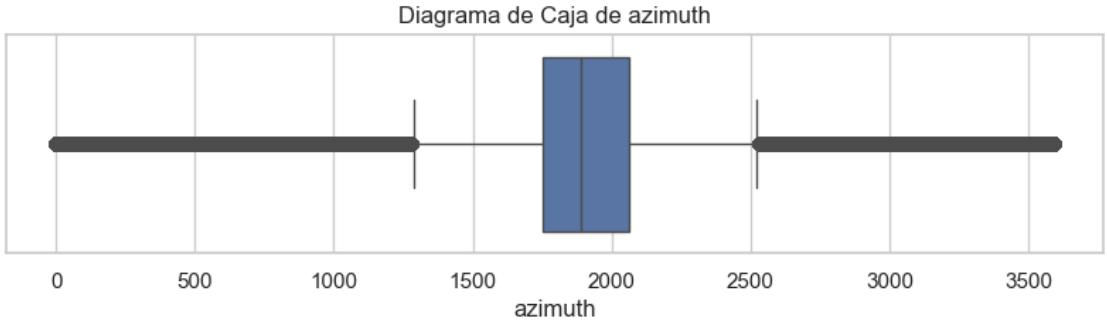
  - **Dispersión:** Amplio rango de valores entre 0 y 60,000, lo que indica alta variabilidad.
  - **Colas:** No se observan colas extremas, sugiriendo pocos o ningún outlier significativo.
- **Distribución de la variable ´y´**:
  - **Multimodalidad:** La distribución presenta dos picos principales, indicando dos grupos o subpoblaciones claras.
  - **Sesgo:** El primer pico parece ligeramente sesgado hacia la derecha, mientras que el segundo está más balanceado.
  - **Curtosis:** Moderada, con un primer pico pronunciado y un segundo menos definido.
  - **Dispersión:** Los valores están contenidos entre 0 y 40,000, con mayor concentración entre 10,000 y 15,000, y otro grupo entre 30,000 y 35,000.
  - **Colas:** Las colas no son extremadamente largas, pero hay una pequeña disminución gradual hacia los extremos.
- **Distribución de la variable ´azimuth´**:
  - **Multimodalidad:** La distribución es predominantemente unimodal, con un pico central bien definido alrededor de 2000, aunque hay pequeñas concentraciones en los extremos.
  - **Sesgo:** La distribución parece simétrica alrededor del pico principal, sin sesgo significativo.
  - **Curtosis:** Alta, con un pico central estrecho y pronunciado, lo que indica una concentración significativa de valores alrededor del centro.
  - **Dispersión:** Los valores están concentrados principalmente entre 1500 y 2500, con caídas graduales hacia los extremos en un rango de 0 a 3600.
  - **Colas:** Las colas son relativamente cortas, con una ligera presencia de valores hacia los extremos izquierdo y derecho.
- **Distribución de la variable ´altitude´**:
  - **Multimodalidad:** La distribución es principalmente unimodal, con un pico definido alrededor de 600, aunque hay pequeñas fluctuaciones en los alrededores del pico principal.
  - **Sesgo:** Ligeramente sesgada hacia la derecha, con una caída más gradual en los valores altos (700-900) comparado con los valores bajos.
  - **Curtosis:** Moderada, con un pico relativamente pronunciado en el centro y una dispersión gradual hacia los extremos.
  - **Dispersión:** Los valores están concentrados principalmente entre 500 y 700, con un rango total de aproximadamente 200 a 900.
  - **Colas:** Las colas son visibles pero no extremadamente largas, indicando pocos valores extremos fuera del rango principal.
- **Distribución de la variable ´pressure´**:
  - **Multimodalidad:** La distribución es principalmente unimodal, con una alta concentración de valores en el rango más bajo (cercano a 0), seguida de una larga cola que se extiende hacia valores más altos.
  - **Sesgo:** Fuertemente sesgada hacia la derecha, con la mayor densidad concentrada en valores cercanos a 0.
  - **Curtosis:** Alta, con un pico extremo en valores bajos y una dispersión muy gradual hacia el resto del rango.
  - **Dispersión:** Los valores están distribuidos en un rango amplio de 0 a 1000, pero la mayoría se encuentra muy cerca de 0.
  - **Colas:** Una cola larga y extendida hacia la derecha, lo que indica la presencia de valores más altos que ocurren con menor frecuencia.

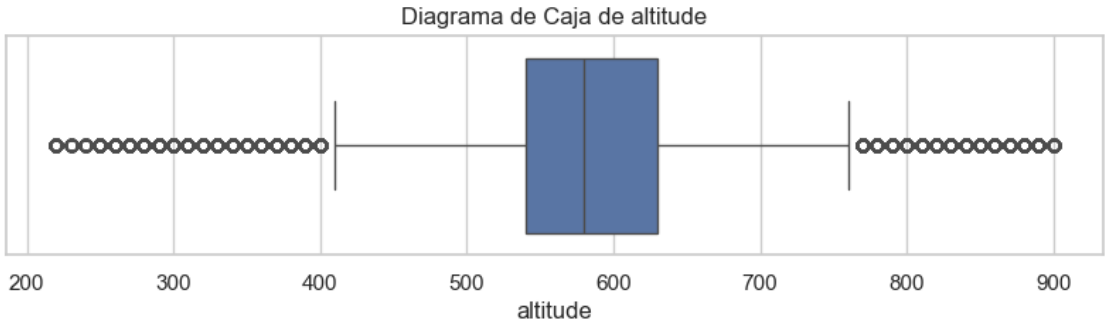
Diagramas de Caja

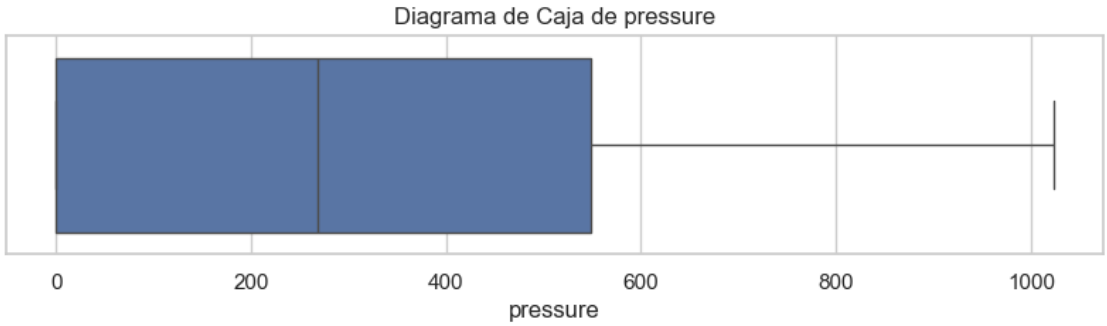
[16]:
```
for col in numerical_columns:
    plt.figure(figsize=(10, 2))
    sns.boxplot(x=df_raw[col])
    plt.title(f"Diagrama de Caja de {col}")
    plt.show()
```











Descripción de los diagramas de caja:

- **Descripción del boxplot de `x`**
  - **Mediana:** La línea central dentro de la caja está cercana a los 33,534, lo cual coincide con el valor reportado como mediana en las estadísticas.
  - **Rango intercuartílico (IQR):** La caja abarca de aproximadamente 21,385 (Q1) a 45,724 (Q3), lo que indica un amplio rango para los valores centrales del 50% de los datos.
  - **Simetría:** Los bigotes son casi simétricos respecto a la mediana, sugiriendo una distribución relativamente balanceada sin un sesgo extremo.
  - **Outliers:** No se observan puntos fuera de los bigotes, indicando que no hay valores atípicos significativos detectados en los datos.
  - **Dispersión:** Los bigotes se extienden desde aproximadamente 2,215 (mínimo) hasta 61,504 (máximo), mostrando una alta dispersión en los valores de la variable.
- **Descripción de boxplot de `y`**
  - **Mediana:** La línea central dentro de la caja está cerca de los 14,982, indicando que la mitad de los datos se encuentra por debajo de este valor.
  - **Rango intercuartílico (IQR):** La caja abarca desde aproximadamente 11,692 (Q1) hasta 30,542 (Q3), mostrando una dispersión amplia en los datos centrales del 50%.
  - **Simetría:** Los bigotes son ligeramente más largos hacia el lado derecho, lo que sugiere un sesgo positivo en la distribución.

  - **Outliers:** No se observan puntos fuera de los bigotes en este gráfico, aunque los valores cercanos al máximo podrían analizarse más detalladamente como potenciales outliers.
  - **Dispersión:** Los datos están distribuidos en un rango que va desde 10 (mínimo) hasta 40,630 (máximo), con una mayor densidad entre Q1 y Q3.

- **Descripción del boxplot de `azimuth`**
  - **Mediana:** La línea central dentro de la caja está cerca de los 1,890, lo que indica que la mitad de los datos se encuentra por debajo de este valor.
  - **Rango intercuartílico (IQR):** La caja abarca desde aproximadamente 1,750 (Q1) hasta 2,060 (Q3), mostrando una concentración significativa de valores en este rango.
  - **Simetría:** Los bigotes son relativamente equilibrados, aunque se extienden más hacia los valores extremos (0 y 3,590).
  - **Outliers:** Hay puntos identificados fuera de los bigotes, tanto en el extremo inferior (cercanos a 0) como en el extremo superior (cercanos a 3,590), lo que indica la presencia de valores atípicos.
  - **Dispersión:** La mayor densidad de datos se concentra en el rango entre Q1 y Q3, mientras que los valores extremos se distribuyen de manera más dispersa hacia los límites del rango total.

- **Descripción del boxplot de `altitude`**
  - **Mediana:** La línea central dentro de la caja está cerca de los 580, indicando que la mitad de los datos está distribuida de manera equilibrada en torno a este valor.
  - **Rango intercuartílico (IQR):** La caja abarca desde aproximadamente 540 (Q1) hasta 630 (Q3), lo que muestra una concentración significativa de datos dentro de este rango.
  - **Simetría:** Los bigotes son asimétricos, extendiéndose más hacia el extremo inferior que hacia el superior, lo que indica un sesgo leve hacia los valores bajos.
  - **Outliers:** Se observan puntos fuera de los bigotes tanto en los extremos inferiores (cercanos a 200) como superiores (cercanos a 900), lo que identifica valores atípicos significativos.
  - **Dispersión:** La mayoría de los datos están concentrados dentro del rango intercuartílico, con una dispersión moderada hacia los valores extremos.

