Classification of Blood Clot Origins in Ischemic Strokes 🩸

This notebook explores the task of classifying the etiology of blood clots in whole-slide digital pathology images, specifically identifying whether they are of Cardioembolic (CE) or Large Artery Atherosclerosis (LAA) origin. Through an extensive exploratory data analysis (EDA), we describe the dataset, analyze missing and duplicate values, examine the distribution of image sizes, classify variables, and review the label distribution for the training set, along with plenty of other analysis. This EDA provides a comprehensive understanding of the data and helps identify potential preprocessing steps for optimal model performance.

Following the EDA, we preprocess the images to standardize them for model input. The preprocessing involves resizing, converting images to grayscale, normalizing pixel values, and applying Gaussian blur to reduce noise. These steps ensure that the images are suitable for a Convolutional Neural Network (CNN) by preparing them with a consistent size, format, and reduced noise, enabling more efficient training and improved classification accuracy.

Authors:

- Daniel Valdez
- Emilio Solano
- Adrian Flores
- Andrea Ramírez

---

(1) Import Libraries ⬇️

In [ ]:
```
# Data manipulation and visualization
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
import subprocess
from PIL import Image
import tifffile as tifi
import cv2

# Set the maximum allowable pixels to a higher number.
Image.MAX_IMAGE_PIXELS = None

# Standard libraries
```

```
import warnings
warnings.filterwarnings('ignore')

# ===== ===== Reproducibility Seed ===== =====
# Set a fixed seed for the random number generator for reproducibility
random_state = 42

# Set matplotlib inline
%matplotlib inline

# Set default figure size
plt.rcParams['figure.figsize'] = (6, 4)

# Define custom color palette
palette = sns.color_palette("viridis", 12)

# Set the style of seaborn
sns.set(style="whitegrid")
```

(2) Data Upload 📄

In [ ]:

```
# Note:
# -----
# For this specific task, we are only using the training data (`train.csv`)
# The test dataset (`test.csv`) will not be used until the very end when mak
# We will train, validate, and tune your model using the training data only.
#
# The reason the test data is only used at the very end is to prevent any bi
# We want the test data to remain completely unseen until after we've finali
# as a true evaluation of our model's performance.
# ---------------------------------------------

df = pd.read_csv('../input/mayo-clinic-strip-ai/train.csv')  # Load the trai
df.head()  # Display the first 5 rows of the DataFrame for a quick inspectic
```

Out[ ]:

	image_id	center_id	patient_id	image_num	label
0	006388_0	11	006388	0	CE
1	008e5c_0	11	008e5c	0	CE
2	00c058_0	11	00c058	0	LAA
3	01adc5_0	11	01adc5	0	LAA
4	026c97_0	4	026c97	0	CE

(3) Exploratory Analysis 🔎

(1) Descripción de los Datos

In [ ]:
```
# Print the number of records in the DataFrame
print("The given dataset has", df.shape[0], "registers and", df.shape[1], "c
```

```
The given dataset has 754 registers and 5 columns.
```

Observaciones 💡 -->

> - El conjunto de datos contiene más de mil imágenes de patología digital de alta resolución de diapositivas completas. Cada diapositiva representa un coágulo de sangre de un paciente que sufrió de un accidente cerebrovascular isquémico agudo.

> - En el conjunto `train.csv` , con el que se trabajará por el momento, se cuenta con 754 registros y 5 columnas, lo que indica que tiene una dimensión relativamente pequeña. Cada uno de los 754 registros representa una anotación única con relación a una de las imágenes dentro del directorio `train/` .

Fuente: Página oficial de Kaggle

In [ ]:
```
# Basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 754 entries, 0 to 753
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   image_id    754 non-null    object
 1   center_id   754 non-null    int64 
 2   patient_id  754 non-null    object
 3   image_num   754 non-null    int64 
 4   label       754 non-null    object
dtypes: int64(2), object(3)
memory usage: 29.6+ KB
```

- `image_id` : Un identificador único para la instancia, con el formato {patient_id}_{image_num}. Corresponde a la imagen {image_id}.tif.
- `center_id` : Identifica el centro médico donde se obtuvo la diapositiva.
- `patient_id` : Identifica al paciente del que se obtuvo la diapositiva.
- `image_num` : Enumera las imágenes de coágulos obtenidas del mismo paciente.
- `label` : La etiología del coágulo, que puede ser CE (embolia cardioembólica) o LAA (ataque isquémico). Este campo es el objetivo de clasificación.

Nota importante: Las diapositivas que conforman los conjuntos de entrenamiento y prueba representan coágulos con una etiología (es decir, origen) que se conoce como CE (cardioembólica) o LAA (aterosclerosis de grandes arterias).

Fuente: Página oficial de Kaggle

(2) Clasificación de Variables

Nombre	Descripción	Tipo
image_id	Identificador único de la imagen.	Cualitativa (Nominal)
center_id	Identificador del centro médico donde se tomó la diapositiva.	Cualitativa (Nominal)
patient_id	Identificación del paciente de la diapositiva.	Cualitativa (Nominal)
image_num	Número que indica la secuencia de imágenes de un mismo paciente.	Cuantitativa (Discreta)
label	Clasificación del coágulo: CE (cardioembólica) o LAA (aterosclerosis).	Cualitativa (Nominal)

Observaciones 💡

> - En nuestro conjunto de datos, la mayoría de las variables son de tipo **cualitativo nominal**.

> - Por otro lado, solo una variable es de tipo **cuantitativo discreto**, que corresponde al número de imagen.

(3) Exploración y Limpieza Inicial de los Datos

(1) Análisis de Data Faltante

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
image_id      0
center_id     0
patient_id    0
image_num     0
label         0
dtype: int64
```

Observaciones 💡

> - A partir de este breve análisis de los datos faltantes, podemos observar que no hay columnas con valores nulos. Esto significa que no es

> necesario realizar ningún tipo de imputación en el conjunto de datos. Todas las variables están completamente pobladas, lo que garantiza la integridad de los datos para su análisis posterior.

(2) Previsualización de Imágenes

In [ ]:
```python
# Assuming your DataFrame is named df
base_path = "../input/mayo-clinic-strip-ai/train/"

# Add the full path to the df
df['image_path'] = base_path + df['image_id'] + '.tif'

# Preview the DataFrame to ensure the new column is added correctly
df.head()
```

Out[ ]:

	image_id	center_id	patient_id	image_num	label	image_path
0	006388_0	11	006388	0	CE	../input/mayo-clinic-strip-ai/train/006388_0.tif
1	008e5c_0	11	008e5c	0	CE	../input/mayo-clinic-strip-ai/train/008e5c_0.tif
2	00c058_0	11	00c058	0	LAA	../input/mayo-clinic-strip-ai/train/00c058_0.tif
3	01adc5_0	11	01adc5	0	LAA	../input/mayo-clinic-strip-ai/train/01adc5_0.tif
4	026c97_0	4	026c97	0	CE	../input/mayo-clinic-strip-ai/train/026c97_0.tif

Observaciones 💡

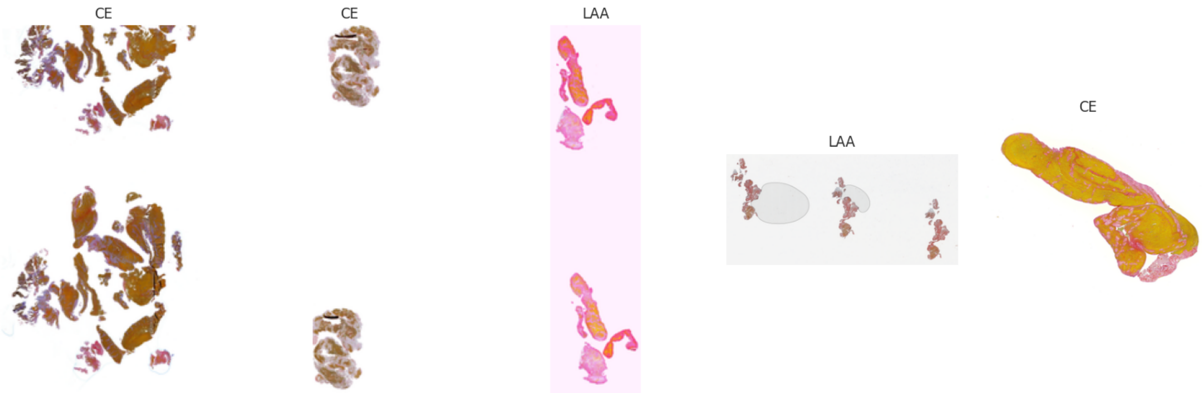
> - Una de las principales técnicas de preprocesamiento será **incluir el path o la dirección de cada imagen** dentro del DataFrame. Esto permitirá acceder a las imágenes de manera más sencilla, ya que, como se ha mencionado anteriormente, todas las imágenes se encuentran almacenadas en el directorio `train`.

In [ ]:
```python
def plot_images(df, num_images=5):
    # Select first n images
    sample_df = df.head(num_images)

