Term Project Report

CMPT 318, Spring 2022, Group 6

Rauf Shimarov, 301397321
Kevin Che Su, 301386066,
Parth Vakil, 301320486

Abstract

This report is intended to analyze electricity consumption data to show anomalies, if any, that are present within the data. Using these intrusion detection methods we can gain cyber situational awareness in the analysis of automated control processes. First, we will perform PCA on the data and choose the appropriate response variables. After cleaning and preparing the data for the analysis, we will build multiple Hidden Markov Models with a different number of states that potentially define the normal patterns of electricity consumption. We then will showcase the different BIC and log-likelihood values that we generated based on different HMM states, and decide on the best model. Lastly, we will apply the model to the provided potentially anomalous electricity consumption data to detect anomalies, if any.

Introduction

Problem Scope

The problem our team has been assigned to address was to detect anomalies that deviate from the normal operation of the critical infrastructure such as electric power grids, public water utilities, and smart transportation networks, that rely on supervisory control systems. These efforts are put forward to protect against not only cyber attacks, but also initially unexplainable deviations from normal patterns. An example where operation patterns deviated from the norm was the 2003 Northeast Blackout. After an alarm failed to warn operators and after hours of investigation, it was determined that a tree branch sagging on a high-voltage powerline was the root cause of “contributing to at least 11 deaths" and leaving 50 million people without power. Our team is determined to protect critical infrastructure from cyber threats. These critical infrastructure components ensure the daily lives of people are uninterrupted and daily processes run smoothly. These efforts protect and help mitigate the impact on essential services such as water distribution, energy distribution, and other economic services. This is what our team is looking to avoid and therefore, we are running numerous statistical analysis techniques on the data we have been provided.

Our team is asked to build an anomaly-detection-based intrusion detection system using Hidden Markov Model that would allow us to detect anomalies in the provided 3 files with potentially anomalous electricity consumption data over 1 year. We are also provided a dataset of electricity consumption that can help to define the normal behavior of the electricity consumption

over 3 years. This analysis will be facilitated by a team of 3, and this report will describe our findings, the model development process, and how the model can help to detect anomalies.

Exploratory Data Analysis

Methodology

Our team first began by taking a look at the data and making sure it was usable. We started off by scaling the data as it would be needed for PCA since we need to normalize the data to compare different variables with different ranges of values. Another reason that scaling is required is to build the multivariate HMM models later on. As we explored the given electricity consumption dataset over 3 years, we also noticed missing values for certain days and had to clean the data later on. To build our HMM model, we had to split the data into training and testing data. We noticed that 2007 and 2008 were the only two datasets that were full years and decided to include that in the training data as well as 2006 which was only a couple of days' worth of data. 2009 had about 11 months of data and was used as our testing data. It worked out to be a 70/30 split, with 70% training data and 30% for our testing data. Next, our team looked for general trends within the data. As we were trying to make a successful HMM, we wanted to minimize the BIC value, but by doing so we also increased the complexity of the model which can lead to overfitting. Lastly, we want to maximize the log-likelihood while also keeping it at a negative value as the increasing number of states can result in positive values thus indicating divergence.

Data Cleaning

The quality of data is important in order to come up with precise modeling and calculations. Unnecessary or invalid data can have a negative impact on our final results. The process that makes sure our data is correct and consistent is called data cleaning. Data cleaning can be done both manually or automatically which mainly depends on the size of our datasets. The larger the dataset, the more overwhelming it would become to do the data cleaning manually. In other words, the cleaner our dataset, the fancier our result would be.

Our dataset has a number of NA values. There are two possible ways to approach data cleaning in our case: we either try to interpolate the missing values or we can simply remove them from our data. Going with the first approach might be dangerous since although it can make some good predictions, it can also create some noise. Therefore, our team decided to go for a safer approach of removing the missing values and configuring the HMM parameters accordingly (ntimes parameter specifically).

Response Variables Selection

In the original dataset, some of the response variables had a large range of values while others had a small range. As a result, if we want to compare the values of the response variables, we need to scale each of the response variables. Otherwise, the comparison of two response variables with a different range of values will be invalid. Hence, before applying PCA, we scaled our dataset using the scale() R function.

