Term Project

Group 19

Kiarash Zamani, 301423525
Sahba Hajihoseini, 301433997
Quang minh Dinh, 301449800
Kayin Lam Chen, 301219238

In this project multivariate Hidden Markov Models are used on electricity consumption data for the purpose of anomaly detection. The initial step is to perform Principal Component Analysis (PCA) on the response variables and choose a subset of variables suitable for training HMM models. The data is scaled and then partitioned into train and test datasets, where various HMMs with a different number of states are trained on the train data for a selected time period. The best-performing model based on log-likelihood and BIC is then selected and the same model is used to determine the log-likelihood for the test data. This project concludes with computing the log-likelihood for observation sequences within the predetermined time window in three datasets with anomalies, followed by comparison and interpretation of the degree of anomalies present in each dataset.

CMPT318 - Spring 2023

Table of Content

1. Introduction....4

2. Background....4

2.1. Principal Component Analysis....4

2.2. Hidden Markov Model....5

3. Methodology....5

3.1. Feature Engineering....5

4. Characteristics of the Solution and Rationalization of Design Choices....8

4.1. Interpretation of the PCA's result and Choice of response variables....8

4.2. The choice for partitioning the data into train data and test data....10

4.3. Choices of HMM's Number of Hidden States....11

4.4. Interpretation of the Log-likelihood Results....13

5. Reinforcement Learning Paradigm....14

5.1. Overview....14

5.2. Advantages compared to the classical machine learning approach....15

6. Discussion....15

6.1. Problems encountered and solutions....15

7. Conclusion....16

7.1. Results....16

7.2. Lessons learned....17

8. References....19

Table of figures

Figure 1: The variance percentages of all PCs. ……8

Figure 2: The PCA plot of the new dataset.……..……..9

Figure 3:The comparison of the models with different states …….……..12

Figure 4: The best model with 10 states found by find_best_models function…….……..13

Figure 5: Log-likelihood of datasets with anomaly……..……17

1. Introduction

Data analysis of supervisory control systems to determine unsupervised intrusion detection using time series analysis and forecasting can be highly improved by using machine learning techniques. Throughout the term, our team has learned ways to analyze large amounts of data time series by using R. This project will focus on analyzing a time series data set of electric consumption obtained using supervisory control systems, then design and train a model to detect anomalies in other data sets. Thus we will focus on exploring ways for anomaly-detection-based intrusion methods used for situational awareness in the analysis of automated control processes. For this project, we will be using Principal Component Analysis to determine a suitable representative subset of variables for the training of Multivariate Hidden Markov Models, and then use the models to test for anomaly detection.

2. Background

2.1. Principal Component Analysis

One of the major problems of multivariate datasets is that it is often difficult to process and visualize them systematically to analyze the underlying relationships due to the high correlations between several different response variables [1]. A common way to work around this problem is by using Principal Component Analysis (PCA) to project the original datasets to a lower dimensional space, most commonly 2 while keeping as much information as possible for visualization and interpretation. This process is done by finding the orthogonal axes called Principal Components (PC), in a way such that the highest variations across the response variables are captured by the first principal component (PC1), and the next highest variations are captured by PC2. The new dataset created by only using the first few PCs is much less complex than the original dataset, at the same time still preserves most of the information from the original dataset, which makes it easier to visualize and analyze the underlying relationships

of the data. The major disadvantage of using PCA is that accuracy is lowered at the expense of simplicity, which is acceptable considering the benefits it brings [2].

Mathematically, PCA can be calculated using Singular Value Decomposition (SVD), in which the eigenvalues represent the variations of the PCs, and the proportions of each response variable in the original dataset that contributed to the construction of the PCs are called loading scores [3].

2.2. Hidden Markov Model

A hidden Markov model (HMM) is a probabilistic Markov model that is used to model data, where the underlying states are not directly observable but can be inferred from the observable outputs. Hidden Markov Models share the Markov property that refers to the memoryless property of a stochastic process, that is the conditional probability of distribution of future states solely depends on the present state of the model. HMMs are widely used in speech recognition, natural language processing, bioinformatics, etc. HMMs can be used for detecting anomalous data points in time series data, giving it the application of intrusion detection in cyber security. To measure the quality of the HMM model to the observed data one can use log-likelihood. Log-likelihood is the logarithm of the probability of observing a particular sequence of data, given the HMM parameters. The Bayesian Information Criterion (BIC) is a statistical measure used for model selection in HMM. It is a criterion that balances the fit of the model to the data with the complexity of the model.