    # Create subplots
    fig, axes = plt.subplots(1, num_images, figsize=(15, 5))

    # Flatten axes to make it easier to iterate
    axes = axes.flatten()

    for i, (img_path, label) in enumerate(zip(sample_df['image_path'], sampl
        img = Image.open(img_path)
        img.thumbnail((300,300), Image.Resampling.LANCZOS)
```

```python
        axes[i].imshow(img)
        axes[i].set_title(label)
        axes[i].axis('off')  # Hide axes

    plt.tight_layout()
    plt.show()
```

In [ ]:

```python
# Show 5 images
plot_images(df)
```



Observaciones 💡

> - Una de las primeras observaciones clave es la gran variabilidad en los tamaños de las imágenes presentes en el conjunto de datos. Estas imágenes muestran diferentes valores de relación de aspecto, lo cual puede presentar desafíos al momento de alimentar estos datos en un modelo predictivo.

> - También hemos detectado una variación considerable en las características cromáticas de las imágenes. Por ejemplo, algunas de ellas presentan fondos blancos, mientras que otras tienen fondos grises. Esto puede introducir ruido en el modelo si no se trata adecuadamente, ya que los modelos de visión por computadora son sensibles a las variaciones en las condiciones de iluminación y color de las imágenes.

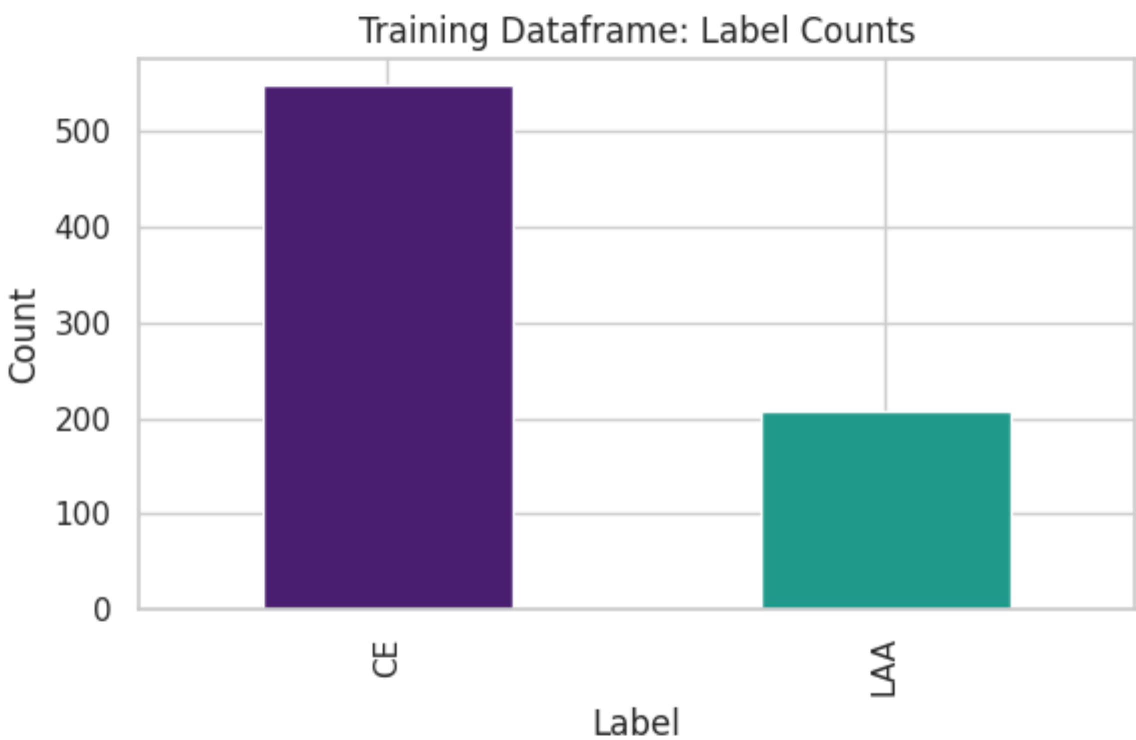
> - Adicionalmente, las imágenes presentan una mezcla de elementos que pueden afectar la precisión del modelo. Algunos coágulos están claramente definidos, mientras que otros están parcialmente oscurecidos o presentan sombras y brillos que pueden confundir al modelo.

Todas extas observaciones reflejan los primeros desafíos que enfrentaremos en el preprocesamiento de imágenes, un paso que se realizará más adelante.

(3) Distribución de Labels

In [ ]:
```
def label_distribution(df, name='Training Dataframe'):
  label_counts = df['label'].value_counts().sort_index()
  total_counts = label_counts.sum()

  # Calculate percentages
  label_percentages = (label_counts / total_counts) * 100

  # Plot counts
  plt.plot(1, 2, 1)
  label_counts.plot(kind='bar', color=[palette[0], palette[6]])
  plt.title(f'{name}: Label Counts')
  plt.xlabel('Label')
  plt.ylabel('Count')
  plt.tight_layout()
  plt.show()
```

In [ ]:
```
label_distribution(df)
```



Observaciones 💡

> - Nuestro conjunto de datos presenta un **fuerte desbalance**, ya que se cuenta con aproximadamente 550 imágenes de coágulos de **etiología cardioembólica (CE)**, lo que representa más del 70.0% del total de muestras.

> - En contraste, solo se disponen de 200 imágenes de coágulos de **etiología aterosclerótica de grandes arterias (LAA)**, lo que corresponde a **menos del 30.0%** del conjunto de datos.

> - Este desequilibrio entre las clases puede tener un impacto negativo en

> el rendimiento del modelo predictivo, ya que probablemente este se incline a **predecir mayormente la clase CE** debido a la falta de representatividad de la clase LAA. Un modelo sesgado de esta manera podría no capturar adecuadamente las características distintivas de los coágulos LAA, reduciendo su capacidad para hacer predicciones precisas.

> - Para mitigar este problema, se tendrá que explorar técnicas como **submuestreo de la clase mayoritaria**, o el uso de algoritmos tales como **data augmentation**. Aunque esto forma parte de los siguientes pasos a tomar.

(4) Análisis de Tamaños de Imagen

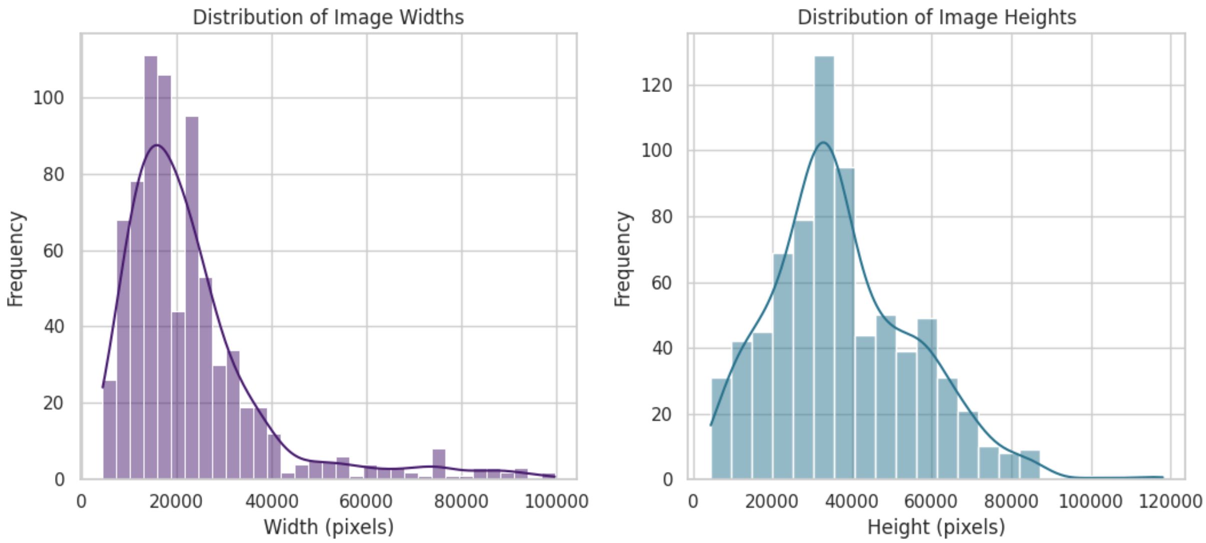
In [ ]:
```
def get_image_size(image_path):
    try:
        with Image.open(image_path) as img:
            return img.size  # Returns (width, height)
    except Exception as e:
        print(f"Error loading {image_path}: {e}")
        return None