- **Descripción del boxplot de `pressure`**
  - **Mediana:** La línea central dentro de la caja está cerca de los 269, indicando que la mitad de los valores de presión está por debajo de este punto.
  - **Rango intercuartílico (IQR):** La caja abarca desde aproximadamente 0 (Q1) hasta 548 (Q3), mostrando que una gran proporción de los datos está concentrada en este rango.
  - **Simetría:** Los bigotes son notablemente asimétricos, extendiéndose mucho más hacia el lado superior, lo que refleja una distribución sesgada positivamente.
  - **Outliers:** No se identifican puntos fuera de los bigotes, lo que sugiere que no hay valores atípicos significativos.
  - **Dispersión:** Aunque el rango de valores va desde 0 hasta 1,023, la mayor concentración se encuentra dentro del IQR, mientras que los valores altos están más dispersos.

1.4.2 Análisis de Series de Tiempo

Dado que estamos trabajando con series de tiempo y que los datos se procesarán de esta forma, es más util observar la distribución temporal de los datos así como el equivalente a lo que sería un diagrama de caja de forma temporal. Esto nos indica valores máximos, mínimos, quartiles y media para cada sample tiempo.

[17]:
```
def sample_windows(data, homework, variable, samples=2):
    sample = data[data['homework']==homework]['x'].sample(n=samples)
    return sample

def plot_random_signals(df, variable, title, homework, samples=3):
    layout = go.Layout(title=title,
        xaxis=dict(title='Timestamp'),
        yaxis=dict(title='Value'),
        hovermode='closest'
    )
    fig = go.Figure(data=[
        go.Scatter(x=list(range(len(data))), y=data, mode='lines', name='Sample␣
 ↪{}') for data in sample_windows(df, homework=homework, variable=variable,␣
 ↪samples=samples)
    ], layout=layout)
    fig.show()
```

[18]:
```
plot_random_signals(df, 'x', 'X signals for 3 random samples of homework 1',␣
 ↪homework=1, samples=3)
```

[19]:
```
plot_random_signals(df, 'y', 'Y signals for 3 random samples of homework 1',␣
 ↪homework=3, samples=3)
```

Como puede observarse, las variables temporales tanto de X como Y no tienen la misma cantidad de samples. Esto se debe a que los datos fueron recolectados de diferentes personas que dibujaron de manera distinta y tiempos distintos. A continuación se puede observar que para X y Y las longitudes varían y tienen las siguientes características de acuerdo con su longitud.

[20]:
```
df['x'].apply(lambda x: len(x)).describe()
```

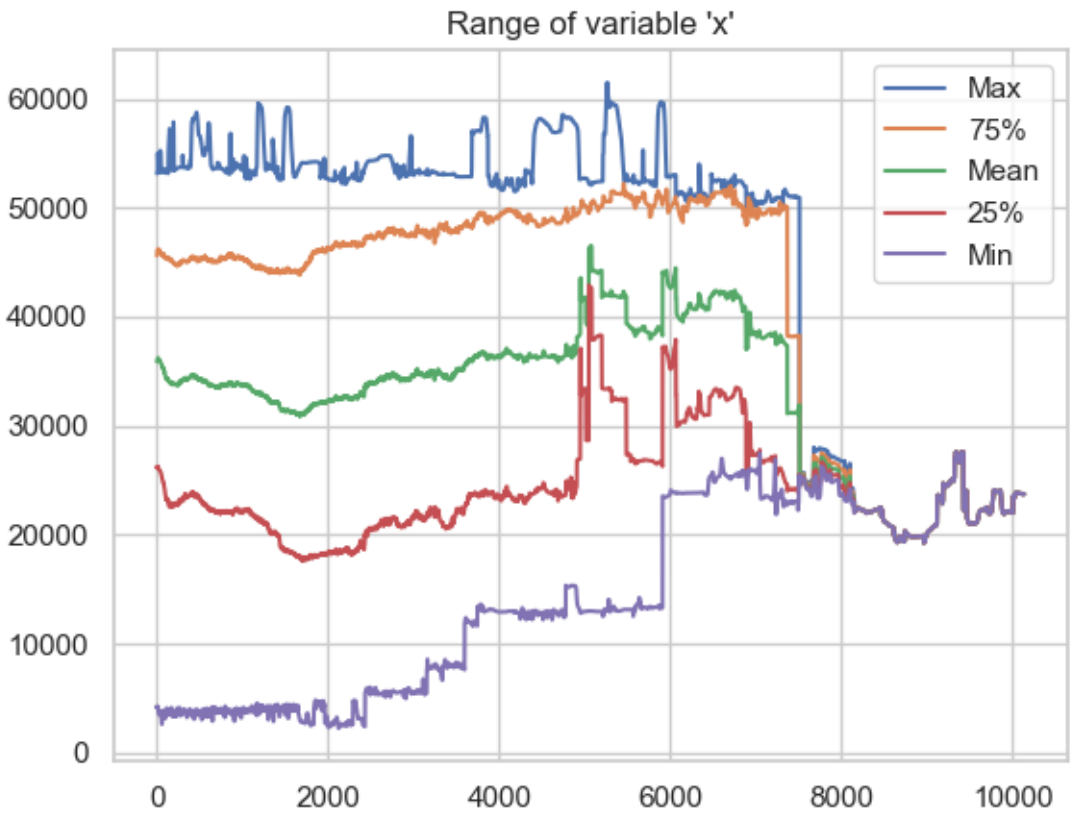
[20]:
```
count     1588.000000
mean      1688.040302
std        917.135949
min          2.000000
25%       1162.750000
50%       1577.000000
75%       2046.500000
max      10136.000000
Name: x, dtype: float64
```

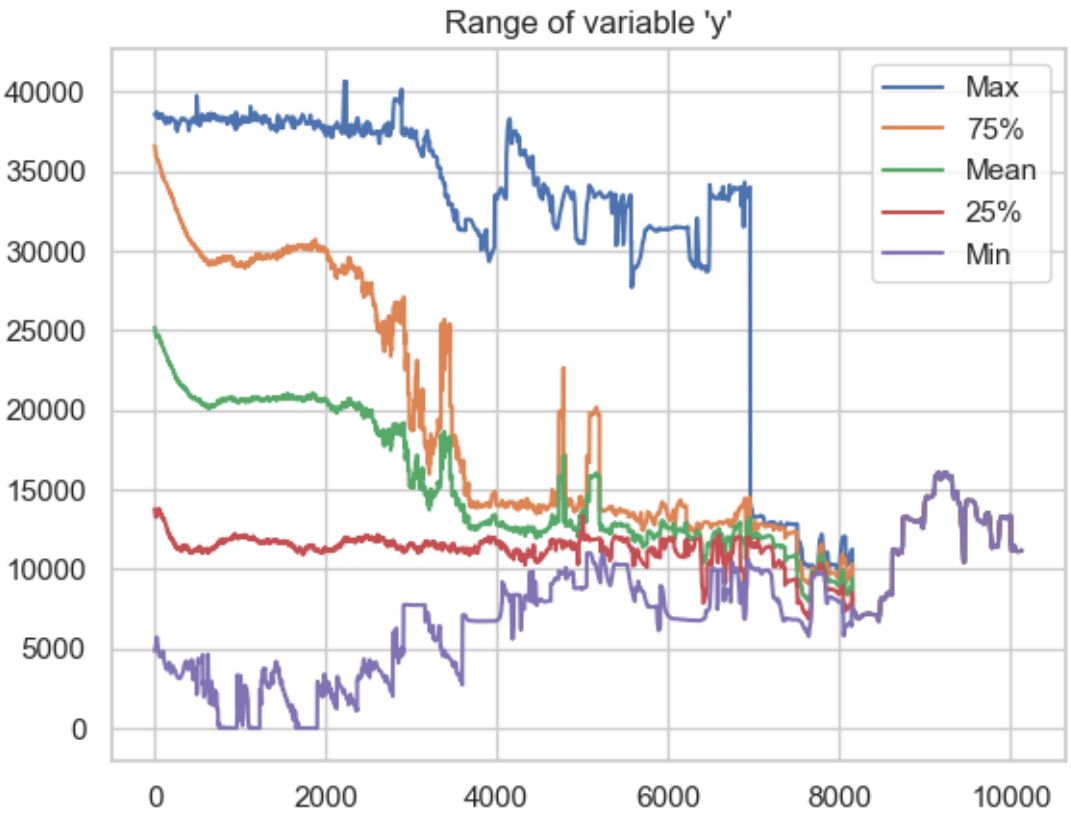
Los ejemplos tienen una media de 1688 samples, el que menos samples tiene 2 y el que más tiene son 10,136 samples.

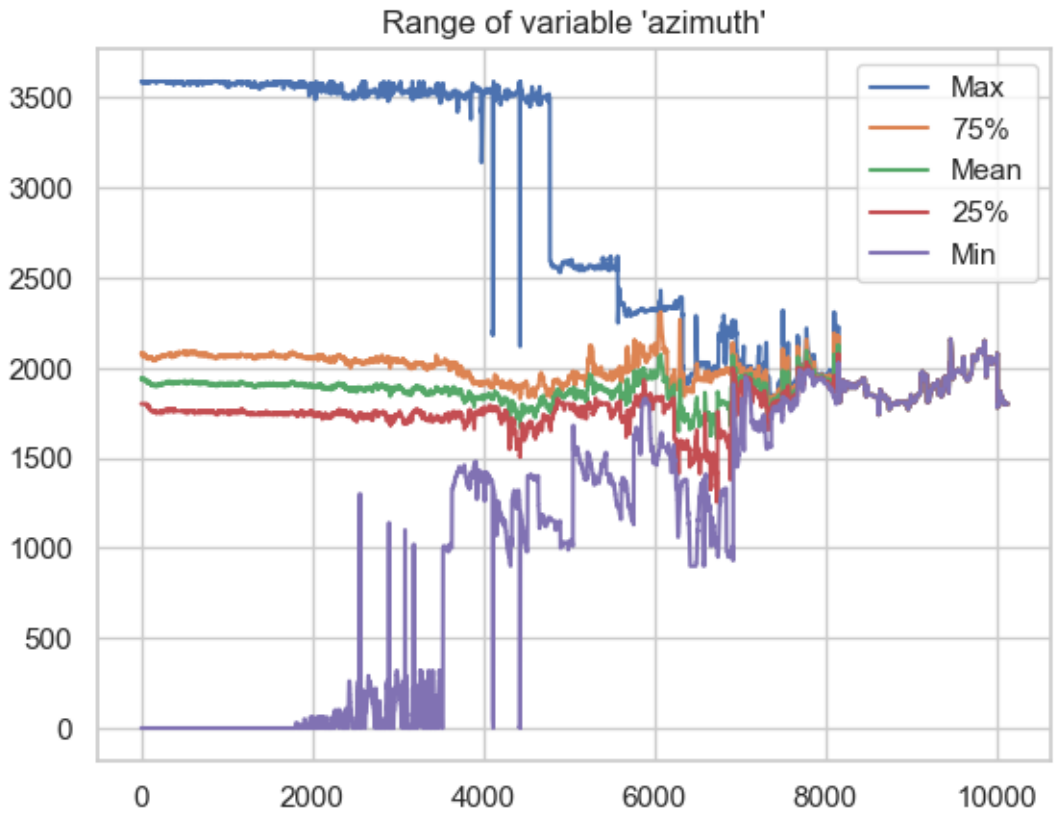
Distribución de datos de señales de series de tiempo (min, max, media, quantiles 25% y 75%) A continuación se muestran los rangos máximos y mínimos para los diferentes instantes de tiempo de las señales numéricas, así como la media y los quantiles 25% y 75%. Puede observarse que el valor los valores colisionan porque pocos dibujos tienen una longitud mayor a 8000 muestras, en estos casos las métricas de maximo, mínimo y media dan el mismo valor al ser la única muestra disponible.

