Familiarización de un entorno de trabajo para hacer un Análisis de una red de tráfico utilizando IA.

Resumen: Garantizar la privacidad de la información del paciente, así como que esta no sea manipulada antes de realizar un diagnóstico y tratamiento del paciente cuando estamos trabajando con un sistema IoT, es sumamente importante, por lo cual primeramente necesitamos conocer los datos que tenemos, saber que respecta cada uno, así mismo con qué herramientas podemos ayudarnos para hacer un trabajo adecuado.

Introducción: En esta primera entrega, nuestro objetivo es familiarizarnos con nuestro entorno de trabajo y comenzar a manipular de forma experimental los datos, así mismo, explorar las bibliotecas y funciones con las que hemos de trabajar para generar una solución eficiente. Trabajaremos con un dataset que contiene registros relacionados con el cuidado de la salud, esta información intenta ser interceptada y modificada en tránsito, más adelante buscaremos como detectar estos ataques. Por lo tanto, en esta etapa investigaremos el contenido y la orientación de los datos para realizar un análisis más preciso, buscamos garantizar que tanto los datos lleguen íntegros a su destino para qué se puede hacer un tratamiento y diagnóstico correcto, así como que resguardar la privacidad del paciente.

Dataset:

1. **Descripción del dataset:** El dataset alberga más de 16 mil registros relacionados con el cuidado de la salud, se combinan datos biométricos con network flow(Red de flujo), en este dataset se albergan registros normales y con MITO (Man-In-The-Middle -> alteración de los paquetes en tránsito).

2. **Arquitectura/testbed:**

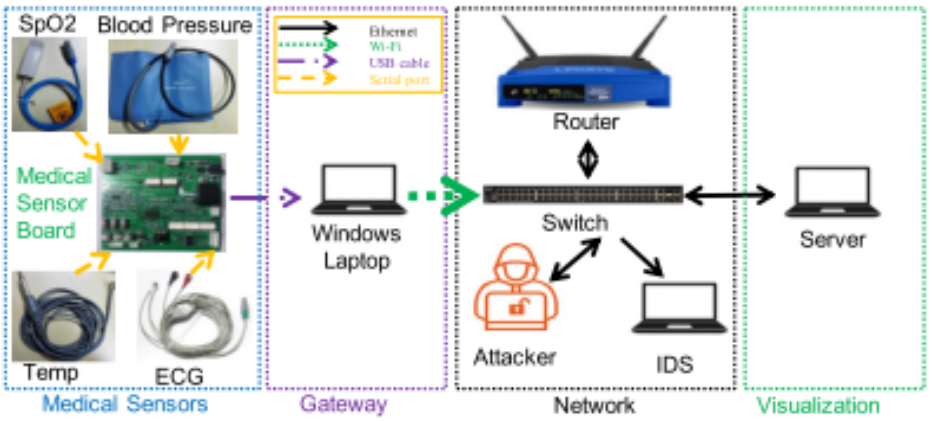

Fig 1. EHMS testbed [2]

La tarjeta (Placa médica) está conectada a una computadora con SO Windows mediante el puerto USB, un software en C++ captúralos datos censados. La

computadora es el (gateway) que transfiere los datos al servidor mediante Wifi usando el protocolo TCP/IP.
Todas las máquinas están conectadas por Ethernet, menos el gateway, El switch está conectado a Internet mediante el router, al cual el gateway está conectado por Wifi.

3. **Atributos:**

Nombre (Metric)	Descripción	Biometrico/Flow metric	Tipo
SrcAddr	**Source Bytes**	**Flow metric**	**Categorical**
DstBytes	**Destination Bytes**	**Flow metric**	**Categorical**
SrcLoad	**Source Load**	**Flow metric**	**Numeric**
DstLoad	**Destination Load**	**Flow metric**	**Numeric**
SrcGap	**Source missing bytes**	**Flow metric**	**Numeric**
DstGap	**Destination missing bytes**	**Flow metric**	**Numeric**
SIntPkt	**Source Inter Packet**	**Flow metric**	**Numeric**
DIntPkt	**Destination Inter Packet**	**Flow metric**	**Numeric**
SrcJitter	**Source jitter**	**Flow metric**	**Numeric**
DstJitter	**Destination Jitter**	**Flow metric**	**Numeric**
sMaxPktSz	**Source Maximun Transmitted Packet size**	**Flow metric**	**Numeric**
dMaxPktSz	**Destination Maximum Transmitted Packet size**	**Flow metric**	**Numeric**
sMinPktSz	**Source Minumum Transmitted Packet size**	**Flow metric**	**Numeric**
dMinPktSz	**Destination Minimum Transmitted Packet size**	**Flow metric**	**Numeric**
Dur	**Duration**	**Flow metric**	**Numeric**

Trans	**Aggregated Packets Count**	**Flow metric**	**Numeric**
TotPkts	**Total Packets Count**	**Flow metric**	**Numeric**
TotBytes	**Total Packets Bytes**	**Flow metric**	**Numeric**
Loss	**Retransmitted or Dropped Packets**	**Flow metric**	**Numeric**
pLoss	**Percentage of Retransmitted or Dropped Packets**	**Flow metric**	**Numeric**
pSrcLoss	**Percentage of Destination Retransmitted or Dropped Packets**	**Flow metric**	**Numeric**
Rate	**Number of Packets per second**	**Flow metric**	**Numeric**
SrcMac	**Source MAC**	**Flow metric**	**Categorical**
DstMac	**Destination Mac**	**Flow metric**	**Categorical**
Sport	**Source Port**	**Flow metric**	**Categorical**
Dport	**Destination Port**	**Flow metric**	**Categorical**
Temp	**Temperature**	**Biometric**	**Numeric**
SpO2	**Peripheral Oxygen Saturation**	**Biometric**	**Numeric**
Pulse_Rate	**Pulse Rate**	**Biometric**	**Numeric**
SYS	**Systolic Blood Preassure**	**Biometric**	**Numeric**
DIA	**Diastolic Blood Preassuere**	**Biometric**	**Numeric**
Heart_Rate	**Heart_Rate**	**Biometric**	**Numeric**
Resp_Rate	**Respiration Rate**	**Biometric**	**Numeric**
ST	**ECG ST segment**	**Biometric**	**Numeric**
Label	**Attacked / Normal**		**Categorical**

4. **Clases:**
   Intrusión
   No intrusión

Análisis de la red de tráfico: Se comunican por el protocolo TCP / IP

Experimentación:
Trabajé en con mathplotlib para la generación de los gráficos, así que para llegar a los resultados tuve que hacer un cargado del dataset en un DataFrame con ayuda de la librería de pandas.