The goal of PCA is to reduce the number of features in a high-dimensional dataset. We used the prcomp() R function to execute PCA. The result of this function is a set of 7 principal components. Each principal component is a linear combination of the original response

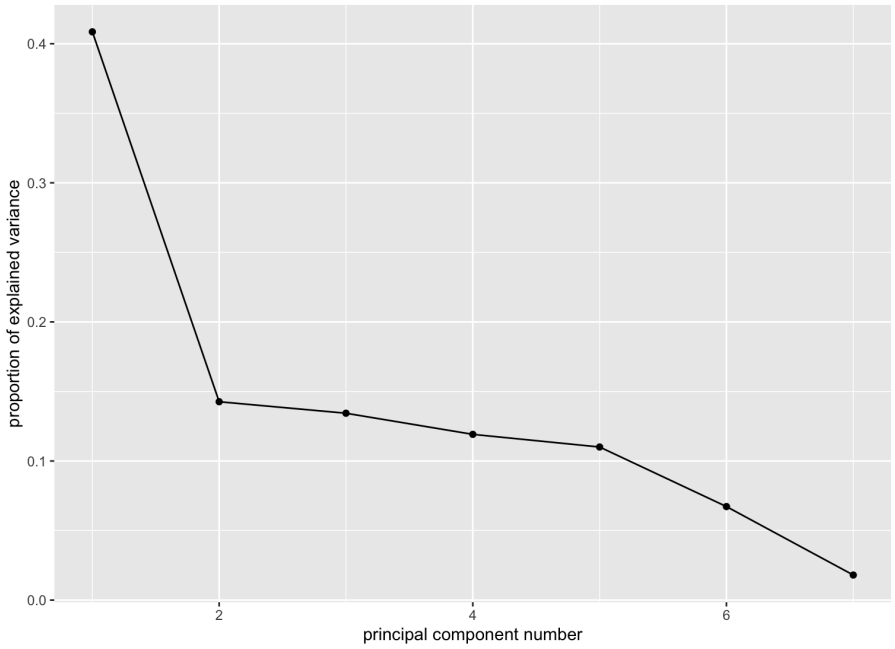
variables. For example, if we print pca$rotation, we can see that the first principal component (PC1) represents about 55% of the Global_intensity data, 46% of Global_active_power data, and 38% of Sub_metering_3 data. The proportions are called the loadings, absolute values of which show how much each response variable contributes to every principal component.

```
> print(pca$rotation)
                              PC1         PC2          PC3         PC4         PC5
data.Global_active_power   -0.4687199  0.13475701 -0.087364024 -0.06854240  0.26144877
data.Global_reactive_power -0.1947535 -0.74422839  0.166001666  0.60786960  0.06446593
data.Voltage                0.3305256 -0.13245740 -0.035064907 -0.13266015  0.91900003
data.Global_intensity      -0.5595970  0.01912909  0.001155733 -0.06409154  0.13818872
data.Sub_metering_1        -0.2988388 -0.12874880  0.728446337 -0.47839198  0.04294132
data.Sub_metering_2        -0.2837926 -0.41294122 -0.651209714 -0.42742059 -0.05496931
data.Sub_metering_3        -0.3874711  0.47218329 -0.094190943  0.43879699  0.24282899
                                PC6         PC7
data.Global_active_power   -0.76918472 -0.29968496
data.Global_reactive_power -0.03290397 -0.07677664
data.Voltage                0.08172722  0.05602833
data.Global_intensity       0.08117594  0.81036445
data.Sub_metering_1         0.24064565 -0.27362731
data.Sub_metering_2         0.28686667 -0.23846336
data.Sub_metering_3         0.50378618 -0.33574973
```

Using the summary() function, we can see how much of our original data is represented by a certain principal component. For example, PC1 has about 41% of the total variance, i.e. PC1 represents about 40% of the original dataset

```
> summary(pca)
Importance of components:
                          PC1    PC2    PC3    PC4    PC5     PC6     PC7
Standard deviation     1.6911 0.9992 0.9698 0.9133 0.8777 0.68606 0.35513
Proportion of Variance 0.4086 0.1426 0.1343 0.1192 0.1101 0.06724 0.01802
Cumulative Proportion  0.4086 0.5512 0.6855 0.8047 0.9147 0.98198 1.00000
```

Using the ggscreeplot() function from the ggbiplot package, we built a scree plot to show the proportion of explained variance for each principal component.



