

# Anomaly Detection and Attack Classification in Industrial IoT Networks Using the WUSTL-IIoT-2021 Dataset

<b>Course:</b>	Tools and Techniques
<b>Project Type:</b>	Term Project
<b>Student Name:</b>	Shabbir Minhas
<b>Roll Number:</b>	MSCS22059
<b>Instructor:</b>	Kalim Majeed
<b>Department:</b>	CS
<b>University:</b>	Information Technology University
<b>Submission Date:</b>	08-12-2025

## Task 1: Data Loading and Dataset Description

### 1.1 Objective of the Task

The objective of this task is to load the given dataset into the analysis environment and develop a basic understanding of its structure, size, and purpose before proceeding with detailed exploratory data analysis and modeling.

### 1.2 Dataset Description

The dataset used in this project is the **WUSTL-IIoT-2021 dataset**, a publicly available Industrial Internet of Things (IIoT) intrusion detection dataset. It is specifically designed for research on **anomaly detection, cyber-attack classification, and zero-day attack detection** in industrial environments.

This dataset represents **realistic network traffic flows** generated from both normal IIoT operations and multiple types of cyber-attacks. It includes flow-level statistics such as:

- Source and destination IP addresses

- Source and destination ports
- Packet and byte counts
- Data transfer rates
- Flow duration
- Jitter and loss measurements
- Application-level byte information

Each record in the dataset represents a **single network flow**, making it highly suitable for machine learning-based intrusion detection systems.

```
[6]: # Show first 5 rows
full_data.head()
```

	StartTime	LastTime	SrcAddr	DstAddr	Mean	Sport	Dport	SrcPkts	DstPkts	TotPkts	...
0	2019-08-19 12:23:28	2019-08-19 12:23:28	192.168.0.20	192.168.0.2	0	59034	502	10	8	18	...
1	2019-08-19 15:13:24	2019-08-19 15:13:24	192.168.0.20	192.168.0.2	0	55841	502	10	8	18	...
2	2019-08-19 13:41:31	2019-08-19 13:41:31	192.168.0.20	192.168.0.2	0	63774	502	10	8	18	...
3	2019-08-19 12:43:19	2019-08-19 12:43:20	209.240.235.92	192.168.0.2	0	61771	80	4	0	4	...
4	2019-08-19 14:49:44	2019-08-19 14:49:48	192.168.0.20	192.168.0.1	3	0	0	14	0	14	...

5 rows x 49 columns

## 1.3 Data Loading Process

The dataset was loaded using the Python programming language and the Pandas library. The Python file contains the code for data loading.

## 1.4 Dataset Size and Structure

After loading the dataset, it was observed that:

- The dataset contains **1,194,464 records (rows)**.
- It includes **49 features (columns)**.
- The features consist of both **numerical and categorical variables**.
- The dataset includes a **target class label (Traffic)** representing the traffic type and attack category.

The dataset contains five traffic classes:

- Normal
- Denial of Service (DoS)
- Reconnaissance (Reconn)
- Backdoor
- Command Injection (Commlnj)

## 1.5 Relevance to the Course “Tools and Techniques”

This dataset is highly relevant to the **Tools and Techniques** course project as it enables hands-on implementation of:

- Data preprocessing techniques
- Exploratory Data Analysis (EDA)
- Feature engineering methods
- Machine learning classification models
- Anomaly and zero-day attack detection strategies

The project integrates multiple tools such as **Python, Pandas, NumPy, Matplotlib, and Scikit-learn**, demonstrating the practical application of data science and cybersecurity tools.

# Task 2: Exploratory Data Analysis (EDA)

EDA was performed using Python (Pandas, NumPy, Matplotlib) and Results can be seen in the GitHub → full notebook/scripts

## 2.1 Dataset Structure

- Shape of data
  - Shape of dataset: (1194464, 49)
- Shows columns
  - ['StartTime', 'LastTime', 'SrcAddr', 'DstAddr', 'Mean', 'Sport', 'Dport',  
'SrcPkts', 'DstPkts', 'TotPkts', 'DstBytes', 'SrcBytes', 'TotBytes',  
'SrcLoad', 'DstLoad', 'Load', 'SrcRate', 'DstRate', 'Rate', 'SrcLoss',  
'DstLoss', 'Loss', 'pLoss', 'SrcJitter', 'DstJitter', 'SIntPkt',  
'DIntPkt', 'Proto', 'Dur', 'TcpRtt', 'IdleTime', 'Sum', 'Min', 'Max',  
'sDSb', 'sTtl', 'dTtl', 'sIpld', 'dIpld', 'SAppBytes', 'DAppBytes',  
'TotAppByte', 'SynAck', 'RunTime', 'sTos', 'SrcJitAct', 'DstJitAct',  
'Traffic', 'Target']
- Show datatypes
  - `object` (5): StartTime, LastTime, SrcAddr, DstAddr, Traffic
  - `int64` (24): various packet counts, bytes, IDs, ports, TTLs, labels
  - `float64` (20): loads, rates, durations, jitter, loss, etc.
- Summary explanation
  - The dataset contains a total of **1,194,464 records and 49 features**. Each record represents a network flow observed in an Industrial IoT (IIoT) environment.

The features include:

- **Temporal attributes:** StartTime, LastTime
- **Network identifiers:** SrcAddr, DstAddr, Sport, Dport, Proto
- **Traffic statistics:** packet counts, byte counts, load, rate, jitter, and loss
- **Application-level features:** SAppBytes, DAppBytes, TotAppByte
- **Target variables:** Traffic (attack class) and Target (numerical encoding)

## 2.2 Missing Values Analysis

- Count of missing values
  - There are no missing values as we can see while performing the EDA
- Percentage missing
  - No missing values in any column.

