Term Project Report

CMPT 318 - Spring 2022
Group 16

Authors
Bright Wu (301417539)
Ivy Zhu (301355788)
Kevin Tang (301357455)

Abstract

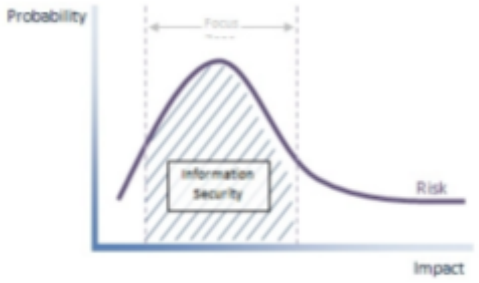
The increasing integration of computerized supervisory control systems in public utilities infrastructure has given rise to greater vulnerabilities to malicious disruptions from advanced persistent threats. This report explores methods of anomaly-detection based intrusion detection to counteract the influences of such threats. In the context of electrical grid data, it begins with the process of Principal Component Analysis (PCA) to designate the relevant response variables for a anomaly analysis, and uses optimally trained Hidden Markov Models to calculate the likelihood of example anomalous data, so as to demonstrate how such techniques may discover a manifestation of a threat in the form of anomalous statistics. The working process of this demonstration is supported by the R code output and graph found in the body of this report.

Table of Contents

1. Introduction

The growing application of computerized automation technologies has provided significant improvements in the reliability and efficiency of industrial operations. This application is especially noticeable in the field of public utilities, as demonstrated in the examples of power grids and water supply networks. With the mounting integration of computerized “smart” control, these utilities have become increasingly reliant upon the integrity of their supervisory control systems, which now projects greater attack surfaces for threats as an effect of said integration. Given the potential adverse consequences of control system failure due to malicious actions, it is of interest for methods to be explored on how to efficiently locate and identify appearances of such threats so that the risk of failures are minimized.

Based on the Figure “correlation between the probability of a risk occurrence and its potential impact”, what can be inferred is that when we need to locate and identify threats it is most effective to spend more effort on the areas of risk that are most likely to occur. While we may be concerned about risks that are highly unlikely to occur but could have a serious impact, we should not spend a lot of time addressing such risks. Therefore, in analyzing and resolving the data in this report, for the nuisance data, we also prefer to derive the nuisance data that are most likely to occur.



Correlation between the probability of a risk occurrence and its potential impact

The likelihood of risks with serious implications increases with the continued advance in technology. One category of these risks are called "Advanced Persistent Threats"(APT), which are difficult to locate and have potential to cause lasting damage if not swiftly located and eliminated in a system. There already exists a range of tools to defend against APT such as polymorphic techniques and moving target defense. In this demonstration, we use an example among the "Anomaly-Based Detection Methods", that is, mitigating risks and potential threats by
identifying anomalous data in system operations.

We proceed through three stages in this demonstration to accurately identify anomalies in data sets. In feature engineering, we need to select a specific subset of response variables from a given dataset and infer the impact of the variables on the relevance of the data by evaluating the variables in the original model that are informative to the model. In this stage, we employ PCA as our method of analysis. In Hidden Markov Model training, the data are scaled by the corresponding variables found in the previous stage and the continuous data are divided into separate data sets to facilitate our search for patterns in the data. In addition, we divided the data into a training group for training Hidden Markov Models and a test group for calibrating whether the trained model is valid. After this, we examine values of Log-likelihood and Bayesian Information Criterion (BIC) to obtain the optimal number of state parameters in the ideal Hidden Markov Model. In anomaly detection, we apply the trained Hidden Markov Model to identify the given anomalous datasets, and by comparing their log-likelihoods relative to the testing set to prove that they are indeed anomalous.

2. Feature Engineering

The first step to anomaly detection is to choose a specific subset of response variables from the data set, and only with reference to these variables will the models be eventually constructed to infer whether an entry in the total set of observations is normal. This choice is produced as the conclusion of Principal Component Analysis (PCA), which calculates how much influence each variable has over the correlation of data points with one another.

2.1. Principal Component Analysis

To efficiently process the data of the given context and begin PCA, the data set itself must first be normalized to avoid disproportionate variances due to the magnitude of its feature values, so feature scaling, a data pre-processing method, is performed on the raw data. It is necessary because the process of PCA is sensitive to the range and magnitude of feature values because it maximizes variance, and as such it would incur a loss of result accuracy if it is executed on unscaled features of the raw data.

