ISYE 6748 Applied Analytics Practicum
Fortiphyd Logic: Anomaly Detection in Industrial Control Systems

Krishna Kumar and Sumit Machwe

Fall 2022

Contents

1 Introduction

Goal of this project is to devise generalizable anomaly detection in industrial control system sensor data. Fortiphyd Logic is a cybersecurity startup firm that specializes in fortifying industrial networks from malicious cyber attacks.

According to the paper[1] written by the firm's founder, David Fromby, most of the industrial IoT device infrastructure (programmable control logic devices) is vulnerable to cyber-attacks and may result in massive economic losses.

Early detection of such attacks can not only save critical time to recovery but also encourage the PLC manufacturing companies to implement better security controls around the devices and prevent value loss.

For this project, we analyze the sensor data from a scaled-down version of an industrial water treatment plant. The plant consists of six stage filtration process. Water was continuously run for 11 days through the system. Network and physical attacks were launched during this period. The data consists of physical properties related to the plant and treatment process as well as network traffic data. Data files consist of non-anomalous and anomalous data. Please refer to this paper for further details.[2]

2 Approach

Developing a generic anomaly detection process is hard. Unfortunately, one model fits all does not apply, especially when we do not want to incorporate domain knowledge about the dataset. We can however develop a framework that can be applied to other use cases with reasonably minimal changes. The approach we took required not to delve into the physics (or chemistry) of the dataset, but rather identify dependent variables that can be used to describe normal state behavior over time and which can be used in building unsupervised learning model.

In pursuit of such an approach, we discovered 'pycaret' library which organizes various Anomaly Detection algorithms. Under the hood, it uses 'PyOD' open-source library.

2.1 Solution outline

1. Prepare data for analysis and feature selection. Please refer to section 2.3 and 2.4
2. Partition training data vertically across 6 sub-systems.

$$\begin{aligned} S &= \{P_1, P_2, P_3, P_4, P_5, P_6\} \\ M &= \{\text{OCSVM, iforest, Histogram, KNN, LOF, MCD}\} \\ \text{Number of Models} &= S \times M = 6 \times 6 = 36 \\ \text{Models not Built}^3 &= 8 \\ \text{Total Models Built} &= 36 - 8 = 28 \end{aligned} \tag{1}$$

3. Performance Metrics
4. *Why not produce confusion matrix?*

   Because we have used an expansionary definition of actual attacks. Please refer to assumption section 2.2
5. Analyze and conclude the findings. The objective is to find the best-performing model and model group based on performance matrices. Please refer to section 5.
6. key design decisions:
   (a) **Unsupervised** algorithm is preferred even though we have labeled data. In order to develop a generalizable anomaly detection approach, we do not emphasize on number or attributes of parameters.
   (b) We have also used one of the **supervised** learning approaches 'KNN' in order to seek better performance with labeled data.

2.2 Assumptions

Following assumptions apply to our implementation of Anomaly Detection:

1. **Actual attack data identifier**: It is difficult to isolate sensor malfunction from a malicious attack. Without sufficient data knowledge, it is difficult to identify actual attacks in the given attack period. Hence we decided that every record in a given attack window is an actual attack.
2. **Data Partitioning**: The given SWaT training dataset is relatively large (495000 records, 53 features). Subjecting this data to train underlying models could not succeed due to memory constraints. In order to overcome memory issues, and in discussion with project sponsors:
   (a) **Partition training data by days [22-Dec-2015 to 28-Dec-2015]**. However, due to reduced training data the models could not provide acceptable performance. The percentage of identified attacks for OCSVM model was around *19%*.
   (b) **PCA and Multicollinearity**: To reduce feature space, we applied principal component analysis upto 10 dimensions. However, training could not succeed since PCA is a highly memory-intensive process. *However, given better infrastructure, we recommend applying PCA.*

Our EDA highlighted the correlation between sensors and actuators. We tried removing multicollinearity to retain statistical significance. However, we faced memory constraints while removing multicollinearity during training. *However, given better infrastructure, we recommend removing multicollinearity.*

(c) **Partition training data by sub-systems**. Our assumption for this approach is that each subsystem can be analyzed independently and there are negligible interactions across sub-systems. We chose this approach as we could get significantly better results. The percentage of identified attacks for OCSVM model was around **91%**.

3. **Infrastructure Limitations**: Our conclusions and results are based on analyzing all sub-systems except:

$$\textbf{OCSVM} : \{P_3,\ P_4,\ P_5,\ P_6\}$$
$$P_4 : \{\text{MCD, Histogram, KNN, LOF}\}$$

- One class SVM model takes a long time to build on 'Google Colab Pro'. We were able to build the OCSVM models for P_1 and P_2 sub-systems. However, models for other sub-systems failed due to memory constraints.
- For some reasons still unknown to us, 4 models for P_4 sub-systems could not be built due to memory constraints.
- Due to memory constraints, we could not perform PCA or remove multicollinearity. However, based on our EDA analysis, we recommend doing both.

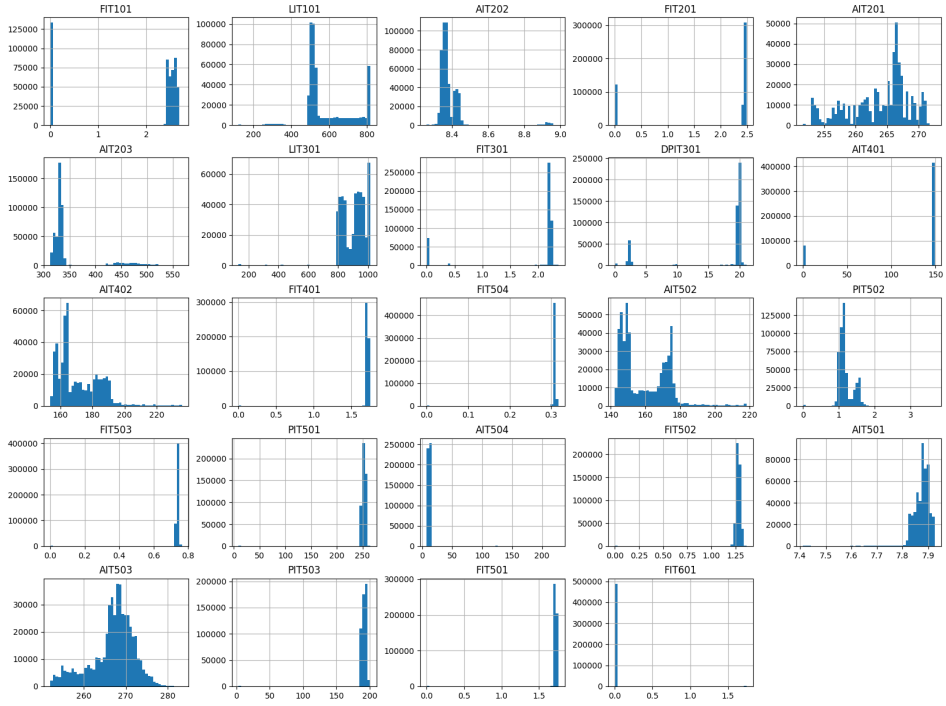
This may cause bias in our analysis and conclusions.

2.3 Exploratory Time series Data Analysis

2.3.1 Data Distribution

We reviewed the sensor and actuator data from various stages of the water treatment plant. The below plot (Figure 1) shows data distribution of numeric features for all subsystems.

Figure 1: Subsystem Sensors $P_1, P_2, P_3, P_4, P_5, P_6$ - Data Distribution Histogram



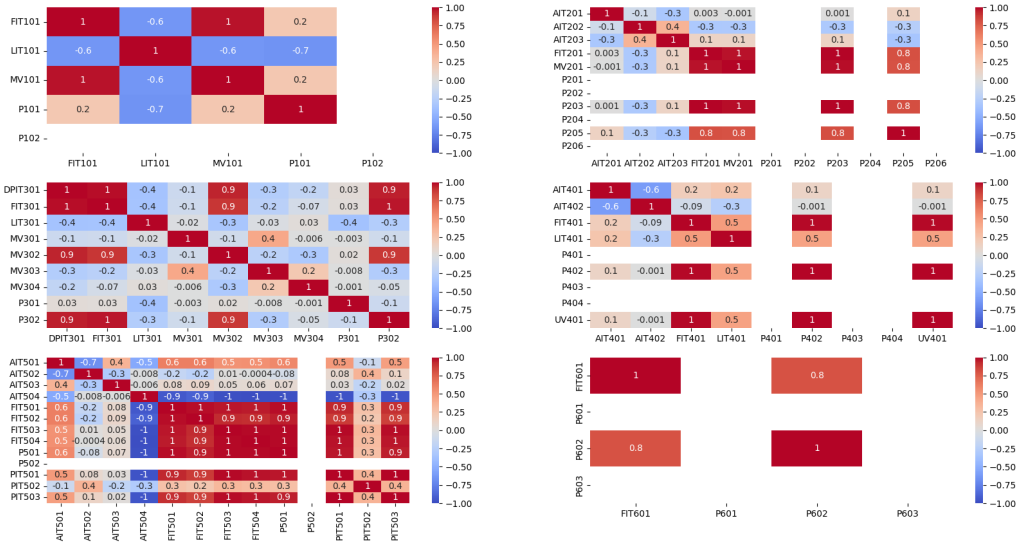
The data distribution shows the multi-modal normal distribution with long tails for $FIT101$, $UT101$, $AIT202$,

$AIT201$, $AIT203$, $LIT301$, $AIT402$, $AIT502$, $PIT502$, $AIT501$, and $AIT503$. Specific sub-systems such as $FIT501$, $FIT601$, $FIT401$, $FIT504$, and $AIT504$ do not seem to have Gaussian distribution as data distribution is concentrated to specific ranges. Based on the large data size, we can reasonably assume a normal distribution (Based on Central Limit Theorem - Gaussian Mixture model - Maximum Likelihood Probability). It is possible to transform the data to see if it has an unbiased normal distribution - but we decided not to do it and proceed with modeling.

2.3.2 Heatmap

For each sub-system, we created a correlation heatmap to determine if the sensor/actuator data are correlated. If they are highly correlated, we may decide to do dimensionality reduction using PCA (Principal Component Analysis). Please refer to the below plot (Figure 2) for the correlation Heatmap across subsystems Based

Figure 2: Subsystem Sensors $P_1, P_2, P_3, P_4, P_5, P_6$ - Correlation Heatmap



on the correlation Heatmap, we can conclude that specific features within subsystems (for example $FIT101$ & $LIT401$, $FIT201$ & $P203$, $FIT301$ & $P302$, $FIT401$ & $P206$, $FIT501$ & $FIT502$, $FIT601$ & $P602$) are highly correlated. We should use PCA and remove multicollinearity while modeling.[?]