## 2.3 Statistical Summary of Numerical Features

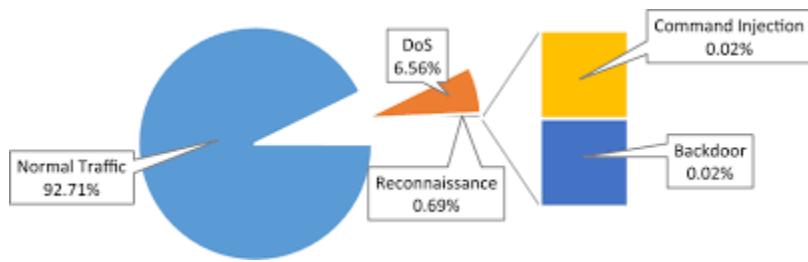
- Means
- Min/max
- Standard deviation

	Mean	Sport	Dport	SrcPkts	DstPkts	TotPkts	Dst
count	1.194464e+06						
mean	1.286870e-01	5.445253e+04	7.907604e+02	1.665579e+02	1.688389e+01	1.756631e+02	7.60157e-01
std	6.859156e-01	1.200834e+04	3.299492e+03	5.266192e+04	1.137763e+03	5.266221e+04	7.50869e+00
min	0.000000e+00						
25%	0.000000e+00	5.221800e+04	5.020000e+02	1.000000e+01	8.000000e+00	1.800000e+01	5.08000e+00
50%	0.000000e+00	5.663500e+04	5.020000e+02	1.000000e+01	8.000000e+00	1.800000e+01	5.08000e+00
75%	0.000000e+00	6.104300e+04	5.020000e+02	1.000000e+01	8.000000e+00	1.800000e+01	5.08000e+00
max	5.000000e+00	2.765721e+06	6.552200e+04	2.773967e+07	3.092160e+05	2.773967e+07	8.25155e+00

- 8 rows × 44 columns

## 2.4 Class Distribution

- Attack vs Normal
  - Target
    - 0 92.715059 ( Normal)
    - 1 7.284941 ( Attack)
  - “Normal is ~92% and Attack is ~7%”



- Severe imbalance
- Implications for ML/zero-day detection
  - The large number of records and the diversity of network-level statistics make this dataset highly suitable for machine learning-based anomaly detection and classification.

## 2.5 Univariate Analysis

- Histograms
- Distribution of key features
- Skewness explanation

Histograms of key numerical features such as `TotBytes`, `TotPkts`, `Dur`, and `Rate` were generated using Python to examine their individual distributions (refer to Task 4). The plots reveal strong right-skewness, with most traffic concentrated at low values and a small number of extreme observations corresponding to potential anomalous behavior.

## 2.6 Multivariate Analysis

- Correlation heatmap
- Feature relationships
- Attack-class behavior differences

# Task 3: Data Wrangling and Cleansing

## 3.1 Objective of Data Wrangling

The purpose of this step is to transform the raw WUSTL-IIoT-2021 dataset into a **clean, consistent, and analysis-ready form**. Data wrangling ensures that missing values, inconsistent types, and noisy features are handled appropriately before building machine learning models for anomaly detection and attack classification.

## 3.2 Removal of Identifier and Timestamp Features

To prepare the dataset for machine learning, several identifier and timestamp-related columns were removed because they do not directly contribute to predictive learning and may introduce noise or bias. The following columns were dropped:

- `StartTime`, `LastTime` — time-based features
- `SrcAddr`, `DstAddr` — source and destination IP addresses
- `sIpId`, `dIpId` — IP identification fields

These fields primarily serve as **identifiers rather than behavioral indicators**, and including them in machine learning models could lead to overfitting without improving generalization.

**After dropping these columns, the dataset shape became:**

- Final shape after column removal: (1,194,464 rows × 43 columns)

This confirms that **6 non-essential columns** were successfully removed.

## 3.3 Missing Values Verification

After feature removal, a comprehensive missing values analysis was performed across all remaining 43 features. The results showed:

- No missing values were present in any of the remaining columns.
- Total NaN values after cleansing: 0

```
[14]: # Total missing values per column  
missing_counts = full_data.isnull().sum().sort_values(ascending=False)  
print(missing_counts)
```

```
StartTime      0  
SIntPkt       0  
Proto         0  
Dur           0  
TcpRtt        0  
IdleTime      0  
Sum           0  
Min           0  
Max           0  
sDSb          0
```

This confirms that the dataset is **fully complete and does not require any imputation techniques**, making it highly suitable for supervised and unsupervised machine learning tasks.

## 3.4 Duplicate Records Check

To ensure data integrity, a duplicate check was performed on the cleaned dataset. The results showed:

- **Total duplicate rows: 0**

This confirms that:

- Every network flow record is unique.
- There is no redundancy that could bias class distributions or model performance.

---

► `print("Duplicate rows:", full_data.duplicated().sum())`

```
Duplicate rows: 0
```

## 3.5 Final Cleaned Feature Set

After cleansing, the dataset contains the following categories of features:

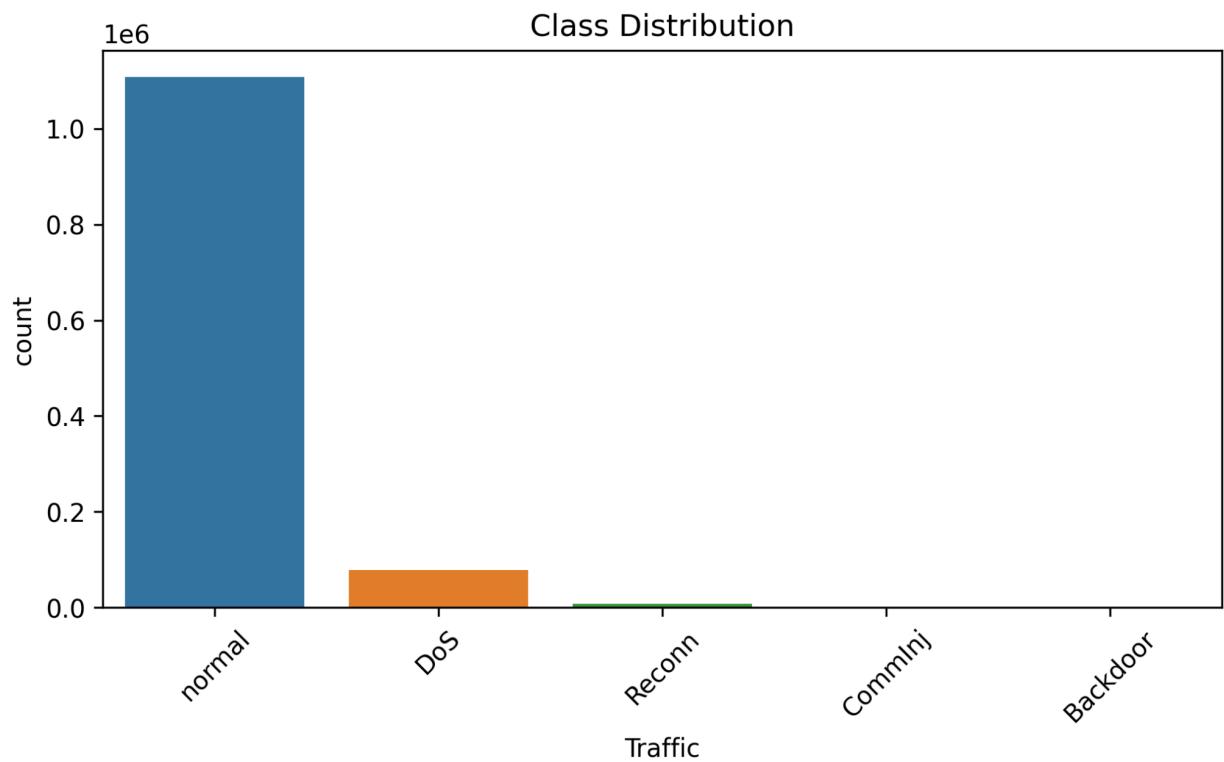
- **Traffic statistics:** SrcPkts, DstPkts, TotPkts, SrcBytes, DstBytes, TotBytes
- **Load & rate features:** SrcLoad, DstLoad, Load, SrcRate, DstRate, Rate
- **Loss & jitter features:** SrcLoss, DstLoss, Loss, pLoss, SrcJitter, DstJitter
- **Flow behavior:** Dur, IdleTime, TcpRtt, Sum, Min, Max
- **Application-level statistics:** SAppBytes, DAppBytes, TotAppByte
- **Traffic class labels:** Traffic and Target

These cleaned features now strictly represent **network behavior**, making them optimal for:

- Anomaly detection
- Zero-day attack detection
- Attack classification tasks

## Task 4: Build multiple charts and explain observations

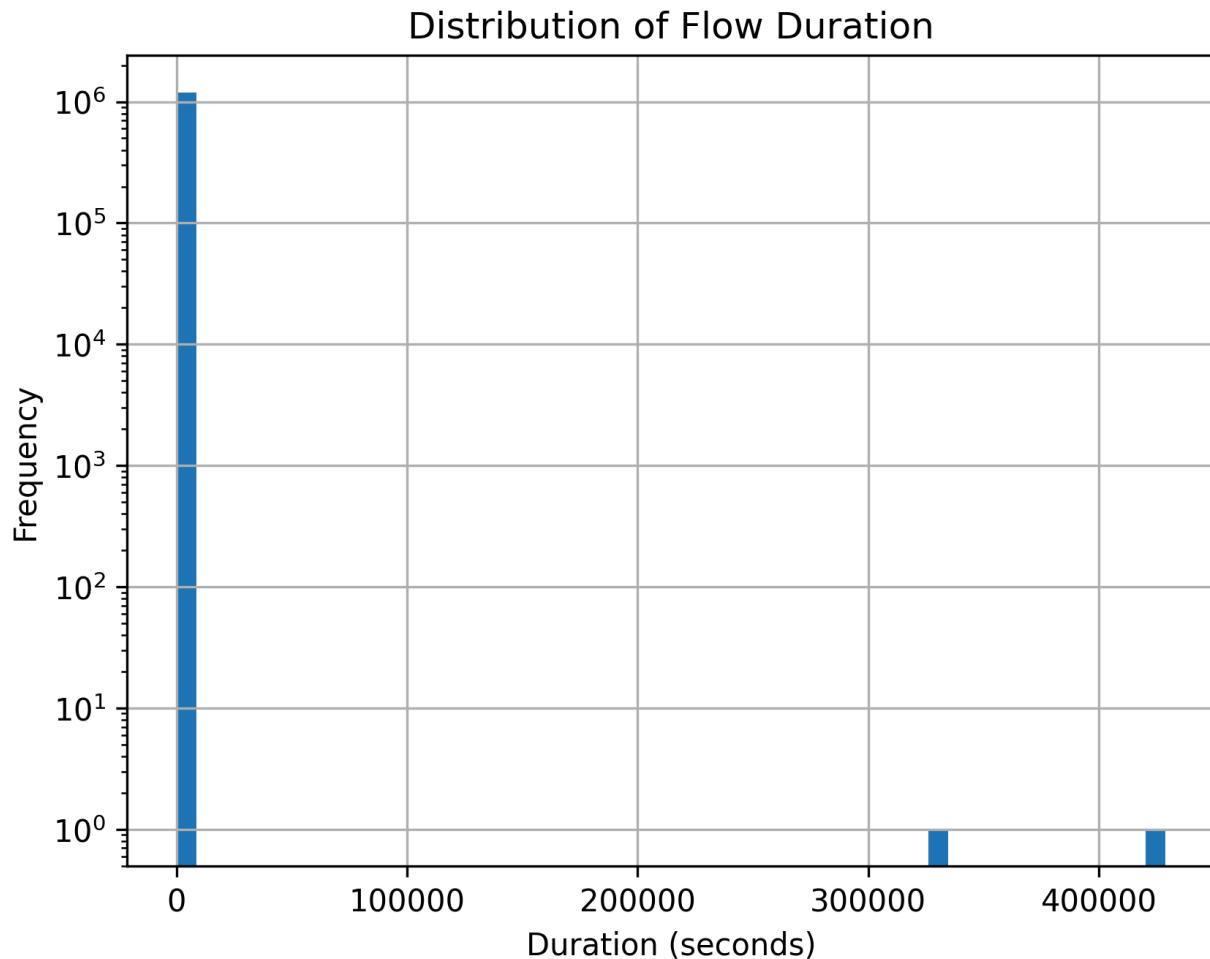
Chart 1: Class distribution (bar plot)



### Class Distribution Plot

The bar chart shows a highly imbalanced distribution of traffic classes. Normal traffic dominates the dataset, while attack classes such as **Backdoor** and **CommInj** appear very rarely. This confirms that the problem is not only a multi-class classification task but also an **anomaly detection** problem, where rare events are of primary interest. Special care will be needed in evaluation and model design to avoid bias toward the majority class.

