Assessing Data Quality in Image Recognition Datasets of Swedish Financial Reports

LUND
UNIVERSITY

Maren Demuth

Master Thesis
Spring Term 2024

Supervisor: Magnus Wiktorsson
Examiner: Ted Kronvall

Lund University
Centre for Mathematical Sciences

Abstract

A great amount of financial statements of Swedish companies are registered with the Swedish companies registration office, Bolagsverket, as paper copies. These copies are then scanned and made available to the public as image-PDFs, meaning the financial information contained in them is not digitised and therefore not easily processed in an automated manner. Being able to do so, is valuable however, e.g. to offer automated credit risk assessments or investigate fraud cases through modern technologies. In order to digitise this financial information, the company I work for has developed image recognition algorithms that can create a structured data representation of the financial statements. In some cases however, the image recognition fails at creating an accurate representation of what is written in the financial statement. This poses a data quality issue where further applications onto the digitised financial data might be misconstrued and offer a skewed or outright wrong perspective of the underlying financial situation of a certain company. It is therefore the goal of this thesis to develop a solution that can identify cases in which the image recognition data does not match its true counterpart.

The solution developed in this thesis is three-fold. First, the quality of the extracted data is evaluated through the assignment of so-called *error labels*. Second, a Random Forest classifier is trained to be able to predict these error labels and lastly, a *quality score* is calculated to offer a suggestion of the best possible representation for each value in a financial statement. It is shown that this approach obtains reasonable results and does indeed beat an existing approach to solving the same problem by a considerable margin. The solution build in this thesis therefore offers a valuable extension to the image recognition program, by allowing for a data quality assessment of the extracted information and therefore increasing the confidence one can have in the digitised financial data to accurately represent the original paper copies.

Acknowledgements

Rarely any work happens in isolation and neither has the work that lead me to this, my Master thesis. I am grateful for the richness in support I have received from many wonderful people. Let me start by offering thanks to my coworkers, who have been instrumental in helping me get set up in working with and understanding the data itself. I would like to make special notice of Sebastian Søderberg, who has made great effort in explaining the intricacies of financial statements to me and has been welcoming my questions without hesitation.

Gratitude is naturally also extended to the company I work for (who, due to the sensitivity of discussing the inner workings of their product in this thesis wishes not to be named) for the opportunity to write my thesis with them and providing the necessary data.

Additional thanks goes to my supervisor Magnus Wiktorsson, who has shown incredible perseverance with my work and made it possible for me to complete my Master's degree while working at the same time. I am very grateful for the flexibility and support you have offered me in this lengthy process.

Last but not least, my thanks goes to my partner Manel, for your wealth in patience and compassion as you accompanied me through all of this.

Popular Science Summary

In this thesis, I build a solution that can identify whether or not a data point is of good or bad quality. These data points are originally created by a program that extracts information from financial reports. Now, this program works by using image recognition technology and the problem is, that sometimes it simply extracts the wrong information from the financial report. I want to figure out when that happens.

A financial report contains numbers that describe a company's financial situation, usually over the period of a year. For example, it contains the company's revenue and cost figures. In Sweden, these reports are often registered as paper copies with the financial authorities and then they get scanned into PDFs. Because these PDFs are basically just images, the information contained in them cannot easily be processed with computers. It would be very valuable to be able to do that however, for example to offer automated credit risk assessments on these companies. In order to be able to do that, the company I work for has therefore developed this image recognition program which can extract information from the PDFs and store it in a table format.

The central problem of my thesis is that sometimes, the program will not extract the correct information. In the PDF, the revenue value might read 4502 SEK, but the program has extracted the value 8502 SEK instead. I want to be able to identify when that happens to make sure that the extracted information is of high quality. To solve this problem, I build a Machine Learning program that can predict whether an extracted number matches the number written in the PDF or not. As it turns out, this is a decent way to identify the incorrect extractions. This means that my Machine Learning program can be integrated with the image recognition program that the company I work for develops, to improve the extracted information it offers.

Contents

Abstract i

Acknowledgements ii

Popular Science Summary iii

1 Introduction 3
- 1.1 Data Quality . . . 3
- 1.2 Image Recognition of Swedish Financial Reports . . . 4
  - 1.2.1 About the Image Recognition Data . . . 5
- 1.3 Problem Statement and Sketch of the Approach . . . 7
- 1.4 Alternative Approaches . . . 8
- 1.5 Thesis Structure . . . 8

2 Theoretical Background 9
- 2.1 Definition of Data and Data Types . . . 9
- 2.2 Data Quality . . . 10
- 2.3 Statistical Learning Theory . . . 12
  - 2.3.1 Bias-Variance Trade-Off . . . 14
- 2.4 Random Forest Classifier . . . 16
  - 2.4.1 Growing a Decision Tree . . . 16
  - 2.4.2 Combining Multiple Decision Trees . . . 19
  - 2.4.3 Prediction Probabilities in the Implementation . . . 19
- 2.5 Machine Learning Tools . . . 19
  - 2.5.1 Data Preparation Techniques . . . 20
  - 2.5.2 ML Training Improvements . . . 20
  - 2.5.3 Evaluation Methods . . . 21

3 Application and Results 23
- 3.1 About the Implementation . . . 23
- 3.2 Types of Error in the Image Recognition . . . 23
  - 3.2.1 Examples of Scraper Mistakes . . . 25
  - 3.2.2 Assigning Error Labels . . . 28
- 3.3 Random Forest Classification . . . 30
  - 3.3.1 Feature Selection, Encoding and Model Configuration . . . 30
  - 3.3.2 Classification Results . . . 33
- 3.4 Scraper Value Selection . . . 37
  - 3.4.1 Quantifying Quality . . . 37
  - 3.4.2 Quality Scoring Results . . . 39
  - 3.4.3 Model Comparison Results . . . 40

4 Discussion 43
- 4.1 Limitations of the Solution . . . 43
- 4.2 Future Outlook . . . 44
- 4.3 Conclusion . . . 45

A Examples of Scraper Mistakes **46**

B Confusion Matrices by Scraper **51**

C Quality Score Histograms by Match Flag and Scraper **52**

D Selected Sample of Suggested and Merged Values **53**

References **54**

1 Introduction

1.1 Data Quality

By far, this is not the first nor the last thesis concerning itself with 'data' and 'Machine Learning'. Words so ubiquitous that sometimes it seems, not even my grandpa can escape hearing about 'AI' and 'data-driven decision making'. The omnipresence of these technologies even leads some people to refer to our current times as the *Age of Data*. From CEOs seeking to cut costs to politicians scrambling to address global pandemics, decisions are being made by 'looking at the numbers', trusting that 'the data does not lie'. And it makes sense, numbers and statistics can offer incredible insight on whatever one wishes to study. Without knowing how much incoming versus outgoing money a company has, it would be quite the challenge to successfully manage a business and offer secure employment. Without knowing how many people fall ill from a disease and at which rate, intervention and cure might be futile efforts.

Gathering records and creating data is not just a phenomenon of recent decades. Indeed, the earliest forms of writing (now known as *proto-cuneiform* writing) were motivated by economic accounting initiatives, recording quantities of sales goods, labour and livestock in circa 3300 BCE Mesopotamia.[11] From clay tablets to microchips, the creation of data is essential to human organisation and decision making.

What certainly has changed over the thousands of years of recording data, is the vast amount of information that is being gathered every day. It is hard to imagine a human living 200 years ago that knew the precise amount of steps they took each day or how high their heart rate went climbing up the stairs. I would claim that we are not living in the Age of Data because data is so essential to our processes, but rather because virtually everything is being measured, recorded and tracked these days.

Personally, I have worked with creating 'Key Performance Indicator dashboards' and setting up 'monitoring solutions' for some years and whenever I would discuss the numbers shown in any of these applications with my colleagues, I would often find myself saying sentences along the lines of 'Yes, but: consider that this only measures X and not Y' or 'Yes, but: this number only shows how often A happens under this condition, not in every case'. In order to draw valid conclusions from any of these dashboards, graphs, tables or plots, it is essential to understand the conditions under which each number, or data point, was created. Is the 'number of users' some software product might have counting all existing user IDs or is it based on who has logged in to the platform in the last month? In order to provide the according number, a decision needs to be made on how 'users' should be measured, thereby selecting which aspect of reality a data point should represent. The context that goes into the creation of data is thus necessary for the correct interpretation of the numbers, but also highlights the difference that exists between the data itself and the phenomenon of the real world that it aims to represent.

Indeed, this representational quality of data leads the author of [19] to describe its function as *semiotic*, meaning that data is a stand in for things other than itself, and that data can be seen as a model of reality. A model in which only selected characteristics of the chosen aspects of reality are being represented.

If one accepts that data is simply a model of reality and thereby distinct from it, the question naturally arises how well data actually represents said reality. What happens, when a data point does not actually match its real counterpart? In any scientific experiment, the idea of measurement error is well established. One speaks of 'noise' in the data that indicates such misrepresentations

between measurement values and the underlying real phenomenon. In a broader context, studying cases in which data and reality do not match, falls under the category of *data quality*. When assessing the quality of a particular set of data, one aims to address, among other things, how well the data matches the part of reality that it was created to represent. Coincidentally, this is what I have chosen to make the topic of my thesis.

While sure enough, I apply said buzz-wordy technologies of Machine Learning in my work, it is with the aim of challenging the very fundamental assumption to any data-driven approach: the assumption that the data used for creating abstractions of reality actually represents this reality sufficiently well.

As my grandpa likes to joke: 'Statistics is the art of lying in its finest form'[1]. While of course, the vast majority of numbers and statistics are created with good intent, there is an inkling of truth to my grandpa's words, expressed by the idea that sometimes, numbers do lie, or at least do not offer a good representation of reality. Personally I believe that the more significance one places on insights derived from data, the more one relies on it to make decisions, the higher the burden placed on this fundamental assumption of accurate representation to hold true. By choosing data quality to be the over-arching topic of my thesis, I hope to put a bit of focus on the care that should be taken whenever data is used to learn about reality.

Now, this sets the stage for my thesis rather broadly. The next section narrows down the particular application I have chosen for my work.

1.2 Image Recognition of Swedish Financial Reports

I currently find myself working for a company that offers financial services, amongst which are credit risk assessments on other companies. These assessments are based on a broad range of data points for each company, including the financial statements that a company typically releases every year. These statements are often publicly available and contain valuable information on a company's financial health. Unfortunately, in Sweden, these financial statements are largely filed in non-digital formats which make automated processing of them difficult. Often, the financial data is only available through scanned images of a printed document, requiring extraction methods such as image recognition algorithms to create a digital representation of the scanned images.

To put it in the language introduced above, the company I work for creates abstractions (the credit risk assessments) of another company's financial situation (that company's reality) based on a dataset that contains extracted information from the image of the financial statement (the representation)[2]. In order for the credit risk assessment to be reliable, one needs to assume that the extracted dataset accurately represents the company's financial reality. To test whether this assumption is true shall be the objective of this thesis.

The Swedish authority that is responsible for handling the financial statements of companies is called Bolagsverket, or the Swedish Companies Registration Office. And while they generally offer the option to hand in financial reports in a digital file format called XBRL (eXtensible Business Reporting Language), many companies still choose to hand in a physical copy of their report to

[1] Loosely translated from the original German: 'Statistik ist die Höchstform der Lüge.'

[2] One could argue that the original financial statement, before it becomes a scanned image file, is already just a representation of the actual financial situation of the company and of course, that is valid. And spun even further, can one even know what the *true* financial reality looks like? Is there such a thing as *objective truth*? Interesting questions indeed, but in order to make any practical progress, I simply assume that the scanned image file as it was filed with the authorities is a company's financial reality.

Bolagsverket. In order to digitise these physical reports, Bolagsverket scans them as TIFF files and compiles a PDF which they then make available through their website search tool and subscription services.

The financial information contained in each report is very valuable as it offers crucial insights into the financial health of each company and many industries and third parties rely on this information in order to operate their businesses. To address this need, there are many companies which offer to redistribute the financial report from authorities such as Bolagsverket, and they often not only redistribute the information from the financial report, but also add their own interpretations and services to this, e.g. in the form of credit assessments and risk analyses. As mentioned before, the company I work for is one of these companies.

In order to provide products like this, the companies are challenged by the fact that especially in Sweden a large part of the financial statements are filed as printed copies, not using the digital XBRL format. In order to obtain digitised and structured financial information on other companies, they need to rely on manual data entry where a person reads the scanned PDF and types the information in a structured file format, or increasingly, image recognition algorithms which seek to automate this labour. While differing in their efficacy, both methods are prone to produce data quality issues in the form of incorrectly extracted data points or misrepresentations of reality, as I called it earlier. In the context of this problem, I consider the scanned PDF to be the company's financial 'reality' and the extracted dataset, manually entered or created through an image recognition algorithm, the representation. Whenever scanned PDF and the corresponding data point of the extracted dataset differ, it poses a data quality issue and it is my goal to create a statistical model that is able to detect these cases.

1.2.1 About the Image Recognition Data

As is established above, the reality that shall be represented through data are the financial statements Swedish companies file with Bolagsverket as physical copies. Through Bolagsverket the company I work for has obtained a large dataset of scanned PDFs, which I consider the 'reality', and also implemented image recognition algorithms to extract the information contained in each, which form the 'representation'.

The content of a financial statement generally follows a similar structure. For a limited liability company in Sweden, called 'Aktiebolag', Bolagsverket requires the financial report to contain a Director's report ('Förvaltningsberättelse'), Income Statement ('Resultaträkning'), Balance Sheet ('Balansräkning'), Notes and often an Auditor's report. [4] While the Director's and Auditor's report as well as the Notes sections of the statement are very text-laden, the Income Statement and Balance Sheet contain the financial figures, such as a company's assets, debts and revenue, which are firstly numeric and secondly, very relevant for gaining insights on the financial health of the company in an automated manner. It is these two parts of the financial statement that the image recognition focuses on, so that the resulting extracted dataset is largely numeric.

At this point it is worth mentioning that of course, when one attempts to extract financial data from the scanned PDFs, it is not only the numeric values that are extracted, and therefore potentially a misrepresentation of reality, but also the text that gives meaning to the numerical values, i.e. the financial figure names. For example, if a company has an annual revenue of 250000 SEK, it is not just the value of 250000 that needs to be extracted, but also the associated meaning of it, in this case the text string 'revenue'. Both extractions, the numeric and the text-based, can suffer from data quality issues. For the text-based ones these can often take the form of typos,

wrongly-read special characters, but also issues like undetected line breaks. Identifying these issues is complicated enough, but can be addresses through domain knowledge checks. That is, the financial figure names are standardised across reports and while not every report necessarily contains all financial figures, it is possible to know which ones to expect. So if the image recognition algorithm returns the word 'Nettoomsatting' instead of the correct 'Nettoomsättning' (English: revenue), a simple string distance measure allows for a reasonable enough mapping of these two strings, and therefore a sufficiently identified meaning of the financial figure value associated with it.

On the other hand, the domain knowledge constraints are much less precise for the financial figure values. While sure enough, some financial figures cannot take negative values, or a value of a trillion SEK would be very unreasonable, these bounds are not small enough to make for an efficient detection of what the actual value is. It is because of this reason, that I have chosen to only focus on the data quality issues pertaining to the extracted financial figure values and not the financial figure names in this thesis. In the dataset that I have used for this project the text strings are therefore already cleaned and mapped to a sufficient meaning by using the domain knowledge constraints described above. Data quality issues encountered in the correct extraction of financial figure names are therefore not considered here.

Now, in order to extract financial data from the PDFs, the company I work for has implemented several different versions of image recognition algorithms. Each version is called a scraper and in this thesis I worked with the output of six different ones. While the underlying implementation for the image extraction part has been the same across all scrapers, each scraper itself was configured in slightly different ways. For example, one scraper would sharpen the PDF image before attempting an extraction, another would try to increase contrast and so forth. This means that for every financial figure included in a PDF, there are six different outputs. The challenge, and ultimate goal my work, is to identify which of these six values actually corresponds to the value as it is displayed in the original image, if any, and then to make a recommendation which value should be chosen as the best possible representation of the image one.

Since the actual values contained in the image are not easily machine-readable, this would technically be an unsupervised learning task, as there are no available labels, or true values, for each of the extracted ones. And indeed, as part of the initial scoping of this problem, I have tried a few unsupervised attempts at solving the problem, but with little success. Luckily, the company I work for has a secondary source for the financial data contained in each PDF available. While this source comes with limitations itself (which are described in Section 4.1), the available data is comparable and so it was possible to have at least an approximation of the true values for each scraped one. This ultimately allowed me to use supervised learning algorithms to identify misrepresentations of reality, which is described in more detail in later sections.

Table 1: Illustrative Data Sample

ID	**Financial Figure Name**	**Scraper Version**	**Company Age**	**Company Type**	**Scraped Value**	**Secondary Source Value**
Report 1	ebitda	Scraper 3	2208	AB	-31123422.00	NaN
Report 1	short_term_debt	Scraper 6	2208	AB	18230.00	18000.00
Report 2	inventories	Scraper 2	2815	AB	0.00	0.00
...	...	...	...	...	...	...

An illustrative sample of the dataset used for this thesis can be seen in Table 1. The columns ID, Financial Figure Name and Scraper Version uniquely identify each scraped value. The columns Company Age (measured in days) and Company Type (e.g. 'AB' which stands for 'Aktiebolag' - a Swedish Limited Liability Company) give further information on each company. Lastly, the Secondary Source Value column is used as the approximation to the unknown true value as it is printed in each report.

1.3 Problem Statement and Sketch of the Approach

The goal of this thesis is two-fold. First, I aim to assess the data quality of the image recognition data with the focus on determining whether or not each value accurately represents its true counterpart, that is the financial figure value as it is printed in the report itself. And second, I use the data quality assessment to make a selection across the six different scraper outputs, deciding which of them is most likely to be an accurate representation of the true value. In order to build a solution that can hopefully achieve this goal, I split my approach into three steps.

Step 1. I start out by drafting a heuristic on what constitutes 'data quality' in my chosen application domain of the image recognition efforts for Swedish financial reports. In essence, I want to answer the question: what is the difference between scraped values considered to be of 'good quality' and those considered 'bad quality'? As is described in more detail later on in Section 2.2, in this step, I aim to define the data quality dimension of accuracy for my application domain.

Step 2. In order to be able to assess whether or not the image recognition outputs do match with their true counterparts in an automated manner, I do require a dataset for the true values. Luckily, a secondary source for the financial data is available. While this secondary source does come with some limitations (described in Section 4.1), it is independent of the image recognition algorithm and sufficiently represents the true values. However, this secondary source is only available for a subset of all non-digitally filed financial statements and it is expensive to acquire. Because of this, I do not want my solution to rely on it going forward and ultimately, in this Step 2, I aim to replace the secondary source by training a predictive algorithm (in particular a Random Forest Classifier) to replace it. This trained classifier is then used to assess the accuracy of scraper values.

Step 3. Lastly, I use the data quality assessment model from Step 2 to make a selection for each financial figure, which scraper output is the best possible one to accurately represent the true value. To evaluate the success of my solution, I compare it to the current solution that the company I work for has implemented. This 'current solution' equally provides a selection across the six scrapers, but it has several drawbacks (for more details, see Section 3.4). If my solution performs better than this existing solution, I deem my efforts successful and vice versa.

