LUT UNIVERSITY
School of Engineering Science
Degree Programme in Industrial Engineering and Management

Anni Heikkinen

COMPARATIVE ANALYSIS OF OUTLIER DETECTION METHODS IN THE CONTEXT OF PROCUREMENT SPEND ANALYSIS

Supervisors:	Professor Pasi Luukka
	D.Sc. (Tech.) Samuli Kortelainen

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT

School of Engineering Science

Tuotantotalouden koulutusohjelma

Anni Heikkinen

Poikkeamien tunnistamiseen käytettävien algoritmien vertailu hankinnan kuluanalyysin kontekstissa

Diplomityö

2021

63 sivua, 12 kuvaa ja 11 taulukkoa

Tarkastajat: Professori Pasi Luukka ja TkT Samuli Kortelainen

Hakusanat: poikkeamien tunnistus, liiketoiminta-analytiikka, koneoppiminen, kuluanalyysi

Poikkeamat ovat väistämätön osa suuria data määriä. Koneoppimista hyödyntäviä poikkeamien tunnistamisalgoritmeja on ajan mittaan kehitetty. Tämän työn tavoitteena on etsiä mahdollisimman hyvä vaihtoehto valituista koneoppimisalgoritmeista tunnistamaan poikkeamia annetusta kuludatasta, hankinnan kuluanalyysin näkökulmasta. Algoritmien suorituskyky mainitussa kontekstissa analysoidaan ja yhtä algoritmeista suositellaan case yritys Sievolle eteenpäin kehitettäväksi ja implementoitavaksi osaksi sen tuotetta. Valitut algoritmit vertailussa ovat One-Class Vector Machine, pääkomponenttianalyysiin perustuva poikkeamien tunnistus, Isolation Forest ja Local Outlier Factor.

Kirjallisuuskatsaus käy lävitse kaikki neljä algoritmia teoreettisella tasolla. Lisäksi hankinnan kuluanalyysi ja kuludata esitellään. Poikkeama sekä poikkeamien tunnistaminen käsitteinä määritellään. Työn empiirisessä osassa käytetään oikeaa case yrityksen asiakkaan kuludataa. Algoritmien parametrit optimoidaan käytettävään kuludataan sopiviksi. Optimoituja algoritmeja testataan iteratiivisesti datalla ja suorituskykyä tarkastellaan kahden mittarin avulla – Excess-Mass ja Mass-Volume. Analyysiä tukee myös muut mittarit kuten poikkeamien suhteellinen osuus sekä algoritmin laskennallinen suorituskyky.

Excess-Mass ja Mass-Volume mittareiden perusteella yksi algoritmeista suoriutui parhaiten kuludatan tapauksessa. Syvemmän analyysin perusteella, case yritykselle suositellaan kyseistä algoritmia jatkokehitettäväksi sekä aiemmin mainittujen mittareiden että muiden algoritmin kokonaissuorituskykyyn liittyvien tekijöiden perusteella.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT

School of Engineering Science

Degree Programme in Industrial Engineering and Management

Anni Heikkinen

Comparative analysis of outlier detection methods in the context of procurement spend analysis

Master's thesis

2021

63 pages, 12 figures, and 11 tables

Examiners: Professor Pasi Luukka and D.Sc. (Tech.) Samuli Kortelainen

Keywords: outlier detection, business analytics, machine learning, spend analysis

Outliers are inevitable when working with large amounts of data. Outlier detection methods utilizing machine learning have been developed and are readily available to use. This thesis aims to determine the best alternative outlier detection machine learning algorithm for a given spend data set from the procurement spend analysis perspective. The algorithms' performance is analyzed, and a recommendation of an algorithm to be developed and implemented as a part of the case company Sievo's product is given. Outlier detection algorithms chosen for the comparison are the One-Class Support Vector Machine, Principal Component Analysis based method, Isolation Forest, and Local Outlier Factor.

The literature review gives information about the four algorithms in theory. In addition to this, spend analysis and spend data are presented, and the concept of an outlier and outlier detection is defined. In the empirical section, a real spend data set is used from the case company's client. The algorithms' parameters are tuned to be the most suitable for the sample spend data. The tuned algorithms are iteratively tested with the data, and the performance is measured using Excess-Mass and Mass-Volume. Other measures like counts and proportions of the detected outliers are also evaluated in the comparison to support the analysis.

The final results are that one of the studied algorithms performs best in terms of Excess-Mass and Mass-Volume measures for the particular spend data case. Based on further analysis, that algorithm is the suggested one for the case company for further development considering the above-mentioned measures and other factors related to the algorithm's overall performance.

ACKNOWLEDGEMENTS

This Master’s thesis completes my 4 and a half years of studies at LUT University. First, I would like to thank my family for supporting me along the way, especially during this thesis’ writing process. Special thanks to my partner Aleksi for your help and reassurance during my studies and these exceptional times of remote studying and working.

Writing this thesis has been a rather short but very educational journey professionally. I would sincerely like to thank my supervisors Pasi Luukka and Samuli Kortelainen, for guiding me through this project and challenge me to push further. I would also like to acknowledge Sievo for allowing me to work on my thesis part-time and provide this exciting thesis topic for me. Thank you, fellow Sievonians, for your patience during this unusual remote working time while writing this thesis. From my Sievo colleagues, I would like to address Matan Shenhav. Thank you for helping me, especially with the data and technical matters related to this process.

27.1.2021
Anni Heikkinen

TABLE OF CONTENT

LIST OF ABBREVIATIONS

ERP	Enterprise Resource Planning
P2P	Procure-to-Pay
PO	Purchase Order
MRO	Maintenance, Repair, and Operations
AP	Accounts Payable
GL	General Ledgers
OCSVM	One-class Support Vector Machine
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
iForest	Isolation Forest
iTree	Binary tree, a part of Isolation Forest
LOF	Local Outlier Factor
kNN	k-nearest neighbour
ROC	Receiver Operating Characteristic
EM	Excess-Mass
MV	Mass-Volume
PR	Precision-Recall
CSV	Comma-separated values
IQR	Interquartile Range

LIST OF SYMBOLS

S	A general estimate of a subset of the feature space
X	full sample data set
X_t	a training subset of X
x_l	lth training data point belonging to X_t
x	a single data point in X
$\mathbb{R}^N$	set of N-dimensional real numbers
w	perpendicular vector to the decision boundary

ρ	bias term
ξ	slack variable
$\varphi(\cdot)$	an implicit transformation function
$sgn(\cdot)$	a signum function
v	regularization parameter
X_i	the *i*th row of X
X_j	the *j*th column of X
D	diagonal matrix
P	orthonormal matrix
Σ	covariance matrix
$\overline{e_j}$	*j*th eigenvector
λ_j	*j*th eigenvalue
$\overline{\mu}$	centroid
T	a node of iTree
T_m	a child node of T
T_r	a child node of T
$\mid X \mid$	cardinality of X
$h(x)$	path length of x
$H(\cdot)$	estimates a harmonic number
$\text{avg}(\cdot)$	calculates average value
s	outlier score calculated for data point x
M	a parameter value of the number of neighbour data points
$N_M(x)$	the volume of the neighbourhood of x
o	a neighbour data point of x in X
e	resulting outlier score of one data point in X
E	a decision function including all the outlier scores for data points in X
$\mathbf{X}_z$	the *z*th independent and identically distributed random variable
$Leb(\cdot)$	calculates the Lebesgue integral
$\mathbb{P}_z$	probability of an event z
t	discretized Lagrangian multiplier
$\mathbb{1}$	indicator function
u	a unique element in the decision function of data set X

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

In this introductory chapter, the background and purpose of this thesis are presented. The research questions, limitations, delimitations and assumptions, and the case company Sievo are also introduced.

1.1 Background and the purpose of the study

Outlier detection has been an interesting subject for a long time. During the past few decades, the methods have quickly advanced due to the development of machine learning, pattern recognition, and data mining tools. (Mehrotra, et al., 2017) The earlier work on the outlier detection field has been in the statistics community. Drawbacks in outlier detection's statistical approaches have been deficient algorithmic scalability and not focusing on interpreting the models and results. (Aggarwal, 2013) Outliers appear in various fields and can reveal, for instance, failures in mechanical systems or credit card frauds (Hodge & Austin, 2004). In the field of procurement, organizations are increasingly collecting their data to analyze it. (Aggarwal, 2013) Spend analysis studies organizations spend data to find, for example, savings. Instead of outlier detection just being part of spend data cleansing, outliers can also indicate invoicing fraud of a vendor or employee. From the error point of view, outliers as part of the data can potentially lead to false conclusions and decisions in the business context.

In procurement and strategic sourcing, spend analysis has become an important asset to obtain potential savings. Spend analysis starts with the data. Organizing and enhancing the data provides a holistic view of the spend structures and patterns of the organization. Valid and good quality data ensures such analysis and view of the procurement processes. (Pandit & Marmanis, 2008) Outliers in the data might cause losing already mentioned potential savings or lead to false conclusions. The other side of the outlier detection in strategic sourcing is detecting fraudulent behavior of a supplier or simple human errors.

Even though outliers in spend analysis have not been studied very much, it has been recognized by the experts in the procurement data analytics field. Ruck (n.d) discusses outlier detection to ensure data quality in the procurement analytics field. Suplari (2019) points out outliers in procurement data analytics in their blog post. Fraudulent behavior detection could also be an application for outlier detection in procurement analytics, which Coupa Software Incorporated (Computer Weekly News, 2020) tries to tackle with clustering-based outlier detection. Data plays a crucial role in spend analysis, and good data is required to accomplish accurate analysis (Heath, 2006). One of the challenges listed by Heath (2006) was the incompleteness, inaccurateness, and inconsistency of the data. By proper outlier detection from the spend, data is one step closer towards better and more effective spend analysis.

The purpose of this study is to combine the two topics introduced – machine-learning-based outlier detection and strategic sourcing via data analytics. As data is an essential part of spend analytics, outliers cannot be avoided. Outlier detection algorithms could potentially provide a solution and a tool for spend analysis to discover, for instance, simple errors or fraudulent behavior in the sourcing processes and potentially increase the overall quality of the analysis. In this thesis, spend data is gathered from one anonymous customer of the case company Sievo. Outlier detection is delimited in identifying price outliers as well as possible. Four rather basic outlier detection algorithms were chosen for the study, and the purpose is to find the most suitable one for outlier detection in the spend data context. Finding the best alternative is accomplished by testing the algorithms in practice with real spend data and evaluating the unsupervised algorithms' performances.

Ruck (n.d) and Suplari (2019) provide some insight into the actual detection of the outliers from spend data and statistical approaches. This study's significance comes with the utilization of machine learning in the spend analysis field that Coupa (Computer Weekly News, 2020) touches on with the clustering-based approach to fraud detection in procurement. The topic is also essential for the case company. Machine learning-based implementation for outlier detection could help Sievo develop its product and the outlier detection to be more accurate. This thesis is an initial study on the machine learning-based outlier detection in spend data giving first directions and insights for further research around the topic.

1.2 The objective and research question of the study

This thesis aims to study the outlier detection in the spend data context and perform a comparative analysis within this specific context. The outcome is a suggestion of an outlier detection algorithm that the case company Sievo could potentially use as a part of the data preprocessing. The algorithms chosen for the study are One-Class Support Vector Machine (Schölkopf, et al., 2001), PCA-based outlier detection (Shyu, et al., 2003), Isolation Forest (Liu, et al., 2008), and Local Outlier Factor (Breunig, et al., 2000). The first two of these are requested by the case company to be included in the research. The last two algorithms were chosen based on the preliminary review before the thesis as conceptually slightly different options than the first two. Isolation Forest was chosen because it is an ensemble method. Local Outlier Factor is a density-based model alongside the Isolation Forest, which made it an interesting algorithm to be part of the study with two very different algorithms. Comparison of the algorithms is made based on the empirical study, which includes measuring the performance. The theoretical literature review strives to give the necessary information and knowledge about the chosen algorithms. A summary of the algorithms is gathered at the end of the theory section.

The research questions of the thesis are formulated as follows:

How do the One-Class SVM, PCA, Isolation Forest, and Local Outlier Factor algorithms perform compared to each other in price outlier detection in spend data?

Which algorithm is suggested for the case company to develop a machine learning-based outlier detection?

The study only includes the algorithms' basic forms, but a simple hyperparameter tuning is performed to find the optimal parameters for each algorithm for the task. A detailed analysis of the outliers that the algorithms identify is not in the scope of this thesis. Labeled data that would include either labeling for regular data points or both inliers and outliers is not available. Labeling each data point by hand is laborious. Thus, the algorithms in the study are implemented unsupervised, which means using unlabeled data.

1.3 Delimitations, limitations, and assumptions

Time limits the scope of this thesis to some extent. Many more outlier detection algorithms exist, but the number of examined algorithms in this thesis is delimited to four. The scope of the study has also been delimited to utilize the algorithms unsupervised. Creating labeling for the supervised and semi-supervised implementations requires an extensive amount of manual work of going through the data with an expert from the sample data owner company. Hence the labeling is not available for this study. The aforementioned sets an assumption that the chosen evaluation method of the unsupervised algorithm can rank the methods. The performance of the evaluation method itself is not studied further or compared to other evaluation methods.

1.4 Case company Sievo

Sievo is a Finnish software company and one of the global leaders in Procurement Analytics, providing intelligent spend management software. The company was established in 2003 and has over 180 employees in Helsinki, Finland, and Chicago, United States. Sievo manages over 350 billion dollars of spend yearly, and customer references include international companies like ISS, Carlsberg, and Schindler. (Sievo, 2020a) Sievo extracts the raw spend data. Spend data can be from ERP systems, general ledgers, purchase orders, and other systems. The most common of these are ERP systems, where Sievo collects the data into one platform. In Sievo, the raw data is cleansed and enriched. Cleansing and enriching include spend data classification, supplier consolidation, and for example, translations and currency conversions. Then the data is collected into reports to be analyzed. (Sievo, 2020b)

It is Sievo’s interest to provide more accurate and robust data and analytics for its customers. Outliers in data affect the data’s quality and accuracy, and Sievo is interested in detecting and cleansing the outliers from the data with a machine learning-based approach. Sievo’s customers are also interested in this as they are aware that their data may contain outliers and errors, leading to inaccurate spend analysis and insights.

