Feature engineering for efficiently detecting Modbus/TCP anomaly

Jinseok Park, Geonwoo Lee
July 22, 2021

Abstract

As the number of smart factory increases, the factory's network changes from a closed network to an open concept, increasing the need to respond to attacks that can come through external networks. Accordingly, in this study, we investigate which learning base detection model and which feature set can effectively detect anomaly quickly and accurately on Modbus/TCP networks. Among the SVM, DT, Random Forest, Kmeans, DNN, and Autoencoder models, DT has the fastest throughput and stable accuracy compared to other models, while DNN models and Autoencoder models have high accuracy and acceptable process speed.

---

1 Introduction

Recently, in the advent of the Fourth Industrial Revolution, smart factories with strong connection-oriented characteristics have emerged, escaping from the closed network concept. Unlike a closed network base factory environment, the Industrial Control System(ICS) of a smart factory tends to be connected to the external internet, and it means that system is exposed to several security threats that have not been considered before. Therefore, there is a need to come up with countermeasures against attacks coming from external networks.
In this paper, we studied which detection model and which feature extraction method would be most efficient to detect network anomalies, such as DDoS, during network attacks. We limit the target to Modbus/TCP for network protocol and only considered Machine Learning-based detection models because the Rule-based model needs manual rule generation and management taking more cost than learning-based model in the long term.
About learning-based anomaly detection models, we focused on these two factors.

. feature extraction : which feature best represent abnormal behavior?

. detection model: which model is most accurate and efficient(fast)?

We conducted intensive research on the two topics and conducted them between May and early June this year.

2 Approaches

2.1 Dataset

To train anomaly detection models, dataset and their quality are very important. Unfortunately, we could not get hardware to simulate the CPS environment, so we should continue our research by using the existing dataset. The dataset used in the study, CYBER-SECURITY MODBUS ICS DATASET [1], is a dataset that exists in IEEE dataport. It simulated the following small industrial networks and then generated several anomalies, such as DDoS/MITM.
We created a dataset by extracting features from each packet of the original dataset and labeling each of them by attack type. The composition of the dataset that we created is as follows.

Datasets class type

Type	number(packet)
Normal	1208554(during attack) + 521686
MITM	47083
Modbus Query Flooding	893233
Ping Flooding	754232
TCP SYN Flooding	698758
Total	4123546

Portion of Train and Test set

Usage	number(packet)
Train	2474127 (60%)
Test	1649419 (40%)

We limited the number of feature data from attack packets to balance normal and attack packets feature data in the datasets, and used 60% of them for training, and used the rest 40% for testing. We did feature engineering by using this dataset, by limiting some features in training and testing anomaly detection models.

2.2 Overview

To simultaneously evaluate the extracted feature set and the performance of the detection model, we construct the framework as follows.