2.3.3 Sub System Breakup

sub system	continuous variables	categorical variables (actuators)	Ignore (constants)
P_1	$FIT101$, $LIT101$, $MV101$, $P101$, $P102$	$MV101$, $P101$, $P102$	$P102$
P_2	$AIT201$, $AIT202$, $AIT203$, $FIT201$, $MV201$, $P201$, $P202$, $P203$, $P204$, $P205$, $P206$	$MV201$, $P201$, $P202$, $P203$, $P204$, $P205$, $P206$	$P201$, $P202$, $P204$, $P206$
P_3	$DPIT301$, $FIT301$, $LIT301$, $MV301$,$MV302$, $MV303$, $MV304$, $P301$, $P302$	$MV301$, $MV302$, $MV303$, $MV304$, $P301$, $P302$	
P_4	$AIT401$, $AIT402$, $FIT401$, $LIT401$, $P401$, $P402$, $P403$, $P404$, $UV401$	$LIT401$, $P401$, $P402$, $P403$, $P404$, $UV401$	$P401$, $P403$, $P404$
P_5	$AIT501$, $AIT502$, $AIT503$, $AIT504$, $FIT501$, $FIT502$, $FIT503$, $FIT504$, $P501$, $P502$, $PIT501$, $PIT502$, $PIT503$	$P501$, $P502$	$P502$
P_6	$FIT601$,$P601$, $P602$, $P603$	$P601$, $P602$	$P601$, $P603$

Table 1: Partition SWaT Dec 2015 spreadsheet into 6 sub systems
Ignore (Constants): Based on data analysis, these variables were found to have constant values and would not contribute to describing the variance in the model. They can be removed from the model-building exercise.

2.3.4 Visualize Normal Data by sub-systems

Visualizing training and test data is essential for analysis. Below graphs plot the continuous (Sensor data) and categorical (actuator data) variables on a log scale. Data being plotted is the mean value over 1 minute interval. Original data is provided at per second granularity.

