Automated Data Analytics to provide Business to Business Customer Insights

T. J. Langhout
S. A. J. van Leeuwen
C. I. Ort
W. S. Volkers

Supervisors:

prof. dr. ir. D. H. J. Epema
ir. M. Kerkhof
drs. L. Gunneweg
drs. T. Boevink

pwc

TU Delft

B2B CUSTOMER ANALYSIS TOOL

Bachelor End Project Final Report

by

Tjard Langhout
Sander van Leeuwen
Cherwin Ort
Bas Volkers

Supervisors: prof. dr. ir. D. H. J. Epema (TU Delft)
drs. L. Gunneweg (PwC)
drs. T. Boevink (PwC)
ir. M. Kerkhof (PwC)

June 2020

PREFACE

This is the bachelor's thesis of Tjard Langout, Sander van Leeuwen, Cherwin Ort and Bas Volkers as part of the Bachelor Computer Science and Engineering at the Delft University of Technology. Over the course of eleven weeks we have worked with the Deal Analytics branch of PwC Amsterdam to create an application that that applies data analysis on transactional data of companies.

We were helped by many people in realising this project. We received a lot of feedback and continuous help from every member of the PwC Amsterdam Deal Analytics branch, which allowed us to optimise the analyses and user interface.

We would like to thank a few members that helped in the process. Specifically, we would like to thank Lise Gunneweg and Thijs Boevink for the daily meetings and for always being ready to help. Furthermore, Martin Kerkhof guided us during weekly meetings and made sure we always had our priorities in order.

Finally, a special thanks to our supervisor, Prof. Dick Epema, chair of the Distributed Systems group at the Delft University of Technology. He has guided us in keeping the project on the right track and has provided clear feedback on all steps of the process.

Tjard Langhout
Sander van Leeuwen
Cherwin Ort
Bas Volkers
Delft, June 2020

CONTENTS

SUMMARY

The Deal Analytics group of PricewaterhouseCoopers Amsterdam has requested a tool for automatising the business-to-business customer analysis. This analysis was performed manually, which left room for performance improvement.

This report discusses how the final product was developed. After two weeks of initial research, a complete system was designed and implemented in the subsequent nine weeks.

The tool consists of two distinct parts, a front-end and a back-end. The front-end allows the user to customise the analysis to its own preferences, and communicates with the back-end to efficiently perform the analysis. With the help of user evaluations, the front-end has been designed such that it is usable by any PwC employee within the Deals branch.

The back-end uses data analysis techniques and machine learning to analyse customer behaviour. Strong points and growth opportunities of a company are found using techniques such as customer segmentation, regression analysis, and cross-sell analysis.

The product has been tested using a variety of techniques to ensure that the software does not crash on unexpected input. The final product is evaluated based on the requirements, design goals and success criteria set at the start of the project and can be considered successful.

1 INTRODUCTION

In order to complete the Bachelor Computer Science and Engineering, students have to create a product with a company over the course of eleven weeks. This project, named the Bachelor End Project (BEP), allows students to get practical experience in software development and apply what they have learned during their bachelor. A group of four students is tasked with finding a company that has a request for a product for which no off-the-shelf solution exists yet.

During the start of the second semester of 2019/2020, we have come in contact with Thijs Boevink from the Deal Analytics branch of PwC Amsterdam. After meeting online with Lise Gunneweg and Martin Kerkhof, also members of the Deal Analytics branch, we were excited to apply our knowledge to the problem PwC presented: automatising the business to business (B2B) customer analysis.

1.1 B2B CUSTOMER ANALYSIS

The Deal Analytics group, a relatively new branch within the Deals branch of PwC Amsterdam provides data-driven insights to improve decision making and value creation throughout the entire deal lifecycle in the context of Mergers and Acquisitions (M&A). One of the data analyses they perform is the B2B customer analysis, where the growth potential of a B2B company is analysed. This growth potential is determined by providing insights into the value of the client and its customers, cross-selling opportunities and new customers of customer segments.

PwC can perform B2B customer analysis for both the buyer and the seller side of a(n) (M&A) process. When a company wants to sell, they want to let potential buyers in on the growth potential of the company. This is beneficial for the seller as providing an accurate estimate of the growth potential may increase the value of the company.

After a letter of intent has been signed by the buyer, indicating they have a serious interest in buying the company, the buyer performs financial due diligence on the selling company. During this due diligence stage, the buyer may also gain access to the transaction data of the seller. In this stage, PwC can also perform the B2B customer analysis for the buyer. The buyer obtains a good overview of the strengths and weaknesses of the company on sale, including where potential growth opportunities lie. The buyer uses this overview to determine how much the company is worth to them.

