INVENTORY STOCK ANOMALY DETECTION WITH VARIATIONAL AUTOENCODER

(DEĞİŞKEN OTOKODLAYICI KULLANARAK ENVANTER STOĞUNDAKİ ANOMALİLERİN TESPİTİ)

by

Halil ARĞUN, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

INDUSTRIAL ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Jan 2022

TABLE OF CONTENTS

LIST OF SYMBOLS

AD	: Anomaly Detection
AE	: Auto Encoder
VAE	: Variational Auto Encoder
DL	: Deep Learning
LOF	: Local Outlier Factor
IRI	: Inventory Record Inaccuracy
ANN	: Artificial Neuron Network
IQR	: Interquartile Range
MAE	: Mean Absolute Error
LSTM	: Long Short-Term Memory
CNN	: Convolutional Neural Network

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Retail companies monitor inventory stock levels regularly and manage stock levels based on forecasted sales to sustain their market position. The accuracy of inventory stocks is critical for retail companies to create a correct strategy. Many retail companies try to detect and prevent inventory record inaccuracy caused by employee or customer theft, damage or spoilage and wrong shipments. Our study aimed to detect inaccurate stocks using the Variational autoencoder (VAE) method, and we used the real inventory stock data of one of Turkey's largest supermarket chains. This method learns the distribution of data, and it is a great advantage to use this in data that changes over time. The VAE learns the usual pattern from normal time series data and detects anomalies by identifying the unseen data pattern, possibly reducing time and effort while gathering error data. In addition, this method can be applied to any product level. However, we use the interquartile range method to define the threshold for our model; therefore, it becomes parametric. On the other hand, generally, researchers use public data to develop methods, and it is challenging to apply machine learning algorithms to real-life data, especially in unsupervised learning. We show how to handle real-life data noises, missing values etc. The experimental findings show that the proposed approach can detect anomalies in the low and high inventory stock quantity and quickly apply to other time series anomaly detection problems.

ÖZET

Perakende şirketleri, envanter stok seviyelerini düzenli olarak izler ve pazar konumlarını korumak için satışlara dayalı tahminlere göre stok seviyelerini yönetir. Envanter stoklarının doğruluğu, perakende şirketlerinin doğru bir strateji oluşturması için kritik öneme sahiptir. Birçok perakende şirketi, çalışan veya müşteri hırsızlığı, hasar veya bozulma, yanlış sevkiyatlar nedeniyle envanter stoğundaki yanlışlıkları tespit etmeye ve önlemeye çalışmaktadır. Değişken oto kodlayıcı yöntemini kullanarak hatalı stokları tespit etmeyi amaçlayan çalışmamızda Türkiye'nin en büyük süpermarket zincirlerinden birinin gerçek envanter stoğu verileri kullanılmıştır. Değişken oto kodlayıcı yöntemi verilerin dağılımını öğrenir, bu yüzden zamanla değişen verilerde bunu kullanmak büyük bir avantajdır. Değişken oto kodlayıcı, normal zaman serisi verilerinden verilerin dağılımını öğrenir ve başka bir zaman dilimindeki anormallikleri tespit eder; bunları yaparken hem zamandan hem de emekten tasarruf sağlar. Bunun yanında yöntem herhangi bir ürün seviyesine uygulanabilir. Ayrıca, modelimizdeki anomali eşik değerlerini tanımlamak için çeyrekler arası aralık yöntemini kullanıyoruz; bu sayede eşik değerlerimiz parametrik hale geliyor. Öte yandan, genellikle araştırmacılar yöntem geliştirmek için halka açık verileri kullanır fakat özellikle denetimsiz öğrenme alanındaki makine öğrenmesi algoritmalarını gerçek hayattaki verilerine uygulamak zordur. Çalışmada gerçek hayattaki verilerdeki problemlerin örneğin verideki eksik ve ekstrem değerlerin vb. nasıl ele alınacağını gösteriyoruz. Deneysel sonuçlar, önerilen yaklaşımın düşük ve yüksek seviyedeki envanter stoğundaki anormallikleri tespit edebildiğini ve diğer zaman serisi anormallik tespit problemlerine hızla uygulanabileceğini gösteriyor.

1. INTRODUCTION

Many retail companies keep their inventory levels under constant control and manage the inventory level according to the estimated future sales to support their market position. Many companies develop a dynamic structure that automatically predicts demand instead of static methods or buying a service or product that automatically predicts the demand. In the retail industry, automatic demand forecasting plays a vital role in inventory management for companies. The correct operation of automated inventory management is based on the company information system that provides accurate stock information (Chehrazi, 2020). Generally, the company information system's stock data are based on daily sales and shipments calculations. However, the company information system's stock values may differ from the actual stock in the store and warehouse. If the stock in the system is lower than the existing stock, it may lead to excessive products, resulting in extremely high inventory levels (Chuang et al., 2016). On the contrary, the order of products may be delayed, and the company can not satisfy customer demand. In either case, the faulty stock automated system cannot operate to its full potential, resulting in a loss of profit. Inventory record inaccuracy (IRI), the name given to errors in stocks in the literature, can be caused by manual adjustments, theft, damage, and wrong shipments. IRI causes **1% sales and 3% gross profit loss** (Shabani et al., 2021) in the retail industry.

Considering that there are millions of stock-keeping units (SKU) in the retail industry, we can easily express that it is challenging to ensure stock accuracy and follow it. Many retail companies implement Enterprise Resource Planning (ERP) systems (Natsvlishvili et al., 2020) to ensure consistency of inventory stock. To prevent wrong inventory stocks, the human factor decreases day by day, but mistakes can be made in accepting goods or sending goods from one store to other stores and ERP can not handle these errors. These

directly affect the inventory accuracy in the system. Machine learning is one of the methods that can apply to detect these errors in the inventory stock.

1.1. Inventory Management

Stevenson (2021) defined *inventory* as a stock or store of goods. Companies generally stock hundreds or even thousands of items in their inventory, from little products like pencils, glasses, string, and buttons to major items like machinery, cranes, construction equipment, and trucks. The majority of things in a company's inventory are, of course, tied to the sort of business it does. As a result, manufacturing companies provide raw materials, buy components, semi-finished products, and completed goods. Fresh and canned foods, packaged and frozen meals, home supplies, periodicals, baked goods, dairy, fruit, and other items are all available in the inventory of supermarkets. We can categorize inventory into six groups: raw materials, semi-finished products, finished goods, equipment and supplies, maintenance and repairs, and goods and services in transit (Stevenson, 2021).

Inventory management is an essential aspect of operations management (Stevenson, 2021). Most organizations and supply chains rely on effective inventory management, influencing operations, marketing, and finances. On the other hand, poor inventory management stymies operations, lowers customer satisfaction, and raises operational expenses. Some businesses have outstanding inventory management, while others have adequate inventory management. Many, on the other hand, have poor inventory management. They have insufficient or excessive inventory, poor inventory tracking, or misplaced priorities. What is missing is a clear picture of what needs to be done and how it should be done.

Inventory control issues can result in both understocking and overstocking of products. Late deliveries, lost sales, customer complaints, and production bottlenecks arise from understocking; overstocking wastes space and money that could be better spent elsewhere. Although excessive overstocking may appear to be the lesser of two evils, the cost of excessive overstocking can be staggering when inventory holding costs are high,

and things can easily spiral out of control. The overall purpose of inventory management is to deliver exceptional customer service while keeping inventory costs within reasonable bounds. When it comes to inventory management, the two most essential considerations (decisions) are when to order and how much to order.

Stevenson (2021) indicate that there are five requirements for effective inventory management:

- A way to keep track of what's in stock and what's on order.
- A dependable demand projection that includes a cautionary note regarding possible forecast errors.
- Information on delivery timeframes and variability in delivery time.
- Inventory holding, ordering, and shortfall costs are all plausible estimates.
- A classification system for items in stock.

The first requirement is crucial since it assures that the others exist. The accurate stock can aid in creating accurate forecasts at the right time. We focus on how to increase the accuracy of inventory. The overview of the inventory management process is shown in *Figure 1.1*.

Figure 1.1 Inventory Management stages (Epstein, 2022)

1.2. Time Series

A time series is a series of data points collected in equal time intervals. A time series collects observations or data for a particular variable collected over a defined and ordered set of periods. Observations of a variable are recorded against equal time intervals in a time series model.

If we define time periods as $t_1, t_2, t_3, \dots, t_n$, and observations are $X_1, X_2, X_3, \dots, X_n$; The set of X is referred to as time series and time between observations equal each other. We can show time series as:

$$X_t = \mathrm{X}\{X_t \; where \; t = 1,2,3 \dots n\} \qquad (1)$$

Also, if we want to show the relationship between observation time and data points as a function, it is $\mathrm{X} = \mathrm{f(t)}$.

Time series analysis is more accurate when the time series is long. However, there are short, medium, and long-range forecasting approaches in time series data analysis. The analysis' accuracy is improved by using the appropriate time series approach for the data set. On the other hand, time series models are useful when investigating the previous behaviour of variables. We can use it to predict or predict the behaviour of the business or economic variables. Policy decisions and future planning can be formulated using time series models. Interdependencies between two or more time series can also be defined using them. Time series data analysis is essential to identify fundamental forces and structures in a time series variable and identify a suitable prediction model. Current and previous observations can forecast a time series' future value. A time series' neighbouring observations are interdependent. Consider the inflation rate of a country: the previous year's inflation rate impacts this year's inflation rate, and this year has an impact on the future year's inflation rate (Illukkumbura, 2021).

Generally, researchers use monthly, weekly, daily and hourly time series. For example, the daily USD price collection is a daily time series; if we look at EUR/USD weekly, it becomes a weekly time series. An example of price change in EUR/USD is shown in

Figure 1.2. Also, we can easily change time periods from small to bigger intervals by applying summation or average of values.

Figure 1.2 An example of weekly time series

1.3. Anomaly detection

Although machine learning usage areas are increasing (Sarker, 2021), one of these areas is finding different points from the usual situation. Detecting anomaly points in records problems in machine learning is processed as anomaly detection (AD). AD means that a situation or formation is different from its natural flow. In other words, it is a significant dissociation of a point itself from its past value or predicted future value. These classes are usually highly unequally distributed, with the normal class dominating the anomaly one. However, recognizing anomalies can provide helpful information about the application inside. Some examples are as follows (Aggarwal C. C., 2013):

- *Intrusion detection systems:* Computers gather a large amount of data (information) regarding system calls, logs, network traffic, and various other user actions. Because of malevolent activities or prohibited instances, data might sometimes indicate strange patterns. Intrusion detection is the process of detecting these actions.

- *Credit-card fraud:* Fraud detection is one of the most critical financial system cases. Credit card usage shows different patterns in many fraud cases, such as buying costly items from different locations. Anomaly detection techniques can detect such patterns.
- *Interesting sensor events:* Sensors are used in many real applications to monitor the environment for the detected incident.
- *Medical diagnosis:* Information about patients is gathered via a variety of tests and scans in numerous medical applications. Unusual patterns usually indicate diseases in data.
- *Time-series monitoring:* Variables change over time, but sometimes they increase or decrease suddenly. These events can provide important information.
- *Law enforcement:* Anomaly detection has a variety of uses in law enforcement, particularly when unexpected patterns may be identified over time with several actions by an asset. Trading operations frequently need to recognize unexpected patterns in data to detect fraud in financial transactions.
- *Earth science:* Many systems capture a large quantity of data regarding weather patterns and climate changes. This data reveals a unique pattern of human activity trends and environmental changes.

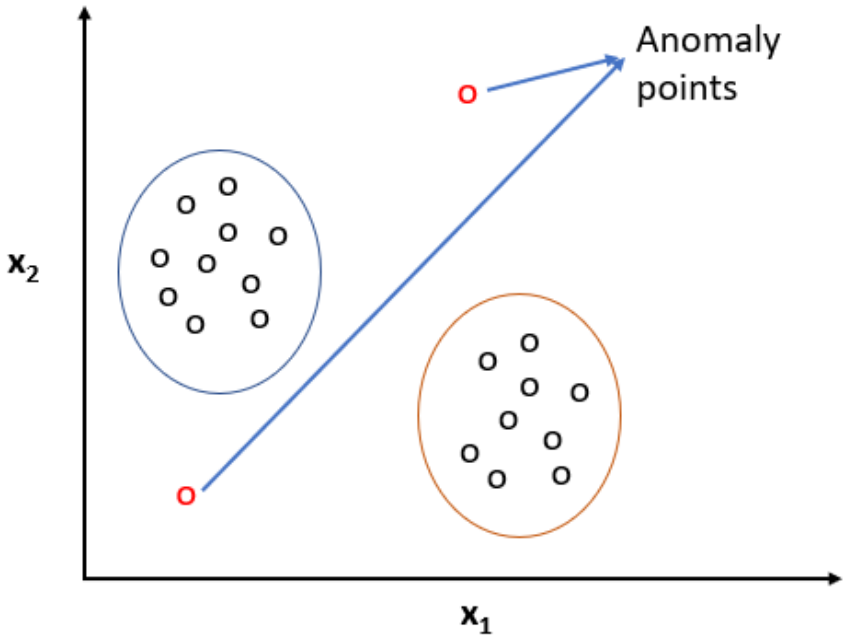

Figure 1.3 An example of anomaly points

In these applications, there is a typical situation of cases. It is referred to as the normal data model, and for unusual circumstances, there is abnormal data. Normal data is also known as inliers, and abnormal data is called an anomaly. An example of anomaly points are shown in *Figure 1.3*. There are many algorithms for detecting anomalies in the data (Chalapathy & Chawla, 2019). The output of these algorithms can be two types:

- *Anomaly score:* Most anomaly detection algorithms output a score for each data point. This score provides the level of the anomaly. The most suspicious points can be shown by ranking the data points. Also, the analyst can decide on a threshold to flag anomaly points.

- *Binary label:* Some algorithms give binary labels that indicate whether or not a data point is an outlier. Although some algorithms return binary labels directly, outlier scores can also be transformed to binary labels.

When comparing anomaly score and binary label, we can easily say that anomaly score has more advantages than binary because we can order score and define a threshold to convert from anomaly score to binary label. Especially ordering is critical for the business unit to analyze most suspicious events.

Anomaly detection methods have benefited from machine learning in recent years, especially from deep learning. In this context, studies on anomaly detection are generally carried out by applying supervised learning algorithms on labelled data. However, obtaining labelled data in many applications is very time-consuming and costly. In addition, each different situation for labelling may require a separate area of expertise. Therefore, supervised learning algorithms are limited by the availability of labelled data.

1.4. Why unsupervised time series anomaly detection?

Deep Learning's incredible success has been mainly due to supervised machine learning algorithms based on deep neural networks (Hinton et al., 2012) . To achieve good performance and state-of-the-art outcomes, large labelled datasets are required for training these algorithms. However, as previously said, large-scale labelled datasets are difficult to access by, and the annotation process necessitates domain expertise.

Despite the recent success of supervised learning, earlier unsupervised learning has produced outstanding results. For example, Hinton G. E.(2006) demonstrated outstanding results in dimensionality reduction using autoencoders in the early days of the deep learning era. Following that, the success of supervised learning in crucial tasks like speech recognition and image classification drew the research community's attention, which considerably enhanced and improved it, while unsupervised learning mainly was ignored.

Unsupervised learning is still a complex topic to tackle, as it consistently underperforms supervised learning in various tasks. In terms of deep learning, machine learning algorithms typically focus on supervised models, while there has been a recent movement toward using unsupervised approaches. Because labelled data are scarce, researchers are working to develop unsupervised learning models to apply them to problems and tasks that have previously been disregarded.

Finally, it is worth noting that the human reaction to unanticipated world discoveries, precisely the human form of realizing anomaly detection, is mainly unsupervised. Here is a simple example inspired by another unsupervised learning example given by Yann LeCun (LeCun, 2018). In the early days of life, a newborn is typically perplexed by the environment he senses. On the other hand, a newborn can create his sense of normalcy, that is, learn and become aware of it, after experiencing the new world for a while and gathering some observations of the surroundings without any monitoring. He is frequently disturbed when he sees an unfamiliar face since he does not recognize it.