1.5 Structure of the study

The structure of this study consists of five chapters. The first chapter introduces the topic and describes the significance of the study. The research question is presented, as well as the objective of the study. The study's scope is delimited in the introductory chapter, and the last two sections introduce the case company and the structure of the thesis. The second chapter is a literature review on the topics of this thesis. The literature about the spend analysis and spend data is reviewed. The concept of an outlier is defined to gain more understanding of the topic and give information to discuss the reasons for spend data outliers based on the literature. In addition to these, the second chapter includes the theory behind all the algorithms used alongside the theoretical background of the evaluation method used to rank the algorithms in the empirical study part in chapters three and four.

Chapter three describes the process and methodology that leads to the results in the fourth chapter. Chapter three describes the data preprocessing steps required and goes through the studied sample data set in an exploratory data analysis, including basic statistics and visualizations. The implementation of the algorithms is described in the third chapter. The performance evaluation method appliance in practice is the last section of chapter three, leading to chapter four. The study results are presented in the fourth chapter, the base for conclusions and discussion in chapter five. Chapter five summarizes the findings from the results and presents the conclusions drawn from the results. Also, recommendations for further research are given in the fifth chapter.

2 LITERATURE REVIEW

In this chapter, previous research and knowledge are gathered around the topic of this study. Spend analysis and spend data are introduced, and how the purchasing processes affect the spend data. Outliers are presented and defined alongside combining the theory behind outliers and spend data. All the outlier detection algorithms later tested in chapter 3, and the evaluation method to rank the algorithms are defined in this chapter's later parts.

2.1 Spend analysis

Strategic sourcing is playing a big role in the procurement operations of many organizations nowadays. The ability to access, organize, and analyze spend data plays a large role in more strategic and effective sourcing. Spend analysis is a process to do that. It analyzes historical purchasing data, also referred to as spend data, to identify savings opportunities and reduce costs. (Aberdeen Group, 2004; Pandit & Marmanis, 2008; Talluri & Narasimhan, 2004) Example questions that spend analysis aim to answer are, for example, the following:

- How much did the company spend on purchasing product X?
- How much did the company spend with supplier Y on product Z?
- What was the spend amount in a specific category N in the first quarter of last year?

The more complex type of questions that spend analysis could answer are:

- Are there suppliers that are more valuable and strategic than others, and which suppliers are they?
- What was the trend in spending over the last years?
- How much spend is there associated with preferred suppliers, and what percentage of this spend is contracted?

The existing data itself is not necessarily able to answer the questions introduced above. Pandit and Marmanis (2008) introduce the ETLA approach to implement the spend analysis solution to get there. E stands for extracting T for transforming, L for loading, and A for analysis in the ETLA approach. In addition to these, Aberdeen Group (2004) divides the transforming step into four: validation, cleansing, classification, and enhancing.

The data must be extracted from the system to get it into use. These systems can be Enterprise Resource Planning (ERP) systems or other internal and external data systems and files. Organizations usually have multiple different systems, and those may also be geographically scattered. To get comprehensive visibility of the organization’s procurement operations and spending, all these data sources must be synchronized (Pandit & Marmanis, 2008). In the transforming step, the data is normalized and enriched. This step also includes aggregating and organizing the spend data for the analysis. After the extraction, the Aberdeen Group (2004) identified three steps that include similar tasks as Pandit and Marmanis’ (2008) transformation step. The validation step ensures that the data is accurate and complete. After that, cleaning and classification are performed. The classification step can also include vendor consolidation and categorization.

The same vendor in the data can be entered in different ways. Pandit and Marmanis (2008) bring up IBM as an example vendor that can be entered as “I.B.M” or “International Business Machines” in addition to the regular “IBM” spelling. The spend is coming from the same vendor, but both would be analyzed as separate vendors if not consolidated into one. Similarly, categorization helps to gather, for example, purchasing data of the same material from different sources together if the same material had, for instance, different identifiers or naming in different source systems. Overall, categorization maps the data to a standard classification system that can be enterprise-specific and brings the data to a lower level for spend analysis. (Pandit & Marmanis, 2008; Aberdeen Group, 2004)

Cleansed data can be enriched or enhanced with some additional information. Vendor data can be enriched with additional information about the vendor’s diversity, quality, performance, or credit ratings. In addition to vendor enrichment, other kinds of enhancing information can also improve the spend data classification. After all the raw data preprocessing, it can be loaded into a database for further use. This loading part is the next step after transforming the data. (Pandit & Marmanis, 2008)

The data that has been preprocessed, cleaned, enhanced, and loaded to the database is ready for analysis. Necessary reports are run to get insights from the data. Implementing visualizations and taking business intelligence tools to look at the data and spend patterns from different dimensions. The most powerful way is to start analyzing the data from a higher level and advance to the transactional level. (Pandit & Marmanis, 2008)

2.2 Spend data

Spend transactions are the base for spend analysis. The process of generating the data is called the Procure-to-Pay (P2P) process. The process flow goes from contract to purchase order (PO) and from PO to invoice and payment (Figure 1). This process mostly applies to direct spend, which is the product's cost of being manufactured, such as purchasing raw materials. Some of the Maintenance, Repair, and Operations (MRO) spend is also generated by the P2P process. An example of MRO spend is spare parts or maintenance required for the production machinery. The P2P process is not as used in indirect spend, including other general and administrative costs like marketing costs or purchasing office supplies. (Pandit & Marmanis, 2008; Chowdhary, et al., 2011)

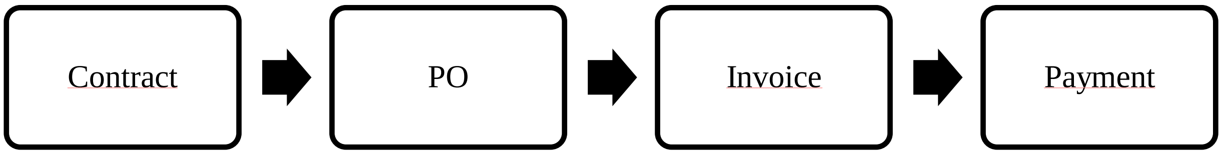

Figure 1 Procure-to-Pay process flow (Pandit & Marmanis, 2008)

Indirect spend does not usually follow the P2P process of purchasing. There are also other types of spend data than just the type that the P2P process generates. Invoices without a purchase order and contract are common alongside Purchasing cards (P-Cards) and company credit cards. From these, only invoices, transactions, and P-cards provide data with standard dimensions like purchase orders in the P2P process. Invoice information is typically Accounts Payable (AP) data, including both invoice information and payment information. Master data for general ledgers (GL), cost centers, vendors, and other fields with descriptions to identification numbers is important information from the spend analysis point of view to help to model the dimensions. Names for vendors instead of just vendor numbers could help, especially in the vendor consolidation process. (Pandit & Marmanis, 2008; Chowdhary, et al., 2011) Examples of the features that are likely to be included in the data from the P2P process and invoice transactions can be found in Table 1.

Table 1 Examples of features in the P2P process and invoicing in accordance with Pandit and Marmanis (2008)

Vendor	Total amount	Invoice due
PO Number	Invoice number	Billed quantity
GL Account	Discounts	Billed amount
Cost center	Surcharges	
Item description	Total amount billed	
Quantity	Taxes	
Price	Invoice amount	
Payment terms	Invoice date	

Corporate P-Cards and credit cards have similar features to each other. Still, P-card transaction history is more structured and suitable for spend analysis purposes than credit card data. P-Card is built for purchasing purposes, and the transaction structure is built to be a suitable part of procurement and sourcing processes. Transaction data from corporate cards are structured and digital, thus easily available to be extracted as part of spend analysis. The issue with the credit card data is that all employees do not have a corporate credit card. They go through the process of purchasing by using their own credit card and submitting the receipts for verification. That process possibly affects the data quality since the employee usually does the expense report filing, leading to abnormalities in patterns of individual expense reimbursements discussed more in the next section. Both P-Cards and corporate credit cards are an important source of indirect spend data. (Pandit & Marmanis, 2008) Table 2 has an example of features of P-Card data that credit card data include.

Table 2 Example set of features in P-Card data (Pandit & Marmanis, 2008)

Account number	Vendor
User	Cost center
Transaction ID	GL account
Transaction date	Item description
Transaction amount	

Besides the P2P process, invoice transactions, AP master, and card data, other data sources can be included in the spend analysis. Receiving docks may have a system to track when shipments are delivered based on PO numbers. The receiving system can provide information about the delivered quantity. Third-party transportation companies can provide transaction details about transportation mode, location, and status. Other third-party data, like market research or supplier intelligence, enriches the data for the analysis even further. (Pandit & Marmanis, 2008)

2.3 Outliers

Outlier, anomaly, deviation, novelty, exception, or noise, all these terms describe data points in a data set that are somehow abnormal or significantly different from the majority of the data (Aggarwal, 2013; Hodge & Austin, 2004). The most suitable term for this occurrence in this thesis is an outlier. Barnett and Lewis (1987) define an outlier as “An outlying observation, or ‘outlier,’ is one that appears to deviate markedly from other members of the sample data set in which it occurs.” Aggrawal (2013) also brings up Hawkins (1980) formal definition of an outlier: “An outlier is an observation which deviates so much from the other observations as to arouse suspicions ta different mechanism generated it.” The term “observation” in the definitions refers to data points in a data set. Both are commonly referred definitions, but universally accepted one does not exist. Mehrotra et al. (2017) shortly conclude outliers being substantial variations from the norm. This definition is a more statistical way of viewing outliers and normality, which, in this case, refers to the statistical norm of the data set. In Figure 2, outliers are displayed and marked according to the definitions above. Blue marks represent the normal data points, and orange marks are potential outliers.

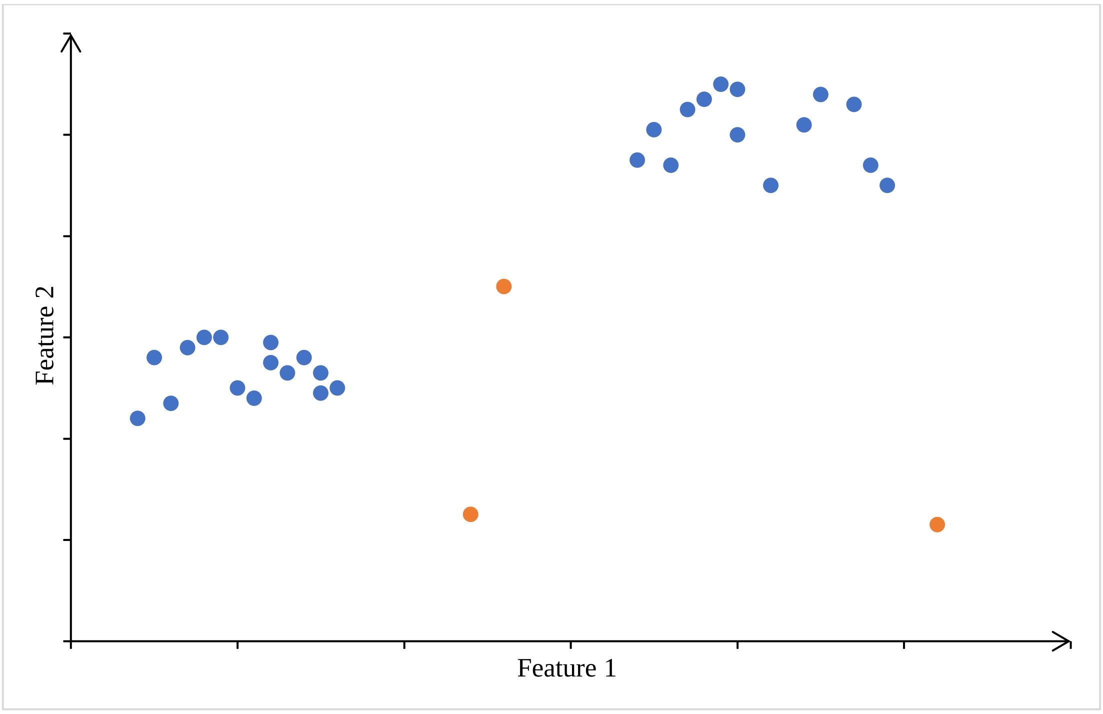

Figure 2 A simple artificial example data set with few potential outliers marked orange

Sridhar and Suman (2019) state that outlier is a value deviating from what is expected, but outlier detection can vary depending on the situation. Even though outliers are very often considered as errors, they may also include important information. When the data of a process behaves unusually, it leads to outliers, and the outlier gives information about the unusual characteristics within the process. (Aggarwal, 2013) There are multiple example applications where these unusual data points can be useful information instead of an error. Credit card fraud detection is one commonly presented use case for outlier detection, where the outliers are not removed as a part of data cleansing but as unusual data points that can be identified as fraud. Credit card fraud is an unauthorized usage of credit cards (Maimon & Rokach, 2005). This unauthorized use may show up as differing patterns like using the card in a geographically distant location, purchasing unusual items, or spending unusual amounts of money. Patterns like this in credit card transaction data are useful for banks or credit card providers to identify to minimize the clients' and the credit card providers' financial losses. (Maimon & Rokach, 2005; Aggarwal, 2013)

The other end of outliers also being useful when they are actual errors, and the application area defines the nature of the outliers. Outliers being errors are, for instance, typographical errors in the system data entry caused by human error. Once this type of outlier is detected, it can be corrected and restored as a normal data point. (Hodge & Austin, 2004) Depending on the data use case, an error type of outliers may cause incorrect results or wrong defined model in modeling and analysis. (Maimon & Rokach, 2005) Outliers like this are also good to verify if they are actual errors. An example of this is research on the human population. A small group of people seems to be very heavy or a group of people much taller than the others in the research. These data points can be typographical errors if humans do the data collection. The other option here is that these are natural occurrences that simply seem like an outlier. (Hodge & Austin, 2004) Overall, it is useful to detect and verify outliers from the data to ensure the modeling, analysis, and result correctness.

