**Anomaly Detection Term Project**

**Abstract**

Anomaly detection presents itself as a capable method for providing effective cybersecurity measures against certain attacks. This report details the process and approach to detecting anomalies within multiple datasets containing user electricity consumption data. Employed techniques include Principal Component Analysis and Multivariate Hidden Markov Model Training for prepping the data, and Log Likelihood and Moving Average analysis for detecting anomalies within prepared data. We then explore the subject of Reinforcement Learning and how it can be applied to provide anomaly and intrusion detection. Finally, we discuss our learnings and concluding thoughts.

**GROUP 26**

Vinay Lalwani #301319204

Dan Amarasinghe #301335606

Rakim Zubair #301340433

*CMPT 318 - DECEMBER 2020*

*SIMON FRASER UNIVERSITY*

**Table of Contents**

**i. Introduction** --------------------------------------------------------------------------------------------------------- 3

**ii. Training and Testing HMMs -**------------------------------------------------------------------------------ 4 - 12

1. Principal Component Analysis ------------------------------------------------------------------------- 4 - 9
2. Training the multivariate HMMs ------------------------------------------------------------------- 10 - 11
3. Testing and verifying HMMs ------------------------------------------------------------------------------ 12

**iii. Anomaly Detection -**-------------------------------------------------------------------------------------------- 13 - 28

1. Log-likelihood method ------------------------------------------------------------------------------------- 13
2. Moving average method ------------------------------------------------------------------------------ 14 - 28

**iv. Technical Essay** --------------------------------------------------------------------------------------------- 29 - 31

**v. Contributions** ------------------------------------------------------------------------------------------------ 32 - 33

**vi. Conclusion ---**------------------------------------------------------------------------------------------------------ 33

**vii. References** -------------------------------------------------------------------------------------------------------- 34

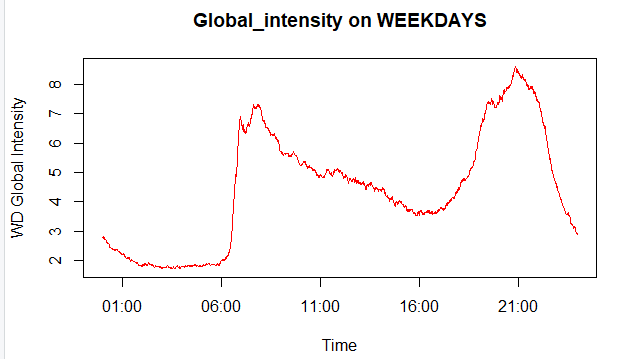
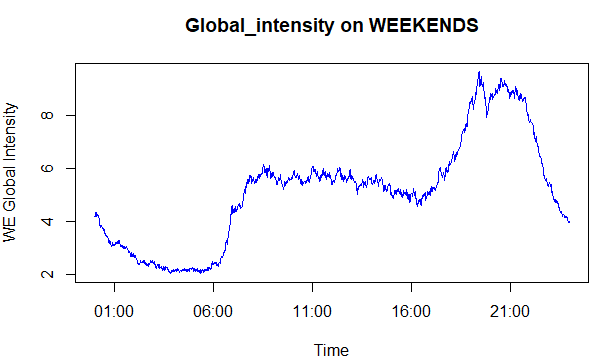
**i. Introduction**

The basis of this project was to use a dataset with values for particular readings and use various data analysis tools on it. It required the utilization of semi-supervised anomaly detection techniques to precisely detect anomalies in the provided dataset. The project report is divided into three main components: HMM (training and testing), Anomaly detection and Technical Essay. The HMM Training allowed for the creation of a suitable model which could be used to test the rest of the data. The second component to this assignment was the Anomaly Detection. In this component, the resulting model from the previous part was used to calculate the log-likelihood of the Anomalous datasets. Furthermore, the moving average method was employed to see how various thresholds affected the number of anomalies detected. The final component was the Technical Essay, in which a method of machine learning known as Reinforcement Learning was explored alongside its basic principles and applications in intrusion detection. It is then followed by a conclusion, teammate contribution and list of references used. The teammate contribution also contains a personal report outlining work done individually and as a team, as well as how the experience of doing this project was holistically.The work was distributed amongst the teammates with consideration of other courses of the teammates, with the tasks being distributed fairly so no teammate is forced to work more than necessary. This project explores many concepts and functions in the R language, which allowed us to experience the fields of Cybersecurity and Data Analysis at a somewhat basic level.

**ii. Part 1 - Training and Testing HMMs**

**a. Principal Component Analysis**

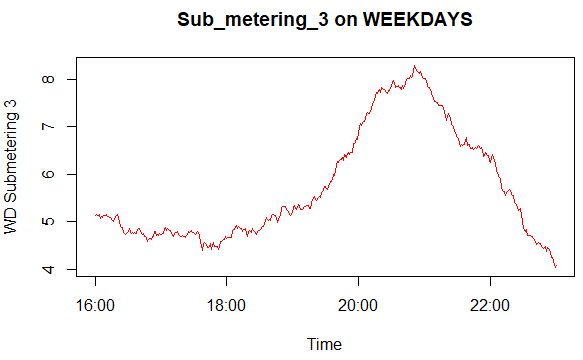
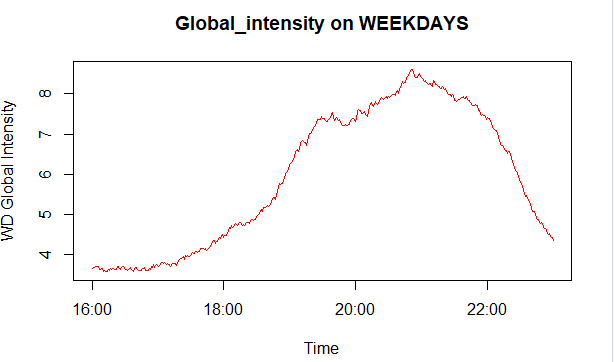
In order to choose the most suitable response variables for the sake of this project, principal component analysis was performed on the provided dataset. Similar to previous assignments, the first step was to determine time windows which best represented the dataset. This was done to avoid performing a PCA on the dataset as a whole, but rather only on the data that was relevant to the time windows used in the project.Analysis of the data showed that weekdays and weekends had differing energy consumption patterns, particularly in the early mornings.

****

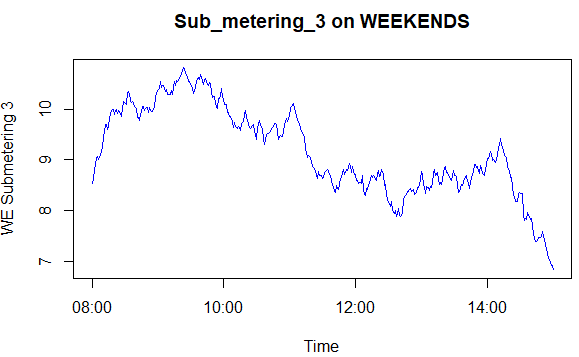
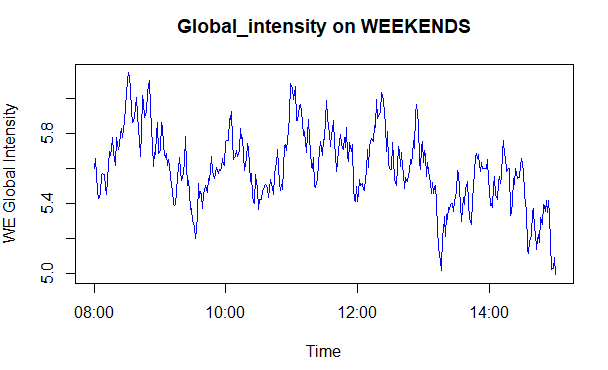
For this reason, all anomaly detection methods were applied separately to the weekend and weekday datasets**.** The chosen time windows were weekdays from 4:00pm to 11:00pm, and weekends from 8:00am to 3:00pm. These windows showed a clearly recognizable energy consumption pattern that could be easily associated with real world activities. Another one of the major design choices that were made was to omit all NA values in the dataset. Through the use of the na.omit() function, all rows that had even a single NA value present in a column were removed. Of course, this meant that some data would be lost, but for the specified time windows the difference was negligible. After omitting the NA values, the weekend dataset lost 848 out of 324591 (0.003%) of its entries, whereas the weekend dataset lost 1335 out of 130089 (0.01%) of its entries. An alternative could have been to use the na.approx() function, which replaces all NA values with approximates using the values that come before and after. These new values are not actual recorded data, but something that is manually being entered into the dataset, thus taking away from its integrity.