1. **Distribución de las clases-frecuencia:**
   Para el análisis de la distribución de las clases hicimos un análisis de las etiquetas, es decir, cuantos de los registros están clasificados como normales y cuáles como un ataque, donde 0 es normal(no intrusión) y 1 presenta una intrusión (ataque).

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

EHMS = pd.read_csv('../WUSTL-EHMS/wustl-ehms-2020.csv')

df = pd.DataFrame(EHMS)

# Hacemos un conteo de las ocurrencias de cada etiqueta
label_counts = df['Label'].value_counts()

# Plot the label counts
plt.figure(figsize=(8, 6))
label_counts.plot(kind='bar', color=['pink', 'magenta'])
plt.title('Count of Labels')
plt.xlabel('Label')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()

# Display the counts for each label
label_counts
```

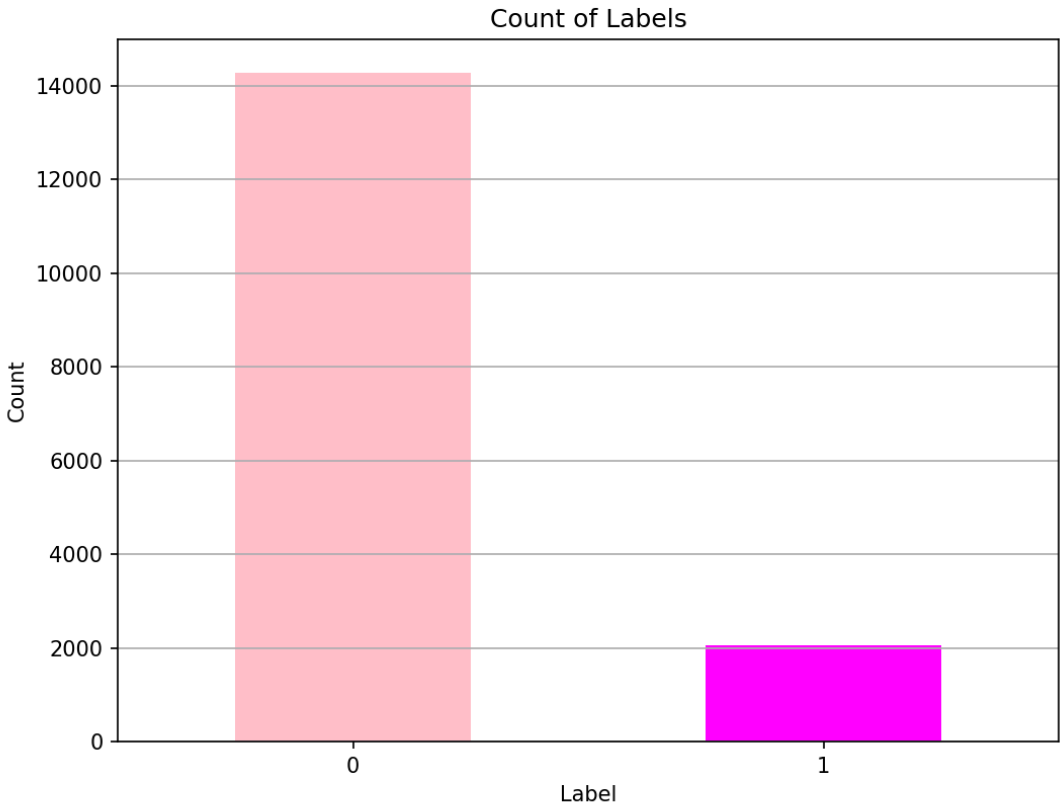

Fig 2. Distribucion de las clases-frecuencia

2. **Distribución de las características-atributo Frecuencia:**
   Para la distribución de los atributos se realizó una gráfica de los datos biométricos, ya que los demás datos son respectivos a la red y comunicación, y los tamaños de transferencia y ese tipo de aspectos son constantes, si se gusta comprobar el código está comentado.

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

# Cargar el dataset
EHMS = pd.read_csv('../WUSTL-EHMS/wustl-ehms-2020.csv')

df = pd.DataFrame(EHMS)

# Convertir la columna 'Label' a categórica
df['Label'] = df['Label'].astype('category')

# Análisis de cada columna
column_analysis = []
```

```
for column in df.columns:
    col_type = df[column].dtype
    if column == 'Label':
        # Procesar como dato categórico
        column_analysis.append({
            'Column': column,
            'DataType': col_type,
            'Category': 'Categorical',
            'Min': None,
            'Max': None
        })
    elif pd.api.types.is_numeric_dtype(df[column]):
        col_min = df[column].min()
        col_max = df[column].max()
        column_analysis.append({
            'Column': column,
            'DataType': col_type,
            'Category': 'Numeric',
            'Min': col_min,
            'Max': col_max
        })
    else:
        column_analysis.append({
            'Column': column,
            'DataType': col_type,
            'Category': 'Categorical',
            'Min': None,
            'Max': None
        })

# Convertir el análisis a un DataFrame para una visualización más
fácil
analysis_df = pd.DataFrame(column_analysis)

# Mostrar el análisis
print(analysis_df)


#vamos a graficar la distribución de los datos biometricos
biometric_columns = ['Temp', 'SpO2', 'Pulse_Rate', 'SYS', 'DIA',
'Heart_rate', 'Resp_Rate','ST']
```

```
#Cuidados en un ambiente hospitalario, para evitar el robo de
información médica o la alteración de la misma

for column in biometric_columns:
    plt.figure(figsize=(8, 6))
    plt.hist(df[column], bins=20, color='skyblue')
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.grid(axis='y')
    plt.show()

#flow_metric_colums    =    ['SrcBytes',    'DstBytes',    'SrcGap',
'DstGap','SrcJitter','DstJitter','sMaxPktSz','dMaxPktSz','sMinPkt
Sz','dMinPktSz','Dur','Trans','TotPkts','TotBytes']
#for column in flow_metric_colums:
#    plt.figure(figsize=(8, 6))
#    plt.hist(df[column], bins=20, color='pink')
#    plt.title(f'Distribution of {column}')
#    plt.xlabel(column)
#    plt.ylabel('Count')
#    plt.grid(axis='y')
#    plt.show()
```