2.4 Outliers in spend analysis

Outliers in spend can be data input errors, errors in conversions, and confusion in measurement units. Spend data outliers can also indicate fraudulent behavior in the procurement processes (Phillips & Lanclos, 2014). The utilization of outlier detection methods could help the organization find and evaluate abnormal transactions to quickly resolve them, whether it is a fraud or merely a human error (Ball, 2019).

Phillips and Lanclos (2014) identified activities that could potentially provide an opportunity for fraud in procurement processes. One could direct a contract to a favored vendor in the procurement planning phase instead of choosing the vendor with the most suitable offering. Deficient, fraudulent, or biased market research may restrict the competition leading to a situation where the material may be purchased overpriced. During the invoicing process, the vendor can consciously mischarge the purchaser, which is accidentally not noticed. (Phillips & Lanclos, 2014) Even though Phillips and Lanclos (2014) research acknowledge only the U.S Government procurement fraud scheme, the actions mentioned above can also apply to private sector procurement processes.

Errors in the data impact data quality. Pandit and Marmanis (2008) have recognized different types of errors that spend data might include. These errors can be caused by misspellings, missing information, or inconsistencies in the attribute values. The errors are divided into two types of errors – single-source and multisource. In single-source errors, the errors come from one source, and the cause of it does not have any connections to other data sources. Multisource errors are caused by the many data sources that the data is received from. Combining the data sources might cause errors in the data. From the price outlier point of view, the following errors are potential to produce such errors and then outliers per Pandit and Marmanis (2008):

- Single-source errors:
  - Misspelling errors, like spelling “Supplier” incorrectly as “Suplier.”
  - Terms referring to the same thing phrased or formulated differently like status terms “completed” and “100%.”
  - The discrepancy between the values and schema, e.g., the date can be in an invalid format for a column having a date as a data type causing incorrect values
- Multisource errors:
  - Different units of measurement formats between source systems, e.g., metric system versus the imperial system.
  - Different formats of values, for example, marking decimals with point or comma.

All the listed are data point-level errors meaning that the errors are specific to a single data point and not the whole schema. Outliers caused by these kinds of errors in the data are important to detect and possibly eliminate to ensure the data quality for further use. (Pandit & Marmanis, 2008)

Outliers or abnormal transactions in the procurement data can also be something other than fraud or error. A drastic change in the price, for example, may seem like a mistake at first glance but has a reasonable explanation behind it. Outlier detection could also bring the abnormalities up for management to review if actions like changing the contract or vendor are required (Ball, 2019). Outlier detection in procurement does not necessarily have to aim to cleanse the data. It could also support the whole procurement process in terms of taking corrective actions to obtain savings. Another example is price opportunities. The price of material could drastically drop. In case that is not an error, the procurer could utilize the information and opportunity to acquire even more savings.

Outliers caused by errors are essential to detect and possibly eliminate to ensure the data quality for further use. The other point of view of the outliers is fraud detection and contract compliance, meaning examining the procurement process to detect violations within each procurement process activity. In addition to these, outliers can provide insights about actions that would require attention and, without outlier detection, could be left unnoticed.

2.5 Outlier Detection

Different outlier detection algorithms have been developed over time using different kinds of characterized approaches. The distance-based approach detects the points that are further away from the other points and marks them outliers. The density-based approach considers outliers being lower density regions in the data set. In the rank-based approach, outliers are data points that nearest neighbours have others as their nearest neighbours. (Mehrotra, et al., 2017) The nature of the data defines the fundamental approach of outlier detection on top of the distance, density, and rank-based approaches. The supervised approach is when there is a training data set with classification with existing labels. It is used to train the algorithm to recognize and classify the outliers as outliers. In a semi-supervised way, normal or abnormal behavior is taught with partial labels in the data. The unsupervised nature of the data means that there are no labels to train the algorithm. Hence it does not know what kind of criteria defines an outlier or a normal data point. (Hodge & Austin, 2004; Mehrotra, et al., 2017; Li & Fu, 2017)

The data in outlier detection is usually partially labeled or unlabeled, so the approach is an un- or semi-supervised algorithm. As already mentioned, semi-supervised outlier detection involves partially labeled data. The ideal would be that the model learns the characteristics of normal data. Meaning that the model would be able to detect outliers as they differ from the regular data points. (Sridhar & Suman, 2019) The semi-supervised approach is suitable for both static and dynamic data due to learning only the normality. (Hodge & Austin, 2004)

Labeling of the data requires manual work. If the amount of data is large, that is not very often done if the task to be performed is not a common classification. In many outlier detection cases, the model is unsupervised, where the data is unlabeled. The distances and comparisons are made based on the entire data set (Mehrotra, et al., 2017). The model is trained without actual training data having labels. The model’s purpose is to recognize the most distant point or points and, as an output, give them as a potential outlier. (Hodge & Austin, 2004) In this thesis, all the models are utilized and researched as unsupervised, even though most of the next algorithms can also be implemented as supervised and semi-supervised. Against the Hodge and Austin (2004) approach, the data set is split into training and test sets.

2.5.1 One-class Support Vector Machine

One-class Support Vector Machine (OCSVM or OSVM) is a variant from Support Vector Machines (SVM) for outlier detection purposes. SVM has been performing at least as well as the competing methods in terms of generalization (Burges, 1998). OCSVM is basically a semi-supervised model variant from SVMs, but it can also be used for unlabeled data in unsupervised cases. (Amer, et al., 2013; Shin, et al., 2005) The basic theory behind Support Vector Machines is the statistical learning theory findings by Vapnik (1995) and was initially developed for pattern recognition. Based on Vapnik’s theory, Schölkopf et al. (2000) proposed an enhanced SVM algorithm for regression and classification. Slightly later, Schölkopf et al. (2001) proposed an extension for their previous SVM algorithm to the case of unlabeled data called a single class SVM.

Compared to basic SVM, One-Class SVM algorithms goal is to find and learn an optimal hyperplane or set of hyperplanes as a boundary separating the data from the origin. A small part of the data points can lie outside this hyperplane, and those are considered outliers, which is also the main difference between the traditional SVM and OCSVM. The basic SVM for classification purposes finds and learns a hyperplane between the two classes instead of origin and the whole data set. (Amer, et al., 2013) The One-Class SVM algorithm’s output is a binary value, 1 representing inliers and -1 marking outliers. An outlier score is also possible output that gives more information about the “outlier-ness” measure of the data point instead of a binary value telling if the point is an outlier or not. (Amer, et al., 2013; Sridhar & Suman, 2019) Figure 3 is an artificial example data set with a hyperplane separating outliers (-1) and inliers (+1) on each side of the hyperplane.

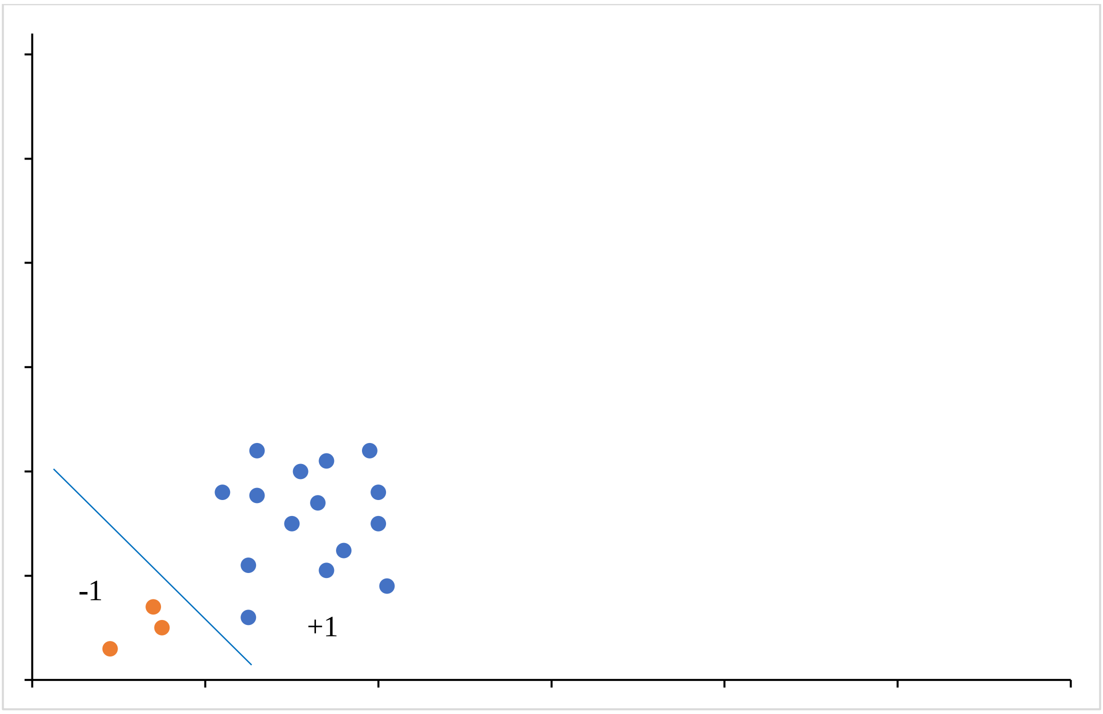

Figure 3 One-Class Support Vector Machine illustration where the blue line represents the learned decision boundary

The key idea behind the OCSVM algorithm is that abnormal data points are expected to contribute less to the decision boundary than normal data points. The Gaussian kernel guarantees this boundary. All the data points in the kernel space lie in the same quadrant, which can be concluded from all the kernel entries being non-negative. Based on the above, the Gaussian kernel deals well with any dataset. (Amer, et al., 2013)

One-Class SVM estimates function f that is positive on S and negative on the complement $\overline{S}$ and can be presented as follows:

$$f(x) = \begin{cases} +1 & if\ x \in S \\ -1 & if\ x \in \overline{S} \end{cases} \tag{1}$$

Where S is an estimate of a subset of the feature space bounded by a priori specified value $v \in (0,1)$ that a test point from the dataset's probability distribution P lies inside S. $\overline{S}$ is the complement of S. (Manevitz & Yousef, 2001)

Let x_1, x_2, ..., x_l be the training data points that belong to the one class X_t that is a subset of vector space $\mathbb{R}^N$ (Manevitz & Yousef, 2001). According to Amer et al. (2013), $g(\cdot)$ can be defined so:

$$g(x) = w^T \varphi(x) - \rho, \tag{2}$$

where w is a perpendicular vector to the decision boundary, x is a single data point, ρ represents the bias term, and $\varphi(\cdot)$ is the implicit transformation function defined by the kernel for projecting the data into a higher-dimensional space. The decision function of OCSVM returning positive values for inliers and otherwise negative is defined:

$$f(x) = sgn(g(x)), \tag{3}$$

where $g(x)$ is defined in Equation (2) (Amer, et al., 2013) and $sgn(\cdot)$ is the signum function. The following quadric programming problem needs to be solved to separate the data set from the origin:

$$\begin{gathered} min_{w,\xi,\rho} \frac{\|w\|^2}{2} - \rho + \frac{1}{vn} \sum_{i=1}^{l} \xi_i \\ subject\ to\colon\ w^T \varphi(x_i) \geq \rho - \xi_i, \qquad i = 1,2,\dots,l \quad \xi_i \geq 0, \end{gathered} \tag{4}$$

where ξ_i is the slack variable for point i allowing it to lie on the opposite side of the boundary, l is the size of the training data set, and v is the regularization parameter (Amer, et al., 2013; Manevitz & Yousef, 2001).

By defining the decision boundary as

$$g(x) = 0 \tag{5}$$

the distance of any chosen data point to the decision boundary can be computed as follows: (Amer, et al., 2013):

$$dist(x) = \frac{|g(x)|}{||w||} \tag{6}$$

Hence the algorithm is attempting to maximize the distance $\frac{\rho}{||w||}$ when the origin is added to the equation, which can also be presented as minimization of $\frac{||w||^2}{2} - \rho$ (Amer, et al., 2013).

One-Class SVM algorithm is a popular unsupervised outlier detection technique. OCSVM can model any pattern of normal behavior being either linear or non-linear well. SVM based models are appealing options due to the good generalization if parameters are configured correctly. (Erfani, et al., 2016) In terms of generalization errors, Amer et al. (Amer, et al., 2013) mention that SVMs have been performing in outlier detection at least as well as other algorithms. It is also argued that SVMs can be used for dimensionality reduction purposes, making it an attractive option for outlier detection as it could potentially overcome the so-called "curse of dimensionality" (Amer, et al., 2013).

2.5.2 PCA-based outlier detection

Principal Component Analysis (PCA) is a method used for dimensionality reduction purposes. Usually, these principal components are determined before implementing a machine-learning algorithm to reduce the number of dimensions. Shyu et al. (2003) propose a PCA-based approach for outlier detection. Principal Component Analysis is already an unsupervised algorithm, so the outlier detection is also practical for a case with unlabeled data.