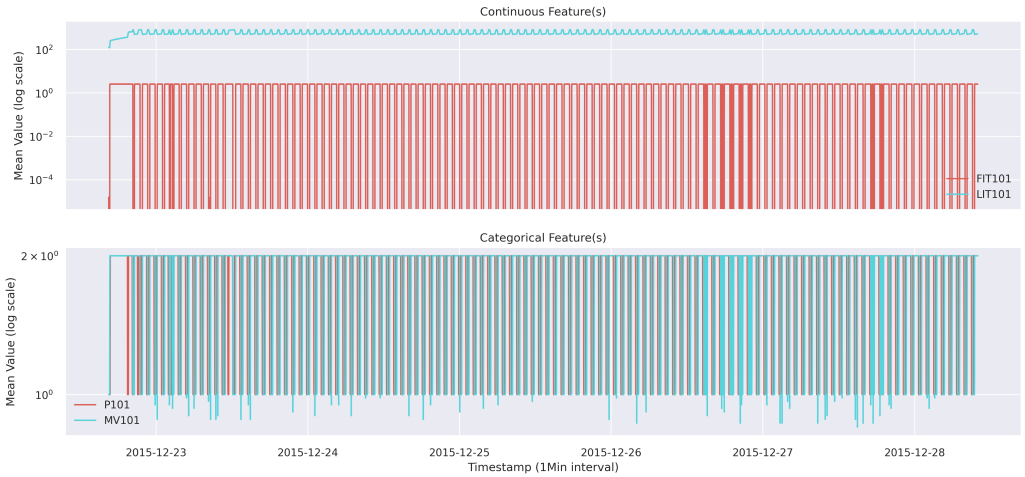


Figure 3: P_1 Sub-system data (log scale), aggregated in 1-minute interval

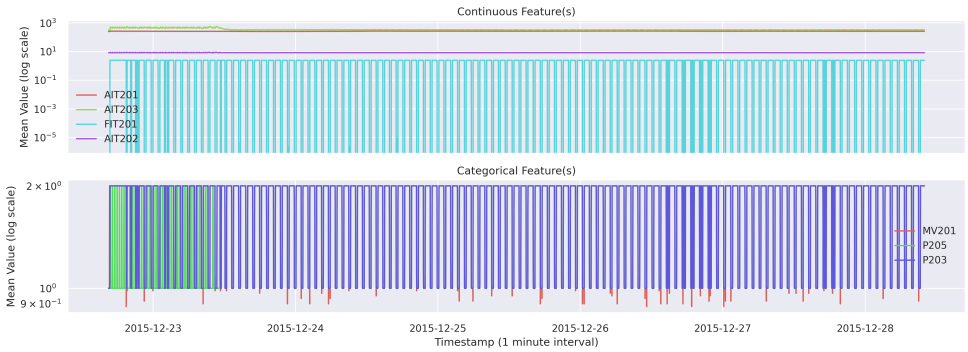


Figure 4: P_2 Sub-system data (log scale), aggregated in 1-minute interval

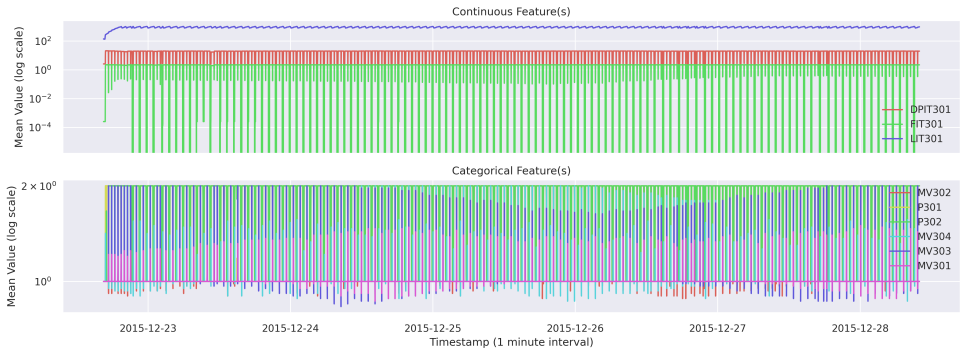


Figure 5: P_3 Sub-system data (log scale), aggregated in 1-minute interval

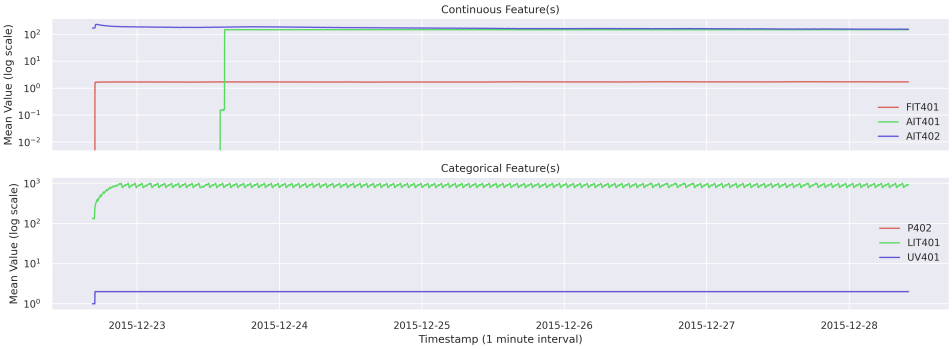


Figure 6: P_4 Sub-system data (log scale), aggregated in 1-minute interval

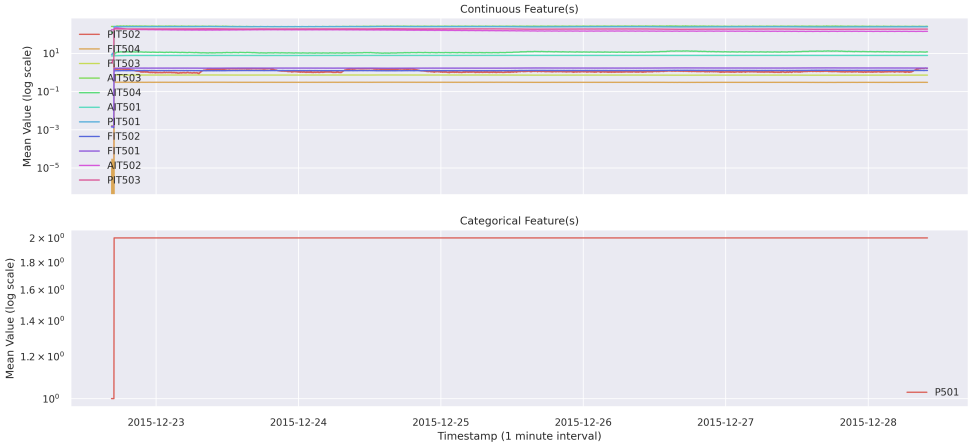


Figure 7: P_5 Sub-system data (log scale), aggregated in 1-minute interval

Figure 8: P_6 Sub-system data (log scale), aggregated in 1-minute interval

2.3.5 Visualize Attack Data by sub-systems

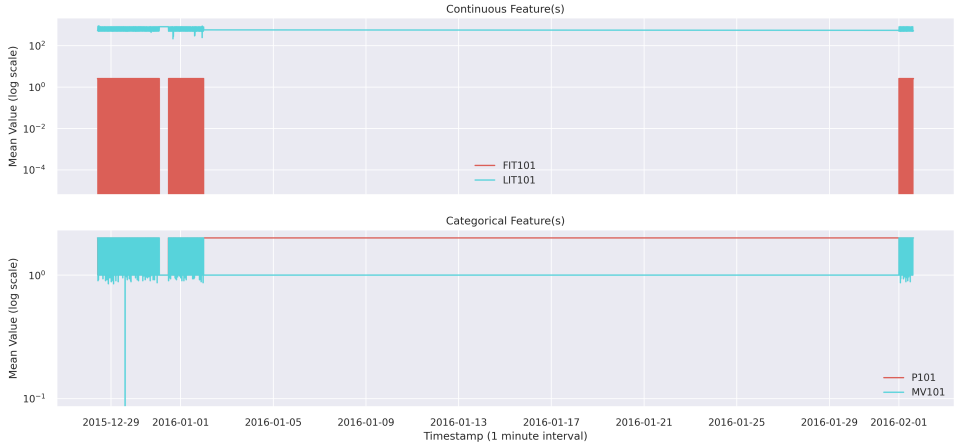


Figure 9: P_1 Sub-system data (log scale), aggregated in 1-minute interval

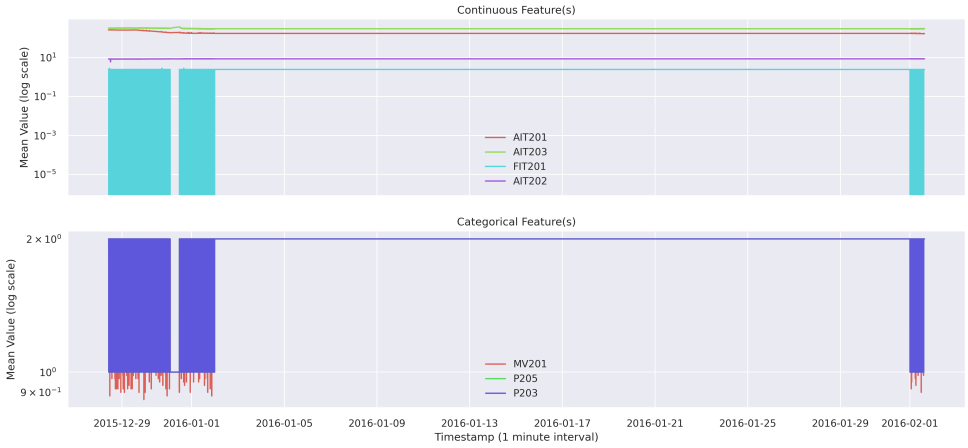


Figure 10: P_2 Sub-system data (log scale), aggregated in 1-minute interval

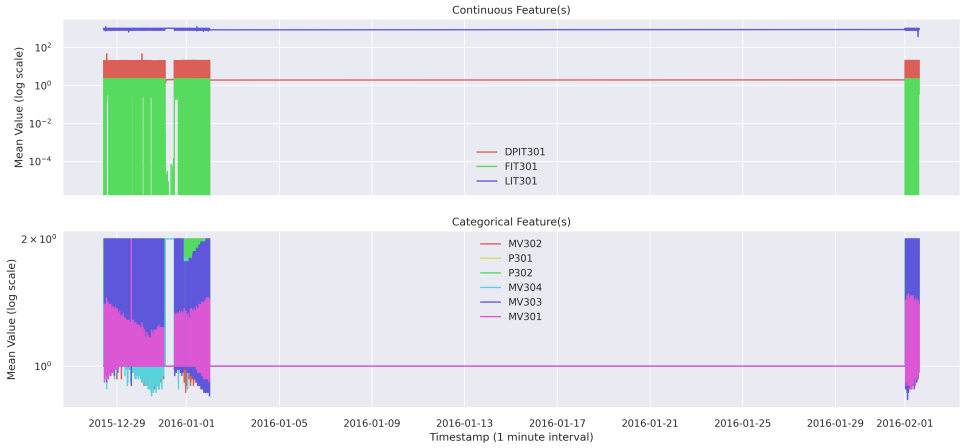


Figure 11: P_3 Sub-system data (log scale), aggregated in 1-minute interval

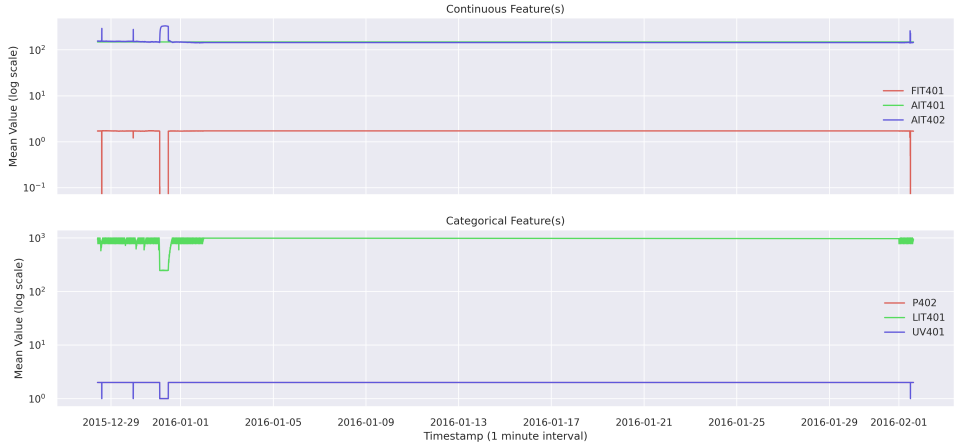


Figure 12: P_4 Sub-system data (log scale), aggregated in 1-minute interval.
We can see anomalous behavior in Sensor and Actuator data P_5 sub-system on 01-Jan-2016.

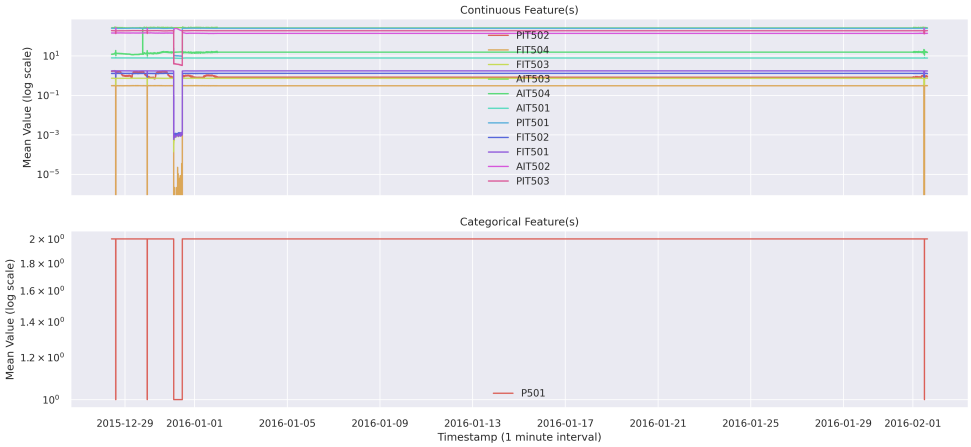


Figure 13: P_5 Sub-system data (log scale), aggregated in 1-minute interval.
Similar to P_4, we can see anomalous behavior in Sensor and Actuator data P_5 sub-system on 01-Jan-2016.

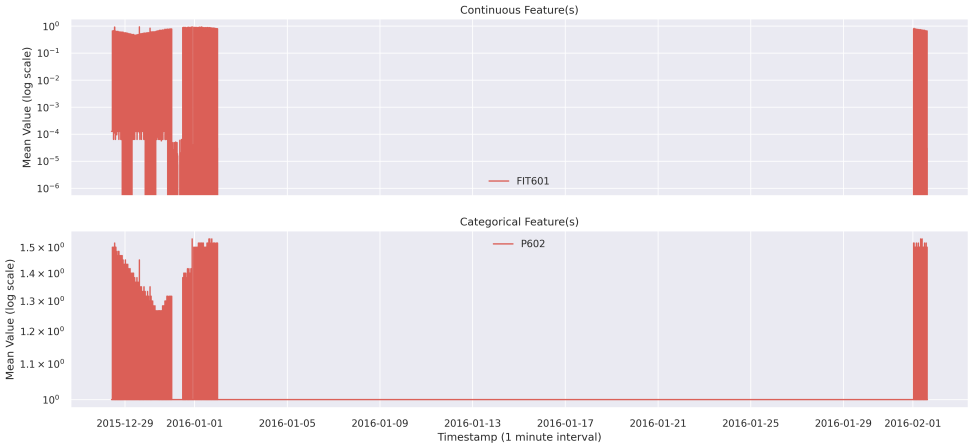


Figure 14: P_6 Sub-system data (log scale), aggregated in 1-minute interval

2.4 Feature selection

Almost all features required for the model are provided in the data. However, the following data-cleaning tasks are required in order to utilize the dataset:

1. Create valid data frames with headers and values
2. Convert DateTime strings to Timestamp and sensor/actuator data values to numeric.
3. Valid column names and values removing leading/trailing spaces

As part of feature engineering, each sub-system sensor/actuator data will be extracted as a feature for unsupervised learning.
We will also ignore the label data 'Normal/Attack' as per recommendation.

2.5 Develop Unsupervised Model

To develop the model, we will take the following steps:

1. We will train the model for each of the Sub-systems separately. This decision was taken due to memory/infrastructure constraints for training the full model with all records and features. We assume that each of the sub-systems is independent and hence it will not bias the model's predictions. We validated this assumption with Project sponsors.
2. Merge non-anomalous and anomalous data while creating a 'class' feature.
3. Ensure that all data points either exhibit Gaussian (Normal) Distribution, or that they are transformed such that their distribution is normal. This is an important step before applying any probability-based learning algorithm.
4. Split the data into Training, Cross Validation (CV), and Test. However, the Training set will consist of completely non-anomalous data, whereas CV and Test datasets will distribute anomalous data 50% each.
5. Implement Unsupervised algorithm that computes Local Outlier Factor (LOF)[3] using Mahalanobis Distance. LOF is an anomaly score computed for each sample. The locality is defined in terms of k-nearest neighbors using some distance measure. Mahalanobis works better with higher dimensional data.

2.6 Model validation

We trained the models on first 8 days of SWaT: 22-Dec-2015 to 28-Dec-2015 10:00 AM. And tested it against data from 28-Dec-2015 to 02-Jan-2016.

3 Algorithm Description

There are subtle differences between Outlier Detection and Novelty detection.[12] Outliers are essentially observations that significantly deviate from all other observations in a given training data. They can be treated as contamination and are generally discarded before using the training dataset. On the other hand, Novelty detection attempts to figure out if a new/unseen observation has a similar distribution as the training data. If the distributions do not match, we declare the new observation as an anomaly.

There are various algorithms that help with detecting outliers as well as Novelties. We explored the below-listed algorithms on the provided dataset to come up with a robust anomaly detection model:

- Isolation Forest
- Minimum Covariance Determinant (MCD)
- One-Class Support Vector Machine (OCSVM)
- Local Outlier Factor (LOF)
- Histogram-Based Outlier
- K-nearest Neighbor (KNN)

3.1 iforest

An efficient way of performing outlier detection in high-dimensional datasets is to use random forests. The 'iforest'[5] algorithm 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Recursive partitioning can be

represented by a tree structure. The number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

Anomalies have a shorter path. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies. 'iforest' can be used to detect Novelty data-point.

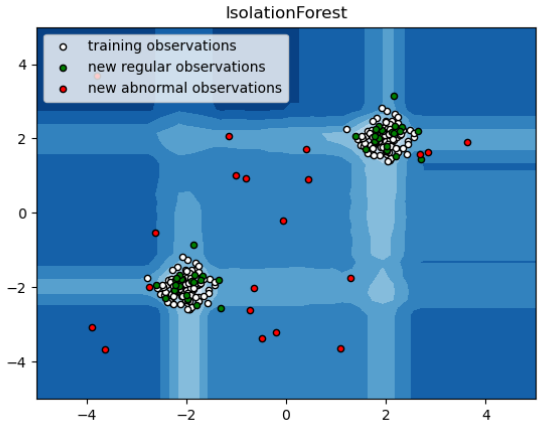


Figure 15: Source: Scikit-learn Isolation Forest Example[13]

3.2 Minimum Covariance Determinant (MCD)

Anomalies are detected by assuming that non-anomalous data are generated by a particular probability distribution and declaring points with low probability density as anomalies. For elliptically distributed (e.g. Gaussian) data, this can be done by computing the Mahalanobis distance from each point to the mean and defining anomalies as points with distance above some threshold.