At this point, I like to emphasise that my use of a Machine Learning model is, for lack of a better word, a bit 'old-school'. Not only have Neural Networks been the fastest growing ML algorithms, in terms of published research papers, in the period from 2013 to 2022 [16], Deep Learning has become a ubiquitous buzzword. The trend does seem to be to grow large ML models that require little human input and mostly identify patterns in the data autonomously. However, this comes at the cost of interpretability and lack of oversight, giving rise to the term 'black-box models'

where it is unknown to the human observer how the algorithm has arrived at its output. While complex Machine Learning algorithms are relatively cheap to apply after they have been trained, the computational cost to train them is high which also means that the energy consumption and environmental impact is a factor to be considered.

Since my thesis problem, to assess the scraper output data quality and offer a best-possible value selection, is ultimately an intermediate step in the over-arching solution to providing digitised financial data on Swedish companies, it is a requirement that my solution is relatively light-weight and can be included in a production pipeline without accruing high cost. In addition to this, I have also considered my personal experience and skills for the selection of a suitable Machine Learning algorithm and have ultimately set the scope of my solution to rely on a more 'traditional' Machine Learning algorithm in the form of a Random Forest classifier.[3]

1.4 Alternative Approaches

Before describing the actual chosen approach, I want to mention the alternative approach I tried to apply as part of the initial exploration of the thesis problem. Namely, I originally set out to solve this problem using Outlier detection methodologies. Outliers being data points that to some degree are 'different' to others, or are considered atypical, extreme or otherwise not fitting into the existing expectations towards the data. This concept of *outlyingess* does not necessarily mean that data points are wrong, in the sense that they misrepresent reality, but it gives an indication that a data point might be 'suspicious'. Often, for every data point an outlier score is calculated that represents the degree to which this data point is different from others and I intended to use this scoring as an input to assessing data quality. Unfortunately however, the methods I tried to apply to this thesis problem did not lead to satisfying results. While some extreme values could correctly be identified as actual mispresentations of reality, a large part of the quality issues stem from extracted values that do not necessarily appear very different from other values and as such, the outlier methodology is simply not a good candidate for this thesis problem. This has lead me to explore other possible methods, ultimately leading to the approach detailed in the previous section.

1.5 Thesis Structure

Beyond the introduction, this thesis is split into three further sections. Section 2 describes the theoretical background relevant to this thesis. It begins by offering a classification of data types as well as a data quality definition and moves on to describing principles of Statistical Learning Theory, the mathematical framework for Machine Learning, as well as detailing the workings of the Random Forest and Decision Tree models. This section ultimately concludes with mentioning selected Machine Learning tools that I applied in my approach.

Following that, Section 3 reports the details of my actual application. From information about the implementation in Python, to the initial heuristic as well as outputs from the Random Forest classification and ultimate value selection, it offers an account of the results I obtained by applying the previously described approach to the image recognition data of Swedish financial reports. Lastly, Section 4 sums up my solution by discussing limitations of the approach as well as ideas for potential improvements and finally ends my thesis with a conclusion. Additionally, the appendix includes excerpts of actual financial reports and references are stated as well.

[3]While technically, a Random Forest model is still difficult to interpret directly, the narrow choice of my application domain in Step 2 ensures sufficient transparency and control for the overall solution.

2 Theoretical Background

2.1 Definition of Data and Data Types

When one thinks of the representation of a dataset in mathematics, it is often the matrix X that comes to mind. X usually is nicely defined, with n rows and p columns. One might even know some characteristics of the matrix, whether it is invertible, sparse or its diagonal has interesting properties etc. How the values of the matrix X came to be is less often considered. When indeed, as everybody that has tried to tinker with or read about Machine Learning algorithms has experienced, there is a long path of 'data wrangling' before that matrix, or the code variable `X`, is ready to be plugged into the training function of a Machine Learning library.

In order to offer some perspective on this process of 'data wrangling', i.e. the handling, transforming, cleaning or 'making ready' of some dataset for further analysis and abstraction, this section explores what *data* might be and how different types of it can be distinguished. To do so, I follow the classifications given in [2, p.6-8] which defines data as follows.

Definition 2.1. Data represent real world objects, in a format that can be stored, retrieved, and elaborated by a software procedure, and communicated through a network. The process of representing the real world by means of data can be applied to a large number of phenomena, such as measurements, events, characteristics of people, the environment, sounds and smells. Data are[4] extremely versatile in such representation.

The authors describe several view points from which different *types of data* may be identified. Firstly, one common distinction is made on the basis of the structure in which data is available, with unsurprising category names such as:

- *Structured data:* is data that is given in a well-defined structure where each value of the dataset can be associated with a meaning (e.g. the column name) and the list of all possible meanings is known and fixed. Examples of this are traditional relational tables of a database.
- *Semistructured data:* also called 'schema-less' data, is available in a more flexible structure where data values are still clearly associated with a meaning, but the list of all possible meanings is not necessarily known at creation and can be expanded upon at any point. Common file formats for storing such data are XML or JSON.
- *Unstructured data:* refers to information expressed through for example natural language that follows no specific structure and where values do not necessarily have a clearly identifiable meaning.

This view of data types is a practical one when considering the ease of handling data. The more structure a dataset offers, the easier it is to assign meaning to each value and to automate the processing of it. The less structure a dataset follows, the more external knowledge and input is required to make sense of the information contained and the more difficult it is to create procedures that are well-equipped to handle the data meaningfully.

There are other views on working with data from which different data type definitions arise. Again, drawing from the work of [2], such distinctions can be made from the view point of data

[4]The word 'data' is grammatically a plural noun which many academic sources correctly address. In more common language and industry contexts, 'data' is spoken of as a noun in the singular, which I prefer and therefore use in this thesis.

as a product, in which the 'product-readiness' of data is evaluated. First, that is done through considering *raw data* which is data as it has been collected from a source without any transformations added to it, then *component data items* which are temporary in nature and represent a layer of transformation performed on the raw data. And lastly, the *information products* which are the resulting datasets of several processing activities.

Further view points describe data type distinctions based on *elementary* and *aggregated* data. The former is data representing 'atomic phenomena of the real world' and the latter collections of such elementary data through abstractions like adding or averaging values. The sources from which data stems also offers another categorisation: into, among others, *federated data* (coming from several sources) and *web data* (being scraped from websites). And lastly, when considering a time dimension of the data, it also makes sense to distinguish data that is *stable* (unlikely to change over time), *long-term-changing* (changes happen infrequently or over long time periods) and *frequently-changing* (changes happen frequently, e.g. in real-time streaming data).

Now, these classifications very much focus on the data as a whole. But, if we consider the values contained in a dataset, they themselves can be of differing types. A very common and broad distinction is that of *numerical* and *categorical* values. While numerical data will most often be any real number[5], representing counts, averages, percentages, categorical values will be text-based. Furthermore, data values might also be missing which is referred to as *blanks*, *NULL-values* or *NaNs*.

To apply these various distinctions and definitions of data at large to the dataset relevant for this thesis, the dataset described in Section 1.2.1 is a structured one that can be considered an 'information product', since data point meanings are well-defined and much cleaning and transforming has already happened. Furthermore, the dataset contains elementary as well as aggregated data, since some financial figures are actually aggregates of others. Additionally, while the data for this thesis stems from several thousands of financial statements and multiple scrapers, the information is comparable and each scraper and statement combination is unique, requiring no record-matching (i.e. identifying the same object across different sources), so I would consider it a 'single-source' dataset. And lastly, while the dataset is stable, in the production environment of the company I work for the same data would be considered long-term-changing, reflecting the maintenance and development efforts undertaken on the image recognition code base.

2.2 Data Quality

Data Quality (DQ) is often considered a multi-dimensional discipline with varying definitions.[8][7] An aspect that is often mentioned in such definitions is the idea of 'fitness for use' where data is considered of good quality, if it meets the expectations set upon it by the people that use the data and if it is able to address the purpose for which the data was created.[19]

Data quality aspects are called *dimensions* and their quantification the according *metrics*. A single dimension can be assessed through several different metrics. Nowadays, there are many DQ frameworks ([7] provides a review of such) and tools (see [8] for another review) that aim to offer generalised DQ assessments, but the authors of both papers identify some limitations. First, there is a lack of standards when it comes to DQ definitions, dimensions and metrics. While the ISO 8000

[5]Of course, a numeric value stored in a computer cannot be truly infinite, so a number considered continuous in a dataset is actually just a discrete approximation of it. Which in itself highlights the importance of thinking of data as a *representation* of reality, not reality itself. That is, if one thinks of the mathematical concept of infinity as being part of the real world in any case.

standard for data quality exists [3], it is defined on a very high level and lacks the granularity to be effectively applied to a wide range of data sets and applications. Second, DQ is context dependent and evaluation is highly subjective and third, the authors of [8] even suggest that decades of research into generalised DQ dimensions and metrics did not lead to successful generalised frameworks and the focus should therefore shift to more practical approaches.

Nonetheless, some overlap across DQ dimensions can be seen. The most commonly described ones are:

- **Accuracy:** describes how close data is to the exact or true values that the data was intended to represent.
- **Relevancy:** is the extend to which data meets current and potential needs of its users.
- **Timeliness:** measures the length of time between data being available to the underlying event or phenomenon of the real world occurring.
- **Coherence:** relates to the ability of data to be reliably combined in different ways and for various applications and in such a way that the differences in data can be explained by difference in the event or phenomenon it describes.

The descriptions of each dimension stem from [9]. Often, metrics to evaluate either dimension are defined as the ratio of items that pass a certain quality test related to the dimension and the number of items that were tested, i.e.

$$M = \frac{\text{No. items that pass a quality test}}{\text{No. tested items}} \tag{1}$$

For this thesis, I am testing whether or not each scraped value offers a correct representation of its counterpart, the reality, and as such, I am considering a problem of the DQ dimension accuracy. While the evaluation of other DQ dimensions could potentially offer valuable insights on the image recognition data as well, they are not directly related to my thesis problem and I therefore deem them out of scope.

Since accuracy assessment requires knowledge of the true values, which often are unavailable, there are no generalised metrics defined for it.[8] Because this lack of 'objective truth' occurs for a lot of tasks,[19] in fact excludes the accuracy dimension from their DQ framework completely and instead proposes a validity dimension. This validity dimension describes the extend to which a given data point adheres to the domain knowledge constraints applicable to it, e.g. a value of 4.2m in a data column that describes human height would be considered outside the permissible range and therefore the value would be deemed 'invalid'.

I am in the lucky position to have access to a secondary data source for the financial data, however, which allows me to build a solution that can (hopefully) assess DQ accuracy. In that, my accuracy metric follows the basic definition described in (1) and I call it the *success rate* of my solution which ultimately in Step 3 of my approach is calculated as

$$R = \frac{\text{No. values that match the true one}}{\text{No. values in the validation data}} \tag{2}$$

Note that the quality test in the numerator of this success rate applies a binary quality test of whether or not the extracted value matches its true counterpart. As I describe in more detail later

on, I actually consider several aspects of each value that potentially match the true one or not, which is not reflected in this definition. This is because, ultimately, when I offer my data point selection in Step 3, it is the binary nature of a 'full' match between extracted and true value that determines the success of the solution. With only a partial match between the values I cannot claim my result to be successful.

It seems that overall, there is unfortunately no general 'mathematical theory for data quality assessment' available. Many industry relevant frameworks and tools exist to address certain needs of data quality management, but scientific research is often limited by the fact that data quality is context specific and difficult to generalise. In this thesis, I build a targeted solution to assess the single dimension of accuracy for a very specific data generating process (that of the financial statement image recognition). This solution includes the training of a Machine Learning algorithm for which a much more well defined mathematical framework exists. This framework is described in the following section.

2.3 Statistical Learning Theory

Statistical Learning Theory (SLT) provides the mathematical framework to Machine Learning algorithms and concerns itself with the inference problem of estimating functions based on a set of observations. As such, its origins lie in the 1960s where the first theoretical foundations for these learning models were formulated. With the introduction of Support Vector Machines in the 1990s, the popularity of Statistical Learning Theory (and hence Machine Learning) increased sharply.[29] Since this thesis applies a supervised learning algorithm, the following sections focus on theory relevant to such tasks.

According to [27] there are three components to a mathematical model that is capable of learning from data:

(i) A **generator** which consists of the random vector $X = (X_1, \ldots, X_p)$. X is drawn independently from a fixed, but unknown, probability distribution $P(X)$ which is defined on the *feature space* $\mathscr{X} \subset \mathbb{R}^p$.[6] In more modern contexts, the elements of the random vector are referred to as *inputs* or *features*.[13]

(ii) A **supervisor** that returns the output vector Y (also called *response*) for every input X in accordance with a conditional distribution function $P(Y|X)$ which is also fixed, but unknown. Y takes values in the set $\mathscr{Y}$ and the joint probability distribution $P(X, Y) = P(X)P(Y|X)$ is defined over the space $\mathscr{X} \times \mathscr{Y}$.

(iii) A **learning machine** which summarises a set of functions $\{f(X, \alpha) : \alpha \in \Lambda\}$ where α is a parameter from the arbitrary set Λ. This set can for example simply contain scalars, vectors or more abstract elements.[28]

The idea of the learning problem can then be expressed as choosing the function from $\{f(X, \alpha) : \alpha \in \Lambda\}$ that predicts the response Y in 'the best possible way' - which is to be defined. This function selection is based on a training set of n random, identically distributed (iid) observations $(x_1, y_1), \ldots (x_n, y_n)$ which are drawn from the probability distribution $P(X, Y)$.

[6] Note that formally, P should be defined as a probability measure on a σ-algebra of $\mathscr{X}$. I will omit the details of measure theory for the purpose of this thesis, and assume that the necessary conditions for measurability are met.

Continuing with the problem set up described in [27], in order to solve this learning problem, a *loss function* $L(Y, f(X, \alpha))$ is introduced. This loss function measures the discrepancy between the response Y of the supervisor to a given input X and the response $f(X, \alpha)$ provided by the learning algorithm. This discrepancy is also called the *prediction error*. The expected loss is then given by the *risk function* (also referred to as the *expected prediction error* in [13]),

$$R(\alpha) = \int L(Y, f(X, \alpha)) dP(X, Y) \tag{3}$$

This results in a refined statement of the learning problem, as follows.

Definition 2.2. It is the goal of the learning problem to find a function $f(X, \alpha_0)$ which minimises the risk $R(\alpha)$ over the class of functions $\{f(X, \alpha) : \alpha \in \Lambda\}$ when the joint probability distribution $P(X, Y)$ is fixed, but unknown and an iid sample $(x_1, y_1), \ldots (x_n, y_n)$ is given.

Furthermore it is assumed that the learning model does not make any further assumptions on $P(X, Y)$ except what is stated in the definition and the sample of data may assume non-deterministic values, e.g. there might be data quality issues[7] in the training data itself.[10, p.15]

The existence of a data sample for the response Y means that the learning problem as defined above is indeed a task in *supervised* learning, as opposed to *unsupervised* learning where the response of the supervisor is unavailable and the focus rather lies in understanding structures in the generator data such as clusters.[13, p.2]

The risk function as defined in Equation (3) describes the general setting of the learning problem. Depending on the data type of the response Y, the supervised learning problem can be divided into different specific settings. Namely,

- **Regression problems:** If the response is of numerical or quantitative nature, the learning problem is often referred to as a *regression* model. A commonly used loss function for regression problems is the *squared error loss*, [13]

$$L(Y, f(X, \alpha)) = (Y - f(X, \alpha))^2 \tag{4}$$

- **Classification problems:** Given a response that is of categorical or qualitative nature, i.e. $Y \in \mathscr{Y} = \{1, \ldots, k\}$ where k is the number of categories (also referred to as *labels* or *classes*), the learning problem is one of *classification*. In older texts this is also called a problem in *pattern recognition* and a typical loss function is the *0-1 loss*, [28]

$$L(Y, f(X, \alpha)) = \begin{cases} 0, \, Y = f(X, \alpha) \\ 1, \, Y \neq f(X, \alpha) \end{cases} \tag{5}$$

While both the probability distribution $P(X)$ and conditional distribution $P(Y|X)$ are assumed to exist, they are also unknown, which means that the risk function in (3) cannot actually be calculated

[7] At this point, the literature often mentions *noise* in the data and gives examples such as measurement errors or transcription mistakes when data is being generated. As is mentioned in the previous section, in the domain of data quality, this is also referred to as an accuracy issue, or a misrepresentation of reality - exactly the kind of situation I set out to study.

and thus, not formally minimised either. In order to solve the learning problem as defined in 2.2, an approximation to $R(\alpha)$ is required and this approximation is given by the *empirical risk function*,

$$R_{\text{emp}}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i, \alpha)) \tag{6}$$

which is calculated using the n data points in the training data. The idea is then to approximate $L(Y, f(X, \alpha_0))$ which minimises $R(\alpha)$ by $L(Y, f(X, \alpha_n))$ which minimises $R_{\text{emp}}(\alpha)$. This is called the *empirical risk minimisation principle* (ERM).[28]

Statistical Learning Theory concerns itself with the conditions for consistency of the ERM principle, meaning the conditions under which the empirical risk converges in probability to the actual smallest risk (see below). Furthermore, SLT describes the associated rate of convergence and describing how algorithms that control this convergence, meaning the generalisation ability of the ERM principle, can be constructed.[27] Details of these aspects are deemed outside the scope of this thesis and it is therefore assumed that the ERM principle is a consistent approach to solving the learning problem, meaning the following two conditions are assumed to hold.

(i) The sequence of expected risks for the functions $L(Y, f(X, \alpha_n))$, $n = 1, 2, \ldots$ of which each minimises the empirical risk $R_{\text{emp}}(\alpha_n)$ converges in probability to the minimal possible value of the actual risk, i.e.

$$R(\alpha_n) \xrightarrow{P} R(\alpha_0) \text{ as } n \to \infty \tag{7}$$

(ii) The sequence of calculated empirical risks equally converges in probability to the minimal possible value of the actual risk, i.e.

$$R_{\text{emp}}(\alpha_n) \xrightarrow{P} R(\alpha_0) \text{ as } n \to \infty \tag{8}$$

While the ERM principle is statistically consistent, it is still faced with the challenge whether the function that minimises $R_{\text{emp}}(\alpha)$ is actually a good approximation to the function that minimises $R(\alpha)$ or not. Since the empirical risk is calculated based on a fixed set of training data, it might look very different for a new, previously unseen set of data points. In order for the ERM principle to be a powerful tool in solving the learning problem, the difference between the empirical and actual risk function should be sufficiently small. In such a case, the chosen function $f(X, \alpha_n)$ is said to *generalise*, meaning it performs similarly over the training data as well as unseen data.[10, p.91-92]

As it turns out, the ERM principle is prone to over-fitting, meaning in its basic form it will result in a function $f(X, \alpha_n)$ that does not generalise well. Modern implementations of ML algorithms therefore rely on several improvements to the ERM principle, such as regularisation or restricting the set of functions considered in the learning algorithm.[20, p.16] The latter is described in more detail in the following section.

2.3.1 Bias-Variance Trade-Off

Since the introduction of Support Vector Machines mentioned above, many more types of learning models have been introduced: from Neural Networks, to Clustering algorithms to Tree-based methods such as Random Forest.[16] Each model comes with its own set of assumptions on the data and the type of functions it considers to solve the learning problem. More formally, each model aims to solve the learning problem by finding the best possible function in a set $F_{\text{model}} = \{f(X, \alpha) : \alpha \in \Lambda_{\text{model}}\}$.