The idea underlying this simple example underpins the principles of unsupervised anomaly detection, which will be the primary emphasis of this thesis. Because the dataset that inspired this thesis is fully unlabeled, this thesis aims to improve unsupervised anomaly detection.

1.5. Motivation & Purposes

In the previous sections, we talked about the importance of inventory management. Accurate inventory stocks are essential for the customer experience and the company expenses. In this framework, when we show the inventory stocks numerically, we can examine them as a daily time series. We can say that the errors in the inventory are an anomaly; to find these errors, we can use anomaly detection methods based on deep learning. Since the errors in the inventory stock are hidden in multidimensional data, one of the deep algorithm methods, Variational Auto Encoder (VAE), can catch these errors.

This study aims to detect the anomalies before they are reflected on the store by using the real inventory stock data of one of Turkey's largest supermarket chains with the VAE method. Inventory record inaccuracy is a big problem in the retail sector (Chehrazi, 2020), and detecting the anomaly before it occurs and generating an alarm in case of a mistake can prevent this error before it happens. The points we aim to contribute to the literature in this study are as follows:

- Detect anomalies in data whose characteristics change over time; because people's consumption habits can change or the company's strategy can change.
- Detect anomalies in univariate time series; to give an example, a company want to observe just one product.
- Detect anomalies in multivariate time series; if the company wants to analyze more than one product at the same time.
- Reduce the number of false alarms in the anomaly detection system; companies have limited sources to investigate the alarm.
- Establishing a dynamic anomaly detection structure,
    - Adding a new feature should be easy
    - Extracting the current features should be easy
    - Defining the thresholds should be dynamic

1.6. Thesis Outline

The rest of this thesis is organized as follows:

- Section 2 presents the background of inventory record inaccuracy(IRI) and some unsupervised anomaly detection methods

- Section 3 provides a theoretical background for the methods and models used, such as artificial neural networks and variational autoencoders. In particular, it gives thorough information on training and the many varieties of VAE.

- Section 4 explain our methodology in the five parts; they are data gathering, data explanations, data preprocessing, application of VAE and Interpretation of result

- Section 5 describes our experimental results for univariate and multivariate time series.

- Section 6 explains the findings in the thesis and makes recommendations for further research.

2. LITERATURE REVIEW

In this section, the relevant literature has been reviewed in two stages. Firstly, the studies carried out to make the stock correct based on inventory management have been examined. Then, unsupervised time series anomaly detection methods and the studies they used are explained.

2.1. Inventory Record Inaccuracy

Inventory management processes are one of the most significant key factors for the inventory carrying companies aiming minimum operational cost while providing high-quality service with maximum product availability. Therefore, the information gathered from automated inventory management information systems is crucial for a successful business. However, while the correct data supports the right decisions, incorrect ones can lead to revenue losses due to out-of-stock cases arising from Inventory Record Inaccuracy (IRI) (Kang & Gershwin, 2005). The presence, effects, causes, reduction and measurement methods of IRI consists of the five main classifications regarding studies of IRI.

The presence of IRI in companies, regardless of having enterprise resource planning (ERP) systems or not, is identified via experimental proofs from several papers which define the difference between Q_R (recorded inventory quantity) and Q_P (psychical stock quantity) (Shabani et al., 2021). For instance, Kang and Gershwin (2005) disprove the

popular opinion that retailers are good at knowing the number of products they actually have in their stores while working with distinguished sponsor companies of Auto-ID Center at MIT. One of these global retail companies' stores is better at perfect inventory accuracy rather than other stores; there is a maximum 80% match between Q_R and Q_P. In contrast, two-third of inventory records were inaccurate. De Horatius and Raman (2008) also demonstrate similar results with their study on systematic variation in IRI. By observing 37 stores belonging to one retailer, it is discovered that only 35% of the 370,000 inventory records are matched with the quantity at the store and do not show IRI problems. In contrast, 65% of leading retail chain records are inaccurate.

Customer service level can also be seriously affected by IRI, and even at a low level. (Cannella et al., 2015). Canella et al. (2015) also found that IRI can risk supply chain stability due to their modelling, covering a numerical simulation that supposes varied mistakes. Kull et al. (2013) also refer to damaging effects of IRI on operating performance by observing the daily variation of IRI that is indicated by daily collected data from discrete-event simulation experiments to a multi-spectral retailer. It is revealed that IRI declines service levels, so implementations based on ignorance of daily IRI variation need to be revised to multi day counting for assessing daily IRI and identifying its reason. Barratt et al. (2018) argue that MCDC (multi-channel distribution center) is highly sensitive about IRI. The inventory system's stability and convenience can be negatively affected by IRI if it is at substantial levels and has a changing level of consistency. Product availability can suffer from a low level of IRI, which can lead to a reputational loss for retailers.

Some of the causes of IRI are described by Chuang & Oliva (2015) as backroom and shelf shrinkage as well as part-time labour with a lack of feedback loop and unsuccessful in reducing IRI. Theft, damage and poor quality are also considered as factors besides transactions (De Horatius & Raman, 2008). Based on the dependency of transactions, IRI can have permanent or temporary reasons.

The frequency of audit, the quantity of annual selling, cost of the product, volume of monetary, variety of products, mode of distribution, physical distribution model, the density of inventory are also casual agents of IRI (De Horatius & Raman,2008). Ton and Raman (2010) state that increased product variety and inventory levels result in high deformity rates. Employee related issues like turnover, training, workload, the effort also consists of drivers.

In order to reduce IRI, studies provide different management options. Kök and Shang (2007) highlight replenishment policy development as the design of better cycle count. Kök and Shang (2014) refer to RFID (Radio Frequency Identification) technology as an opportunity to monitor continuously via RFID readers.

De Horatius et al. (2008) introduce the Bayesian Inventory Record, which is used to estimate the probability distribution of the inventory in the presence of stochastic IRI triggers. The distribution is utilized to create a strict replenishment rule in an operational scenario. One of the essential aspects of the computations is replenishment, which captures the fluxes of the goods in the same manner that NRI does in the proposed IRI measure.

2.2. Unsupervised Time Series Anomaly Detection

Since we consider inventory stock errors as anomalies in time series, we need to examine past studies from this aspect. Anomaly detection is assessed as supervised, semi-supervised, or unsupervised, supported by whether the labels are used within the training process (Zhang et al., 2021). We concentrate on unsupervised learning algorithms, which are mainly grouped into classical anomaly detection and deep anomaly detection methods. The overview of anomaly detection methods can be found in *Figure 2.1*.

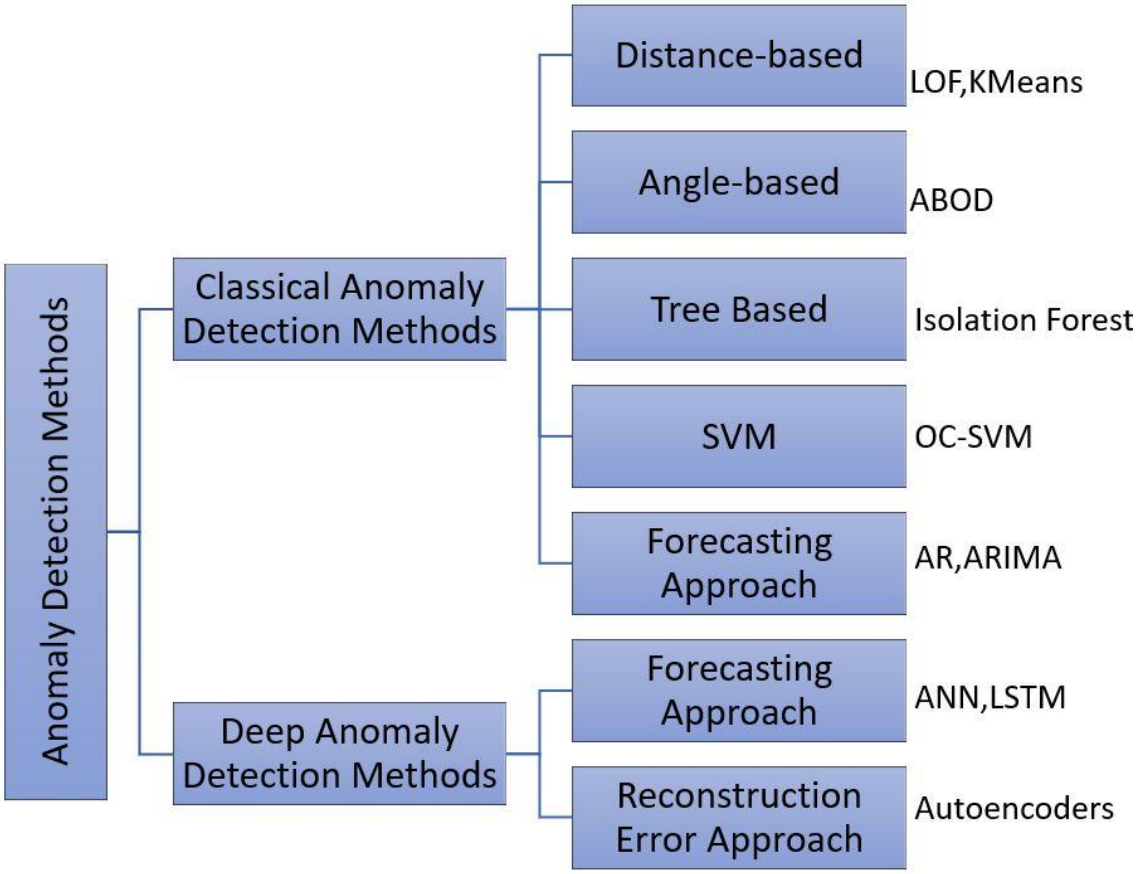

Figure 2.1 The overview of anomaly detection methods

2.2.1. Classical anomaly detection methods

Classical methods use linear calculation, and some of them are distance-based, density-based, angle-based methods. These methods are based on statistical and similarity calculation; therefore, the training phase takes too much time when data is increasing.

Density-based methods focus on data points in the specific region of space. Density-based methods are significantly related to clustering, and distance-based methods use the distances specific region, and we can say these regions are similar to clusters. Density-based methods detect locality-sensitive outliers.

The one popular density-based method is the Local Outlier Factor (LOF) (Breunig, 2000). LOF can measure data points' patency to adjust variations of different local densities. It considers that samples whose density is significantly lower than their neighbours are an outlier. LOF calculates the score for each data point by computing the average densities of the neighbours to the density of the point itself. Breunig (2000) uses the LOF methods to discover data outliers, data inaccuracies and traffic irregularities in real-world scenarios such as accidents, traffic jams, and low volume. An example of LOF outlier scores is shown *Figure 2.2*.

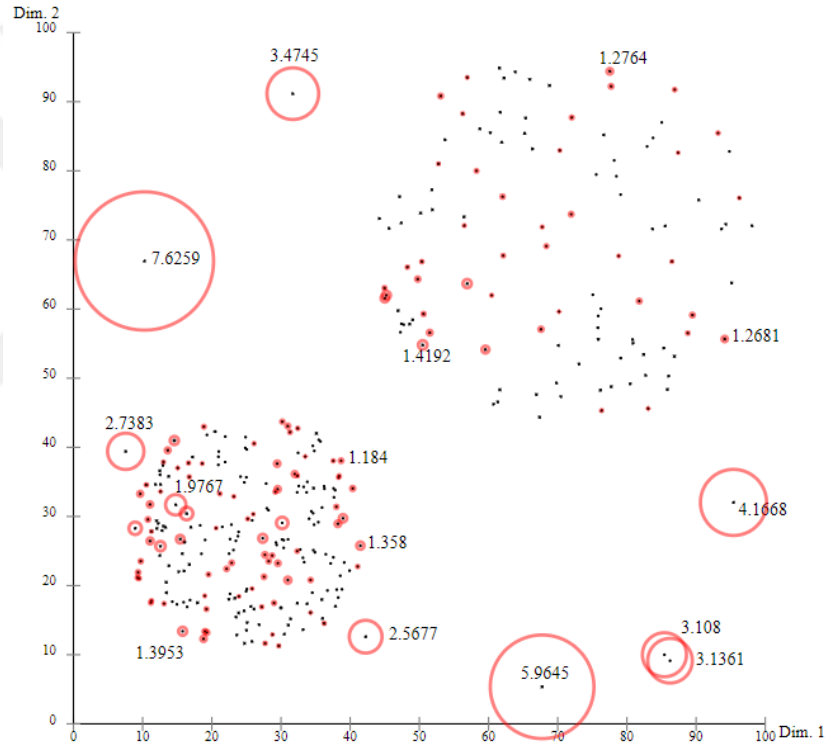

Figure 2.2 LOF outlier scores example, (Chire, 2016)

Another density-based method is clustering. There is a very strong relationship between clustering and anomaly detection. There may be some outlier values as well as points that are similar to each other. Clustering aims to divide a large data collection or sample into subgroups based on point similarity and evaluate them individually. Also, anomaly detection aims to find not similar points with other points. Generally, many algorithms defined *outlier values* as a side-product in their outputs. However, it is essential to

understand why these points are the outlier and how they are similar to their clusters. To find the reason for the anomaly, we have to find outlier scores that give the points different from other points (Aggarwal & Reddy, 2014) . The simple definition of a for is to find the distance between points and cluster centroids. Generally, there is more than one cluster, and we have to find the distance between points and their closest cluster. To understand cluster-based techniques, we can focus on the basic clustering algorithm that is K-means. K-means separate the data to a fixed number(k) of clusters. Each cluster is a group of data points grouped by their similarity and have a centroid, and all points in the clusters are close to their cluster centroids. Each point is assigned to the closest cluster centroid, and the process aims to keep the centroids small. Jianliang et al. (2009) use K-means to detect intrusion Detection, and simulations show that this approach effectively detects unknown intrusions in real-world network connections.

Another popular method is isolation forest is similar to decision tree algorithms. The algorithm aims to isolate the anomaly values by randomly selecting an attribute from the dataset and then randomly splitting the value between the minimum and maximum values. The randomly partitioning of the attribute proves a shorter path for the anomaly points that will be separated from normal data. Since they scale well with huge datasets and offer rapid prediction speeds. Furthermore, they operate well with various features, such as discrete and continuous, so the features do not need to be normalized. The disadvantage of tree-based methods is that they are susceptible to variables in data with a small number of samples. For example, they can mark all special days as anomalies. There are many applications of the isolation forest, such as network traffic anomaly (Tao et al. 2018), credit card fraud detection (Ounacer et al. 2018) and monitoring machines (Li et al. 2021).

Another popular classic method is a one-class support vector machine (OC-SVM) that differentiate one class of observations from another by using hyper-planes in multidimensional space (Zhang et al., 2007). One-class support machines are linear and logistic regression variants with margins used to avoid overfitting, like regularization in regression models. In addition, the sum of squared errors can be employed as a loss function in SVMs. An SVM model attempts to segregate data by as large a separation as

feasible while penalizing incorrect predictions on the wrong side of the gap. The SVM model then forecasts by dividing points to one side of the gap.

Another classical approach uses time series forecasting methods such as AutoRegressive (AR) and AutoRegressive Integrated Moving Average (ARIMA) to predict and calculate the difference between actual and predicted values. If the difference is high, the point has the potential for anomalies (Pena et al., 2013).

2.2.2. Deep anomaly detection methods

We can divide deep learning anomaly detection models into two groups: forecasting and using the reconstruction error of input values (Zhang et al., 2021). In forecasting models, the aim is to train the model with past values and make predictions for the future using this model. The difference between the estimates and the actual values gives us information about the degree of abnormality of the points. Recurrent neural network (RNN) and long short-term memory (LSTM) are very popular for sequence prediction. In (Filonov et al. 2017), authors use RNN based model to detect anomalies in multivariate time series data to prevent cyber-attacks by minimizing the mean squared error between actual and predictions. In (Lindemann et al. 2020), they use LSTM to detect anomalies in manufacturing processes by using advantages of long-term effects of processes. Ergen and Kozat (2019) combine LSTM and OC-SVM to increase the learning performance of LSTM because the training of deep learning forecasting models takes time and memory. The reconstruction model focuses on several approaches for lower reconstruction error. For example, Autoencoders are frequently used for anomaly detection by learning to rebuild a given input. The model is only trained on normal data. When it cannot rebuild the input with the same correctness as ordinary data reconstruction, the input sequence is labelled anomalous data (Zhang et al., 2021).

