



Text mining of the company's annual reports in PDF format  
**Svetlana Zamyatina**



Dissertation  
Master in Modelling, Data Analysis and Decision Support Systems



Supervised by  
**PhD João Gama**  
**PhD Bruno Miguel Delindro Veloso**



2022

# Acknowledgments

First of all, I would like to show my gratitude to my supervisors, Professors João Manuel Portela da Gama and Bruno Miguel Delindro Veloso, that accepted to take on this challenge with me and taught me so much during the process.

Also, to Professors Maria Eduarda da Rocha Pinto Augusto da Silva and Jorge Miguel Silva Valente, who was always available to clarify my doubts.

To the institution that hosts me, the Faculty of Economy of the University of Porto (FEP), as well as to all of its docents that guide me during this Master's program, I am thankful for everything I have learned until now.

To my parents, Valentina and Anatoliy, for making this journey possible and always supporting me every step of the way.

To my husband, Armando Jorge, and our daughter Ekaterina for always being such a great role model that pushed me to become a better person and invest in my education, for his endless patience, and for believing in me even when I didn't. He made the process easier and encouraged me to always try my best.

To Banco de Portugal and Professor Nuno C. Azevedo for all the support and availability during the internship that gave the mote and data to develop this thesis.

Finally, to all my friends.

Svetlana Zamyatina

# Abstract

The digitalization of the economy actively influences both financial and credit institutions in general, and the activities of Banks in particular, which determines the tasks of quality rethinking the consequences of introducing banking innovations, manifested in the creation of modern banking technologies and innovative banking infrastructure. Marked gives mediate the need to form a theoretical and methodological basis for studying the development of banking innovations in the digital economy.

A current challenge in banking information is the need to treat customer financial statement data more efficiently. Much of the information that appears in financial statements is in an unstructured format, normally written in PDF format. All these PDF format barriers make the banking information management process difficult. However, automatic tasks can make the process easier.

An example is the automatic extraction of relevant information from Financial statement documents in PDF format, including extracting information from tables.

In this Thesis, we created a system capable of automatically determining which pages of the financial statements contain the Balance – sheet and Income Statement; in the second phase, we developed an algorithm that may be applied to other financial statements. In the third phase, we extracted the previously identified into an editable file, according to a pre-defined structure, preferably in CSV, Excel formats.

The Bank of Portugal provides financial statements.

**Keywords:** Text mining, Python, tables extraction, recognition, Financial statements, PDF format.

# Resumo

A digitalização da economia influencia ativamente as instituições financeiras e de crédito em geral, e as atividades dos Bancos em particular, o que determina as tarefas de qualidade repensando as consequências da introdução de inovações bancárias, manifestadas na criação de tecnologias bancárias modernas e infraestruturas bancárias inovadoras. Dados marcados medem a necessidade de formar uma base teórica e metodológica para estudar o desenvolvimento das inovações bancárias na economia digital.

Um desafio atual nas informações bancárias é a necessidade de tratar os dados das demonstrações financeiras dos clientes de forma mais eficiente. Muitas das informações que aparecem nas demonstrações financeiras estão em um formato não estruturado, normalmente escrito em formato PDF. Todas essas barreiras do formato PDF dificultam o processo de gerenciamento de informações bancárias. No entanto, tarefas automáticas podem facilitar o processo.

Um exemplo é a extração automática de informações relevantes de documentos de Demonstrações Financeiras em formato PDF, incluindo a extração de informações de tabelas. Nesta Tese, criamos um sistema capaz de determinar automaticamente quais páginas das demonstrações financeiras contêm o Balanço Patrimonial e a Demonstração do Resultado; na segunda fase, desenvolvemos um algoritmo que pode ser aplicado a outras demonstrações financeiras. Na terceira fase, extraímos os previamente identificados em um ficheiro editável, de acordo com uma estrutura pré-definida, preferencialmente nos formatos CSV, Excel.

O Banco de Portugal disponibiliza demonstrações financeiras.

**Palavras-chave:** Mineração de texto, Python, extração de tabelas, reconhecimento, Demonstrativos financeiros, formato PDF.

# Glossary

DM – Data mining

TM – Text mining

PDF – Portable Document Format

CSV - Comma Separated Values

OCR – Optical Character Recognition

NLRK – Natural Language Toolkit

PEP – Python Enhancement Proposal

SQL - Structured Query Language

HTML - HyperText Markup Language

JSON - JavaScript Object Notation

DOCX – Office Open XML Document

ODT – Open Document Text

# Contents

Acknowledgment . . . . .	i
Abstract . . . . .	ii
Resumo . . . . .	iii
Glossary . . . . .	iv
<b>1 Introduction</b>	<b>1</b>
1.1 Context . . . . .	1
1.2 Research problem . . . . .	2
1.3 Motivation and goals. . . . .	2
<b>2 Literature Review</b>	<b>4</b>
2.1 Tasks of the text analysis . . . . .	4
2.1.1 Stages of text mining . . . . .	4
2.1.2 Possible documents structures . . . . .	6
2.2 PDF format . . . . .	7
2.2.1 Advantages and disadvantages of PDF format. . . . .	7
2.2.2 PDF data types . . . . .	7
2.2.3 Description of the PDF document . . . . .	8
2.2.4 Document structure . . . . .	9
2.2.5 PDF contains . . . . .	9
<b>3 Research Methodology and instruments</b>	<b>11</b>
3.1 Table definition . . . . .	11
3.2 Main task of the PDF parsing. . . . .	11
3.3 Python language . . . . .	12
3.3.1 Python libraries to read PDF files and parse tables. . . . .	13
3.3.1.1 Splitting the documents library: Pikepdf . . . . .	15

3.3.1.2	Text extraction libraries: Pdfplumber, PDFMiner, Pdf2image, OCR, Pytesseract . . . . .	15
3.3.1.3	Preprocessing (or lemmatization) libraries: SpaCy, Nltk . . . . .	17
3.3.1.4	Table Extraction libraries: Tabula, Camelot. . . . .	17
3.3.1.5	Table Handling library: Pandas. . . . .	19
<b>4</b>	<b>The main components of the algorithm</b>	<b>20</b>
4.1	Development workflow. . . . .	20
4.1.1	Text preprocessing . . . . .	23
4.1.2	Scoring of the words . . . . .	28
<b>5</b>	<b>Experiments and results</b>	<b>30</b>
5.1	Testing the Camelot library . . . . .	30
5.2	Extracting the tables using the markup of the image . . . . .	31
5.3	Version 1. . . . .	35
5.4	Version 2. . . . .	38
5.5	Version 3. . . . .	42
5.6	Version 4. . . . .	47
5.7	Version 5. . . . .	49
<b>6</b>	<b>Conclusions and Future Work</b>	<b>58</b>
	<b>Appendix</b>	<b>i</b>
	<b>References</b>	<b>v</b>

# List of Figures and Tables

<b>Figure 1.</b> Stages of the text mining.....	i
<b>Figure 2.</b> Examples of the data structure (adapted from Mike Bergman, 2005) .....	6
<b>Figure 3.</b> PDF elements document .....	i
<b>Figure 4.</b> PDF Document structure .....	ii
<b>Figure 5.</b> User Space coordinates on a PDF page .....	9
<b>Figure 6.</b> Table example from one of PDF files .....	ii
<b>Figure 7.</b> Standard example .....	iii
<b>Figure 8.</b> Extracted data using the Tabula library from a page containing two tables .....	iii
<b>Figure 9.</b> Distance measurement parameters with Tabula .....	18
<b>Figure 10.</b> Element types found on the PDF page .....	iii
<b>Figure 11.</b> Extracted data using the Camelot library from a page containing two tables ..	iv
<b>Figure 12.</b> Development workflow .....	21
<b>Figure 13.</b> Text Preprocessing .....	24
<b>Figure 14.</b> PDF page before processing .....	26
<b>Figure 15.</b> Processed text .....	26
<b>Figure 16.</b> Scoring of the words .....	28
<b>Figure 17.</b> Extracted tables in CSV format during testing of the Camelot library .....	30
<b>Figure 18.</b> The result of testing the Camelot library .....	31
<b>Figure 19.</b> The page with 2 tables of the CTT Company in PDF format .....	32
<b>Figure 20.</b> The markup of the image using the Camelot library .....	32
<b>Figure 21.</b> The extracted tables using the Tabula library .....	33
<b>Figure 22.</b> Workflow of the code development .....	34
<b>Figure 23.</b> The DataFrame with divided into groups with the total number of pages found for each group .....	36
<b>Figure 24.</b> Page in PDF format with 2 tables located on the same page .....	36
<b>Figure 25.</b> The result of the extraction of the 2 tables located on the same page .....	37
<b>Figure 26.</b> The diagrams of the accuracy of finding the correct tables using Version 1 ..	38
<b>Figure 27.</b> The HAM 6 code learning algorithm is based on reporting of the 6 companies .....	38
<b>Figure 28.</b> The DataFrame with divided into groups with the largest number of total _words in each group .....	40
<b>Figure 29.</b> The Extracted to xlsx files page 135 of the company LISGRÁFICA .....	40
<b>Figure 30.</b> The Extracted to xlsx files page 134 of the company LISGRÁFICA .....	41
<b>Figure 31.</b> The diagrams of the accuracy of finding the correct tables using Version 2 ...	42
<b>Figure 32.</b> The HAM 12 code learning algorithm is based on reporting of the 12 companies .....	42
<b>Figure 33.</b> The Extracted to xlsx file page with extracted NIPC and Year .....	44



<b>Figure 34.</b> Page in PDF format with the table Income Statement is vertical .....	45
<b>Figure 35.</b> The Extracted to xlsx files page using the pdfplumber library .....	45
<b>Figure 36.</b> The diagrams of the accuracy of finding the correct tables using Version 3 ...	46
<b>Figure 37.</b> The pages recognition result uses the algorithm that calculates the maximum number the matches words from the Group .....	47
<b>Figure 38.</b> The pages recognition result uses the algorithm that calculates the largest number of total _words in each group .....	48
<b>Figure 39.</b> The diagrams of the accuracy of finding the correct tables using Version 4 ...	49
<b>Figure 40.</b> Page in PDF format to extract data using the code Version 5 .....	50
<b>Figure 41.</b> The result of the data extraction using code Version 5 (without using OCR) and exported in xlsx .....	51
<b>Figure 42.</b> The results of the tested code Version 5 effectively identify the pages we need.....	51
<b>Figure 43.</b> Page in PDF format (digitalized) to extract data using the code Version 5 with using OCR .....	52
<b>Figure 44.</b> The result of the data extraction using code Version 5 with using OCR and exported in xlsx .....	53
<b>Figure 45.</b> The diagrams of the accuracy of finding the correct tables using Version 5 ..	54
<b>Figure 46.</b> Installing libraries required for analysis .....	55
<b>Figure 47.</b> Selection of the first place of the developed rating.....	56
<b>Figure 48.</b> The diagram of the quality of progress .....	57

<b>Table 1.</b> Comparison table of the used Python libraries.....	14
--------------------------------------------------------------------	----

# 1 | Introduction

This project is dedicated to building an algorithm capable of offering useful tools to select, process, and extract data from a document corpus. The main goal was to create a dynamic algorithm capable of performing data preprocessing and extracting the necessary information from documents in PDF format (Portable Document Format).

During the preparation of the Thesis, we presented the proposed methodology used to achieve the stated goals, the various application modules, and various operations that constitute the selection, preprocessing, and extraction of the necessary data applied to several data sets. We have considered the state-of-the-art analysis of the approaches used in TM (Text Mining) to achieve this solution, compared the various document processing methods, and pointed out how this structure was developed. Below we will describe several algorithms used in processing to extract text from tables with conversion to CSV (Comma Separated Values format), which is the intended purpose.

## 1.1 Context

TM has evolved to offer an approach to efficiently extracting information from text, creating tools to analyze the causal relationship between features, also in unstructured data, including those containing tables.

Every year the importance of TM increases in a wide variety of fields of activity. This allows us to develop and improve new algorithms that will later help extract texts more clearly and accurately, including in the form of tables with the ability to save results in various formats, for example, in CSV format. This can be helpful for companies in various fields of activity, including in financial institutions in general and in the banking sector, for the clarity and reliability of data extraction in accordance with the goal.

It is a complex process of obtaining high-quality information, although the knowledge gained through parsing followed by data processing and analysis can achieve high accuracy in the process of extracting data.

## 1.2 Research problem

Today, TM has a set of powerful techniques that still have significant limitations. Existing TM approaches take little advantage of the existing structure and relationships in texts. TM algorithms treat texts as a collection of words, which can make the algorithms less efficient. Some techniques can code and manipulate relational information to help improve TM algorithm's performance. In this regard, the main goal of our research was to develop an effective algorithm with applied some known TM tools that can qualitatively extract the data we need from documents in PDF format while facilitating the workflow.

## 1.3 Motivation and goals

DM (Data mining) is one of the most relevant and demanded areas. Today's business and manufacturing processes generate huge amounts of data, and it is becoming increasingly difficult for companies to interpret and respond to large amounts of data that change dynamically at runtime.

DM is an interdisciplinary field of knowledge at the intersection of traditional statistical analysis, artificial intelligence, machine learning, and large database development. The essence of the philosophy of DM is partially expressed in the name of this area of knowledge, which consists of two concepts: the search for valuable information in a large database (data) and mining. It is in sifting through a sieve of their tools a huge amount of "raw", often unstructured data in search of nuggets meaningful, non-trivial information - knowledge.

The original definition of the term, which Grigory Pyatnitsky-Shapito [33] gave, is as follows: "DM is the process of discovering in raw data previously unknown non-trivial, practically useful and accessible interpretations of knowledge necessary for decision-making in various areas of human activity."

TM - text mining is a specific area of DM that analyzes textual. By analogy with the term DM, the term TM can be given the following definition - this is a non-trivial process of discovering new, potentially useful, and understandable patterns in unstructured text data.

Because TM is such a useful tool today, it's important to improve its presentation. For this reason, this dissertation is intended to explore state of the art in TM and create a flexible system that allows us to perform multiple text preprocessing tasks to test which ones are the most efficient and get the best result.

This work intends to develop an algorithm that uses traditional TM methods and evaluates the extent to which traditional approaches apply to extracting text from tables. Unstructured text processing involves the use of more powerful computing resources.

In this thesis, we presented the proposed methodology used to achieve the stated goals, the various application modules, and various operations that constitute the selection, preprocessing, and extraction of the necessary data applied to several data sets. We considered the state-of-the-art analysis of the approaches used in TM to achieve this solution, compared the various document processing methods, and pointed out how this structure was developed.

The theoretical basis of the dissertation was the work of foreign and domestic experts, substantiating scientific and theoretical directions and concepts, principles of formation and development of banking activities in the context of the introduction of TM, and key definitions concerning the goals of the dissertation being implemented (Development of a platform capable of performing data extraction; Extracting data from tables; Develop an algorithm which may be applied in other Financial statements; Processing documents in PDF format; Saving the results in CSV format).

## 2 | Literature Review

The theoretical basis of the dissertation was the work of foreign and domestic experts, substantiating scientific and theoretical directions and concepts, principles of formation and development of banking activities in the context of the introduction of TM, and key definitions concerning the goals of the dissertation being implemented: Development of the algorithm capable of performing data extraction; Extracting data from tables; Develop an algorithm which may be applied in other Financial statements; Processing documents in PDF format; Saving the results in CSV format.

For this purpose, this chapter is structured as follows:

Section 2.1 sets some terminology on text analysis tasks, provides background on TM stages, and gives examples of possible document structures.

Section 2.2 introduces terminology and background work on PDF format: Highlights advantages and disadvantages of PDF format; gives the PDF data types definition and description of the PDF document and document structure; examples of possible PDF file contain are given.

### 2.1 Tasks of the text analysis

#### 2.1.1 Stages of TM

For the first time, TM techniques appeared in the mid-1980 s, and in the next decade, the development of technology allowed them to improve significantly. In an interdisciplinary sense, TM lies at the intersection of information retrieval, DM, machine learning, statistics, and computational linguistics.

The need to use huge amounts of corporate information that exists in an unstructured form has long been known. But special technologies that allow us to work with texts and not quantitative data appeared only in the late 90s [1].

Relevance of the problem:

Text Mining technology is one of the varieties of DM methods and involves extracting knowledge and high-quality information from text arrays. This usually happens through the identification of patterns and trends [2].

The statistical study of patterns. This technology of in-depth analysis of texts can "sift" large volumes of unstructured information and identify only the most significant of them so that a person does not have to spend time on the extraction of valuable knowledge "manually" [1].

Examples of such documents are web pages, PDF documents, regulatory documents, etc. Generally, such documents can be complex and large and include text and graphical, tabular information [2].

The results of TM can be used for mathematical forecasting, bank risk analysis, and market analysis [1].

TM involves the process of structuring input textual data, extracting patterns from already structured data, and the final evaluation and interpretation of the results [1].

Analyzing text documents can be represented as a sequence of several steps. This methodology's entire data processing cycle is represented by six successive stages [1]. The main six stages of the TM have presented in *Fig. 1 Stages of the text mining*.

**Stage 1. Determining the objectives of the study.** This is the beginning of almost any meaningful activity. Good goal setting requires a deep understanding of all aspects of the situation in which the research is being conducted and a clear definition of the result that we want. To do this, it is necessary to study the problem to be solved by the study[1].

**Stage 2. Assess the availability and nature of the data.** This stage includes the following tasks:

- Identification of text sources. The text may be digital or written on paper and maybe located inside or outside the organization under study.
- Evaluation of the availability and applicability of data.
- Collection of primary data.
- Evaluation of the content of the data (whether they contain the information necessary for the study).
- Evaluation of the quantity and quality of data.

After the exploration part of the study is completed, we can start collecting data from various sources [1].

**Stage 3. Data preparation.** Data preparation is a necessary stage for TM because the specificity of this method, compared to DM, lies in the more time-consuming stages of collecting and processing data.

The data preparation stage consists of the following phases:

- **Corpus creation.** In linguistics, a corpus is a large structured collection of texts. At this stage, collecting all text documents related to the problem under study is necessary. The researcher has to decide what data and volumes it is necessary to collect and analyze to solve the problem. It should be remembered that all DM methods are highly dependent on the accuracy of the results obtained and their number. Once the documents are collected, they need to be transformed so that they are presented in a single form (for example, in a text file) for computer processing.

- Data preprocessing [34] is the first and essential step in machine learning because the quality of the data and the useful information that can be obtained from it directly affects the learning ability of our algorithm, so it is extremely important that we pre-process our data. Without preliminary data processing, a banking specialist will not be able to get the job done.

Main tasks at the stage of data preprocessing:

- Processing of zero values;
- Data normalization;
- Emission control;
- Processing of categorical features [34].

**Stage 4. Development and calibration of the model.** At this stage, knowledge extraction methods are applied [1].

**Stage 5. Checking the results.** After the model is created and configured, we must perform a general check of all actions. For example, we need to ensure that the selection is made correctly. It also happens that in constructing a study, the main goal for which it was started is lost. At this stage, it is necessary to check whether the model solves the formulated problem and thus serves the achievement of the goal. If something is missed, it is necessary to go back to the stage that created the mismatch between the goal and the result [1].

**Stage 6. Implementation.** If it was decided that the model solves the problem based on the checks results, it can be applied. At its simplest, an implementation may take the form of writing a report on the results of a study. In the complex - building an intelligent system based on the built model so that it can be reused for decision making [1].

Visualization can also be used as a text analysis tool. To do this, key concepts are extracted, which are presented graphically.

Based on the results of this chapter, the following conclusions can be drawn.

Knowledge discovery in the text is a non-trivial process of discovering new, potentially useful and understandable patterns in unstructured text data [3].

## 2.1.2 Possible document structures

Text information extraction consists of extracting information from semi-structured [4] or unstructured data.

The figure shows an example of structured, semi-structured, and unstructured data.



**Fig. 2 Examples of the data structure (adapted from Mike Bergman, 2005).**

Unstructured data does not require formatting, as it is normally in free text form. The text has structure, but that is just a structure that follows linguistic rules meant for humans and not computers. As mentioned earlier, many documents fall into this category [4].

The semi-structured ones have some structure, such as the European Curriculum Vitae presentation model [4].

As for structured data, they already obey strict rules and restrictions. Relational databases are one of the possible examples of this type of data [4].

## **2.2 PDF format**

### **2.2.1 Advantages and disadvantages of PDF format**

PDF (Portable Document Format), is not just a well-known and convenient format, but the only one that is the standard. The developers of the format set themselves the task of creating such a document display format so that the document opens and looks the same on any device. The PDF it is not as simple as DOCX (Office Open XML Document) or ODT, (OpenDocument Text) yet it is still natively a text rather than a binary format [5].

Advantages and disadvantages [5].

Pros:

- standardization and popularity: it opens on any device with any operating system exactly in the form in which it was created;
- takes up little space on the hard drive because it supports many compression algorithms.
- security: the user can configure security settings for his file, for example, prohibit printing, prohibit editing, use an electronic signature to determine the authenticity of a document, etc. [5].

Minuses:

- PDF editor - paid;
- editing pdf files in specialized programs is more complex than any other graphic files [5].

### **2.2.2 PDF data types**

PDF supports several basic data types, some of which I will describe below: strings, arrays, dictionaries, streams, and objects.

**Strings:** Lines PDF inherited from PostScript as a result, a line in .pdf means a sequence of 8-bit characters surrounded by parentheses. The string can be wrapped to the next line using a backslash, which is not part of the string and, among other things, escapes special characters [6].



**Arrays:** in PDF are enclosed in square brackets and are simply a sequence of grouped objects. For example: [(Hello,)10(world!)]. Arrays sometimes contain text strings [7].

**Dictionaries:** These are key-value pairs wrapped in << H >>. A dictionary is often used to give the object that contains it the properties described in it. This data will help us determine how, for example, to decrypt the stream, find out its length, or, conversely, discard the current object as uninteresting (if it is an image) [7].

**Streams:** represent a sequence of eight bits of data between the stream and endstream keywords. Any binary data, be it compressed text, an image, or an embedded font, will be presented as a stream. A stream is always inside an object (just below) and is characterized at least by its length (option /Length N in the dictionary) and very often by its compression method (for example, /Filter /FlateDecode). PDF supports a sufficient number of compression formats (including the /CryptDecode encryption format). In streams, we can search for the text we want to get from a PDF document [7].

Objects are the largest structure to work with. An object can contain any other type of data, from an ordinary number to a stream, framed by the obj and endobj keywords. An object has its own ID within the document by which it can be referenced[7].

Where is it used:

The PDF format is used to store and transmit textual and graphic information on the network, for example, to transfer letterhead to printing to store financial statements. Great for demonstrating developments: easy to show, difficult to edit. This makes intellectual property theft more difficult [6].

### 2.2.3 Description of the PDF document

The PDF text format is becoming the standard for electronic document management worldwide. Therefore, programmers regularly face the task of extracting text from such files. Along with it, more complex tasks appear, for example, obtaining the text structure of a document. [7]. The basic structure of a PDF file is shown in figure 3 [7]: *Fig. 3 PDF elements document.*

PDF is a text format but has a rather complex structure. It can contain media, links, and more. Consider a simple test document.

The file structure includes four sections [7]:

Each PDF document has the following elements:

- **The header** is the first line of the file. It contains information about the PDF version.
- **Body.** All the content of the document is in the body of the file. The data types in the body will be discussed in the next section.

- **xref Table.** This table contains links to all objects in the document. Thanks to it, we do not need to read the entire document to find the desired object. An xref table is made up of sections. Each section corresponds to a new version of the document. The reference table starts with the word xref. Each section starts with two numbers: the ID of the first object and the number of objects in the section. Two sequences of bytes represent objects. The first corresponds to the position of the first byte of the object in the file, and the second corresponds to the generation number [7].

- **Tailer.** This section gives information about the location of key objects in the file. For example, it contains the offset of the xref table from the beginning of the file. Therefore, any programmatic reading of a PDF document starts at the tail [7].

