Vrije Universiteit Amsterdam

Master Internship Report

Unsupervised Outlier Detection in Support of Financial Due Diligence

Author: Lotte Huft (2596520)

1st supervisor: prof. dr. Mark Hoogendoorn
daily supervisor: dr. Jan van Angelen (SINCERIUS)
2nd reader: prof. dr. André C.M. Ran

August 25, 2022

Preface

I wrote this paper as a thesis for the Master Business Analytics followed at the Vrije Universiteit van Amsterdam. It is part of an internship I followed at SINCERIUS as a contribution to the modernization of the Financial Due Diligence branch of the organization.

I would like to thank Jan van Angelen from the Business Intelligence team for being my external supervisor. In addition, I would like to thank all the consultants at SINCERIUS for giving their expert opinions to help me to be able to evaluate my results and everyone who helped me in any other way before and during my thesis.

I would also like to thank Professor Mark Hoogendoorn for being my supervisor and helping me during this whole process. He greatly supported me with his guidance and insight. Lastly, I would like to thank Professor André C.M. Ran for being my second reader.

Summary

In this thesis the added value of unsupervised outlier detection during the Financial Due Diligence process is discussed. More specifically, it is performed on monthly revenue data per client of a set of companies. Based on a review of related literature three algorithms have been chosen and their results are compared to each other and to results from expert interviews with consultants within the field. This is done to answer the research question; "Which of the researched algorithms used to perform an unsupervised outlier detection on client revenue data yields the result which most closely resembles the outlier detection performed by FDD consultants?".

From the results it is shown that from the algorithms tested Local Outlier Factor performs best when compared to the experts' opinions. However, even the results from Local Outlier Factor still leave a lot of room for improvement by performing further research.

The most significant areas of improvement for future research are concluded to be re-implementing the algorithms on a larger data set with both a broader selection of company types as well as a greater selection of similar companies. Another important area to pay attention to is the feature engineering and selection.

Contents

1 Introduction

When the transaction of ownership of a company or part of a company occurs, i.e., Mergers and Acquisitions (M&A), different kinds of investigations are done into the company in question. These investigations are often referred to as due diligence. Examples of these types of due diligence are Strategic, Financial, and Legal Due Diligence. During Financial Due Diligence (FDD), one of the main objectives is finding anything notable in the heaps of financial data these companies have. Parts of the data are obtained from audit files. These files contain all the financial transactions of a company and have many different uses, e.g., during audit. (1) (2)

The data from audit files is mostly analyzed manually, only using some software tools to transform and visualize data. This leaves a lot of room by improvement when implementing forms of Machine Learning. Since the Transaction Services (TS) market is a highly competitive one, having a leg up to the competition can contribute in whether a company is chosen to perform the FDD. In addition, making a process quicker can save time within a FDD, which in turn can fasten the turnover time of projects and leave more time to take on more new deals.

The field of Machine Learning is undeniably still a developing one with an increasing popularity in different kinds of fields. Every once in while a paper will be written about the so-called trends and challenges still to be solved. Examples of this are the papers by Jordan and Mitchell (3) and by Holzinger *et al.* (4). These papers give some demonstration of the magnitude the Machine Learning industry has already reached and at the same time show some of the many problems that are yet to be solved. The financial sector is not staying behind on the advancements. In the book by Puneet Mathur (5) the influence

Machine Learning has on three different sectors has been thoroughly researched. Finance is named here as one of these sectors, with the other two being retail and healthcare. In Chapter 14, fraud detection is pointed out as one of the fields within finance where the advantageous of Machine Learning have already been recognized and, partly, realized. Machine Learning has now become an important part of the field. Therefor the field of fraud detection will be used as a source of existing research in this thesis. In the same chapter some application examples of Machine Learning within a couple of other finance segments are described. Two of these segments are "Finance advisory and management services" and "Accounting", which are both somewhat related to Financial Due Diligence. However, these fields are also classified as the two with the lowest Technology Maturity Level. This confirms the observation that not all of finance has adopted Machine Learning. In the sector of FDD most of the analysis seem to still be done by hand.

One of the possible areas of improvement is performing an outlier detection on the "revenue per client" data. This data consists of all monetary income transactions of a company, specifically those that can be linked to a specific client of the company in question. The data can be summed to portray these transactions monthly. As a part of a FDD, for most companies this data is analyzed to locate abnormal behavior from clients. This is done to make sure the revenue of a company is sustainable. For example, if a company gets a large part of its revenue from a single client, this will be flagged. Another example could be high spikes or dips in the revenue linked to an otherwise relatively consistent customer. Abnormal behavior will consequently be queried at the company in question to be able to conclude if the abnormal behavior is also something to be alarmed about.

The locating of these abnormal behaviors, or outliers in other terms, is the problem that this thesis will be focusing on. This is a task that can be very time consuming and sensitive to errors when done by humans, even when these humans are sector experts. In the current set up, the revenue per client data is mostly analyzed using Excel templates. Specifically, by visualizing the trend lines, of the top clients, and searching for outliers by looking at these visuals. Although Excel is an excellent tool for data analysis, it will hopefully be easier, faster, and less prone to errors when this task is done using Machine Learning. Since this specific problem is not one that has been extensively researched, this thesis will be comparing it to similar problems where outlier detection is used on financial data, mainly the aforementioned fraud detection.

In the paper by Xiaogang Su and Chih-Ling Tsai (6) the history of the Machine Learning problem of outlier detection is explained. In the early stages performing a statistical analysis was the most sophisticated way of finding outliers in a data set and this had to be done under assumptions of the distribution of the data. In the current day, this might still be a feasible way to approach some problems within the field. However, with the evolution of data collection the size of data sets has greatly increased, both by number of dimensions and number of observations. These statistical analysis approaches are not scalable enough to be able to deal with these types of sets within a timely matter. In addition, these approaches rely on the data to be quantitative and will not work for ordinal or categorical data.

Furthermore, some more current approaches are discussed in the paper and a distinction is made between techniques to be used when the true outliers of the data are known and those used when this information is not available. In the latter case a technique referred to as unsupervised Machine Learning should be used.

Another part of the paper discusses the problem that data used for outlier detection is inherently unbalanced, which makes classifying new data into outliers and non-outliers a complicated task.

The data used in this thesis is completely unlabeled and it is also hard to classify each of the instance within it as an outlier with a 0 or 1 certainty. In addition to the previously discussed challenges with supervised learning, this is enough reason to decide to approach the outlier detection on the client revenue data as an unsupervised Machine Learning problem.

In this thesis, three algorithms will be tested against the problem of outlier detection in client revenue data. These algorithms have been proven to work well for similar problems. The research question deducted from this problem is "Which of the researched algorithms used to perform an unsupervised outlier detection on client revenue data yields the result which most closely resembles the outlier detection performed by FDD consultants?".

The second chapter of this thesis will be focusing on the background of this problem. This contains the current situation at SINCERIUS and some more information about the company. In addition, this chapter will be discussing the related Machine Learning background. Some literature related to the problem of this thesis is discussed in chapter three. This includes some literary background of Unsupervised Outlier detection and elaborating on some of the commonly used algorithms within this field.

In the four chapter, the enquiring, exploration, description, and processing of the data are be discussed.

The methods used to research the question at hand are described in chapter five and in chapter six the experimental setup is given. After, the results derived from these methods are reported in chapter seven.

Lastly, chapter eight will be the conclusion and discussion of this thesis.

Find the appendix and references at the end of the report.

2

Background

2.1 Application Domain

This thesis is written about research done under an internship done at SINCERIUS. This company is made up of multiple branches. They started as a company providing Transaction Services in the form of Financial Due Diligence for M&A transactions. After some years, a Business Intelligence (BI) team was added, which now focuses on the improvement of different internal processes but also sells Financial and Operational tools to improve the insight the clients have of their data. Lastly, a Performance Improvement (PI) branch was established. The PI team helps companies elevate their business by using their extensive knowledge to produce analyses on how to improve their operations. This thesis will be building on the desire within SINCERIUS to always be improving. The focus will be on the processes within the TS team. The team has years of audit files available of a great number of companies in different branches and with a variety of backgrounds. At SINCERIUS, these audit files play a great part in determining whether a company is as fiscally healthy as they present to be. As of this moment, they mainly dissect these audit files using EXCEL.

2.2 Machine Learning

According to Plasek (2016) (7) defining the history of Machine Learning is impossible. He also claimed that the innovation of the field has not come close to what historical figures like Alan Turing deemed as possible. However, in 1959, A.L. Samuel (8) wrote a paper about writing a program that was able to play checkers. In this same paper he would also define the term Machine Learning. He has also described the field of Machine Learning as "Field of study that gives computers the ability to learn without being explicitly programmed". All this has made him a crucial part of the history of this science.

Machine Learning problems can typically be put in one of four classes. The next definitions follow those of the book by Rebala *et al.* (9). Firstly, supervised algorithms can make predictions on new data after learning from examples of data where the target values are already known. These target values can also be referred to as labels. It learns what key characteristics decide what the label should be and subsequently use them as a bases to predict the labels of any new data. This method can be used when, for example, doing classification or forecasting. In the case of unsupervised algorithms, the opportunity is created to still learn about data in an automated way when no labels are available. These algorithms are trained to identify trends within the data and create groups based on those similarities. A variation between supervised and unsupervised learning is semi-supervised learning, where only a part of the training data has been labeled to save time. Only a small amount of the data needs to be labeled for the algorithms to work, instead of the large amount supervised learning needs. It then will use techniques similar to unsupervised learning to create clusters and link those to the labeled data. Lastly, Reinforcement learning is a technique used when the algorithm has many to decide from and/or the current situation is ever changing. For example, when an algorithm plays a human in chess. Every turn there are a lot of possible moves and depending on what the opponent's next move is a whole new set of moves is created at every turn. To solve this and other problems with a similar setup, it must learn by trial and error. The longer it learns, the better it gets at estimating the long-term consequences of its choices while trying to maximize a predetermined score. For every decision it makes, it takes into consideration its own current state as well as the current state of the external environment.

Continuing on the subject of unsupervised learning since this is the class of Machine Learning this thesis will be focusing on. Like previously stated, unsupervised machine learning groups similar data by searching for similarities, assigning data to different groups without particularly knowing what these groups entail. The formal term for this is clustering. A deduction from the clustering problem is outlier detection. Outlier detection can be summarized finding data points that can be classified as anomalies. (10)
In other words, the algorithm tries to separate the data in a, smaller, group of outliers and a group of "normal" data. In the following section some examples of techniques that unsupervised outlier detection methods can be based on will be explained in more detail. This follows the methods as named in the paper by Wang *et al.* (11).

Firstly, probabilistic models rely on the assumption that most data originates from the same distribution. If the likelihood that an instance is sampled from this particular distribution is very small, it has a high chance of being an outlier according to this type of model. Next, clustering-based models define outliers as instances not part of clusters as a result of a clustering algorithm. On the other hand, distance-based models compare the distance from each instance to the closest k instances around it, i.e., the k-th nearest neighbor distance. If the k-th nearest neighbor distance of an instance is larger than average, it is probably an outlier by the rules of distance-based models. These methods however do not work well for data sets with multiple clusters with different densities. A similar method that works better for these types of data sets are the density-based methods. These methods compare the density of the area around an instance to those of instances close to it, i.e., it compares the local density to the local densities of its k-nearest neighbors. Since these techniques really focuses on comparing data to other local data, they are also called local models. Lastly, time series outlier detection models are used to find outliers in data sets ordered on some timescale. These outliers can either be a single instance with an unexpected value or a complete window of time. Expected values of time series can be found by taking in consideration some noise, a general trend, and the seasonality. When an instance or a window of time deviates too much from this expectation, it might be an outlier.