3. Methodology

3.1. Feature Engineering

The dataset used in this project provides the measurements of a total of 7 response variables: *Global_active_power*, *Global_reactive_power*, *Voltage*, *Global_intensity*, *Sub_metering_1*, *Sub_metering_2*, *Sub_metering_3* for every minute in a time period of ~3

years from December 16th, 2006 to December 1st, 2009. To choose the appropriate features for the training of the multivariate Hidden Markov Models on the normal electricity consumption data, we perform PCA and analyze the loading scores of 7 response variables for PC1.

Before PCA can be performed on the dataset, we first create a function *extract_time_window* which will do all the data extraction. It will convert the date column to readable dates by using the *as.Date()* function and add a new column for the day of the week and fix the format of the time column. To fill the missing N/A values, we consider two methods: directly exclude them or use interpolation. Since excluding the N/A values would introduce observation sequences with different lengths and could damage the models, we decide to use interpolation, which estimates a value in accordance with the surrounding existing values.

Since each of the samples that would be needed for the model in the training process is a time series in a time window of a specific weekday, we decided to perform PCA on a similar dataset that best models the training dataset.

For PCA we created a function *get_resp_PCA*, which we used to create a new dataset by calculating the mean for all the response variables for that time frame. As PCA is very sensitive to the variances of the response variables, we standardized the new dataset using the function *mutate_at().* This is to avoid the biased results that might be introduced by PCA in case there are large differences between the ranges of the initial response variables. Another function we created was *get_resp* which we use to select features and rename the features for easier use for HMM training and testing.

Lastly, we create the compute_pca_variance function that computes all the pca calculations. The *prcomp()* function, we use to calculate the PCA of the new dataset in the specified time frame.

```
# Function to calculate pca and variance percentage
compute_pca_variance <- function(data_resp){
  # PCA calculation
  pca <- prcomp(data_resp, scale=TRUE)
```

```
  # Principal components summary
  summary(pca)
  # Calculate the percentage of variance
  var_per <-pca$sdev^2
  var_per <-round(var_per/sum(var_per)*100,1)

  result <- list ("pca" = pca, "var_per" = var_per)
  return (result)
}
```

We then applied the functions we created to obtain the pca summary.

```
time_window <- extract_time_window(df)
pca_resp <- get_resp_PCA(time_window)

# PCA calculation
pca_var <- compute_pca_variance (pca_resp)
pca <- pca_var$pca
p_var <-pca_var$var_per
pca
summary (pca)
p_var
```

```
summary (pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
Standard deviation     1.9475 1.2280 0.9764 0.72808 0.33643 0.29833 0.11678
Proportion of Variance 0.5418 0.2154 0.1362 0.07573 0.01617 0.01271 0.00195
Cumulative Proportion  0.5418 0.7572 0.8934 0.96917 0.98534 0.99805 1.00000
```

Table 1: Summary of the PCA results on the new dataset.

To calculate the proportions of each response variable in the original dataset that contributed to the construction of PC1, we get the loading scores for PC1 using the *rotation* component. The negative loading scores push the data to the left and positive scores push the data to the right.

```
loading_scores <- pca$rotation[,1]
resp_rank <- sort(abs(loading_scores), decreasing = TRUE)
pca$rotation[names(resp_rank), 1]
```

```
> pca$rotation[names(resp_rank), 1]
       Sub_metering_3              Voltage   Global_active_power      Global_intensity        Sub_metering_2
          -0.50433423           0.48940215           -0.48596413           -0.45333197            0.24507861
Global_reactive_power         Sub_metering_1
           0.06055742           -0.02703307
```

Table 2: The loading scores of all response variables for PC1.

From the above results, we can see that the data varies the most across *Sub_metering_3*, *Voltage*, *Global_active_power,* and *Global_intensity*.

