Technische Universiteit Eindhoven University of Technology
Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Outlier Detection in Event Logs of Material Handling System

Master Thesis

Özge Köroğlu

Supervisors:
Dr. Dirk Fahland
ir. Hilda Bernard
ir. Koen Verhaegh

Eindhoven, July 2019

Abstract

Having a fast and well-performing process is desirable for businesses. To ensure efficiency, performance related problems regarding the process should be identified and necessary measures have to be taken to solve them. However, identifying problems is not always enough to ensure efficiency especially if the process is complex and consists of many components. In this kind of process, there is a need to detect the problem along with the exact process step and the reasons for the problem. In this way, the problematic steps of the process can be identified and fixed. One way of finding problems is by identifying outliers in each step. In this thesis, event logs from a baggage handling system are used to find the performance-related outliers in each step of the system using statistical approaches. Cases are detected as outliers if their duration significantly differs from the majority of the other cases. Each outlier is classified into one type of problem happening in the process. After each performance-related problem is detected, we find the problematic parts of the system that suffers from these performance-related problems more often. Finally, we visualize the outliers and problematic parts in the system to provide a simple and automatic way to grasp the performance-related problems happening in the system over time for the given event log.

Keywords: Outlier detection, event log extraction, outlier pattern classification, blockage detection, clustering, Performance Spectrum Miner.

Preface

This master thesis was submitted to Information Systems research group of the Mathematics and Computer Science department of Eindhoven University of Technology (TU/e) to complete my Big Data Management and Analytics Master study as a part of Erasmus Mundus program. I would like to express my gratitude to the people who helped me during my master project.

First of all, I would like to thank my supervisor Dirk Fahland for his guidance, patience, and immense knowledge during my thesis. Since the beginning of the thesis, he has provided me with valuable suggestions and constructive feedback. I also want to thank Vadim Denisov for the generous technical support I received from him during my thesis.

Secondly, I would like to thank Hilda Bernard and Koen Verhaegh for their help and valuable insights during my master thesis and internship. It was a pleasure working with them throughout the project. In our weekly meetings, all the discussions that we had were really helpful for me. Also, I want to thank my manager who provided me the opportunity to join his team as an intern.

Finally, I would like to thank my parents who gave me the opportunity to pursue a good education. Very special thanks to my loving boyfriend Tarik for his endless support during this period. He was always there for me when I needed. Also a big thanks to my dear friend Anna, whom I've worked side by side within this period. Her positive energy kept me motivated and she always cheered me up when I needed it. I appreciate all of your support during this thesis. Thank you.

Ozge Koroglu
Eindhoven, July 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introduction to the project. Section 1.1 will describe the problem leading to this master thesis. In Section 1.2, both academic and business motivations will be explained. The research questions will be introduced in Section 1.3. Then in Section 1.4, the followed methodology for the research will be summarized. Lastly, the structure of the thesis will be given.

1.1 Problem Context

A baggage handling system is a type of conveyor system used in airports for carrying baggage from check-in or arrived aircraft to a departing aircraft. Baggage handling systems, in general, are busy and complex systems. For example, one of the baggage handling system considered in this project is a busy system that carries 3.7 million bags per year. In major airports, bags from more than 500 flights are handled per day. In such big and complex systems, the risk to face performance-related problems is high. In terms of the baggage handling systems, the more load the system has the higher probability that part of the system either will stop working completely or will slow down due to some problems caused by the congestion in the system parts such as conveyor belts. Some parts of the system are not designed to be able to handle all the load and thus they break down easily in the face of increased bag load. There are also parts in the system that regularly face performance-related issues due to the fact that all the bags from other conveyor belts are merged into that part. In that case, the performance issues would be expected in high bag load. Also, in conveyor belts merging into the main one, we would expect a delay due to waiting time for diverting operations. However, just knowing all possible reasons for performance-related issues in the system, is not sufficient to detect in which situations and on which days these problems occur and how severe the problem is. All the bottlenecks in the system that slows down the process should be identified to ensure an efficient and smooth process. We should identify the parts, where recurring performance problems occur in order to allow stakeholders to take special measures about them to fasten the process. The other problem that has to be solved is to make available data more structured. In order to conduct worthy analysis, valuable insights should be obtained from the data that has an excessive amount of information available which is the main challenge for all the data science related tasks.

1.2 Motivation

This master thesis is conducted with a process automation company, therefore, it carries two objectives. The first objective is conducting a meaningful analysis for the company to help them identify problems in their system by providing a visual tool and interpreting the results. On the other hand, the second objective is coming up with a scientific method to detect and classify outlier behavior for a given environment. Both objectives will be explained in more detail in the following two sections.

1.2.1 Business Motivation

Performance problems of the baggage handling system cause bottlenecks and slow down the process. This slowness could eventually cause bags to miss their flights which is one of the worst scenario for a process automation company. If this becomes a regular problem, customers (airports) could switch to another provider for their system and due to that, the company could lose money but most importantly credibility about the reliability of their system. To prevent that, measures have to be taken in every step in the process starting at first designs for the system of the airport. After the system is installed and running at airports, data is regularly collected from almost all the components of the system. Having this valuable data available, one can draw what is happening in the system and identify problems related to it.

Systems sometimes suffer from abnormal behavior. This behavior could be caused by faulty component, maintenance, properties of items or simply load of the system. Detecting these kinds of patterns and pointing where they often occur is highly valuable information for companies. Interpreting these results leads to understanding where the problematic components are and under which conditions they are behaving abnormally. Therefore, to ensure an efficient process, we should identify the problems that slow down the baggage handling process.

1.2.2 Academic Motivation

The goal of the study is to identify performance problems by analyzing event data using process mining techniques. Detecting performance problems is achieved by detecting outliers in the system. Detecting outliers is one of the biggest challenges in event data analysis and process mining. Because as stated in [21], most of the time, modeling the data normality is a challenging task since there could be many possible normal behaviors observed in the data. Also, how to define the border between normal and outlier behavior is not always clear in some data sets since this definition is domain-oriented. For some domains, small deviations from the normal behavior would indicate an abnormality. On the other hand, for other domains, the deviation must be larger for a point to be labeled as an outlier.

Finding an appropriate algorithm and correct parameters to detect outliers is important for the study. The nature of the data and the purpose of the research play important roles in finding the correct algorithm. Because each existing algorithm has a unique effect on result and picking the wrong one could completely change the project. Depending on the algorithm, choosing the correct parameters to detect outliers is a challenging task since each parameter could heavily affect the results.

One of the objectives of the study is that identifying relevant patterns of outlier behavior to understand the types of problems in the system. Outliers are just the first step. If outliers occur based on a structured mechanism, they form patterns that can become features. Outlier analysis is therefore always interested in searching for patterns to allow structured reasoning. Moreover, we want to find a way to detect these outlier patterns automatically as a dataset is usually too large to find outliers manually. The Performance Spectrum Miner [10] was developed for visualizing performance and it was already useful in prior studies on the baggage handling system data where outlier behavior was visualized. Performance Spectrum Miner (PSM) is a visual analytics tool for event data that allows visualizing the performance of the process in each process step. In this way, various performance patterns can be detected.

Lastly, the visualization of outliers has always been an important aspect of data science. Because without a proper visualization, it is hard to understand where and why the outlier behavior happens. There is a chance that one component of the system is heavily affected by others. Hence, the abnormal behavior in one could be reflected in the other component too. Detecting this correlation is not always easy because outlier behavior could be happening due to the effect caused by

a sequence of other components. If the system is big and complex, calculating all possible correlations would be inefficient and might be impossible. This brings a need for a proper visualization tool for this purpose.

1.3 Research Questions

The general objective of the thesis is to find a way to automatically detect outlier behavior for the given system. To be able to achieve this objective, several research questions are stated below.

1. **Is it possible to have a general purpose technique that can automatically identify performance related deviations within the performance spectrum?**

   Using the Performance Spectrum as a starting point, one of the main goals of the project is to automatically identify abnormal behavior in different parts of the system. However, different kinds of abnormalities can be observed in systems. Firstly, the characteristics of these abnormalities should be detected before any further analysis. In event logs, outliers can be observed in different settings such as performance-based or workflow-based. In this analysis we are interested in the performance of the system, therefore, the duration of the bags between different parts of the system is the point of interest. On the other hand, since event logs are the source of data in this analysis, the time-series aspect gains importance. Consequently, a proper outlier detection method covering all these points should be found and implemented.

   The baggage handling system consists of many components that are specialized to handle different tasks. For instance, check-in desks allow bags to enter the system and baggage storage holds the bags until their flights. Therefore, the characteristics of these components are different, and this leads to observe different performance behaviors in each of these components. For this reason, performance-related deviations should automatically be detected in each component of the system separately.

2. **Can we find patterns of outlier behavior and which outlier patterns make sense for the system?**

   Finding just abnormalities is not enough to identify all the problems in the system. Outliers are in fact patterns in the data that do not conform with the normal behavior. They even can be divided further into a different group of patterns behaving similarly. Establishing repetitive patterns in the data enables us to understand what is happening in the system and what are the common problems that have been faced. But not every pattern is meaningful to investigate further. An outlier pattern can repeat itself often, but that does not guarantee that the system is suffering greatly from it. It could be normal for the given part of the system to have fluctuations. Therefore patterns obtained from the data should be investigated carefully before further analysis.

3. **Can all states of normal performance behavior for each segment and over multiple segments of the system be identified?**

   If the research questions 1 and 2 are answered, we have lots of features and outlier patterns. However, the baggage handling system is big and complex, we have a lot of data covering months and there is not always constant load on the system. Therefore, how can we understand whether certain problems frequently or rarely occur in the segments? The answer to this question would indicate the severity of the problem for the segment. Knowing the severity of the problem would allow us to differentiate the normal segment performance from the special ones. This differentiation needs to be understandable for engineers working in the system.

4. **Which context factors cause outlier behavior in the system and what is the effect of outliers on performance of the system?**

   After automatically detecting outliers and their patterns in the system, it is crucial to find out why these abnormalities happen and what actions can be taken to prevent it in the future. It is also important to know how these outlier behavior is affecting the performance of the system. The detected outliers and patterns should enable users to automatically find the hot spots of the system that often suffers from abnormalities. By looking at the behavior of points in those hot spots and the behavior around them manually, users would be able to find the reasons why abnormalities happen in those parts.

1.4 Methodology of the Thesis

This thesis mostly follows the CRISP-DM methodology which stands for the cross-industry process for data mining. It is a well-defined structure to plan ahead of the tasks for the project. The cycle explains the steps of this methodology.

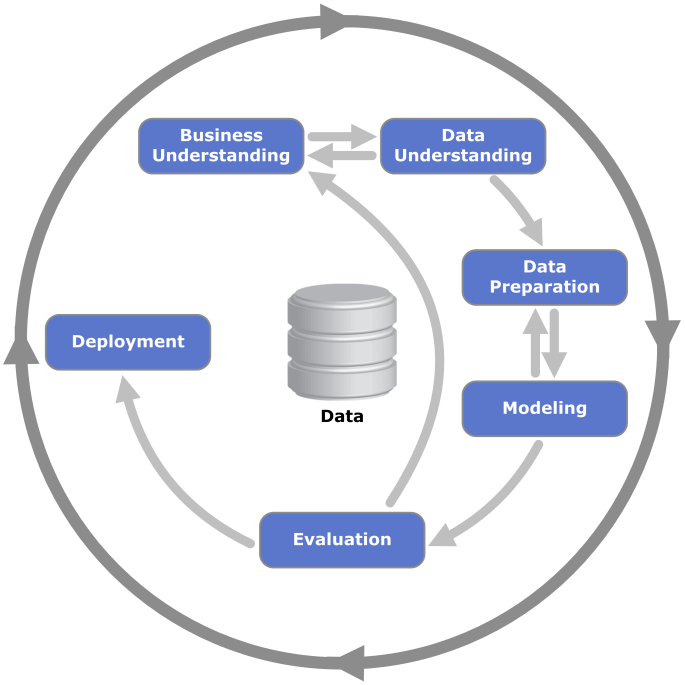


Figure 1.1: CRISP-DM cycle

The first step is business understanding. In this step, the main goal is to understand business objectives. The company this thesis is conducted with has a complex system that consists of many components mostly acting together. Therefore before doing any analysis, a detailed system understanding is necessary. Several weeks were dedicated to understand the nature of the business and what this thesis can accomplish for it. Business goals are listed and verified with the relevant people in the company. We summarize the results of business understanding in Section 3.1.

The second step is data understanding. The data for this analysis is coming from different sources and structured in a spread way. After setting business objectives, data is studied thoroughly and the decisions such as which information among many in the data is more valuable to use in the thesis were decided in this step. We summarize data understanding and give the detailed problem formulation obtained in Sections 3.2 and 3.3.

The third step is data preparation which usually takes up most of the time to accomplish. Initial data had to go through lots of transformations to be in the correct form for the outlier detection method. We summarize data preparation in Chapter 4.

The fourth step is modeling. Preprocessed data is used in the modeling step. Outlier detection, outlier classification, and segment behavior classification were conducted in this step. Sometimes when analysis got stuck, we revisited the data preparation step and made several changes to the data to make it compatible with algorithms used. By this step, we answer the research questions 1, 2, and 3. We explain the modeling process in Chapters 5 and 6.

The fifth step is evaluation. Data analysts who are responsible for the relevant system had tried our approach from the thesis and answered pre-defined analysis questions about it. They evaluated how the method is answering the analysis questions. They also compared the results obtained from this method to the ones they observe from their system and stated their opinions about the reliability of the proposed method. We summarize the results of the evaluation in Chapter 8.

The sixth and final step is deployment. We implemented the methodology as an extension of the Performance Spectrum Miner. Outliers are visualized in this framework and we made the relevant analysis available inside the PSM. Based on the positive outcomes after the successful evaluation, it was decided to deploy the extended tool within the company. We summarize the implementation in Chapter 7.

In the following, before we dive into business understanding, we explain external technical and theoretical concepts in Chapter 2. We conclude the thesis in Chapter 9 with an overview of contributions, limitations and future work.

Chapter 2

Preliminaries

This chapter aims to provide a basic understanding of all the external concepts and methods that are used in the project.

2.1 Event Logs

An event log can be seen as a collection of cases and each case corresponds to a sequence of events which is often called a *trace*. In event logs, each event refers to a case, an activity, and a point in time.[37] Event data can come from a variety of sources such as databases or simple data sources like CSV files and spreadsheets. The important point regarding the event log is that every row in the event log corresponds to an event. An event log must have:

- Case ID: Instance identifier of the event.
- Activity: Actions in the event log
- Timestamp: Date and time information of the event.

Besides these mandatory fields, an event log can also have other attributes related to the nature of the data; e.g. a resource or dimensions.

Table 2.1 represents an example event log where ID, timestamp and location correspond to Case ID, timestamp, and activity respectively.

ID	Timestamp	Location
1111111	21-05-19 13:44:54.948	X
1111111	21-05-19 13:45:42.760	Y
1111112	21-05-19 13:45:47.277	Y
1111112	21-05-19 13:49:21.290	Z

Table 2.1: Example event log

2.2 Performance Spectrum Miner (PSM)

Performance Spectrum Miner (PSM) can be defined as a visualization tool for event logs. An event log is put into the PSM and the PSM visualizes the flow of the cases over time. The main goal of the PSM is to display the changes in the performance of the cases over time. Several performance patterns can be discovered in the visualization created by the PSM.

Using the Performance Spectrum, the PSM divides the event log based on the observed flow between two process steps(*segment*) and shows them on the time axis. As stated by Denisov, Fahland, & van der Aalst, (2018) [10], segments are defined as follows:

Where A is set of activity names, $(a, b) \in AxA$ is called a segment describing a step from activity a to b. In the baggage handling system, a segment is defined as the movement of the bags from location a to b. Each segment is assigned a name usually in the format of a:b where a is the start activity and b is the end activity of the case.

The Performance Spectrum Miner divides the event log into the segments it defined. This process is called *the event log segmentation*. The event log segmentation process takes the event log that is given in Table 2.1 , it divides the event log into segments as displayed in Table 2.2 and finally it visualizes the segmented logs as in Figure 2.1.

ID	Timestamp	Activity	Duration (ms)
1111111	21-05-19 13:44:54.948	X:Y	48188
1111112	21-05-19 13:45:47.277	Y:Z	124013

Table 2.2: Example event log after the segmentation

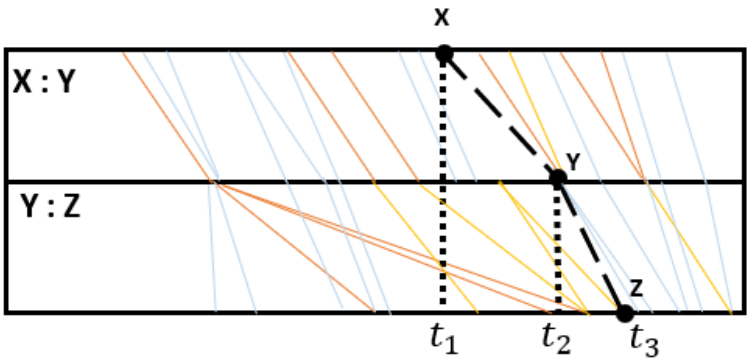


Figure 2.1: Example performance spectrum

For event log segmentation, the Performance Spectrum Miner translates the Table 2.1 into a set of sequences of time-stamped activities. Where each sequence $< (a, t_a), (b, t_b)... >$ describes the movement with its timestamp. For example the Table in 2.1 results in the sequences <(X:Y, 21-05-19 13:44:54.948), (Y:Z, 21-05-19 13:45:47.277)>

Per each case, the duration between locations is calculated from the timestamp information. This step is important because performance-related insights are crucial for the rest of the analysis.

In the paper by Denisov, Fahland, & van der Aalst, (2018) [10], the duration is calculated as

For each occurrence of a segment (a, b) in a trace $< (a, t_a), (b, t_b)..... >$ we can measure the time for the bag to reach to point b from the point a. The duration is calculated as $t = t_b - t_a$.

In Figure 2.1, two segments has been displayed. All the cases between activity X and activity Y is shown in segment $X : Y$. Similarly, all the cases between activity Y and Z belong to the segment $Y : Z$. Activity pairs (X, Y) and (Y, Z) have directly-follow relationship between them. Meaning there is at least one occurrence that X is directly followed by Y and Y is directly followed by Z in the event log. In Figure 2.1, one occurrence that activity X is followed by activity Y

has been highlighted. In this occurrence, activity X and activity Y have been executed at time t_1 and t_2 respectively. The duration for this occurrence between activity X and activity Y is calculated as $t_{(X,Y)} = t_2 - t_1$. Each line in the performance spectrum corresponds to a case and the inclination of the lines account for the duration of the case. The skewed lines indicate slower cases. If the line is vertical we can deduct that the case is executed almost immediately. The color of the line comes from the performance-classifier. The default performance-classifier shows the quartile that the case belongs.[9] Default PSM classifier is defined as follows:

- Color Blue : Cases that belong to 0-25% are assigned a blue color.
- Color Light-blue : Cases that belong to 26-50% are assigned a light-blue color.
- Color Yellow: Cases that belong to 51-75% are assigned a yellow color.
- Color Orange : Cases that belong to 76-100% are assigned an orange color.

Users can add their own classifiers. Based on the classifiers, the logic of the classification and class colors can change. In addition to these standard classifiers, there is also the possibility to label each case in a segment with a different class. In Chapter 7, we will use this to label cases in a segment based on what kind of outlier they are.

2.3 Outliers

In general terms, outliers are described as extreme values that are different from other observations in the data. Outlier points lie an abnormal distance from other data points. In other words, outliers diverge from normal behavior in the data. Many definitions for outliers exist, and some of these definitions are as follows:

Grubbs [18] defines outliers as

An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs

According to Hawkins, (1980) [22] outlier is

An observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism

Outliers may happen due to measurement, data entry, sampling errors or they are simply extreme points in the sample that causes the distribution to be heavy-tailed. Reasons for outliers can be explained in a more detailed way:

1. Measurement errors: Measurement errors could either be random or systematic errors. [35] Random errors are statistical variations in the data that have no pattern. They may occur due to the imprecise measuring instrument. This is also known as a scaling error. Systematic errors, however, occur when measuring instrument leads to measurements with a consistent and repeatable error. These measurements are consistently wrong in one direction.

2. Data entry errors: These errors are mostly caused by human behavior. They can come in the form of errors in addition/subtraction of numbers, decimal mistakes, incorrect data formatting, and many more human-related mistakes.

3. Sampling errors: Sampling errors would occur if the used data was designed for another population or analysis and not suitable for the aimed purpose.

4. Extreme values: Lastly, the most important reason for outliers is when some extreme valued data points exist in the data. Being extreme is subjective and changes according to business insights, the aim of the analysis and patterns in the data. This kind of outlier points are not errors and they usually mean that the distribution has high skewness. Finding these kinds of outliers enables organizations to see the nature of their system. Also, it helps them to understand faulty/abnormal behavior so that they can find out the reasons behind it.

2.4 Outlier Detection Methods

In this section, we discuss several methods to detect outliers. Studies about detecting outliers in the data date back to 19th-century [12]. Since then, many outlier detection methods have been proposed. These outlier detection methods have been a topic of many researches. For instance, Hodge, & Austin, (2004) [23] provide a deep understanding of the outlier detection techniques in the fields of machine learning and statistics. According to Hodge and Austin, there are 3 main outlier detection approaches namely; *statistical approaches*, *neural network approaches* and *machine learning modelling approaches*. Another extensive study about the outlier detection has been conducted by Chandola, Banerjee, & Kumar (2009) [6] and the authors suggest that there are *classification-based*, *nearest neighbor-based*, *clustering-based* and *statistical-based* outlier detection techniques. We will describe the existing outlier detection methods in three main approaches like Hodge and Austin suggested. However, statistical methods carry more value for the research since we use a statistical approach to find outliers in the baggage handling system. Therefore, we will explain statistical approaches in more detail.

2.4.1 Statistical Outlier Detection Methods

Statistical approaches assume that the data follows some statistical distribution (usually normal distribution) and they fit a model to represent points on this distribution. Outliers would be detected if they do not fit the model. The first examples of using statistics in the outlier detection field were only able to find univariate outliers in the data set. [23] Another point about the statistical approaches is that they are suited for quantitative real-valued data sets.[23] If the data consists of ordinal values, these values must be transferred to be used in the statistical approaches. This would increase the processing time in the case of complex transformations. The statistical approaches can be divided into parametric and non-parametric techniques[6]. Parametric outlier detection techniques take into account the underlying distribution of the data while non-parametric techniques do not assume the underlying distribution. We will now discuss both parametric and non-parametric outlier detection techniques.

Parametric outlier detection techniques :

- **Box Plot Rule :** Box plot is one of the simplest statistical methods to detect both univariate and multivariate outliers. Box plots provide a visual representation that allows us to see the deviating points easily. Laurikkala, (2000) [28] uses the informal box plot to find outliers in the real world medical data set. Box plot describes the data visually by using its summary attributes such as its minimum value, lower quartile (Q1), median, upper quartile (Q3) and maximum value. Inter Quartile Range (IQR) is calculated by subtracting Q1 from Q3 (Q3-Q1). The box plot finds the outliers by labeling the points that lie beyond the upper and lower limits and these limits are defined as:

$$\text{upper limit} = \text{upper quartile} + 1.5 * \text{IQR} \tag{2.1}$$

$$\text{lower limit} = \text{lower quartile} - 1.5 * \text{IQR} \tag{2.2}$$

- **Grubbs Test :** Grubbs test is used to detect univariate outliers in the data.[19] The test assumes that the data follows a Gaussian distribution. The Grubbs test detects one outlier

in each run. The detected outlier is then excluded from the data and the test is executed again. To conduct the Grubbs test, we first have to calculate the Z-score for each point in the data. Z-score is calculated as follows:

$$z = \frac{X - \bar{X}}{s} \tag{2.3}$$

where
X = sample mean
s = sample standard deviation
The data point is detected as an outlier if,

$$z > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}} \tag{2.4}$$

where
N = Sample size
$t^2_{\alpha/(2N),N-2}$ = outlier threshold taken by a t-distribution

- **Z-Score Method :** Another parametric method to detect the outliers is called Z-Score. It uses the mean and the standard deviation to detect the outliers. For each point, the algorithm calculates how many standard deviations it is away from the mean. Using the Z-Score method could result in wrong outcomes for small samples since the maximum Z-score is at most $\frac{n-1}{\sqrt{n}}$. According to the book by Kreyszig [27], the Z-score of each point is defined as:

$$Z_i = \frac{Y_i - \bar{Y}}{s} \text{ where } Y \sim \mathcal{N}(\mu, \sigma^2)\,. \tag{2.5}$$

where
Y_i = sample mean
s = sample standard deviation

A common rule to detect outliers is labeling points whose Z-score is greater than 3 as outliers.

- **Modified Z-Score Method :**The common practice in the field of outlier detection is labeling range of the points between mean and points resting in three standard deviations around the mean as normal while labeling points outside of this range as an outlier. This method is called **Z-Score** and particularly, points are scored based on how much they deviate from the mean. However, this approach is dependent on the mean, and the mean is not robust statistics against outliers. The sample mean and the standard deviation are greatly influenced by outliers. Also, the distribution of the sample is assumed to be normal in this method, yet outliers are included in the sample are creating a tail that makes the distribution detract from the normality assumption. Therefore, a more robust method has to be chosen for outlier detection. Instead of using mean and the standard deviation, estimators like median and the deviation from the median would be calculated since the median is more robust to outliers in the sample. This method that uses median and deviation from the median is called **modified Z-Score method**.

  In the paper published by Iglewicz, & Hoaglin, (1993) [25] modified Z-score method is described as follows:

  For a univariate data set x_1, x_2, ..., x_n

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD} \tag{2.6}$$

  where:

x_i = Sample median

MAD = The median absolute deviation is calculated by taking differences between each point and the median of the sample. Then the median of the absolute differences gives the MAD.

$$MAD = median(\mid x_i - \tilde{x} \mid) \tag{2.7}$$

The constant 0.6745 is the 0.75th quartile of the normal distribution. This constant is selected because MAD converges to 0.75th quartile of the normal distribution.

A point in the sample is labeled as an outlier if $|M_i| > threshold$. According to Iglewicz, & Hoaglin, (1993) [25] the threshold should be 3.5. Which means points that have a higher modified Z-score than 3.5 will be labeled as outliers. This threshold, however, could be selected something different than 3.5 according to characteristics of the sample. But the selected threshold must be justified.

Non-parametric outlier detection techniques :

- **Proximity-based methods:** In proximity-based approaches, the spatial proximity of each data point is checked and points that deviate from the proximity of other points are labeled as outliers. For these methods, no assumption about the data distribution is required. According to Aggarwal, (2017) [5] the most common ways to define the proximity of the data points are as follows:
  - Cluster-Based: Cluster-based methods cluster similar points together. Data points that do not belong to any cluster are considered as outliers.
  - Distance-Based: Distance-based methods use the distance function between a point and each of its neighbor. If the distance exceeds a specific threshold then the point is detected as an outlier. **K-means** is the most applied distance-based outlier detection method. In the k-means method, the distance of a data point to its k-nearest neighbor is measured, and data points that have large k-nearest neighbor distances are detected as outliers. [5]
  - Density-Based: Density-based methods looks at the density of the data point and its neighbors. The data point is detected as an outlier if the density of it is significantly lower than its neighbors. **DBSCAN** follows a density-based clustering approach to find outliers.[13] The DBSCAN algorithm groups the data points together if they are close to each other. If the point is too far away from all of its neighbors then it is detected as an outlier according to the DBSCAN algorithm.

2.4.2 Machine Learning Outlier Detection Methods

In machine learning modeling approaches, a labeled training set is used to train classification models in order to distinguish outliers from the normal points.

- **Linear Regression Model:** According to Fox, (2008) [16], in simple regression, an outlier is an observation whose dependent variable value is conditionally unusual given the value of the independent variable. From this definition, outlier detection with a linear regression model is formed in two steps. [6] In the first step, the linear regression model is fitted into the data set. Secondly, for each data point, the residual is calculated to be used as criteria for outlier detection. The residual of the data point can be defined as parts of the point that is not explained by the regression model. The residual of the point is used as a score that measures the outlier strength of the point. According to Weisberg, (2005) [40], observations with large residuals are candidates for outliers. In order to successfully detect outliers, a robust linear regression model has to be selected because the presence of outliers in the training set greatly affects the performance of the regression model.

2.4.3 Neural Networks Outlier Detection Methods

The neural network approach has been applied to the outlier detection problem in multi-class and also in one-class approaches. [6] In a multi-class approach, firstly the neural network is trained on the labeled data and then each data point is given as an input to the neural network. If the neural network does not accept the input then the point is classified as an outlier. [8] One-class approach one the other hand, does not require labeled data. Neural network models require parameter tuning and they are difficult to train. Therefore, we will not use a neural network model to detect outliers in the baggage handling system.

2.5 Distance Measures

In order to calculate the similarity between two histograms later on the thesis, a distance has to be computed. Here, we investigate the distance measures that can compare two histograms.

2.5.1 Wasserstein Metric

Wasserstein metric is a distance function that is defined between probability distributions on a metric space. The metric is defined by Vaserstein in 1969 [38]. However, the name Wasserstein metric first appeared in the paper published by Dobrushin, (1970) [11]. On a metric space $\mathbb{R}^d$ Vasershtein introduced the metric $W(\mu, \nu) = \inf_{(X,Y)} \mathbb{E}(|X - Y|)$ where the inf takes all random variables X, Y with $X \sim \mu$ and $Y \sim \nu$. It aims to provide a way to compare probability distributions. Rubner [34] uses the Wasserstein metric to create Earth Movers Distance (EMD). This distance can be seen as the minimum amount of work required to transform one distribution into another where work is the amount of distribution weight that must be moved, multiplied by the distance it has to be moved. In more analytic form, [33] calculates the Wasserstein distance between two distributions u and v as follows:

$$l_1(u, v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| \mathrm{d}\pi(x, y) \tag{2.8}$$

where $\Gamma(u, v)$ is the set of probability distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are u and v.

Wasserstein metric is widely used to compare two discrete distributions such as histograms. The metric transforms one histogram to another with a minimum cost considering cross-bin information.[34]

2.5.2 Quadratic-Form Distance

Quadratic form distance is a cross-bin distance to calculate the similarity between two probability distributions.[20]. If P and Q are two histograms and A is the bin-similarity matrix, the Quadratic Form distance is calculated as:

$$QF^A(P, Q) = \sqrt{(P - Q)^T A (P - Q)} \tag{2.9}$$

Cross-bin information is obtained from the similarity matrix $A = [a_{ij}]$ where a_{ij} is the similarity between bins i and j.

2.5.3 Chi-Squared Histogram Distance

Chi-Squared Histogram Distance is a distance to compute the similarity between two histograms that have an equal number of bins. This distance only takes into account the difference between the corresponding bins, therefore, it is sensitive to distortions [41]. If x and y are the probability

distributions with random variables, i = 1, 2,..., n, the chi-square distance between these two histograms is calculated by [32]

$$\frac{1}{2}\sum_{i=1}^{n}\frac{(x_i - y_i)^2}{(x_i + y_i)} \tag{2.10}$$

2.5.4 Bhattacharyya Distance

Bhattacharyya Distance calculates the similarity between two probability distributions. This distance is bin-to-bin similarity measure and it has been proposed by Bhattacharyya in 1943 [2]. For probability distributions p and q on the same domain X, the Bhattacharyya distance is calculated as follows:

$$D_B(p,q) = -\ln\left(BC(p,q)\right) \tag{2.11}$$

where

$$BC(p,q) = \sum_{x\in X}\sqrt{p(x)q(x)} \tag{2.12}$$

2.6 Clustering Methods

Clustering means grouping the observations in the data in a way that each observation looks similar to all the others in the same group but significantly differs from all the observations that belong to different groups. Many different clustering techniques exist for data analysis. However, we will investigate the candidate clustering techniques for our project.

2.6.1 Hierarchical Clustering

Hierarchical clustering aims to build a hierarchy of the clusters. It usually represented with a tree-structure diagram named as a dendrogram. A dendrogram shows the arrangement of the clusters produced by the hierarchical clustering algorithm.[14] There are two types of hierarchical clustering. Agglomerative and Divisive clustering [30]

- **Agglomerative clustering:** In this bottom-up approach, each observation forms its own cluster first and then two closest clusters are merged until we obtain one big cluster that contains all the initial clusters. The agglomerative clustering requires parameters for the method to define the similarity between the clusters. One parameter for this purpose is distance metric and the other is the linkage criterion. Distance metric is used to define cluster proximity and various metrics could be used such as *Euclidean*, *Manhattan*, *cosine* or some precomputed distance. On the other hand, the linkage criterion defines a way to merge clusters. Such as *ward criteria* minimizes the variance of the merged clusters while *complete criteria* uses the maximum distances between all observations of the two sets. The agglomerative clustering also requires a *number of cluster* parameter to decide where to stop the algorithm. The algorithm stops when the desired number of the cluster is obtained.

- **Divisive clustering:** In this top-down approach, every observation starts in one big cluster and then step by step they are split until each observation forms just one cluster.