In Principal Component Analysis, the data is transformed from correlated data into uncorrelated variables – principal components (PCs). (Ghorbel, et al., 2015) The principal components are linear combinations of random variables, and these variables must have certain properties. As already mentioned, PCs must be uncorrelated, the first PC's variance must be the highest, the second one has the second-highest, and so on. The combined total variation of the principal components needs to match the total variation of the original variables. (Shyu, et al., 2003) PCA-based outlier detection approach assumes outliers to be different from the normal data points. The detection algorithm is built so that the normal group's correlation matrix is generated, and PCA is used on that. As this approach is unsupervised, it is not clear that the normal group data set would contain only normal data points. The initial expectation is that the number of outliers or suspicious data points is much smaller than normal data points. PCA method also assumes that outliers do not have the same correlation structure as the normal data points. (Shyu, et al., 2003)

Principal Component Analysis calculates the eigenvalue of the covariance or the correlation matrix that usually give different principal components. In this definition, the covariance matrix is used. In the principal component analysis, let *n* represent the number of data points in the data set having *b* variables. The *i*-th data point is a row of the data set denoted by $X_i = [x_{i1} \ldots x_{ib}]$ where x_{ij} is the *i*th data point of X_j. Let the covariance matrix be denoted as $b \times b$ of the data set by Σ. Here the (*i,j*)th entry is the covariance between the *i*th and *j*th dimensions and diagonalized as

$$\Sigma = P \cdot D \ \cdot \ P^T, \tag{7}$$

where *D* is a diagonal matrix and *P* is an orthonormal matrix having corresponding columns to the orthonormal eigenvectors of Σ. These vectors provide the directions of the axes that the data should be projected. (Aggarwal, 2013)

The way of modeling the "outlier-ness" or abnormality level of a data point is done using eigenvalue to calculate a normalized distance between the data point and the centroid along each principal component's direction. Let $\overline{e_j}$ be the *j*th eigenvector, and along that direction, an eigenvalue (variance) of λ_j. (Aggarwal, 2013) To the centroid $\bar{\mu}$, outlier score of a single data point *x* is given by the following equation (Aggarwal, 2013):

$$Score(x) = \sum_{j=1}^{b} \frac{\left|(x - \bar{\mu}) \cdot \overline{e_j}\right|^2}{\lambda_j} \tag{8}$$

In the research of Shyu et al. (2003), the outcome is that the PCA-based method surpasses more traditional outlier detection methods like the k-nearest neighbour and Local Outlier Factor (LOF). Other advantages of the PCA-based method are that it does not assume the data to follow any distribution. As the PCA method's one of the original use cases is dimensionality reduction, it performs well in high-dimensional cases. (Shyu, et al., 2003) PCA-based outlier detection methods have also been expanded as non-linear dimensionality reduction is starting to find its application in the outlier detection field. An example of these is kernel principal component analysis (KPCA). (Ghorbel, et al., 2015) However, the scope of this thesis remains in the basic approach to PCA-based outlier detection.

2.5.3 Isolation Forest

Isolation Forest also referred to as iForest, is an outlier detection method where the abnormal data points are isolated instead of learning and characterizing the normal behaviour of the data. Isolation forest is an ensemble method, which means combining a collection of some methods. In the case of Isolation Forest, the ensemble consists of isolation trees (iTrees). Isolation Forest assumes that the outliers being rare and distant from the centers of the normal clusters. In other words, outliers represent the minority in the data set, and the values are very different from the normal data points. (Liu, et al., 2008) This method can be considered partially density-based and distance-based without actual calculation of distances or densities.

Liu et al. (2008) define isolation as "separating an instance from the rest of the instances," in which the term "instance" refers to a data point. Liu et al. (2008) also define outliers as rare and distant from the other data points, and they are easier to be separated from those other data points. The basic idea behind Isolation Forest is that it is a collection of tree structures that are used to partition the data set recursively. These structures are collected in each iteration where cuts parallel to axes and randomly selected are used to isolate each data point from the others. Partitioning of an outlier has fewer partitions than the data points closer to the others. Thus, the path in the case of outliers is much shorter, which means that different values distant from the others are more likely to get isolated earlier in the partitioning process. (Liu, et al., 2008) Figure 4 shows the cuts of one data point (marked orange) to be isolated from the group of blue data points.

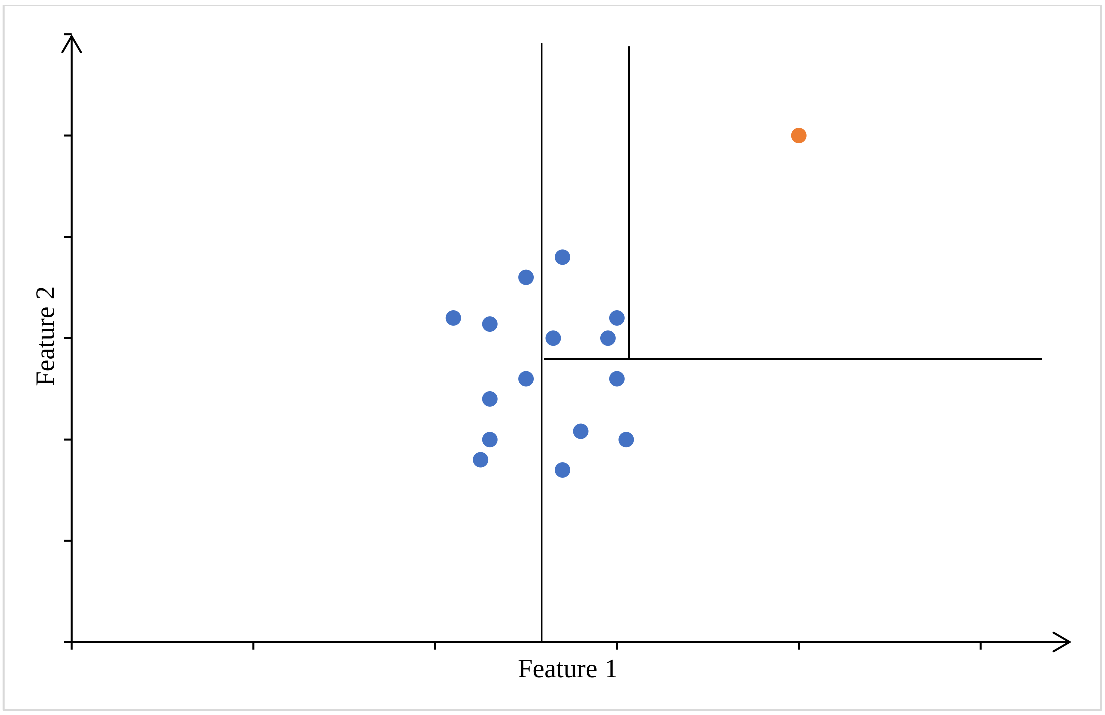

Figure 4 Example of isolating potential outlier (orange) from the rest of the data set (Hariri, et al., 2019)

Let T be one node of an iTree, which can be either an external node without children or an internal node with one test node and precisely two child nodes – T_m and T_r. A test contains attribute q and splits value p where $q < p$ dividing the data points into T_m and T_r. A given sample data set X having n data points from d-variate distribution is recursively divided by selecting q and p randomly until one of the following is reached:

(a) the tree reaches the height limit
(b) $| X | = 1$, where $| X |$ is cardinality of X
(c) all data points of X have the same value.

The iTree is a binary tree where each node has exactly zero or two daughter nodes. All data points are also assumed to be distinct, and each one is isolated into an external node when the Isolation Tree is fully grown. (Liu, et al., 2008)

For outlier detection, anomaly ranking or score is required. Path length $h(x)$ of a data point x is not comparable itself as the maximum height of the iTree grows in the order of n, and the average height grows in the order of $log\ n$. The structure of an iTree is equivalent to a Binary

Search Tree (BST). Therefore, the estimation of average path length is calculated based on an unsuccessful BST search. For data set of n data points, the average path length of an unsuccessful BST search is:

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{9}$$

Where $H(n$-1) is estimated harmonic number $ln(n$-1) + 0.5772156649 (Euler's constant). (Liu, et al., 2008) To normalize $h(x)$ for given n, c(n) is used, and the outlier or anomaly score s of a data point x is

$$s(x,n) = 2^{-\frac{avg(h(x))}{c(n)}} \tag{10}$$

where $avg(h(x))$ is the average of $h(x)$ and leading to the following scores

- when $avg(h(x)) \rightarrow$ c(n), $s \rightarrow 0.5$;
- when $avg(h(x)) \rightarrow 0$, $s \rightarrow 1$;
- $avg(h(x)) \rightarrow n$ - 1, $s \rightarrow 0$

where s is monotonic to $h(x)$. (Liu, et al., 2008)

From the score s the following can be concluded: (i) any data points returning s very close to 1 are definitely outliers, (ii) data points having s much smaller than 0.5 are relatively safe to be seen as inliers, and (iii) in case of all data points returning s approximately 0.5 the sample data set does not include any distinct outliers. (Liu, et al., 2008)

Ding and Fei (2013) have studied iForest, and they have listed a few advantages of the Isolation Forest method. The isolation Forest method does not utilize any distance or density-based calculations, reducing the method's computational cost. The complexity of iForest grows linearly, which means that the memory requirement of using the method is low. The last advantage mentioned is that the method's ensemble nature turns weaker individual iTrees into a robust algorithm when combined and used together. Based on these advantages, the Isolation Forest seems a good algorithm from the performance point of view. (Ding & Fei, 2013)

2.5.4 Local Outlier Factor

Local Outlier Factor (LOF) is an outlier detection method introduced by Breunig et al. (2000). It is one of the earliest outlier detection methods. The general idea is to determine how local the data point is with respect to the surrounding neighbour data points. It is a density-based method where the outliers are considered to be distant from the restricted neighbourhood. (Breunig, et al., 2000; Kotu & Deshpande, 2016; Celebi & Aydin, 2016)

Local Outlier Factor is calculated for each data point, and the value indicates the "outlier-ness" of the data point. The key technique in LOF is the relative density score of the data point *x*, which is the measurement of the data point's local deviation with respect to its nearest neighbours. Relative density is in concept level calculated as follows:

$$Relative\ density\ of\ x = \frac{density\ of\ x}{average\ density\ of\ all\ instances\ in\ the\ neighbourhood} \quad (11)$$

(Kotu & Deshpande, 2016) For outliers, this calculated density is lower than the density of the neighbourhood. In other words, the space around an outlier is sparser compared to data points that have a lower distance to their neighbours. (Mehrotra, et al., 2017; Celebi & Aydin, 2016; Kotu & Deshpande, 2016)

According to Schubert et al. (2012 cited Breunig, et al., 2000), LOF uses the k-nearest neighbours (kNNs) for context and reference of data point *x*, using a density model called local reachability density (lrd). Let *o* represent a neighbour data point, *M* a parameter value for the minimum number of neighbour data points, and $N_M(x)$ parameter value for the neighbourhood's volume where *o* belongs as well. Local reachability density for a single data point *x* is defined as follows:

$$lrd_M(x) = 1/\frac{\sum_{p \in N_M(x)} reachability\text{ - }dist_M(x,o)}{|N_M(x)|} \quad (12)$$

Where $reachability\text{-}dist_M(x,o)$ is defined as:

$$reachability\text{-}dist_M(x,o) = max\{M\text{-}dist(o), dist(x,o)\} \tag{13}$$

In Equation (13), *dist* is a distance between *x* and *o,* and *M*-dist(*o*) is the distance between *o* and the *M*th nearest neighbour of *o*. The following comparison calculates the final score:

$$LOF_M(x) = \frac{\sum_{p \in N_M(x)} \frac{lrd_M(o)}{lrd_M(x)}}{|N_M(x)|} \tag{14}$$

(Breunig, et al., 2000; Schubert, et al., 2012)

The Local Outlier Factor method has been successful in outlier detection cases where the data sets density has been non-uniform in multiple different fields. Varying density has been an issue with the simplest density-based clustering outlier detection methods, and LOF was successfully proposed to overcome this challenge. However, in the LOF method, the local outlier factor is calculated for each data point, decreasing the efficiency and calculation time of this algorithm significantly when the number of data points increases. (Kotu & Deshpande, 2016; Su, et al., 2019) Researchers have been extending the basic LOF model to increase efficiency and accuracy (Schubert, et al., 2012). In this thesis, the method used is LOF in its original format that Breunig et al. (2000) proposed. In the research of Breunig et al. (2000), LOF turned out efficient and practical for large data sets.

2.6 Summary of the outlier detection methods

A comparison of the outlier detection algorithms presented in the previous section can be found in Table 3. It includes advantages and disadvantages listed based on the previous section’s literature review.

Table 3 Summary of the outlier detection algorithms

	One-Class SVM	**PCA**	**Isolation Forest**	**LOF**
Basic concept	Finding a hyperplane separating the data from the origin. Data points between the hyperplane and origin are labeled as outliers.	The normal group's correlation matrix is used to calculate the principal components. The reconstruction error is used to determine if the data point is an outlier.	Outliers being rare and distant, are easier to isolate from the other data points when partitioning the data set recursively.	The relative density around the data point measures the data point's local deviation with respect to its k-nearest neighbours.
Advantages	- Good generalization level - Finds patterns well in both linear and non-linear cases	- Goes well with high dimensional cases - The data is not expected to follow any distribution	-No distance or density calculations required reducing the computational requirements -Ensemble method quality turns individual models into a more robust algorithm	- Good performance in cases where the density of the data set varies - Suitable for non-uniform density cases
Disadvantages	- Rather high computational requirements (Amer, et al., 2013) - Poor performance with more complex data sets (Amer, et al., 2013)	- The linear nature of the model is not suitable for all practical cases (Hodge & Austin, 2004)	- Does not consider the relative importance of the features (Chen & Wu, 2018)	- Local density might not be enough for more complex data sets (Gao, et al., 2011) - The quality of the results depends highly on the determination of the number of the k-nearest neighbours (Gao, et al., 2011)

Based on the literature review and the above summary (Table 3), one cannot determine which would be the best outlier detection method for spend analysis purposes.

2.7 Evaluation of unsupervised learning methods

Evaluation and ranking must be performed somehow to determine the best alternative algorithm for outlier detection. In unlabeled data and unsupervised learning, the evaluation, comparison, and benchmarking are more complicated than when the data is labeled. Traditional criteria like Receiver Operating Characteristic (ROC) curve cannot be used (Goix, 2016a). Goix et al. (2015) introduce Excess-Mass (EM) and Mass-Volume (MV) curves to get the information on how well each method performs in unsupervised outlier detection so that the best one of the alternatives can be chosen.