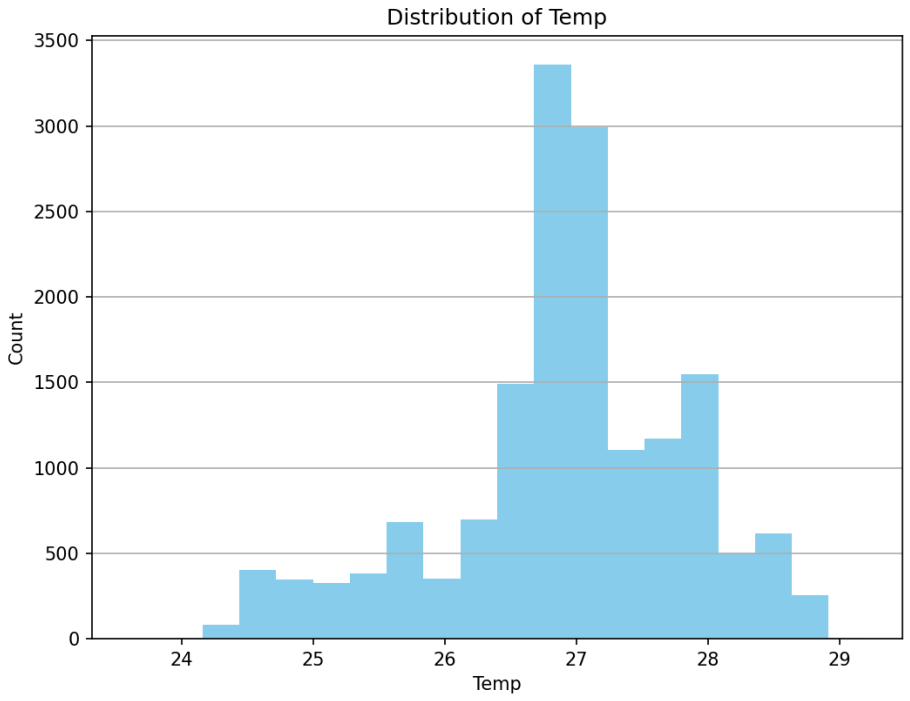

Fig.3 Distribución de la temperatura

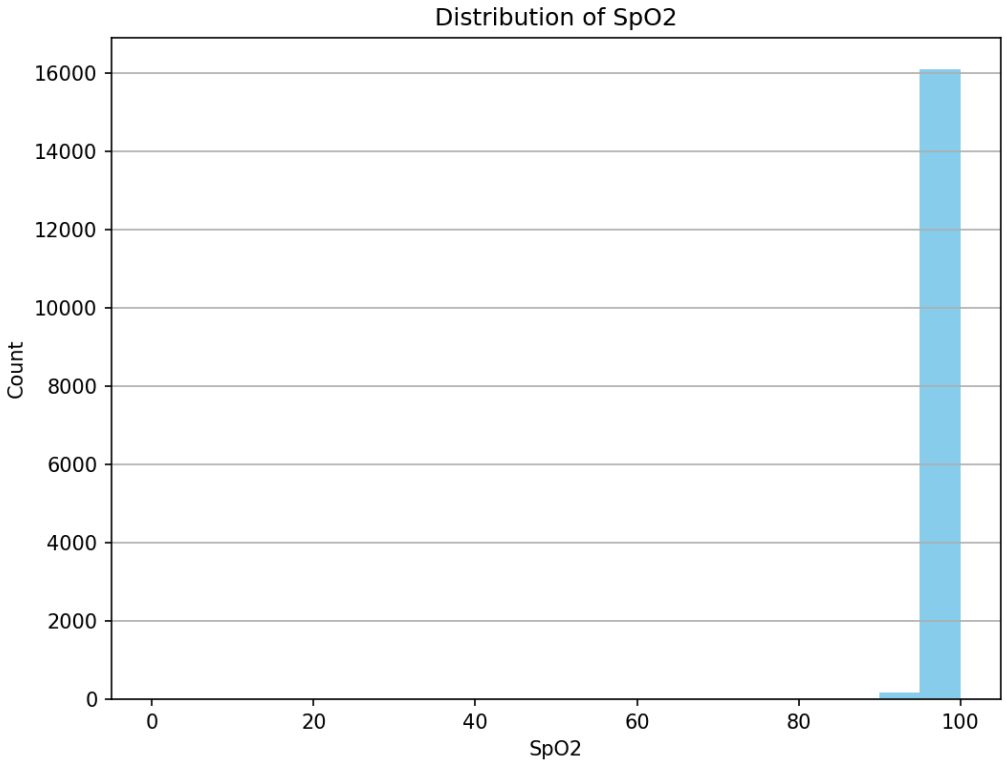

Fig.4 Distribución de la saturación del oxígeno en la sangre

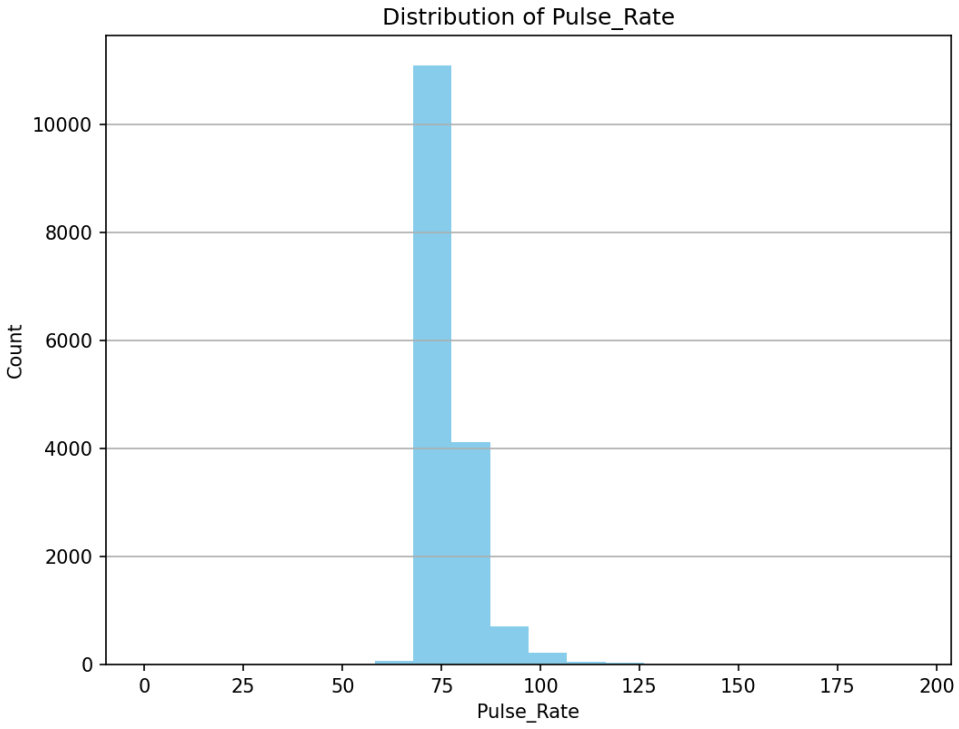

Fig.5 Distribución del pulse rate

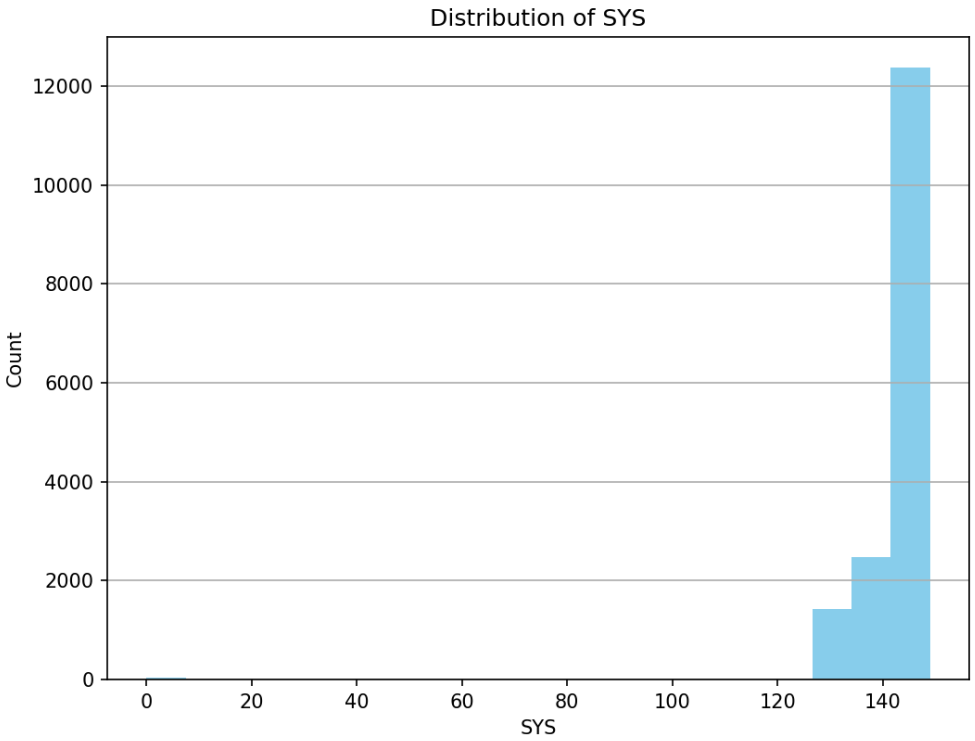

Fig.6 Distribución de la presión sistólica.