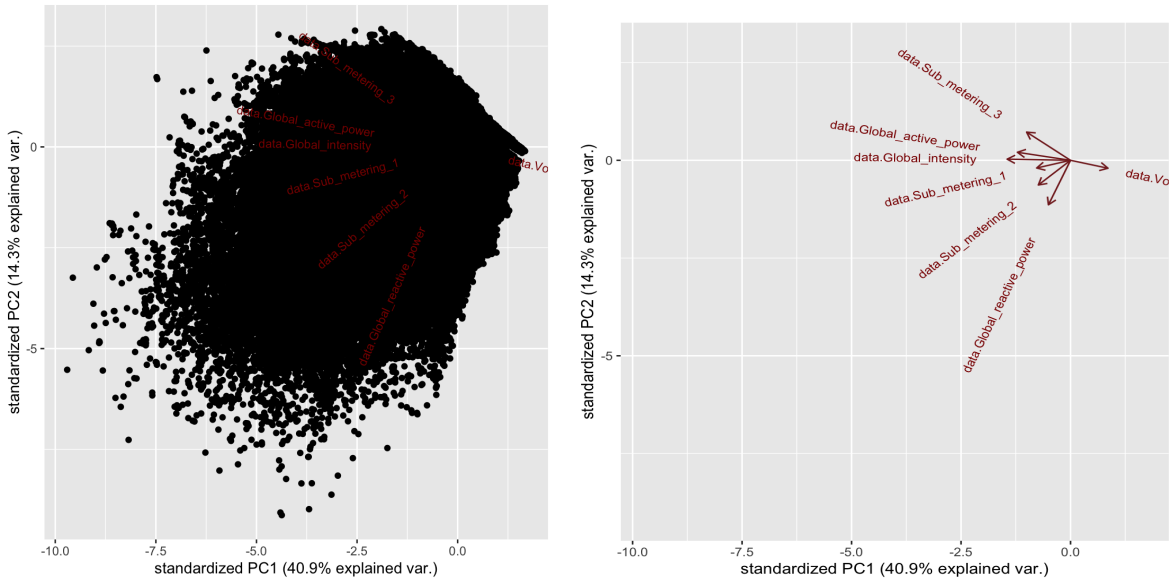
According to the scree plot, PC1 covers about 40% of the variance. The second and other PCs are considerably smaller than PC1 (with PC2 covering only about 14%), so we decided to use PC1 only for our feature selection.

We also calculated the variances of PCs by squaring their standard deviations. As a result, PC1 is the only PC that has a variance greater than 1. According to the tutorials provided, only PCs that have a variance greater than one should be considered in the analysis. The following is the ranking of the response variables according to the PC1 loadings:

1. data.Global_intensity 0.5595970
2. data.Global_active_power 0.4687199
3. data.Sub_metering_3 0.3874711
4. data.Voltage 0.3305256
5. data.Sub_metering_1 0.2988388
6. data.Sub_metering_2 0.2837926
7. data.Global_reactive_power 0.1947535

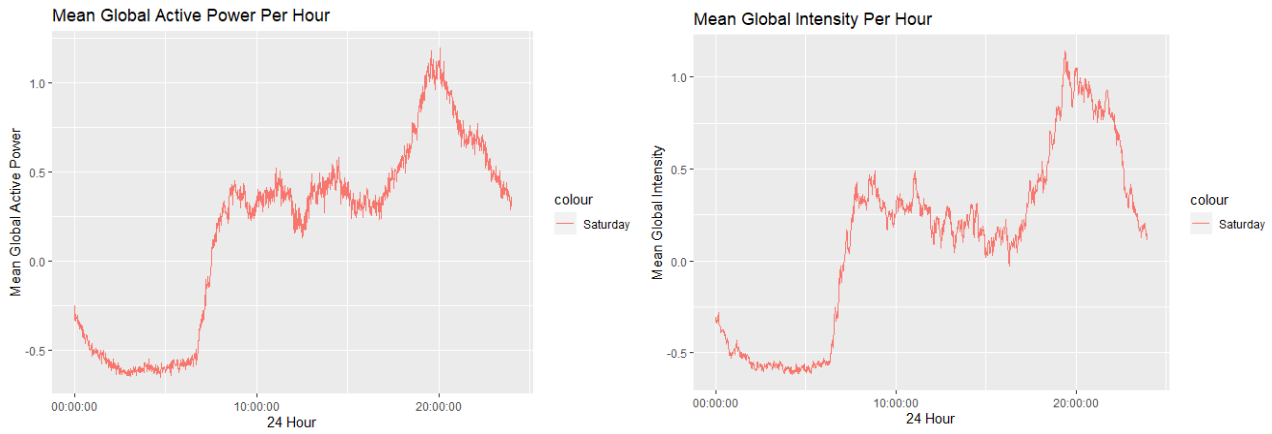
Hence, we choose **Global_intensity, Global_active_power data** as our response variables since they have the highest loadings in PC1. We used the ggbiplot() function to build a biplot of PCA and verify our conclusion on the response variable selection.