## 2.2.4 Document structure

The document's content comprises objects from the body of the file. They may contain links to each other. The reference structure of objects is a tree. At the root is an object called the document directory. Its children are important building blocks, one of which is the page tree.

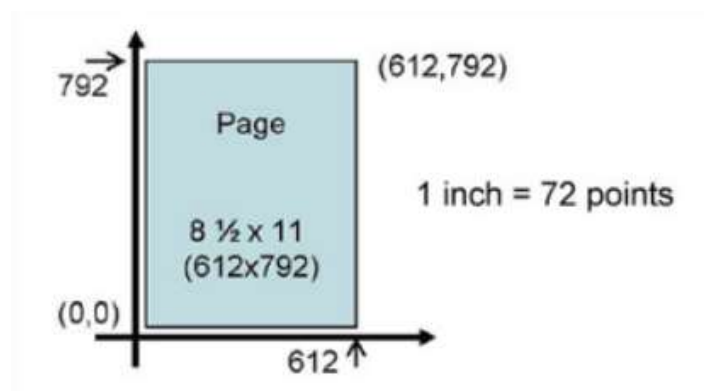
The document directory refers to the root of the page tree. The leaves of the tree are the pages. Each tree node contains information about the parent, children, and the number of leaves among the children. Each page document page can be found by the /Page entry, the root of the page tree by the /Pages entry, and the document directory by /Catalog.

The entire structure of a PDF document can be represented in figure 4 [7]: *Fig. 4 PDF Document structure.*

The figure above, shows that the document catalog contains links to the page tree, path hierarchy, article flows, named recipients, and interactive form. [7].

## 2.2.5 PDF file contains

The PDF file contains instructions for placing a character at x, y coordinates on a 2-D plane without any knowledge of words, sentences, or tables [8].



**Fig. 5 User Space coordinates on a PDF page**

The coordinate system on a PDF page is called User Space. This is a flat 2-dimensional space, just like a piece of paper. The units of User Space are called "points", and 72 points/inch. The origin or 0,0 point is located in the bottom left-hand corner of the page. Horizontal, or X, coordinates increase to the right and vertical, or Y, coordinates increase [8] towards the top (Fig. 5).

## 3 | Research methodology and instruments

In working on the Thesis, it was necessary to process documents of the company's financial statements in PDF format to extract text information from the tables.

The main goal of the thesis was to create an algorithm capable of automatically determining which pages of the Financial statements contain the Balance - sheet and Income Statement. To develop an algorithm that may be applied to other financial statements and to extract the previously identified into an editable file, according to a pre-defined structure, preferably in CSV, Excel formats.

The Bank of Portugal provides financial statements.

### 3.1 Table definition

*“A table is an object which uses linear visual cues to simultaneously describe logical connections between the discrete content entries in the table. A content entry is the basic component of information in the table [...] (and) can be any visual symbol”. Cameron (1989).*

According to different sources, ‘table’ has various definitions [9]. In this Thesis, we are developing an automated method of extraction of tables from PDF files. We analyzed some documents manually to see which of the definitions suited best for this work.

Looking at fig. 6 we can see that layout table content differs from standard (Fig. 7), which has repetitive structure or the same data type in a column or a row and clear contours. In addition, some spaces are separating the columns. This implies that the definition of Astrakhantsev et al. [9] is appropriate for this work. The table is a set of cells with some text content.

A table example from one of the PDF files is presented in Fig. 6 *Table example from one of the PDF files*. A standard example of a table is presented in Fig. 7 *Standard example*.

### 3.2 Main task of the PDF parsing

Parsing means to parse a word or text. Thus, parsing is a method in which a string or text is parsed and broken down into syntactic components. Then the received data is converted into a suitable format for further processing and use in applied research. It turns out that one data format turns into another, more readable one [10].

**During the internship I used PDF parsing method:**

- splitting a PDF document into separate pages and saving them;
- selecting pages by keywords,
- extracting text and pages from PDF and conversion of found pages into excel tables, merging multiple xlsx sheets into one sheet [10].

### 3.3 Python language

Python is an interpreted high-level, general-purpose programming language developed in the early 1990s by Guido van Rossum [12]. Python is a beautiful, concise, and feature-rich language. It is used in fields of activity that are completely incompatible with each other, it can create projects alone and not look at the possibilities of competitive languages.

A high-level, interpreted, object-oriented, imperative, strongly typed, general-purpose language that is dynamically typed.

High-level languages are designed for ease of use and speed of writing a program. They use certain abstractions - data structures, auxiliary functions, etc.

It is important to note that Python is a scripting language. This means the code is checked for errors and immediately executed without additional compilation or rework. This approach is also called interpretive. [15].

Python is one of the most used languages in Data Science. Algorithms of programs with machine learning and analytical applications are written on them. It is used for data storage and cloud services. It also helps to parse data.

An unusual language feature is the separation of code blocks with spaces.

Primitive types in Python include booleans, arbitrary precision integers, floating points, and complex types. Built-in Python container types: string, list, tuple, dictionary, and set [13].

All values are objects, including functions, methods, modules, and classes.

At the moment, the version of the Python 3 language is being actively developed. Language development is conducted through PEP (Python Enhancement Proposal) language extension proposals, which describe innovations, adjust based on community feedback, and final document decisions [14]. Accordingly, we decided to use the latest release Python 3.10, for our work.

A Python variable is a symbolic name that is a reference or pointer to an object. Once an object has been assigned to a variable, we can refer to the object by that name. But the data itself is still contained in the object.

A variable is a name used to refer to a location in memory. A Python variable is also known as an identifier and is used to store a value.

Primitive types in Python include boolean, arbitrary precision integer, floating-point, and complex. Python's built-in container types are string, list, tuple, dictionary, and set [13]. All values are objects, including functions, methods, modules, and classes. An unusual language feature is the separation of code blocks with spaces.

The class system supports inheritance (single and multiple) and metaprogramming. It can be inherit from most built-in and extension types [11].

The software (application or library) in Python is made in the form of modules, which can be assembled into packages. Modules can be located both in directories and in ZIP archives. The module is connected to the program using the import statement. Once imported, a module is represented by a separate object that gives access to the module's namespace.

The module can be reloaded with the `reload()` function. It is also possible to connect modules designed in other programming languages. This makes Python very flexible [14].

Python's most powerful and attractive feature is the rich internal standard library. (12) A set of modules for working with the operating system allows you to write cross-platform applications.

In addition to the standard library, many libraries provide an interface to all system calls on different platforms; in particular, on the Win32 platform, all Win32 API calls are supported, as well as COM calls to the extent not less than that of Visual Basic or Delphi. The number of Python application libraries in various areas is huge (web, databases, image processing, word processing, numerical methods, operating system applications, etc.).

### 3.3.1 Python libraries to read PDF files and parse tables

One of the interesting parts for us is the application of libraries for working with PDFs and tables: [pikepdf](#) - splitting a document into pages and saving the result; [os](#) - creating new directories; [tqdm](#) - creating a progress bar for cycles.

Libraries [pdfminer](#) and [pdfplumber](#) - are an approach to reading and parsing data from PDF page-by-page. [Spacy](#) - text lemmatization. [Camelot](#) and [Tabula](#) - the most popular and effective ways to extract tables from PDF. [Pdf2image](#) - convert pdf to image; [Pytesseract](#) - extract text from PDF using OCR (Optical Character Recognition). The library also works with the computer utility. [Pandas](#) - manipulating tables obtained with Tabula.

Table 1 provides a comparison of the Python libraries we used. Namely, what are their advantages and disadvantages. Based on the characteristics of the libraries and in the process of testing them, only those libraries that showed the best results were selected for further development of the code.

№	Library	Advantages	Disadvantages
<b>Splitting the documents library</b>			
1	<b>Pikepdf</b>	<ul style="list-style-type: none"> <li>• Easy to install and use</li> </ul>	There are no disadvantages
<b>Text extraction libraries</b>			
2	<b>Pydfplumber</b>	<ul style="list-style-type: none"> <li>• works fast</li> <li>• the highest quality of recognition among libraries that do not use OCR</li> </ul>	<ul style="list-style-type: none"> <li>• only works with documents generated automatically</li> </ul>
3	<b>PDFminer</b>	<ul style="list-style-type: none"> <li>• works fast</li> </ul>	<ul style="list-style-type: none"> <li>• only works with documents generated automatically</li> <li>• recognizes less text than pdfplumber</li> </ul>
4	<b>Pytesseract</b>	<ul style="list-style-type: none"> <li>• recognizes almost all texts correctly</li> <li>• works with scanned documents</li> </ul>	<ul style="list-style-type: none"> <li>• works for a long time</li> <li>• need to convert pdf to jpeg first</li> <li>• installation may take a long time</li> </ul>
5	<b>Pyf2image</b>	<ul style="list-style-type: none"> <li>• high-quality recognition</li> <li>• works fast</li> </ul>	<ul style="list-style-type: none"> <li>• difficult to install (may have problems with the poppler library)</li> </ul>
<b>Preprocessing (or lemmatization) libraries</b>			
6	<b>SpaCy</b>	<ul style="list-style-type: none"> <li>• there is a lemmatizer for Portuguese</li> </ul>	<ul style="list-style-type: none"> <li>• lemmatizer works for a long time</li> <li>• splitting into tokens (words) is not very convenient</li> <li>• less relevant list of stop words</li> </ul>
7	<b>NLTK</b>	<ul style="list-style-type: none"> <li>• flexible and convenient implementation of splitting into tokens</li> <li>• most relevant list of stop words</li> </ul>	<ul style="list-style-type: none"> <li>• no lemmatizer for Portuguese</li> </ul>
<b>Table Extraction libraries</b>			
8	<b>Tabula</b>	<ul style="list-style-type: none"> <li>• higher quality table recognition</li> </ul>	<ul style="list-style-type: none"> <li>• installation may take a long time</li> <li>• does not work with scanned pdf files</li> </ul>
9	<b>Camelot</b>	<ul style="list-style-type: none"> <li>• easy installation</li> <li>• more features: visual debugging - visualization of text and tables found on the page</li> </ul>	<ul style="list-style-type: none"> <li>• worse quality of table recognition</li> <li>• does not work with scanned pdf files</li> </ul>
<b>Table Handling library</b>			
10	<b>Pandas</b>	<ul style="list-style-type: none"> <li>• easy to install</li> <li>• popular library with a wide range of features</li> <li>• easy to save the table in excel or csv</li> <li>• available documentation</li> </ul>	There are no disadvantages

**Table 1. Comparison table of the used Python libraries**

### 3.3.1.1 Splitting the documents library: Pikepdf

- [Pikepdf](#) library - splitting a pdf document into separate pages (one page - one new document). The name of new documents is generated as follows: the suffix "\_n" is added to the name of the main file, where n is the serial number of the page [24].

### 3.3.1.2 Text extraction libraries: Pdfplumber, PDFMiner, Pdf2image, OCR, Pytesseract

- **Library PDFplumber** is an approach to read and parse data from PDF page-by-page. It has method *open()*, which returns an instance of the `pdfplumber.PDF` class. The instance `pdfplumber.PDF` allows us to extract text, words and tables from the page or convert it into an image. *extract\_tables()* extracts tabular data from the page [12] and returns a 2-D list of the data. This library cannot save the table into a CSV file.

I will complete the PDF parsing task using the [PDFplumber](#) library[21]:

- Extract text from pdf page
  - Analyze the content for the presence of words from the first group, from the second group, from the black list
  - Add a page number to the list if it satisfies the condition [12].
- **PDFMiner** is a tool for extracting information from PDF documents. Unlike other PDF-related tools, it focuses entirely on getting and analyzing text data. [PDFMiner](#) allows one to obtain the exact location of text on a page, and other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats. It has an extensible PDF parser that can be used for other purposes than text analysis [25], [26].
- **Pdf2image** is a package that converts PDF files into image files [7].

**Poppler** is a Python binding to the poppler-cpp library. It allows us to read, render, or modify PDF documents.

There are three methods for using [Pdf2image](#):

- use the path directly to read the PDF file;
  - first, use *open* to *open* and then use *convert\_from\_bytes* to parse;
  - use temporary files for reading [28].
- **Optical Character Recognition (OCR)** is a technology that allows you to convert various documents, such as scanned documents, PDF files, or digital camera photos, into editable, searchable formats [29].

OCR involves detecting textual content in images and translating the images into encoded text that a computer can easily understand. An image containing text is scanned and analyzed to identify its characters. Once identified, the character is converted into machine-coded text.

In other words, OCR systems convert a two-dimensional image of text, which may contain machine-printed or handwritten text, from its graphical representation to machine-readable text. OCR as a process is usually divided into several sub-processes that must be performed with maximum accuracy [29]:



- Image pre-processing;
- Text localization;
- Segmentation;
- Recognition;
- Post-processing.

The sub-processes in the list above may vary, but these are approximate steps needed to achieve automatic character recognition. In OCR, its main purpose is to identify and capture all unique words using different languages from written text characters. As part of the master's thesis, the shell for Tesseract OCR - Pytesseract will be used [29].

It is not uncommon for part of the content of a document, or the entire document, to be a scanned image. In such cases, there is no text data in it, and we have to resort to OCR.

While OCR can help with some of the problems described, it also has its drawbacks:

- Long processing time;
- Running OCR on a PDF scan usually takes an order of magnitude longer than extracting text from PDF directly;
- Difficulties with non-standard characters and glyphs.

OCR has a long processing time, so we used pdfplumber whenever possible when developing our code. But pdfplumber's tools do not work with scanned documents unlike OCR, which performed great in our tests [29].

- **Pytesseract** - extract text from PDF using OCR. The library also works with the computer utility [29]. Tesseract depends on a multi-step process in which steps can be distinguished:
  - Search for words;
  - Search for lines;
  - Character classification.

Searching for words is done by enclosing text strings in small rectangles, and the strings and rectangles are analyzed for fixed pitch or proportional text. Text lines are broken into words depending on the spacing between characters. The recognition is then carried out in two steps. First, an attempt is made to recognize each word in turn. The approved words are passed to the adaptive classifier as training data. The adaptive classifier is then able to recognize the text better.

To avoid all possible drops in output accuracy, we need to ensure that the image is pre-processed properly[29].

It is necessary to understand how to recognize PDF scans to achieve the main goal, to extract information from tables to analyze them further. Therefore, it is required first to translate PDF scans to format Jpeg and then extract the text itself using [Pytesseract](#). I used Google's Pytesseract. Since it requires the installation of the Google engine on a computer, and it was working with documents that contain confidential information, all actions were carried out on an internal computer that did not have access to the network. After downloading the necessary file and writing its path in the code, I started processing files using Python [29].

### 3.3.1.3 Preprocessing (or lemmatization) libraries: SpaCy, Nltk

[Spacy](#) library is a relatively new package and is currently considered the standard in the Natural Language Processing industry. It has pre-built models that can parse text and perform various Natural Language Processing-related functionality [30].

**Lemmatization** is the process of converting a word to its base form. The difference between stemming and lemmatization is that lemmatization takes context into account and converts the word to its meaningful base form, while stemming simply removes the last few characters, often resulting in incorrect meaning and misspellings.

A feature of SpaCy is also an architecture designed to process complete documents without pre-processing in preprocessors that divide the document into phrases[30].

The main features of spaCy:

- support for about 60 languages;
  - trained models are available for different languages and applications;
  - support for pretrained vectors and word embedding;
  - high performance and accuracy;
  - a ready-made model of an on-the-job training system;
  - linguistically motivated tokenization.
- **NLTK (Natural Language Toolkit)** is the leading Python platform for creating NLP programs. As well as word processing libraries for classification, tokenization, stemming, markup, filtering, and more [31].

We will use this tool to work with stop words.

Stop words are words that are thrown out of the text before / after text processing. When we apply machine learning to texts, such words can add a lot of noise, so eliminating irrelevant words is necessary [31].

Stop words are usually understood by articles, interjections, conjunctions, etc., which do not carry a semantic load. At the same time, one must understand that there is no universal list of stop words, everything depends on the case.

[Nltk](#) has a predefined list of stop words. Before using it for the first time, we need to download it: `nltk.download("stopwords")` [31].

### 3.3.1.4 Table Extraction libraries: Tabula, Camelot

- **Tabula**

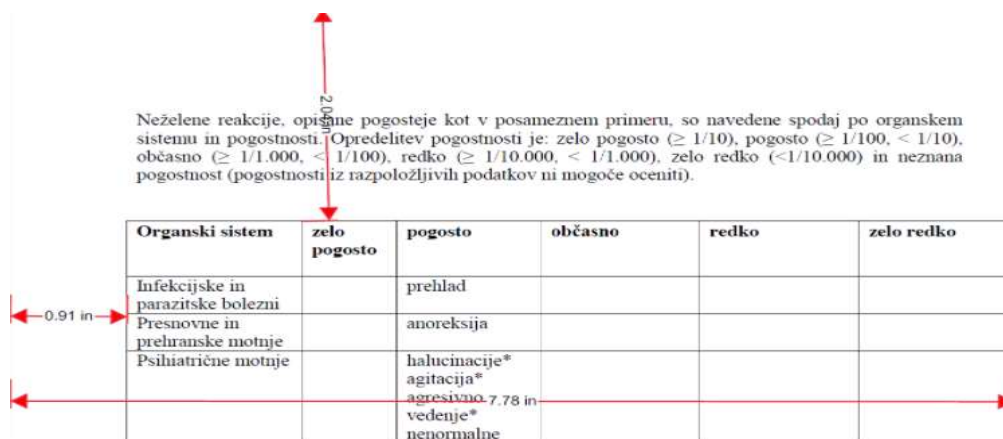
The most popular and effective way to extract tables from PDF is [Tabula](#). Such tasks are perfectly solved by the *tabula-java* tool, which is specially designed to solve such problems. To work with this tool, the *tabula-py* wrapper library was written in Python, which allows us to extract data in any form convenient for us: *pandas DataFrame*, *JSON (JavaScript Object Notation)*, *CSV*.

We can get table data as a list of DataFrames using the method `read_pdf()` or generate a CSV file with tables from PDF using `convert_into()`. All of them methods take special parameters [11]:

- *area* is a portion of the page to analyze, this implies the possibility to manually identify the area where to search for the table or tune this parameter for working with different PDF files;
- *columns* are X coordinates of column boundaries, setting this parameter allows us to work with tables that are not repetitively structured;
- *stream* is force PDF to be extracted using stream-mode extraction (if no ruling lines are separating each cell, as in a PDF of an Excel spreadsheet);
- *lattice* is force PDF to be extracted using lattice-mode extraction (if there are ruling lines separating each cell, as in a PDF of an Excel spreadsheet)
- *pages* are optional values specifying pages to extract from, this parameter gives us the possibility to work with some pages if it is needed [17].

Summing up, this method has a major advantage in that it allows us to read tables that are not repetitively structured and extract the data from each cell.

Tabula needs the area specified as the *top*, *left*, *bottom* and *right* distances. To obtain them, we can measure the distances from the *top* of the page to the beginning of the table [18].



Neželene reakcije, opisane pogostejše kot v posameznem primeru, so navedene spodaj po organskem sistemu in pogostnosti. Opredelitev pogostnosti je: zelo pogosto ( $\geq 1/10$ ), pogosto ( $\geq 1/100$ ,  $< 1/10$ ), občasno ( $\geq 1/1.000$ ,  $< 1/100$ ), redko ( $\geq 1/10.000$ ,  $< 1/1.000$ ), zelo redko ( $< 1/10.000$ ) in neznana pogostnost (pogostnosti iz razpoložljivih podatkov ni mogoče oceniti).

Organski sistem	zelo pogosto	pogosto	občasno	redko	zelo redko
Infekcijske in parazitske bolezni		prehlad			
Presnovne in prehranske motnje		anoreksija			
Psihiatrične motnje		halucinacije* agitacija* agresivno vedenje* nenormalne			

**Fig. 9 Distance measurement parameters with Tabula**

- **Camelot**

[Camelot](#) is another way of parsing tables from PDFs. This method also allows us to save results into CSV files. One useful feature of Camelot is access to a report with an accuracy metric, the page the table was found on, and the percentage of whitespace present in the table. To extract tables Camelot suggests a `read_pdf()` method, which returns a list of parsed tables [19].

Camelot is an open-source Python library that can help us easily extract tables from PDF files. It was built on top of pdfminer, another tool for extracting text from PDF documents.

It comes with many useful features such as [20]:

- Customizable - Works well in most cases, but can also be customized,
- Visual Debugging Using the Matplotlib Library,
- The output is available in several formats, including CSV, JSON, Excel, HTML, and even the Pandas framework,
- Open source - MIT license,
- Detailed documentation.

We need to install Camelot via conda, pip, or directly from the source. If installing with pip, the following dependencies must be installed: Tkinter and Ghostscript.

Generally, Camelot uses two parsing methods to extract tables [20]:

**Stream:** Look for spaces between words to identify a table.

**Hash:** Look for lines on a page to identify a table. Lattice is used by default.

In addition, we can also construct the elements found on the PDF page based on the specified type, for example, 'text', 'grid', 'contour', 'line', 'joint' etc. They are useful for debugging and playing around with different settings to get the best result [20]. They are presented in *Fig. 10 Element types found on the PDF page*. An example of the extracted data with Camelot containing two tables on one page is presented in *Fig. 11 Extracted data using the Camelot library from a page containing two tables*.

### 3.3.1.5 Table Handling library: Pandas

- **Pandas**

Pandas is a high-level Python library for data analysis. We call it high-level because it is built on top of the lower-level NumPy library, which is a big performance boost. In the Python ecosystem, pandas are the most advanced and fastest growing data science library. Two major structures of pandas are [22]:

- *Series* is an object similar to a one-dimensional array (a Python list, for example), but its distinguishing feature is the presence of associated labels, the so-called indexes, along each element from the list. This feature turns it into an associative array or dictionary in Python.
- *DataFrame* is a tabular data structure like a regular table. Every table always has rows and columns. The columns in a *DataFrame* are *Series* objects whose rows are their immediate members [22].

Pandas supports all the most popular data storage formats: csv, excel, SQL (Structured Query Language), clipboard, HTML (HyperText Markup Language) and more. In this project we are interested in saving results into csv or excel files. It could be done by using methods *to\_csv()* or *to\_excel()*. [23].

Grouping data is one of the most commonly used and powerful methods in data analysis. In pandas, the *.groupby* method is responsible for grouping. This feature can be very helpful for this work because we are looking for the ability to aggregate data extracted from PDF tables to get correct results [22].

## 4 | The main components of the algorithm

The main goal of the Thesis is to process documents of the company's Financial statements in PDF format to extract text information from the tables and save the extracted tables in the formats CSV, Excel, and to create an algorithm that is capable of automatically determining:

- Which pages of the Financial statements contain the Balance - sheet and Income Statement;
- To develop an algorithm that may be applied to other financial statements;
- To extract the previously identified into an editable file, according to a pre-defined structure, preferably in CSV, Excel formats.

### 4.1 Development workflow

This Section will describe the complete workflow development. From the start to the end to achieve our main goal and receive the best final result.



Fig. 12 Development workflow

- **Start:**

This is where almost any meaningful activity begins. Good goal setting requires a deep understanding of all aspects of the situation in which the research is being conducted and a clear definition of the result that we want to get. To do this, it is necessary to study the problem to be solved by the study.

- **Input: PDF document**

PDFs are a good source of data, and most organization only release their data in PDFs. The first step is to collect data. In our case, this is the Financial Statement in PDF format provided by the Banco de Portugal during the internship. It should be noted that the financial statements in PDF format do not have a unified form, and the quality of the documents provided is fundamentally different.