Below are some graphs from previous assignments which applied the same time windows:

**WEEKDAYS, 4:00pm to 11:00pm**

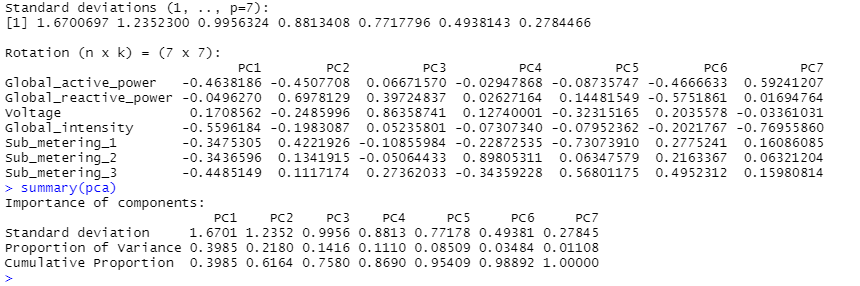


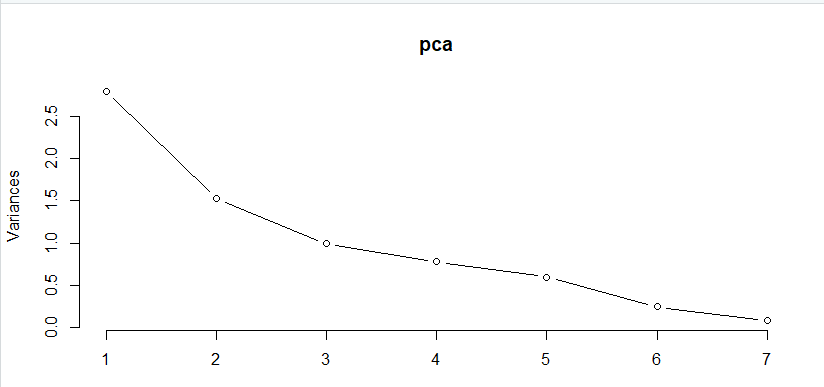
**WEEKENDS, 8:00am to 3:00pm**



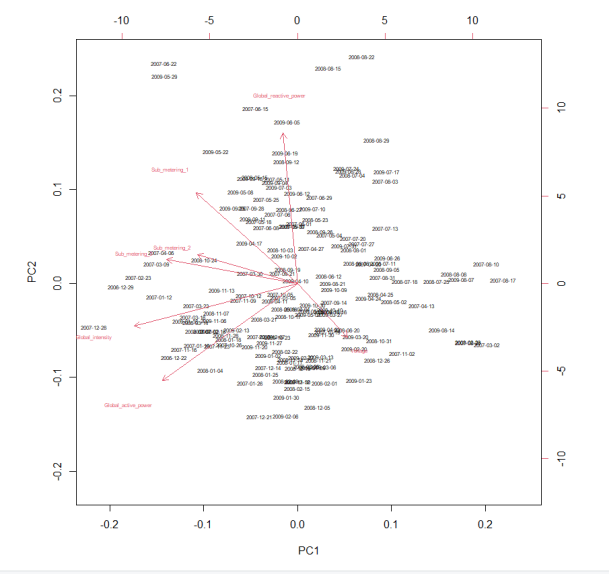
These graphs show a clear energy consumption pattern, and show how the responses Global\_intensity and Sub\_metering\_3 could be correlated with each other. These time windows demonstrate a clear increase, somewhat of a plateau, and a decrease in energy consumption, making them fitting time windows for analysis. These changes the pattern can be explained through routine activities, with examples like using electronics such as TVs and computers, turning on the heating in colder weather, etc.

From the dataset, each week was considered a unique sample which resulted in 155 samples to work with. To obtain a single value for each sample, the average value was calculated for each response variable over the whole time window for each week in the dataset. This was done using the ‘xts’ library, through the use of the endpoints() and period.apply() functions. The resulting ~155 values (each corresponding to a certain week in the data) were used to perform PCA using the prcomp() function. For weekdays from 4:00 to 11:00pm, this resulted in the following data:



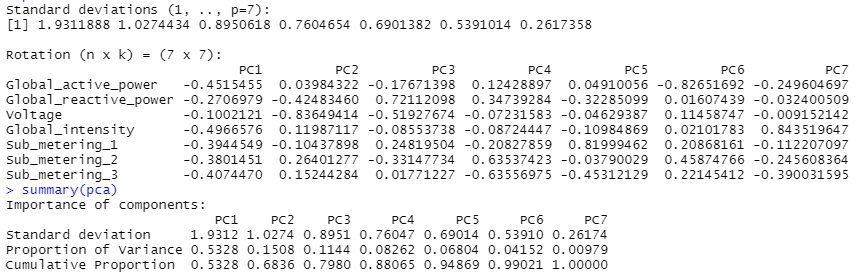


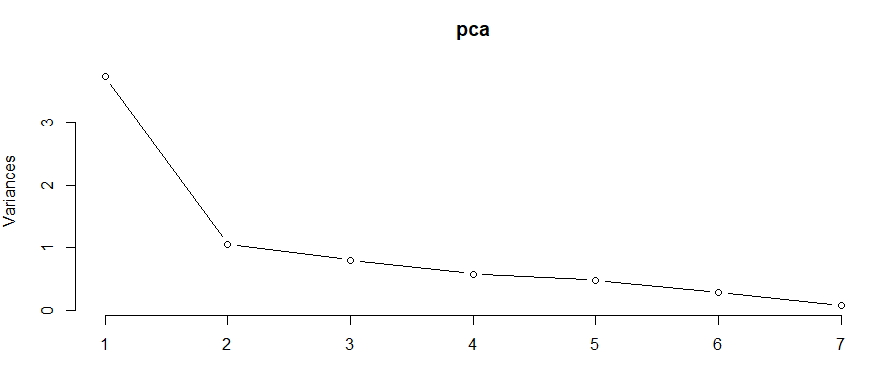
*This graph highlights the proportion of variability each PC component contributes.*



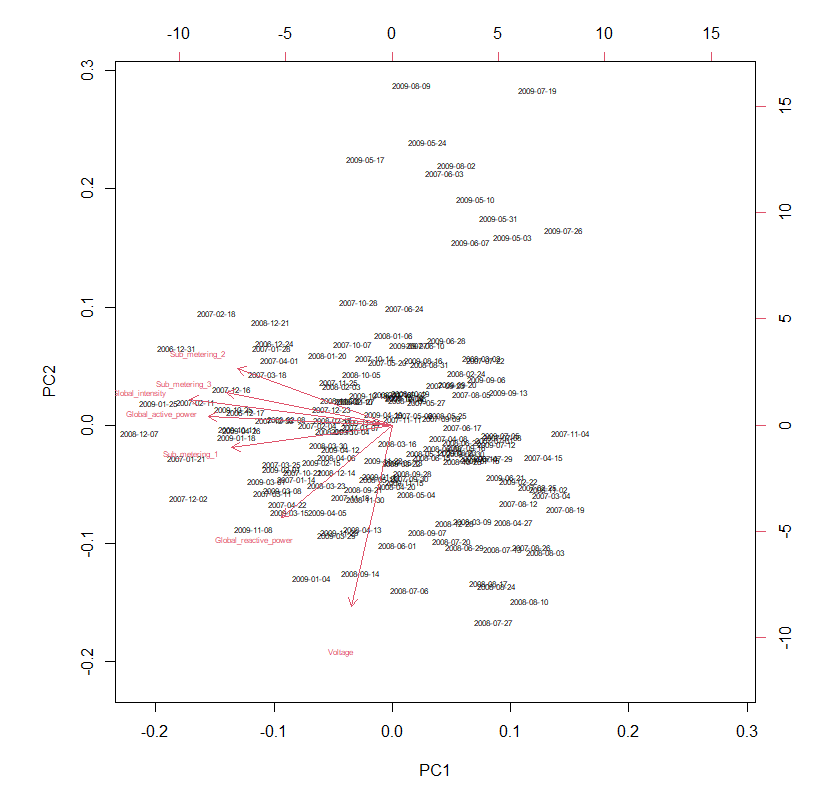
*Biplot of Weekday Data*

Applying the same method to weekends from 8:00am to 3:00pm, the results are:





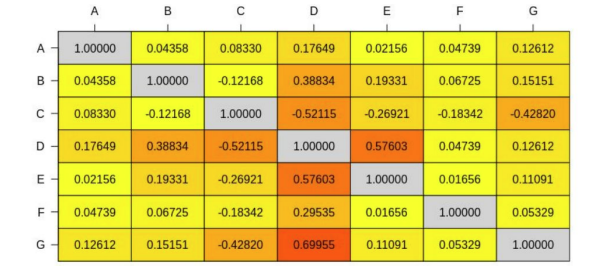
*This graph highlights the proportion of variability each PC component contributes.*



*Biplot of Weekend Data*

Averaging the two outputs (weekdays and weekends) shows that PC1 was responsible for ~45% of the variance in the dataset, with the next closest component being PC2 with ~18%. PC1 was the clear frontrunner, and was analyzed to find the most influential variables within the principal component. This was done by getting the absolute value of each response variable, putting them all in a vector and sorting them from highest to lowest. For both datasets, the sorted vector showed that the top 4 variables were (in descending order): Global\_intensity, Global\_active\_power, and Sub\_metering\_3, and Sub\_metering\_1.