The Mahalanobis distance requires the parameters of the distribution (mean and covariance matrix).[14] Since these are unknown, they must be estimated from the data. MCD[6] is a method for estimating the mean and covariance matrix in a way that tries to minimize the influence of anomalies. The idea is to estimate these parameters from a subset of the data that has been chosen to (hopefully) not contain anomalies. The idea behind minimizing the determinant is that the determinant of a covariance matrix measures how broad the distribution is. MCD, therefore, selects the subset of the data that is most tightly distributed. This is to exclude anomalies, which are likely to lie further away from the rest of the data.

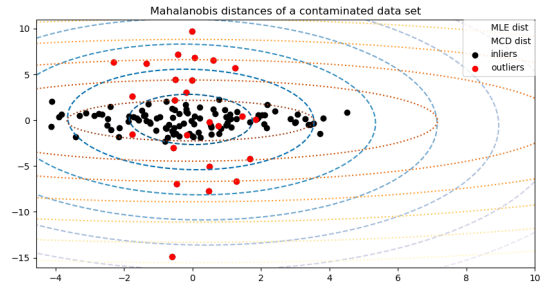


Figure 16: Source: Scikit-learn Robust Covariance[15]
Separating inliers from outliers using a Mahalanobis distance

3.3 OneClass SVM

One-Class Support Vector Machine (OCSVM)[7] answers the problem that only one class is available. It models the properties of the normal class. When the model is applied to unknown data that may contain anomalous data, the model will be able to detect them.

Support Vector Machine (SVM) is a supervised learning algorithm that can be used for both classification and regression problems. CVM can create a non-linear decision boundary to separate two classes and can be

applied to the higher dimensional dataset.

The One-class SVM by Schölkopf et al. [4] separates all the data points from the origin in a higher dimensional space and maximizes the distance from this hyperplane to the origin. In other words, the origin is the class that the algorithm tries to separate from the normal class. The OCSVM outlier score is the distance (also called Similarity Measure) of a data point to the hyperplane. It is computed by a kernel function such as the Radial Basis Function (RBF), linear, polynomial, or sigmoidal function.

Below figure shows comparision between various outlier detection algorithms.

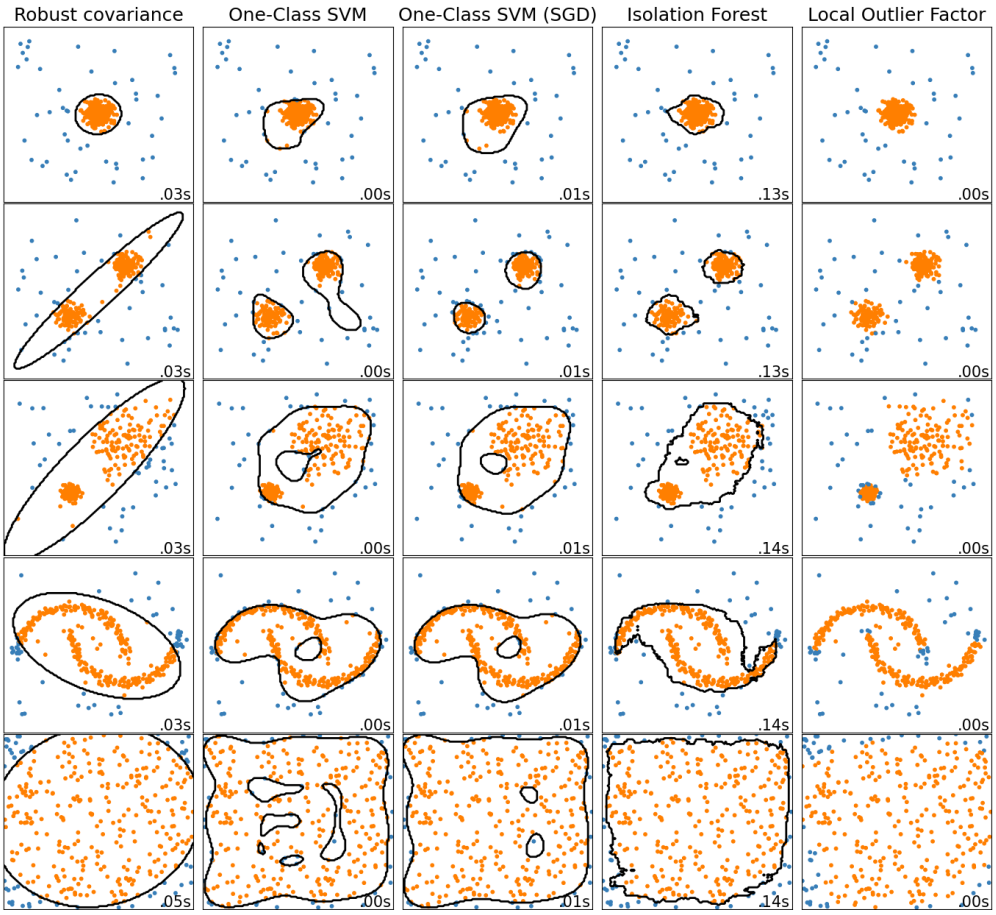


Figure 17: Source: Scikit-learn Comparing anomaly detection algorithms for outlier detection on toy datasets[16]

3.4 Local Outlier Factor

LOF[8] computes a score reflecting the degree of abnormality of the observations. It measures the local density deviation of a given data point with respect to its neighbors. The idea is to detect the samples that have a substantially lower density than their neighbors. Local density is obtained from the k-nearest neighbors. lof score = average local density of its k-nearest neighbors / own local density.

$$LRD_k(A) = \frac{1}{\sum_{X_j \in N_k(A)} \frac{RD(A,X_j)}{\|N_k(A)\|}}$$

$$LOF_k(A) = \frac{\sum_{X_j \in N_k(A)} LRD_k(X_j)}{\|N_k(A)\|} \times \frac{1}{LRD_k(A)} \tag{2}$$

RD = reachability distance of A from its neighbors

LRD = local reachability density

LOF = Local Outlier Factor

3.5 Histogram-Based Outlier

In Histogram Based Outlier[9] detection approach, we compute the count statistic, called a histogram, for each dependent variable in training data. If a value of observation falls in the tail of a histogram, the value is an outlier. It is possible that some values of the observation are outliers in terms of the corresponding variables, but some values are normal. If many values of observation are outliers, the observation is very likely to be an outlier.
With this intuition, the Histogram-based Outlier Score (HBOS) uses the histogram to define the "outlier-ness", called the univariate outlier score of a variable. An observation should have a corresponding univariate outlier score for each variable. These univariate outlier scores can be summed up to measure the overall "outlier-ness" of an observation.

3.6 KNN

The unsupervised k-NN[10] method computes the Euclidean distance of observation to other observations. There are no parameters to tune. Since an outlier is a point that is distant from neighboring points, the outlier score is defined as the distance to its k^{th} nearest neighbor. Each point will have an outlier score. The algorithm finds those points with high outlier scores and declares them as anomalous.

4 Performance Metrics

This section provides our methodology to evaluate the performance of machine learning models for anomaly detection. Based on the best-performing model/model group, we provide our recommendations. We also provide our recommendations for further improvement of model performance and generalization of anomaly detection for industrial control systems.

Definition of metrics used to evaluate ML model performance.

1. **Percentage of identified attacks**:

   For evaluating the model performances, we used an absolute metric **% of Identified attacks**. For a given attack window, we considered the total records within the attack window as **Actual Attacks (denominator)**. Based on the trained model's prediction for given features, we predict Anomaly Score. We standardize Anomaly Score and mark the record as anomalous if it breaches a specific sigma threshold (say $\sigma = 0.5$ or $\sigma = 1.0$). We count the total no. of breached records as **Predicted Attacks (numerator)**

   $$\%\ \text{of Identified attacks} \equiv \frac{\text{Predicted Attacks}}{\text{Actual Attacks}} * 100 \tag{3}$$

   Anomaly Score :[11]The anomaly Score is the value computed by the corresponding algorithm. Outliers / Not-Normal points are identified with large anomaly scores.

2. **Propensity of attack**:

   For evaluating the model performances, we used another absolute metric **Propensity of attack**. For a given attack window, we considered the total records within the attack window as **Actual Attacks**. Based on the trained model's prediction for given features, we identify the earliest time interval (in seconds) within which the Attack is identified.

   Propensity of Attacks $\equiv$ Earliest Time Interval (in seconds) within which the actual attack is identified

5 Results, analysis and conclusion

5.1 Overall results

Sr. No	Model Group	Actual Attacks (Count)	Identified Attacks (Count)	% of Identified attacks
1	**SVM**	101524	91996	90.62
2	Histogram	253810	176998	69.74
3	iforest	304572	194666	63.91
4	MCD	253810	135462	53.37
5	KNN	253810	68174	26.86
6	lof	253810	32122	12.66

Table 2: Model-Group Performance Sorted by % of Identified attacks (Descending) for $\sigma = 0.5$
MCD: Minimum Covariance Determinant, lof: Local Outlier Factor, KNN: K-Nearest Neighbor, SVM: One Class SVM

Sr. No	Model Group	Actual Attacks (Count)	Identified Attacks (Count)	Mean of Propensity (In Seconds)
1	**SVM**	101524	91996	43.82
2	iforest	304572	194666	215.38
3	Histogram	253810	176998	451.88
4	MCD	253810	135462	488.98
5	KNN	253810	68174	784.25
6	lof	253810	32122	1423.04

Table 3: Model-Group Performance Sorted by Mean of Propensity (Ascending) for $\sigma = 0.5$
MCD: Minimum Covariance Determinant,lof: Local Outlier Factor,KNN: K-Nearest Neighbor, SVM: One Class SVM

Sr. No	Model Group	Actual Attacks (Count)	Identified Attacks (Count)	% of Identified attacks
1	**MCD P_3**	50762	50393	99.27
2	SVM P_2	50762	49700	97.91
3	Histogram P_3	50762	49432	97.38
4	Histogram P_1	50762	47732	94.03
5	MCD P_1	50762	47598	93.77
6	iforest P_2	50762	46117	90.85

Table 4: Model Performance Across Attacks Sorted by % of Identified attacks (Descending) for $\sigma = 0.5$
textsfMCD: Minimum Covariance Determinant, SVM: One Class SVM

Sr. No	Model Group	Actual Attacks (Count)	Identified Attacks (Count)	Mean of Propensity (In Seconds)
1	**MCD P_3**	50762	50393	0.0
2	iforest P_1	50762	28415	1.16
3	Histogram P_3	50762	49432	13.81
4	SVM P_2	50762	49700	20.45
5	Histogram P_3	50762	47732	24.45
6	MCD P_1	50762	47598	33.94