2.6.2 K-Means Clustering

K-means clustering aims to divide the n-observations in the data into k clusters where each observation is in the cluster with the nearest mean. The algorithm starts by selecting k centroids. These centroids are the mean value of all the observations in the cluster. After defining centroids, all observations are compared to all centroids and each observation is assigned to its closest centroid regarding the Euclidean distance. The algorithm stops when there is no observation left to assign to the centroids.

2.6.3 K-Medoids Clustering

K-Medoids clustering is very similar to the K-Means clustering in a way that both partition the data into k clusters. Contrary to the k-means algorithm, the k-medoids algorithm selects observations as centers for clusters. These selected points are called *medoids*. Medoids are the most centrally located points in the cluster. The K-Medoids algorithm is more robust to noise in the data than K-means. Because the mean is greatly influenced by the noise while medoid is robust to the noise.

2.7 Average Silhouette Score

For techniques like K-Means, K-Medoids, and Agglomerative clustering there is a need to predefine the number of clusters. The silhouette score helps to determine the optimal number of clusters for k-means, k-medoids, and hierarchical clustering.

The silhouette score measures the quality of clusters by checking each point's fit with its cluster. The average silhouette calculates the average silhouette of observations for each candidate number of clusters. The optimal number of cluster is defined to be the one that has the maximum average silhouette score.

The silhouette is calculated with the following steps:

For each data point i in the cluster C_i

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{2.13}$$

where $d(i, j)$ is the distance between observations i and j in the cluster C_i

$a(i)$ corresponds to the average dissimilarity of i to all other observations of its cluster. Then we calculate $b(i)$ as follows:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{2.14}$$

Where $b(i)$ corresponds to the neighbor cluster measured by calculating the mean distance of i to all the other clusters that do not contain it. $b(i)$ is selected to be the smallest mean which can be considered as the second-best cluster for the point i.

The silhouette score $s(i)$ for each observation i is calculated as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.15}$$

The silhouette score is interpreted as follows:

- $s_i = 1$: The observation fits its cluster perfectly.
- $s_i = 0$: The observation lies between two clusters.
- $s_i =$ -1 : The observation does not fit its cluster at all.

2.8 Related Work

In this section, we position our research into the existing research areas. For this purpose, we study the existing literature in the relevant research areas.

The research areas for this thesis can be classified as outlier definition in event logs, outlier detection and outlier pattern classification in the event logs. Hence, we will look for literature in these areas of interest.

Outliers in event logs can be in the form of workflow or performance patterns. Ghionna et al.,(2008) [17], Bose, Chandra & van der Aalst (2013) [3] and Bouarfa & Dankelman, (2012) [4] define deviations from the workflow as outliers. On the other hand, Teinemaa, Leontjeva & Masing, (2015) [36], Bautista, Wangikar & Akbar, (2013) [7], Berger, (2017) [1] define outliers based on the duration of the event. Our research focuses on finding performance-related outliers hence, we use the duration of the events to detect outliers in the baggage handling system.

There is extensive literature dedicated to outlier detection techniques. These techniques are summarized in papers by Hodge & Austin, (2004) [23] and Chandola, Banerjee & Kumar, (2009) [6]. These papers present a wide range of outlier detection techniques. However, we are only interested in statistical outlier detection techniques. Hence, we focus our research on finding an appropriate statistical outlier detection technique for our thesis. Also, since the thesis focuses on finding outliers based on the duration of the events, there is only one criterion which is the duration of an event. For this reason, we will look for univariate outliers in our thesis. Iglewicz & Banerjee, (2001) [26], Leys et al., (2013) [29] and, Vijendra & Shivani (2014) [39] provide robust techniques to find univariate outliers.

Outlier pattern classification forms an important part of the thesis. According to Hu & Sung, (2003) [24] after identifying possible outliers, it is often profitable to study the underlying reasons why they happen. Hence, they investigate outliers for certain patterns. Similarly, Fawzy, Mokhtar, & Hegazy, (2013) [15] detects and classifies outlier behavior in the sensor network to find the source of the problem. Pawar, (2014) [31], conducts a very detailed survey to investigate the type of outlier patterns for the time series data. According to the author, periodic outlier patterns can be discovered from the time series data.

Chapter 3

Problem Understanding For Outlier Detection

Section 3.1 explains the concept of the baggage handling system and indicates the performance-related problems regarding it. Also, in this chapter, the business questions that the thesis is trying to solve are listed. Later, the structure of the data from the baggage handling system is described in Chapter 3.2. We explain the business problem along with the steps to translate it into a technical data science approach in Section 3.3. In the end, a pipeline to solve the problem of detecting outliers in the baggage handling system is given.

3.1 Business Understanding

This research focuses on finding performance-related problems in the baggage handling system. First, we will focus on providing a clear definition of the baggage handling system together with the description of various performance-related problems regarding it. Then the business objectives will be given.

3.1.1 Baggage Handling System

The baggage handling system is a type of conveyor system used in airports for carrying baggage from check-in or arrived aircraft to a departing aircraft. Along its way, bags go through various process steps in different areas which are connected via conveyors. To execute the baggage handling processes, the system consists of many different physical equipments along with different applications to run the system. There are three main processes baggage handling system in airports handle:

1. Transporting baggages from the check-in counters to aircraft.
2. Transporting baggages to related aircraft during transfers
3. Transporting baggages from aircraft to baggage reclaim area.

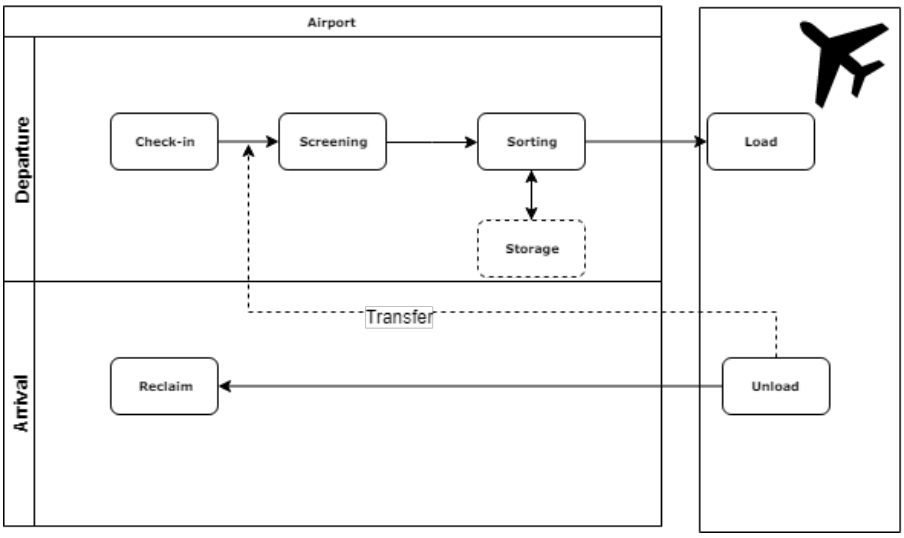


Figure 3.1: Airport Process Flow Diagram

The process flow diagram in Figure 3.1, shows the main processes of the baggage handling systems in airports. In a typical baggage handling system, bags are registered into the system at check-in counters and receive a unique ID that enables tracking them throughout the system. Then bags are screened and later sent to the sorting area. The sorting process can be described as sending bags to designated locations and if the bag is too early for the flight then it is forwarded to storage to wait for its flight. After the sorting process, bags are put into containers to be loaded into related aircraft. In the case of arrival, bags are treated differently regarding whether they are an arrival or transfer bags. If the bag is an arrival bag then it is sent to the baggage reclaim area for passengers to collect. If the bag is a transfer bag then it is sent firstly to the screening area and then through the sorting and finally to the related aircraft.

Figure 3.2 shows a physical representation of the baggage handling system. Bags enter the system at check-in counters and are loaded onto the conveyor belts. Several conveyor belts from check-in counters merge to go to scanners. After the scanning, bags are diverted into different conveyor belts to go to either storage area or to lateral to be loaded into aircraft. Transfer inputs are also forwarded into the scanner and then optionally to storage. In arrival, bags are put into conveyor belts that move into baggage reclaim areas and from there, passengers collect their bags. Bags are assigned a unique ID at check-in and that allows every movement of the bag to be recorded in the system with the help of sensors placed in the machines along with the conveyor belts.

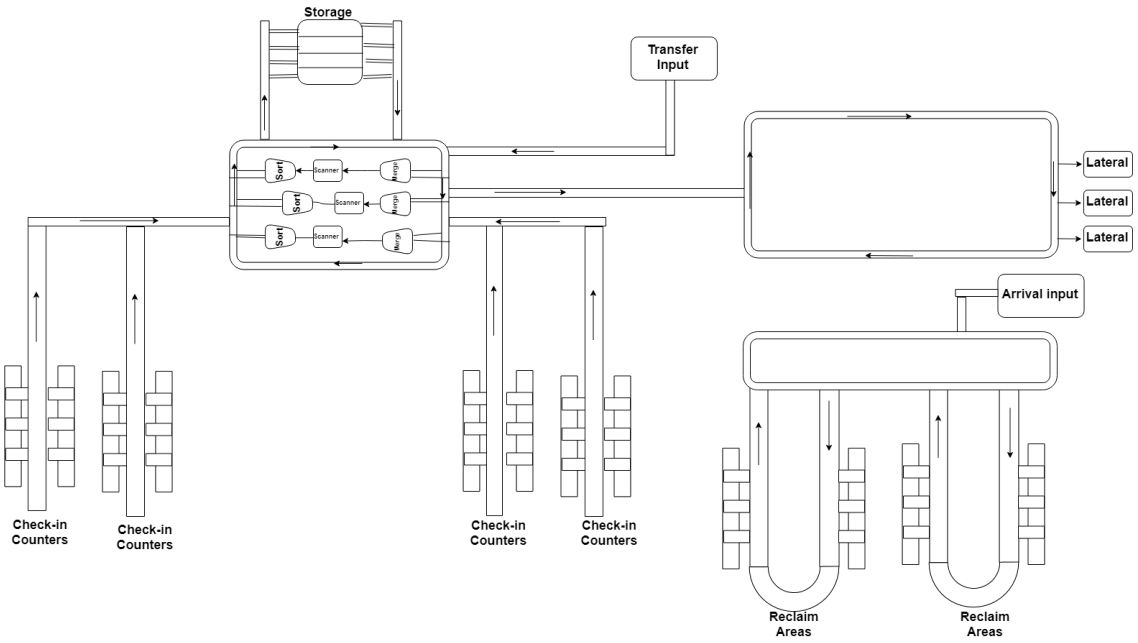


Figure 3.2: Physical Representation of Baggage Handling System

3.1.2 Problems in Baggage Handling System

The baggage handling system is a complex system and there could be many performance-related problems occurring in different components. The aim of this master thesis is to address possible performance-related problems regarding the equipment of the system by analyzing all the components. The most generic type of problem that could be observed in the baggage handling system is the state when bags are queued up and cannot go forward to the next part. To illustrate this problem we have to understand that the baggage handling system is mostly comprised of meters and sometimes several hundreds of meters of conveyor belts to carry all the items. Together with all the conveyor belts, the system resembles the road system. Just like on roads, this system might also suffer from *traffic jams*. The possible reasons for the traffic jams in the baggage handling system are as follows:

- **Load of the system :** Traffic jams in the baggage handling system occur mainly when there is a high baggage load in the system. For example in Figure 3.3, conveyor belts 1, 2 and 3 are merging to go to conveyor belt 4. We can observe that conveyor belt 4 is quite busy, and that is causing a *dieback effect* on merging paths especially in conveyor belt 1. The dieback effect happens when the later part in the process affects the behavior of past parts. Meaning, bags in conveyor belt 1 are not able to move through the conveyor belt 4 because load balancers on the system are preventing new bags to enter conveyor belt 4 since it has reached its capacity. This bottleneck of the system happens when the system receives more bags than it could handle. Bags passing through the conveyor belt 1, take more time to complete their journeys, and this could eventually lead those bags to miss their flight.

  However, baggage handling systems are designed to work even under high load without the dieback effect. Thus, if dieback occurs, the system is not working as intended. This kind of behavior can happen before or after the long public holidays such as Christmas or school holidays.

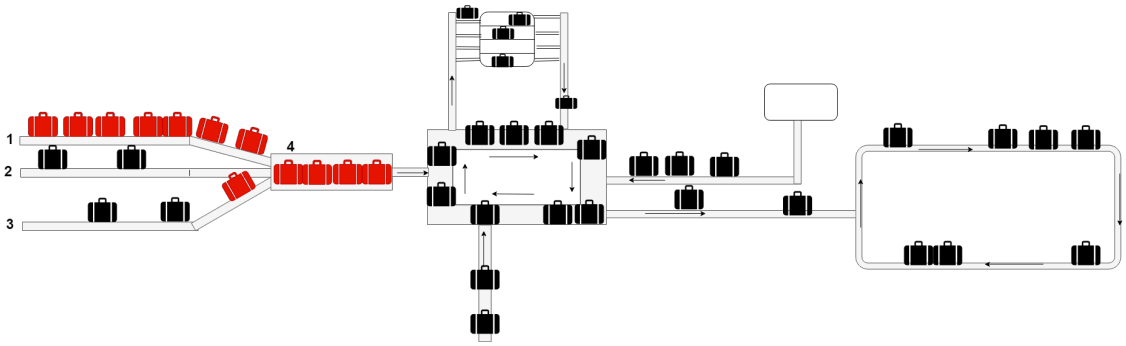


Figure 3.3: Scenario 1: High load of bags slowing down the system

- **Unavailability in system parts:** Sometimes the problem happens due to unavailability in one of the machines in the system. If the machine stops working for a while, all the bags that have to go through that machine will have to wait until the faulty component begins to operate normally again or they simply have to be forwarded to an alternative machine. To illustrate this, we can look at Figure 3.4, If the conveyor belt 4 stops working temporarily because of some mechanical accident, all the other bags trying to move forward to 4 will stop moving. They either will wait for the problem to be fixed in the belt 4 or they will be forwarded to alternative parts to belt 4. If there is not an alternative, manual intervention will be needed. This again is one of the common scenarios happening in the baggage handling systems.

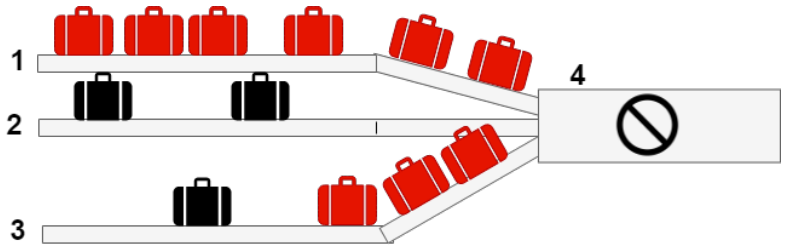


Figure 3.4: Scenario 2: Unavailability in System Parts

- **Item properties :** Another cause for the problem might be happening because of the item the system is carrying. Sometimes bags are not in ideal shape or size for example skiing tools or musical instruments. Also, bags with loose belts can cause problems in the system because the belts can get stuck in the system. Even though most of the time there is a specialized process in the baggage handling system to handle non-optimal shaped items, some objects might still be missed and put into the system with all the optimal shaped bags. This could lead the problematic bag to get stuck in some part of the system and cause other bags behind it to wait until the problem is solved. Figure 3.5 shows that accidentally one oversized baggage such as a musical instrument was let into the system and got stuck in one point. This creates a *blockage* behind that oversized object causing the system to slow down.

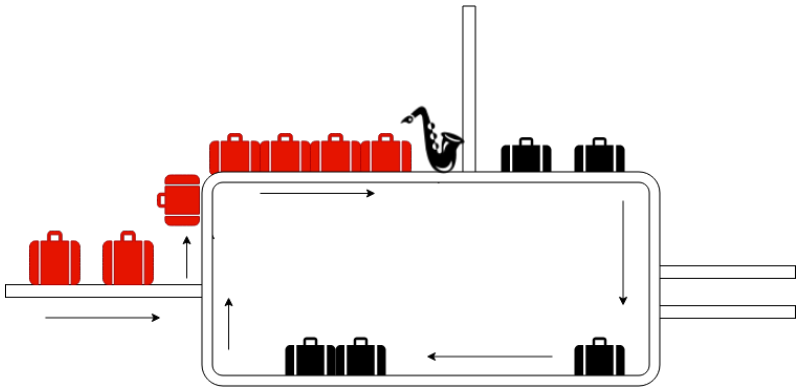


Figure 3.5: Scenario 3: Properties of item causing blockage in the system

All of these scenarios slow down the entire process by causing a traffic jam, and they might cause bags to miss their designated flight.

3.1.3 Business Questions

The main stakeholders of this thesis are the process and reliability engineers of the automation company whose responsibility is to ensure a smooth process for the baggage handling system in relevant airports. Stakeholders will use the findings from this thesis to answer the following business questions:

Is there a way to

1. Automatically identify abnormal behaviors in the segments of the system so that actions can be taken to improve/fix the system?
2. Automatically provide all the outliers that are similar so that it can be used to create a new extension of the system or apply a permanent solution?
3. Identify the segments of the system that experience more performance-related problems so that the source of the problem in the system can be diagnosed?
4. Categorize the abnormalities into actual incidents versus temporary one time problem versus recurring problems so that relevant teams and competencies are involved to get a solution?
5. Provide a ground truth about the types of outlier behavior the system commonly shows?
6. Understand which days are similar in the sense of outlier behavior?
7. Visualize the problems occurring in the system?
8. Automatically identify the properties that are leading to the abnormality, so the relevant action can be proposed or taken?

In the rest of the thesis, we will answer these business questions.

We will conduct the analysis for each segment of the baggage handling system. The reason for analyzing each segment individually over time instead of all segments together are the following:

- Characteristics of the segments: Segments have to be investigated individually because the characteristics of different components of the system are not the same. For example, some conveyor belts are 10 meters long while others are 250 meters long. Hence, the time needed to complete these paths certainly is not the same.

- Distance between segments: We cannot compare the duration of bags in the check-in counter conveyor with the storage part because they are far apart from each other in the baggage handling system. This distance between them causes the bag to be affected by various factors. Therefore, we would not expect to observe similar behavior for the bags that travel this long distance.

3.2 Data Understanding

This section describes the dataset used in this research. Firstly, data sources will be explained and then the dataset structure used in this thesis will be presented. Finally, the section ends with an explanation of how the dataset was converted into an event log and how segments, different areas of the baggage handling system, were extracted for the purpose of the analysis.

3.2.1 Understanding Data Sources

The dataset used for this research is coming from one of the biggest airports in Europe for one terminal that has around 230 check-ins and bags from approximately 470 flights are handled per day. There are also semi and fully automatic robots that are used as a part of this system to stack bags into pallets or containers that are loaded into flights. Data for the analysis was collected from the sensors that are installed on the machines of the baggage handling system. Every time a bag passes a sensor, the information of the exact passing time and the unique ID of the bag along with the exact location of the sensor is stored in the system. Many other attributes about the flight of the bag and machines of the system are recorded along with the information whether the bag made it to its designated flight or not. Figure 3.6 illustrates this recording process. Since the available data contains an extensive amount of information regarding the different areas of interests, the data source is available in the form of different reports and each of the reports is stored separately.

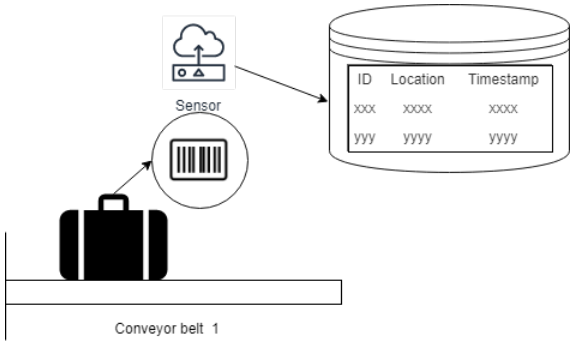


Figure 3.6: Sensor data recording

3.2.2 Nature of the Input Data

Data is available in the form of different reports that are named as *baggage movement reports*. These reports are in CSV format and each has multiple attributes about the process. The relevant reports for the thesis are created as a result of the streaming data coming from sensors in the system and they are stored on a daily basis. 6 months of historical data was provided for the thesis meaning 180 different daily reports were made available to use in this thesis. The data has a unique timestamp in milliseconds along with a unique identifier for the bag which is the same in every report. Also, the location of the bag in every movement is recorded in these reports. Table 3.1 shows the structure of the reports, reports have a unique bag identifier, a location of the baggage handling system where the baggage has passed along with the timestamp of this

movement. There are also many other attributes regarding the bag and the system present in these reports.

ID	Timestamp	Location	...
1111111	2019/05/21 23:21:01.058	X	...
1000000	2019/05/21 23:21:07.945	Y	...

Table 3.1: Example Report Structure

Baggage movement reports were provided in the form of different reports in CSV format. Each report has the timestamp information regarding different activities such as bag movement, the state of the machine, flight information or error status in the baggage handling system. Essentially, the type of reports is the following:

- Event report : This report gives information of events happening at particular time. For example, when a bag passes a sensor in the system, information about the bag along with the timestamp is recorded.
- State report : Data in this report is created periodically regarding to state of an object. For instance, the availability of a machine is recorded periodically in this report.
- Summary report : This report mostly contains aggregated information about bags and their bag/flight properties.
- Performance report : In this report, raised errors and exceptions are held about the components of the baggage handling system.

3.2.3 Converting Input Data to Event Logs

The *baggage movement reports* give events with location and time information per bag. That allows us to analyze segment-level performance as required by the PSM. However, to do that, we first need one event log that contains all the necessary events.

The conversion to event log is performed as follows; the unique bag ID becomes a case and timestamp of the movement becomes a timestamp of the event log while the location of the movement becomes the activity of the event log. Subsequently, each row in the source data becomes one event in the event log. Figure 3.7 shows the event log conversion. Attributes of the event log were defined as one case refers to one bag while timestamp is the time that events are recorded when bags pass sensors on conveyors. The activity attribute is the location of the sensor positioned in the machines.

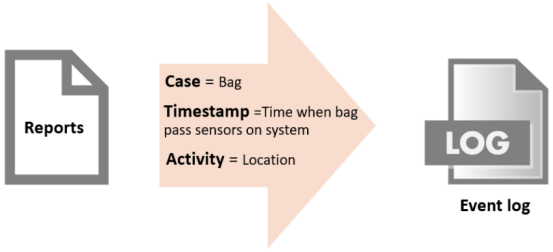


Figure 3.7: Event log conversion parameters

The obtained event log contains the information of unique bag ID, location of the sensor and the exact time in milliseconds when the bag passed through that location. Table 3.2 displays an example event log obtained from baggage handling system. The obtained event log contains all events of all the bags.

ID	Timestamp	Location
1111111	21-05-19 13:44:54.948	X
1111111	21-05-19 13:45:42.760	Y
1111112	21-05-19 13:45:47.277	Y
1111112	21-05-19 13:49:21.290	Z

Table 3.2: Example event log

The resulting log contains all events of the bags in the system. Ultimately for the case study, at the formation of the event log, 165,886,759 events regarding 6,020,431 different bags were recorded at 9,046 locations for a period of 6 months in the baggage handling system.

3.2.4 Event Log Segmentation

The problem with the obtained event log is, it only shows the movement of the bag inside the baggage handling system. Each row in the event log corresponds to one movement, and we only know the location of each bag in the system at a particular time. Knowing this is not enough to analyze the performance of the system due to the fact that performance-related issues cannot be identified only with the timestamp information. For each bag, we need to know the time spent between two consecutive locations on its trace to be able to measure its performance. For this reason, we have to break down the process into smaller related process steps.

At this point in the analysis, we have the event log and we need the performance information for each segment of the baggage handling system. In preliminaries, we presented the PSM which performs exactly this functionality (See details in Chapter 2). Hence, we use the PSM to obtain the segments and performance of the bags in these segments.

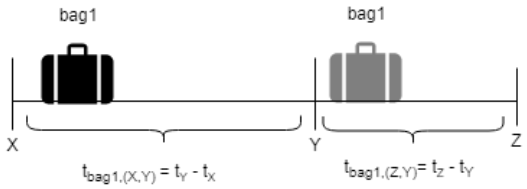


Figure 3.8: Example segmentation

In Figure 3.8, the segmentation and duration calculation is illustrated; the duration for one specific bag to complete the segments (X:Y) and (Y:Z) is calculated. The same calculation is performed for other bags as well. Chapter 2 explained how this calculation is done elaborately.

3.2.5 Challenges in Event Data/Segment Extraction

The Performance Spectrum Miner tool was used for both event log and segment extraction. However, the tool is not fit to process the entire 6 months of data at the same time. The initial data size was 8 GB and for this reason, the entire dataset did not fit into the memory of a single machine configuration. The solution was separating the data into smaller files and processing each of them separately. The data was split into 27 different files each of them covering 7 days of data. With this split, the Performance Spectrum Miner processed approximately 200-400 MB of data in each run. It eventually took more time, but the entire data was converted to event log and segments were extracted.

3.3 Problem Understanding

The business problem of the thesis is finding the problematic parts of the baggage handling system. The business questions raised in 3.1.3 are the questions the stakeholder wants to have answered. To answer those questions, we need the following data features:

- Classification of individual bags as outliers in a segment: As stated in the research question 1, we are interested in finding performance-related deviations in each segment. To do that, each bag should either be classified as an outlier or a normal-behaving bag per each segment.
- Classification of types of outliers and outlier patterns: The research question 2 states the need to find the outlier patterns in the system. Accordingly, each outlier bag should be assigned with one detected type of outlier behavior in each segment.
- Classification of segments: To answer the research question 3, we need to classify the segments to capture different performance behaviors.

3.3.1 Defining The Data Science Problem

We find the required data features with the following steps:

- **Finding the seasonality patterns in the data:** The domain of the baggage handling system is airports and airports operate under flight schedules. Hence, the behavior of the baggage handling system is subject to seasonality that comes from the flight schedules. We must analyze the seasonality patterns to allow detecting outliers that are robust against the seasonality. Seasonality detection will be described in Chapter 4.
- **Finding the outliers for each segment:** If the duration is calculated for each bag passing through one part of the system via other parts such as conveyor belt to scanner machine, we can figure out what is the duration that bags usually take to complete this path by comparing travel times of all bags for that particular part of the system. Having found the normal behavior for parts enables us to look for outliers too. The bags that deviate from the baseline that is formed by the majority behavior in the segment could be classified as outliers for that particular segment. Figure 3.9, shows the performance spectrum of **segment A**. In Chapter 2, we introduced the Performance Spectrum Miner and it explains how event data can be visualized over time. In the figure, each line corresponds to a bag and timestamp information is used in the time axis. From the figure, it is visible that bags normally complete this segment almost immediately because the lines are almost vertical. However, the pile of bags showed in dark blue, completed this part slower than the other bags because the lines have a smaller slope to the time axis which indicates a slow movement. We can deduce from this image that light blue bags behave normally on this segment while the behavior of dark blue ones can be considered as outliers. However, the definition of an outlier bag is a contextual subject for the baggage handling system, meaning it changes from one segment to another. But, in the entire system, the main criteria to classify outlier bags will be the duration attribute. The outlier detection in the segments of the baggage handling system will be explained in Chapter 5.
- **Classifying the outlier behavior into categories:** Some outlier behaviors form a pattern, and they should be classified together to allow applying a permanent solution for them. The process of classifying the abnormal behavior into the categories will be explained in 5.3.
- **Measuring the significance of the outlier behavior:** To comprehend how serious the outlier problem is, we need to create measures reflecting the importance of the observed outlier behavior in the system. These measures will be introduced in 5.4.
- **Classifying the segment behavior:** Since we are interested to know which component of the system experience more abnormal behavior and slow down the process of carrying bags,

we need to calculate the performance of the segment from the average baggage behavior that goes through the segment. To find which segments are showing a good performance and which are not, we need to find representative outlier behaviors from the historical data and rank these different outlier behaviors based on their performance. Classification of the segment will be explained in Chapter 6

- **Creating a dashboard:** Creating the dashboard is an important step because, for this thesis to be meaningful for the stakeholders, we need to visualize the results clearly. On this dashboard, stakeholders should be able to see all the problematic segments of the system, and the dashboard should also allow stakeholders to see all types of different outlier behavior that has been happening in their system. In this way, stakeholders can understand their system, and quickly identify the issues related to it. The dashboard creation will be described in Chapter 7

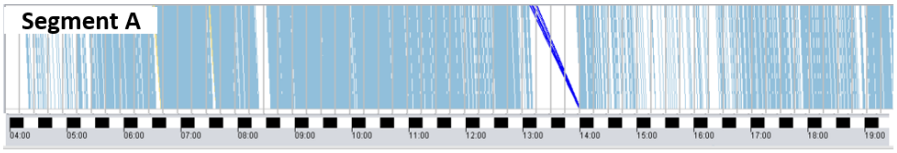


Figure 3.9: Performance Spectrum of Segment A

3.3.2 Detailed Problem Breakdown and Analysis Pipeline

Figure 3.10 shows the complete resulting pipeline of the thesis. We address all the data preprocessing steps in Chapter 4. Later, the outlier detection, the outlier classification and the blockage attribute detection methods are explained in Chapter 5, Then we describe the segment behavior classification in Chapter 6. Finally, the dashboard creation is described in Chapter 7.

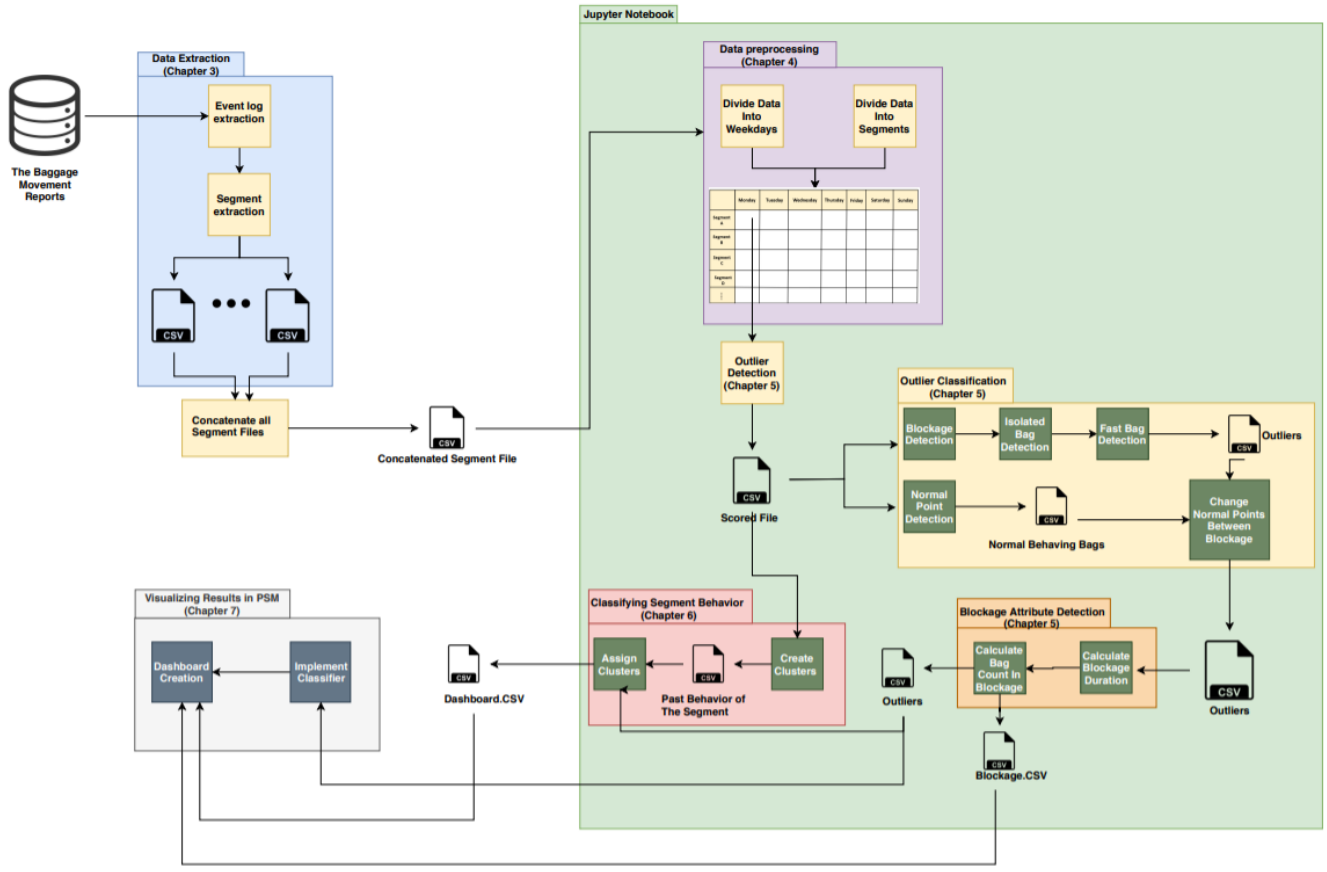


Figure 3.10: Pipeline of the project

Chapter 4

Data Preprocessing For Outlier Detection

This chapter will explain all the data preprocessing steps before the outlier detection method. The seasonality in the dataset is detected and handled by separating the dataset into seasonal patterns.

