Familiarization with a Work Environment for Analyzing Traffic Network Using AI

Summary: Ensuring patient information privacy, as well as preventing its manipulation before diagnosis and treatment when working with an IoT system, is extremely important. Therefore, we first need to know the data we have, understand what each data point pertains to, and identify the tools we can use to perform an adequate job.

Introduction: In this first delivery, our goal is to familiarize ourselves with our work environment and start experimentally manipulating the data, as well as exploring the libraries and functions we will work with to generate an efficient solution. We will work with a dataset containing records related to healthcare, this information attempts to be intercepted and modified in transit, and later we will look for ways to detect these attacks. Therefore, at this stage, we will investigate the content and orientation of the data to perform a more precise analysis. We seek to ensure that the data arrives intact at its destination for proper treatment and diagnosis, as well as to safeguard patient privacy.

Dataset:

Dataset description: The dataset houses over 16,000 records related to healthcare, combining biometric data with network flow. This dataset includes both normal records and records with MITM (Man-In-The-Middle -> packet alteration in transit).

Dataset:

1. **Dataset description:** The dataset houses over 16,000 records related to healthcare, combining biometric data with network flow. This dataset includes both normal records and records with MITM (Man-In-The-Middle -> packet alteration in transit).

2. **Arquitectura/testbed:**

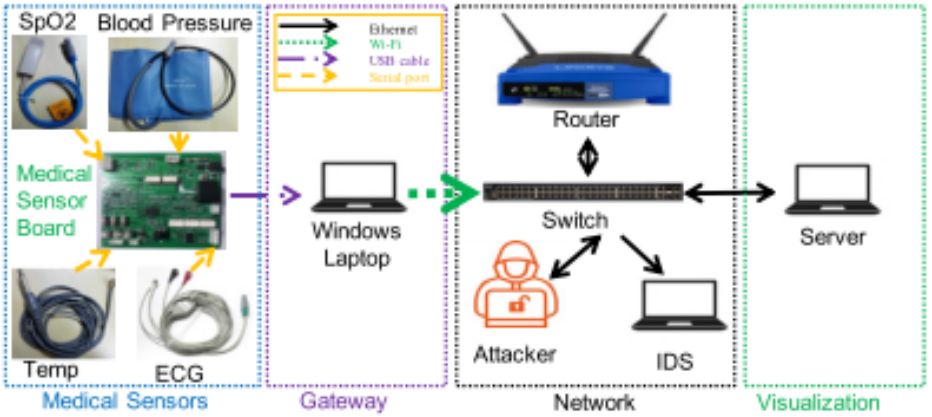

Fig 1. EHMS testbed [2]

The medical board is connected to a Windows OS computer via USB port, and software in C++ captures the sensed data. The computer acts as the gateway, transferring the data to the server via WiFi using the TCP/IP protocol.

All machines are connected via Ethernet except the gateway. The switch is connected to the Internet through the router, to which the gateway is connected via 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. **Classes:**
   Intrusion
   No intrusion

Analysis of the traffic network: The system uses the protocol TCP / IP

Experimentation:
I worked with matplotlib for generating the graphs, so to reach the results, I had to load the dataset into a DataFrame with the help of the pandas library.

1. **Class-Frequency Distribution:** For the analysis of class distribution, we analyzed the labels, that is, how many records are classified as normal and which ones are classified as an attack, where 0 is normal (no intrusion) and 1 indicates an intrusion (attack).

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

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

df = pd.DataFrame(EHMS)