In the last assignment, features were chosen based on the correlation matrix that was calculated in assignment 1. The goal was to choose a ‘main’ feature (denoted Fm), and at least two other variables that had a high correlation coefficient with Fm (but not with each other). Interestingly, it was obvious even in past assignments that Global\_intensity (D) was the most important response variable as it had the highest correlation with all the other features. The other two variables that were selected in assignment 3 were Sub\_metering\_1 (E) and Sub\_metering\_3 (G). These choices were not too far off from the results of the PCA, which favored Global\_active\_power (A) over Sub\_metering\_1 (E).

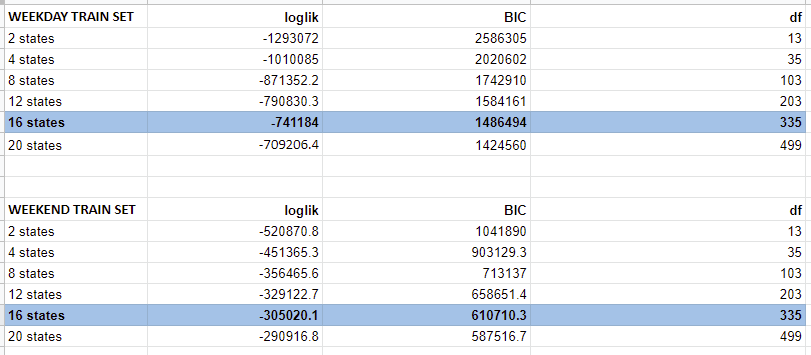


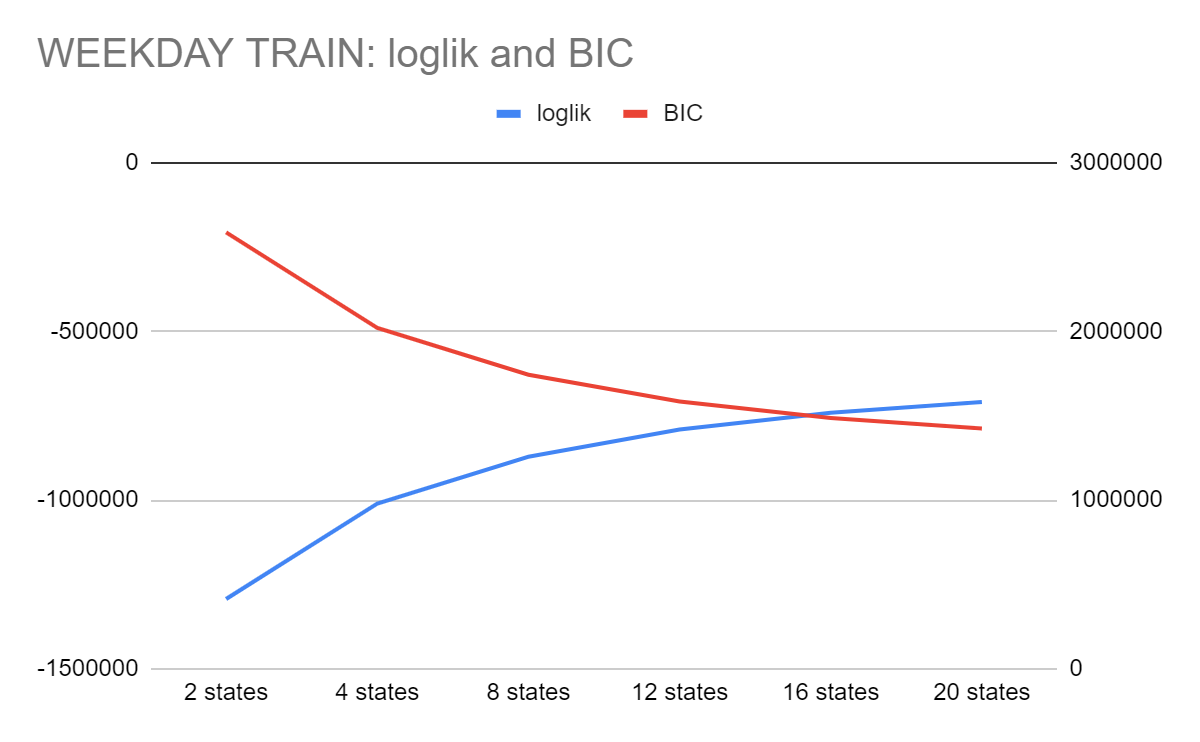
The principal component analysis brought us to the conclusion that the three most important response variables in the data were Global\_intensity, Global\_active\_power, and Sub\_metering\_3.

**b. Training the multivariate HMM**

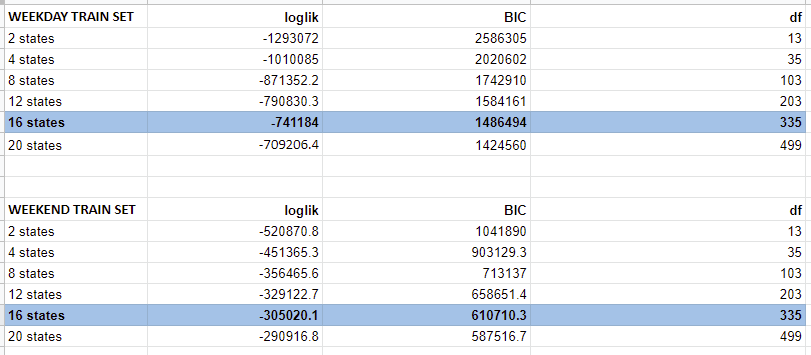
The next step was to train the multivariate HMM based on the results of the principal component analysis. Here, the goal was to end up with a suitable probabilistic model that would represent ‘normal’ system behavior. Much like the last assignment, the data was split into a train set (3 years, 2006 - 2008) and a test set (1 year, 2009). Since ‘depmix’ requires some level of random number generation, the seed was set to ‘1’ in order to get consistent results over multiple runs of the program. Next, the model was trained with varying numbers of states, in order to find a model that was neither underfitted nor overfitted on the data but a good balance of the two. The ‘gaussian’ family was used for Global\_intensity and Global\_active\_since they are continuous data. However, for the response ‘Sub\_metering\_3’ which deals with counts of watt-hours (discrete data), the ‘poisson’ family was used. Once again, this was done separately for the weekday and weekend time windows, and the results were as follows:

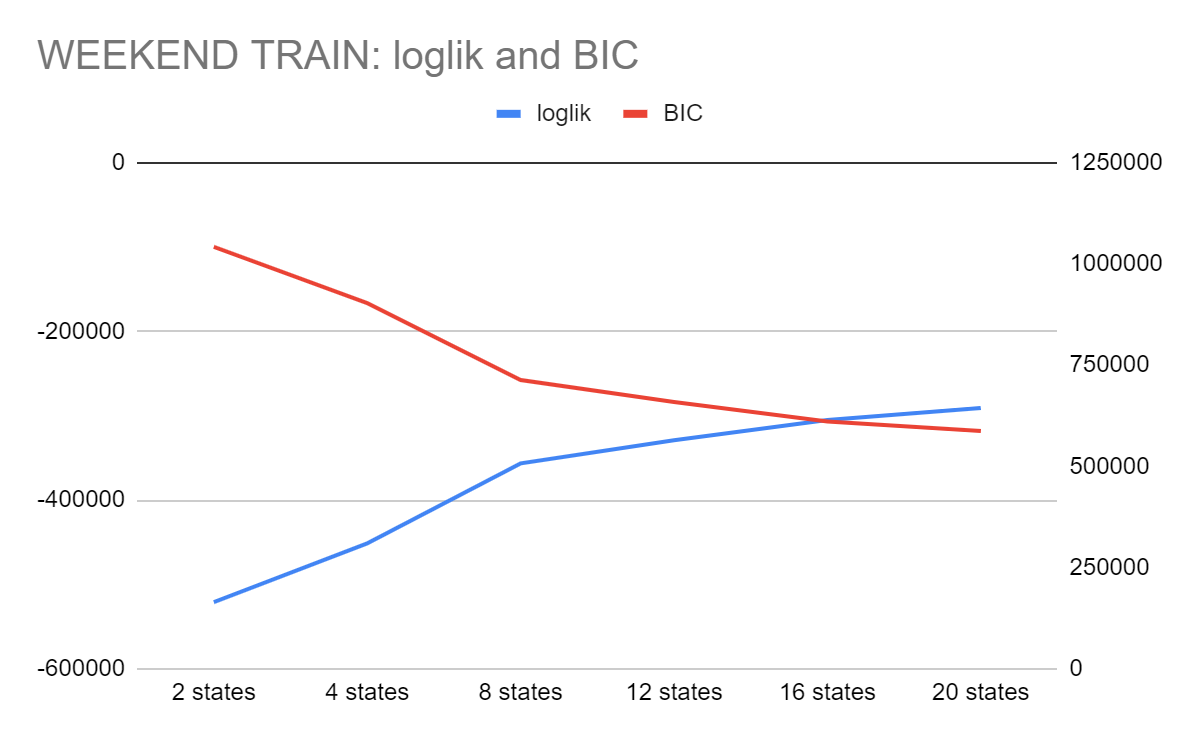
**WEEKDAYS (4:00pm to 11:00pm)**

****

****

**WEEKENDS (8:00am to 3:00pm)**

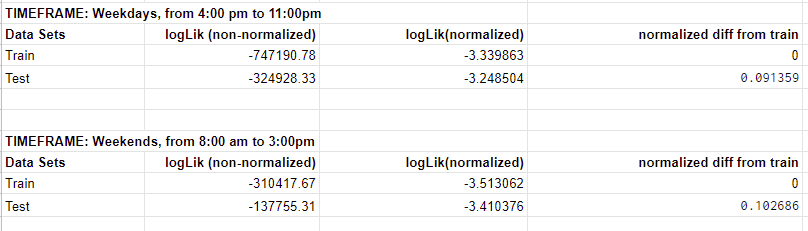
****

****

From these results, it seems that both the weekend and weekday models agree on 16 states being the best choice for the model. At this number of states, the log-likelihood and BIC values intersected, meaning that it’s where the model struck the best balance of overfitting and underfitting on the data. Of course, that is not to say that either value was ideal, but the whole point of anomaly detection is that there are tradeoffs that need to be made.