- **Split document into the PDF pages:**

First, we open a PDF file, create a reader object, and iterate through all the pages using the reader object's *tqdm enumerate(pdf.pages)* method.

```
pdf = Pdf.open(filename)
for n, page in tqdm enumerate(pdf.pages):
```

The next step is to create a unique filename, using the original filename plus an index.

```
output_filename = f"{name}/{name}-page-{n}.pdf"
```

Finally, we save the files in PDF format.

```
new_pdf = Pdf.new()
new_pdf.pages.append(page)
new_pdf.save(output_filename)
```

- **For each page:**

Further development of the algorithm will be applied to each page separately after the process of splitting the document into PDF pages.

At this stage of development, I use the *listdir* function to work with each page. *Listdir* returns a list containing the names of the entries in the directory specified by path. The list is in random order and does not include the special entries '.' and '..' even if they are present in the directory. If a file is removed from a directory or is added to it during a call to this function, it is not specified whether the name of this file will be included.

```
print(name)
files = []
for file in tqdm(os.listdir(name)):
```

To skip system files that may automatically appear in the folder:

```
if file[0] == ".":  
    continue
```

- **Extract text:**

***Where are we able to extract the text?***

To read text from a PDF file and extract it, we used the *pdfplumber* library. It is worth noting that two libraries for reading text were tested (*pdfminer* and *pdfplumber*). The *pdfplumber* library recognized the text with better quality.

```
text = ""  
with pdfplumber.open(f"{name}/{file}") as pdf:  
    first_page = pdf.pages[0]  
    text = first_page.extract_text()
```

If the plumber library did not find the text on the page, then we will read it using AI (OCR).

```
image = get_img_from_pdf(f"{name}/{file}")  
text = ocr_core(image)
```

After extracting the text, we move on to the Preprocessing stage, which will be discussed in more detail in Chapter 4.2.1

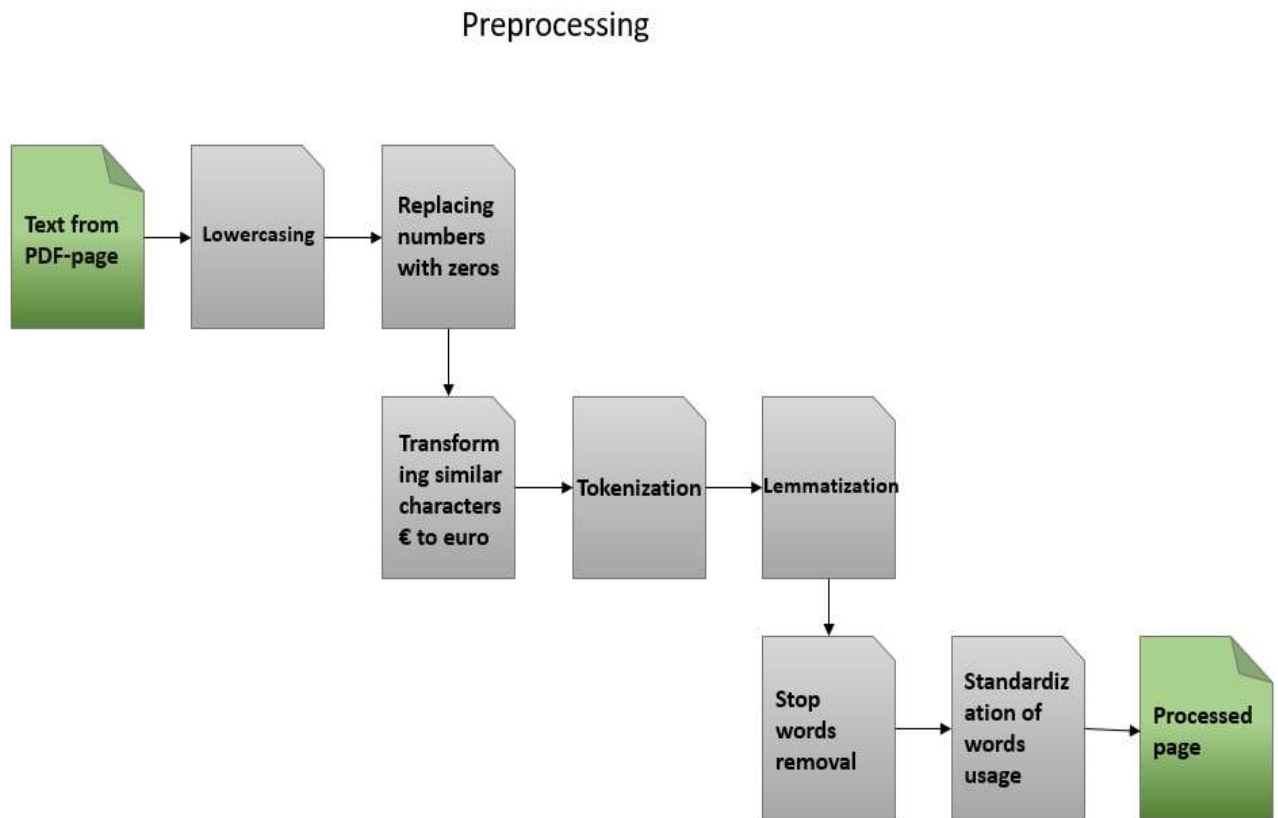
## **4.1.1 Text Preprocessing**

The preprocessing module presented in this section has as its main goal the cleaning up of the document's text being processed under several operations to remove, or at least, reduce this type of informality characteristic making the work of the main goal easier.

The goal of data preprocessing is to produce “clean text” machines can analyze error-free.

Clean text is human language rearranged into a format that machine models can understand. Text cleaning can be performed using simple Python code that eliminates stopwords and simplifies complex words to their root form.





**Fig. 13 Text Preprocessing**

Below, we enumerate and describe the different preprocessing methods implemented:

- **Text from a PDF- page:** This module works for each split PDF page. Very often, the words that appear in documents/pages have many structural variants. So before the information retrieval from the documents, the data preprocessing techniques are applied to the target data set to reduce the size of the data set, which will increase the effectiveness of the Information Retriever System.

- **Lowercasing:** This operation is responsible for converting upper case characters to lower representation. The advantages provided by this operation are centred on the analysis of words written in different ways.

```
text = text_splited.lower()
```

- **Replacing all numbers with zeros:** we will replace all numbers with 0 in order to further simplify data processing:

```
for i in range(1, 10):
```

```
text = text.replace(str(i), "0")
```

and

```
for ind in range(len(words)):
```

```
if "0" in words[ind]:
```

```
words[ind] = "00"
```

- **Transforming similar characters € to euro:** we will simplify Euro characters:

```
text = text.replace("€", "euro")
```

- **Tokenization:** This operation is a step that breaks long lines of text into smaller chunks or tokens. Large chunks of text can be turned into sentences and sentences into words. Further processing is usually done after a piece of text has been properly composed. Tokenization is also known as text segmentation or lexical analysis.

Segmentation is sometimes used to break up of a large chunk of text into pieces larger than words (such as paragraphs or sentences), while tokenization is for the process of breaking up, purely into words.

We will split the text into words and remove all punctuation for our task. To do this, use the function :

```
words = tokenizer.tokenize(text)
```

Join the words into one line separated by spaces: `document = ' '.join(words)`

- **Lemmatization:** This operation is the process of converting a word to its base form. The difference between stemming and lemmatization is that lemmatization takes context into account and converts the word to its meaningful base, form while stemming simply removes the last few characters, often resulting in incorrect meaning and misspellings. To normalize Portuguese words, use the following function [32]:

```
nlp = spacy.load('pt_core_news_sm'). tokenizer = RegexpTokenizer(r'\w+')
stops = set(stopwords.words('portuguese')).
```

Sent the string to parse the Portuguese language (then we can get lemmas):

```
text = nlp(document).
```

- **Stop words removal:** This operation removes the most common words in the language in the analysis. We remove the stop words and store the lemmas in a list:

```
words = [word.lemma_for word in text if (word.text not in stops) and (word.text not in ['_'])]
```

- **Standardization of words usage:**

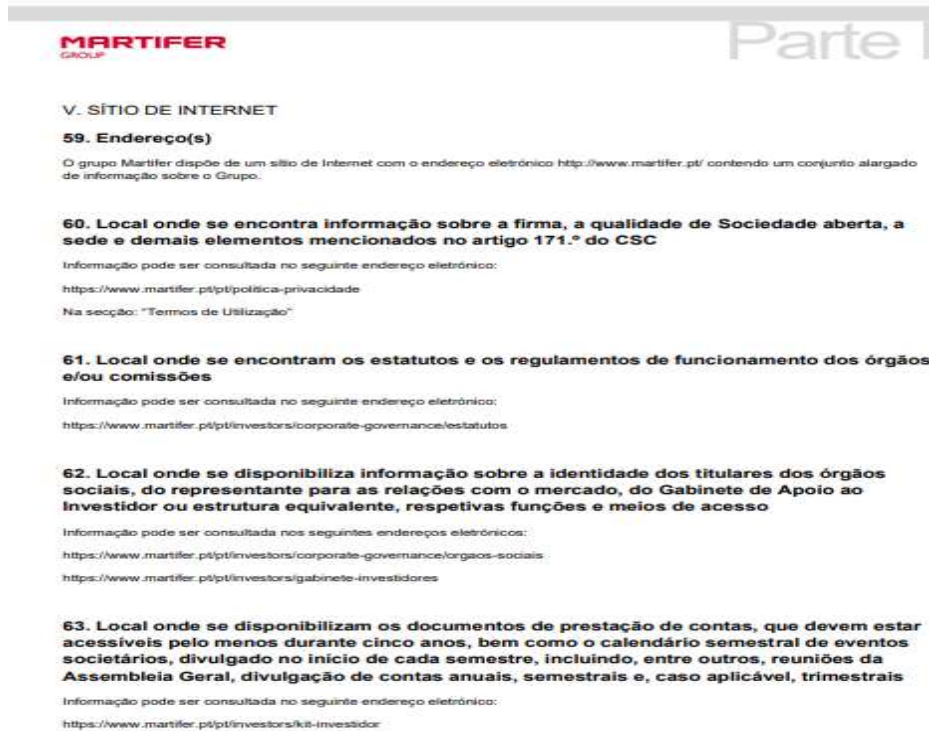
is a process that we do not just remove endings of the words, but bring the word to its initial form. If we apply standardization of words like: "activo", "ativo" or "activos"; or "passivos" and "passivo" we will convert different forms to one.

```
if words[ind] == "activo" or words[ind] == 'ativo' or words[ind] == "activos":
    words[ind] = "ativos"
if words[ind] == "passivos":
    words[ind] = "passivo"
set_words = set(words)
```

## • Output: processed page

Any text cleaning approach is about attention to detail and boiling our data down to only its most crucial bits without losing its context – and that's a hard balance to strike.

This is how we turned the complex, multi-element text from a PDF page into a series of ordered words designed to find up-to-date information in the future and extract data relevant to our goal. The graph shows PDF page before processing: *Fig. 14 PDF page before processing*. Fig.15 demonstrates our processed text.



**Fig. 14 PDF page before processing**

Now our processed text looks like this.

```
[ 'parte', 'i', 'v', 'sítio', 'Internet', '00', 'endereço', 's', 'grupo', 'martifer', 'dispor', 'sítio', 'Internet', 'endereço', 'eletrónico', 'Http', 'Www', 'martifer', 'pt', 'conter', 'conjunto', 'alargar', 'informação', 'sobre', 'grupo', '00', 'local', 'onde', 'encontrar', 'informação', 'sobre', 'firma', 'qualidade', 'sociedade', 'abrir', 'se de', 'demais', 'elemento', 'mencionar', 'artigo', '00', 'º', 'csc', 'informação', 'poder', 'consultar', 'seguinte', 'endereço', 'eletrónico', 'https', 'Www', 'martifer', 'pt', 'pt', 'politico', 'privacidade', 'Secção', 'termo', 'utilização', '00', 'local', 'onde', 'encontrar', 'estatuto', 'regulamento', 'funcionamento', 'órgão', 'comissão', 'informação', 'poder', 'consultar', 'seguinte', 'endereço', 'eletrónico', 'https', 'Www', 'martifer', 'pt', 'pt', 'investors', 'Corporate', 'governancer', 'estatuto', '00', 'local', 'onde', 'disponibilizar', 'informação', 'sobre', 'identidade', 'titular', 'órgão', 'social', 'representante', 'relação', 'mercado', 'gabinete', 'apoio', 'investidor', 'estrutura', 'equivalente', 'respetivo', 'função', 'meio', 'acesso', 'informação', 'poder', 'consultar', 'seguinte', 'endereço', 'eletrónico', 'Https', 'Www', 'martifer', 'pt', 'pt', 'investors', 'Corporate', 'governance', 'Orgaos', 'social', 'https', 'Www', 'martifer', 'pt', 'pt', 'investors', 'Gabinete', 'investidor', '00', 'local', 'onde', 'disponibilizar', 'documento', 'prestação', 'conta', 'dever', 'acessível', 'menos', 'durante', 'cinco', 'ano', 'bem', 'calendário', 'semestral', 'evento', 'societário', 'divulgar', 'início', 'cada', 'semestre', 'incluir', 'outro', 'reunião', 'Assembleia', 'Geral', 'divulgação', 'conta', 'anual', 'semestral', 'caso', 'aplicável', 'trimestral', 'informação', 'poder', 'consultar', 'seguinte', 'endereço', 'eletrónico', 'https', 'Www', 'martifer', 'pt', 'pt', 'investors', 'kit', 'investidor', 'relatório', 'governo', 'societário', '00']
```

**Fig. 15 Processed text**

Although there are more text preprocessing techniques, in this dissertation we only used the ones previously described since each of them is associated, it helps to clean up the data to further develop the code and get the maximum result for the intended purpose.

• **Word Inclusion and Exclusion Lists: Are there words from the "black\_list" in the text?**

We also can define a list of indexed words. This list will be used to search for individual words and classify source documents based on the relevance of the words that appear per goal. In our case, this list is called "form\_sure". We defined "black\_list" as words that will be excluded from indexing. We included in the "black\_list" list those words that are used very often, carry little meaningful information and prevent the extraction of the correct tables from the company's Financial statements, in our case, are: Balance-sheet and Income statements.

```
if len(set_words & set(black_list)) != 0:  
    continue
```

• **Are there words in the text from "For\_sure"?**

We also can define a list of indexed words. This list will be used to search for individual words and classify source documents based on the relevance of the words that appear per the goal. In our case, this list is called "form\_sure".

```
if len(set_words & set(for_sure)) < 1:  
    continue
```

• **Count in the text words from Group 1.**

We will count the words in the text from Group 1 that satisfy our search conditions for words in the text from "For\_sure". How many group words were found in the text (each word is counted once).

```
count_group1 = 0  
for word in group1:  
    if word in words:  
        count_group1 += 1
```

• **Count in the text words from Group 2.**

Similarly, we will count the words in the text from Group 2 that satisfy our search conditions for words in the text from "For\_sure".

```
count_group2 = 0  
for word in group2:  
    if word in words:  
        count_group2 += 1
```

After counting the words relevant to our main goal by groups, we move on to the Scoring Stage, which will be discussed in more detail in chapter 4.1.2.

## 4.1.2 Scoring of the words

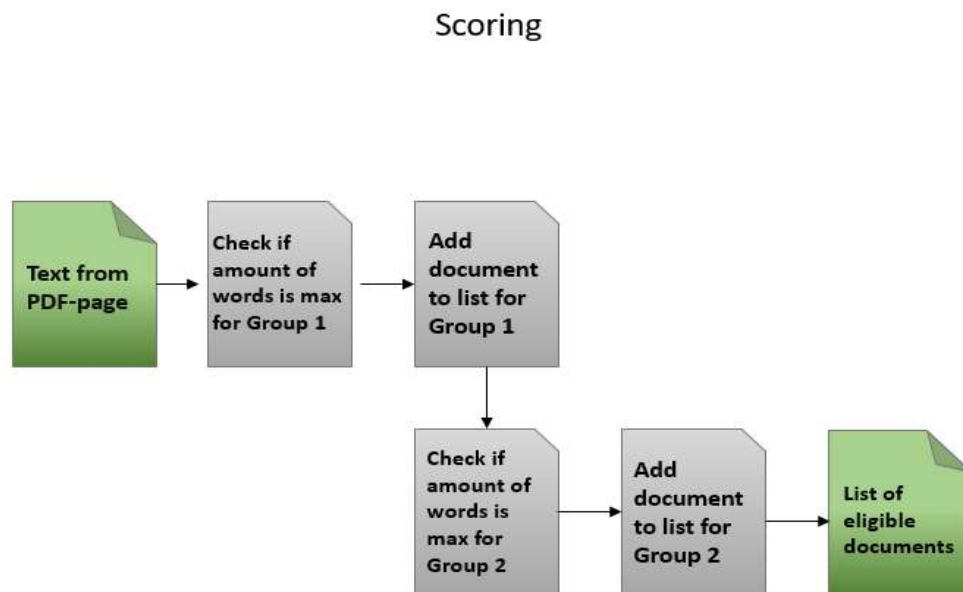


Fig. 16 Scoring of the words

### Scoring of the words:

This is a popular and simple feature extraction technique used when working with text. It describes the occurrences of each word in the text.

#### • Text from a PDF – page:

This module is applied to each page separately after splitting the document into PDF pages. For each PDF page, we apply this model (scoring) of classifying the information contained in the document into different groups if the characteristic that separates these groups is unknown. However, other factors associated with the characteristic of interest to us are known.

### Bag of words:

Machine learning is often used to process text documents using a feature text model, in which features are defined for each document separately. Signs can be various informational characteristics of the text: both linguistic and statistical and structural: for example, the frequency of certain words (or their categories) in a document, the frequency of use of special characters, the ratio of parts of speech of words, the presence of certain syntactic constructions or sections of text, the date of creation, etc.

Varieties of the attribute model are the Bag of words model, in which the text is characterized by a set of its significant words (usually, these are all significant words, more precisely, their lemmas).

- **To check if the amount of words is a maximum for each Group and to add the document to the List for each Group:**

In our case, 2 Groups of words were created to extract the 2 required Tables from the Financial Statements. Group 1 contains words to extract from the Financial Statements Balance-sheet, Group 2 contains a set of words to extract the Income statement. How many group words were found in the text (each word is counted once).

If the number of words from the first group is the largest on the specified page, then we write down the document's name to list Group 1. Moreover, remember how many maximum words coincided with the group at the moment:

```
if count_group1 > documents1['count_unique']:
documents1['names'] = [file]
documents1['count_unique'] = count_group1
```

Similarly, we check if the amount of words is a maximum for Group 2 and add the document to the List for Group 2.

- **Adding the file to the list of eligible documents:**

If the number of words from the group is equal to the maximum of those found earlier, then we write the name of this file to the list of eligible documents:

```
elif count_group1 == documents1['count_unique']:
documents1['names'].append(file)
```

As a result of applying the word scoring technique, for a document in PDF format, we received a list of documents, broken down by Groups and by the largest number of occurrences of words in each Group, which will allow us to continue developing the code and extract the necessary tables from the Financial Statements with the greatest accuracy.

- **Convert the document to xlsx tables:**

The structure of all written code algorithms is built as follows: all found pages that are divided into several xlsx files are merged into one xlsx file. A folder is automatically created into which uploaded the correct xlsx files, merged, and saved.

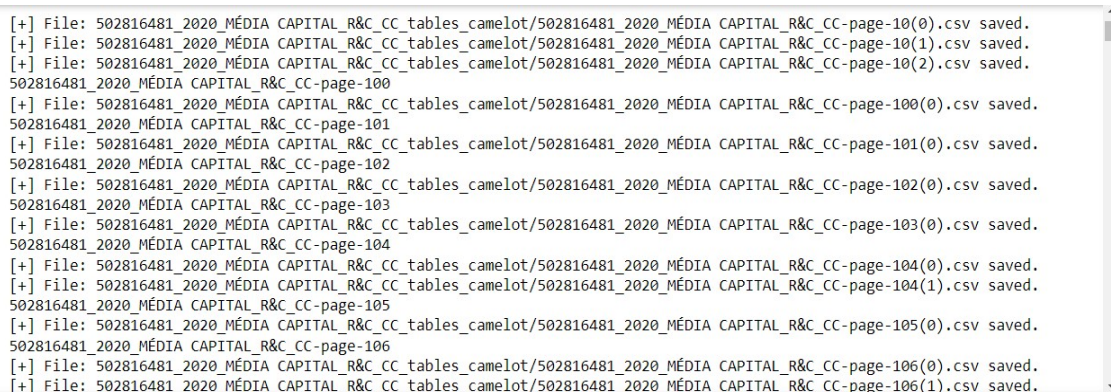
- **Output: xlsx documents**

As a result of applying our TM workflow for a document in PDF format, we have extracted the required Balance-sheet and Income statement from Financial Statements with the greatest accuracy. Furthermore, they imported the received data into xlsx format.

## 5 | Experiments and results

### 5.1 Testing the Camelot library

At the beginning of code development, in order to obtain relevant information and achieve the main goal, the Camelot library was tested. The testing results showed that the files uploaded to the csv format are uploaded in large quantities, including some pages being split into several files.



```
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-10(0).csv saved.
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-10(1).csv saved.
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-10(2).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-100
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-100(0).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-101
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-101(0).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-102
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-102(0).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-103
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-103(0).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-104
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-104(0).csv saved.
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-104(1).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-105
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-105(0).csv saved.
502816481_2020_MÉDIA CAPITAL_R&C_CC-page-106
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-106(0).csv saved.
[+] File: 502816481_2020_MÉDIA CAPITAL_R&C_CC_tables_camelot/502816481_2020_MÉDIA CAPITAL_R&C_CC-page-106(1).csv saved.
```

**Fig. 17** Extracted tables in CSV format during testing of the Camelot library

The ability to recognize the contained text is very weak. The received data in csv format turned out to be almost unreadable. In this regard, it was decided to continue further work on the main goal and further development with the help of the Tabula library, which contains more functions and tools for working with text and tables. Name of the code: **Camelot\_testing\_library**.



502816481\_2020\_MÉDIA CAPITAL\_R&C\_CC-page-10(0) - Excel (A Ativação do Produto Falhou)

Ficheiro Base Inserir Esquema de Página Fórmulas Dados Rever Ver PDFelement Diga o que pretende fazer...

Cortar  
 Copiar  
 Pincel de Formatação

Calibri 11 A A  
 N I S A  
 Tipo de Letra

Moldar Texto  
 Unir e Centrar  
 Alinhamento

Geral  
 \$ % 000   
 Número

Formatação Condicional  
 Formatar como Tabela  
 Estilos de Célula  
 Estilos

Inserir Eliminar  
 Células

A1 m iillhhaarreess ddee â,-â,-,"2 00 22 00","2 00 11 99","Vaarr %%", "2 SS 22 00 22 00","2 SS 22 00 11 99","Vaarr %%"

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	m iillhhaarreess ddee â,-â,-,"2 00 22 00","2 00 11 99","Vaarr %%", "2 SS 22 00 22 00","2 SS 22 00 11 99","Vaarr %%"															
2	Recebimentos,"181 632","194 294","(7%)","104 183","88 771","17%"															
3	Pagamentos,"(164 592)","(181 370)","9%","(86 355)","(91 739)","6%"															
4	F lluxx ooss ddaass aattiivviiddaaddeess ooppeerraacciioonnaaiiss ((11)),"1 77 00 44 11","1 22 99 22 44","3 22 %%", "1 77 88 22 88","(22 99 66 77)"),"n..aa.."															
5	Recebimentos,"212","2 978","(93%)","96","1 397","(93%"															
6	Pagamentos,"(3 156)","(5 313)","41%","(849)","(2 311)","63%"															
7	F lluxx ooss ddaass aattiivviiddaaddeess ddee iinnvveessttiimm eennttoo ((22)),"(22 99 44 44)","(22 33 33 55)","(22 66 %%)","(77 55 44)","(99 11 55)"),"1 88 %%"															
8	Recebimentos,"14 904","124 791","(88%)","(46 232)","86 054","n.a."															
9	Pagamentos,"(8 327)","(132 794)","94%","44 195","(79 462)","n.a."															
10	F lluxx ooss ddaass aattiivviiddaaddeess ddee ffiinnaancciaamm eennttoo ((33)),"6 55 77 77","(88 00 00 44)","n..aa..","(22 00 33 77)"),"6 55 99 22","n..aa.."															
11	Caixa e equivalentes no inÃ-cio do perÃ-odo,"2 966","382","676%","8 603","256",">999%"															
12	Var. caixa e seus equivalentes (4)=(1)+(2)+(3),"20 674","2 585","700%","15 038","2 711","455%"															
13	Efeito das diferenÃas de cÃmbios,"(0)","(0)","3%","(1)","(0)","(164%"															
14	C aaiix aa ee eeqquiiivvaalleenntteess nnoo ffiinnaall ddo ppeerrÃ-Ã-ooddo,"2 33 66 44 00","2 99 66 66","6 99 77 %%", "2 33 66 44 00","2 99 66 66","6 99 77 %%"															

**Fig. 18 The result of testing the Camelot library**

## 5.2 Extracting the tables using the markup of the image

For the best recognition, the location of tables on a single page in PDF format, a combination of Camelot and Python libraries I used to develop this code. Name of the code **area\_Camelot\_Tabula**.