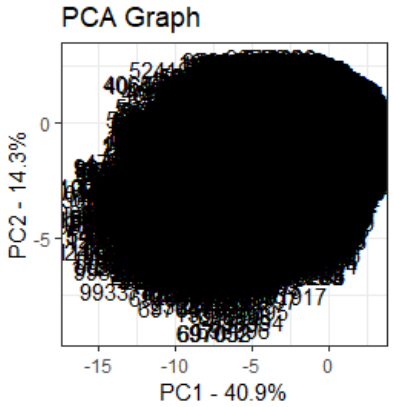
The feature scaling is done through the line: `df <- scale(df[c(3:9)], center = TRUE, scale = TRUE)`. This would balance the non-time variables in the raw data set "df" so they weigh equally during the later processes.

The analysis then begins the PCA. The process groups individual entries of the input data set into correlated clusters through linear transformation and "produces linear combinations of the original variables …, also known as principal components" (Holland). These principal components are then used to generate axes that show the distribution of variance in each principal component. Precise details of this technique are not included in this report for the purpose of convenience due to the complex mathematics involved.

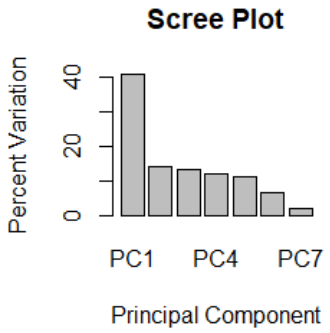
After omitting empty entries in the raw data set, the entire PCA process is performed automatically in R through the line: pca <- prcomp(df, center = TRUE, scale = TRUE). It automatically means centers, scales, and executes PCA on the raw dataset "df", then places the results in "pca".

```
> 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
```

From the summary of the PCA results, it can be seen that the principal components PC1 and PC2 constitute the two most weighted components in terms of their respective proportion in overall variance.



Through accessing the "sdev" (standard deviation) component of the PCA results, the percentage proportions of overall variance in each principal component can be obtained with the formula: $sdev^2/sum(sdev^2)*100$. These percentages are shown graphed as a Scree Plot.



2.2. Interpreting Loading Scores

To determine the most relevant response variables to be used in the anomaly modeling, the series of loading scores from the results of PCA is needed. Loading scores are "weights for each original variable when calculating the principal component" (Lohninger). "A positive loading means that a variable correlates positively with the principal component; a negative loading indicates a negative correlation" (Holland). A high positive or negative loading score of a variable indicates that said variable contributes strongly to the principal component. They are obtained with the following lines of R code.

```
> loading_scores <- pca$rotation[,1]
> scores <- abs(loading_scores)
> scores_ranked <- sort(scores, decreasing=TRUE)
> top_2_attributes <- names(scores_ranked[1:2])
> print(top_2_attributes)
[1] "Global_intensity"    "Global_active_power"
```

In the above code, the loading scores are extracted from the rotation component from the PCA results. The absolute value of each loading score is then calculated and the set of absolute values are ranked in descending order. The two most significant loading scores in terms of magnitude are taken to determine the optimal response variables. Given that the two most significant loading scores are of the attributes (columns) **Global_intensity** and **Global_active_power** in the data set, the subset of **[Global_intensity,**

Global_active_power] is chosen from the full set of variables for the training of Hidden Markov Models in the later stage of this anomaly detection process.

3. Hidden Markov Model Training and Testing

This method of anomaly detection relies on Hidden Markov Models (HMMs), which are "powerful statistical tools for modeling generative sequences that can be characterized by an underlying process generating an observable sequence." (Blunsom). The nature of the data set as a time series of changing latent states (individual electricity usage activities) with only record of observations (global power measurements) is highly compatible with such models. As such, the anomaly detection method can employ the training of optimal Hidden Markov Models to calculate the normality of each entry in observed data and eventually locate anomalies through any deviations from this calculated normality.