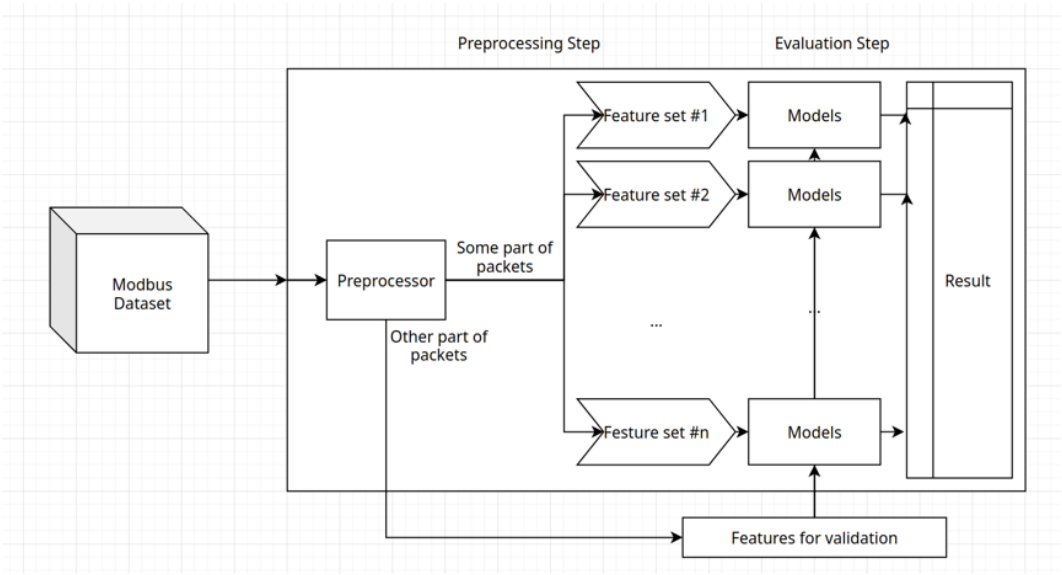

Figure 1: framework overview

The framework is largely divided into pre-processing and evaluation step. In the preprocessing step, feature information is extracted from packets in the pcap file, and features are fused to form multiple feature sets. In the subsequent evaluation phase, models are trained and tested for each feature set to evaluate their performance.

2.3 Preprocessing step

We determine that basic information which is extractable from individual packets can be utilized to detect anomalies in Modbus/TCP environments. Therefore, protocol type, data length, TCP flag, ARP opcode, ICMP type, ICMP code information was extracted from each packet, and the above information was set to feature set candidates. Information extracted from these individual packets will be collectively referred to as "Packet Information" from now on and its detailed explanation is shown below.

Packet Information

Features	feature location in full vector
Protocol type	[0]
Data length	[1]
TCP flag	[2∼7]
ARP op	[8∼11]
ICMP type	[12∼19]
ICMP code	[20∼27]

In addition, since abnormalities occur across multiple packets rather than one packet, we cut the packet's sequence into a two-second window and extract features below. We will collectively call this "Flow Information" from now on.

Flow information(N Sec)

Features	feature location in full vector
number of packets from same src ip	[28]
number of packets from same dst ip	[29]
number of packets from same src ip, src port	[30]
number of packets from same dst ip, dst port	[31]
number of IP address to same dst ip	[32]
number of source port from same src ip to same dst ip	[33]
number of destination port from same src ip to same dst ip	[34]
Total bytes size from same src ip	[35]
Total bytes size from same dst ip	[36]
number of ICMP packets from same src ip	[37]
number of TCP SYN packets from same src ip	[38]
number of ICMP packets to same dst ip	[39]
number of TCP SYN packets from same dst ip	[40]

2.4 Evauation step

This step is the part where the model is trained and tested using the feature set extracted earlier. The result of this step is performance indicators for each model and used feature set such as learning time, processing time, and accuracy. The learning-based model used in this study is as follows.

. SVM : SVM is a machine learning algorithm that calculating maximum-margin hyperplane that can classify the input data. in this study, we used rbf(Radial Basis Function) as kernel, give C=8, gamma=0.1, and limited maximum iteration to 500 times.

. DT : DT make decision tree that can classify new input by evaluating the input value with tree of conditions. in this study, we give default parameter to DT model(no limitation of max depth and max leaf of tree).

. Random Forest : Random Forest make Forest containing multiple DT from input data and make final decision by combining each decision from DT. We set the number of DT to 50 in Random Forest model.

. K Means : K Means make n clusters that minimize the total within-cluster variance. We set number of clusters to 5.

. DNN : DNN is supervised neural network model, and its model shape is shown as below.

. Autoencoder model : We first trained the Autoencoder in an unsupervised manner. Then we make new neural net using Autoencoder and did supervised training. The shape of final Autoencoder model is shown as below.