The biplot shows the original variables as vectors. The more a certain vector is parallel to the PC1 axis, the higher loading the response variable that corresponds to this vector has. As we can see from the biplot, **Global_intensity and Global_active_power data** are the top 2 vectors that are parallel to the PC1 axis the most.



Model Training

Determining Time Window



Our team decided to analyze the electricity consumption on Saturdays since we consider the data on the weekends to be more active and interesting. To draw meaningful observations, we aggregated and took the average for each time unit on Saturdays for the two response variables (Global Active Power and Global Intensity) over 3 years of observations. From our graphs above, it is clear that from 7 am to 12 pm, there is a steep increase in energy consumption. We consider this phenomenon to be interesting to analyze, so we chose to analyze a 7 AM to 12 PM time window on Saturdays. We also consider the size of the time window to be large enough to train a relatively reliable model.

Parameter Selection

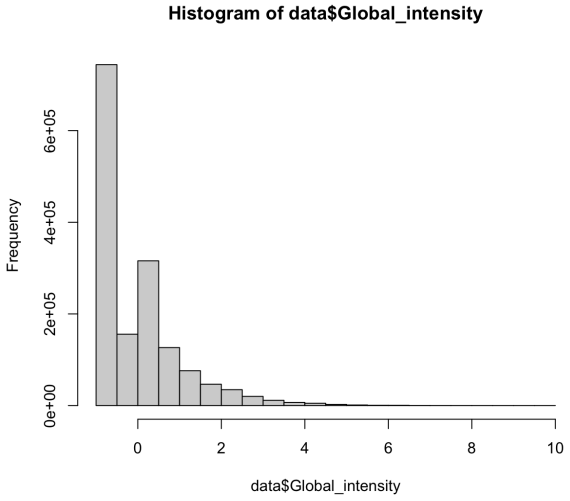
For the HMM model training we use the depmix function from the depmixS4 R package.

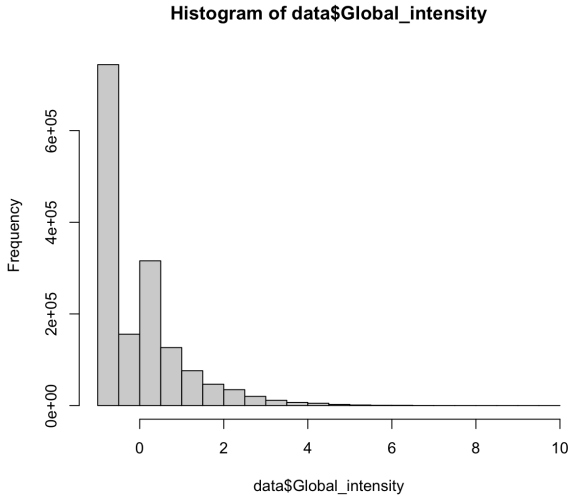
The main parameter we need to configure is the number of states in the model. We tried to train 11 models with different numbers of states ranging from 4 to 24 states. The decision on which model is the best among 11 is described in the next section.

One parameter of the depmix function that we need to configure is ntimes. The attribute is necessary as we are extracting the time windows for each Saturday of the week for a total of 52 Wednesdays in a year, for each of the 3 years we have. This is required so our learning algorithm knows that they are not separate data chunks, but different data for the same time window. The ntimes vector includes the size of each data chunk.

To get the ntimes vector, we extract the time windows for each Saturday, group them by date, and count the number of observations for the specified time window. This will account for the fact that we remove some rows that had no observations.