4.1 Splitting Data Based on Time Series Seasonality

Since the data comes from airport systems, it shows a certain pattern because there is a flight schedule that leads to moments of peak load in the system. Taking this as a baseline leads to the assumption that the human aspect shows some patterns such as people drop their bags at certain hours before their flight. We would assume that different people who have the same flight would act similarly and leave their bags at check-in desks around the same time. This also has to do with how airports operate. The check-in for the specific flight is only allowed for a certain period of time so it gives a limited time frame to people to leave their bags. Therefore, the bags for the same flight enter the baggage handling system around the same time and they are most likely processed together. The chart in Figure 4.1 displays the duration distribution of all the check-in counters in the system throughout one week. As seen in the figure, there is no continuous operation for the baggage handling system proving that the daily behavior exists. We see from this figure that there are 7 different groups each representing a day. And the duration of the bags variates for each day. For instance, check-in counters do not operate the same on Wednesday and Saturday for this week. On Wednesday we observe fewer deviations compared to Sunday indicating a weekly seasonality. To sum up, we can say repetitive human behavior as a result of a stable flight schedule causes seasonality patterns in data.

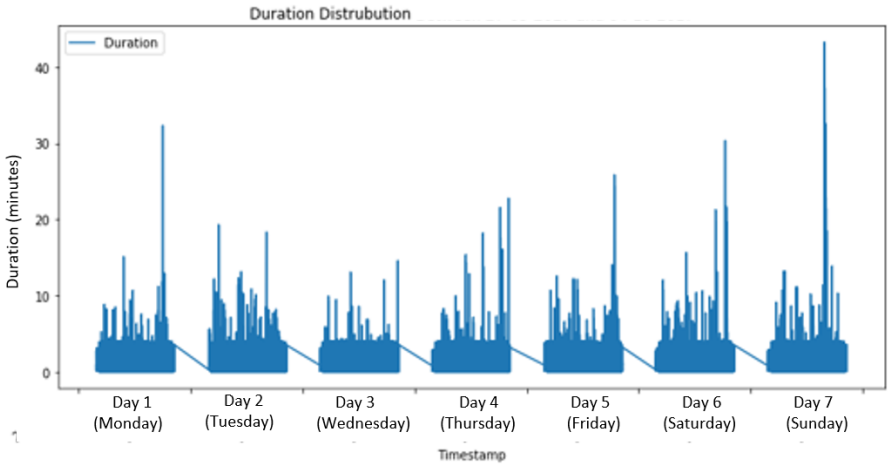


Figure 4.1: Duration distribution of the check-in counters for a week

Detecting seasonality patterns will be crucial for the analysis because, in order to find abnormal behavior first, we have to understand what is the normal behavior for each segment in the baggage handling system. Hence, it is interesting to find time series patterns in segments. For example, if we detect monthly seasonality, it means in the baggage handling system; every January is similar. In other words, each segment of the baggage handling system such as a conveyor belt will show similar behavior every January. If the segment does not behave similarly to its other January behaviors, then that particular segment can be considered to have abnormal behavior. Therefore, for the outlier detection method, it is important to divide the data based on different patterns and to find abnormalities taking into account the standard behavior observed in each of the patterns.

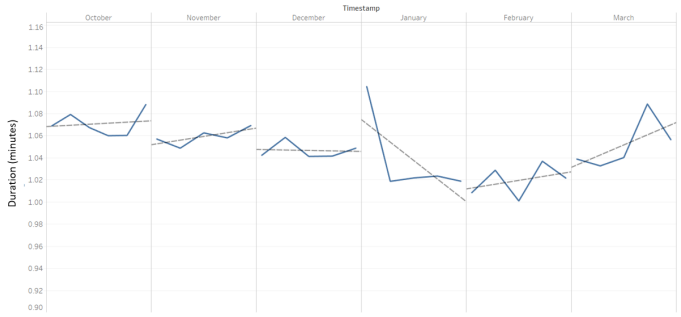
We aim to detect the existing time-series seasonality in the baggage handling system. For that purpose, we will look for various seasonality patterns in the data and detect one pattern that fits it. Then we will split the dataset accordingly.

The dataset was investigated for the possible time-series seasonality patterns.

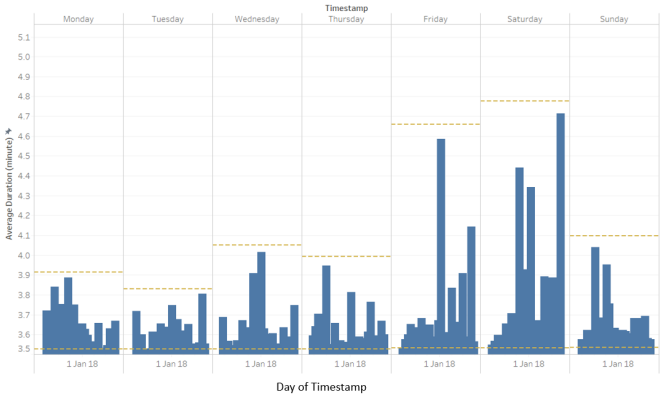
- **Yearly Seasonality :** Since the dataset only consists of 6 months of the event log, it is impossible to check yearly patterns.
- **Monthly Seasonality :** We could not detect a monthly seasonality in the dataset. Because if we look at Figure 4.2a, the change in the average duration the bags take is shown based on the week number. Since each month has 4 weeks, the chart shows the average duration of bags for each 4 weeks of the months that are present in the dataset. From this figure, it can be observed that there are no clear monthly seasonal patterns. For example, the first week of each month does not always behave similarly nor we do not always observe an increase in the average duration of the bags until the end of the month.
- **Weekly Seasonality :** Most of the bag movements can be correlated with the flight schedule and the flight schedule has defined for weekdays. Hence, each different weekday (from Monday to Sunday) could be investigated on its own under the assumption that flight schedules are either the same or similar for weekdays. For example with this assumption, we can say that each Monday has the same flight schedule therefore, we can expect similar bag behavior for Mondays. In fact in Figure 4.2b, it is clear that bags behave similarly according to which day of the week they enter the segment. This chart is showing the average duration of the bags per each date divided into the day of the week. We observe different days of the week differ from each other. Especially, Friday and Saturdays significantly differ

from Monday and Tuesdays. Fridays and Saturdays are more likely to show much higher time whereas Mondays and Tuesdays are more likely to have consistently lower times. For instance, all the Mondays in the data are between the average duration range between 3.5 and 3.9 minutes. On the other hand, Fridays and Saturdays have an average duration range between 3.5 and 4.7 minutes. Hence, from the chart in the figure, we see that even though the average duration range for different days of the week is not significantly different, we see slightly different behavior, especially for Friday and Saturdays. The higher average duration of the bags on these days could be considered normal since on these days we would expect more load at airports. To sum up, this chart is showing a slight difference between the days of the week that worth investigating.

- **Daily Seasonality :** As displayed in Figure 4.1, the system does not operate continuously for the entire day. We see before every new day, there is a resting period for the system and that indicates there is a distinction between days. Detecting daily seasonality means we will use the hours of the day as features to identify whether there is an outlier behavior on a particular day. However, the baggage handling system is a big and complex system and is dependent on many internal/external factors. Therefore considering that segments show daily seasonal patterns and dividing the dataset based on hours would be assuming too much. For example, if we say every day between 12:00 - 13:00, the bags in the particular segment behaves the same we would be saying that people arrive at the airport at the same hours and leave their bags in that specific hour every day which is mostly unlikely. Thus, we would not want to split the dataset hourly even if we detect daily seasonality. If daily seasonality is detected then, it can be used in the high-level analysis.



(a) Monthly seasonality in segment A



(b) Weekly seasonality in segment A

Figure 4.2: Seasonality patterns of segment A

Based on the insights about the flight schedule and the observed bag behavior, it has been

decided to split data based on days of the week to allow comparing each day of the week, such as Mondays, to each other as explained above.

4.2 Splitting Data Into Segments

To pinpoint where the performance-related problems are happening, the baggage handling system has to be broken down to its segments and each segment has to be investigated distinctively. Segment separation is a necessary step for the thesis as detecting normal behavior in each segment is one of the main motivations of the thesis. We are interested to see in which segment of the baggage handling system, there are more performance-related problems, taking into account the bags deviating from the normal behavior of the segment. With this aim, we split the dataset into smaller subsets corresponding to segments of the baggage handling system.

Figure 4.3 shows all the splits that have been done on the data. First, the data is divided into 7 subsets, and each of the weekday subsets is also divided into different segments. There are more than 2000 segments in the data therefore, the original data was divided into more than 14,000 subsets. The outlier detection method will run distinctively for each subset of segment and weekday.

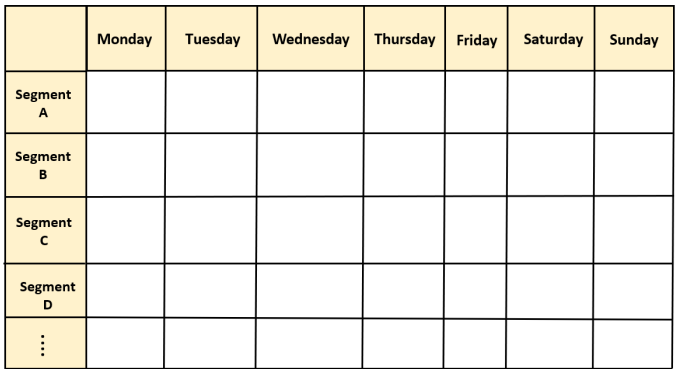


Figure 4.3: Dataset split before outlier detection

4.3 Eliminating Duplicate Timestamps

We have found that sometimes the dataset contains the same timestamps for multiple events for different bags. In theory, it is impossible for two bags to appear in the exact same place at the same time. However, this anomaly exists in the data. Data recorder positioned in the segments recorded multiple events with the same timestamp. Even the millisecond of record time is the same for the events. As this does not make sense physically, after a discussion with experts from the company, we decided to leave only one event for the specific timestamp and eliminate others.

In the next chapter, we will explain the method of outlier detection within each subset defined by a segment and weekday. Later in the chapter, we will describe the outlier classification by comparing the outliers in a segment and weekday.

Chapter 5

Outlier Detection in Event Logs of Baggage Handling System

This chapter explains all the steps regarding the outlier detection process in the event logs of the baggage handling system. Firstly, in Section 5.1 we revisit the business problem and explain the definition of the outlier behavior for the baggage handling system in the business context. Secondly, the method to assign scores to each bag to determine the outliers is introduced in Section 5.2. Then, the outlier pattern discovery is demonstrated in Section 5.3. Finally, Section 5.4 gives the description of the attribute detection process for certain outlier patterns.

5.1 Overview of the Problem Definition

The problem that the thesis is trying to solve is finding the problematic segments of the baggage handling system as explained in Chapter 3. To find the problematic parts of the system, we must measure the individual bag performance per segment. If the overall performance for all the bags passing through the segment is calculated, it would be equivalent to the segment performance. Thus, there is a need to score each bag in segments based on their duration in the segment. This score should measure how much the duration of a particular bag differs from the baseline which forms the majority in the segment. If the calculated outlier score of the bag is significantly different than the majority of outlier scores then this bag would be classified as an outlier. But the important point is that the same bag might travel with the same duration in a different segment without being considered as an outlier in that segment. Thus, the outlier detection should be in the level of segments.

Taking the average outlier score of the bags in a particular segment corresponds to the performance for that segment. For instance, if most of the bags have higher scores in one segment, in other words, if there are many outlier bags inside this segment then it would be an indicator of how bad this particular segment is performing.

5.2 Outlier Scoring Method

As explained in Section 5.1, the outlier strength of all the bags has to be measured according to their duration in each segment. Depending on the technique, bags should be labeled as an outlier or normal behaving in a segment level. Therefore throughout its trace, a bag can be labeled as an outlier in one segment and normal in the next. This allows us to see whether the bag had problems in each segment or it just suffered from a temporary problem in one segment. The scoring is performed distinctively per segment and weekday ensuring more accurate results since

in the smaller subsets the relevance to target is maximized.

There are several existing outlier detection techniques in the field. Selecting the correct algorithm for the business motives and the data has been a great challenge for this thesis. In Chapter 2, we discussed different types of techniques. Here are the requirements for selecting the right technique for our problem:

1. Lack of labeled data: Machine learning approaches could not have been used in this thesis since labeled data for the outliers is not present. In fact, outliers have never been detected in this system before. Since machine learning methods require a labeled dataset to train models, this approach is not convenient to use in this thesis.

2. Type of the outlier: Outliers could either be univariate or multivariate. Univariate outliers are the outliers that occur within one variable while multivariate outliers are a combination of at least 2 variables. In the case of outlier detection in the baggage handling system, outliers are found based on only a single variable which is the duration. Statistical approaches are useful to find univariate outliers, while proximity-based approaches are more suitable to find multivariate outliers.

3. Simple implementation: One of the initial goals of the thesis is to implement the outlier detection method to the Performance Spectrum Miner as an extension. Therefore, to avoid implementation complexity, we look for simple but also solid approaches such as statistical approaches.

For all the reasons stated, different statistical methods were researched. There are many statistical methods to detect the univariate outliers such as *Grubbs method, box-plot method, Z-score method* and *modified Z-score method.* Each of these methods is a candidate for the outlier detection algorithm in the baggage handling system. Hence, we execute the outlier detection with the candidate methods. Based on the visual investigation of resulting outlier points from various methods, we select the **Modified Z-Score Method** to detect outliers. Please refer to the appendix B to see the visual comparison of different outlier detection techniques. The modified Z-Score method is used to score outlier strength per each bag in a given weekday and segment. The algorithm for the modified Z-Score method can be described as follows:

Algorithm 1: Outlier Detection

Let B be the ordered list of all bags and T be the ordered set of timestamps belong to outlier points where $\forall b \in B$ and $r \in \{1, 2, 3, ..., n\}$ where n is the size of the set B and r is the index of each bag.

Input : $B_{weekday,segment}$, threshold

median ← median(B.duration)
difference_list ← []

for b_r *in* B **do**
    difference ← $abs(b_r.\text{duration} - median)$
    append difference to difference_list
MAD ← $median$(difference_list)
for b_r *in* B **do**
    difference ← $abs(b_r.\text{duration} - median)$
    Modified_Z_Score ← 0.6745 ∗ difference/MAD
    b_r.Modified_Z_Score ← Modified_Z_Score
    if *Modified_Z_Score* > *threshold* **then**
        b_r.outlier ← 1
    else
        b_r.outlier ← 0

Output: $B_{weekday,segment}$

The outlier detection algorithm takes the event log per weekday & segment and the threshold as an input. The event log has *bag ID*, *timestamp* , *location* and *duration* attributes. The *duration* attribute is the only valuable attribute for the outlier detection algorithm. Based on the duration of each bag, the outlier detection algorithm labels the bags either outlier or not. Therefore, the output of the algorithm is an event log that is structured like the input but with an additional one field called *outlier* indicating whether the bag is an outlier per weekday & segment or not.

The outlier detection algorithm iterates over the event log that contains all the bags per weekday & segment. The first step of the algorithm is to calculate the median duration of bags. After calculating the median, per each bag, b the algorithm takes the absolute difference between b and the median and then it appends this value into a list. In the next step, it calculates the median of those differences in the list. In other words, we calculate the median absolute deviation (MAD). Then, those differences per each bag are divided by the median absolute deviation, and this division is multiplied by 0.6745 to approximate the standard normal distribution. The obtained calculation is called **Modified Z Score** corresponding to the score that measures the strength of an outlier. Each bag is assigned with the score and to decide which bags are chosen as outliers, we need to select a threshold. The threshold is a parameter for this algorithm and can be chosen any value according to point of interest of the analyst. The threshold is used as a cut-off value for the outlier detection algorithm meaning the bags that have a higher score than the threshold is labeled as an outlier while bags scored lower than the threshold are labeled as normal behaving bags.

For the purpose of this thesis, we will show some segments which are called as segment A, B, C, D as running examples.

Figure 5.1: Outlier score distribution plot of segment A

Figure 5.1 shows the probability densities regarding obtained modified Z scores of the bags from segment A for one day. The modified Z score is in the range of 0 and 300 for this segment. The higher the score is, the stronger outlier we observe. As it can be seen on the chart, most of the bags (70%) are assigned a score around zero which means they are normal behaving bags. However, some bags have higher outlier scores indicating an outlier behavior. The threshold defines where to separate normal behaving bags from the outlier bags in the segment. For example for this segment, we can choose 12 as a threshold because it is the score where we start to observe a lower probability density. With this threshold, 13 bags would be assigned as outliers while 70214 bags are assigned as normal behaving bags. However, we can also use score 42 as a threshold because it is the score where we start to observe serious outliers that have a higher score. In this case, 9 bags would be assigned as outliers while 70218 bags are assigned as normal behaving bags.

5.2.1 Normality Assumption of the Modified Z-Score Method

The Modified Z-Score method assumes normality in the sample. Hence before using this method, we have to make sure that points in the sample follow an approximately normal distribution. Otherwise, detected outliers could be the cause of the non-normality of the sample. There are different techniques in the field to test the normality in the samples. For instance, histograms provide nice visualization to check the normality in the data. Bell-shaped, symmetric histograms mean that the sample distribution is normal. There are also formal normality test that could be applied to the sample if visual representation is not providing clear understanding.

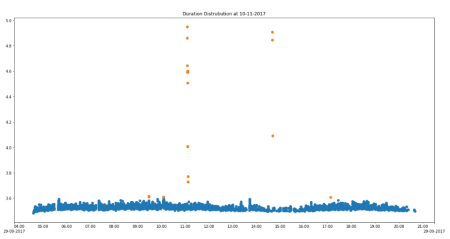
We want the mean of the duration to be representative of the actual majority class, and that is what the normality assumption gives. For this, we need a lot of samples in case we do not have normally distributed data. The important segments in the baggage handling system have on average 1000 bags per day. Hence, if a segment has for example 30 bags then this segment would not be representative of the actual majority class. We assume with the stakeholders that having at least 30 bags in the segment is a requirement. We observe more than 30 bags in most of the segments per day. There are still segments that do not have 30 bags going through it per day but according to stakeholders, these segments are not interesting parts of the system. The aim is to find hot-spots of the system, meaning finding segments that experience problems that affect more bags. Therefore, we exclude the segments that have less than 30 bags per day because they are not relevant to the analysis.

5.2.2 Parameters for Outlier Scoring Algorithm

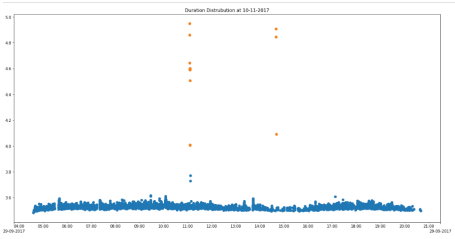
The only parameter for the Modified Z-score method is the value *threshold*. However, threshold selection forms a very crucial part for the outlier detection since it will be the cut-off value for the given sample dividing outliers and normal points. Selecting a lower threshold might easily lead to many false-positive results meaning defining more points as an outlier even though they slightly deviate from the rest of the points. On the other hand, selecting a higher threshold might lead to to missing important outlier points by labeling them as normal.

The threshold selection for the outlier detection method in the baggage handling system is challenging in a way that there are so many segments with a different number of bags passing through them. Selecting a segment-specific threshold for each segment is impossible because we have more than 2000 segments, hence we chose to identify one globally valid threshold that leads to acceptable classification for all segments. We have explored different threshold values and evaluated the correctness of the classification by visual inspection with domain experts who provide feedback. (The results from different thresholds are in appendix A).

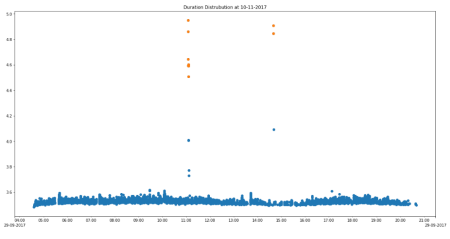
According to visual inspections, for the rest of the analysis, we will use the threshold score of **50**. However since this score is given to the algorithm as a parameter, one can try different values as a threshold as well. For example in Figure 5.2, we observe how threshold selection changes the outlier detection. We performed the outlier detection algorithm with different thresholds for segment A whose distribution is given at 5.1. According to visual inspections, it can be observed that threshold 50 is the best divider for this segment. Because selecting 5 as a threshold gives any bags near normal behavior as an outlier which is unacceptable according to the feedback from the domain experts. With threshold 20, we observe some deviations in two points but they can be overlooked since the stakeholders are interested in more serious outliers. If we assign a threshold score of 80, then we are missing some important outlier bags. This discussion illustrates the trade-off that we made with choosing 50 as a global threshold.



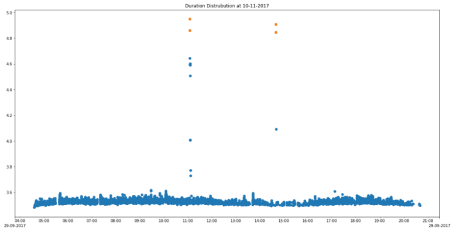
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20



(c) Labeled outliers with threshold 50



(d) Labeled outliers with threshold 100

Figure 5.2: Labeled outliers with different threshold for segment A

5.2.3 Output of the Outlier Scoring Algorithm

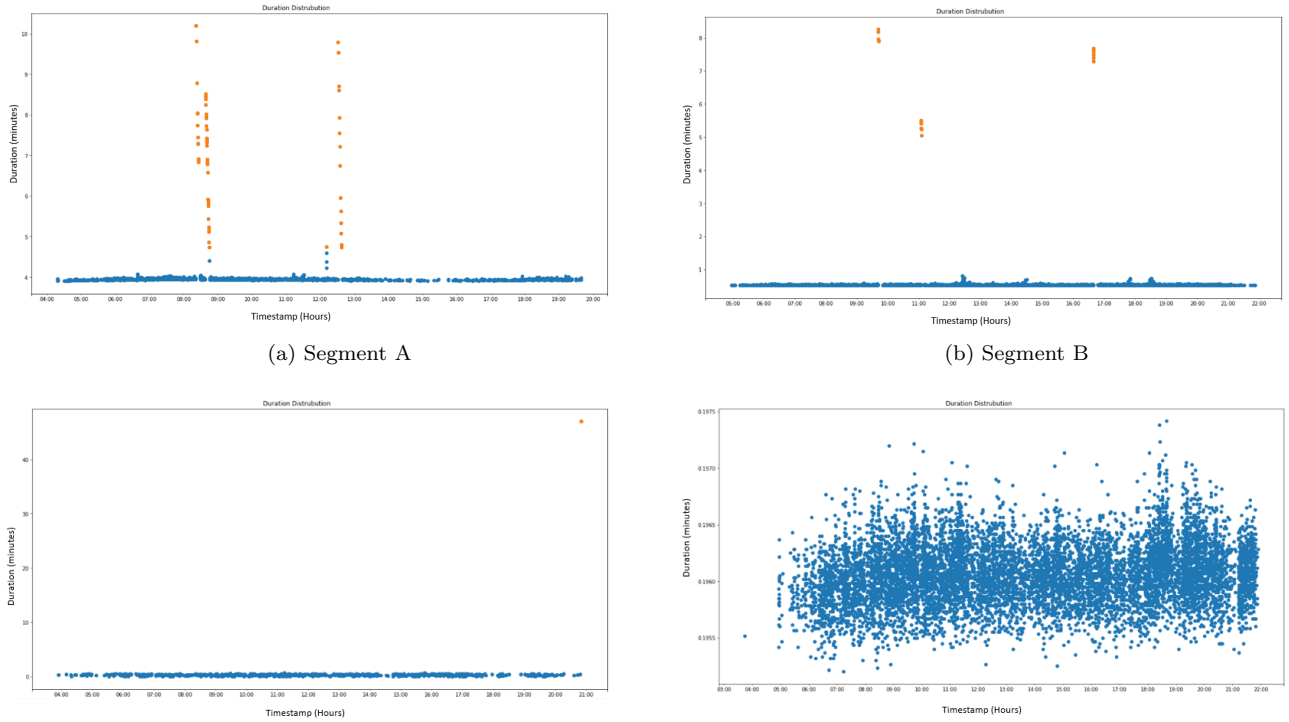
The Modified Z-score algorithm is used to score each bag in a specific weekday and segment. According to this algorithm, each bag is assigned a score that measures its deviation from the median of the sample. In other words, this score measures its outlier strength. Then, according to the pre-defined threshold, each bag is either labeled as an outlier or not. Outlier scoring and detection is performed for each weekday and on each segment the system has.

Figure 5.3 displays the result of the outlier detection algorithm in different segments. Each figure only represents the results from one specific day. These charts are showing the duration of each bag throughout the day. While the orange-colored points correspond to the outliers detected in this segment, blue points correspond to the normal behaving bags. In Segment A, 3 different outlier bundles are visible, the first outlier behavior happened at around 08:30 and many bags got involved in this outlier behavior. The second outlier bundle happened around 9:00 and 36 bags were involved in it. This abnormal behavior involves bags that took more time completing the segment A. In normal conditions, bags are completing this segment in 4 minutes. However, in the first outlier bundle, we observe that bags took 7 to 10.5 minutes to complete this segment which is clearly an outlier behavior for the segment A. Also in the same segment, there is one single bag whose duration is 7.5 minutes and it is labeled as an outlier.

In segment B, there are three outlier groups and all of them have many bags involved.

In segment C, there is only one point flagged as an outlier which lasted 45 minutes while the rest of the bags take approximately 0.5 minutes to complete this segment.

In segment D, none of the bags are labeled as outliers since all of the bags show similar behavior. We do not see any deviations from normal behavior.



(a) Segment A

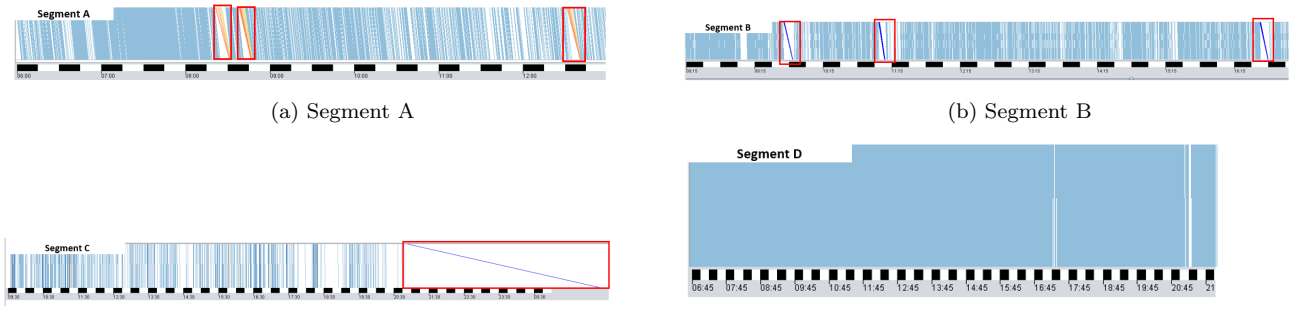
(b) Segment B

(c) Segment C

(d) Segment D

Figure 5.3: Output of outlier detection method

We can confirm the result of the outlier detection algorithm through the PSM. The segments are displayed in the PSM as in Figure 5.4. Three main abnormal behavior bundles detected as a result of the outlier detection algorithm in segment A and segment B are also visible on the PSM. The outlier point in segment C can be seen as a line on the PSM that is represented as a skewed line indicating a slow movement. Also, there are no slow bags visible on segment D. All the bags have almost vertical lines, therefore, we do not see any performance problems for this segment verifying the outlier detection result.



(a) Segment A

(b) Segment B

(c) Segment C

(d) Segment D

Figure 5.4: Output of outlier detection method on PSM

Table 5.1 displays the result of the outlier detection algorithm. The result of the outlier detection algorithm is the same format as the original event log consisting of attributes *Bag ID*, *Timestamp* , *Activity* and *Duration*. In addition to the original event log, this algorithm produces two new attributes; *Outlier* and *Modified Z-Score*. *Outlier* defines whether the bag is an outlier or

not for the segment. While 1 means 'outlier', 0 means 'normal behaving bag'. On the other hand, *Modified Z-Score* attribute holds the modified Z- Score obtained as a result of the algorithm. For the rest of the analysis, we will refer to this table.

Bag ID	Timestamp	Activity	Duration (minute)	Outlier	Modified Z-Score
100000	2019-05-21 07:41:05.782	Segment A	3.497	0	0.630
100001	2019-05-21 08:13:34.145	Segment A	3.584	0	1.369
100002	2019-05-21 09:10:24.856	Segment A	7.321	1	542.124
100003	2019-05-21 09:15:14.154	Segment A	4.949	1	106.712
...	...	...	...	...	...
110000	2019-05-21 08:13:11.114	Segment B	0.555	0	0.072
110001	2019-05-21 10:14:17.412	Segment B	8.256	1	786.423

Table 5.1: Output of the outlier score algorithm

5.3 Outlier Classification Method

Once all the outlier bags are identified in the baggage handling system, we need to classify certain outlier patterns because from the output of the outlier detection method, we observed that some outlier behaviors are often repeated throughout the system. Also, from the observations at Figure 5.3, we observe that there are bundles of outlier behavior along with the individual ones. It gives rise to the need for classifying patterns of outliers. Finding these outlier patterns enables us to see recurring problems in the system and furthermore stakeholders can evaluate these problems to find where and most importantly why they happen so that they can find a solution to fix it. We explained the problems that the baggage handling system has been facing in Chapter 3. Taking these problems into account and observing the bag behavior in several segments, outlier bags have been classified into 4 categories as *blocking*, *stuck*, *isolated* and *fast* bags. These categories can be described as:

Blocking and Stuck Bags : The common problem of the baggage handling system happens when the bags got accumulated on one segment and unable to move forward to the next segment. In Chapter 3 we compared these accumulated points to traffic jams on the roads. The result of the outlier detection algorithm also conformed that traffic jams exist in the baggage handling system. We will call each of these traffic jams a "*blockage*" in the baggage handling system. Blockages happen when multiple bags in one segment stop moving or each of them moves very slowly around the same time. Like the outlier behavior, blockages need to be inspected in the level of segments too. Because a bag can be stuck in the blockage in one segment but behave at its normal pace in the next segment. We can also differentiate the bags that start the blockage and the bags that are affected by this blockage since the timestamp information is present for all the bags. If the bags are ordered according to their timestamp in the segment, we can call the first bag in the blockage a *blocking bag* and the other bags a *stuck bag* . Figure 5.5 shows the blocking and stuck outlier patterns in the Performance Spectrum Miner. We observe one bag starting a blockage and other bags getting stuck by this blockage.

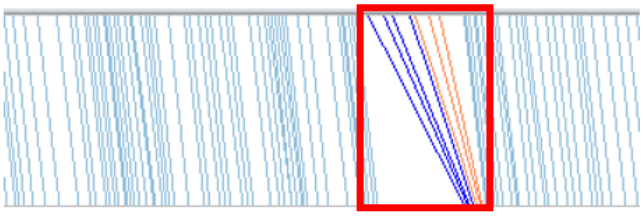


Figure 5.5: Blocking and Stuck Bag Pattern in PSM

Isolated Bags : Another problem in the baggage handling system is that sometimes, only one bag can behave abnormally without being affected by other bags. These bags are not causing any blockage behind them nor are they stuck because of other bags. We classify these bags as *isolated bag* because there is no other abnormal behavior happening near it. This kind of outlier could be happening because of various reasons, such as; if the bag entered the baggage handling system too early then it would wait for laterals to be open and while waiting, it stays in one segment. This leads to long duration in one segment, in other words, an abnormal behavior. Additionally, there could be a temporary malfunction in the system that only affected one bag after which the system went back to its normal state. Figure 5.6 shows the isolated outlier pattern in the PSM. We observe, one bag behaved abnormally and none of the other bags are affected by this abnormal behavior.

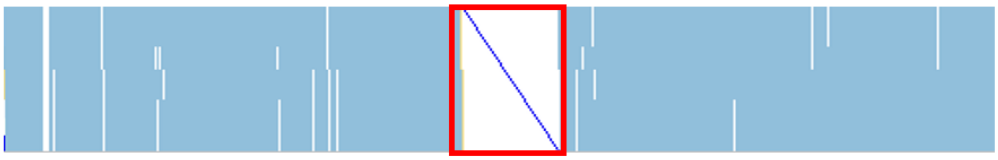


Figure 5.6: Isolated Pattern in PSM

Fast Bags : In the baggage handling system, some bags are faster than the rest of the bags that traveled on the same path. We call these *fast bag*. This could indicate either something happened such as manual intervention or could also mean without traffic, the bag can pass the path faster. This raises the question of why other bags are not behaving in the same way. Figure 5.7 shows the fast bag outlier pattern in PSM. The bag in the red rectangular acted too fast compared to other bags in the segment. The line belonging to the bag is almost vertical, indicating that this bag passed this segment almost immediately.

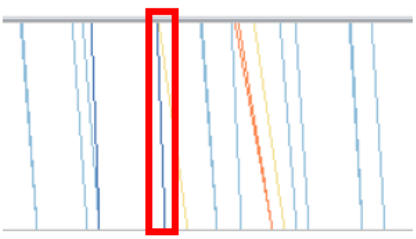


Figure 5.7: Fast Bag Pattern in PSM

Until this point of the analysis, each bag is assigned an outlier score and according to a threshold, outlier bags are labeled. Now we classify all the existing outlier patterns in the baggage handling system in different steps. Figure 5.8 shows all the required steps to classify bags into

different outlier patterns. First, we use the result of **Outlier Detection Algorithm** as an input. For the outlier bags (Outlier=1) we first execute **Blockage Detection Algorithm** to find all the blocking and stuck bags in the system then **Isolated Bag Detection Algorithm** is executed to find isolated bags in the system. After this step, we find fast bags by executing the **Fast Bag Detection Algorithm**. On the other hand, for normal behaving bags (Outlier=0) **Normal Behaving Bag Detection Algorithm** is executed to label the bags. Finally, we execute the **Separation Of Blockages Algorithm** to separate blockages if normal behaving bags are detected in between.