3

Review of Related Literature

In a paper by Domingues *et al.* (12) 14 algorithms were compared to each other on 15 different data sets. These data sets contained both real data sets and synthetic ones. To show the performance of these algorithms, the ROC AUC and the PR AUC were calculated. AUC is an abbreviation for the Area Under the Curve and ROC stands for Receiver Operating Characteristics. The ROC-Curve represents the trade-off between the correctly classifying the positive examples, true positives, and falsely classifying negative examples as positive, false positives. In the case of outlier detection this translates to the trade-off between correctly classifying outliers and incorrectly classifying non-outliers as outliers. The PR AUC is a similar measure for binary classification, where PR stands for Precision-Recall and the PR AUC is the trade-off between these two measures. Precision shows what part of the data classified as positive are true positives. Recall shows what part of the real positives have been classified as positive. The PR AUC has been introduced as a substitute for the ROC AUC when dealing with data that contains a large imbalance between the classes. To explain this a bit further, a threshold is set for a classification on problem where only 0.1% of the data is a true positive. This threshold shows to have a low false positive rate, like 0.1, and a high true positive rate, like 0.9, which seems to be a good result. Although, this comes down to 90% of the true positive class being correctly classified, more than 99% of what is classified as positives will be false positives. Therefore, ROC AUC does not work well in these cases. All this is discussed further in a paper by Zhou *et al.* (13).

In addition to testing the performance of each algorithm per data set, the robustness of the algorithms was also tested by Domingues *et al.* (12). This was done by changing the number of features, the sample size, and the noise density. When changing the number of features and the sample size, the training time, prediction time, and memory the algorithms needed was also tracked. This is all done to test how scalable these algorithms are for different types of data.

When looking at the mean average precision over all data sets the best performing algorithms looked to be iForest, RKDE, PPCA, and OCSVM. When testing the robustness of these algorithms all seem to obtain good results. DPGMM and GMM also shows good robustness.

When looking at the increase of training and prediction time when increasing the number of features and the number of samples, LSA and iForest had the best overall result. Other algorithms did perform well one some of these criteria but lacked overall scalability. For example, LOF and RKDE do very well when the number of features is increased for both training and prediction time but increasing the number of samples makes them go over 24 hours, after which the algorithms reach a timeout for this research. In addition, these two algorithms also reached the max memory usage in that part of the testing. As a conclusion in this paper, iForest is as the overall best performing algorithm. RKDE and OCSVM are also given as good performers, yet both are less scalable than the previously mentioned iForest.

Another comparative paper where multiple unsupervised outlier detection algorithms are tested against each other with the help of 14 different data sets is the paper by ur Rehman *et al.* (14). To be specific, Four, similar, proposed methods are tested against 19 state-of-art unsupervised algorithms using the ROC AUC as an evaluation metric. Although the proposed methods seem to perform the best in the results shown for the synthetic data sets, LOF obtained the best ROC AUC values out of the state-of-art algorithms. In addition, LOF also had a far shorter computational time. In the tests using real data, the proposed algorithms again gave the overall best results. However, not all state-of-art algorithms that the paper previously mentioned were included in the results. When looking at the results that were given, LDF and k-NN did outperform the proposed methods on some data sets. Furthermore, the best performing state-of-art algorithms varies somewhat between the different data sets.

Now to discuss some studies that are also dealing with outlier detection in financial settings to show which algorithms are proven to work well for data similar to the set used in this thesis. Firstly, the paper by Ahmed *et al.* (15) compares ten techniques to find outliers in financial data, specifically transaction data of stocks on the Australian Stock Exchange. It uses a set outlier ratio and uses ROC AUC as an evaluation measure. This paper concludes that all but two algorithms performed worse than a random classifier. The algorithms that did perform well were LOF and CMGOS with a true positive rate close to 100%.

Next, the paper by Verhoeyveld *et al.* (16) compares two algorithms to a baseline solution to perform outlier detection on tax declaration to identify fraudulent behavior. The algorithms tested are FWAD and LOF, with the baseline being the absolute sum of some tax ratios, also called fraud indicators. These fraud indicators are also given to the algorithms as features. In this paper the evaluation is done by checking how many of known fraud cases are contained in what the algorithms classify as the most suspicious cases. This research concludes that depending on the sector the optimal method can vary, and some sector specific research might be needed before implementing any of them. Time complexity is also important in this research. Since the baseline performs very well in this area and LOF perform rather poorly, the advice is given to first try the baseline technique. If this does not result in a desirable outcome, it is suggested to try LOF. The paper also suggests another algorithm, FWLOF, that might work well while still adhering to time constraints. However, this algorithm was not included in the research itself.

Another paper using outlier detection to detect fraud is the one by Mensah *et al.* (17). In this paper, the techniques iForest, OCSVM, and k-Means are used to discern transactions where online travel agents try to defraud airlines when selling commissioned travel tickets. These algorithms are tested on a data set without any transformations as well as on a normalized and a standardized version. The evaluations were done using precision and recall. The F_1 and F_2 scores were also calculated with the equation:

$$F_\beta = (1+\beta^2)\frac{precision * recall}{\beta^2 * precision + recall} \tag{3.1}$$

These scores give a combined measure of both recall and precision, where as β increases so does the weight of the recall. The F_2 score was used as the main metric to conclude which algorithm performed best. iForest turned out to be the best performing, but only on the data without any transformations. OCSVM performed second best and got the best score without any transformations. k-Means performed best when the data was either standardized or normalized, with normalization giving the best score.

4

Data

This research uses data obtained from audit files. Audit files are used to exchange financial information between companies. These files can be extracted from most financial systems companies might use and are saved as XML-files. They contain all the financial transactions a company has made in a year. For finding the outliers in the revenue, the focus will lie on the transaction data. The selection of transaction data that only includes revenue data is made by filtering on the account numbers linked to revenue accounts.

In the next part this data will be described further as well as the addition of some extra features from different sources. Next, the properties and characteristics are explored. Lastly, the creation of new features by transforming existing ones is elaborated.

4.1 Data Exploration

In this section, the data will be discussed in more depth. After collecting data on 15 different projects, which sometimes contained multiple companies, the unnecessary columns and rows were dropped, and the remaining data was grouped. The filtering of rows was done so only the transaction data linked to revenue would remain. Columns that were dropped were inconsistent references of descriptions which did not supply any interesting information. Another column that was dropped was the transaction date, since the goal is to approach a simulation of the way it would normally be looked at, which is by month. Month and year are individual columns, which are kept in the data instead of a 'start of month date' since dates are hard to deal with features. Now by grouping the data is reduced to 179,219 rows and the columns year, month, company, and customer id as well as the summed amount of revenue.

The earliest data starts in 2017 and the last available year is 2021. The data contains 35 unique companies and a total of 60,673 unique customers across all companies. However, approximately 51 thousand of these customers belong to one company. The distribution of the other companies and the number of unique customers each has had in the past years can be seen in figure 4.1. The number of unique customers per company is shown after it was normalized. This was done to make lower counts still readable, since there is such a large difference. The bottom eleven all have seven or less customers.

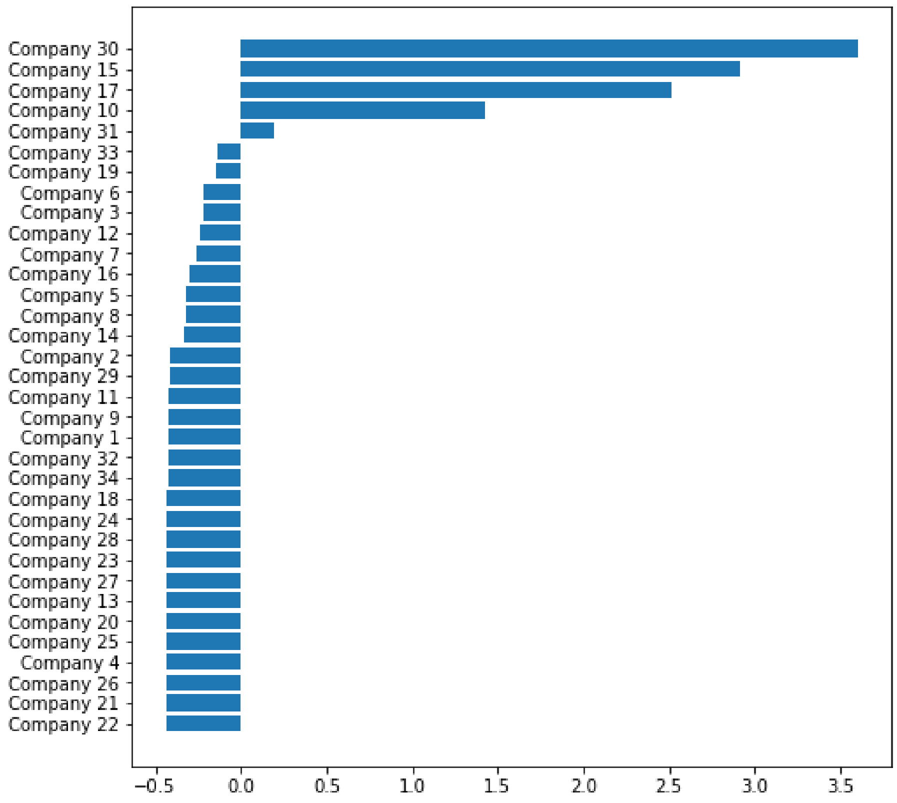

Figure 4.1: Distribution of customers among companies

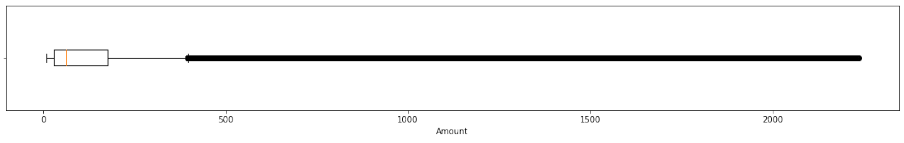

Figure 4.2: Boxplot of distribution of total monthly revenue per customer

One would normally expect revenue to be positive but for reasons like refunds this might not always be the case. During the data exploration it was found that on a monthly per client bases in our data the revenue was positive around 98% of the time. Within the same measures, the minimum value is -81 thousand, the maximum 1.7 million, and the mean 948, with a standard deviation of 1,130. The distribution of the fifth till the ninety-fifth percentile of the summed revenue is shown in figure 4.2. This percentile is taken instead of the total data set to be able to get a better image of the distribution. Else there was not much visible due to the data's long right tail. This tail is still very distinguished in the boxplot, in other words the data is heavily left-skewed. These extreme and negative values might give an indication on where outliers can be found.

In the following part, some different cross-sections of the total revenue are shown and discussed.

In figure 4.3 it shows that the years 2017 and 2021 have much less revenue. However, this is most likely caused by these years having less data. When only looking at the data where companies have reported a full year and averaging the revenue per year over the number of companies a completely different trend is shown. This is shown in figure 4.4. It can be concluded that on average there is a strong trend of increasing revenue for the companies in the data set. This is the case for most of the individual companies within the data.

