University of Delaware

Engineering Machine Learning Systems
Checkpoint 1

Saeid Rajabi
Department Of Electrical and Computer Engineering
College of Engineering

Contents

1 Dataset Collection **2**

2 Data Preprocessing **3**

3 Exploratory Data Analysis (EDA) **4**

4 Initial Thoughts on Model Selection **6**

1 Dataset Collection

The dataset originates from an IoT botnet data source from UCI [1], containing network traffic data from various IoT devices under benign and attack conditions. The devices include the Danmini Doorbell, Ecobee Thermostat, Philips Baby Monitor, and several security cameras.

Instances: Around 7 million records. Features: The dataset has 125 features. These features represent various metrics and statistics on traffic.

The data of each device were stored separately and divided into directories for benign and attack traffic. The attack traffic data was subdivided by specific attack types (e.g., combo, junk, scan, tcp, udp).

I merged all the separate files into a single dataset. The final dataset is prepared for analysis and modeling, with each record labeled by its device type and classified as either benign or attack traffic.

Numerical features: Features include mean, variance, weight, and magnitude metrics for packet transmission levels (e.g., MI_dir_L5_mean, HH_L0.01_variance).

Categorical features: I represented device types with one-hot encoding. This method allows models to learn unique traffic patterns for each device, which helps detect attacks specific to each device type and start from all zeros for more efficiency.

Record counts per device:
Danmini_Doorbell: 49548 benign records, 968750 attack records
Ecobee_Thermostat: 13113 benign records, 822763 attack records
Ennio_Doorbell: 39100 benign records, 316400 attack records
Philips_B120N10_Baby_Monitor: 175240 benign records, 923437 attack records
Provision_PT_737E_Security_Camera: 62154 benign records, 766106 attack records
Provision_PT_838_Security_Camera: 98514 benign records, 738377 attack records
Samsung_SNH_1011_N_Webcam: 52150 benign records, 323072 attack records
SimpleHome_XCS7_1002_WHT_Security_Camera: 46585 benign records, 816471 attack records
SimpleHome_XCS7_1003_WHT_Security_Camera: 19528 benign records, 831298 attack records

2 Data Preprocessing

For binary classification, benign traffic was labeled as 0, and attack traffic as 1. For multi-class classification, benign traffic was labeled as 0, with each attack type assigned a unique label from 1 to 9. We assigned one-hot encoded vectors to each type of device, indicating which device generated the traffic.

After merging the datasets, we removed duplicates using *chunk_imputed.drop_duplicates* function. We examined the dataset for missing values, confirming that no records had missing data by using a chunk-based loading and verification process.

```
# Function to check for missing values in a chunk
def check_missing_values(chunk):
    missing_values = chunk.isnull().sum()
    total_missing = missing_values.sum()
    if total_missing > 0:
        print("Missing values in this chunk:")
        print(missing_values[missing_values > 0])
    else:
        print("No missing values in this chunk.")
```

I standardized all numerical features to maintain consistent scaling, which is essential for the performance of machine learning models. Using one-hot encoding for device types allowed the model to recognize distinct patterns in device-specific data.

The dataset was highly imbalanced, with over 90% of instances labeled as attacks. We used SMOTE and random undersampling to achieve a balanced distribution of 50% benign and 50% attack records.

Given the large size of the dataset, I loaded and processed it in chunks. This approach avoids memory overload and allows efficient handling of large datasets. For each chunk, SMOTE generated synthetic instances for the minority class, helping it reach a target proportion relative to the majority class. After processing each chunk, I saved the balanced data to a new file for further analysis, ensuring that both classes had comparable representation.