To get started, we should install the required libraries for our work:  
tabula, matplotlib.pyplot as plt, camelot, numpy.

We should specify the folder's name of the folder and specific page in PDF format. Our example is the page with 2 tables of the CTT Company.



Em 2020 o montante de 7,0 MC de itens específicos diz respeito a: (i) reestruturações empresariais, que se situaram em 3,3 MC (-8,6 MC) face ao período homólogo, (ii) projetos estratégicos, que registaram 0,9 MC (-4,0 MC) essencialmente em estudos de apoio à renegociação do novo contrato de concessão, e (iii) outros rendimentos e gastos, que registaram 2,8 MC (+1,4 MC), dos quais se destaca a penalização em preços pela ANACOM (+1,0 MC) imposta pelo incumprimento dos Indicadores de Qualidade de Serviço de 2019, os gastos relacionados com a pandemia de COVID-19, nomeadamente em equipamentos de proteção individual, nebulizações, medição de temperatura, reforço das limpezas (+1,1 MC) e o pagamento de um prémio extraordinário aos colaboradores que durante o período de confinamento estiveram sempre na linha da frente, com enorme profissionalismo e entrega total (+0,5 MC).

#### EBIT e Resultado Líquido

O EBIT no 4T20 cresce 32,1% (+4,2 MC) situando-se em 34,5 MC no ano de 2020, 12,8 MC abaixo (-27,0%) do registado em 2019, fortemente penalizado pelo decréscimo do EBITDA (-11,0 MC), pelo crescimento das imparidades e provisões (+6,7 MC) e das depreciações e amortizações (+7,9 MC), que não compensaram o decréscimo verificado nos itens específicos (-11,2 MC).

#### EBIT por área de negócio

	2019	2020	Valor	Milhões €
			Δ	
<b>EBIT</b>	<b>47,3</b>	<b>34,5</b>	<b>-12,8</b>	<b>-27,0%</b>
Correio e Outros	42,9	9,9	-32,9	-76,8%
Correio	98,6	66,4	-32,2	-32,6%
Estrutura central	-55,7	-56,5	-0,7	-1,3%
Expresso e Encomendas	-12,1	-0,5	11,6	95,9%
Banco CTT	-4,9	4,6	9,5	193,1%
Serviços Financeiros e Retalho	21,5	20,5	-1,0	-4,5%

Os resultados financeiros consolidados atingiram -11,4 MC, refletindo uma melhoria de 0,4 MC (3,2%) face a 2019.

#### Resultados financeiros

	2019	2020	Valor	Milhões €
			Δ	
<b>Resultados financeiros</b>	<b>-11,8</b>	<b>-11,4</b>	<b>0,4</b>	<b>3,2%</b>
Rendimentos financeiros líquidos	-10,4	-9,6	0,7	6,9%
Gastos e perdas financeiros	-10,4	-9,7	0,8	7,3%
Rendimentos financeiros	0,1	0,02	-0,04	-68,4%
Ganhos/perdas em subsidiárias, associadas e empreendimentos conjuntos	-1,4	-1,7	-0,3	-24,3%

Os gastos e perdas financeiros incorridos ascenderam a 9,7 MC, incorporando maioritariamente os gastos financeiros com benefícios pós-emprego e de longo prazo aos empregados de 4,5 MC, juros suportados dos passivos de locação no âmbito da aplicação da IFRS 16 no valor de 3,3 MC e juros de financiamentos bancários no montante de 1,7 MC.

Os CTT obtiveram em 2020 um **resultado líquido** consolidado atribuível a detentores de capital do Grupo CTT de 16,7 MC, 12,5 MC (-42,9%) abaixo do verificado no ano anterior, fortemente impactado pela evolução negativa do EBIT (-12,8 MC).

Fig. 19 The page with 2 tables of the CTT Company in PDF format

For visualization, both of the above libraries were tested, Camelot reflected the image better.

```
tables = camelot.read_pdf(f"{folder_name}/{page}", flavor='stream')
ax = camelot.plot(tables[2], kind='contour')
```

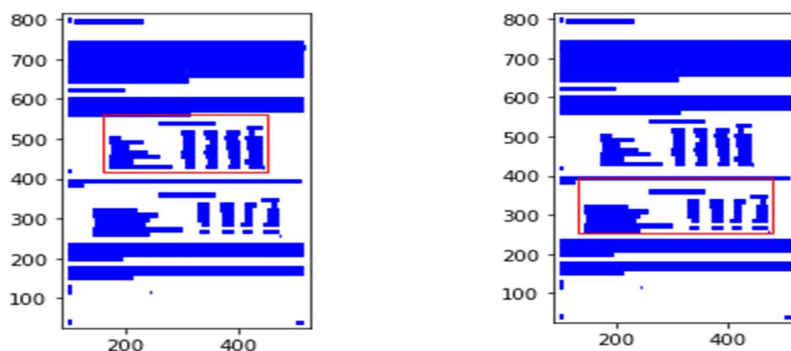


Fig. 20 The markup of the image using the Camelot library

To extract tables using the Tabula library, use the markup of the image obtained previously, for that, we enter the variable Area.

```
tables = read_pdf(f"{folder_name}/{page}", pages="all",
area = [[300, 100, 420, 800],
[480, 100, 600, 800]]
) # [top,left,bottom,right]
```

	A	B	C	D	E	F
1						Milhões €
2	0		2019	2020	Valor	Δ
3	1	EBIT	47,3	34,5	-12,8	-27,0%
4	2	Correo e	42,9	9,9	-32,9	-76,8%
5	3	Correo	98,6	66,4	-32,2	-32,6%
6	4	Estrutura e	55,7	56,5	-0,7	-1,3%
7	5	Expresso e	12,1	0,5	11,6	95,9%
8	6	Banco CTT	4,9	4,6	9,5	193,1%
9	7	Serviços Fi	21,5	20,5	-1,0	-4,5%
10						

	A	B	C	D	E	F
1						Milhões €
2	0		2019	2020	Valor	Δ
3	1	Resultado	-11,8	-11,4	0,4	3,2%
4	2	Rendimen	-10,4	-9,6	0,7	6,9%
5	3	Gastos e p	-10,4	-9,7	0,8	7,3%
6	4	Rendimen	0,1	0,02	-0,04	-68,4%
7	5	Ganhos/perdas em subsidiárias, associadas e				
8	6		-1,4	-1,7	-0,3	-24,3%
9	7	empreendimentos conjuntos				
10						

**Fig. 21 The extracted tables using the Tabula library**

The quality of table recognition with the help of combinations of libraries is quite high. The only negative is that this work is labour-intensive and time-consuming.

To achieve the main goal, I will develop a more automated algorithm that can save time and improve results and performance.

To achieve the desired and the best result for the intended goal, 5 main versions of the code were developed, which are presented and detailed in this section.

Workflow of the code development

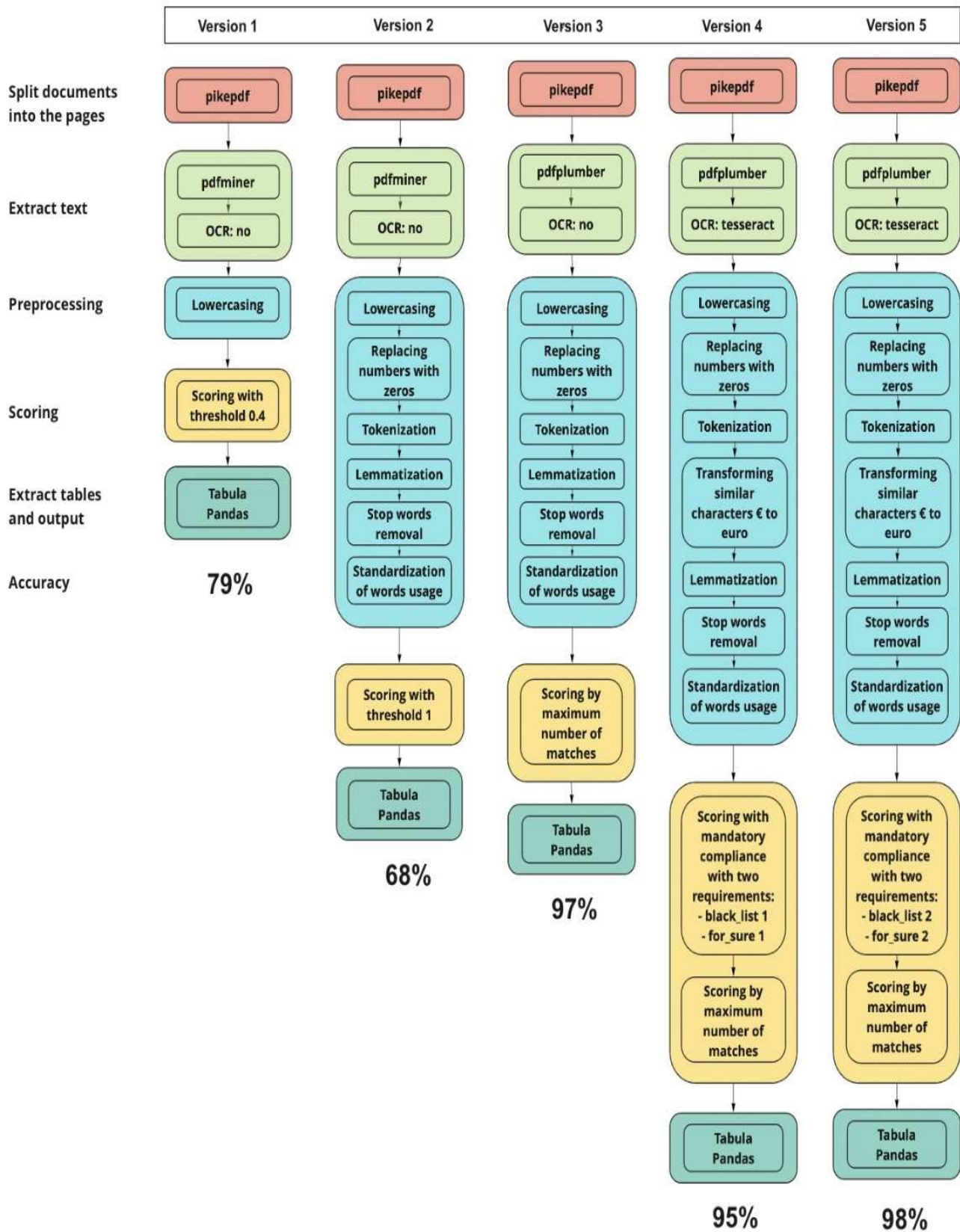


Fig. 22 Workflow of the code development

## 5.3 Version 1

**Split documents into pages:** we used library pikepdf - splitting a pdf document into separate pages (one page - one new document). The name of new documents is generated as follows: the suffix "\_n" is added to the name of the main file, where n is the serial number of the page [21].

### **Extract text:**

At the first stage, to start working on writing the code, the most frequently occurring words in Financial statements were provided by the Bank. The words provided were divided by me into 2 main groups, which represent two main reports:

Group 1 - Balance-sheet and Group 2 - Income statement.

The more relevant words that were selected to search for the Group 1: "ativo", "activo", "corrent", "caixa", "capital", "passivo", "fornecedor", "client", "financiamento".

The more relevant words that were selected to search for the Group 2: "consolidad", "gasto", "rendimento", "resultado", "outro", "imposto", "perda", "financeiro", "operacional", "pessoal".

The algorithm Version 1 tested 17 documents.

In Version 1, we used the most popular and effective way to extract text and tables from PDFs is Tabula. It is a simple Python wrapper for tabula-java that can read tables from PDFs and convert them to Pandas DataFrames.

To extract the text, we used library pdfminer is, an approach to read and parse data from PDF page-by-page. The instance pdfminer allows us to extract text, words, and tables from the page or convert it into an image.

**Preprocessing:** In Version 1 we only used *Lowercasing*. This operation is responsible for converting upper case characters to lower representation. The advantages provided by this operation are centred on the analysis of words written in different ways.

*text = text.lower()*

**Scoring:** Scoring with threshold: In Version 1 the threshold was calculated with a parameter  $\geq 0.4$ . To do this, the number of words found in the group is divided by the number of words in the group.

*if count\_group1 / len(group1)  $\geq 0.4$*

A similar calculation was made with a threshold of 0.4 with Group 2.

**Extract tables and output:** Library pdfminer cannot save the table into a CSV file. In this regard, we used Tabula / Pandas to convert data to xlsx format. We use Tabula to retrieve tables, display, and save the results. Pandas contain tools for storing tables.

It is worth noting that a similar approach for extracting tables, displaying and saving results, and further storing tables was applied in **all five Versions** developed since it is more efficient.

The code allows viewing the results in a DataFrame with division into groups. The total number of pages found for each group is displayed by searching for similar words on the page. The last column shows the number of correct pages found.

```
In [10]: df = pd.DataFrame.from_records(results)
df
```

```
Out[10]:
```

	filename	keywords_g1	keywords_g2	right
0	2020_500077797_CorticeiraAmorim_R&C_CC	7	7	1
1	2020_501669477_Ibersol_R&C_CC	8	12	2
2	2020_502028351_SONAECON_R&C_CC	10	10	2
3	2020_502293225_Cofina_R&C_CC	12	9	1
4	2020_502593130_Semapa_R&C_CC	4	12	2
5	2020_503541320_GLINTT_R&C_CC	4	4	2
6	2020_504453513_NOS_R&C_CC	12	16	2
7	2020_508548527_Ramada_R&C_CC	11	9	1
8	500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PA...	3	3	2
9	500166587_2020_LISGRÁFICA_R&C_CC	2	3	2
10	502816481_2020_MÉDIA CAPITAL_R&C_CC	8	7	1
11	2020_500077568_CTT_R&C_CC	7	5	1
12	500101221_2020_ESTORIL_R&C_CC	3	4	2
13	503264032_2020_REN_R&C_CC	6	14	2
14	2020_500978654_VAA_R&C_CC	2	2	0
15	2020_505127261_Martifer_R&C_CC	14	12	2
16	504882066_2020_BENFICA_R&C_CC	2	3	2

Fig. 23 The DataFrame with divided into groups with the total number of pages found for each group

The structure of all written code algorithms is built as follows: all found pages that are divided into several xlsx files are merged into one xls file.

Using the example of NOS Financial Statements, we can see that the algorithm extracted 2 tables located on page 178 and saved both tables in a single xlsx file with good recognition quality.



**DEMONSTRAÇÃO DA POSIÇÃO FINANCEIRA CONSOLIDADA**  
EM 31 DE DEZEMBRO DE 2019 E 2020  
(Montantes expressos em milhares de euros)

	NOTAS	31-12-2019	31-12-2020
<b>ATIVO</b>			
<b>ATIVO NÃO CORRENTE</b>			
Ativos não correntes	8	1.034.813	991.413
Propriedades de investimento		633	637
Ativos intangíveis	9	1.014.064	1.041.067
Encargos de contratos com clientes	10	163.101	162.123
Direitos de uso	11	218.283	280.097
Investimentos em empreendimentos conjuntos e associadas	12	18.244	10.897
Contas a receber - outros	13	4.064	7.504
Impostos a recuperar	14	149	149
Outros ativos financeiros não correntes	15	439	579
Ativos por impostos diferidos		80.428	87.782
<b>TOTAL DO ATIVO NÃO CORRENTE</b>		<b>2.334.342</b>	<b>2.507.448</b>
<b>ATIVO CORRENTE</b>			
Inventários	16	34.081	43.628
Contas a receber - clientes	17	341.712	290.452
Contas a receber - outros	18	48.059	41.602
Impostos a recuperar	19	28.126	28.610
Impostos a recuperar	14	4.631	2.894
Custos diferidos	19	43.954	34.054
Ativos não correntes detidos para venda		450	450
Caixa e equivalentes de caixa	21	12.819	153.285
<b>TOTAL DO ATIVO CORRENTE</b>		<b>533.884</b>	<b>685.179</b>
<b>TOTAL DO ATIVO</b>		<b>3.088.176</b>	<b>3.172.643</b>
<b>CAPITAL PRÓPRIO</b>			
Capital social	22.1	5.152	5.152
Prémios de emissão de ações	22.2	854.219	854.219
Ações próprias	22.3	(14.650)	(14.859)
Reserva Legal	22.4	1.030	1.030
Outras reservas e resultados acumulados	22.4	16.041	12.007
Resultado líquido		143.494	92.000
<b>CAPITAL PRÓPRIO EXCLUINDO INTERESSES QUE NÃO CONTROLAM</b>		<b>1.005.281</b>	<b>949.549</b>
Interesses que não controlam		7.542	5.885
<b>TOTAL DO CAPITAL PRÓPRIO</b>		<b>1.012.823</b>	<b>955.434</b>
<b>PASSIVO</b>			
<b>PASSIVO NÃO CORRENTE</b>			
Empréstimos obtidos	24	1.216.847	1.363.514
Provisões	25	94.059	73.345
Contas a pagar - outros	29	3.855	40.050
Acrescimos de custos	26	467	505
Provisões diferidas	27	5.123	4.729
Instrumentos financeiros derivados	28	345	451
Passivos por impostos diferidos	15	11.626	5.025
<b>TOTAL DO PASSIVO NÃO CORRENTE</b>		<b>1.333.343</b>	<b>1.487.833</b>
<b>PASSIVO CORRENTE</b>			
Empréstimos obtidos	24	143.281	127.125
Contas a pagar - fornecedores	28	229.499	252.607
Contas a pagar - outros	29	33.835	47.438
Impostos a pagar	14	68.202	51.981
Acrescimos de custos	26	203.726	175.860
Provisões diferidas	27	31.634	33.228
Instrumentos financeiros derivados	28	135	944
<b>TOTAL DO PASSIVO CORRENTE</b>		<b>743.551</b>	<b>738.546</b>
<b>TOTAL DO PASSIVO</b>		<b>2.076.894</b>	<b>2.226.409</b>
<b>TOTAL DO CAPITAL PRÓPRIO E PASSIVO</b>		<b>3.088.176</b>	<b>3.172.643</b>

O anexo faz parte integrante da demonstração da posição financeira consolidada a 31 de dezembro de 2020  
O Contabilista Certificado O Conselho de Administração

**DEMONSTRAÇÃO CONSOLIDADA DOS RESULTADOS POR NATUREZA**  
DOS EXERCÍCIOS FINDOS EM 31 DE DEZEMBRO DE 2019 E 2020  
(Montantes expressos em milhares de euros)

	NOTAS	4º TRIM 19 REPORTADO	12M 19 REPORTADO	4º TRIM 19 REAPRESENTO*	12M 19 REAPRESENTO*	4º TRIM 20	12M 20
<b>RECEITAS</b>							
Prestação de serviços		380.227	1.455.935	332.564	1.345.108	318.411	1.262.950
Outras receitas		28.043	89.141	28.043	89.141	29.210	84.309
<b>TOTAL</b>	<b>30</b>	<b>414.048</b>	<b>1.599.230</b>	<b>366.387</b>	<b>1.458.404</b>	<b>348.309</b>	<b>1.347.886</b>
<b>CUSTOS, PERDAS E GANHOS</b>							
Custos com o pessoal	31	22.793	85.176	22.715	84.830	21.876	85.331
Custos diretos	32	143.482	524.058	99.475	368.889	102.860	348.775
Custo das mercadorias vendidas	33	21.983	64.228	21.982	64.228	24.832	74.312
Marketing e publicidade	34	15.652	37.216	15.652	37.216	19.845	24.024
Serviços de suporte	34	22.319	82.335	22.441	82.237	25.055	86.281
Fornecimentos e serviços externos	34	29.843	112.063	27.952	112.420	27.213	109.542
Outros custos / (ganhos) operacionais		123	516	123	516	331	719
Impostos indiretos		8.354	32.844	8.406	32.844	8.355	32.747
Provisões e ajustamentos	35	11.654	18.934	11.739	18.983	4.635	11.493
Depreciações, amortizações e perdas por imparidade	8, 9, 10, 11 e 37	123.344	421.318	123.341	421.313	104.597	409.842
Custos de reestruturação	38	713	7.732	713	7.732	1.033	5.523
Perdas / (ganhos) com a alienação de ativos, líquidos		98	(647)	97	(547)	1.136	2902
Outros custos / (ganhos) não recorrentes, líquidos	39	3.093	10.726	3.092	10.726	1.136	56.796
<b>TOTAL DOS CUSTOS, PERDAS E GANHOS</b>		<b>405.972</b>	<b>1.997.399</b>	<b>397.731</b>	<b>1.287.837</b>	<b>329.088</b>	<b>1.230.576</b>
<b>RESULTADOS ANTES DE PERDAS / (GANHOS) EM EMPRESAS PARTICIPADAS, RESULTADOS FINANCEIROS E IMPOSTOS</b>		<b>6.076</b>	<b>201.831</b>	<b>6.656</b>	<b>200.767</b>	<b>22.220</b>	<b>137.310</b>
Perdas / (ganhos) em empresas participadas, líquidas	12 e 36	3.318	1.022	3.318	(291)	6.099	2.099
Custos de financiamento	40	4.212	20.441	4.211	20.441	8.908	22.218
Perdas / (ganhos) em valores cambiais, líquidas		175	139	172	139	90	548
Perdas / (ganhos) em ativos financeiros, líquidas		151	142	149	142	2	53
Outros custos / (provisões) financeiros, líquidos		1.039	3.026	1.040	3.025	1.081	1.814
<b>RESULTADO ANTES DE IMPOSTOS</b>		<b>8.914</b>	<b>25.790</b>	<b>8.910</b>	<b>35.749</b>	<b>10.052</b>	<b>35.723</b>
Imposto sobre o rendimento	(43b)	(43b)	(176.041)	(254)	(174.978)	(5.167)	(191.578)
<b>RESULTADO LÍQUIDO POR AÇÃO</b>		<b>15</b>	<b>(150.251)</b>	<b>(164)</b>	<b>(139.229)</b>	<b>(5.115)</b>	<b>(155.855)</b>
<b>RESULTADO CONSOLIDADO LÍQUIDO DAS UNIDADES OPERACIONAIS EM CONTINUAÇÃO</b>		<b>5.381</b>	<b>143.243</b>	<b>5.526</b>	<b>142.421</b>	<b>10.094</b>	<b>85.236</b>
Resultado consolidado líquido das unidades operacionais descontinuidas	47	-	-	(145)	822	-	6.407
<b>RESULTADO CONSOLIDADO LÍQUIDO</b>		<b>5.381</b>	<b>143.243</b>	<b>5.381</b>	<b>143.243</b>	<b>10.094</b>	<b>91.643</b>
Ativos / (passivos) em unidades operacionais descontinuidas		5.401	143.494	5.401	143.494	12.879	92.000
Interesses que não controlam	23	(20)	(251)	(20)	(251)	216	(357)
<b>RESULTADO LÍQUIDO POR AÇÃO DAS UNIDADES OPERACIONAIS EM CONTINUAÇÃO</b>		<b>41</b>	<b>0,01</b>	<b>0,01</b>	<b>0,01</b>	<b>0,28</b>	<b>0,03</b>
Básico - euros	41	0,01	0,28	0,01	0,28	0,03	0,18
Diluído - euros	41	0,01	0,28	0,01	0,28	0,03	0,18

\*Reapresentação decorrente da classificação da NOS International Carrier Services como uma unidade operacional descontinuada (nota 47).