# Applying the function to the DataFrame
df['image_size'] = df['image_path'].apply(get_image_size)

# Separate width and height if needed
df['width'], df['height'] = zip(*df['image_size'].dropna())
```

Observaciones 💡

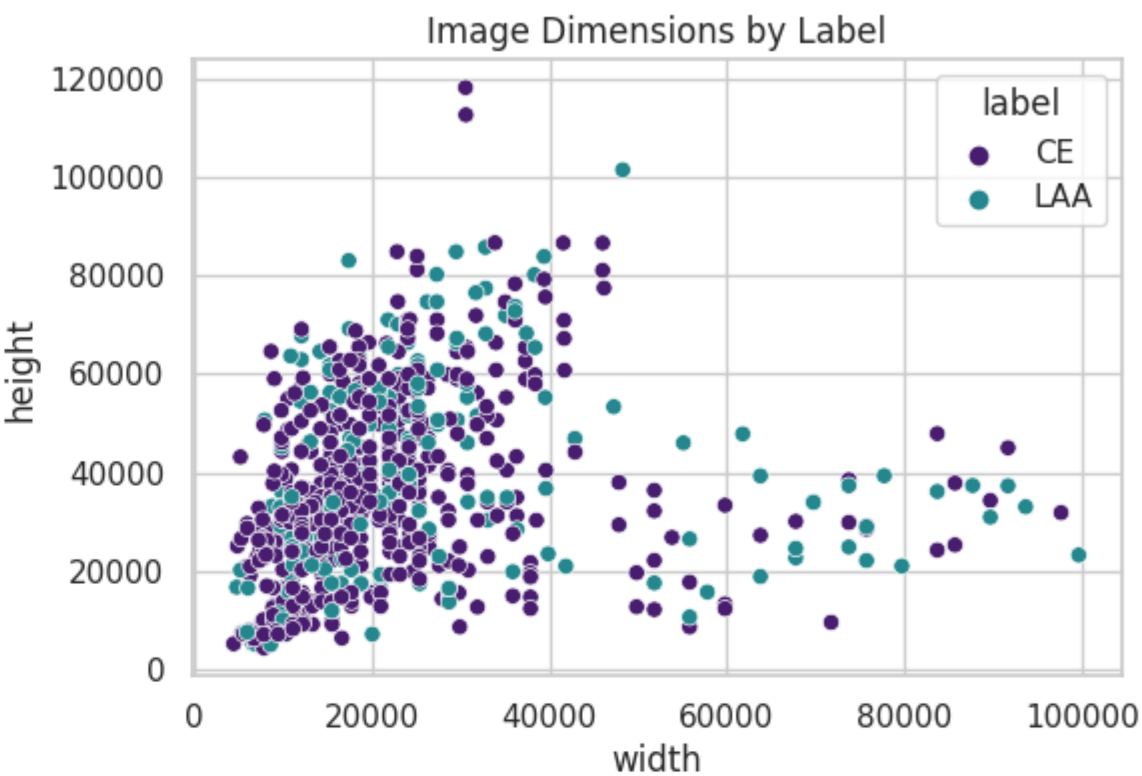
> - Como parte del proceso de preprocesamiento de datos, se extraerán las dimensiones (ancho y alto) de cada imagen del DataFrame. Estas dimensiones se almacenarán en dos columnas separadas (width y height) para facilitar el acceso y análisis de esta información. Este paso es importante ya que nos permitirá analizar la distribución de los tamaños de las imágenes, lo cual puede ser útil para determinar de qué manera se deberá redimensionar estas en los próximos pasos.

In [ ]:
```
# Set up the plot size and style
plt.figure(figsize=(16, 5))

# Plot the distribution of the width
plt.subplot(1, 3, 1)
sns.histplot(df['width'], kde=True, color=palette[0])
plt.title('Distribution of Image Widths')
plt.xlabel('Width (pixels)')
plt.ylabel('Frequency')

# Plot the distribution of the height
```

```
plt.subplot(1, 3, 2)
sns.histplot(df['height'], kde=True, color= palette[4])
plt.title('Distribution of Image Heights')
plt.xlabel('Height (pixels)')
plt.ylabel('Frequency')

# Show the plots
plt.tight_layout()
plt.show()
```



In [ ]:
```
sns.scatterplot(data=df, x='width', y='height', hue='label', palette=[palett
plt.title('Image Dimensions by Label')
plt.show()
```



In [ ]:
```
# Display descriptive statistics for image dimensions
print("Training Set: Image Width Statistics")
print(df['width'].describe())

print("\nTraining Set: Image Height Statistics")
print(df['height'].describe())
```

```
Training Set: Image Width Statistics
count      754.000000
mean     22988.594164
std      15653.642619
min       4417.000000
25%      13215.250000
50%      18700.000000
75%      26376.750000
max      99699.000000
Name: width, dtype: float64

Training Set: Image Height Statistics
count       754.000000
mean      37622.196286
std       18058.750676
min        4470.000000
25%       25402.500000
50%       34981.500000
75%       48919.750000
max      118076.000000
Name: height, dtype: float64
```

Observaciones 💡

> - En estos nuevos gráficos se confirma nuevamente la presencia de múltiples valores tanto en altura como en ancho de las imágenes. La mayoría de las imágenes presentan un ancho concentrado alrededor de los 20,000 píxeles, mientras que la altura se encuentra predominantemente alrededor de los 40,000 píxeles, como lo muestra claramente el histograma.

> - Además, al continuar con este análisis, podemos observar en el diagrama de dispersión que existe una relación predominante entre el tamaño de las imágenes y el tipo de etiología del coágulo del paciente.

> - Se han identificado **datos atípicos**, específicamente en relación con las dimensiones de las imágenes. Sin embargo, estos valores no son de gran relevancia, ya que se normalizarán al redimensionar las imágenes para su inclusión en el modelo.

A continuación veremos cuál es la imagen que posee las dimensiones más grandes y sus detalles.

In [ ]:
```
# Find the largest image without creating a new column for area
largest_image_index = (df['width'] * df['height']).idxmax()
```

```python
largest_image = df.loc[largest_image_index]

# Print the details neatly
print("Largest Image Details:")
print("-" * 30)
for column in df.columns:
    print(f"{column}: {largest_image[column]}")

# Open and display the image using Matplotlib
image_path = largest_image['image_path']
img = Image.open(image_path)
img.thumbnail((400,400), Image.Resampling.LANCZOS)

plt.imshow(img)
plt.axis('off')  # Hide axes
plt.title(f"Largest Image: {largest_image['image_id']}")
plt.show()

del img
```

```
Largest Image Details:
------------------------------
image_id: 6baf51_0
center_id: 11
patient_id: 6baf51
image_num: 0
label: LAA
image_path: ../input/mayo-clinic-strip-ai/train/6baf51_0.tif
image_size: (48282, 101406)
width: 48282
height: 101406
```

Largest Image: 6baf51_0

(5) Cross-Tab Analysis

In [ ]:
```python
# Create a cross-tabulation of 'center_id' and 'label'
```

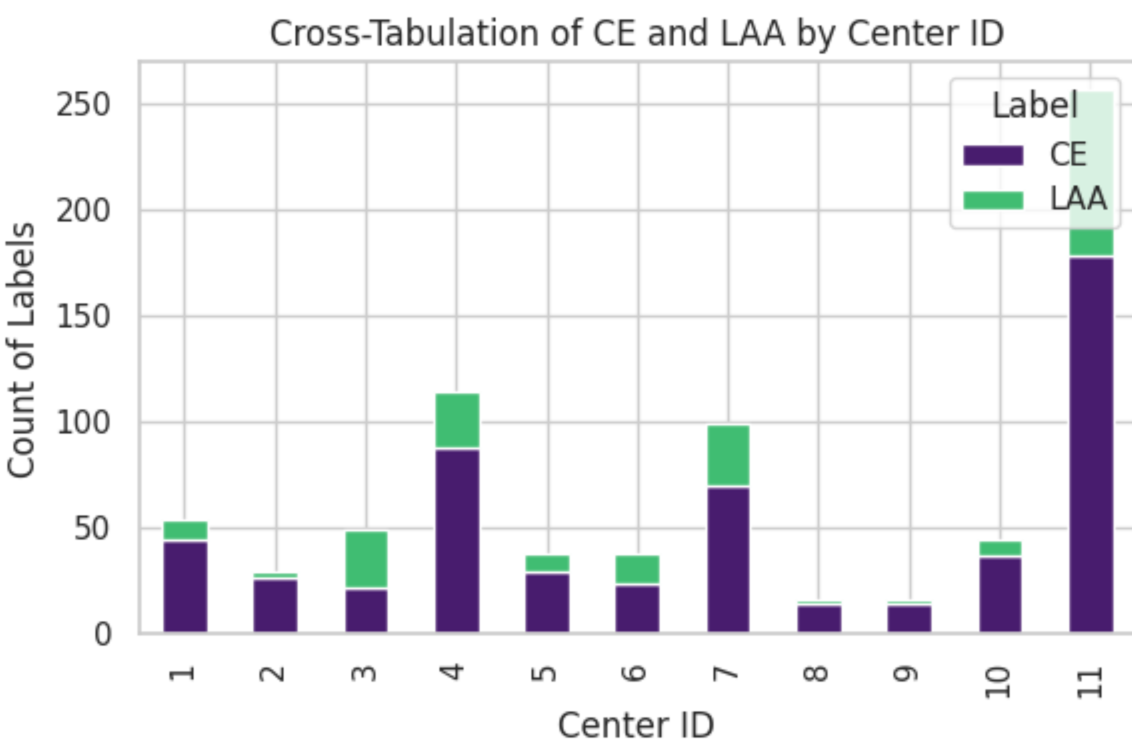
```
cross_tab = pd.crosstab(df['center_id'], df['label'])