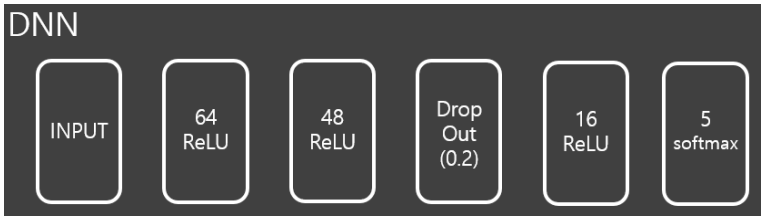

Figure 2: Shape of DNN Model

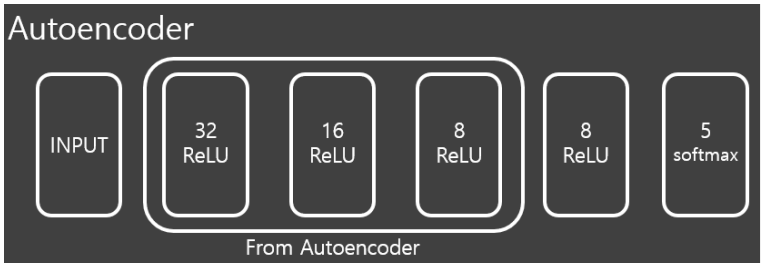

Figure 3: Shape of Autoencoder Model

3 Evaluation

Section 3 discusses the evaluation result of the framework.

3.1 Environment

We set up the evaluation environment as follows.

Evaluation Environment

OS	Windows 10
CPU	AMD Ryzen 3600XT (6C12T)
GPU	NDIVIA GeForce GTX 1660 SUPER
RAM	24GB
Runtime	Python 3.8 & Jupyter notebook

3.2 Feature set

The number of all possible feature sets is 2^{41}, and testing all possible feature sets is almost impossible. So we arbitrary selected five feature set that is considered as meaningful in this project. Below are the feature sets we selected.

1. Full feature: Using all extracted features as feature set
2. Extended Flow information: All flow information with basic packet information(protocol type, data length)
3. Flow information only: Feature set contains only flow information
4. Limited flow information: Some flow information([35, 37, 38]) were dropped from above feature set
5. Packet information Only: Feature set with all packet information and without flow information

3.3 Perfomance

We evalauted the performance of each model with selected feature sets.