LSTM autoencoder models can reveal anomalies using long-term effects (Malhotra et al., 2016). Variational Autoencoders are autoencoders representing the relationship between

two random variables, latent variable z and visible variable x. A prior for z is typically a multivariate unit $\mathcal{N}(0; 1)$. VAE learns the distribution of variables, and this feature provides a dynamic structure for different variables. In (An & Cho, 2015), the authors define the reconstruction probability for anomaly detection as the average probability of the original data provided by the distribution. Anomalies are data points with more significant reconstruction possibilities and vice versa.

3. THEORETICAL BACKGROUND

This section presents the theoretical background of artificial neuron networks, how we can build a network, and how the network is trained. After that, we explain autoencoders and the types of autoencoders.

3.1. Artificial Neuron Network

Computers can now easily do millions of operations per second due to advances in technology. People have achieved significant advances in their profession by utilizing the power of computers. Despite this progress, many commonplace functions for humans, such as reading text and recognizing images, are extremely challenging for computers. Artificial intelligence has evolved to implement intelligent behaviour in machines in this approach. Artificial neural networks (ANN) are a successful example of this type of development. Because of its intricate structure inspired by the human brain, ANN can conduct various functions such as translating, combining features. ANN is commonly described as a transfer function from input x to output y. Inputs can range from sensor data to cameras, but they are nearly always recorded as a single or vector of real numbers, $\mathrm{x} = (x_1, x_2, x_3, \dots, x_m), x_i \in R$. The output of the network is also a form of real numbers, $\mathrm{y} = (y_1, y_2, y_3, \dots, y_m), y_i \in R$; however, it can be comprehended as the value of another signal, the likelihood of a picture belonging to a specific class, or anything else that real numbers can show. In the following sections, we give the details for the basics of ANN network structure, how the neurons are activated and how they are working.

3.1.1. What are artificial neurons?

The human brain is contained in a network of nerve cells that receive, process, and send electrical or chemical information. Similarly, an artificial neural network (ANN) is a network of artificial neurons. Each artificial neuron processes a collection of input variables and outputs a result. First, all of the inputs are weighted, and then the total is summed with a bias. This weighted sum is then transferred to an activation function $\varphi(.)$ that gives the neuron's transfer function linear and nonlinear characteristics. We can show the calculation as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \quad (2)$$

In equation (2), x is the neuron's input, w is the weight of the network, b is the bias term, and y is the output. The relationship between terms is shown in *Figure 3.1*.

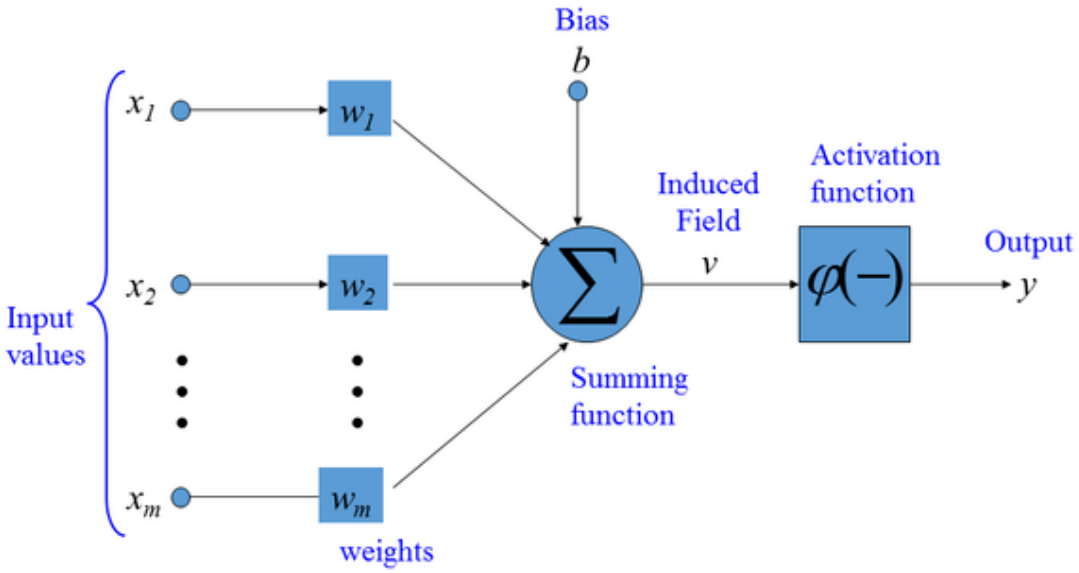

Figure 3.1 An artificial neuron *(Melli, 2021)*

3.1.2. How are the neurons activated?

Artificial neurons have activation functions that vary based on the network structure. Activation functions work according to the sum of the values coming from a neuron. According to the threshold of the function, it decides whether the neuron will be activated or not. For example, if we use a binary activation function and the result is less than 0, our neuron will not be activated because it will return 0. The activation function utilized can either increase or underperform the functionality of the artificial neuron. Their primary function is to transform the signal at the node into an output signal. Even if the neurons are activated, the outputs of the activation function are used as inputs for another neuron. Some popular activation functions are shown below (Wilson, 2012):

Table 3.1 Some activation function equations

Name	**Equation**
Identity	$f(x) = x$
Binary	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$
Logistic	$f(x) = \frac{1}{1 + e^{-x}}$
ReLU	$f(x) = \begin{cases} 0 & , \quad x < 0 \\ x & , \quad x \geq 0 \end{cases}$
SoftPlus	$f(x) = \log_e(1 + e^x)$

3.1.3. How to build a network structure?

A single artificial neuron cannot model highly complex processes. In fact, perceptron, one of the first artificial neurons to learn to separate data into one of two categories, has proven to perform only linearly separable tasks. Researchers began research connecting neurons in a network like the human brain to solve this problem. Various network architectures have been developed, and various structures have been shown to be effective in tackling different types of issues. These network architectures are feedforward neural

networks in which neurons are arranged in successive layers and information flows from one layer to the next.

Suppose we want to categorize the brand and color of a car in a photograph. There may be cars of brands A, B and C, each of which can be grey, blue or red. If we want to classify one step, we need to separate 3 × 3 = 9 different possibilities and need an equal number of decision units. However, by dividing the decision-making process into two stages, we can first determine the type and color. This is how we solve the decisions, and we will only need 3 + 3 = 6 decision units since they are not interconnected. Because the transfer functions we want to model are so variable and unknown, there is no way to fully prove how helpful depth is and when it is needed. However, empirical evidence suggests that greater depth yields better generalization results for a wide variety of tasks.

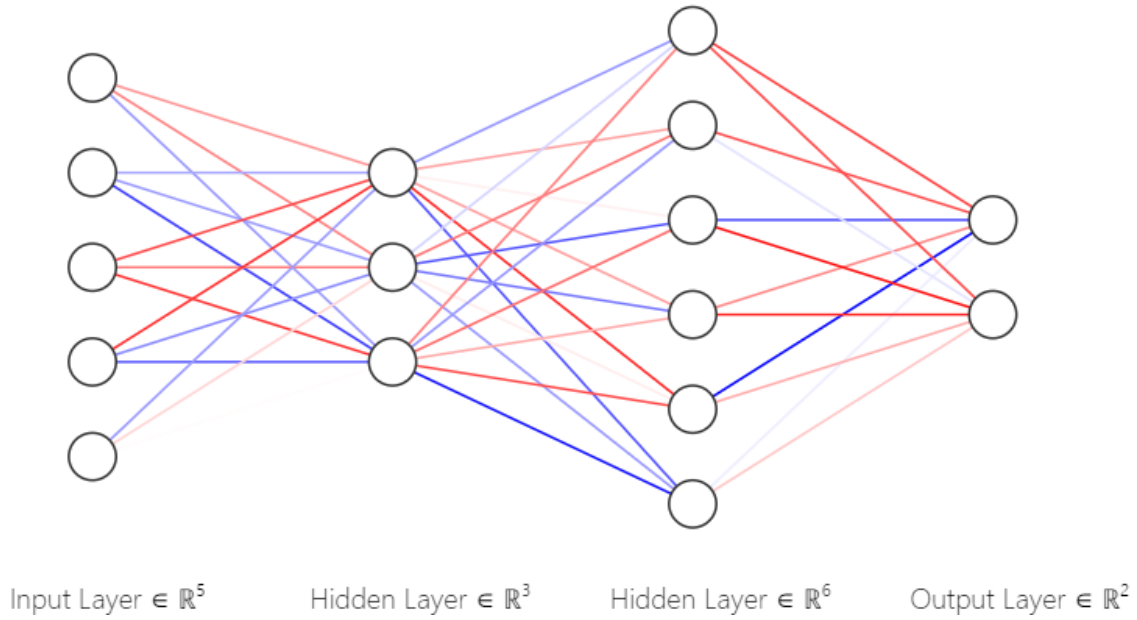

Figure 3.2 An example of ANN

In the example shown in *Figure 3.2*, the ANN has five neurons in the input layer, ANN includes two hidden layers, and one of them has three neurons the other one has six

neurons. Also, the output layer has two neurons. Depending on the complexity of the problem, we can increase layer and neuron numbers.

3.1.4. How to train neural networks?

Even though transfer functions we want to simulate with ANNs are generally uncertain, the network weights must be learned from data. This process is referred to as network training. Before beginning training, we first have to define the cost function because of comparing the input and outputs. Generally, mean squared error (3) and mean absolute error (4) are used as cost functions:

Mean squared error: $$\frac{1}{N}\sum_{i=1}^{n}(output - input)^2 \quad (3)$$

Mean absolute error: $$\frac{1}{N}\sum_{i=1}^{n}|output - input| \quad (4)$$

The issue of changing the weight parameters is now reduced to lowering the value of the cost function, and various optimization techniques can be applied. Today's most common algorithms use the information obtained from the derivatives of the cost function based on the weights. The most popular approach is the gradient descent-based technique that uses only the first derivative.

Gradient descent is the process of finding the lowest point on a surface by first taking a step in the direction of the steepest slope, then taking another step in that direction, assessing where the slope is now sharpest. The surface correlates to the cost function value, and the steps correspond to minor changes in the dependent variables, i.e. neural network weights. The slope is defined by the partial derivatives of the cost function, and it can be demonstrated that the sharpest slope corresponds to the cost function's gradient

vector. As a result, gradient descent necessitates taking repeated steps in the right direction of the cost function's negative gradient.

Another two important terms in the training batch size and epoch number. Batch size is a hyperparameter that specifies how many samples should be performed before updating the internal model parameters. A batch may be thought of as a for-loop that iterates across one or more samples and makes predictions. The estimations are compared to the predicted output variables at the conclusion of the batch, and an error is computed. The update procedure is used to enhance the model based on this inaccuracy. The number of epochs is a hyperparameter that controls how many times the learning algorithm runs over the whole training dataset. Each sample in the training dataset has had the chance to change the internal model parameters once each epoch. There are one or more batches in an epoch.

3.2. Autoencoders

An autoencoder is a neural network that has been trained to replicate the input to the output (Kingma & Welling, 2013). Autoencoders are limited in their architecture or training techniques to require them to only duplicate the input properly. Because it can't represent everything accurately, these limitations drive the autoencoder to learn just the most significant properties of the training data. Autoencoders have been used for anomaly detection, dimensionality reduction, and feature extraction, among other things, since learned features frequently give a better representation of the data. The autoencoder usually consists of two parts; the encoder part aims to represent the input data with fewer variables with a code. The decoder part tries to reveal the input data again by using these code outputs. A simple autoencoder architecture is shown in *Figure 3.3*. The architecture has one hidden layer as an encoder, and the latent variable has a 2 neuron which means we present 3 inputs to 2 latent variables; the decoder is the same with the encoder part, and finally, we have the output layer the same with the input layer. The difference between input and output variables are defined as reconstruction error, and we can take the absolute value of reconstruction error depending on our purpose. In the training phase, the autoencoders aim to minimize the reconstruction error.

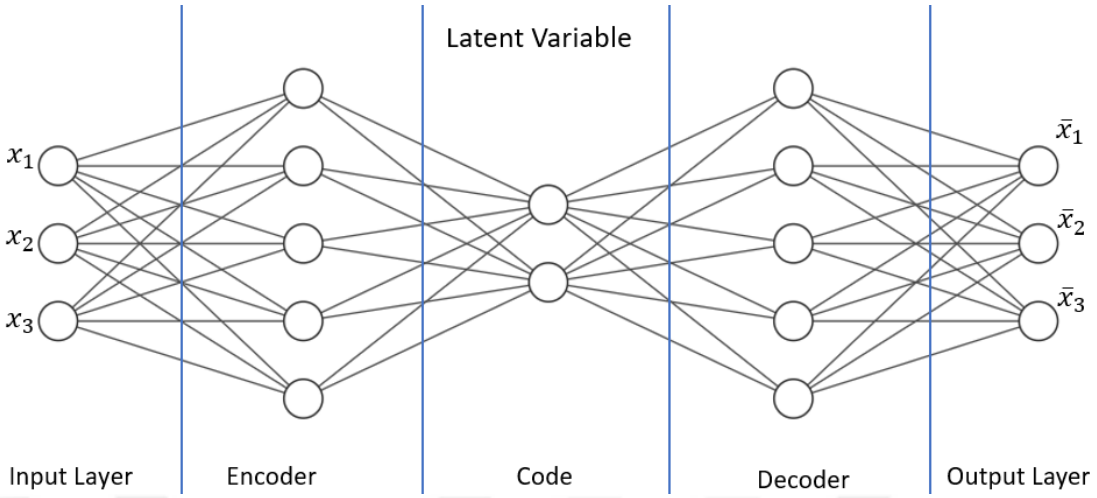

Figure 3.3 Architecture of an autoencoder

Autoencoders may be under-complete, in which case latent variable size is less than the input layer size, forcing them to learn a compressed representation of the data. For dimensional reduction tasks, an autoencoder can be employed in this framework. In fact, an autoencoder is demonstrably equivalent to Principal Component Analysis (PCA), where the weights of the K hidden units are the same subspace as the first K principal components of the data if the autoencoder has only one hidden layer, and the functions are linear, and the loss is mean squared error (Murphy, 2012). This shows the connection between PCA and autoencoder.

Let's say input $x \in R^n$ and f is the mapping function for latent variables(z). The encoders calculate

$$z = f(x) = v_f(w * x + b_z) \tag{5}$$

where v_f represents an activation function of the encoder, w is the weights, and b is a bias term. After that, the decoder(g) calculates outputs from latent variables to outputs.

$$\tilde{x} = g(z) = g\big(f(x)\big) = v_g(w' * x + b_x) \qquad (6)$$

where v_g represents an activation function of the encoder, w' is the weights, and b is a bias term.

Autoencoder trying to find set of parameters $\emptyset$=(W, b_x, b_z) to minimize the loss function that is also called reconstruction error.

There are many types of autoencoders, and some of them are explained in the following sections. The autoencoder architecture is very flexible, and the researcher also can combine different or modify the networks.

3.2.1. Denoising Autoencoders

Denoising autoencoders are a modification of basic autoencoders. It's important to mention that denoising autoencoders were not designed to automatically denoise an image. The training starts with adding noise to input values (they are corrupted), and the model aims to create original input from the corrupted input (Vincent et al., 2008). The corruption process typically adds Gaussian noise to the inputs at the beginning training process. Also, we can randomly add noises or remove some points in the inputs. For example, in *Figure 3.4*, some input values are masked.