## Chart 2: Distribution of flow duration (**Dur**)



### Flow Duration Distribution

The histogram of flow duration (**Dur**) is highly skewed, with many short-lived flows and a smaller number of long-duration flows. When plotted on a logarithmic scale, the heavy-tailed nature of the distribution becomes apparent. This behavior is typical of real network traffic and is relevant for anomaly detection because certain attack types may manifest as unusually short or unusually long flows.

## Chart 3: Total bytes by class (boxplot)

### Total Bytes per Flow by Class

The boxplot of **TotBytes** across different traffic classes reveals differences in the volume of data transferred during benign and malicious flows. Certain attack classes show heavier tails or

higher median values compared to normal traffic, indicating that they often involve larger data transfers or more intense communication patterns. This suggests that `TotBytes` is a potentially informative feature for distinguishing between normal and attack traffic.

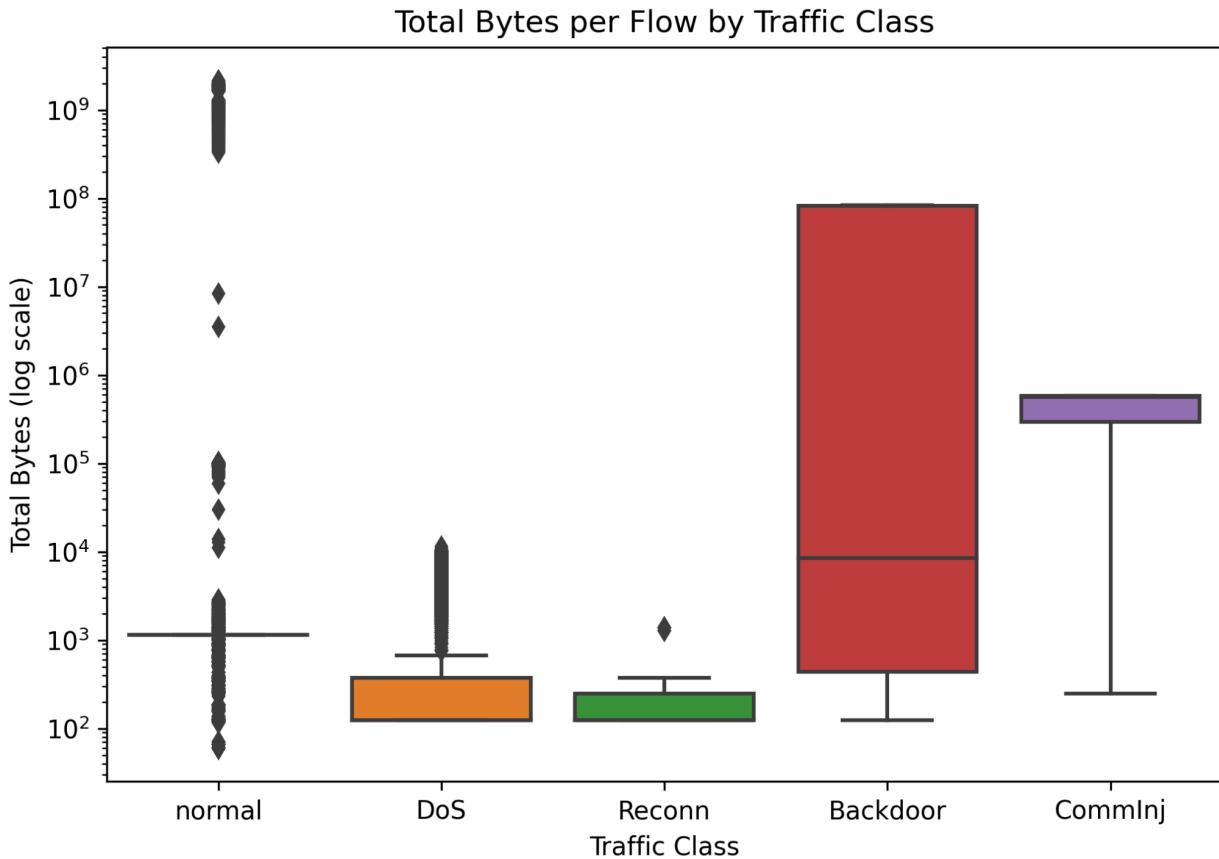
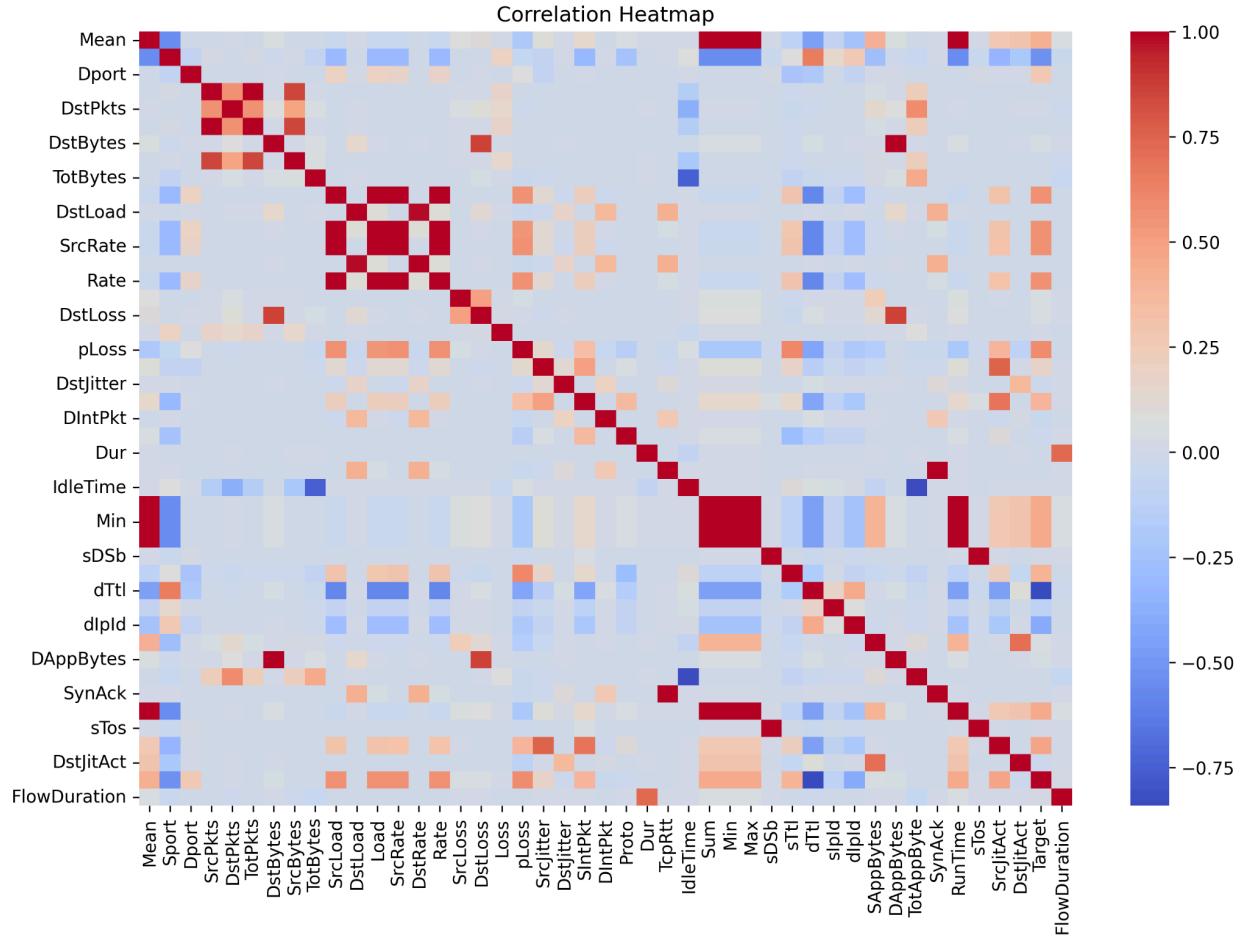