# Display the cross-tabulation table
cross_tab.head()
```

Out[ ]:

label center_id	CE	LAA
1	44	10
2	26	3
3	22	27
4	88	26
5	29	9

In [ ]:

```
# Plot a stacked bar plot for better visualization
cross_tab.plot(kind='bar', stacked=True, color=[palette[0], palette[8]])

# Adding titles and labels
plt.title('Cross-Tabulation of CE and LAA by Center ID')
plt.xlabel('Center ID')
plt.ylabel('Count of Labels')
plt.legend(title='Label', loc='upper right')

# Show plot
plt.tight_layout()
plt.show()
```



Observaciones 💡

> - El Centro 11 tiene la mayor cantidad de imágenes, predominantemente etiquetadas como CE (179) en comparación con LAA (7). Este desequilibrio significativo sugiere un posible enfoque o especialización en casos cardioembólicos en este centro.

> - El Centro 4 también muestra una fuerte prevalencia de CE con 88 casos frente a 26 casos de LAA.

> - Los Centros 3 y 7 demuestran una distribución más equilibrada, con el Centro 3 teniendo 22 CE y 27 LAA, y el Centro 7 reportando 70 CE frente a 29 LAA.

> - Otros centros, como el Centro 2 y el Centro 5, presentan un menor número de casos en general, con el Centro 2 mostrando 26 CE y 3 LAA, lo que indica una posible subrepresentación de casos de aterosclerosis de grandes arterias.

Vemos que en general, los datos sugieren que los centros están más especializados en etiologías de accidente cerebrovascular, particularmente en casos **cardioembólicos**.

(6) Tablas de Frecuencia

In [ ]:
```
# Define a function for formatted printing
def print_frequent_values(df):
    # Iterate over each column in the DataFrame
    for column in df.columns:
        frequency_values = df[column].value_counts().head(10)
        print(f"\n{'='*50}\nTop 10 most frequent values for column '{column}
        print(f"{'Index':<5} {'Value':<30} {'Frequency':<10}")
        print('-' * 50)  # Separator line

        for index, (value, frequency) in enumerate(frequency_values.items(),
            print(f"{index:<5} {str(value)[:30]:<30} {frequency:<10}")

        print(f"{'='*50}")
```

In [ ]:
```
# Call the function
print_frequent_values(df)
```

```
==================================================
Top 10 most frequent values for column 'image_id':
Index Value                          Frequency
--------------------------------------------------
1     ffec5c_1                       1
2     006388_0                       1
3     008e5c_0                       1
4     00c058_0                       1
5     01adc5_0                       1
6     026c97_0                       1
7     028989_0                       1
8     029c68_0                       1
9     032f10_0                       1
10    0372b0_0                       1
==================================================

==================================================
Top 10 most frequent values for column 'center_id':
Index Value                          Frequency
--------------------------------------------------
1     11                             257
2     4                              114
3     7                              99
4     1                              54
5     3                              49
6     10                             44
7     5                              38
8     6                              38
9     2                              29
10    8                              16
==================================================

==================================================
Top 10 most frequent values for column 'patient_id':
Index Value                          Frequency
--------------------------------------------------
1     91b9d3                         5
2     56d177                         5
3     09644e                         5
4     3d10be                         5
5     a4c877                         4
6     f40c69                         4
7     4f6fb1                         4
8     5987c0                         4
9     a26055                         3
10    abc4a3                         3
==================================================

==================================================
Top 10 most frequent values for column 'image_num':
Index Value                          Frequency
--------------------------------------------------
1     0                              632
2     1                              89
3     2                              21
4     3                              8
```

```
5     4                              4
==================================================

==================================================
Top 10 most frequent values for column 'label':
Index Value                          Frequency
--------------------------------------------------
1     CE                             547
2     LAA                            207
==================================================

==================================================
Top 10 most frequent values for column 'image_path':
Index Value                          Frequency
--------------------------------------------------
1     ../input/mayo-clinic-strip-ai/ 1
2     ../input/mayo-clinic-strip-ai/ 1
3     ../input/mayo-clinic-strip-ai/ 1
4     ../input/mayo-clinic-strip-ai/ 1
5     ../input/mayo-clinic-strip-ai/ 1
6     ../input/mayo-clinic-strip-ai/ 1
7     ../input/mayo-clinic-strip-ai/ 1
8     ../input/mayo-clinic-strip-ai/ 1
9     ../input/mayo-clinic-strip-ai/ 1
10    ../input/mayo-clinic-strip-ai/ 1
==================================================

==================================================
Top 10 most frequent values for column 'image_size':
Index Value                          Frequency
--------------------------------------------------
1     (15496, 12017)                 2
2     (15410, 29532)                 2
3     (18772, 17529)                 2
4     (14355, 53736)                 1
5     (14368, 16628)                 1
6     (23078, 40562)                 1
7     (51843, 36423)                 1
8     (17545, 42426)                 1
9     (24121, 52541)                 1
10    (25276, 40548)                 1
==================================================

==================================================
Top 10 most frequent values for column 'width':
Index Value                          Frequency
--------------------------------------------------
1     37885                          6
2     51843                          5
3     10533                          5
4     12062                          4
5     75771                          4
6     73777                          4
7     55831                          4
8     7814                           4
9     18660                          3
```

```
10    83747                         3
==================================================

==================================================
Top 10 most frequent values for column 'height':
Index Value                         Frequency
--------------------------------------------------
1     7339                          3
2     29538                         3
3     12017                         2
4     36924                         2
5     33182                         2
6     17529                         2
7     35972                         2
8     35086                         2
9     31366                         2
10    38737                         2
==================================================
```

Observaciones 💡

> - Tal como se espera, el valor más frecuente en la columna `center_id` es 11, que aparece 257 veces, lo que indica que es el centro médico principal involucrado en la mayoría de los casos. Otros centros notables incluyen el 4, con 114 ocurrencias, y el 7, con 99.

> - En la columna `patient_id`, los IDs más frecuentes (por ejemplo, 91b9d3, 3d10be y 09644e) aparecen cada uno 5 veces, lo que indica que estos pacientes tienen múltiples imágenes tomadas para análisis. Esta repetición sugiere un posible enfoque en pacientes específicos con imágenes recurrentes, posiblemente debido a un seguimiento continuo de condiciones particulares.

> - Finalmente, la columna `image_num` muestra que 0 es el valor más común, con 632 ocurrencias, lo que sugiere que el conjunto de datos consiste principalmente en las primeras imágenes tomadas de varios pacientes. Los valores posteriores, con 1 apareciendo 89 veces y 2 solo 21 veces, indican que hay menos imágenes subsiguientes por paciente. Este patrón podría sugerir que la mayoría de los casos implican evaluaciones iniciales en lugar de imágenes de seguimiento.

(7) Análisis de Pacientes

In [ ]:
```python
# Count unique patients
unique_patient_count = df['patient_id'].nunique()