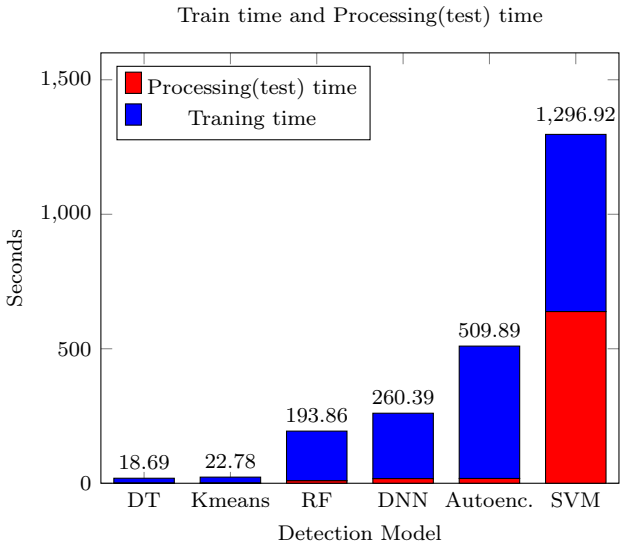

Figure 4: Feature Set #1

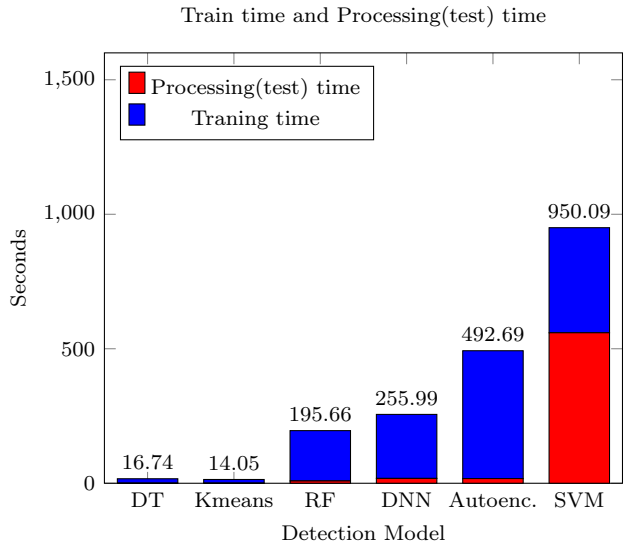

Figure 5: Feature Set #2

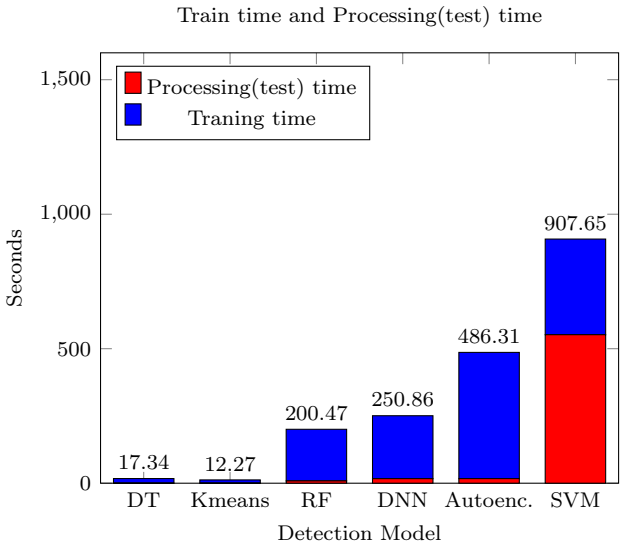

Figure 6: Feature Set #3

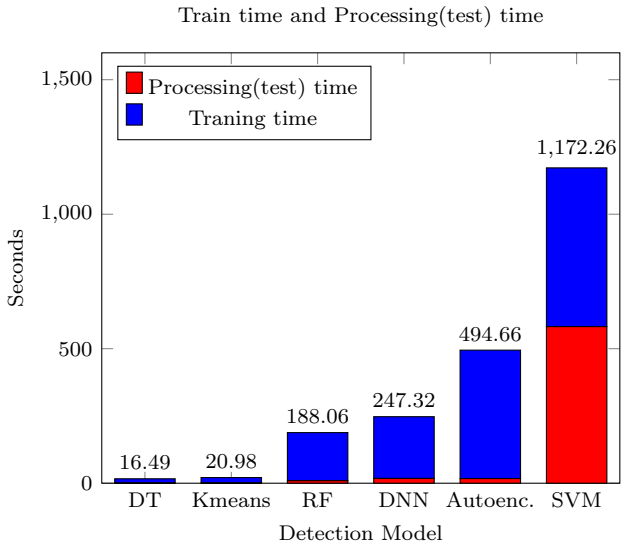

Figure 7: Feature Set #4

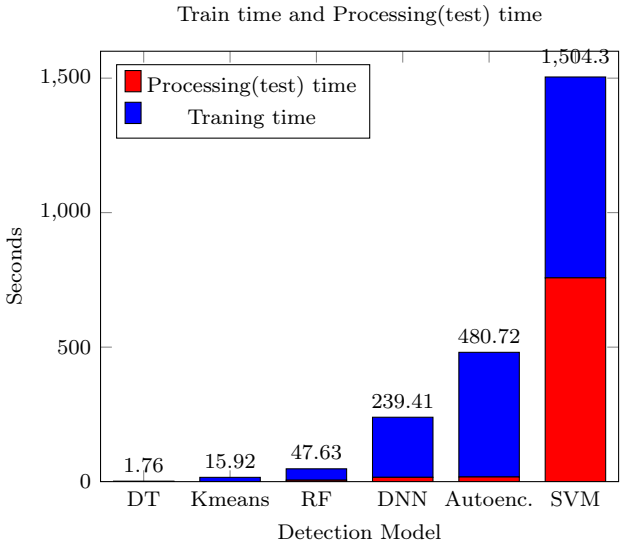

Figure 8: Feature Set #5

Each detection model showed relatively similar performance across the feature sets. To sum up, it is as follows;

- In most cases, DT, Kmeans showed the highest performance among detection models in terms of training and test(process) time.
- DNN, Autoencoder, Random Forest models showed the second-fastest performance. They spent more training times, but they showed fast detection speed in comparison with DT and Kmeans model
- SVM showed low performance in most cases, including both training time and test time.
- Reducing the number of features in feature set reduced training time and test time of some model, such as SVM.

3.4 Accuracy

We evaluated the performance of each model with selected feature sets.