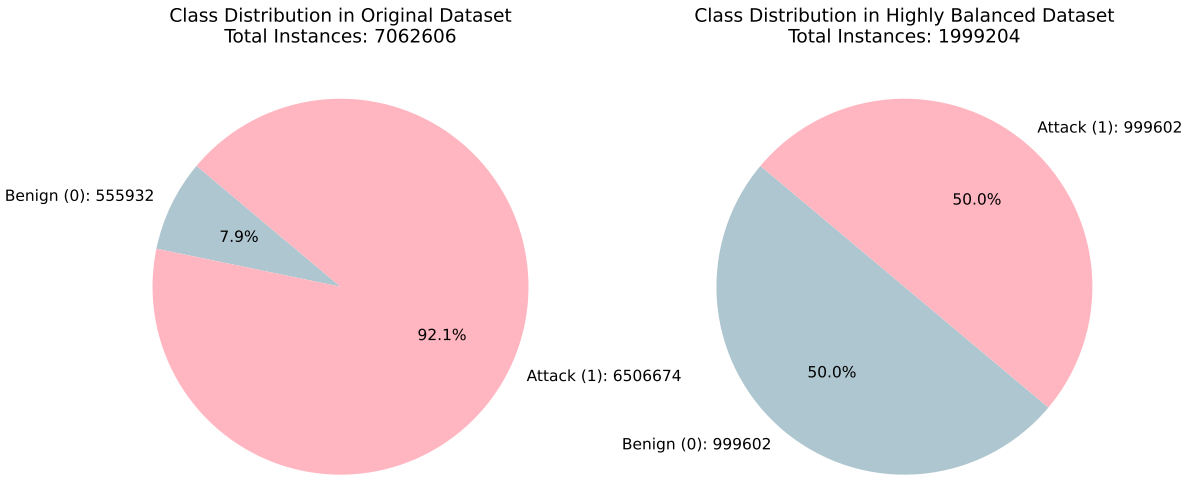

Figure 1: Class distributions before and after balancing

3 Exploratory Data Analysis (EDA)

I computed basic statistics for each feature (mean, median, mode, range).

A CSV file attached to this report contains all the statistics for all 115 features.

For the visualization, a Pearson correlation heatmap was created for numerical features. The heatmap identified redundant features, suggesting opportunities for dimensionality reduction in 2.