The Mass-Volume calculation is an extension of the ROC curve, where partial preorder is produced on the scoring functions, which means that the selection of optimal elements defines the set of scoring functions having a minimum MV curve everywhere. (Clémençon & Thomas, 2018) The excess-Mass curve is calculated similarly to the Mass-Volume curve, but the optimal elements are determined by mass maximization. Mass-Volume and Excess-Mass curves do not work so well with high-dimensional data. Goix (2016a) mentions that MV and EM performance decreases drastically if the number of dimensions exceeds eight. However, a way to come around this issue has been proposed. This approach works so that the number of features to calculate MV and EM is chosen randomly, and the performance score is obtained by averaging the different draws of the features. Interpretation of these values goes as follows: best-performing algorithms have the lowest EM and highest MV score. (Goix, 2016a)

In unsupervised cases, the distribution F of the normal data set without outliers is unknown, which means that the MV and EM curves need to be estimated. The decision function is an output of an outlier detection algorithm, which includes the outlier scores of each data point before the scores are turned into binary values 1 and 0 or +1 and -1. Let $e \in E$ be a decision function, $\mathbf{X}_1, \dots, \mathbf{X}_z$ independent and identically distributed random variables with cumulative distribution function F representing the normal behavior and set $\mathbb{P}_z$ to be:

$$\mathbb{P}_z(e \geq t) = \frac{1}{z}\sum_{i=1}^{z} \mathbb{1}_{E(\boldsymbol{X}_i)\geq t}, \tag{15}$$

where $\mathbb{P}_z$ is the probability of an event z and t is a discretized version of the Lagrangian multiplier. In Equation (15), $\mathbb{1}_E$ indicator function of E representing the membership of $\mathbf{X}_z$ in E. (Goix, 2016a; Goix, 2016b).

Then the MV and EM curves are defined as follows:

$$\widehat{MV}_s(\alpha) = \inf_{u\geq 0}\{\text{Leb}(e \geq u)\; s.t.\, \mathbb{P}_n(e \geq u) \geq \alpha\} \tag{16}$$

$$\widehat{EM}_s(t) = \sup_{u\geq 0} \mathbb{P}_n(e \geq u) - t\text{Leb}(e \geq u) \tag{17}$$

for any $\alpha \in (0,1)$ and $t > 0$ where the volume $\text{Leb}(e \geq u)$ is the Lebesgue integral, and u is a unique element in the decision function of the sample data set X. (Goix, 2016a). $\text{Leb}(e \geq u)$ is estimated with Monte-Carlo approximation in case of fewer than 8 dimensions in the data set. Empirically, the performance criteria based on EM and MV can be computed using the following equations:

$$\hat{C}^{EM}(e) = \left\|\widehat{EM}_e\right\|_{L^1(I^{EM})} \qquad I^{EM} = \left[0, \widehat{EM}^{-1}(0.9)\right] \tag{18}$$

$$\hat{C}^{MV}(e) = \left\|\widehat{MV}_e\right\|_{L^1(I^{MV})} \qquad I^{MV} = [0.9,0.999] \tag{19}$$

(Goix, 2016a). As the accuracy is evaluated, one natural interval of I^{MV} could be from 0.9 to 1. MV deviates from 1 when the support is infinite. Thus $I^{MV} = [0.9,0.999]$ is chosen. Similarly with $I^{EM} = [0, \text{EM}^{-1}(0.9)]$, where $\text{EM}^{-1}{}_e(0.9) = \inf\{t \geq 0, \text{EM}_e(t) \leq 0.9\}$, as $\text{EM}_e(0)$ is equal to 1. (Goix, 2016b)

In the existing literature, MV and EM as an evaluation criterion have also performed well in the high dimensional cases. Compared to ROC and Precision-Recall (PR) curve, Mass-Volume, and Excess-Mass curves have agreed with the supervised criterion in approximately 80% of the cases. In Goix's (2016a) research, 12 different data sets were used to rank iForest, OCSVM, and LOF algorithms alongside ROC and PR. The study concluded that MV and EM could rank unsupervised outlier detection methods without labels in the data to some extent. In the cases where ROC and PR disagree with each other, MV's and EM's performance is not giving as promising results either. (Goix, 2016a)

3 RESEARCH METHODS

This chapter introduces the data, research methodology, and the process of the empirical study.

3.1 Data set preprocessing and exploratory data analysis

The dataset used in this research is an actual transactional dataset from an anonymous Sievo client. Due to information security reasons, no information about the client company is revealed at any point. The dataset is exported from the Sievo database using Microsoft SQL Server Manager and query in SQL language into Comma-separated values (CSV) format file. Data rows with spend amount and quantity above zero and year 2019 were chosen from the database. In addition to this, transactions that reverse each other has been eliminated. These kinds of counter transactions appear when the operator has made a mistake in submitting a purchasing document to the system. This mistake can be corrected by first submitting a reversing document with the same spend amount in the opposite direction. Only after that, the user would submit a new, correct transaction. For example, the user may have created a document with an incorrect amount of 10. A new document with the amount of -10 would be created to cancel the first document. In Sievo, these canceled double transactions are detected by comparing the transaction lines and amounts.

The file containing the data is in a virtual machine hosted by Sievo, imported in Python for further analysis. The whole undivided dataset in the CSV-file imported in Python contains 51 columns and 2 002 387 rows. All the features contain identification numbers unique within the Sievo database and determined separately by Sievo for each transaction row. The data set does not contain original data values from the client's raw source data, which means that all the data are anonymized. It is impossible to determine the actual values of the data's features. Year information, spend amount, and quantity are exceptions as they contain the transactions' actual values. Since all the other values are anonymized, one cannot determine the actual vendor company, plant, material, or document description of the spend transaction. Only comparisons like differences between the features within the examined rows can be examined.

The data are prepared for exploratory analysis and algorithm implementation. The whole script of the data preparation and algorithm implementation is done in Python language and within a Jupyter notebook. The preparation starts with importing the CSV-file containing the data into a dataframe object using a Python software library called Pandas. The dataframe format makes manipulating and handling the dataset more convenient. With the imported dataframe, the preparation continues with choosing a subset of the data that has been classified to one category already in the Sievo application. The category was chosen based on the number of spend transaction rows. The category chosen included the largest number of rows at the exporting point of time.

Each purchase's price is calculated by dividing the spend amount by the purchased quantity and saved as a new column in the dataframe. After calculating the price, four of the most important features are chosen for the dataset alongside with price for further analysis. Based on the previous chapter's literature review, vendor information was first chosen as an essential feature for detecting price outliers. Vendors are the factor in the sourcing process that affects the price as they are the party defining the price and invoicing based on that. Vendors are also affecting the possible fraudulent invoicing. The other two features chosen for further analysis are material and material group information. Material has a high impact on defining the price. Material group information gives slightly higher-level information about the material and may also be useful when looking for price outliers in transactional data. Material information is chosen because larger deviations within one material (or material group) might signal outliers. At the end of the data preprocessing, the number of features in the dataset is 4. The number of rows is 44 219.

The data set's basic shape can be seen in Figure 5, where the scatter plots show the shape of the data and histograms in the diagonal, showing the shapes of value counts of each feature. Even though the data is in scatter plots, the other variables except price are categorical. Since the categorical values are in numerical identification number format, the values can be visualized in a scatter plot. Price as a numerical value is also easy to visualize against all the other features.

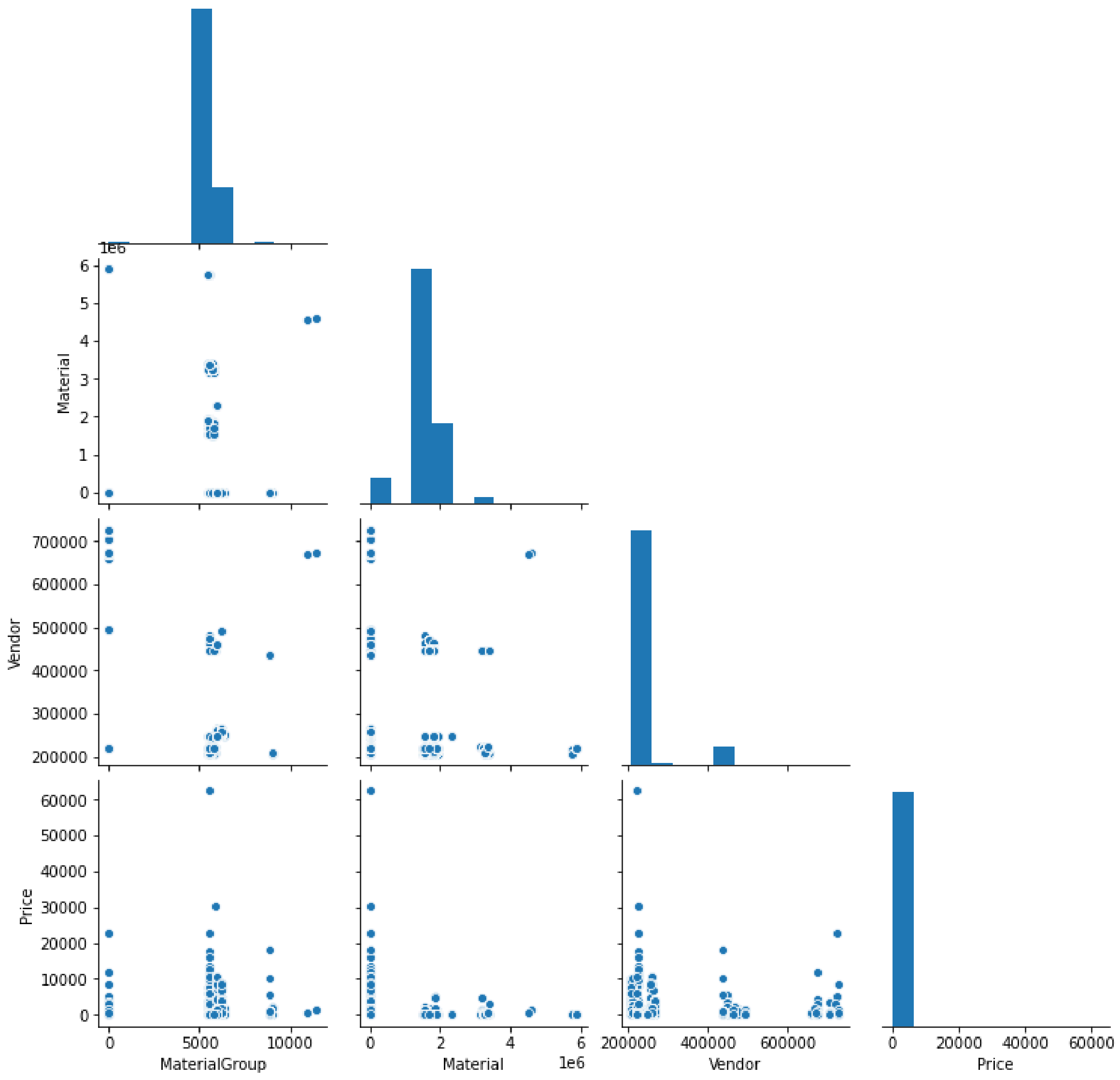

Figure 5 Basic shape of the data set in a scatter plot matrix

There are a few clearer clusters in the categorical graphs that do not include the price. In the vendor-material scatter plot, there are some clusters in the middle of the graph. Those clusters tell that most probably there are specific vendors that the company uses to purchase specific materials. Price seems to have few values that are significantly different from the other data points. As 94.4% of the data points have price value under 500, Figure 6 shows the same scatter plot matrix but with a subset of the data with data points having price less than 500. The shape of the price against the other features can be seen more clearly. There are widely different prices within one material group, and the same applies to materials and vendors. There are no clear price clusters in any of the features. The prices seem to be distributed between specific material groups and somewhat clearly between certain materials and vendors. The previously mentioned is visible in the scatter plots with price as the data points are distributed vertically along specific identification values.

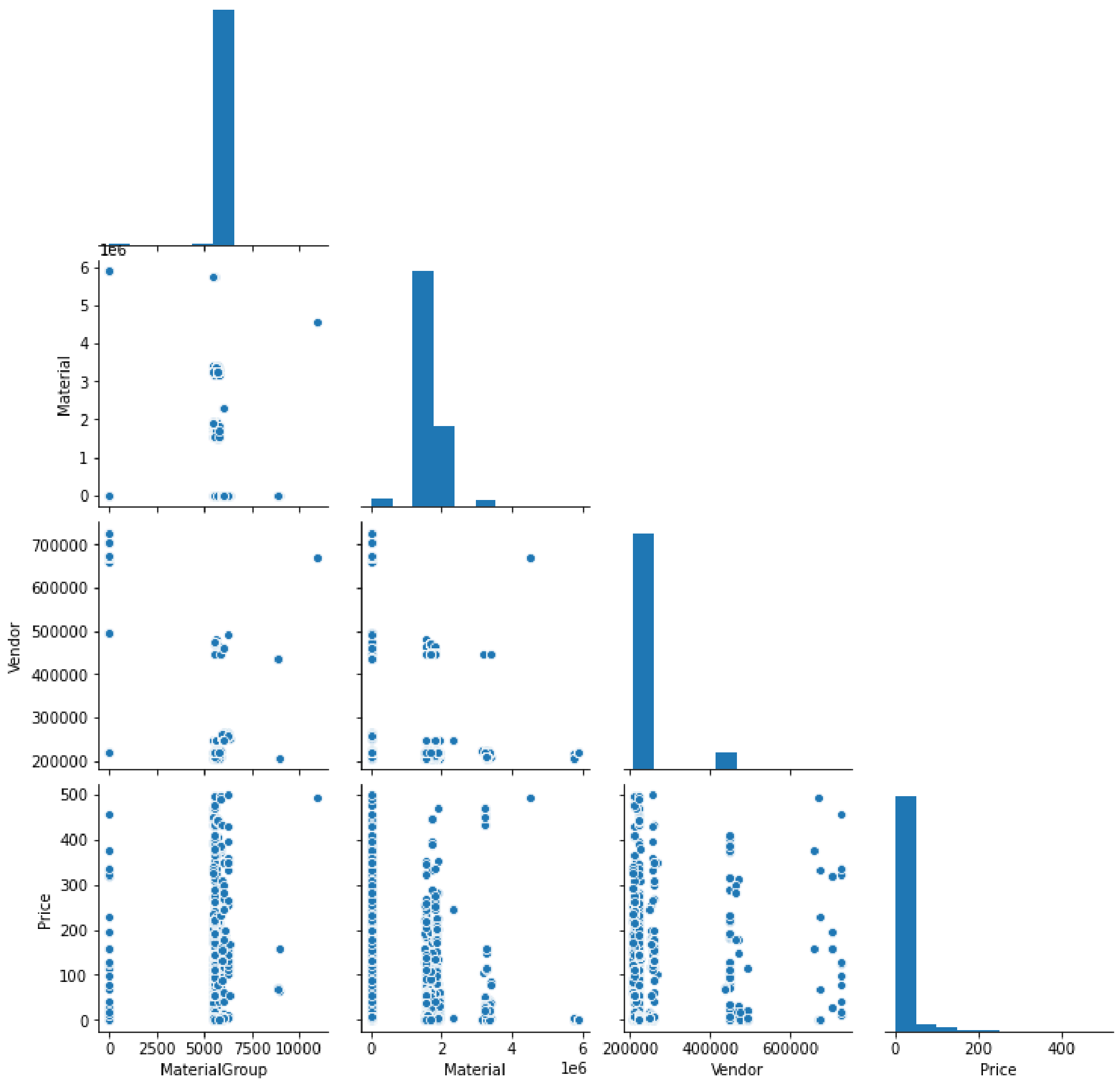

Figure 6 Scatter plot matrix of the data with price value less than 500

Table 4 shows the statistical summary of the price feature in the dataset. The difference between the 75th percentile and the maximum value is significantly large. From the statistical point of view, this could indicate the existence of outliers with extreme price values in the dataset. High standard deviation confirms that the price values are spread out rather widely.