**c. Testing and verifying HMMs**

After obtaining the trained model, the next step was to verify its accuracy by feeding in the test data set and comparing the log-likelihood. A new depmix model was set up the same way as the trained model, meaning it had the same number of states (16), same response variables, same time windows, etc. The getpars() function was used to obtain the parameters from the trained model, and set these parameters on the test model. The forwardbackward() function was called on the resulting test model to obtain the log-likelihood of the test sequence being generated. This is an inference algorithm that aims to solve the third problem of HMMs, dealing with unknown parameters. It took in the parameters from the trained model, and calculated the likelihood of the test sequence being produced from those parameters. Since the train set was larger than the test set, the log-likelihoods were normalized by dividing them by the length of each data (in this case, the number of rows).

****

The normalized difference from train is the absolute value of the train log-likelihood minus the absolute value of the log-likelihood obtained from forwardbackward(). The results show that there is a fairly small difference between these 2 values, and from this the following deductions can be made:

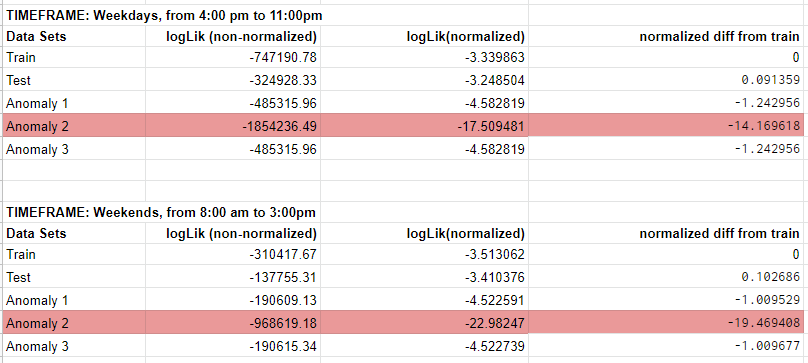
1. The trained model does a good job of representing ‘normal’ system behavior in the data
2. Assuming the above is true, then there are no anomalies in the test dataset (it matches expected system behavior)

With this, the trained model was verified and could now be used to detect any anomalies or patterns that deviate from the expected behavior.

**iii. Part 3 - Anomaly Detection**

**a. Log-likelihood method**

Using the same method as before, the 3 anomaly datasets were fed to an instance of the trained model. As learned in class, if the log-likelihood differed greatly from that of the trained model, then there are some degree of anomalies present in the dataset. The goal here was to test each provided dataset separately, and analyze the varying degrees of anomalies present in each dataset. The following results were obtained:

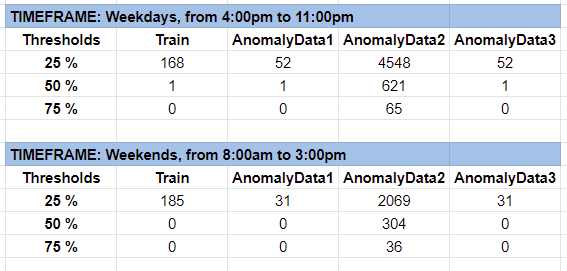


Interestingly, both ‘AnomalyDataset1’ and ‘AnomalyDataset3’ resulted in the exact same log-likelihood, whereas ‘AnomalyDataset2’ had the most anomalies by far. These results can also be verified by the Moving Average and Mean-SD methods used below which show ‘AnomalyDataset1’ and ‘AnomalyDataset3’ showing similar anomalies for the 3 variables whereas ‘AnomalyDataset2’ has more anomalies than the other two.

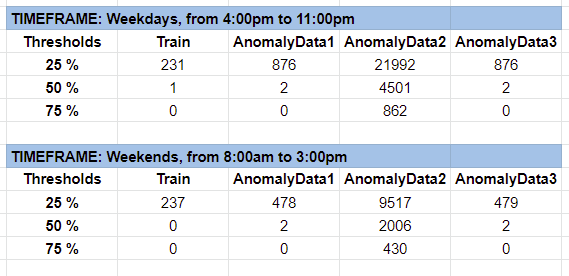
**b. Moving average method**

To further explore anomaly detection methods, the moving average method was also applied. In this case, Exponential Moving Average was used over Simple Moving Average, because it puts more weight on recent values, resulting in more accurate and representative data. For each anomaly dataset, the moving average for each chosen response variable was calculated. A window size of 10 was chosen as that provided a sufficient sample of information without intaking too much data. Then, thresholds were set at 25%, 50% and 75% of the values to determine whether the data was anomalous or not. These percentages were chosen since they encompassed all of the anomalous values and adequately partitioned the values, allowing us to see the effects that different thresholds had on the anomalous data. The Moving Average function was applied to the Training dataset to get an idea of what kind and how many anomalies were present and to give us a base to compare the results of the other datasets to. The number of anomalies in each dataset were counted, resulting in this data:

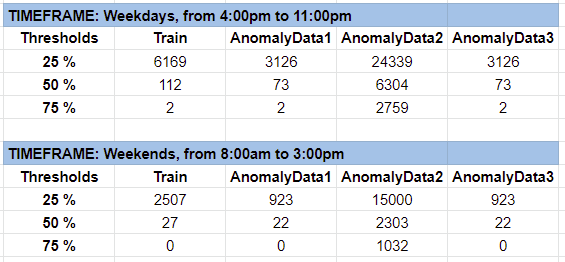
**Global Intensity**



**Global Active Power**



**Submetering 3**



The results show a strong correlation with this method’s results and log-likelihood method’s results. Both methods presented anomaly datasets 1 and 3 having the same proportion of anomalies, with dataset 2 containing drastically more anomalies. Additionally, it was clear that the number of detected anomalies increased as the threshold got looser. A common challenge faced in many anomaly detection techniques is that of a high false alarm rate. In order to be precise, techniques must strike a good balance between true positives, true negatives, false positives, and false negatives. In this case, a threshold of ±25% might be too strict, as it may mistake completely normal data points to be anomalies (false positives). On the other hand, ±75% may not be strict enough and miss some anomalies (false negatives). Clearly, the threshold should be a middle point of the two so as to only detect true positives and true negatives. This is too idealistic, however, as no technique will ever be 100% perfect in detecting anomalies. The best that can be done is to find a good balance of false positives and false negatives, and for this particular case the threshold of ±50% may be a good candidate.

To verify the findings of the Moving Average, another method was used to find outliers to see whether the readings from the new method matched those of the log-likelihood and Moving Average methods. It is a method known as the ‘Mean and Standard deviation method’ in which the respective values of the datasets were found (including the anomalous ones). The ‘min’ and ‘max’ values were calculated based on the formula for those two boundaries as:-

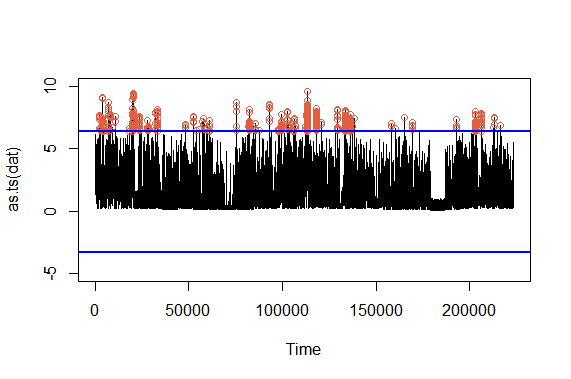
Min = mean(data) - n\*sd(data)

Max = mean(data) + n\*sd(data)

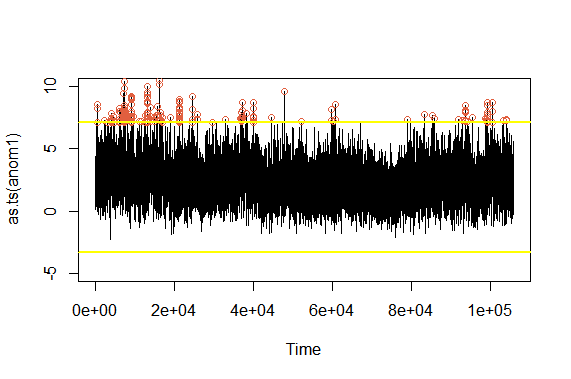
Wherein ‘n’ is the value of how many standard deviations away from the mean are being used for the data. A value of 4 standard deviations was used for Global Active Power and Global Intensity since the data points was very scattered and spread out which allowed us to use a higher SD value whereas for Submetering\_3, a standard deviation value of 2.5 was chosen since the data was very compact and had many 0 values which tended to interfere with the data. The purpose of using this method was to visualize how the anomalies look while also providing a backup to demonstrate that the log-likelihood and Moving Average methods match with this method, showing that the anomalies found are valid and not a mistake. The next 12 pages display the graphs for the Mean-SD method.