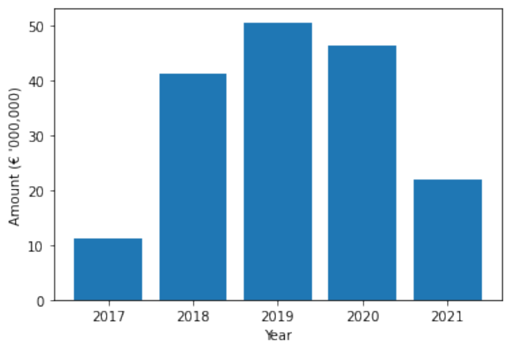

Figure 4.3: Distribution of total revenue amount over the years

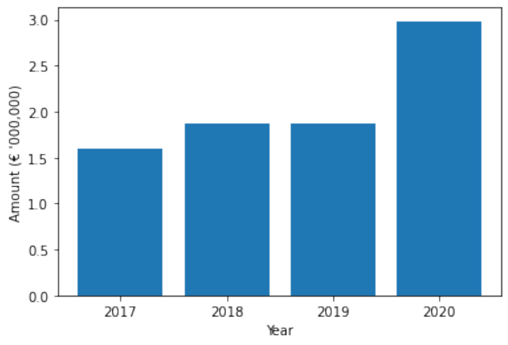

Figure 4.4: Distribution of average revenue amount of the companies over the years

When looking at figure 4.5 some seasonality might be present but most noticeable is the revenue in January, which is much higher than in any other month. When averaging this data on the number of companies that reported revenue in that month, the trend does not change a lot. Most noticeably, the peak in January came down a bit. The peak in January can be explained by two companies that reported all their revenue in January. The new distribution can be seen in figure 4.6.

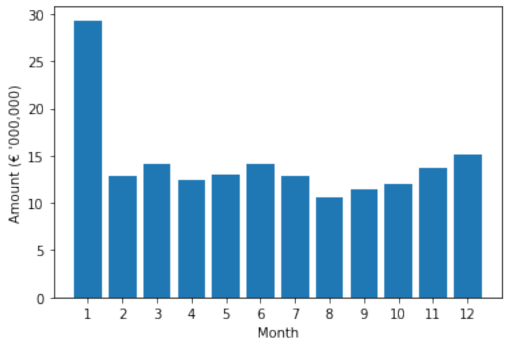

Figure 4.5: Distribution of total revenue amount over the months

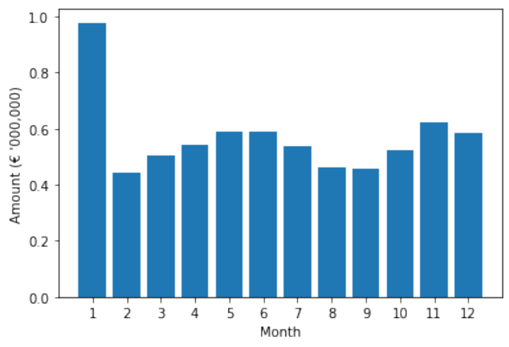

Figure 4.6: Distribution of average revenue amount of the companies over the months

Lastly, in figure 4.7 it shows that just like the number of clients per company, the total amount of revenue per company is also vastly varying. This is also visible in figure 4.8, where the variance of the revenue is shown in a histogram. The cubic root of this variance had to be taken to make the figure somewhat readable, since some companies have a very high month to month variance of revenue.

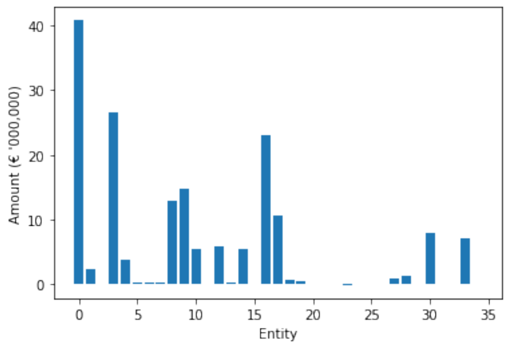

Figure 4.7: Distribution of total revenue amount over the companies

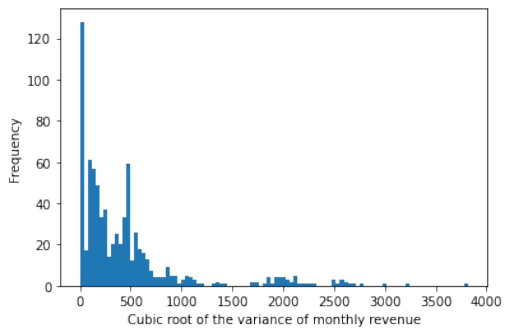

Figure 4.8: Variance of the distribution of revenue company

These trends and high variability give some insight on where outliers might be found. However, it also shows a possible pitfall of having an imbalanced data set. If there is not enough data for similar kinds of companies, the algorithms might completely classify them as outliers. Even though the behavior shown is completely normal, it is just not represented over the companies in the data set.

More information is added to this revenue data by adding features obtained from other sources where descriptions of the companies are given. Nine of these features are binary and do not differ over the months or clients within a company. In figure 4.9 the spread of these binary features has been visualized. These features all give some information about each company, like what kind of sector the company is part of. So, figure 4.9 shows what kind of companies are in the total data set. In appendix C all these features are listed, including their definition. In the appendix, two other features are listed. Namely, FTE_company and Saldo_EBITDA. FTE_company represents the Full-time equivalent (FTE) of a company in a specific year. In this definition, 1 FTE equals a full work week. Therefore, two employees working half a work week add up to 1 FTE. A histogram of the distribution of this feature is shown in figure 4.10. Saldo_EBITDA shows the total EBITDA (Earnings Before Interest, Tax, Depreciation, and Amortization) in each month per company. EBITDA is an important measure when talking about the health of a company and taking it into consideration when trying to find out whether a monthly revenue is normal could be important. EBITDA is calculated by Revenue - Cost Of Sale - Operational expenses. The visual representation of how this is distributed over the data can be found in figure 4.11.

These features are added because they give insight in the type of companies the algorithms are dealing with. Hopefully this will give them a way to group future new data with similar companies in the current data. The distinctions these types of features make can be informative on the revenue patterns the company might have and therefor what does not align with them, i.e., might be an outlier.

Finally, the underlying distribution of the summed revenue feature was research. When taking the summed amount from the fifth till the ninety-fifth percentile, the data seems to fit the Johnson's S_u distribution with a location parameter $\theta = -1.9942$, a scale parameter $\sigma = 0.6018$, and two shape parameters $\delta = 7.8589$ and $\gamma = 4.0215$. The data fitted to the Probability Density Function of this distribution can be seen in figure 4.12. Johnson's SU distribution was proposed as a variation on the normal distribution, transformed to include kurtosis and skewness into its parameters (18). This can be used to deal with asymmetric data with fatter tails than the usual normal distribution. In the paper by Choi *et al.* (19) it is shown that this distribution is good at encapsulating financial time series. In addition, Cayton *et al.* (20) proposes a two-step version of the Johnson's SU distribution, which in the same paper is proven to work well for forecasting losses of financial asset portfolios. For these reasons, it is relevant and not surprising to know that this financial data also behaves according to this distribution.

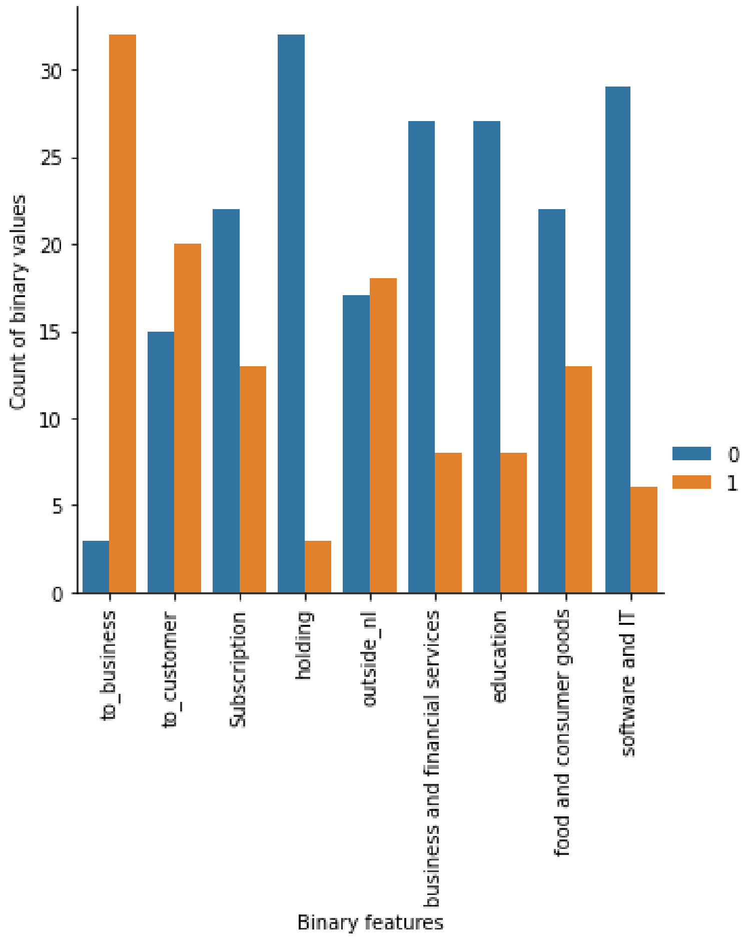

Figure 4.9: Spread of 0 and 1 values of the binary features per distinct company

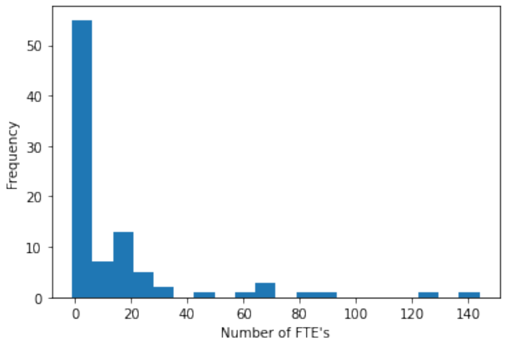

Figure 4.10: Distribution of the number of FTE's in a year per company

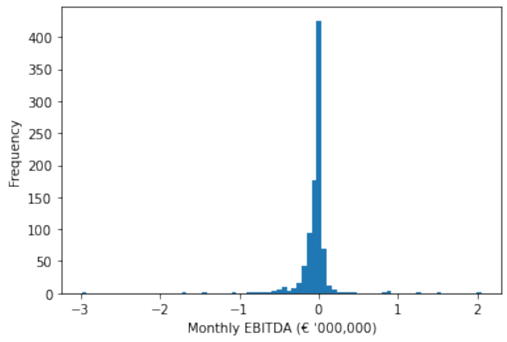

Figure 4.11: Distribution of the monthly EBITDA amount per company

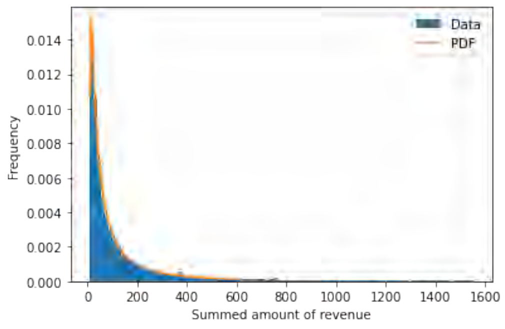

Figure 4.12: Summed amount fitted to the Johnson's S_u distribution, $\theta = -1.9942$, $\sigma = 0.6018$, $\delta = 7.8589$, $\gamma = 4.0215$

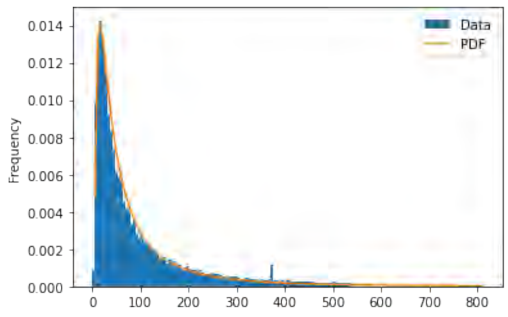

Figure 4.13: Summed amount fitted to the Johnson's S_u distribution after filled in missing values, $\theta = -2.5260$, $\sigma = 0.8208$, $\delta = 5.0398$, $\gamma = 4.3652$

4.2 Data pre-processing

To be able to effectively train the Machine Learning models, data often must be pre-processed in some kind of way. This could, for example, be pertaining to dealing with missing data or performing data normalization, but the first part of data pre-processing is always splitting the data in a train, validation, and test data set.