Como prática corrente, apenas as contas anuais são auditadas, sendo que os valores trimestrais não foram auditados de forma autónoma.

O anexo faz parte integrante da demonstração consolidada dos resultados por natureza para o exercício findo em 31 de dezembro de 2020.  
O Contabilista Certificado O Conselho de Administração

Fig. 24 Page in PDF format with 2 tables located on the same page



2020_504453513_NOS_R&C_CC-page-178 [Modo de Compatibilidade] - Excel (A Ativação do Produto Falhou)											
Ficheiro Base Inserir Esquema de Página Fórmulas Dados Reverter Ver PDFElemento Digite o que pretende fazer...											
Cortar Copiar Pincel de Formatação Área de Transferência Tipo de Letra Alinhamento Número Estilos Formatação Condicional Formatar como Tabela Estilos de Célula Inserir Eliminar Formatar Soma Automática Preenchimento Limpar Ordenar e Filtar Localizar e Selecionar Partilha											
D1											
	A	B	C	D	E	F	G	H	I	J	K
		31-12-2019	31-12-2020		NOTAS REPORTADO	REPORTADO	4o TRIM 19	REEXPRESSO*	12M 19.1	4o TRIM 20	12M 20
2	ATIVO										
3	ATIVO NÃO CORRENTE										
4	Ativos fixos tangíveis Propriedad	1 034 813 653	991 613 637	RÉDITOS Prestação de serviços	380 227	1 485 935	332 566	1 345 108	318 611	1 262 980	
5	Ativos intangíveis	1 014 066	1 041 087	Vendas	28 063	89 141	28 063	89 141	29 210	96 305	
6	Encargos de contratos com client	163 101	162 123	Outras receitas	5 758	24 155	5 758	24 155	6 488	18 597	
7	Direitos de uso Investimentos em	218 383 18 244	260 097 10 897	CUSTOS, PERDAS E GANHOS	30 414 048	1 599 230	366 387	1 458 404	354 309	1 367 886	
8	Contas a receber - outros	4 064	7 504	Custos com o pessoal	31 22 793	85 176	22 715	84 830	21 876	85 331	
9	Impostos a recuperar Outros ativ	149 439	149 579	Custos diretos Custo das mercadorias ven	32 33 145 402 21 983	524 058 64 228	99 475 21 982	384 889 64 228	100 080 24 832	348 77674 312	
10	Ativos por impostos diferidos	80 426	82 782	Marketing e publicidade	15 655	37 216	15 655	37 216	10 045	24 504	
11	TOTAL DO ATIVO NÃO CORRETE	2 534 342	2 557 468	Serviços de suporte	34 22 519	82 335	22 441	82 257	25 055	96 281	
12	ATIVO CORRENTE Inventários	34 081	43 628	Fornecimentos e serviços externos Outros	34 29 843 123	112 863 516	27 952	123 112 670 516	27 213 331	100 542719	
13	Contas a receber - clientes	361 712	290 652	Impostos indiretos	8 354	32 844	8 406	32 844	6 255	32 747	
14	Ativos de contratos com clientes	88 059 28 128	61 002 28 610	Provisões e ajustamentos Depreciações	35 8,9,10,11 e 37 123 344 11 6	18 934 421 318	11 739 123 341	18 983 421 313	4 835 104 597	11 493409 842	
15	Impostos a recuperar	4 631	2 894	Custos de reestruturação	36 713	7 732	7 713	7 732	1 033	5 523	
16	Custos diferidos	43 954	34 054	Perdas / (ganhos) com a alienação de ativ	98	(547)	97	(547)	(290)		
17	Ativos não correntes detidos par	450	450	Outros custos / (ganhos) não correntes	39 3 093	10 726	3 092	10 726	1 136	50 796	
18	Caixa e equivalentes de caixa	12 819	153 285	Outros custos / (ganhos) não correntes	405 572	1 397 399	357 731	1 257 657	329 088	1 230 576	
19	TOTAL DO ATIVO CORRENTE	553 834	615 175	RESULTADOS ANTES DE PERDAS / (GANHOS) EM EMPRESAS		201 831	6 656	200 747	25 220	137 310	
20	TOTAL DO ATIVO	3 088 176	3 172 643	PARTICIPADAS, RESULTADOS FINANC	8 476	1 022	3 318	1 022	(29)	9 099	
21	CAPITAL PRÓPRIO			Perdas / (ganhos) em empresas particip	12 e 36 3 318	20 661	4 211	20 661	6 908	22 218	
22	Capital social	5 152	5 152	Custos de financiamento	40 4 212	139	142	139	90	548	
23	Prémio de emissão de ações	654 219	654 219	Perdas / (ganhos) em variações cambiais	175	142	149	142	2	53	
24	Ações próprias	(14 655)	(14 859)	Perdas / (ganhos) em ativos financeiros	11 151	3 826	1 060	3 825	1 081	3 814	
25	Reserva Legal	1 030	1 030	Outros custos / (provetos) financeiros	liq 40 1 059	25 790	9 910	25 789	10 052	35 732	
26	Outras reservas e resultados ac	16 041	12 007	Dívidas - euros	6 914	(254)	(254)	174 978	15 167	101 578	
27	Resultado líquido	143 494	92 000	RESULTADO ANTES DE IMPOSTOS	(438)	176 041		32 557	2 073	16 342	
28	CAPITAL PRÓPRIO EXCLUINDO	1 005 281	949 549	Imposto sobre o rendimento	15 (5 820)	32 798	(5 780)				
29	Interesses que não controlam	7 042	6 685								
30				RESULTADO CONSOLIDADO LÍQUIDO DAS UNIDADES							
31	TOTAL DO CAPITAL PRÓPRIO	1 012 322	956 234	OPERACIONAIS EM CONTINUAÇÃO	5 381	143 243	5 526	142 421	13 094	85 236	
32	PASSIVO			Resultados consolidados líquidos das unidades 47 -			(145)	822	-	6 407	
33	PASSIVO NÃO CORRENTE			descontinuadas							
34	Empréstimos obtidos	1 216 847	1 363 514	RESULTADO CONSOLIDADO LÍQUIDO	5 381	143 243	5 381	143 243	13 094	91 643	
35	Provisões	94 959	73 345	atribuível a							
36	Contas a pagar - outros	5 855	40 050	Detentores do capital da empresa-mãe	5 401	143 494	5 401	143 494	12 879	92 000	
37	Acrescimos de custos	667	505	Interesses que não controlam	23 (20)	(251)		(20)	(251)	(357)	
38	Provetos diferidos	5 123	4 729	RESULTADO LÍQUIDO POR AÇÃO	655						
39	Instrumentos financeiros derivad	265	655	Básico - euros	41 0,01	0,28	0,01	0,28	0,03	0,18	
40	Passivos por impostos diferidos	11 626	5 025	Dívidas - euros	41 0,01	0,28	0,01	0,28	0,03	0,18	
41	TOTAL DO PASSIVO NÃO COR	1 333 343	1 487 823	RESULTADO LÍQUIDO POR AÇÃO DAS UNIDADES OPERACIONAIS							
42	PASSIVO CORRENTE			EM CONTINUAÇÃO							
43	Empréstimos obtidos	143 281	167 126	Básico - euros	41 0,01	0,28	0,01	0,28	0,03	0,17	
44	Contas a pagar - fornecedores	259 499	252 007	Dívidas - euros	41 0,01	0,28	0,01	0,28	0,03	0,17	
45	Contas a pagar - outros	33 835	47 438								
46	Impostos a pagar	68 202	51 981								
47	Acrescimos de custos	203 726	175 860								
48	Provetos diferidos	33 834	33 228								
49											
50	Instrumentos financeiros derivad	135	346								
51	TOTAL DO PASSIVO CORRENTE	742 511	728 586								
52	TOTAL DO PASSIVO	2 075 854	2 216 409								
53	TOTAL DO CAPITAL PRÓPRIO	3 088 176	3 172 643								
54	ATIVO										
55	ATIVO NÃO CORRENTE										

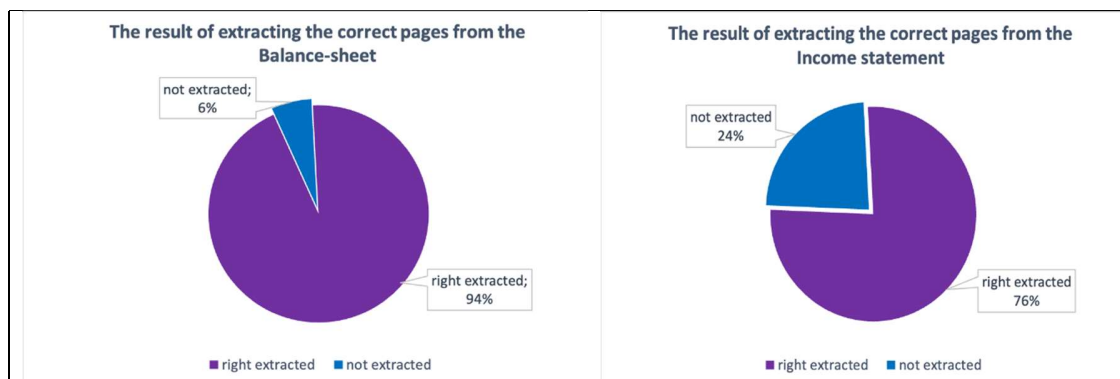
\*Reexpressão decorrente da classificação da NOS International Carrier Services como uma unidade operacional descontinuada (nota 47).

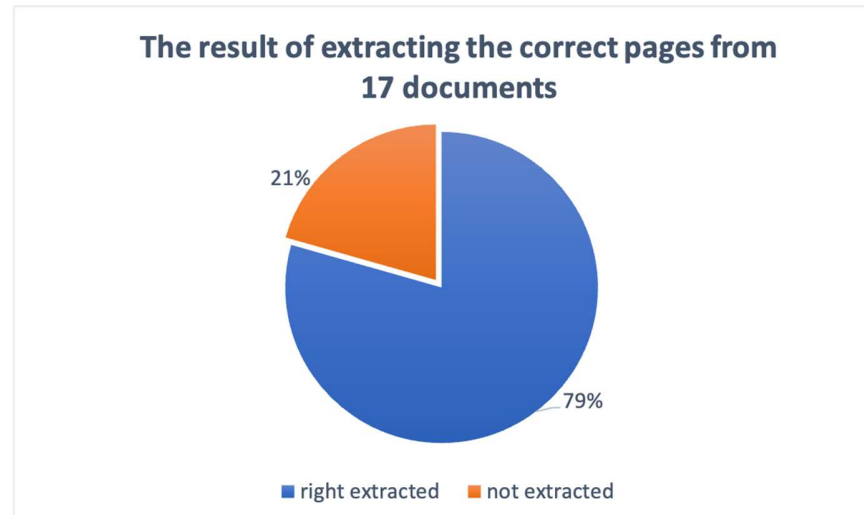
Como prática recorrente, apenas as contas anuais são auditadas, sendo que os valores trimestrais não foram auditados de forma autónoma.

**Fig. 25 The result of the extraction of the 2 tables located on the same page**

For all written code algorithms a folder is automatically created into which PDF pages are uploaded. Similarly, a folder is created in which the correct xlsx files are saved.

The first algorithm Version 1 tested 17 documents. The algorithm found many additional similar pages. The accuracy of finding the correct tables equal 79 %.





**Fig. 26** The diagrams of the accuracy of finding the correct tables using Version 1

**Accuracy calculation:** we calculated the accuracy of all developed Versions in the following way, the number of correctly recognized pages was divided by the total number of pages that needed to be recognized and calculated as a percentage.

## 5.4 Version 2

**Split documents into pages:** we used library pikepdf - splitting a pdf document into separate pages (one page - one new document) [21]. It is worth noting that this library was used for all written code algorithms because it is effective.

### Extract text:

In Version 2, we tested 17 documents for words extracted from the HAM 6 code (learning from reporting of the 6 companies. In this code, the words contained in Group 1 and Group 2 are HAM, and words that do not suit us, this is spam), for algorithm learning was used a link to the right pages.

Code training from reporting of the 6 companies has the name `Select_group_words_version1`.

```
In [5]: # it is important to indicate the page of the first group first, and then the second
ham = {
    "500166587_2020_LISGRÁFICA_R&C_CC": ["500166587_2020_LISGRÁFICA_R&C_CC-page-135.pdf", "500166587_2020_LISGRÁFICA_R&C_CC-page-136.pdf", "500166587_2020_LISGRÁFICA_R&C_CC-page-137.pdf"],
    "500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC": ["500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC-page-123.pdf", "500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC-page-124.pdf", "500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC-page-125.pdf"],
    "502816481_2020_MÉDIA CAPITAL_R&C_CC": ["502816481_2020_MÉDIA CAPITAL_R&C_CC-page-123.pdf", "502816481_2020_MÉDIA CAPITAL_R&C_CC-page-124.pdf", "502816481_2020_MÉDIA CAPITAL_R&C_CC-page-125.pdf"],
    "2020_503025798_The Navigator Company_R&C_CC": ["2020_503025798_The Navigator Company_R&C_CC-page-2.pdf", "2020_503025798_The Navigator Company_R&C_CC-page-3.pdf", "2020_503025798_The Navigator Company_R&C_CC-page-4.pdf"],
    "2020_508548527_Ramada_R&C_CC": ["2020_508548527_Ramada_R&C_CC-page-178.pdf", "2020_508548527_Ramada_R&C_CC-page-179.pdf", "2020_508548527_Ramada_R&C_CC-page-180.pdf"],
    "2020_502593130_Semapa_R&C_CC": ["2020_502593130_Semapa_R&C_CC-page-139.pdf", "2020_502593130_Semapa_R&C_CC-page-140.pdf", "2020_502593130_Semapa_R&C_CC-page-141.pdf"]
}
```

**Fig. 27** The HAM 6 code learning algorithm is based on reporting of the 6 companies

As a result of training the algorithm based on 6 documents, the most relevant words for Group 1 and Group 2 were obtained.

The more relevant words that were selected to search for the Group 1: "00", "ativos", "caixa", "capital", "consolidar", "corrente", "equivalente", "euro", "financeiro", "fixo", "noto", "outro", "passivar", "passivo", "provisão", "próprio", "reservar", "resultar", "total".

The more relevant words that were selected to search for the Group 2: "00", "antar", "consolidar", "demonstração", "euro", "externo", "financeiro", "fornecimento", "imparidade", "imposto", "noto", "outro", "perda", "pessoal", "rendimento", "resultar", "serviço", "sobrar".

For text recognition, we use the **pdfminer** library.

### **Preprocessing:**

In Version 2 we used:

- Lowercasing. This operation is responsible for converting upper case characters to lower representation.
- We use spacy for lemmatization, bringing words back to their original form.
- Replace all numbers in the range (1, 10) for "0".
- Removed stop words and saved lemmas to a list. How many words of the Group were found in the text (each word is counted once).
- Produced standardization of word usage.
- Tokenization - this operation is a step that breaks long lines of text into smaller chunks or tokens. Large chunks of text can be turned into sentences, and sentences into words. Further processing is usually done after a piece of text has been properly composed. Tokenization is also known as text segmentation or lexical analysis. Segmentation is sometimes used to break up of a large chunk of text into pieces larger than words (such as paragraphs or sentences), while tokenization is for the process of breaking up, purely into words.

**Scoring:** Scoring with threshold: In Version 2 the threshold was calculated with a parameter  $\pm 1$ . To do this, the number of words found in the group is divided by the number of words in the group.

*results\_line["keywords\_g1"]  $\pm 1$*

A similar calculation was made with a threshold of 1 with Group 2.

An algorithm was developed that calculates the ratio of the found words, determined by us for each group, to the total number of words in this group. That is, 'count\_group1 / len(group1)' for Group 1 and 'count\_group2 / len(group2)' for Group 2. A search for equality of occurrences was also carried out. An algorithm was written that sorts the pages in Group 1 and Group 2 by the largest number of "Total \_words" in each group. The results of this algorithm quite effectively identify the pages we need.

Using the example of the company LISGRÁFICA\_R&C\_CC, the pages required for extraction were determined correctly. Pages 135 and 134.



Out[97]:

	page	group1	group2	total_words
41	500166587_2020_LISGRÁFICA_R&C_CC-page-135	5.000	5.000	80
43	500166587_2020_LISGRÁFICA_R&C_CC-page-137	3.846	5.128	78
5	500166587_2020_LISGRÁFICA_R&C_CC-page-102	3.125	3.125	64
4	500166587_2020_LISGRÁFICA_R&C_CC-page-101	3.175	6.349	63
38	500166587_2020_LISGRÁFICA_R&C_CC-page-132	3.175	3.175	63
29	500166587_2020_LISGRÁFICA_R&C_CC-page-124	3.279	1.639	61
123	500166587_2020_LISGRÁFICA_R&C_CC-page-83	3.571	1.786	56
36	500166587_2020_LISGRÁFICA_R&C_CC-page-130	4.082	4.082	49
18	500166587_2020_LISGRÁFICA_R&C_CC-page-114	4.255	4.255	47
0	500166587_2020_LISGRÁFICA_R&C_CC-page-0	3.571	0.000	28
8	500166587_2020_LISGRÁFICA_R&C_CC-page-105	3.846	7.692	26
37	500166587_2020_LISGRÁFICA_R&C_CC-page-131	11.538	7.692	26

**Fig. 28 The DataFrame with divided into groups with the largest number of total\_words in each group**

Extracted to xlsx files pages 135 and 134 of the company LISGRÁFICA\_R&C\_CC look like this.

500166587_2020_LISGRÁFICA_R&C_CC-page-135 -																
Ficheiro Base Inserir Esquema de Página Fórmulas Dados Rever Ver PDFelement Diga o que pretende fazer...																
Calibri 11 A A N I S T Moldar Texto Geral Unir e Centrar % 000 0,00 0,00 Formatação Condiciona																
Área de Transferência Tipo de Letra Alinhamento Número																
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2	ATIVOS NÃO CORRENTES:	Notas	2020	2019												
3	Ativos intangíveis	12	538 551	260 814												
4	Ativos fixos tangíveis	13	3 361 859	5 583 273												
5	Direitos de uso	12	1 307 248	1 481 172												
6	Investimentos financeiros		2 347	1 825												
7	Ativos por impostos diferidos	11	43 595	68 507												
8	Clientes e contas a receber	16	617 532	1 095 550												
9	Outros ativos não correntes	14	70 413	155 730												
10	Total de ativos não correntes		7 241 545	9 646 871												
11	ATIVOS CORRENTES:															
12	Existências	15	102 621	52 261												
13	Clientes e contas a receber	16	990 278	2 178 815												
14	Outros ativos correntes	17	25 315	506 476												
15	Imposto sobre o rendimento	26	116 249	116 250												
16	Caixa e seus equivalentes	18	16 301	14 964												
17	Total de ativos correntes		2 880 764	5 216 566												
18	TOTAL DO ATIVO		10 122 309	12 863 437												
19																
20	CAPITAL PRÓPRIO E PASSIVO															
21	CAPITAL PRÓPRIO:															
22	Capital	19	9 334 831	9 334 831												
23	Ações próprias	19	(527 531)	(527 531)												
24	Reserva legal	19	1 866 966	1 866 966												
25	Resultados transitados	19	(32 185 870)	(26 716 175)												
26	Outras variações no capital próprio	19	5 152 807	5 152 807												
27	Resultado consolidado líquido do período		(3 000 472)	(5 469 695)												
28	Total do capital próprio		(19 359 269)	(16 358 799)												
29	PASSIVO:															
30	PASSIVOS NÃO CORRENTES:															
31	Provisões outros riscos e encargos	25	-	927 481												
32	Empréstimos obtidos	20	5 653 203	727 198												
33	Locações	21	790 425	1 510 117												
34	Outros passivos não correntes	23	5 172 717	5 652 340												
35	Fornecedores e contas a pagar	24	3 965 733	5 498 191												
36	Passivos por impostos diferidos	11	2 234 438	2 622 686												
37	Total de passivos não correntes		20 816 516	21 938 015												
38	PASSIVOS CORRENTES:															
39	Empréstimos obtidos	20	3 301 972	5 874 740												
40	Locações	21	1 549	5 932												
41	Fornecedores e contas a pagar	24	3 792 613	4 254 473												
42	Outros passivos correntes	23	2 480 027	1 039 528												
43	Imposto sobre o rendimento	26	48 901	59 550												
44	Total de passivos correntes		5 665 062	284 223												
45	TOTAL DO CAPITAL PRÓPRIO E DO PASSIVO		10 122 309	12 863 437												

**Fig. 29 The Extracted to xlsx files page 135 of the company LISGRÁFICA\_R&C\_CC**

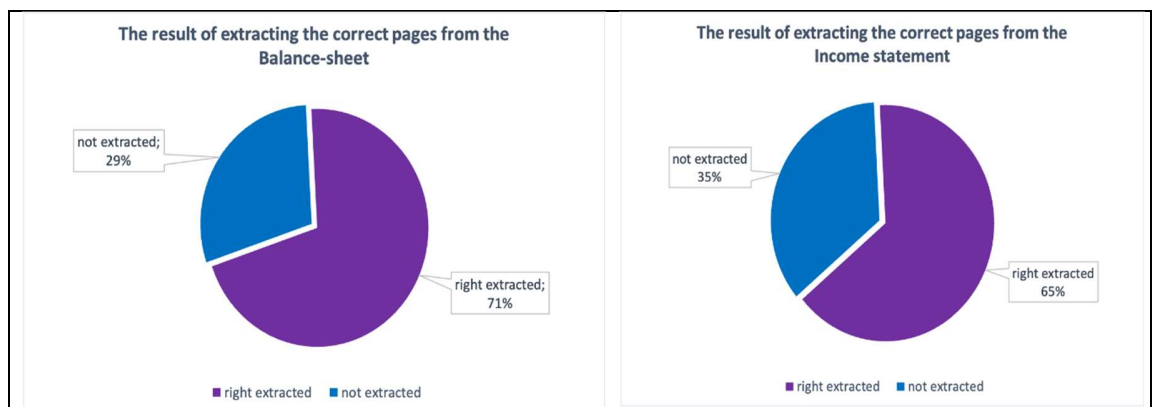
500166587\_2020\_LISGRÁFICA\_R&C\_CC-page-134 -

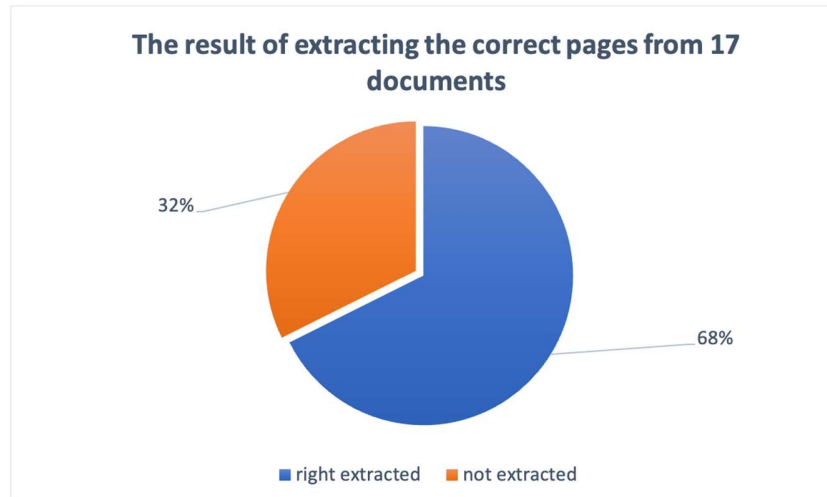
	A	B	C	D	E	F	G
1							
2	<b>PROVEITOS OPERACIONAIS:</b>	<b>Notas</b>	<b>2020</b>	<b>2019</b>			
3	Vendas	4	9 287 163	13 202 862			
4	Outros proveitos operacionais	5	1 589 410	1 380 708			
5	Total de proveitos operacionais		10 876 573	14 583 570			
6	<b>CUSTOS OPERACIONAIS:</b>						
7	Custo das mercadorias vendidas	6	(2 956 823)	(4 388 925)			
8	Fornecimentos e serviços externos	7	(3 142 068)	(4 967 894)			
9	Custos com o pessoal	8	(3 218 437)	(4 449 308)			
10	Depreciações e Amortizações	12 e 13	(1 103 586)	(1 187 940)			
11	Perdas por imparidade em clientes e out	22	(612 235)	(930 605)			
12	Provisões (aumentos/reduções)	25	(73 946)	(37 662)			
13	Outros custos operacionais	9	(1 666 432)	(3 410 213)			
14	Total de custos operacionais		(12 773 527)	(19 372 547)			
15	Resultados operacionais		(1 896 954)	(4 788 977)			
16	<b>RESULTADOS FINANCEIROS:</b>						
17	Custos financeiros	10	(1 459 146)	(1 368 519)			
18	Proveitos financeiros	10	486	909			
19	Resultados antes de impostos		(3 355 614)	(6 156 587)			
20	Imposto sobre o rendimento do período	31	355 142	686 892			
21	Resultado consolidado líquido do período		(3 000 472)	(5 469 695)			
22	Rendimento integral		(3 000 472)	(5 469 695)			
23	Atribuível a:						
24	Acionistas da empresa-mãe		(3 000 472)	(5 469 695)			
25	Resultado por ação						
26	Básico	29	(0.0162)	(0.0295)			
27	Diluído	29	(0.0162)	(0.0295)			

**Fig. 30 The Extracted to xlsx files page 134 of the company LISGRÁFICA\_R&C\_CC**

### Extract tables and output:

Saved the results in a DataFrame, also saved to an xlsx file. By the analogy to all implemented algorithms.