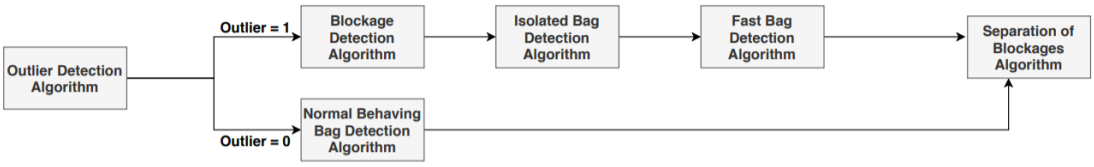


Figure 5.8: Pipeline for the outlier classification

5.3.1 Blockage Detection

We will now detect all the blockages in each segment of the baggage handling system and classify these blockages based on whether it is blocking the segment or gets blocked in the segment.

On a conceptual level, the blockage detection algorithm aims to detect and label the bags in blockage as blocking and stuck bags. We can assume that if bags start to behave abnormally around the same time and in the same location then they belong to a blockage. To check the blockages, we first have to define a time window because we want to detect bags that belong to the same blockage and this is only possible if we look at a certain time window. For this analysis, 3 minutes is selected as the time window as a result of visual inspections of the data. However, we have also tried other time windows. In the end, 3-minute window was most suitable for the analysis since it was reasonable for the business context and it has been proved with the visual inspections. Other time window settings are present in Appendix C. Hence, in order to detect blockage behavior, we have to check the 3-minutes window that each outlier bag belongs to. If we see any other outlier bag in the 3-minute window, then we know that they belong to the same blockage. Consequently, the first bag in the same time window would be labeled as a blocking bag while others are labelled as stuck bags.

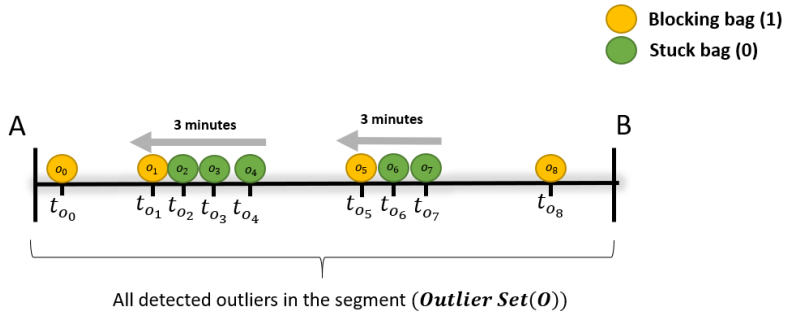


Figure 5.9: Blockage detection representation

The blockage detection algorithm can be explained as in Figure 5.9. For the segment A:B, we sort every bag according to their timestamp t where $t_0 < t_1 < t_2 < ... < t_8$. According to the algorithm, we assign the first bag (o_0) as a blocking bag and then iteratively shift the other bags

3 minutes and see whether we detect another outlier bag in this 3-minute window. For example, we know that t_1 happened 5 minutes after t_0. Therefore we cannot say that the bag o_1 belong to the same blockage with the bag o_0. We now start a new blockage with the bag o_1 and o_2, o_3, o_4 are all in the 3 minutes window with the o_1. Hence, they are assigned as stuck bags. The same logic is applied to the blockage that starts with o_5 and ends with o_7. We assign the last bag in the segment (o_8) as a blocking bag even though it is not correct because o_8 is not followed by any other outlier bag. In the next steps of outlier classification, we will deal with this problem.

The blockage detection in the baggage handling system is performed with the following steps:

For each weekday and segment in the event log:

1. Obtain the bag list (B)
2. Sort the bags ascending according to their timestamp.
3. Apply the following algorithm to the ordered bag list B to discover blockages.

Algorithm 2: Blockage Detection

Let B be the ordered list of the bags and T be the ordered list of timestamps belong to bags where $\forall b \in B$ and $r \in \{1, 2, 3, ..., n\}$ where n is the size of the set B and r is the index of each bag.

Input : $B_{weekday,segment}, T_{weekday,segment}$

```
O ← []
TO ← []
foreach b_r in B do
    if b_r.outlier = 1 then
        append b_r to O
        append T_r to TO
foreach o_r in O do
    if o_r.timestamp = min(TO) then
        o_r.outlier_type ← 1
    else
        if shift(TO_r, −3minutes) > TO_(r−1) then
            o_r.outlier_type ← 1
        else
            o_r.outlier_type ← 0
```

where 1 and 0 corresponds to blocking bag and stuck bag respectively.

Output: $O_{weekday,segment}, TO_{weekday,segment}, B_{weekday,segment}$

The blockage detection algorithm takes the ordered list of bags per weekday & segment as input. The structure of the input is given in table 5.1. The obtained set of bags is sorted based on the timestamp attribute before it is given as an input to the blockage detection algorithm. On the other hand, the output of the blockage detection algorithm will be the ordered list of bags per weekday & segment with points classified as a blocking bag or a stuck bag. The structure of the output will be like the input with one additional property which is the attribute called *outlier_type*. The attribute, *outlier_type*, holds the information of the type of outlier per outlier bag. As explained in 5.3, there are 4 types of outliers observed in the baggage handling system: *blocking*, *stuck*, *isolated* and *fast* bags. The output of the blockage detection algorithm detects the *blocking* and *stuck* bags per weekday & segment and labels them in the *outlier_type* attribute.

The blockage detection algorithm iterates over all the bags in the ordered list of bags. It first puts the bags that labeled as outliers as a result of the outlier detection algorithm into the list O. Timestamps of the outlier bags are also recorded to the list TO. After that, the algorithm iterates over the outlier bag list O. It assigns 1, blocking bag, to the first outlier bag in the list. The algorithm checks this condition by taking the minimum timestamp of the list TO and comparing it to the timestamp of the bag o_r. $(min(TO))$ Then, for each outlier bag o, it shifts the timestamp of the bag to 3 minutes before and checks if the shifted timestamp is greater than the timestamp of the previous outlier bag. If that condition holds, it assigns 1, blocking bag, to the *outlier_type* attribute. If the condition does not hold then 0, stuck bag, is assigned to the *outlier_type* attribute. With this condition, the algorithm checks if two subsequent bags are too far from each other by scanning the 3-minute window for other outliers.

5.3.2 Isolated Bag Detection

Isolated bag detection is important because it allows us to see the temporary one-time problems in the system. Stakeholders can investigate the reasons behind these temporary problems by looking at the time when an isolated bag appeared and they can see the state of the segment for that specific time.

The isolated bag detection algorithm finds and labels the isolated bags in the segments. Isolated outlier behavior happens when there is no other outlier behavior around the same time on the same location. As in the blockage detection algorithm, we also have to depend on a time window. Given an outlier bag, if we cannot detect any other outlier behavior in a 3-minute interval, we say that that particular bag is isolated.

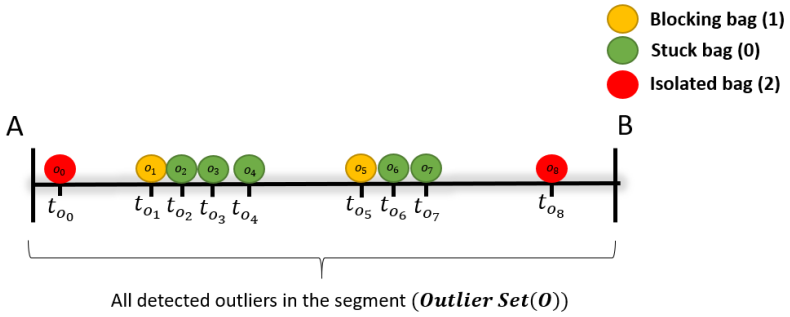


Figure 5.10: Isolated bag detection representation

We express the blockage detection algorithm in Figure 5.9, after we perform the isolated bag detection algorithm, outlier labels are changed for some bags. In Figure 5.10, we see the result of the isolated bag detection algorithm and according to it, bags o_0 and o_8 are labeled as isolated bags since they are not followed by any other outlier bag in this segment. In this way, we correct the wrongly classified bags in the blockage detection algorithm. Hence, the blockage detection algorithm and the isolated bag algorithm complete each other. We will now explain the isolated bag detection process.

For each weekday and segment in the event log:

1. Obtain the outlier lists O and T0 with detected blockages from blockage detection algorithm
2. Apply the following algorithm to the ordered outlier list with detected blockages O

Algorithm 3: Isolated Bag Detection

Let O be the ordered list of outlier bags with detected blockages and TO be the ordered set of timestamps belong to outlier points where $\forall o \in O$ and $r \in \{1, 2, 3, ..., n\}$ where n is the size of the outlier list O and r is the index of each outlier point.

Input : $O_{weekday,segment}, TO_{weekday,segment}, B_{weekday,segment}$
foreach o_r *in* O **do**
  if $o_r.timestamp = max(TO)$ *and* $o_r.outlier\_type \neq 0$ **then**
    o_r.outlier_type $\leftarrow 2$
  else if $o_r.timestamp = min(TO)$ *and* $o_{(r+1)}.outlier\_type = 1$ *and* $o_r.outlier\_type = 1$ **then**
    o_r.outlier_type $\leftarrow 2$
  else
    if $(o_{(r+1)}.outlier\_type$ *in* $(1, 2)$ *and* $o_r.outlier\_type = 1$ **then**
      o_r.outlier_type $\leftarrow 2$

where 0,1,2 corresponds to stuck bag, blocking bag and isolated bag respectively.

Output: $O_{weekday,segment}, TO_{weekday,segment}, B_{weekday,segment}$

The isolated bag detection algorithm takes the list of outliers with the detected blockages per each weekday & segment as an input. In other words, it takes the output of the blockage detection algorithm as an input. The input event log has the attributes of *bag ID*, *timestamp* , *location*,*duration* and *outlier_type* where *outlier_type* only consists of 0, stuck bag, or 1, blocking bag. The input is sorted based on the timestamp of the outlier bags. Nonetheless, the output of the isolated bag detection algorithm is an event log with the same attributes as the input. The only difference is that, instead of only having values of 0 and 1, the *outlier_type* will have the value of 2, isolated bag, as an outlier type too. Some blockage outlier types will be converted to the isolated outlier type.

The algorithm, iterates over all the bags in the ordered list of outliers. All the outlier bags are either labeled as 1 or 2 in the input. To detect isolated bags, the algorithm first checks if the bag is on the edge of the given list. If the outlier bag is the last bag in the outlier list based on its timestamp ($max(TO)$) and if it is not a stuck bag then we assign 2, isolated bag, for that particular outlier bag. The reason behind this step is that the last bag on the list cannot be a blocking bag since there are no other bags to follow it. Similarly, if the bag is the first on the outlier list ($min(TO)$) and if it is labeled as a blocking bag, the algorithm checks the outlier type of the next bag. If the next bag is also a blocking bag then we change the labeling of the bag to the isolated bag. The reason for this step is that the blockage detection algorithm automatically labels the first bag as a blocking bag and to assure the bag is labeled correctly, we need to check whether it is a blocking bag or an isolated bag. In this way, we cover the edges. If the bag is not on the edge and labeled as a blocking bag then the algorithm checks the next bag. If the next bag is either a blocking bag or another isolated bag then we change the labeling to the isolated for that particular bag.

5.3.3 Normal Behaving Bag Detection

The normal behaving bags also have to be labeled for the rest of the analysis.

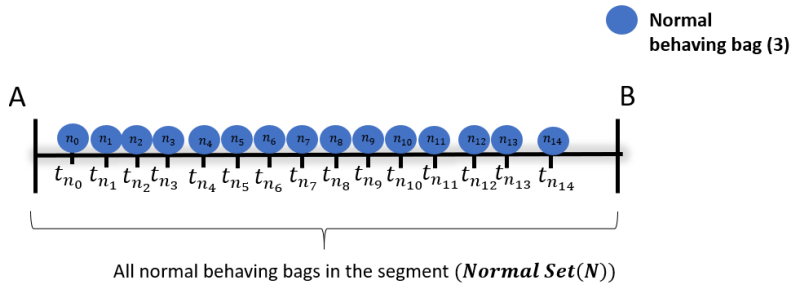


Figure 5.11: Normal bag detection representation

The normal behaving bag detection algorithm takes all the normal behaving points according to the outlier detection algorithm and assigns a label, normal behaving bag, to each of them. Figure 5.11 shows all the normal behaving bags in the segment A:B where each bag n is sorted according to its timestamps where $t_{n_0} < t_{n_1} < ... < t_{n_{14}}$

Algorithm 4: Normal Behaving Bag Detection

Let Nr be the ordered list of the normal-behaving bags and TN be the ordered list of timestamps belong to the normal-behaving bags where $\forall nr \in Nr$ and $r \in \{1, 2, 3, ..., n\}$ where n is the size of the set Nr and r is the index of each bag.

Input : $B_{weekday,segment}, T_{weekday,segment}$
Nr $\leftarrow$ []
TN $\leftarrow$ []
foreach b_r *in* B **do**
  if $b_r.outlier = 0$ **then**
    append b_r to Nr
    append T_r to TN
foreach nr_r *in* Nr **do**
  nr_r.outlier_type $\leftarrow$ 3
where 3 corresponds to normal behaving bag

Output: $Nr_{weekday,segment}, TN_{weekday,segment}$

The normal behaving bag detection algorithm takes the ordered list of bags per each weekday & segment as an input. The structure of the input is given in table 5.1. The obtained list of normal behaving bags is sorted based on the timestamp attribute. The output of the normal behaving bag detection algorithm will be the ordered list of normal behaving bags per weekday & segment with points classified as normal with an additional attribute *outlier_type*.

The algorithm puts the non-outlier bags ($b_r.outlier = 0$) into another list Nr and then in this list, it iteratively assigns 3, normal behaving bag, to the *outlier_type* attribute for each non-outlier bag.

5.3.4 Fast Bag Detection

Some bags in the system get labeled as outliers in some segments even though their duration is low compared to the other bags in the system. Detecting these bags prevents us from misclassifying bags because if some pile of bags passes the segment very fast, they would be detected as an outlier possibly a blockage even though they are not. We, therefore, have to make sure to separate these fast bags from the rest.

Algorithm 5: Fast Bag Detection

Let O be the ordered list of all the outlier points where $\forall o \in O$ and B be the ordered list of all the bags where $\forall b \in B$

Input : $O_{weekday,segment}$,$TO_{weekday,segment}$, $B_{weekday,segment}$

$median = median(B.\text{duration})$

foreach *o in O* **do**
  if $o.duration < median$ **then**
    o.outlier_type $\leftarrow 4$
where 4 corresponds to fast bag

Output: $O_{weekday,segment}, TO_{weekday,segment}$

The fast bag detection algorithm takes the list of outliers and set of all the bags as an input, and it gives the same structured outlier list with a difference in the *outlier_type* attribute for some bags since the outlier type of some bags changes after the algorithm.

The algorithm first calculates the duration median for the list that contains all the bags for segment & weekday. Then for each outlier bag o in the outlier list, it checks if the duration of the bag is less than the duration median. If the condition holds, the algorithm changes the *outlier_type* attribute to 4 which corresponds to a fast bag.

5.3.5 Separation of Blockages in the Presence of Normal Bags

The problem with the outlier classification algorithm until now is that the normal behaving bags and outlier bags are investigated separately. This leads to ignoring the time perspective. Since both subsets are analyzed differently we do not take normal behaving points into the account when discovering outlier patterns. As a result, when both subsets are merged, we found that there are some normal behaving points between blockages. In its nature, blockages are a pile of outlier points happening around the same time and in the same location. Hence, having normal points between the blockages breaks the pattern. We can no longer assume the blockage since there are some points performing well despite being in the middle of the blockage. The segment is not really in the state of a blockage if some bags can move at a normal pace. The solution to that problem is to separate the blockage if a normal behaving bag is detected in the middle of it. In this way, several different blockages are formed instead of having one big blockage. With this approach, we take care of all bags in the relevant weekday&segment rather than just focusing on different subsets, being outliers and normal behaving set of the relevant weekday&segment.

The separation of blockages algorithm takes all the classified bags that belong to the weekday & segment. All the bags in the input are classified with one outlier type, and the aim of the algorithm is to correct wrongly classified blockages by separating them into different small blockages if the normal behaving bags are detected in between. The output of the algorithm is the same as the input with the only difference being modified outlier types.

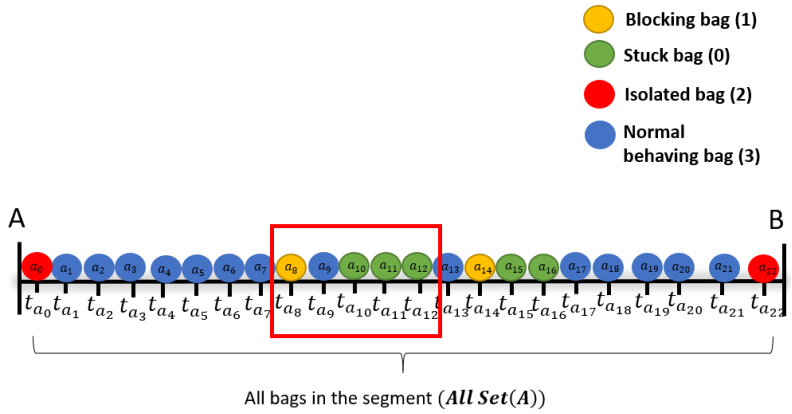


Figure 5.12: The separation of blockages algorithm representation

Figure 5.12, shows the reason why we need this step. After the merge of outlier bags and normal behaving bags, each bag is sorted according to its timestamp t_a where $t_{a_0} < t_{a_1} < ... < t_{a_{22}}$ In the figure, some blockages have normal behaving bags in between like shown in the rectangular. The algorithm separates the blockage by assigning an isolated bag to bag a_8 since it is not followed by an outlier point. Also bag a_{10} is labeled as a blocking bag since it is followed by two other outlier bags.

Algorithm 6: Separation of Blockages

Let A be the ordered list of all bags and TA be the ordered list of timestamps belong to the bags where $\forall a \in A$ and $r \in \{1, 2, 3, ..., n\}$ where n is the size of the set A and r is the index of each point.

Input : $O_{weekday,segment}, TO_{weekday,segment}, Nr_{weekday,segment}, TN_{weekday,segment}$

A $\leftarrow$ MERGE(O, Nr)
TA $\leftarrow$ MERGE(TO, TN)

foreach a_r *in* A **do**
  if $a_r.timestamp \neq max(TA)$ *and* $a_r.timestamp \neq min(TA)$ **then**
    if $a_r.outlier\_type = 0$ *and* $a_{(r-1)}.outlier\_type$ *in* $(2, 3, 4)$ **then**
      a_r.outlier_type $\leftarrow 1$
    if $a_r.outlier\_type = 1$ *and* $a_{(r+1)}.outlier\_type \neq 0$ **then**
      a_r.outlier_type $\leftarrow 2$

where 0,1,2,3,4 corresponds to stuck bag, blocking bag, isolated bag, normal behaving bag, fast bag respectively.

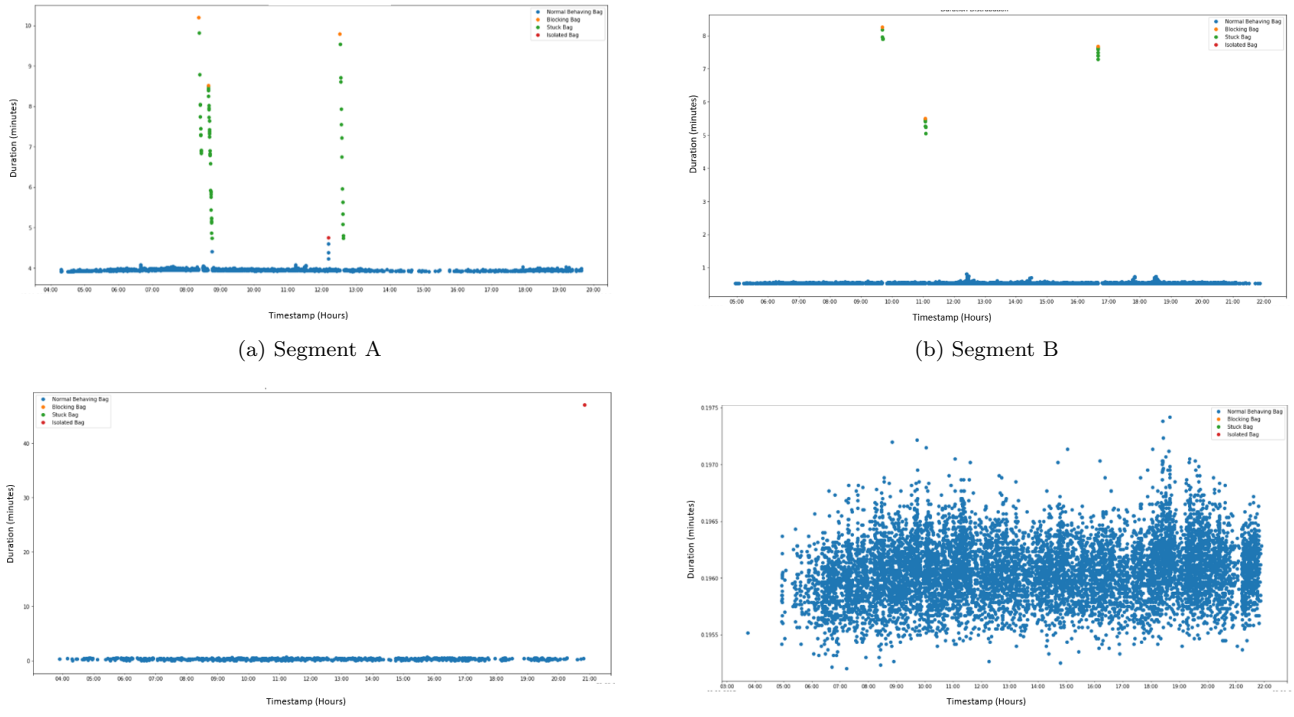
Output: $A_{weekday,segment}$

The algorithm first merges the normal behaving bag list (Nr) with all the classified outlier list (O). Then it iterates over all bags in the merged list A. It first checks whether the bag is on the edge of the list or not and if not it checks if the outlier type of the bag is 0, stuck bag, and it also checks if the bag that passes in the same segment just before it labeled as 2, 3, 4 meaning isolated, normal and fast bag respectively. If the condition holds, the algorithm changes the stuck bags under these conditions to a blocking bag to start a new blockage. Similarly, if the blocking bag is not followed by any stuck bags then the algorithm changes the outlier type to isolated bag.

5.3.6 Output of the Outlier Classification Algorithm

The outlier classification algorithm consists of many steps which we explained as different algorithms. When all the steps are performed on the event log, we obtain a dataset per weekday & segment with every bag labeled as either one of the outlier types or as a normal behaving bag. Having every bag classified enables us to see types of problem in the baggage handling system that we explained in Section 3.1.2 with the information of when and where they happen.

Figure 5.13 shows the result of the outlier classification algorithm. We classified the outliers displayed in Figure 5.3 into different performance-related problem in the system. The blockages in segment A are detected and showed in different colors. Blocking bags and stuck bags are shown in orange and green respectively. Normal behaving bags are assigned the color blue, and we observe one isolated bag which colored red in segment A. The reason to have this isolated point is because of the threshold selection. If we lower the current threshold (outlier score of 50), we would have gotten 4 bags in a blockage. In segment B, we only have three blockages and they are all colored accordingly. In segment C, we observe one bag whose duration is 45 minutes and since there is no other outlier behavior happening around it, this bag is classified as an isolated bag and colored in red. On the other hand, in segment D, we do not detect any abnormal behavior hence we do not have an outlier pattern here. More results can be found in Appendix D.



(a) Segment A

(b) Segment B

(c) Segment C

(d) Segment D

Figure 5.13: Output of outlier classification method

5.4 Attribute Detection for Blockages

Blockages in the system can involve many bags and just identifying which bag belongs to the blockage is not sufficient to understand how serious the blockage problem is. We need to measure the importance of the blockage because, for instance, a blockage that lasted 1 minute with 3 bags involved is not as serious as a blockage that lasted 5 minutes with 50 bags involved. Stakeholders

would want to focus on the areas that have serious blockages since those areas are likely the problematic parts of the baggage handling system. Therefore, we introduce two measures to calculate the seriousness of the blockage. The first measure is called *blockage duration* and it measures how long each blockage lasted. The second measure is named *blockage bag count* and it calculates the total number of bags in each blockage.

The attribute detection for blockages consists of two steps as displayed in Figure 5.14. First, the blockage duration is calculated for each bag in every segment. If the bag does not belong to any blockage then *blockage duration* is assigned 0. If the bag belongs to a blockage then the blockage duration is calculated and it is assigned to every bag in the same blockage. This algorithm produces a file called **Outlier.CSV** this file contains all the fields from the event log in addition to those fields, it also contains the outlier type and blockage duration for each bag. After the blockage duration for each bag has been assigned, the total number of bags in each blockage is calculated. In this step, a file called **Blockage.CSV** that contains all information about the blockages is created.

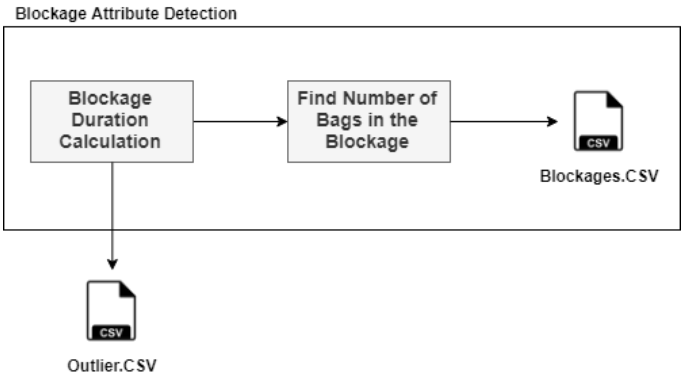


Figure 5.14: Attribute Detection For Blockages

5.4.1 Blockage Duration Calculation

A blockage is the state of the baggage handling system in which several bags suffer from slow movement. Knowing the duration of each blockage helps us to separate the significant blockages from the insignificant ones. With the blockage duration measure we can know for how long the system was not operating in its normal state. Hence, this measure carries great value for the stakeholders to discover the impact of the problem in their system.

The blockage duration calculation algorithm takes the list of all classified points as an input per weekday & segment. Every bag in the input has been classified either as a type of outlier or a normal behaving bag. On the other hand, the output of the blockage detection algorithm has one extra field which is called *blockage_duration*. The attribute *blockage_duration* holds the information of the duration of the blockage per each bag. If the bag does not belong to a blockage then this attribute gets a value of 0. Besides, all the bags that belong to the same blockage have the same blockage duration since the blockage duration is not the attribute of individual bags but the attribute of the entire blockage. The output event log has fields of *bag ID*, *timestamp*, *location*, *duration* along with *blockage_duration*.

The blockage duration calculation algorithm first assigns 0 to *first_bag*, *last_bag*, *last_bag_finish* and $deltaT$ variables. The *first_bag* variable corresponds to the blocking bag of each blockage. Similarly, *last_bag* variable corresponds to the last stuck bag that got caught in the blockage based on its timestamp. Furthermore, the *last_bag_finish* variable corresponds to the time that the last stuck bag left the segment. Finally, the $deltaT$ variable corresponds to the blockage duration per

Algorithm 7: Blockage Duration Calculation

Let B be the ordered set of all the bags and T be the ordered set of timestamps belong to the all the bags where $\forall t \in T$, $\forall b \in B$ and $r \in \{1, 2, 3, ..., n\}$ where n is the size of the set B and r is the index of each outlier point.

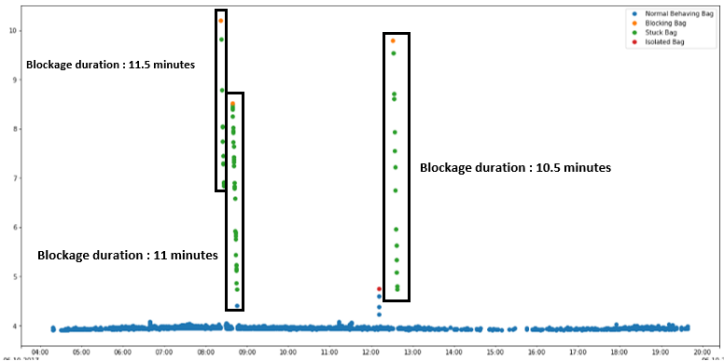
$first\_bag \leftarrow 0$
$last\_bag \leftarrow 0$
$last\_bag\_finish \leftarrow 0$
foreach b_r *in* B **do**
  if $b_r.outlier\_type = 1$ **then**
    $stuck\_list \leftarrow []$
    $first\_bag \leftarrow b_r$
    $first\_bag\_ID \leftarrow b_r.BagID$
    foreach b_r *in* $B[r+1:]$ **do**
      if $b_r.outlier\_type = 0$ **then**
        $last\_bag \leftarrow b_r$
        $last\_bag\_finish \leftarrow T_r + b_r.duration$
        append last_bag to stuck_list
        $last\_bag\_ID \leftarrow b_r.BagID$

      else
        break
    $PatternID \leftarrow first\_bag\_ID + last\_bag\_ID$
    $deltaT \leftarrow last\_bag\_finish - first\_bag$
    b_r.deltaT $\leftarrow deltaT$
    b_r.PatternID $\leftarrow PatternID$
    foreach *s in stuck_list* **do**
      s_r.deltaT $\leftarrow deltaT$
    s_r.PatternID $\leftarrow PatternID$
  else if $b_r.outlier\_type = 0$ **then**
    continue
  else
    b_r.deltaT $\leftarrow 0$
    b_r.PatternID $\leftarrow 0$

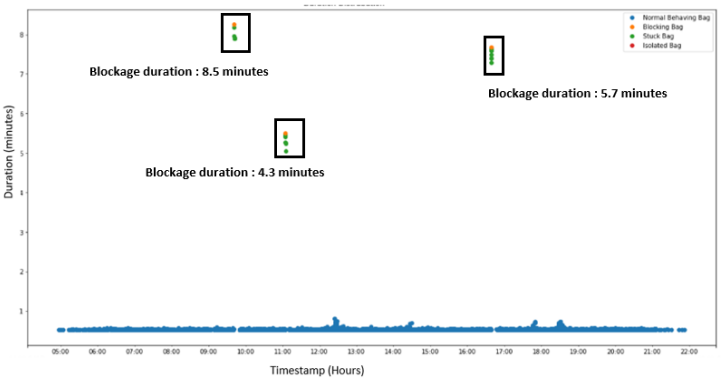
where 0,1,2,3,4 corresponds to stuck bag, blocking bag, isolated bag, normal behaving bag, fast bag respectively.

each blockage. The algorithm iteratively checks all the bags in the input. For each bag b, it checks if the bag is a blocking bag or not. If the bag is not a blocking bag nor a stuck bag we assign 0 to its blockage duration since the bag does not belong to any blockage. If the bag is a blocking bag then we assign the bag to the *first_bag* variable and create a list to hold all the points belong to the same blockage. After assigning the blocking bag, the algorithm iterates over another set that has been obtained by excluding the blocking bag and having other bags that happen after that particular blocking bag. The reason for this step is that we are trying to find all the stuck bags that belong to the same blockage as the blocking bag and stuck bags pass the segment after the blocking bag. If the next bag is a stuck bag then we iteratively change the value of *last_bag* and each of these stuck bags are also added to a list. Ultimately the *last_bag* value belongs to the last stuck bag in the blockage. The next step in the algorithm is to assign *last_bag_finish* variable by adding the duration attribute for the last stuck bag to its timestamp. With this step, we obtain the exact times that the blockage has ended. Lastly, the difference between timestamps of last stuck bag in the blockage and the blocking bag is assigned to the blocking bag as blockage duration. Besides, we also create a unique blockage pattern ID for each blockage by combining the ID of the first bag and the last bag in the blockage. This pattern allows us to distinguish the blockages.

Figure 5.15 displays the blockage detection algorithm results. In segment A, blockages lasted 11.5, 11 and 10.5 minutes and in total, we observe 33 minutes of blockages for that particular day in segment A. In segment B, blockages lasted 8.5, 4.7 and 5.7 minutes and in total 18.9 minutes of blockage for that specific day in segment B.



(a) Segment A



(b) Segment B

Figure 5.15: The blockage duration metric representation

Table 5.2 displays the resulting file from the *Blockage Duration Calculation* algorithm. This file is called **Outlier.CSV** and it contains all the fields from the event log. In addition to those fields, it also contains the outlier type and blockage duration for each bag.