In this case this was done on a company level, so every company's data will be in one of the data sets completely. When splitting these companies, it was discovered that some had very little instances. It was decided to delete all companies with less than 1.000 instances since they do not have enough data to find any patterns or relations within the company. This means deleting all companies with less than 1.000 rows in the data set after grouping on month, year, and customer. In practice this means that the company deleted with the largest number of data rows had 789 rows. All these companies also meet at least one out of two requirements that make them unfit for this research. The first requirement is that the company reported all its revenue in two or less months a year. From experience this usually means that the data is not reliable in the first place. The second requirement regards the number of customers the company reports on. When the number of customers is lower than fifteen, it would be easier to do a quick manual analysis, since missing something important is less likely when all the customers can be taken into consideration. In addition, it was earlier discovered that one of the companies had a much larger number of instances than the other ones. This company was capped at around 10.000 instances by randomly selecting customers and deleting all the data of the customers that were not selected.

After this process, 44,795 rows or instances are left in the data set and these origin from only 12 companies. It should be noted that it would have been desirable to have data from more companies. This might not be enough data to properly learn from the different types of trends within the data, especially since it was discovered that the data has quite a high variability.

Another part of data pre-processing is dealing with missing values. The data is not measured at a set time interval, but a new data row is just added every time a customer pays for something that can be linked to revenue. Therefor there is no direct measure to check whether there is any missing data, and the assumption can and needs to be made that the data does not include any missing data. However, sometimes customers will not have revenue in every month, for these cases the months between their first occurrence in the data and the last will be assigned a revenue of zero. This is done to be able to better follow the trend of clients that may not have revenue in every month. However, this does add around 300 thousand zero values to the data. This changes the distribution of the data, but it still fits the Johnson's SU distribution with different parameters. This new distribution can be found in figure 4.13.

In this case the method feature engineering was a paramount part of the data pre-processing. Entity and customer id are not features that could be used to train the models. Therefor, including new features is a suitable way to add back some of the information about which transactions belong to the same customers. Adding features can also make sure that similar customers and companies get grouped together.

In appendix D the features added from transforming the summed amount of revenue are given with a description. The selection of these particular features is based on expert interviews. From those could be concluded that these features would cover all the information looked at when doing the outlier detection manually. Therefor it was concluded that this information would also be beneficial to have when training the algorithms. To give a bit more of a insight on these features table 4.1 presents some simple statistical facts about them. When looking at the distribution of all these features it shows that mean_per_client, mean_last_12_months_client, mean_last_3_months_client, percentage_of_total, normalized_amount_total, and normalized_amount_company all have very similar distributions to the summed amount, as shown in figure 4.12. One feature with some remarkable characteristics seems to be features growth_percentage_client. After looking at the distribution, but also the distribution when filtering the data on a 0.05 to 0.95 interval, it seems to contain some very high and low data points These could probably be explained by instances where a growth percentage is calculated with one of the columns approaching zero, but not being zero exactly. The 0.05 to 0.95 interval distribution is shown in figure 4.14. The peak in the middle is much higher than the other frequencies. This peak is precisely at a growth percentage of zero, which mainly represents all the instances without any revenue.

Table 4.1: Added features derived from the revenue with description

	Median	mean	std	min	max
mean_per_client	-20,34	-350,61	5002,12	-846399,00	9959,25
mean_per_company	-93,39	-347,57	1952,51	-82782,53	4641,04
number_of_clients	50919,00	45527,19	15413,54	1,00	50919,00
number_of_clients_in_year	18994,00	18400,59	7270,50	1,00	25685,00
number_of_non_zero_months_mean	10,37	10,47	3,18	1,00	43,04
growth_percentage_client	1,00	9,22E+11	2,73E+14	-5,89E+16	6,55E+16
growth_percentage_LY_client	1,00	5,37	2989,01	-229797,00	2021249,00
mean_last_12_months_client	-5,85	-243,37	4352,55	-550797,63	9110,25
mean_last_12_months_company	-729574,20	-653756,71	301092,01	-1056839,18	8772,75
mean_last_3_months_client	0,00	-288,79	5491,40	-842513,16	25130,24
mean_last_3_months_company	-773950,84	-777358,50	319930,98	-1418311,47	10298,00
total_month_of_company	-851894,72	-902212,75	371373,02	-3825786,88	17000,00
percentage_of_total	0,00	0,00	0,03	-6,68	8,47
normalized_amount_total	0,05	0,00	1,00	-245,42	11,83
normalized_amount_company	0,00	-350,61	6854,67	-1678152,54	80523,87

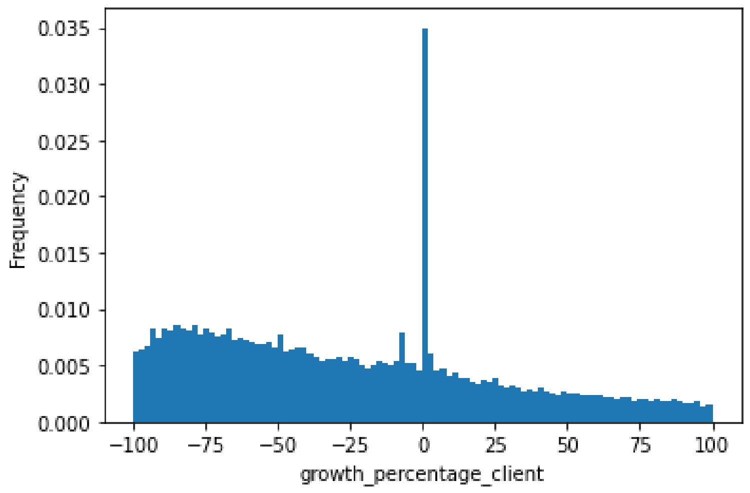

Figure 4.14: Distribution of growth percentage per client

5

Methods

5.1 Isolation Forest

The algorithm iForest (21) finds outliers by separating data instances until they isolate from all other data. This is done by building tree like structures that are called isolation trees, or iTrees for short. These iTrees have a binary structure and an example of what it could look like can be seen in figure 5.1. A data instance becomes isolated when it is an external node, which means no other instances are also part of this node. Starting in the root node with a complete data set, a binary separation is made at every node. These binary separations are made on the values of one feature per node. An iTree is done once all the data instances have been isolated. Since outliers are expected to be different than "normal" data, they will need to go through less binary separations to become isolated. Therefore, data instances that isolate closer to the root have a greater chance of being an outlier.

In the training stage of iForest multiple of these iTrees are created, also called a forest of iTrees. The iTrees are created repeatedly to account for randomness. For example, an outlier might take a long time to isolate in one tree by coincidence. To create the forest of iTrees, two input parameters need to be specified. The first being the amount of iTrees that the algorithm will create, this uses the notation t. The second parameter is the amount of data instances the algorithm will use to create one tree. This is also called the sample size and uses the notation ψ. These instances for a sample are randomly picked from the complete train data set and for every tree this is the same complete data set. So, after randomly picking a sample of size ψ and creating an iTree where all the instances in the sample are isolated, the iTree is added to a collection of trees, which in turn will be the

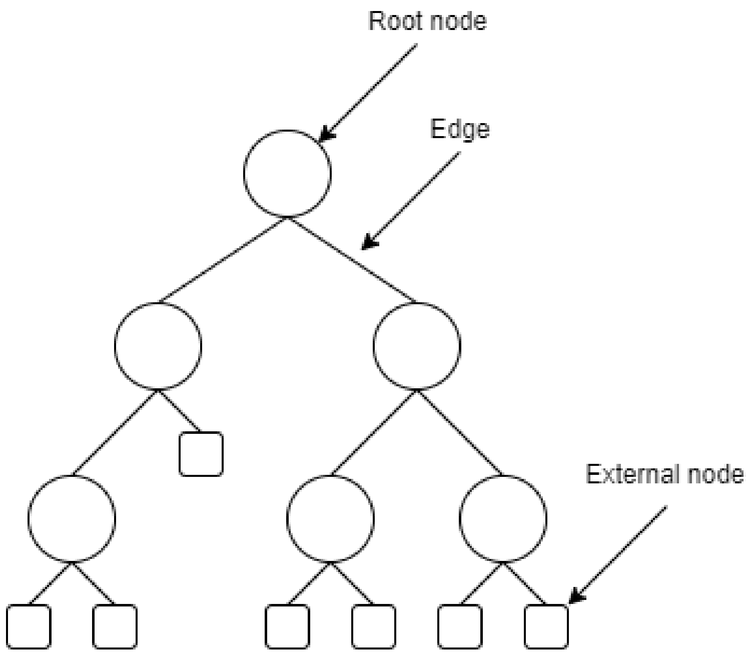

Figure 5.1: Structure of tree

iForest. The iForest is complete when it contains t iTrees. In the worst-case scenario, this process will lead to a time complexity of $O(t\psi^2)$ and a space complexity of $O(t\psi)$.

The next step in this process is the evaluation stage. During this stage, every instance in the evaluation set gets assigned an anomaly score. This is done by running them through the iTrees created in the training stage. This means every instance x will go through all the splits until it ends up in an external node. This will give a path length $h(x)$ for an instance for every iTree. The path length is the number of edges between the root node and the external node where the instance ends up in, see figure 5.1 for reference. Finding the path length for a long path can take a long time so a h_{lim} can be added as an input parameter. When the current path length equals h_{lim}, the total path length will be approximated by adding an estimation of the average path length of the remaining sub-tree to the current path length. The sub-tree in this case is part of the iTree that is reachable after the current node. In equation 5.1 the equation to estimate this average path length is given, where ψ is the number of instances of the data left at this point in the tree during training. This is also equal to the number of external nodes that are at the end of the sub-tree, since every data instance needs to be isolated. The harmonic series, $H(i)$, is used in equation 5.1 to estimate the path length. This series can be denoted as $H(i) = \sum_{n=1}^{i} \frac{1}{n}$ and be approximated by using the natural logarithm and the Euler-Mascheroni constant as shown in equation 5.2.

$$c(\psi) = \begin{cases} 2H(\psi - 1) - \frac{2(\psi-1)}{\psi} & for\ \psi > 2 \\ 1 & for\ \psi = 2 \\ 0 & otherwise \end{cases} \tag{5.1}$$

$$H(i) \approx ln(i) + \gamma,\ \gamma = Euler - Mascheroni\ constant \tag{5.2}$$

Now the path length through each iTree has been calculated for an instance, the anomaly score s can also be calculated. First calculate the average path length for the complete sample size ψ with equation 5.1. This $c(\psi)$ can be plugged into equation 5.3 together with the mean of all path lengths of an instance, $E(h(x))$. When this score is close to 1, it means that the average path length of the instance is much smaller than the overall estimated path length, $\frac{E(h(x))}{c(\psi)}$ will be small, and the instance is an anomaly. When the score is lower than 0.5, it likely is not since the average path length of the instance is larger than the overall estimated path length, and $\frac{E(h(x))}{c(\psi)}$ is larger than 1. However, if every instance in the evaluation set returns a score around 0.5, it means the set has no real anomalies and all paths have a similar length, which means that for all instances $\frac{E(h(x))}{c(\psi)} \approx 1$.

$$s(x, \psi) = 2^{\frac{-E(h(x))}{c(\psi)}} \tag{5.3}$$

The evaluation stage can be repeated for the test stage.

In this paper, the implementation of the Isolation Forest algorithm was done using the Python package sklearn.ensemble.IsolationForest, which origins from the library of scikit-learn. The paper by Lui *et al.* has concluded that $\psi = 2^8 = 256$, $t = 100$, and $h_{lim} = ceil(log_2(\psi))$ are good input parameters for experiments.