3.1. Data Set Modifications

Before the training of Hidden Markov Models can begin, the raw data set must be modified to enable an accurate result. The data set is first feature scaled on its chosen subset of response variables, "**Global_intensity**" and "**Global_active_power**", it is necessary because the machine learning algorithm of depmix, the R tool that will be used to train the Hidden Markov Models in this section, is sensitive to the range and magnitude of feature values, as such its algorithm would incur a potential loss of result accuracy if it is performed upon unscaled features. This application of feature scaling is similar to the case in PCA, but with fewer variables involved.

```
# scale the attributes global_intensity and global_active_power in the table
data_set$Global_intensity = scale(data_set$Global_intensity)
data_set$Global_active_power = scale(data_set$Global_active_power)
```

A time window is then applied unto the scaled data set with the below lines of R code.

```
# choose time window(7:00AM - 9:00AM) on Tuesday
data_set$Date = as.Date(data_set$Date, "%d/%m/%Y")
data_set$weekday <- weekdays(data_set$Date)
data_set$Time = paste(data_set$Date, data_set$Time)
data_set$Time = as.POSIXlt(strptime(data_set$Time, " %Y-%m-%d %H:%M:%S"))
data_set = filter(data_set, data_set$weekday=='Tuesday' &
                    hour(data_set$Time)>=7 &
                    hour(data_set$Time)<10)
```

It replaces the entries of the "Time" column in the data set with a standardized and combined R datetime object, then filters the entire set down to entries that occur after 7:00 AM and before 10:00 AM on Tuesdays. This is the chosen time window for this demonstration, and its purpose is to divide the continuous data set into a collection of separate samples so that patterns can be found by depmix's algorithms.

Finally, the data set is further divided into two distinct groups: the training set and the testing set. The training set acts as the reference data for the training of the Hidden Markov Models, and each model learns from the patterns of this set; the testing set is the data used to gauge the validity of the resultant models. Because the model learns more effectively with a larger data set, it is designated to be twice the size of the testing set in this demonstration. As such, the base data set, a time series of 3 years, is divided in the following manner: year 1-2 for training set, year 3 for testing set.

The below lines of R code performs this partition. (Note: year 2006 is trivial)

```
# split data set into training set(year 1-2) and test set(year 3)
training_set = subset(data_set, year(data_set$Time)==2006 |
                        year(data_set$Time)==2007|
                        year(data_set$Time)==2008)
testing_set = subset(data_set, year(data_set$Time)==2009)
```

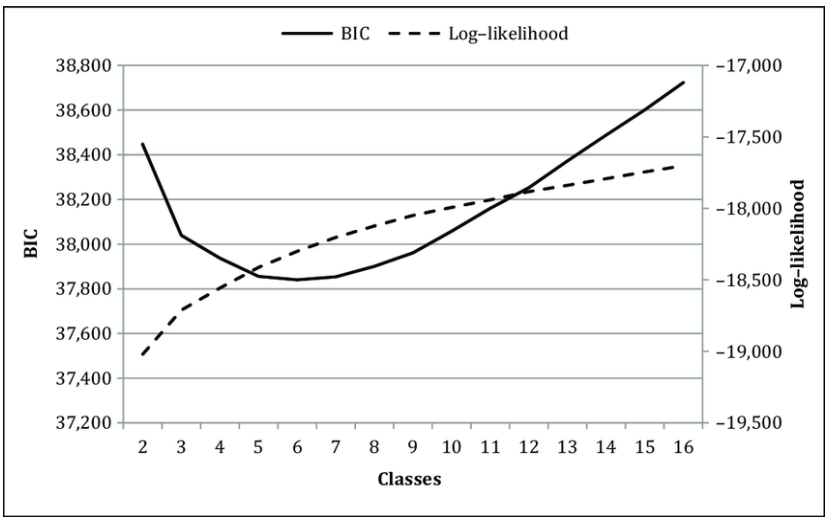
This completes the necessary modifications and the resultant data sets are ready for Hidden Markov Model training.

3.2. Optimal States Parameter

In this context, the quality of prediction of each trained Hidden Markov Model is indicated by two variables, the Likelihood and the Bayesian Information Criterion (BIC). The Likelihood is $P(O|\lambda)$, which is the probability of a sequence of observations given a particular model; the BIC score is a metric that measures the accuracy and quality of a Hidden Markov Model that "penalizes by adding (state) parameters to the model" (Maydeu-Olivares and García-Forero 190-196), that is, it disfavors overfitting Hidden Markov Models with high accuracy and high number of state parameters.

"The higher the value of the log-likelihood, the better a model fits a dataset." (Zach) "Given a data set, a researcher chooses either the AIC or BIC, and computes it for all models under consideration. Then, the model with the lowest index is selected." (Maydeu-Olivares and García-Forero 190-196) Therefore, a lower BIC score and a higher Log-likelihood (modified form of the likelihood score for better interpretation) value is desired.