Timestamp	Activity	CaseID	Duration (Minute)	Blockage Duration (Minute)	Outlier Type
2019-05-21 07:41:05.782	Segment A	10000000	0.1036	1,095	1
2019-05-21 07:41:42.772	Segment A	10000001	0.1031	1,095	0
2019-05-21 07:42:05.623	Segment A	10000002	0.1028	1,095	0
2019-05-21 07:42:27.920	Segment A	10000003	0.1034	0	3
2019-05-21 07:42:30.457	Segment A	10000004	0.1077	0	2
.					
2019-05-21 10:47:11.800	Segment B	11000001	2.0427	0	3
2019-05-21 10:47:12.851	Segment B	11000002	2.0426	0	3

Table 5.2: Outlier.CSV

5.4.2 Number of Bags in Blockages

Next to wanting to know the duration, we also want to know how many bags were involved in the blockage. Obtaining the total number of bags in the blockages carries an important value for the analysis in a sense that if we know how many bags got involved in the blockage we would know the importance of this blockage. If a great number of bags are affected by one blockage then we can assume that the segment is having a really serious problem. On the other hand, we can ignore some blockages if it only affects, for example, two bags. This measure is calculated by counting the bags that have the same PatternID as introduced in algorithm 7. This measure along with the blockage duration measure provides a good understanding of all the blockages in the system.

Activity	Blockage ID	Blockage Duration (Minute)	Total Number of Bags in Blockage	Average Blockage Duration Per Bag	Blockage Start	Blockage End
Segment A	1	1,095	3	0.365	2019-05-21 07:41:05.782	2019-05-21 07:42:06.102
Segment A	1	1,095	3	0.365	2019-05-21 07:41:05.782	2019-05-21 07:42:06.102
Segment A	1	1,095	3	0.365	2019-05-21 07:41:05.782	2019-05-21 07:42:06.102
Segment A	2	4.256	2	2.128	2019-05-21 09:17:27.920	2019-05-21 09:21:35.912
Segment A	2	4.256	2	2.128	2019-05-21 09:17:27.920	2019-05-21 09:21:35.912
...	...	...	...	...	...	...
Segment B	5	10.415	2	5.2075	2019-05-21 22:47:11.800	2019-05-21 22:57:48.189
Segment B	5	10.415	2	5.2075	2019-05-21 22:47:11.800	2019-05-21 22:57:48.189

Table 5.3: Blockage.CSV

Table 5.3 displays the resulting file after the calculation of total bags in the blockages. This file is called **Blockage.CSV** and it contains all the information about the blockages in each segment. For every segment, all the blockages along with their ID, duration, the total number of bags, average blockage duration per bag, and start-end time are listed.

In this chapter, we explained the process of detecting the outlier bags with respect to their performance in a segment. Modified Z-Score is calculated for each bag to quantify the severity of the performance problem each bag has. We also identified some outlier patterns such as isolated bags and blockages which are formed when several bags are stuck together. Finally, we quantified each blockage with respect to their duration and the involved number of bags. Now that we have all required information on outlier types and segment performance, we will explain the process of classifying segments based on their performance.

Chapter 6

Classifying Outlier Score Distribution For Segments

To find the ground truth about the types of outlier behavior the segments commonly show, this chapter explains the process of classifying outlier score distributions for each segment. Section 6.1, describes the method to find representative outlier score distribution clusters given historical data of 6 months of event logs. Section 6.2 on the other hand explains the process of matching daily test data to relevant clusters.

6.1 Finding Outlier Score Distribution Clusters

The problem with the analysis so far is that we do not have labeled data indicating outlier behavior. As stated in Chapter 3, one of the main objectives of this thesis is to provide ground truth about the types of outlier behavior in the baggage handling system. Until this point, all the methods have been conducted for the historical 6-months of event log. We aim to use the results obtained from this historical data to provide us a ground truth about the outlier behavior in the system. However, we cannot assume that the majority of the historical data is without any outliers. Hence, using the historical analysis result as it is might lead to creating a wrong ground truth for the analysis. Taking this problem into account, we would like to know that given any day, how bad the segment behaves compared to all other days in the historical data. We approach this question by ranking the different days in the historical data with regard to how bad the outliers are and describe their differences.

The technique to find how bad the segment behaves compared to other days can be described as using the outlier score distribution that we obtained after the outlier detection algorithm to find similar days for each segment. If we assume that historical data is perfect, meaning the majority of the data is without outliers, we would obtain similar outlier score distributions for each day regarding the segment. The reason we obtain similar outlier score distributions is that the bags would have been assigned similar scores as most of them do not deviate from the baseline behavior which forms the majority. However, historical data is not perfect. For some days, we have observed many serious outliers during these 6 months. Some of these serious outliers are displayed in Chapter 5. As a result of not having the perfect data, outlier score distributions of different days vary greatly. Therefore, we will group the days that have a similar outlier score distribution together to be able to rank the different outlier behavior each of the group represents. Table 5.3 displays the resulting CSV after the calculation of total bags in the blockages. This file is called **Blockage.CSV** and it contains all the information about the blockages in each segment. For every segment, all the blockages along with their ID, duration, the total number of bags, average blockage duration per bag, and start-end time for them are listed.

6.1.1 Data Preparation

The technique to find how bad the segment behaves compared to other days uses the data obtained from the outlier detection algorithm as an input. The structure of this data is given in table 5.1. The data has approximately 6 months of event log which corresponds to approximately 25 weeks. However, some segments only operated a few times. For example, there could be a segment which operated on only 2 days throughout the 6 months of data. We can think approximately but not necessarily 25 of each weekday of the baggage movement data is available for each segment, such as 26 Mondays or 14 Tuesdays for segment A.

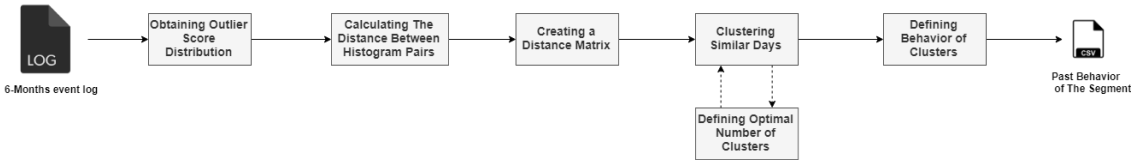


Figure 6.1: Structure of the Representative Clustering Analysis

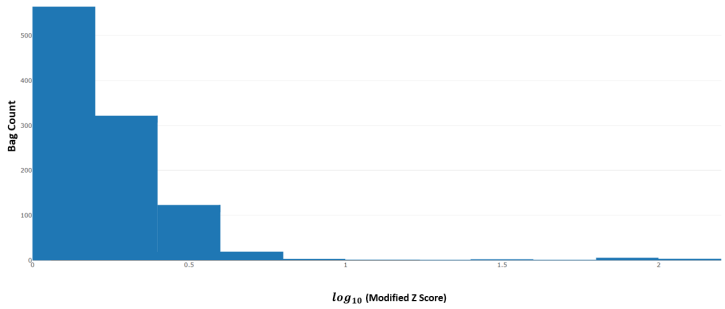
The technique to find how bad the segment behaves compared to other days is executed in multiple steps. Figure 6.1 shows all the required steps required for the technique. In the first step, we obtain the outlier score distribution for each existing day for the segment. Secondly, we convert these distributions into histograms to be able to measure the distance between different days. Then, these outlier score histograms are compared via a distance metric because in this way we can determine which days have similar bag behavior by looking at the distance between them. In the next step, a distance matrix is created from the pair of distances between days to be used as input for the clustering algorithm. Finally, by using the distance matrix, we perform clustering to group all the similar behaving days. To select the best performing clustering we conduct a goodness-of-fit test for the *number of outlier parameter* which corresponds to the information of how many clusters the clustering algorithm will create. Based on the results of the test, we select the optimal cluster number for the segment. In the end, we obtain clusters of days, each of them indicating a different segment behavior such as the days that the segment performed at its best and the days that it performed poorly by ranking the clusters according to their average outlier score. We conduct this analysis for each segment of the baggage handling system and receive a file containing each segment that consists of all the clusters along with the information of behavior they represent. In the following sections, we will elaborate on the steps of this technique.

6.1.2 Obtaining the Outlier Score Distribution

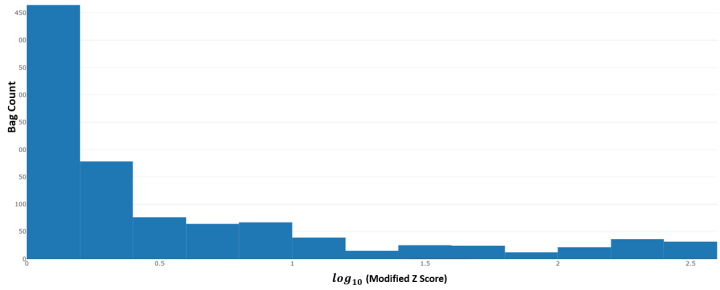
The outlier score distribution is obtained with the following steps:

- Given a dataset $B_{day,segment}$, i.e. list of bags for a given day, we calculate the modified Z-score for each bag $b \in B_{day,segment}$ using algorithm 1.
- We create an histogram of all modified Z-score values for all bags $b \in B_{day,segment}$

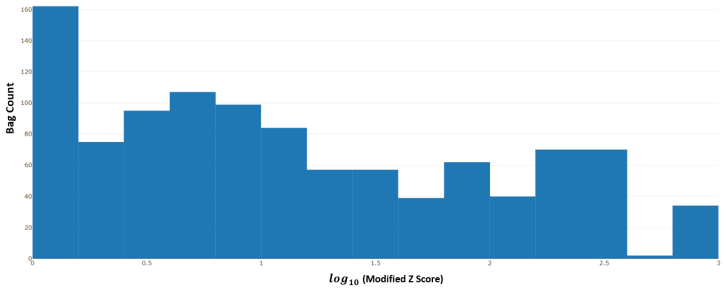
Figure 6.2 displays the outlier score histograms from 3 different days in segment A. According to this image, although we observe deviations, day 1 seems to have a good day since most of the points are accumulated around the logarithmic outlier score of zero. Day 2, however, expresses a bag behavior which is worse than day 1 since the bags are more spread out. On this day, we observe more bags that deviates from zero and have higher outlier scores. Day 3 seems to be the worst behaving day among the three. The outlier score scale is large and many bags deviated from zero. Based on this figure, we can say that Day 3 could have been a disastrous day for the segment.



(a) Day 1



(b) Day 2



(c) Day 3

Figure 6.2: Daily distributions of outlier scores for segment A

6.1.3 Calculating Distance Between Histograms of Outlier Score

To be able to cluster the similar days together, we first need to measure the distance between every pair of days to compare how similar some days are. If the distance between two days is zero then we can conclude these days are the same in terms of outlier behavior. The bigger the distance is, the more different the day pair is. There are many distance measures as discussed in Chapter 2. For the analysis, we use the *Wasserstein metric* to measure the similarity between two histograms for the following reasons:

- It is widely used to compare histograms.
- It computes many pointwise distances in the histogram making it robust to noise. Since we observe long tails indicating a noise in the outlier score distributions, this feature becomes important.
- Histograms do not have to be the same size unlike in other distance metrics. The number of bags in the segment is different in different days, therefore, the days are not equally sized. This leads to their histograms to not have equal bin size. To compare histograms, the equal bin size is a requirement for most of the methods. Therefore, the Wasserstein metric becomes most fit for our purpose.

6.1.4 Creating Distance Matrix from the Distance Metric

A distance matrix is a table to show the distance between each pair of observations. For a given segment, we aim to compare any two days from the set D of days based on how similar they are. We do that by defining a DxD matrix where we compute the distance for each pair (d1,d2) of days. In this matrix, every cell corresponds to the distance between the days of the segment. The diagonal of the matrix becomes 0 since it is the distance between the same days. We will need this distance matrix for the clustering algorithm later on.

Figure 6.3 shows the distance matrix of all the days that exist in the event log of segment A. Each pair of days are assigned a distance measuring how similar they are. For instance, Day 1 is most similar to Day 2 and it differs greatly from Day 26. Only the upper triangular matrix is calculated and shown because it gives the same information as the lower triangular matrix.

	Day 1	Day 2	Day 3	...	Day 26
Day 1	0	0.78	1.52	...	1.96
Day 2		0	0.53	...	0.72
Day 3			0	...	0.085
⋮				0	...
Day 26				...	0

Figure 6.3: Distance matrix of days that Segment A operated on

6.1.5 Clustering the Similar Days of the Segment

To summarize the historical data, we cluster similar behaving days. The assumption about the baggage handling system is that it does not operate arbitrarily but it shows certain characteristics under specific recurring load scenarios. Hence, we exploit this assumption by grouping similar days together. This technique will help the stakeholders to understand:

- Which sets of days are similar for the system.
- Which types of outlier behaviors the system shows regarding the clusters of similar outlier behaviors.

We cluster the days by using the distance matrix that has been obtained in the previous step. There are many clustering methods as we discussed in Chapter 2. Our requirements for the clustering method are as follows:

1. We obtained a meaningful distance matrix in the previous step. So we seek an algorithm that can take the computed distance matrix as an input.
2. Although we are clustering, we want to see the structure in the cluster with respect to the similarity to let the user explore and understand the data if necessary.

The first requirement excludes using methods such as k-means since it does not support to use the distance matrix. The second requirement is satisfied by the usage of *dendograms*. Hence, we use the *Agglomerative Clustering* method since it meets all requirements for our analysis.

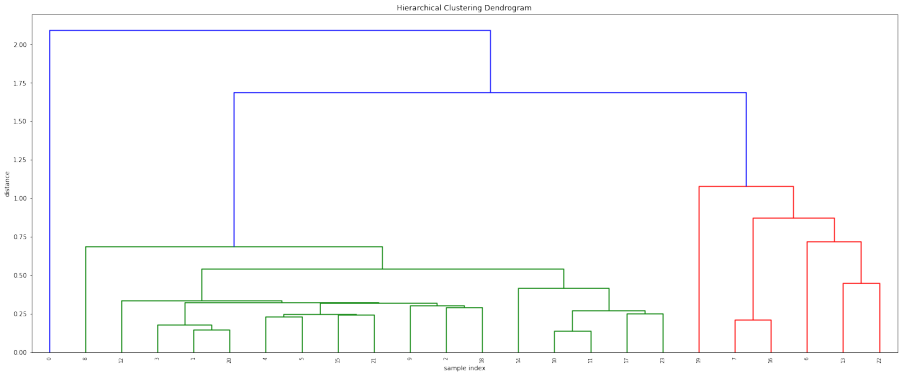


Figure 6.4: Dendrogram of Segment A.

Figure 6.4 shows the dendrogram of days in segment A. According to the dendrogram, we see 3 different main clusters indicating three different types of behavior observed in the segment A. As seen on the dendrogram, Day 1 is represented as a blue line,and is a cluster on its own, meaning only day 1 is showing a behavior that is different than the rest of the days. We see a huge cluster formed in the middle of the dendrogram colored in green and a cluster with a pile of days colored as red.

Figures 6.5 , 6.6 and 6.7 show the outlier score distributions of some days that belongs to the blue, green, and red clusters respectively. From these distributions we see the Agglomerative Clustering Algorithm is performing well since the distributions within a cluster look similar and distributions in different clusters look different as in Figures 6.5, 6.6 and 6.7 . On the other hand, each cluster is representing a different behavior in segment A. We see that the clusters are formed well since they are different from each other.

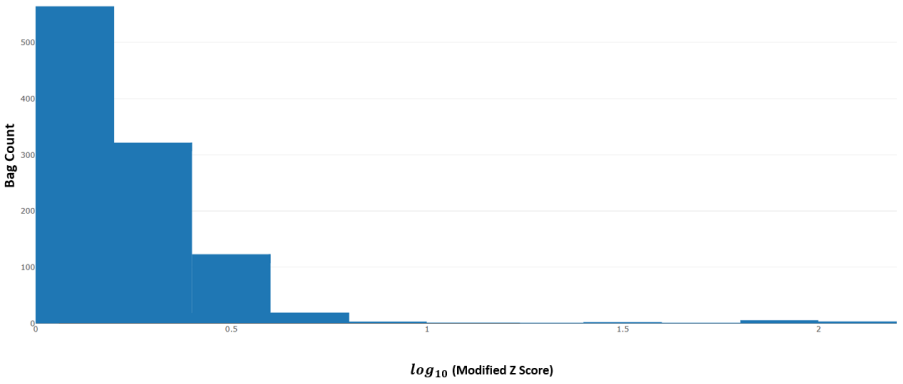
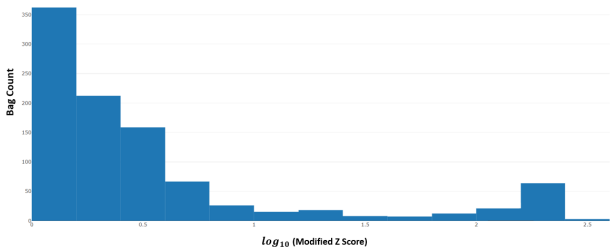
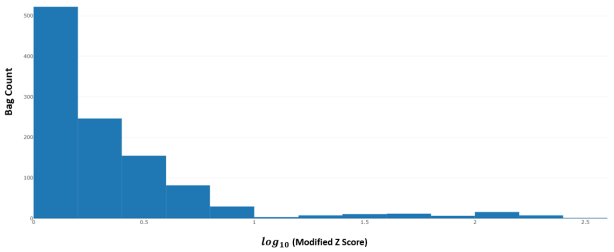


Figure 6.5: Outlier Score Distribution of Blue Cluster

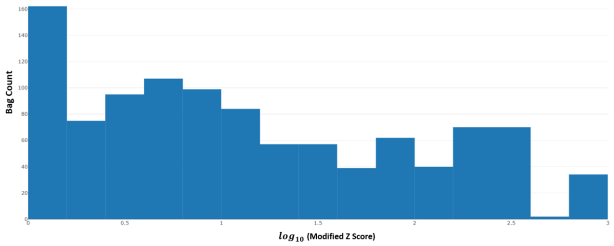


(a) Day 4

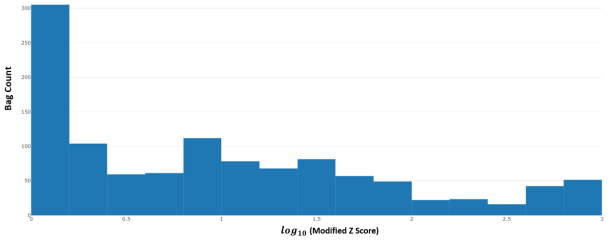


(b) Day 5

Figure 6.6: Outlier Score Distribution of Green Cluster



(a) Day 6



(b) Day 7

Figure 6.7: Outlier Score Distribution of Red Cluster

As a result of the clustering algorithm, each day has been assigned a cluster number indicating which cluster it belongs to. All the days that have the same cluster number are merged to represent the unique behavior of the segment. The outlier score distribution of each day is also merged according to their clusters to form one distribution for the entire cluster. From the merged distribution, we obtain the average outlier score along with the minimum and maximum values of the outlier score. These attributes of distribution are later used to comprehend how well the cluster is behaving. Apart from the distribution attributes, we also obtain information on how many bags and how many days belong to the same cluster. Figure 6.8 shows the output of the clustering algorithm. The days that belong to the same cluster are merged and a combination of attributes from each day becomes an attribute of the cluster. For example, outlier scores of each day that form the cluster are merged to obtain outlier score distribution for the cluster. The result of the clustering algorithm in Figure 6.8 displays that 5 clusters have been formed in the segment. However, the number of cluster varies from segment to segment taking into account the optimal number of cluster. We will explain how to get this number in the following section.

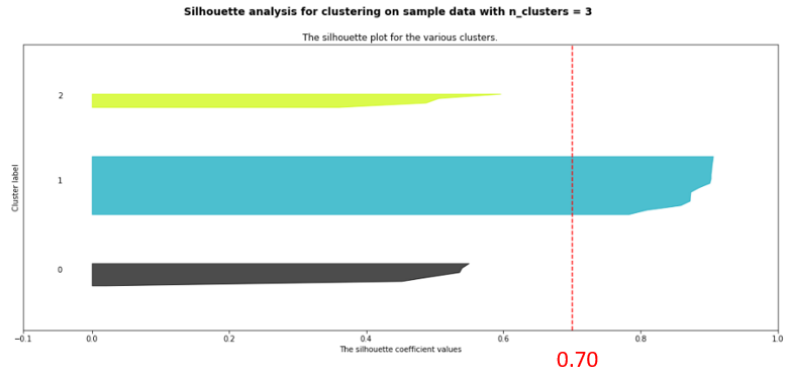
Cluster	Distribution	Average Outlier Score	Minimum Outlier Score	Maximum Outlier Score	Bag Count	Day Count	Cluster Rank
0	[1.7, 0.3, ...,0.16]	**2.25**	0	119.43	39498	14	1
1	[1.94, 1.69, ...,0.52]	**2.47**	0	892.28	6040	2	2
2	[0.64, 0.45, ...,1.99]	**28.26**	0	10863.28	13092	4	4
3	[41.09, 97.36, ...,23.04]	**55.83**	0	92796.62	8407	2	5
4	[1.09, 1.51, ...,0.14]	**4.48**	0	2087.33	4393	2	3

Figure 6.8: The result of the Clustering Algorithm

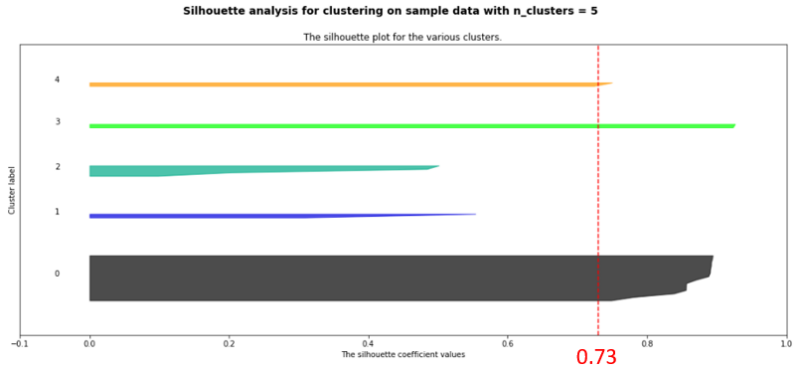
6.1.6 Defining Optimal Number of Clusters

The agglomerative Clustering algorithm requires a distance threshold to cut the tree or it has to be given a constant cluster number so that the algorithm could stop after having the defined number of clusters. For the first method, the dendrogram that the algorithm produces could be investigated to see at what distance we should cut the tree. But since we have thousands of segments in the baggage handling system, manually investigating each dendrogram becomes an impossible task. Hence, we want to automatically determine the right cluster number for each segment. Now the question is how to determine whether a number of cluster is best for a given segment? Because if we force a cluster number to the algorithm, we may classify some days into wrong clusters which leads to poor performing clustering results. For instance, while one segment is represented the best with 3 different clusters because we only observe 3 different types of behavior in the past, other segments could be best represented with 5 clusters.

To solve the issue stated above, we propose to simply try different numbers of clusters and keep the clustering with the highest quality. To measure the quality of the cluster, we use the **Average silhouette method**. The average silhouette method computes the silhouette score for each day in the clusters and takes the average of this score for the entire cluster. The average silhouette score is calculated for each number of clusters and the optimal number of cluster is found to be the one that has the maximum silhouette score. In Figure 6.9, we see the comparison of two different values of the *number of cluster parameter*. For the particular segment, the average silhouette score is found out to be 0.70 and if we assign 5 as a *number of cluster parameter*, the average silhouette score becomes slightly higher than assigning 3 as a *number of cluster parameter*. We repeat the same calculation for the set of different values of the *number of cluster parameter*. Since there is a maximum of 26 different days for each segment, we have to keep the number of clusters limited. We cannot allow the algorithm to separate the segment into 26 clusters meaning each day forming a cluster on its own. For a similar reason, we also cannot allow the segment to form only one cluster which is the combination of all the days. These edge cases do not add any value to our analysis, therefore, we exclude some values from the set of the possible number of clusters. In our analysis, we tried the algorithm with values starting at 3 until 10 for the *number of cluster parameter*, meaning a segment can have a minimum of 3 and a maximum of 10 clusters. As stated in the business motives in Chapter 3, the reason to start at 3 is that stakeholders are mostly interested in seeing the best behaving, standard behaving, and the worst behaving days for the segment. Hence, we need at least 3 clusters to represent these behaviors. The reason for selecting 10 as a maximum is that it has been agreed on with engineers as a reasonable maximum and still allows for grouping days together as we have maximum 26 days for a segment.



(a) Number of cluster : 3



(b) Number of cluster : 5

Figure 6.9: Silhouette score comparison on segment A

6.1.7 Defining the Behavior of Clusters

We just obtained an optimal number of clusters for each segment to summarize for each day what kind of outlier behavior it shows. However, the clusters are currently unlabeled and not meaningful for a user. For this reason, the need to differentiate obtained clusters to represent various behaviors for the segment arises. If the average outlier score for each cluster is calculated, we can rank the clusters according to their average outlier score to see which cluster represents the good days for the segment and which represents the worst ones. To illustrate the ranking of clusters, we can look at the Figure 6.10. **Cluster 0** has been assigned with a rank **1** because it has the lowest average outlier score. On the contrary, **cluster 3** has been assigned with a rank **5** since the average outlier score for this cluster is the highest among all. For segment A, cluster 0 represents its best behavior while cluster 3 corresponds to its worst behavior.

Cluster	Distribution	Average Outlier Score	Minimum Outlier Score	Maximum Outlier Score	Bag Count	Day Count	Cluster Rank
0	[1.7, 0.3, ...,0.16]	**2.25**	0	119.43	39498	14	1
1	[1.94, 1.69, ...,0.52]	**2.47**	0	892.28	6040	2	2
2	[0.64, 0.45, ...,1.99]	**28.26**	0	10863.28	13092	4	4
3	[41.09, 97.36, ...,23.04]	**55.83**	0	92796.62	8407	2	5
4	[1.09, 1.51, ...,0.14]	**4.48**	0	2087.33	4393	2	3

Figure 6.10: Cluster ranking in segment A

While the outlier score can rank the clusters for one segment, different segments will have a different number of clusters. A segment of rank 3 with 7 clusters performs better than a segment of rank 3 clusters. Thus, instead of ranking the N clusters for a segment in an absolute way, we assign each cluster a normalized ranking score of $rank/N$. In this way, we rank clusters from 0.11 to 1 while ranks near 0 indicate good-behaving segments and ranks near 1 indicate worst-behaving segments. Figure 6.11 shows the final output of the clustering algorithm where we obtain the standardized ranking by dividing each cluster rank by the cluster number. For example, this particular segment in Figure 6.11, we have 5 clusters and to obtain the standardized rank we divide each rank by 5.

Cluster	Distribution	Average Outlier Score	Minimum Outlier Score	Maximum Outlier Score	Bag Count	Day Count	Cluster Rank	Cluster Standardized Rank
0	[1.7, 0.3, ...,0.16]	2.25	0	119.43	39498	14	1	0.2 → $\frac{1}{5}$
1	[1.94, 1.69, ...,0.52]	2.47	0	892.28	6040	2	2	0.4 → $\frac{2}{5}$
2	[0.64, 0.45, ...,1.99]	28.26	0	10863.28	13092	4	4	0.8 → $\frac{4}{5}$
3	[41.09, 97.36, ...,23.04]	55.83	0	92796.62	8407	2	5	1 → $\frac{5}{5}$
4	[1.09, 1.51, ...,0.14]	4.48	0	2087.33	4393	2	3	0.6 → $\frac{3}{5}$

Figure 6.11: Output of Standardization of ranking

Figure 6.12 shows the general scale for the clusters. Based on how the scale is defined, if the standardized cluster ranking is between 0 and 0.33, we can say that this cluster represents the best behavior of the segment while we can also conclude if the standardized cluster ranking is between 0.67 and 1 we would say it represents the worst behavior of the segment.

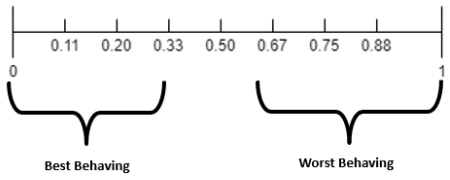


Figure 6.12: Standardized cluster ranking

To summarize, we analyzed and ranked the behavior of each segment and day relative to all other days. We classified each day with respect to all other days by clustering, and each cluster is on a scale from 0 (best) to 1 (worst). Having done all these analysis for the historical data, now we address the case of classifying a new day regarding how good or bad it performs in the sense of outliers.

6.2 Assigning Appropriate Cluster to Daily Outlier Behavior

When the new data arrives we need to define how it is behaving regarding the historical data. The idea to do that is to use the clustering from Section 6.1 as the classes into which the new day shall be categorized as:

1. For the new (day,segment) pair, find the cluster that is most similar regarding its modified Z-score distribution.
2. The normalized rank of this most similar cluster is the severity of the outlier behavior on that day.

Now, we present the steps to compute this:

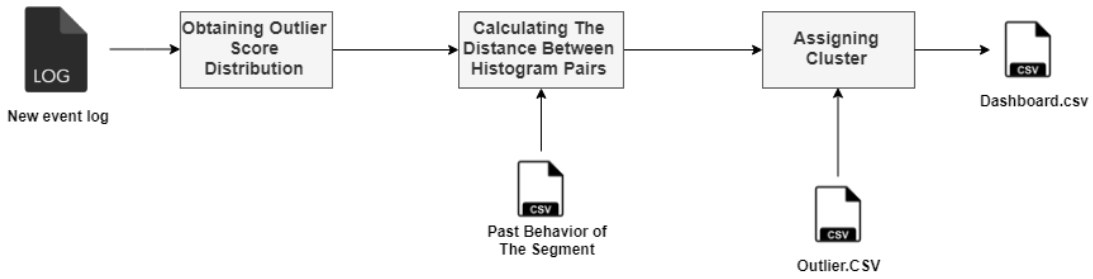


Figure 6.13: Structure of assigning appropriate clusters

In the first step, the outlier score distribution for the new event log is obtained. Secondly, this distribution is converted into a histogram. Then, we measure the distance between the histogram obtained from the new event log and each of the histogram of distribution from the clusters in the **Past Behavior of The Segment** file. In the next step, the cluster that has the minimum distance to the outlier score distribution of the new event log is selected. As a result, we assign this cluster to the relevant segment of the new event log. Relating past behavior with a new event log allows us to see how the new day is behaving. For example, if the segment from the new event log is assigned the cluster that is ranked as 1, we would understand that this segment is showing its worst behavior we have seen so far on this new day.

In the following subsections, we will explain each of the steps from Figure 6.13.

6.2.1 Obtaining Outlier Score Distribution

When an event log from a new day comes, we first obtain the outlier score distribution for each segment that exists in the event log. This step is necessary to see how the segments from the new day are behaving. We need to convert the outlier score distribution to a histogram to be able to compare the past and the present behavior. For that, we need to have the shape attributes of both distributions. Therefore, we convert all the outlier score distributions from the segments to histograms using the same algorithms as used in the historical data.

6.2.2 Calculating the Distance Between Histogram Pairs

In this step, for each segment, we obtain a file that contains the past behavior clusters. We compare the histograms of outlier scores from each cluster to the histogram of the outlier scores from the new event log. A histogram of the cluster is formed from combining or including the data from all days in the cluster. Since the number of bags in the cluster and the new day is different, while comparing two histograms we convert both of them to show the value of the probability

density function at the bin rather than the number of samples in each bin. Figure 6.14 shows all the calculated distances between the segment outlier score distribution from the new event log and the past behavior cluster distributions of that particular segment. From the distances, we see that this segment in the new event log is mostly similar to Cluster 2.

	Distance
Cluster 1	0.72
Cluster 2	**0.12**
Cluster 3	0.86
Cluster 4	1.22
Cluster 5	1.99

Figure 6.14: Calculated distances between all the clusters from the past behavior of the segment and the new event log

We then select the cluster with the minimum distance to the current dataset as the cluster that corresponds best to the current day. To illustrate this we can revisit Figure 6.14. Cluster 2 seems to have the minimum distance with a new event log. Therefore, we conclude that the segment shows behavior that is very similar to the behavior captured in Cluster 2.

6.2.3 Assigning Cluster

After finding the correct cluster for the segment, we can assign this cluster to the current behavior of the segment. We can assume that the current behavior shows indications from the past behavior. For example, if cluster 2 in 6.14, corresponds to the worst behavior of the segment, we can assume that the current behavior of the segment is also showing the worst behavior.

In this step, the file, Outlier.CSV, that is created as a result of the outlier detection and classification algorithm is given as an input. The structure of the Outlier.CSV file is given in Table 5.2. After each segment in the Outlier.CSV file has assigned with an appropriate cluster, another CSV file with the information of clusters is created and it is called **Dashboard.CSV**. The structure of Dashboard.CSV is represented in Figure 6.15. This file contains the information about each segment including the cluster it belongs to and aggregated data of the segment and cluster behavior such as total/average blockage duration and average outlier score for both segments and the cluster it belongs to along with the total number of each outlier type in segments. Later in the analysis, this file will be used to show the problematic segments based on the aggregated values it contains.

Activity	Average Duration (Minute)	Total Blockage Duration	Total Number of Bags in Blockage	Average Blockage Duration Per Bag	Isolated Bag Count	Fast Bag Count	Total Number of Outliers	Total Number of Bags	Cluster	Cluster Average Outlier Score	Segment Average Outlier Score	Importance
Segment A	2.22	458.33	9	49.51	0	0	9	734	1	8.94	7.41	66.69
Segment B	1.28	362.1	36	13.59	1	0	37	416	1	13.65	15.87	587.19
Segment C	0.54	109.72	3	35.36	0	0	3	900	0.5	1.51	1.79	5.37
...	...	...		...					...	...	...	
Segment Z	0.83	0	0	0	1	0	1	4893	0.2	1.21	1.33	1.33

Figure 6.15: Dashboard.CSV

In the following chapter, we will explain the implementation of the methods that we have described so far in Chapters 5 and 6.