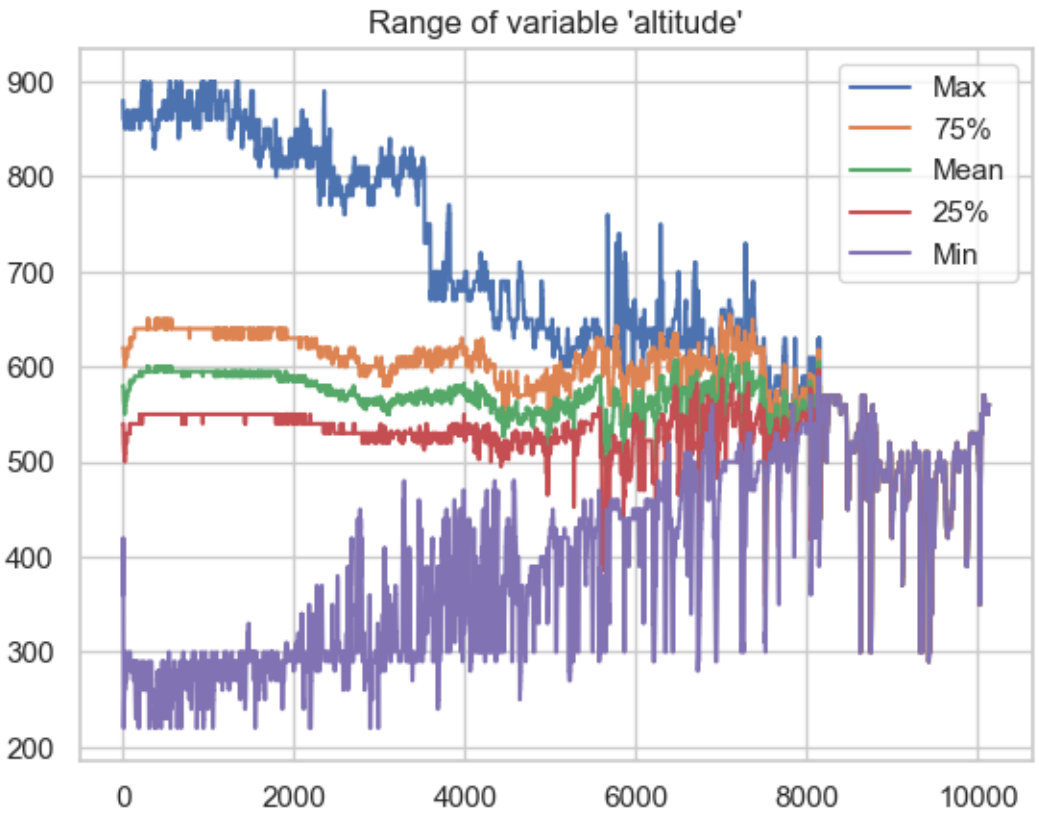
[21]:
```
def plot_ranges(variable):
    data = df[variable].apply(pd.Series)
    quantiles = data.quantile([0.25, 0.75], axis=0)
    plt.plot(data.max(axis=0), label='Max')
    plt.plot(quantiles.iloc[1], label='75%')
    plt.plot(quantiles.mean(axis=0), label='Mean')
    plt.plot(quantiles.iloc[0], label='25%')
    plt.plot(data.min(axis=0), label='Min')
    plt.legend(loc="upper right")
    plt.title(f"Range of variable '{variable}'")
    plt.show()
```

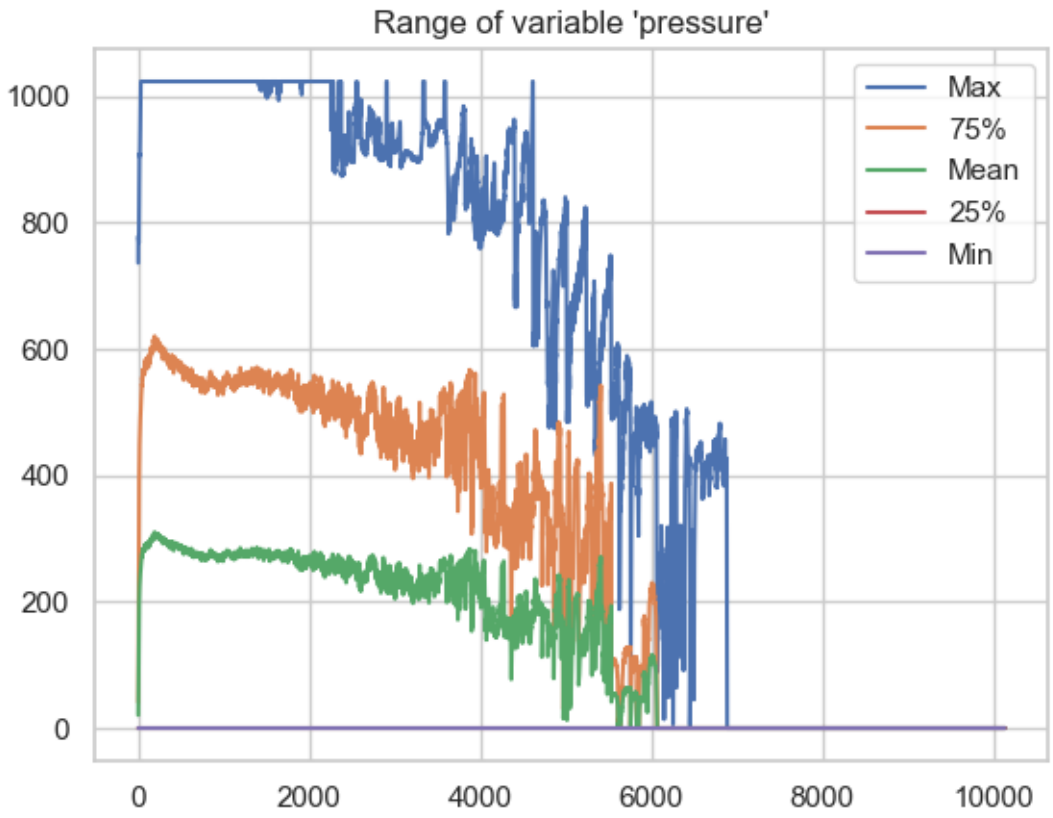
[22]:
```
for x in numerical_columns:
    plot_ranges(x)
```




Range of variable 'y'
Max
75%
Mean
25%
Min
40000
35000
30000
25000
20000
15000
10000
5000
0
0
2000
4000
6000
8000
10000


Range of variable 'azimuth'
Max
75%
Mean
25%
Min
3500
3000
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000


Range of variable 'altitude'
Max
75%
Mean
25%
Min
900
800
700
600
500
400
300
200
0
2000
4000
6000
8000
10000



Podemos obtener los valores máximos y mínimos para las señales, en especial las señales X y Y tienen valores del orden de decenas de miles pues representan coordenadas. Es probable que debamos normalizar las señales para que sus valores queden entre 0 y 1 para tenerlos listos para los algoritmos de clasificación basados en redes neuronales recurrentes. Por otro lado, si lo que se quiere es utilizar las imágenes generadas por las coordenadas será necesario mantenerlas tal y como están. En el preprocesamiento se utilizan ambas alternativas, se normalizan las señales y se dejan las originales X y Y disponibles de igual forma.

[23]:
```
for var_name in numerical_columns:
    print(f"Max '{var_name}':", df[var_name].apply(pd.Series).max().max())
    print(f"Min '{var_name}':", df[var_name].apply(pd.Series).min().min())
    print("\n")
```