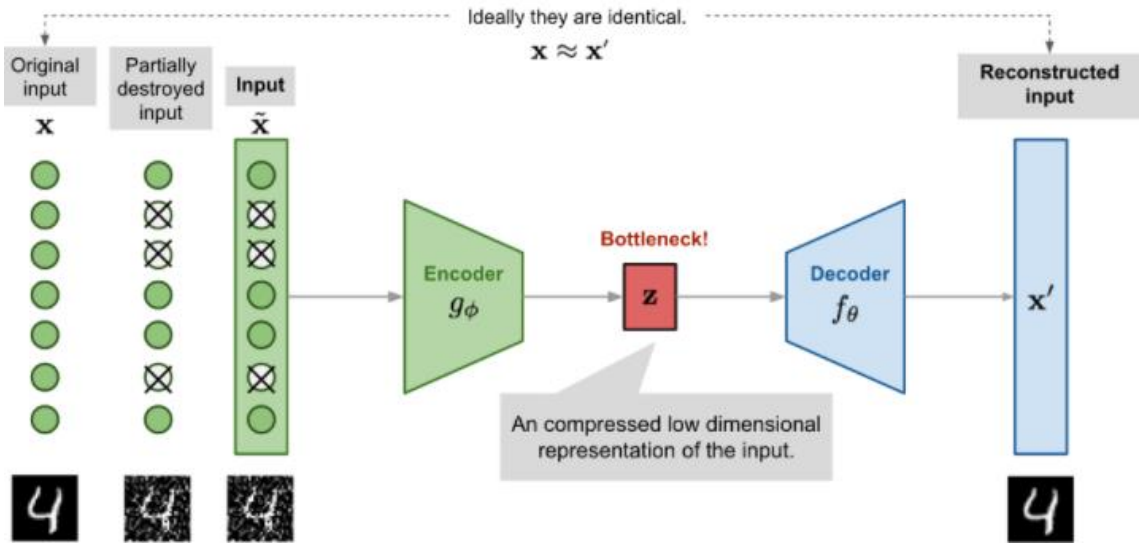

Figure 3.4 A denoising autoencoder model architecture *(Weng, 2018)*

3.2.2. Sparse Autoencoders

We can use a sparsity penalty to design the autoencoder so that only part of the nodes, called active nodes, have non-zero values this type of autoencoders are called sparse autoencoders. We usually include a penalty term in the loss function to ensure that only a portion of the nodes is active. This pushes the autoencoder to represent each input as a collection of a few nodes, allowing it to identify beneficial features for representing the data. Because the hidden layer is only active for a small subset of the nodes in the code at any given moment, this strategy works even if the code size is larger than the input.

3.2.3. Variational Autoencoders[1]

The variational autoencoder concept differs from that of all other autoencoder models in that it is thoroughly founded in variational Bayesian and graphical model approaches (Weng, 2018). We want to map the input into a distribution rather than a fixed vector. Let us refer to this distribution as p_θ, which is parameterized by θ. The following

[1] Calculation and explanation of VAE refer to (Kingma & Welling, 2019)

equations define the relationship between the data input x and the latent encoding vector z .

- Prior $p_\theta(z)$
- Likelihood $p_\theta(x|z)$
- Posterior $p_\theta(z|x)$

Assume we know the true parameter (θ^*) for this distribution. To produce a sample that resembles a real data point ($x^{(i)}$),we perform the following steps:

1. Firstly, take a sample ($z^{(i)}$) from a prior distribution $p_{\theta^*}(z)$
2. Secondly, generate a value $x^{(i)}$ from a conditional distribution $p_{\theta^*}\left(x\middle|z = z^{(i)}\right)$

The optimum parameter (θ^*) is the one that optimizes the possibility of producing real data samples:

$$\theta^* = \arg\max \theta \prod_{i=1}^{n} p_\theta\left(\boldsymbol{x}^{(i)}\right) \tag{7}$$

We usually use log probabilities to convert the right-hand product into a sum:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} \log p_\theta\left(\mathbf{x}^{(\mathbf{i})}\right) \tag{8}$$

Let us revise the equation to depict better the data generating process by including the coding vector:

$$p_\theta\left(x^{(i)}\right) = \int p_\theta\left(x^{(i)}\middle|z\right) p_\theta(z)dz \tag{9}$$

Unfortunately, calculating $p_\theta\left(\boldsymbol{x}^{(\boldsymbol{i})}\right)$ in this manner is quite costly to validating all potential values of $\boldsymbol{z}$ and add them. To help with speedier searching, we'd want to create a new approximation function that outputs a probable code given the input $\boldsymbol{x}$, $q_\phi(z|x)$, and parameterized with ϕ .

The structure is now like that of an autoencoder:

- Similar to the decoder $f_\theta(x|z)$,the conditional probability $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ defined a generative model. $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is also referred to as a probabilistic decoder.
- The approximation function $q_\phi(\mathbf{z}|\mathbf{x})$ is the *probabilistic encoder* and has a similar role with $g_\phi(z|x)$.

The computed posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be fairly similar to the actual $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. Kullback-Leibler divergence can be used to calculate the distance between these two distributions. When the distribution Y is used to represent X, the KL divergence ($D_{\mathrm{KL}}(X|Y)$) assesses how much information is lost. In *Figure 3.5*, the relationship between the terms is shown.

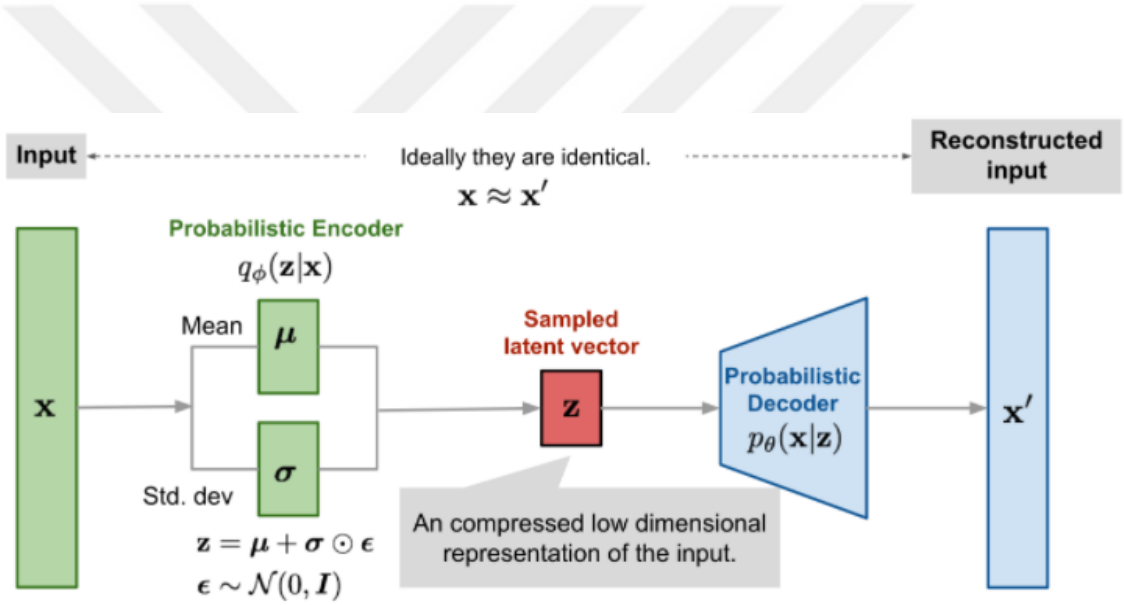

Figure 3.5 Variational autoencoder architecture *(Weng, 2018)*

In our problem, we want to keep $D_{\mathrm{KL}}\left(q_\phi(z|x)|p_\theta(z|x)\right)$ as low as possible with respect to ϕ. We can show our equation:

$$D_{\mathrm{KL}}\left(q_\phi(z|x)|p_\theta(z|x)\right) = \int \mathrm{q}_\phi(\mathrm{z}|\mathrm{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \mathrm{dz} \tag{10}$$

$$= \int q_\phi(z|x) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{z}, \boldsymbol{x})} dz \tag{11}$$

$$= \int q_\phi(z|x) \left(\log p_\theta(x) + \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}, \boldsymbol{x})} \right) dz \tag{12}$$

$$= \log p_\theta(x) + \int q_\phi(z|x) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\theta(\boldsymbol{z})} dz \tag{13}$$

$$= \log p_\theta(x) + \mathbb{E}_{z\sim q_\phi(z|x)} \left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z})} - \log p_\theta(x|z) \right] \tag{14}$$

$$= \log p_\theta(x) + D_{\mathrm{KL}} \left(q_\phi(z|x) || p_\theta(z) \right) - \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) \tag{15}$$

Finally, we have:

$$D_{\mathrm{KL}} \left(q_\phi(z|x) | p_\theta(z|x) \right) = \log p_\theta(x) + D_{\mathrm{KL}} \left(q_\phi(z|x) || p_\theta(z) \right) - \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) \tag{16}$$

If we rearrange equation (16), it becomes:

$$\log p_\theta(x) - D_{\mathrm{KL}} \left(q_\phi(z|x) | p_\theta(z|x) \right) = \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - D_{\mathrm{KL}} \left(q_\phi(z|x) || p_\theta(z) \right) \tag{17}$$

When learning actual distributions, the left-hand side of the equation is what we want to optimize: we want to maximize the chance of creating real data ($\log p_\theta(\boldsymbol{x})$) while also minimizing the difference between the real and projected posterior distributions. It's worth noting that $p_\theta(x)$ is proportional to q_ϕ.

When we take the negative value of above, it defines the loss function:

$$L_{\mathrm{VAE}}(\theta, \phi) = -\log p_\theta(\boldsymbol{x}) + D_{\mathrm{KL}} \left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) | p_\theta(\boldsymbol{z}) \right) \tag{17}$$

$$= -E_{z\sim q_\phi(z|x)} \log p_\theta(x|z) + D_{\mathrm{KL}} \left(q_\phi(z|x) | p_\theta(z) \right) \tag{18}$$

$$\theta^*, \phi^* = \arg \min_{\theta,\phi} L_{VAE} \tag{19}$$

This loss function is known as the variational lower bound, or evidence lower bound, in Variational Bayesian techniques. The "lower bound" component of the term originates from the fact that KL divergence is never negative, hence $-L_{VAE}$ is the lower bound of $\log p_\theta(\mathbf{x})$.

$$-L_{\mathrm{VAE}}(\theta, \phi) = \log p_\theta(\boldsymbol{x}) - D_{\mathrm{KL}} \left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) || p_\theta(\boldsymbol{z}|\boldsymbol{x}) \right) \leq \log p_\theta(\mathbf{x}) \tag{20}$$

As a result, we maximize the lower bound on the likelihood of obtaining real data samples by minimizing the loss. It is very hard to calculate the correct parameters to provide both of them, so we use the reparameterization trick.

The expectation term in the loss function causes $z \sim q_\phi(z|x)$ samples to be generated. We can't backpropagate the gradient since sampling is a stochastic process. The reparameterization approach is used to make it trainable: The random variable $\boldsymbol{z}$ may frequently be expressed as a deterministic variable, $\mathrm{z} = \mathcal{T}_\phi(\mathrm{x}, \epsilon)$ where is an auxiliary independent random variable and the transformation function $\mathcal{T}_\phi$ parameterized by transforms $\boldsymbol{\epsilon}$ to $\mathbf{z}$. Not only does the reparameterization approach work for Gaussian distributions, but it also works for other types of distributions. We make the model trainable in the multivariate Gaussian case by learning the distribution's mean and variance, and explicitly applying the reparameterization approach, while the stochasticity remains in the random variable $\epsilon \sim \mathcal{N}(0; 1)$.

4. METHODOLOGY

This section introduces the proposed anomaly detection framework for a retail company. We first describe how to decide product categories and collect data from the big data platform. Secondly, we introduce variables and how to create time features. Next, we explain how we overcame the missing and extreme values. Then, we present how to apply VAE. Finally, we describe how to use the result of VAE and how to evaluate the generation of alarms. The workflow of the methodology is shown *Figure 4.1*.

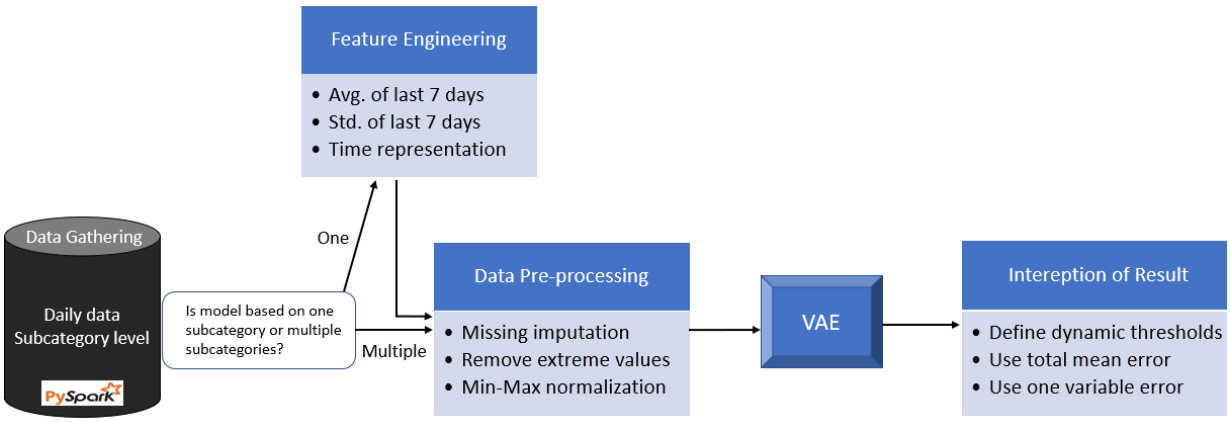

Figure 4.1 The overview of the methodology

4.1. Data Gathering

We collected data from the big data platform by using pyspark. Pyspark has been released to support the collaboration of Apache Spark and Python, and it is a Python API for Spark. In the meeting we held with the business unit, they suggested that we do it based on subcategory, which is the next level of SKU in the product hierarchy, as the brands of the

products change over time. To give an example for product hierarchy, A brand pasteurized milk and B brand pasteurized milk are in the same p*asteurized milk subcategory*; Organic milk, pasteurized milk, yoghurt and ayran are in the *Milk &Yogurt category*; Ice-cream, cheese and milk&yoghurt are in the *dairy main category*. We pick a store and sum up the number of products under the same subcategory daily, so we have a dataset including day and inventory stock quantity. The product hierarchy is shown in *Figure 4.2*.