After the deal has been made, PwC can use the analysis to assist in the strategy for value creation. But also outside of an M&A context, when a company wants to increase its revenue they can also make use of the customer analysis. Since there are many situations in which the B2B customer analysis is relevant, it is important that the analysis is performed quickly with accurate results.

Currently, B2B customer analysis is done mostly manually. Different programs and data analysis tools are combined by hand which can make performing the analysis a tedious activity. By automating this process, the efficiency of generating insights in the deal lifecycle can be improved and the Deal Analytics employees can provide their services in larger volume and/or with more quality. Automatising the labour intensive steps of the process allows the focus to shift from performing the analysis to interpreting the results.

1.2 STRUCTURE OF THIS DOCUMENT

The purpose of this report is to describe how, during the eleven weeks of the project, a tool was developed that automatises the B2B customer analysis.

Before the tool could be developed, two weeks were devoted to researching the problem and designing the product. During this phase, all requirements were set and for each step of the analysis, the optimal solution was determined. However, before any requirements could be formulated, the problem had to be analysed such

that it was clear what exactly was expected of the team. After the problem was sufficiently analysed, the research and design phase could begin.

The result of the research phase was the research report. The original research report is available in Appendix A. During the course of the project, the design of the final product had to be slightly altered a few times. The final design of the complete system is described in Chapter 2.

How the design was implemented is described next. The architecture consists of two main parts, a user interface, and software running the analyses. The implementation of the user interface is set out in Chapter 3 and the implementation of the analyses software is explained in Chapter 4. Next, the tools used to develop the application and the general development process are described in Chapter 5, and hereafter the evaluations of the success criteria, design goals and requirements are reported in Chapter 6.

In the discussion in Chapter 7 the project will be reflected upon, looking at the setbacks, alternative approaches and future recommendations. Finally, in the conclusion in Chapter 8, the success of the complete project is discussed.

2 DESIGN

This chapter will discuss the design process behind the final solution to the problem. The design goals formulated at the start of the project, the requirements that were derived based on the design goals and design patterns that will be used in code design are discussed.

2.1 DESIGN GOALS

Design goals are targets that are considered while designing, implementing and testing the solution to the problem. The main goals that were set for this project are usability, reliability and privacy.

2.1.1 Usability

Usability is the first main design goal of the project. The tool should not only be usable by the Deal Analytics group, but also by other groups within the Deals branch of PwC. It is therefore important that the user can use the software without extensive knowledge about data analytics. The tool should be self-explanatory and the analysis executable with default settings. To test the user-friendliness and guide the design of the system, user evaluations have been held. More details are described in Section 5.4.3

Another aspect of usability is a consistent data format between input and output data: the original structure of the client's input data should be kept intact. This is important because it ensures the user understands the format of the output data and can continue working with this data. As described in Section 4.10, there will be a single excel file as output.

2.1.2 Reliability

Besides usability, another key aspect of the project is reliability. Reliability has two meanings for this project. First, it answers the question of how reliable the output and conclusion of the data analysis is. If the data analysis is performed on a lacking data set, the conclusions of the analysis are not reliable. In every step of the analysis, the quality and quantity of the input data will have to be kept in mind. The user should be made aware of the quality of the uploaded data and given an indication of the reliability of the analysis.

Furthermore, another interpretation of reliability is how reliable the tool itself is. The tool must not crash on input the user gives, either through interacting with the interface itself or through uploading data. It is therefore important to perform extensive input validation to ensure the tool stays running. Another advantage of performing extensive input validation is the increase in the security of the system. Besides the tool not crashing on user input, the tool also should not crash during any step of the analysis process. Instead, errors should be displayed to the user and steps of the analysis should be skipped if needed.

2.1.3 Privacy

Privacy is the last main design goal of this project. The user uploads sensitive client data, which should not be distributed further. In order to preserve the privacy of the data, the data should only be stored temporarily in server memory. Both the input data and the results of the analysis are not stored outside of the tool's memory. An aspect of this design goal to consider is that reloading the web page removes any progress on the analysis. This aspect has to be kept in mind in designing the system.

2.2 REQUIREMENTS

After the design goals were formulated, both functional and non-functional requirements were set. The functional requirements are split into four categories us-

ing the MoSCoW method: *Must Have*, *Should Have*, *Could Have* and *Won't Have*,,,,,,,,,,,,,,,,,,,,,. Within these categories, the requirements are split into six subcategories: Interface, Data, Matching, Analysis, Output, Testing and Error Handling.

The original aim of the project was to implement all requirements stated in the *Must Have* and *Should Have* categories. Furthermore, the idea was that in case of a major setback, at least everything in *Must Have* category should be accomplished. Additionally, if the project was finished earlier than the due date, then the requirements under *Could Have* could also be implemented.