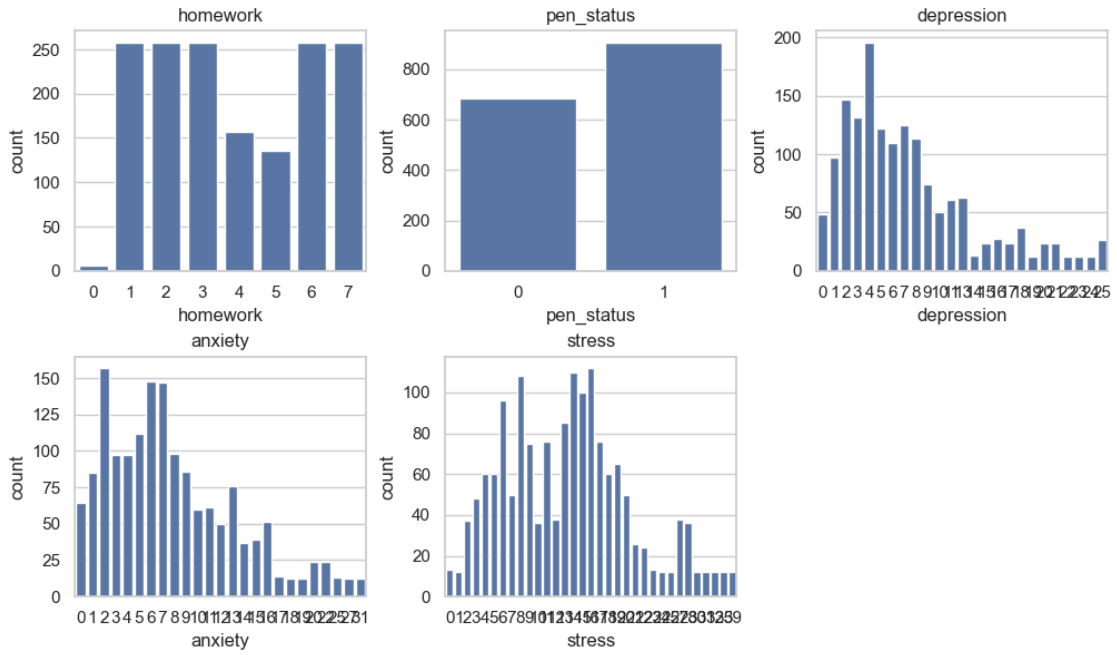
```
Max 'x': 61504.0
Min 'x': 2215.0


Max 'y': 40630.0
Min 'y': 10.0


Max 'azimuth': 3590.0
```

```
Min 'azimuth': 0.0


Max 'altitude': 900.0
Min 'altitude': 220.0


Max 'pressure': 1023.0
Min 'pressure': 0.0
```

1.4.3 Visualizaciones para Datos Categóricos

Gráfico de conteo

[24]:
```
make_countplot(df, categorical_columns, fig_size=(10,8))
```



[25]:
```
for column in categorical_columns:
    print("Para {} se tienen {} valores únicos".format(column, df[column].
 ↪nunique()))
    print("Que son los siguientes: \n{}\n".format(df[column].value_counts()))
```

```
Para homework se tienen 8 valores únicos
Que son los siguientes:
homework
1    258
2    258
```

```
3    258
6    258
7    258
4    157
5    135
0      6
Name: count, dtype: int64

Para pen_status se tienen 2 valores únicos
Que son los siguientes:
pen_status
1    906
0    682
Name: count, dtype: int64

Para depression se tienen 25 valores únicos
Que son los siguientes:
depression
4     196
2     147
3     132
7     125
5     122
8     113
6     110
1      97
9      74
13     63
11     61
10     50
0      49
18     37
16     28
25     27
17     24
20     24
21     24
15     24
14     13
24     12
23     12
22     12
19     12
Name: count, dtype: int64

Para anxiety se tienen 25 valores únicos
Que son los siguientes:
anxiety
```

```
2     157
6     148
7     147
5     112
8      98
4      97
3      97
9      86
1      85
13     76
0      64
11     61
10     60
16     51
12     50
15     39
14     37
20     24
22     24
17     14
25     13
19     12
31     12
18     12
27     12
Name: count, dtype: int64

Para stress se tienen 33 valores únicos
Que son los siguientes:
stress
16    112
14    110
8     108
15    100
6      96
13     85
11     76
17     76
9      75
19     65
5      60
18     60
4      60
20     50
7      50
3      48
12     38
27     38
```

```
2      37
10     36
28     36
21     26
22     24
23     13
0      13
1      12
31     12
32     12
30     12
39     12
35     12
25     12
24     12
Name: count, dtype: int64
```

Descripción de los Countplots

1. `homework`:
   - **Valores únicos**: 8
   - **Distribución de los valores**:
     - Los valores 1, 2, 3, 6, y 7 tienen el mismo recuento (258), lo que sugiere una distribución uniforme para estos niveles.
     - El valor 4 tiene un recuento menor (157), mientras que el 5 (135) y el 0 (6) son valores poco frecuentes.
   - **Interpretación del Countplot**:
     - Este gráfico mostrará barras significativamente más altas para los valores más comunes (1, 2, 3, 6, 7) y barras mucho más pequeñas para los valores menos comunes (4, 5, 0).
     - Los valores altos indican que la mayoría de los datos están concentrados en ciertos niveles de tarea (`homework`).
2. `pen_status`:
   - **Valores únicos**: 2 (0 y 1)
   - **Distribución de los valores**:
     - El valor 1 tiene un recuento de 906, mientras que el 0 tiene 682.
   - **Interpretación del Countplot**:
     - Este gráfico mostrará solo dos barras, con la barra correspondiente al valor 1 siendo más alta que la del valor 0.
     - Esto sugiere que el estado 1 de la pluma ocurre con mayor frecuencia que el estado 0.
3. `depression`:
   - **Valores únicos**: 25

- **Distribución de los valores**:
  - Los valores más comunes son 4 (196), 2 (147), y 3 (132).
  - Los valores menos comunes son los últimos de la lista, como 24, 23, 22, 19, y 25, todos con un recuento de 12.
- **Interpretación del Countplot**:
  - Este gráfico mostrará una alta variabilidad entre las barras. Las barras correspondientes a los valores 4, 2, y 3 serán las más altas, mientras que habrá muchas barras pequeñas para valores menos frecuentes.
  - La distribución indica que los niveles de depresión están dispersos en varios niveles, pero ciertos valores son más frecuentes.

4. `anxiety`:
   - **Valores únicos**: 25
   - **Distribución de los valores**:
     - Los valores más comunes son 2 (157), 6 (148), y 7 (147).
     - Los valores menos comunes incluyen 25, 19, 31, 18, y 27, todos con un recuento de 12 o menor.
   - **Interpretación del Countplot**:
     - Similar al caso de `depression`, el gráfico tendrá barras altas para los valores más comunes y barras mucho más bajas para los menos frecuentes.
     - La distribución sugiere una alta variabilidad en los niveles de ansiedad, con una concentración en algunos valores específicos.
5. `stress`:
   - **Valores únicos**: 33
   - **Distribución de los valores**:
     - Los valores más comunes son 16 (112), 14 (110), y 8 (108).
     - Los valores menos comunes incluyen 1, 31, 32, 30, 39, 35, 25, y 24, todos con un recuento de 12 o menor.
   - **Interpretación del Countplot**:
     - Este gráfico mostrará barras muy altas para los valores más comunes, con una disminución gradual en la altura de las barras hacia los valores menos comunes.
     - La amplia cantidad de valores únicos indica una distribución más dispersa y variada en los niveles de estrés.

Observaciones Generales

- Las variables como `homework` y `pen_status` tienen pocos valores únicos, lo que da lugar a gráficos más simples y menos variables.
- Las variables `depression`, `anxiety`, y `stress` tienen una mayor cantidad de valores únicos, lo que se traduce en gráficos más complejos con una distribución más dispersa.

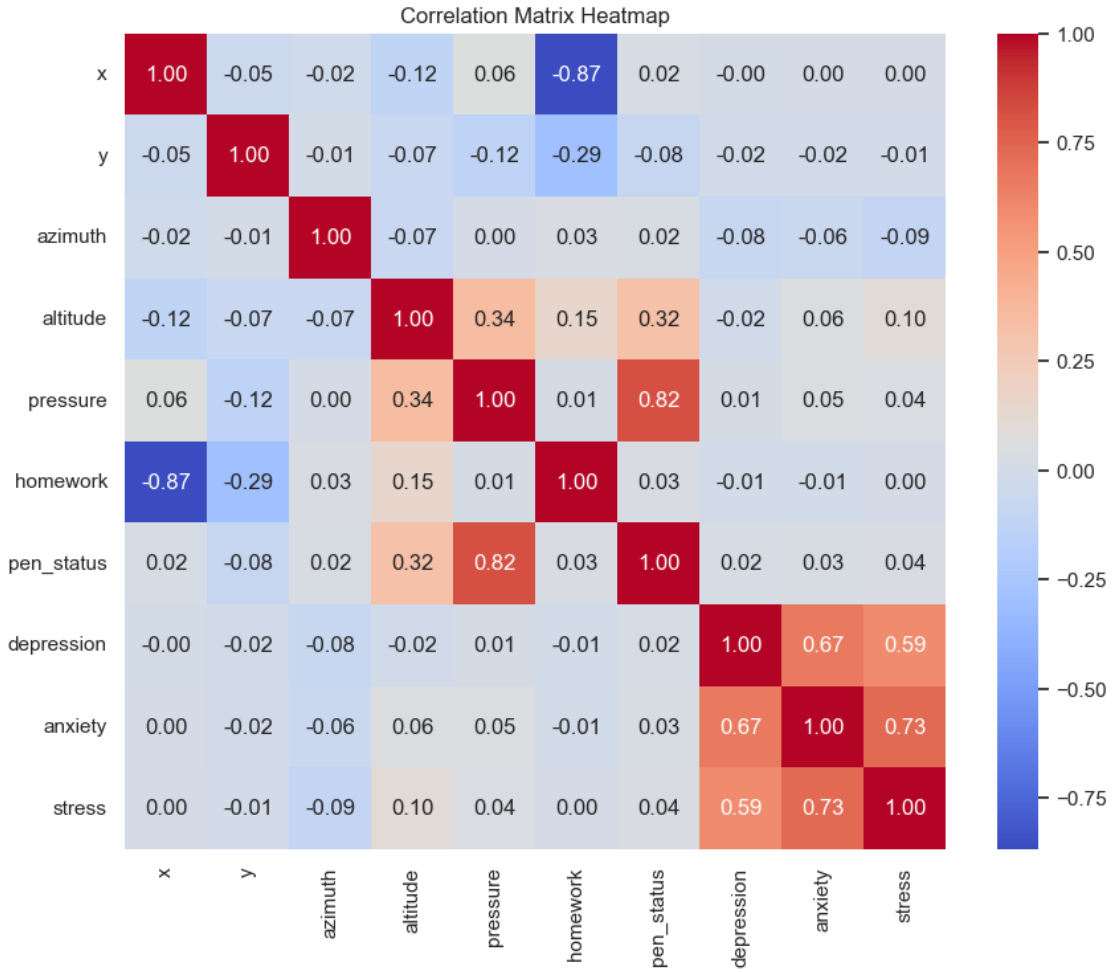
1.5 Análisis Bivariado y Multivariado

1.5.1 Análisis de Correlación para Variables Numéricas

Matriz de Correlación Para incluir en la matriz de correlación las variables categóricas haremos one-hot encoding de las etiquetas.