Table 4 Statistical summary of price

Count	44 219
Mean	175.0313
Standard deviation	890.2701
Min	0.000071
25th percentile	0.0923
50th percentile	0.5005
75th percentile	4.3052
Max	62 396.2773

Within the 44 219 rows, there are not as many unique categorical values in the dataset. Table 5 shows a summary of the value counts of other variables than price.

Table 5 Summary of value counts of categorical variables

	Material group	**Material**	**Vendor**
Maximum count	10 306	3 323	6 824
Minimum count	1	1	1
Count of unique values	85	935	94

Based on the exploratory data analysis, there are potential outliers in the dataset. The scatter plot matrices (Figure 5 and Figure 6) show that some data points are relatively isolated from the other data points. The statistical summary of the price (Table 4) shows that price outliers potentially exist.

For comparison purposes, a threshold value for the proportion of outliers in the data set is needed. The case company utilizes a statistical method for that. Briefly explained, it calculates the median prices of material and supplier combinations. Upper and lower limits to the median price are determined, and all material supplier combinations with a price outside these limits are labeled as outliers. This method resembles a statistical method called Interquartile Range (IQR). From this particular data set described above, the statistical method determined approximately 1% to be outliers. However, a human expert has not gone through these outliers, and they are not validated. The 1% threshold value only gives a direction for the performance analysis of the algorithms.

3.2 Methodology and implementation of the algorithms

As already mentioned in the previous section, this research's execution was done in Jupyter notebook with Python. In Python, lots of ready-made packages are available. Machine Learning library Scikit-learn created by Pedregosa et al. (2011) is used for most of this research. Python Outlier Detection (PyOD) is another Python library for outlier detection built by Zhao et al. (2019). These libraries include the outlier detection algorithms introduced in the previous chapter. To evaluate the algorithms' performance for the benchmarking, Excess-Mass, and Mass-Volume measures by Goix (2016a) are used. The author provides a ready-to-use function for these in the GitHub repository. In addition to EM and MV measures and curves, each algorithm's execution time is measured with Python module time. The execution time measurement works so that the timer is started before the predicting function is called. The timer stops when the algorithm has calculated the labels for the test data points. The execution time is the difference between the start and end of the timer in seconds.

3.2.1 Parameter tuning

Parameter tuning is implemented by utilizing a for loop and a dataframe, including all possible combinations of chosen parameters and values. First, the most critical hyperparameters are chosen, and for those parameters, the values tried in the tuning loop are determined. The dataframe includes all the possible combinations of these values by calculating the Cartesian Product from the value lists given. Each row of this combination dataframe is an input to initialize the parameter tuning model in a loop. In each iteration loop, the model is taught with the training data. In the same iteration round, the decision function of the test data is calculated to get the performance measure Excess-Mass. EM is calculated within each iteration of the loop and saved to a list for analyzing the initialized algorithm's performance. Most optimal parameters for the model are found by finding the hyperparameter values with the highest resulting EM measure.

Hyperparameters tested for One-Class SVM are kernel type, gamma, and degree if kernel type is polynomial. Kernel type is used to precompute the kernel matrix, which determines the hyperplane between outliers and inliers. Gamma is the kernel coefficient for all the different kernel types. It determines how far the influence of a single data point reaches in training. High values of gamma mean that the influence reach is close, and for low values, the opposite. Gamma value "auto" is one divided by the number of features in the data. Value "scale" divides one by multiplying the number of features by the data set's variance. The degree is determined for the polynomial kernel type only, and all the other kernels ignore the degree parameter. The table, including the possible combinations, is built as short as possible to keep the algorithm's running time in control. The polynomial kernel type has eight different combinations because of the degree changes. The other two kernels have just two options as the degree does not affect the calculation. Table 6 shows each hyperparameter's possible values and the optimal combination bolded, giving the maximum EM value.

Table 6 Hyperparameters and values for OCSVM tuning

Parameter	**Values**
kernel	{polynomial, **RBF**, sigmoid}
gamma	{scale, **auto**}
degree	{1, 2, 3, 4}

Figure 7 shows the development of the Excess-Mass value within the combinations. The peaking combination with the highest Excess-Mass is the one bolded in Table 6. There can be seen that from 12 options, only one gives a significantly higher result than the others.

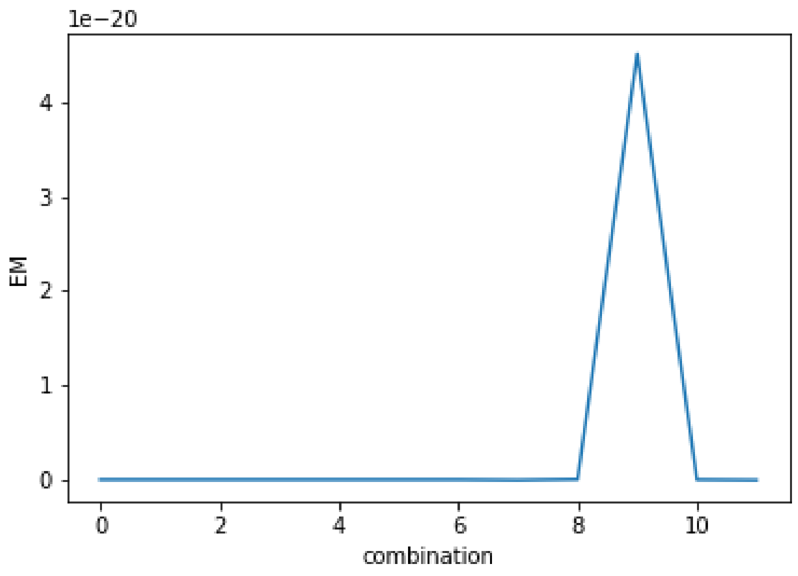

Figure 7 Excess-Mass scores of each OCSVM hyperparameter combination

For PCA, the optimal hyperparameters to be determined are the number of components used for calculating the PCA (n-components), contamination, standardization, and the type of singular value decomposition (SVD) solver. Contamination is a threshold value for the decision function, and it determines the proportion of the outliers. For example, if the contamination is set to 50%, half of the data points are labeled as outliers. Standardization is either done or not, and if done, the data is converted to zero mean and unit variance (Zhao, et al., 2019). SVD solver called as “full” calls standard LAPACK solver from Python SciPy package. SVD solver “arpack” calls ARPACK solver from the same SciPy package and has a requirement that the number of components chosen cannot be equal to the number of features in the data. The rows in the parameter table with the “arpack” solver and 4 components were dropped out before running the parameter tuning. “Randomized” solver uses the SVD method of Halko et al. (2011).

Table 7 shows the values that constructed the parameter table for the PCA tuning and the chosen ones bolded. Interestingly, all the possible combinations with 1 component gave exactly the same highest EM values. The aforementioned can be observed from Figure 8. Thus, some decisions about the other parameters for the actual study were made based on other criteria.

Table 7 Hyperparameters and values for PCA tuning

Parameter	Values
n-components	{**1**, 2, 3, 4}
contamination	{0.001, **0.01**, 0.1}
standardization	{true, **false**}
SVD solver	{full, arpack, **randomized**}

Multiple combinations give precisely the same Excess-Mass value. It is known that the number of outliers is around 1% in the data set. Therefore, contamination was chosen to get the value of 0.01. Standardization does not affect the results either. Hence it has been left out to avoid any extra calculation time. For the SVD solver, the documentation by Zhao et al. (2019) suggests the "randomized" solver for lower dimensional cases. Therefore it was chosen.

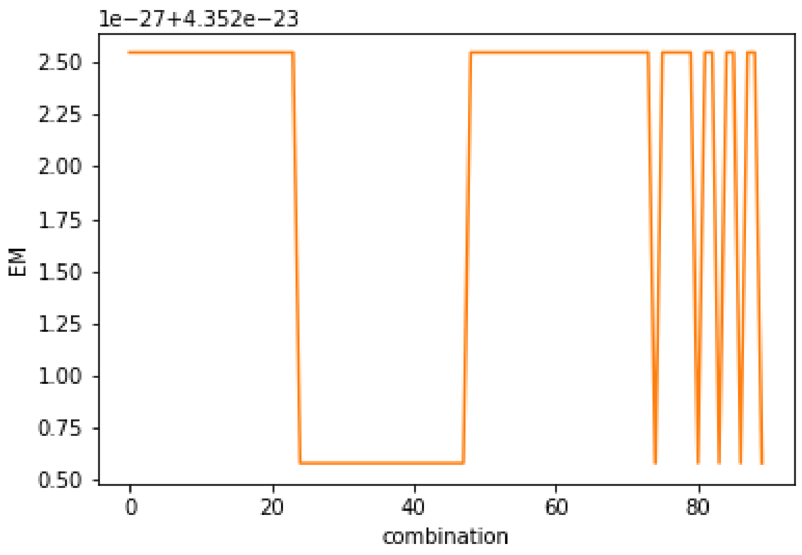

Figure 8 Excess-Mass scores of each PCA hyperparameter combination

Isolation Forest parameter tuning includes finding the optimal number of base estimators (n-estimators), maximum number of data points used for all the trees in the ensemble (max samples), bootstrapping, and the maximum number of features used to train each base estimator (max features). "Auto" maximum number of data points in each tree is minimum from the number of data points in the data set and 256. The bootstrapping determines if the trees would be fitted on random subsets of training data. Table 8 shows the parameter values that are used to build the combinations for parameter tuning. The optimal values are bolded.

Table 8 Hyperparameters and values for iForest tuning

Parameter	**Values**
n-estimators	{10, 50, **100**, 500}
max samples	{100, 500, 1000, **all**, auto}
bootstrap	{**true**, false}
max features	{1, **2**, 3, 4}

Figure 9 shows the Isolation Forest’s development of Excess-Mass values over the different combinations of parameters. The index of the best combination is 109. From the figure can be concluded that many parameter combinations give pretty good results, which means that the algorithm is somewhat stable in terms of changes in the parameters.

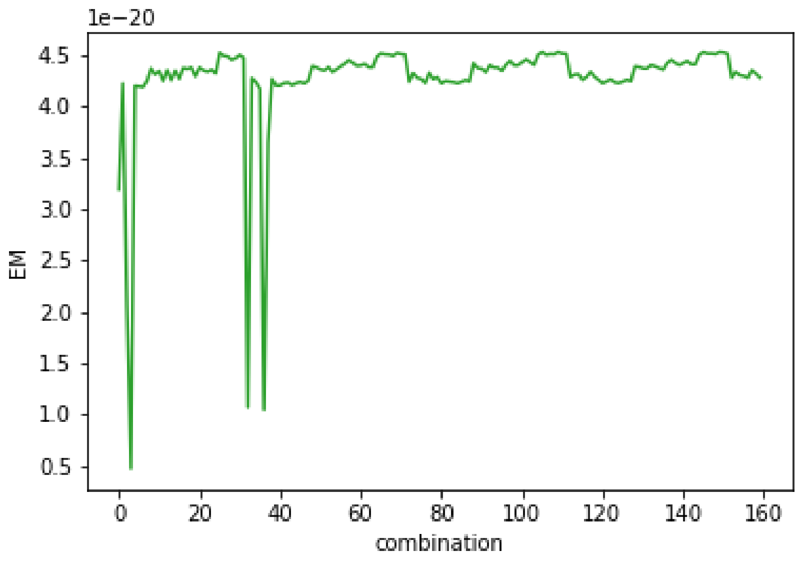

Figure 9 Excess-Mass scores of each iForest hyperparameter combination

Local Outlier Factor has only two critical parameters to be tuned – the number of neighbours (n-neighbours) and contamination. The number of neighbours determines how many neighbours are used to calculate the relative density around the data point. Contamination determines the proportion of the outliers in the dataset. Table 9 has the values used for LOF parameter tuning. With the most optimal contamination, 0.001 with 90 neighbours gave the highest EM value.

Table 9 Hyperparameters and values for LOF tuning

Parameter	Values
n-neighbours	{10, 20, 30, …, **70**, 80, 90, 100}
contamination	{**0.001**, 0.01, 0.1, 0.2}

From Figure 10 can be observed that the Local Outlier Factor Excess-Mass measures with different hyperparameters are rather stable, excluding the few very poorly performing ones. In the graph, combination number 24 gives the best performing parameters, also shown in Table 9.