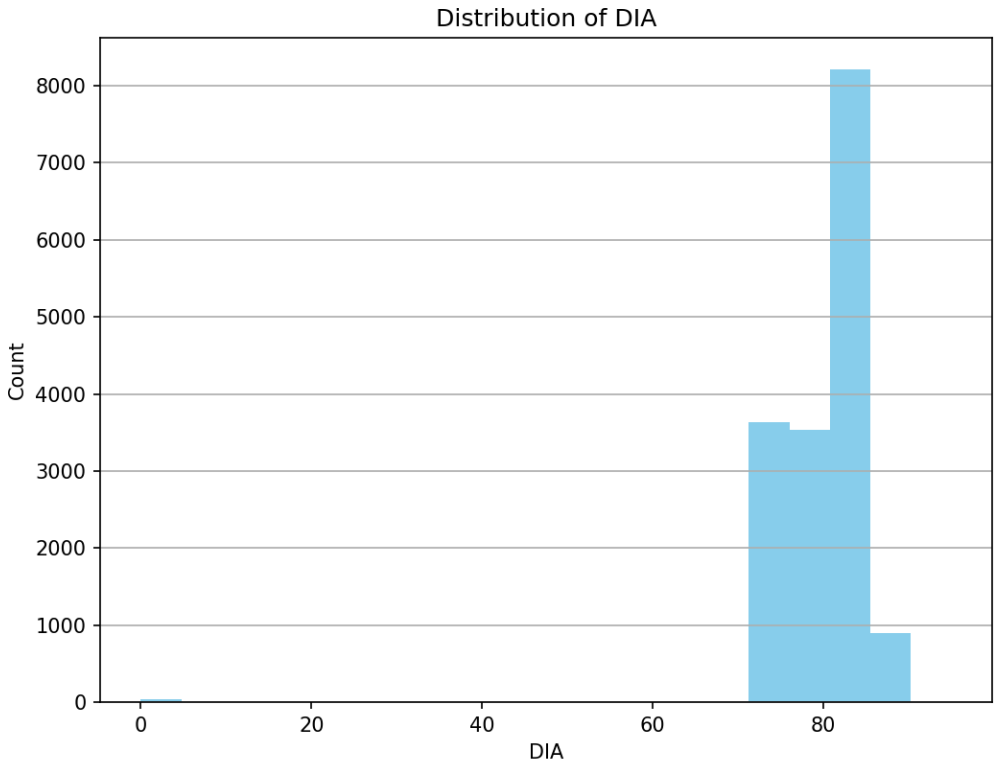

Fig.7 Distribución de la presión diastólica

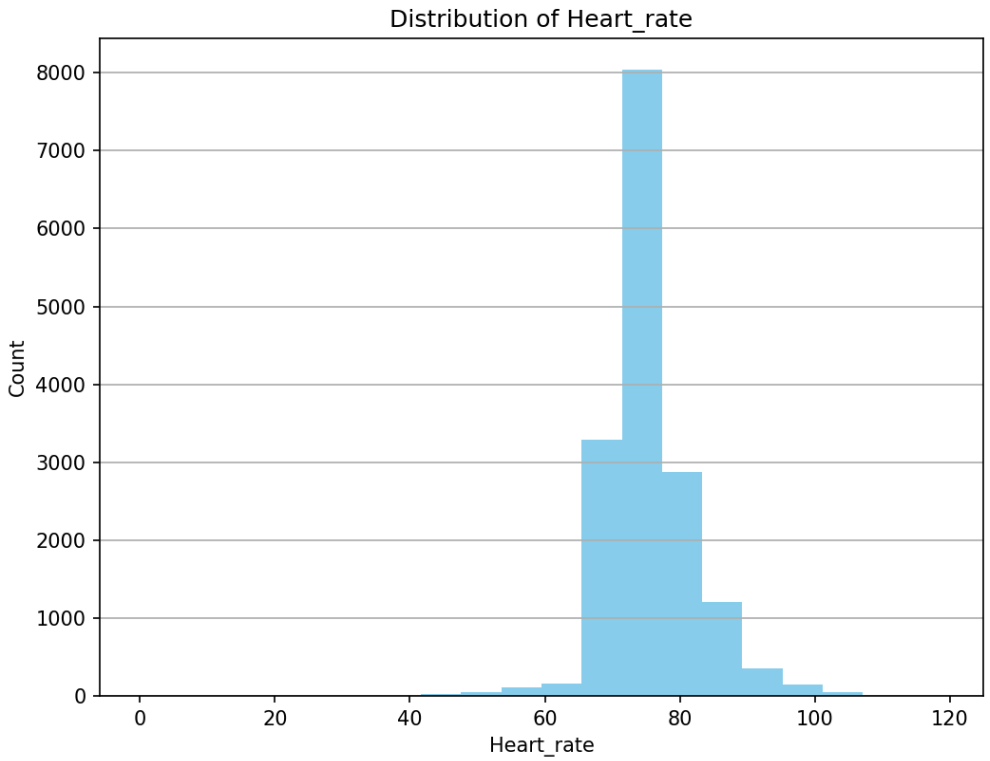

Fig.8 Distribución del heart rate

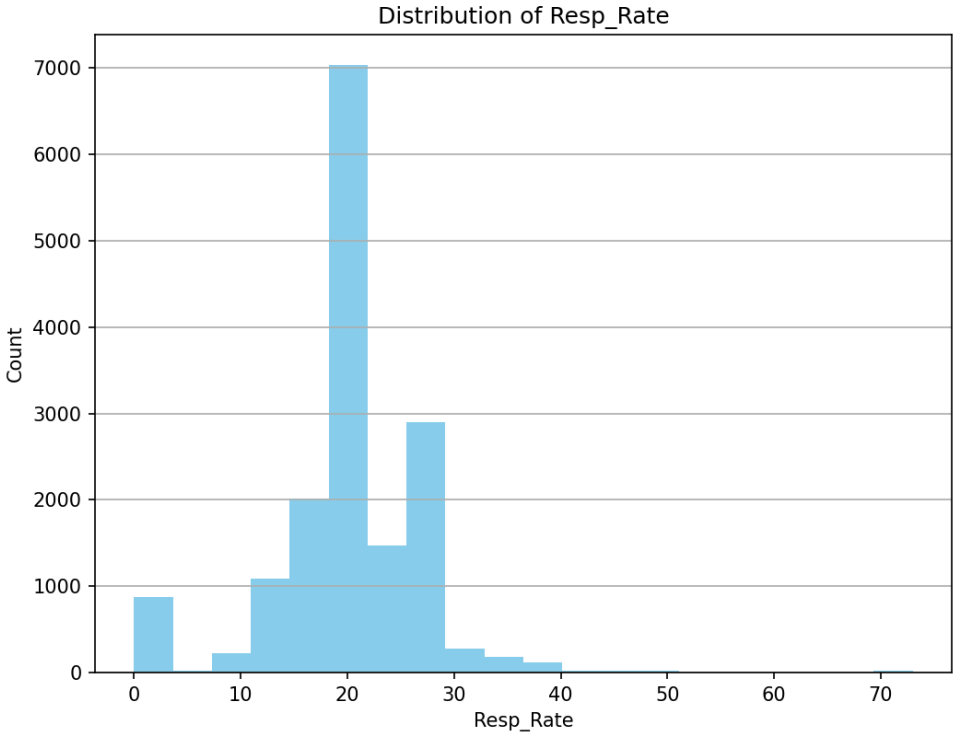

Fig 9. Distribución del Respiration Rate

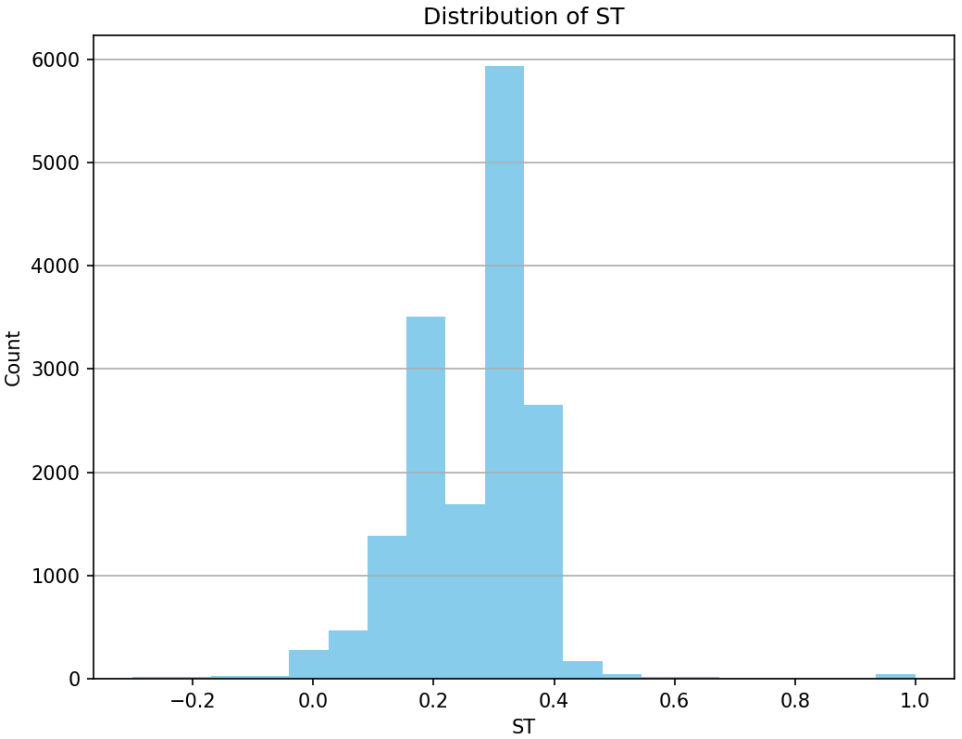

Fig.10 Distribución del ECG ST segment

Las gráficas de los datos de tipo "flow metric" no se clasificaron debido a que las gráficas se verían como se muestran a continuación, las variantes son las MAC y los puertos, pero no sería razonable hacer esas gráficas.

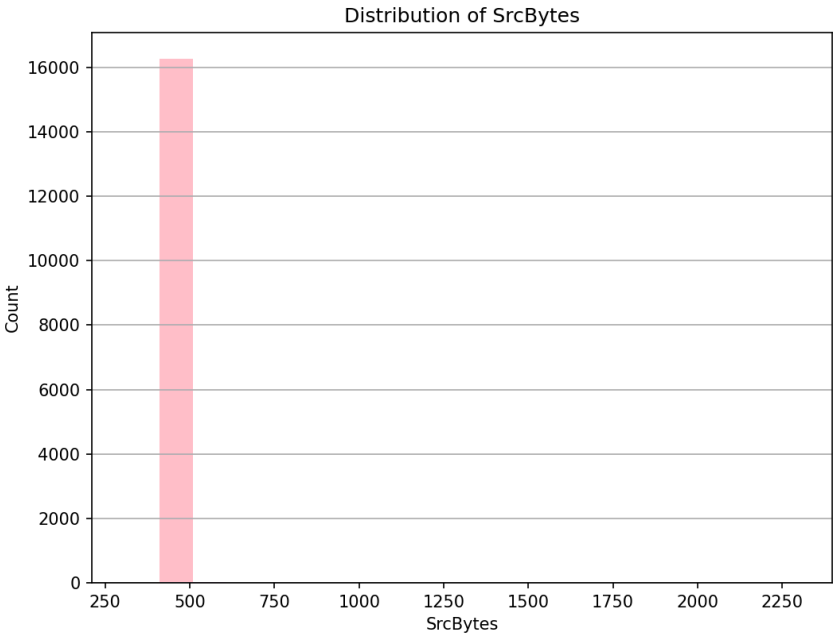

Fig.11 Distribución de los SrcBytes

3. **Distribucion de caracteristicos-formato(categorical,numerico)**
   Para poder hacer esta distribución se ocupó la función `select_dtypes` que nos permite seleccionar del DataFrame un tipo de dato, en los categóricos colocamos los tipos objetos.

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

EHMS = pd.read_csv('../WUSTL-EHMS/wustl-ehms-2020.csv')

df = pd.DataFrame(EHMS)
#print(EHMS.head())