The *Must Have* requirements, therefore, include a functional web-page that is able to send data to a back-end to execute a subset of the analyses on provided client data. The *Should Have* requirements contain more interesting analyses and a more intuitive user interface. Finally, the *Could Have* requirements contain analyses that were deemed very time consuming and something beyond the original scope of the project. A full list of requirements can be found in Appendix B. An evaluation of the final product with respect to the requirements is done in Chapter 6

2.3 SUCCESS CRITERIA

Before the development of the project could begin, success criteria had to be defined. When all success criteria have been met in the final product, the project can be considered 'done' and successful.

The project can be considered a success when the system complies with the three main design goals as described in Section 2.1 and at least the *Must Have* requirements are implemented. Additionally, the greater part of the *Should Have* requirements should be implemented such that the tool will actually be used in the deal life cycle by PwC employees of the Deals branch.

2.4 HIGH-LEVEL OVERVIEW

The application will be presented as a web-based tool. A user can surf towards the website of the tool and upload its client's data after which this data is sent for analysis to servers. These servers will retrieve information about the customers from other API's, and perform analyses. The main advantages of this setup are discussed in Section 2.5.

The application will have two distinct parts. The first part is what the user interacts with. This is called the front-end or User Interface (UI) and runs on the computer of the user. The front-end is a website which the user can load when connected to the PwC network. In the front-end, the user can start, monitor, and review the results of the analysis. This part is elaborated upon in Section 3.

When the user submits data of a client to be analysed, this data is sent to the back-end which is a program running on a cluster of servers. The submitted data consists of information about the client and its customers: the records of sales between the client and the customer, when something was sold, and for how much. This submission and the subsequent processing steps are visualised in Figure 2.1.

After the data has been received in the back-end, the first step is to preprocess the data before any data analysis can be applied. The purpose of this step is to sufficiently prepare the data for further steps. The preprocessing step is explained in more detail in Section 4.4.

The next step is to identify existing customers from the transaction data. This means that the customers of sales from different records have to be compared against each other to find which sale belongs to which customer. For example: *Company*, *COMP-ANY* and *Company B.V.* should all be considered the same company. This step is called customer identification and is explained in Section 4.5

After unique customers are identified, extra information about each customer is queried from Orbis[1] in the 'matching' step. This extra information includes characteristics such as the address, size and industry of the customers. The matching step is further elaborated in Section 4.6.

After extracting this information from the Orbis database, three additional interesting customer statistics are collected, using the internal transaction data and API's. These statistics are the distance between the client locations and the customer locations, the lifetime value of each customer, and general industry statistics of the customers. How these statistics are calculated is described in Section 4.7.

1 https://orbiseurope.bvdinfo.com/

After the data collection step is finished, insights into the customers and their purchasing behaviour can be generated. This includes calculating the customer concentration, identifying the characteristics of valuable customers and segmenting the customers into groups, as well as analysing the cross-sell potential by finding products commonly bought together and creating recommendations for existing customers. The methods used for these calculations are elaborated on in Sections 4.8 and 4.9, respectively.

Finally, the generated insights are formatted into tables stored in a single Excel file. This Excel file is then returned back to the front-end, where it is available for the user to download. How this file is generated is described in Section 4.10.

