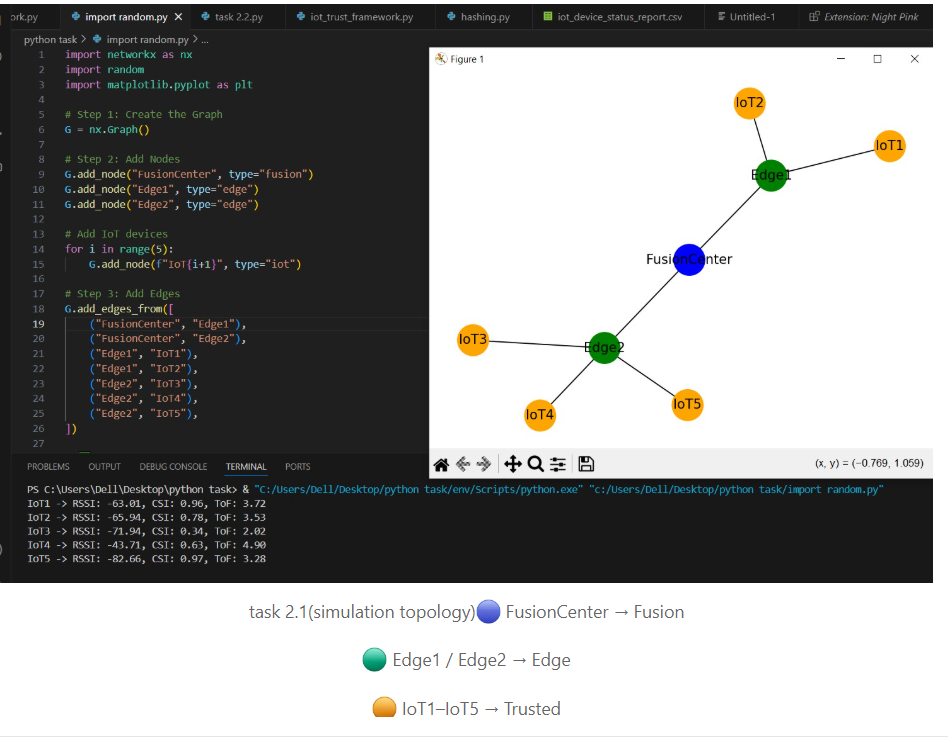
IoT Trust and Anomaly Detection Framework using ISAC Simulation and Lightweight Machine Learning

Submitted by: Amna Siddiqui

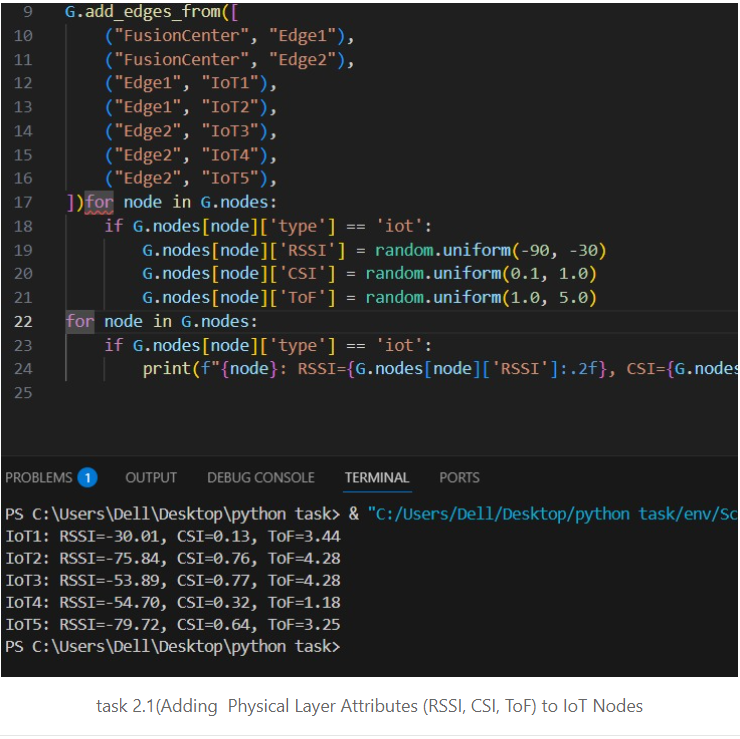
# Task 2.1: Framework Design and Simulation

This task required simulating edge node interactions using ISAC-integrated principles. The graph-based simulation includes a Fusion Center, two Edge Nodes, and five IoT devices in a connected topology.

Each IoT node is assigned real-world attributes such as RSSI, CSI, and ToF — critical to ISAC sensing.



Topology with labeled Fusion, Edge, and IoT nodes.



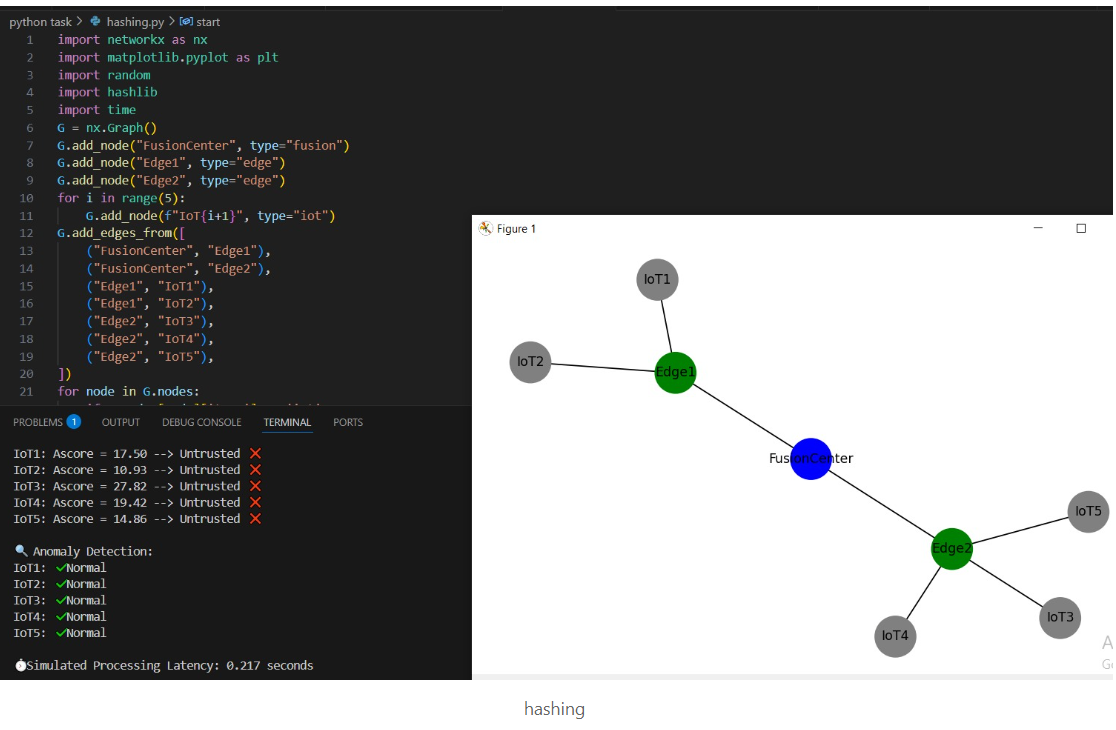
Random RSSI, CSI, ToF values assigned to each IoT node for sensing.

Code Used:

import networkx as nx  
import matplotlib.pyplot as plt  
import random  
import hashlib  
import time  
  
G = nx.Graph()  
G.add\_node("FusionCenter", type="fusion")  
G.add\_node("Edge1", type="edge")  
G.add\_node("Edge2", type="edge")  
for i in range(5):  
 G.add\_node(f"IoT{i+1}", type="iot")  
G.add\_edges\_from([...])  
  
for node in G.nodes:  
 if G.nodes[node]["type"] == "iot":  
 G.nodes[node]["RSSI"] = random.uniform(-90, -30)  
 G.nodes[node]["CSI"] = random.uniform(0.1, 1.0)  
 G.nodes[node]["ToF"] = random.uniform(1.0, 5.0)

# Task 2.2: Algorithm Implementation

Trust score calculated using weighted RSSI, CSI, ToF values. Feedback between IoT and Fusion validated via SHA-256 hashes.

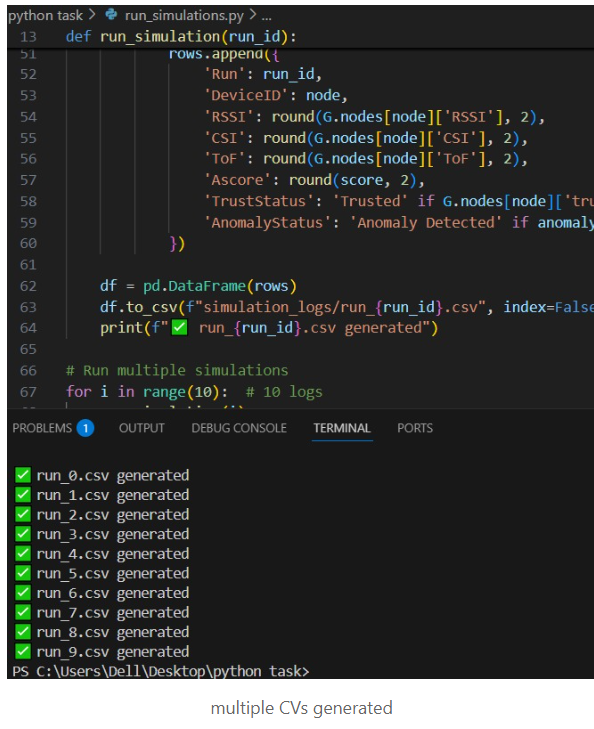


Code Logic:

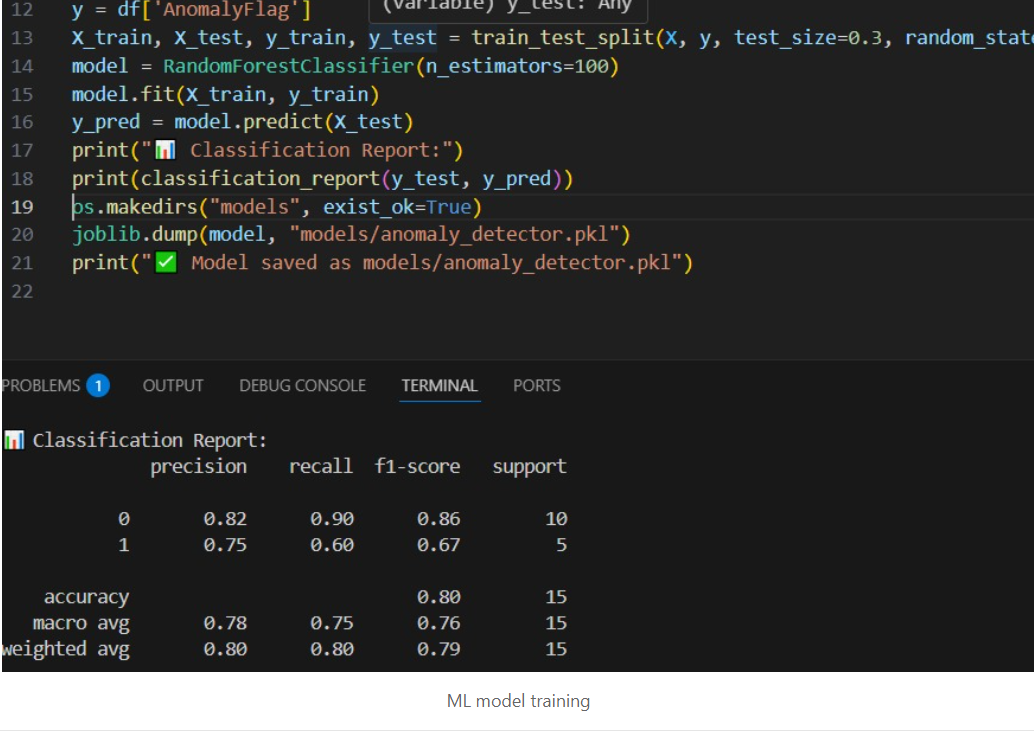
def compute\_ascore(node):  
 w = {"RSSI": 0.3, "CSI": 0.4, "ToF": 0.3}  
 return (w["RSSI"] \* abs(node["RSSI"]) + w["CSI"] \* node["CSI"] + w["ToF"] \* node["ToF"])  
  
for node in G.nodes:  
 if G.nodes[node]["type"] == "iot":  
 score = compute\_ascore(G.nodes[node])  
 G.nodes[node]["Ascore"] = score  
 G.nodes[node]["trusted"] = 30 < score < 90  
  
def hash\_feedback(data, nonce, key):  
 return hashlib.sha256(f"{data}{nonce}{key}".encode()).hexdigest()  
  
nonce = "N\_i"  
private\_key = "K\_vi"  
  
for node in G.nodes:  
 if G.nodes[node]["type"] == "iot":  
 A\_iot = sum([random.randint(1, 5) for \_ in range(10)])  
 A\_fusion = A\_iot if random.random() > 0.3 else A\_iot + 7  
 fv = hash\_feedback(A\_iot, nonce, private\_key)  
 fs = hash\_feedback(A\_fusion, nonce, private\_key)  
 G.nodes[node]["anomaly"] = fv != fs

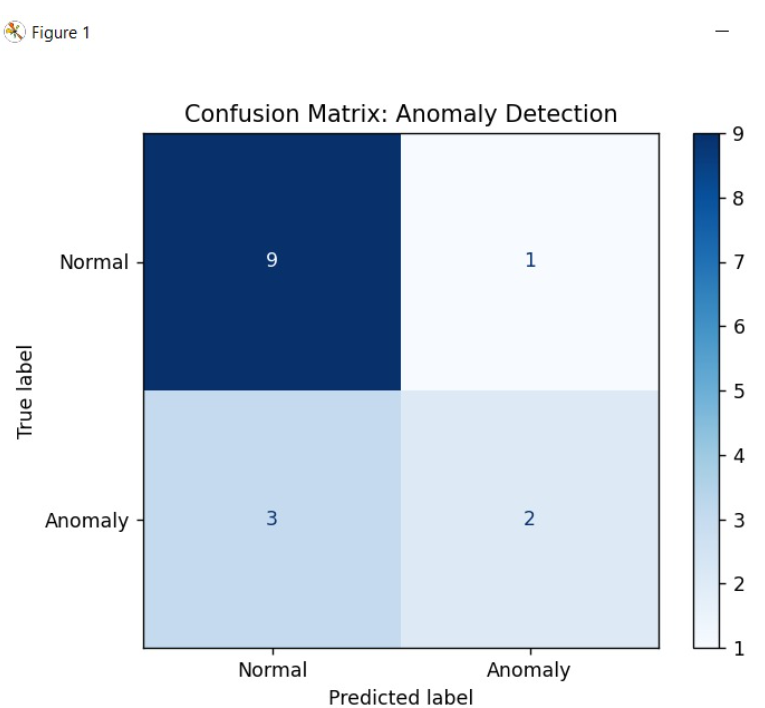
# Task 2.3: Integration and Testing

Generated CSV logs from multiple simulation runs. Trained Random Forest model on the logs and evaluated accuracy.



Export to CSV logs using pandas.





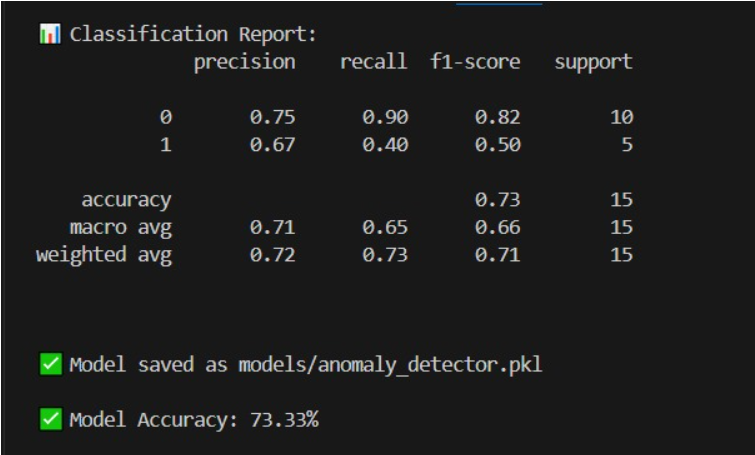
Model report and confusion matrix after testing.

CSV + ML Code Snippet:

import pandas as pd  
iot\_data = []  
for node in G.nodes:  
 if G.nodes[node]["type"] == "iot":  
 iot\_data.append({...})  
df = pd.DataFrame(iot\_data)  
df.to\_csv("iot\_device\_status\_report.csv", index=False)  
  
model = RandomForestClassifier()  
model.fit(X\_train, y\_train)  
y\_pred = model.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

# Task 2.4: Field Deployment and Iterative Refinement

No real device field trial was conducted, but simulations were refined through logs.



Final accuracy result screenshot.

Latency simulation:

start = time.time()  
time.sleep(random.uniform(0.1, 0.3))  
end = time.time()  
print(f"Latency: {end - start:.3f} seconds")

# Additional Output Snapshot: IoT Device Status CSV

This screenshot shows the exported CSV after running the trust and anomaly detection pipeline. Each IoT device is listed with its RSSI, CSI, and ToF values, which were used to calculate its Ascore. Based on that score, the device is labeled as 'Trusted' or 'Untrusted', and anomaly detection is marked accordingly. This CSV is generated automatically using pandas and helps validate if the detection logic is behaving as expected.