4. Characteristics of the Solution and Rationalization of Design Choices

4.1. Interpretation of the PCA's result and Choice of response variables

From *Table 1*, we can see that PC1 has the highest standard deviation, meaning it is the one with the highest total variation of the data at 54.2% and PC2 explains 21% of the total variation. Meaning PC1 roughly represents a bit over half of the dataset and PC2 contains ¼ of the dataset. Their cumulative proportion is 75% of the total variance which means the first 2 Principal components can be a good representation of the dataset.

To better visualize the PCA result, we calculated the percentage of variance that each PC contributes by dividing each of the PC's variances by the total variance and multiplying them by 100 (percent) when we used the *compute_pca_variance* function. We then created a Scree plot [4] using the *barplot()* function. (Shown in *Figure 1*)

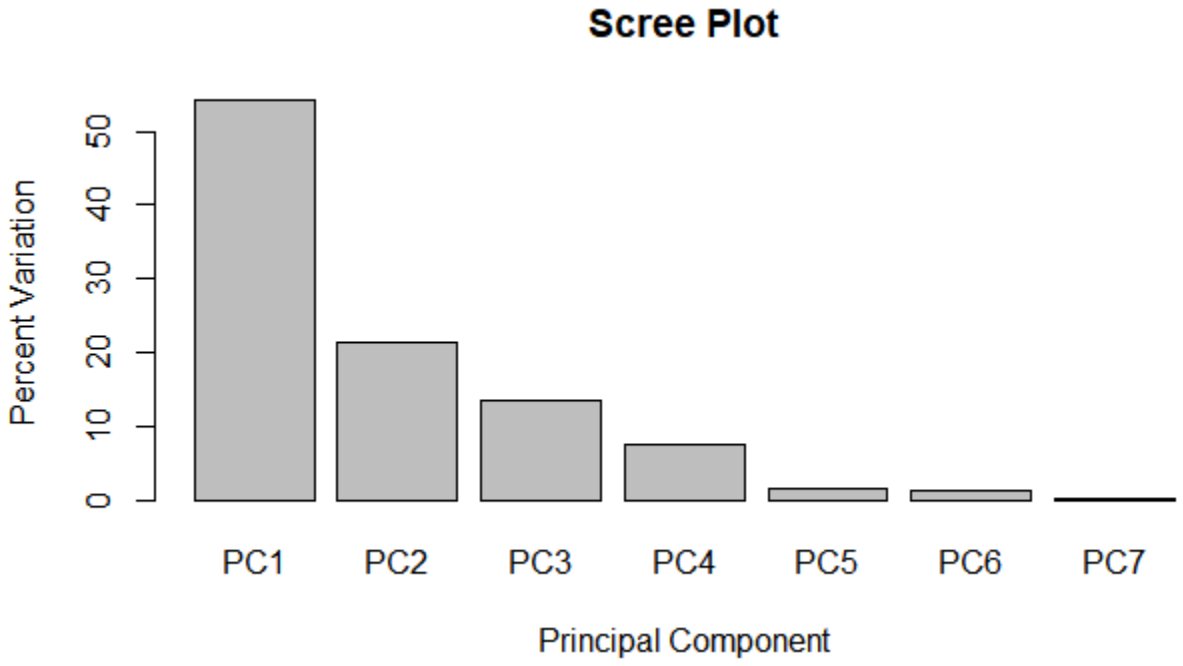

Figure 1: The variance percentages of all PCs.

We can see from *Figure 1* that PC1 has the highest variation, after using the function

p_var, we can see that PC1 variation is 54.2%. In other words, it contains 54.2% of the information from the dataset. The other PC variations are as follows, as listed in Table 3.

```
> p_var
[1] 54.2 21.5 13.6  7.6  1.6  1.3  0.2
```

Table 3: Principal Components variance

We then plot the PCA using the *ggbiplot()* function, which produces the following plot:

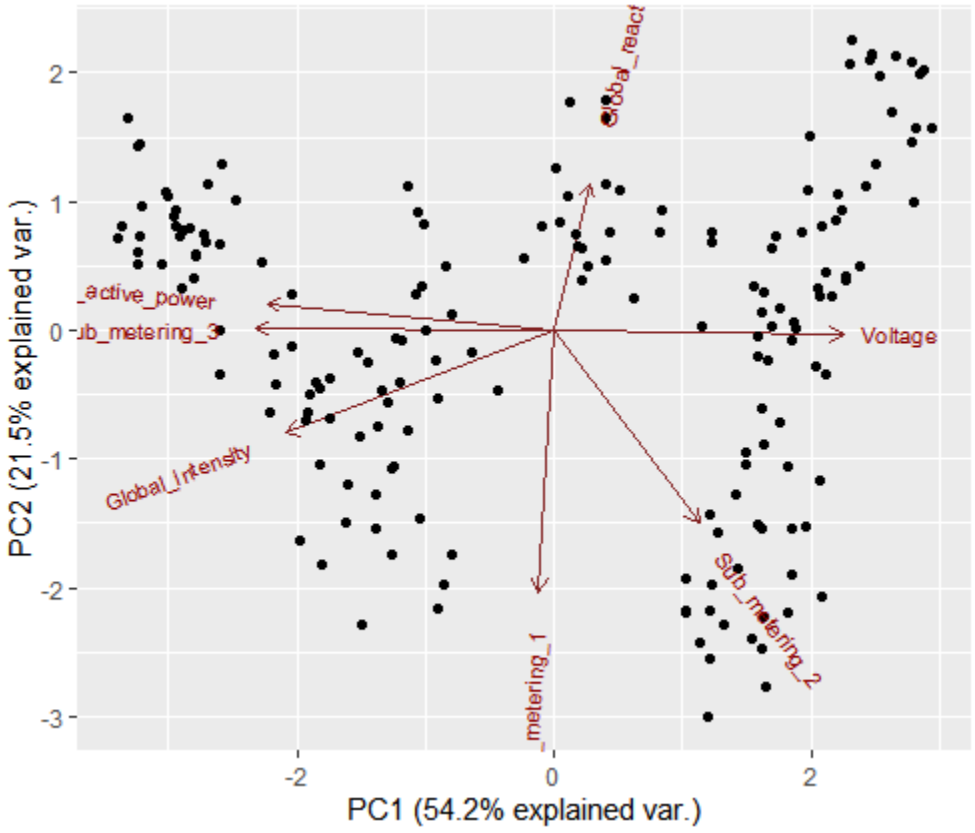

Figure 2: The PCA plot of the new dataset.

Figure 2 shows the underlying relationships of the data in the dataset and how each response variable contributes to the constructions of PC1 and PC2. We can see that the variation of PC1 is higher than PC2, and there are two major data clusters to the left and to the right of the figure, suggesting that more attention should be placed on PC1 rather than PC2, although PC1 only contains 54.2% of the information from the dataset. Each response variable vector when being projected to an axis represents the loading score of the corresponding

response variable for the PC corresponding to the axis. Among the 7 vectors, we can see that *Sub_metering_3*, *Voltage*, *Global_active_power,* and *Global_intensity* have the highest cardinalities with regards to the x-axis (PC1), which is in agreement with our previous result. From Table 2, we can see that the 4 features with the highest loading scores are *Sub_metering_3*, *Voltage*, *Global_active_power,* and *Global_intensity*. And since a positive loading means that a variable contributes to some degree to the PC and a negative loading means that not having the variable contributes to some degree to the principal component. The highest loading score magnitude means that the variable plays a larger part in contributing to the principal component [5]. Since the variance between the 4 features with the highest variance are all within 40-50, which does not represent a big difference between the features, we decided to try a few feature combinations such as *Global_intensity* and *Global_active_power* or *Voltage* and *Global_active_power* and the results were subpar compared to *Global_intensity* and *Voltage*. Therefore, we decided to choose *Global_intensity* and *Voltage* as the two main variables we will work with for training and testing of HMM, which makes sense as we are measuring the electricity consumption.

4.2. The choice for partitioning the data into train data and test data

In this project, we chose Monday and Tuesday from 13:00:00 to 16:00:00 respectively as the weekday of interest for train data and test data respectively. Furthermore, to evaluate the performance of our model on unseen data we partitioned our train and test data in an 80-20 ratio where 80% of the samples are used for training, and the remaining 20% are used for testing. Our decision to choose the mentioned time window was based on careful analysis of the dataset, our knowledge from our previous assignments, and the availability of data during those time periods. To evaluate the performance of the HMM models, it is important to choose the proper time frames that are representative of typical activity patterns and minimize any noise in the data, such as overlapping events or irregular patterns of activity.