# Print the result neatly
print(f"Unique Patients Count: {unique_patient_count}")
```

```
Unique Patients Count: 632
```

Observaciones 💡

> - Como se ha mencionado anteriormente, solamente algunos pacientes tienen más de una imagen asociada a su persona, por lo que en nuestro conjunto de datos hay imágenes de 630 pacientes **únicos**.

(8) Análisis de Valores Duplicados

In [ ]:
```
# Check duplicate rows in dataset
df = df.drop_duplicates()
# Print the number of records in the DataFrame
print("The given dataset has", df.shape[0], "registers and", df.shape[1], "c
```

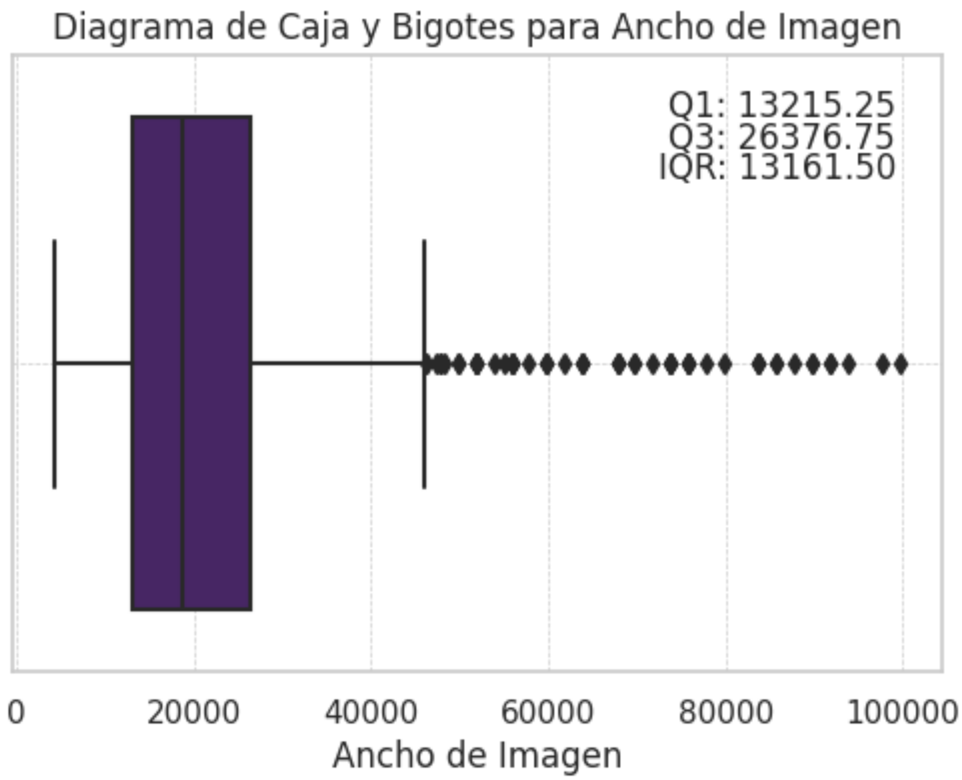
```
The given dataset has 754 registers and 9 columns.
```

Observaciones 💡

> - Vemos que no existen valores duplicados en el conjunto de datos.

(9) Análisis de Datos Atípicos

In [ ]:
```
# Calculate quartiles
# Calculate the first quartile (Q1)
Q1 = df["width"].quantile(0.25)
# Calculate the third quartile (Q3)
Q3 = df["width"].quantile(0.75)
# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Create the box and whisker plot with quartiles
sns.boxplot(x="width", data=df, palette=palette)
plt.title("Diagrama de Caja y Bigotes para Ancho de Imagen")
plt.xlabel("Ancho de Imagen")

# Add quartile information as text annotations
plt.text(0.95, 0.9, f"Q1: {Q1:.2f}", transform=plt.gca().transAxes, ha="righ
plt.text(0.95, 0.8, f"IQR: {IQR:.2f}", transform=plt.gca().transAxes, ha="ri
plt.text(0.95, 0.85, f"Q3: {Q3:.2f}", transform=plt.gca().transAxes, ha="rig

# Adding grid with custom style
plt.grid(True, linestyle='--', linewidth=0.5)  # Adding grid with dashed lin
plt.show()
```



Observaciones 💡

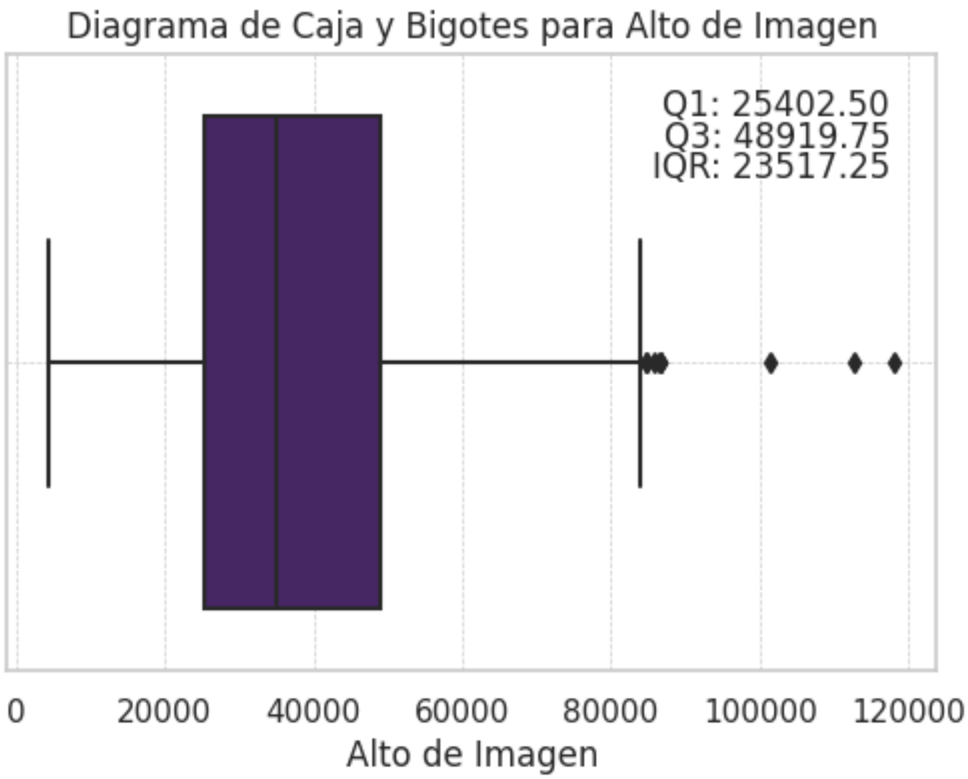
> - Se observa en las estadísticas del conjunto de datos para el ancho de las imágenes que la mediana (50%) es de 18,700 píxeles, lo que indica que la mitad de las imágenes tienen un ancho igual o menor a este valor.

> - El primer cuartil (25%) se encuentra en 13,215 píxeles y el tercer cuartil (75%) en 26,376.75 píxeles, lo que sugiere que la mayoría de las imágenes tienen un tamaño moderado, con un rango intercuartílico (IQR) de 13,161.5 píxeles.

> - El valor máximo de 99,699 píxeles, así como la desviación estándar de 15,653.64 píxeles indica sugieren que existe una variabilidad notable entre los tamaños de las imágenes y revelan la existencia de imágenes significativamente más anchas en el conjunto

In [ ]:
```
# Calculate quartiles
# Calculate the first quartile (Q1)
Q1 = df["height"].quantile(0.25)
# Calculate the third quartile (Q3)
Q3 = df["height"].quantile(0.75)
# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Create the box and whisker plot with quartiles
sns.boxplot(x="height", data=df, palette=palette)
plt.title("Diagrama de Caja y Bigotes para Alto de Imagen")
plt.xlabel("Alto de Imagen")
```

```
# Add quartile information as text annotations
plt.text(0.95, 0.9, f"Q1: {Q1:.2f}", transform=plt.gca().transAxes, ha="righ
plt.text(0.95, 0.8, f"IQR: {IQR:.2f}", transform=plt.gca().transAxes, ha="ri
plt.text(0.95, 0.85, f"Q3: {Q3:.2f}", transform=plt.gca().transAxes, ha="rig

# Adding grid with custom style
plt.grid(True, linestyle='--', linewidth=0.5)  # Adding grid with dashed lir
plt.show()
```



Observaciones 💡

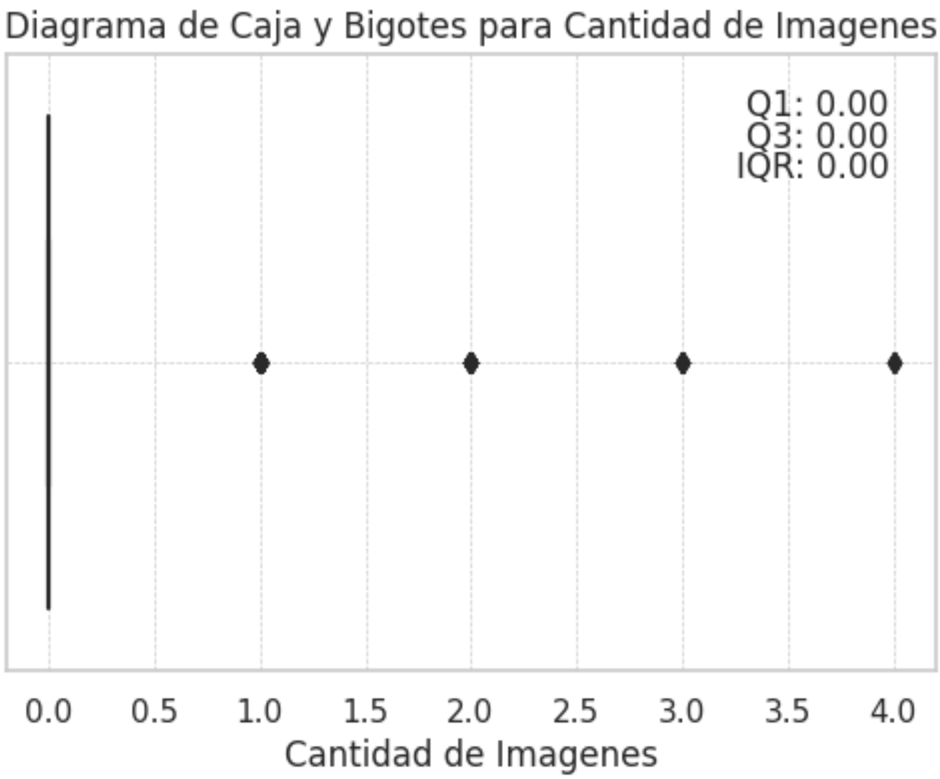
> - Se observa que la mediana (50%) es de 34,981.5 píxeles, lo que indica que la mitad de las imágenes tiene una altura igual o menor a este valor.

> - El primer cuartil (25%) se sitúa en 25,402.5 píxeles y el tercer cuartil (75%) en 48,919.75 píxeles, sugiriendo que la mayoría de las imágenes poseen una altura moderada, con un rango intercuartílico (IQR) de 23,517.25 píxeles.

> - La media de 37,622.20 píxeles refleja la altura promedio de las imágenes, mientras que la desviación estándar de 18,058.75 píxeles indica una considerable dispersión en las alturas.

In [ ]:
```
# Calculate quartiles
# Calculate the first quartile (Q1)
Q1 = df["image_num"].quantile(0.25)
# Calculate the third quartile (Q3)
Q3 = df["image_num"].quantile(0.75)
```

