Electric Energy Consumption Data Anomaly-Detection: Using Hidden Markov Models to Detect Anomalies in Industrial Control System Data

By Group 7

Abstract: This report details the methodology for anomaly-based detection for stream data using supervisory control systems. To expand, in order to analyze energy consumption time series data for anomalous entries, this project focuses on conducting unsupervised intrusion detection using time series analysis and forecasting. Feature engineering is enriched through conducting a Principle Component Analysis (PCA). Furthermore, Hidden Markov Models with different states are trained and later tested to determine the best performing one. The best model is used to determine a threshold for normal behavior. This threshold is used to differentiate between normal and anomalous observations which can lead to identifying irregular activity and potential threats.

Table of Contents

Table of Figures

1 Introduction

As technology develops, automation has become an essential part of operation success for critical infrastructures. Without automation, critical infrastructures such as electric power grids, public water utilities, and smart transportation networks would be susceptible to higher costs, damages, and dangers [3]. While automation is essential for the new era of critical asset management, it exposes these assets to a higher risk of attack. Automation leaves critical infrastructure vulnerable to complex cyber risks because it only takes one failed algorithm or internet leak to cause a cascade of damages. To elaborate, Michael Parrant, Director & Cyber Insurance Practice Leader at Aon, notes "In many large retail organizations 'digital control towers' have been established;…bring down the tower and an entire industry can be brought to its knees" [1].

Naturally, because automation creates a new form of vulnerability in critical infrastructure, there have been many unfortunate incidents caused by the overreliance on automation. To illustrate, the Cybersecurity and Infrastructure Security Agency (CISA) reported that in 2015, Ukraine experienced an abnormal number of power outages due to "remote cyber intrusions." The attackers managed to access the electricity grid systems by obtaining "legitimate credentials" and using administration tools at the "operating system level" or "remote industrial control system (ICS) client software via virtual private networks (VPNs)." These vulnerabilities were exploitable because ICS automation increased the attack surface area. Because of these massive power outages, Ukraine faced disruptions to essential services, severe economic losses due to business closures, and political unrest as citizens questioned their safety and the legitimacy of their government [2].

With the potential repercussions of a weak security system being astronomic, cybersecurity is becoming one of the fastest-growing industries as more organizations recognize the importance of protecting their information [3]. Logically, mitigating the risk of cyberattacks has become a lucrative and essential endeavor, not only to reduce damage to critical assets but to protect the safety of citizens as well. In order to analyze automated control processes like ICSs for anomalies, there are many intrusion detection methods that cyber intelligence can use. It is best to implement a wide range of these methods at different layers of the security system to ensure a robust defense. As such, this report explores an intrusion detection method based on anomaly detection.

1.1 The problem scope of this project

This project will analyze energy consumption time series data for anomalous entries. To expand, in order to study anomaly-based detection methods for stream data and supervisory control systems, this project focuses on conducting unsupervised intrusion detection using time series analysis and forecasting. Feature scaling and engineering will be enriched through conducting a Principal Component Analysis (PCA) as this technique provides deeper insights into response variables. Furthermore, this project aims to explore the capabilities of Hidden Markov Models (HMMs) by testing models through various states. This process aims to enhance the anomaly detection process. However, these goals come with obstacles. In anomaly detection, there is often a need to find an optimal way to clean dirty data; overcome a lack of existing data; balance precision and recall. It is essential to resolve these concerns as a high-quality anomaly-detection system can only be obtained by minimizing the room for inaccuracies in a real-life scenario.

1.2 Technical Background

1.2.1 Supervisory Control

Supervisory control systems are a subcategory of Industrial Control Systems (ICSs). They are a critical success in the industrial automation of cyber-physical systems (CPS). Specifically, these systems monitor and manage physical processes using a variety of controllers at remote locations. Because these systems are automated, there is a lower demand for human operation which increases the affordability, quality, and efficiency of critical asset supervision [3].

1.2.2 Anomaly detection in time series

Outliers and anomalies are terms that are used interchangeably when describing observations that deviate from the normal so much that it seems like it came from an external operator. Being able to separate typical behavior from these atypical behaviors is essential in cybersecurity applications. For example, in time-series data, information is ordered chronologically. Because of this characteristic, time series analysis is an important cybersecurity application. The ability to observe behaviors over some time allows systems to detect suspicious activity that may be related to security threats or system bugs [4].