5.2 Local Outlier Factor

As it is in the name, this method of finding outliers is done by giving each instance a Local Outlier Factor.(22) This factor defines how much of an outlier a data point is. This factor is calculated by comparing the density of its neighborhood with the density of its neighbors' neighborhoods, i.e. if the neighborhood of its neighbors is much denser than its own the data point is likely to be a (local) outlier.

The basis of this algorithm lies on the distance between data points within the set, where $d(p, q)$ represents the distance between two data points p and q. For this case, the often-used Euclidean distance as described in equation 5.4 was used to calculate the LOF. When writing the Euclidean distance out for a multi-dimensional problem with n dimensions, like the one this paper discusses, it would look like the one in equation 5.5.

$$d(p, q) = \sqrt{(p - q)^2} \tag{5.4}$$

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{5.5}$$

In addition, the factor is only reliant on one parameter, namely the k. This parameter represents the number nearest neighbors a data point will (at least) have in its local neighborhood and can be any positive integer. In figure 5.2 a 2D-representation of what such a neighborhood of a data instance p would look like. In this figure a data instance o is represented as a dot on the boundary of the neighborhood. This data point is the k^{th} farthest from p. For example, if $k = 10$, like in figure 5.2, o is the data point with the tenth farthest distance but could also be the eleventh for $k = 11$. It is possible for more data points to have the exact same distance to p as o. In that case those are also part of the neighborhood of p and the neighborhood has more than k data instances. The distance between p and the data point k^{th} farthest from an object is also called the k-distance and the actual neighborhood of an object is called the k-neighborhood.

Before giving the formula for the LOF, these terms need to be formally defined. Firstly, about the k-distance of an object p in the data set, k-distance(p), two things are true:

1. At least k other objects have a distance equal to or smaller than it from p.

2. At most $k - 1$ objects have a distance smaller than it from p.

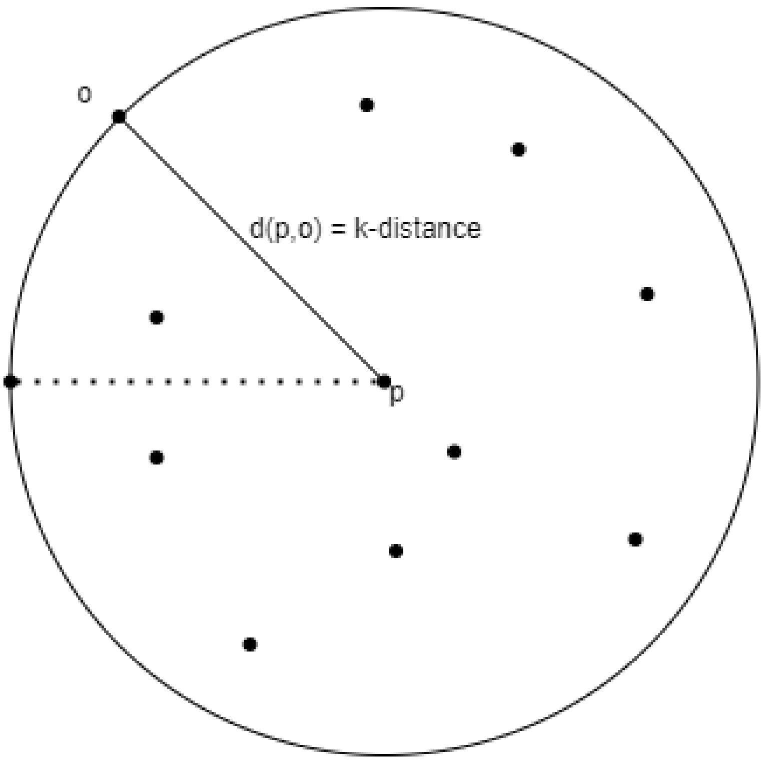

Figure 5.2: 2D representation of the k-distance for a data instance p

Secondly, the k-neighborhood of p is denoted as $N_k(p)$ and includes the points not farther away from p than the previously discussed k-distance. Every data point within this neighborhood is what is called the k-nearest neighbors of p.

The following part is to elaborate some term needed to calculate the LOF and describe the coinciding equations.

Reachability distance This distance represents the maximum between the actual distance between two data points and the k-distance of one of them. Formally, $reach - dist_k(p, o)$ is the reachability distance of p with respect to object o and is given by $reach - dist_k(p, o) = max\{k - distance(o), d(p, o)\}$. This is also made visual in figure 5.3

Local reachability distance This is the inverse average of the reachability distance of an object p with respect to each of its k-nearest neighbors. This can also be referred to as $lrd_k(p)$ and is given in equation 5.6.

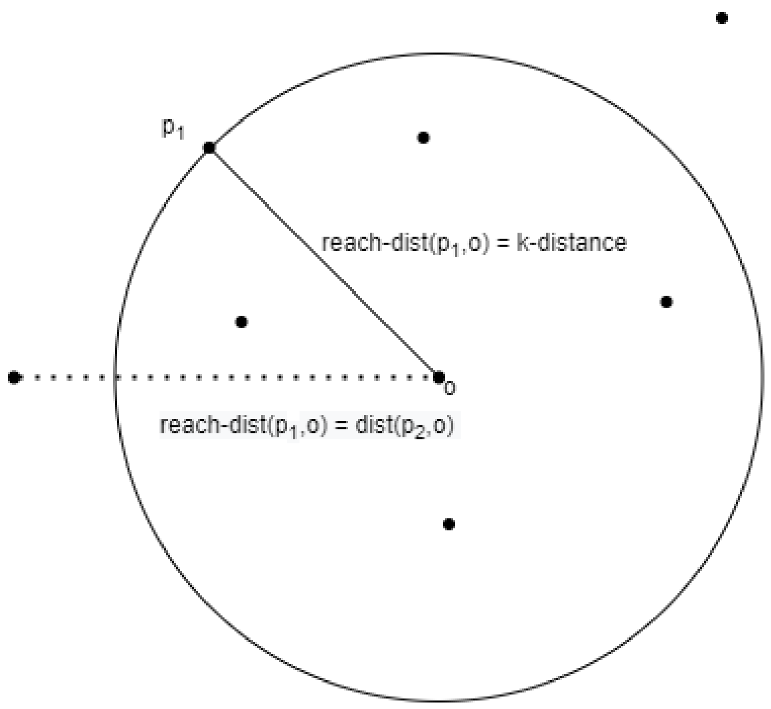

Figure 5.3: 2D representation of the reachability distance

$$lrd_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} reach - dist_k(p, o)} \tag{5.6}$$

Local outlier factor The formula where this algorithm got its name from is given in equation 5.7. This calculates the average of the ratio between $lrd_k(o)$ and $lrd_k(p)$, where o is an object in the neighborhood of p.

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \tag{5.7}$$

Find the official definitions of what is described above as defined in the original paper by Breunig *et al.* in appendix B

This is all implemented using the sklearn.neighbors.LocalOutlierFactor package in Python, also originating from the scikit-learn library. In addition, the paper by Breunig *et al.* describes that a k between 10 and 20 works well for most cases. For this reason, it was set to 20 within this experiment.

5.3 One-Class Support Vector Machine

OCSVM (23) is an outlier detection algorithm derived from the SVM algorithm. SVM is an algorithm that was extensively research by Vladimir Vapnik and was first introduced in the paper by Boser *et al.* (24). SVM can be used for either regression or classification problems and is inherently a supervised algorithm. OCSVM is a version of SVM classification, modified in a way to be able to classify data into a normal class in an unsupervised manner. If a data point cannot be classified into the normal class, it is seen as an outlier.

SVM classifies data by creating a hyperplane to separate the data from one class from the other data, i.e., creating two classes. The algorithm can be modified to be able to recognize multiple classes by repeating this for each class. A hyperplane is like a line but with more dimensions, or a multi-dimensional space. In the case of SVM the number of dimensions a hyperplane is equal to the number of features the data has minus one.

The goal of the SVM is not only to separate the class but also to do this in the most optimal way possible. This is done by maximizing the distance between the data points closest to the hyperplane and the hyperplane itself. This is called the max margin. These data points which help decide the location of the hyperplane are called support vectors.

To translate this into a one class problem, the training data into a feature space of higher dimension, which is done using a kernel $k(x, y)$ to project the data into the feature map ϕ. After doing this, most data can and needs to be separated from the origin of this feature space by creating a hyperplane. The data that cannot be separated by this hyperplane are seen as outliers. This hyperplane can be found by solving the quadratic program in equation 5.8.

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}||w||^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho \\ \text{s.t.} \quad & (w * \phi(x_i))\rho - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{5.8}$$

In this equation, $i \in [l]$, where l is the number of instances in the training set. This makes ξ_i the slack variable of a certain data point. The slack variable causes some points to be able to be in the margin separating the hyperplane and the support vectors. This variable is desired to be as small as possible.

When tuning the OCSVM algorithm for this research, $\nu \in (0, 1)$ was one of the most important ones. This parameter represents the trade-off between the number if support vectors the hyperplane will use and the number of outliers the algorithm will allow. By the agency of it being the upper bound of the fraction of outliers, as well as the lower bound of the fraction of support vectors. This trade-off simultaneously represents the two goals of creating the hyperplane in an OCSVM. Namely, separating most data points from the origin and doing this with a large margin from the hyperplane.

If the optimization problem in equation 5.8 can be solved for w and ρ then a signum function $f(x)$ 5.9 can be derived. This function will be $+1$ for all data points contained by the hyperplane and -1 for all other data points. This function can be used to decide whether new data is classified as an outlier under the previously trained OCSVM.

$$f(x) = sgn((w * \phi(x)) - \rho) \tag{5.9}$$

The standard formula of a kernel is given in equation 5.10. In this research, the Gaussian kernel, as shown in equation 5.11, was used to map the data. This kernel is proven to have the best performance when it comes to OCSVM in a paper by Bounsiar *et al.* (25).

$$k(x, y) = (\phi(x) * \phi(y)) \tag{5.10}$$

$$k(x, y) = e^{-||x-y||^2/c} \tag{5.11}$$

To implement to OCSVM algorithm, the sckikit-learn library was used once again, this time with the usage of the sklearn.svm.OneClassSVM package.

6

Experimental setup

6.1 Data split

Data collection was done manually by going through different projects to get the right data from a variety of companies. Then the data was rewritten to CSV format so they could be put into a pandas DataFrame.

After this, the data is split into three sets. While doing this, a company is always complete in one data set and not split up into multiple sets. The first set used is the train data set; this is used to let the algorithms decide what outliers are. They will all make their own set of rules on how to classify an instance as an outlier or not. This data set contains 6 companies and 29.058 instances. Next, the evaluation set is used to see how the algorithms perform on completely new data. After taking the results of the evaluation set in considerations, the algorithms can still be modified and re-trained on the train set. This set contains 3 companies and 15.737 instances. Lastly, the test set should not be looked at until the very end and at that point the algorithms and data should not be altered any further. This data set is used to test how well the final algorithms perform. This data set also contains 3 companies and consists of 16.573 instances.

6.2 Implementing algorithms

The next part is implementing the described algorithms. This was all done using multiple Python libraries but the most important one was sklearn. Within these algorithms some parameters need to defined. The parameters and their chosen values are shown in table 6.1. Most of these values were already given in the methods chapter and these values are based on prior research where they have been proven to work well. The only parameter that was not assigned a value yet was ν for the OCSVM algorithm. The value 0.1 was chosen based on the average fraction of outliers defined by experts within these and similar data sets. These algorithms all have built-in scoring functions modelled after the scoring techniques described in the methods chapter of this thesis. These scores are how these algorithms show how much of an outlier they think data is. The algorithms are first trained, on the train data. For the test set the outlier scores are obtained by running the trained algorithms on the set. The scores are then taken as the results and must be evaluated.