For example, F_{model} might only include functions that are linear, or only second degree polynomials etc. Ideally, F_{model} is chosen in such a way that it matches the true relationship of the generator and supervisor data, i.e. $P(X, Y)$. But again, since the joint probability is unknown, this might prove difficult to do. It might then happen that the choice of F_{model} is not ideal, meaning the best possible function that can be obtained is limited through the choice of F_{model}. Because of this, the set F_{model} is also referred to as the *bias of the supervised algorithm.*[10, p.19]

This thesis considers a multi-label classification problem, meaning the set of functions considered by the learning algorithm take values in the set of possible labels $\mathscr{Y} = \{1, \ldots, k\}$ where k is the number of labels. With the 0-1 loss function defined in (5), the best possible function that minimises the risk $R(\alpha)$ as described in the learning problem definition in 2.2, is then given by the so-called *Bayes classifier* which simply chooses the most likely label for each observation $x \in X$, i.e.[13, p.21]

$$f(x, \alpha_0) = \underset{l \in \mathscr{Y}}{\operatorname{argmax}} P(l|X = x) \tag{9}$$

Now as mentioned, since $P(Y|X)$ is unknown, the Bayes classifier is merely a theoretical construct. By the ERM principle however, this classifier can be approximated using the empirical risk function in (6) which for the 0-1 loss is minimised by the function that makes the smallest amount of prediction errors in the training data.[28, p.33] In more general contexts, the Bayes classifier is simply defined as the function in $\{f(X, \alpha) : \alpha \in \Lambda\}$ that minimises $R(\alpha)$.[10, p.26]

Now, if F_{all} is the set of all possible functions to tackle *any* learning problem, then $F_{\text{model}} \subset F_{\text{all}}$ and the best possible function in F_{all} is the Bayes classifier $f(X, \alpha_0)$. The goal is to find a function that is as close as possible to the Bayes classifier, in terms of minimising the risk. In order to evaluate how close some solution $f(X, \alpha_i)$ is to the Bayes classifier, their difference in risks can be decomposed as follows,

$$R(\alpha_i) - R(\alpha_0) = \underbrace{R(\alpha_i) - R(\alpha_m)}_{\text{estimation error}} + \underbrace{R(\alpha_m) - R(\alpha_0)}_{\text{approximation error}} \tag{10}$$

where $f(X, \alpha_m)$ is the best possible function in F_{model}. The *estimation error*, as an equivalent to statistical variance, represents the uncertainty found in the training data, while the *approximation error* indicates how far away the best possible function for the chosen model and the Bayes classifier are. As such, the latter represents the bias introduced by the choice of model.

Ultimately the difference $R(\alpha_i) - R(\alpha_0)$ should be as small as possible. This translates into the problem of minimising both the estimation as well as approximation error. The problem however is, that if F_{model} is chosen in such a way that $f(X, \alpha_i)$ easily converges to $f(X, \alpha_m)$, i.e. the estimation error (variance) is small, this also means that F_{model} is rather restricted and hence the 'distance' to the Bayes classifier (meaning the approximation error, or bias) is large.[10, p.99] Now, the other way around equally applies, where choosing F_{model} to include a large family of functions likely decreases the approximation error, but at the same time increases the estimation error. This gives rise to the term *Bias-Variance Trade-Off* where choosing an appropriate model for some training data is a matter of balancing the estimation and approximation error.

In terms of ML, this is also often referred to as the problem of over-fitting versus under-fitting. An over-fitted ML model indicates the scenario in which the approximation error is small, due to the choice of a large F_{model}, but at the same time, the estimation error is large. For example, using a 10-degree polynomial to model a dataset that shows largely linear relationships is likely an

over-fitted model.[8] And again, the inverse, whereby F_{model} is small, is called an under-fitted model with high approximation error, but low estimation error.[10, p.100]

To sum this up, SLT and the ERM principle form the foundation for training Machine Learning algorithms that are able to predict some desired target variable based on an available input dataset. The Bias-Variance Trade-Off highlights, how this training process needs to happen in a way that the resulting model has a good generalisation ability and is not over-fitted to the training data itself. The theoretical body on SLT and conditions for obtaining generalising models is far more extensive than what is presented here, but for the purposes of this thesis, I deem the stage set and the next sections move on to a description of the Random Forest algorithm as well as more practical considerations for implementing ML models.

2.4 Random Forest Classifier

A Random Forest (RF) Classifier is a type of ML algorithm that was first introduced by Leo Breiman in [5]. In this paper, he introduces a definition for a Random Forest classifier as follows.

Definition 2.3. A Random Forest is a classifier consisting of a collection of tree-structured classifiers $\{f(X, \alpha_j), j = 1, 2, \ldots, b\}$ where $\{\alpha_j\}$ are iid random vectors and each tree casts a unit vote for the most popular class at input x.

Note that the random vector a_j defines the characteristics of the j^{th} tree in terms of its split variables, end points at each node as well as terminal node values.[13, p.589] These characteristics are elaborated on in the following section.

The mentioned tree-structured classifiers are also known as *Decision Trees*. Since a RF model is a collection of multiple Decision Trees, it forms a combination of simpler 'base models' and as such it is considered an *ensemble method*.[13, p.605] As Decision Trees form the basis for the RF model, the next section is dedicated to them.

2.4.1 Growing a Decision Tree

Decision Trees (DT) describe another set of functions that can be used to solve the learning problem in Definition 2.2. The idea behind them is to partition the feature space $\mathscr{X}$ into the sets $T_1, \ldots, T_t$ such that $T_m \cap T_o = \emptyset$ for $m \neq o, m = 1, \ldots, t$ and $o = 1, \ldots, t$ and offer a prediction for the response y_i for an observation x_i in each T_m according to the function

$$f(x_i, \alpha) = \sum_{m=1}^{t} l_m I(x_i \in T_m) \tag{11}$$

where $l_m \in \mathscr{Y} = \{1, \ldots, k\}$ is the majority label in each set T_m, I is the indicator function and α indicates the configuration of the DT.

While in general a partition of some set can be arbitrary, as long as the partitioning sets are disjoint, in the context of DT, it is preferred to have a partition that can easily be described. Because of this, the partitioning of the feature space happens according to a process called *recursive binary splitting*. In essence, this process starts by considering the entire feature space, splits it into two subsets according to some chosen criteria and then repeats the same process for each newly created subset (also referred to as a *node*) until some stopping condition is met. The sets $T_1, \ldots, T_t$

[8] In very modern contexts, this is also referred to as an *overkill*.

obtained at the last iteration form the partition of $\mathscr{X}$ and are called the *leaves* of the tree, or *terminal nodes.*[13, p.305]

The difficulties arise in deciding how each set should be split. Since the feature space is p-dimensional, there are p options of deciding which variable of $X = (X_1, \ldots, X_p)$ should be considered for each split. In addition, the *split point* s needs to be decided, meaning the threshold for deciding which observations of the training data are collected in which of the two resulting sets after the split. And lastly, the stopping condition for the recursive splitting process needs to be set.

To find the optimal split at each node, a measure for the 'goodness of split' at each node $\widetilde{T}_{m'}$ is required. Recall that the overall aim of solving the learning problem is to find a function that minimises the risk $R(\alpha)$, meaning the expected prediction error. Since a DT predicts the majority label at each terminal node, it is reasonable to relate the 'goodness of split' criterion to a measure of *impurity* $\phi(m')$ of the nodes. This impurity measure characterises the diversity of distinct labels in each node, with a low impurity (indicating high homogeneity of the labels in a node) being preferred. In particular, the proportion of label k observations in any node $\widetilde{T}_{m'}$ is given by

$$p_{m'k} = \frac{1}{\left|\widetilde{T}_{m'}\right|} \sum_{x_i \in \widetilde{T}_{m'}} I\left(y_i = k\right) \tag{12}$$

where $\left|\widetilde{T}_{m'}\right|$ is the number of observations in $\widetilde{T}_{m'}$ and m' indexes all nodes obtained through the splitting process. The majority label $\widetilde{l}_{m'}$ at a node $\widetilde{T}_{m'}$ is then the label that maximises $p_{m'k}$. From this, several node impurity measures can be defined. In [13, p.309] the following ones are mentioned,

- Misclassification error,

$$\phi(m') = \frac{1}{\left|\widetilde{T}_{m'}\right|} \sum_{x_i \in \widetilde{T}_{m'}} I\left(y_i \neq \widetilde{l}_{m'}\right) = 1 - p_{m'\widetilde{l}_{m'}} \tag{13}$$

- Gini index,

$$\phi(m') = \sum_{j=1}^{k} p_{m'k}(1 - p_{m'k}) \tag{14}$$

- Cross-entropy,

$$\phi(m') = -\sum_{j=1}^{k} p_{m'k} \ln(p_{m'k}) \tag{15}$$

where generally the Gini index and cross-entropy are preferred to the misclassification error. As Breiman has shown in [6, p.28-32], minimising the risk for the DT classifier translates to minimising the tree impurity measure which in turn, means that at each node m' the split point s should be chosen in such a way that the decrease of impurity after the split is maximised. Formally, let $\widetilde{T}^{\mathrm{R}}_{m'}$ and $\widetilde{T}^{\mathrm{L}}_{m'}$ be the two sets obtained after splitting the node $\widetilde{T}_{m'}$ at s. Then the decrease of node impurity is given by

$$\Delta_s \phi(m') = \phi(m') - \frac{\left|\widetilde{T}^{\mathrm{R}}_{m'}\right|}{\left|\widetilde{T}_{m'}\right|}\phi(m'^R) - \frac{\left|\widetilde{T}^{\mathrm{L}}_{m'}\right|}{\left|\widetilde{T}_{m'}\right|}\phi(m'^L) \tag{16}$$

and ultimately the optimal splitting point s^* is given by

$$s^* = \underset{s}{\operatorname{argmax}}\, \Delta_s \phi(m') \tag{17}$$

At each iteration of the splitting process, s^* is calculated for each potential splitting variable X_p and the optimal pair is selected as the split that the DT performs.

Lastly, it needs to be decided at which point the recursive binary splitting should be stopped. As is quite intuitive, growing a tree that is very large likely fits the training data rather well, and sometimes too well leading to over-fitting. On the opposite hand, stopping too early and growing a small tree might mean that the algorithm has not learned important patterns, leading to under-fitting.[13, p.307]

Now one could try to introduce another measure during the recursive binary splitting that indicates whether this split should be the last one or not. Similarly to a good gardener, however, that prunes a tree not when the first shoots grow in spring, but after the branches have grown in winter, stopping splits in the DT too early might mean missing out on good splits later on in the fully-grown 'branch'. Because of this, it is the preferred strategy to grow large Decision Trees and stop the splitting only when there is a given number of observations left in the terminal nodes. To avoid over-fitting, the DT is then sufficiently pruned back using a method called *cost complexity pruning*.[13, p.308]

Pruning in the context of DT refers to the process of 'collapsing' internal nodes of the large tree D_0 by taking the union of all subsets to the according node. In order to govern this pruning, a *cost complexity criterion* for the tree $D \subseteq D_0$ is introduced as follows

$$C_\theta(D) = \sum_{m=1}^{t} |T_m|\, \phi(m) + \theta t \tag{18}$$

where θ is a *tuning parameter* that controls the complexity of the DT model. In order to achieve a balanced fit, one that is neither over- nor under-fitted, θ should be chosen so that $C_\theta(D)$ is minimised. This best-possible parameter is θ^* and it is obtained through so-called *weakest link pruning*. In this process, a sequence of sub-trees to D_0 is obtained by iteratively collapsing the internal nodes of the tree which give the smallest per-node increase in $\sum_{m=1}^{t} |T_m|\, \phi(m)$. This is repeated until only a single-node tree is left and out of the resulting sequence of trees, the one that minimises $C_\theta(D)$ is chosen as the final DT, D_{θ^*}.[13, p.308]

Now, while [13] identifies θ as the only tuning parameter of the DT model, as [25] shows, in practise further parameters need to be set in order to fit a DT to the training data. These parameters that are not adaptively set by the learning algorithm itself are called *hyper-parameters*. For Decision Trees these hyper-parameters include, among others, the chosen impurity measure $\phi(m')$, the minimum amount of observations in terminal nodes and the maximal number of terminal nodes allowed. These parameters, together with the tuning parameter that the model learns itself and the obtained leaf nodes T_m, $m = 1, \ldots, t$ with their majority labels l_m, then are described in the vector α which characterises the DT classifier in (11).

Lastly, even though Decision Trees are 'pruned back' to combat the issue of over-fitting, they are still prone to having high estimation errors (variance), since a small change in the training data might mean a very different selection of optimal splitting points. To improve this, one can combine several DTs to achieve a single prediction.[13, p.312] This is the case in *bagging* and its modification, the Random Forest model. The next section is dedicated to this ensemble method.

2.4.2 Combining Multiple Decision Trees

As is mentioned in the previous section, Decision Trees themselves generally suffer from high variance as they are very sensitive to changes in the training data. A method to improve this, is to grow multiple DTs and then combine their results for a better prediction. One such method is *bagging*, also called *bootstrap aggregation*. What it does, is to create bootstrap samples of the training data, grow a DT on each sample and ultimately combine the predictions of each.[13, p.587] These bootstrap samples are of the same size as the training data and are obtained by randomly drawing observations with replacement.[13, p.249]

Since bagged trees are correlated, the variance reduction of this method is limited by the correlation itself. Random Forest models then offer an improvement to that, by selecting random input variables for the training of each DT on the bootstrap samples, thereby growing trees that are less correlated.[13, p.588]

As stated in Definition 2.3, for a classification problem, the outputs of the DTs grown in the RF model are combined through determining the majority vote cast by the trees. In addition to the tuning and hyper-parameters mentioned in the previous section, the characterising vector α for each DT, must contain the number of randomly selected input variables before each split $\eta \leq p$ as well. Often, $\eta = \sqrt{p}$ is selected.[13, p.589]

2.4.3 Prediction Probabilities in the Implementation

As a last note to RF models it is worth mentioning that in the implementation of the algorithm used in this thesis (the Scikit-learn library `RandomForestClassifier` [21]), for each predicted label of a data point x_i an additional *prediction probability* p_i^{RF} has been calculated. This prediction probability for a single tree corresponds to the proportion of observations that are of that label in the corresponding terminal node, as defined in Equation (12). To obtain a prediction probability for the RF output l_i^{RF}, the individual tree prediction probabilities have been averaged. That is, if l_i^j is the predicted label of the j^{th} tree for input x_i, the RF prediction probability for x_i is given by

$$p_i^{RF} = \frac{1}{b}\sum_{j=1}^{b} p_{m_i l_i^j} = \frac{1}{b}\sum_{j=1}^{b} \frac{L_i^j}{|T_{m_i}|} \tag{19}$$

where T_{m_i} is the terminal node to which x_i belongs and L_i^j is the number of training data observations in T_{m_i} that are of label l_i^j, i.e. $L_i^j = \sum_{x_q \in T_{m_i}} I(y_q = l_i^j)$.

This concludes the theory section on Random Forest models and the next section moves on to some practical considerations for building ML solutions.

2.5 Machine Learning Tools

This section describes a selection of methods commonly used when developing Machine Learning algorithms. It is by far not exhaustive on the different tools and techniques applied in ML and rather focuses on the ones relevant to this thesis. I divide them into three sub-sections. The first one focuses on data preparation techniques, i.e. methods required to prepare the data in order for it to comply with the standards the ML implementation expects. The second details improvement methods of the training part and lastly, I describe how the ML model can be evaluated on a test data set.

2.5.1 Data Preparation Techniques

In this thesis, I make use of publicly available libraries in the programming language Python that have implemented a wide array of ML models, including Random Forest classifiers. Section 3.1 elaborates on this. These libraries often define more or less narrowly how they expect data to be passed on to them. Often enough, it is required for the data to be a real-valued matrix. Since the dataset I used for this thesis contains a lot of categorical variables, including the prediction target (the response Y), it is necessary for these to be encoded. The simplest type of encoding comes in the form of *Label Encoding* in which a non-negative integer is assigned to each unique category.[24] This is actually assumed to be the case already in the formulation of the classification problem in Section 2.3 where the set of possible labels G is described as being the set $\{1, \dots, k\}$ for k unique labels. As is described in the Application later on, the response variable is actually of categorical nature and the set G is the label-encoded stand in.

As an alternative to label encoding, it is also possible to introduce a binary variable for each distinct label, indicating whether or not a certain observation belongs to each label or not. This is called *One-Hot Encoding*. More formally, let X_j be a column in the $n \times p$-matrix of input data X that can take k_j unique categorical values. For each unique value c in X_j a new column $X_{j,c}$ is introduced which then can take values in $\{0, 1\}$ depending on whether or not each observation in X_j is of category c or not. This results in k_j new columns for each categorical variable X_j in X.

Besides the new columns that get created when using One-Hot Encoding as opposed to Label Encoding, another difference between the two methods is that Label Encoding preserves the ordinal nature of the original categorical values, while One-Hot Encoding represents categories without any hierarchy between them. This means that Label Encoding is preferred when an ordinal relationship of the categories in a variable is present and vice versa.

2.5.2 ML Training Improvements

While there are many methods to improve a ML algorithm, I have applied only two different ones and these are described here.

First off, due to the optimal split point search in the tree growing process, tree-based ML algorithms are sensitive to a large number of input variables. In order to reduce the number of features in the input data, one can select the ones deemed more relevant in successfully predicting the target and simply remove the others. This is called *feature selection*. For this thesis, I have made use of the `feature_importance` property of the Scikit-learn implementation for the RF model which calculates the importance of each feature based on the summed reduction of the node impurity measure ϕ (in this case the Gini index) each feature achieves.[13, p.593] The feature importance across all input variables is normalised to the interval $[0, 1]$ to obtain the relative feature importances.[21] Finally only those features that score higher than some set threshold are selected.

Another issue often encountered in multi-label classification problems is that of *class imbalances*. This occurs, when the frequencies of each label in the training data are not roughly equal and there are labels that clearly are observed more often than others. Due to the over-abundance of one or a few labels, the learning algorithm might become biased towards the majority classes which can lead to poorer generalisation ability.[30] In order to address this problem, *class weights* are introduced. In the Scikit-learn implementation, the weight w_l for the label l is calculated according to

$$w_l = \frac{n}{k \sum_{i=1}^{n} I(y_i = l)} \tag{20}$$

where n is the number of observations in the training data and k the number of distinct labels. For each split in the tree growing process, the decrease in node impurity of Equation (16) is then evaluated through the weighted sums.[25]

2.5.3 Evaluation Methods

This last section of the theoretical background describes several methods to assess the generalisation ability of the learning model. Recall from Section 2.3 that a model is said to generalise well, if it performs similarly over the training data, as well as new, previously unseen *test data*. In order to ultimately decide, if the trained ML algorithm offers a good approximation to the true relationship between input data and the response, the model performance on a test dataset is evaluated.