1.2.3 Hidden Markov Models (HMMs)

Hidden Markov Models are a type of statistical model that attempts to model the likelihood of a sequence of observable events occurring along with correlated hidden states. In terms of working with time series data, Hidden Markov Models work well because they help uncover the underlying behaviors that generate the resulting time series data [4].

1.2.4 Feature Scaling and Engineering

Feature scaling is a data preprocessing technique commonly used in machine learning to normalize the range of features in a dataset. The goal of feature scaling is to ensure that all features have the same scale, typically between 0 and 1. This normalization process helps machine learning algorithms converge faster and prevents features with large scales from dominating those with smaller scales during the training process [6].

There are two main feature scaling techniques:

The first is normalization. Normalization simply adjusts the features of the data to be on a similar scale. It compresses the scale, squeezing it into a smaller interval, typically scaling it to a range of 0 to 1. This means that normalization does not affect the shape of the distribution or the outliers of a dataset. In fact, normalization is most useful when there are no outliers in a dataset. The mathematical formula for calculating a new point after normalization is as follows: X_new = (X - X_min)/(X_max - X_min) [6].

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Figure 1.1 of formula for normalization

The second is standardization. Standardization aims to fit the shape of the data to a normal curve so they have a mean of σ and a standard deviation of 1. In most cases, outliers can be rounded off unless there are too many. Standardization does not get affected by outliers as much as normalization does since there is no predefined range for standardization. The mathematical formula for calculating a new point after standardization is as follows: X_new = (X - mean)/Std [6].

$$x' = \frac{x - \bar{x}}{\sigma}$$

Figure 1.2 of formula for standardization

Feature engineering, on the other hand, involves selecting a subset of response variables that are most suitable for training multivariate HMMs. Within the realm of feature engineering, there is a powerful technique called “Principal Component Analysis” (PCA), which is used to identify the most useful features from a dataset [6]. Essentially, PCA involves a linear transformation that condenses a dataset with multiple variables (such as responses and samples) into a reduced set of variables termed principal components [7]. This method is used in this project to determine the ideal variables to train our HMM with.

2 Methodology

In addressing the challenge of unsupervised intrusion detection, our methodology uses a four-step approach. First is feature scaling, where input variables are normalized to ensure a consistent scale and prevent bias towards certain features. Second is feature engineering, where the most meaningful information is extracted from the dataset, enhancing anomaly detection capabilities. Third is Hidden Markov Model (HMM) training and testing, where the HMMs are trained to recognize patterns, then the most ideal model is determined through observations of key metrics. Lastly, is anomaly detection, where the optimal model is used on our given dataset to detect potential intrusions or anomalies. Each of these steps plays a pivotal role in the development of an effective solution for detecting anomalies in stream data from supervisory control systems.

2.1 Feature Scaling

The first and most foundational step of our methodology was feature scaling. Feature scaling is a crucial step in data preprocessing for machine learning algorithms, especially for HMMs which rely heavily on the distribution of the data features. Scaling ensures that all features have a

similar impact on the model's learning, preventing any single feature from dominating the learning algorithm [8]. This is important because real-world datasets often have significant variations in feature magnitude, which may pose challenges for machine learning algorithms. By transforming the features onto the same scale, the impact of extreme values is reduced which prevents bias and helps the machine learning algorithm interpret data more effectively [8]. During this stage, missing values and non-numeric values were two challenges that we encountered which prevented scaling. Therefore, before scaling our features, we used linear interpolation to handle the missing data and then converted all values to their numeric equivalents.

Between the two feature scaling methods, standardization and normalization, we chose to apply standardization to our dataset, because it simplifies the anomaly detection process by regulating the data. In HMM-based anomaly detection, anomalies are often defined as points lying outside three standard deviations (99.7%) from the mean. This makes standardization the ideal feature scaling technique for anomaly detection because of its ability to maintain the original data distribution while centering it around 0 and scaling it to have a standard deviation of 1. This makes it easy to determine how many standard deviations away from the mean a particular data point lies [10]. Furthermore, standardization generally fits data to a Gaussian distribution, which is a distribution HMMs work well with.