**Fig. 31** The diagrams of the accuracy of finding the correct tables using Version 2

As we can see from the results of the tested algorithm 2, this code does not recognize the tables we need very efficiently, with the accuracy of finding the correct tables equal to 68 %. At the same time, it should be noted that the number of filtered pages corresponding to the HAM 6 word groups for recognizing the tables we need has significantly decreased, which in our case is a positive trend for continuing to work with the HAM algorithm.

## 5.5 Version 3

**Split documents into pages:** we used library pikepdf. [21].

**Extract text:**

In Version 3, we tested 17 documents with words extracted from the code HAM 12 (learning from reporting of the 12 companies. In this code, the words contained in Group 1 and Group 2 are HAM and words that do not suit us, this is spam), with the accuracy of finding the correct tables equal 97 %. Code training from reporting of the 12 companies has name **Select\_group\_words\_version2**. For algorithm learning was used a link to the right pages.

```
ham = {"500166587_2020_LISGRÁFICA_R&C_CC": ["500166587_2020_LISGRÁFICA_R&C_CC-page-135.pdf", "500166587_2020_LISGRÁFICA_R&C_CC-page-136.pdf"],
"500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC": ["500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC-page-111.pdf", "500136971_2020_IMOBILIARIA CONSTRUTORA GRÃO-PARA S.A._R&C_CC-page-112.pdf"],
"502816481_2020_MÉDIA CAPITAL_R&C_CC": ["502816481_2020_MÉDIA CAPITAL_R&C_CC-page-123.pdf", "502816481_2020_MÉDIA CAPITAL_R&C_CC-page-124.pdf"],
"2020_501669477_Ibersol_R&C_CC": ["2020_501669477_Ibersol_R&C_CC-page-177.pdf", "2020_501669477_Ibersol_R&C_CC-page-178.pdf"],
"2020_508548527_Ramada_R&C_CC": ["2020_508548527_Ramada_R&C_CC-page-111.pdf", "2020_508548527_Ramada_R&C_CC-page-112.pdf"],
"2020_503541320_GLINTT_R&C_CC": ["2020_503541320_GLINTT_R&C_CC-page-87.pdf", "2020_503541320_GLINTT_R&C_CC-page-88.pdf"],
"2020_504453513_NOS_R&C_CC": ["2020_504453513_NOS_R&C_CC-page-178.pdf", "2020_504453513_NOS_R&C_CC-page-179.pdf"],
"2020_500077568_CTT_R&C_CC": ["2020_500077568_CTT_R&C_CC-page-202.pdf", "2020_500077568_CTT_R&C_CC-page-203.pdf"],
"2020_500077797_CorticeiraAmorim_R&C_CC": ["2020_500077797_CorticeiraAmorim_R&C_CC-page-296.pdf", "2020_500077797_CorticeiraAmorim_R&C_CC-page-297.pdf"],
"2020_502028351_SONAEOM_R&C_CC": ["2020_502028351_SONAEOM_R&C_CC-page-117.pdf", "2020_502028351_SONAEOM_R&C_CC-page-118.pdf"],
"2020_502293225_Cofina_R&C_CC": ["2020_502293225_Cofina_R&C_CC-page-112.pdf", "2020_502293225_Cofina_R&C_CC-page-113.pdf"],
"2020_502593130_Semapa_R&C_CC": ["2020_502593130_Semapa_R&C_CC-page-139.pdf", "2020_502593130_Semapa_R&C_CC-page-140.pdf"]}
}
```

**Fig. 32** The HAM 12 code learning algorithm is based on reporting of the 12 companies

Together with colleagues, it was decided to partially change the words obtained by the HAM 12 algorithm to more relevant ones: in Group 1, the word "consolida" was added and the word "passivar" was excluded; in Group 2, the words "imposto", "atribuível", "outro" were added, and the words "externo", "fornecimento", "sobrar", "antar" were excluded.

As a result, the most relevant words for Group 1 and Group 2 were obtained. The more relevant words that were selected to search for the Group 1: "00", "ativos", "caixa", "capital", "consolida", "corrente", "diferir", "euro", "financeiro", "fixo", "outro", "passivo", "provisão", "próprio", "reservar", "resultar", "total".

The more relevant words that were selected to search for the Group 2: "00", "serviço", "consolidar", "euro", "imposto", "demonstração", "noto", "perda", "rendimento", "imparidade", "resultar", "outro", "pessoal", "atribuível", "financeiro".

For text recognition, I use the **pdfplumber** library. Library pdfplumber is an approach to read and parse data from PDF by page-by-page. PDF allows us to extract text, words, and tables from the page or convert it into an image.

### **Preprocessing:**

In Version 3 we used similar directives applied in Version 2.

**Scoring:** In Version 3 the search for equality of occurrences was also carried out. An algorithm was created that calculates the maximum number of matches from the group that could be found. A search for equality of occurrences was also carried out. An algorithm was created that calculates the maximum number of words from the group that could be found. If the number of words from the first group is the largest on the specified page, then we write down the document's name, and remember how many maximum words matched the group at the moment.

```
if count_group1 > documents1['count_unique']:
documents1['names'] = [f"{name}/{file}"]
documents1['count_unique'] = count_group1
results_line["keywords_g1"] = 1
if file in ham[filename][0]:
results_line["right"] += 1
```

If the number of words from the group is equal to the maximum of those found earlier, then we write the name of this file too.

```
elif count_group1 == documents1['count_unique']:
documents1['names'].append(f"{name}/{file}")
results_line["keywords_g1"] += 1
if file in ham[filename][0]:
results_line["right"] += 1
```

The results of this algorithm quite effectively identify the pages we need.



The next task set by the Bank was to extract information containing the name **NIPC** and **Year** for all Financial Statements, with further uploading along with a table in **xlsx**.

I solved this problem by extracting relevant information from the name indicated in the Reporting of Companies.

```
NIPC = filename.split("_")[1]
```

```
year = filename.split("_")[0]
```

```
meta_info = pd.DataFrame({'NIPC': [NIPC], 'year': [year]})
```

Now, the results exported in **xlsx** look like this. An example of uploading the Report Martifer\_R&C\_CC in **xlsx**. Page 94.

2020_505127261_Martifer_R&C_CC-94 -							
Ficheiro Base Inserir Esquema de Página Fórmulas Dados Rever Ver PDFelement Diga o que pretende fazer...							
<div> <div>Cortar Copiar Colar Pincel de Formatação</div> <div> <div>Calibri 11</div> <div>N I S</div> <div>Área de Transferência</div> </div> <div> <div>Alinhamento</div> <div>Número</div> </div> </div>							
A	B	C	D	E	F	G	H
1	NIPC	Year					
2	505127261	2020					
3							
4	€	NOTAS	ANO 2020	ANO 2019	2020	2019	
5					(NÃO AUDITAD	(NÃO AUDITADO)	
6	Vendas e prestações de serviços	3, 4	226.121.546	235.914.875	121.341.587	124.554.492	
7	Outros rendimentos operacionais	5	23.132.718	31.020.752	16.715.982	26.748.520	
8	Custo das mercadorias vendidas e matérias con	6	(70.964.787)	(84.663.616)	(37.701.984)	(48.183.181)	
9	Subcontratos	7	(70.765.306)	(64.505.967)	(36.858.124)	(29.552.968)	
10	Fornecimentos e serviços externos	8	(30.354.904)	(31.016.039)	(16.075.431)	(17.287.051)	
11	Gastos com o pessoal	9	(36.738.171)	(37.423.304)	(19.356.944)	(18.447.537)	
12	Perdas de imparidade de ativos financeiros	25	3.265.548	(8.911.182)	1.376.294	(8.774.558)	
13	Outros gastos operacionais	10	(24.327.072)	(11.542.126)	(17.019.967)	(8.347.456)	
14		3	19.369.572	28.873.393	12.421.413	20.710.261	
15	Amortizações e depreciações	3, 18, 19, 20	(6.006.771)	(8.528.944)	(2.756.702)	(3.866.774)	
16	Provisões	3, 11, 35	(551.291)	669.829	(923.029)	265.765	
17	Perdas de imparidade de ativos não financeiros	3, 11	222.415	(2.732.810)	9.647	(2.732.810)	
18		3	13.033.925	18.281.467	8.751.329	14.376.442	
19	Rendimentos e ganhos financeiros	12	2.407.918	3.753.711	1.007.581	3.344.417	
20	Gastos e perdas financeiros	12	(7.802.665)	(7.700.919)	(3.843.115)	(3.613.689)	
21	Ganhos / (perdas) em empresas associadas e conjuntamente						
22		3, 13	248.900	8.026.010	(504.545)	1.788.967	
23	controladas						
24	Ganhos / (perdas) monetárias líquidas	43	207.623	83.284	147.085	111.049	
25	Resultado antes de imposto sobre o rendimento		8.095.701	22.443.553	5.558.335	16.007.186	
26	Imposto sobre o rendimento	14	(1.411.517)	(996.873)	(964.217)	(97.162)	
27	Resultado líquido do exercício	3	6.684.184	21.446.679	4.594.118	15.910.024	
28	Atribuível:						
29	a interesses que não controlam	31	385.981	(2.099.565)	(161.229)	(2.196.468)	
30	aos detentores do capital da empresa-mãe	16	6.298.203	23.546.244	4.755.346	18.106.488	
31							
32	Resultado líquido por ação:	16					
33	básico e diluído		0,0644	0,2408	0,0486	0,1852	
34							

**Fig. 33** The Extracted to **xlsx** file page with extracted **NIPC** and **Year**

## Extract tables and output:

Example of uploading Report CTT\_R&C\_CC in xlsx. Page 203. It should be noted that the table Income Statement is vertical in the document, but the **pdfplumber** library does a very good job of recognizing these types of tables.

RELATÓRIO INTEGRADO 2020

204

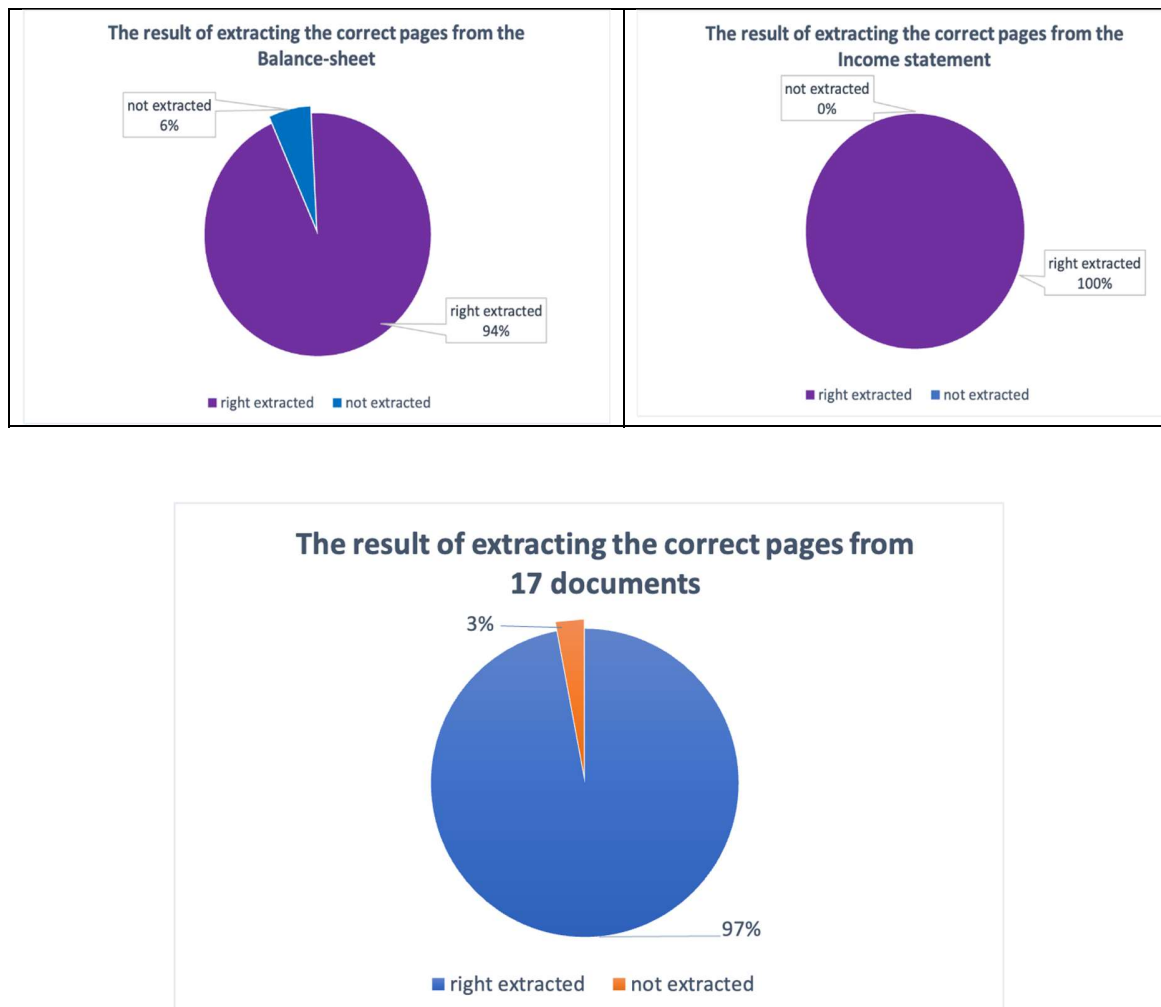
Fig. 34 Page in PDF format with the table Income Statement is vertical

2020\_500077568\_CTT\_R&C\_CC-page-203 - Excel (A Ativação do Produto Falhou)

	Notas	31.12.2019	31.12.2020	31.12.2019	31.12.2020	31.12.2019	31.12.2020	31.12.2019	31.12.2020
5 Vendas e serviços prestados	40	688.021.669	672.854.025	181.631.937	188.951.581	522.297.559	468.833.332	135.789.237	124.632.891
6 Margem Financeira	41	29.315.856	44.636.907	10.421.070	11.814.868	-	-	-	-
7 Outros rendimentos e ganhos operacionais	41	22.948.405	27.749.403	8.682.378	10.211.951	40.541.244	44.710.790	12.527.182	14.952.111
8 Custo das mercadorias vendidas e das matérias consumidas	17	740.285.930	745.240.335	200.735.385	210.978.400	562.838.803	513.544.122	148.316.420	139.585.002
9 Fornecimentos e serviços externos	42	(14.261.450)	(19.218.064)	(4.805.042)	(5.897.765)	(13.588.474)	(18.607.910)	(4.620.763)	(5.736.972)
11 Gastos com o pessoal	43	(242.776.520)	(256.144.789)	(64.942.709)	(74.338.907)	(121.098.644)	(111.195.328)	(31.309.271)	(30.280.487)
12 Imparidade de contas a receber (perdas/reversões)	44	(356.004.365)	(342.488.107)	(93.868.310)	(91.046.599)	(310.883.876)	(293.331.088)	(81.201.296)	(78.115.122)
13 Imparidade de outros ativos financeiros bancários	44	(7.800.406)	(5.613.098)	(3.603.244)	(901.621)	(1.905.392)	(2.794.597)	(1.247.811)	(429.035)
14 Provisões (aumentos/reversões)	44	(3.095.636)	(8.916.969)	(1.409.457)	(1.333.741)	-	-	-	-
15 Depreciações / amortizações e imparidade de investimentos (perdas/reversões)	32	905.250	(853.298)	393.979	69.532	1.367.746	(83.122)	669.600	(209.822)
16 Resultados de outros ativos e passivos financeiros bancários	45	(54.223.229)	(62.135.823)	(14.760.277)	(16.080.957)	(41.077.288)	(46.597.825)	(11.242.399)	(12.024.269)
17 Outros gastos e perdas operacionais	46	(16.233.140)	(16.194.526)	(4.730.529)	(4.437.048)	(8.823.425)	(8.752.418)	(2.383.682)	(2.638.063)
18 Ganhos/perdas com alienação de ativos	47	488.912	451.469	34.690	155.309	452.776	678.502	28.900	63.944
19 Gastos e perdas financeiros	48	(693.000.585)	(710.733.205)	(187.690.900)	(193.742.415)	(495.556.578)	(480.683.786)	(131.306.721)	(129.369.826)
20 Rendimentos financeiros	48	47.285.345	34.507.130	13.044.485	17.235.985	67.282.225	32.860.335	17.009.699	10.215.175
21 Ganhos/perdas em subsidiárias, associadas e empreendimentos conjuntos	48	(10.421.170)	(9.660.185)	(2.920.989)	(2.350.307)	(9.094.665)	(8.366.012)	(2.567.708)	(2.033.491)
22 Resultado antes de impostos	48	63.609	20.091	(133.260)	9.336	351.179	525.238	(46.674)	164.195
23 Imposto sobre o rendimento do período	10/11/12	(1.400.621)	(1.741.529)	(788.869)	(658.864)	(12.795.844)	(958.448)	(3.299.162)	6.095.223
24 Resultado líquido do período	49	35.527.163	23.125.507	9.201.367	14.236.150	45.742.896	24.061.113	11.096.154	14.441.102
25 Resultado líquido do período atribuível a:									
26 Detentores de capital	29	29.196.933	16.669.309	6.344.751	12.339.831	-	-	-	-
27 Interesses não controlados	28	87.767	97.225	58.549	11.087	-	-	-	-
28 Resultado por ação:	28	6,19	6,11	0,04	0,08	0,19	0,11	0,19	0,11

Fig. 35 The Extracted to xlsx files page using the pdfplumber library

Results of the implemented algorithm Version 3 with the accuracy of finding the correct tables equal 97 %.



**Fig. 36** The diagrams of the accuracy of finding the correct tables using Version 3

## 5.6 Version 4

### Extract text:

With code Version 4, we tested 19 documents with words extracted from the code HAM 12 (learning from reporting of the 12 companies). With a similar set of the words for each Group, applied in Version 3.

For text recognition, we use the **pdfplumber** library.

To improve the recognition of documents of poor quality, we installed the Tesseract - OCR library. This function will handle the core OCR processing of images. OCR (Optical Character Recognition) – In other words, OCR systems convert a two-dimensional text image from its graphical representation to machine-readable text. The algorithm is built so that if the pdfplumber library does not recognize the page / text , the **Tesseract - OCR** library is included in the work.

### Preprocessing:

In Version 4 we added transforming similar characters € to euro:

```
text = text.replace("€", "euro").
```

Other directives we use in Version 4 are similar to those that we applied in Version 3.

**Scoring:** When developing Version 4, we used Scoring with mandatory compliance with two requirements. We additionally created **Black\_list** variable - words that should not be present on the pages necessary for our purpose, since they contribute to the extraction of additional pages. In Version 4 these words are: **"fluxo", "separar", "individual", "indivíduo", "ias", "hiperinflacionária", "ifrs"**.

**For\_sure** variable - a word that must be present on the pages required for the search. In this Version 4, this word is **"demonstração"**.

In Version 4 the search for equality of occurrences was also carried out. An algorithm was created that calculates the maximum number of matches from the group that could be found.

If the number of words from the first group is the largest on the specified page, then we write down the document's name and remember how many maximum words matched the group at the moment.

If the number of words from the group is equal to the maximum of those found earlier, then we write the name of this file too.

The results of this algorithm quite effectively identify the pages we need.

```
In [40]: print(documents1) # what files did we find
         print(documents2)
```

```
{'names': ['500101221_2020_ESTORIL_R&C_CC-page-149.pdf'], 'count_unique': 13}
{'names': ['500101221_2020_ESTORIL_R&C_CC-page-150.pdf'], 'count_unique': 12}
```

**Fig. 37 The pages recognition result uses the algorithm that calculates the maximum number the matches words from the Group**



Saved the results in a DataFrame, which is also saved to an xlsx file.

An algorithm has been applied that calculates the number of words found, determined by us for each group, to the total number of words in this group. That is, 'count\_group1 / len(group1)' for Group 1 and 'count\_group2 / len(group2)' for Group 2. An algorithm was written that sorts the pages in Group 1 and Group 2 by the largest number of “Total \_words” in each group. The results of this algorithm quite effectively identify the pages we need.

```
In [48]: df.sort_values(['count_group2 / len(group2)'], ascending=False)
```

```
Out[48]:
```

	page	count_group1 / len(group1)	count_group2 / len(group2)	total_words
5	500101221_2020_ESTORIL_R&C_CC-page-150	0.411765	0.800000	69
23	500101221_2020_ESTORIL_R&C_CC-page-180	0.411765	0.733333	101

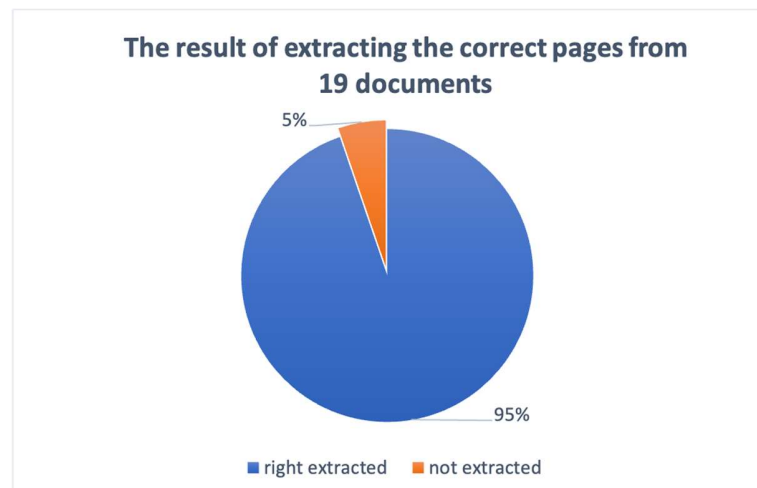
**Fig. 38 The pages recognition result uses the algorithm that calculates the largest number of total \_words in each group**

#### Extract tables and output:

We used Tabula / Pandas to convert data to xlsx format.