Chart 4: Correlation heatmap of selected numerical features

#### Correlation Heatmap of Traffic Features

The correlation heatmap shows strong positive correlations between packet-based and byte-based features (e.g., `TotPkts` and `TotBytes`), which is expected since flows with more packets typically carry more bytes. Some features, such as `Dur`, `Rate`, and `Load`, exhibit more moderate correlations, indicating that they capture complementary aspects of traffic behavior. This analysis helps identify redundant features and informs later decisions on feature selection or dimensionality reduction.



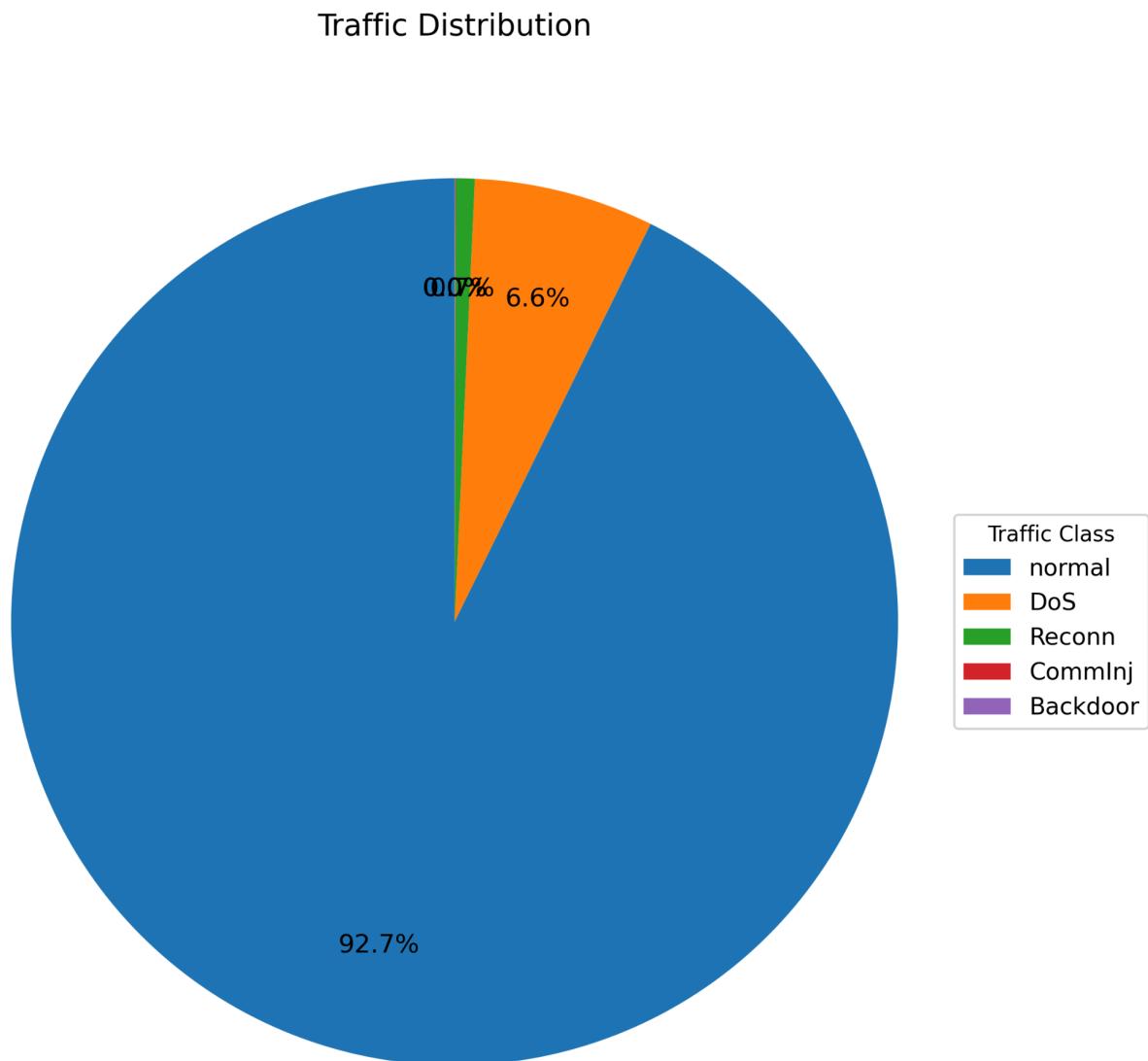
**Chart 5: Pie Chart - Distribution of Network Traffic**