2.2 Feature Engineering

Feature engineering involves selecting a subset of response variables that are most suitable for training multivariate HMMs. We used PCA to determine the most suitable response variables to

work with. Specifically, we computed the principal components of the original dataset, plotted the results, and then interpreted them. The first principal component (PC1) showed an impact on around 53% of the variation, demonstrated in the bar plot below.

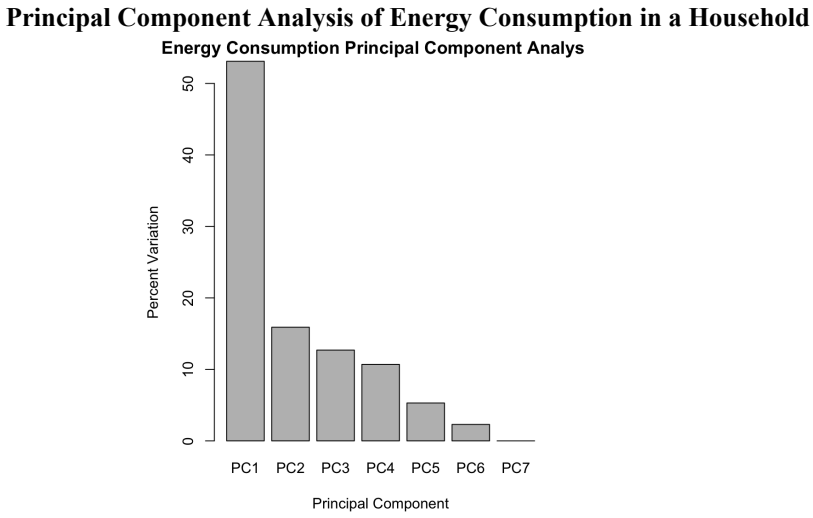

Figure 2.1 of principal component analysis results

Further analysis revealed that "Global_intensity" and "Global_active_power" were the most significant variables that affected PC1. We opted for these two features since they had the two highest loading scores, i.e. they affected PC1 the most. We decided on using only two because the third variable had less impact compared to the first two and to reduce model complexity.

```
Global_intensity   Global_active_power        Sub_metering_1 Global_reactive_power
       0.5519617             0.5415710             0.4182490             0.2503012
  Sub_metering_2               Voltage        Sub_metering_3
       0.2405504             0.2368147             0.2247455
```