It was also research how important a feature in the data set was to the result. For Isolation Forest, this was done using a Python library called SHAP and for the other two algorithms this was done using the function permutation_importance. This function, similar to the score function, is built-in to the sklearn library of these specific functions. This resulted in a top 4 most important features per algorithm.

Table 6.1: Chosen parameter for the different algorithms

IF	LOF	OCSVM
$\psi = 2^8 = 256$	$k = 20$	$\nu = 0.1$
$t = 100$		
$h_{lim} = ceil(log_2(\psi))$		

6.3 Evaluation

To evaluate the results, first a global overview of the patterns of the algorithms was created. This was done using heatmaps. These were created per company and were separated by month on the horizontal axis, one month per row, and by client on the vertical axis, one client per column. This particular cross section of the multi-dimensional data was chosen to mimic the way the data is used in the current situation. Each algorithm can give a data point a form of an outlier score. The heatmaps are used to make these scores readable for large amounts of data. This causes a comprehensible view of how outliers are spread throughout the data. The heatmaps have a scale from green to red. This represents the scale of a low outlier score to a high one, respectively. Since the three algorithms all gave quite varying intervals of outlier scores, the scores were all standardized to fit into a 0 to 1 scale. Also, the clients are ordered from highest to lowest revenue. These heatmaps are provided to give a first glance on how the algorithms center the outliers in the different companies.

Thereafter, the same set of data were reviewed by a couple of field experts, in this case these were Financial Due Diligence Consultants. The data is setup with the same cross section as for the heatmaps, but instead of colors the cells contain the amount of revenue. They have flagged the parts of the data they would have paid extra attention to if they would be to use the data in a Due Diligence project. These results they were used to score the algorithms on both precision as well as recall.

First, the average monthly outlier score was calculated. This resulted in some pronounced dips, meaning likely outliers. These dips were then scored against the months the experts identified, using both recall and precision.

Second, a similar analysis was made on a per client level. However, in this case it was harder to find a clear cutoff point between likely outliers and likely normal occurrences. To still be able to score the algorithms on this, the top 5, 10, and 30 most likely outliers were taken and compared to the expert results. The recall and precision were calculated here too but, in this case, it should be noted that the number of outliers the experts flagged were higher than 10 and lower than 30 for all companies. Therefore, the top 5 and 10 could never get a perfect precision.

Lastly, a second set of expert interviews were conducted. The experts were given the same source data but this time 30 occurrences, per company, were highlighted and the experts were asked whether they agreed if these were outliers. One occurrence refers the a specific month of a single client. The 30 occurrences given to the experts were the combined top 10 most likely outliers defined by the three algorithms. The choice was made to evaluate the results in this was because it was again hard to draw a line between what were and were not outliers according to the algorithms. In addition, when looking at specific occurrences, the results of the first interviews did not always agree completely person per person. When doing the interviews like this the results were almost identical. Since this technique still does not give a set of definitive outliers only the precision could be calculated.

7

Results

In this chapter, the first part will be giving high-over analysis of the trends of what is classified as likely to be an outlier between the different algorithms and companies. In addition, the differences in feature importance between the different the algorithms are discussed. After, the same type of analysis will be given on the results of the expert interviews. Lastly, a more detailed and score-based comparison will be made between the results given by the algorithms and the results from the expert interviews.

7.1 Pattern analysis of the results from the implemented algorithms

In the figures E.1, E.2, and E.3, as part of appendix E, the results of testing the algorithms on three different companies in the form of heatmaps are shown.

Analyzing the results of Company A, as seen in figure E.1, leads to the following observations. In this case, Isolation Forest ranks the clients with the highest total revenue very high on the outlier scale for every month. However, when a client's revenue is less than its monthly average, that month will score a bit lower. These are often the important clients to look at when looking for anomalies in the monthly revenue data. Mostly because when these clients change their habit, the relative impact on the company will be considerably higher. In this case, out of 152 clients, the top 5 clients make up almost 50% of the client bound revenue.

In addition, some slight outlier behavior is flagged around the last quarter of every year. This could be an identifier of some sort of seasonality. Also, the year 2021, which is the last year in the available data for this company, is leaning towards the outlier side of the scale.

Local Outlier Factor has a completely different result. Although it also seems to flag the last quarters of a year, it does not look at the clients with a high revenue as outliers. They seem to be classified as the most normal data. In the first year of the data, and especially when looking at the clients with a lower revenue, a group of some data flagged as slight outliers can be observed.

Lastly, the last row seems to be varying heavily between inliers and outliers. One explanation for this could be that this client had months with positive revenue as well as months where the revenue is negative. This is generally very uncommon. This variation between inliers and outliers can also be seen in the classification done by Isolation Forest yet less prominent.

The results of One-Class SVM algorithm shown in the heatmaps however do not lead to easily analyzable patterns for any of the companies in the test data.

Although highlighting different months for each company, LOF also seems to focus on a seasonality pattern for Company B and C. For Company B it signals the whole month of January for each year as having a lot of highly outlying data and from December till April as being at least moderately outlying. Meanwhile, the other months are leaning more green. Once again, the customers with a higher revenue veer more towards being inliers. A closely related pattern can be seen for Company C.

The algorithm Isolation Forest seems to act a specific way throughout the different companies as well. Like with Company A, it seems to locate the most outliers within the clients with a relatively higher revenue for both Company B and C. In addition, both companies do seem to have a seasonality pattern within their outliers similar to the ones LOF identified. Lastly, for Company C, the last year in the data set is also seen as having more outliers than the other years. This could be due to is being an incomplete year, since this was also the case for Company A.

For each company the 10 occurrences that were seen as most likely to be outliers had no overlap between the algorithms. When looking at the clients that were given as most likely to be outliers there however was some overlap. In tables 7.1, 7.2, and 7.3 the fraction of overlap between the different algorithms for each company is shown for the top 5, 10, and 30 clients given to be most likely to be outliers.

Lastly, the feature importance analyses of the three algorithms gave rather differing results. The four most important features per algorithm can be found in table 7.4. Some noticeable differences can be named. One being the fact that three of the four features identified as important for IF are company specific binary features, namely to_customer, subscription, and education. Next, the important features for the OCSVM model all say something about revenue amount with only total_month_of_company regarding a company wide revenue comparison. The other three features all tell something about the amount of revenue of each client. On the other hand, the important features for LOF turned out to be of a more varying range. Two of the features being related to revenue with on being client specific, growth_percentage_client, and one being month and company specific, total_month_of_company. Of the other two features one is a year and company specific continuous value, and the other is a company specific binary, Saldo_EBITDA and outside_nl respectively. From IF the feature importance per instance could be derived using SHAP. From the top 10 most likely outliers per company the five most important features were summarized, and these results are shown in table 7.5.

Table 7.1: Overlap between top results per client of the different algorithms for Company A

	Top 5			Top 10			Top 30		
	IF	LOF	OCSVM	IF	LOF	OCSVM	IF	LOF	OCSVM
IF	1,00	0,00	0,00	1,00	0,00	0,00	1,00	0,07	0,30
LOF	0,00	1,00	0,00	0,00	1,00	0,00	0,07	1,00	0,17
OCSVM	0,00	0,00	1,00	0,00	0,00	1,00	0,30	0,17	1,00

Table 7.2: Overlap between top results per client of the different algorithms for Company B

	Top 5			Top 10			Top 30		
	IF	LOF	OCSVM	IF	LOF	OCSVM	IF	LOF	OCSVM
IF	1,00	0,00	0,00	1,00	0,00	0,00	1,00	0,13	0,03
LOF	0,00	1,00	0,00	0,00	1,00	0,00	0,13	1,00	0,10
OCSVM	0,00	0,00	1,00	0,00	0,00	1,00	0,03	0,10	1,00

Table 7.3: Overlap between top results per client of the different algorithms for Company C

	Top 5			Top 10			Top 30		
	IF	LOF	OCSVM	IF	LOF	OCSVM	IF	LOF	OCSVM
IF	1,00	0,00	0,20	1,00	0,00	0,20	1,00	0,17	0,43
LOF	0,00	1,00	0,00	0,00	1,00	0,20	0,17	1,00	0,50
OCSVM	0,20	0,00	1,00	0,20	0,20	1,00	0,43	0,50	1,00

Table 7.4: Top 4 most important features per algorithm

IF	LOF	OCSVM
to_customer	growth_percentage_client	mean_per_client
Subscription	total_month_of_company	mean_last_12_months_client
education	Saldo_EBITDA	total_month_of_company
mean_per_company_scaled	outside_nl	mean_per_client_not_zero

Table 7.5: Feature importance of IF for the top 10 results per company

	Company A	Company B	Company C	Total
normalized_amount_total	10	10	10	30
number_of_non_zero_months	10	10	10	30
FTE_company	0	0	1	1
total_month_of_company	0	5	9	14
growth_percentage_LY_client	9	0	0	9
Month	8	9	9	26
percentage_of_total	0	10	9	19
softwareandIT	3	1	1	5
mean_last_12_months_company	10	2	0	12
Saldo_scaled	0	0	1	1
to_customer	0	3	0	3

7.2 Comparative results

In the next section the results that were given by the algorithms are compared to the results obtained from the expert interviews. Some generalizations about the types of comments that were often made about the three companies were made. These are noted here:

- The comments were mainly made about the customers with the highest average revenue
- The company wide seasonal patterns of peaks and drops of the revenue flow.
- Large clients that start to bring in less revenue or have brought in no revenue in the last months. This could signal a big client is leaving or has already left, which could cause in a big drop in total revenue.
- A new client with a high monthly revenue. It is interesting to know if this trend will continue since it is the start of an increase of total revenue.

- A seemingly one-off client with a high revenue in that one month. This could also be the start of a new client, e.g., this client might not get invoiced monthly.
- Clients with a high total revenue with a different month-to-month pattern than most other high paying customers. This could be that they pay in a different month or that they might not pay monthly while others do.

In the expert interviews, 14 out of Company A's 152 clients were flagged as containing something important or, in other words, as containing outliers. For Company B, the experts flagged 15 out of 210 and for Company C this number was 20 out of 72. In table 7.9 it is shown how many of those clients were classified as most likely to be an outlier by the different algorithms. In the table this is divided in to the Top 5 most likely as well as the Top 10 and 30. From this table IF works best for all three companies, with the best performance for Company A.

Next, the experts flagged some months as overall containing outliers. These months can be found in figure 7.1 as highlighted in yellow and green. The yellow months are what they defined as the seasonal patterns and the green would be any other months containing outliers that they did not see as seasonality.

The other columns are showing the results of the algorithms. Where the red highlighted months represent all the months scoring lower than 0.4 on average, which means they are likely to be outliers. This value was chosen since this value seemed to represent a good split between the months with a peak outlier score and the months where the outlier score dipped. A second color coding was added for IF, since it seemed to have a second set of, slightly higher, dips in the score. These months were given an orange color.

Lastly, the most notable thing when looking at OCSVM is that it sees all months in Company C as likely outliers. This is cause by all but one month having an average score below 0.002, while the remaining month has a score close to one. The dips that follow after adjusting can be found in the last column.

The Recall and Precision of these results can be found in the tables 7.6 and 7.7. This includes the split between yellow and green months in Company A as well as the discussed adjustments made to the results of IF and OCSVM. The best performing algorithms per test case are given in bold text.

This shows that LOF performs best in most cases but IF also performs well in some cases.