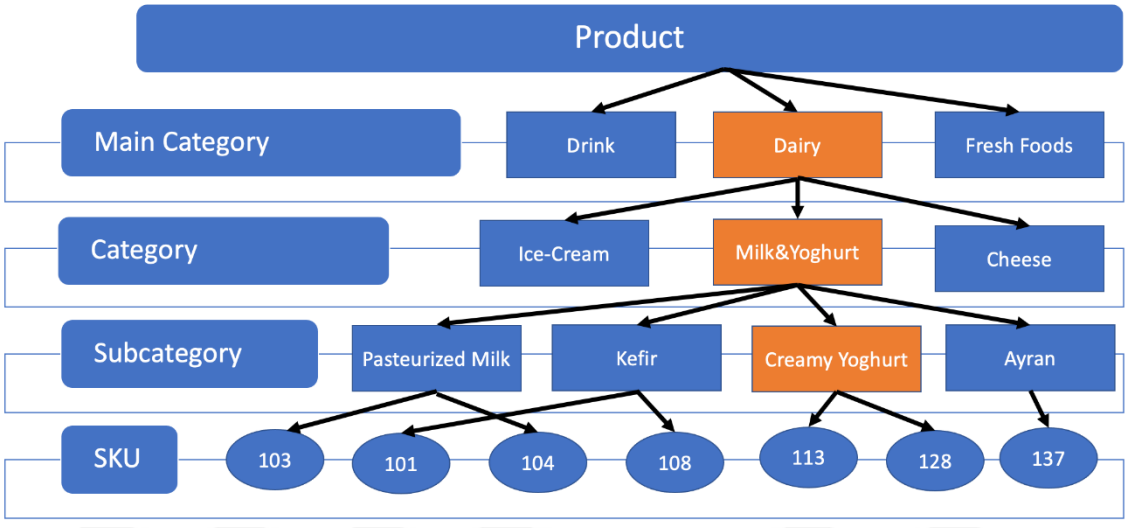

Figure 4.2 An example of product hierarchy

If the firm wants to examine just one subcategory, it can take only that subcategory from the data and include it in the model. In addition, if he wants to consider more than one subcategory at the same time and analyze the interaction between them in the model, they should consist of other categories in the model. The first case is handled as a univariate time series, while the other is treated as a multivariate time series. Examples for univariate and multivariate time series are shown below in Figure 4.3:

date	kaymakli_yogurt
2016-01-01	0.309524
2016-01-02	0.269841
2016-01-03	0.420635
2016-01-04	0.373016
2016-01-05	0.412698

date	ayran	kefir	salep
2016-01-01	0.543011	0.676056	0.272727
2016-01-02	0.500000	0.676056	0.227273
2016-01-03	0.736559	0.661972	0.227273
2016-01-04	0.725806	0.647887	0.227273
2016-01-05	0.569892	0.394366	0.500000

Figure 4.3 Univariate & multivariate time series

4.2. Data Explanations & Feature Engineering

After collecting the data, we need to decide which type of anomalies we want to detect. Because in univariate time series, we only have the day and the inventory stock of that day. To detect the univariate time series anomalies, we create some base variables that can give information about time series. The first step is to create features based on current (It is shown as x_t) and past quantity, and these features give information about the history of data and changes in data. The list of variables and their calculation is shown in the table:

Table 4.1 Calculations of variables

Variable	Explanation	Calculation
x_t	Current day stock quantity	Calculation no needed
$x_{pctchange}$	Percent change from previous day	$\frac{(x_t - x_{t-1})}{x_{t-1}}$
$x_{avglast7}$	Average of last 7 days	$\frac{1}{7}\sum_{i=0}^{6} x_{t-i}$
$x_{stdlast7}$	Standard deviation of last 7 days	$\sqrt{\frac{1}{7-1}\sum_{i=0}^{6}(x_{t-i} - \bar{x})}$
$x_{CVlast7}$	Coefficient of variation	$\frac{x_{stdlast7}}{x_{avglast7}}$

The second step is the time representation that is critical to catch seasonality and pattern depending on the date. The most popular method is one-hot encoding, where each categorical value is transferred to a particular feature in the dataset containing binary 1 or 0. One of the disadvantages of this method is that it increases the size of the data. For example, if we use one-hot encoding for the day of year variable, we must add 365 columns to our data. To not increase the size of the data, based on (Chakraborty & Elzarka, 2018) we defined time as sine and cosine functions. Thus, we can represent cyclical features with only two variables in the (x,y) coordinates on a cycle. When we think numerically, we can say that the difference between the 1st day of the year and the 365th day is quite large. If we include this information in the model as is, it may not detect that there is only one day in between. So, we aim to create two different components and take advantage of the proximity of their location on the circle. We draw the plot of the sin and cos component of a day in the year in *Figure 4.4*. Then We showed the place of these numbers on the circle in *Figure 4.5*.We create features for the year's day, week of the year, and month. The calculation for each variable is shown below:

Table 4.2 Represent time as a sin-cos function

Variable	Explanation	Calculation
daysin	Sine of the year's day	$\sin\left(\frac{2 * \pi * current\ day}{365}\right)$
daycos	Cosine of the year's day	$\cos\left(\frac{2 * \pi * current\ day}{365}\right)$
weeksin	Sine of the week of the year	$\sin\left(\frac{2 * \pi * current\ week}{52}\right)$
weekcos	Cosine of the week of the year	$\sin\left(\frac{2 * \pi * current\ week}{52}\right)$
monthsin	Sine of the month of the year	$\sin\left(\frac{2 * \pi * current\ month}{12}\right)$
monthcos	Cosine of the month of the year	$\cos\left(\frac{2 * \pi * current\ month}{12}\right)$

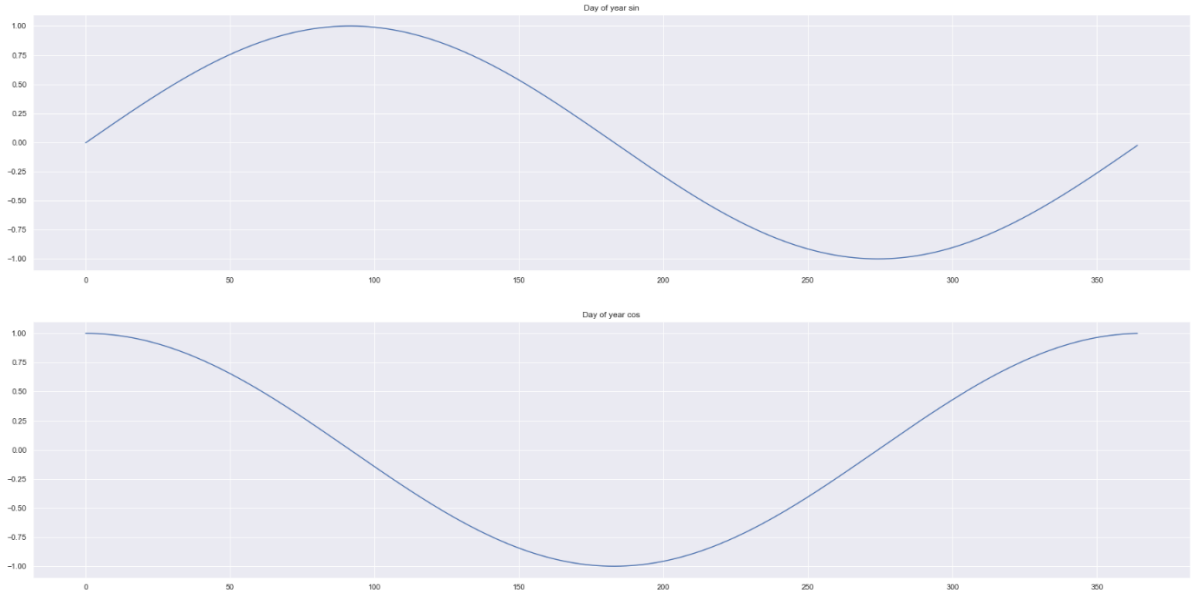

Figure 4.4 The day of year as a sin and cos wave in separately

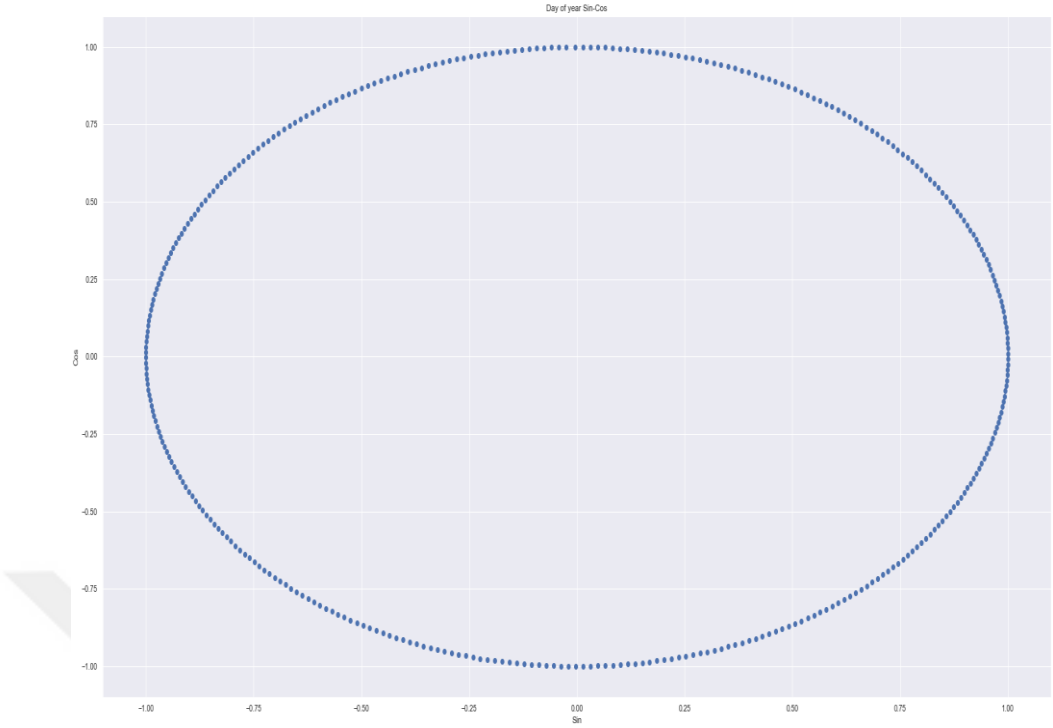

Figure 4.5 The day of year as a function of sin and cos

On the other hand, we do not create base features such as average and time representation features for multivariate time series because the company wants to catch the anomaly that caused interaction between the subcategories. For example, let's write a scenario, what happens if a store may order traditional milk instead of pasteurized milk? The traditional milk subcategory inventory stock will increase, and new products do not replace the pasteurized milk subcategory stock. The store has too many stocks in the traditional milk subcategory; it is overstocking. But the store will stock out because of less pasteurized milk. This mistake directly affects the cost and customer satisfaction.

4.3. Data Preprocessing

Data preprocessing refers to the steps involved in transforming or encoding data to be easily interpreted by an algorithm (Garcia et al., 2015). Each data point may have unique properties, making it challenging to standardize the data. Furthermore, several issues may

occur while gathering data. Data may be collected from several sources, there may be an issue with the system's flow, or the data going to the source may be incorrect.

For instance, suppose you collect data from numerous sources for students at school. Similarly, it is doubtful that all data with hundreds of records will be correct. For example, the age information or the student's surname may be missing in some records.

We preprocess the data to make it easier to understand and use. This procedure removes inconsistency or duplications in data that might impair a model's accuracy. Data preparation also guarantees that no values are wrong or missing due to human mistakes or defects. In short, applying data preparation techniques improves the completeness and exactness of the data.

We do data preprocessing in three parts; the details are explained in the following sections.

4.3.1. Data imputation

The substitution of approximated values for missing or unreliable data elements is known as data imputation (Efron, 1994). The replacement values are meant to provide a data record that is not prone to errors while editing. In our problem, some values of inventory stock can be missing because of system errors, or there is no stock; we impute missing values with 0. Also, occasionally inventory stock is less than zero because, on some days, the number of products returned by customers may be higher than the product sold in subcategories; we replace negative stocks with 0.

4.3.2. Removing extreme values

On the other hand, a dataset may contain extreme values that are outside the acceptable ranges and dissimilar to the overall data (Coles et al., 2001). Identifying and even deleting these can help enhance machine learning modelling and model skills in general. We use the interquartile range (IQR) method that is the difference between the 75th and 25th percentiles ($Q_3 - Q_1$) of the data (Kokoska & Zwillinger, 2000). Generally, extreme values are defined as $1.5 * IQR$ below Q_1 or $1.5 * IQR$ above Q_3. Based on our conversation with the business unit, we do not want to lose any anomalies in data; therefore, we take $3 * IQR$ instead of $1.5 * IQR$. Also, the business unit analyzes the extreme values separately. According to these, we replace the extreme values with null values. To impute these null values, we use linear interpolation that defines linear polynomials to create new data points inside the range of a discrete set of known data points. The linear interpolation equation (5) is shown below, and the figure for the equation can be found in *Figure 4.6*.

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \tag{21}$$

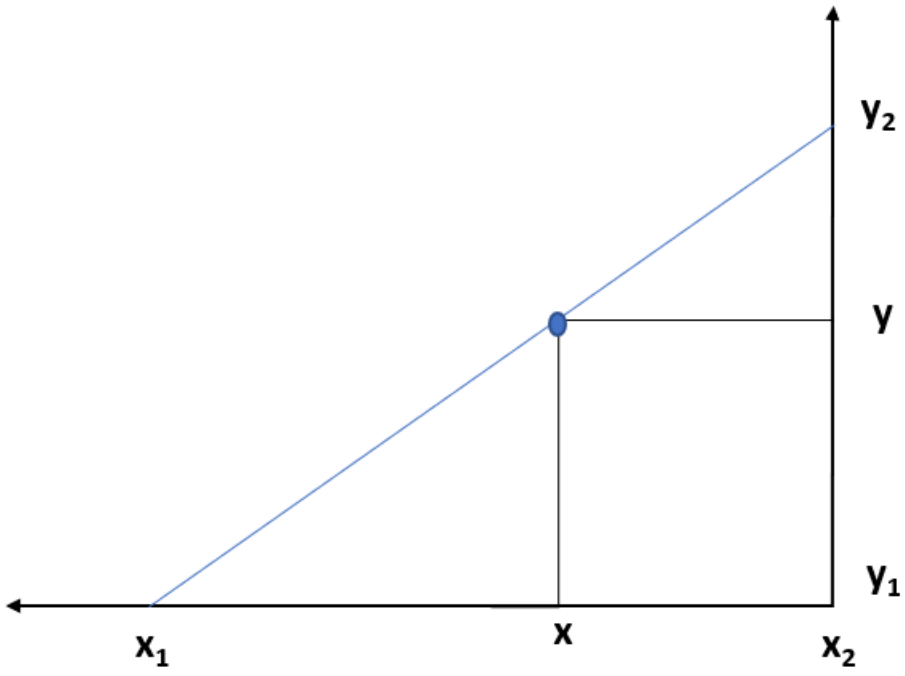

Figure 4.6 Linear interpolation point example

4.3.3. Data Normalization

The second step is data transformation, converting data from one format to another useful format. We use feature-wise min-max normalization transformation because variables measured at different scales do not contribute equally to the model fitting and learned function and may result in bias. We can formulate min-max transformation as:

$$x_{new} = \frac{x_i - min(x)}{max(x) - min(x)} \tag{22}$$

When we look at *Table 4.3*, the original values are between 8 and 12. When we apply min-max normalization, the range becomes 0-1. The maximum value is 12, and it becomes 1 in the new representation. We apply this method to all variables in the dataset.

Table 4.3 An example of Min-Max Normalization

Original Value	Calculation	Normalized Value
10	$\frac{(10-8)}{(12-8)}$	0.5
8	$\frac{(8-8)}{(12-4)}$	0
12	$\frac{(12-8)}{(12-8)}$	1
9	$\frac{(9-8)}{(12-8)}$	0.25

4.4. Variational Autoencoder

The theoretical background of VAE is explained in 4.4, and we want to use some advantages of VAE. Firstly, when we compare AE and VAE, VAE provides more options to adjust, allowing us to have more flexibility in modelling our latent distribution.

Secondly, VAE learns the data distribution; this is very important when your data changes over time. Also, we can combine VAE with other neural network architectures, such as long short-term memory (Nguyen et al., 2021) and convolution neural networks (Metlapalli et al., 2020).

We can easily say that customer behaviours change over time, and the companies have to change their strategy based on customers. Companies collect a lot of data about their customers and consumer habits. To increase customer satisfaction and decrease cost, stores should be managed effectively, including the inventory store. The methods should be dynamic, that is why we apply VAE.

4.5. Interpretation of Result

The output of VAE is the new representation of each input value. We compute the reconstruction error for each feature by calculating the absolute value of the difference between prediction and real values. Also, we calculate the mean of the reconstruction error that is also referred to as the mean absolute error (MAE) for each day; let us define this error as $total\_anomaly\_score$. According to the business unit, the system should not generate alarms frequently, so we decided; if the reconstruction error is greater than $Q_3 + 1.5 * I\text{QR}$, for each feature, we put a flag as an anomaly for these records' value. On the other hand, we put a flag for $total\_anomaly\_score$ if it is higher than $Q_3 + 1.5 * I\text{QR}$ for $total\_anomaly\_score$. Because of using this method, our anomaly definition becomes parametric, and our threshold is based on the IQR method. We can focus on a specific variable or analyze the combination of all variables together by using $total\_anomaly\_score$. An example of calculation for a variable given is shown below:

$error = \lvert x_{pctchange} - \bar{x}_{pctchange} \rvert$	Difference between real and output value
$Q_1 = 25^{th}\ error$	Order errors from lower to higher take 25th error
$Q_3 = 75^{th}\ error$	Order errors from lower to higher take 75th error

$IQR = Q_3 - Q_1$	Interquartile range
$\boldsymbol{flag} = \begin{cases} 1, if\ error > Q_3 + 1.5 * IQR \\ 0, \quad else \end{cases}$	Create flag based on IQR

We calculate the flag for all variables. After that, we create an anomaly flag table that includes all variables and the total anomaly score based on our anomaly definition. The business unit can look at a single variable or other variables together by using the total anomaly score flag.