4.3. Choices of HMM's Number of Hidden States

The number of hidden states in HMM is a parameter that determines the complexity and flexibility of the model. Choosing the right number of states is essential to avoid overfitting and underfitting. In this project, we use Bayesian Information Criterion (BIC) to choose the right fit. The models with the lower BICs are considered the best fits. The following code shows the function to find the best candidate models as it trains various multivariate Hidden Markov Models on the train data with different numbers of states. *find_best_models()* function uses the *fit()* function from *dempixS4* package to estimate the parameters of HMM. it takes the model as an argument and returns the fitted object model which contains the values that are used to calculate the log-likelihood, BIC, and, AIC. Then the function extracts the minimum BIC found in these models and uses it to filter out the models with the best (minimum) BIC as well as removing the Null variable at this point. In the end, it stores the model with minimum BIC and returns them.

After multiple trials with different ranges of numbers of states, we have concluded that a model with 10 states is the best model. This decision was made based on several factors that we explore in the next few sentences. Models with less number of states gave us less accurate results (higher log-likelihood) which was an indication of an under-fitted model. As one can observe in figure 3, the difference between the log-likelihood of the train and test models is less in a model with 10 states (77) than in other models with lower BIC. Making this model a better fit for our data despite its inferior BIC compared to models with higher numbers of states that are overfitting the model to the given dataset.

n_states	bic	train	test	d1	d2	d3
7	30346	-119	-233			
8	25820	-100	-234	-245	-409	-245
9	21248	-81	-217	-240	-458	-240
10	18450	-69	-146	-223	-538	-223
11	15028	-54	-139	-195	-648	-195
13	8654	-26	-116	-189	-674	-189

Figure 3: Comparison of the models with different states

train_best_models	list [1]	List of length 1
[[1]]	list [4]	List of length 4
model	S4 (depmixS4::depmix.fitted)	S4 object of class depmix.fitted
loglik	double [1] (S3: logLik)	-76.47477
bic	double [1]	20358.35
nstates	integer [1]	10

Figure 4: The best model with 10 states found by find_best_models function.

```
find_best_models <- function(data_resp, num_states, ntime) {
  # Calculate the number of time windows (number of observations)
  sequence_num = nrow(data_resp) %/% ntime

  # Create a list of observation lengths
  ntimes <- rep(ntime, sequence_num)

  # create a list to save the models, loglikes, and BIC values
  train_results <- list()

  # looping through the number of states
  for (i in num_states) {
    model <- depmix(list(feat1~1, feat2~1), data=data_resp, nstates=i,
                    family= list(gaussian(),gaussian()), ntimes=ntimes)

    # fit the model by calling fit
    fmodel <- fit(model)

    # avoiding NULL models
    if (!is.null(fmodel)) {
      loglik <- logLik(fmodel) / sequence_num
      bic <- BIC(fmodel)

      # Save the results
      train_results[[i]] <- list(model=fmodel, loglik=loglik, bic=bic,
nstates=i)
    }
  }

  # Select the best performing models based on BIC and overall fit on train
data
```

```
  valid_models <- Filter(function(x) !is.null(x$bic), train_results)

  # Choosing the best minimum BIC as the best BIC
  best_bic <- min(sapply(valid_models, function(x) x$bic))

  # Find models with minimum BIC
  best_models <- lapply(valid_models, function(x) if (x$bic == best_bic &&
!is.null(x$model)) x)
  best_models <- Filter(Negate(is.null), best_models)

  return(best_models)
}

# Choose the range of number of states for the models
num_states <- 6:10

# Calling the function to find the best models
train_best_models <- find_best_models(train_resp, num_states,
window_sample_num)
```