# Convertir tipos de datos si es necesario para categorizar, lo
estaba detectando como numerico
df['Dport'] = df['Dport'].astype('object')

# Identificar las características categóricas y numéricas
categorical_columns =
df.select_dtypes(include=['object']).columns
numerical_columns = df.select_dtypes(include=['number']).columns

# Contar las características categóricas y numéricas
categorical_count = len(categorical_columns)
numerical_count = len(numerical_columns)

# Graficar la distribución de las características por formato
plt.figure(figsize=(8, 6))
plt.bar(['Categorical', 'Numerical'], [categorical_count,
numerical_count], color=['Orange', 'Pink'])
plt.title('Distribution by Category')
plt.xlabel('Feature Type')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()

categorical_columns, numerical_columns
```

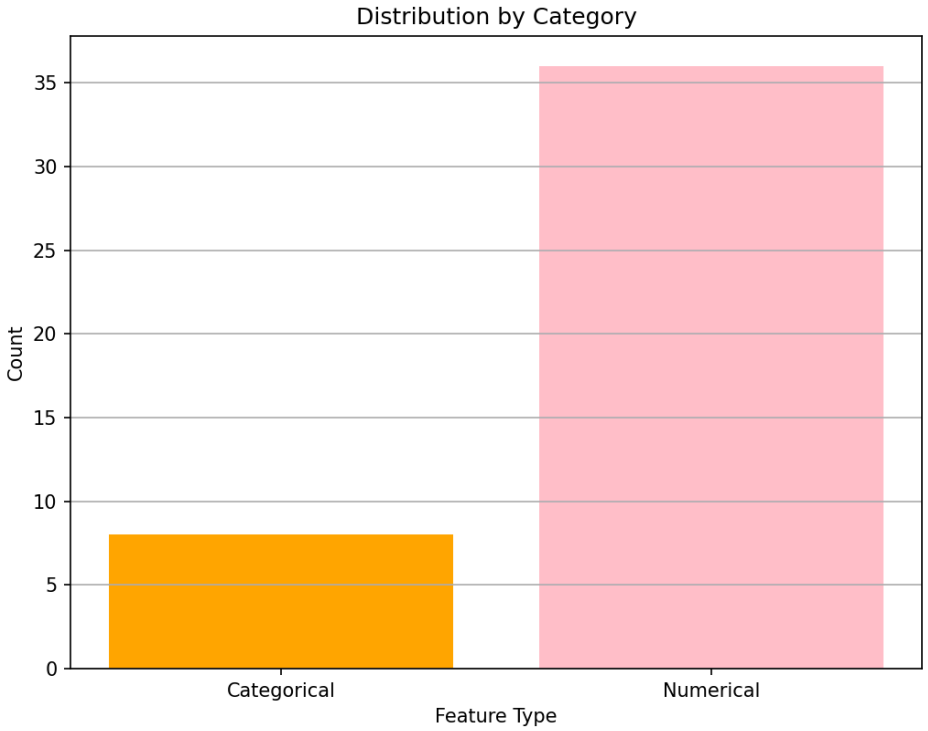

Fig.12 Distribución por categoría

4. Distribución de características de la red de trafico-protocolos de comunicación

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

# Cargar el dataset
file_path = '../WUSTL-EHMS/wustl-ehms-2020.csv'