Chapter 7

Implementation

Chapter 5 presented our approach to detect and classify outliers in the baggage handling system. Chapter 6 uses the result of the outlier detection method to create a standard representation of baggage behavior. Using the methodologies presented both in Chapters 5 and 6, this chapter explains the implementation process as an extension of Performance Spectrum Miner. Firstly, this chapter explains the implementation of the outlier classifier in Section 7.1. Secondly, the creation of a visual dashboard that shows the problematic segments of the baggage handling system is discussed in Section 7.2.

7.1 Implementation of the Outlier Classifier In the PSM

As stated in Chapter 3, visualizing the problems happening in the system is one of the main objectives of the thesis. To visualize the outliers in each segment, we use the Performance Spectrum Miner because valuable insights regarding the performance of segments can be obtained with it. To visualize the outlier bags in the Performance Spectrum Miner, the implementation of a classifier regarding the outliers is a necessary step.

As explained in Chapter 2, the color of the lines comes from a classifier in the PSM. Hence, we need to implement a classifier that allows classifying the outlier bags according to outlier patterns that we obtained in the 5.3 such as blocking bag, stuck bag, isolated bag, and fast bag. Accordingly, we can see the types of outliers in different colors in PSM. To classify all the traces into a different type of outliers, a dictionary that maps the ID of the bags to one of those classes can be built and used within a classifier.

We implement the outlier type classifier as a result of a series of steps. The steps of the implementation are given in Figure 7.1. The first step is using the event log that has been obtained from the initial reports as an input for the outlier detection Python pipeline. The pipeline executes the series of outlier detection and classification methods as explained in Chapter 5. As a result, the pipeline produces a CSV file (Table 5.2) where every bag is assigned either a type of outlier or a normal behaving bag in the directory of the PSM data on the disk. The resulting CSV file is used in the outlier classifier script to implement the classifier. The output of the classifier is defined in a separate JAR file. Later, this JAR file is used in the Performance Spectrum Miner with a class-path injection to visualize all the outliers in the given event log.

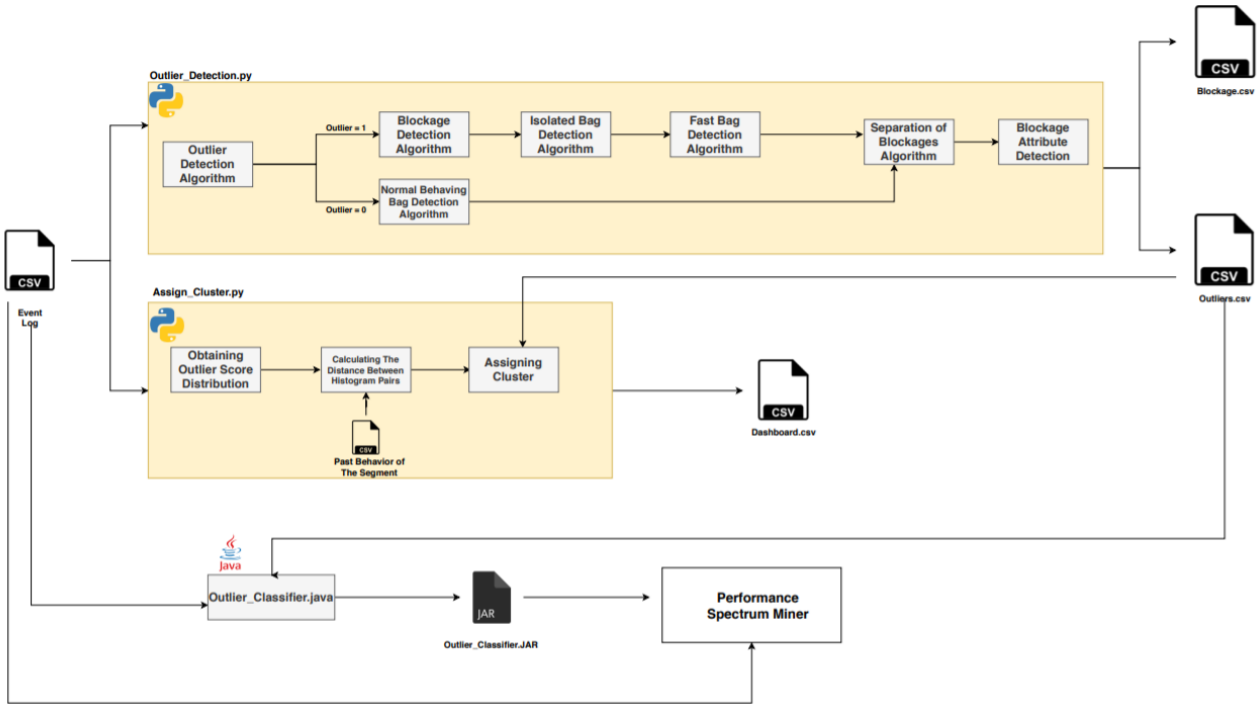


Figure 7.1: Implementation of Outlier Classifier

In order to implement the classifier, the resulting CSV file of the *outlier_detection.py* is required. Apart from the resulting CSV file of the outlier detection pipeline, we also need the original event log that the outlier detection analysis is based on. *Outlier_Classifier.java* takes these two files as inputs and it creates a directory that maps the triple (*bag ID*, *segment name*, *timestamp*) from the original event log to the Outliers.CSV file to get the outlier type information. As a result, for each event in the original event log, the classifier classifies the event to its outlier type. Then, a JAR file is created from the *Outlier_Classifier.java* to be used in the Performance Spectrum Miner. As a final step, the performance spectrum miner uses the created JAR file with a class-path injection as displayed in Figure 7.2 and visualizes the event log with the outlier classifier.

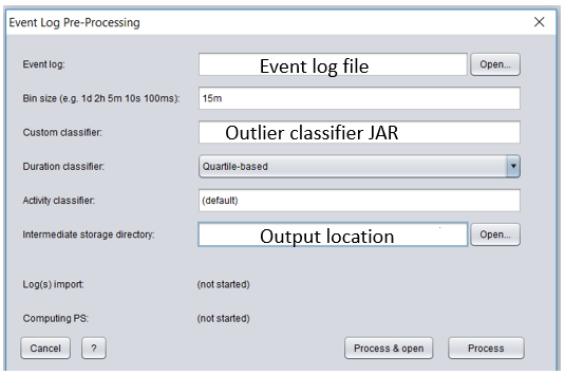


Figure 7.2: Class-path injection of *Outlier_Classifier.JAR* file

Figure 7.3 displays the result of the outlier classifier in the PSM. Each bag is represented with an outlier classifier and is assigned a color according to it. Figure7.4 shows the colors that are associated with the outlier classes. Normal behaving, blocking, stuck, isolated and fast bags

receive the colors of grey, yellow, green, red, and dark blue respectively. In Figure 7.3, we see for each segment most of the bags are colored grey, meaning they are behaving normally in their segment. However, there are some blockages visible on segments B, C and D. As seen on the figure, bags in the blockages are slower compared to normal behaving bags as the bags in the blockages have a bigger projection on the time axis.

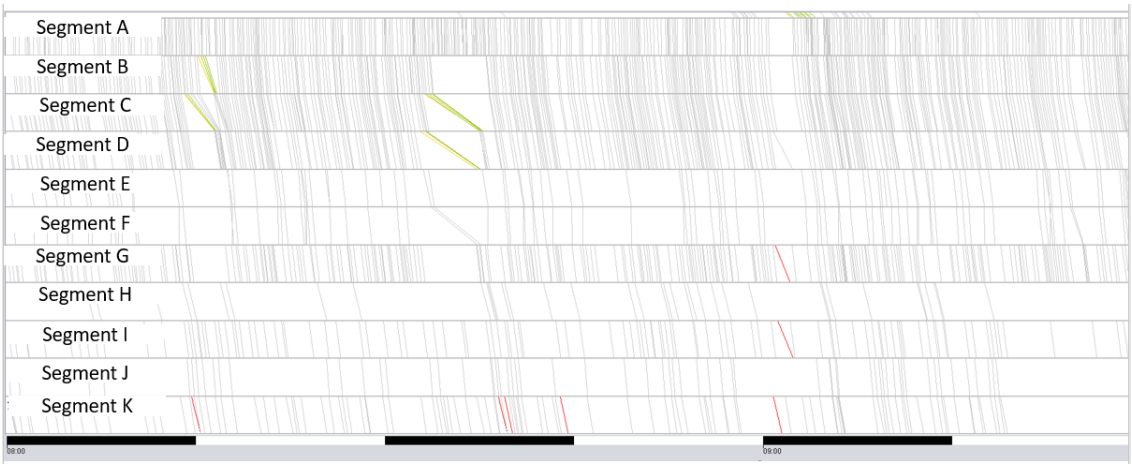


Figure 7.3: The result of the outlier classifier

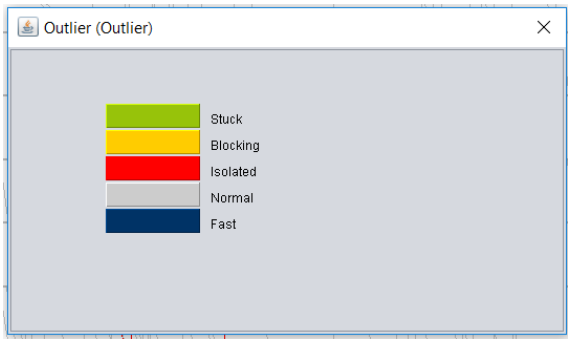


Figure 7.4: Outlier Class Colors

7.2 Visual Dashboard for Problematic Segments

To work out the problem of detecting problematic segments of the system more clearly as stated in Chapter 3, we first need to visualize these problematic segments through a dashboard. Because every segment is classified and clustered but there are thousands of segments with a lot of data and data features, engineers need help to see all points of interest starting with the most severe ones. Thus, we create a dashboard inside the performance spectrum miner. Dashboard creation is formed by three steps as displayed in Figure 7.5. In the first step, a dashboard that shows the problematic segments is created. In this step, the file (Figure 6.15) that has been obtained as a result of the outlier detection and clustering pipelines is used as an input. The second step is to create another window to give information about the blockages that happened in problematic segments. The **Blockage.CSV** file (Table 5.3) that has been obtained before is used as an input in this step. The third and final step is visualizing these blockages on the Performance Spectrum Miner.

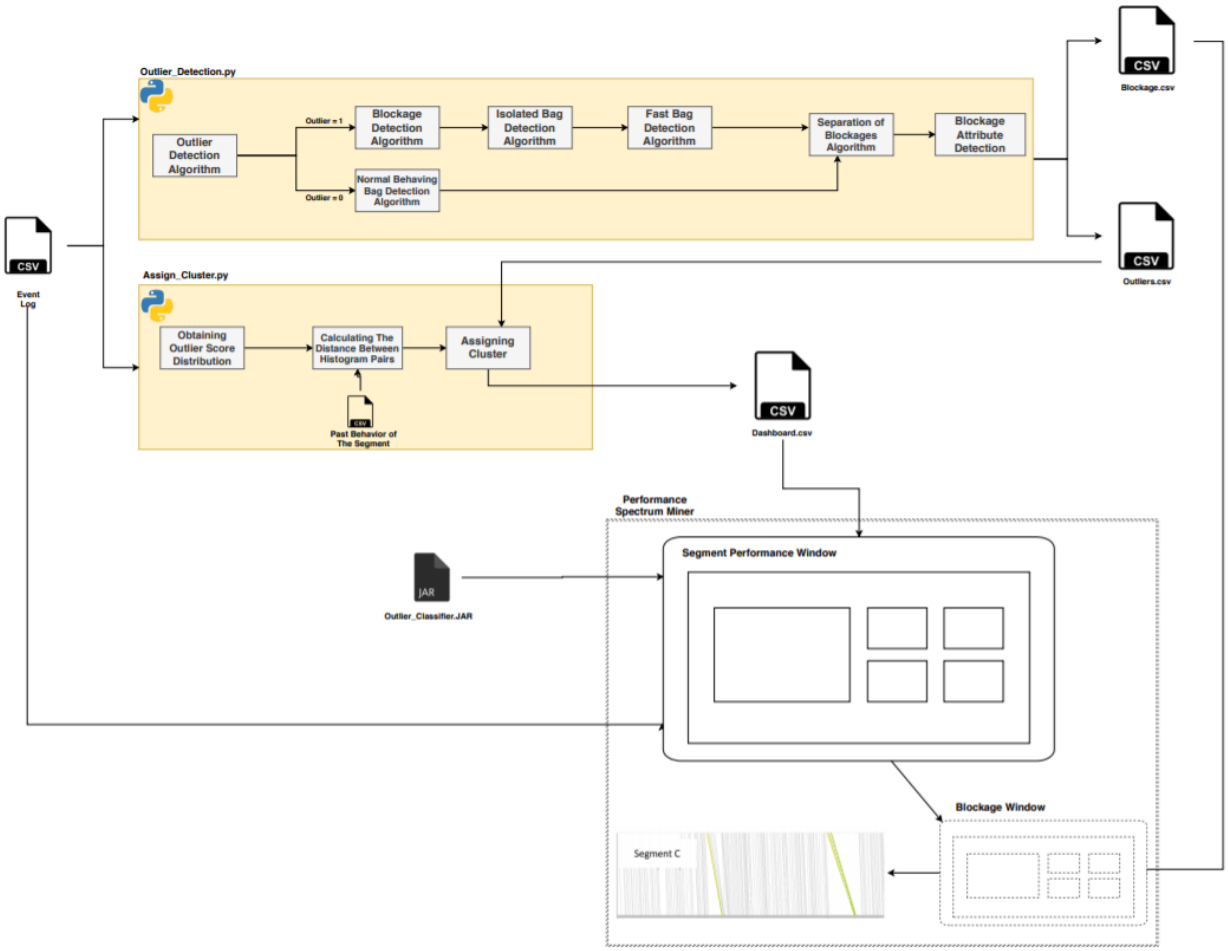


Figure 7.5: Implementation of the dashboard

7.2.1 Displaying Problematic Segments

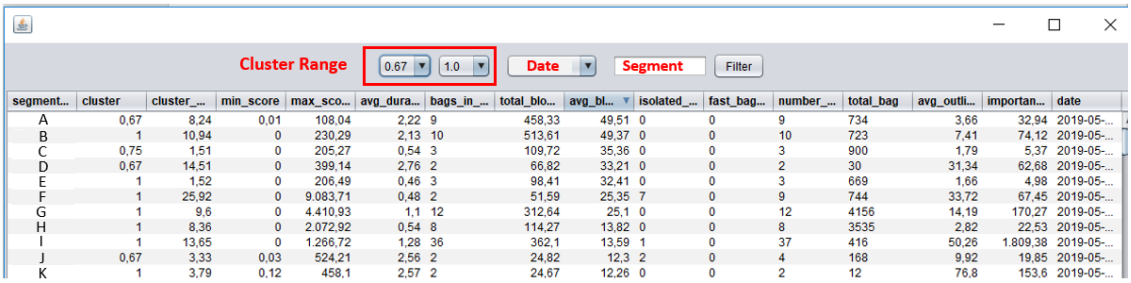
In the baggage handling system, some segments suffer from outlier behavior more than the others. Throughout the analysis we have called these segments *problematic segments*. Stakeholders of this analysis want to see these problematic segments automatically based on the outlier behavior that the segments have. Hence, a dashboard is needed to display all these problematic segments for the event log they upload to the PSM. For this purpose, we have implemented a separate window inside the performance spectrum miner. In this window, a detailed analysis that has been obtained as a result of the outlier detection and clustering pipelines is displayed.

Figure 7.6 shows the dashboard for the problematic segments. This dashboard has several functionalities including:

- Ability to filter the segments based on the behavior they show: Users can select the type of segments based on the performance using the cluster range. In Chapter 6, we explained the standardized cluster ranking for the segments. Each segment is assigned with a cluster that represents one of its past behaviors. To make the segments comparable, we introduced a standardized ranking scale that has values between 0 and 1. Having the average outlier score of the segment as a baseline, the rankings near-zero corresponds to the best behavior of the segment while rankings near 1 represent the worst behavior of the segment. Thus, in the dashboard, users can select the cluster range to filter the segments based on their behavior.

The dashboard in Figure 7.6 shows all the segments of the baggage handling system that have the cluster ranking between 0.67 and 1 meaning we only display the segments that express its worst behavior for the given day.

- Ability to choose the date to display results: Since the outlier detection, outlier pattern classification and the cluster assigning is conducted separately for each day of the event log, the dashboard allows selecting different dates to display if the uploaded event log consists of more than one day.
- Ability to choose problematic segment criteria: Users are allowed to filter the segments they are interested in based on performance for a given date. As a final functionality, users can sort the table based on any variable they want. In this way, they can create their own definition of a problematic segment. For instance, for some users, a problematic segment means if most of the bags express abnormal behavior. However, for other users, the average blockage duration of the segment could imply a problem. This sorting functionality gives the user the flexibility to choose any criteria they want to see as an indicator of a problem.



segment...	cluster	cluster_...	min_score	max_sco...	avg_dura...	bags_in_...	total_blo...	avg_bl...	isolated_...	fast_bag...	number_...	total_bag	avg_outli...	importan...	date
A	0,67	8,24	0,01	108,04	2,22	9	458,33	49,51	0	0	9	734	3,66	32,94	2019-05-...
B	1	10,94	0	230,29	2,13	10	513,61	49,37	0	0	10	723	7,41	74,12	2019-05-...
C	0,75	1,51	0	205,27	0,54	3	109,72	35,36	0	0	3	900	1,79	5,37	2019-05-...
D	0,67	14,51	0	399,14	2,76	2	66,82	33,21	0	0	2	30	31,34	62,68	2019-05-...
E	1	1,52	0	206,49	0,46	3	98,41	32,41	0	0	3	669	1,66	4,98	2019-05-...
F	1	25,92	0	9.083,71	0,48	2	51,59	25,35	7	0	9	744	33,72	67,45	2019-05-...
G	1	9,6	0	4.410,93	1,1	12	312,64	25,1	0	0	12	4156	14,19	170,27	2019-05-...
H	1	8,36	0	2.072,92	0,54	8	114,27	13,82	0	0	8	3535	2,82	22,53	2019-05-...
I	1	13,65	0	1.266,72	1,28	36	362,1	13,59	1	0	37	416	50,26	1.809,38	2019-05-...
J	0,67	3,33	0,03	524,21	2,56	2	24,82	12,3	2	0	4	168	9,92	19,85	2019-05-...
K	1	3,79	0,12	458,1	2,57	2	24,67	12,26	0	0	2	12	76,8	153,6	2019-05-...

Figure 7.6: Problematic Segment Dashboard

7.2.2 Displaying Blockages

Blockages form an important part of the analysis. Most of the problems in the baggage handling system cause blockages to occur. Hence, investigating where and when the blockages occur carries a great value for the stakeholders. To display all the information regarding blockages, a separate frame has been created. This frame is displayed if someone clicks the segment on the problematic segment dashboard. This new dashboard displays all the blockages that happened on the selected segment and the day. This dashboard is called the *Blockage Dashboard* and it shows the **Blockage.CSV** file that has been created as a result of the outlier detection and pattern classification algorithms. The dashboard shows all the attributes regarding the blockage such as the number of bags involved, the blockage duration, and the start and end timestamp of the blockage.

Figure 7.7 displays the blockage dashboard. This dashboard shows all the blockages of segment A for the selected date. The attributes of the blockages are also displayed in this dashboard such as Pattern ID, which represents the unique blockage ID for each blockage. Blockage duration and the number of bags in the blockage attributes are also present to express how serious the blockage is. The average duration is another field that the user can use to find the importance of the blockage. It corresponds to the average duration that the bags in the blockage took in the segment. Finally, the attributes of time namely, the starting and end timestamp of the blockage allow the user to see the exact time the blockage has occurred.

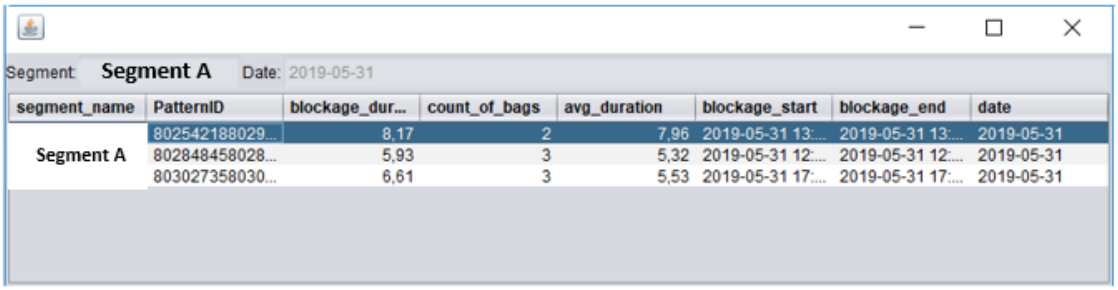


Figure 7.7: Blockage Dashboard

The starting and end timestamp of the blockage also allow for additional functionality in the blockage dashboard. By clicking on the blockages, users can see the selected blockage on the Performance Spectrum Miner. This functionality takes the time interval of the blockage and shows this interval on the PSM. In this way, users automatically can see where the blockage has happened without searching for it in the PSM. Figure 7.8 shows this functionality. When a user clicks on the blockage on the third row of the blockage dashboard, PSM automatically displays this blockage.

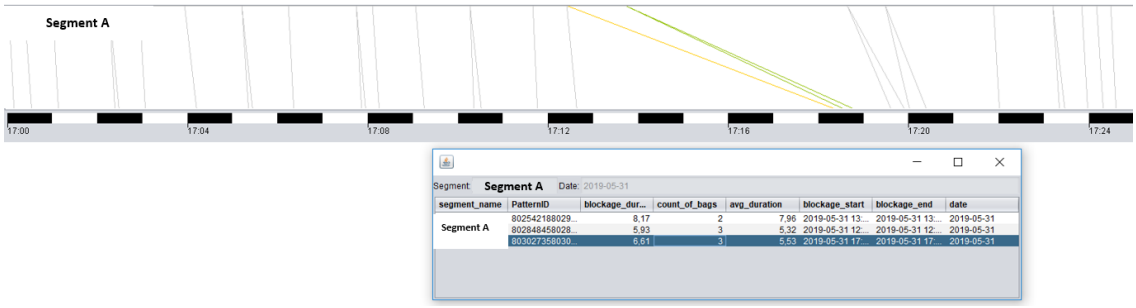


Figure 7.8: Visualizing blockages on PSM

In this chapter we explained the implementation of the methods given in Chapters 5 and 6 as an extension of the PSM. In the following chapter, we will use the extended PSM to evaluate the results obtained from our methods with the stakeholders.

Chapter 8

Evaluation

This chapter presents the result of the steps that we performed to evaluate the methods that were introduced in the Chapters 5 and 6. Section 8.1 describes the experimental setup for the evaluation. Section 8.2 describes the analysis questions that need to be answered during the evaluation phase. Section 8.3 shows the result of every analysis question. Finally, Section 8.4 explains how do findings add value to the day-to-day practice of stakeholders.

8.1 Experiment Setup

This section explains the required experiment steps to conduct the analysis using the methods we described in Chapters 5 and 6. The aim of this project was to automatically identify which part of the baggage handling systems suffer from abnormal operations and under which conditions these abnormal operations arise. With this aim, we conduct the steps as described in Section 3.3.1. Using these steps, we answer the business questions stated in Section 3.1.3.

The evaluation is conducted using the reports provided by the stakeholders. Historical data analysis is conducted using the 6 months of event data that has been provided at the beginning of this thesis. However, the daily analysis is conducted with a one-day event log that has been provided by the stakeholders for testing purpose. Th one day event log is not contained in the historical data, therefore, it allows us to test our methodology on new data. In this way, we can evaluate with the stakeholders how well the methodology is working.

For the evaluation of our technique, we have involved 2 stakeholders for the analysis questions in 8.2.1, 8.2.2, 8.2.4, 8.2.5 and 8.2.6.

8.2 Analysis Questions

We conduct the evaluation to answer the business questions regarding the performance of the baggage handling system. Hence, our analysis questions are investigated under eight main objectives each of them corresponding to a business motive.

8.2.1 Identifying Abnormal Behavior and Providing Similar Outliers in Segments

1. **Which bags show an abnormal behavior in the segments of the system?**

   Given new data, we want to find all outliers in each segment of the baggage handling system. Stakeholders can investigate the trace of each bag to know in which segments they are labeled as outliers. Hence, if the bag is late for the flight, they could see the segment(s) that cause this.

2. **Which outlier pattern does each bag belong to?**

   Given new data, we want to obtain the outlier patterns as described in 5.3. This is to provide stakeholders, information that they are interested to know about what kind of outlier behavior happens in their system. Differentiating the problems helps them to find a solution for each detected common problem.

8.2.2 Identifying the Segments that Experience More Performance-related Problems

1. **Which parts of the system suffer from the most significant abnormal behavior on the given day?**

   We are interested to find the problematic segments of the system. Problematic segments are the ones that suffer from abnormal behavior more often than the rest. To see the severity of the problem, we use some measures such as average outlier score for the segment and the number of outliers that have been observed in the segment. A combination of these two measures could indicate the severity of the abnormal behavior for the segment.

2. **For a given day, which segments suffer from serious blockage problems?**

   Blockages are the main indicators of the problems that occur in the baggage handling system. Detecting the segments that suffer most from the blockages allows the stakeholders to investigate these segments further to solve the blockage problem. Therefore we intend to find all the segments that have serious blockages for a given day according to some measures for the segment such as average blockage time or the total number of bags in the blockages of the segment.

3. **For a given day, what is the total amount of time bags spent in blockages and which attributes does each blockage have?**

   Blockage attribute detection is an important part of the analysis. The significance of each blockage is not the same even in the same segment, Stakeholders of the analysis might be interested to see more serious blockages and only investigate them further. Therefore, we have introduced blockage measures such as the number of bags in the blockage, the blockage duration, and the average duration that the bags in the blockage have in the segment. These measures help the stakeholder to understand the importance of each blockage they observe in the segment.

4. **For a given day, which segments suffers more from isolated outliers ?**

   Identifying the segments that have more isolated outlier bags, helps us to see segments that often have single accidents rather than blockages. The stakeholders can look at these segments and find the segments that they do not expect to have so many isolated bags, because for some segments having many isolated bags is normal, especially if the segment consists of loops. Bags that are unable to move forward to the next segment are stuck on the loop and this results in a long duration in the segment. Therefore, according to their insight, stakeholders can identify the real single accidents in the system and eventually find ways to avoid them.

8.2.3 Categorizing the Abnormalities in Segments Based on Occurrence

1. **Which parts of the system regularly suffer from serious blockages?**

   Some segments of the baggage handling system suffers from the blockages more often than the other parts. We can call these segments the *hot-spots* of the system. Finding these segments enables the stakeholders to focus on the ways to enhance these segments to avoid more performance-related problems in the future. Hot-spots could be detected based on the average blockage duration of the segment during the 6 months of period. We also know for

how many days we observed a blockage during these 6 months. A combination of these two attributes can give an overview of the blockage problem for the segment. For example, we approximately have 24-26 weeks on the historical data. If we conduct the analysis on Fridays, we would have 24-26 different Friday behavior to investigate. The number of days that the blockages are observed along with the average blockage duration on all Fridays could help us detecting the segments that have a recurring blockage problem.

2. **Which parts of the system shows non-repeating but serious blockages?**

   Sometimes, segments experience a very important blockage that affects many bags on one day. However, we can also find out that the same segment does not suffer from the serious recurring blockages during the 6 months period. Identifying these one-time problems could be useful in the sense of finding disastrous days. Even if the segments that do not regularly experience the blockage problem show some serious blockages on a day. We could conclude that day as a disastrous day if there are more segments also showing the same kind of behavior.

8.2.4 Providing a Ground Truth About the Types of Outlier Behavior in the System

1. **What can be said about the daily behavior of a segment taking into account past behavior?**

   One of the main objectives of the thesis was to find the problematic segments in the baggage handling system. To do that, we have classified the past behavior of each segment with a ranking that shows how well the segment behaves according to its average outlier score. Given a new event log, we match one of these past behaviors with a current one for each segment to understand the overall baggage behavior for the current day. Therefore we can see the segments in the current event log that operates in their worst behavior along with the ones that operate in their best behavior according to the similar past behavior obtained from the initial 6-months event log.

8.2.5 Visualizing the Problems Occurring in the System

1. **Can we visualize the outliers in the PSM?**

   Outliers have been found and classified in chapter 5. To visualize the outliers in each segment of the system, we have created a classifier for the PSM as explained in chapter 7. Hence using the classifier, we are able to visualize each outlier bag with their types represented as different colors in the PSM.

2. **Can we automatically visualize blockages in the PSM?**

   In chapter 7, we have explained the functionalities of the extension we have implemented to the PSM. From the blockage dashboard, we are able to visualize any blockage on the PSM by clicking on it. The time interval of the selected blockage is shown on the PSM. They are there in the main PSM screen by using the classifier we implemented.

8.2.6 Identifying the Properties That Are Leading to the Abnormality

1. **Are there any cause-effect relationship between segments / sequence of segments in the sense of abnormal behavior?**

   We have explained the *die-back effect* in chapter 3. Essentially, the die-back effect happens where the abnormal behavior propagates backward in the system. The die-back effect causes the blockage in one segment to affect the paths that merge into it. The bags in the merging segments either slow down or stop since there is a blockage on the next segment. The die-back effect could be detected by looking at the Performance Spectrum Miner. If the segments

are sorted on the PSM based on their position in the physical system reflecting how they travel, we can see when the blockages cause a die-back effect on the previous segments by visually inspecting the PSM.

2. **Can we find temporary malfunctions in the system that cause abnormal behavior?**

   If a part of the system such as a scanner machine experience a temporary malfunction, all the segments that are related to that part could suffer from the temporary effects. This effect could be detected on the Performance Spectrum Miner if we investigate all the segments that start or ends at that faulty part.

8.3 Results and Interpretation

Now that we have all the analysis questions and approaches, in the following, we will provide results of each of the analysis questions.

8.3.1 Identifying Abnormal Behavior and Providing the Similar Outliers in Segments

1. **Which bags show an abnormal behavior in the segments of the system?**

   We solve this question by using statistics over the outlier labels returned by Algorithm 1. Table 8.1 displays the result of the Algorithm 1 for the given day. This result shows the percentage of outlier bags in the system. However, a bag may be counted twice or more times as an outlier if it is also labeled as an outlier in different segments. We have found that the system consists of 1.32% outliers. As displayed in Table 8.2, that percentage corresponds to 12.792 detected outlier bags for the entire system on the given day.

   Stakeholders do not find detecting the individual outlier bags significantly important. In their existing dashboards, they can see if the bag shows abnormal behavior. However, their analysis is more aggregated. Instead of investigating each segment and finding normal behavior for it, they investigate the area level. For example, in their system, they check the aggregated information for check-in area involving all the check-in desks and conveyor belts whereas our approach allows them to find problems in more specific locations such as on one conveyor belt. Hence, to detect more specific outliers and their exact spot, they find our approach useful.

Outliers	Non-Outliers
1.32%	98.68%

Table 8.1: Outlier and non-outlier percentages in the entire system

Outliers	Non-Outliers
12.792	958.149

Table 8.2: Outlier and non-outlier counts in the entire system

2. **Which outlier pattern does each bag belong to?**

   We solve this question by using statistics over the outlier classification returned by Algorithm 6. Table 8.3 displays the result of the outlier classification algorithm for the given day. We have found that outliers of the system comprise of 18.05%, 40.67%, 38.95%, 2.34% blocking, stuck, isolated and fast bags respectively. As displayed in Table 8.4, that percentages

corresponds to 2.309, 5.202. 4.982 and 299 detected blocking, stuck, isolated and fast bags respectively for the entire system on the given day. We observe that the stuck bags are the most common outlier types for the given day.

Stakeholders find the outlier classification useful for their analysis. Even though they are familiar with the type of outlier problems happening in the baggage handling system, this approach actually labels the bags based on the problems. According to stakeholders, this classification allows them to filter out the problems they want to focus on. Also, they are interested in the aggregated information regarding these problems at the segment level so that they can understand the sources of the problems.

Blocking Bag	Stuck Bags	Isolated Bags	Fast Bags
18.05%	40.67%	38.95%	2.33%

Table 8.3: Outlier type percentages in the entire system

Blocking Bag	Stuck Bags	Isolated Bags	Fast Bags
2.309	5.202	4.982	299

Table 8.4: Outlier type counts in the entire system

8.3.2 Identifying the Segments that Experience More Performance-related Problems

1. **Which parts of the system suffer from the most significant abnormal behavior on the given day?**

   We have found and visualized the problematic segments for the given day. To find the problematic days, in PSM we first select the cluster ranking to show the segments that are having the worst behavior. If we revisit the Figure 6.12, the worst behavior corresponds to the ranking between 0.67 and 1. After selecting the segments in their worst behavior, we sort the table in Figure 8.1 according to the variable **importance**. This variable is created by multiplying the total number of outliers in the segments to the average outlier score in the segment for the given day. The table that is sorted based on the importance variable shows the segments that suffered from significant abnormal behavior. According to Figure 8.1, Segment A seems to be the segment that has suffered the most from abnormal behavior.