**WEEKDAY Mean-Standard Deviation graphs**

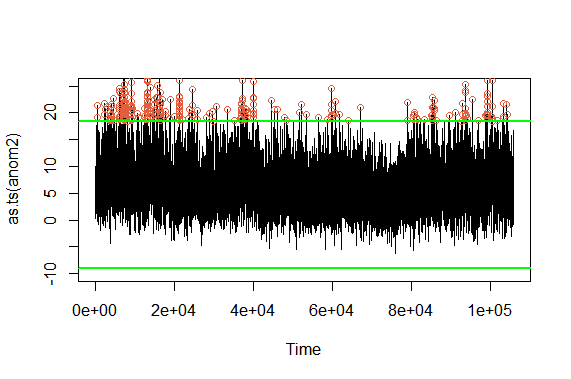
**Global Active Power**



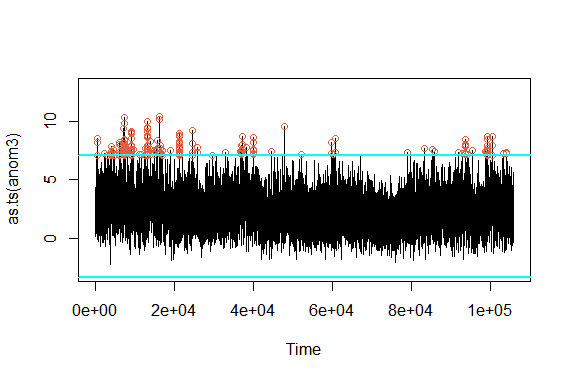
*Main Dataset*



*Anomaly Dataset 1*

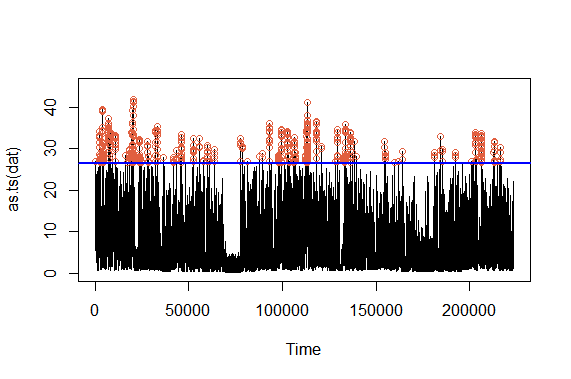


*Anomaly Dataset 2*

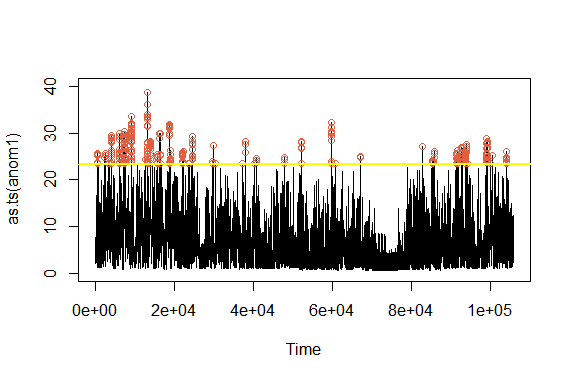


*Anomaly Dataset 3*

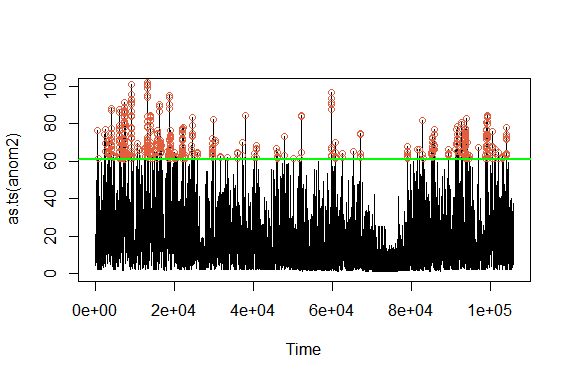
**Global Intensity**



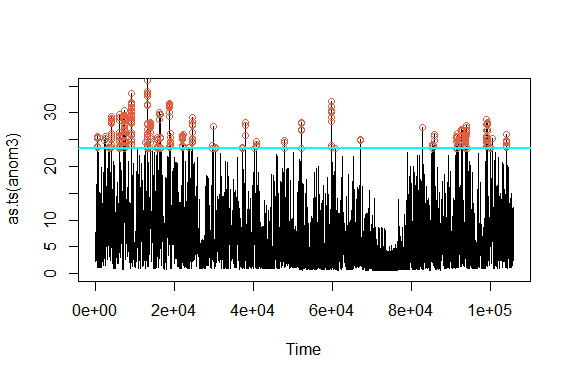
*Main Dataset*



*Anomaly Dataset 1*

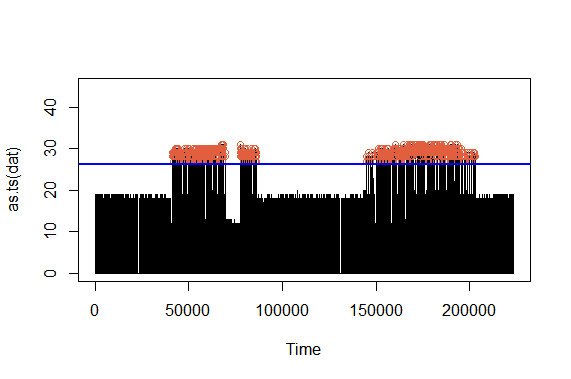


*Anomaly Dataset 2*

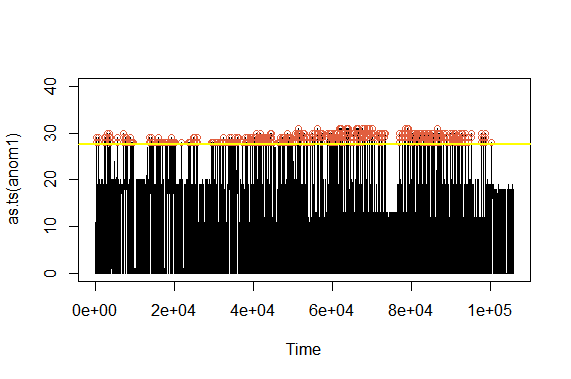


*Anomaly Dataset 3*

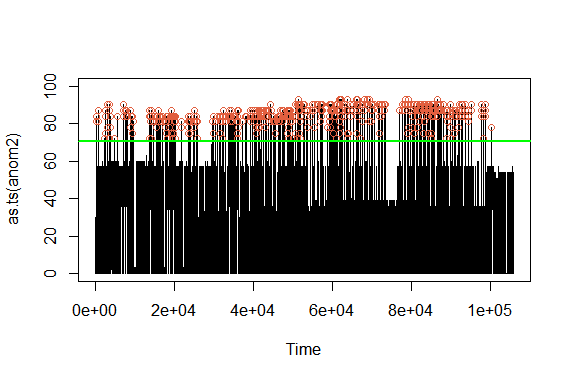
**Submetering\_3**



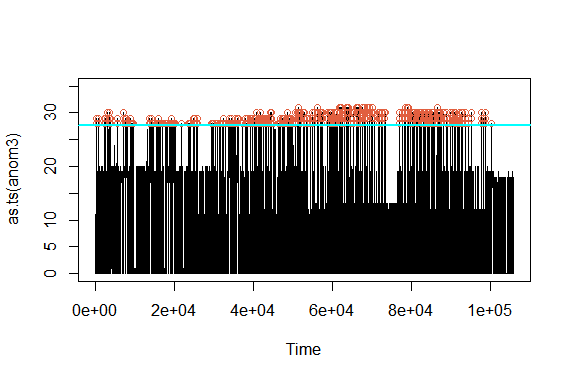
*Main Dataset*



*Anomaly Dataset 1*

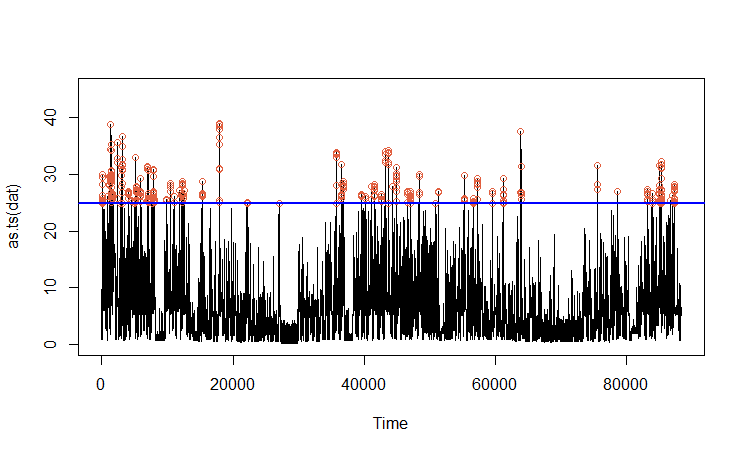


*Anomaly Dataset 2*

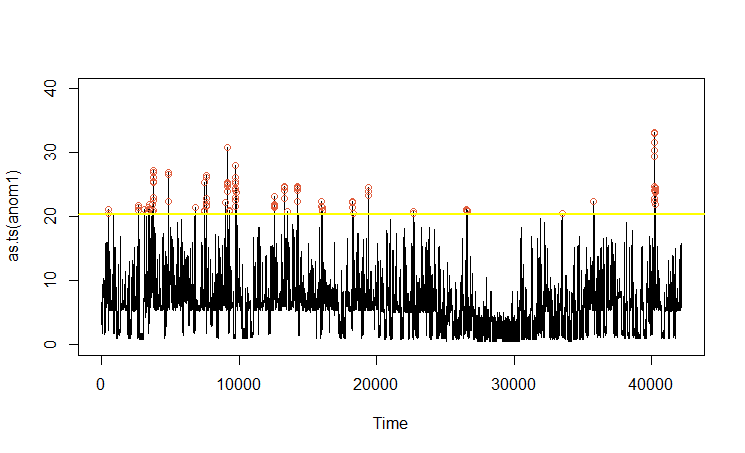


*Anomaly Dataset 3*

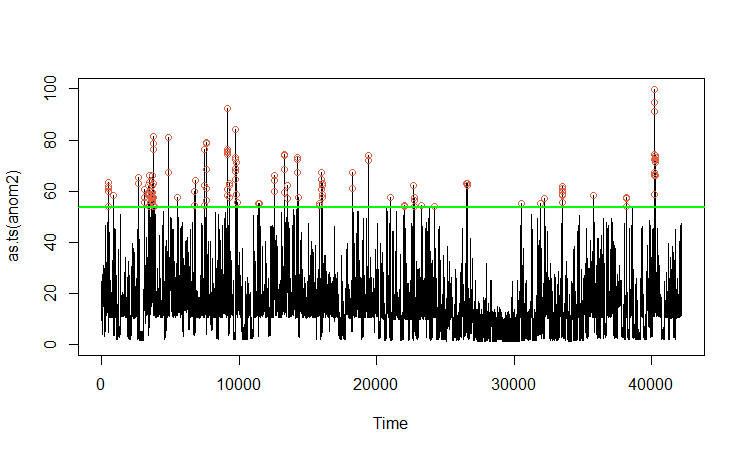
**WEEKEND Mean-Standard Deviation graphs**