Table 5: Model Performance Across Attacks Sorted by Mean of Propensity (Ascending) for $\sigma = 0.5$
MCD: Minimum Covariance Determinant

Sr. No	Model Name	Attack ID	Actual Attacks (Count)	Identified Attacks (Count)	% of Identified attacks
1	iforest P_1	1	940	940	100.0
2	Histogram P_2	10	160	160	100.0
3	Histogram P_2	21	721	721	100.0
4	Histogram P_2	20	395	395	100.0
5	Histogram P_2	19	259	259	100.0
6	Histogram P_2	17	717	717	100.0

Table 6: Model Performance for Specific Attack ID Sorted by % of Identified attacks (Descending) for $\sigma = 0.5$

Sr. No	Model Name	Attack ID	Actual Attacks (Count)	Mean of Propensity (In Seconds)	% of Identified attacks
1	**iforest** P_1	1	940	940	0.0
2	Histogram P_2	10	160	160	0.0
3	Histogram P_2	21	721	721	0.0
4	Histogram P_2	20	395	395	0.0
5	Histogram P_2	19	259	259	0.0
6	Histogram P_2	17	717	717	0.0

Table 7: Model Performance for Specific Attack ID Sorted by Mean of Propensity (Ascending) for $\sigma = 0.5$

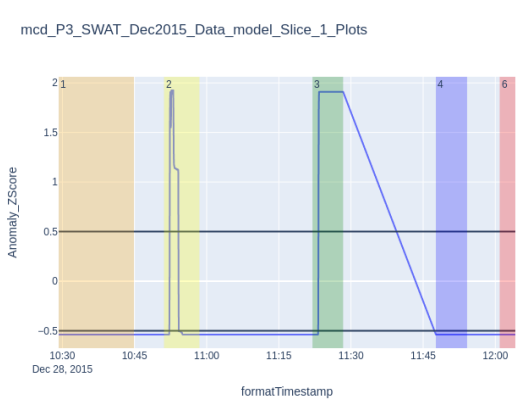
Sr. No	Model Group	Actual Attacks(Count)	Identified Attacks(Count)	% of Identified attacks
1	**Histogram**	253810	151693	59.77
2	iforest	304572	137166	45.04
3	MCD	253810	109245	43.04
4	KNN	253810	46946	18.50
5	lof	253810	32122	12.66
6	SVM	101524	9818	9.67

Table 8: Model-Group Performance Sorted by % of Identified attacks (Descending) for Standard Deviation of $\sigma = 1.0$
MCD: Minimum Covariance Determinant,lof: Local Outlier Factor,KNN: K-Nearest Neighbor

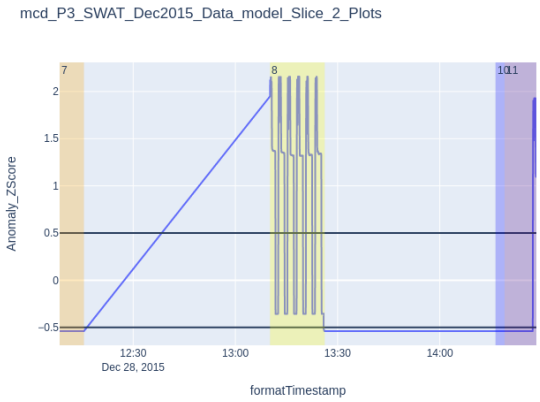
Sr. No	Model Group	Actual Attacks(Count)	Identified Attacks(Count)	Mean of Propensity (In Seconds)
1	**iforest**	304572	137166	339.11
2	Histogram	253810	151693	569.08
3	MCD	253810	109245	645.99
4	SVM	101524	9818	880.45
5	KNN	253810	46946	1087.95
6	lof	253810	32122	1423.04

Table 9: Model-Group Performance Sorted by Mean of Propensity (Ascending) for $\sigma = 1.0$
MCD: Minimum Covariance Determinant, lof: Local Outlier Factor, KNN: K-Nearest Neighbor

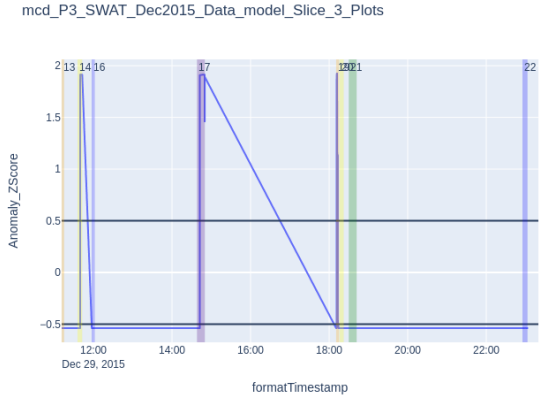
5.2 Best Performing model Plots



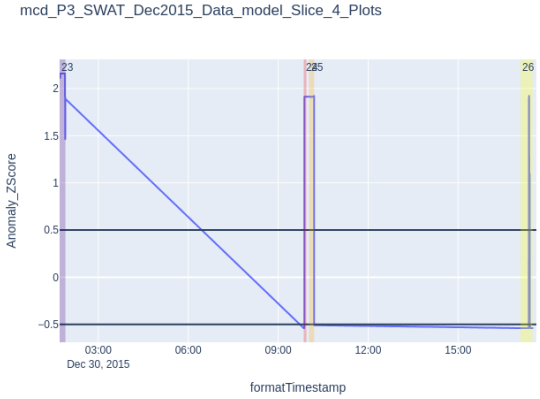
(a)



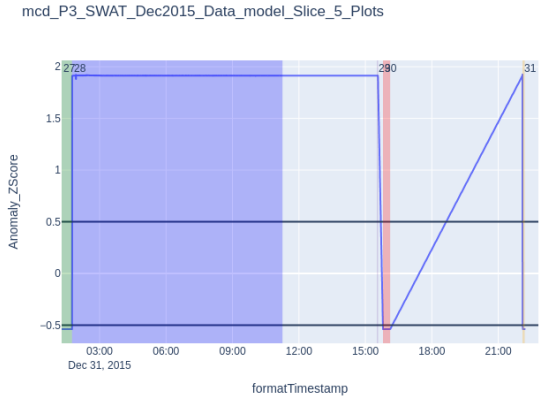
(b)



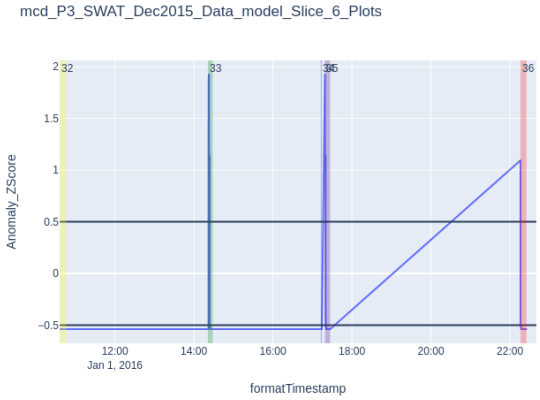
(c)



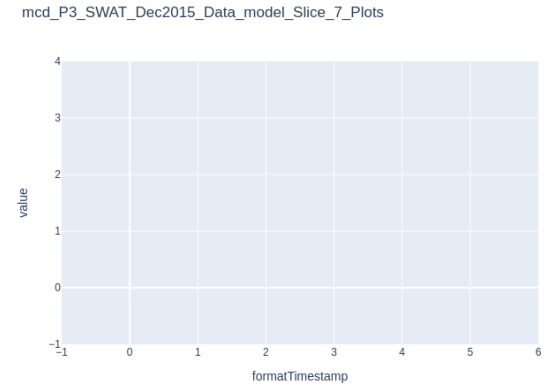
(d)



(e)

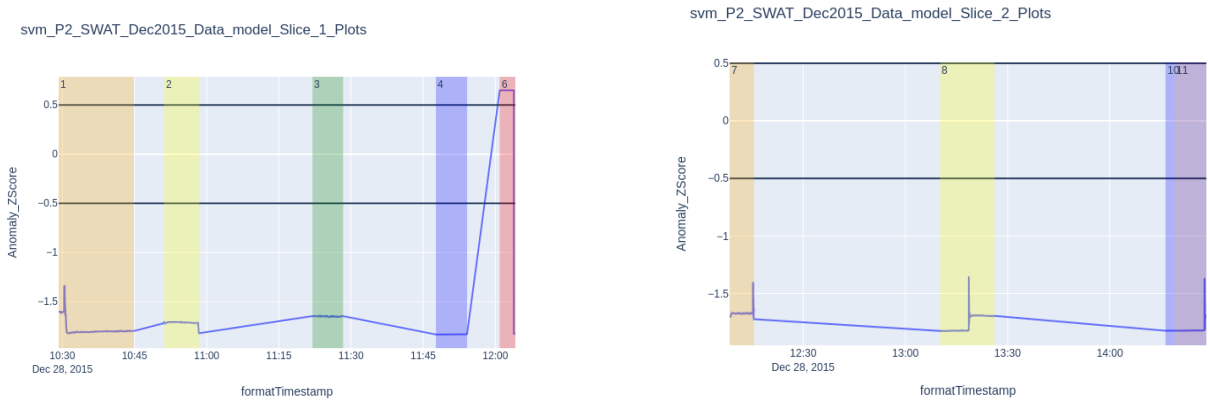


(f)

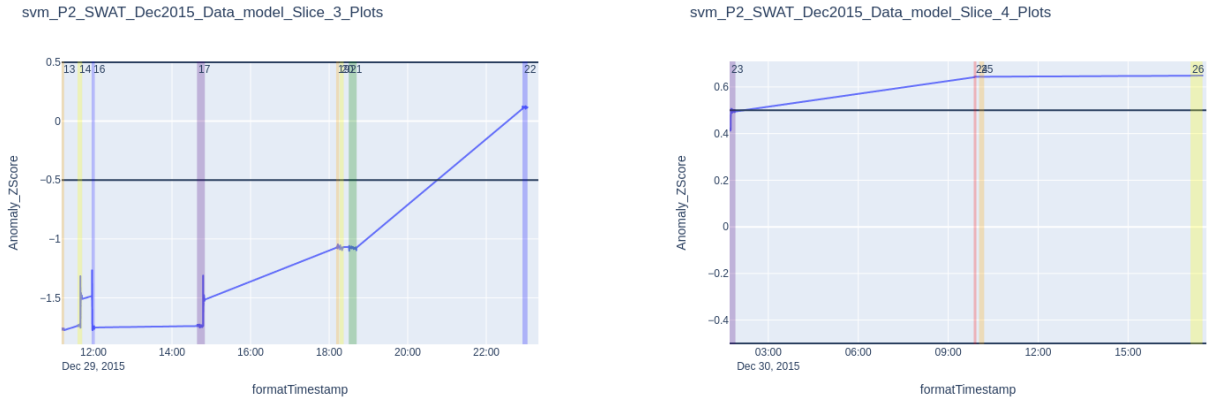


(g)