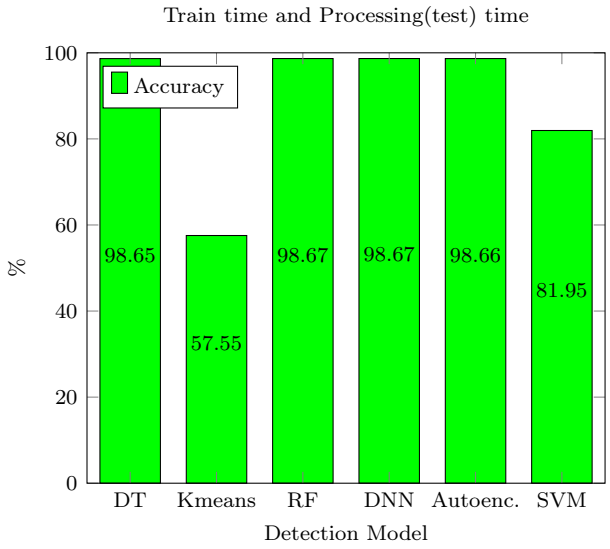

Figure 9: Feature Set #1

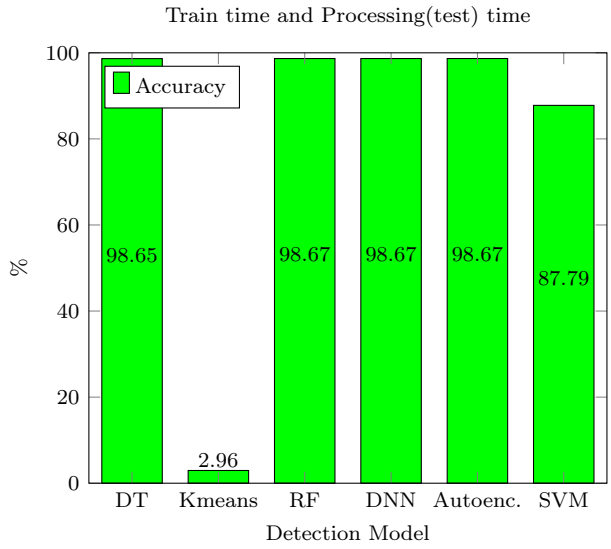

Figure 10: Feature Set #2

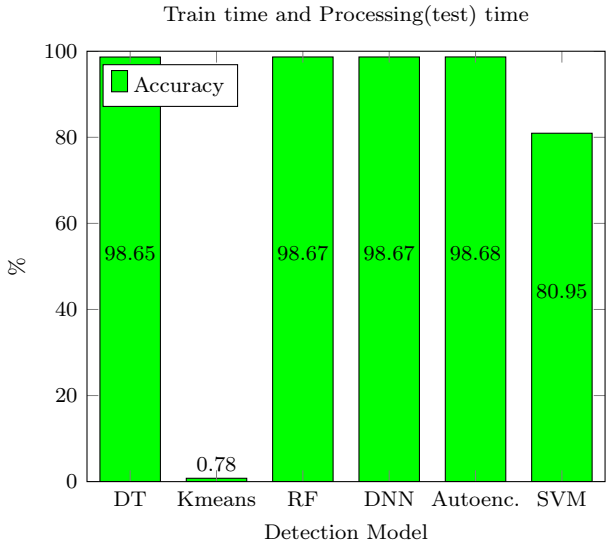

Figure 11: Feature Set #3

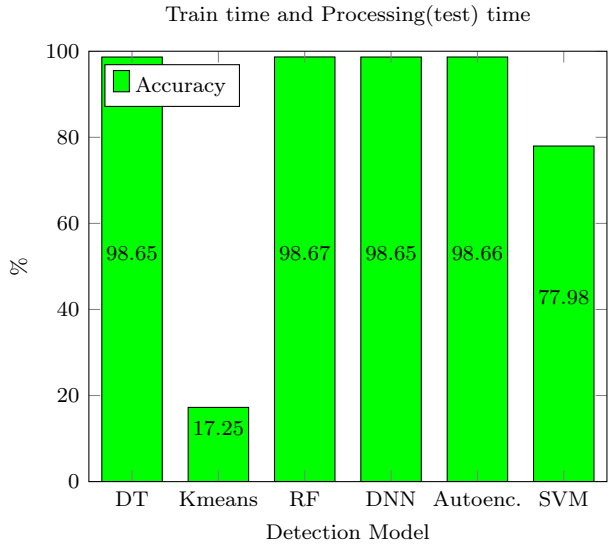

Figure 12: Feature Set #4

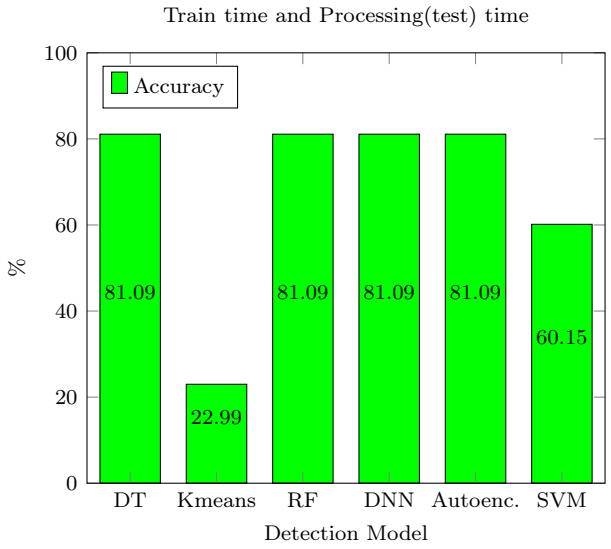

Figure 13: Feature Set #5

Each detection model showed relatively similar performance across the feature sets. To sum up, it is as follows;

- In most cases, feature sets containing flow information showed high accuracy across detection models. It is, even if the accuracy of a specific detection model is low on average, the model with a feature set including flow information showed higher accuracy than one without flow information relatively.
- DT, DNN, Autoencoder, Random Forest showed the highest accuracy among detection models in terms of detection accuracy.
- SVM showed the second accurate result. But accuracy difference between the above models is big.
- Kmeans model was inaccurate than others, lower than 50% in most cases. It showed a weak correlation between feature selection and accuracy.

3.5 Evaluation Summary

From the evaluation of the feature set and detection model, we could reach some conclusions.

- Flow information is more efficient than packet information when detecting Modbus/TCP anomaly. Detecting anomaly with packet information of each packet also works, but the accuracy of only using flow information is almost the same as using all features.
- Removing some part of feature increase performance slightly but reduces accuracy in a trade-off.
- In this project, DT was the best model in terms of performance accuracy. We guess that it is because attack types in the dataset are simple or DT is indeed a good model to detect an anomaly.
- DNN, Random Forest, and autoencoder model also showed an acceptable level of performance and high accuracy.
- Other two models are not recommended to be used. SVM shows less accuracy and worse performance than other models, and Kmeans model failed to detect anomalies properly.