Summing up, in the development of the Version 4, we tested the financial statements of the 19 companies. In this particular case, Tesseract - OCR library was not useful to us, because the quality of the documents provided allows pdfplumber to extract the necessary tables and recognize them with the maximum ability inherent in this library. Results of the implemented algorithm Version 4 with the accuracy of finding the correct tables equal 95 %.





**Fig. 39** The diagrams of the accuracy of finding the correct tables using Version 4

## 5.7 Version 5

**Split documents into pages:** we used library pikepdf [21].

**Extract text:**

With code Version 5 we tested the Financial statements of the 20 companies. Of particular interest is the Financial statements of GALP GÁS NATURAL\_R&C\_CC, MOTA-ENGIL\_R&C\_CC, EDP\_R&C\_CC, which is freely available and posted on the Internet. The quality of the placed documents are very poorly. The documents are likely images. With a similar set of the words for each Group, applied in Version 4.

For text recognition, we used the **pdfplumber** library.

To improve the recognition of documents of poor quality, we installed the **Tesseract - OCR** library. We installed the python - poppler. It allows to read, render, or modify PDF documents.

We used Pillow's Image class to open the image and pytesseract to detect the string in the image:

```
pytesseract.pytesseract.tesseract_cmd = 'C:\Program Files\Tesseract-OCR\tesseract.exe'
def ocr_core(img):
    """
    This function will handle the core OCR processing of images.
    """
    text = pytesseract.image_to_string(img, lang='por')
    return text
```

## Preprocessing:

In Version 5 we used similar directives applied in Version 4.

**Scoring:** When developing Version 5, we additionally created **Black\_list** variable - words that should not be present on the pages necessary for our purpose, since they contribute to the extraction of additional pages. Black\_list contains the words "**fluxo**", "**separar**", "**ias**", "**hiperinflacionária**", "**ifrs**".

**For\_sure** variable - words that must be present on the pages required for the search. For\_sure contains the words "**demonstração**", "**balançar**".

In Version 5 the search for equality of occurrences was also carried out similar applied in Version 4.

The results of the code Version 5 (without used OCR) exported in xlsx look like this. An example of uploading the Report EDP\_R&C\_CC in xlsx. Page 234.

Demonstração dos Resultados Consolidados para os períodos findos em 31 de Dezembro de 2019 e 2018			
Milhares de Euros	Notas	2019	2018
Receitas de vendas e serviços de energia e outros	7	14.333.009	15.278.085
Custo com vendas de energia e outros	7	-9.115.859	-10.178.903
		5.217.150	5.099.182
Outros proveitos	8	691.886	562.677
Fornecimentos e serviços externos	9	-897.543	-956.961
Custos com o pessoal e benefícios aos empregados	10	-620.196	-651.540
Outros custos	11	-652.473	-715.379
Imparidades de clientes e devedores	25	-33.207	-20.850
		-1.511.533	-1.782.053
		3.705.617	3.317.129
Provisões	35	-101.530	-287.938
Amortizações e imparidades	12	-1.765.619	-1.444.812
		1.838.468	1.584.379
Proveitos financeiros	13	387.817	456.245
Custos financeiros	13	-1.057.591	-1.010.390
Equivalências patrimoniais em joint ventures e associadas	21	25.011	10.858
Resultado antes de impostos e CESE		1.193.705	1.041.092
Impostos sobre os lucros	14	-225.901	-99.666
Contribuição extraordinária para o sector energético (CESE)	15	-68.477	-65.345
		-294.378	-165.011
<b>Resultado líquido do período</b>		<b>899.327</b>	<b>876.081</b>
<b>Atribuível a:</b>			
<b>Accionistas da EDP</b>		<b>511.751</b>	<b>519.189</b>
<b>Interesses não controláveis</b>	32	<b>387.576</b>	<b>356.892</b>
<b>Resultado líquido do período</b>		<b>899.327</b>	<b>876.081</b>
Resultado por Acção (Básico e Diluído) - Euros	29	0,14	0,14

Fig. 40 Page in PDF format to extract data using the code Version 5

500697256\_2019\_EDP\_R&C\_CC-page-234 - 1

Ficheiro Base Inserir Esquema de Página Fórmulas Dados Rever Ver Diga o que pretende fazer...

Cortar Copiar Pincel de Formatação

Calibri 11 A A

N I S Tipo de Letra

Alinhamento

Unir e Centrar

Geral

\$ % 000 ,00 ,00

Número

Format Condi

V31

	A	B	C	D	E	F	G	H	I
1	NIPC	year							
2	500697256	2019							
3									
4	Milhares de Euros	Notas	2019	2018					
5	Receitas de vendas e serviços de energia e outros	7	14.333.009	15.278.085					
6	Custo com vendas de energia e outros	7	-9.115.859	-10.178.903					
7			5.217.150	5.099.182					
8	Outros proveitos	8	691.886	562.677					
9	Fornecimentos e serviços externos	9	-897.543	-956.961					
10	Custos com o pessoal e benefícios aos empregados	10	-620.196	-651.540					
11	Outros custos	11	-652.473	-715.379					
12	Imparidades de clientes e devedores	25	-33.207	-20.850					
13			-1.511.533	-1.782.053					
14			3.705.617	3.317.129					
15	Provisões	35	-101.530	-287.938					
16	Amortizações e imparidades	12	-1.765.619	-1.444.812					
17			1.838.468	1.584.379					
18	Proveitos financeiros	13	387.817	456.245					
19	Custos financeiros	13	-1.057.591	-1.010.390					
20	Equivalências patrimoniais em joint ventures e associadas	21	25.011	10.858					
21	Resultado antes de impostos e CESE		1.193.705	1.041.092					
22	Impostos sobre os lucros	14	-225.901	-99.666					
23	Contribuição extraordinária para o sector energético (CESE)	15	-68.477	-65.345					
24			-294.378	-165.011					
25	Resultado líquido do período		899.327	876.081					
26	Atribuível a:								
27	Accionistas da EDP		511.751	519.189					
28	Interesses não controláveis	32	387.576	356.892					
29	Resultado líquido do período		899.327	876.081					
30	Resultado por Acção (Básico e Diluído) - Euros	29	0,14	0,14					
31									

**Fig. 41 The result of the data extraction using code Version 5 (without using OCR) and exported in xlsx**

The results of the tested code Version 5 quite effectively identify the pages we need and are shown below and the final results the accuracy are in the diagrams.

```
509148247_2019_GALP GÁS NATURAL_R&C_CC
100% 133/133 [04:31<00:00, 2.04s/it]

In [122]: print(documents1) # what files did we find
          print(documents2)

{'names': ['509148247_2019_GALP GÁS NATURAL_R&C_CC-page-37.pdf'], 'count_unique': 15}
{'names': ['509148247_2019_GALP GÁS NATURAL_R&C_CC-page-38.pdf'], 'count_unique': 14}

In [207]: print(documents1) # what files did we find
          print(documents2)

{'names': ['2020_500978654_VAA_R&C_CC-page-90.pdf'], 'count_unique': 16}
{'names': ['2020_500978654_VAA_R&C_CC-page-91.pdf'], 'count_unique': 14}
```

**Fig. 42 The results of the tested code Version 5 effectively identify the pages we need**

## Demonstração da Posição Financeira Consolidada

Galp Gás Natural Distribuição, S.A.

Demonstração da posição financeira consolidada em 31 de dezembro 2019 e em 31 de dezembro de 2018

(Montantes expressos em milhares de Euros - € k)

Ativo	Notas	2019	2018
<b>Ativo não corrente:</b>			
Ativos tangíveis	5	917	507
Goodwill	9	2.275	2.275
Ativos intangíveis	6	1.175.433	1.077.335
Direitos de uso de ativos	7	13.915	-
Participações financeiras em associadas	10	-	12.506
Ativos por impostos diferidos	17	15.582	16.015
Outras contas a receber	12	28.265	15.047
Outros ativos financeiros	13	6	3
<b>Total de ativos não correntes:</b>		<b>1.236.393</b>	<b>1.123.688</b>
<b>Ativo corrente:</b>			
Inventários	11	1.995	1.695
Clientes	12	11.334	12.093
Outras contas a receber	12	42.714	51.946
Imposto corrente sobre o rendimento a receber	17	2.594	-
Caixa e seus equivalentes	14	42.705	48.107
<b>Total dos ativos correntes:</b>		<b>101.342</b>	<b>113.841</b>
<b>Total do ativo:</b>		<b>1.337.735</b>	<b>1.237.529</b>
<b>Capital Próprio e Passivo</b>			
<b>Capital próprio:</b>			
Capital social	23	89.529	89.529
Reservas		9.454	7.468
Resultados acumulados		108.905	120.324
<b>Total do capital próprio atribuível aos acionistas:</b>		<b>207.888</b>	<b>217.321</b>
Interesses que não controlam	24	19.590	19.519
<b>Total do capital próprio:</b>		<b>227.477</b>	<b>236.840</b>
<b>Passivo:</b>			
<b>Passivo não corrente:</b>			
Dívida financeira	15	674.626	609.270
Responsabilidades por locações	7	13.014	-
Outras contas a pagar	16	220.718	217.400
Responsabilidades com benefícios de reforma e outros benefícios	18	60.295	55.802
Passivos por impostos diferidos	17	20.496	7.272
Provisões	19	65.190	53.316
<b>Total do passivo não corrente:</b>		<b>1.054.340</b>	<b>943.060</b>
<b>Passivo corrente:</b>			
Dívida financeira	15	5.268	8.349
Responsabilidades por locações	7	1.115	-
Fornecedores	16	9.596	11.111
Outras contas a pagar	16	39.940	33.770
Imposto corrente sobre o rendimento a pagar	17	-	4.399
<b>Total do passivo corrente:</b>		<b>55.918</b>	<b>57.629</b>
<b>Total do passivo:</b>		<b>1.110.258</b>	<b>1.000.689</b>
<b>Total do capital próprio e do passivo:</b>		<b>1.337.735</b>	<b>1.237.529</b>

Fig. 43 Page in PDF format (digitalized) to extract data using the code  
Version 5 with using OCR

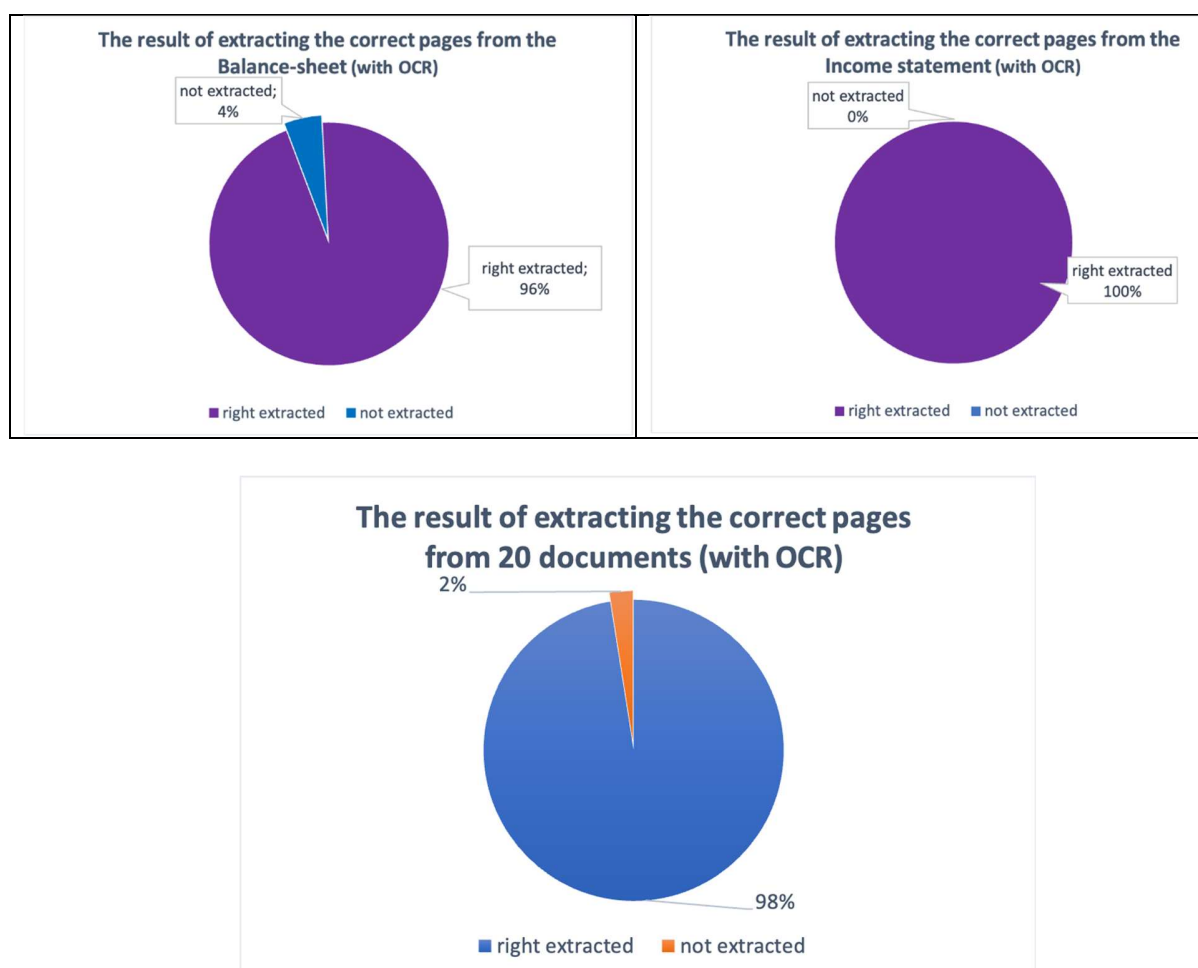


Now, the results of the code Version 5 with used OCR exported in xlsx look like this. An example of uploading the Report GALP GÁS NATURAL\_R&C\_CC in xlsx. Page 37.

509148247_2019_GALP GÁS NATURAL_R&C_CC-page-37 -					
Ficheiro Base Inserir Esquema de Página Fórmulas Dados Rever Ver Diga o que pretende fazer...					
124					
A	B	C	D	E	F
1 NIPC	year				
2 509148247	2019				
3					
4 (Montantes expressos em milhares de Euros - C.k)					
5 Ativo	Notas	2019	2018		
6 Ativo não corrente:					
7 Ativos tangíveis	5	917	507		
8 Goodwill	9	2275	2275		
9 Ativos intangíveis	6	1.175.433	1077.335		
10 Direitos de uso de ativos	7	13915	-		
11 Participações financeiras em associadas	10	-	12.506		
12 Ativos por impostos diferidos	17	15582	16015		
13 Outras contas a receber	12	28265	15.047		
14 Outros ativos financeiros	13	6	3		
15 Total de ativos não correntes:		1 236393	1 123 688		
16 Ativo corrente:					
17 Inventários	11	1 995	1 695		
18 Clientes	12	11334	12.093		
19 Outras contas a receber	12	42 714	51946		
20 Imposto cor rente sobre o rendimento o receber	17	2594			
21 Ca íxa e seus equivalentes	14	42.705	48907		
22 Total dos ativos correntes:		101.342	113.841		
23 Total do ativo:		1 337 735	1 237 529		
24 Capital Próprio e Passivo	Notas	2019	2018		
25 Capital próprio:					
26 Capital social	23	89 529	89 529		
27 Reservas		9454	7 468		
28 Resultados acumulados		108.905	120324		
29 Total do capital próprio atribuível aos ocionistas:		207 888	217 321		
30 Interesses que não controlam	24	19 590	19 519		
31 Total do capital próprio:		227 477	236840		
32 Passivo:					
33 Passivo não corrente:					
34 Dívida financeira	15	674626	609270		
35 Responsabilidades por locações	7	13014	-		
36 Outras contas a pagar	16	220.718	27400		
37 Responsabilidades com benefícios de reforma e outros benefícios	18	60.295	55802		
38 Passivos por impostos diferidos	17	20496	7272		
39 Provisões	9	65.190	53316		
40 Total do passivo não corrente:		1 054 340	943.060		
41 Passivo corrente:					
42 Dívida financeira	15	5268	8349		
43 Responsabilidades por locações	7	1f5			
44 Fornecedores	16	9596	11 111		
45 Outras contas a pagar	16	39940	33770		
46 Imposto corrente sobre o rendimento o pagar	17		4399		
47 Total do passivo corrente:		55 918	57 629		
48 Total do passivo:		1.110.258	1.000.689		

**Fig. 44 The result of the data extraction using code Version 5 with using OCR and exported in xlsx**

Summing up, in developing code Version 5, we tested the financial statements of 20 companies. In this particular case, Tesseract - OCR library was useful, because the quality of the documents provided GALP GÁS NATURAL\_R&C\_CC did not allow pdfplumber to extract the necessary tables and recognize them. With the help of Tesseract - OCR, the correct pages of the required reporting were recognized and extracted. Results of the implemented code Version 5 with the accuracy of finding the correct tables and extracting them equal 98 %.



**Fig. 45 The diagrams of the accuracy of finding the correct tables using Version 5**

It is worth noting that a similar Project was implemented during a student internship at the Banco de Portugal in Lisbon last year. In developing the code, we did not rely on an already implemented project. The algorithm developed by us is intellectual property.

At the heart of the project previously implemented at the Bank of Portugal was the testing of 5 links from the Internet:

### **1. Testing the Tabula library.**

Testing was done to retrieve two different tables using the "tables" expression. Although the code was correct, the program did not recognize the two tables effectively. Thus, the hypothesis of extracting two tables together requires additional testing. And at the same time, it turned out that the intervals between tables and within tables are represented by NaN (cells without value).

During testing, the conclusions of the student were made:

- Tabula is a good library for pdf document analysis, although it presents some shortcomings.
- The conversion of the document from pdf to CSV may be relevant in the search for a set of words or numerical values since it can detect most of the characters.

## 2. Testing Pandas libraries with Tabula.

The student selected a specific area for data extraction. After selecting the area of interest, the pages of the analyzed document were indicated, as well as the document's title.

Reading the list of regions is done through the *read\_pdf()* function and defining the output format in JSON. Thus, a list is obtained, however, it does not contain values. Although the same document was tested as the previous demo, failed to get the expected result.

The final conclusions:

- Extracting data through the box is inefficient and impractical because the same area is defined for all document pages.

## 3. Tesseract-OCR Library

The student was unable to install the library, so it was not possible to test it.

## 4. Testing the Tabula library - py.

Tabula-py allows you to extract tables from PDF to DataFrame, JSON or save the file as CSV or JSON.

The following demo used the tabula library to read a PDF document.

It is normal for errors to appear, since tabula cannot read images, only text.

Similar to what was developed earlier, the *read\_pdf* method on the path is used, which gives access to the document in PDF format. The *dfs2* variable is then output, which includes the double path method.

## 5. Camelot Library

It was impossible to use this library because the installation of Ghostscript is required. During the attempt to install Ghostscript student credentials were placed to which.

## Final Solution:

In this demo, the following libraries were used to parse, extract and read data from a pdf document.

```
In [46]: import csv
         from collections import Counter
         import tabula
         import pandas as pd
         from pandas import DataFrame
         import PyPDF2
```

**Fig. 46 Installing libraries required for analysis**

Variables have been created that will be needed to consider the size of the selected pdf document. Then csv files are created for each page of the document.



After receiving the various CSV, the words that made meaning to find the desired Income statement. It should be noted that all selected words are generic to adapt to each document's words. For example, the word "Deprecia" can be formulated in terms of "Gastos de depreciação" or "Depreciações e amortizações". Thus, the program can read a word that has places or stands both in the plural and in the singular.

The pages that contain the search words are then selected and a sequence is created that will be ordered according to the frequency of each page (rating). The result is the first place in an elaborate ranking.

```
In [21]: total.nlargest(1)
Out[21]: Página 179    18
         dtype: int64
```

**Fig. 47 Selection of the first place of the developed rating**

### Score system:

If all or more than 50% of the words we want to find have a *high frequency*, the system will deduct a point from the page score. Otherwise, if all or more than 50% of the words we want to have a *low frequency*, the system would add a point to the page score.

The accuracy of extracting the required pages from the Financial Statements of the project implemented by **the student from Lisbon was approximately 50%**. At a time when the accuracy of the algorithm **developed by us equals 98%**.

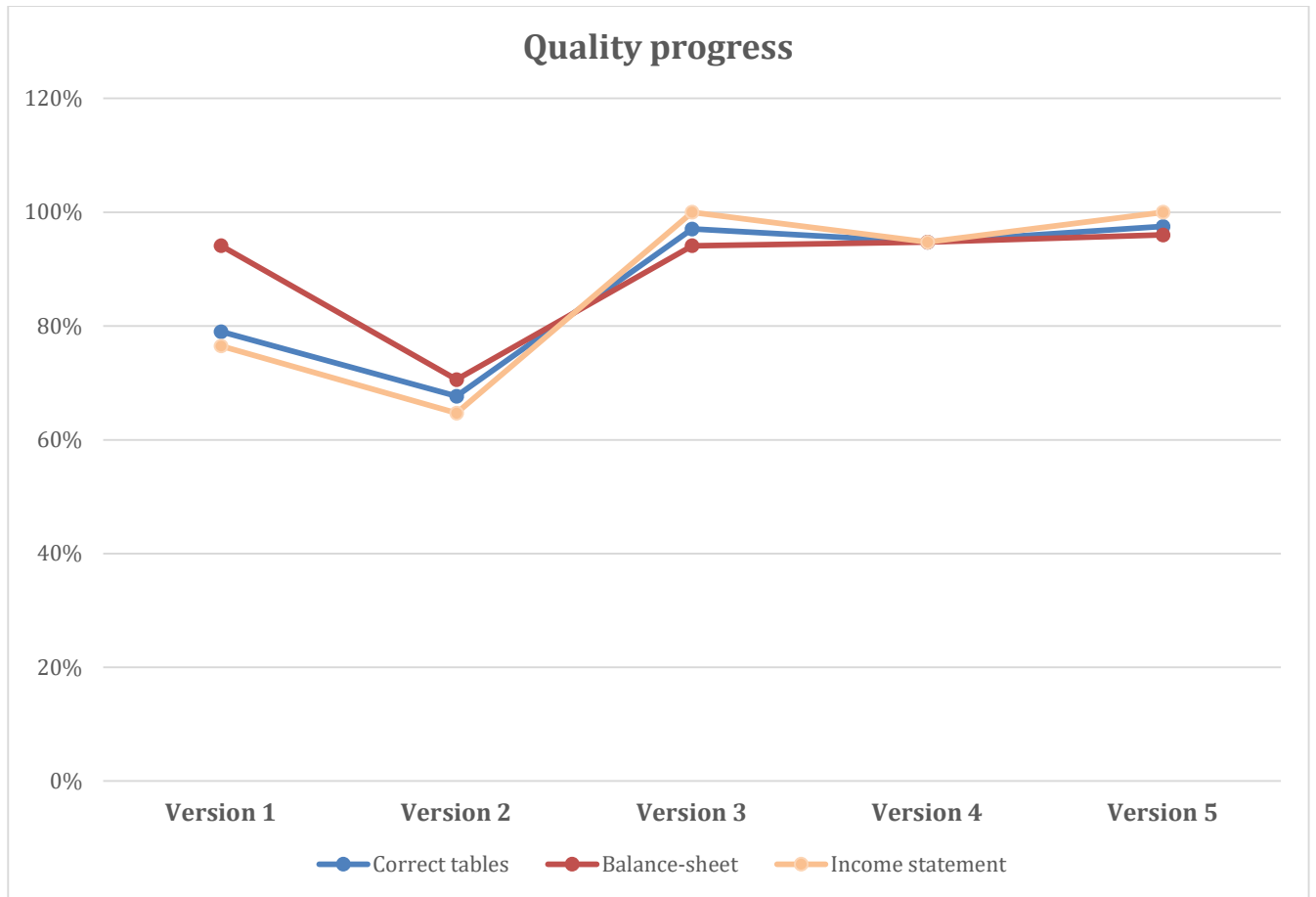
We calculated the accuracy of all developed Versions in the following way, the number of correctly recognized pages was divided by the total number of pages that needed to be recognized and calculated as a percentage.