df = pd.read_csv(file_path)

# Atributos relacionados con protocolos de comunicación TCP/IP
tcp_ip_columns = ['SrcAddr', 'DstAddr', 'Sport', 'Dport',
'SrcBytes', 'DstBytes', 'SrcLoad', 'DstLoad',
                  'SrcGap', 'DstGap', 'SIntPkt', 'DIntPkt',
'SIntPktAct', 'DIntPktAct', 'SrcJitter', 'DstJitter',
                  'sMaxPktSz', 'dMaxPktSz', 'sMinPktSz',
'dMinPktSz', 'Dur', 'Trans', 'TotPkts', 'TotBytes',
                  'Load', 'Loss', 'pLoss', 'pSrcLoss',
'pDstLoss', 'Rate', 'SrcMac', 'DstMac', 'Packet_num']

total_columns = len(df.columns)
print(total_columns)
columns_no_iot = total_columns - len(tcp_ip_columns)
columns_iot = len(tcp_ip_columns)
```

```
# Graficar cuantos atributos pertenecen a IoT y cuantos no
plt.figure(figsize=(8, 6))
labels = ['Non-TCP/IP attributes', 'TCP/IP attributes']
counts = [columns_no_iot, columns_iot]
colors = ['pink', 'magenta']

plt.bar(labels, counts, color=colors)
plt.title('Distribution of the attibutes of the traffic-protocols
of comunication')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
```