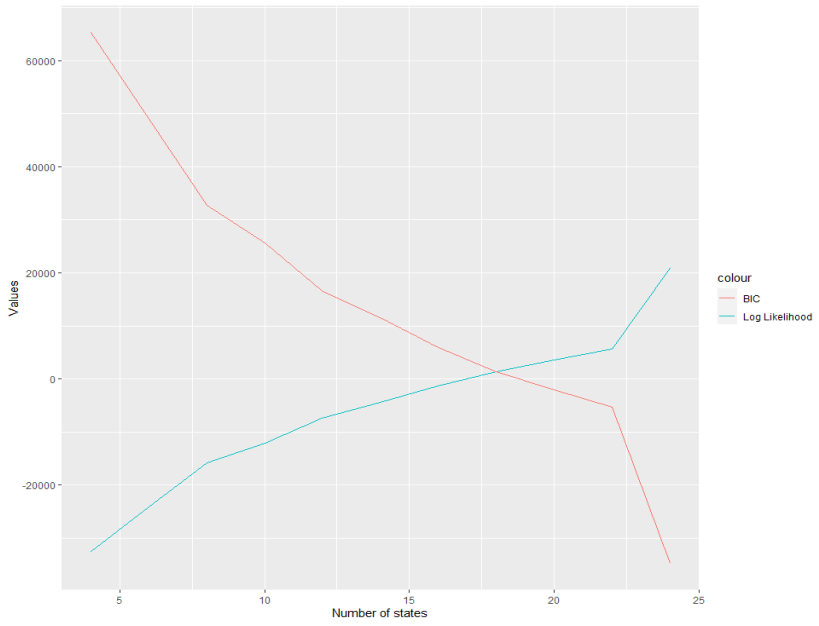
Another parameter that we need to configure is the family “parameter” for which we are given only two options in this term project - either multinomial or gaussian. We tried plotting Global_intensity and Global_active_power data. See the histograms below.





The data doesn't seem to be normal, so we used a multinomial family for both response variables for our first attempt in training the models. However, it took us 1 hour and 40 minutes to train the model for 4 states only (which is the smallest possible number of states in this project). Hence, due to a lack of computing resources, we decided to use the Gaussian family which converged much faster. Although our data is not normal, the Gaussian family results should be close to the multinomial family.

Best Multivariate Hidden Markov Models



The diagram above displays the BIC and log-likelihood values for each state from 4 to 24 for chosen Saturdays. Although BIC and log-likelihood intersect at the point with about 18 states and the model with 18 states is potentially good, our team decided to consider only the negative log-likelihood values, so we ignore all the states after 16. As shown in the graph, the log-likelihood, represented by the blue line, is maximized and the BIC, represented by the green line, is minimized when the number of states is at about 16 (given that we ignore the parts of the graph where the log-likelihood is positive). We chose the models with 14 and 16 states as our best models to continue our analysis.

States	4	6	8	10	12	14
BIC	65351.36	48998.86	32783.9	25647.78	16644.65	11559.28
Log-Likelihood	-32515.4	-24194.3	-15900.6	-12105	-7334.54	-4481.56

States	16	18	20	22	24
BIC	5935.85	1340.052	-2041.37	-5372.72	-34802
Log-likelihood	-1318.17	1372.78	3497.91	5639.378	20871.19

The table above shows the BIC and log-likelihood values for the number of states we ran.

Model Testing

To test our model, we created new models using the depmix() function with all parameters set the same as the training model, except the data parameter where we input testing data instead of training data. Then, we use the getpars() function to obtain the HHM parameters of our training model (A, B, and pi), and transfer these parameters to our testing model using the setpars() function. In addition, we used the forwardbackward() function to test our testing model and compute the log-likelihood for the testing model. We then normalized the log-likelihood of testing and training data to compare them because the lengths of the two data sets were different. We compare the training and testing log-likelihoods to make sure we are not overfitting. The following are our results:

States	Norm. Training LgLk	Norm. Testing LgLk
14	-43.09188	-56.0582
16	-12.67467	-17.43695

From the result above, we can see the difference between training and testing log-likelihoods for the model with 16 states is less than for the model with 14 states. Hence, our best model is the model with 16 states.

The training and testing log-likelihoods are close enough to claim that our model does not overfit the training data. This allows us to proceed to the next step of detecting the anomalous in the data that the model hasn't seen.