4.4. Interpretation of the Log-likelihood Results

The log-likelihood is a parameter that shows how well the model fits the dataset. Here we are evaluating the performance of the selected models by comparing their log-likelihood, so that the higher the log-likelihood, the better the fit of the model to the data. The *get_best_model* function in the provided code calculates the log-likelihood for each candidate model and selects the one with the highest log-likelihood as the best model. The function takes 3 arguments, the list of the selected models, the response data, and the number of states. Inside the function, we use *forwardbackward()* from the depmixS4 package to find the log-likelihood for each mode, and we store them in the vector. From this vector, the model with the highest log-likelihood is returned.

```
get_best_model <- function(best_models, data_resp, ntime) {
  # Make a vector to save the loglikes
  log_likelihoods <- vector("list", length = length(best_models))
```

```
  for (i in seq_along(best_models)) {
    log_likelihoods[[i]] <- calculate_likelihood(best_models[[i]],
data_resp, ntime)
  }

  # choosing the maximum loglike as the best one
  max_log_likelihood <- max(unlist(log_likelihoods))

  # finding the model with maximum loglike
  best_model <- best_models[[which.max(unlist(log_likelihoods))]]
  logLikRet <- list(testLogLik = max_log_likelihood, trainLogLik =
best_model$loglik)

  return(list(model=best_model, logLikRet=logLikRet))
}
```

5. Reinforcement Learning Paradigm

5.1. Overview

Reinforcement learning is a machine learning method that involves rewards or penalties while training to enable optimal decision-making based on previous experiences. One key characteristic of reinforcement learning is that it involves an “agent” exposed to an environment. This agent can be in different states, and depending on what type of action the agent performs, the agent can change states accordingly. Another important characteristic is that depending on the action the agent does, we can present them with a reward or penalty, to encourage or discourage certain behavior, which is established by the policy. This reward and punishment system often results in delayed feedback because the agent needs to learn from the previous experience. Another characteristic of reinforcement learning is that the timing or the sequence of the actions and the choices between reward and punishment are very important, therefore decisions are made in sequential order. [6]

5.2. Advantages compared to the classical machine learning approach

This project is focused on anomalous detection of household electricity consumption for a period of 3 years, thus it would be advantageous to use reinforced learning instead of classical machine learning. Although the training for reinforcement learning is longer and more tedious in the beginning, one advantage of using it instead of classical machine learning is that once the model has been trained successfully, it normally performs better than the classical machine learning models, as it is better at judging the data after being trained with the reward and penalty system. Another advantage is that we could make use of the presence of anomalous data, and use the reward and penalty system to train the models better. For instance, every time a model finds anomalous data from the time series data set, we reward it, so the model learns to focus on finding anomalies instead of hitting normal data. Also, another advantage of reinforcement learning is the flexibility it offers, as it is better at handling cases where the data is incomplete, which is true for the dataset being used in this project. In those cases, reinforcement learning models would be able to decide what to interpret from the N/A data, based on their previous experience. Lastly, Zhang et al (2022) [7] showed that time series anomaly detection via reinforcement learning has an overall better performance than classical machine learning. With that knowledge, and from the above reasons, we have enough evidence to conclude that using reinforcement learning would be of advantage for this project.

6. Discussion

6.1. Problems encountered and solutions

One of the few things we struggled with was choosing the right number of states. We initially overfitted the model by using a higher number of states than necessary and failed to notice it until we used the same model on the datasets with anomalies. We then backtracked and unfortunately under-fitted the model due to our belief at the time that the more variation we see in our results, the more accurate our results are which is not the case.

Another problem we struggled with was choosing the right features for our model. When we plotted our model using the PCA components, we noticed that there were components that were perpendicular to each other and others that were in completely opposite directions. Therefore, we had to experiment with different features to find out which feature sets are the best to use for our HMM model.

7. Conclusion

7.1. Results

For this project we performed a Principal Component Analysis on a time series dataset to choose a set of feature variables where we then performed a multivariate Hidden Markov Model training and testing.

From the training and testing of the HMM model, we obtained that in final run the Log Likelihoods of our model with 10 states, BIC of 20358.35 and Log Likelihood of -76.47477, tested on the 3 given datasets with injected anomalies are the following:

Dataset_with_Anomalies_1 → -210.5856

Dataset_with_Anomalies_2 → -558.8563

Dataset_with_Anomalies_3 → -210.5856

These results indicate that while Dataset_with_Anomalies_1 and Dataset_with_Anomalies_3 have less anomalies and are rather identical, Dataset_with_Anomalies_2 has a larger amount of anomalies indicated by the much lower Log Likelihood. (Illustrated in *Figure 5*)