Figure 2.2 of principal component loading scores

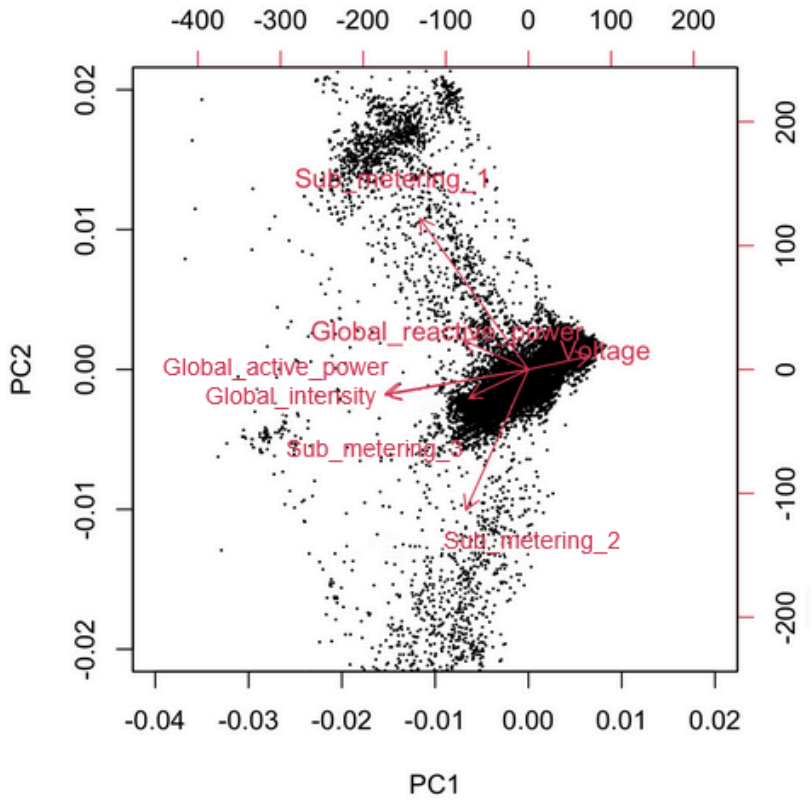

Figure 2.3 of loading scores affecting the direction of observations

2.3 HMM Training and Testing

After identifying the optimal variables to use, HMM training and testing began. Our HMM training involved partitioning the data into train and test sets, training multivariate HMM models with varying numbers of states, and evaluating their performance using metrics such as log-likelihood and Bayesian Information Criterion (BIC).

2.3.1 Discretization

We proceeded with a continuous distribution of the variables as discretization can cause crucial information and trends to be lost [9]. As a result it could impact the models performance making it harder to accurately capture underlying patterns in the data.

2.3.2 Partitioning Data

We set our time window to be on Saturdays from 5:30pm to 7:30pm as many people will be home to consume energy. Having more people consume energy gives us more data to work with. This allows more regular observations and reduces the likelihood of having zeros. Additionally, it is more likely for power to be consumed on a weekday due to a holiday than power to not be consumed on a weekend. These considerations led us to choose the final time window. Moreover, the dataset was partitioned into a training set containing the first three years of data and a test set that contained the remaining data from the fourth year. This ensures that the model is trained on a historical dataset and then tested on a different, unseen dataset. Setting 75% of the data for training helps the model learn the patterns of the data while preventing it from overfitting. Additionally, having at least 25% of data for model testing ensures a more accurate result.

2.3.3 Model Training

Multivariate HMM models are trained on the training data using various numbers of states. The number of states determines the complexity of the model and its ability to capture the patterns in the data. Due to a lack of computational power, we implemented a strategy to streamline the training process while maintaining high accuracy levels. Rather than training models for every single state ranging from 4 to 20, we chose to train models at intervals of 4 states, starting from state 4 and progressing to 8, 12, and so forth. This approach substantially decreased the training time without compromising accuracy.

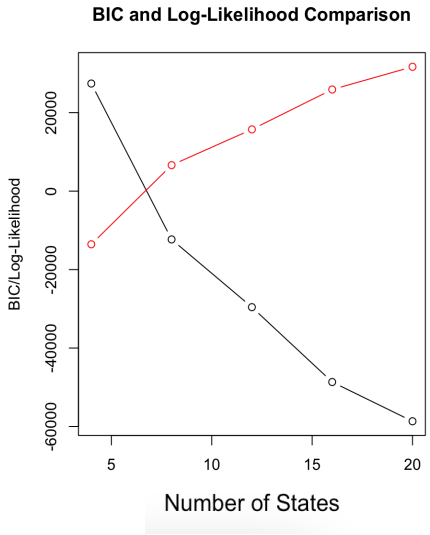

Figure 2.4 of training strategy results

2.3.4 Model Evaluation

We evaluated the models' performances by considering two important statistical metrics, log-likelihood and Bayesian Information Criterion (BIC), to determine the best model. Our goal was to identify a model that had a high log-likelihood while also maintaining a low BIC. We found that the elbow of the graph occurs around 12 states but would need more info to find exactly where. Testing from 11-14 states would show that 13 states would yield the best results.

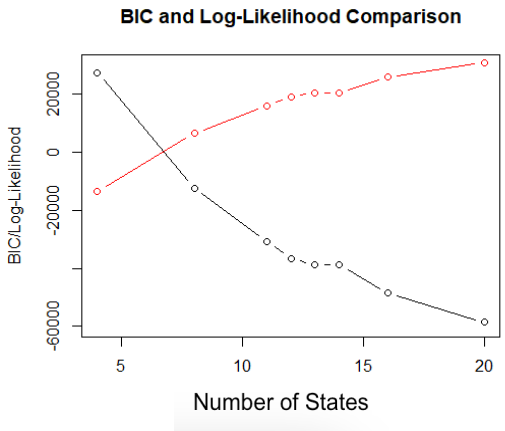

Figure 2.5 showing the plot BIC and log likelihoods of states 11-14 compared to the original results

states	TrainLogLiks	TestLogLiks
11	0.8376378	0.9288017
12	0.9808097	1.0468150
13	1.1376420	1.1359313
14	1.1852411	1.2414303

Figure 2.6 showing the normalized train and test log-likelihoods of HMMs with states 11-14

After taking the normalized log-likelihoods of the training and testing, we can see that state 13 has very close values, meaning it's neither overfitted nor underfitted to the training data.