   Stakeholders find the dashboard in Figure 8.1 useful to do pattern analysis in their system. They especially want to visualize the disaster days to detect the problematic segments that many of the bags suffered from the slow movement. However, they found some segments very long in the physical plan. For example, according to them, one segment that we identified is a 400-meter long conveyor belt. It is not desirable for stakeholders to focus on a large area to fix problems because they do not know where exactly the problem comes from. This segment problem is caused because of how the data is recorded and thus, unfortunately we could not offer a solution to fix it within the scope of this project.

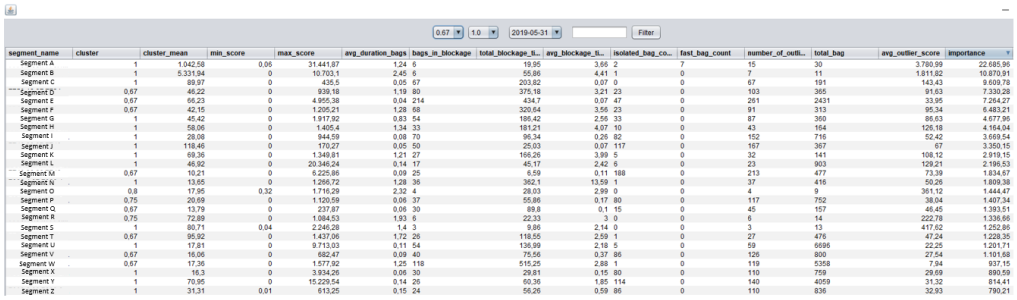


Figure 8.1: Problematic Segments

2. **For a given day, which segments suffer from serious blockage problems?**

   According to some stakeholders, the problematic segments is the one that has suffered the most from blockages. The dashboard in Figure 8.2 displays the segments that have the most serious blockage problems. The table is sorted based on the total blockage duration. Hence we see the segments that have very long blockages first.

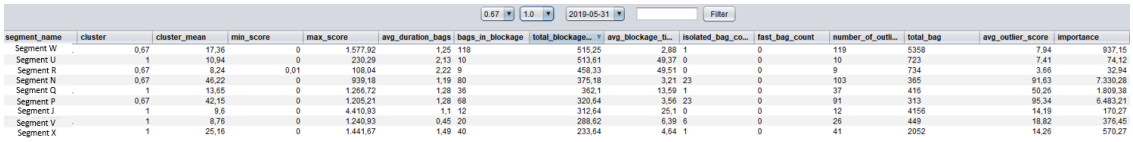
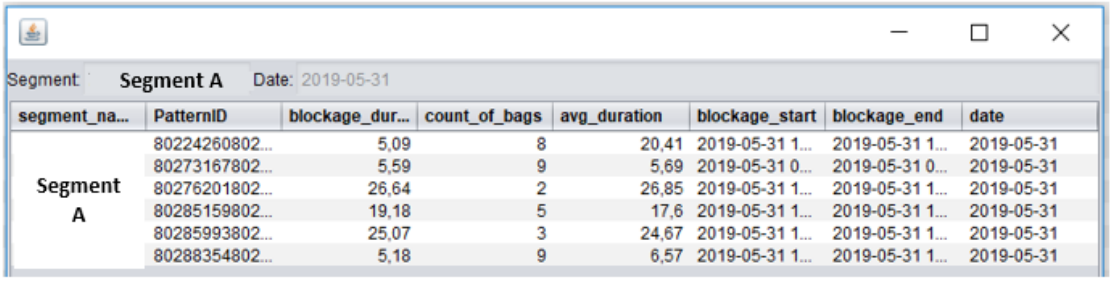


Figure 8.2: Segments that have serious blockages

3. **For a given day, what is the total amount of time bags spent in blockages and which attributes does each blockage have?**

   To investigate the blockages further, a blockage dashboard is created as explained in Section 7.2.2. The dashboard in Figure 8.3, shows all the blockages that happened in the given day and the selected segment along with blockage attributes.

   Stakeholders have found the blockage dashboard that is displayed in Figure 8.3 useful since it allows them to see each bundle of blockages. They plan to use this dashboard to find the exact time when a serious blockage(s) happened on a disastrous day.



Segment: **Segment A** Date: 2019-05-31

segment_na...	PatternID	blockage_dur...	count_of_bags	avg_duration	blockage_start	blockage_end	date
Segment A	80224260802...	5,09	8	20,41	2019-05-31 1...	2019-05-31 1...	2019-05-31
	80273167802...	5,59	9	5,69	2019-05-31 0...	2019-05-31 0...	2019-05-31
	80276201802...	26,64	2	26,85	2019-05-31 1...	2019-05-31 1...	2019-05-31
	80285159802...	19,18	5	17,6	2019-05-31 1...	2019-05-31 1...	2019-05-31
	80285993802...	25,07	3	24,67	2019-05-31 1...	2019-05-31 1...	2019-05-31
	80288354802...	5,18	9	6,57	2019-05-31 1...	2019-05-31 1...	2019-05-31

Figure 8.3: Blockages in segment A

4. **For a given day, which segments suffers more from isolated outliers ?**

   In similar ways as before, we find the segments that have the most isolated outliers by sorting the table based on the *isolated bag count* attribute. Figure 8.4 shows the segments that have the most isolated bag for the given day. Having many isolated bags means many single

incidents in the segment. Hence the stakeholders can investigate these segments further to understand why many single incidents occur.

According to the stakeholders, the isolated bag analysis for each segment is not enough by itself. They believe that an isolated bag outlier type occurs as a result of the bags but not because of the segments since they reflect single incidents in the system. Hence they are interested to see more statistical information to prove their point. For example, they want to know what is the percentage of the bags that are labeled as isolated bags in some other segments too. With this percentage, they can understand whether the problem comes from the segment or the bags. Unfortunately, we could not implement this statistical information due to the time restriction of the project.

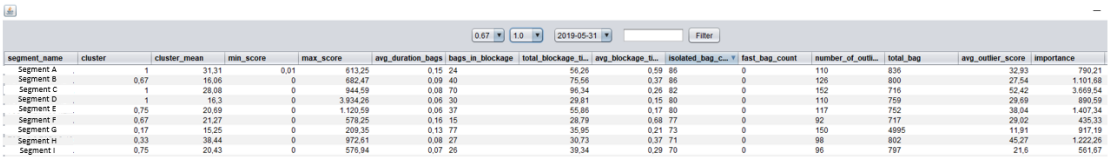


Figure 8.4: Segments that have affected by isolated outlier behavior

8.3.3 Categorizing the Abnormalities in Segments Based on Occurrence

1. **Which parts of the system regularly suffer from serious blockages?**

   To find recurring blockage behavior in the segments, we investigate the historical 6-months data. For each segment, we have found the *average blockage time* and we calculate *number of days that the blockage observed* attribute by counting all the days that the blockage was observed from the timestamp information. If we multiply these two attributes we obtain a new variable called **Blockage Importance** and it indicates how often the serious blockages have been observed in the system. We obtain Table 8.5 by using the results of the Algorithm 7 on 6-months of historical data. Table 8.5 is sorted descending on the blockage importance variable and it shows that Segment A is the segment that most regularly suffered from serious blockages during the 6 months.

Segment	Average Blockage Time	Number of Days Blockage Observed	Blockage Importance
Segment A	3.23	22	73.7
Segment B	3.62	19	68.76
Segment C	2.95	23	67.78
Segment D	2.80	24	67.24

Table 8.5: Repetitive blockages

2. **Which parts of the system shows non-repeating but serious blockages?**

   To find serious but not recurring blockages, we again use the historical 6-months data. We calculate *average blockage time* and the *number of days that the blockage observed* but this time we only include the segment if they have blockages at most 2 days in 6 months of period. We obtain Table 8.6 by using the results of the Algorithm 7 on 6-months of historical data. Table 8.6 display the results that are sorted based on the average blockage time the segments have. Accordingly, Segment E had the most important blockage on one particular day. Stakeholders can investigate the other segments in the table to compare the days the segments had these blockages on. If the day is the same for different segments then this could indicate a disastrous day that affected the entire system.

Segment	Average Blockage Time	Number of Days Blockage Observed
Segment E	5.17	1
Segment F	4.31	1
Segment G	3.83	1
Segment H	3.50	1

Table 8.6: Non repeating but serious blockages

8.3.4 Providing a Ground Truth About the Types of Outlier Behavior in the System

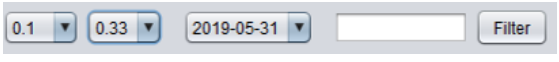
1. **What can be said about the daily behavior of a segment taking into account past behavior?**

   As explained in detail in Chapter 6, we have analyzed and ranked the behavior of each segment and day relative to all other days. We classified each day with respect to all other days by clustering, and each cluster is on a scale from 0 (best) to 1 (worst). Then this scale is implemented on the PSM as described in Chapter 7. According to the scale, we can see the segments that show their best and worst behavior for the given day. We also know how good or bad the new day performs in the sense of outliers as a result of the steps described in Section 6.2. Figure 8.5 displays the different filters according to segment ranking of the given day. We can classify the rankings as:

   - 0.1 - 0.33 : Best-behaving segments for the given day
   - 0.33 - 0.67 : Standard-behaving segments for the given day
   - 0.67 - 1 : Worst-behaving segments for the given day

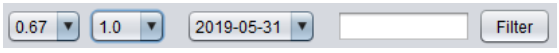
   According to the selected rank interval, we display the segments that fit the selected criteria for the given day. For example, if the rank interval 0.67 - 1 is selected, the average outlier score for segments is calculated as 49.50 while it is 6.209 and 2.792 for the ranges 0.33 - 0.67 and 0.1 - 0.33 respectively for the given day.

   According to the stakeholders, finding a ground truth about the segments adds a lot of value to their practice, because they believe in this way the best-performing days for the segments can be used as a baseline for future analysis such as predictive-maintenance as they do not have any other labeled dataset regarding the performance-problems in the system.



(a) Best-behaving segments

(b) Standard-behaving segments



(c) Worst-behaving segments

Figure 8.5: Daily behavior of segments

8.3.5 Visualizing the Problems Occurring in the System

1. **Can we visualize the outliers in the PSM?**

   The PSM view in Figure 8.6 shows the bags in different colors according to their types. (Refer to Figure 7.4 for the meaning of the colors.) Hence, we have provided a classifier that shows each outlier type in different colors. This way, the abnormalities in each segment are visible.

   Stakeholders find the visualization of outliers clear.

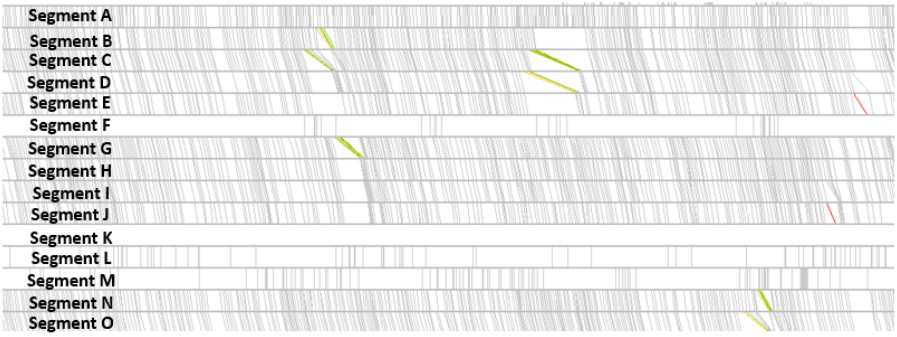


Figure 8.6: Outlier visualization in the PSM

2. **Can we automatically visualize blockages in the PSM?**

   The functionality we implemented in Chapter 7, allows us to visualize each blockage in blockage dashboard on the PSM. Figure 8.7, shows the visualization that is obtained by clicking a blockage that consists 14 bags. Hence, users can see the visualization of any blockage by clicking on them in blockage dashboard. This visualization is in the main view as well, this functionality just zooms in to the relevant segment and time interval.

   Stakeholders find this functionality useful to investigate the serious blockages and the relevant segments. In this way, they will see which other segments are affected by particular blockage.

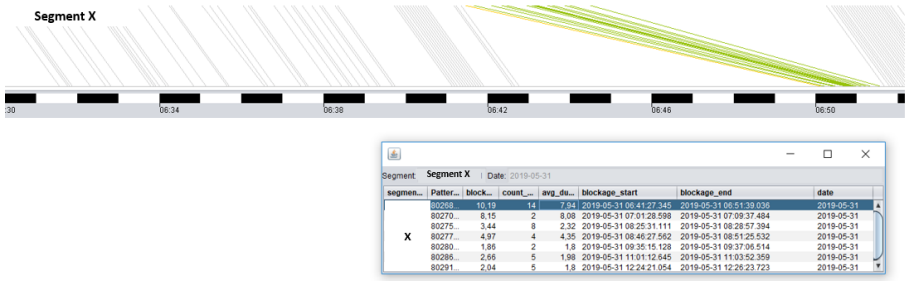


Figure 8.7: Blockage visualization in the PSM

8.3.6 Identifying the Properties that are Leading to the Abnormality

1. **Are there any cause-effect relationship between segments / sequence of segments in the sense of abnormal behavior?**

   We can find the dieback effect on the system by visually investigating the sequence of segments that physically follow each other. Diebacks are found manually by visually exploring the sorted PSM. In Figure 8.8, we see the segment **B:C** experienced a blockage and after

this blockage, the segment **A:B** also started to experience a blockage. The segment B:C comes after the segment A:B on the system. Hence, we see from the figure that the blockage in B:C had caused a dieback effect on the previous segment A:B.

The stakeholders find this analysis useful since they can detect the exact source of the problems. However, they believe rather than trying to find dieback effect manually, they would prefer another dashboard that automatically lists dieback effects and the segments that are involved. This will be a future work.

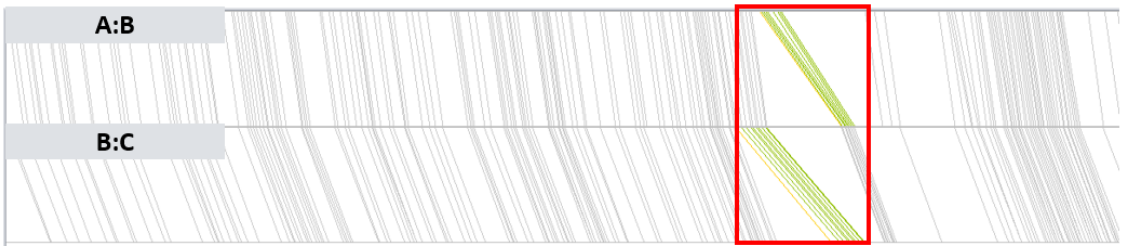


Figure 8.8: Dieback-Effect

2. **Can we find temporary malfunctions in the system that cause abnormal behavior?**

   To find temporary malfunctions, we investigate the PSM visually. This type of analysis is manual. We filter the segments based on one component it consists and look for the temporary malfunctions in the resulting view since the PSM gives the visual features. Figure 8.9, displays three different segments A:X, B:X and C:X. The common attribute for these segments is that they all have the same machine/conveyor belt X as the ending point. We filter the PSM to visualize the segments that have X as an ending point, The figure displays a malfunction in X that affected all the incoming paths to it. Accordingly, we can say that X had some temporary malfunction between the specific time range. Hence, by visually exploring the PSM, we can spot the temporary malfunctions in parts of the system.

   The stakeholders find this analysis useful if they are provided a list of detected faulty components in the system. With the current version, they believe temporary malfunctions are hard to detect manually since there are too many segments.

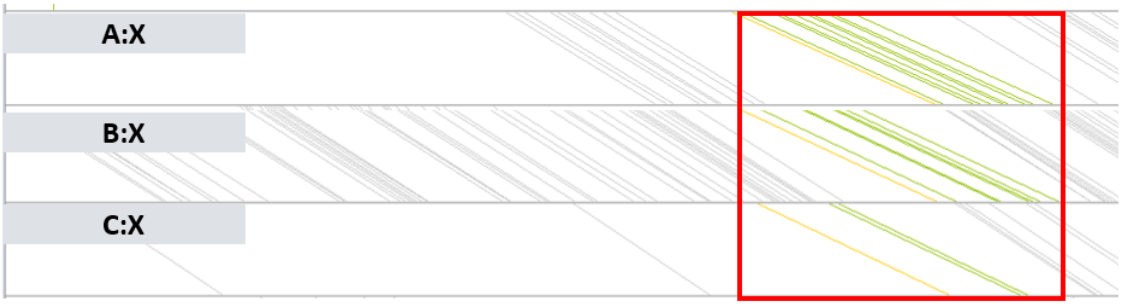


Figure 8.9: Temporary problems in the system

8.4 Practical Contributions to Stakeholders

This section explains how this project is adding value to stakeholder's day to day practice. Stakeholders found the project useful because of the following reasons:

- Distinguishing types of outliers: Before the tool, the users were able to see which bags were slow in which segment by investigating the logs of the baggage handling system in the PSM. Slow bags used to be defined based on their distance to the segment median. In a way, these bags correspond to the outlier bags. However, this thesis classifies the outlier types into blocking, stuck, isolated and fast outlier behaviors as displayed in Section 8.3.1. This classification is adding value as it allows them to distinguish the problems. They can now focus on any of the four problems to solve.

- Finding the cause of problems in the disaster days: Instead of trying to find which segments caused the disaster in PSM, they will easily and quickly find which areas are problematic by looking at the dashboards that are displayed in Section 8.3.2.

- Identifying the root cause of problems: As displayed in Section 8.3.6, they can distinguish the problems that are caused by dieback effect or temporary malfunctions. This again will allow them to find the area whose effect propagates backward to the system. Fixing the problems in the source of dieback effect leads to other problems in different areas to disappear too. Also, for the given day, they will be able to find the faulty components that have caused a malfunction in the system.

- Obtaining a baseline: They will use the best-behavior of the segments as a baseline for predictive analysis. They were lacking a labeled training dataset to base their predictive-analysis upon. We have classified each observed behavior in the segments and according to that classification, the cluster that represents the best-behavior has significantly fewer outliers and this cluster will be used as training data for their future analysis.

In this chapter, we evaluated our methodology and explained how do the project add value to the day-to-day practice of stakeholders. In the following chapter, we will conclude our project by stating contributions, limitations and future work.

Chapter 9

Conclusions

This chapter concludes the project. Firstly, the contributions of the project are stated in Section 9.1. The research questions are revisited in Section 9.2. Then in Section 9.3, we discuss limitations of the proposed method. Finally, possible improvements to the method are considered in Section 9.4.

9.1 Contributions

Within this master thesis, using a statistical approach, we identified the outlier bags with respect to their duration in each segment. We have classified the outlier behavior into four patterns: blocking, stuck, isolated, and fast bags. Then, using the outlier scores obtained from the outlier detection method, we have found different types of performance behavior for each segment in the system by grouping the days that have a similar outlier score distribution together. Then we ranked the different outlier behavior each of the groups represents to identify how bad the segment is behaving based on its previous behavior. To represent our findings, we have extended the PSM to display the points of interest starting with the most severe ones. The contributions of our method could be listed as follows:

1. Detecting outlier bags in specific locations of the system: We have found the outliers for each weekday and segment pairs. This is a contribution because outliers are found on a more specific level than before. Performance classifiers in the PSM already allow seeing slow bags in the segments using statistics. For instance, the distance to the duration median is used to detect slow bags in each segment. We strengthen this approach by adopting a more robust outlier detection technique called modified Z-score. This technique also scores each bag regarding its outlier strength. Hence, we obtain quantitative information on, not just outliers, but for each bag. Outlier scores of the bags are useful to find the performance of the segments and this leads to finding problematic segments that have bags with higher outlier scores.

2. Classifying the outlier types: The most important contribution of the thesis is that we provided high-level algorithms to classify the outlier types in the system based on their position in a sequence of outliers within a time window. We assigned one type to each outlier point. These outlier types correspond to the reasons for outlier behavior. Hence by assigning each outlier bag to one outlier type, we also define the reason why the bag is an outlier. Our outlier pattern classification method could also be used in similar domains such as traffic. We provide a way to identify blockages hence, any domain that have blockage behavior related to cars, humans, or products in their system could use this approach to detect blockages.

3. Blockage properties: We provided high level algorithms to find properties regarding the

blockages in the system such as blockage duration and the number of bags involved in the blockage. In this way, we quantify the blockages to find the severity of each of them.

4. Classifying the days: We obtain clusters of days representing different performance behaviors for each segment given the historical data. Our method is flexible in creating clusters since it does not require a fixed number of clusters as a parameter. We rank each of these clusters to represent the best-behavior, standard-behavior, and the worst-behavior of each segment based on the average outlier score.

5. Classifying new day behavior: With our clustering method, when we receive a new day of data, each segment can be immediately classified into one of the clusters, so we can assess if it is a regular day or a very exceptional day (either good or bad) for the segment. Also, since we obtain all this information for each segment, we are not generalizing the outlier score per day, i.e. one day can have a good outlier score for one segment and a bad outlier score for another. The clustering is done per each segment to obtain more accurate results.

6. Identifying a baseline behavior for each segment: The representation of segment behavior is a contribution of the thesis in a way that it allows creating a baseline for future analysis. If the best-behavior of the segment is used, we can ensure that this behavior does not contain many outlier points and it is a representation of how the segment would work if it does not suffer from performance problems. Hence, we create a representation of the perfect segment behavior to be used as a baseline for future analysis such as predictive maintenance in the system.

7. Finding problematic segments of the system: We provide a high level of transparency to the user, because the dashboards we implemented in the PSM helps the user to identify which segments have more unexpected behavior on the given day, and the user can then visualize the data of that segment in the PSM. The contribution can be described as we provide an automatic way to find and display all the problematic segments for the given day.

9.2 Research Questions

Here we will revisit the research questions that were introduced in Section 1.3 to discuss whether we fulfilled them or not.

1. **Is it possible to have a general purpose technique that can automatically identify performance related deviations within the performance spectrum?**

   We answered this question in Chapters 5 and 7. In Chapter 5, we introduced outlier detection and classification methods. As a result of these methods, bags are labeled as outliers according to their duration in the segment & weekday they belong. Then, each bag has been assigned with a relevant outlier pattern. In Chapter 7, we implemented a classifier for the performance spectrum miner to show all the bags in different colors according to their outlier pattern in each segment. Hence, we automatically identify and show the outliers for a given event log.

2. **Can we find patterns of outlier behavior and which outlier patterns make sense for the system?**

   This question is answered in Chapter 5. We have found 4 different patterns of outliers behavior as blocking, stuck, isolated, and fast bag patterns. These patterns make sense for the system because these behaviors have been observed in the baggage handling system before and they are acknowledged problems. For example in the baggage handling system we observe many bags accumulate in one part and unable to move forward. We have classified this behavior as a blockage in our method and identified which bag is the first bag that started the blockage and which bags got stuck in this blockage. Also, isolated outlier behavior is

also a problem in the system since it indicates single accidents. Hence, we classified relevant outlier patterns for the system.

3. **Can all states of normal performance behavior for each segment and over multiple segments of the system be identified?**

   We have answered this question in Chapter 6. We have identified different types of performance behavior for each segment in the system by grouping the days that have a similar outlier score distribution together. Then we ranked the different outlier behavior each of the group represents. As a result, we have obtained the normal behavior which is captured as the standard-behaving cluster and the special behaviors such as best and worst behavior for the segment. However, we could not find the normal behavior for multiple segments since we have more than 2000 segments in the system. The problem arises because we do not know which segments to investigate together among 2000.

4. **Which context factors cause outlier behavior in the system and what is the effect of outliers on performance of the system?**

   We detected only two possible reasons for the outlier behavior in the system. The first one is die-back effect as explained in Chapter 3 and the second is a temporary malfunction in the system that has been caused by a faulty component. These two possible reasons have been visualized in Chapter 8.3.

Moreover, we positively evaluated the project with stakeholders from the business, which further strengthens the results of this project.

9.3 Limitations of the Proposed Method

We will discuss the main limitations of the thesis:

1. Lack of labeled data: We were not provided with labeled data that indicates outliers for the system. Hence, we faced some troubles in validating the performance of the different outlier detection methods. Visual inspection was the only possible way for us to compare the results that have been obtained from different outlier detection methods.

2. Deciding the outlier detection technique: Bag movements have been recorded in many different parts of the baggage handling system. This causes more than 2000 segments to occur in the event log. Since we have too many segments, finding common methods to solve performance-related problems becomes a challenging task. We have mentioned the need for visual inspection to decide the best technique for outlier detection. However, visual inspection is only possible for a limited number of segments. We could not investigate all the segments to decide whether the technique works for all of the segments so we only investigated most used segments from different parts of the system. However, this could result in misclassifying outliers in some segments because the segments have different characteristics. For example, bags can spend a lot of time in manual parts of the system where people are involved. But, in some segments, they hardly spend a second. This causes a variation in the bag behavior between segments. Therefore, finding a technique that can deal with all kinds of variation is a limitation for this project.

3. Threshold selection for the outlier detection algorithm: The threshold selection is explained in Section 5.2.2. We have given this selected threshold as a parameter to the outlier detection algorithm. However, it is still generic for the segments. One given threshold is used to identify the outliers in all the segments. This causes a problem since the bags behave differently in different segments. The outlier detection algorithm assigns a wide variety of scores to bags resulting in the outlier score distribution to differ from segment to segment. When we assign one generic threshold to separate outlier bags and the normal-behaving bags in the system,

for some segments that threshold results in very clear separation whereas for other segments it could result in wrongly classified outlier points. According to the selected threshold, we can have many false-positive or false-negative errors. For our analysis, we assume that the threshold 50 is a good separator for the majority of the segments based on the visual inspections with different threshold values. However, since it is an assumption, the threshold selection remains a limitation for the project.

4. Normality assumption for the outlier detection algorithm: In Section 5.2.1 we have assumed that for the mean of the duration to be representative of the actual majority class, we need at least 30 bags in one segment per day. With this assumption, we have excluded all the segments that have less than 30 bags per day. This assumption becomes a limitation for the project as we cannot investigate all segments. However, this is also done to the limited source data.

5. Time-window selection for the blockage detection algorithm: The blockage detection algorithm is explained in Section 5.3.1. We define a time window to detect the bags that belong to the same blockage. According to the visual inspections with different time windows, we decided to use a 3-minute window for the project. However, we can still miss some bags that actually belong to a blockage. For example, we can miss labeling a serious outlier bag into a blockage if it is 3.1 minutes away from the last point that got stuck in the blockage. Different time-window selections do not change how the blockages are detected in a significant way since we check this time-window between consecutive bags and not the whole blockage. But, since it is still an assumption it also becomes a limitation for the project, but can easily be tweaked if necessary.

6. Segment definition: We have detected in the data that some segments have only a few bags. Most of the time, this indicates a data recording error, because the movement of the bag is not recorded as it is in the physical schema. This problem could be illustrated with an example. In Figure 9.1 the segment **A:E** is observed in the data but only one bag passed through that path. If we look at the image, we see the normal behavior for bags would be first A:B then B:C then C:D finally D:E. This raises the problem of labeling the bag in the wrong way. If the bag is showing abnormal behavior in any segment on its way, the algorithm is not labeling it as an outlier since it is the only bag in that segment. We could approach this problem as summing average duration times for the intermediate segments and compare with the actual duration of the only bag in that segment. However, we do not know for sure that the bag is supposed to follow the exact same route since there are many other alternatives too. Hence this problem stays as a limitation for the project. Though this is caused by data recording quality, which was deemed out of scope for our project.

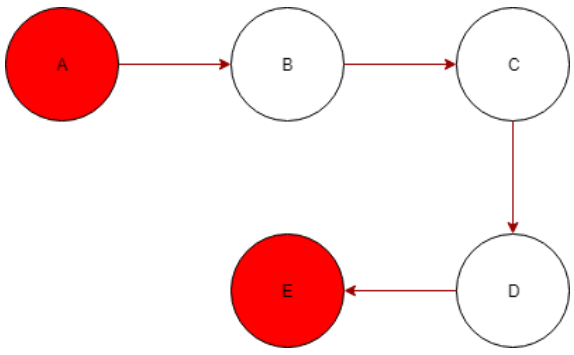


Figure 9.1: Segment definition problem

7. Updating the clustering approach: We have used the historical 6-months of data to create clusters of days to reflect different outlier behaviors for the segments. However, the result

of the clustering reflects a snapshot of the current data. When a new day arrives and if it sits between two existing clusters, then the cluster assignment would be random. Detecting this wrong assignment is a challenge. However, for now it remains as a limitation for the project.

9.4 Future Work

There are possible improvements to our current approach. We could not perform these improvements due to the time restriction.

One of the possible improvements is using a better outlier detection technique such as clustering. Because clustering does not require labeled data and it is very powerful in classifying outliers. We could not use the clustering technique because initially, the aim was implementing the entire outlier detection technique as an extension of the PSM. Therefore, we looked for simpler approaches to implement.

If the current methodology is implemented then the threshold and time frame selections could be improved. We left the threshold selection as a parameter for all segments. However, both threshold and time window could be automatically detected for each segment based on the behavior that the majority of the bags show in the segment. This could lead to a more robust outlier detection algorithm that applies to each segment.

The entire approach starting from the outlier detection to segment classification could be implemented in the PSM. This would make everything more automatic as currently we have to execute 2 different pipelines to obtain CSV files and only after that those files are used in the PSM. However, if every step of the method is implemented in the PSM the execution would be simpler and more efficient.

As a solution to the problem of updating clusters as explained in Section 9.3, every time new data arrives the clustering method could be updated. The other solution could be that we can assign the cluster as labels to the data and when analyzing the new data, the cluster labels could be used to predict the cluster of the new data.

Another improvement could be an automatic prediction of outliers and outlier patterns in the system. Depending on insights into outlier behavior and outlier patterns we found, one can build real-time predictive/prescriptive models so steps can be taken to prevent incidents.

The last improvement could be providing a better interface to show problematic segments, blockages, and their reasons. The current version only has basic functionalities to read and display the given CSV file. For example, in the current version of the project, we can only see the die-back effect visually. However, the method could be improved to automatically discover the die-back effect on multiple sequences of segments. This new functionality could be implemented in the PSM as follows; when an outlier behavior is detected on one segment, it could cumulatively check the next related segment for the possible die-back effect until it no longer detects the effect. In the end, the segments that have a die-back effect on each other for the given day could be displayed on a separate dashboard.

With our current solution and the listed future work we believe we have provided a solid first step in detecting outliers in material handling system and we look forward to see the future developments in this field.