### Class Distribution of Network Traffic

The pie chart illustrates the proportional distribution of traffic classes in the WUSTL-IIoT-2021 dataset. It is evident that the dataset is **severely imbalanced**, with **normal traffic accounting for approximately 92.7% of all records**. The **Denial-of-Service (DoS)** class represents around **6.6%** of the data, while the remaining attack classes—**Reconnaissance (Reconn)**, **Command Injection (ComInj)**, and **Backdoor**—together contribute **less than 1%** of the total samples.

This extreme imbalance confirms that the dataset is highly representative of real-world network environments, where malicious activity is significantly rarer than normal traffic. From a machine learning perspective, this imbalance poses a major challenge, as traditional classifiers may become biased toward predicting the majority class. Therefore, this observation strongly justifies the use of **anomaly detection techniques and specialized evaluation metrics** such as precision, recall, F1-score, and class-wise performance rather than overall accuracy alone.

Furthermore, the very small proportion of certain attack types (e.g., Backdoor and Commlnj) makes this dataset particularly suitable for **zero-day attack detection experiments**, where models must detect previously unseen or extremely rare attack behaviors.



**Note:** While multiple visualizations were generated as part of the exploratory data analysis using Python, only the most relevant plots have been included in this report. The complete set of charts and their corresponding code can be found in the accompanying Python notebook/script submitted with this project.

## Task 5: Feature Engineering & Encoding

### 5.1 Objective of Feature Engineering

The objective of this task is to transform the cleaned dataset into a machine-learning-ready format by appropriately separating categorical and continuous features, preparing them for encoding and scaling, and ensuring compatibility with classification and anomaly detection algorithms

### 5.2 Feature Categorization

After completing data cleansing, the features were categorized into **categorical** and **continuous** groups based on their data type and role in network traffic behavior.

#### Categorical Feature

The following feature was identified as categorical and suitable for encoding:

- **Proto** — Network protocol type (e.g., TCP, UDP, ICMP)

This feature represents protocol-level behavior and cannot be directly used in numerical machine learning algorithms without encoding.

#### Continuous Numerical Features

All remaining features were treated as continuous numerical attributes, including:

- Traffic statistics:  
SrcPkts, DstPkts, TotPkts, SrcBytes, DstBytes, TotBytes
- Load and rate features:  
SrcLoad, DstLoad, Load, SrcRate, DstRate, Rate

- Loss and jitter metrics:  
SrcLoss, DstLoss, Loss, pLoss, SrcJitter, DstJitter
- Flow behavior attributes:  
Dur, IdleTime, TcpRtt, Sum, Min, Max
- Application-level statistics:  
SAppBytes, DAppBytes, TotAppByte
- Additional behavioral indicators:  
SynAck, RunTime, sTos, SrcJitAct, DstJitAct
- Statistical descriptors:  
Mean, sDSb, sTtl, dTtl, SIntPkt, DIntPkt

These continuous features capture fine-grained network behavior and are critical for distinguishing normal traffic from various types of cyber-attacks.

### 5.3 Feature Scaling Configuration Using TabPreprocessor

To ensure that continuous features contribute uniformly during model training, a feature scaling configuration was applied using the `TabPreprocessor`. All continuous numerical variables were explicitly selected for scaling using the `cols_to_scale` parameter. This step ensures that features with large numerical ranges, such as byte counts and traffic rates, do not dominate smaller-magnitude features during learning. The actual scaling transformation is applied during the model training phase through the preprocessing pipeline.

### 5.4 Target Label Encoding

The categorical traffic class labels in the `Traffic` feature were converted into numerical form using a label encoding technique to make them compatible with machine learning algorithms.

```
# Encoder for filtered_data
label_encoder = LabelEncoder()
filtered_data['Traffic'] = label_encoder.fit_transform(filtered_data['Traffic'])
```

## 5.5 Final Feature Set for Model Input

Following feature categorization, the final model input structure consists of:

- **1 categorical feature:**  
Proto
- **Over 40 continuous traffic and behavioral features**
- **2 target variables:**
  - Traffic (multi-class label)
  - Target (numerical encoding)

This structured feature configuration enables the application of:

- Supervised classification models
- Anomaly detection models
- Zero-day attack detection strategies

## Task 6: Model Building

### Model Description

The proposed model is a self-supervised representation learning framework based on an encoder–decoder neural network architecture. The encoder learns compact latent representations of Industrial IoT traffic from unlabeled data, while the decoder reconstructs the original features to guide representation learning. The learned embeddings are then used for unsupervised clustering using K-Means to analyze traffic patterns and assess cluster purity with respect to true attack labels.

#### Unlabeled Training Feature Selection

The training dataset was assigned as the unlabeled feature set by excluding all class labels.

## **Tabular Preprocessing of Unlabeled Data**

The unlabeled training features were transformed using the tabular preprocessing pipeline to generate encoded and scaled inputs.