5. EXPERIMENTS

This thesis is aimed to find anomalies in the inventory stock by using real-life data. We define inventory records inaccuracy as an anomaly; when the proposed system detects the inventory errors, the inventory management will be more effective, and customer satisfaction will increase. On the other hand, generally, researchers are forced to use artificial data in their studies because of data privacy. Therefore, it is extremely important to use real-life data and demonstrate how to handle problems. However, another goal of this thesis is to provide a generic framework for unsupervised anomaly detection that can be used in all-time series data.

We collect the main dataset for a store by using pyspark from the big data platform. Our data includes real inventory stock of 18 different subcategories under the milk&yoghurt category. Our dataset is between January 2016 and December 2019, with 2020 and 2021 omitted due to the pandemic. The company want to see the result for normal years, and after that, they can apply this method to other time intervals. Our sample size is 1461 with 18 columns, and raw data includes inventory stock information daily, so we have 1461 different days.

date	ayran	boza	kefir	salep	aromali_sut	ozellikli_sut_uzun_omurlu	sade_sut_uzun_omurlu
2016-01-01	0.543011	0.7	0.653333	0.272727	0.575521	0.236842	0.620553
2016-01-02	0.500000	0.7	0.653333	0.227273	0.552083	0.231579	0.588933
2016-01-03	0.736559	0.7	0.640000	0.227273	0.528646	0.231579	0.557312
2016-01-04	0.725806	0.7	0.626667	0.227273	0.505208	0.231579	0.529644
2016-01-05	0.569892	0.6	0.386667	0.500000	0.802083	0.215789	0.640316

Figure 5.1 A snapshot of the dataset[2]

Depending on the preference, anomaly detection can be univariate and multivariate time series. Firstly, we propose univariate time series anomaly detection for the creamy yoghurt subcategory. After that, we develop a model that includes 18 different subcategories.

5.1. Univariate Time series Anomaly Detection

Univariate time series anomaly detection provides the company to focus on just one subcategory. Especially, products with a short expiration date are very critical to catching. So, we select the creamy yoghurt subcategory from the main dataset that includes 1461 samples and one column (inventory stock quantity). After that, we create features as we mentioned in 4.3, such as percent change from the previous day, average and standard deviation of last seven days, coefficient of variation, time representation with sine & cosine for the day of the year, week of year and month of the year. At the end of the feature engineering, we have a total of 11 variables. After that, we impute missing and negative values with 0. The representation of daily inventory stock data is shown in *Figure 5.2*.

[2] The numbers are not real because of data privacy

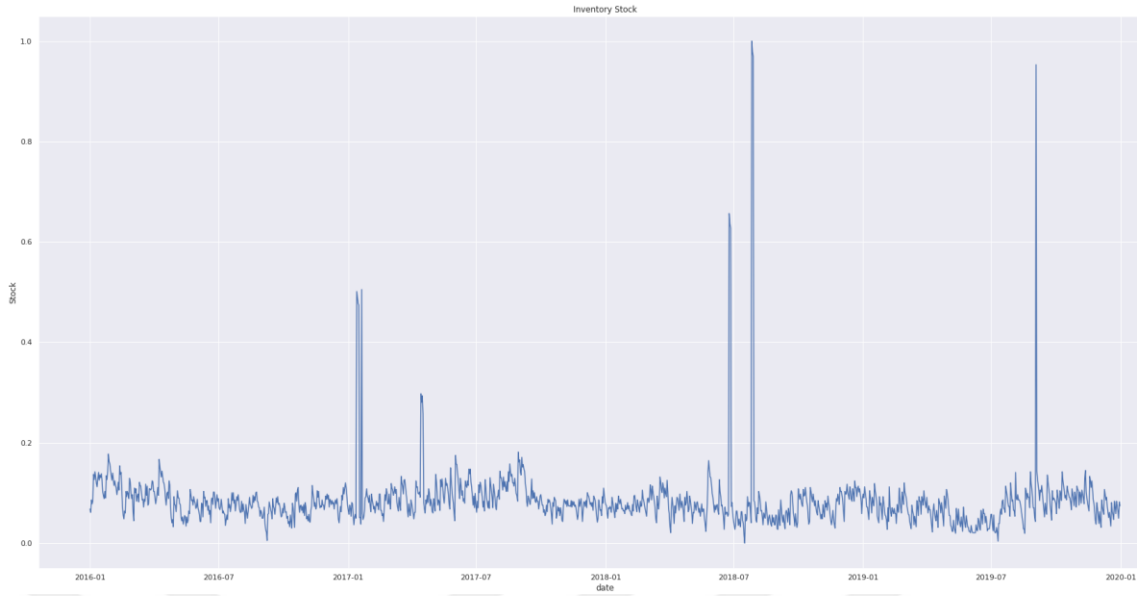

Figure 5.2 Daily stock of creamy yoghurt

When we look at *Figure 5.2*, we can easily see some values are very high. Then we apply our *I*QR method to our data, if the values are cached by the IQR method, we replace them with null values. To fill null values, we do imputation with linear interpolation. After that, these extreme values are replaced with more consistent values. The graph of actual inventory stock is shown in *Figure 5.3*.

Figure 5.3 Daily stock of creamy yoghurt after removing extreme values

In the data transformation step, we apply min-max normalization to our data; so, all variables are between 0 and 1. Nextly, we create a VAE model using the Keras library in python. In the model, we have three layers for encoder and decoder separately. To explain the encoder (11-20-3), we have 11 neurons equal to the variable number in the input layer. The hidden layer includes 20 neurons, and the latent variable consists of three neurons; the decoder is symmetric with the encoder (3-20-11). We use the batch size is 32, and the epoch number is 250. The representation of architecture is shown in *Figure 5.4*.

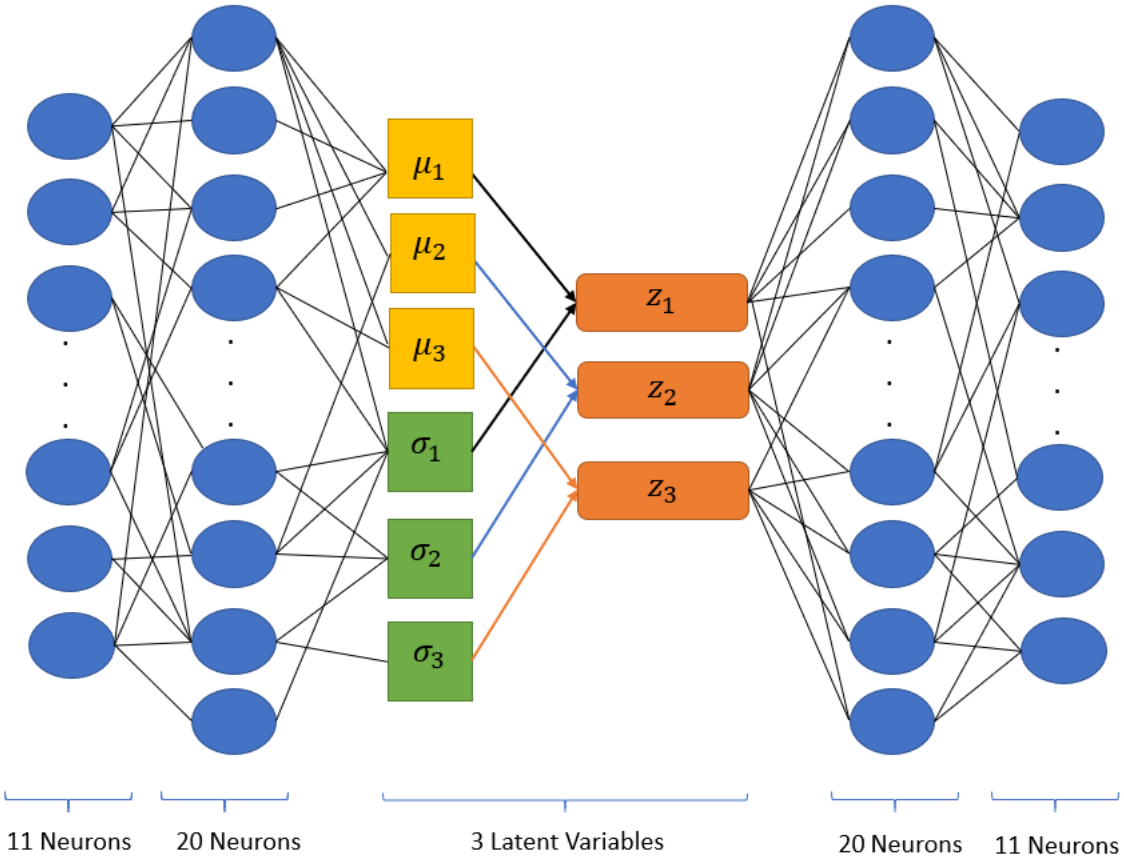

Figure 5.4 The architecture of VAE

We train the model to use mean squared error as the loss function. After 250 epochs, the final loss is 0.3113. If we increase the epoch number or create a more complex structure, the model can overfit the data. Also, when we look at the history of the loss function in *Figure 5.5*, we can say the model is learning.

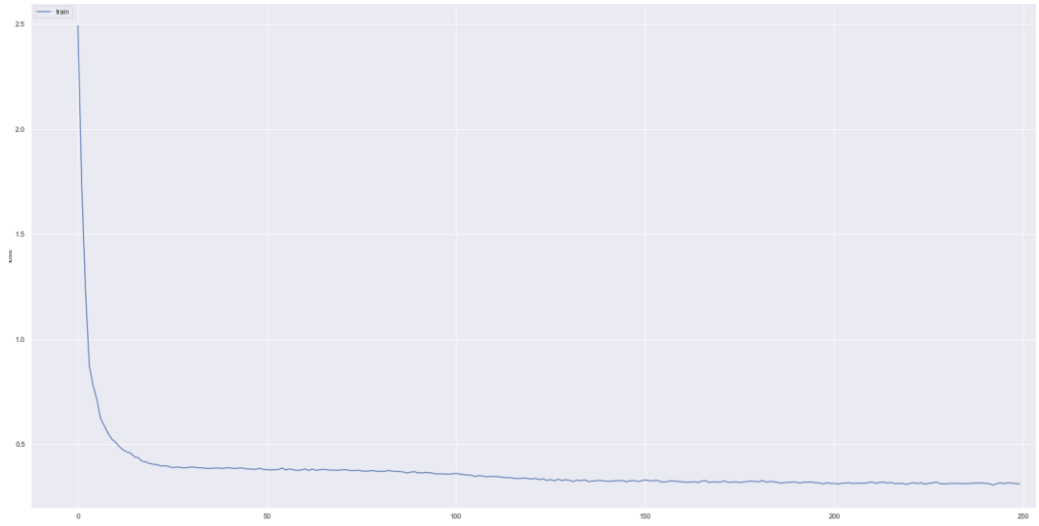

Figure 5.5 The loss function history

After fitting our model, we calculate the absolute error for each feature and the total anomaly score. The graph of the total anomaly score can be found in *Figure 5.6*. When we look at the graph, we can say that it seems like a normal distribution. So, approximately 95% of the score is within two standard deviations of the mean.

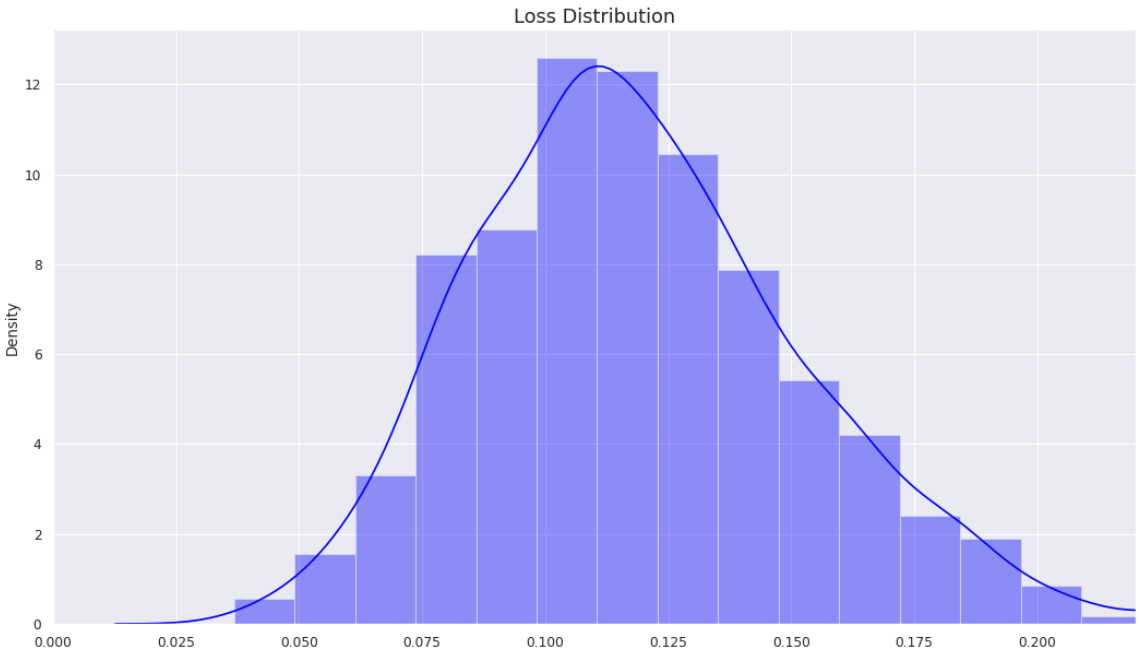

Figure 5.6 The distribution of loss

We define a day as an anomaly if the total anomaly score is greater than $Q_3 + 1.5 * I\text{QR}$ or the error of actual inventory stock is higher than $Q_3 + 1.5 * I\text{QR}$ therefore, we have 48 anomalies in our data and it is just 3.2% of all records. Because it produced few alarms, the business unit could easily review the recordings. Another good thing about the model is that it marks high values as anomalies and detects low values as anomalies. The business unit also can focus on other variables anomalies by using their error. For example, percent change from the previous day is critical to catching demand intensity suddenly. If they look at the anomaly points for this feature, they can analyse and make inferences about customer behaviours. The number of anomaly flags in the dataset is given in *Table 5.1*. When we examine the table, we can easily see thresholds are different for each variable. Because of this, we can analyse all variables separately, or we can just look at the total anomaly score.

Table 5.1 The thresholds and number of anomalies

Variable	Flag threshold	Number of anomalies
$\boldsymbol{x_t}$	0.396	27
$\boldsymbol{x_{pctchange}}$	0.034	59
$\boldsymbol{x_{avglast7}}$	0.366	42
$\boldsymbol{x_{stdlast7}}$	0.259	47
$\boldsymbol{x_{CVlast7}}$	0.303	53
total_anomaly_score	0.181	28

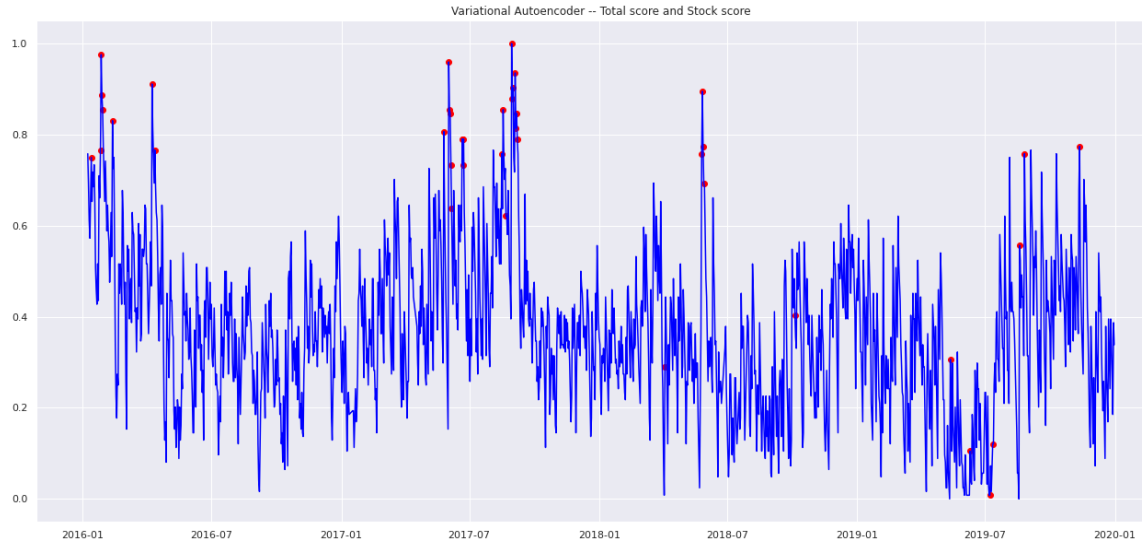

Figure 5.7 Anomaly points in creamy yoghurt

In *Figure 5.7*, we can see anomaly points in creamy yoghurt that is based on the total anomaly score and actual stock. Generally, high-level inventory stocks are marked; however, we can see some low ones. Another critical variable is the percent change of inventory stock. If we look at the graph, we see that most of the big jumps are marked by the model. We want to look at the anomalies based on percent change score, it is very easy and it is shown in *Figure 5.8*. This analysis is can be done for each variable.