Lastly, the 10 occurrence that were classified as most likely to be outliers, per company and per algorithm, were shown to a different set of experts and their opinion was used to score these outcomes. Since the 10 occurrences had no overlap between the different algorithms, each company had 30 occurrences that were shown to the experts. The precision that was calculated for these results are shown in table 7.10. This able demonstrates that, just like the per client results, IF performs best. OCSVM gave no correctly identified outliers in any of the companies and LOF did not either for two of the companies.

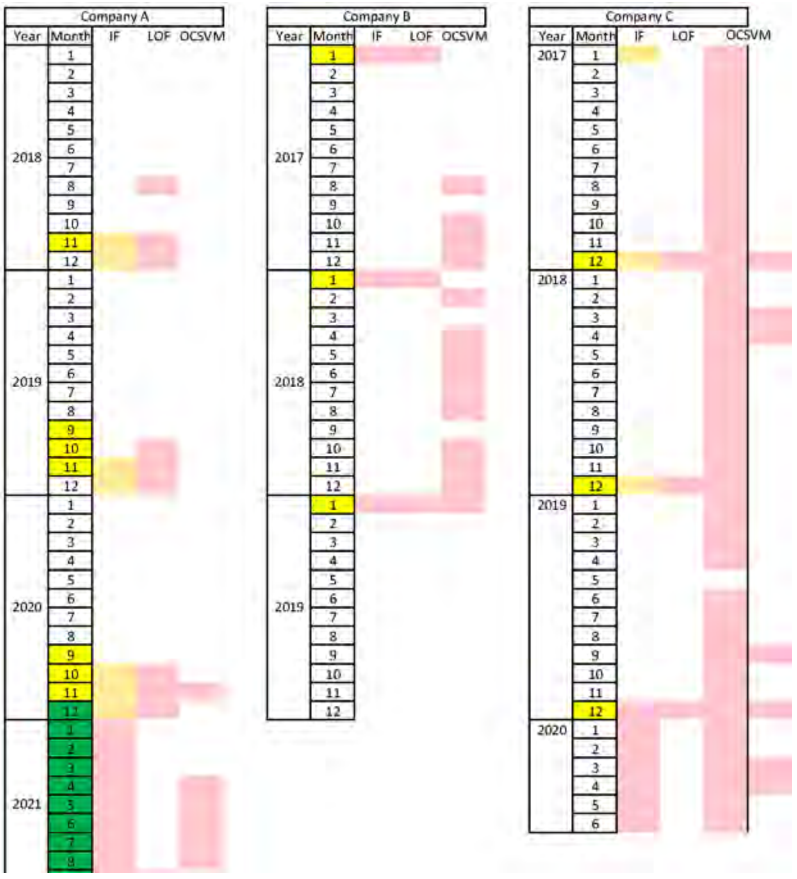

Figure 7.1: Monthly results per company and algorithm

Table 7.6: Top results per month compared to expert opinions, Company A

		IF	IF - Adj.	LOF	OCSVM
Recall	Green (seasonality)	0,90	**1,00**	0,20	0,50
	Yellow (other)	0,00	0,57	**0,71**	0,14
	Total	0,53	**0,82**	0,41	0,35
Precision	Green (seasonality)	**1,00**	0,63	0,20	0,42
	Yellow (other)	0,00	0,25	**0,50**	0,08
	Total	**1,00**	0,88	0,70	0,50

Table 7.7: Top results per month compared to expert opinions, Company B and C

	Company B			Company C				
	IF	LOF	OCSVM	IF	IF - Adj.	LOF	OCSVM	OCSVM - Adj.
Recall	**1,00**	**1,00**	0,33	0,33	**1,00**	**1,00**	**1,00**	0,67
Precision	**1,00**	**1,00**	0,07	0,14	0,30	**1,00**	0,07	0,29

Table 7.8: Top results per client compared to expert opinions scored on recall

	Company A			Company B			Company C		
	IF	LOF	OCSVM	IF	LOF	OCSVM	IF	LOF	OCSVM
Top 5	**0,22**	0,00	0,00	**0,07**	0,00	0,00	**0,15**	0,00	0,10
Top 10	**0,57**	0,00	0,00	**0,27**	0,00	0,00	**0,25**	0,00	**0,25**
Top 30	**0,93**	0,07	0,14	**0,67**	0,00	0,00	**0,50**	0,20	0,40

Table 7.9: Top results per client compared to expert opinions scored on precision

	Company A			Company B			Company C		
	IF	LOF	OCSVM	IF	LOF	OCSVM	IF	LOF	OCSVM
Top 5	**0,80**	0,00	0,00	**0,20**	0,00	0,00	**0,60**	0,00	0,07
Top 10	**0,80**	0,00	0,00	**0,40**	0,00	0,00	**0,50**	0,00	0,17
Top 30	**0,43**	0,03	0,07	**0,33**	0,00	0,00	**0,33**	0,13	0,27

Table 7.10: Precision scores of the 10 most likely outliers, per algorithm, per company

	Company A			Company B			Company C		
	IF	LOF	OCSVM	IF	LOF	OCSVM	IF	LOF	OCSVM
Precision	**1.0**	0.5	0	**0.8**	0	0	**0.7**	0	0

8

Conclusion and Discussion

In conclusion, the three algorithms seem to behave similar in all three companies discussed in this paper. They seem to prioritize the same kind of patterns when looking for outliers in different sets of data, all the while still creating some sort of uniqueness with them. Namely, LOF will find some sort of different seasonal pattern for each company. IF also finds seasonality in all three companies, with Company B having the most pronounced pattern. It also classifies centers a lot of outliers around the clients with the highest revenue. OCSVM classifies outliers as being scattered, seemingly without a distinct pattern.

In addition, IF performs the overall best on every test, except for when looking at the seasonal pattern of Company A. LOF is also quite good at finding the seasonal patterns, especially for Company B and C. However, when looking at the per client and per occurrence results, all but IF perform very poorly. Therefore, this algorithm has the most potential in this case of unsupervised outlier detection.

This gives the answer to the research question of "Which of the researched algorithms used to perform an unsupervised outlier detection on client revenue data yields the result which most closely resembles the outlier detection performed by FDD consultants?". In most cases IF most closely resembles the outlier detection performed by the FDD consultants on client revenue data.

This thesis ended up not including a lot of different companies, which might have caused the data to not have enough diversification within every set of data. Continuing this research or performing similar experiments in the future could definitely benefit from having a broader set of companies to test, validate, and train on. With this, it is not only important to have different types of companies, but also collecting companies with similar characteristics.

In addition, it would be helpful to perform a semi-supervised outlier detection, so the algorithms have some sort of idea of what outliers could look like. This can also help a lot during the validation stage to be able to easier identify which algorithms are still performing poorly. If this was spotted earlier on in the process, OCSVM and LOF could maybe have been tweaked in a way that they would have shown a better performance.

Moreover, this thesis relied highly on what information experts use to make the same analysis, for example during feature engineering. This might not be the best approach and therefore more time can be put into researching literature on effective feature engineering for similar problems.

Lastly, the algorithms all have very different features that are most important to the final outlier score they give to data. This might mean different things and therefore it might be effective to re-evaluate which features are considered, especially for the worse performing algorithms.

Appendix A

List of abbreviations

M&A	Mergers and acquisitions
FDD	Financial Due Diligence
TS	Transaction Services
BI	Business Intelligence
PI	Performance Improvement
PR	Precision-Recall
ROC	Receiver Operating Characteristics
AUC	Area Under the Curve
iForest	Isolation Forest
iTree	Isolation Tree
LOF	Local Outlier Factor
OCSVM	One Class Support Vector Machines

Appendix B

Local Outlier Factor definitions

(22)

Definition 1 (k-distance of an object p)**.** For any positive integer k, the k-distance of object p, denoted as $k - distance(p)$, is defined as the distance $d(p, o)$ between p and an object $o \in D$ such that:

i. for at least k objects $o' \in D|\{p\}$ it holds that $d(p, o') \leq d(p, o)$

ii. for at most $k - 1$ objects $o' \in D|\{p\}$ it holds that $d(p, o') < d(p, o)$

Definition 2 (k-distance neighborhood of an object p)**.** Given the k-distance of p, the *k-distance neighborhood of* p contains every object whose distance from p is not greater than the k-distance, i.e. $N_{k-distance(p)}(p) = \{q \in D/\{p\}|d(p, q) \leq k - distance(p)\}$.
These objects q are called the k-nearest neighbors of p.

Definition 3 (reachability distance of an object p w.r.t. object o)**.** Let k be a natural number. The *reachability distance* of object p with respect to object o is defined as
$reach - dist_k(p, o) = max\{k - distance(o), d(p, o)\}$

Appendix C

Explanation of extra features related to the companies

Feature name	description
FTE_company	Total of FTE in a particular year of a single company
FY	Binary, does this year have a full year of data for this company
software and IT	Binary, does this company fall in this sector
education	Binary, does this company fall in this sector
business and financial services	Binary, does this company fall in this sector
food and consumer goods	Binary, does this company fall in this sector
to_business	Binary, is this company's target clientele other businesses
to_customer	Binary, is this company's target clientele individual customers
Subscription	Binary, is this company's business model subscription-based
holding	Binary, is this company the holding of a group of companies
outside_nl	Binary, does this company do business outside the Netherlands (does not need to be exclusively)
Saldo_EBITDA	Total EBITDA (Earnings Before Interest, Tax, Depreciation and Amortisation) value of the company in this month (EBITDA = Revenue - Cost Of Sale - Operational expenses)

Appendix D

Explanation of extra features derived from the revenue

Feature name	description
mean_per_client	Mean revenue of a particular client of a company over all months
mean_per_client_not_zero	mean_per_client, but without the months where there was no revenue and than some
mean_per_company	Mean revenue of a company over all months
number_of_non_zero_months	Describes how many months a client had revenue over all months
number_of_non_zero_months_mean	Mean of number_of_non_zero_months over all clients per company
growth_percentage_client	Increase in revenue of a client compared to the previous month
growth_percentage_company	Increase in revenue of a company compared to the previous month
growth_percentage_LY_client	Increase in revenue of a client compared to the same months last year
growth_percentage_LY_company	Increase in revenue of a company compared to the same months last year
mean_last_12_months_client	Mean monthly revenue over the last 12 months of a particular client of a company
mean_last_12_months_company	Mean monthly revenue over the last 12 months of a company
mean_last_3_months_client	Mean monthly revenue over the last 3 months of a particular client of a company
mean_last_3_months_company	Mean monthly revenue over the last 3 months of a company
percentage_of_total	What percentage of the total revenue of the company came from this client in this month
normalized_amount_client	Normalize the revenue data per client
normalized_amount_company	Normalize the revenue data per company
normalized_amount_total	Normalize the revenue data over all rows