[26]:
```
df_raw['depression'] = df_raw['depression'].astype(int)
df_raw['anxiety'] = df_raw['anxiety'].astype(int)
df_raw['stress'] = df_raw['stress'].astype(int)
```

[27]:
```
# Compute correlation coefficient
correlation = df_raw[numerical_columns + categorical_columns].corr()

# Visualize the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix Heatmap")
plt.show()
```



Descripción de la matriz de correlación

- La matriz muestra las correlaciones entre distintas variables del conjunto de datos. Las correlaciones más fuertes, positivas o negativas, se destacan con colores más cercanos al rojo o azul respectivamente, mientras que las correlaciones cercanas a 0 son más claras.
- **Observaciones destacadas:**
  - Existe una correlación negativa fuerte entre `x` y `homework` (-0.87).
  - La variable `pen_status` tiene una correlación positiva alta con `pressure` (0.82).
  - Variables relacionadas con emociones, como `depression`, `anxiety` y `stress`, tienen correlaciones moderadas entre sí (valores entre 0.59 y 0.73).
  - La mayoría de las correlaciones entre variables como `azimuth`, `altitude`, `pressure` y las emocionales son bajas o cercanas a 0, lo que indica poca relación entre ellas.

Nota importante: Dado que se trata de un análisis aplicado a series de tiempo, estas correlaciones deben interpretarse con precaución. Las relaciones identificadas en esta matriz podrían no ser relevantes debido a la naturaleza temporal de los datos. En lugar de este análisis estático, podría ser más apropiado utilizar técnicas específicas para series de tiempo, como autocorrelación o análisis de dependencia temporal.

Pairplots No es recomendable utilizar sns.pairplot para analizar series de tiempo, ya que está diseñado para visualizar relaciones entre variables estáticas (no dependientes del tiempo). Las series de tiempo tienen una estructura secuencial donde el orden de los datos es crucial, y un pairplot no captura esta dependencia temporal.

1.6 Visualización gráfica de las tareas

[28]:
```
# Expandir columnas con listas en filas separadas
df_expanded = df.explode(['x', 'y', 'timestamp', 'azimuth', 'altitude',␣
 ↪'pressure']).reset_index()

# Convertir columnas expandidas a numéricas
for col in ['x', 'y', 'timestamp', 'azimuth', 'altitude', 'pressure']:
    df_expanded[col] = pd.to_numeric(df_expanded[col])

# Mostrar las primeras filas del conjunto de datos expandido
df_expanded.head()
```

[28]:

	Subject	homework	pen_status	x	y	timestamp	azimuth	altitude	\
0	1	1	0	48331	31876	672620	1830	530	
1	1	1	0	48318	31963	672628	1830	530	
2	1	1	0	48305	32053	672635	1830	530	
3	1	1	0	48305	32159	672643	1830	530	
4	1	1	0	48305	32159	672650	1830	530	

	pressure	depression	anxiety	stress
0	0	2	8	13
1	0	2	8	13

```
2          0          2          8       13
3          0          2          8       13
4          0          2          8       13
```

[29]:
```python
def plot_subject_data(subject_id, pen_status, homework_id, x_size=5, y_size=5):
    """
    Función para graficar datos de dispersión para un sujeto específico,
    con un estado de lápiz y una tarea específicos.

    Parámetros:
    subject_id (int): ID del sujeto.
    pen_status (int): Estado del lápiz (activo/inactivo).
    homework_id (int): ID de la tarea.
    """
    # Filtrar los datos
    sample_subject_data = df_expanded[(df_expanded['Subject'] == subject_id) &
                                      (df_expanded['pen_status'] == pen_status)␣
 ↪&
                                      (df_expanded['homework'] == homework_id)]

    # Verificar si los datos filtrados no están vacíos
    if not sample_subject_data.empty:
        # Crear el gráfico de dispersión
        plt.figure(figsize=(x_size, y_size))
        plt.scatter(-sample_subject_data['y'], sample_subject_data['x'], s=3,␣
 ↪c='blue', alpha=0.6)
        plt.title(f"Gráfico de Dispersión: Sujeto {subject_id}, Estado del␣
 ↪Lápiz {pen_status}, Tarea {homework_id}")
        plt.xlabel("Posición -y")
        plt.ylabel("Posición x")
        plt.grid(True)
        plt.show()
    else:
        print(f"No se encontraron datos para Sujeto {subject_id}, Estado del␣
 ↪Lápiz {pen_status}, Tarea {homework_id}.")
```

[30]:
```python
for homework_id in range(1, 7):
    plot_subject_data(subject_id=1, pen_status=1, homework_id=homework_id)

plot_subject_data(subject_id=1, pen_status=1, homework_id=7, x_size=12,␣
 ↪y_size=2)
```

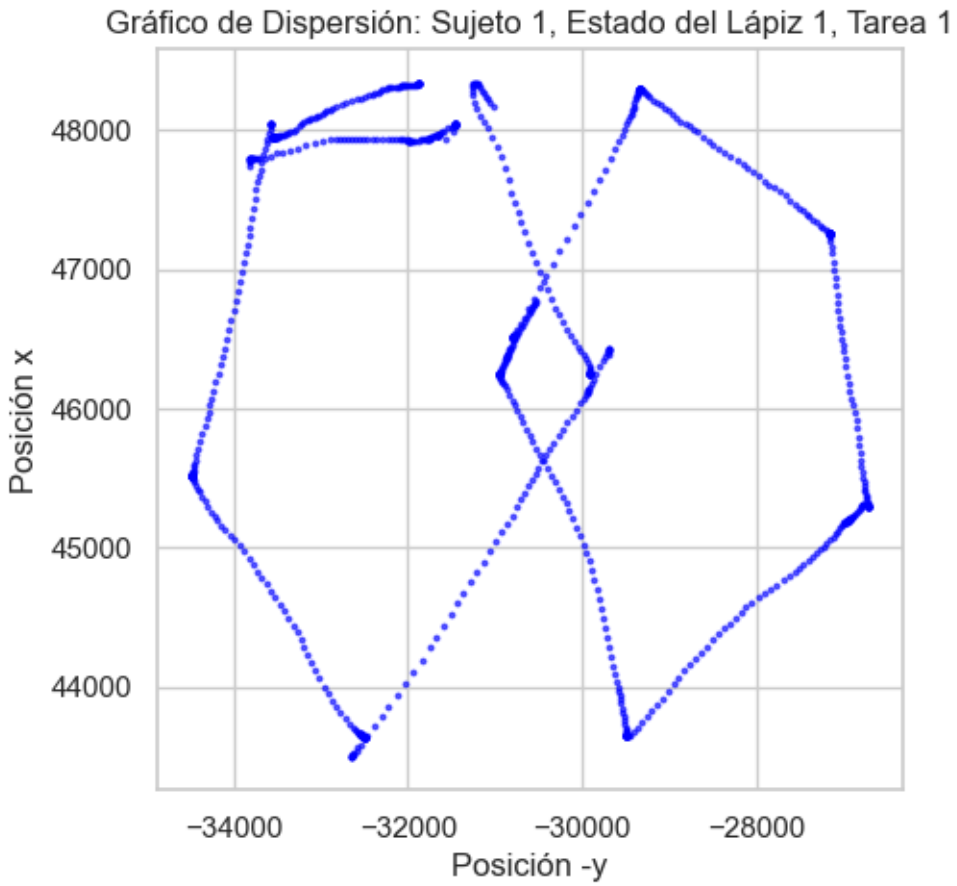


Gráfico de Dispersión: Sujeto 1, Estado del Lápiz 1, Tarea 1
Posición x
48000
47000
46000
45000
44000
−34000
−32000
−30000
−28000
Posición -y

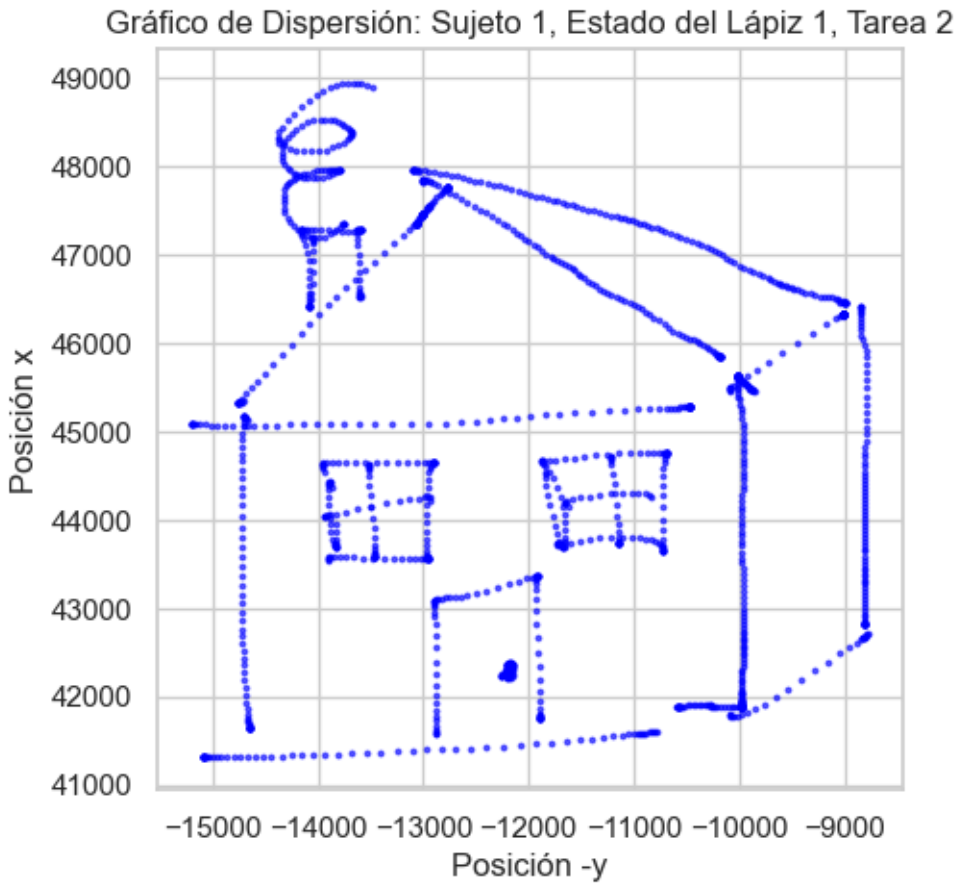


Gráfico de Dispersión: Sujeto 1, Estado del Lápiz 1, Tarea 2
Posición x
49000
48000
47000
46000
45000
44000
43000
42000
41000
−15000 −14000 −13000 −12000 −11000 −10000 −9000
Posición -y