**High accuracy in achieving the goal is due to the following:** a thorough study of the Python libraries related to working with TM and their huge number of features and tools. We studied Documentation for the library's packages.

Tested the main libraries to achieve the best result, including testing the Camelot library, Tabula, which, when used together, showed a decent result for manual extraction of tables, but this process is labour intensive. For the analysis of TM, preprocessing of the text was carried out to clean up the text and facilitate the extraction of the high-quality and necessary information. In our case, we used: Lowercasing, Replacing numbers with zeros, Transforming similar characters € to euro, Tokenization, Lemmatization, Stop word removal, and Standardization of word usage. We tested Tesseract - OCR, which helped us recognize and extract the correct pages of the required reporting.

Work was done on Scoring of the words, Bag of the words, when in the process of testing the algorithm, relevant words were selected corresponding to the words for Group 1 the Balance-sheet, and Group 2 - Income statement. Scoring was done with different Thresholds. We additionally created the Black\_list variable - words that should not be present on the pages necessary for our purpose and For\_sure variable - words that must be present on the pages required for the search allowed us to achieve the best result. We have worked through all the stages inherent in working with TM documents, in connection with which we have achieved progress and great results.

The dynamics of the development of the implemented algorithms and the progress to extract the necessary tables: Balance-sheet and Income statement are reflected in Fig. 46. The diagram of the quality progress.



**Fig. 48 The diagram of the quality progress**

## 6 | Conclusions and Future Work

Has successfully developed the TM algorithm to extract the required Financial Statement tables with a high accuracy equal 98 %.

We have developed an algorithm to find document pages with a high likelihood of holding tables displaying the information we wish to extract. This is a necessary step in a long information extraction process from tables in the text. We have applied the algorithm in the context of companies' financial statements – mandatory reports published by accounting companies for their activities at the end of the year. The algorithm has been developed by using data analysis techniques and tools from machine learning to text mining on a set of 20 financial statements. Information extraction from such reports is relevant as a support for the decisions of many economic agents, but current approaches are mainly manual and time-consuming.

The basis of the algorithm that we implemented was text preprocessing; searching for relevant words for each group created to extract tables from which then we need to extract information; choosing the right words for the Blacklist, words that we need to exclude, and choosing the right words For\_sure, that must be present on the page. The approach we took to work with extracting information from tables showed excellent results.

In addition, the approach includes some parameters that allow the user to influence the algorithm's performance and further adapt it to the features of the tables that he wants to detect. The performance of the algorithm was measured on a given set of reports, as well as on a separate test set. The results show that we can immediately save much work without losing important information.

Another main advantage of this approach is that any PDF document can be converted to an image, after digitalizing and using optical character recognition software (OCR) that recognizes the text and necessary tables for further information extraction with a high probability of accuracy. Knowing certain keywords and exact words that are contained in Blacklist has the potential to improve the search for required tables to 100% with very low economic costs and will be sought in future work. The remaining tasks for information extraction from tables will also be researched in the future.

It is worth noting that a similar project was implemented during the student's internship at Banco de Portugal in Lisbon last year. The accuracy of the implemented project by another student is **approximately 50 %**. During the development of the code, we did not rely on an already implemented project. The algorithm developed by us is intellectual property.

Thanks to the fact that we have carefully studied the Python libraries associated with working with TM, and with their huge number of functions and tools. We have gone through all the Library Package Documentation and tested the core libraries. After the work we implemented, we achieved the best result by accurately extracting the tables we needed **equal to 98%**.

In the future, it would be interesting to apply other approaches to recognize the tables we need, such as Automatic Selection of Table Areas in Documents, or try to solve our main goal using the KNIME software.

Another important work to be continued in the future is the correlation of the results of this study with officially tested new data from the source. Such associations will be useful both for testing the proposed approach and improving the knowledge inference and extraction process.

# Appendix

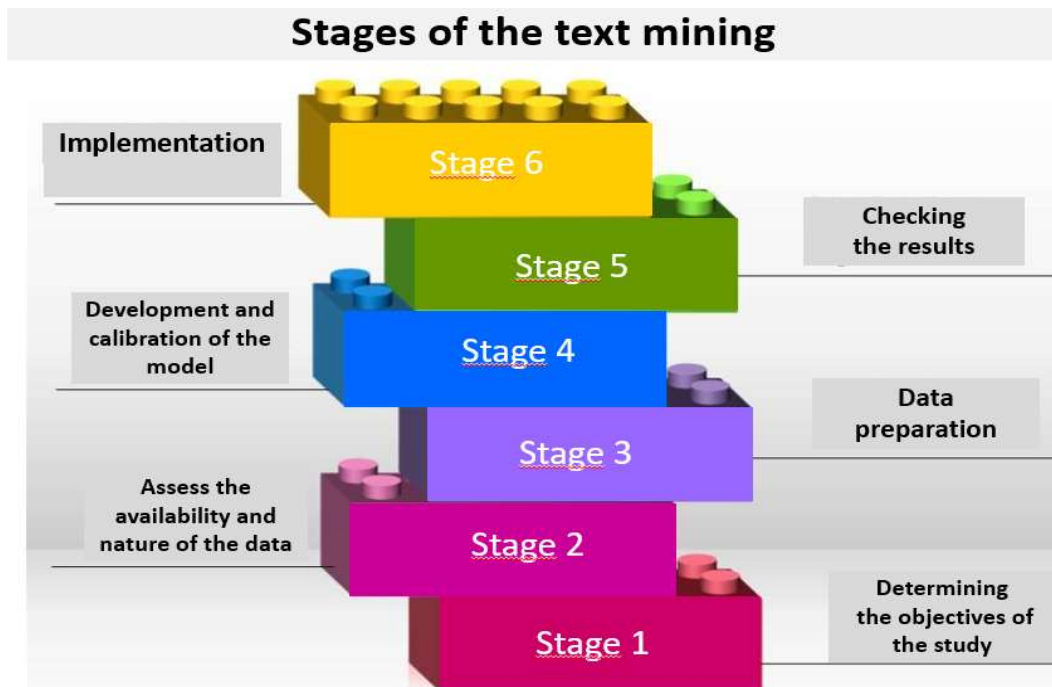


Fig. 1 Stages of the text mining

Header
Body
'xref' Table
Trailer

Fig. 3 PDF elements document

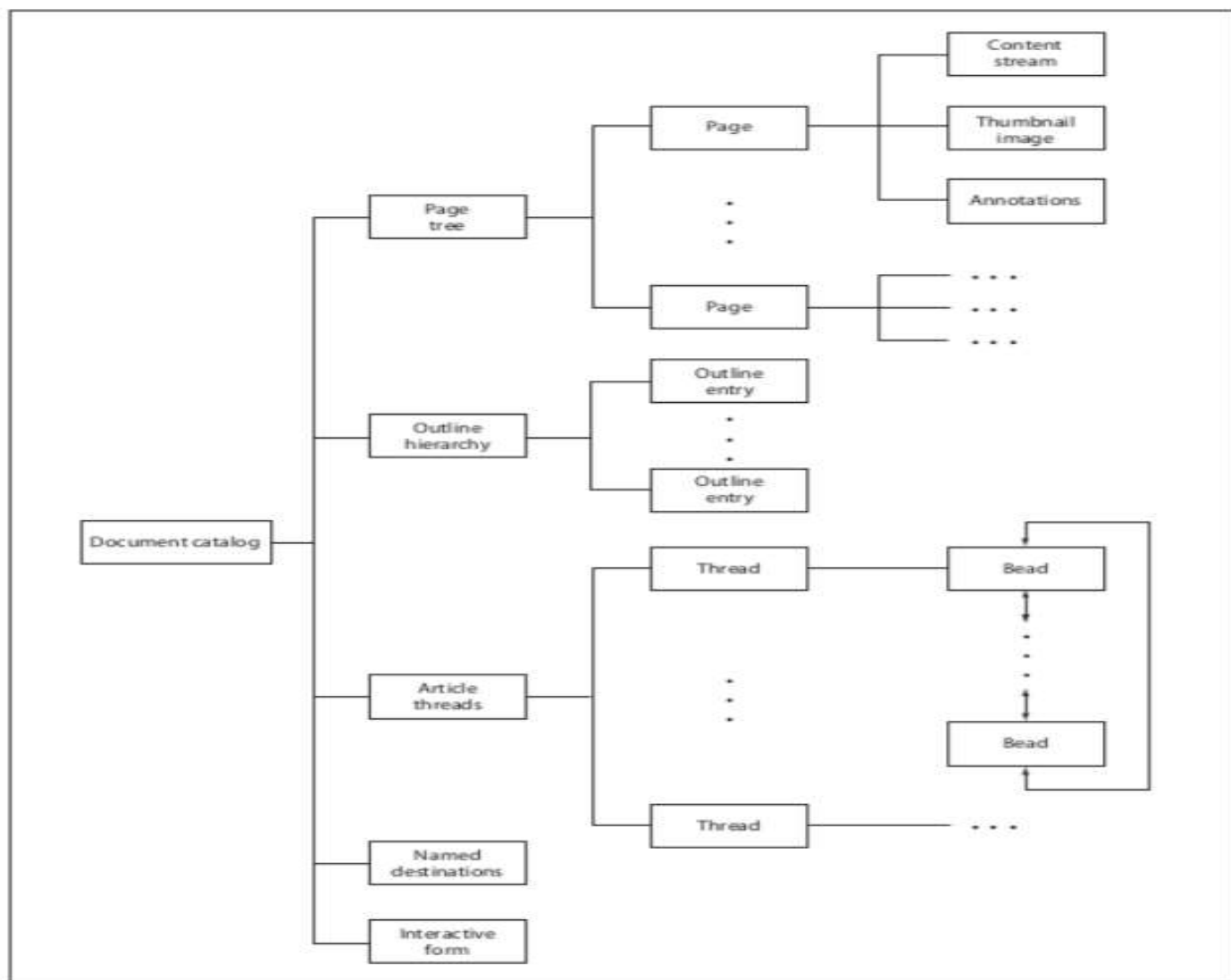


Fig. 4 PDF Document structure

(Montantes expressos em euros)	Notas	dezembro 2020	4º Trimestre 2020 (não auditado)	dezembro 2019 (reexpresso)	4º Trimestre 2019 (não auditado e reexpresso)
Vendas	1s), 30 e 37	76.546.308	15.578.874	83.256.879	17.535.197
Prestações de serviços	1s), 30 e 37	52.540.194	14.004.576	49.141.618	12.262.609
Outros rendimentos	1q), 23, 31 e 37	3.171.635	1.766.925	2.446.082	885.880
		132.258.137	31.350.375	134.844.579	30.683.686
Custo das vendas	1i) e 14	(66.669.523)	(13.099.006)	(71.800.735)	(14.217.962)
Fornecimentos e serviços externos	1h), 32 e 37	(22.457.967)	(5.496.983)	(25.651.248)	(7.233.046)
Gastos com o pessoal	1q), 1y), 42, 43 e 45	(46.400.159)	(12.804.434)	(46.361.557)	(12.072.576)
Amortizações e depreciações	1c), 1d), 1f), 1h), 5, 6 e 7	(8.885.581)	(2.251.520)	(10.185.528)	(3.924.223)
Provisões	1j), 1o), 1x) e 24	(93.293)	23.319	(112.070)	(56.233)
Perdas de imparidade	1j), 1o), 1x) e 24	(143.241)	(77.455)	(404.428)	(244.091)
Outros custos	33	(371.281)	(114.291)	(420.758)	(135.040)
		(145.021.045)	(33.820.370)	(154.936.324)	(37.883.172)
Ganhos e perdas em empreendimentos conjuntos e associadas	1b), 9 e 35	46.031.392	33.718.719	62.851.437	32.127.045
Ganhos e perdas em ativos registados ao justo valor através de resultados	1b), 11 e 35	21.626.448	(7.718.627)	619.935	619.935
Gastos e perdas financeiros	1h), 1m), 1w), 1x), 22, 34 e 37	(3.553.917)	(863.388)	(2.410.473)	(896.404)
Rendimentos e ganhos financeiros	1w), 22, 34 e 37	2.920.559	905.100	2.346.847	815.662
Resultados antes de imposto		54.261.574	23.571.809	43.316.001	25.466.752
Imposto sobre o rendimento	1p), 12 e 36	4.457.255	7.672.721	(6.108.175)	(8.094.407)
<b>Resultado líquido consolidado do exercício das operações continuadas</b>		<b>58.718.829</b>	<b>31.244.530</b>	<b>37.207.826</b>	<b>17.372.346</b>
Resultado líquido do exercício de operações descontinuadas	40	-	-	12.568.216	-
<b>Resultado líquido consolidado do exercício</b>		<b>58.718.829</b>	<b>31.244.530</b>	<b>49.776.042</b>	<b>17.372.346</b>

Fig. 6 Table example from one of PDF files

Household income	'Expected' monthly contribution*	Average monthly contribution	Under- or over-payment
Less than £25,000	£0	£54	+£54
£25,001 - £35,000	£51	£89	+£38
£35,001 - £45,000	£151	£134	-£18
£45,001 - £55,000	£252	£141	-£111
£55,001 - £65,000	£353	£193	-£160
£65,001+	£375	£287	-£88

Fig. 7 Standard example

	0	1	2	3	4
0	x1	x2	x3	x4	y
1	0	0	1	0	0
2	0	1	1	0	0
3	0	1	0	1	1
4	1	0	0	0	0
5	1	1	0	0	1

	0	1	2	3	4
0	x1	x2	x3	x4	y
1	1	1	0	1	?

Fig. 8 Extracted data using the Tabula library from a page containing two tables

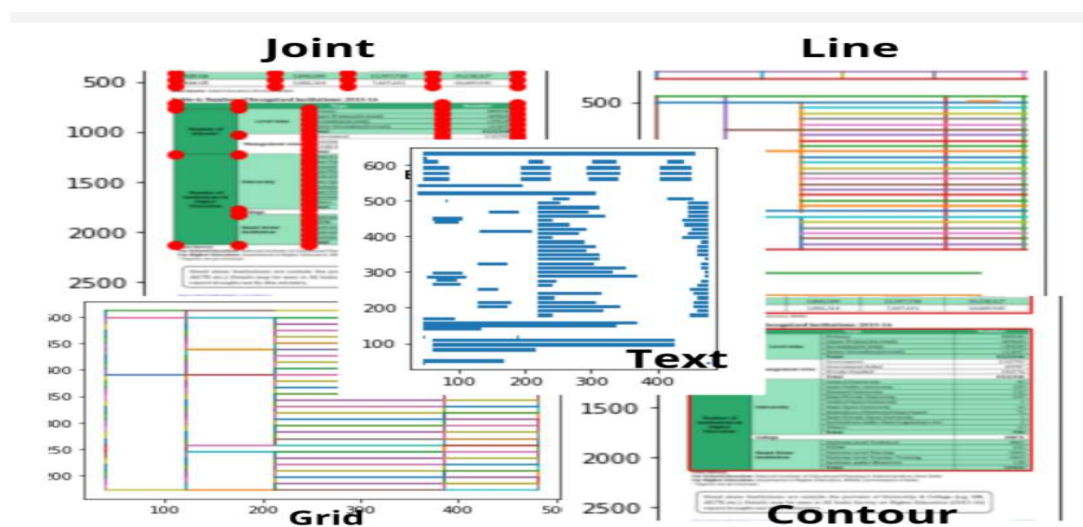


Fig. 10 Element types found on the PDF page



**Fig. 11** Extracted data using the Camelot library from a page containing two tables.



# References

- [1] Barseghyan, A. A., S. I. Elizarov, M. S. Kupriyanov, I. I. Kholod, M. D. Tess. (2009, January 1). *Analysis of data and processes*. St. Petersburg, 512 p. B26. ISBN 978-5-9775-0368-6.
- [2] Shmueli, G., Bruce, P. C., Gedeck, P., & Patel, N. R. (2019). Data mining for business analytics: concepts, techniques and applications in Python. *Data Mining for Business Analytics*. ISBN - 978-1119549840.
- [3] Michael J. A. Berry, Gordon Linoff. (1997, June 10). *Data Mining Techniques*. Jhon Wiley & Sons, Inc.
- [4] Mike Bergman. (2005, November 1). *Semi-structured Data*.  
[https://www.slideshare.net/dtunkelang/scale-structure-and-semantics/17-Semistructured\\_Data\\_Michael\\_K\\_Bergman](https://www.slideshare.net/dtunkelang/scale-structure-and-semantics/17-Semistructured_Data_Michael_K_Bergman)
- [5] Vasyi Holiney (2021, February 28). *PDF format: features, advantages and disadvantages*.  
<https://www.logaster.ru/blog/pdf/>
- [6] Leonard Rosenthol (2013). Developing with PDF: Dive Into the Portable Document Format. "O'Reilly Media, Inc." ISBN– 9781449327910. <https://vdoc.pub/download/developing-with-pdf-dive-into-the-portable-document-format-3fp0df4u4grg>
- [7] Documentation management (2008, July 1). *Portable document format – Part 1*. First Edition 2008-7-7. PDF 32000-1:2008. [https://pdf-lib.js.org/assets/with\\_large\\_page\\_count.pdf](https://pdf-lib.js.org/assets/with_large_page_count.pdf)
- [8] Tutorials, Tools, Scripts, and Samples for scripting PDF. *Documentation for the PDF Page Coordinates (page size, field placement, etc.) package*.  
<https://www.pdfscripting.com/public/PDF-Page-Coordinates.cfm>
- [9] Ana Costa e Silva, Alípio Mário Jorge, Luís Torgo. (2006, January). Design of an end-to-end method to extract information from tables. *International Journal on Document Analysis and Recognition*. 8(2):144-171. DOI: 10.1007/s10032-005-0001-x.  
(PDF) [Design of an end-to-end method to extract information from tables \(researchgate.net\)](https://www.researchgate.net/publication/220211111_Design_of_an_end-to-end_method_to_extract_information_from_tables)
- [10] Andrey Shirshov (2014, October 3). *Parse in Python*. <https://habr.com/ru/post/239081/>
- [11] Tabula – py. Read tables in a PDF into DataFrame. *Documentation for the Tabula-py library package*.  
[https://tabula-py.readthedocs.io/en/latest/tabula.html#tabula.io.build\\_options](https://tabula-py.readthedocs.io/en/latest/tabula.html#tabula.io.build_options)

- [12] Jeremy Singer-Vine (2021). *Plumb a PDF for detailed information about each char, rectangle, line, et cetera — and easily extract text and tables.*  
<https://github.com/jsvine/pdfplumber>
- [13] Sebastian Bassi (2007). *A Primer on Python for Life Science Researchers.*  
<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.0030199>
- [14] Kalyani Adawadkar. (2017). Python Programming-Applications and Future. *International Journal of Advance Engineering and Research Development*, 4.72. e-ISSN : 2348-4470/p-ISSN : 2348-6406.  
[https://scirp.org/\(S\(lz5mqp453edsnp55rrgjet55\)\)/reference/referencespapers.aspx?referenceid=2739833](https://scirp.org/(S(lz5mqp453edsnp55rrgjet55))/reference/referencespapers.aspx?referenceid=2739833)
- [15] Yogesh Rana. (2019, February). Python: Simple though an Important Programming language. *International Research Journal of Engineering and Technology (IRJET)*, e-ISSN: 2395-0056/p-ISSN: 2395-0072. ISO 9001:2008. <https://www.irjet.net/archives/V6/i2/IRJET-V6I2367.pdf>
- [16] Siddharth Sachdeva (2021, September 2). *PyPDF2 Library for Working with PDF Files in Python.* <https://www.analyticsvidhya.com/blog/2021/09/pypdf2-library-for-working-with-pdf-files-in-python/>
- [17] Tabula. High level interface. *Documentation for the Tabula-py library package.*  
[https://tabula-py.readthedocs.io/en/latest/tabula.html#tabula.io.build\\_options](https://tabula-py.readthedocs.io/en/latest/tabula.html#tabula.io.build_options)
- [18] Tejas Thakar. (2017, 2 August). *Tabula extract tables by area coordinates.*  
<https://stackoverflow.com/questions/45457054/tabula-extract-tables-by-area-coordinates>
- [19] Camelot: *PDF Table Extraction for Humans.* <https://camelot-py.readthedocs.io/en/master/https://ichi.pro/ru/izvlechenie-tablicnyh-dannyh-iz-pdf-fajlov-stalo-prose-s-camelot-141567380147336>
- [20] Vinayaka Mehta (2018, March 5). *Extracting tabular data from PDFs just got easier with Camelot.* [https://www.education.gov.in/sites/upload\\_files/mhrd/files/statistics-new/ESAG-2018.pdf](https://www.education.gov.in/sites/upload_files/mhrd/files/statistics-new/ESAG-2018.pdf)
- [21] Nikolay Pavlyuk (2021, December 14). *Main tasks of PDF parsing.*  
<https://tproger.ru/articles/osnovnye-zadachi-parsinga-pdf/>

- [22] McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython* " O'Reilly Media, Inc."
- [23] Mueller A. (2017). *Introduction to machine learning with Python. Data Scientist's Guide*. Moscow Vilyams.
- [24] Pikepdf. *Documentation for the pikepdf library package*. [pikepdf Documentation — pikepdf 5.4.1.dev17+g9f5fa86 documentation](#)
- [25] PDFMiner. Python PDF parser and analyzer. *Documentation for the pdfminer library package*.  
[https://pdfminer-docs.readthedocs.io/pdfminer\\_index.html](https://pdfminer-docs.readthedocs.io/pdfminer_index.html)
- [26] PDFminer.six. *Documentation for the pdfminer library package*.  
[Welcome to pdfminer.six's documentation! — pdfminer.six \\_\\_VERSION\\_\\_ documentation \(pdfminersix.readthedocs.io\)](#)
- [27] Stanislav Pankevich. (2021). A python module that wraps the pdftoppm utility to convert PDF to PIL Image object.  
[GitHub - Belval/pdf2image: A python module that wraps the pdftoppm utility to convert PDF to PIL Image object](#)
- [28] Pdf2image. *Documentation for the pdf2image library package*.  
<https://pdf2image.readthedocs.io/en/latest/index.html>
- [29] Matthias A Lee, Emilio Cecchini, Igor Marques. (2019). *A Python wrapper for Google Tesseract*. [https:// GitHub - madmaze/pytesseract: A Python wrapper for Google Tesseract](https://github.com/madmaze/pytesseract)
- [30] SpaCy. *Documentation for the spaCy library package*. [spaCy · Industrial-strength Natural Language Processing in Python](#)
- [31] NLTK. *Documentation for the NLTK library package*. [NLTK :: Natural Language Toolkit](#)
- [32] John Bauer Christopher D. Manning, Mihai Surdeanu. (2014). *The Stanford core NLP natural language processing toolkit*. DOI: [10.3115/v1/P14-5010](https://aclanthology.org/P14-5010/)  
<https://aclanthology.org/P14-5010/>
- [33] Grigory Pyatnitsky-Shapito (1989, August 29), *Detroit, MI. Workshop on Knowledge Discovery in Data (KDD-89), held at IJCAI-1989*.
- [34] Bykov, K. V. (2021, December 30). Features of data preprocessing for machine learning. *Text direct in Young scientist. - 2021. - No. 53 (395)*.