Appendix E

Heatmap results of implemented algorithms

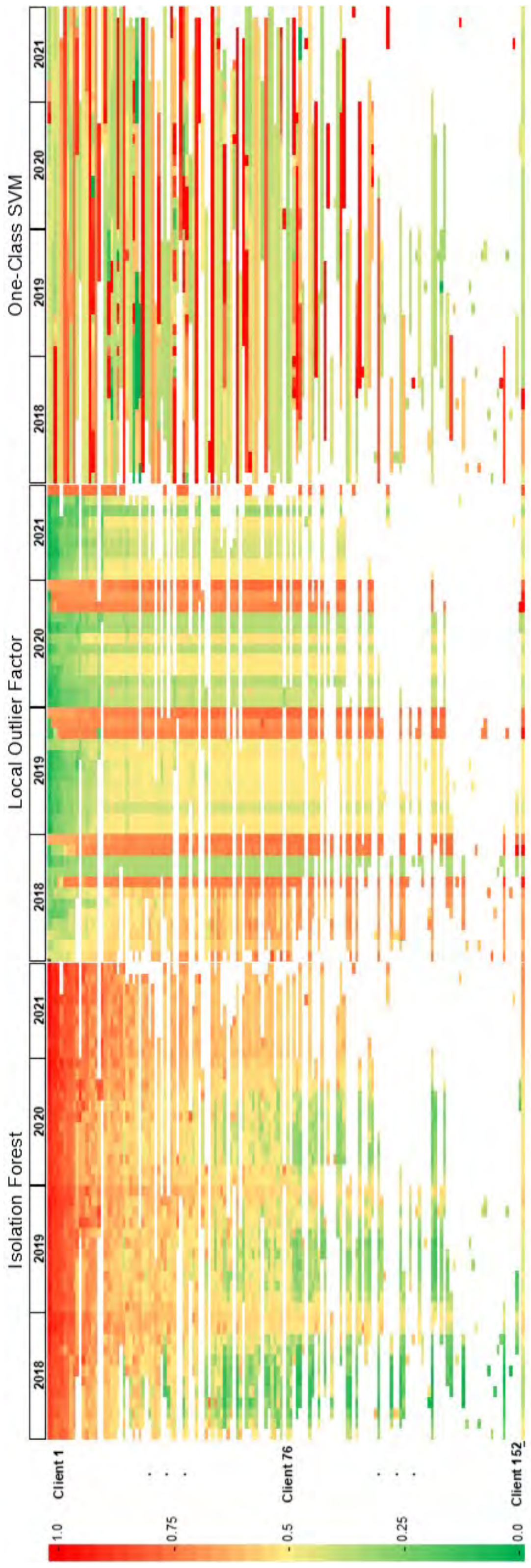

Figure E.1: Results Company A, rows ordered by total revenue of a client

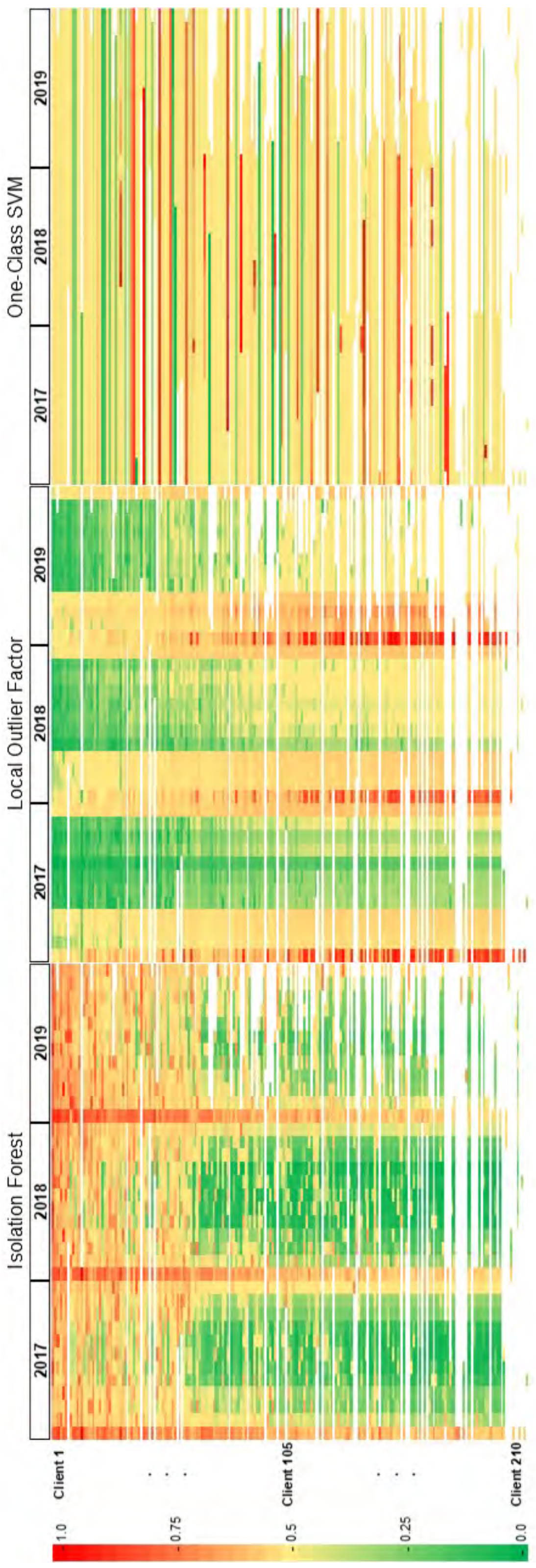

Figure E.2: Results Company B, rows ordered by total revenue of a client

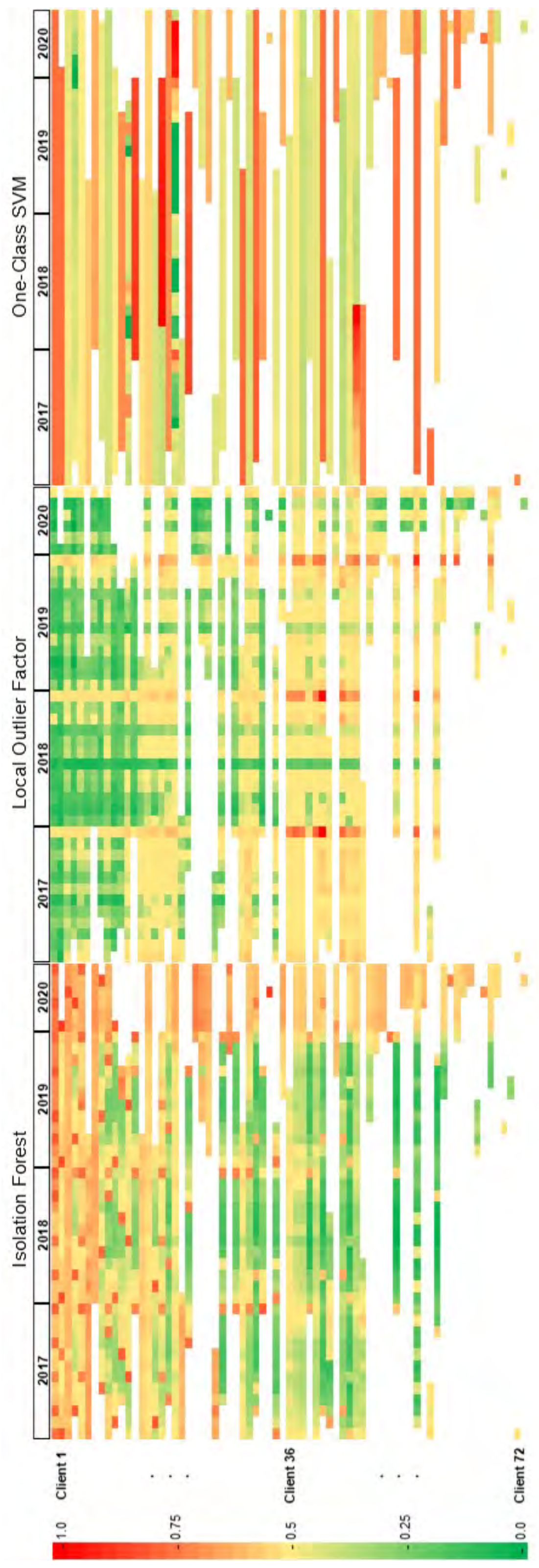

Figure E.3: Results Company C, rows ordered by total revenue of a client

References

[1] Jeffrey Berkman. *Due Diligence and the Business Transaction: Getting a Deal Done*. Apress, 1st ed. edition, 2014. 1

[2] Thorsten Feix. *End-to-End MA Process Design*. Springer Medizin Verlag, Heidelberg, Duitsland, 1st ed. 2020 edition, 2020. 1

[3] M. I. Jordan and T. M. Mitchell. **Machine learning: Trends, perspectives, and prospects**. *Science*, **349**(6245):255–260, 2015. 1

[4] Andreas Holzinger, Peter Kieseberg, Edgar Weippl, and A Min Tjoa. **Current Advances, Trends and Challenges of Machine Learning and Knowledge Extraction: From Machine Learning to Explainable AI**. *Lecture Notes in Computer Science*, pages 1–8, 2018. 1

[5] Puneet Mathur. *Machine Learning Applications Using Python: Cases Studies from Healthcare, Retail, and Finance*. Apress, 1st ed. edition, 2018. 1

[6] Xiaogang Su and Chih-Ling Tsai. **Outlier detection**. *WIREs Data Mining and Knowledge Discovery*, **1**(3):261–268, 2011. 2

[7] Aaron Plasek. **On the Cruelty of Really Writing a History of Machine Learning**. *IEEE Annals of the History of Computing*, **38**(4):6–8, 2016. 5

[8] A. L. Samuel. **Some Studies in Machine Learning Using the Game of Checkers**. *IBM Journal of Research and Development*, **3**(3):210–229, 1959. 5

[9] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An Introduction to Machine Learning*. Springer Publishing, New York, Verenigde Staten, 2019. 6

[10] Emre Celebi. *Unsupervised Learning Algorithms*. Springer Publishing, New York, Verenigde Staten, 1 edition, 2018. 6

[11] Xiaochun Wang and Xiali Wang. *New Developments in Unsupervised Outlier Detection*. Springer Medizin Verlag, Heidelberg, Duitsland, 1st ed. 2021 edition, 2021. 6

[12] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. **A comparative evaluation of outlier detection algorithms: Experiments and analyses**. *Pattern Recognition*, **74**:406–421, 2018. 8

[13] Qian M. Zhou, Lu Zhe, Russell J. Brooke, Melissa M. Hudson, and Yan Yuan. **A relationship between the incremental values of area under the ROC curve and of area under the precision-recall curve**. *Diagnostic and Prognostic Research*, **5**(1), 2021. 8

[14] Atiq ur Rehman and Samir Brahim Belhaouari. **Unsupervised outlier detection in multidimensional data**. *Journal of Big Data*, **8**(1), 2021. 9

[15] Mohiuddin Ahmed, Nazim Choudhury, and Shahadat Uddin. **Anomaly Detection on Big Data in Financial Markets**. *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017. 9

[16] Jellis Vanhoeyveld, David Martens, and Bruno Peeters. **Value-added tax fraud detection with scalable anomaly detection techniques**. *Applied Soft Computing*, **86**:105895, 2020. 10

[17] Caleb Mensah, Jan Klein, Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei. **Detecting Fraudulent Bookings of Online Travel Agencies with Unsupervised Machine Learning**. *Lecture Notes in Computer Science*, pages 334–346, 2019. 10

[18] N. L. Johnson. **Systems of Frequency Curves Generated by Methods of Translation**. *Biometrika*, **36**(1/2):149, 1949. 15

[19] Pilsun Choi and Kiseok Nam. **Asymmetric and leptokurtic distribution for heteroscedastic asset returns: The SU-normal distribution**. *Journal of Empirical Finance*, **15**(1):41–63, 2008. 15

[20] Peter Julian Cayton and Dennis Mapa. **Time-varying conditional Johnson SU density in value-at-risk (VaR) methodology**. 01 2012. 15

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. **Isolation-Based Anomaly Detection**. *ACM Transactions on Knowledge Discovery from Data*, **6**(1):1–39, 2012. 21

[22] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. **LOF**. *ACM SIGMOD Record*, **29**(2):93–104, 2000. 24, 42

[23] B Schölkopf, R Williamson, A Smola, J Shawe-Taylor, and J Platt. **Support Vector Method for Novelty Detection**. *NIPS*, **12**:582–588, 1999. 27

[24] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. **A training algorithm for optimal margin classifiers**. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, 1992. 27

[25] Abdenour Bounsiar and Michael G. Madden. **Kernels for One-Class Support Vector Machines**. *2014 International Conference on Information Science Applications (ICISA)*, 2014. 28