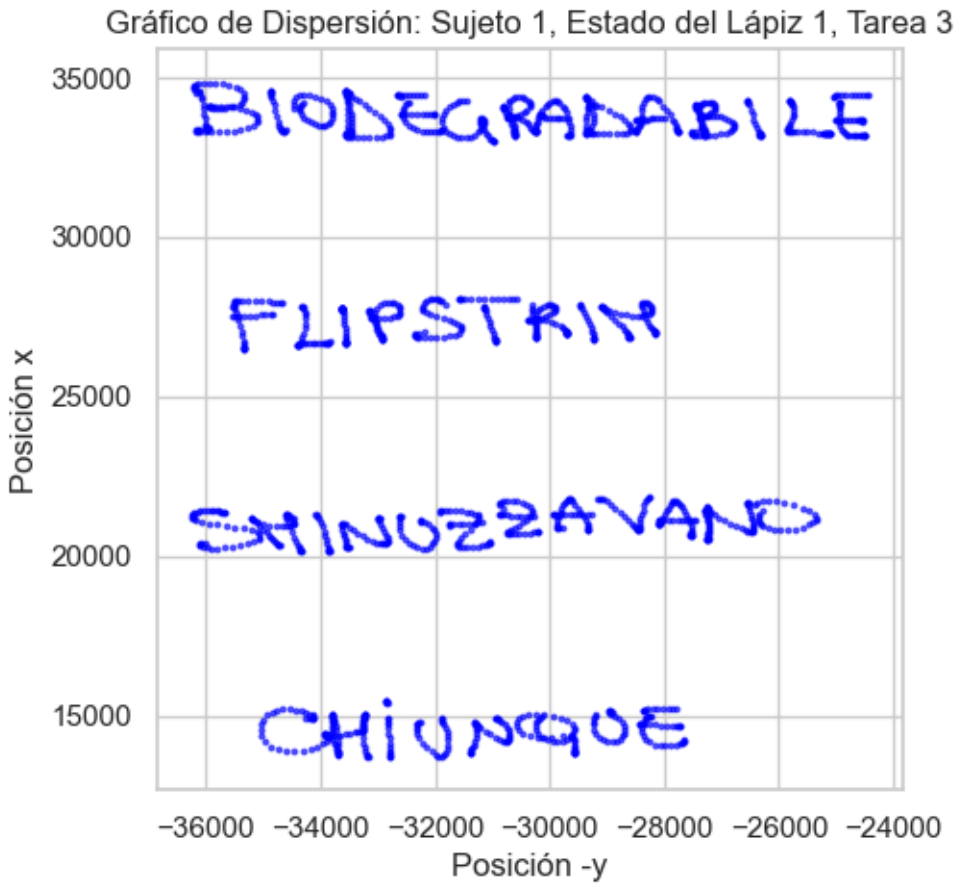


Gráfico de Dispersión: Sujeto 1, Estado del Lápiz 1, Tarea 3
Posición x
35000
30000
25000
20000
15000
BIODEGRADABILE
FLIPSTRIP
SMINUZZAVANO
CHIUNQUE
−36000
−34000
−32000
−30000
−28000
−26000
−24000
Posición -y

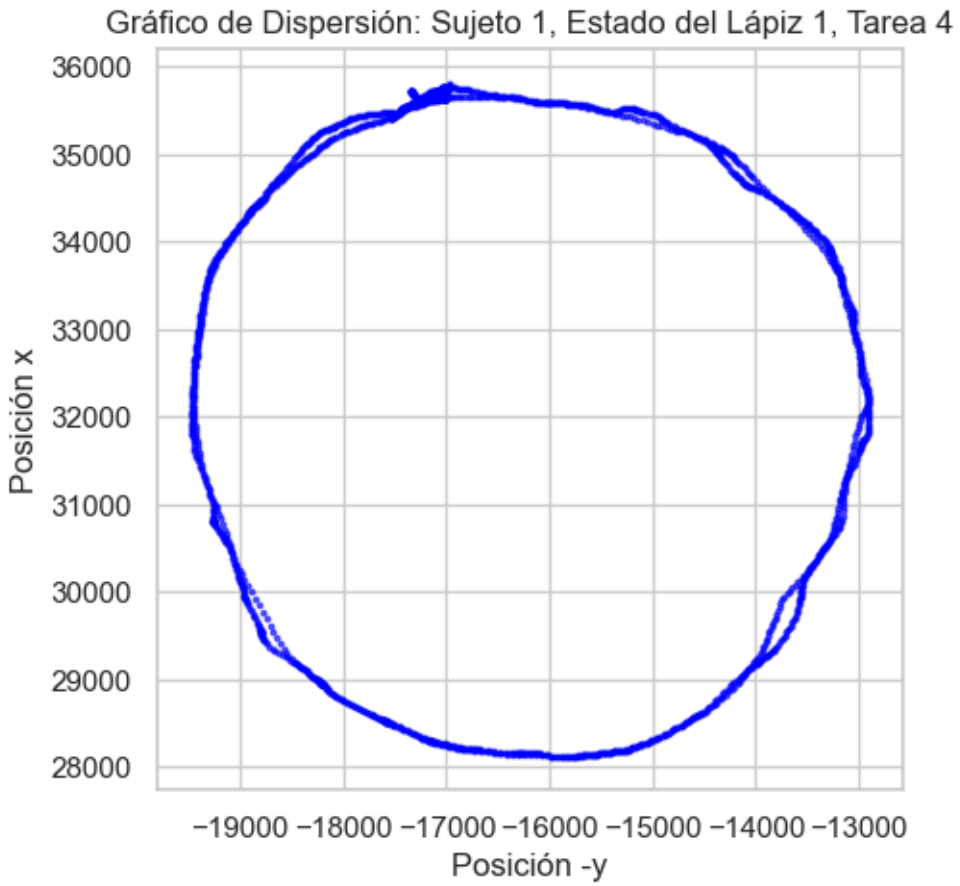


Gráfico de Dispersión: Sujeto 1, Estado del Lápiz 1, Tarea 4
36000
35000
34000
33000
32000
31000
30000
29000
28000
Posición x
−19000 −18000 −17000 −16000 −15000 −14000 −13000
Posición -y

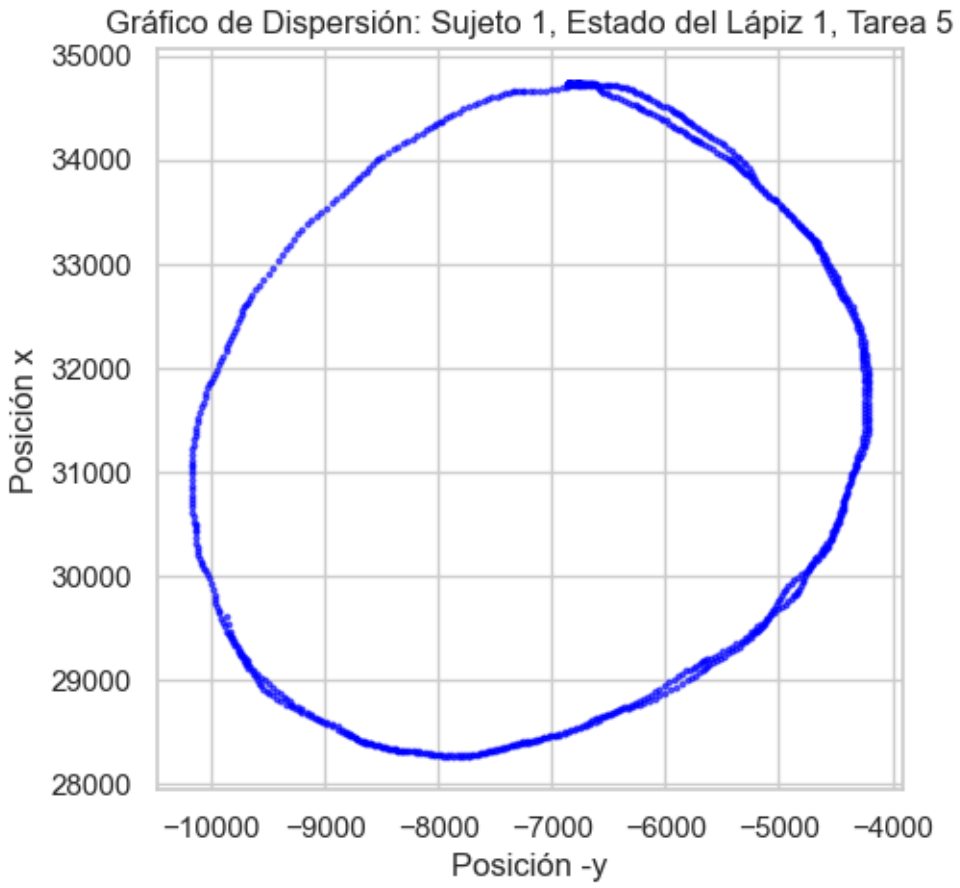
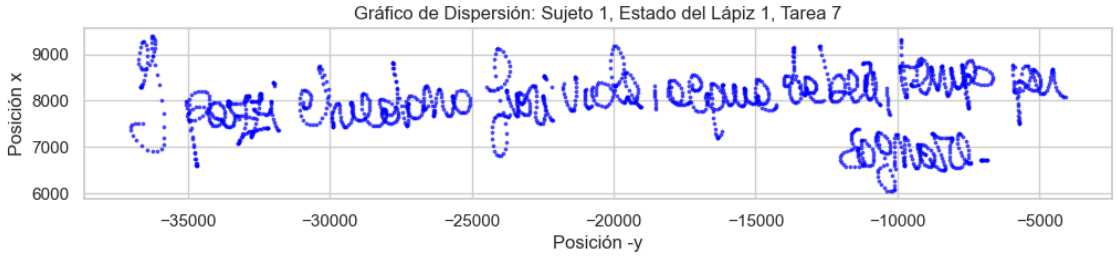


Gráfico de Dispersión: Sujeto 1, Estado del Lápiz 1, Tarea 5
Posición x
35000
34000
33000
32000
31000
30000
29000
28000
−10000
−9000
−8000
−7000
−6000
−5000
−4000
Posición -y



1.7 Preprocesamiento de los datos

1.7.1 Manejo de Valores Faltantes

No hay datos faltantes en el conjunto de datos. Nuevamente, como se había mencionado, no hay necesidad de realizar alguna técnica o algoritmo para datos faltantes en nuestro set de datos.