## **Decoder Network Initialization**

A multilayer perceptron–based decoder network was initialized to reconstruct the original feature space from learned latent representations.

## **Encoder Network Initialization**

A multilayer perceptron–based encoder network was initialized using the preprocessed tabular feature configuration to learn compact data representations.

## **Encoder–Decoder Training Controller Setup**

A training controller was initialized to manage joint encoder–decoder optimization using Adam, learning rate scheduling, early stopping, and GPU acceleration.

## **Execution of Self-Supervised Pretraining**

The encoder–decoder model was pretrained on the unlabeled, preprocessed training data for 100 epochs with a large batch size while recording the total training time

## **Saving the Pretrained Representation Model**

The trained tabular preprocessor and encoder weights were serialized and saved to disk for reuse in downstream tasks.

## **Supervised Training Hyperparameter Configuration**

Key training hyperparameters including maximum epochs, learning rate, weight decay, early stopping patience, and model checkpoint filename were defined.

## **Preprocessing and Tensor Conversion of Labeled Features**

The selected labeled features were transformed using the fitted tabular preprocessor and converted into GPU-ready tensors.

## **Latent Feature Extraction Using the Encoder**

The preprocessed input features were **passed** through the encoder network to generate compact latent embeddings.

## **Conversion of Latent Embeddings to NumPy Format**

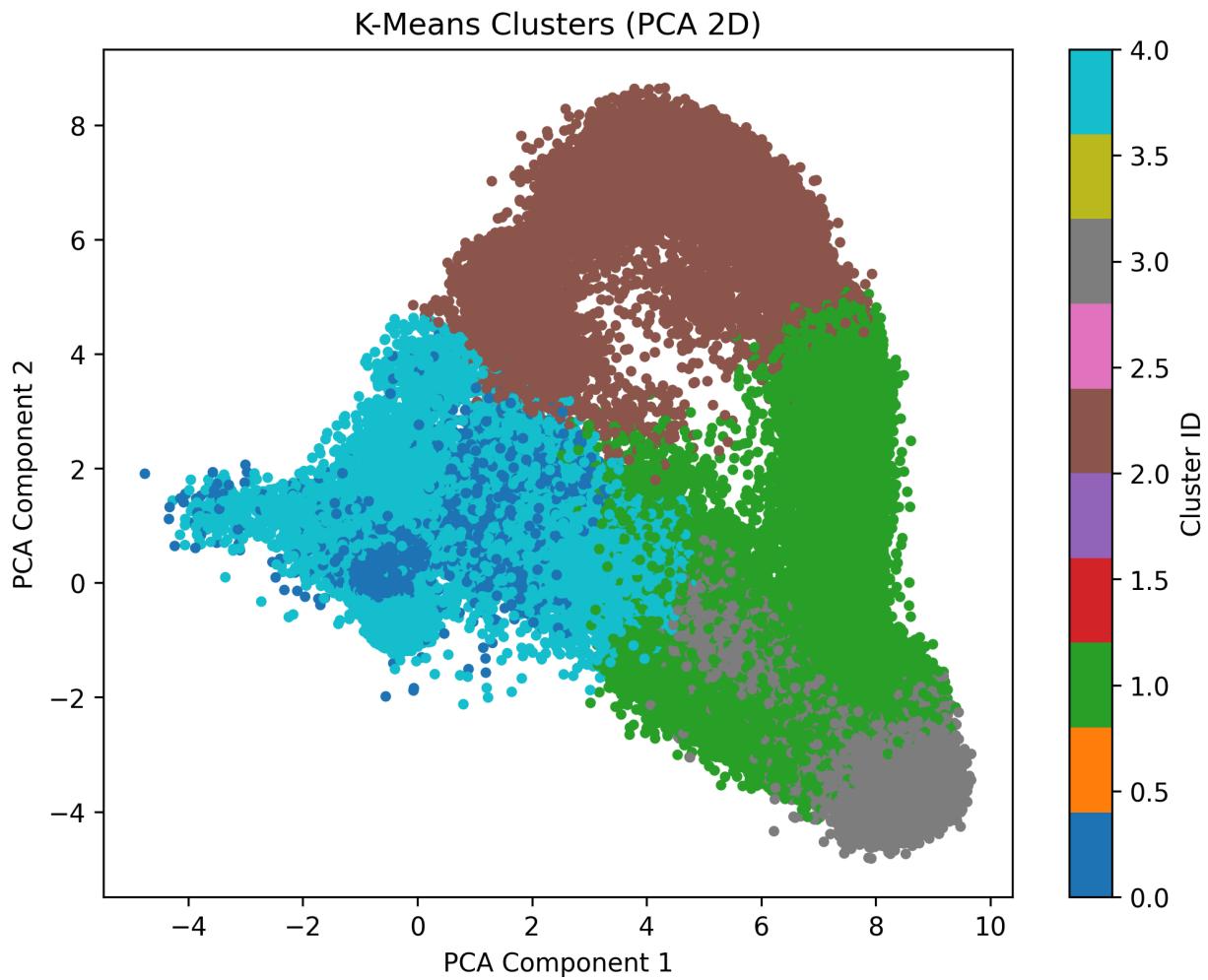
The encoder-generated latent feature embeddings were transferred from GPU to CPU and converted into NumPy arrays.

## **K-Means Clustering on Learned Embeddings**

K-Means clustering was applied to the encoder-generated latent embeddings to group the data into five clusters.

## **Clustering Results and Purity Analysis**

The learned embeddings were clustered using K-Means



## Cluster purity

**cluster purity** was computed by comparing the predicted cluster assignments with the true class labels. The resulting **purity score of 0.9272** indicates that the majority of data points within each cluster belong to the same traffic class.