```
# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Create the box and whisker plot with quartiles
sns.boxplot(x="image_num", data=df, palette=palette)
plt.title("Diagrama de Caja y Bigotes para Cantidad de Imagenes")
plt.xlabel("Cantidad de Imagenes")

# Add quartile information as text annotations
plt.text(0.95, 0.9, f"Q1: {Q1:.2f}", transform=plt.gca().transAxes, ha="righ
plt.text(0.95, 0.8, f"IQR: {IQR:.2f}", transform=plt.gca().transAxes, ha="ri
plt.text(0.95, 0.85, f"Q3: {Q3:.2f}", transform=plt.gca().transAxes, ha="rig

# Adding grid with custom style
plt.grid(True, linestyle='--', linewidth=0.5)  # Adding grid with dashed lin
plt.show()
```



In [ ]:

```
# Define the limits to determine outliers
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

# Count the number of outliers
outliers = df[(df["image_num"] < lower_limit) | (df["image_num"] > upper_lim
num_outliers = len(outliers)

# Calculate the percentage of outliers
percentage_outliers = (num_outliers / len(df)) * 100

print("Outliers Percentage in 'image_num':", round(percentage_outliers, 2))
```

```
Outliers Percentage in 'image_num': 16.18
```

In [ ]:
```
print("\nTraining Set: 'image_num' Statistics")
print(df['image_num'].describe())
```

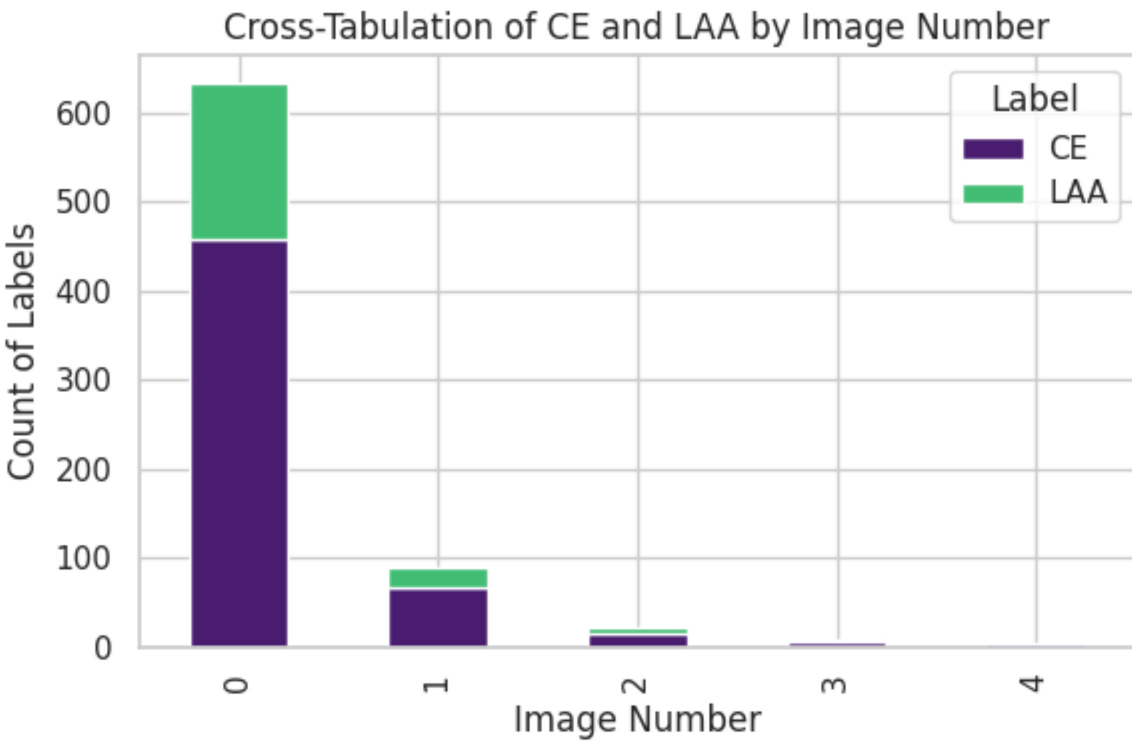
```
Training Set: 'image_num' Statistics
count    754.000000
mean       0.226790
std        0.599046
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        4.000000
Name: image_num, dtype: float64
```

In [ ]:
```
# Create a cross-tabulation of 'center_id' and 'label'
cross_tab = pd.crosstab(df['image_num'], df['label'])

# Display the cross-tabulation table
cross_tab.head()
```

Out[ ]:

label image_num	CE	LAA
0	457	175
1	66	23
2	15	6
3	6	2
4	3	1

In [ ]:
```
# Plot a stacked bar plot for better visualization
cross_tab.plot(kind='bar', stacked=True, color=[palette[0], palette[8]])

# Adding titles and labels
plt.title('Cross-Tabulation of CE and LAA by Image Number')
plt.xlabel('Image Number')
plt.ylabel('Count of Labels')
plt.legend(title='Label', loc='upper right')

# Show plot
plt.tight_layout()
plt.show()
```



Observaciones 💡

> - El valor de todos los cuartiles es de 0, lo que indica que el conjunto de datos consiste principalmente en las primeras imágenes tomadas de varios pacientes. Sin embargo, el valor máximo de 4 indica que hay algunos pacientes que presentan hasta cuatro imágenes de coágulos, lo que introduce cierta variabilidad en el conjunto de datos.

> - Los puntos atípicos representan alrededor del 16.18% de la totalidad de los datos.

(4) Image Preprocessing 📷

In [ ]:
```
def preprocess_image(image_path, idx):
    print(f"Processing image at index {idx}")  # Print the current index
    # Read the image from the file
    img = tifi.imread(image_path)
    #img = cv2.imread(image_path)
    img = cv2.resize(img, (0,0), fx=0.05, fy=0.05)
    # Convert the image to grayscale (if required by your model)
    # If your CNN expects color images, skip this step
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Reason: Convert to grayscale if the model is designed for single-chann
    # Normalize pixel values to the range [0, 1]
    img = img / 255.0
    # Reason: Normalization ensures that pixel values are in a consistent ra
    # Apply Gaussian blur to reduce noise (optional, depending on the noise
    img = cv2.GaussianBlur(img, (5, 5), 0)
```

```
    # Reason: Noise reduction can help the CNN by removing small details tha
    img = np.expand_dims(img, axis=-1)
    # Reason: Consistent image size is required for CNN input.
    return img
```

In [ ]:

```
# Apply the function
df['image'] = df.apply(lambda row: preprocess_image(row['image_path'], row.n
```

```
Processing image at index 0
Processing image at index 1
Processing image at index 2
Processing image at index 3
Processing image at index 4
Processing image at index 5
Processing image at index 6
Processing image at index 7
Processing image at index 8
Processing image at index 9
Processing image at index 10
Processing image at index 11
Processing image at index 12
Processing image at index 13
Processing image at index 14
Processing image at index 15
Processing image at index 16
Processing image at index 17
Processing image at index 18
Processing image at index 19
Processing image at index 20
Processing image at index 21
Processing image at index 22
Processing image at index 23
Processing image at index 24
Processing image at index 25
Processing image at index 26
Processing image at index 27
Processing image at index 28
Processing image at index 29
Processing image at index 30
Processing image at index 31
Processing image at index 32
Processing image at index 33
Processing image at index 34
Processing image at index 35
Processing image at index 36
Processing image at index 37
Processing image at index 38
Processing image at index 39
Processing image at index 40
Processing image at index 41
Processing image at index 42
Processing image at index 43
Processing image at index 44
Processing image at index 45
Processing image at index 46
Processing image at index 47
Processing image at index 48
Processing image at index 49
Processing image at index 50
Processing image at index 51
Processing image at index 52
Processing image at index 53
Processing image at index 54
Processing image at index 55
```