4 Future Work

Section 4 discusses the limitation of the project and future research topics.

4.1 Collecting Dataset and Detection Model

In this project, we used a packet third-party dataset available since we do not have access to the real-world or simulated industrial network. The dataset used in this project is well-generated, but attack types are not varied enough. Collecting more datasets is required for these reasons.

Coverage: models have to detect various attack types/scenarios.

Validation: accuracy of the trained model have to be robust against multiple datasets in detecting an anomaly.

Therefore, collecting available datasets with more attack types/scenarios and generating dataset by ourself can be one of the future work. Also, recently devised learning-based models need to be applied to the framework. Including the latest learning-based models into our framework also should be considered as future work.

4.2 Feature Analysis

Because of lack of time, we did not analyze exactly which feature is helpful to detect exactly which attack type. It should be done because we can reduce features of a feature set that are not useful to detect any anomaly attack. Analyzing the relation between each feature and accuracy of detecting specific attack type/scenario.

5 Conclusion

We proposed the framework for evaluating feature set and learning-based models to efficiently and accurately detect Modbus/TCP anomalies. We evaluated five different feature sets with various learning-based models. Among feature sets and models, feature sets containing the information about packet flow showed high accuracy, and DT, DNN, Random Forest, and Autoencoder models showed high performance and accuracy enough to be applied in the real world. We expect the project result can help design the ICS anomaly detection model in the future.

References

[1] Ivo Frazão; Pedro Abreu; Tiago Cruz; Helder Araújo; Paulo Simões, . Cyber-security modbus ics dataset, 2019.