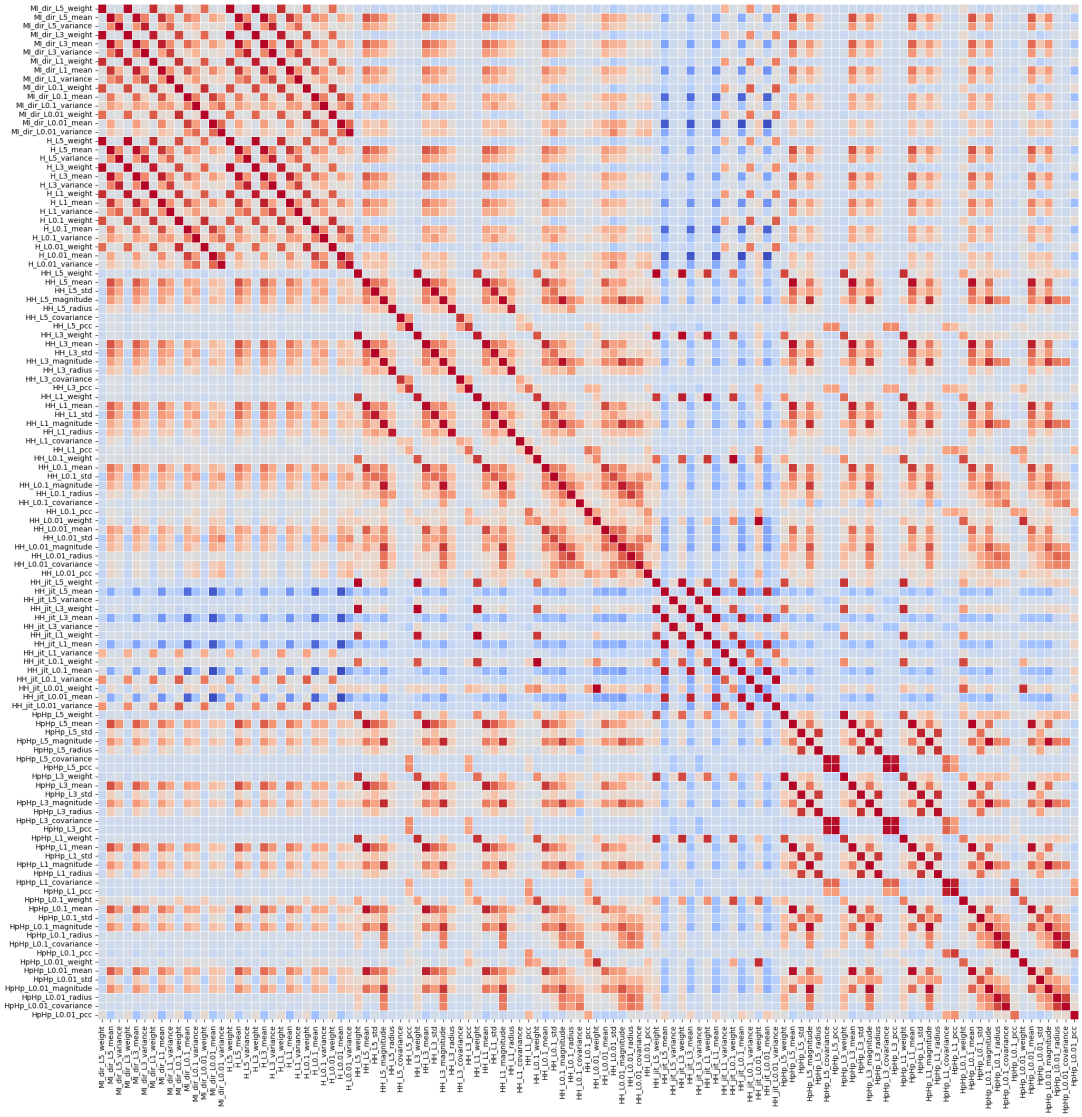

Figure 2: Heatmap between all the features with themselves

Table 1: Dataset Statistics

Feature	**Mean**	**Median**	**Mode**	**Range**
MI_dir_L5_weight	81.65	68.94	3138.76	261.29
MI_dir_L5_mean	178.13	167.31	9428.80	681.64
MI_dir_L5_variance	15383.57	15097.06	1052287.81	154122.32
MI_dir_L3_weight	129.20	109.56	5220.99	365.02
MI_dir_L3_mean	178.23	172.09	10301.65	603.21
MI_dir_L3_variance	17061.87	17285.07	858729.59	150873.96
MI_dir_L1_weight	367.05	316.11	14593.59	812.52
MI_dir_L1_mean	178.48	179.48	9745.88	480.71
MI_dir_L1_variance	18502.39	19367.82	522951.54	119373.83
MI_dir_L0.1_weight	3397.75	3026.49	154210.40	5772.74
MI_dir_L0.1_mean	178.85	185.73	8919.95	331.49
MI_dir_L0.1_variance	19209.55	20983.96	645031.62	92588.98
MI_dir_L0.01_weight	19722.85	18878.05	1090856.62	34552.47
MI_dir_L0.01_mean	178.74	187.50	9295.76	289.57
MI_dir_L0.01_variance	19456.00	21366.15	877342.62	75436.52
H_L5_weight	81.65	68.94	3138.76	261.29
H_L5_mean	178.13	167.31	9428.80	681.64
H_L5_variance	15383.57	15097.06	1052287.81	154122.32

4 Initial Thoughts on Model Selection

An MLP is a strong choice for my task due to its ability to learn complex, non-linear relationships across features. I need this capability to identify subtle patterns in network traffic that signal anomalies or attacks. MLPs also excel at capturing interactions between features, which is important given the high dimensionality and complexity of the dataset in this project.

I can scale MLPs by increasing layers or neurons, which helps when dealing with larger datasets or more intricate patterns. To avoid overfitting, I apply L2 regularization and dropout layers. This approach ensures that the model generalizes well, even with my imbalanced data.

References

[1] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.