The below graph shows that there exists an intersection between the output curves of Log-likelihood and BIC score as the number of Hidden Markov Model state parameters (classes) increase. Because the BIC score tends to stop decreasing and turn to increase beyond a certain number of Hidden Markov Model state parameters, the value of the number of state parameters at which the monotonically increasing Log-likelihood intersects the decreasing BIC score before it turns to increase would be the optimal number of state parameters to have in the ideal Hidden Markov Model for purposes of accurate anomaly detection. This would ensure that the resultant model has the highest Log-likelihood balanced against the lowest BIC score.



Comparison between function curves of BIC score and Log-likelihood vs. number of state parameters (Neuts et al. 793-808)

This optimal number of state parameters is found using R code with a trial and error method: an increasing number of state parameters are placed in each training iteration of the Hidden Markov Model until it reaches the threshold where the output Log-likelihood value is first greater than the output BIC score and the BIC score goes in the "knee" and gradually ceases to decrease.

The training of each iteration of Hidden Markov Model performed with the below lines of R code.

```
nt <- rep(c(180), times=107)
model <- depmix(response = list(Global_intensity ~ 1, Global_active_power ~ 1),
                data = training_set,
                family = list(gaussian(),gaussian()),
                nstates = n,
                ntimes = nt)
fitModel <- fit(model)
```

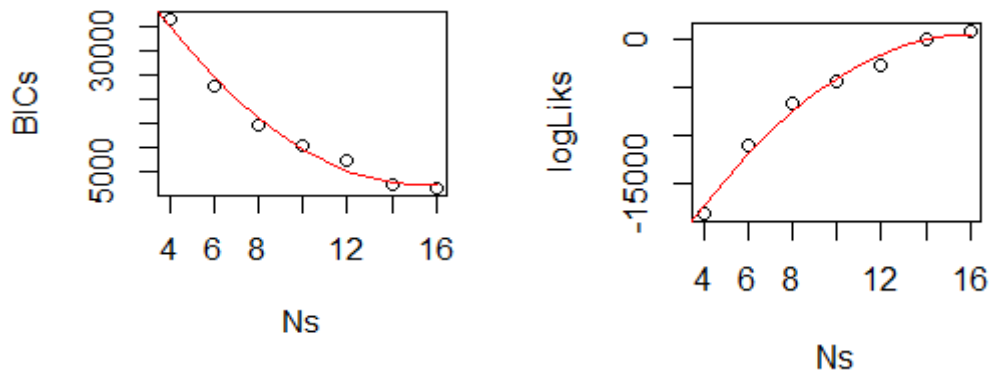
Here, the total amount and individual length of the set of separate time series is defined for the "ntimes" parameter, this corresponds to the earlier operation of dividing the data set into individual time windows. Then, the depmix method is called to train the Hidden Markov Model on the training data set with designated "nstates" (number of state parameters)

using the Gaussian Process, which is the more efficient process in this context. After optimizing the base model with the “fit()” method, a fully trained Hidden Markov Model is thus produced. The “logLik()” method and the “BIC()” method can then be called on the resultant “fitModel” to check if its values of Log-likelihood and BIC have reached threshold.

In the case of this demonstration, the output Log-likelihood and BIC score at states = 15 is respectively 255.5218 and 2290.81, while the output Log-likelihood and BIC score at states = 16 is respectively 1513.121 and 120.9104. The Log-likelihood is last smaller than BIC score on states = 15, therefore the optimal number of state parameters for the ideal Hidden Markov Model of this data set is 15.

```
> fitModel_1 <- fit(model_1)
converged at iteration 155 with logLik: 255.5218

> print(logLik(fitModel_1))
'log Lik.' 255.5218 (df=284)

> print(BIC(fitModel_1))
[1] 2290.81


> fitModel_2 <- fit(model_2)
converged at iteration 279 with logLik: 1513.121

> print(logLik(fitModel_2))
'log Lik.' 1513.121 (df=319)

> print(BIC(fitModel_2))
[1] 120.9104
```

This is also where the BIC score begins to flatten and cease decreasing, as shown in the below graphs. While further increases in the number of state parameters may improve Log-likelihood or BIC score, there would be significant diminishing returns due to the increasing complexity of the resultant model.