```
Processing image at index 56
Processing image at index 57
Processing image at index 58
Processing image at index 59
Processing image at index 60
Processing image at index 61
Processing image at index 62
Processing image at index 63
Processing image at index 64
Processing image at index 65
Processing image at index 66
Processing image at index 67
Processing image at index 68
Processing image at index 69
Processing image at index 70
Processing image at index 71
Processing image at index 72
Processing image at index 73
Processing image at index 74
Processing image at index 75
Processing image at index 76
Processing image at index 77
Processing image at index 78
Processing image at index 79
Processing image at index 80
Processing image at index 81
Processing image at index 82
Processing image at index 83
Processing image at index 84
Processing image at index 85
Processing image at index 86
Processing image at index 87
Processing image at index 88
Processing image at index 89
Processing image at index 90
Processing image at index 91
Processing image at index 92
Processing image at index 93
Processing image at index 94
Processing image at index 95
Processing image at index 96
Processing image at index 97
Processing image at index 98
Processing image at index 99
Processing image at index 100
Processing image at index 101
Processing image at index 102
Processing image at index 103
Processing image at index 104
Processing image at index 105
Processing image at index 106
Processing image at index 107
Processing image at index 108
Processing image at index 109
Processing image at index 110
Processing image at index 111
```

```
Processing image at index 112
Processing image at index 113
Processing image at index 114
Processing image at index 115
Processing image at index 116
Processing image at index 117
Processing image at index 118
Processing image at index 119
Processing image at index 120
Processing image at index 121
Processing image at index 122
Processing image at index 123
Processing image at index 124
Processing image at index 125
Processing image at index 126
Processing image at index 127
Processing image at index 128
Processing image at index 129
Processing image at index 130
Processing image at index 131
Processing image at index 132
Processing image at index 133
Processing image at index 134
Processing image at index 135
Processing image at index 136
Processing image at index 137
Processing image at index 138
Processing image at index 139
Processing image at index 140
Processing image at index 141
Processing image at index 142
Processing image at index 143
Processing image at index 144
Processing image at index 145
Processing image at index 146
Processing image at index 147
Processing image at index 148
Processing image at index 149
Processing image at index 150
Processing image at index 151
Processing image at index 152
Processing image at index 153
Processing image at index 154
Processing image at index 155
Processing image at index 156
Processing image at index 157
Processing image at index 158
Processing image at index 159
Processing image at index 160
Processing image at index 161
Processing image at index 162
Processing image at index 163
Processing image at index 164
Processing image at index 165
Processing image at index 166
Processing image at index 167
```

```
Processing image at index 168
Processing image at index 169
Processing image at index 170
Processing image at index 171
Processing image at index 172
Processing image at index 173
Processing image at index 174
Processing image at index 175
Processing image at index 176
Processing image at index 177
Processing image at index 178
Processing image at index 179
Processing image at index 180
Processing image at index 181
Processing image at index 182
Processing image at index 183
Processing image at index 184
Processing image at index 185
Processing image at index 186
Processing image at index 187
Processing image at index 188
Processing image at index 189
Processing image at index 190
Processing image at index 191
Processing image at index 192
Processing image at index 193
Processing image at index 194
Processing image at index 195
Processing image at index 196
Processing image at index 197
Processing image at index 198
Processing image at index 199
Processing image at index 200
Processing image at index 201
Processing image at index 202
Processing image at index 203
Processing image at index 204
Processing image at index 205
Processing image at index 206
Processing image at index 207
Processing image at index 208
Processing image at index 209
Processing image at index 210
Processing image at index 211
Processing image at index 212
Processing image at index 213
Processing image at index 214
Processing image at index 215
Processing image at index 216
Processing image at index 217
Processing image at index 218
Processing image at index 219
Processing image at index 220
Processing image at index 221
Processing image at index 222
Processing image at index 223
```

```
Processing image at index 224
Processing image at index 225
Processing image at index 226
Processing image at index 227
Processing image at index 228
Processing image at index 229
Processing image at index 230
Processing image at index 231
Processing image at index 232
Processing image at index 233
Processing image at index 234
Processing image at index 235
Processing image at index 236
Processing image at index 237
Processing image at index 238
Processing image at index 239
Processing image at index 240
Processing image at index 241
Processing image at index 242
Processing image at index 243
Processing image at index 244
Processing image at index 245
Processing image at index 246
Processing image at index 247
Processing image at index 248
Processing image at index 249
Processing image at index 250
Processing image at index 251
Processing image at index 252
Processing image at index 253
Processing image at index 254
Processing image at index 255
Processing image at index 256
Processing image at index 257
Processing image at index 258
Processing image at index 259
Processing image at index 260
Processing image at index 261
Processing image at index 262
Processing image at index 263
Processing image at index 264
Processing image at index 265
Processing image at index 266
Processing image at index 267
Processing image at index 268
Processing image at index 269
Processing image at index 270
Processing image at index 271
Processing image at index 272
Processing image at index 273
Processing image at index 274
Processing image at index 275
Processing image at index 276
Processing image at index 277
Processing image at index 278
Processing image at index 279
```

```
Processing image at index 280
Processing image at index 281
Processing image at index 282
Processing image at index 283
Processing image at index 284
Processing image at index 285
Processing image at index 286
Processing image at index 287
Processing image at index 288
Processing image at index 289
Processing image at index 290
Processing image at index 291
Processing image at index 292
Processing image at index 293
Processing image at index 294
Processing image at index 295
Processing image at index 296
Processing image at index 297
Processing image at index 298
Processing image at index 299
Processing image at index 300
Processing image at index 301
Processing image at index 302
Processing image at index 303
Processing image at index 304
Processing image at index 305
Processing image at index 306
Processing image at index 307
Processing image at index 308
Processing image at index 309
Processing image at index 310
Processing image at index 311
Processing image at index 312
Processing image at index 313
Processing image at index 314
Processing image at index 315
Processing image at index 316
Processing image at index 317
Processing image at index 318
Processing image at index 319
Processing image at index 320
Processing image at index 321
Processing image at index 322
Processing image at index 323
Processing image at index 324
Processing image at index 325
Processing image at index 326
Processing image at index 327
Processing image at index 328
Processing image at index 329
Processing image at index 330
Processing image at index 331
Processing image at index 332
Processing image at index 333
Processing image at index 334
Processing image at index 335
```

```
Processing image at index 336
Processing image at index 337
Processing image at index 338
Processing image at index 339
Processing image at index 340
Processing image at index 341
Processing image at index 342
Processing image at index 343
Processing image at index 344
Processing image at index 345
Processing image at index 346
Processing image at index 347
Processing image at index 348
Processing image at index 349
Processing image at index 350
Processing image at index 351
Processing image at index 352
Processing image at index 353
Processing image at index 354
Processing image at index 355
Processing image at index 356
Processing image at index 357
Processing image at index 358
Processing image at index 359
Processing image at index 360
Processing image at index 361
Processing image at index 362
Processing image at index 363
Processing image at index 364
Processing image at index 365
Processing image at index 366
Processing image at index 367
Processing image at index 368
Processing image at index 369
Processing image at index 370
Processing image at index 371
Processing image at index 372
Processing image at index 373
Processing image at index 374
Processing image at index 375
Processing image at index 376
Processing image at index 377
Processing image at index 378
Processing image at index 379
Processing image at index 380
Processing image at index 381
Processing image at index 382
Processing image at index 383
Processing image at index 384
Processing image at index 385
Processing image at index 386
Processing image at index 387
Processing image at index 388
Processing image at index 389
Processing image at index 390
Processing image at index 391
```

```
Processing image at index 392
Processing image at index 393
Processing image at index 394
Processing image at index 395
Processing image at index 396
Processing image at index 397
Processing image at index 398
Processing image at index 399
Processing image at index 400
Processing image at index 401
Processing image at index 402
Processing image at index 403
Processing image at index 404
Processing image at index 405
Processing image at index 406
Processing image at index 407
Processing image at index 408
Processing image at index 409
Processing image at index 410
Processing image at index 411
Processing image at index 412
Processing image at index 413
Processing image at index 414
Processing image at index 415
Processing image at index 416
Processing image at index 417
Processing image at index 418
Processing image at index 419
Processing image at index 420
Processing image at index 421
Processing image at index 422
Processing image at index 423
Processing image at index 424
Processing image at index 425
Processing image at index 426
Processing image at index 427
Processing image at index 428
Processing image at index 429
Processing image at index 430
Processing image at index 431
Processing image at index 432
Processing image at index 433
Processing image at index 434
Processing image at index 435
Processing image at index 436
Processing image at index 437
Processing image at index 438
Processing image at index 439
Processing image at index 440
Processing image at index 441
Processing image at index 442
Processing image at index 443
Processing image at index 444
Processing image at index 445
Processing image at index 446
Processing image at index 447
```