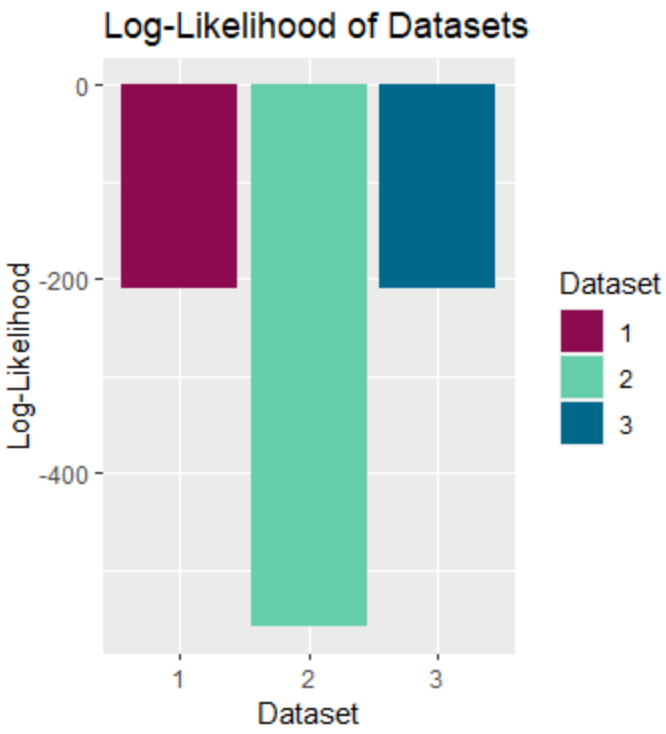

Figure 5: Log-likelihood of datasets with anomaly

7.2. Lessons learned

For this project, the main lessons we learned are the uses and application of Principal Component Analysis and Multivariate Hidden Markov Models in time series analysis. PCA is a very important tool that we could apply in the future to simplify our dataset when we work with large amounts of time series datasets that involve multiple variables. And Multivariate Hidden Markov Models are extremely useful when we want to predict some trend in a certain dataset. Moreover, finding the correct number of states is critical to avoid overfitting and underfitting.

On the other hand, we learned skills such as troubleshooting, as we encountered a few problems while working on the project, we tried various methods to reach a good solution, and using trial and testing and then analyzing the result was the best way to reach a good conclusion. Anomaly detection in time series datasets proved to be slightly difficult as

sometimes, we are unsure if the actual anomalies observed are anomalies or just a random occurrence.

Lastly, from the Reinforcement learning tutorial, we learned there are improved and more optimized ways to work with large time series datasets for anomaly detection, which would be interesting to apply in the future.

8. References

[1] Lohninger, H. (n.d.). *PCA*. Principal component analysis. Retrieved March 29, 2023, from http://www.statistics4u.com/fundstat_eng/cc_pca.html

[2] *A step-by-step explanation of principal component analysis (PCA)*. Built In. (n.d.). Retrieved March 29, 2023, from https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[3] StatQuest with Josh Starmer. (2018, April 2). *StatQuest: Principal Component Analysis (PCA), Step-by-Step* [Video]. YouTube. https://www.youtube.com/watch?v=FgakZw6K1QQ

[4] Scree plot. (2022, August 19). In *Wikipedia*. https://en.wikipedia.org/wiki/Scree_plot

[5] Understanding scores and loadings - David T. Harvey∗ Bryan A. Hanson (2022) https://cran.r-project.org/web/packages/LearnPCA/vignettes/Vig_04_Scores_Loadings.pdf

[6] Developer, A. S. K. S., Author: Brendan Martin Founder of LearnDataSci, Satwik Kansal Software DeveloperSoftware Developer experienced with Data Science and Decentralized Applications, & Brendan Martin Founder of LearnDataSciChief Editor at LearnDataSci and software engineer. (n.d.). *Reinforcement Q-learning from scratch in Python with Openai Gym*. Learn Data Science - Tutorials, Books, Courses, and More. Retrieved March 29, 2023, from https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

[7] Zhang, J. E., Wu, D., & Boulet, B. (2022). Time Series Anomaly Detection via Reinforcement Learning-Based Model Selection. *ArXiv*. /abs/2205.09884