Also introduced in the previous sections is the loss function that measures the discrepancy between the actual response and the prediction output. Since this thesis deals with a multi-label classification problem, the loss function that details the number of misclassifications can be seen as a $k \times k$ matrix C (where k is the number of unique labels). Each element of the matrix is then given by

$$C_{s,t} = \sum_{i=1}^{n_{\text{test}}} I\left(y_i = l_s, f(x_i) = l_t\right) \tag{21}$$

where l_s and l_t are the s^{th} and t^{th} label respectively and n_{test} is the number of observations in the test data set. This matrix is also called the *confusion matrix*. The diagonal elements of C characterise correct predictions, whereas all non-diagonal elements show misclassifications. From this, the *accuracy* of the model can be calculated as follows

$$A = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} I\left(y_i = f(x_i)\right) \tag{22}$$

As there are different types of misclassifications, one considers the following four cases for a label l,

$$TP_l = \sum_{i=1}^{n_{\text{test}}} I(y_i = l, f(x_i) = l) \tag{23}$$

$$FN_l = \sum_{i=1}^{n_{\text{test}}} I(y_i = l, f(x_i) \neq l) \tag{24}$$

$$FP_l = \sum_{i=1}^{n_{\text{test}}} I(y_i \neq l, f(x_i) = l) \tag{25}$$

$$TN_l = \sum_{i=1}^{n_{\text{test}}} I(y_i \neq l, f(x_i) \neq l) \tag{26}$$

where TP indicates the number of *true positives*, FN the *false negatives*, FP stands for *false positives* and lastly, TN indicates *true negatives*. These four counts are summarised by the quantities *recall* R (also called *sensitivity* or *true positive rate*) and *precision* P which for the label l are given by

$$R_l = \frac{TP_l}{TP_l + FN_l} \text{ and } P_l = \frac{TP_l}{TP_l + FP_l} \tag{27}$$

Additionally, *specificity* S (or *true negative rate*) may be defined as

$$S = \frac{TN}{TN + FP} \tag{28}$$

From this, the *false positive rate* is given by $1 - S$. Now, the accuracy measure A is sensitive to class imbalances, as it will be dominated by the majority class of the test dataset. So instead, a *balanced accuracy* can be calculated by

$$A_{\text{balanced}} = \frac{1}{k} \sum_{l=1}^{k} R_l \tag{29}$$

as is described in [22].

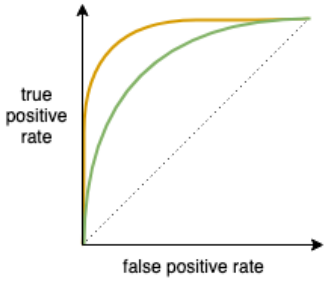


Figure 1: ROC curve for two illustrative binary classification models (green and orange). The performance of the orange model is considered better than the green one. The dashed line indicates a model that predicts labels by chance.

Lastly, the true positive rate and false positive rate may be plotted against each other for varying thresholds of prediction probabilities. That is, for different choices of thresholds of the RF prediction probability as defined in Equation (19) to decide on the majority label, the resulting true positive and false positive rates are calculated and plotted against each other in the so-called *receiver operating characteristics* curve (ROC). This is illustrated in Figure 1. The larger the area under the curve is, meaning the closer the curve runs to the top left corner of the graph, the better the performance of the model. To quantify the ROC curve, the area under the curve (AUC) for a model is calculated. The AUC then serves as another measure to evaluate model performance.

Now, since the true positive and false positive rate are calculated for each label, meaning they consider the binary case, an extension to the multi-label classification case is necessary. This is achieved by a pairwise comparison of AUC scores for the distinct labels, namely the multi-label AUC score is calculated as the average AUC of all possible pairwise combinations of labels.[23]

This concludes the section on Machine Learning tools that are relevant to this thesis. The following part focuses on the application of the methods presented here to the image recognition data with the aim of assessing its data quality.

3 Application and Results

While the previous section dives into the theoretical intricacies relevant to the thesis problem, this section now describes the details of my chosen approach. It is ultimately my goal to first, assess how well the scraped values represent the true one as it is printed in the financial statements and then, to offer a suggestion which of the different scraper values that are available for each financial figure should be the one to be selected in the final dataset. To accomplish this, I divide my approach into three steps which are described below.

Step 1. I start out by investigating how the image recognition outputs differ from the actual values. Based on that knowledge, I define the data quality aspect that I wish to identify. This quality aspect is represented through so-called *error labels* which capture the differences between scraped and true values that I am able to identify systematically. Since the true values are technically only available as the original image files, I make use of a secondary source to serve as an approximation to the true values.

Step 2. In this step, I then build a statistical model to predict the previously introduced error labels. This allows me to be able to assess the data quality of the image recognition outputs without the need for a secondary source. The statistical model in question is a Random Forest classifier.

Step 3. Lastly, in order to make a selection for the best possible scraped value to represent the true one, I quantify my quality assessment by introducing a *quality score*. The scraped value which obtains the highest score is ultimately selected as my suggestion for the best possible representation of the true financial figure. I then contrast my results with an alternative model for the value selection.

3.1 About the Implementation

I wrote the scripts for this work in Python (Version 3.11.5) [26], since first and foremost I am most comfortable with this programming language and second because it is an industry standard, used by the company I work for, and many extensive and well-documented Machine Language libraries exist for it. In particular, I used the Anaconda [1] distribution package and created a Jupyter [15] notebook for my work. Furthermore, I have made use of the Python packages Scikit-learn (Version 1.3.0) [18], Pandas (Version 2.0.3) [17], NumPy (Version 1.24.3) [12] and Matplotlib (Version 3.7.2) [14].

3.2 Types of Error in the Image Recognition

This section starts out by describing Step 1 of my approach, understanding the data quality aspects relevant for the image extraction problem and introducing a method to systematically identify them.

As previously mentioned, when using Machine Learning algorithms to predict whether a numerical value is of good quality or not, it is easy to think of it as a regression problem - after all, I am trying to see how close an extracted number is to the true one. I could tackle this problem by making a prediction for what the true value of each financial figure in a particular report should be and then select the value across the different scrapers that is closest to the predicted one. And indeed, I

have tried that, but with little success. The predictions were just not very good, likely because the available training data was not very suited for a regression task.

Additionally, it is not necessarily the numerical aspect of the scraped financial figure values that indicates whether the image recognition was successful or not. Instead, it is the simple fact of whether or not the string of digits that was extracted matches the string of digits that was written in the report itself, that indicates the successful extraction of information. Because of this, I have chosen to tackle this problem as a classification task instead.

In the most simple problem statement, identifying whether extracted and true values match is of binary nature. True, whenever the two values match and False, whenever they do not. Being able to correctly differentiate those two scenarios is a powerful tool to evaluate the quality of the image extraction data. However, by only considering each value in its entirety, it does not offer a lot of granularity on why a scraped value might be assigned a True or 'matching-reality' label or vise versa. From a binary label alone, one cannot answer questions such as: Do the values not match, because one digit is wrong? Or are none of the digits matching each other? Since it is unlikely that the predictions would reach perfect accuracy, this lack of granularity could easily mean that a lot of scraped values get mistakenly disregarded as not good enough or even worse, incorrectly considered as being of good quality and therefore assumed to accurately reflect the original financial figure value.

While less than perfect accuracy is a concern for all predictive modelling, the binary case in addition to that only considers each value in its entirety and labels the whole value as matching reality or not, when actually the image recognition algorithm often only gets *part* of a number wrong. For example, the true value might read 597895 and the image recognition extracts the value 59789**6**, where the last 5 is incorrectly scraped as a 6. When comparing these two numbers as a whole, the binary label would be False or 'mismatching-reality' as in fact, those two numbers do not match precisely. However, a single wrong digit, especially in the last position, is not of grave consequence in this case. For the purpose of creating a digital representation of the images of financial statements, it is better to have a number that only has a single wrong digit than to have no representation of the financial figure at all.[9]

I have therefore chosen to predict not just whether a value matches its true one, but which part of it matches. The parts of a number that I have considered for this problem are the sign, the string of significant digits (that is the number excluding leading and/or trailing zeros) and the order of magnitude (OoM). Note that for most of the extracted values it is not possible to infer the OoM from the string of significant digits alone, because financial reports often state their numbers with differing currencies. For example, a financial figure value might be printed as 5672 (corresponding to the string of significant digits from which the inferred OoM would be 3) while the currency is given as KSEK, meaning the actual value is 5672000 SEK (corresponding to a correct OoM of 6).

While the sign is certainly important for the correct interpretation of a financial figure value, unfortunately the reporting standards for annual accounts regarding the presentation of negative values are not unified. Sometimes, when reporting a loss it is assumed that the stated number is negative and because of this, the minus is omitted. In other, albeit rare, cases, the minus is printed on the right side of a number and then again, a minus might indicate a double-negative

[9] It is worth to note that the position of such a single wrong digit is significant, however. Representing a company's revenue that is truly 597895 SEK as 59789**6** SEK or **6**97895 SEK does make a difference. Unfortunately though, while my solution did have moderate success at identifying a single wrong digit, additionally identifying the *position* of the wrong digit goes beyond its capabilities. The different types of mistakes that can happen in the string of digits are discussed in more detail in a later section.

for financial figures that are inherently negative values. This varied approach to the reporting of negative amounts unfortunately means that the minus sign becomes unreliable. And while it is true that my problem statement only concerns itself with the correct extraction of information and not the assigning of meaning to the extracted information, i.e. the minus sign is just another character to be correctly or incorrectly scraped independent of what it means for the financial figure value, the secondary source is equally affected by the lack of strict reporting standards for negative values and I therefore deem it unreliable for representing the true value. Because of this, I have only considered the string of significant digits and the order of magnitude to create my measure of quality, which I call the *error labels*. These error labels reflect how well each part of the scraped value matches the corresponding secondary source value, of which the latter serves as the approximation to the true value as it is in the image of the financial report itself. It is the goal of my approach to create a Machine Learning model, in particular a Random Forest Classifier, that can predict these error labels with sufficient accuracy, so that the quality of future scraped values can be evaluated without a corresponding secondary source value being available.

3.2.1 Examples of Scraper Mistakes

Before the error labels themselves get created (and described in this text), I want to take a step back and introduce the heuristic on which they are based. Namely, the types of things that go wrong in the image extraction part, the *scraper mistakes*. These scraper mistakes are instances in which the scraped value does not match the corresponding value that is printed in the PDF which is considered the true value. As such, the scraper mistakes are the actual misrepresentations of reality in the data that I want to identify. In order to do so, I started out with a manual inspection of a random set of reports and compared the scraper outputs to them. A selection of mistakes I found when doing this can be seen in Table 2. While there are many different mistakes that happen in the image recognition process, some occur more frequently than others. I can certainly not claim to have found them all, as the sheer amount of financial statements is just too great to go through all of them manually. That being said, the more frequent scraper mistakes can generally be grouped into the following categories:

- Digit mistakes
- Mistakes in the order of magnitude (OoM)
- Positioning mistakes
- Combinations of multiple mistakes

In the category of **digit mistakes**, one might find problems in which the scraper only extracted parts of the string of digits, with either the beginning or ending digits missing. Examples of these can be seen in Figure 5 in the appendix. Alternatively, it is only a single digit (or not more than a few) that has been extracted wrongly, as is the case in Figure 7, equally included in the appendix (as are the following figures displaying scraper mistakes).

For the problem of **mismatching order of magnitudes** between scraped and true value, generally speaking the OoM is either the same or it differs. There are some special cases of differing OoMs that occur frequently. One of these is an OoM of the scraped value that is off by a factor of 1000, since financial figure values are often stated in currency values of SEK, KSEK (where 1 KSEK = 1 000 SEK) or MSEK (where 1 MSEK = 1 000 000 SEK), see Figure 6c. Additionally

to that, sometimes the currency abbreviation is given in Swedish, e.g. with Tkr as the equivalent to KSEK (see Figure 6d) or it is stated not as an abbreviation at all, but is written as part of a sentence somewhere else in the report (see Figure 6a and Figure 6b). These non-standardised ways of stating the currency of financial figure values pose quite the challenge in correctly extracting the OoM and therefore often result in scraper outputs that are off by a factor of 1000.

Additionally, the OoM might be wrong when the scraper extracts numbers that are not financial figures at all, like dates, IDs or page numbers. This was particularly troublesome in early versions of the scrapers, but has been addresses in more recent versions so that these mistakes occur less frequently. One type of mistake that remains however, is the case of two column values being extracted as one (see Figure 9d). This often results in scraper values with extreme OoMs.

Another common mistake is made in regards to the **positioning of values** and their meaning, i.e. the financial figure name or corresponding time interval. Examples of this are given in Figure 8 where the scraper either swapped the columns, i.e. reading the value stated for the previous financial year instead of the most recent one, or extracted the wrong line, when the financial figure is a summed item and the figure name is not written in the same line as its value.

Lastly, any of these mistakes can of course happen in **combination** with others. For example, a single wrong digit together with a wrong OoM as in Figure 9a. Or, the scraped and true value differ greatly, so that it is not actually apparent which mistakes were made as is the case in the remaining images of Figure 9.

As stated above, ultimately it is the goal to identify these scraper mistakes as each poses a data quality issue. Namely, for each wrongly scraped number, the financial reality of that company gets misrepresented. And if the wrong number gets used to create abstractions based on it, e.g. in the form of credit risk assessments, the mistake gets exacerbated and potentially inaccurate or plain wrong consequences are being drawn. Of course, this is relative to the type of mistake and the quantity of financial figures in a report, but the principle persists. The question now becomes, how can one identify these scraper mistakes in an automated manner without the need for manual inspection? This is where the error labels and the available secondary source come into play. While generally, the true values are only given in the non-digitised (and hence, 'not-automatable') PDF format, in this particular problem I am lucky enough to have a secondary source available that can serve as an approximation for the true values. This secondary source is independent from the scraping process, and while it might have its own issues (as described in Section 4.1), it is sufficiently close to the true values to be used as a representation of them.

With this, I can compare each scraped value to the corresponding secondary source one (as a stand-in for the true value), which then allows me to identify mistakes in the digits and the order of magnitude. It does not however, offer context as to which values are 'around' each other. And by that I mean the positioning context that one obtains when looking at each value as it is printed in the PDF, with the financial figure names as well as applicable time intervals in a semi-tabular structure. This means that it is not possible to identify the scraper mistakes pertaining to the positioning of a value and its meaning by simply comparing the scraped and secondary source number. As is described in the following section, the error labels therefore only capture mismatches in the string of digits and/or the order of magnitude. A scraper mistake caused be the incorrect reading of a column or line therefore most often will only show up as a mistake of mismatching digits/the OoMs. With my solution as is, it is not possible to truly differentiate these cases and similarly so for combined scraper mistakes. That being said, it is possible to expand the error labels to potentially be able to identify positioning mistakes as well. For example, if the previous year's

value of a financial figure is available in the secondary source, one could check the scraper value against that previous value to see, if a potential 'column swap' happened. Additionally, several financial figures are related to each other, e.g. one being the sum of others, and to check for mistakes in which the wrong line was read, it might be possible to check a scraped value against all secondary source values for the same report to see if any matches can be found. This would then indicate a 'line swap'. Beyond that, this could also be a targeted effort, by utilising domain knowledge around which financial figures are related / stated close to each other in the report, e.g. only checking balance sheet figures against each other versus what is printed in the income statement, or even more granular by looking at the 'sum total of assets' and all assets sub-figures etc.

Table 2: Sample of scraper mistakes together with their true value and assigned error label

Fig.	Scraper Mistake	Financial Figure	Financial Figure (SE)	Scraped Value	Secondary Source Value	True Value	Error Label
5a	missing_first_digits	short_term_receivables _from_sales_and_services	Kundfordringar	902	361000	360902	meh digits, 1000 OoM diff
5b	missing_first_digits	equity	Eget Kapital	279	291000	291279	meh digits, 1000 OoM diff
5c	missing_last_digits	equity	Eget Kapital	541	542000	541645	good digits, 1000 OoM diff
5d	missing_last_digits	ebit	Rörelseresultat	5338	5338000	5338325	good digits, 1000 OoM diff
9a	multiple	short_term_debt	Kortfristiga Skulder	2101	9101000	9101000	meh digits, 1000 OoM diff
6a	oom_elsewhere	ebit	Rörelseresultat	31444	31444000	31444000	good digits, 1000 OoM diff
6b	oom_elsewhere	short_term_receivables	Kortfristiga Fordringar	428785000	429000	428785	meh digits, 1000 OoM diff
6c	oom_elsewhere	profit_loss_before_tax	Resultat Före Skatt	913	913000	913000	good digits, 1000 OoM diff
6d	oom_missing	depreciation	Avskrivningar Och Nedskrivningar Av Materiella Och Immateriella Anläggningstillgångar	59035	59035000	-59035000	good digits, 1000 OoM diff
9d	two_columns_together	profit_loss_before_tax	Resultat Före Skatt	251434589855	252000	251434	extreme scraper OoM
9b	unclear	liabilities_and_equity	Eget Kapital Och Skulder	1	1618000	1617570	meh digits, wrong OoM
9b	unclear	short_term_debt	Kortfristiga Skulder	18	393000	393394	meh digits, wrong OoM
9c	unclear	short_term_debt	Kortfristiga Skulder	10112	17418000	17418000	meh digits, 1000 OoM diff
8a	wrong_column	depreciation	Avskrivningar Och Nedskrivningar Av Materiella Och Immateriella Anläggningstillgångar	4670	4307000	-4307116	meh digits, 1000 OoM diff
8a	wrong_column	short_term_debt	Kortfristiga Skulder	3884	7506000	7506135	meh digits, 1000 OoM diff
7a	wrong_digit	operating_costs	Övriga Externa Kostnader	7505	6000	-5750	meh digits, exact OoM
7b	wrong_digit	short_term_receivables	Kortfristiga Fordringar	9010639	9911000	9910699	meh digits, exact OoM
7c	wrong_digit	liabilities_and_equity	Eget Kapital Och Skulder	278356	328000	328356	meh digits, exact OoM
8c	wrong_line	short_term_debt	Kortfristiga Skulder	9456	40000	40196	meh digits, wrong OoM
8d	wrong_line	property_plant _and_equipment	Materiella Anläggningstillgångar	61118000	80612000	80612000	meh digits, exact OoM

As these examples illustrate, some scraper mistakes are easier to identify than others. The error labels will therefore group several different scraper mistakes together, as can be seen in Table 2 above. The next section describes in more detail how the error labels get created.

3.2.2 Assigning Error Labels

As described above, to evaluate the quality of the scraped values I compare two parts of each number, its order of magnitude (OoM) and the string of significant digits (SSD), to the equivalent parts of the secondary source number. I start out by introducing slight modifications to both values. These are namely:

- Taking the absolute value of both values
- Strip the values of trailing and leading zeros to obtain the SSD
- Round the scraped value to the nearest thousand
- Strip the rounded scraper value of trailing and leading zeros
- Remove the first digit from the secondary source value (which is done to potentially identify cases in which the image extraction missed the first digit, see examples in Figure 5a and Figure 5b in the Appendix)

Following this, I calculate some derived data points:

- Difference between secondary source and scraper value as well as secondary source and rounded scraper value
- OoM of both values and their difference
- Similarity of the SSDs (both for rounded and exact scraper value versus the secondary source value) using the `SequenceMatcher` from the Python library `difflib`

Using these new columns, I create two flags: one to represent how well the OoMs match, the *OoM flag*, and one for the quality of matching the SSD, the *digit match flag*. These two flags can take a limited number of values which can be see in Table 5. In particular, the OoM flag values are assigned as shown in Table 3 and similarly, for the digit match flag, I assign the values as described in Table 4.