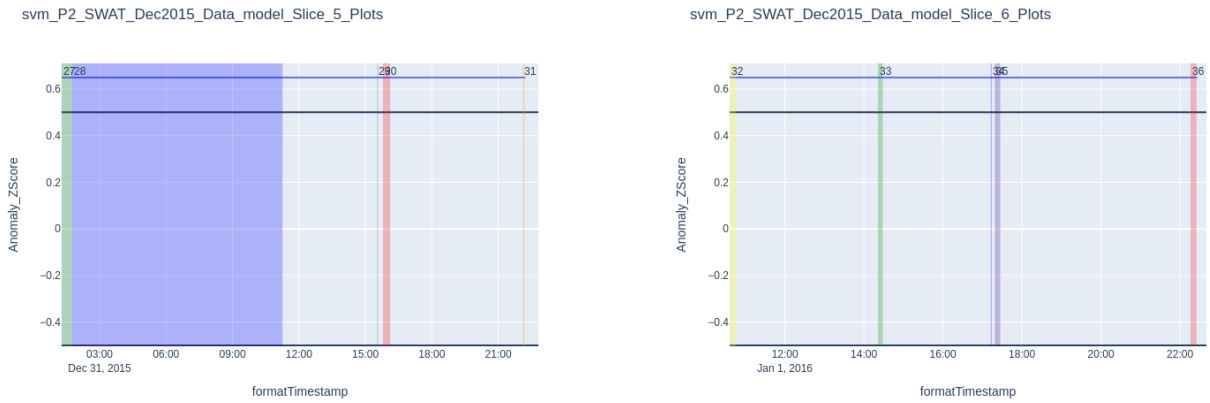
Figure 18: Minimum Covariance Determinant (MCD) Subsystem P3 model performance



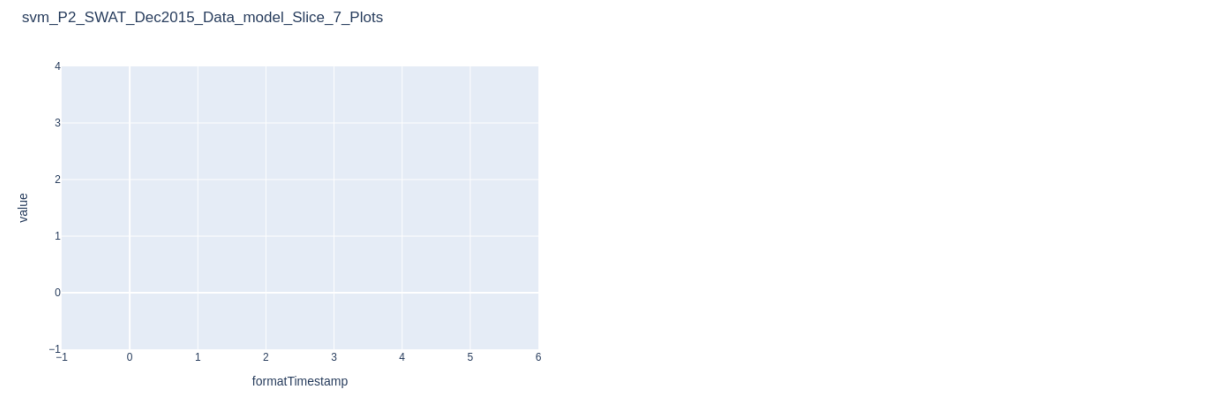
(a) (b)



(c) (d)

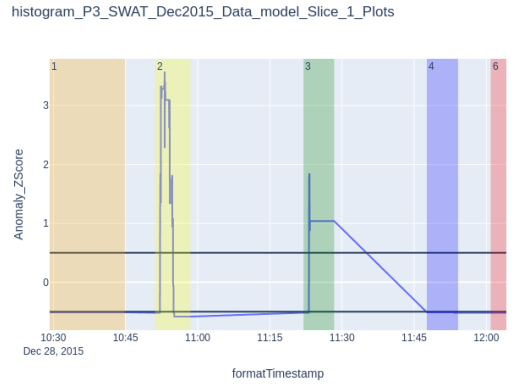


(e) (f)

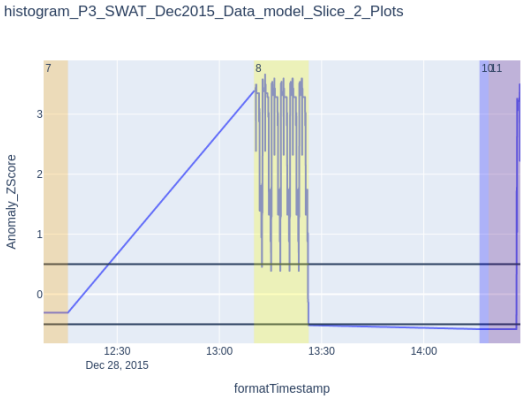


(g)

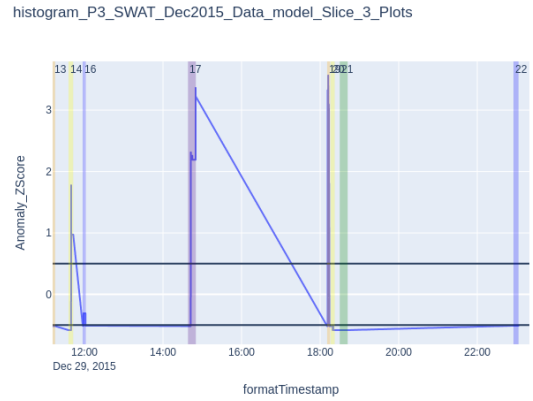
Figure 19: SVM Subsystem P2 model performance



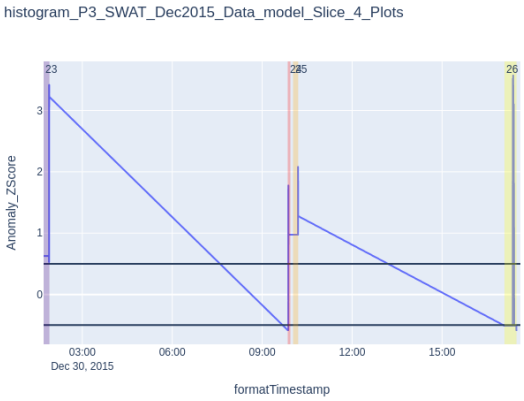
(a)



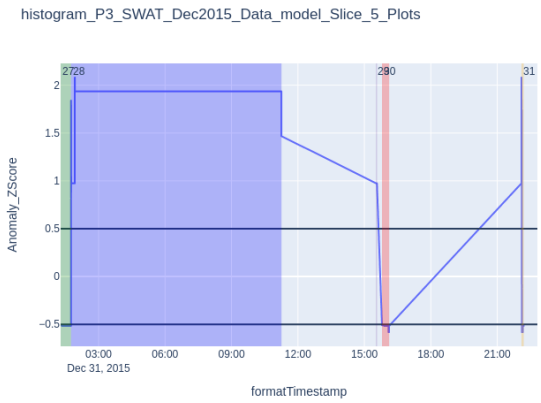
(b)



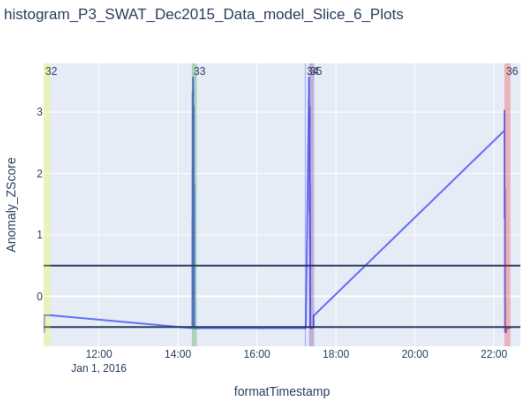
(c)



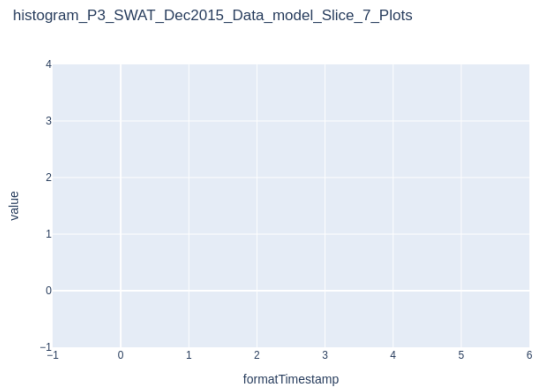
(d)



(e)

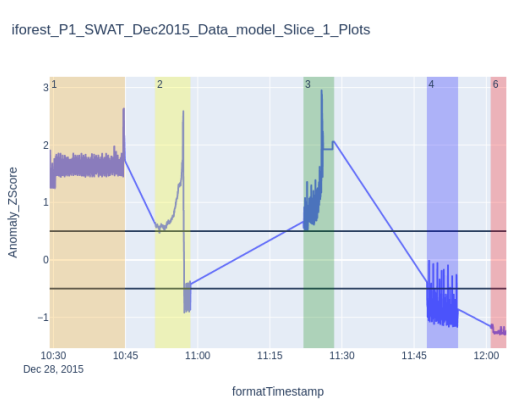


(f)

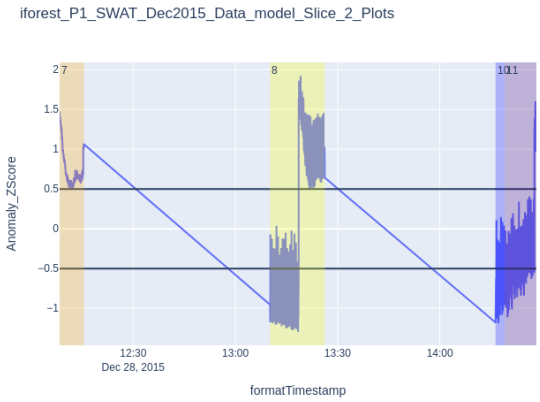


(g)

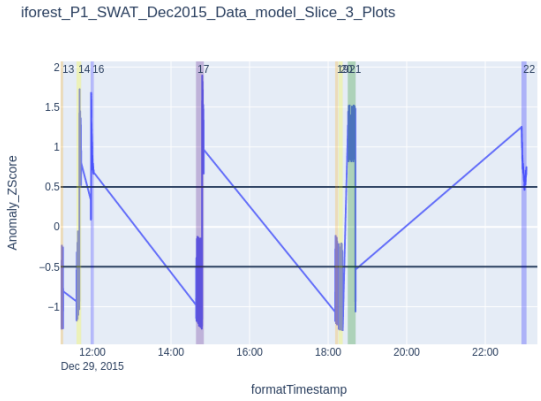
Figure 20: Histogram Subsystem P3 model performance