**Global Intensity**

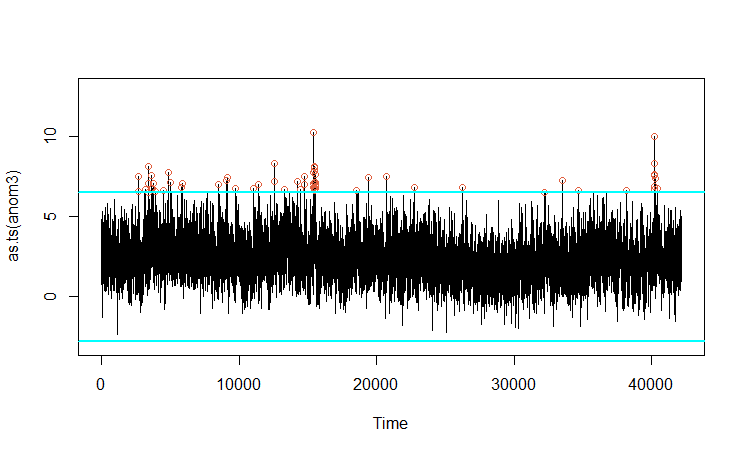
*Main Dataset*

****

*Anomaly Dataset 1*

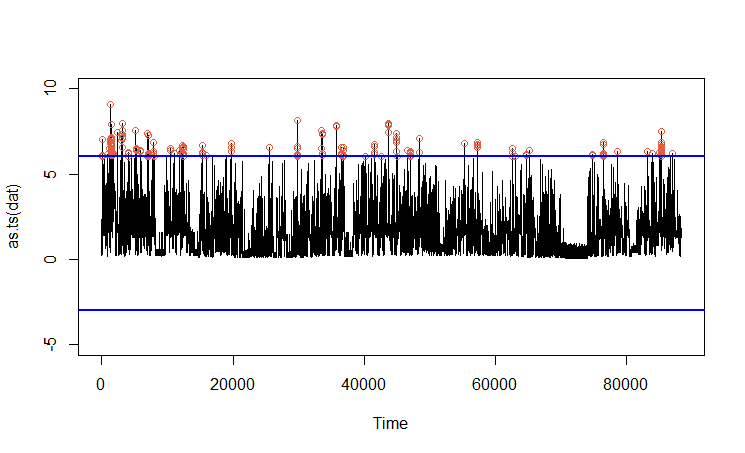
****

*Anomaly Dataset 2*

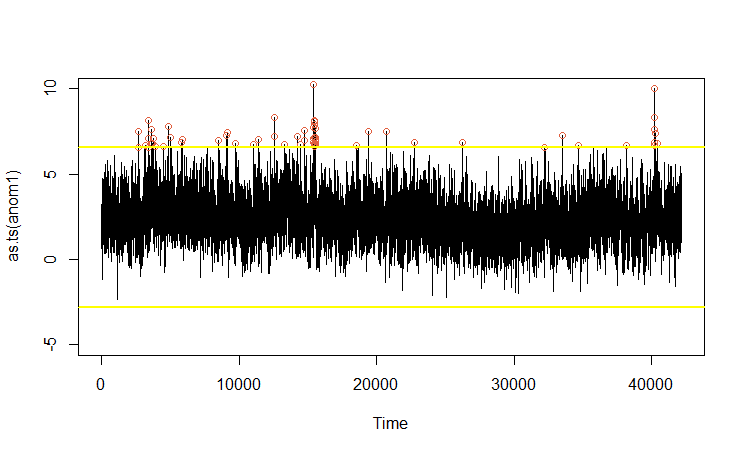
****

*Anomaly Dataset 3*

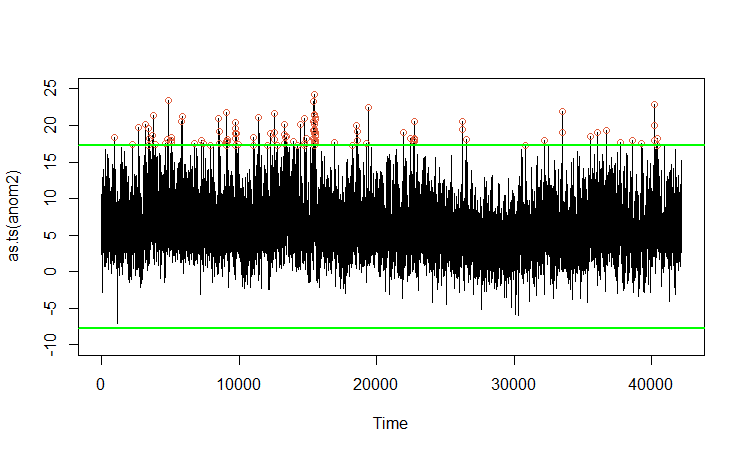
**Global Active Power**

****

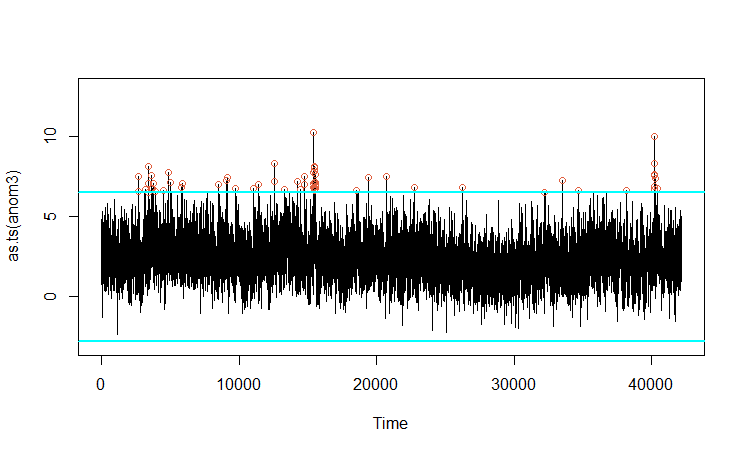
*Main Dataset*

****

*Anomaly Dataset 1*

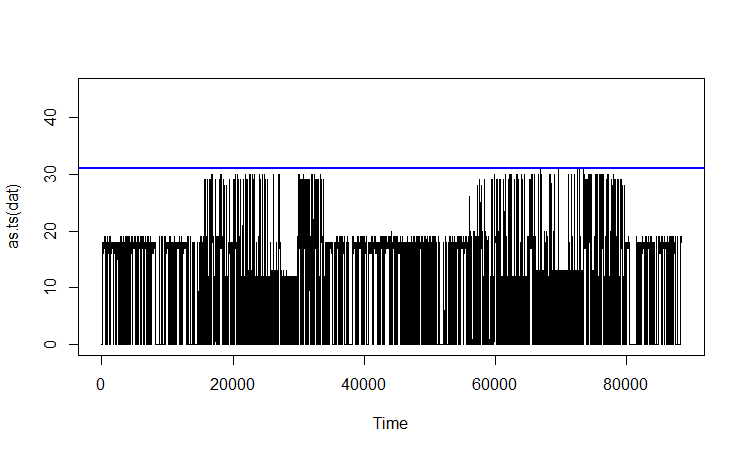
****

*Anomaly Dataset 2*

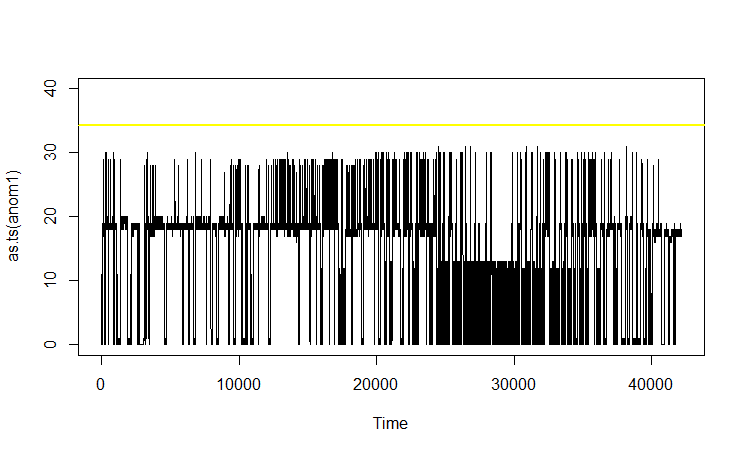
****

*Anomaly Dataset 3*

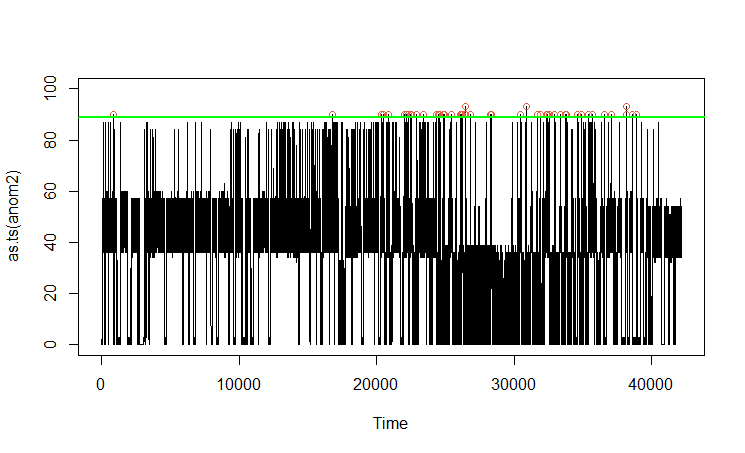
**Submetering 3**

****

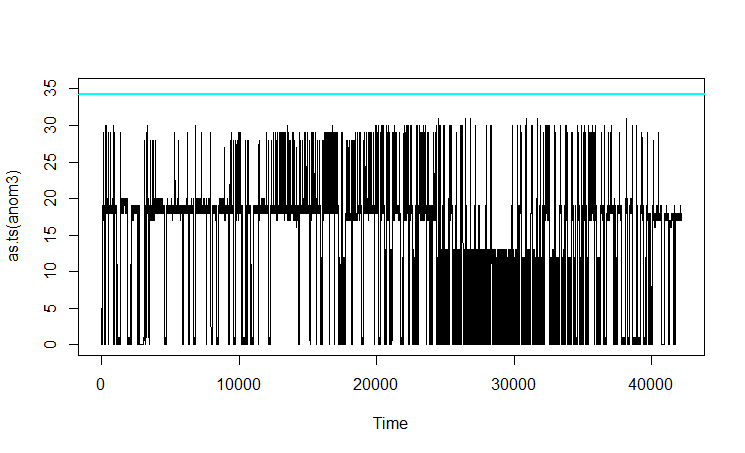
*Main Dataset*

****

*Anomaly Dataset 1*

****

*Anomaly Dataset 2*

****

*Anomaly Dataset 3*

**iv. Technical Essay**

Under the umbrella of machine learning, reinforcement learning stands out as a unique and powerful approach towards teaching an AI to perform a task. In reinforcement learning, an agent, namely an artificial intelligence program, performs actions in its environment which results in changes in the environment’s state. That is to say, the agent’s actions follow a Markov decision process. Depending on the state that is reached, the agent may receive a reward or a punishment, for example, an increase or decrease in its overall score. Learning can be achieved through a brute force approach, in which the agent will perform random sequences of actions, some of which yielding positive rewards. Over many iterations and an extended duration, the agent can learn which sequences of actions result in rewards (which is known as a policy), and thus “learns” how to complete the task. It continues to iterate upon itself, growing smarter and more optimized as time passes, until its progress ultimately plateaus and the agent reaches an optimal policy. As with other methods of machine learning, this type of approach is enabled by the fact that many iterations and actions can be taken in parallel, due to the agent’s actions being done virtually, sometimes through simulations. Unlike a human agent, an artificial intelligence agent is much less bounded by time, as it can easily perform thousands of actions per second in different scenarios, thus being able to “brute force” the correct sequence of actions through sheer probability.

Reinforcement learning has several significant advantages over other methods of machine learning. Compared with supervised learning, in which the agent is shown an example of how to complete the given task and then attempts to gather the correct sequences to match the provided solution, reinforcement learning can yield more efficient and creative solutions. This is because while supervised learning can land at a solution relatively quickly, since it was modelled after a human’s solution, the agent’s actions may not be optimal, as an AI does not have the same limitations as a human, and thus may be able to complete the given task in a novel and unique way that a human may not have been able to derive. An example of this advantage can be seen in artificial intelligence that has been trained to complete specific video games. Since these agents run in a virtual environment, they can exploit certain elements of game interaction that may lead to more efficient solutions, such as inputting multiple movements at once and performing actions that ignore the need for reaction time, both of which are human limitations that do not apply to the AI. This enables reinforcement learning to be able to achieve highly efficient and optimized solutions, sometimes even for highly complicated problems. At times, reinforcement learning may be the only approach possible for solving a problem, as conventional methods may not be able to deal with the high difficulty or complexity of a task. It is also less susceptible to repeating errors in its action sequence compared to other machine learning methods, as sequences that contain errors will be less favored in being fed through the cycle and are more likely to be phased out from the overall sequence over time.