3.3. Comparing Train Data Likelihood and Test Data Likelihood

Having obtained the optimal number of states parameters, the optimal trained Hidden Markov Model can be applied unto the testing data set to show its effectiveness. This requires the transfer of model parameters (A, B, π) from the trained model onto the empty frame of a new model based on testing set data.

```
# construct shell HMM on testing set
nt <- rep(c(180), times=48)
new_model <- depmix(response = list(Global_intensity ~ 1, Global_active_power ~
1),
                    data = testing_set,
                    family = list(gaussian(),gaussian()),
                    nstates = 15,
                    ntimes = nt)
# transfer optimal model parameters
new_model <- setpars(new_model, getpars(fitModel))
```

In the above lines of R code, a new Hidden Markov Model is constructed by calling the depmix method on the testing data set with the same parameters as the earlier example. The ntimes parameter is changed to accomodate the differing size of the testing data set. Then the set of parameters is extracted from the earlier fitted Hidden Markov Model and applied

onto the new model, this provides a fitted Hidden Markov Model based on testing set data that has trained parameters.

```
> print("Normalized Training Set Log Likelihood:")
[1] "Normalized Training Set Log Likelihood:"

> training_loglik = forwardbackward(fitModel_1, return.all=FALSE, useC=TRUE)[3]

> normalized_training_loglik = as.numeric(training_loglik) / nrow(training_set)

> print(normalized_training_loglik)
[1] 0.01326697

> print("Normalized Testing Set Log Likelihood:")
[1] "Normalized Testing Set Log Likelihood:"

> testing_loglik = forwardbackward(model_3, return.all=FALSE, useC=TRUE)[3]

> normalized_testing_loglik = as.numeric(testing_loglik) / nrow(testing_set)

> print(normalized_testing_loglik)
[1] -0.1853452
```

The Log-likelihood of the training set model and of the testing set model is shown above as calculated by the Forward-Backward Algorithm. The value is normalized by dividing it with data set size to account for the volume of data that each model has to process. The normalized Log-likelihood of the training data on the model is 0.01326697, and the normalized Log-likelihood of the testing data is -0.1853452, the absolute difference is 0.17207823. It is a reasonable deviation and shows that the trained Hidden Markov Model is accurate to the characteristics of the data set.

4. Anomaly Detection

After an optimal trained Hidden Markov Model is obtained, it can be applied onto any data to determine its degree of normalcy, as indicated by its Log-likelihood under the trained model relative to the Log-likelihood of the trained model with the testing data set as its data parameter.

4.1. Fitting Hidden Markov Model to Anomalous Data

Before applying the optimal trained Hidden Markov Model, any data set will have the modifications shown earlier done so that it is ready to serve as data parameter in the model, this includes feature scaling its relevant response variables and filtering it down to a specific time window. This is done in R through the below lines of R code in the context of three anomalous data sets.

```
a_data_sets = list(anomalous_data_set_1, anomalous_data_set_2,
anomalous_data_set_3)
for (n in c(1:3)) {
        # scale variables
        a_data_sets[[n]]$Global_intensity = scale(a_data_sets[[n]]$Global_intensity)
        a_data_sets[[n]]$Global_active_power =
        scale(a_data_sets[[n]]$Global_active_power)
        # choose time window(7:00AM - 9:00AM) on Tuesday
        a_data_sets[[n]]$Date = as.Date(a_data_sets[[n]]$Date, "%d/%m/%Y")
        a_data_sets[[n]]$weekday <- weekdays(a_data_sets[[n]]$Date)
        a_data_sets[[n]]$Time = paste(a_data_sets[[n]]$Date, a_data_sets[[n]]$Time)
        a_data_sets[[n]]$Time = as.POSIXlt(strptime(a_data_sets[[n]]$Time,
        " %Y-%m-%d %H:%M:%S"))
        a_data_sets[[n]] = filter(a_data_sets[[n]],
a_data_sets[[n]]$weekday=='Tuesday'&
        hour(a_data_sets[[n]]$Time)>=7 & hour(a_data_sets[[n]]$Time)<10)
}
```

To apply the optimal trained Hidden Markov Model onto the data set, it will undergo the same process performed on the testing data set earlier, that is to transfer the model parameters (A, B, π) of the optimal model onto a newly constructed Hidden Markov Model with the intended data set as its data parameter. This is done in R through the below lines of R code in the same context, and the Log-likelihood of each respective anomalous data set is calculated with the Forward-Backward Algorithm and normalized according to its data size.

```
# construct shell HMMs for the three anomalous data sets and transfer parameters
a_hmms = list()
ant <- rep(c(180), times=51)
for (n in c(1:3)) {
```

```
        a_hmms[[n]] <- depmix(response = list(Global_intensity ~ 1,
        Global_active_power ~ 1),
                data = a_data_sets[[n]],
                family = list(gaussian(),gaussian()),
                nstates = 15,
                ntimes = ant)
        a_hmms[[n]] <- setpars(a_hmms[[n]], getpars(fitModel_1))
        print("Log Likelihood:")
        print(as.numeric(forwardbackward(a_hmms[[n]], return.all=FALSE,
        useC=TRUE)[3])/nrow(a_data_sets[[n]]))
}
```