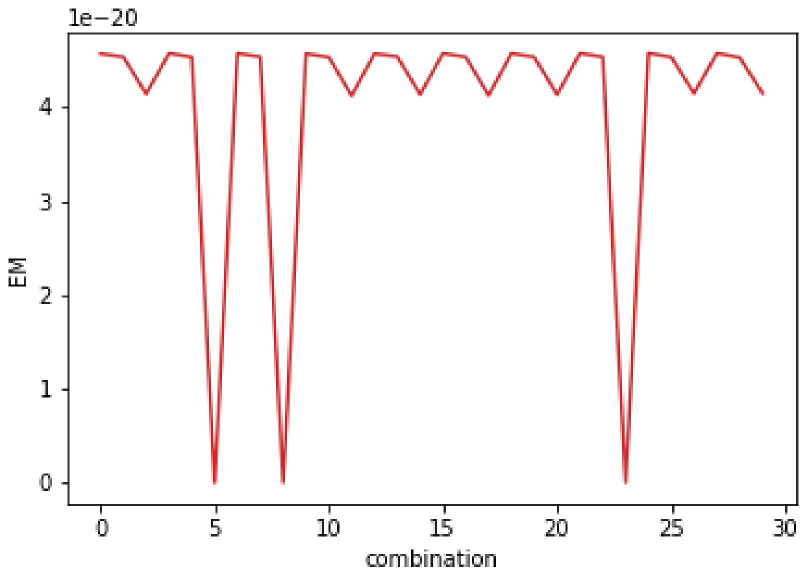

Figure 10 Excess-Mass scores of each LOF hyperparameter combination

The most optimal parameters for each algorithm are chosen to initialize each algorithm described in the next subsection.

3.2.2 Implementing the algorithms

The algorithms' actual testing happens by iterating the train-test split, training, and testing the model in a loop. All the important measures for further analysis are saved from each iteration loop, and then the average results are presented in the next chapter. In a format of simplified pseudo-code following Python code style, the iteration loop of each algorithm looks as follows:

```
1. model = Algorithm(parameters)
2.
3. for i in range(1,n):
4.
5.   df_train, df_test = train_test_split(df, test_size=0.50,
6.   random_state = i)
7.
8.    model.fit(df_train)
9.    model_labels = model.predict(df_test)
10.  s_df = model.decision_function(df_test)
11.  s_unif = model.decision_function(unif)
12.  em = EM(s_df, s_unif)
13.  mv = MV(s_df, s_unif)
```

Here, the first line initializes the model with the given parameters. Parameter tuning in the previous subsection gives the parameter values for each algorithm. The loop iterates the lines from 5. to 13. for n times. The data set (df) is split into a training and testing set (df_train and df_test). Here the ratio between training and testing is 50%, which means that half of the data points are for training and the other half is for testing. The train-test-split's random state determines how many times the split is randomly done before assigning the training and testing sets. The eighth line fits the initialized model to the training data, i.e., trains the model. Line 9 predicts the outlier and inlier labels for the test data set, and line 10 saves the decision function that determines the labeling for the test data. The same decision function is saved for the uniformly distributed data to calculate the Excess-Mass and Mass-Volume measures in lines 12 and 13. All the necessary results from each iteration are saved into a list for further analysis.

The first algorithm tested and implemented for the data is the One-Class Support Vector Machine. The used Python package was the Scikit-learn OCSVM algorithm, which is built based on the LIBSVM library by Chang & Lin (2011). The One-Class SVM implementation by Chang and Lin is based on the OCSVM theory of Schölkopf et al. (2001). The parameters and the values for those can be found in Table 6, where the optimal values are bolded. The algorithm is iterated 100 times, and OCSVM predict-function returns label -1 for outliers and 1 for inliers.

PCA based outlier detection algorithm implemented in this research is from the PyOD library. The calculation is based on Shyu et al. (2003) research, built by Zhao et al. (2019). Tuned parameters for PCA can be found in Table 7. Train-test-split, training, and evaluation are iterated 100 times as one iteration takes less than 1 second for PCA. The labeling in PCA is 1 for outliers and 0 for the inliers.

The Isolation Forest evaluated in this study is from the Scikit-learn library by Pedregosa et al. (2011). The algorithm's theoretical background from two studies by Liu et al. (2008) and (2012). Table 8 shows the optimal parameters used for initializing the Isolation Forest. The calculation time of the iForest is less than one minute. Thus, the number of iterations done is 100 as the script's total running time stays moderate. Scikit-learn Isolation Forest returns -1 for outliers and 1 for inliers.

The LOF algorithm used is from the Scikit-learn library, and it is built based on the study by Breunig et al. (2000). Tuned parameters for LOF can be found in Table 9, and the loop iterates the splitting, training, and evaluation measure calculation 100 times. Like Isolation Forest, LOF returns -1 for outliers and 1 for inliers when calling predict-function.

3.3 Evaluation and benchmarking method

Comparing the different outlier detection algorithms to determine the best one for the purpose is done by measuring the EM and MV measures. The performance is evaluated using Goix's (2016a) Mass-Volume and Excess-Mass measures to rank each algorithm compared to each other. In addition to the MV and EM measures, some other aspects are also considered part of this comparison and benchmarking, such as calculation time and implementation requirements. As the best performing algorithm is potentially going to be implemented as a part of an actual product, things like calculation time and implementation requirements are also essential to take into consideration on the side of the performance of the algorithm being mainly under evaluation.

As already mentioned, the author of the EM and MV functions is Goix (2016a; Goix, 2016b), and they are available in the authors GitHub repository. The maximum value of *t* was defined to be 0.9, and α was determined to be a minimum of 0.9 and 0.999 at maximum. Volume support is used to calculate *t* and both measures – EM and MV. Volume support is a product of an array that is the difference between the maximum and the minimum value of the data set. Value *t* is defined as a range between 0 and 100 divided by volume support with a step of 0.01 divided by the volume support. Data with uniform distribution is drawn by using the Numpy library in Python that includes random uniform function. The uniform distribution data is the same size as the studied test data set. The minimum and maximum values also follow the original data set of this study. As a final step before the actual EM and MV calculation, the algorithms' decision functions were defined for both uniformly distributed and the actual data. After defining all the EM and MV curve calculation parameters, the EM and MV scores are calculated. For plotting purposes, a value called "amax" was determined. "Amax" defines how many values are taken to when plotting the curve. Here the "amax" is -1, which means that all the values are taken into the plot except the very last values for the EM curves.

4 RESULTS

In this chapter, all the results obtained from the study described in the previous chapter are introduced. In addition to Excess-Mass and Mass-Volume measures, outlier counts, proportions, and calculation times are presented to get more insights. All the results are averages of the iterated values, as each algorithm was repeatedly run 100 times with different training and testing data sets.

Figure 11 shows the Excess-Mass curves for all the models. The area under the EM curve is the largest with the best performing algorithm based on EM criteria. Based on visual interpretation, one cannot clearly tell whether the area under iForest's curve is larger than OCSVM's. The ranking of PCA and Local Outlier Factor is clearer in the graph. The EM score of PCA seems to be the lowest based on the figure. LOF seems to have performed better but not as well as OCSVM or Isolation Forest.

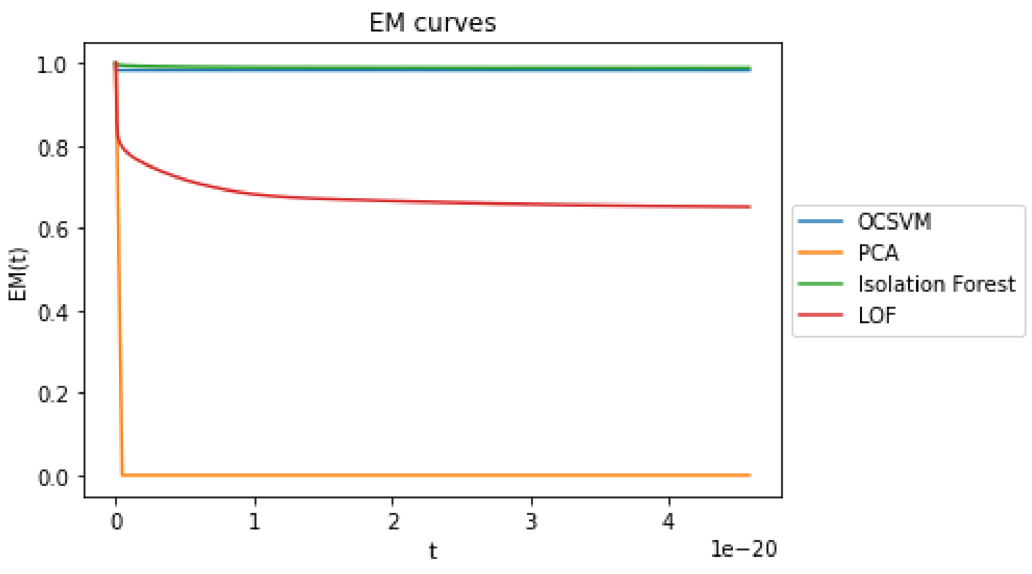

Figure 11 Excess-Mass curves

The Mass-Volume curves can be found in Figure 12. The algorithm with the smallest MV score is the best performing model. In other words, a smaller area under the curve is the most desirable in terms of performance. Based on the visual interpretation of Figure 12, Isolation Forest seems to have the smallest area under the MV curve. OCSVM's score is not as close to the iForest's as it is with EM measure. Overall ordering of the MV results is clearly visible. PCA keeps on performing poorly, and LOF somewhat better than PCA.

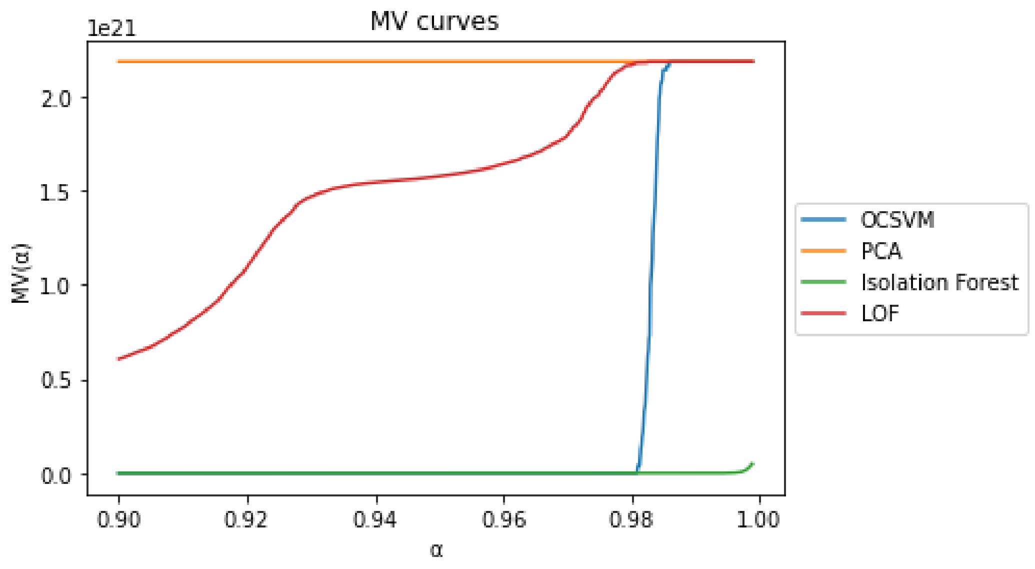

Figure 12 Mass-Volume curves

Table 10 summarizes the results seen in the above curves showing the exact values of the scores. The best results are bolded, and the worst are underlined. The ordering of OCSVM and iForest was not evident in Figure 11. Table 10 shows that the algorithms' scores are very close to each other, but Isolation Forest has a slightly higher EM score making it the best performing algorithm in terms of EM measure.

Table 10 Table of results including EM and MV scores and execution time of the algorithm in seconds

	EM score	**MV score**	**Calculation time (sec)**
OCSVM	4.5036e-20	3.4284e+19	<u>14.5504</u>
PCA	<u>4.3523e-23</u>	<u>2.1584e+20</u>	**0.0031**
Isolation Forest	**4.5339e-20**	**5.368e+16**	1.7164
LOF	4.3691e-23	1.5411e+20	0.7299

Table 10 verifies that Isolation Forest has the lowest MV score, observed already from Figure 12. Only based on the MV score, the other algorithms' scores are not very close to iForest. Both measures agree with each other about the ordering. OCSVM is the second-best algorithm, followed by LOF. The algorithms' computational performance is also visible in Table 10, and OCSVM calculation seems to take significantly more time than the others. Calculating labels with PCA takes barely any time making it the fastest algorithm of all.

Table 11 includes the counts of outliers and inliers of all the algorithms. As it is known that the proportion of outliers is less than 1%, the proportion of outliers labeled by each algorithm was also calculated. Even though one data point cannot be partially labeled to be an outlier, the counts are not integers. On each iteration round, the counts of the labels were saved, and later the average of both counts was calculated, which explains the results not being whole numbers.

Table 11 Each algorithm's outlier and inlier count and proportion of outliers

	OCSVM	**PCA**	**iForest**	**LOF**
Inliers	10 173.2	21 894.8	20 885.0	22 101.2
Outliers	11 936.8	215.2	1 225.0	8.8
Proportion of outliers	53.99%	0.97%	5.54%	0.04%

The share of outliers labeled by OCSVM is significantly higher than the same share of other algorithms, allocating over 50% of the data points to be outliers. PCA and LOF got closer to the 1% threshold given by the case company's existing logic. Isolation Forest's ratio of 5.54% is quite far from the threshold of 1%.

The ranking of the algorithms based on the EM and MV is the following. Isolation Forest performed best, followed by One-Class Support Vector Machine, Local Outlier Factor, and PCA, respectively. However, considering other factors like the calculation time and the inlier-outlier counts, the ordering could also be different, which is discussed in the next chapter.