```
Processing image at index 448
Processing image at index 449
Processing image at index 450
Processing image at index 451
Processing image at index 452
Processing image at index 453
Processing image at index 454
Processing image at index 455
Processing image at index 456
Processing image at index 457
Processing image at index 458
Processing image at index 459
Processing image at index 460
Processing image at index 461
Processing image at index 462
Processing image at index 463
Processing image at index 464
Processing image at index 465
Processing image at index 466
Processing image at index 467
Processing image at index 468
Processing image at index 469
Processing image at index 470
Processing image at index 471
Processing image at index 472
Processing image at index 473
Processing image at index 474
Processing image at index 475
Processing image at index 476
Processing image at index 477
Processing image at index 478
Processing image at index 479
Processing image at index 480
Processing image at index 481
Processing image at index 482
Processing image at index 483
Processing image at index 484
Processing image at index 485
Processing image at index 486
Processing image at index 487
Processing image at index 488
Processing image at index 489
Processing image at index 490
Processing image at index 491
Processing image at index 492
Processing image at index 493
Processing image at index 494
Processing image at index 495
Processing image at index 496
Processing image at index 497
Processing image at index 498
Processing image at index 499
Processing image at index 500
Processing image at index 501
Processing image at index 502
Processing image at index 503
```

```
Processing image at index 504
Processing image at index 505
Processing image at index 506
Processing image at index 507
Processing image at index 508
Processing image at index 509
Processing image at index 510
Processing image at index 511
Processing image at index 512
Processing image at index 513
Processing image at index 514
Processing image at index 515
Processing image at index 516
Processing image at index 517
Processing image at index 518
Processing image at index 519
Processing image at index 520
Processing image at index 521
Processing image at index 522
Processing image at index 523
Processing image at index 524
Processing image at index 525
Processing image at index 526
Processing image at index 527
Processing image at index 528
Processing image at index 529
Processing image at index 530
Processing image at index 531
Processing image at index 532
Processing image at index 533
Processing image at index 534
Processing image at index 535
Processing image at index 536
Processing image at index 537
Processing image at index 538
Processing image at index 539
Processing image at index 540
Processing image at index 541
Processing image at index 542
Processing image at index 543
Processing image at index 544
Processing image at index 545
Processing image at index 546
Processing image at index 547
Processing image at index 548
Processing image at index 549
Processing image at index 550
Processing image at index 551
Processing image at index 552
Processing image at index 553
Processing image at index 554
Processing image at index 555
Processing image at index 556
Processing image at index 557
Processing image at index 558
Processing image at index 559
```

```
Processing image at index 560
Processing image at index 561
Processing image at index 562
Processing image at index 563
Processing image at index 564
Processing image at index 565
Processing image at index 566
Processing image at index 567
Processing image at index 568
Processing image at index 569
Processing image at index 570
Processing image at index 571
Processing image at index 572
Processing image at index 573
Processing image at index 574
Processing image at index 575
Processing image at index 576
Processing image at index 577
Processing image at index 578
Processing image at index 579
Processing image at index 580
Processing image at index 581
Processing image at index 582
Processing image at index 583
Processing image at index 584
Processing image at index 585
Processing image at index 586
Processing image at index 587
Processing image at index 588
Processing image at index 589
Processing image at index 590
Processing image at index 591
Processing image at index 592
Processing image at index 593
Processing image at index 594
Processing image at index 595
Processing image at index 596
Processing image at index 597
Processing image at index 598
Processing image at index 599
Processing image at index 600
Processing image at index 601
Processing image at index 602
Processing image at index 603
Processing image at index 604
Processing image at index 605
Processing image at index 606
Processing image at index 607
Processing image at index 608
Processing image at index 609
Processing image at index 610
Processing image at index 611
Processing image at index 612
Processing image at index 613
Processing image at index 614
Processing image at index 615
```

```
Processing image at index 616
Processing image at index 617
Processing image at index 618
Processing image at index 619
Processing image at index 620
Processing image at index 621
Processing image at index 622
Processing image at index 623
Processing image at index 624
Processing image at index 625
Processing image at index 626
Processing image at index 627
Processing image at index 628
Processing image at index 629
Processing image at index 630
Processing image at index 631
Processing image at index 632
Processing image at index 633
Processing image at index 634
Processing image at index 635
Processing image at index 636
Processing image at index 637
Processing image at index 638
Processing image at index 639
Processing image at index 640
Processing image at index 641
Processing image at index 642
Processing image at index 643
Processing image at index 644
Processing image at index 645
Processing image at index 646
Processing image at index 647
Processing image at index 648
Processing image at index 649
Processing image at index 650
Processing image at index 651
Processing image at index 652
Processing image at index 653
Processing image at index 654
Processing image at index 655
Processing image at index 656
Processing image at index 657
Processing image at index 658
Processing image at index 659
Processing image at index 660
Processing image at index 661
Processing image at index 662
Processing image at index 663
Processing image at index 664
Processing image at index 665
Processing image at index 666
Processing image at index 667
Processing image at index 668
Processing image at index 669
Processing image at index 670
Processing image at index 671
```

```
Processing image at index 672
Processing image at index 673
Processing image at index 674
Processing image at index 675
Processing image at index 676
Processing image at index 677
Processing image at index 678
Processing image at index 679
Processing image at index 680
Processing image at index 681
Processing image at index 682
Processing image at index 683
Processing image at index 684
Processing image at index 685
Processing image at index 686
Processing image at index 687
Processing image at index 688
Processing image at index 689
Processing image at index 690
Processing image at index 691
Processing image at index 692
Processing image at index 693
Processing image at index 694
Processing image at index 695
Processing image at index 696
Processing image at index 697
Processing image at index 698
Processing image at index 699
Processing image at index 700
Processing image at index 701
Processing image at index 702
Processing image at index 703
Processing image at index 704
Processing image at index 705
Processing image at index 706
Processing image at index 707
Processing image at index 708
Processing image at index 709
Processing image at index 710
Processing image at index 711
Processing image at index 712
Processing image at index 713
Processing image at index 714
Processing image at index 715
Processing image at index 716
Processing image at index 717
Processing image at index 718
Processing image at index 719
Processing image at index 720
Processing image at index 721
```

Observaciones 💡 -->

1. **Conversión a Escala de Grises**

   - **Código**: `img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`

- **Descripción**: Convierte la imagen de color (BGR) a una imagen en escala de grises.
- **Propósito**: Reducir la complejidad del modelo ya que se espera una entrada de un solo canal (grayscale). Permitirá que el entrenamiento sea más rápido.

2. **Normalización de Valores de Pixel**

   - **Código**: `img = img / 255.0`
   - **Descripción**: Normaliza los valores de los píxeles a un rango de [0, 1].
   - **Propósito**: Normalizar los valores de los píxeles asegura que estén en un rango consistente, lo que ayuda a mantener una iluminación y contraste uniformes en todas las imágenes. Esto facilita la comparación de patrones y características, haciendo que las imágenes sean más adecuadas para el análisis. Además, permite que los modelos de aprendizaje automático aprendan patrones de manera más efectiva, sin que las diferencias en las condiciones de iluminación afecten el rendimiento.

3. **Aplicación de Desenfoque Gaussiano**

   - **Código**: `img = cv2.GaussianBlur(img, (5, 5), 0)`
   - **Descripción**: Aplica un filtro de desenfoque gaussiano para reducir el ruido en la imagen.
   - **Propósito**: El desenfoque gaussiano reduce el detalle y el ruido en la imagen aplicando una función gaussiana a cada píxel y sus píxeles circundantes. Esto suaviza los bordes y elimina pequeños detalles que pueden no ser útiles para el aprendizaje, ayudando a la red neuronal a centrarse en las características más relevantes. La reducción de ruido mejora la capacidad de generalización del modelo y puede facilitar tareas como la detección de bordes o la segmentación al reducir las variaciones no deseadas en la imagen.

4. **Expansión de Dimensiones**

   - **Código**: `img = np.expand_dims(img, axis=-1)`
   - **Descripción**: Expande las dimensiones de la imagen para agregar un canal adicional.
   - **Propósito**: Asegura que la imagen tenga un formato consistente para la entrada de la red neuronal, especialmente si la red espera una entrada con un número específico de canales (por ejemplo, [alto, ancho, canales]).

5. Redimensionamiento de Imagen

   - **Código**: img = cv2.resize(img, (0,0), fx=0.05, fy=0.05)
   - **Descripción**: Redimensiona la imagen aplicando un factor de escala del 5% en los ejes horizontal (fx) y vertical (fy).
   - **Propósito**: Reduce el tamaño de la imagen para disminuir la cantidad de información que se procesa en la red neuronal convolucional (CNN), lo que puede ser útil para ahorrar recursos computacionales o ajustar la entrada a las dimensiones esperadas por el modelo.

Referencia

> - https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c