Table 3: OoM flag values with a descriptions of the assignment conditions

OoM flag	**Description**
exact OoM	The OoM difference between scraped and secondary source value is exactly 0
1000 OoM diff	The absolute OoM difference is exactly 3
extreme scraper OoM	The absolute OoM difference is greater than 4 and the scraper OoM is greater than 9
wrong OoM	Any remaining case

Table 4: Digit match flag values with a descriptions of the assignment conditions

Digit match flag	Description
good digits	SSDs match exactly SSD of the rounded scraper value matches the secondary source SSD exactly SSD difference is exactly 1 Scraped value equals the secondary source value divided by 1000 Scraped value or rounded scraped value matches the secondary source value without the first digit
ok digits	No good digit match has been determined and the digit similarity (for either exact or rounded scraper value) is bigger than 0.5
bad digits	Any remaining case

Table 5: Assigned error labels with number of occurrences and underlying order of magnitude (OoM) and digit match flags

OoM Flag	Digit Match Flag	Occurrences	Error Label	Aggregated Occurrences
wrong OoM	good digits	3229	good digits, wrong OoM	3229
	bad digits	26894	meh digits, wrong OoM	35509
	ok digits	8615		
extreme scraper OoM	good digits	2	extreme scraper OoM	2379
	bad digits	1797		
	ok digits	580		
exact OoM	good digits	182480	good digits, exact OoM	182480
	bad digits	18844	meh digits, exact OoM	30425
	ok digits	11581		
1000 OoM diff	good digits	19918	good digits, 1000 OoM diff	19918
	bad digits	8081	meh digits, 1000 OoM diff	12319
	ok digits	4238		

The error label is then just the combination of both the OoM flag and the digit match flag - with the slight modifications that first, I have further grouped the labels 'ok digits' and 'bad digits' into the eloquent label 'meh digits' and second, whenever the scraper OoM is extreme, it constitutes a clear domain knowledge violation[10] and the value should be disregarded independent of the quality of the SSD match, so the error label in this case is just the OoM flag. I have introduced both these groupings in the error label to avoid too small classes (i.e. with too few occurrences in the data), since imbalanced classes can be tricky for the prediction algorithm to handle well.

Since these error labels now serve as the target for the prediction, how well defined they are, meaning how well they capture the quality of the image extraction, is crucial for the success of this approach to solve the ultimate problem - that of selecting the best possible value across scrapers to most closely represent the true financial figure value. Over the course of developing a solution to this problem I went through many iterations of identifying and grouping error labels and finally arrived at what is presented here. One could say that these error labels are still quite basic and

[10]Values with the OoM exceeding 9 correspond to financial figure values in the trillions of SEK, since reports often state their values with the currency as TSEK (thousand SEK) or MSEK (million SEK). Often, the scraper values in these magnitudes happen when two separate values get appended to each other and read as a single one.

only capture a part of the types of mistakes the scrapers make. For example, the case of when a scraper reads the wrong line of the report (it extracts a correct value, but it gets assigned to the wrong financial figure), does not get identified by my error labels and instead is simply put into the 'bad digits' category. Trying to expand and refine the error labels is therefore a suited starting point for introducing future improvements to this project.

3.3 Random Forest Classification

The previous section is dedicated to Step 1 of my approach, that is defining the target of the prediction. It therefore answers the question: what is the quality aspect of the image recognition outputs that I want to be able to assess through a statistical model? For this quality aspect, I have chosen the error labels which represent the match between SSD and OoM between scraped numbers and their true counterpart. This section then moves on to Step 2 of the approach, namely the prediction of these error labels.

As is described above, the error label creation depends on a secondary source that serves as a sufficient approximation to the true values. Now this secondary source is only available for a subset of all financial data and acquiring more, e.g. for next year's release of financial statements, is expensive, making the predictive assessment of data quality an attractive option (and hence, giving rise to this thesis project). As representatives of the data quality of the image recognition outputs, the error labels serve as the target of this prediction. The particular predictive algorithm in question is a Random Forest Classifier which this section describes in more detail.

3.3.1 Feature Selection, Encoding and Model Configuration

In Section 2.1 I have introduced different concepts on how to define *data* and how to classify datasets. As is established there, many distinct perspectives on data exist, each offering a different flavour of a definition. For the purposes of this thesis, and in particular statistical modelling, the relevant ones are: that this dataset is structured, contains numerical as well as categorical columns and any column can potentially be `NaN` (not a number, also called a `NULL` value or a *blank*).

The categorical columns require encoding in order to be used as input to the Random Forest Classifier method of the `sklearn.ensemble` library. This has been done using the `OneHotEncoder` function of the `preprocessing` module, equally from the `sklearn` library. The workings of One-Hot Encoding are described in Section 2.5.1. While also being of categorical nature, the error label as the prediction target has been encoded using a simple mapping to natural numbers as is shown in Table 6. This is done to keep the target univariate, instead of creating a multi-variate target as would be the case, if One-Hot Encoding would have been applied.

Table 6: Error labels together with their encoded label

Error Label	**Encoded Error Label**
good digits, exact OoM	0
meh digits, 1000 OoM diff	1
meh digits, wrong OoM	2
meh digits, exact OoM	3
extreme scraper OoM	4
good digits, wrong OoM	5
good digits, 1000 OoM diff	6

The resulting dataset after encoding contains 424 columns and 286259 rows, where rows containing any NULL values have been removed. Each row represents a single scraped financial figure value and the same financial figure of the same report can be represented by different scrapers as multiple rows. This dataset was further split into three parts: a training dataset for fitting the Random Forest Classifier, a test set for evaluating its performance and ultimately a validation set that has been kept aside from the training process to evaluate Step 3 of my approach, the selection of best possible value across scrapers (see the next section).

Note that the split into three datasets has not been done purely random. Since a single financial report contains multiple financial figures of which several are related to each other, selecting rows randomly to split the dataset into three, would mean running the risk of having rows in the test and training dataset be correlated with each other. So instead, to ensure that training, test and validation data are independent of each other, I have selected the report IDs at random. This gives three sets of report IDs, forming a partition to the full set of unique report IDs, and the data is then divided by selecting all rows belonging to each set of random report IDs. This of course means that some rows within each subset are still correlated with each other, see Section 4.1 for more details on this problem.

Lastly, not all of the columns that are available have been selected as input to the final Random Forest model, as these excluded features likely do not carry any explanatory power for predicting the error labels. See Table 7 for descriptions of the available and ultimately selected features. Feature selection has been performed on the training dataset as described in Section 2.5.2, giving rise to the following three dataset sizes (where the 36+1 columns indicate 36 features and the single target),

- **Training data**: $96506 \times (36 + 1)$
- **Test data**: $46850 \times (36 + 1)$
- **Validation data**: $142903 \times (36 + 1)$

After the dataset has been prepared and split, the training data is ready for the actual model fitting and prediction of error labels. The chosen parameters for the Random Forest classifier are as follows:

- `n_estimators = 1000` (meaning the RF model grows 1000 DTs)
- `ccp_alpha = 0.00001` (referring to the tuning parameter θ in the cost complexity criterion $C_\theta(D)$ of Equation (18) in Section 2.4.1)
- `bootstrap = True` (Bootstrap samples are drawn to grow the individual DTs; note that the independence of the samples is limited by the fact that rows in the training data are correlated, since multiple scrapers aim to extract the same financial figure and several financial figures stem from the same PDF, for more detail refer to Section 4.1)

where all remaining parameters have been chosen as the default defined by the `RandomForestClassifier` function itself. Without mentioning all, the node impurity measure ϕ has been chosen to be the Gini index, the number of features considered when identifying the best possible split to be $\eta = \sqrt{p}$ where p is the total number of (encoded) features and the minimal number of observations in a leaf node is set to 1. Note that the class weights as defined in Equation (20) in Section 2.5.2 are introduced to the RF model through setting the `sample_weight` parameter of the `fit` method.

The next section details the results obtained with this RF model.

Table 7: List of available as well as selected features for the final Random Forest classifier

Dataset Column	Considered for Training Data	Included in Final Feature Selection	Column Data Type	Description	Comment
report_id	FALSE	FALSE	Numerical	Unique identifier for the financial statement	The values this column takes are numerical, but the meaning of these numbers is categorical, e.g. an ID of 4568021 is a numerical value, but it carries no ordinal meaning.
company_id	FALSE	FALSE	Numerical	Unique identifier for the company	See row above.
company_name	FALSE	FALSE	Categorical	Name of the company	
date_of_incorporation	FALSE	FALSE	Timestamp	Date at which the company was incorporated	Can be either numerical (e.g. days since 01-01-1980 which equals Unixtime) or categorical (e.g. as a string {1992-01-06})
company_age	TRUE	TRUE	Numerical	Age of the company	Days since date_of_incorporation (as calculated on 2024-02-05)
company_status	TRUE	FALSE	Categorical	Life cycle stage of the company, e.g. is it currently active and conducting business, is it being dissolved or going through bankruptcy proceedings, etc	
company_status_valid_from	FALSE	FALSE	Timestamp	Start date from which the current status is applicable	
company_type	TRUE	FALSE	Categorical	Legal type of the company, e.g. 'AB' standing for 'Aktiebolag'	
employees_number	TRUE	TRUE	Numerical	Number of employees of the company	
industry_code	TRUE	TRUE*	Categorical	Associated main industry for the company	*only some of the one-hot encoded columns for the possible values are included
currency	TRUE	FALSE	Categorical	Currency used in the financial statement	
vat	TRUE	TRUE	Binary	Whether or not the company is registered for VAT	
source_scraper	TRUE	TRUE	Categorical	Version of the scraper	
report_start_date	FALSE	FALSE	Timestamp	Start date of the company's financial year	
report_end_date	FALSE	FALSE	Timestamp	End date of the company's financial year	
report_length_days	TRUE	TRUE	Numerical	Days between start and end date of the financial year	Derived from report_start_date and report_end_date and not included in original data
time_of_publication	FALSE	FALSE	Timestamp	Date at which the report was published	
financial_figure	TRUE	TRUE*	Categorical	Name of the financial figure	*only some of the one-hot encoded columns for the possible values are included
scraped_value	TRUE	TRUE	Numerical	Value of the financial figure as extracted by the scraper	
secondary_source_value	FALSE	FALSE	Numerical	Corresponding secondary source value	
error_label	TRUE	TRUE	Categorical	Data quality measure and prediction target	

3.3.2 Classification Results

With the RF model as defined in the previous section, a sample of resulting predictions can be seen in Table 8. What is apparent from this sample is the overabundance of 'good digits, exact OoM' error labels. This is in-line with the label occurrences shown in Table 5 and additionally, this result is reasonable given that it is the aim of the image recognition algorithm to produce values that actually match what is printed in the report. In this very small sample of table, 6 out of 10 labels are predicted correctly and the others are not.

Table 8: A sample of actual error labels together with their Random Forest prediction on the test dataset, together with the scraped value and its secondary source counter part. The prediction is wrong in rows 3, 7, 8 and 10.

Row	Secondary Source Value	Scraped Value	Error Label	Prediction
1	49900000	49900000	good digits, exact OoM	good digits, exact OoM
2	5911000	5911384	good digits, exact OoM	good digits, exact OoM
3	-1572000	-2021	meh digits, 1000 OoM diff	good digits, exact OoM
4	26000	26271	good digits, exact OoM	good digits, exact OoM
5	50000	50000	good digits, exact OoM	good digits, exact OoM
6	10657000	10656778	good digits, exact OoM	good digits, exact OoM
7	26000	26383	good digits, exact OoM	meh digits, exact OoM
8	126000	126	good digits, 1000 OoM diff	meh digits, 1000 OoM diff
9	1509000	1509000	good digits, exact OoM	good digits, exact OoM
10	55000	55000	good digits, exact OoM	meh digits, wrong OoM

A full count of all matching and mismatching prediction labels can be seen in the confusion matrix in Figure 2. This figure displays the encoded error labels (see Table 6). In the ideal case, only the values of the matrix diagonal would be non-zero as these indicate correctly predicted error labels, but clearly this is not the case here. First off, the label imbalances are again very noticeable, with '0' being the label 'good digits, exact OoM'. Then, the highest amount of miss-classifications, a total of 2900 cases, occur for the true label '3' (meaning 'meh digits, exact OoM') which get mistaken for the label '0' ('good digits, exact OoM'). Indeed, it is about twice as likely for the classifier to wrongly predict the label '0' for the true label '3' than to accurately identify it as such. Additionally, the label '3' gets predicted in roughly the same frequency for the true label '3' (1434 cases) as for the true label '0' (1406 observations). It seems that the model is not particularly good at distinguishing 'good digits' from 'meh' ones. To a degree, this makes sense as the available and selected feature set (see Table 7) does not include a lot of variables that could be powerful in deciding what constitutes 'good' or 'meh' digits in the scraped value. Additionally, the error labels as I defined them do not consider positional context (see Section 3.2) that might give more indication as to whether or not extracted digits are of good quality or not. Another example of this is the second highest miss-classification amount, 1830 cases, where the true label '2' ('meh digits, wrong OoM') get wrongly predicted as the label '0'. This again indicates the bias towards the majority label as well as the rather poor performance on distinguishing the quality of the scraper value digits.

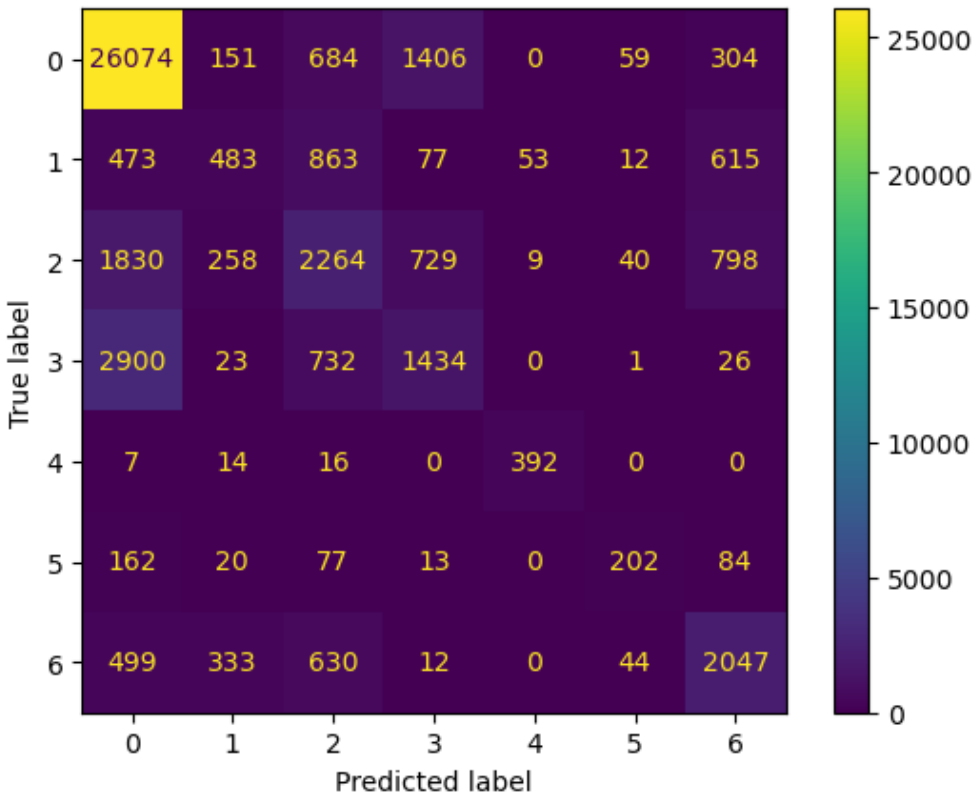


Figure 2: Confusion matrix matching predicted and true error labels for the test dataset

At this point it is worth looking at the consequences of such miss-classifications. Generally speaking, predicting a scraped value to be 'bad' when it is actually of 'good' quality, means that this value is less likely to be selected for the best possible representation in Step 3 and the other way around. Since there are up to six different scraper values for each financial figure, a miss-classification could occur for either one of them, which then results in several different scenarios. For example, a 'truly bad, but predicted good' value could be chosen over a 'truly good, but predicted bad' one in Step 3. What happens in this case is that ultimately a wrong value is passed on as the representation of the financial reality of a company. This in turn could mean that further applications, such as credit risk assessments, would rely on 'bad data' and potentially offer the wrong conclusions on the finances of a company. But, in reality it is rarely the case that these further applications only consider a single value for their calculations which means that a single wrong financial figure value does not carry grave consequences on its own in most cases. Only if a considerable large part of the financial figure values that compromise a report are of bad quality, it would actually impact further applications. Considering that the image recognition algorithm does extract a value that is truly of the type 'good digits, exact OoM' most of the time, the risk for miss-classifications to carry grave consequences is generally low. So while miss-classifications of course mean that the classifier fails at assessing data quality accurately, it only does so for a single value. Ultimately, the value selection in Step 3 and the fact that a financial report contains many different financial figures reduce the impact that a single miss-classification can have.

Now, looking at the performance across all error labels, the ratio between correct predictions and incorrect ones (meaning the prediction accuracy as described in Section 2.5.3) is 70.22%. This is, as expected, not very high and also an overstated result due to the imbalanced label frequencies. In comparison, the balanced accuracy across all error labels is only 51.56%. A list of prediction accuracies by label is given in Table 9. While attempts at correcting for the imbalanced classes have been made (as can be seen in the associated class weights of this table), the effect was rather minute. Only for the error labels 'extreme scraper OoM' and 'good digits, exact OoM' does the prediction accuracy reach above 90%.

Table 9: Error labels and their prediction accuracy on the test set as well as their associated class weights from the training process.

Encoded Label	Error Label	Label Accuracy	Associated Class Weight
4	extreme scraper OoM	0.913753	17.2764
0	good digits, exact OoM	0.909199	0.221727
6	good digits, 1000 OoM diff	0.574194	2.04215
2	meh digits, wrong OoM	0.381916	1.16441
5	good digits, wrong OoM	0.362007	13.6772
3	meh digits, exact OoM	0.280297	1.38378
1	meh digits, 1000 OoM diff	0.1875	3.47444

To offer some perspective on these accuracy numbers, if I chose a naive predictor that in all cases predicts the majority label ('good digits, exact OoM') instead, the prediction accuracy would be equal to the frequency of this label in the test data. In this case, the 'good digits, exact OoM' label occurs 28678 times in 46850 total values, giving a prediction accuracy of 61.21% for the naive predictor. For all remaining labels, the prediction accuracy would be at 0% as none of them would be identified correctly. This then gives a balanced accuracy of only $\frac{61.21\%}{7} = 8.744\%$. Both scores are higher for my RF model than for the naive predictor and especially the balanced accuracy result shows a clear preference for the RF classifier.