5 CONCLUSIONS AND DISCUSSION

Significantly differing data points in data set, outliers, in spend analysis can either lead the decision-maker to false conclusions. Outliers in spend data may also indicate fraud in the procurement processes or be useful information about potential savings. A price outlier can be a simple input error in measurement units as well as knowingly over-invoiced purchase. A drastic price drop may seem like an outlier but can also provide an opportunity for savings if it is not an error.

Outlier detection algorithms studies in this thesis are One-Class SVM (Schölkopf, et al., 2001), PCA (Shyu, et al., 2003), Isolation Forest (Liu, et al., 2008), and Local Outlier Factor (Breunig, et al., 2000). The performance was measured with Excess-Mass and Mass-Volume criteria by Goix (2016a). Because the thesis's case company could implement the suggested machine learning algorithm as a part of their product, each model's computational performance was studied as well. Hence, in addition to EM and MV measures, the calculation times of the algorithms were measured. As the case company has a statistical outlier detection in use, results from that were used as a loose threshold value for evaluating the proportions of outliers labeled by each algorithm.

Overall results from the EM and MV analysis is that Isolation Forest performed the best of all the algorithms. OCSVM had the second-best results, with LOF right behind that with the third-best outcome. PCA-based outlier detection method had the worst EM and MV results. In terms of calculation time, PCA was definitely the most efficient algorithm, and OCSVM significantly the most time-consuming method. PCA was the closest to the threshold outlier proportion value of 1%. Only OCSVM labeled outliers significantly over the threshold value. Isolation Forest, the best performing algorithm, labeled a somewhat higher proportion of outliers than the current statistical approach. Based on the summarized results above, the research questions are answered.

Which algorithm from One-Class SVM, PCA, Isolation Forest, and Local Outlier Factor perform the best in spend data price outlier detection?

Compared to each other, the Isolation Forest performed best based on EM and MV measures. The Isolation Forest calculation time was not the lowest, but it was lower than LOF and OCSVM. When predicting the labels, the algorithms needed to go through approximately 40 thousand rows of data in total. Isolation Forest did this in under 2 seconds, which is a rather good result, especially compared to the 15 seconds of OCSVM calculation. OCSVM clearly assigns too many data points to be outliers with a proportion of over 50% compared to the 1% threshold. On the other end, LOF allocated only 8.8 data points on average to be outliers, which means that less than pro mill of the data would be outliers. The number of labeled outliers closest to the current statistical method was PCA's 0.97%. However, based on the performance measures, PCA did not do very well, and those results of PCA were undoubtedly the worse. Isolation Forest's share differs from the threshold of 1% to some extent, but the difference is less than 5 percentage points. Overall, based on the results, the answer to the first research question is that Isolation Forest performs the best in detecting price outliers from spend data.

Which algorithm is suggested for the case company to develop a machine learning-based outlier detection on top of the existing statistical approach?

From the business and strategic sourcing point of view, the outlier detection algorithm's accuracy is essential. As the current statistical method's accuracy is not measured, it is unsure if the number of outliers is precisely 1%. The threshold value only gives direction to rule out the most extreme deviations from that. In this case, the comparison would exclude OCSVM from the optimal options as the proportion differs significantly from the threshold. Excess-Mass and Mass-Volume give the direction of how accurate the assignment has been. It could be noted that it is more important to have more values classified as outliers to some extent than too few of them. As outliers in spend analysis can be fraud, error, or opportunity, an expert should always review at least the most drastic ones and try to find the reasoning behind them. Of course, that may be laborious when the amount of data is very high. Errors in the data could be corrected to keep the accuracy in the analysis. Fraud could be detected and prevented from happening again in the future, and savings could be obtained if materials can be purchased at a lower price. If the algorithm detects too few potential outliers, some of these might be missed. Of course, a perfect set including only the outliers correctly would be the outlier detection's optimal outcome. With the set up in this thesis, that cannot be obtained due to the lack of labels to verify the results. An expert could go through a reasonable set of data from 500 to 1000 rows of outliers and determine if they require any action, which means that incorrectly labeled outliers, also known as false-positive outliers, could be better than false-negatives in the spend analysis context.

The hyperparameter tuning also gave interesting results and insights about the algorithms. An algorithm with only one parameter combination giving significantly better results can be very unstable in case of changes in the parameters. OCSVM in this study gave such results in the parameter tuning. All the other algorithms' EM scores were more balanced throughout the combinations than OCSVM's. Especially, the Isolation Forest's performance with different combinations of hyperparameters was rather stable. The algorithm's sensitivity to hyperparameter changes could also be considered when choosing an algorithm for a specific use. Incorrectly executed parameter tuning for an algorithm may produce much poorer results in the latter parts of the implementation if the algorithm chosen is sensitive for variations in the hyperparameters.

The calculation time of the algorithms was also measured. As the algorithms would potentially be implemented as a part of a software product, efficiency and computational performance are important factors considering the algorithm selection. When implementing a machine learning algorithm for real use cases, the model would be trained rarely, and in many cases, it may take a lot longer time than getting the results for new data. The calculation time of the results for the new data is critical, as well as the optimization of the predicting function instead of the training of the model. Determining the labeling for approximately 40 thousand rows under 2 seconds is relatively swift even though the Isolation Forest's calculation time was not the lowest (Table 10).

Considering all the algorithms in this study, OCSVM and PCA is easier and quicker for the case company to develop as a part of their product. However, these algorithms' poor performance, PCA's EM and MV scores, and OCSVM's large proportion of outliers eliminates them from being suggested for the case company in this study. Hence, based on all the conclusions and discussions above, the suggested algorithm for further development is the best-performing Isolation Forest in terms of Excess-Mass and Mass-Volume measures. Considering the other factors discussed above as well, Isolation Forest seems like the best alternative for the task and further development.

In the future, it would be reasonable to get labeled data to verify the results. All the algorithms could be tuned and tested with the labeled data set, but especially validating Isolation Forest’s results before taking it to use. From the implementation point of view, determining a normal group of the data for semi-supervised outlier detection could also be an area of development if full labeling of a data set is not doable. The algorithm developed for practical use could utilize the normal labeled set to train the model and detect outliers from new unlabeled data. In addition to the basic algorithms presented in this thesis, many extensions and enhancements of those exist. Testing the enhanced algorithms could provide even better results than the basic ones. Checking and analyzing the resulting outliers of all tested algorithms could be an area of further research to understand the spend data outliers better. Understanding the outliers in the procurement spend data may open an opportunity to combine the results from multiple different algorithms to create an ensemble more robust than only one algorithm. The unsupervised algorithms could also be used as one ensemble, considering all the algorithms’ results. The data points with the most votes from all the algorithms would be elected as outliers. Results from this voting strategy would definitely be an interesting direction of research.

REFERENCES

Aberdeen Group, 2004. *Best Practices in Spending Analysis Cure for a Corporate Epidemic,* s.l.: Aberdeen Group.

Aggarwal, C. C., 2013. *Outlier Analysis.* New York: Springer Science+Business Media New York.

Amer, M., Goldstein, M. & Abdennadher, S., 2013. *Enhancing One-class Support Vector Machines for.* [Online], ACM, p. 8–15.

Ball, B., 2019. *USING AI FOR ANOMALY DETECTION TO IMPROVE YOUR SUPPLY CHAIN TRANSACTIONS,* s.l.: Aberdeen.

Barnett, V. & Lewis, T., 1987. *Outliers in statistical data.* 2nd ed. Chichester: Wiley.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J., 2000. LOF: identifying density-based local outliers. *SIGMOD record,* 29(2), pp. 93-104.

Burges, C. J., 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data mining and knowledge discovery,* 2(2), pp. 121-167.

Celebi, M. E. & Aydin, K., 2016. *Unsupervised Learning Algorithms.* New York: Springer International Publishing.

Chang, C.-C. & Lin, C.-J., 2011. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology,* 2(3), pp. 1-27.

Chen, Y. & Wu, W., 2018. Isolation Forest as an Alternative Data-Driven Mineral Prospectivity Mapping Method with a Higher Data-Processing Efficiency. *Natural resources research (New York, N.Y.),* 28(1), pp. 31-46.

Chowdhary, P. et al., 2011. *Managing Procurement Spend using Advanced Compliance Analytics.* [Online], IEEE, p. 139–144.

Clémençon, S. & Thomas, A., 2018. *Mass Volume Curves and Anomaly Ranking,* s.l.: hal-01516919.

Computer Weekly News, 2020. *Coupa Software Incorporated; Patent Application Titled "System And Method For Repeatable And Interpretable Divisive Analysis" Published Online (USPTO 20200043006).* Atlanta: s.n.

Ding, Z. & Fei, M., 2013. An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window. *IFAC Proceedings Volumes,* 46(20), pp. 12-17.

Erfani, S. M., Rajasegarar, S., Karunasekera, S. & Leckie, C., 2016. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern recognition,* Volume 58, pp. 121-134.

Gao, J. et al., 2011. *RKOF: Robust Kernel-Based Local Outlier.* s.l., Springer, Berlin, Heidelberg, pp. 270-271.

Ghorbel, O., Abid, M. & Snoussi, H., 2015. *A Novel Outlier Detection Model Based on One Class Principal Component Classifier in Wireless Sensor Networks.* [Online], IEEE 29th International Conference on Advanced Information Networking and Applications, pp. 70-76.

Goix, N., 2016a. *How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?.* New York, ICML2016 Anomaly Detection Workshop.

Goix, N., 2016b. *Machine Learning and Extremes for Anomaly,* Paris: TELECOM PARISTECH.

Goix, N., Sabourin, A. & Clémençon, 2015. *On Anomaly Ranking and Excess-Mass Curves.* San Diego, W&CP.

Halko, N., Martinsson, P. G. & Tropp, J. A., 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM review,* 53(2), pp. 217-288.

Hariri, S., Kind, M. C. & Brunner, R. J., 2019. *Extended Isolation Forest,* [Online]: IEEE Transactions on Knowledge and Data Engineering.

Hawkins, D., 1980. *Identification of outliers.* London: Chapman and Hall.

Heath, S., 2006. TACKLING SPEND ANALYSIS. *Contract Management,* 46(1), pp. 40-42,44-45.

Hodge, V. & Austin, J., 2004. A Survey of Outlier Detection Methodologies. *The Artificial Intelligence Review,* 22(2), pp. 85-86.

Kotu, V. & Deshpande, B., 2016. *Predictive analytics and data mining : concepts and practice with rapidminer.* Waltham, Massachusetts: Elsevier Inc..

Li, S. & Fu, Y., 2017. *Robust Representation for Data Analytics Models and Applications.* [Online]: Cham: Springer International Publishing.

Liu, F. T., Ting, K. M. & Zhou, Z.-H., 2008. *Isolation Forest.* [Online], IEEE, pp. 413-422.

Liu, F. T., Ting, K. M. & Zhou, Z.-H., 2012. Isolation-Based Anomaly Detection. *ACM transactions on knowledge discovery from data,* 6(1), pp. 1-39.

Maimon, O. & Rokach, L., 2005. *DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK.* Tel-Aviv: Springer Science+Business Media.

Manevitz, L. M. & Yousef, M., 2001. One-Class SVMs for Document Classification. *Journal of Machine Learning Research,* Volume 2, pp. 139-154.

Mehrotra, K. G., Mohan, C. K. & Huang, H., 2017. *Anomaly Detection Principles and Algorithms.* New York: Springer International Publishing.

Pandit, K. & Marmanis, H., 2008. *Spend analysis : the window into strategic sourcing.* Fort Lauderdale: J. Ross Publishing.

Pedregosa, F. et al., 2011. Scikit-learn: Machine Learning in Python. *Journal of machine learning research,* 12(85), p. 2825−2830.

Phillips, T. B. & Lanclos, R. J., 2014. *Data analytics in procurement fraud prevention,* s.l.: U.S. Government.

Ruck, C., n.d. *4 Steps: Predictive Analytics in Procurement.* [Online]. Available at: https://www.orpheus-it.com/academy/knowledge-spend-analysis/predictive-analytics-in-procurement?highlight=WyJvdXRsaWVyIl0=. [Accessed 14 September 2020].

Schubert, E., Zimek, A. & Kriegel, H.-P., 2012. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery,* 28(1), p. 190–237.

Schölkopf, B. et al., 2001. Estimating the Support of a High-Dimensional Distribution. *Neural computation,* 13(7), pp. 1443-1471.

Schölkopf, B., Williamson, R. C. & Bartlett, P. L., 2000. *New Support Vector Algorithms.* s.l.:Neural Computation.

Shin, H. J., Eom, D.-H. & Kim, S.-S., 2005. One-class support vector machines—an application in machine fault detection and classification. *Computers & Industrial Engineering,* 48(2), pp. 395-408.

Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K. & Chang, L., 2003. *A Novel Anomaly Detection Scheme Based on Principal Component Classifier* , s.l.: s.n.

Sievo, 2020a. *Sievo.* [Online]. Available at: https://sievo.com/. [Accessed 4 September 2020].

Sievo, 2020b. *Spend Analysis 101: Comprehensive Guide to Procurement Spend Analytics.* [Online]. Available at: https://sievo.com/resources/spend-analysis-101. [Accessed 4 September 2020].

Sridhar, A. & Suman, K. A., 2019. *Beginning Anomaly Detection Using Python-Based Deep Learning.* New York: Springer Science+Business Media.

Suplari, 2019. *Data Analytics for Procurement Series: Calculating Outliers in Spend Data – Part 1.* [Online]. Available at: https://www.suplari.com/data-analytics-for-procurement-series-calculating-outliers-in-spend-data-part-1/. [Accessed 14 September 2020].

Su, S. et al., 2019. An Efficient Density-Based Local Outlier Detection Approach for Scattered Data. *IEEE access,* Volume 7, pp. 1006-1020.

Talluri, S. & Narasimhan, R., 2004. A methodology for strategic sourcing. *European Journal of Operational Research,* 154(1), pp. 236-250.

Vapnik, V., 1995. *The Nature of Statistical Learning Theory.* New York: Springer-Verlag.

Zhao, Y., Nasrullah, Z. & Li, Z., 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research,* 20(96), pp. 1-7.