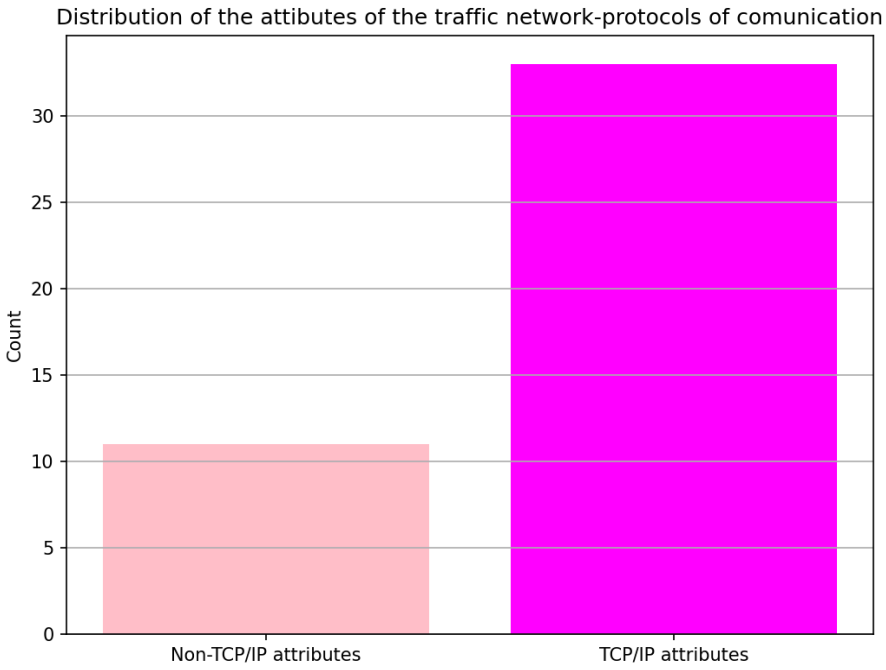

Fig. 13 Distribución de las características de la red de tráfico-protocolos de comunicación.

5. Distribución de dispositivos-aplicaciones en IoT.

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

# Cargar el dataset
file_path = '../WUSTL-EHMS/wustl-ehms-2020.csv'
```

```
df = pd.read_csv(file_path)

# Atributos relacionados con dispositivos-aplicación en IoT
iot_columns = ['SrcAddr', 'DstAddr', 'Temp', 'SpO2',
'Pulse_Rate', 'SYS', 'DIA', 'Heart_rate', 'Resp_Rate', 'ST']
total_columns = len(df.columns)
columns_no_iot = total_columns - len(iot_columns)
columns_iot = len(iot_columns)

# Graficar cuantos atributos pertenecen a IoT y cuantos no
plt.figure(figsize=(8, 6))
labels = ['Non-IoT attributes', 'IoT attributes']
counts = [columns_no_iot, columns_iot]
colors = ['pink', 'magenta']

plt.bar(labels, counts, color=colors)
plt.title('Distribution of dispositives-app IoT')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
```

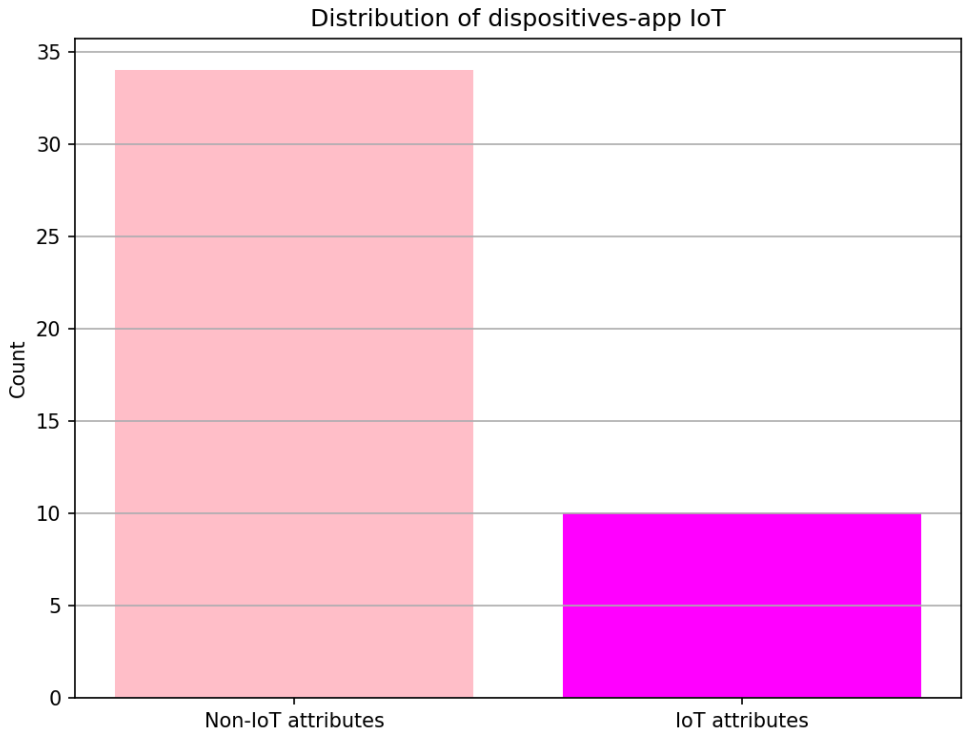

Fig. 14 Distribución de los dispositivos-aplicaciones de IoT

También se hicieron algunas pruebas con las librerias con las que no estaba tan familiarizada, me ayudé de código y ejemplos ya hechos:

```
import pandas as pd
import numpy as np
import math

midwest = pd.read_csv('midwest.csv')

#print(midwest.head())

def calc_information_gain(data, split_name, target_name):
    """
    Calculate information gain given a data set, column to split
on, and target
    """
    # Calculate the original entropy
    original_entropy = calc_entropy(data[target_name])

    #Find the unique values in the column
    values = data[split_name].unique()


    # Make two subsets of the data, based on the unique values
    left_split = data[data[split_name] == values[0]]
    right_split = data[data[split_name] == values[1]]

    # Loop through the splits and calculate the subset entropies
    to_subtract = 0
    for subset in [left_split, right_split]:
        prob = (subset.shape[0] / data.shape[0])
        to_subtract += prob * calc_entropy(subset[target_name])

    # Return information gain
    return original_entropy - to_subtract


def calc_entropy(column):
    """
    Calculate entropy given a pandas series, list, or numpy
array.
    """
    # Compute the counts of each unique value in the column
```

```
    counts = np.bincount(column)
    # Divide by the total column length to get a probability
    probabilities = counts / len(column)

    # Initialize the entropy to 0
    entropy = 0
    # Loop through the probabilities, and add each one to the
total entropy
    for prob in probabilities:
        if prob > 0:
            # use log from math and set base to 2
            entropy += prob * math.log(prob, 2)

    return -entropy

# Calculate the entropy of the Label column in midwest
print(calc_entropy(midwest['age']))
```

También empecé a experimentar con la Liberia de scikit-learn

```
from sklearn.datasets import fetch_openml

X_adult, y_adult = fetch_openml("adult", version=2,
return_X_y=True)

# Remove redundant and non-feature columns
X_adult = X_adult.drop(["education-num", "fnlwgt"],
axis="columns")

print(X_adult.dtypes)
```

Resultados:

Con respecto a la exploración de los datos, tenemos que identificamos la distribución de los registros normales y aquellos afectados por ataques MITM. Observamos una proporción de 80 y 20, asi mismo la identificación de las clases, los atributos, los tipos de datos y el propósito de cada uno de los datos.

Con respecto a la visualización gráfica de los datos tenemos que utilizando Matplotlib, generamos gráficos que representan la distribución de las variables más importantes. Estos gráficos nos ayudaron a visualizar patrones y tendencias en los datos. Hicimos un acercamiento a la herramienta sugerida Power Bi, estamos aún en una fase inicial, buscamos con esta nueva herramienta mejorar la visualización y obtener gráficos más interactivos y detallados, ampliando las posibilidades de visualización..

Hablando de las bibliotecas, tenemos que para el procesamiento de datos, pues ocupamos numpy y pandas, he de mencionar que anteriormente había utilizado numpy asi como sympy para implementar algoritmos de la asignatura de métodos numéricos, hemos de mencionar que pandas ayuda mucho a seccionar, limpiar los datos y transformar. Utilizando Scikit-learn, aplicamos códigos básicos de los del tutorial dado por la documentación, pero logramos captar que esta biblioteca nos ayudará a llevar a cabo los algoritmos de regresión, clasificación y clustering. Con respecto a Hyperopt entiendo que busca mejorar el rendimiento de los modelos, pero me falta experimentarla más.. Por lo cual creo que sentamos una base importante para las etapas posteriores.

Finalmente, logré ver la importancia de la calidad de los datos para obtener resultados precisos. Identificamos y manejamos datos faltantes y anómalos, mejorando la fiabilidad de nuestros análisis.

Conclusiones: En esta etapa, nos sumergimos en el mundo de los datos y su procesamiento. Nos familiarizamos con bibliotecas esenciales como Numpy, Pandas, Scikit-learn, y Hyperopt, y comenzamos a explorar Power BI para mejorar la visualización de gráficos, aunque inicialmente utilizamos Matplotlib. Aprendimos la terminología y la teoría fundamentales, aplicándolas en la práctica con la ayuda de estas bibliotecas. Esto nos permitió ejecutar muchas de las operaciones y fórmulas necesarias para procesar los datos de manera efectiva. Además, el artículo que revisamos nos proporcionó un contexto valioso, preparando el camino para desarrollar soluciones sólidas en las próximas etapas.

Referencias:

1. *Hyperopt Documentation*. (s. f.). https://hyperopt.github.io/hyperopt/

2. Hady, A. A., Ghubaish, A., Salman, T., Unal, D., & Jain, R. (2020). Intrusion Detection System for Healthcare Systems Using Medical and Network Data: A

Comparison Study. *IEEE Access*, *8*, 106576-106584. https://doi.org/10.1109/access.2020.3000421

3. *User Guide*. (s. f.). Scikit-learn. https://scikit-learn.org/stable/user_guide.html

4. *Ganancia de entropía e información en árboles de decisión*. (2020, 4 diciembre). ICHI.PRO. https://ichi.pro/es/ganancia-de-entropia-e-informacion-en-arboles-de-decision-219745150542483

5. *NumPy: the absolute basics for beginners — NumPy v2.0 Manual*. (s. f.). https://numpy.org/doc/stable/user/absolute_beginners.html

6. *Release Highlights for scikit-learn 1.4*. (s. f.). Scikit-learn. https://scikit-learn.org/stable/auto_examples/release_highlights/plot_release_highlights_1_4_0.html#sphx-glr-auto-examples-release-highlights-plot-release-highlights-1-4-0-py