I have chosen to train only a single classifier for all of the six different scrapers. Since each of these scrapers is configured in a slightly different way, the underlying true probability distributions that generate the scraper outputs (and therefore error labels) are likely different as well. So by approximating these different distributions by only a single model (the Random Forest classifier) I am introducing a simplification. It is worth checking whether this simplification is actually valid or whether I might be sacrificing too much predictive complexity. Table 10 shows the classification results by scraper[11]. Not every scraper successfully extracts a value for every financial figure which is why the number of observations by scraper differ. This happens, because the image recognition algorithms do not only need to identify the correct financial figure value, but also the associated name. If a scraper fails to get the financial figure name extracted, the according value needs to be disregarded from the initial data, as the value lacks meaning. That is, it is not known, if the value displays revenue, or assets, loss etc and can therefore not be compared to the secondary source.

Now, the achieved prediction accuracy scores across the scrapers are more or less close to the overall accuracy of 70.22% and similarly so for the balanced accuracy. It does seem that for values

[11]Note that the scrapers are numbered oddly which is simply an artefact of earlier developments and of no further relevance

generated by Scraper 7, the model predicts error labels with the highest accuracy which could be explained by the fact that Scraper 7 also shows the highest ratio of error label '0' observations, i.e. Scraper 7 extracts a value with 'good digits, exact OoM' more often than the other scrapers. Since the classifier is slightly biased towards this majority class, it does make sense that the prediction accuracy for Scraper 7 is also the highest. In fact, the correlation coefficient between True '0' Ratio and Accuracy Score is 0.9847 indicating that for the most part the difference in accuracy scores across scrapers is explained by their differing success rate in extracting a value that is labelled 'good digits, exact OoM'[12]. Based on this it is likely that the configuration differences of the individual scrapers is only of minute relevance and the approximation of the differing true distributions by a single RF model is a valid choice.

In addition, the confusion matrices evaluated on the test dataset and split by the six different scrapers are shown in Figure 10 in the appendix. The pattern of correct and incorrect classifications for each scraper is overwhelmingly similar to the overall confusion matrix in Figure 2, which further shows that the simplification of a single RF model for all scrapers is justified.

Table 10: Classification results across the different scrapers. The column True '0' Cases refers to the number of observations that have the 'good digits, exact OoM' error label in the test dataset and the column True '0' Ratio divides these observations by all observations in the test set for each scraper.

Scraper	Observations	Correct Classifications	True '0' Cases	Accuracy Score	Balanced Accuracy	True '0' Ratio
1	7041	4824	4142	0.685130	0.505717	0.588269
3	8811	6287	5456	0.713540	0.499579	0.619226
4	7189	5080	4521	0.706635	0.514625	0.628877
6	8145	5469	4538	0.671455	0.523031	0.557152
7	7615	5655	5194	0.742613	0.507047	0.682075
8	8049	5581	4827	0.693378	0.519594	0.59970
All	46850	32896	28678	0.702156	0.515552	0.612124

Besides the prediction accuracy, I have also considered the area under the ROC cuve (detailed in Section 2.5.3). While this is technically an evaluation metric for the binary case, it can be expanded to the multi-label case by considering pairwise scores. I have chosen to calculate the one-versus-one score using the `roc_auc_score` function of the `sklearn.metrics` library. The resulting AUC is 86.48% (using an unweighted average across one-versus-one scores) which is a decent result.

Based on these evaluation metrics, the trained Random Forest classifier for predicting the error label is certainly not the very best, but I deem the results acceptable. Since the error label prediction is only Step 2 of my overall approach, its performance does not have to be stellar. The true success of this solution is determined based on the performance of the data point selection across the different scrapers in Step 3. The following section dives deeper into that.

[12]It would also be interesting to compare the individual scrapers in their success of extracting accurate values, but that is outside the scope of my thesis.

3.4 Scraper Value Selection

This section now arrives at the final step and concludes my approach to solving the thesis problem. By applying the Random Forest classifier from the previous step to the validation data, I obtain an error label prediction based on which I calculate a *quality score*. This score aims at representing how good the scraped value matches its true counterpart and is used to select which scraped value, across the six different scrapers, is the best one to represent the true value for each financial figure in each report.

As I mention in the previous section, the ultimate success of my approach is determined by this last step, the scraper value selection. Since this is my only overall model, I do not have an alternative model myself to contrast this approach to. However, in the solution that the company I work for has currently implemented, an alternate model for the scraper value selection exists. This model does not evaluate data quality directly, but relies on a so-called 'image recognition confidence score' which is calculated by an external library. Unfortunately, the exact workings of this confidence score are unknown to me and I do not have the required access to be able to describe it further. What is known however, is that this confidence score is tied to the external library and since it is the currently sole input to the scraper value selection, only scraped values using this external library can be compared using its score. This poses a problem in terms of scalability, as it limits the expansion towards other image recognition algorithms for further improving the quality of scraped financial reports. Additionally, with the unknown definition of the confidence score, it offers little customisation and increases the reliance on unknown factors in the overall effort of digitising financial statements. And lastly, from manual inspections it is known that the scraped value selection based on this confidence score does not always perform in the best way possible. This then gives rise to the motivation for my thesis problem: finding a method that is transparent, controllable and potentially performs better.

This 'current' model for the scraper value selection based on the confidence score supplied by the external library offers an alternative 'best possible value' across the scraper outputs. This value is referred to as the *merged value* and while this name is ambiguous in the context of my thesis, I lack a better alternative and so continue to use it in this text. Ultimately, I use the merged value as an alternative result to compare the selected value from my approach to. If my approach outperforms the merged value one by offering a higher rate of correct representations of reality (based on the secondary source approximation), I deem my approach a success and vice versa.

3.4.1 Quantifying Quality

In order to facilitate selecting the best possible value, i.e. the one most likely to represent the reality of the financial statement accurately, across the six scraped values, I need to quantify the predicted data quality aspect of each value. I do that by introducing a *quality score* for each value which is calculated as follows

$$q_i = \frac{1}{2}\left(p_i + l_i\right) \text{ with } i = 1, \ldots, 6 \tag{30}$$

where p_i is the prediction probability obtained from the Random Forest classifier as described in Section 2.4.3 and l_i is the error label preference score (see Table 11) associated with the predicted error label for scraped value i (I of course calculate the quality score for all scraped values in the validation data, so the index i here indicates the six scraped values for each financial figure). The prediction probabilities as well as error label preference scores are independent of each other and since both inputs to the quality score take values in the interval $[0, 1]$, so does the quality score itself.

For each financial figure in each report, I then select the scraped value with the highest quality score as the best possible value $s^* = s_{i^*}$ to represent that financial figure, where i^* is the scraped value index maximising the quality score, i.e.

$$i^* = \underset{i=1,\dots,6}{\operatorname{argmax}} (q_i) \tag{31}$$

Note that if two scraped values for the same financial figure were to get the same exact quality score, my implementation would pick the first one in the list. If the two scraper values were identical, that would not pose a problem, since the order of identical values does not matter in the selection, but if the scraper values would differ and still have the same exact quality score, the selection is biased towards the order of values in the list variable of the implementation. Ultimately, if my solution assigns the same quality score for two distinct values, it is a sign that the quality score in its current form is not sufficiently distinguishing quality. Luckily, this is not the case for any data point in the validation data.

Table 11: Error labels together with their assigned preference score.

Error Label	**Preference Score**
good digits, exact OoM	1
good digits, 1000 OoM diff	0.8
meh digits, exact OoM	0.6
meh digits, 1000 OoM diff	0.4
good digits, wrong OoM	0.2
extreme scraper OoM	0.0
meh digits, wrong OoM	0.0

Regarding the error label preference score, this score simply gives a ranking which error label is 'more correct', or the other way around, which error label has the least mistakes in matching the true value and is therefore preferred over others. See Table 11 for the score values by error label. The label 'good digits, exact OoM' indicates that scraped value and true one match (almost) exactly [13] and therefore represents the best possible case with the highest preference score. On the other end, the label 'meh digits, wrong OoM' indicates that neither the string of significant digits nor the order of magnitude match between the scraped and true value. A scraped value with this predicted error label, or the label 'extreme scraper OoM', should therefore not be selected as a representation for the true value, and the preference score is set to 0.

Note that the error label 'good digits, wrong OoM' has a lower preference score than 'meh digits, exact OoM' and 'meh digits, 1000 OoM diff' which might be a little counter intuitive. The reasoning behind this is that when representing financial figures, a difference in the OoM constitutes a larger misrepresentation than a difference in the string of significant digits. So an exact OoM match or knowing by how much the OoMs differ is preferred to simply having a wrong OoM in the scraped value, regardless of whether or not the digits might match.

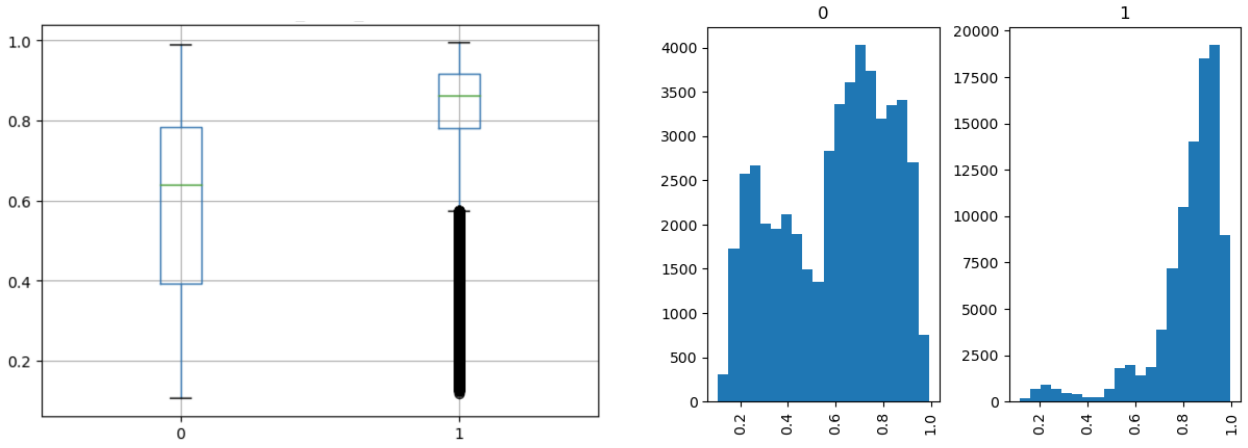
Equipped with the quality score and using the definition in Equation (31) to identify the best possible scraper value, I make my selection of scraper values for all financial figures. As a last step

[13] Since the error labels are based on the secondary source which is only an approximation to the true values, the error labels themselves cannot actually express a truly exact match between scraped and true value, therefore the 'almost' designation.

to this, if the predicted error label for the selected scraper value indicates an OoM difference of a thousand, I offer a correction by multiplying the scraped value by 1000 and setting this value as the ultimate output of my approach: the *suggested value*. If the best possible value has an exact OoM, the suggested value is simply the selected scraper value without manipulation. With this, I finally obtain a list of suggested values which my approach deems to be the extracted value to most closely represent the true value of each financial figure in each report.

3.4.2 Quality Scoring Results

The quality score offers the crucial quantification of the quality aspects I try to determine for the outputs of the image recognition efforts. As with any score, it is assumed that its value is proportional to the actual quality, i.e. the correct representation of reality, but it is always worth considering whether this assumption actually holds. In order to be a powerful score, a higher value should indicate that the associated scraped value is truly a good representation of what is printed in the financial statement and vice versa. Figure 3 shows both a box and whiskers plot as well as histograms of the distribution of the quality score of all the scraper values in the validation data (with 142903 observations), split by a flag indicating whether the scraper value matches its secondary source counterpart (label '1') or not (label '0'). In the ideal case, there would be a clear cut-off point in the quality score, where all scraper values labelled '0' lie below it and all values labelled '1' above. Unfortunately, this is not exactly the case. Looking at the histograms, there is considerable overlap in these two groups for quality scores above 0.5. This means that, while a quality score below 0.5 quite confidently indicates scraped values to be of 'bad quality', i.e. not matching their true counterpart, the other way around, when a quality score is higher than 0.5, does not confidently distinguish 'good' or 'bad' quality. There is definitely room for improvement to make the quality score more powerful in this regard, which is discussed further in Section 4.2.



(a) Box and whiskers plot

(b) Histograms

Figure 3: Distribution of the quality score for the scraper values in the validation data split by a flag indicating whether the value matches the corresponding secondary source one (label '1', 93817 observations) or not (label '0', 49086 observations).

That being said, the sample median of the quality score, indicated as a green line in Figure 3a,

has the value 0.64 for the mismatching values and the value 0.86 for the matching ones. While ideally, the quality score for mismatching values, i.e. incorrect representations of reality, should be zero, a difference between both match cases can be seen. Again, as with the Random Forest classifier in the previous step, these results are not excellent, but I deem them sufficient to continue.

It is worth noting that by looking at the histograms in Figure 11 in the Appendix, there seems to be no great difference in quality score distribution by scraper. The scrapers all show the roughly 0.5 threshold for differentiating the mismatching and matching scraper values and there is not one scraper that is much better or worse than another, in terms of how powerful the quality score distinguishes 'good' or 'bad' quality data points. The only difference between the histograms are the number of scraped values available from each scraper, which is due to the fact that not every scraper necessarily outputs a value for each financial figure in each report.

The next section reveals whether the suggested values outperform their corresponding merged values from the existing approach.

3.4.3 Model Comparison Results

To conclude Step 3 and the approach itself, I compare my suggested values for the validation data to the merged values from the alternative model. A selected sample of values can be seen in Table 13 in the appendix. As said before, I use the success rate R of identifying correct representations of reality as the factor to compare the suggested and merged values by (see the metric definition in Equation (2) in Section 2.2).

There are 27651 rows in the pivoted validation data[14], but for only 27419 a secondary source value is available, meaning I exclude those 232 rows without the secondary source value from the evaluation. I am happy to report that for 20687/27419 (75.45%) my approach selects the correct value while the alternative model only gets 17821/27419 (65.00%) of cases correct. This is already a promising result, but there might of course be cases in which my suggested value is actually worse than the merged one. Table 12 details these different 'match scenarios', using the secondary source value to determine whether scraped or merged values are correct. For the largest part (16389 cases, 59.77%) of the validation data, both the suggested and merged values are correct which is already a decent baseline for both approaches to be useful. As can be seen as well, in 1432 cases (5.223%) my suggested value is actually worse than the merged one - which of course is unfortunate. However, there are 4235 (15.45%) cases in which the suggested value is better than the merged one. Meaning that the loss of correct results in the alternative model is outweighed by the improvements offered by my approach.

[14]In the pivoted validation data each row corresponds to a single financial figure in a given report, with each scraper value displayed in its own column and a new column for the selected scraper value (the suggested value) added. In previous steps, there would be multiple rows for each financial figure, one row for each scraper value.

Table 12: Number of occurrences for each 'match case' between merged, suggested and secondary source value. Note that in 232 observations of the validation data, a secondary source value was unavailable and this table therefore only considers 27419 rows.

	Suggested correct	**Suggested incorrect**	**Suggested Missing**	**Total**
Merged correct	16389	1432	0	17821
Merged incorrect	4235	5261	0	9496
Merged missing	63	39	0	102
Total	20687	6732	0	27419

For the 5261 cases in which both the suggested and the merged model offer incorrect results, the histogram of the quality score distribution is shown in Figure 4. Based on visual inspection, this distribution looks very similar to the overall one in Figure 3b for the mismatching values (label '0'), except for fewer observations in the quality score below 0.5 range which is explained by the fact that now, in the suggested model, only those scraper values with maximal quality score per financial figure are included.

This suggests that there is no clear further aspect of the quality score in its current form that could be exploited to improve these incorrect data point selections. Instead, it is likely that neither of the six scrapers actually obtained a correct value that could have been selected in the first place and the quality score unfortunately does not reflect this in all cases (see the observations in the range above 0.5).

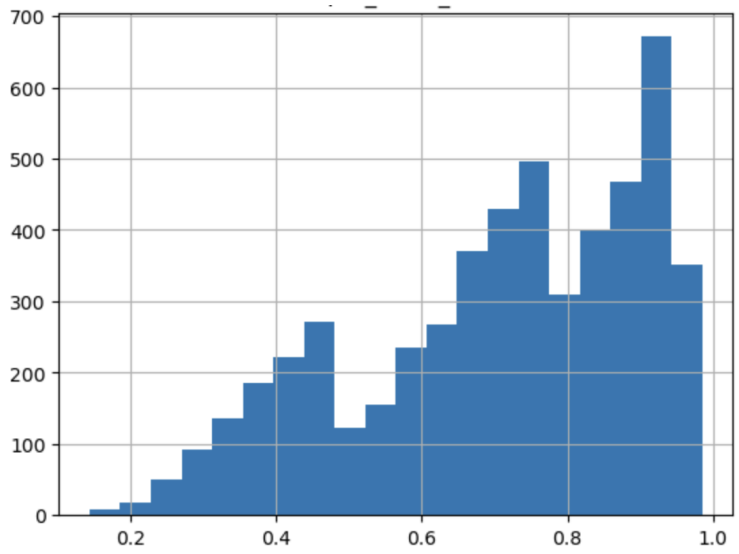


Figure 4: Histogram of the quality score distribution of the validation data where both the merged model and my suggested model lead to incorrect results.

To formally compare the success rate of both models, i.e the number of correct values out of the total, I apply a χ^2-hypothesis test with

$$H_0 : R_s = R_m \tag{32}$$
$$H_1 : R_s \neq R_m \tag{33}$$

where R_s and R_m are the success rates of my suggested model and the merged one respectively. This means that the null hypothesis H_0 reflects the case in which there is no difference between my suggested and the merged values when it comes to selecting a scraped value that accurately represents the true value or not. In turn, the alternative hypothesis H_1 covers the case of suggested and merged model differing in their success rates.

The test statistic is given by

$$\chi^2 = \sum_{i=1}^{4} \frac{(x_i - \mu_i)^2}{\mu_i} \tag{34}$$

where x_i is the observed frequency of correct or incorrect values and μ_i the associated expected value. The observed frequencies are obtained from Table 12 and the expected values are calculated as the product of the joint probabilities and the total values $2N$, giving the following values

$$x_1 = 20687,\ \mu_1 = \frac{(x_1 + x_2)(x_1 + x_3)}{2N} = 19254$$
$$x_2 = 6732,\ \mu_2 = \frac{(x_1 + x_2)(x_2 + x_4)}{2N} = 8165$$
$$x_3 = 17821,\ \mu_3 = \frac{(x_3 + x_4)(x_1 + x_3)}{2N} = 19254$$
$$x_4 = 9598,\ \mu_4 = \frac{(x_3 + x_4)(x_2 + x_4)}{2N} = 8165$$

Note that the 'merged missing' match scenario has been included in the count for incorrect values of x_4, giving an equal amount of observations in both the suggested and merged cases and hence the total number of observations as $2N$. This leads to a test statistic value of

$$\chi^2 = 2\frac{(1433)^2}{19254} + 2\frac{(1433)^2}{8165} = 716.30 \tag{35}$$

With 1 degree of freedom and a significance level $\alpha = 0.05$, the critical value for this test is at 3.841. Since the test statistic far surpasses the critical value, H_0 should be rejected and it can be concluded that the difference between R_s and R_m is statistically significant.

Given these results, I am quite confident that my suggested approach in selecting the best possible scraper value offers an improvement to the existing solution relying on the merged value. Of course, a success rate of 75.45% is still far from perfect and my solution involves a Machine Learning model that entails a much larger maintenance effort, but the results are promising and I deem my approach in tackling the problem of assessing the data quality of the extracted financial data successful.