Log-likelihood is a common way to measure how well a model fits an observed dataset. It is a measure that tells you how likely it is that a model's predictions match the actual outcomes observed in the data. In general, a higher log-likelihood value indicates that a model provides a better explanation or fit to the observed data, meaning it's more likely that the model's predictions match reality. While the log-likelihood of a single model by itself is usually meaningless, it helps immensely when comparing the performance between two or more models.

Additionally, BIC is a criterion for model selection among a finite set of models. It balances model fit with model complexity by penalizing complex models to avoid overfitting. Model fit is typically measured with the likelihood function, which quantifies how well the model fits the actual observed data. Furthermore, model complexity is typically measured by the number of parameters a model has. Complex models tend to have more parameters which increases the probability of the model fitting noise instead of the true underlying patterns, a common

machine-learning phenomenon known as overfitting. In general, models with lower BIC values are preferred over models with higher BIC values.

2.4 Anomaly Detection

Anomaly detection is a method used to measure how greatly a data sequence from a new data stream deviates from the trained model's learned behavior [5]. This deviation shows us how anomalous the data sequence is. For this process, we need a threshold to determine the difference between the normal observations and the anomalous observations [5]. To find our threshold, we first computed the log-likelihood of the training data after finding our best HMM model: the model with 13 states. We then split our test data, which was our final year of the entire dataset, into ten subsets. Since our final year had slightly fewer weeks than a normal year, our 10th subset was slightly smaller. For each of these ten subsets, we computed their log-likelihood using our best-performing HMM model. Most of the models yield greater (worse) values than the trained log likelihood as we ensured that 13 states was the best number of states. However, we did run into a few that were smaller in value such as subset 1, subset 2, and subset 3. This could have been due to external factors such as irregular behavior causing corrupt values/outliers or significant variation of the points in the dataset in that time period. **Upon review, it was determined that the reason a few of the test log likelihoods appeared to be better was because the model performance could have been enhanced using a method such as discretization (ex. smoothing out min and max points, simplifying noise, etc).*

	Subset.Number	Log.Likelihood.Value
1	1	-0.7222541
2	2	1.2890331
3	3	0.8405713
4	4	2.1019781
5	5	1.3338938
6	6	2.2746535
7	7	2.4871108
8	8	1.9384808
9	9	1.2053198
10	10	-0.9178398

Figure 2.7 of test subsets and their log-likelihood values

Then, we took each of these values and subtracted them from the trained log-likelihood value of 1.1376422 to determine how far away each of these subsets deviates from the trained loglike hood. The results of the deviation were the following (all in absolute value):

	Subset.Number	Difference
1	1	1.85989639
2	2	0.15139087
3	3	0.29707091
4	4	0.96433583
5	5	0.19625159
6	6	1.13701127
7	7	1.34946856
8	8	0.80083857
9	9	0.06767758
10	10	2.05548202

Figure 2.8 showing the test subsets and their score differences

We determined our threshold to be subset 7, a value of 2.4871108 since based off of our results, it had the worst log likelihood and it is the furthest away from the trained loglike hood by a magnitude of 1.34946856 (subset 1 and subset 10 appear to be further but that is only because they are negative). This means that any data point between this number and the train

log-likelihood is considered normal behavior and anything above this range is considered to be anomalous.

3 Major Problems Over the Project

Over the course of this project, we encountered issues relating to the application of concepts to the code and time management. For example, when we were trying to find the best HMM model by getting the parameters from the training dataset, we got an error that our parameter lengths did not match. We spent 12 days troubleshooting by referencing several R guidebooks, searching through online forums, and asking the TA for help. After a thorough discussion, we found the cause to be the structure of the dataset. Since our variables are continuous, there are occasions where the test dataset has points that were unaccounted for. We solved this problem by removing those few points.