1.7.2 Tratamiento de variables categóricas

Las variables categóricas 'homework' y 'pen_status' no representan características que puedan aportar valor predictivo sobre las situaciones de depresión, ansiedad o estrés. Más bien representan metadatos o información que permite filtrar los datos para su gestión. Las variables como 'homework' y 'pen_status' tienen baja cardinalidad lo que implicaría el uso de one-hot encoding, sin embargo no serán preprocesadas pues serán utilizadas como método de filtrado no para el entrenamiento de los algoritmos.

Por otro lado las etiquetas como 'depression', 'anxiety' y 'stress' ya se encuentran codificadas en forma de one-hot encoding desde que los datos se preparon con base en el paper Nolazco-Flores, et al. (2021).

1.7.3 Tratamiento de variables numéricas

En el caso de las variables numéricas de series de tiempo, éstas aportan el poder predictivo. Dado que se utilizarán modelos basados en redes neuronales, las características con magnitudes mayores pueden sesgar la predicción. Por esta razón se utilizará una normalización aplicada a series de tiempo. Dado que estamos tratando con series de tiempo implementaremos un transformador de scikit-learn con la capacidad de guardar los parámetros para la reproducción en posteriores etapas.

En el caso de los outliers en este caso no se removerán pues no representan valores individuales, sino que son parte de la señales de series de tiempo y representan ángulos de azimuth y altitude fuera de lo normal, esto podría aportar valor predictivo. Lo que si será necesario es normalizar estos datos.

Ahora comparando ambos datos podemos ver los cambios dentro de los sesgos, los originales por lo general están sesgados a la izquierda, mientras que con los nuevos cambios se estandarizan más los valores, y prácticamente eliminado los valores atípicos.

[31]:
```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.compose import ColumnTransformer

class StandardScalerArray(BaseEstimator, TransformerMixin):
    '''
    Custom scikit transformer for scaling time-series using z-score, it allows␣
↪to save a rolling mean and std-dev for training dataset,
    so the whole preprocessing pipeline can be replicated during inference in␣
↪development and production. It makes it repeatable and scalable.
    '''

    def fit(self, X, y=None):
        self.column_names = list(X.columns)

        # state
        self.mean_of_means = {column_name: 0 for column_name in self.
↪column_names} # means for each feature column
        self.stddev = {column_name: 1 for column_name in self.column_names} #␣
↪mean stddev for each feature column
```

```
        self.samples_count = {column_name: 1 for column_name in self.
↪column_names} # count for each column
        return self

    def transform(self, X, y=None):
        # perform z-score transformation

        return X.apply(self._batch_normalize)

    def _normalize_series(self, window, column_name):
        mean = window.mean()
        stddev = window.std() + 1E-9 # avoid division by zero

        samples_count = self.samples_count[column_name]

        self.mean_of_means[column_name] = (self.mean_of_means[column_name] +␣
↪samples_count * mean)/(samples_count + 1)
        self.stddev[column_name] = (self.stddev[column_name] + samples_count *␣
↪stddev)/(samples_count + 1)
        self.samples_count[column_name] += 1

        return (window - self.mean_of_means[column_name]) / self.
↪stddev[column_name]

    def _batch_normalize(self, column):
        return column.map(lambda col_val: self._normalize_series(col_val,␣
↪column.name))

    def get_feature_names_out(self, column_names):
        return self.column_names


std_scaler_array = StandardScalerArray()

col_transformer = ColumnTransformer([
    ('normalization', std_scaler_array, numerical_columns)
], remainder='passthrough')
col_transformer
```

[31]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('normalization', StandardScalerArray(),
                                 ['x', 'y', 'azimuth', 'altitude',
                                  'pressure'])])
```

Previo a la transformación haremos una división de los datos en entrenamiento, validación y prueba para evitar el filtrado de información. Se utilizarán 70%, 15% y 15% de los datos respectivamente. Se utilizará la estratificación a partir de las variables de 'depression', 'anxiety' y 'stress'.

[32]:
```
from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(df.
 ↪drop(columns=['timestamp', 'depression', 'anxiety', 'stress']),␣
 ↪df[['depression','anxiety','stress']], test_size=0.7, random_state=42,␣
 ↪stratify=df[['depression','anxiety','stress']])

X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,␣
 ↪random_state=42, stratify=y_temp[['depression','anxiety','stress']])
```

Almacenamos los conjuntos generados para su reutilización posterior

[33]:
```
X_train.to_parquet('../data/raw_binary/X_train.parquet')
y_train.to_parquet('../data/raw_binary/y_train.parquet')
X_val.to_parquet('../data/raw_binary/X_val.parquet')
y_val.to_parquet('../data/raw_binary/y_val.parquet')
X_test.to_parquet('../data/raw_binary/X_test.parquet')
y_test.to_parquet('../data/raw_binary/y_test.parquet')
```

[34]:
```
df_normalized = col_transformer.fit_transform(X_train)
column_names = [x.split('__')[-1] for x in col_transformer.
 ↪get_feature_names_out()] # get column names back again
X_train_normalized = pd.DataFrame(df_normalized, columns=column_names) #␣
 ↪transformer generates matrix, convert back to dataframe
```

[35]:
```
X_train_normalized
```

[35]:

	x	y
0	[26.910060356237743, 26.905685563160194, 26.90…	[10.671745482158714, 10.674048820192342, 10.67…
1	[2.95639838169674, 3.0278374332621243, 3.06071…	[0.36987054450271983, 0.35181113692246363, 0.3…
2	[1.1926824044512865, 1.1931867901885518, 1.193…	[4.550762397186965, 4.549722188504138, 4.54972…
3	[-2.3879606076456508, -2.3886637254750274, -2…	[1.7899101877625243, 1.792889687326394, 1.7937…
4	[1.7299819193471089, 1.724431718920416, 1.7170…	[1.486145397046375, 1.4876698841941138, 1.4910…
..	…	…
471	[1.6956188930900546, 1.706458768339123, 1.7157…	[0.3756324148991576, 0.3756324148991576, 0.378…
472	[-0.14184899985457786, -0.1390578482657065, -0…	[1.9697103962977114, 1.973577399067467, 1.9766…
473	[1.2854438256736287, 1.2818587033501556, 1.278…	
474	[1.7965111035491697, 1.7971104210960598, 1.797…	
475	[-0.1565466393729467, -0.20507956978168057, -0…	

```
473  [0.07717319811453098, 0.09376686105605704, 0.1…
474  [0.04161635340722035, 0.04849837180626974, 0.0…
475  [1.6139285002830568, 1.6128196319043855, 1.611…

                                                azimuth  \
0    [20.333107779562777, 20.333107779562777, 20.33…
1    [-0.899540116397932, -0.899540116397932, -0.88…
2    [1.3795236996122597, 1.3795236996122597, 1.379…
3    [-0.39337624272941624, -0.290696178199719, -0…
4    [0.7387940354191571, 1.0619386600960996, 1.223…
..                                                 …
471  [0.5361316685827144, 0.5361316685827144, 0.612…
472  [-0.4429866643193005, -0.4429866643193005, -0…
473  [-0.09914145324613184, -0.09914145324613184, -…
474  [1.494209492027911, 1.576705396268149, 1.74169…
475  [1.918780790209049, 1.918780790209049, 1.91878…

                                                altitude  \
0    [21.479097980238258, 21.479097980238258, 21.47…
1    [1.3918213107285964, 1.8280130723183146, 2.264…
2    [-1.8994255564094138, -1.8994255564094138, -1…
3    [-2.713769646533836, -3.0284561175913876, -3.0…
4    [-1.4044721120234538, -1.4044721120234538, -1…
..                                                 …
471  [-1.7596965378745975, -1.7596965378745975, -1…
472  [-2.761127287125797, -2.256459432911323, -2.25…
473  [-1.5566070179950255, -1.5566070179950255, -1…
474  [-2.548789090165937, -2.548789090165937, -2.54…
475  [-3.600055072286061, -3.600055072286061, -3.60…

                                                pressure homework pen_status
0    [-3.004369491749324, -2.6542682341414325, -2.4…        2          1
1    [-1.7905962187429496, -1.7046047392459542, -2…        4          1
2    [-3.0213305401590023, -2.6708683749523585, -2…        3          1
3    [-3.3869508114599407, -3.143088578695064, -2.9…        7          1
4    [-1.8039092464848987, -1.5788813637950239, -1…        3          1
..                                                 …      …          …
471  [-3.279311790992276, -2.379999900934799, -2.15…        6          1
472  [-2.799228318812496, -2.614170184678273, -2.42…        6          1
473  [-5.324118461721956, -5.0763720151379905, -5.0…        5          1
474  [-2.0186028883834255, -1.3734381308622734, -1…        6          1
475  [-3.0158077211264906, -2.545673686607898, -2.1…        7          1

[476 rows x 7 columns]
```

A continuación se muestran las señales normalizadas.

[36]:
```
plot_random_signals(X_train_normalized, 'x', 'Normalized X signals for 3 random␣
↪samples of homework 1', homework=1, samples=3)
```

[37]:
```
plot_random_signals(X_train_normalized, 'y', 'Normalized X signals for 3 random␣
↪samples of homework 1', homework=1, samples=3)
```

1.7.4 Aplicación de transformación wavelet (WDT)

[38]:
```
class WaveletTransformer(BaseEstimator, TransformerMixin):
    '''
    Custom scikit transformer for wavelet transformation of time-series data.
    '''

    def _extract_wavelet_features(self, data_list, wavelet='db4', level=10):
        """
        Aplicar la Transformada Wavelet a una lista de datos y extraer␣
↪características.