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
```

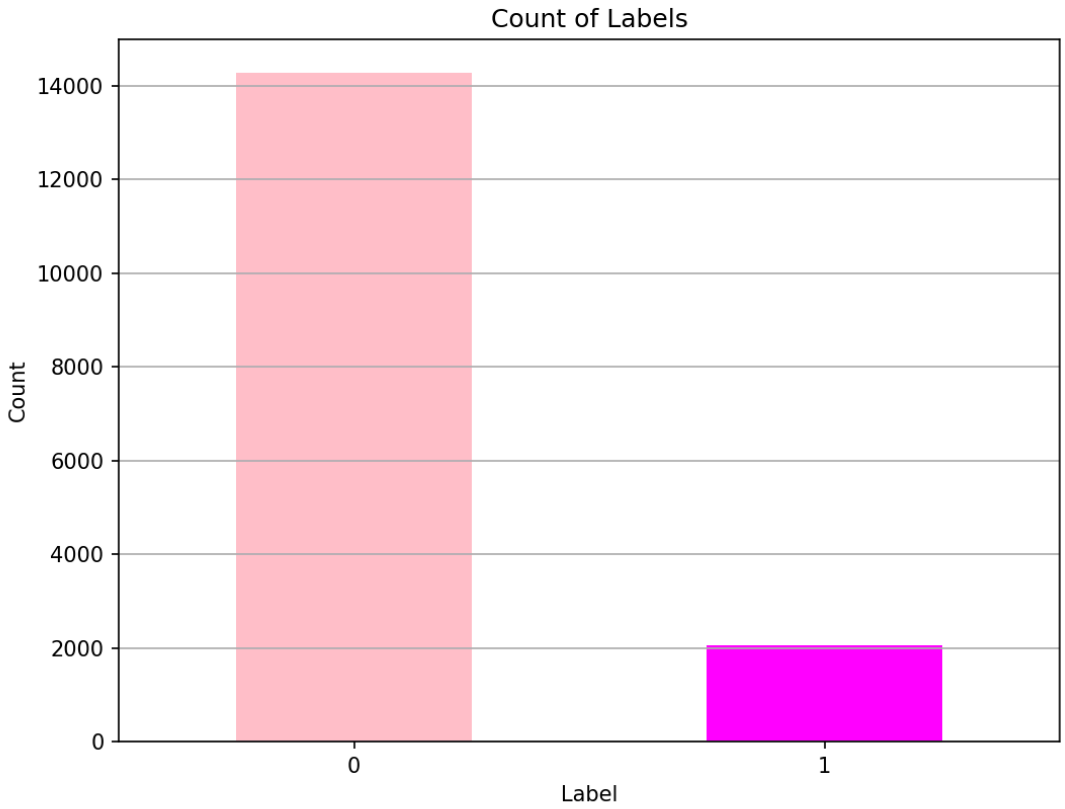

Fig.2 Distribution of the classes - frequency

2. **Attribute-Frequency Distribution:** For the distribution of attributes, a graph of the biometric data was made, as the other data pertain to network and communication, and the transfer sizes and such aspects are constant. If you wish to verify, the code is commented.

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

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

df = pd.DataFrame(EHMS)

#We change manually this attribute 'cause the system did not
detect it as categorical
df['Dport'] = df['Dport'].astype('category')

# Analysis for columns
column_analysis = []

for column in df.columns:
```

```
    col_type = df[column].dtype

    if 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
        })
analysis_df = pd.DataFrame(column_analysis)

# Show the analysis
print(analysis_df)


#Graph the biometric data
biometric_columns = ['Temp', 'SpO2', 'Pulse_Rate', 'SYS', 'DIA',
'Heart_rate', 'Resp_Rate','ST']

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()
```

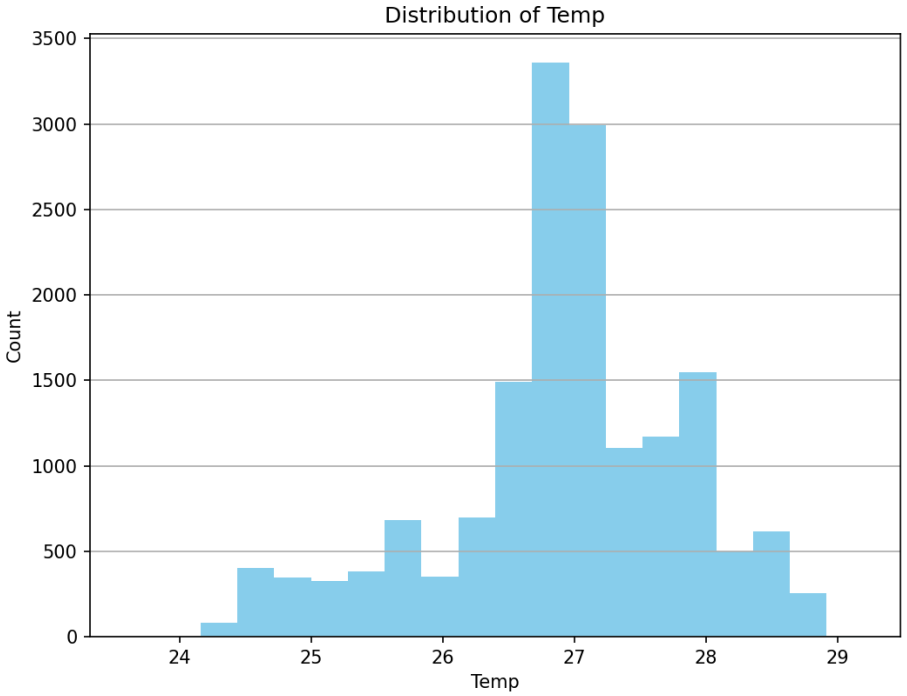

Fig.3 Distribution of temperature

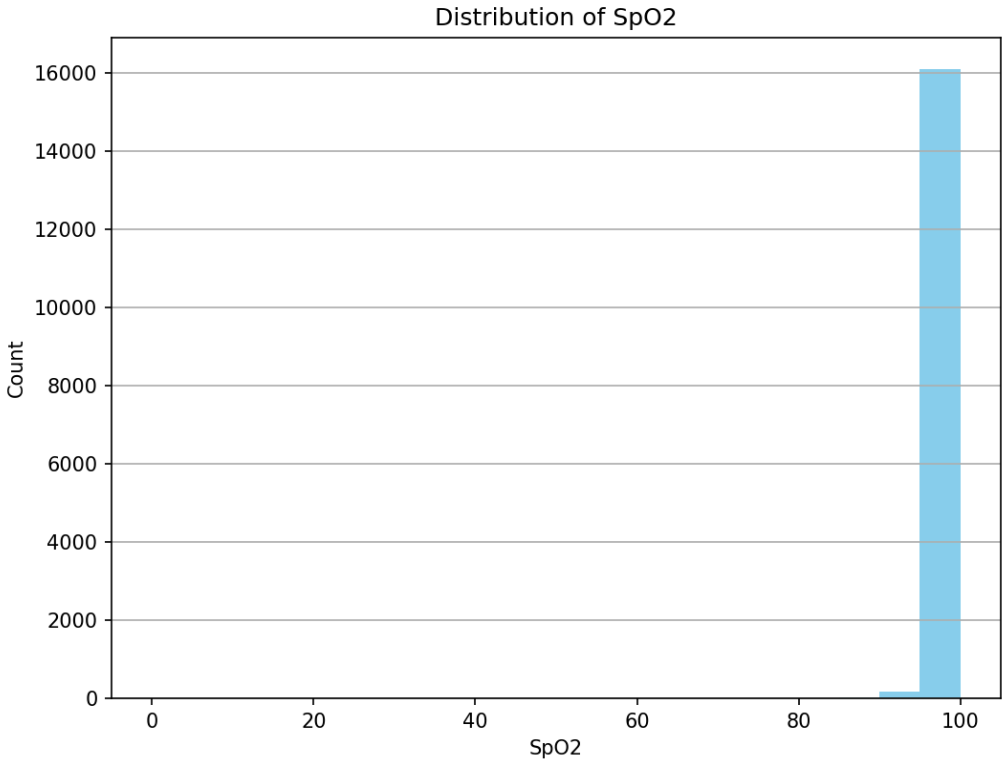

Fig.4 Distribution of the blood oxygen saturation

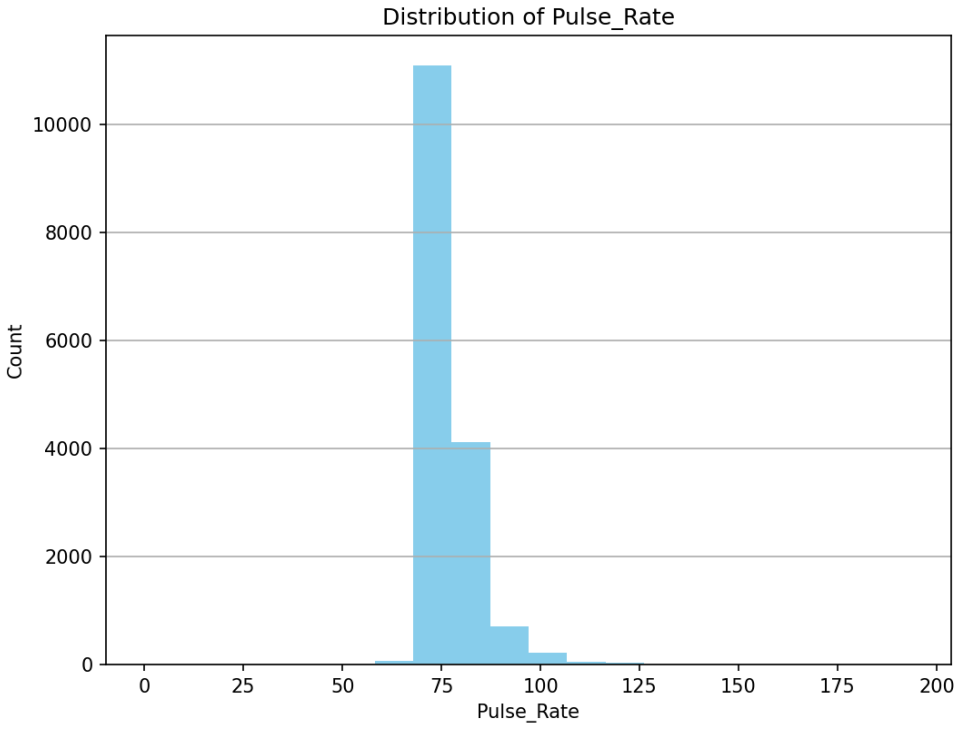

Fig.5 Distribution of pulse rate

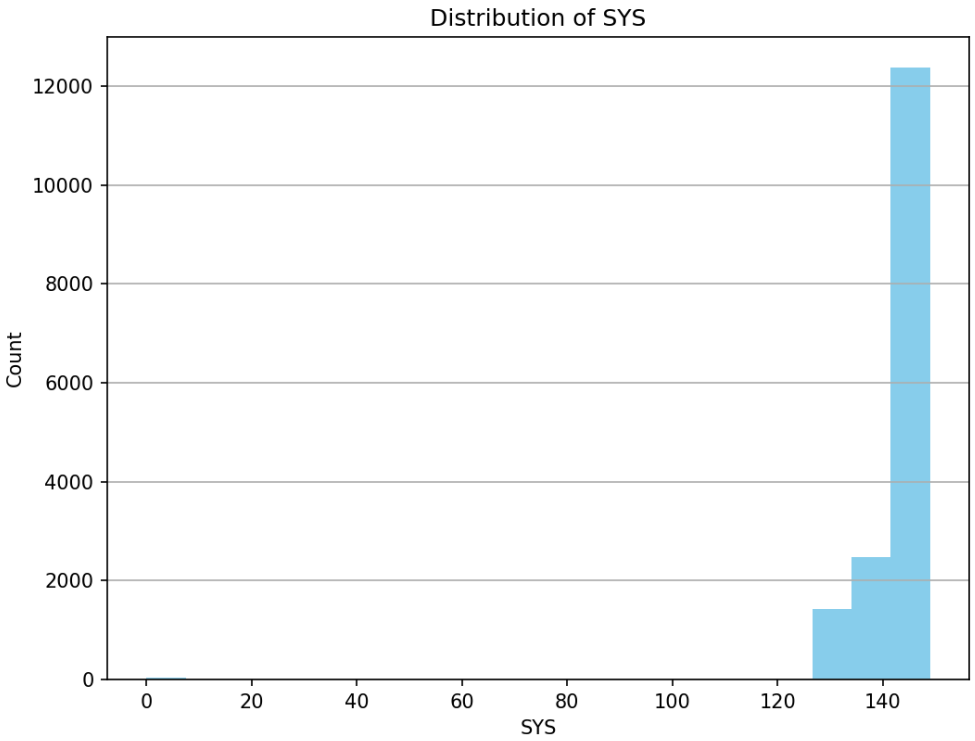

Fig.6 Distribution of the systolic blood preassure

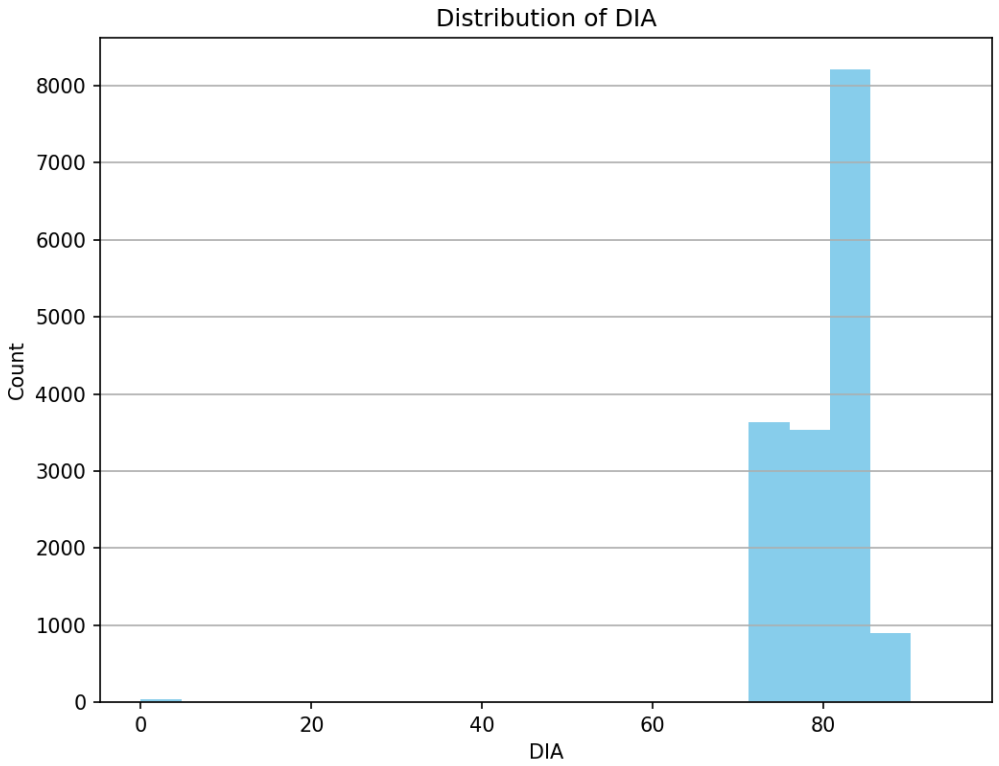

Fig.7 Distribution of the diastolic blood preassure

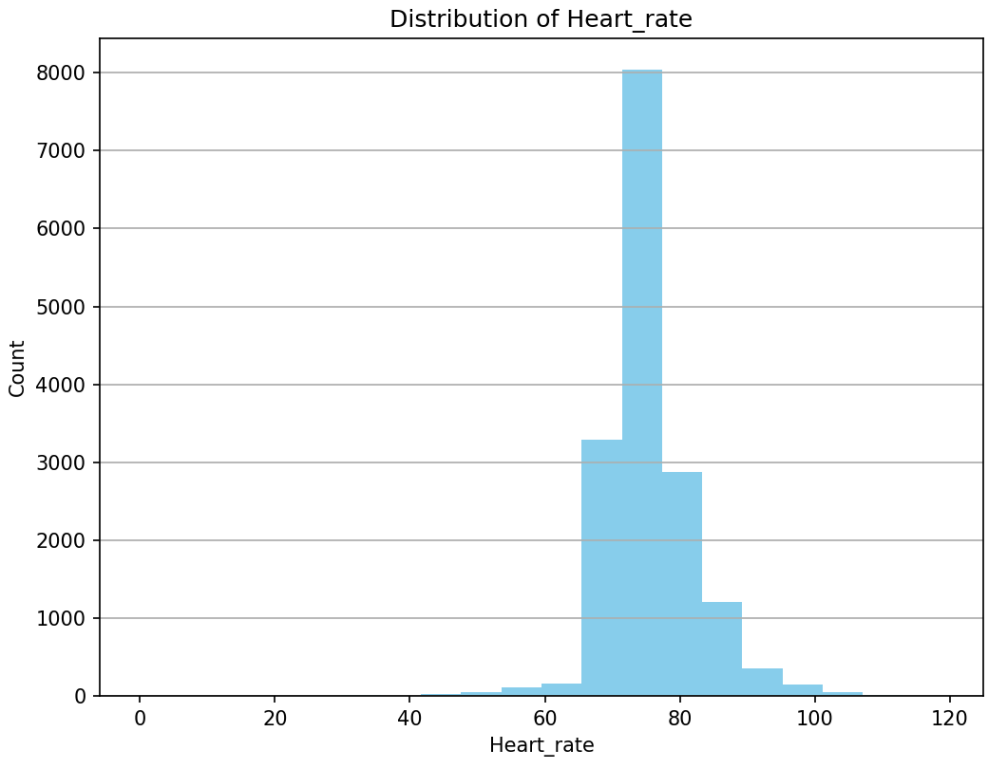

Fig.8 Distribution of heart rate

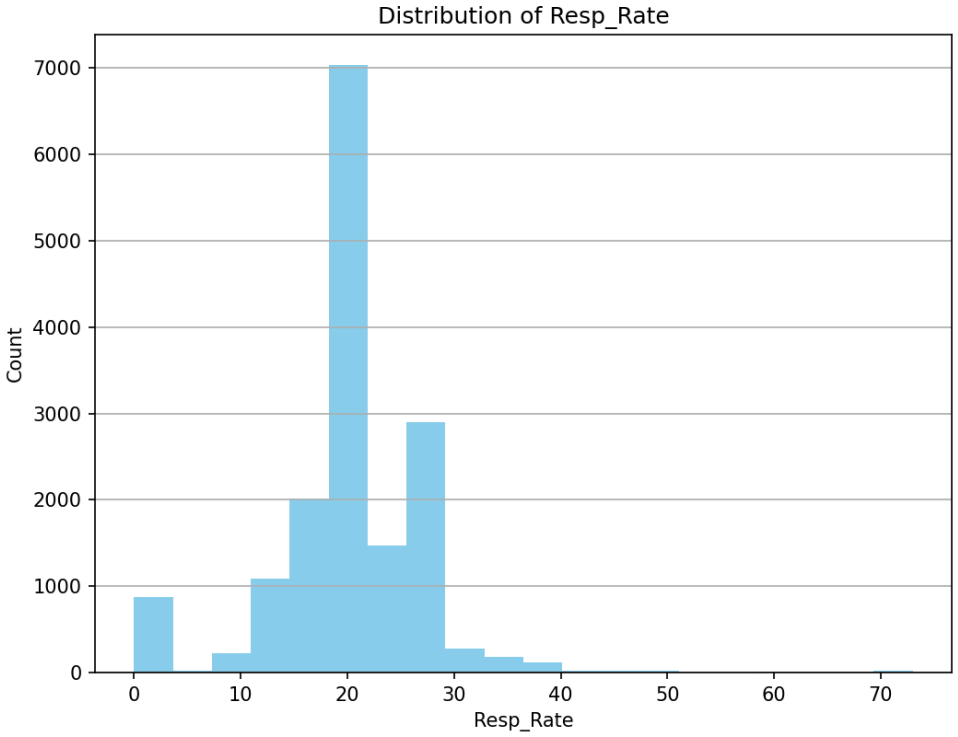

Fig 9. Distribution of theRespiration Rate

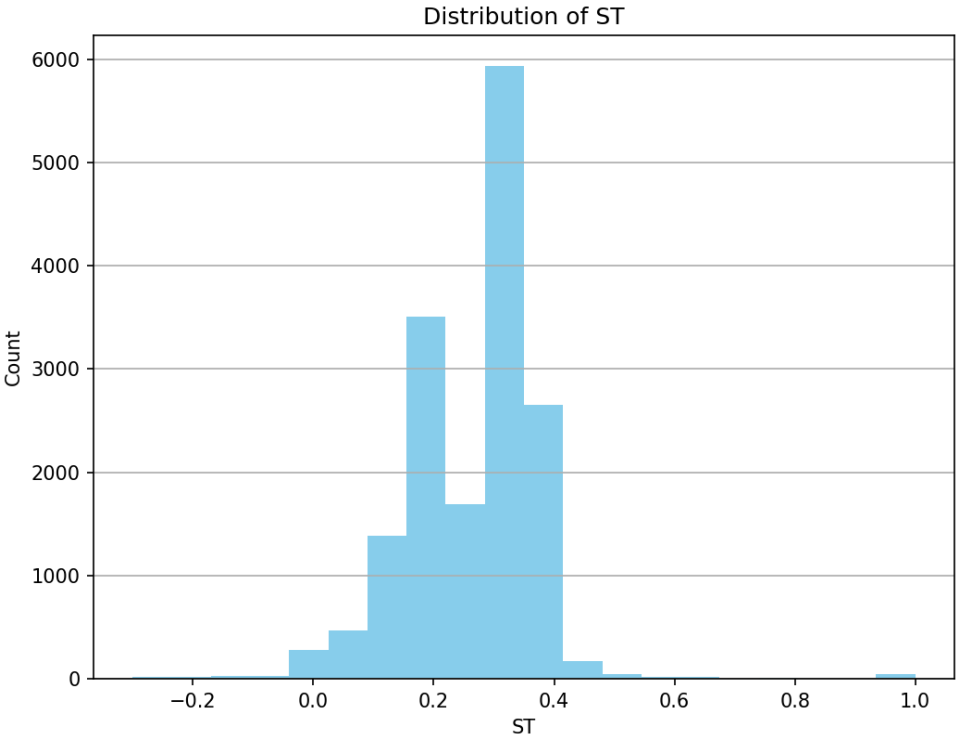

Fig.10 Distribution of ECG ST segment

We did not graph the flow metric data because they have the same value, so we have the same value in all the registers, so we get only one bar as a graph, we are going to process this data in the next iteration.

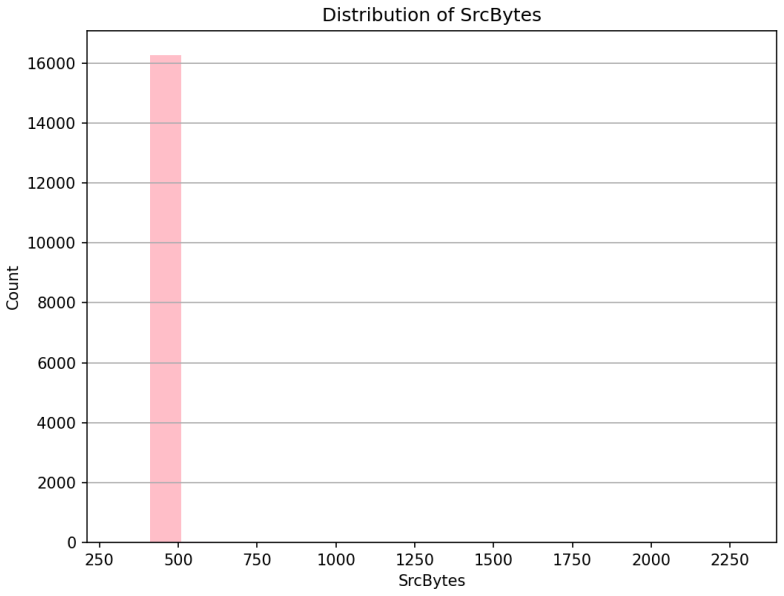

Fig.11 Distribution of the SrcBytes

3. **Distribution of Characteristic Formats (categorical,numeric)**
   To do this distribution we used the function, it help us to select a data type from the DataFrame, to get the categorical attributes we get the ones that are classified as objects.

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

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

df = pd.DataFrame(EHMS)

# We manipulated this attribute 'cause the system detected it as
numeric when is categorical
df['Dport'] = df['Dport'].astype('object')

# Identify the numeric and cathegorical attributes
categorical_columns =
df.select_dtypes(include=['object']).columns
numerical_columns = df.select_dtypes(include=['number']).columns

# Count the numeric and cathegorical attributes
categorical_count = len(categorical_columns)
numerical_count = len(numerical_columns)

# Graph
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
```

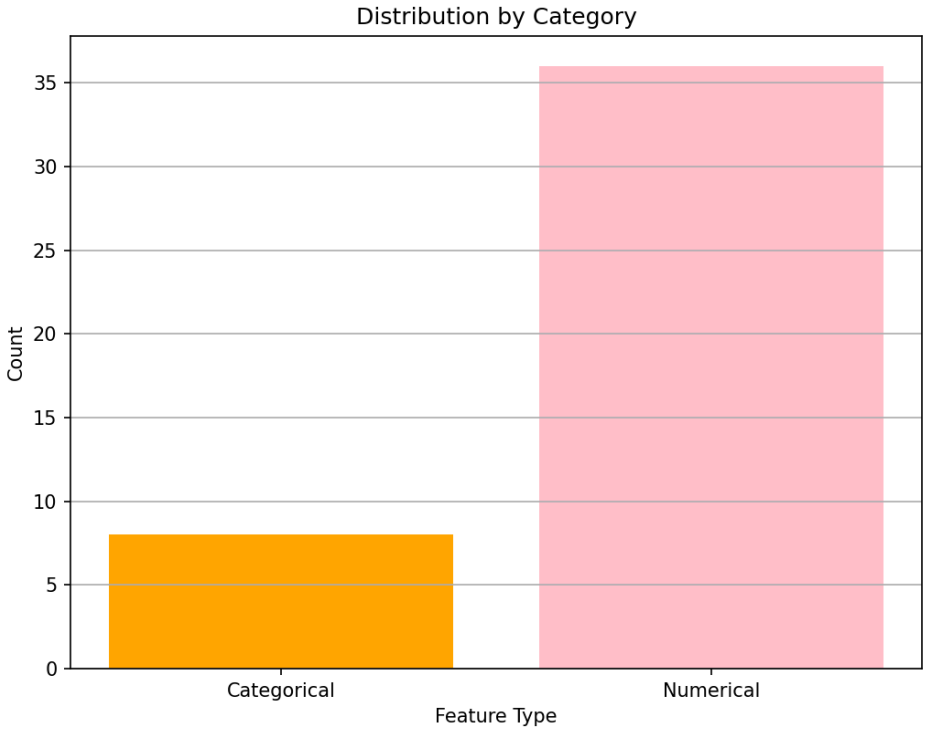

Fig.12 Distribution by attributes format (categorical,numeric)

4. **Distribution of the attributes related to the traffic network-comunication protocols**

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

# Upload the dataset
EHMS= '../WUSTL-EHMS/wustl-ehms-2020.csv'
df = pd.read_csv(EHMS)

# Attributes related to the comunication protocol 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)
```

```
# Grafph how many attributes are part of the IoT attributes
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-comunication protocols')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
```

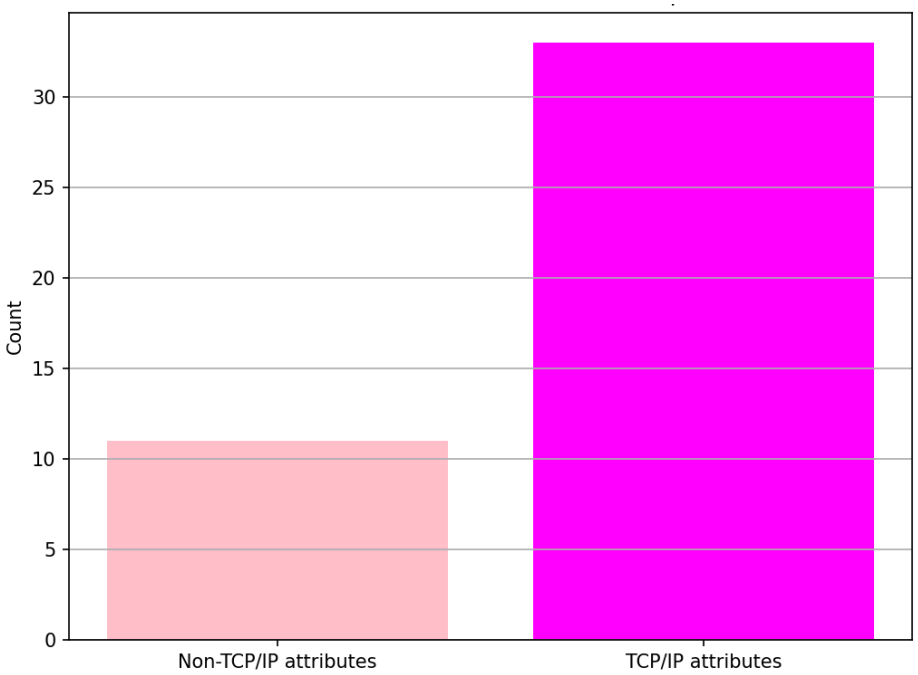

Fig. 13 Distribution of te attributes realted to the traffic network-protocols of comunication

5. Distribution of the IoT devices-app

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

EHMS= '../WUSTL-EHMS/wustl-ehms-2020.csv'
```

```
df = pd.read_csv(EHMS)

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)

# Graph how many attributes are related to IoT
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 the IoT devices-app')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
```

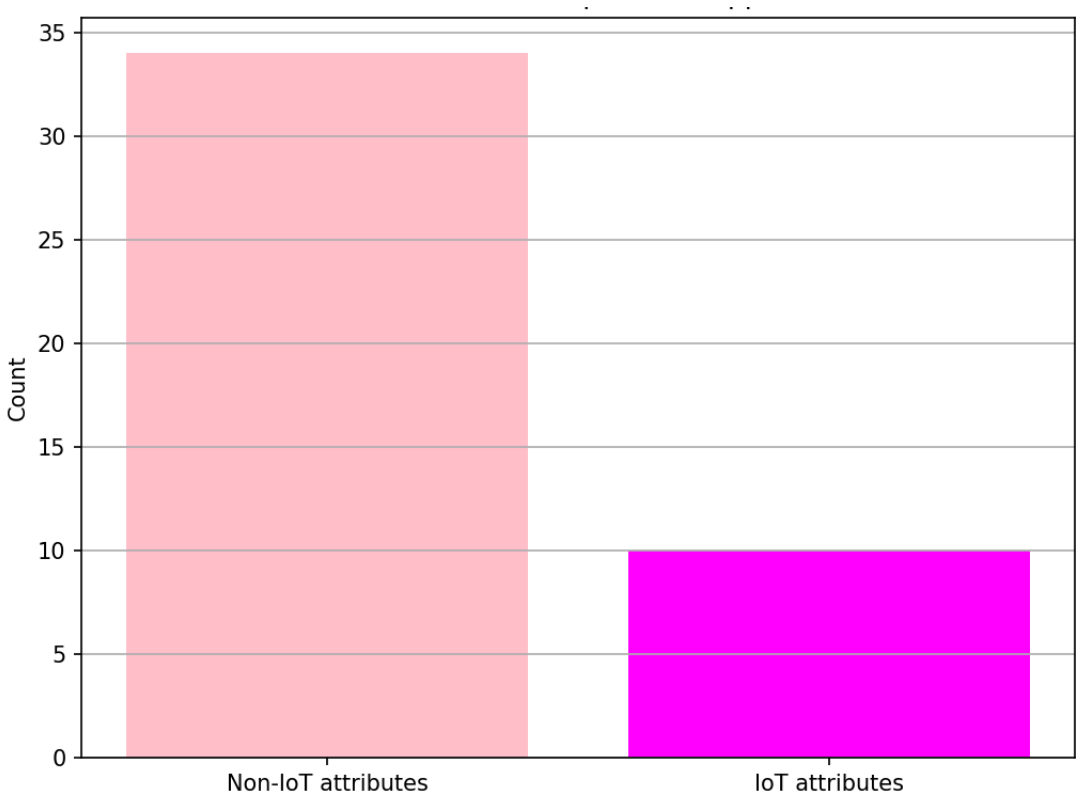

Fig. 14 Distribution od the IoT devices-app

We also did some tests with the libraries, I used some code from Github repositories and examples on blogs

```
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']))
```

Also start to work with the scikit-learn library

```
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)
```

Results:

Regarding data exploration, we identified the distribution of normal records and those affected by MITM attacks. We observed an 80 to 20 ratio, as well as the identification of classes, attributes, data types, and the purpose of each piece of data.

In terms of graphical data visualization, using Matplotlib, we generated graphs representing the distribution of the most important variables. These graphs helped us visualize patterns and trends in the data. We approached the suggested tool Power BI, and we are still in an initial phase, seeking to improve visualization with this new tool to obtain more interactive and detailed graphs, expanding visualization possibilities.

Speaking of libraries, for data processing, we used numpy and pandas. I should mention that I had previously used numpy as well as sympy to implement algorithms for the numerical methods course. Pandas helps a lot in sectioning, cleaning, and transforming data. Using Scikit-learn, we applied basic codes from the tutorial provided by the documentation but managed to understand that this library will help us carry out regression, classification, and clustering algorithms. Regarding Hyperopt, I understand that it seeks to improve model performance, but I need to experiment with it more. Therefore, I believe we laid an important foundation for the subsequent stages.

Finally, I realized the importance of data quality for obtaining accurate results. We identified and handled missing and anomalous data, improving the reliability of our analyses.

Conclusions: In this stage, we delved into the world of data and its processing. We familiarized ourselves with essential libraries like Numpy, Pandas, Scikit-learn, and Hyperopt, and began exploring Power BI to enhance graph visualization, although we initially used Matplotlib. We learned the fundamental terminology and theory, applying them in practice with the help of these libraries. This allowed us to execute many of the necessary operations and formulas to process data effectively. Additionally, the article we reviewed provided us with valuable context, paving the way for developing solid solutions in the upcoming stages.

References:

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