However, that isn’t to say that reinforcement learning does not come with certain disadvantages. Firstly, reinforcement learning requires a large amount of data to work with in order to provide optimal results, which means it is not a good choice for solving simpler problems. Due to this high data requirement, significant computation power is also needed to reduce the time needed to work with the sheer amount of data the agent makes use of. Another common problem that may arise is a behavioural overfit to a sequence of actions in order to maximize rewards. This can occur when the states in which the agent receives a reward are not well designed, resulting in the agent’s actions generating a high amount of rewards without actually completing the intended task. This issue is known as the “alignment problem,” and occurs because it can be very difficult at times to lay out reward states in such a way that is not easily exploited by an AI agent. Additionally, having too many reward states can lead to the solution not being as optimized as it could be, as many reward states have a similar effect to supervised learning where the agent is given an example on how to complete a task. A high amount of rewards can lead to the agent’s behaviour being shaped by the set rewards, which can prevent it from arriving at a truly optimal solution by restricting it to following the path the environment designer decided upon. On the other end of the spectrum, having too few reward states may result in the agent never finding an optimal solution to the problem. This can occur if the correct sequence of actions before a reward is highly complicated and specific, which could result in the agent taking far too long or not finding a solution at all, even through many iterations of random actions. This can be especially true when there are a large amount of possible actions an agent can perform, for example, a robotic arm has many different possible moves, having multiple parts and appendages to maneuver in 3 dimensions. For this reason it may not be feasible to train a robotic arm to perform a complicated action such as making a cup of coffee, with reinforcement learning. High data and computation requirements coupled with the difficulty of avoiding the need to determine specific reward states on a case to case basis also results in reinforcement learning not being a very scalable solution if designed poorly.

As time passes, it is inevitable that humanity will continue to make progress in developing technology with ever increasing complexity and functionality. As such, it is the case that over time, cyber security attack methods will also become increasingly capable and dangerous. The upsurge of sophistication of these attacks can pose serious threats to individual and even national security, which warrants the need of improving current cyber security measures. One possible new cyber attack method that could be imposed in the near future are attacks carried out by an artificial intelligence. This type of attack may be troublesome to defend against with currently available security measures, as AI driven behaviour can be a black box, without any way to predict or anticipate its behaviour. This makes it difficult for humans to design defense measures against such an attack, as the malicious AI could be much more cunning than any human attackers that have been dealt with in the past. For example, an artificial intelligence may perform a cyber attack in such a way that it only leaves minute traces of its existence, and performs its invasion in an unusual or non-accounted-for way.