Bibliography

[1] Frank Berger. Mining event log data to improve a loan application process. 06 2017. 15

[2] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distribution. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943. 13

[3] R. P. Jagadeesh Chandra Bose and Wil M. P. van der Aalst. Process mining applied to the bpi challenge 2012: Divide and conquer while discerning resources. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops*, pages 221–222, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 15

[4] L. Bouarfa and J. Dankelman. Workflow mining and outlier detection from clinical activity logs. *Journal of Biomedical Informatics*, 45(6):1185 – 1190, 2012. 15

[5] Charu C. Aggarwal. *Proximity-Based Outlier Detection*, pages 111–147. 12 2017. 11

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. 9, 11, 12, 15

[7] Arjel D. Bautista, Lalit Wangikar, and Syed M. Kumail Akbar. Process mining-driven optimization of a consumer loan approvals process. pages 219–220, 01 2013. 15

[8] Claudio De Stefano, Carlo Sansone, and Mario Vento. To reject or not to reject: That is the question - an answer in case of neural classifiers. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30:84 – 94, 03 2000. 12

[9] Vadim Denisov, Elena Belkina, Dirk Fahland, and Wil M. P. Aalst. The performance spectrum miner: Visual analytics for fine-grained performance analysis of processes. 09 2018. 8

[10] Vadim Denisov, Dirk Fahland, and Wil M.P. van der Aalst. Unbiased, fine-grained description of processes performance from event data. In *Business Process Management - 16th International Conference, BPM 2018, Proceedings*, Lecture Notes in Computer Science, pages 139–157, Germany, 2018. Springer. 2, 7

[11] R. Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, 15(3):458–486, 1970. 12

[12] F.Y Edgeworth. On discordant observations. *Philosoph*, 23(5):364375, 1887. 9

[13] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *KDD*, pages 226–231. AAAI Press, 1996. 11

[14] Brian. Everitt. *The Cambridge dictionary of statistics / B.S. Everitt.* Cambridge University Press Cambridge, U.K. ; New York, 2nd ed. edition, 2002. 13

[15] Asmaa Fawzy, Hoda M.O. Mokhtar, and Osman Hegazy. Outliers detection and classification in wireless sensor networks. *Egyptian Informatics Journal*, 14(2):157 – 164, 2013. 15

[16] J. Fox. *Applied Regression Analysis and Generalized Linear Models*. SAGE Publications, 2008. 11

[17] Lucantonio Ghionna, Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Outlier detection techniques for process mining applications. In *Foundations of Intelligent Systems*, pages 150–159, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 15

[18] Frank E. Grubbs. Sample criteria for testing outlying observations. *Ann. Math. Statist.*, 21(1):27–58, 03 1950. 8

[19] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969. 9

[20] James Hafner, Harpreet S. Sawhney, Will Equitz, Myron Flickner, and Wayne Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(7):729–736, July 1995. 12

[21] Jiawei Han, Micheline Kamber, and Jian Pei. 12 - outlier detection. In Jiawei Han, Micheline Kamber, and Jian Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 543 – 584. Morgan Kaufmann, Boston, third edition edition, 2012. 2

[22] Douglas M. Hawkins. *Identification of outliers / D.M. Hawkins*. Chapman and Hall London ; New York, 1980. 8

[23] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, October 2004. 9, 15

[24] Tianming Hu and Sam Y Sung. Detecting pattern-based outliers. *Pattern Recognition Letters*, 24(16):3059 – 3068, 2003. 15

[25] B. Iglewicz and D.C. Hoaglin. *How to Detect and Handle Outliers*. ASQC basic references in quality control. ASQC Quality Press, 1993. 10, 11

[26] Boris Iglewicz and Sharmila Banerjee. A simple univariate outlier identification procedure. 01 2001. 15

[27] E. Kreyszig. *Advanced Engineering Mathematics*. Wiley, 1999. 10

[28] Jorma Laurikkala, Martti Juhola, and Erna Kentala. Informal identification of outliers in medical data. *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 07 2000. 9

[29] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764 – 766, 2013. 15

[30] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26(4):354–359, 11 1983. 13

[31] S. S. Pawar. Survey on outlier pattern detection techniques for time-series data. 2014. 15

[32] Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. volume 6312, pages 749–762, 07 2010. 13

[33] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests. *arXiv e-prints*, Sep 2015. 12

[34] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000. 12

[35] J.R. Taylor. *Introduction To Error Analysis: The Study of Uncertainties in Physical Measurements*. ASMSU/Spartans.4.Spartans Textbook. University Science Books, 1997. 8

[36] Irene Teinemaa, Anna Leontjeva, and Karl-Oskar Masing. Bpic 2015: Diagnostics of building permit application process in dutch municipalities. 08 2015. 15

[37] W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer Berlin Heidelberg, 2016. 6

[38] L. N. Vaserstein. Markov Processes over Denumerable Products of Spaces, Describing Large Systems of Automata. *Probl. Peredachi Inf.*, 5:47–52, 1969. 12

[39] Singh Vijendra and Shivani Pathak. Robust outlier detection technique in data mining: A univariate approach. *CoRR*, abs/1406.5074, 2014. 15

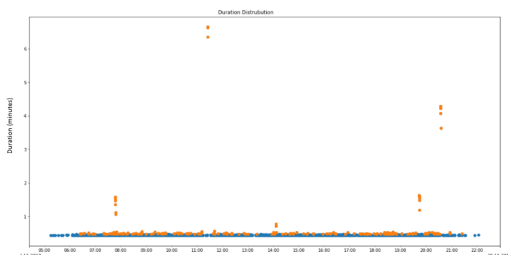
[40] S. Weisberg. *Applied Linear Regression*. Wiley Series in Probability and Statistics. Wiley, 2005. 11

[41] Wei Yang, Luhui Xu, Xiaopan Chen, Fengbin Zheng, and Yang Liu. Chi-squared distance metric learning for histogram data. *Mathematical Problems in Engineering*, 2015:1–12, 04 2015. 12

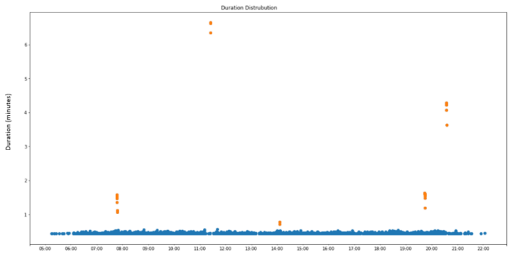
Appendix A

Threshold Selection for the Outlier Detection Algorithm

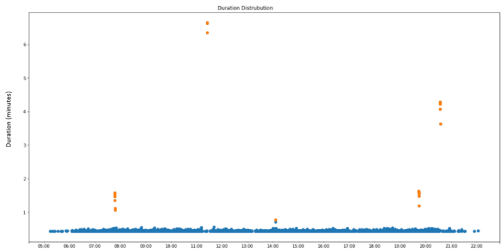
The comparison of different threshold for the outlier detection algorithm is visually displayed on different segments.



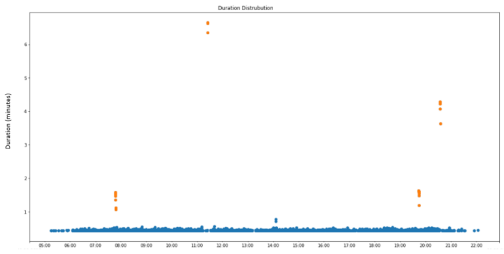
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20

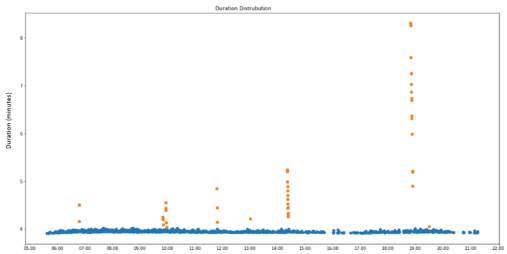


(c) Labeled outliers with threshold 50

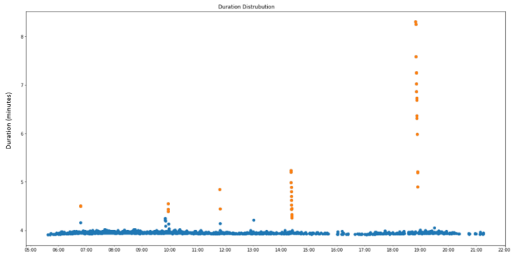


(d) Labeled outliers with threshold 100

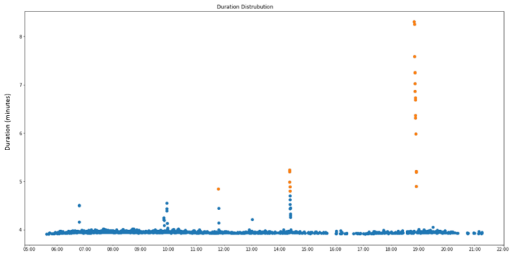
Figure A.1: Labeled outliers with different threshold for segment 1



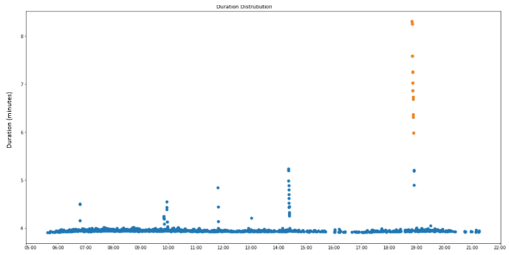
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20

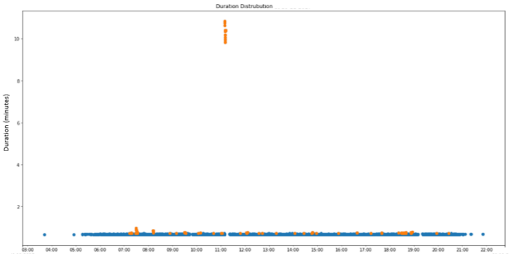


(c) Labeled outliers with threshold 50

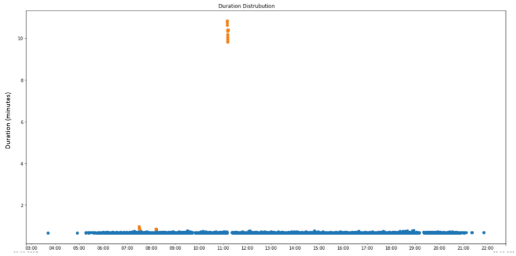


(d) Labeled outliers with threshold 100

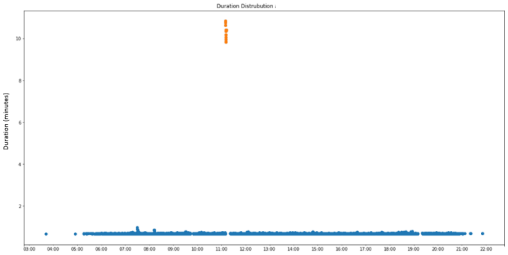
Figure A.2: Labeled outliers with different threshold for segment 2



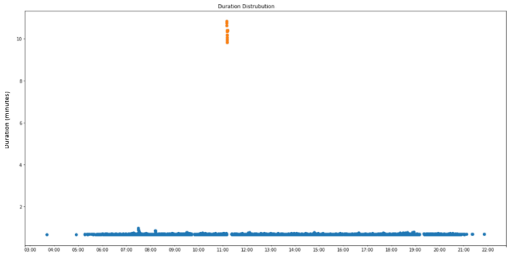
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20

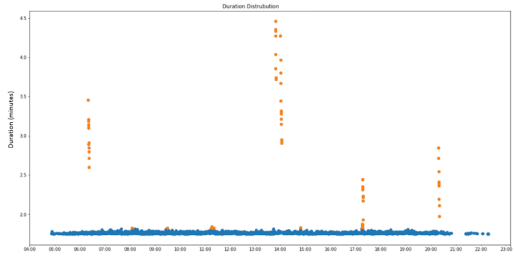


(c) Labeled outliers with threshold 50

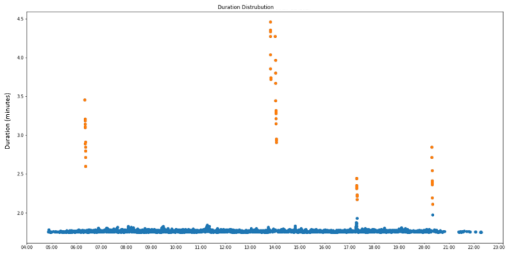


(d) Labeled outliers with threshold 100

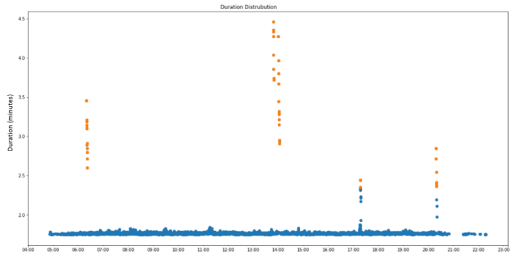
Figure A.3: Labeled outliers with different threshold for segment 3



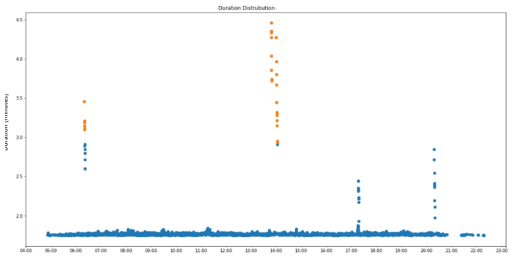
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20



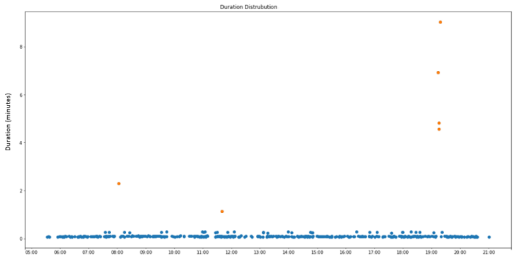
(c) Labeled outliers with threshold 50



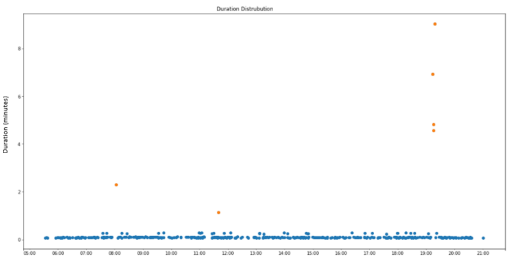
(d) Labeled outliers with threshold 100

Figure A.4: Labeled outliers with different threshold for segment 4

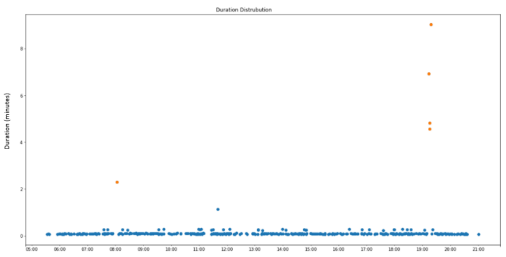
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20

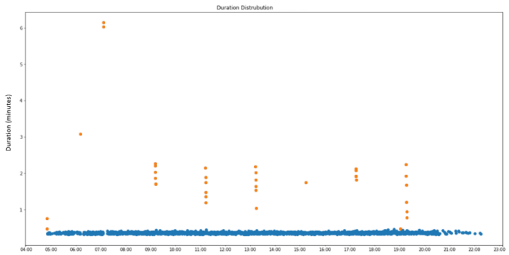


(c) Labeled outliers with threshold 50

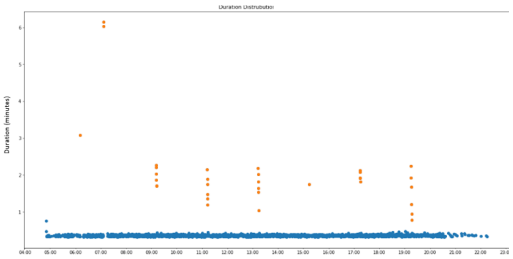


(d) Labeled outliers with threshold 100

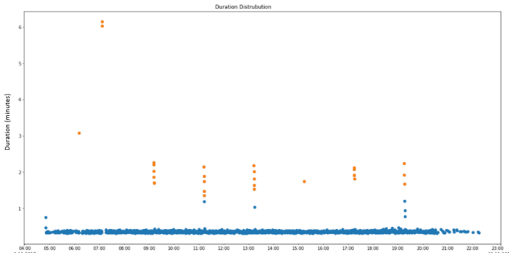
Figure A.5: Labeled outliers with different threshold for segment 5



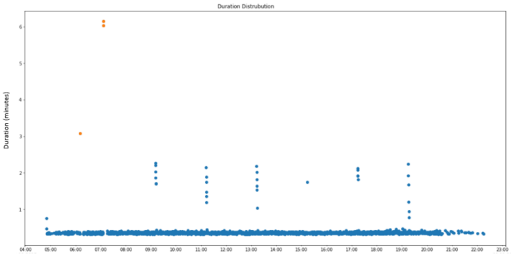
(a) Labeled outliers with threshold 5



(b) Labeled outliers with threshold 20



(c) Labeled outliers with threshold 50



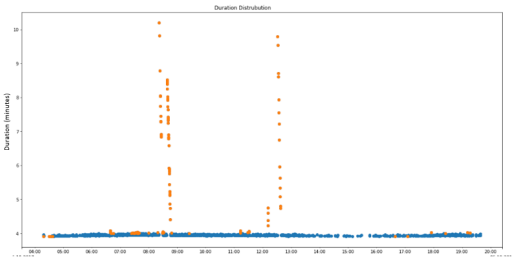
(d) Labeled outliers with threshold 100

Figure A.6: Labeled outliers with different threshold for segment 6

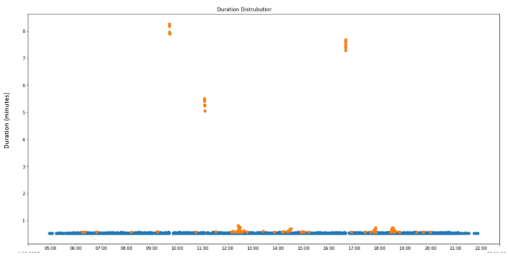
Appendix B

Results from Different Outlier Detection Techniques

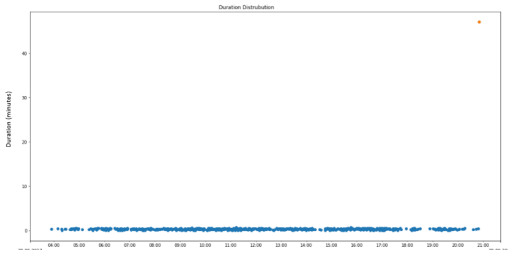
We have tried different outlier detection techniques in our project. Results are displayed for different segments.



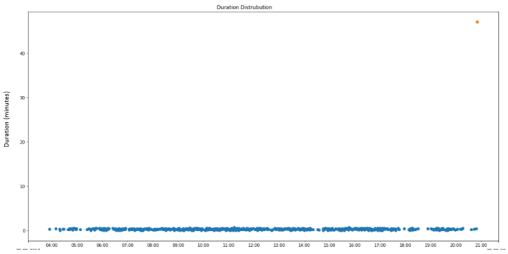
(a) Segment A



(b) Segment B



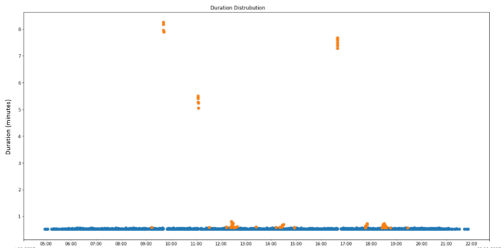
(c) Segment C



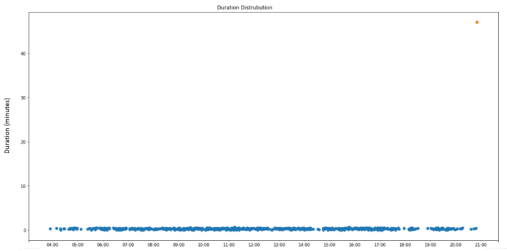
(d) Segment D

Figure B.1: Box-Plot rule (IQR) results

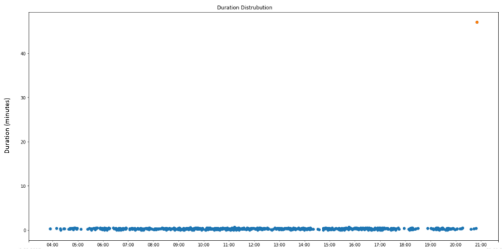
(a) Segment A



(b) Segment B

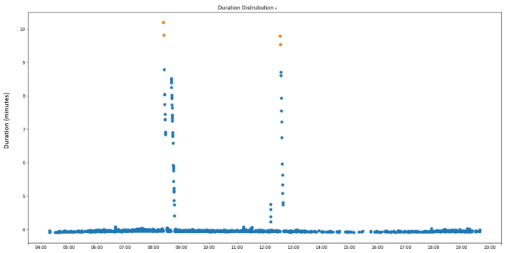


(c) Segment C

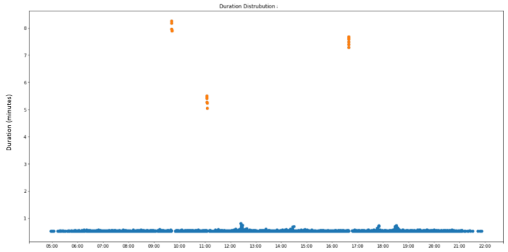


(d) Segment D

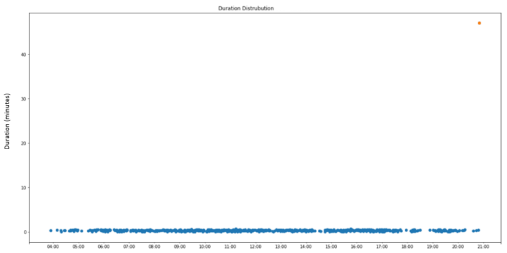
Figure B.2: Grubbs outlier test results



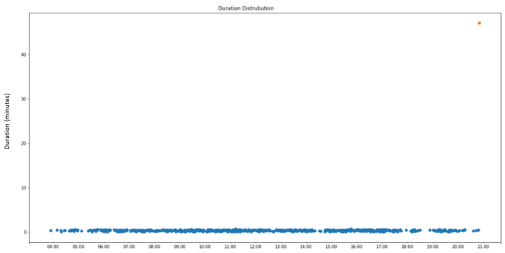
(a) Segment A



(b) Segment B



(c) Segment C



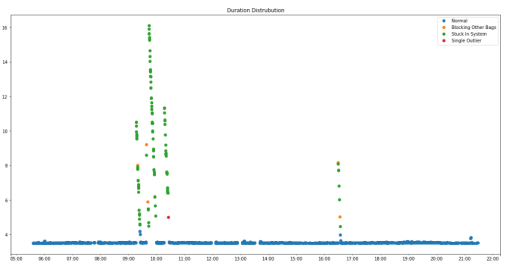
(d) Segment D

Figure B.3: Z-score outlier detection results

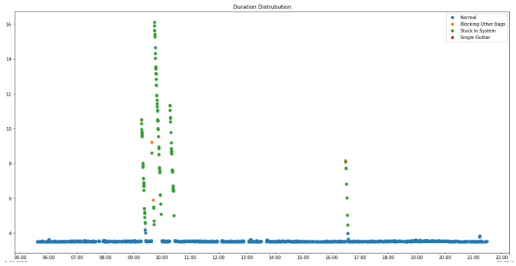
Appendix C

Different Time Window Selections for the Blockage Detection Algorithm

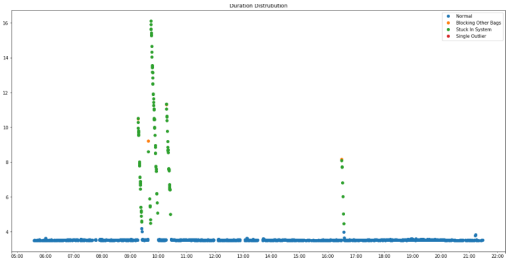
The blockage detection algorithm takes a time window and looks for outliers inside that time window. We have tried different time windows and the results are as follows:



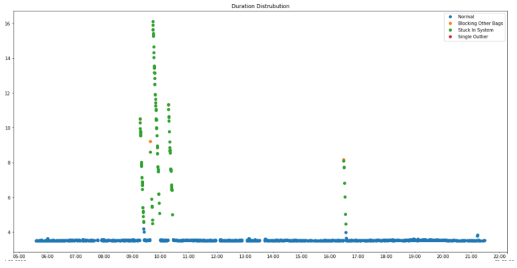
(a) Time window: 1



(b) Time window: 2



(c) Time window: 3



(d) Time window: 5

Figure C.1: Different Time window tries for a segment

Appendix D

More Results from the Outlier Classification Algorithm

We display more results as a result of the outlier detection & outlier classification algorithms.

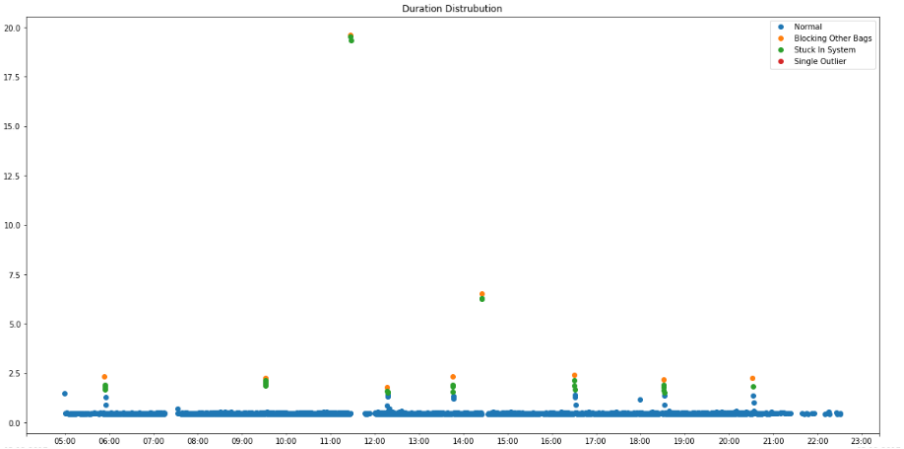


Figure D.1: Segment S

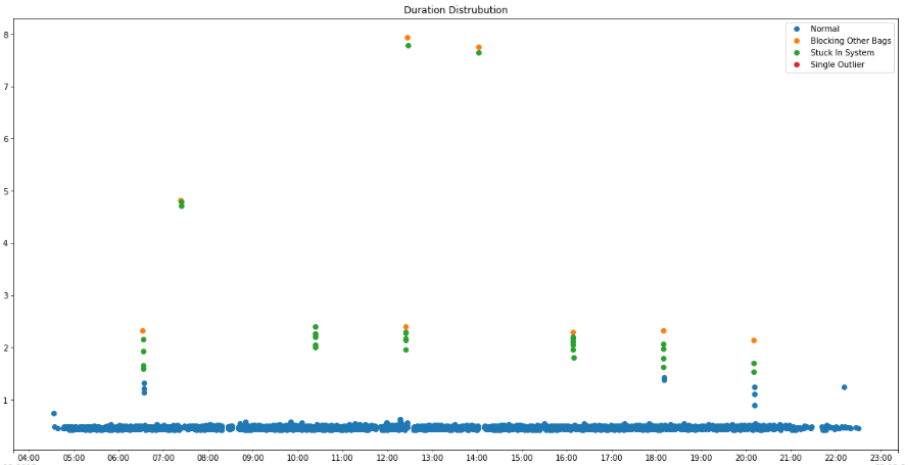


Figure D.2: Segment T

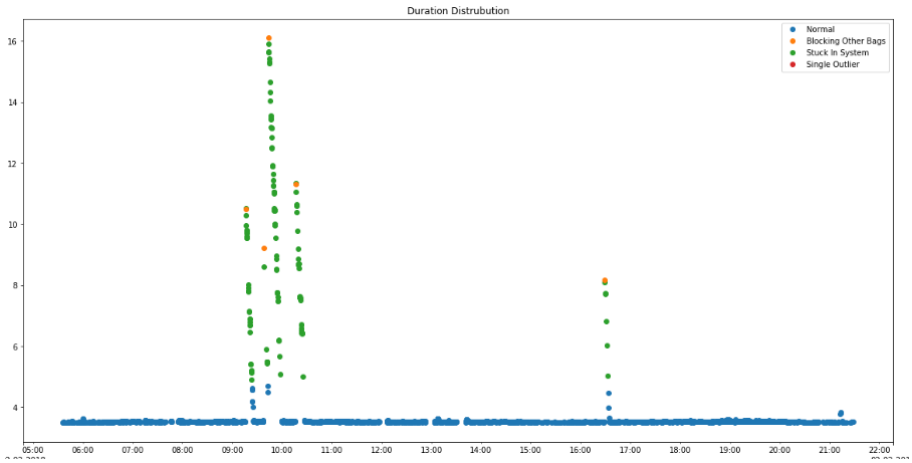


Figure D.3: Segment U

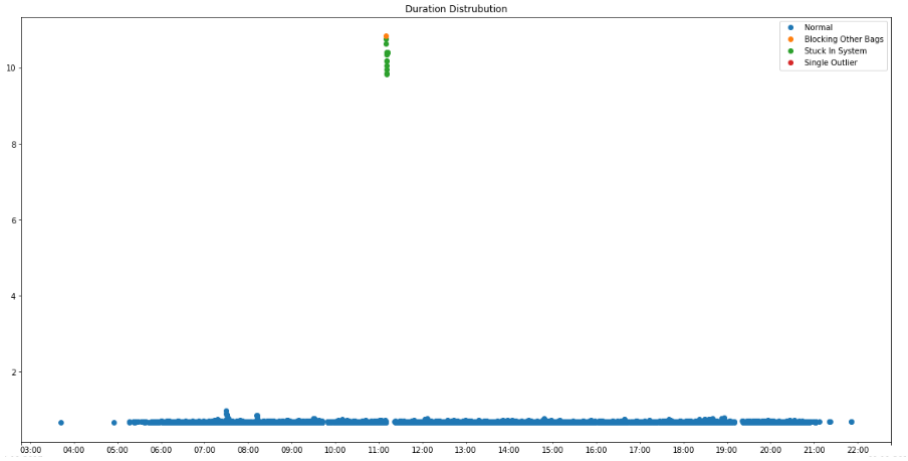


Figure D.4: Segment V

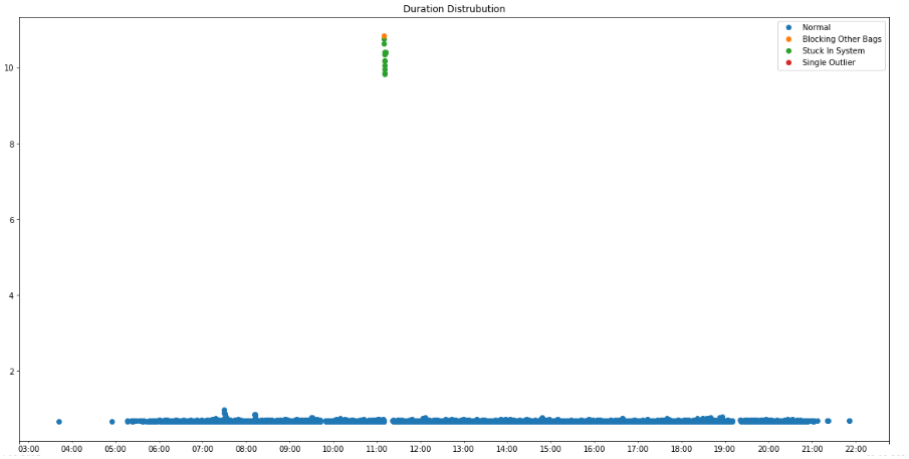


Figure D.5: Segment W

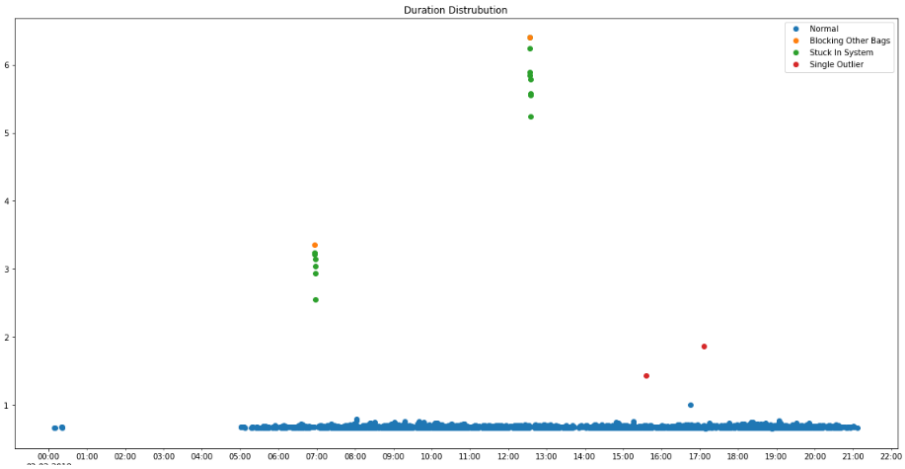


Figure D.6: Segment X

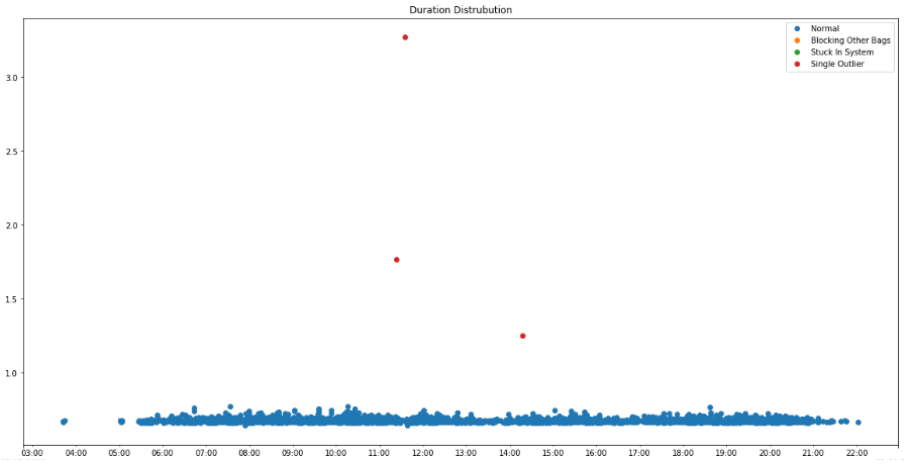


Figure D.7: Segment Y

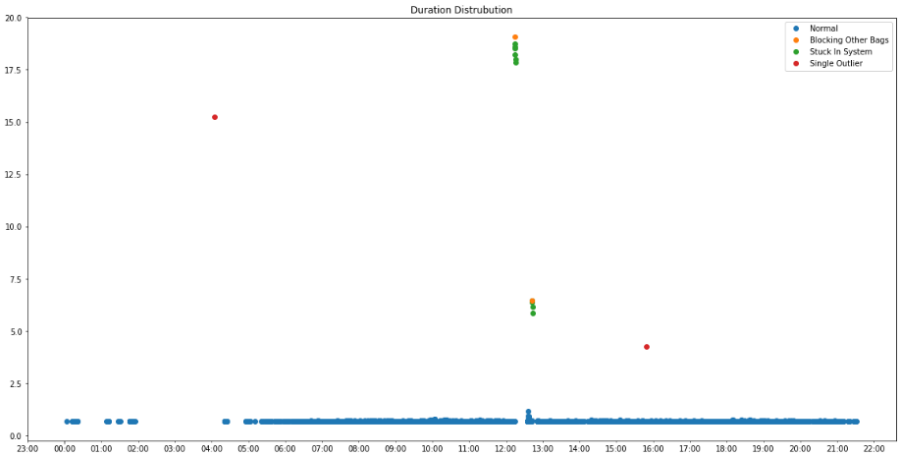


Figure D.8: Segment Z