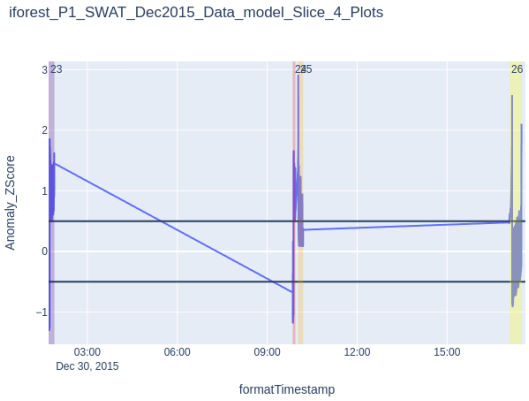
(a)



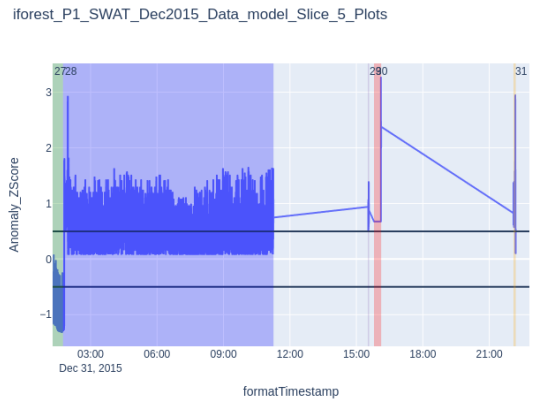
(b)



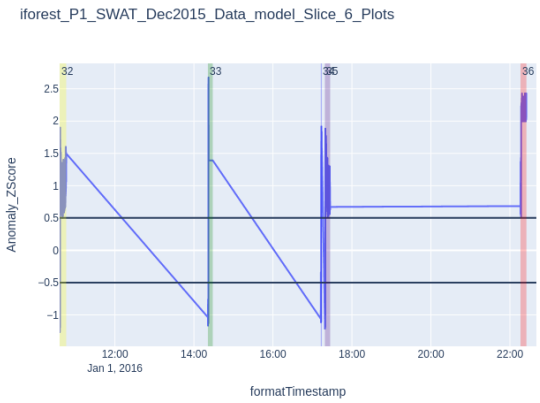
(c)



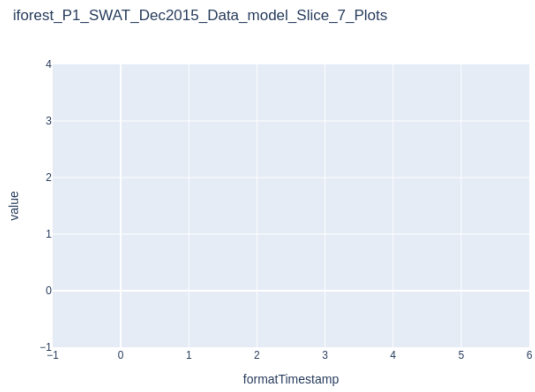
(d)



(e)



(f)



(g)

Figure 21: iforest Subsystem P1 model performance

Figure 22: **Top 3 Performing Models by % of Identified Attacks**

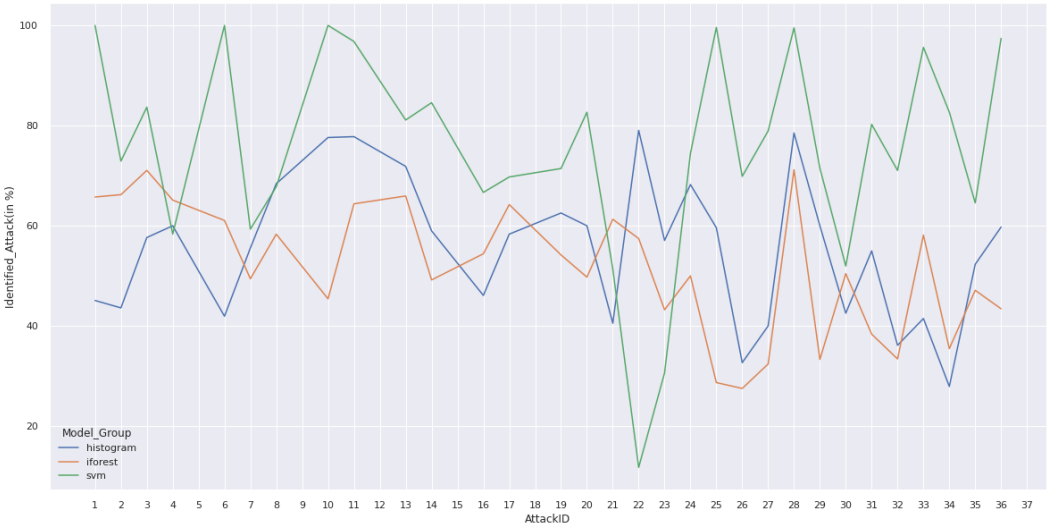


Figure 23: **Bottom 3 Performing Models by % of Identified Attacks**

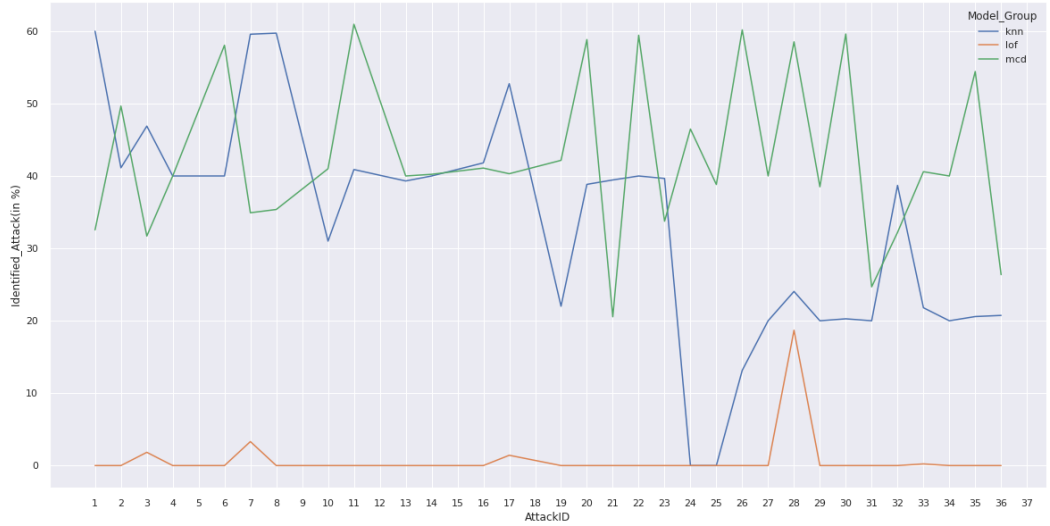


Figure 24: **Top 3 Performing Models by Mean Of PropensityOfAttack (In Seconds)**

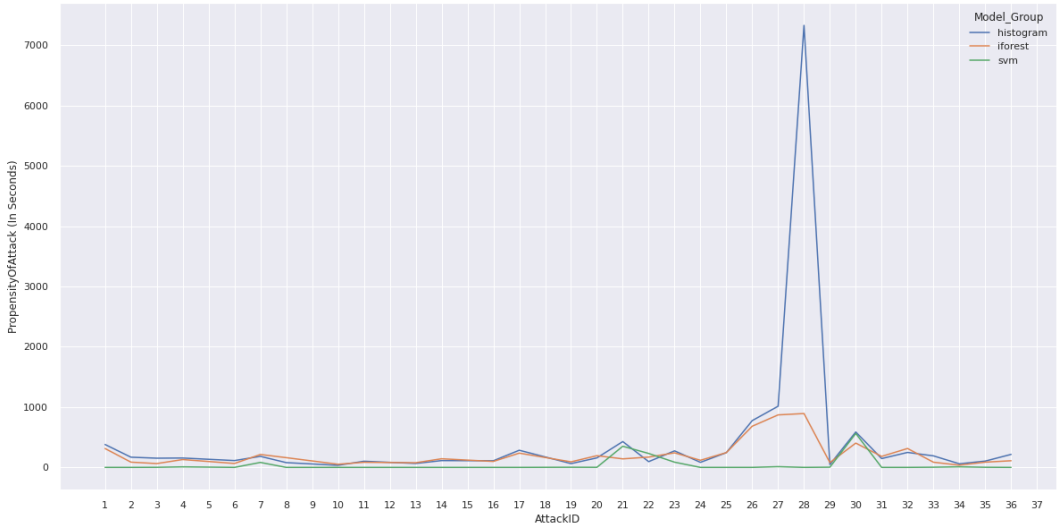
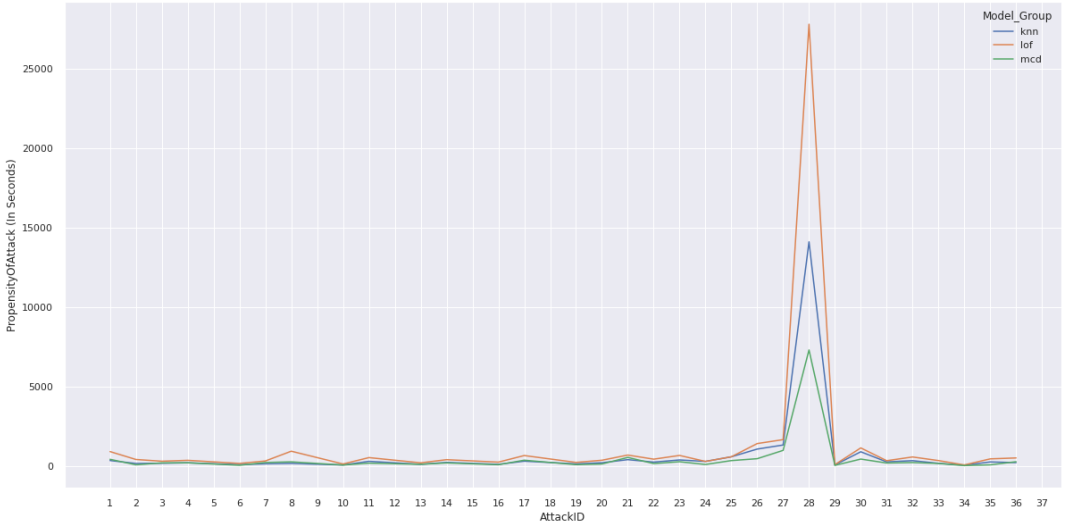


Figure 25: **Bottom 3 Performing Models by Mean Of PropensityOfAttack (In Seconds)**



5.3 Analysis on best performing ML algorithm and Models and Recommendations

1. **Analysis on % Of Identified Attacks**
   (a) We observed that OneSVM model has the best performance metrics (**% of Identified Attacks: 90.62%**). Please refer to Table 2. Based on the underlying methodology of OneSVM, where it creates a non-linear decision boundary to separate two classes, It is not surprising that it consistently identifies the anomaly and performs the best. Please refer to Figure 19, where we have plotted OneSVM anomaly detection for the model trained on the sub-system P2. As well, OneSVM is able to identify the attack earliest as it has the lowest Propensity to identify the attacks. Please refer to Table 3.