In particular, anomaly detection can be used to identify when an online system has had its security compromised, which allows for appropriate countermeasures to be taken in a timely manner. Anomaly detection systems gather information about standard user behavior and determine expected or “normal” patterns of behavior. This is useful in recognizing when irregular (and possibly harmful) actions are being taken. General anomaly detection systems that do not utilize AI can be effective at doing this, but carry several drawbacks. Firstly, there can be a large proportion of warnings/intrusion alarms that are raised without any real threat. This is because while a standard intrusion detection system may be able to find deviations in standard or expected user behaviour, it may not be ill-intentioned behavior that it flags as dangerous, resulting in a significant possibility for false alarms. An overly sensitive security system may result in a loss of faith in the system, or at the very least, an undermining of the severity of the warnings the system presents by the individuals monitoring its feedback. Anomaly detection systems that do not suffer from a high rate of false alarms do exist, but do so at the tradeoff of not being able to recognize unique and unseen types of attacks, which nullifies much of its usefulness in the wake of AI cyber attacks. Creating systems that solve both of these issues results in a system that is overly complex and difficult to implement in a variety of environments, as they require a large degree of case specific parameters to be set, resulting in a solution that is not adaptable nor scalable.

One possible countermeasure for the looming threat of AI cyber attacks is using AI to protect us from AI. Cyber security systems controlled by artificial intelligence can provide a greater degree of defense and attention to detail compared to existing methods, and reinforcement learning presents itself as a highly viable method for developing proficient enough AI to protect ourselves. In addition to this, artificial intelligence powered anomaly/intrusion detection systems, if designed correctly, can avoid many problems that current systems have today. These AI systems suffer less from high false alarm rates, while also maintaining itself as an adaptable security measure, being able to detect varying signs of intrusion and possibly even being able to infer whether simple behavioral anomalies are innocent or malevolent. Furthermore, due to the self learning nature of these systems, they may even be able to provide us with ample protection in the case of a never before seen cyber attack being put in place, as its policy is not shaped by an example, but rather, its experiences. For example, researchers at the University of York employed a method of reinforcement learning with the addition of “tile coding” to create an agent capable of detecting possible DDoS attacks by examining incoming network traffic information and patterns. Their agent was first trained with real-life data, and after learning through many iterations, was able to find anomalous data and behaviour and act accordingly. While they succeeded in creating a viable agent which solved many problems that current detection systems suffer from, it did not come without its own drawbacks. Their agent struggled at times dealing with highly complicated attacks, and in such cases could raise false alarms. Despite this however, the researchers were optimistic that their agent could be further refined in various ways to both add functionality and lessen its shortcomings.

Reinforcement learning is an effective and advantageous approach for developing and teaching proficient AI. In the face of the possibility of artificial intelligence fueled cyber attacks, applying reinforcement learning to develop anomaly detection systems is a promising cybersecurity measure. While it is not without its flaws, it can be much more powerful than other conventional methods due to its ability to detect complex and unique attacks in ways that humans could not implement otherwise. Ultimately, it is not the goal of cybersecurity to be able to block every possible cyber attack, but rather have an optimal method in place to allow for damage reduction, and artificial intelligence presents itself as a quite capable solution for achieving that goal. Cybersecurity researchers are continuing to push the boundary of AI systems and their capabilities, and it is becoming increasingly clear that reinforcement learning and artificial intelligence as a whole will become a valuable tool to add to humanity’s arsenal of cyber security measures in the near future.

**v. Contributions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PCA | HMM(train and test) | Anomaly detection | Technical Essay | Project Report | Presentation Slides |
| Dan A. | 🗸 | - | - | 🗸 | 🗸 | 🗸 |
| Vinay L. | - | 🗸 | 🗸 | - | 🗸 | 🗸 |
| Rakim Z. | 🗸 | 🗸 | 🗸 | - | 🗸 | 🗸 |

Individual Reports

Dan A.

At the beginning of the project, I worked alongside Rakim on getting the Principal Component Analysis working so that we could decide upon which response variables to use for the project. I worked on getting the proper values as well as plotting the variance proportions and biplots of the data. Afterwards, I put the majority of my efforts into the Technical Essay, as we realized that it would take a significant amount of time to complete and it would be better to have one person complete the essay as it would allow for a singular and uninterrupted narration style as well as provide the other members with ample time to figure out the HMM training and the Anomaly Detection. I conducted research online by reading articles, research papers, and watching videos on the subject of Reinforcement Learning and its application in Anomaly Detection. After conducting sufficient research, I began writing the essay, and after a few iterations and some proofreading, it was finished. When I completed the essay, the other members had finalized the code, indicating that we designated tasks amongst the members well. All that was left to do was put together the Project Report (abstract, conclusions, proofreading, etc.) and make the Presentation Slides, which is what the remainder of my effort was put towards.

Vinay L.

When we started the project, the first task I took on was to train and test the HMMs based on how we had done in Assignment 3. While my teammates focused on the Principal Component Analysis, I ran the code for the various states of the Training and Testing HMMs and then gave the getpars and setpars part to my teammates. The various states BIC and log-likelihood values were plotted and analyzed to find the best state fit for both. The second task I took up was to continue on the Anomaly detection and try to figure out the Moving Average function, which took a bit of research and tests to try and understand the workings of the function as well as errors which were being faced. After finishing the Moving Average, I worked on the Mean-Standard Deviation method to firstly verify the results we were seeing in the log-likelihood and Moving Average methods and also to plot the results and see if they correlated with the results we found in the other methods. The last thing to do was a team effort to put together the Project Report and Slides while one of the teammates finished the technical essay, all three of us put together the Report and Slides which was the conclusion to this Project.

Rakim Z.

Since our entire project essentially depended on the variables that we chose to work with, Dan and I worked together on the PCA. I subsetted the entire project dataset into weekdays and weekends, making sure to only use the time windows that we described in the project. We worked together to do the calculations and verified each other’s results, while he did the plotting. Vinay and I both trained our HMM with our train dataset to make sure we obtained the best model, comparing the log-likelihoods and BIC values that we got for different states. By using setpars(), getpars(), and forwardbackward(), I calculated the log likelihood of the test model and verified that our train model was a suitable predictive model. Using the same technique, I applied the log likelihood method on the anomaly datasets to find the varying degrees of anomalies present in them. For the moving average method, I helped Vinay in displaying the graphs and tables for the weekend dataset. While all of this was going on, I was also working on the project report (and slides), making sure that we clearly displayed all of our results. Additionally, I added in explanations for the major problems in intrusion detection (false alarm, NA values, etc) and explained our design choices regarding them.

**vi. Conclusion**

To conclude, this project taught us many things about anomaly detection based intrusion detection methods. We learned about the various tools needed to ensure that we properly conducted anomaly detection, which is very important as credible data is what one seeks when doing this type of analysis. We also learned a fair bit about the potential difficulties of the anomaly detection process as we encountered our fair share of issues. Firstly, dealing with N/A values proved to be quite cumbersome and caused us a few problems initially. Similarly, we had to deal with many 0 values when working with variables such as Submetering, which also caused issues with the data analysis and anomaly detection. The PCA biplots did not appear very appealing due to the large amount of data points, which was unfortunate. Additionally, running the code requires a significant time investment and decent CPU power as it requires training a 16 state HMM. The technical essay component opened our eyes to the world of anomaly detection outside the scope of what we were performing, showcasing how AI can enable orders of magnitudes of more complex and powerful intrusion detection systems.

**vii. References**

Insights, Arxiv. "An Introduction To Reinforcement Learning". Youtube, 2018, https://www.youtube.com/watch?v=JgvyzIkgxF0. Accessed 1 Dec 2020.

Joy, Ashwin. "Pros And Cons Of Reinforcement Learning | Pythonista Planet". Pythonista Planet, <https://www.pythonistaplanet.com/pros-and-cons-of-reinforcement-learning/>.

Koduvely, Hari. "Anomaly Detection Through Reinforcement Learning". Zighra, 2018, <https://zighra.com/blogs/anomaly-detection-through-reinforcement-learning/>.

Servin, Arturo. "Towards Traffic Anomaly Detection Via Reinforcement Learning And Data Flow". Cs.York.Ac.Uk, <https://www.cs.york.ac.uk/yds/pub/07/proceedings_07/11/11.pdf>.

Package ‘zoo’, May 2, 2020, <https://cran.r-project.org/web/packages/zoo/zoo.pdf>

Package ‘pracma’, December 15, 2019, <https://cran.r-project.org/web/packages/pracma/pracma.pdf>

Package ‘depmixS4’, January 20, 2020, <https://cran.r-project.org/web/packages/depmixS4/depmixS4.pdf>

# 

# How to Calculate an Exponential Moving Average in R, October 29, 2020, <https://www.statology.org/exponential-moving-average-in-r/>