4.2. Comparing Test Data Likelihood and Anomalous Data Likelihoods

```
[1] "Log Likelihood:"
[1] -1.294783
[1] "Log Likelihood:"
[1] -1.294792
[1] "Log Likelihood:"
[1] -2.464998
```

The above outputs show that the three designated anomalous data sets, “anomalous_data_set_1”, “anomalous_data_set_2”, “anomalous_data_set_3”, have respectively produced normalized Log-likelihoods -1.294783, -1.294782, and -2.464998. In comparison to the Log-likelihood of -0.1853452 for the testing data set, they have respective absolute differences of -1.1094378, -1.1094378, and -2.2796528. This shows that these anomalous data sets have produced highly improbable observations when examined under the trained Hidden Markov Model.

The trained model is shown to be able to properly classify data sets with anomalies, and threat indicators in the form of such anomalies can be discovered using this method, providing a counter to the stealthy Advanced Persistent Threats that would otherwise be difficult to track down with conventional examination.

5. Conclusion

During the scope of our project, we encountered a few challenges revolving around the implementation of various anomaly detection steps: finding optimal response variables through PCA analysis and figuring out an efficient training model to test for anomaly detection. As we ran into some issues with debugging of our R code, we were not able to find proper loading scores of our PCA's. In addition, we had difficulty adjusting optimal parameters for the training of our HMM. As both optimal response variables and the training of our HMM constitute vital components of our project, we had to implement different methods to identify an optimal solution for each problem.

Through various methods of trial and error, the lessons we have learned through different implementations has enabled us to find optimal response variables within our data set that are most suitable for the training of our multivariate HMM. By learning how to train our HMM model, we were able to test the data and identify any anomalous behavior within the anomalous data set. As zero-day exploits are becoming increasingly common and may cause catastrophic damage to assets of any magnitude, anomaly-based detection methods such as HMM have the ability to discover zero-day exploits by identifying anomalous behavior and will be crucial for the foreseeable future.

Works Cited

Blunsom, Philips. “Hidden Markov Models.” *Citeseerx*, 19 August 2004, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.1016&rep=rep1&type=pdf. Accessed 1 April 2022.

Holland, Steven M. “PRINCIPAL COMPONENTS ANALYSIS (PCA).” *UGA Stratigraphy Lab*, 5 December 2019, http://strata.uga.edu/8370/handouts/pcaTutorial.pdf. Accessed 31 March 2022.

Lohninger, H. “Fundamentals of Statistics.” *Table of Contents*, http://www.statistics4u.info/fundstat_eng/. Accessed 31 March 2022.

Maydeu-Olivares, Alberto, and Carlos García-Forero. “Goodness-of-Fit Testing.” *In International Encyclopedia of Education*, 2010, pp. 190–196. *ScienceDirect*, https://doi.org/10.1016/b978-0-08-044894-7.01333-6. Accessed 1 April 2022.

Neuts, et al. “Market segmentation and their potential economic impacts in an ecotourism destination: An applied modelling study on Hokkaido, Japan.” *Tourism Economics*, vol. 22, 2016, pp. 793-808. *ResearchGate*, 10.1177/1354816616654252. Accessed 1 April 2022.

Zach. “How to Interpret Log-Likelihood Values (With Examples).” - *Statology*, 31 August 2021, https://www.statology.org/interpret-log-likelihood/. Accessed 1 April 2022.

Contributions

Bright Wu (301417539)

- R Code for stage one and stage three
- All feature scaling and time window filtering code
- Report Section: Conclusion
- Office hour / TA consultations and project intelligence
- Presentation slides

Ivy Zhu (301355788)

- R Code for stage one
- All feature scaling and time window filtering code
- Report Section: Introduction
- Presentation slides

Kevin Tang (301357455)

- R Code for stage two and stage three
- Report Section: Abstract
- Report Section: Feature Engineering
- Report Section: Hidden Markov Model Training
- Report Section: Anomaly Detection