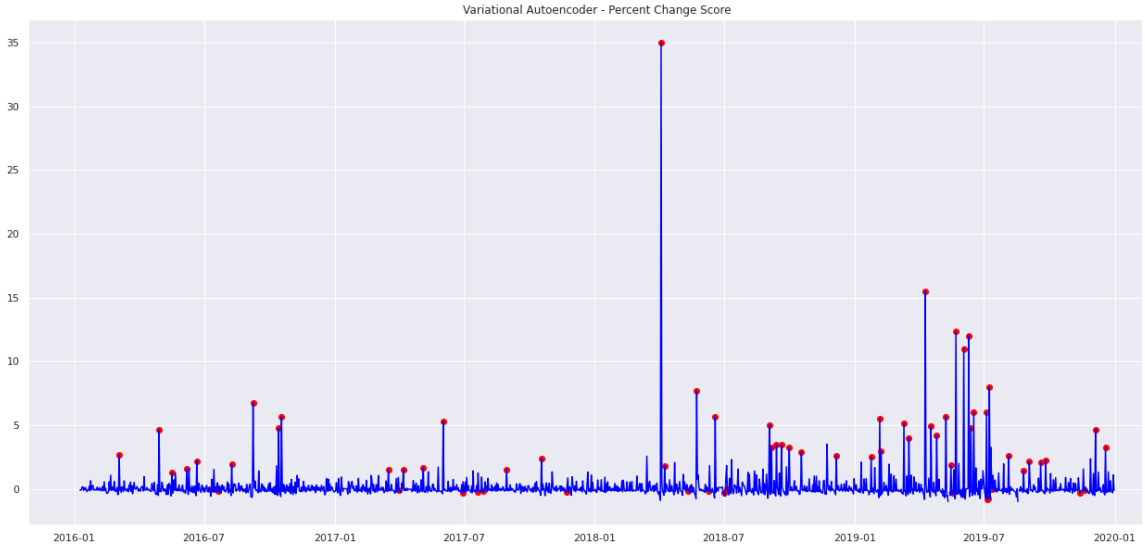

Figure 5.8 Anomaly points based on percent change variable

5.2. Multivariate Time Series Anomaly Detection

Multivariate time series anomaly detection provides the company to focus on more than one time series. The company can create one model to detect anomalies in the 18 subcategories to address our problem. Our data includes real inventory stock of 18 different subcategories under the milk&yoghurt category. In the dataset, we have 1461 samples and 18 columns representing various subcategories' inventory stock.

We skip the feature engineering part because the column number will increase to 60 (4 features for a variable plus time representation) when adding each variable's features. It is easier to analyse 18 variables than the 60 variables. Also, extra features can have interaction with other subcategories variables.

We impute missing and negative values with 0. The representation of daily inventory stock data is shown in *Figure 5.9*. We do not show all subcategories because the data gets mixed up when there are too many lines on the chart.

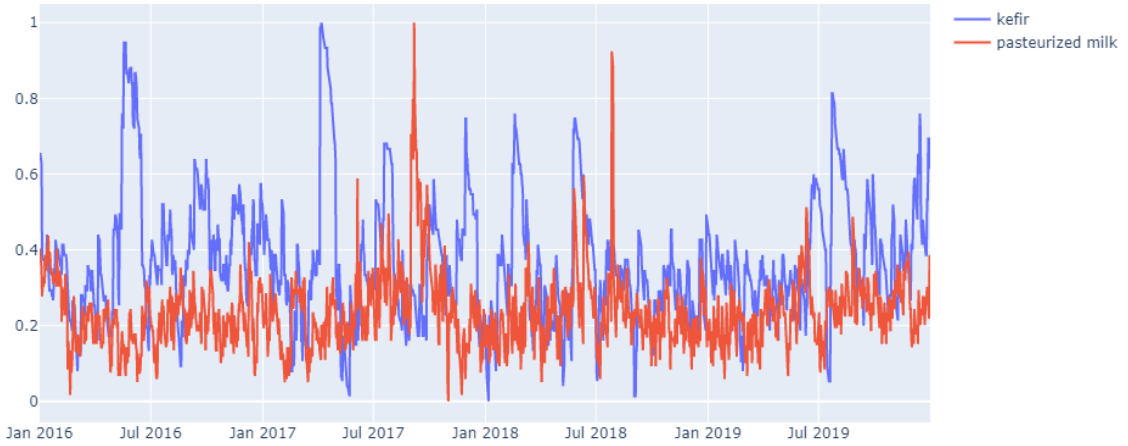

Figure 5.9 Daily stock of kefir and pasteurized milk

When we look at *Figure 5.9*, we can easily see some values are very high. Then we apply our *I*QR method to our data, if the values are cached by the IQR method, we replace them with null values. To fill null values, we do imputation with linear interpolation. After that, these extreme values are replaced with more consistent values. The graph of actual inventory stock is shown in *Figure 5.10*.

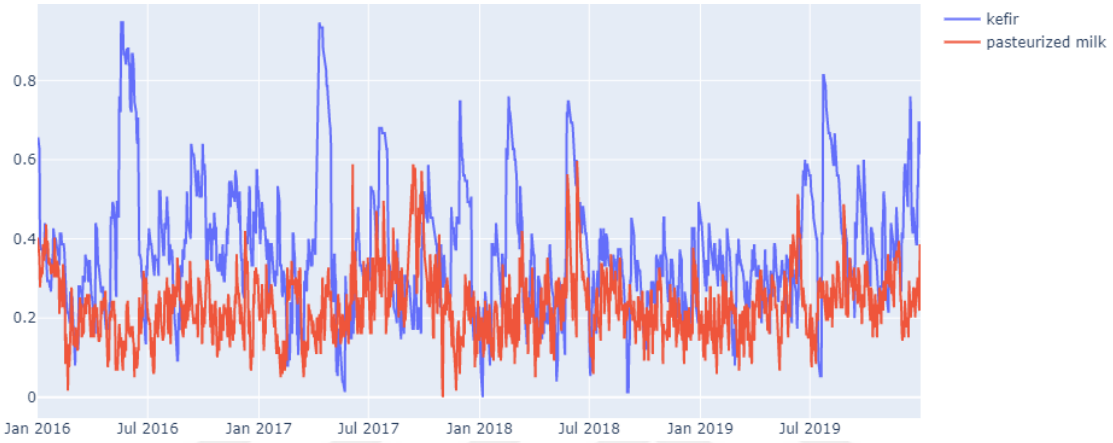

Figure 5.10 Daily stocks after removing extreme values

In the data transformation step, we apply min-max normalization to our data; so, all variables are between 0 and 1. Nextly, we create a VAE model using the Keras library in python. In the model, we have three layers for encoder and decoder separately. To explain the encoder (18-25-5), we have 18 neurons equal to the variable number in the input layer. The hidden layer includes 20 neurons, and the latent variable consists of three neurons; the decoder is symmetric with the encoder (5-25-18). We use the batch size is 32, and the epoch number is 500. The representation of architecture is shown in *Figure 5.11*.

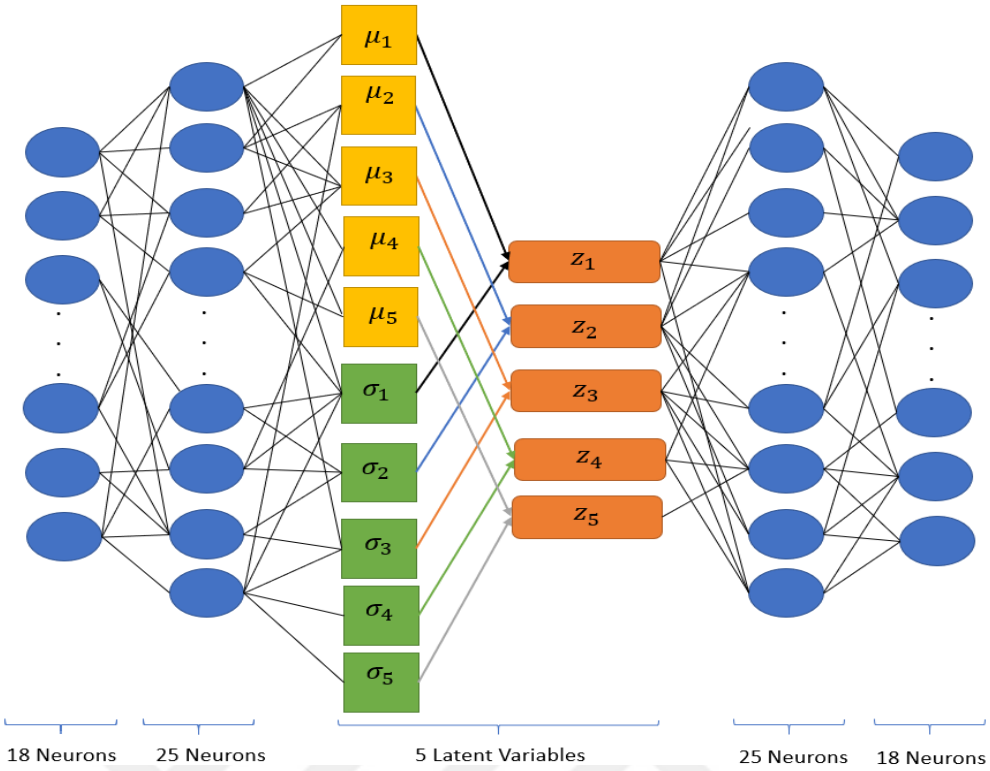

Figure 5.11 The architecture of VAE for multivariate

We train the model to use mean squared error as the loss function. After 500 epochs, the final loss is 0.3325. We increase the epoch number and create a more complex structure than the previous model. Because columns number is increased, the problem becomes more challenging. We do not add extra layers to prevent overfitting, increase training time. Also, when we look at the history of the loss function in figure Figure 5.12, we can say the model is learned.

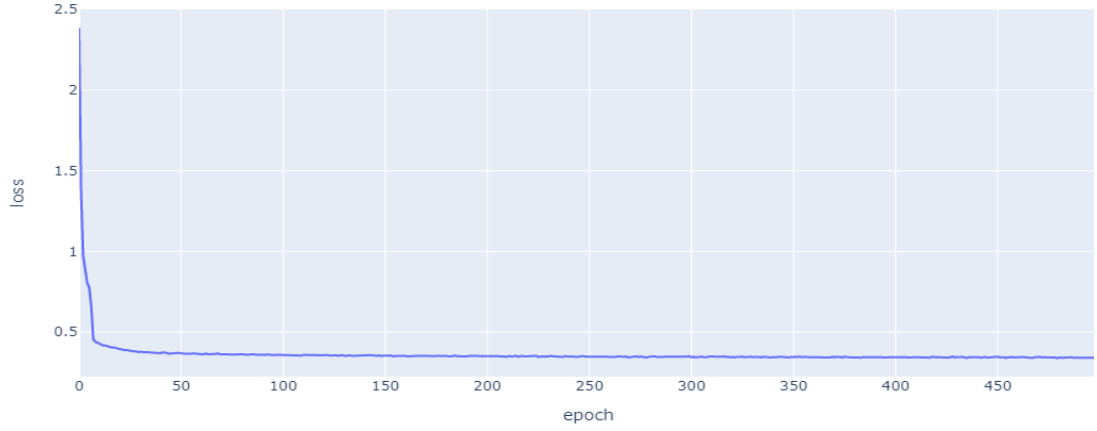

Figure 5.12 The loss function history 2

After fitting our model, we calculate the absolute error for each feature and the total anomaly score. The graph of the total anomaly score can be found in *Figure 5.13*. When we look at the graph, we can say that it seems like a normal distribution. So, approximately 95% of the score is within two standard deviations of the mean. Also, the loss range is lower than the previous model. However, we can analyse loss for each subcategory separately like *Figure 5.14 Loss for different subcategories*.

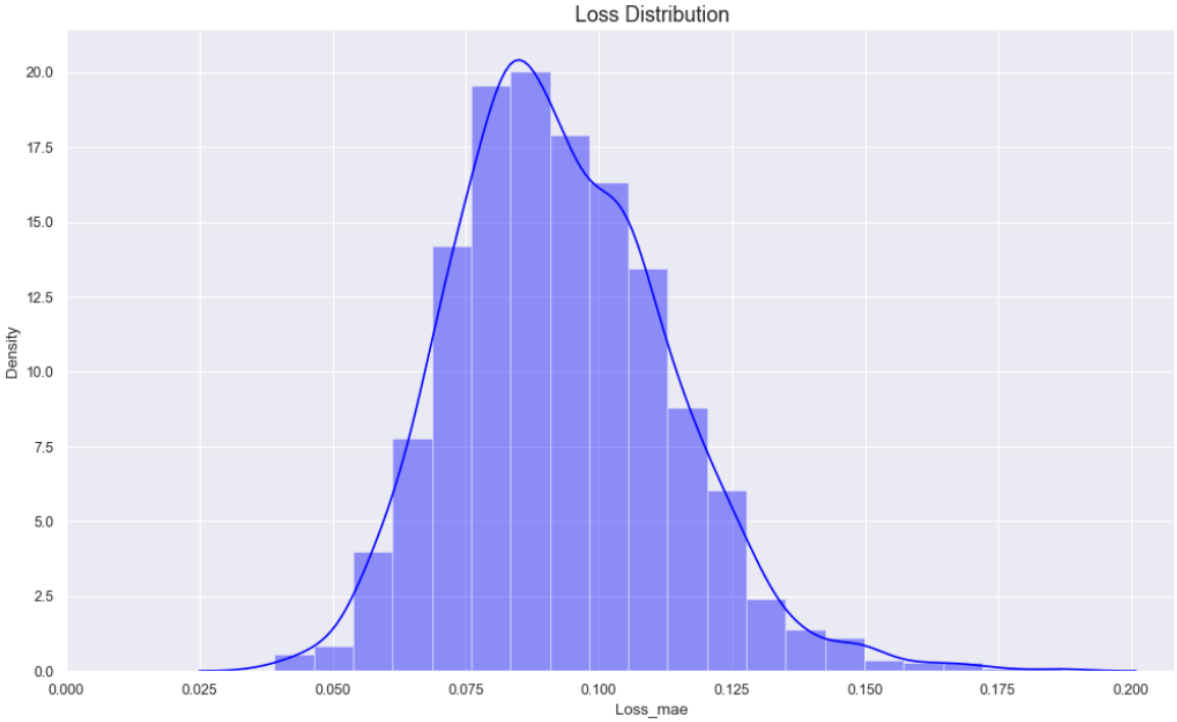

Figure 5.13 The distribution of loss 2

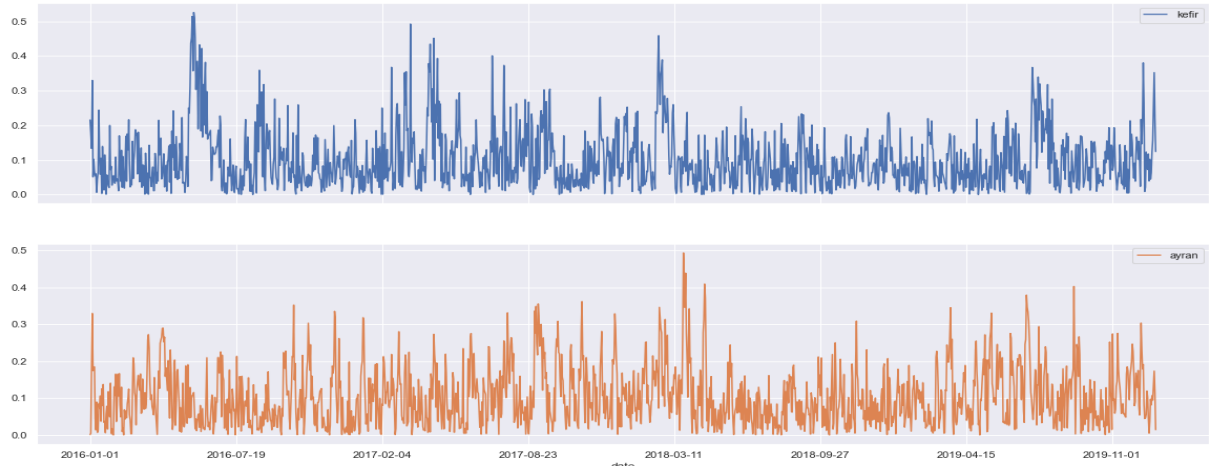

Figure 5.14 Loss for different subcategories

We define a day as an anomaly if the subcategory reconstruction error is greater than $Q_3 + 1.5 * IQR$. Then, we have different anomaly numbers for each subcategory. The model marks high values as anomalies and detects low values as anomalies in the different subcategories. The business unit can focus on one subcategory or more than one category at the same time. The number of anomaly flags in the dataset is given in *Table 5.2*. When we examine the table, we can easily see thresholds are different for each subcategory.