        Parámetros:
        data_list = lista de datos a usar.
        wavelet = transformacion wavelet a usar (Daubechies 4 wavelet)
        level = número de veces que se realiza el proceso de descomposición en␣
↪una señal o imagen
        """
        coeffs = pywt.wavedec(data_list, wavelet, level=level)
        features = []
        for coef in coeffs:
            features.extend([
                np.mean(coef),
                np.std(coef),
                np.min(coef),
                np.max(coef)
            ])
        return features

    def fit(self, X, y=None):
        # stateless transformer
        self.column_names = list(X.columns)

        return self

    def transform(self, X, y=None):
        return X.apply(self._batch_process)

    def _apply_transformation(self, window, column_name):
        return self._extract_wavelet_features(window)
```

```
    def _batch_process(self, column):
        return column.map(lambda col_val: self._apply_transformation(col_val,␣
 ↪column.name))

    def get_feature_names_out(self, column_names):
        return self.column_names


wavelet_transformer = WaveletTransformer()

col_transformer = ColumnTransformer([
    ('wavelet', wavelet_transformer, numerical_columns)
], remainder='passthrough')
col_transformer
```

[38]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('wavelet', WaveletTransformer(),
                                 ['x', 'y', 'azimuth', 'altitude',
                                  'pressure'])])
```

[39]:
```
df_wavelet = col_transformer.fit_transform(X_train)
column_names = [x.split('__')[-1] for x in col_transformer.
 ↪get_feature_names_out()] # get column names back again
X_train_wavelet = pd.DataFrame(df_wavelet, columns=column_names) # transformer␣
 ↪generates matrix, convert back to dataframe
```

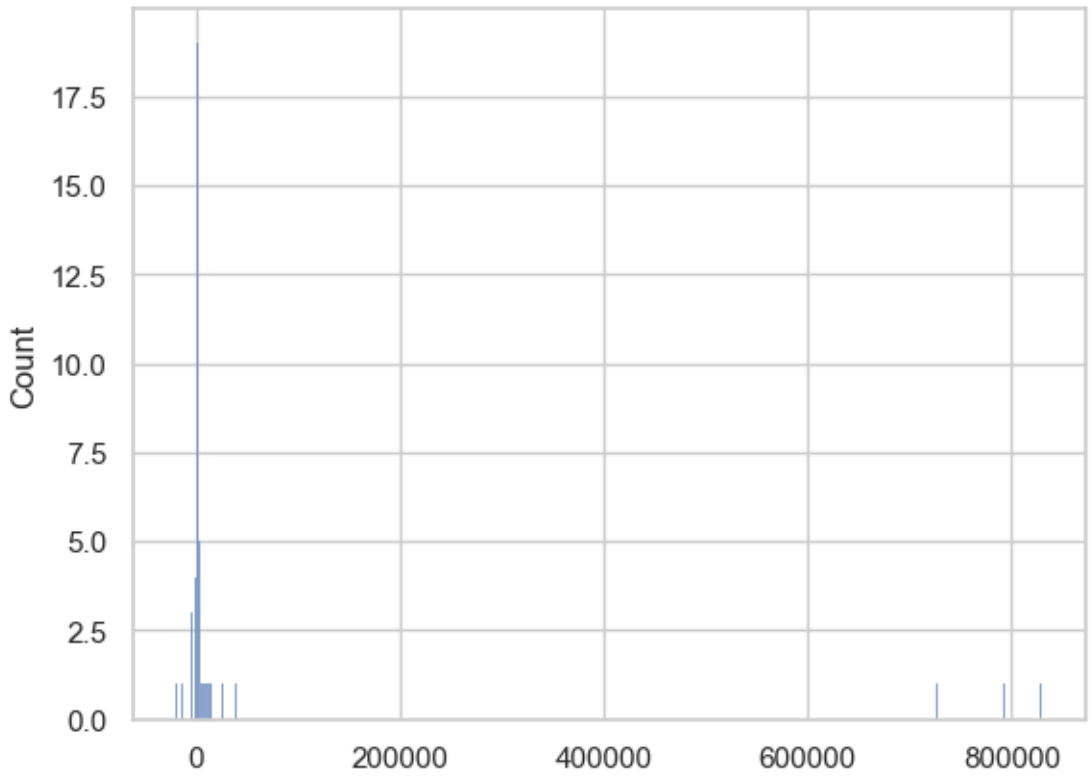
```
C:\Users\arman\AppData\Local\pypoetry\Cache\virtualenvs\proyecto-integrador-
equipo-11-WxcN3L0e-py3.12\Lib\site-packages\pywt\_multilevel.py:43: UserWarning:

Level value of 10 is too high: all coefficients will experience boundary
effects.
```

Ejemplo de aplicación wavelet a la columna de series de tiempo X

Se puede observar que las señales han pasado del dominio del tiempo a una representación en componentes. Esto será útil para las técnicas de clasificación que utilizaremos más adelante, sin embargo será necesario ajustar los parámetros de la conversión conforme se haga más progreso en el proyecto.

[40]:
```
sns.histplot(X_train_wavelet['x'].iloc[451])
```

[40]:
```
<Axes: ylabel='Count'>
```



1.8 Preguntas a contestar despues de abordar el EDA

1. Valores faltantes
- **Pregunta**: ¿Hay valores faltantes en el conjunto de datos? ¿Se pueden identificar patrones de ausencia?
- **Respuesta**: No hay valores faltantes en el conjunto de datos. Esto se verificó utilizando el método `df.isnull().sum()`, el cual confirmó que todas las columnas contienen datos completos.

---

2. Estadísticas resumidas
- **Pregunta**: ¿Cuáles son las estadísticas resumidas del conjunto de datos?
- **Respuesta**: Las estadísticas descriptivas muestran lo siguiente: - Las coordenadas `x` e `y` tienen valores que oscilan entre 0 y 60,000. - `Azimuth` y `Altitude` presentan distribuciones más estrechas. - `Pressure` tiene valores mayoritariamente concentrados cerca de 0.

---

3. Valores atípicos
- **Pregunta**: ¿Hay valores atípicos en el conjunto de datos?
- **Respuesta**: Se identificaron algunos valores atípicos, especialmente en las variables `pressure` y las etiquetas (`depression`, `anxiety`, `stress`). Estos valores fueron tratados mediante normalización durante el preprocesamiento.

---

4. Cardinalidad de las variables categóricas
- **Pregunta**: ¿Cuál es la cardinalidad de las variables categóricas?
- **Respuesta**:
- `homework`: 8
- `pen_status`: 2
- `depression`: 25
- `anxiety`: 25
- `stress`: 33

---

5. Distribuciones sesgadas
- **Pregunta**: ¿Existen distribuciones sesgadas en el conjunto de datos?
- **Respuesta**:
- Las etiquetas (`depression`, `anxiety`, `stress`) presentan distribuciones sesgadas, con algunos valores mucho más frecuentes que otros.
- `pen_status` tiene una distribución binaria balanceada.

---

6. Tendencias temporales
- **Pregunta**: ¿Se identifican tendencias temporales?
- **Respuesta**:
- Las series de tiempo para `x` y `y` muestran patrones similares dentro de cada tarea, aunque existe variabilidad entre diferentes usuarios.
- Las señales pueden considerarse ruidosas.

---

7. Análisis bivariado
- **Pregunta**: ¿Cómo se distribuyen los datos en función de diferentes categorías?
- **Respuesta**:
- Las etiquetas (`depression`, `anxiety`, `stress`) muestran correlaciones moderadas entre sí.
- Las demás variables tienden a ser más independientes entre sí.

---

8. Normalización para visualización
- **Pregunta**: ¿Se deberían normalizar las imágenes para visualizarlas mejor?
- **Respuesta**: No es necesario, ya que el análisis se centra en series de tiempo en lugar de imágenes.

---

9. Desequilibrio en las clases
- **Pregunta**: ¿Hay desequilibrio en las clases de la variable objetivo?
- **Respuesta**: Sí, hay un ligero desequilibrio entre las etiquetas (`depression`, `anxiety`, `stress`). Este desequilibrio será tomado en cuenta al entrenar los modelos.

1.9 Conclusiones

Después de realizar el análisis exploratorio, se identificaron puntos clave:

- **Tareas (`homework`)**:
  Varían de 0 a 7. La mayoría de las muestras corresponden a tareas bien clasificadas.
- **Pluma (`pen_status`)**:
  Presenta una distribución ligeramente inclinada hacia el estado activo (`1`).
- **Etiquetas (`depression`, `anxiety`, `stress`)**:
  Existe un sesgo en las distribuciones, con valores más frecuentes en ciertos rangos.
- **Series de Tiempo (`x`, `y`)**:
  Muestran tendencias similares en usuarios realizando la misma tarea, aunque varían entre individuos.
- **Outliers**:
  Identificados principalmente en las etiquetas y en las series temporales (`azimuth`, `altitude`). Estos serán manejados mediante normalización.

Implicaciones importantes de los hallazgos:

- **Preprocesamiento**:
  - Fue esencial normalizar las series temporales.
  - Como se comentó anteriormente, los outliers en este caso no se removerán pues no representan valores individuales, sino que son parte de la señales de series de tiempo y representan ángulos de azimuth y altitude fuera de lo normal, esto podría aportar valor predictivo.
  - Para la parametrización de los datos aplicando la transformada de Wavelet, como se comentó anteriormente, será necesario ajustar los parámetros de la conversión conforme se haga más progreso en el proyecto.
- **Entrenamiento de Modelos**:
  - Se debe considerar si el ligero equilibrio en las etiquetas al diseñar el pipeline de aprendizaje tiene un impacto en su desempeño.

Con base en este análisis, estamos mejor preparados para continuar con un preprocesamiento más detallada a través de continuar con el siguiente paso de realizar el **Feature Engineering** (Ingeniería de Características) y seleccionar los modelos más adecuados para clasificar las emociones en el conjunto de datos EMOTHAW.

1.10 Referencias

Faundez-Zanuy, M. (2025). Comprehensive analysis of least significant bit and difference expansion watermarking algorithms for online signature signals. Expert Systems with Applications, 267, 126214. https://doi.org/10.1016/j.eswa.2024.126214

Likforman-Sulem, L., Esposito, A., Faundez-Zanuy, M., Clemençon, S., & Cordasco, G. (2017). EMOTHAW: A novel database for emotional state recognition from handwriting and drawing. IEEE Transactions on Human-Machine Systems, 47(2), 273–284. https://doi.org/10.1109/THMS.2016.2635441

Nolazco-Flores, J. A., Faundez-Zanuy, M., Velázquez-Flores, O. A., Cordasco, G., & Esposito, A. (2021). Emotional state recognition performance improvement on a handwriting and drawing task. IEEE Access, 9, 28496–28504. https://doi.org/10.1109/ACCESS.2021.3058443

San Roman, R., Fernandez, P., Défossez, A., Furon, T., Tran, T., & Elsahar, H. (2024). Proactive detection of voice cloning with localized watermarking. arXiv. https://arxiv.org/abs/2401.17264

[40]: