**3.1 MATLAB SIMULATIONS**

The WSCC -9 bus system was implemented in Matlab and serves as the baseline model for all simulations. All naturally occurring events like load changes as well faults were simulated in this. Data was collected from each of the circuit breakers present in the line. This provided instantaneous snapshots of the entire bus system. The data was then extracted to an excel sheet for each line. Five different types of faults- LG, LL, LLG, LLL and LLLG faults were simulated in all the lines. These faults were simulated in each line at varying. The table 3.1 (Refer Fig 2.2) indicates the faults in each line and the particular distances at which they have occurred. Through this process, data was collected for natural events from the system.

**Table 3.1: Different Line Notations**

|  |  |
| --- | --- |
| Notation | Buses |
| a | 4-5 |
| b | 4-6 |
| c | 5-7 |
| d | 6-9 |
| e | 7-8 |
| f | 8-9 |

**Table 3.2: Fault types and distances in each line**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fault Location | LG | LLG | LL | LLL | LLLG |
| a | 20-130,40-110,60-90 | 20-130,40-110,60-90 | 20-130,40-110,60-90 | 20-130,40-110,60-90 | 20-130,40-110,60-90 |
| b | 20-100,40-80,60-60 | 20-100,40-80,60-60 | 20-100,40-80,60-60 | 20-100,40-80,60-60 | 20-100,40-80,60-60 |
| c | 20-100,40-80,60-60 | 20-100,40-80,60-60 | 20-100,40-80,60-60 | 20-100,40-80,60-60 | 20-100,40-80,60-60 |
| d | 20-120,40-100,60-80 | 20-120,40-100,60-80 | 20-120,40-100,60-80 | 20-120,40-100,60-80 | 20-120,40-100,60-80 |
| e | 20-90,40-70,60-50 | 20-90,40-70,60-50 | 20-90,40-70,60-50 | 20-90,40-70,60-50 | 20-90,40-70,60-50 |
| f | 20-90,40-70,60-50 | 20-90,40-70,60-50 | 20-90,40-70,60-50 | 20-90,40-70,60-50 | 20-90,40-70,60-50 |

**3.2 CYBER ATTACKS**

**3.2.1 Setup Modbus Client - Server architecture**

The Modbus setup consists of a two-tier architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. In our case, the PLCs or RTUs are the client which send data or make requests to the server which is the Control Centre consisting of Master Terminal Units (MTUs).

The library used for this is the PyModbus v3.0.0, a full Modbus protocol implementation using twisted/tornado/asyncio for its asynchronous communications core. Since the library is written in python, it allows for easy scripting and/or integration into their existing solutions.

**3.2.2 Encode data to Modbus format**

Data encoded in the Modbus format contains various headers and protocol syntax requirements to enable it to be transmitted over the internet in the form of Modbus Payload. This can be achieved using the PyModbus library and its built-in functions to encode and decode the data and its corresponding payload.

This encoded payload is sent from the client to the server over the internet and stored in the holding registers. The client then sends a GET or Fetch request to the server and receives the encoded data from the database. This is payload is then finally decoded back to raw data using PyModbus.

**3.2.3 Setup cyber-attack architecture**

The cyber-attack architecture consists of three machines, real or virtual. A communication is established between two of the devices over the internet via the Modbus protocol. A third machine will perform the cyber-attack by inserting itself in between the other two in such a way that all the communication is routed through it.

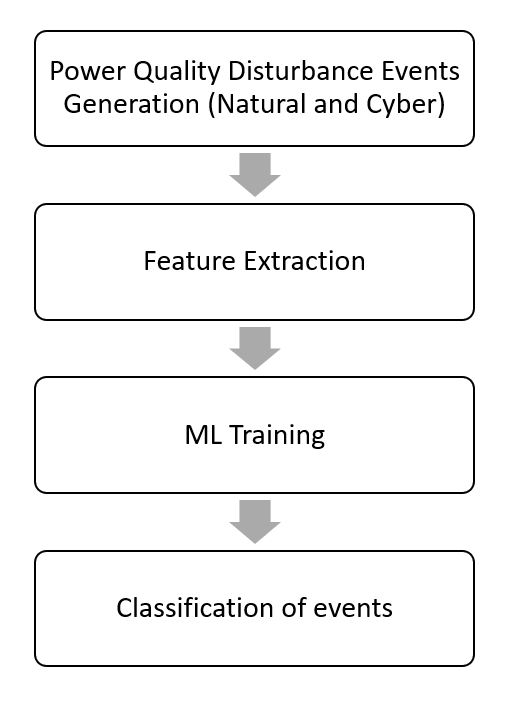
**3.2.4 Perform Man-in-the-middle attacks**

A MITM attack was performed with an Ettercap tool that maliciously modified the Modbus TCP commands between the Master and PLC workstations. Weaknesses of the Modbus TCP that were targeted by the Ettercap tool involved sending commands in clear text and lack of authentication within the Modbus TCP protocol. The Attacker workstation then uses the MAC addresses provided by the Ettercap scan for a MITM with ARP poisoning (ARP spoofing) to send falsified ARP messages. The ARP spoofing results in the linking of the attacker’s MAC address with the IP address of a legitimate computer or server on the network.

**3.2.5 Gather cyber-attack data**

Generate and gather the data produced from conducting cyber-attacks on the network. Perform data wrangling, feature extraction, etc on the gathered data to prepare it for the machine learning model.

**3.3 MACHINE LEARNING TECHNIQUES**

The data collected from the natural events and cyber-attacks were first combined and consolidated. This was followed by performing feature extraction on this data. Pre-processing is the initial step for solving problems related to machine learning. It is a method of converting raw data into a format suitable for training the machine learning model. Raw data is noisy, incomplete, and inconsistent. Figure 3.1 indicates the steps followed from data generation to classification of events.

**Figure 3.1 Methodology for machine learning**

Pre-processing steps:

* Handling of missing or NaN values

Identifying and coping up with missing values is essential for effective cyberattack identification research. There are several methods of handling missing values like average value, majority value or replacing with zero. Out of all these methods, replacing with the NaN values with zero tends to produce the best results.

* Data Standardization

Data standardization is essential before implementing ML algorithms, as data standardization can significantly impact the outcome of the ML training model. Therefore, it is very crucial to have all the data on the same scale. Standard scaler approach was chosen to standardize the data by keeping mean 0 and variance 1.

This pre-processed data was then split into training and test set using a split ratio of 80:20. This is served as input to the machine learning algorithms. For the proposed model and training, evaluation criteria were set on the accuracy, recall, precision, and F1 score, which are given as