4 Discussion

4.1 Limitations of the Solution

This section has been referenced several times in this thesis, indicating the importance of considering the limitations my solution has.

I mention early on that assessing the data quality domain of accuracy requires having some kind of 'objective truth' of what reality actually is. Of course, there are the actual PDFs of the financial statements that allow for a manual inspection of data quality accuracy, but that is not a feasible method to assess data quality of the entire data. I was lucky enough to have a secondary source available for the same financial data that the image recognition algorithm has extracted. While overall this secondary source is deemed of sufficient quality to represent the financial statements themselves, it still comes with some limitations that affect the success of the overall solution.

Firstly, the secondary source only provides rounded values (to the nearest thousand), while the scrapers will show the exact value as it is printed in the statements (if there is not quality issue). This means that 'good digits' in the error labels are determined based on whether or not the rounded values match and not the exact digits. Since most financial statements report numbers in MSEK or KSEK, an amount below 1000 SEK does not have a great effect however, so this rounding limitation is acceptable.

Secondly, the secondary source is not available for all data points, meaning I had to exclude some scraped values from the dataset as the secondary source counterpart is missing. In most cases, the values are unavailable due to not having purchased them, but there might be cases in which the secondary source might have had difficulty obtaining the value (e.g. due to bad scan quality of the financial statements from Bolagsverket, so that not even human eyes could read the correct value). In such a case, it is likely that the scrapers also had difficulty obtaining the correct financial figure, but this then unknown to my solution. From experience, these should only occur in rare cases, however, so again, an acceptable limitation.

Lastly, the secondary source needs to be purchased and the goal of the scraping effort is to replace this cost, meaning the secondary source will not be available in the future which makes retraining the RF model impossible. This should be bearable though, since first, it is expected that the share of digitally filed reports with Bolagsverket is going to increase (therefore decreasing the reliance on image extraction algorithms to obtain digitised financial data on Swedish companies) and second, it is unlikely that the layout of financial reports is going to change much over time, meaning the image recognition algorithm probably will not need to adapt to reading different kinds of reports. Lastly, should it really become necessary, it is possible to still purchase small subsets of secondary source data to retrain the RF classifier.

Beyond the limitations tied to the use of the secondary source, the approach itself has some constraints as well. The first of which is related to the error labels themselves. As mentioned before, these error labels are tied to the types of scraper mistakes that I identified, but they are not able to capture all types. Especially 'positioning mistakes', e.g. reading the wrong line or column of the report or not considering the context of the financial figures printed 'close by', are not included in the error labels. This means that accuracy issues related to these types of mistakes get mixed in to the digit and order of magnitude labels which likely affects how well the model can identify quality issues.

Additionally, the training data rows are not necessarily independent of each other. In each financial report there are several financial figures. Meaning several rows in the training data are

scraped from the same report that might suffer from the same extraction difficulties. For example, if the scan quality of the PDF is bad or the layout of the report is particularly unusual, several rows in the training data will be affected by it, resulting in non-independent rows. Statistical Learning Theory assumes that observations in the training data are iid, which is not necessarily the case here. Since, in reality, it is difficult to obtain truly iid training data however, this is deemed acceptable as well.

This concludes the list of limitations to the solution and the next section focuses on potential future improvements.

4.2 Future Outlook

As is mentioned in the results, the RF classifier results are not stellar. While my solution overall beats the existing approach to selection scraper values, there is clearly room for improvement which is discussed in this section.

Firstly, the error labels are a crucial building block of my solution. They form the abstraction by which I assess data quality, so their success in actually capturing quality issues puts a bound on how successfully the overall solution can be. I already mention previously that they do not include all identified types of scraper mistakes, so the natural possible improvement is to expand these error labels. Namely, when error labels are being created, including 'positioning mistakes' likely leads to an improvement in overall data quality detection. This could be done by introducing the context of all financial figures within a single report to the evaluation of a scraped value. For example, when a scraper value does not match its secondary source counterpart, one could compare the list of all other values in the same report and check, if the scraper value matches another secondary source value in the same report. If that is the case, it is likely that a positioning mistake has occurred where the wrong line has been extracted. Another flavour of this is to utilise the additive nature of financial figures, e.g. the financial figure 'assets' is the sum of 'noncurrent assets' as well as 'current assets' which in turn are sums of yet more granular financial figures. Checking if a scraped value that does not match its counterpart directly, instead matches an aggregate of secondary source values from the same report could indicate another mis-read line mistake. However, this additive nature is more difficult to exploit, because it does not necessarily offer a location of the mistake[15], so it is a less preferred improvement option.

Furthermore, the error labels for extreme order of magnitudes are based on hard-coded thresholds (i.e. an OoM difference greater than four and a scraped value OoM greater than nine). Instead, more sophisticated methods to identify what constitutes an extreme value can be introduced. For example, thresholds could be set per financial figure instead of for all values together, since there are likely differences in what counts as extreme based on the meaning of each financial figure. Also, instead of providing fixed threshold values, they could be determined adaptively, e.g. by using Chebyshev's inequality to identify extreme values.

Another possible improvement can be done in regards to the unbalanced error labels. Since the image recognition algorithms are overall pretty good at what they do, meaning the scraped values are largely accurate, there is a clear dominance of the 'good digits, exact OoM' error labels. While I tried to counteract this by introducing class weights in the model, there are potentially further adjustments to be made.

[15] E.g. if the asset value is 4600 SEK, noncurrent assets is 2800 SEK and current assets 2000 SEK, the sum of both 'sub-figures' is 4800 SEK, indicating that a mistake has occurred. But, with this information alone, it cannot be deduced whether the error lies with noncurrent assets, current assets or both.

Beyond the error labels, further improvements can also be made to the quality score itself in Step 3 of the approach. Instead of averaging the prediction probability and the error label preference score, another combination of these inputs might better capture quality. For example, multiplying both values instead or introducing weights could improve the 'power' of the quality score.

More generally applicable improvements are to simply increase the sample size of the training data, as long as secondary source values are available, and to introduce better handling of NULL values, e.g. through imputation rather than removal of rows from the dataset. Additionally, alternative ML algorithms might be applied to predicting error labels, especially ones that are able to model more complex transformations of the input data and capture semantic and positional context of the financial figures. While this improvement is quite limited with the currently available data, potentially expanding the feature set with additional information from the image recognition algorithms likely increases the predictive power and ultimately offers a more confident and detailed data quality assessment. Given the current feature set however, I would expect that especially the refinement and expansion of the error labels would bring the greatest effect in improving the success of the solution in identifying misrepresentations of reality.

4.3 Conclusion

Overall, I deem my thesis problem as stated in Section 1.3 to be sufficiently solved. Although the ML performance of my trained RF classifier is far from perfect, it is able to identify error labels and therefore data quality accuracy within reasonable bounds. The following data point selection does beat the existing method by a considerable margin, meaning my solution detects data quality issues better than what has been currently implemented.

While there is room for improvement as described in the previous section, I can recommend my solution to be added to the existing financial data service that the company I work for develops. Doing so likely improves how well the extracted information reflects the reality of the scanned financial reports from Bolagsverket.

This thesis solves the data quality question in a very targeted and narrow domain. It does seem that truly general frameworks for addressing data quality are yet to be developed, or might need to continue as more practical approaches.

As a last remark, in this 'Age of Data' that we are living in, numbers, statistics and data are inescapable, omnipresent companions. We see the world through abstractions and small samples, impossible as it would be to comprehend every detail all at once. And indeed, my work in this thesis is just a tiny drop in a vast ocean of research that shows that there is incredible value in the exploration and utilisation of data. Through choosing aspects of a real phenomenon, measuring them and *creating* data, we introduce models of reality that are powerful tools for learning about it and making new discoveries - in a way that would not be possible without this layer of data as a representation. It is my hope that this thesis highlights the importance of thinking of data as being that, a representation, and that 'data-driven' methodologies consider the discrepancy that might exist between the data and its underlying real phenomenon.

Appendix A Examples of Scraper Mistakes

Kortfristiga fordringar	
Kundfordringar	360 902
Övriga fordringar	0
Upparbetad men ej fakturerad intäkt	567 109
Förutbetalda kostnader och upplupna intäkter	10 163
Summa kortfristiga fordringar	**938 174**

(a) Missing first digits; while the true value reads 360902, the scraper extracted the value 902.

BALANSRÄKNING forts.

	Not	2022-12-31	2021-12-31
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital		50 000	50 000
Summa bundet eget kapital		**50 000**	**50 000**
Fritt eget kapital			
Balanserat resultat		213 611	172 669
Årets resultat		27 668	140 942
Summa fritt eget kapital		**241 279**	**313 611**
Summa eget kapital		**291 279**	**363 611**

(b) Missing first digits; while the true value reads 291279, the scraper extracted the value 279.

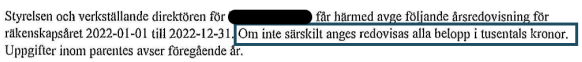
EGET KAPITAL OCH SKULDER	
Eget kapital	
Bundet eget kapital	
Aktiekapital (500 aktie)	50 000
Summa bundet eget kapital	**50 000**
Fritt eget kapital	
Balanserat resultat	260 666
Årets resultat	230 979
Summa fritt eget kapital	**491 645**
Summa eget kapital	**541 645**

(c) Missing last digits; while the true value reads 541645, the scraper extracted the value 541.

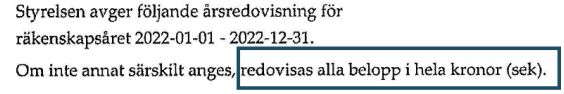
RESULTATRÄKNING	**Not**	**2022-01-01 2022-12-31**
Rörelsens intäkter m.m.		
Nettoomsättning		3 090 687
Övriga rörelseintäkter		4 953 205
		8 043 892
Rörelsens kostnader		
Fastighetskostnader		-472 234
Övriga externa kostnader		-1 052 381
Personalkostnader	2	-78 852
Avskrivningar av materiella anläggningstillgångar		-1 096 757
Övriga rörelsekostnader		-5 343
		-2 705 567
Rörelseresultat		5 338 325

(d) Missing last digits; while the true value reads 5338325, the scraper extracted the value 5338.

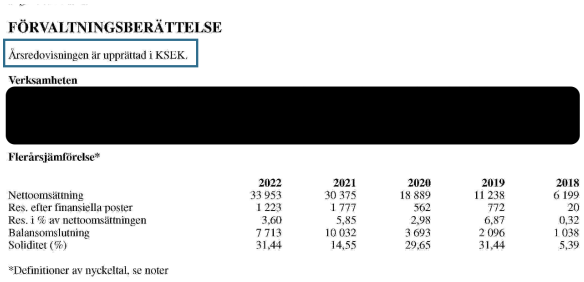
Figure 5: Scraper mistakes, missing first and last digits of a number.


Styrelsen och verkställande direktören för [redacted] får härmed avge följande årsredovisning för räkenskapsåret 2022-01-01 till 2022-12-31. Om inte särskilt anges redovisas alla belopp i tusentals kronor. Uppgifter inom parentes avser föregående år.

(a) OoM stated elsewhere in report; while the true value reads 31444000, the scraper extracted the value 31444.


Styrelsen avger följande årsredovisning för räkenskapsåret 2022-01-01 - 2022-12-31.
Om inte annat särskilt anges, redovisas alla belopp i hela kronor (sek).

(b) OoM stated elsewhere in report; while the true value reads 428785, the scraper extracted the value 428785000.


FÖRVALTNINGSBERÄTTELSE

Årsredovisningen är upprättad i KSEK.

Verksamheten

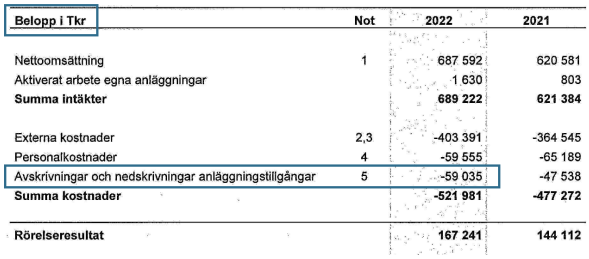
[redacted]

Flerårsjämförelse*

	2022	2021	2020	2019	2018
Nettoomsättning	33 953	30 375	18 889	11 238	6 199
Res. efter finansiella poster	1 223	1 777	562	772	20
Res. i % av nettoomsättningen	3,60	5,85	2,98	6,87	0,32
Balansomslutning	7 713	10 032	3 693	2 096	1 038
Soliditet (%)	31,44	14,55	29,65	31,44	5,39

*Definitioner av nyckeltal, se noter

(c) OoM stated elsewhere in report; while the true value reads 913000, the scraper extracted the value 913.



Belopp i Tkr	Not	2022	2021
Nettoomsättning	1	687 592	620 581
Aktiverat arbete egna anläggningar		1 630	803
Summa intäkter		**689 222**	**621 384**
Externa kostnader	2,3	-403 391	-364 545
Personalkostnader	4	-59 555	-65 189
Avskrivningar och nedskrivningar anläggningstillgångar	5	-59 035	-47 538
Summa kostnader		**-521 981**	**-477 272**
Rörelseresultat		**167 241**	**144 112**

(d) OoM in expected position, but not extracted; while the true value reads -59035000, the scraper extracted the value 59035.

Figure 6: Scraper mistakes, OoM not extracted properly.

Rörelsekostnader	
Övriga externa kostnader	-5 750
Summa rörelsekostnader	**-5 750**
Rörelseresultat	**-5 750**

(a) Wrong digit extracted; while the true value reads -5750, the scraper extracted the value 7505.

Balansräkning	**Not**	**2021-12-31**	**2020-12-31**
TILLGÅNGAR			
Anläggningstillgångar			
Materiella anläggningstillgångar			
Byggnader och mark	5	52 299	0
Maskiner och andra tekniska anläggningar	6	155 621	277 209
Inventarier, verktyg och installationer	7	4 347 617	6 370 189
		4 555 537	**6 647 398**
Summa anläggningstillgångar		**4 555 537**	**6 647 398**
Omsättningstillgångar			
Varulager m m			
Färdiga varor och handelsvaror		4 859 737	1 255 063
		4 859 737	**1 255 063**
Kortfristiga fordringar			
Kundfordringar		8 983 369	1 903 279
Fordringar hos koncernföretag		0	3 569 806
Övriga fordringar		223 571	667 373
Förutbetalda kostnader och upplupna intäkter	8	703 759	1 215
		9 910 699	**6 141 673**
Kassa och bank		1 790 115	3 216 891
Summa omsättningstillgångar		**16 560 551**	**10 613 627**
SUMMA TILLGÅNGAR		**21 116 088**	**17 261 025**

(b) Wrong digit extracted; while the true value reads 9910699, the scraper extracted the value 9010639.

Balansräkning	**Not**	**2021-12-31**	**2020-12-31**
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital		25 000	25 000
Summa bundet eget kapital		**25 000**	**25 000**
Fritt eget kapital			
Balanserat resultat		193 206	0
Årets resultat		7 918	193 206
Summa fritt eget kapital		**201 124**	**193 206**
Summa eget kapital		**226 124**	**218 206**
Obeskattade reserver	3		
Periodiseringsfonder		81 000	81 000
Ackumulerade överavskrivningar		4 074	3 442
Summa obeskattade reserver		**85 074**	**84 442**
Långfristiga skulder			
Övriga skulder	4	0	42 402
Summa långfristiga skulder		**0**	**42 402**
Kortfristiga skulder			
Skatteskulder		2 158	52 798
Övriga skulder		0	21 933
Upplupna kostnader och förutbetalda intäkter		15 000	43 126
Summa kortfristiga skulder		**17 158**	**117 857**
SUMMA EGET KAPITAL OCH SKULDER		**328 356**	**462 907**

(c) Wrong digit extracted; while the true value reads 328356, the scraper extracted the value 278356.

Figure 7: Scraper mistakes, wrong digit extracted.

Resultaträkning	Not	2022-01-01 -2022-12-31	2021-01-01 -2021-12-31
Nettoomsättning		84 349 490	87 845 349
Övriga rörelseintäkter		844 712	526 435
		85 194 202	**88 371 784**
Rörelsens kostnader			
Råvaror och underentreprenader		-54 458 187	-55 353 708
Övriga externa kostnader	1	-10 506 882	-13 397 654
Personalkostnader	2	-13 412 688	-14 108 169
Avskrivningar och nedskrivningar av materiella och immateriella anläggningstillgångar		-4 307 116	-4 670 506
		-82 684 873	**-87 530 037**
Rörelseresultat		**2 509 329**	**841 747**

(a) Wrong column extracted; while the true value reads -4307116, the scraper extracted the value 4670.

Balansräkning	Not	2022-12-31	2021-12-31
	1		
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital		100 000	100 000
Reservfond		20 000	20 000
Summa bundet eget kapital		**120 000**	**120 000**
Fritt eget kapital			
Balanserat resultat		1 922 504	1 959 807
Årets resultat		231 044	-37 304
Summa fritt eget kapital		**2 153 548**	**1 922 503**
Summa eget kapital		**2 273 548**	**2 042 503**
Obeskattade reserver			
Periodiseringsfonder		1 136 500	1 760 000
Summa obeskattade reserver		**1 136 500**	**1 760 000**
Kortfristiga skulder			
Förskott från kunder		100 000	0
Leverantörsskulder		1 424 444	607 109
Skulder till koncernföretag		4 312 900	1 810 000
Övriga skulder		202 009	317 412
Upplupna kostnader och förutbetalda intäkter		1 466 782	1 149 984
Summa kortfristiga skulder		**7 506 135**	**3 884 505**

(b) Wrong column extracted; while the true value reads 7506135, the scraper extracted the value 3884.

Kortfristiga skulder		
Skatteskulder	9 456	-
Övriga kortfristiga skulder	30 740	1 132 773
Upplupna kostnader och förutbetalda intäkter	-	30 740
	40 196	1 163 513
SUMMA EGET KAPITAL OCH SKULDER	95 593	1 228 366

(c) Wrong line extracted; while the true value reads 40196, the scraper extracted the value 9456.

Balansräkning	Not	2022-12-31	2021-12-31
Tkr	1		
TILLGÅNGAR			
Anläggningstillgångar			
Materiella anläggningstillgångar			
Förvaltningsfastigheter	7	19 494	4
Pågående nyanläggningar och förskott avseende materiella anläggningstillgångar	8	61 118	0
		80 612	**4**
Summa anläggningstillgångar		**80 612**	**4**

(d) Wrong line extracted; while the true value reads 80612000, the scraper extracted the value 61118000.

Figure 8: Scraper mistakes, wrong column or line extracted.

Balansräkning

Tkr

	Not	2022-12-31	2021-12-31
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital		100	100
		100	**100**
Fritt eget kapital			
Balanserad vinst eller förlust		54 323	58 575
Årets resultat		5 876	5 747
		60 199	**64 322**
Summa eget kapital		**60 299**	**64 422**
Långfristiga skulder	15		
Skulder till koncernföretag		9 000	11 000
Summa långfristiga skulder		**9 000**	**11 000**
Kortfristiga skulder			
Leverantörsskulder		152	50
Skulder till koncernföretag		259	173
Skulder till Sandvikens kommun		8 441	6 988
Övriga kortfristiga skulder		125	16
Upplupna kostnader och förutbetalda intäkter	16	124	33
Summa kortfristiga skulder		**9 101**	**7 260**
SUMMA EGET KAPITAL OCH SKULDER		**78 400**	**82 682**

(a) Multiple mistakes (wrong digit and OoM not extracted); while the true value reads 9101000, the scraper extracted the value 2101.