Table 5.2 The thresholds and number of anomalies for subcategories

Subcategory	Flag threshold	Number of anomalies
A	0.311	29
B	0.299	49
C	0.293	23
D	0.283	24
E	0.251	33
F	0.296	40
G	0.225	59
H	0.287	38
J	0.284	36
K	0.304	40
L	0.311	28
M	0.288	23
N	0.208	82
Creamy yoghurt	0.289	33
O	0.296	24
P	0.292	30
R	0.295	33
S	0.323	24

However, when looking at the creamy yoghurt subcategory, the anomaly number is 33, and it is greater than the first model (it is 27). We can say, the different subcategories interact with each other.

5.3. Which approach should be used in which case?

First of all, both approaches have their advantages, and of course, the business unit will make the right choice. Univariate time series anomaly detection aims to find anomaly points in one subcategory. All features that we create separately are related to inventory stock. Therefore, when points are marked as an anomaly, we can easily say that it is because of the inventory value. But it is not easy to create the model for each subcategory separately, and it is not easy to manage it. We can create models in the power machine to handle this challenge, but it will be very costly for the companies. At this point, we can consider using multivariate time series anomaly detection. We can use all subcategories in the one model in the multivariate approach. The main disadvantage of this approach is that it may be marked as an anomaly when the point is an anomaly because it interacts with other points. The framework is created as generic so that we can apply it to different stores in different subcategories or main categories.

5.4. Why did we select Variational Autoencoder in this problem?

When we examine the literature, we can easily say that there are many methods for unsupervised anomaly detection.So, we want to explain why the Variational Autoencoder method is more suitable for our needs compared to the existing methods, as follows:

- VAE learns the distribution of the data so the algorithm can easily adapt to new data even if the data changes over time. For example, the standard autoencoders method works statically and learns only the most important properties from the data.
- VAE also calculates an error for each variable in the input data. Thus, it allows examining more than one time series or variable in a single model. Isolation forest, one of the most popular unsupervised anomaly detection methods, calculates only a single anomaly score. So, we have to create different models for each time series.
- We can define our thresholds dynamically in the VAE, so we can easily decrease the number of alarms that are generated by the system.

6. CONCLUSION AND FUTURE WORKS

The discrepancy between the quantity recorded in a company's inventory management system and the number actually physically available is known as the inventory record inaccuracy (IRI). IRI can cause major problems in the retail industry, such as stockouts and revenue loss due to poor stock replenishment.

This study detects the errors in the inventory and defines these as an anomaly. It has a unique positioning as our dataset is taken from real-life while previous studies heavily relied on artificial datasets. Anomaly detection is a prevalent challenge for large industries such as retail industry. This research aims to contribute to the development of the unsupervised approach for anomaly identification. The key benefit of the method is its applicability to different types of time series data. The lack of labels creates the biggest challenge for this study, as they introduce limitations for evaluating the previous cases. Indeed, in the absence of tags and ground truth, evaluating metrics and criteria for unsupervised anomaly detection algorithms remains a difficult practical challenge despite few recent studies on the subject.

Furthermore, in unsupervised anomaly detection, the idea of normality, which is typically intuitive for specialists, proves the presence of challenges to define anomaly in formal terms. However, from the point of anomaly detection, describing what is normal appears to be more rational than determining what is abnormal. In large-scale anomaly detection applications, the definition of what is abnormal is frequently conditioned by the company's ability to respond to these abnormalities. An AD algorithm under such

constraints aims to select the most abnormal observations that are able to checked and confirmed by the company or service provider.

This thesis proposes a generic, unsupervised, and scalable framework for anomaly detection in time series data, which is retail inventory stock data in this study. The suggested approach meets all of the objectives established in the early stages of this research and can detect aberrant behaviour in data from very various domains, contributing to several probable application areas of AD such as finance and health.

One of the primary contributions of this thesis is based on the approach's unsupervised character. While unsupervised learning is significantly more challenging than supervised learning, the proposed system has a promising capacity to learn meaningful data representations and subsequently detect anomalous occurrences. The two anomaly detection approaches have been shown to function well on the real dataset, allowing for a quick way to locate data that do not adhere to usual behaviour. Another contribution is dynamic threshold definition, as it is not easy to analyse all features in the datasets. On the other hand, applying machine learning algorithms to real-life data is not straightforward, especially in unsupervised learning. The study provides approaches on how to overcome the problems (e.g., noises, missing values) deriving from the structure of the data in real life. From a retail perspective, we can apply our method to different industry components such as product categories, stores, and warehouses. Implementation of our approach to the inventory management system could help industry players prevent errors before they occur. The business units could assess and validate the model results, which could further feedback and improve the unsupervised anomaly detection model. However, the business unit examined our anomaly points. For the most part, they confirmed many high-value points as anomalies but they give feedback that lower-valued ones needed further investigation.

Although our approach has shed some light on the subject with promising suggestions, further research could expand our knowledge about unsupervised anomaly detection and overcome the study's shortcomings. Following recommendations could benefit the literature and provide opportunities for real-life applications:

- In the multivariate approach, clustering of the subcategories may improve the overall model accuracy as clusters would have similar characteristics.
- The scope of the current study is limited to one store of a retail company, the replication of the study in different geographical regions could provide additional insights.
- The model's different variants (e.g., models with fewer layers, latent variables, batch size) can be developed at varying complexity levels.
- SKU-based prediction models could be designed, and the differences between prediction and real values can be examined to generate insights.
- As it stands, the development of a comprehensive model to cover all different products and their abnormalities is not very likely. However, the attempts to improve the flexibility and applicability of the model may get us closer to the desired state for unsupervised anomaly detection.

REFERENCES

Aggarwal, C. C. (2013). An Introduction to Outlier Analysis. In *Outlier Analysis* (pp. 1–40). New, York, NY: Springer New York. doi:10.1007/978-1-4614-6396-2_1

Aggarwal, C. C., & Reddy, C. K. (2014). Data clustering. *Algorithms and applications*.

An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE, 2*, 1–18.

Breunig, M. M., Kriegel, H.-P., & Ng, R. T. (2000). LOF: identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 93-104.

Chakraborty, D., & Elzarka, H. (2018). Advanced machine learning techniques for building performance simulation: a comparative analysis. *Journal of Building Performance Simulation*, 193-207. doi:https://doi.org/10.1080/19401493.2018.1498538

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.

Chehrazi, N. (2020). Impacts of Inventory Record Inaccuracy on Retailers' Internal Operations. *Available at SSRN 3637031*.

Chen, T., Liu, X., Xia, B., Wang, W., & Lai, Y. (2020). Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *IEEE Access, 8*, 47072–47081.

Chire. (2016, 5 14). *Wikimedia.* Retrieved 1 2, 2022, from LOF: https://commons.wikimedia.org/wiki/File:LOF.svg

Chuang, H. H.-C., Oliva, R., & Liu, S. (2016). On-shelf availability, retail performance, and external audits: a field experiment. *Production and Operations Management, 25*, 935–951.

Coles, S., Bawa, J., Trenner, L., & Dorazio, P. (2001). *An introduction to statistical modeling of extreme values.* Springer.

DeHoratius, N., & Raman, A. (2008). Inventory Record Inaccuracy: An Empirical Analysis. *Management Science, 54*, 627-641. doi:10.1287/mnsc.1070.0789

Dhiman, A., Gupta, K., & Sharma, D. K. (2021). Chapter 1 - An introduction to deep learning applications in biometric recognition. In V. Piuri, S. Raj, A. Genovese, & R. Srivastava (Eds.), *Trends in Deep Learning Methodologies* (pp. 1-36). Academic Press. doi:https://doi.org/10.1016/B978-0-12-822226-3.00001-5

Efron, B. (1994). Missing data, imputation, and the bootstrap. *Journal of the American Statistical Association*, 463-475.

Epstein, D. (2022, 1 1). *Various Types of Inventory Management Systems for Your Business*. Retrieved from FinancesOnline: https://financesonline.com/various-types-of-inventory-management-systems-for-your-business/

Ergen, T., & Kozat, S. S. (2019). Unsupervised anomaly detection with LSTM neural networks. *IEEE transactions on neural networks and learning systems, 31*, 3127–3141.

EugenioTL. (2021). Variational Autoencoder structure. *Variational Autoencoder structure*. Retrieved from https://commons.wikimedia.org/wiki/File:VAE_Basic.png

Filonov, P., Kitashov, F., & Lavrentyev, A. (2017). Rnn-based early cyber-attack detection for the tennessee eastman process. *arXiv preprint arXiv:1709.02232*.

Garcia, S., Luengo, J., & Herrera, F. (2015). Data Preparation Basic Models. In *Data Preprocessing in Data Mining* (pp. 39–57). Cham: Springer International Publishing. doi:10.1007/978-3-319-10247-4_3

Hinton, G. E. (2006). Reducing the dimensionality of data with neural networks. *Hinton, Geoffrey E and Salakhutdinov, Ruslan R*, 504-507.

Hinton, G., Deng, L., Yu, D., & Dahl. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 82-97.

Illukkumbura, A. (2021). *Introduction to Time Series Analysis.* Independently published.

Jianliang, M., Haikun, S., & Ling, B. (2009). The application on intrusion detection based on k-means cluster algorithm. *International Forum on Information Technology and Applications*, 150-152.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. doi:arXiv:1312.6114

Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.

Kök, A. G., & Shang, K. H. (2007). Inspection and replenishment policies for systems with inventory record inaccuracy. *Manufacturing & service operations management, 9*, 185–205.

Kök, A. G., & Shang, K. H. (2014). Evaluation of cycle-count policies for supply chains with inventory inaccuracy and implications on RFID investments. *European Journal of Operational Research, 237*, 91–105.

Kokoska, S., & Zwillinger, D. (2000). *CRC standard probability and statistics tables and formulae.* Crc Press.

Kriegel, H.-P., Schubert, M., & Zimek, A. (2008). Angle-based outlier detection in high-dimensional data. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 444–452).

LeCun, Y. (2018, 3 6). *Obstacle to Progress in Deep Learning & AI*. Retrieved from https://engineering.nyu.edu/news/revolution-will-not-be-supervised-promises-facebooks-yann-lecun-kickoff-ai-seminar

Li, C., Guo, L., Gao, H., & Li, Y. (2021). Similarity-Measured Isolation Forest: Anomaly Detection Method for Machine Monitoring Data. *IEEE Transactions on Instrumentation and Measurement*, 1-12.

Lindemann, B., Jazdi, N., & Weyrich, M. (2020). Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks. *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, (pp. 1003–1010).

London, I. (2016, 8 31). *Encoding cyclical continuous features - 24-hour time*. Retrieved December 13, 2021, from Encoding cyclical continuous features - 24-hour time: https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

Lou, C., & Zhao, H. (2015). Local density-based anomaly detection in hyperspectral image. *Journal of applied remote sensing, 9*, 095070.

Ma, M. X., Ngan, H. Y., & Liu, W. (2016). Density-based outlier detection by local outlier factor on largescale traffic data. *Electronic Imaging*, 1-4.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Melli, G. (2021, 11 12). *Artificial Neuron*. Retrieved 1 3, 2022, from Gabormelli: https://www.gabormelli.com/RKB/Artificial_Neuron

Metlapalli, A. C., Muthusamy, T., & Battula, B. P. (2020). Classification of Social Media Text Spam Using VAE-CNN and LSTM Model. *Ingénierie des Systèmes d'Information*, 747-753.

Murphy, K. P. (2012). Machine learning: a probabilistic perspective. *MIT press*.

Natsvlishvili, E., Lomtadze, E., & Khatiashvili, N. (2020). *ERP system implementation challenges in Georgian Medium-sized enterprises, retail sector.* Ph.D. dissertation, Ilia State University.

Nguyen, H., Tran, K. P., & Thomassey. (2021). Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*.

Ounacer, S., Bour, E., Oubrahim, H. A., Ghoumari, Y. a., & Azzouazi, M. Y. (2018). Using Isolation Forest in anomaly detection: the case of credit card transactions. *Periodicals of Engineering and Natural Sciences*, {394-400.

Pena, E. H., de Assis, M. V., & Proença, M. L. (2013). Anomaly detection using forecasting methods arima and hwds. *2013 32nd international conference of the chilean computer science society (sccc)*, (pp. 63–66).

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, *2*, 1–21.

Shabani, A., Maroti, G., de Leeuw, S., & Dullaert, W. (2021). Inventory record inaccuracy and store-level performance. *International Journal of Production Economics*, *235*, 108111.

Stevenson, W. J. (2021). *Operations Management,14th Edition.* McGraw-Hill.

Tao, X., Peng, Y., Zhao, F., Zhao, P., & Wang, Y. (2018). A parallel algorithm for network traffic anomaly detection based on Isolation Forest. *International Journal of Distributed Sensor Networks*.

Tayeh, T., Aburakhia, S., Myers, R., & Shami, A. (2020). Distance-based anomaly detection for industrial surfaces using triplet networks. *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, (pp. 0372–0377).

Tokovarov, M., & Karczmarek, P. (2021). A Probabilistic Generalization of Isolation Forest. *Information Sciences*.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. (pp. 1096-1103). Proceedings of the 25th international conference on Machine learning.

Weng, L. (2018). From Autoencoder to Beta-VAE. *lilianweng.github.io/lil-log*. Retrieved from http://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Wilson, W. H. (2012, 6 25). *The Machine Learning Dictionary*. Retrieved from The Machine Learning Dictionary: http://www.cse.unsw.edu.au/~billw/mldict.html#activnfn

Wu, S. (n.d.). *Supervised vs. Unsupervised Machine Learning*. Retrieved 10 5, 2021, from https://nlcshub.com/stem/computerscience/338/

Zadeh, A. H., Sharda, R., & Kasiri, N. (2016). Inventory record inaccuracy due to theft in production-inventory systems. *The International Journal of Advanced Manufacturing Technology, 83*, 623–631.

Zhang, C., Li, S., Zhang, H., & Chen, Y. (2019). VELC: A new variational autoencoder based model for time series anomaly detection. *arXiv preprint arXiv:1907.01702*.

Zhang, R., Zhang, S., Muthuraman, S., & Jiang, J. (2007). One class support vector machine for anomaly detection in the communication network performance data. *Proceedings of the 5th conference on Applied electromagnetics, wireless and optical communications*, 31-37.

Zhang, Y., Chen, Y., Wang, J., & Pan, Z. (2021). Unsupervised deep anomaly detection for multi-sensor time-series signals. *IEEE Transactions on Knowledge and Data Engineering*.