   (b) We also plotted the Top 3 performing models by % of Identified Attacks across all attacks (Table 2 and Figure 22: Attack 1 through Attack 41 - Between 28 December 2015 and 2nd-Jan-2016). We observe from the plot that OneSVM consistently identifies the attack better (higher) among all other models. Only for attacks 21 through 23 (between 29th December 2015 6.30 PM and 1.42 PM), OneSVM performance is worse than the other models (Histogram, iForest, knn , and lof). This need to be looked at closely and analyzed further.

   (c) Apart from OneSVM, the other two best-performing models are Histogram and iforest (Table 2). Histogram determines anomaly based on Count Statistics. The tail distribution of the Histogram is identified as Anomaly. Under Central Limit Theorem, for large Data Sizes, Histograms will converge to Gaussian distribution and the tail will be identified as an outlier. Hence, Histogram does a better job of identifying attacks. (**% of Identified Attacks: 69.74%**)

   (d) Apart from OneSVM and Histogram models (Table 2), the iforest model detects anomalies in high dimensions by randomly selecting the value and then randomly selecting the split value between the maximum and minimum of the selected feature value. Anomalies will require a shorter path from the root. This is one of the best models for identifying Anomalies in high dimensions. In observed model performance, iforest does a good job of identifying Attacks (**% of Identified Attacks: 63.91%**)

2. **Analysis on individual models for % Of Identified Attacks and Propensity (early detection)**
   (a) We also looked into the top three models' ability to identify the attacks and propensity of attacks (earliest to attacks) for a specific attack. We observe (Table 6 and Table 7) that iforest (P1 Subsystem) and Histogram (P2 Sub-System) identify 100% of attacks within 0 Seconds (Propensity). This is quite impressive - given the robustness of models as we observe from Figure 20 and Figure 21. However, across sub-systems and across all attacks, they do not have good performance metrics as we observe in Table 2 , Table 3 ,and Figure 22.

   (b) Next, we looked into the top three models' ability to identify the attacks and propensity of attacks (earliest to attacks) across all attacks for a specific sub-system model. We have a mixed-bag observation (Table 4 and Table 5). We expected OneSVM, Histogram, and SVM to perform better based on the performance metrics of Table 2 and Table 3. However, surprisingly MCD (P3 Sub-system) performed quite impressive for identified attacks (**% of Identified Attacks: 99.27% and Propensity : 0 Seconds**). We plotted Figure 18 to observe its ability to identify the attacks. It appears that MCD (P3 Subsystems) features are densely located and hence its performance is stellar. We can state that if a particular dataset is densely clustered in high dimensions, Mahalanbosis distance will be quite effective in identifying dispersion and anomalies. However, across sub-systems and across all attacks, MCD does not have good performance metrics as we observe in Table 2 ,and Figure 23.

   (c) We also plotted the Bottom 3 performing models by % of Identified Attacks across all attacks (Table 2 and Figure 23: Attack 1 through Attack 41 - Between 28 December 2015 and 2nd-Jan-2016).All these models (KNN, MCD, and LOF) are inferior to OneSVM. LOF (Local Outlier Factor) performs the worst as it has identified only 12% (approx.) of attacks.LOF reflects a degree of an anomaly if an observation is different from the local density of neighbors obtained from the K-Nearest Neighbors. It appears that the data is quite sparsely distributed and hence LOF could not identify the attacks appropriately.

   (d) One of the worse performing models (Table 2), the Minimum Covariance Determinant (MCD) model detects anomalies in high dimensions by computing Mahalanbosis Distance(Based on underlying features). The broader the distribution of MCD, the probability of identifying anomalies are higher. In Our case, MCD does a bad job of identifying Attacks (**% of Identified Attacks: 53.37%**)

3. **Analysis on Propensity of Attacks (early detection)**

(a) For Performance metrics in terms of Propensity of Attacks (earliest to identify), OneSVM performs the best (Please refer to Table 3) across all models. We expected it as OneSVM is quick to identify slight perturbations and detect anomalies. However, The minimum time to identify anomalies (Propensity) is **43 Seconds**, which is quite good. As we observe from (Figure 24: the Top 3 performing Models by Means of Propensity (in Seconds)), OneSVM propensity is close to zero seconds for all attacks but attacks 20 through 22 and attacks 29 through 31. We need to analyze it further.

(b) However, Performance metrics in terms of Propensity of Attacks (earliest to identify), Apart from OneSVM, other models (Please refer to table 3) perform the worst (at least 5 times worse than the OneSVM Model). With these performance propensity metrics, apart from the OneSVM model, other models may not be acceptable for production-level anomaly detection systems.

(c) As we Observe from Figure 24- Top 3 Performing Models by Mean Of PropensityOfAttack (In Seconds), the iforest model is relatively better than the Histogram model. The iForest model performance degrades through attack 26 and attacks 29 - resulting in an overall bad propensity score **251.38 Seconds**. However, for Attacks 24 through 29, Histogram performance degrades which needs to be analyzed further. It has impacted the mean of the Propensity score **451.88 Seconds**.

(d) As we Observe from Figure 25 - Bottom 3 Performing Models by Mean Of PropensityOfAttack (In Seconds), KNN, LOF, and MCD performances are quite bad. For Attacks 26 through 29, performance degrades need to be analyzed further. It has impacted the mean of the Propensity score.

(e) We observe (Table 3, Figure 24, and Figure 25) that the best performance for Propensity (In Seconds) for identifying attacks is **OneSVM: 43 Seconds**. In production systems, we should be able to identify anomalies at the earliest (less than 10 seconds). The degradation of Propensity may be attributed to below :

i. We are assuming all records are real anomalies for a given attack. However, the impact of anomaly has latency and the actual impact will be further in time. It will require domain knowledge and thorough analysis. Hence, due to our expansionary assumptions, Propensity metrics degrade.

ii. As we observe from (Figure 24 and Figure 25), there are spikes in propensity across model groups - which need to be further analyzed. and attacks 29 through 31. We need to analyze it further. This degradation has impacted the mean of the Propensity score (negatively)

(f) For all models and across sub-systems, we could not identify any attack data for Attack 37 through Attack 41 (2nd Jan 2016 between 11:17:02 and 13:13:02). We should have considered removing it from the analysis.

4. **Anomaly detection is Inversely proportional to the threshold of Standard Deviation** Anomaly detection is sensitive to the threshold of standard deviation. Lower the standard deviation, we will be able to detect Anomalies higher (% of Identified Attacks) and early (Propensity of Attacks (in Seconds)) As We Compare and contrast, Table 2 & Table 3 (for Standard deviation of 0.5) and Table 8 & Table 9 (for Standard deviation of 1.0), we observe that for Standard deviation 1, the detected Anomalies (% of Identified Attacks) are lower and (Propensity of Attacks (in Seconds)) is higher for all models across subsystems.

5. **Recommendations :**

(a) As mentioned in the assumption Section, the OneSVM model is quite compute/resource intensive and we could not train the model for Subsystems P_3 through P_6.

(b) Based on our Analysis and the above Performance metrics tables (Table 1 and Table 2), we conclude that the Overall OneSVM model performed the best in terms of % Of Identified Attacks (**90.62 %**) and (Mean) Propensity of Attacks (**43.82 Seconds**)

(c) Under this constraint and for Identifying % Of Attacks and Propensity (early detection) of Attacks, we would like to recommend the OneSVM model for Anomaly detection.

(d) We are indifferent in recommending the Histogram or iForest model for Anomaly detection as they equally performed well.

(e) Based on empirical evidence (observed) during this experimentation, we are not recommending other models such as MCD, KNN , and LOF.

5.4 Next Steps

1. Train the model across all remaining Sub-system models 3 (Especially OneSVM) to ensure that OneSVM remains the preferred and recommended model

2. Extend underlying models to other Systems such as WADI
3. Pursue Other models other to achieve better performance such as (Prophet, XGBoost). We think that Time Series based model may be able to detect anomalies as it captures the correlation of the feature's distribution as a time series.

6 References:

References

[1] Temporal Execution Behavior for Host Anomaly Detection in Programmable Logic Controllers. https://ieeexplore.ieee.org/document/8832261

[2] A Dataset to Support Research in the Design of Secure Water Treatment Systems https://www.researchgate.net/publication/305809559

[3] sklearn.neighbors.LocalOutlierFactor https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

[4] ADBench: Anomaly Detection Benchmark https://www.andrew.cmu.edu/user/yuezhao2/papers/22-preprint-adbench.pdf

[5] Handbook of Anomaly Detection with Python Outlier Detection — (4) Isolated Forest https://towardsdatascience.com/use-the-isolated-forest-with-pyod-3818eea68f08

[6] https://scikit-learn.org/stable/modules/covariance.html#robust-covariance

[7] Handbook of Anomaly Detection: With Python Outlier Detection — (6) OCSVM https://dataman-ai.medium.com/handbook-of-anomaly-detection-with-python-outlier-detection-6-ocsvm-f746dae9f450

[8] Local Outlier Factor (LOF) — Algorithm for outlier identification https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843

[9] Handbook of Anomaly Detection: with Python Outlier Detection — (2) HBOS https://medium.com/dataman-in-ai/anomaly-detection-with-histogram-based-outlier-detection-hbo-bc10ef52f23f

[10] Handbook of Anomaly Detection: With Python Outlier Detection — (8) KNN https://medium.com/dataman-in-ai/anomaly-detection-with-pyod-b523fc47db9

[11] Anomaly Detection: — (11) Anomaly-Score https://scikit-learn.org/stable/modules/outlier_detection.html

[12] Novelty and Outlier Detection https://scikit-learn.org/stable/modules/outlier_detection.html

[13] IsolationForest Example https://scikit-learn.org/stable/auto_examples/ensemble/plot_isolation_forest.html

[14] Robust covariance estimation and Mahalanobis distances relevance https://scikit-learn.org/stable/auto_examples/covariance/plot_mahalanobis_distances.html#sphx-glr-auto-examples-covariance-plot-mahalanobis-distances-py

[15] Covariance Estimation https://scikit-learn.org/stable/modules/covariance.html#robust-covariance

[16] Comparing anomaly detection algorithms for outlier detection on toy datasets https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_anomaly_comparison.html