**Cluster Purity:** 0.9272 (High-quality clustering performance)

```
# Check Clusters Purity -----
def cluster_purity(y_true_labels, cids):
    df = pd.DataFrame({"y": y_true_labels, "c": clusters})
    ct = df.value_counts(["c", "y"]).rename("count").reset_index()
    max_per_cluster = ct.groupby("c")["count"].max()
    return max_per_cluster.sum() / len(df)

purity = cluster_purity(y_true_labels, clusters)
print(f"[Purity] {purity:.4f} (higher is better; in (0,1])")

[Purity] 0.9272 (higher is better; in (0,1])
```

## Supervised Model Training Using Pseudo-Labels

After obtaining high-quality pseudo-labels from K-Means clustering (with a cluster purity of **0.9272**), the problem was reformulated as a **supervised multi-class classification task**. These pseudo-labels were then used to train additional machine learning models to evaluate how well different algorithms can learn from the discovered traffic structure.

This stage enables a **comparative analysis of supervised classifiers** trained on representation-based pseudo-labels.

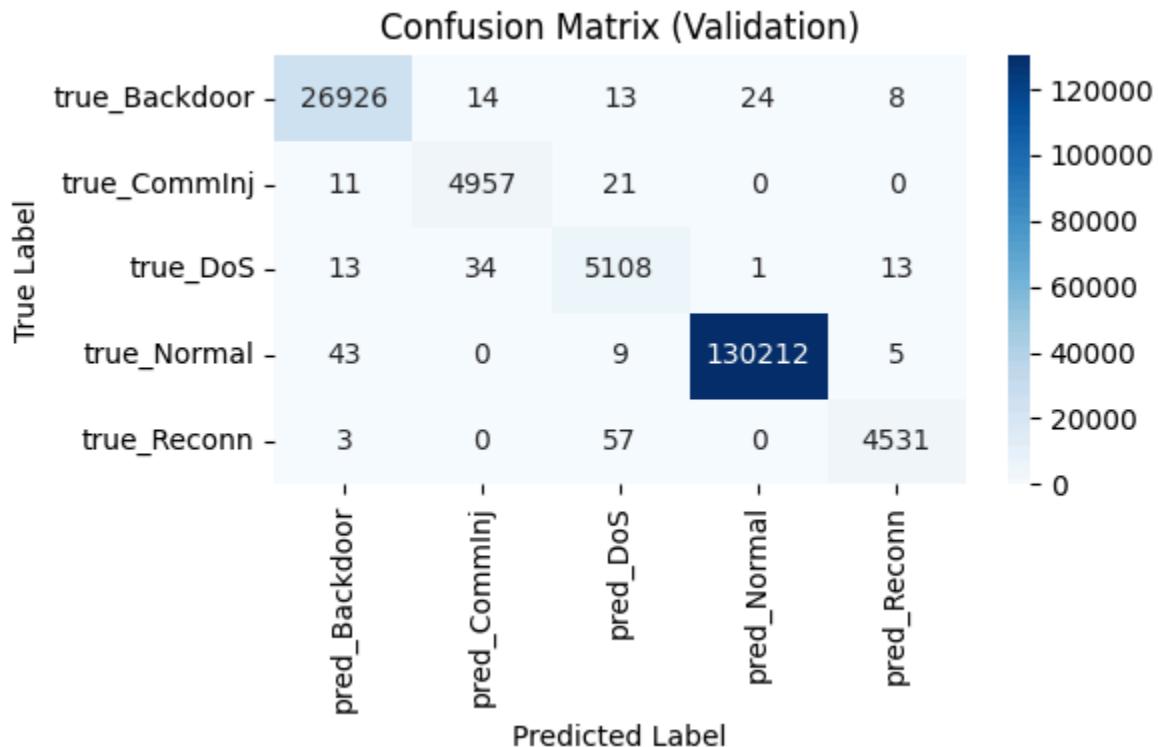
### LightGBM Classifier Training on Pseudo-Labels

LightGBM was selected as the first supervised model due to its high computational efficiency, robustness to class imbalance, and strong performance on large-scale tabular datasets. The model was trained using the latent embeddings as input features and the K-Means pseudo-labels as class targets.

### LightGBM Results

The LightGBM classifier trained on the pseudo-labeled embeddings achieved a **balanced accuracy of 0.9932**, demonstrating excellent classification performance even under severe class imbalance.

## Confusion Matrix



## Classification Report

Balanced Accuracy: 0.9932

Classification report (val):				
	precision	recall	f1-score	support
Backdoor	0.9974	0.9978	0.9976	26985
CommInj	0.9904	0.9936	0.9920	4989
DoS	0.9808	0.9882	0.9845	5169
Normal	0.9998	0.9996	0.9997	130269
Reconn	0.9943	0.9869	0.9906	4591
accuracy			0.9984	172003
macro avg	0.9925	0.9932	0.9929	172003
weighted avg	0.9984	0.9984	0.9984	172003

+ Code

+ Markdown

## XGBoost on pseudo labels

The XGBoost classifier trained on pseudo-labeled embeddings achieved a **balanced accuracy of 0.9942**, demonstrating superior multi-class classification performance under imbalanced conditions.

### Confusion Matrix

---

```
Confusion matrix (val):
             pred_Backdoor  pred_CommInj  pred_DoS  pred_Normal  pred_Reconn
true_Backdoor        26926          14         13          24            8
true_CommInj           11        4957          21            0            0
true_DoS              13          34        5108            1          13
true_Normal            43            0            9        130212            5
true_Reconn            3            0            57            0        4531
<Figure size 640x480 with 0 Axes>
```

### Classification Report

---

```
Balanced Accuracy: 0.9946

Classification report (val):
             precision    recall  f1-score   support
Backdoor      0.9723   0.9912   0.9816     2264
CommInj       0.9876   0.9950   0.9913     4634
DoS          0.9972   0.9972   0.9972     5352
Normal        0.9999   0.9995   0.9997   155279
Reconn        0.9919   0.9899   0.9909     4474

accuracy      0.9989   0.9989   0.9989   172003
macro avg     0.9898   0.9946   0.9921   172003
weighted avg  0.9989   0.9989   0.9989   172003
```

# Model Comparison

Model	Balanced Accuracy
LightGBM	0.9932
XGBoost	<b>0.9946</b>