Balansräkning

Belopp i kr	Not	*2022-12-31*	*2021-12-31*
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital (1000 aktier)	2	100 000	100 000
Reservfond		20 000	20 000
Summa bundet eget kapital		120 000	120 000
Fritt eget kapital			
Balanserad vinst		941 050	844 543
Årets resultat		163 126	246 507
Summa fritt eget kapital		1 104 176	1 091 050
Summa eget kapital		1 224 176	1 211 050
Obeskattade reserver			
Periodiseringsfond		0	0
Summa obeskattade reserver		0	0
Kortfristiga skulder			
Leverantörsskulder		5 695	16 524
Skatteskulder		27 738	25 983
Övriga skulder		284 961	242 250
Upplupna kostnader och förutbetalda intäkter		75 000	38 825
Summa kortfristiga skulder		393 394	323 582
SUMMA EGET KAPITAL OCH SKULDER		**1 617 570**	**1 534 632**

(b) Unclear; while the true value reads 1617570, the scraper extracted the value 1 (for Eget Kapital och Skulder) and the true value 393394 has been scraped as 18 (for Kortfristiga Skulder).

Balansräkning

Belopp i kkr	*Not*	*2022-12-31*	*2021-12-31*
EGET KAPITAL OCH SKULDER			
Eget kapital	*12*		
Bundet eget kapital			
Aktiekapital		1 000	1 000
		1 000	1 000
Fritt eget kapital			
Balanserad vinst		32 640	30 591
Årets resultat		8 252	2 049
		40 892	32 640
Summa eget kapital		41 892	33 640
Obeskattade reserver			
Ackumulerade avskrivningar utöver plan	*13*	6 621	5 302
Periodiseringsfonder	*14*	690	-
		7 311	5 302
Avsättningar			
Uppskjuten skatt	*15*	9 130	8 333
		9 130	8 333
Långfristiga skulder			
Övriga långfristiga skulder		-	300
Skulder till koncernföretag		229 492	200 697
		229 492	200 997
Kortfristiga skulder			
Skuld koncernföretag kort del		7 306	4 424
Leverantörsskulder		1 946	1 890
Skulder till koncernföretag		-	22 055
Övriga skulder		1 564	459
Upplupna kostnader och förutbetalda intäkter		6 602	4 813
		17 418	33 641
SUMMA EGET KAPITAL OCH SKULDER		305 243	281 913

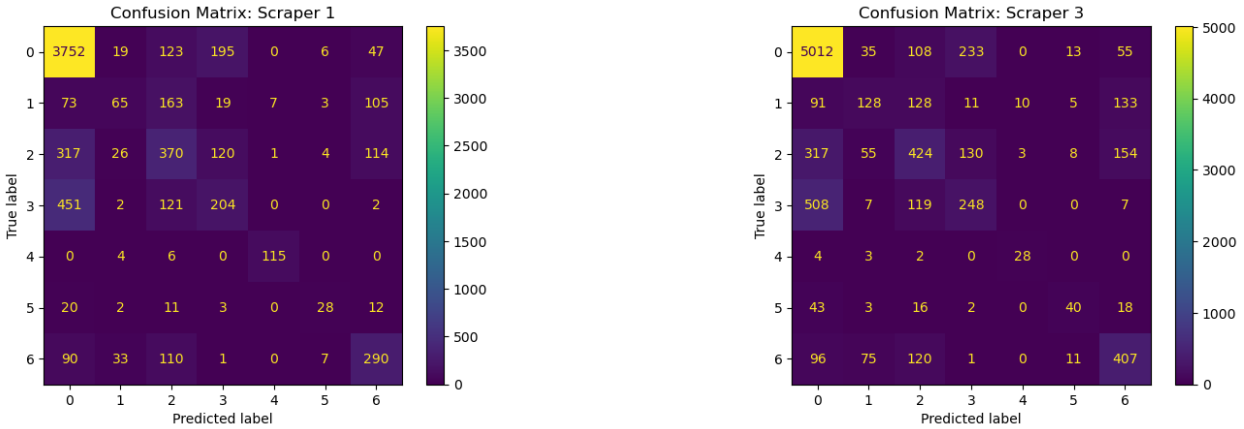
(c) Unclear; while the true value reads 17418000, the scraper extracted the value 10112.

Bokslutsdispositioner		
Förändring av periodiseringsfonder	-22 703	0
Summa bokslutsdispositioner	**-22 703**	**0**
Resultat före skatt	**251 434**	**589 855**

(d) Two columns read together as single value; while the true value reads 251434, the scraper extracted the value 251434589855.

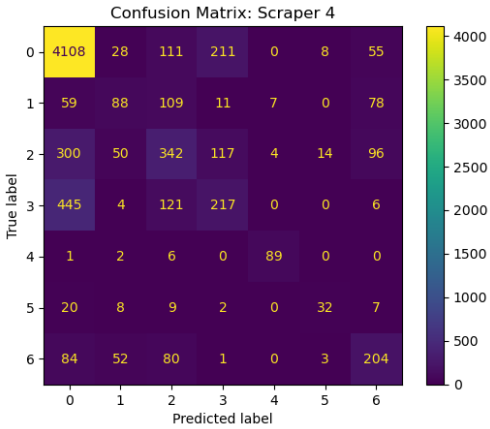
Figure 9: Scraper mistakes, multiple columns read together, multiple mistakes at the same time or mistake is not apparent.

Appendix B Confusion Matrices by Scraper

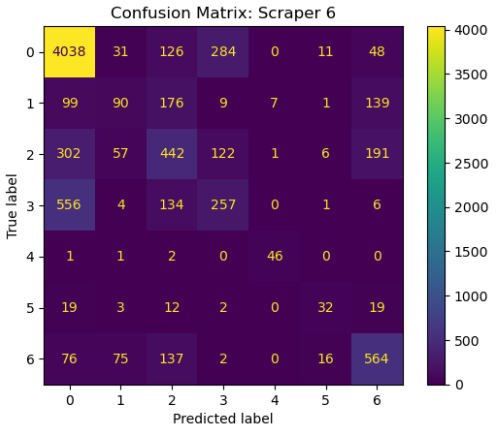


(a)

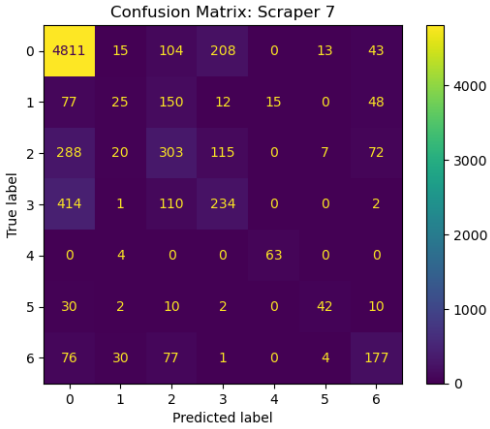
(b)



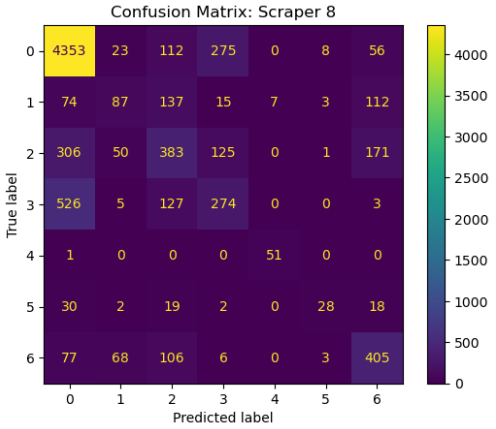
(c)



(d)



(e)



(f)

Figure 10: Confusion matrices evaluated on the test dataset and split by scrapers.

Appendix C Quality Score Histograms by Match Flag and Scraper

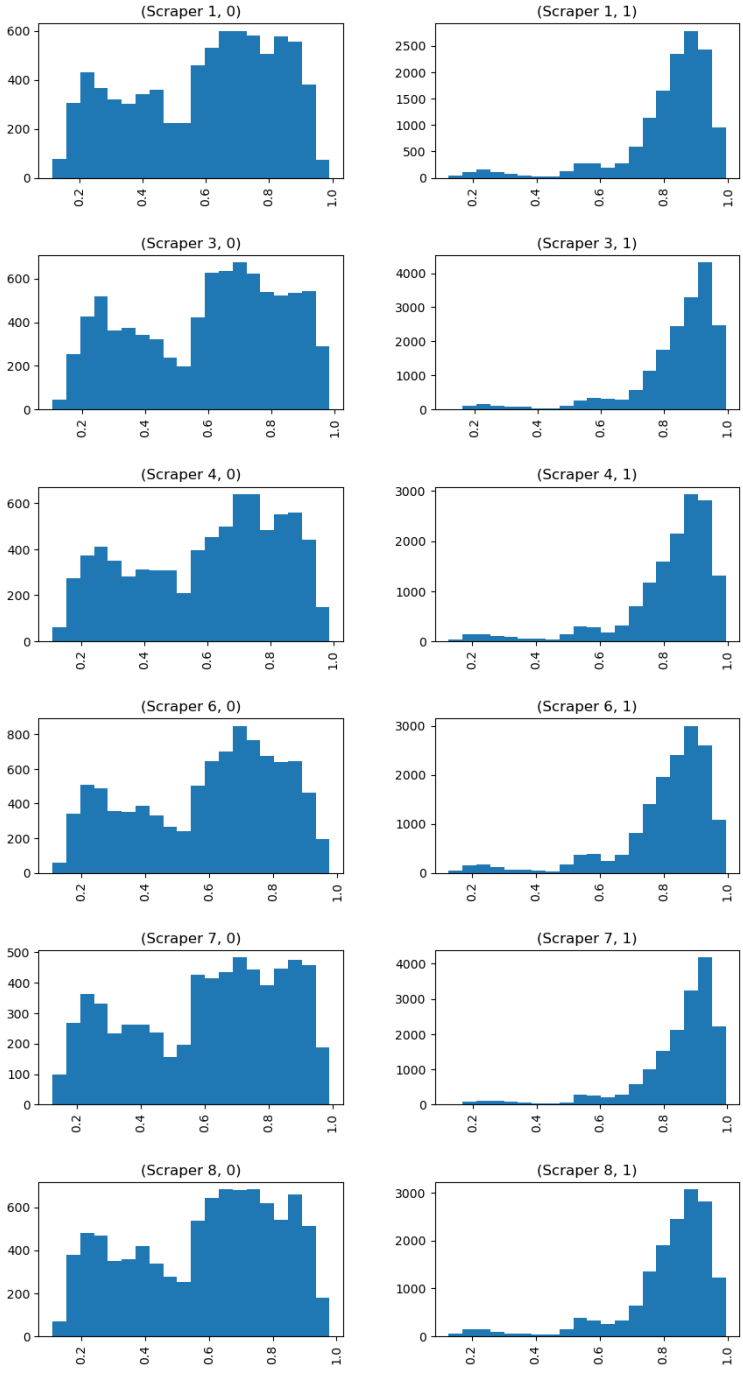


Figure 11: Histograms of quality score distribution of the validation data. Histograms are split by scraper and a match flag where value 0 indicates that the scraped value does not match its true counterpart and 1 indicates that it does match.

Appendix D Selected Sample of Suggested and Merged Values

Table 13: Sample of scraper value selection results compared to the alternative model (the merged value). Models are compared by the 'Match Flag' column that indicates which values correspond to the true one (represented through the secondary source value approximation) or not. Sample is not random, but selected to show more mismatch cases.

Row	Financial Figure	Predicted Error Label	Quality Score	Suggested Value	Merged Value	Secondary Source Value	Match Flag
1	other_short_term_receivables	meh digits, wrong OoM	0.32	18872.00	18.00	72000.00	both different and both incorrect
2	debt	good digits, exact OoM	0.89	2471271.00	2471271.00	2471000.00	both correct
3	assets	good digits, exact OoM	0.92	63971685.00	63971685.00	63972000.00	both correct
4	cash	good digits, 1000 OoM diff	0.66	816000.00	816.00	1000.00	merged correct, suggested incorrect
5	ebitda	good digits, exact OoM	0.72	-19828.00	-19828.00	-4000.00	both same, but incorrect
6	short_term_receivables_ from_sales_and_services	good digits, 1000 OoM diff	0.84	93050000.00	93050.00	93050000.00	merged incorrect, suggested correct
7	assets	good digits, exact OoM	0.98	239197.00	239197.00	239000.00	both correct
8	net_financial_income	good digits, exact OoM	0.84	-113407.00	-113407.00	-117000.00	both same, but incorrect
9	cash	good digits, 1000 OoM diff	0.65	5000.00	0.00	8000.00	both different and both incorrect
10	equity	good digits, exact OoM	0.97	205455.00	205455.00	205000.00	both correct
11	ebitda	good digits, exact OoM	0.86	352862.00	-647138.00	353000.00	merged incorrect, suggested correct
12	short_term_debt	good digits, exact OoM	0.94	168616.00	168616.00	169000.00	both correct
13	untaxed_reserves	meh digits, wrong OoM	0.47	2754000.00	2754000.00	44610000.00	both same, but incorrect
14	long_term_debt	good digits, exact OoM	0.81	26377799.00	26377799.00	26378000.00	both correct
15	short_term_debt	good digits, exact OoM	0.94	2192833.00	2192.00	2193000.00	merged incorrect, suggested correct
16	contributed_capital	good digits, exact OoM	0.99	100000.00	100000.00	100000.00	both correct
17	revenue	good digits, exact OoM	0.82	15945610.00	15945.00	15946000.00	merged incorrect, suggested correct
18	current_assets	good digits, exact OoM	0.90	738250.00	738250.00	738000.00	both correct
19	other_short_term_receivables	good digits, exact OoM	0.66	3901.00	3901.00	2360000.00	both same, but incorrect
20	debt	good digits, exact OoM	0.82	50312000.00	2049937.00	54570000.00	both different and both incorrect
21	equity	good digits, exact OoM	0.94	4053063.00	53.00	4053000.00	merged incorrect, suggested correct
22	profit_loss	good digits, exact OoM	0.95	-492000.00	-492000.00	-492000.00	both correct
23	net_financial_income	good digits, exact OoM	0.92	-2512.00	-2512.00	-3000.00	both correct
24	retained_earnings	good digits, exact OoM	0.82	26209508.00	26209508.00	26210000.00	both correct
25	ebitda	good digits, exact OoM	0.74	872033.00	872033.00	872000.00	both correct
26	retained_earnings	good digits, exact OoM	0.79	-1262887.00	-1257005.00	-1257000.00	merged correct, suggested incorrect
27	retained_earnings	good digits, exact OoM	0.77	18114709.00	1151.00	18115000.00	merged incorrect, suggested correct
28	contributed_capital	good digits, exact OoM	0.97	25000.00	25000.00	25000.00	both correct
29	debt	good digits, exact OoM	0.92	1636433.00	1636433.00	1636000.00	both correct
30	profit_loss	good digits, exact OoM	0.95	401062.00	401062.00	401000.00	both correct

References

[1] *Anaconda Software Distribution*. Version 2-2.4.0. 2020. URL: https://docs.anaconda.com/.

[2] Carlo Batini and Monica Scannapieca. *Data Quality*. Data-Centric Systems and Applications. Springer Berlin, Heidelberg, 2006. ISBN: 978-3-540-33173-5. DOI: https://doi-org.ludwig.lub.lu.se/10.1007/3-540-33173-5.

[3] Peter Benson. *NATO Codification System as the Foundation for ISO 8000, the International Standard for Data Quality*. Accessed: 2024-05-03. 2009. URL: https://oilit.com/papers/Benson.pdf.

[4] Bolagsverket. *Contents of an Annual Report*. Accessed: 2024-01-06. URL: https://bolagsverket.se/en/foretag/aktiebolag/arsredovisningforaktiebolag/delarochbilagoriarsredovisningen.761.html.

[5] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[6] Leo Breiman et al. *Classification and Regression Trees*. Wadsworth, 1984.

[7] Corinna Cichy and Stefan Rass. "An Overview of Data Quality Frameworks". In: *IEEE Access* 7 (2019), pp. 24634–24648. DOI: 10.1109/ACCESS.2019.2899751.

[8] Lisa Ehrlinger and Wolfram Wöß. "A Survey of Data Quality Measurement and Monitoring Tools". In: *Frontiers in Big Data* 5 (2022). ISSN: 2624-909X.

[9] Eurostat. *European Statistical System Handbook for Quality and Metadata Reports*. Publications Office of the European Union, 2021. DOI: 10.2785/616374.

[10] Rodrigo Fernandes de Mello and Moacir Antonelli Ponti. *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Springer International Publishing, 2018. ISBN: 9783319949888.

[11] Nissen Hans. "Uruk: Early Administration Practices and the Development of Proto-Cuneiform Writing." In: *Archéo-Nil. Revue de la société pour l'étude des cultures prépharaoniques de la vallée du Nil* 26.1 (2016), pp. 33–48. ISSN: 1161-0492.

[12] Charles R. et al. Harris. "Array Programming with NumPy". In: *Nature* 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, NY, 2009. ISBN: 978-0-387-84858-7. DOI: https://doi.org/10.1007/978-0-387-84858-7.

[14] John D Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[15] Thomas Kluyver et al. "Jupyter Notebooks – a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87–90.

[16] Lars Lundberg (1962) et al. "Bibliometric Mining of Research Trends in Machine Learning". In: *AI* 5.1 (2024), pp. 208–236.

[17] Wes McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. 2010, pp. 51–56.

[18] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[19] Laura Sebastian-Coleman. *Measuring Data Quality for Ongoing Improvement: A Data Quality Assessment Framework*. Morgan Kaufmann, 2013. ISBN: 9780123970336.

[20] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[21] *sklearn.ensemble.RandomForestClassifier*. Accessed: 2024-05-09. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.predict_proba.

[22] *sklearn.metrics.balanced$_a$ccuracy$_s$core*. Accessed: 2024-05-10. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn.metrics.balanced_accuracy_score.

[23] *sklearn.metrics.roc$_a$uc$_s$core*. Accessed: 2024-05-10. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#r4bb7c4558997-5.

[24] *sklearn.preprocessing.LabelEncoder*. Accessed: 2024-05-06. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder.

[25] *sklearn.tree.DecisionTreeClassifier*. Accessed: 2024-05-09. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[26] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[27] Vladimir N. Vapnik. “An Overview of Statistical Learning Theory”. In: *IEEE Transactions on Neural Networks* 10.5 (1999), pp. 988–999. DOI: 10.1109/72.788640.

[28] Vladimir N. Vapnik. *Statistical Learning Theory*. Adaptive and learning systems for signal processing, communications and control. Wiley, 1998. ISBN: 0471030031.

[29] Ulrike Von Luxburg and Bernhard Schoelkopf. *Statistical Learning Theory: Models, Concepts, and Results*. 2008. arXiv: 0810.4752 [stat.ML].

[30] Ou Wu. *Rethinking Class Imbalance in Machine Learning*. 2023. arXiv: 2305.03900 [cs.LG].