Anomalous Detection

We used our model to find out whether the provided potentially anomalous data is actually anomalous. For each of the three given data files, we extract the same time window over a full year. Then, we run the same testing logic described above (create a new model with data set to the data from a single file, copy HMM parameters from the training model to the new model, compute log-likelihood using forwardbackward() function). The following is a list of provided files from the most anomalous to the least anomalous:

DataWithAnomalies	Anomalies LogLik	(n = 16) Training LogLik
3	-609.0884	-12.67467
2	-374.2862	-12.67467
1	-374.278	-12.67467

As mentioned above, the log-likelihood of our training model, which defines the 'normal' behavior, is -12.67467. The log-likelihoods of all three models above differ from the 'normal' substantially, so, we claim that all three datasets are anomalous. The data from DataWithAnomalies3.txt is the most anomalous since it has the smallest log-likelihood among the three. DataWithAnomalies1.txt and DataWithAnomalies2.txt have about the same log-likelihood.

Lessons Learned

As technology continues to grow, it is becoming even more important for governments, companies, and people to protect themselves from cyberattacks. Furthermore, cybersecurity is essential in keeping critical infrastructures running. A successful attack on a critical infrastructure could cause access to be crippled to key resources such as electricity, water, and

more. Throughout this project we were tasked with using machine learning in order to achieve anomaly-based intrusion detection. It was interesting to see how much can be learned by conducting statistical analysis and using machine learning on raw data. We were able to see which days and time periods had higher electricity consumption as well as which datasets seemed the most anomalous. As we began the process for training our HMM model, we had to adjust multiple parameters in order to find the best model. The first parameter that we had to adjust was the ntimes. This is required so the learning algorithm knows that they are not separate data chunks, but different data for the same time window. The second parameter was the “family” parameter which needed to be configured as either multinomial or gaussian.Due to a lack of computer resources, we decided that using Gaussian will be accurate enough. Another parameter that we needed to select was the number of states to use. In our earlier modeling, we showed the different BIC and log-likelihood values between 4 and 24 states. While the interception point was 18, we decided that the maximum number of states to use should be 16 as we only wanted negative log-likelihood values. Upon further modeling, we compared 14 and 16 as the number of states, and found that 16 states outperformed 14 states. Thus, we decided to 16 as the number of states going forward. To conclude, the project was a great way to apply in-class knowledge to a real world example in order to achieve unsupervised intrusion detection on a stream of data.

References

1. Wood, R. (2020, May 21). *Learn principal component analysis in R*. Medium. Retrieved April 3, 2022, from https://towardsdatascience.com/learn-principle-component-analysis-in-r-ddba7c9b1064
2. Hartmann, k, Krois, J., & Waske, B. (2017, May 1). *Interpretation and visualization*. Interpretation and Visualization • SOGA •. Retrieved April 3, 2022, from https://www.geo.fu-berlin.de/en/v/soga/Geodata-analysis/Principal-Component-Analysis/principal-components-basics/Interpretation-and-visualization/index.html#:~:text=Interpreting%20Biplots,in%20a%20single%20biplot%20display.&text=The%20plot%20shows%20the%20observations,principal%20components%20(synthetic%20variables)
3. StatQuest with Josh Starmer. (n.d.). StatQuest: PCA Main Ideas in only 5 minutes!!! Retrieved April 4, 2022, from https://www.youtube.com/watch?v=HMOI_lkzW08
4. StatQuest with Josh Starmer. (n.d.). *StatQuest: Principal Component Analysis (PCA), Step-by-Step*. Retrieved April 4, 2022, from https://www.youtube.com/watch?v=FgakZw6K1QQ
5. StatQuest with Josh Starmer. (n.d.). *StatQuest: PCA in R*. Retrieved April 4, 2022, from https://youtu.be/0Jp4gsfOLMs
6. Wikimedia Foundation. (2022, March 3). Northeast blackout of 2003. Wikipedia. Retrieved April 3, 2022, from https://en.wikipedia.org/wiki/Northeast_blackout_of_2003
7. Wikimedia Foundation. (2022, March 17). *Ukraine Power Grid Hack*. Wikipedia. Retrieved April 3, 2022, from https://en.wikipedia.org/wiki/Ukraine_power_grid_hack