Time management was a significant issue during the course of this project. Since each part of the code depended on the previous, each part had to be finished promptly. Although we set an internal timeline with due dates to finish everything before the class due date, there were occasions when we could not figure out the issue, thus missing our internal deadlines. This delayed our progress but because we gave ourselves extra time initially, we were able to get our work done on time.

4 Lessons Learned

During the course of this project, we learned how useful PCA is and the significance of finding the best HMM model to determine the threshold for anomalous behavior. PCA is a vital tool that can be used to analyze large datasets with numerous variables as it allows us to see how the data

in the dataset is dispersed, how much variance there is for each variable, and how much information can be represented for a variable in its principal component. The best-performing HMM can be applied to compare the test datasets to the train datasets to see how far the new data streams deviate from the train log-likelihood and to determine the greatest deviation as the threshold for anomalous behavior.

We also gained new practical skills. These include resourcefulness and making the most of available resources such as guidebooks, online discussions, explanation videos, and course material. Furthermore, troubleshooting was an important skill we developed as we had to direct our efforts from different angles to figure out the root of the problem. By using this technique, we were able to reach a conclusion based on our results and observations as well as uncover underlying issues that we were previously unaware of.

5 Conclusion

Industrial control systems play an essential role in cybersecurity, ensuring smooth management of critical assets. This report explored the role of HMMs in anomaly detection regarding critical asset data. To enhance the model development process, feature engineering and scaling processes were conducted with standardization and PCA. Based on our initial analysis, the two variables that strongly correlated with data behavior were “Global_intensitity” and “Global_active_power” based on the PCA. Using these variables with the HMM, it was determined that 13 states were the optimal number of states to model with. When comparing the trained loglike hood to each of the 10 subsets, we determined the threshold to be an absolute value of 2.4871108 from subset 7. Behavior beyond this value would be considered anomalous.

Bibliography

[1]

M. Parrant, “The Impact of Automation on Cyber Risk | Cyber,” *Aon Insights*, Dec. 04, 2018. https://aoninsights.com.au/impact-automation-cyber-risk/#continue (accessed Mar. 31, 2024).

[2]

Cybersecurity & Infrastructure Security Agency, “Cyber-Attack Against Ukrainian Critical Infrastructure,” *Cybersecurity and Infrastructure Security Agency CISA*, Jul. 20, 2021. https://www.cisa.gov/news-events/ics-alerts/ir-alert-h-16-056-01 (accessed Mar. 31, 2024).

[3]

U. Glasser, F. Movafagh, and A. Shahir, *Cybersecurity Section 1 [Lecture Slides] CMPT 318*. Simon Fraser University, 2024.

[4]

U. Glasser, F. Movafagh, and A. Shahir, *Cybersecurity Analytics: Probabilistic Modeling Section 3 [Lecture Slides] CMPT 318*. Simon Fraser University, 2024.

[5]

U. Glasser, F. Movafagh, and A. Shahir, *Anomaly Detection Section 4 [Lecture Slides] CMPT 318*. Simon Fraser University, 2024.

[6]

A. Shahir, *R Tutorials CMPT 318*, Part 1, Part 2, Part 3, Part 4, Part 5. Simon Fraser University, 2024.

[7]

Stat Quest, “StatQuest: PCA main ideas in only 5 minutes!!!,” *www.youtube.com*, Dec. 07, 2017. https://www.youtube.com/watch?v=HMOI_lkzW08&feature=youtu.be (accessed Apr. 06, 2024).

[8]

Shivani, “What is Feature Scaling and Why Does Machine Learning Need It?,” *Medium*, Nov. 16, 2023. https://medium.com/@shivanipickl/what-is-feature-scaling-and-why-does-machine-learning-need-it-104eedebb1c9

[9]

S. Galli, “Data discretization in machine learning,” *Train in Data*, Jul. 04, 2022. https://www.blog.trainindata.com/data-discretization-in-machine-learning/#:~:text=In%20discretization%2C%20we%20convert%20continuous%20variables%20into%20discrete (accessed Apr. 07, 2024).

[10]

Zach Bobbitt, “Standardization vs. Normalization: What’s the Difference?,” Statology, Jun. 09, 2021. https://www.statology.org/standardization-vs-normalization/