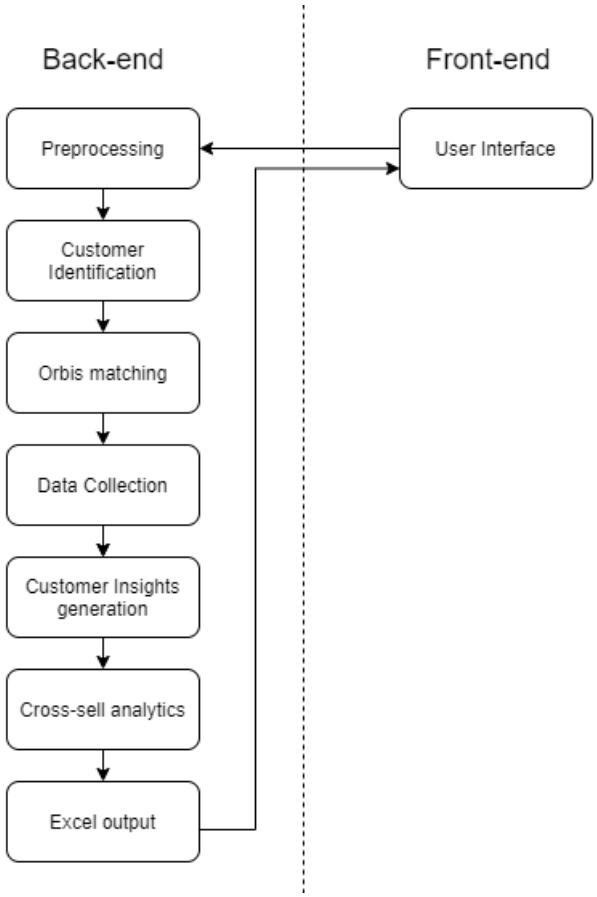


Figure 2.1: Flowchart at high-level

2.5 DESIGN PATTERNS

Design Patterns are used in the software of this application o ensure the architecture of the application is clear, in some way standardised, and easily maintainable. A design pattern is a general and reusable solution to a commonly occurring problem, within a given context in software design. For this project, four different categories of design patterns are considered: architectural, creational, structural and concurrency patterns.

2.5.1 Architectural patterns

Architectural patterns are a special category of design patterns, as they consider the scope of the entire project and not only certain aspects. In this project, two architectural design patterns were considered while designing the code: the Client-server and the Model-view-controller pattern.

The Client-Server pattern consists of two components: multiple clients and a single server. The server component of the application provides concurrent services to multiple clients. There are three main advantages of a client-server architecture in this project:

1. The computation of analysis is off-loaded to faster servers.

2. The risk that the application spreads outside the walls of PwC is reduced.

3. The application can be included in existing PwC web-based tools.

These advantages make a web-based tool a more efficient solution to the problem than other alternatives.

The Model-View-Controller design pattern splits the logic of the application into three interconnected parts. The first part is the model, this is the data about the client and customers which is to be analysed. The second part is the view, this will be represented by the interface the user sees, in the form of a web-page. When the model changes, the view should change: when there are new insights found in the client data, this should be shown on the web-page. These two parts are connected by a controller. The controller handles changes made by the view and projects them onto the model. The controller communicates between front-end and back-end, and matches, expands, and analyses the model.

2.5.2 Creational patterns

Creational design patterns deal with how objects are created and maintained. The main goal of creational design patterns is that objects are created in a flexible and controllable manner. Two creational design patterns are used in this project: The Factory method pattern and the Singelton pattern.

The Factory Method design pattern uses factory methods to create subclasses of a superclass, without specifying the exact subclass. This is useful when different subclasses contain the same methods and purpose, but use different means to achieve this purpose. During run-time, the most applicable subclass can dynamically be created using the factory methods. An example in our code is how internal data is matched with data from Orbis: a factory is used to either match using the Orbis API, an SQL server with a shallow copy of Orbis NL, or a local .CSV file with a shallow copy of Orbis NL.

The Singleton design pattern ensures a class has only one instance and provides global access to the instance of that class. This is useful when exactly one global instance of the class is needed in the system. An example is how progress is tracked: all results are stored in one variable. It should not be possible to make a second instance of this variable, because then the results would be spread across those variables, hiding some results.

2.5.3 Structural patterns

Structural design patterns deal with organising classes and objects in a way that allows realising complex relationships among different classes and also provide tools to add new functionality to existing classes. In this project, two structural design patterns are relevant: the Composite pattern and the Decorator pattern.

The Composite design pattern is a structural design pattern that uses a tree-like structure for objects. It allows the composition of objects into a tree structure, where composite objects behave similarly as if they were individual objects. The interaction with these objects is the same, whether they are composite or individual objects.

The Decorator design pattern allows the addition of new functionality to an existing class, without changing the structure of that class. The Decorator acts as a wrapper for the existing class. New functionality can easily be added without having to bother with the structure of the original class. The API shows the usage of this design pattern: when the client requests data from the API, the API dynamically adds functionality to the subroutine handling the request, to check if the client is a registered PwC employee.

2.5.4 Concurrency patterns

Concurrency design patterns deal with the prevention of the problems that often occur when using multiple threads in a program. Three concurrency design patterns are used in the project: the Thread Pool, Asynchronous Method Invocation, and the Read-Write lock.

The Thread Pool design pattern maintains multiple threads that wait for tasks that can concurrently be executed. Maintaining a thread pool allows for faster per-

formance since the frequent creation and destruction of threads is avoided. This pattern is used when an analysis is started: this is done by using one of the threads in the pool.

The Asynchronous Method Invocation design pattern ensures that a client can start a process at a server, without blocking the actions of the client. This means the client is immediately informed a process is started but does not yet get results in this response. The client can request the results later. Its synchronous counterpart would leave the client waiting in a blocking state, unable to perform other actions. This is implemented in our tool by starting the analyses in the back-end upon request of the front-end, and providing the front-end with instructions on how to follow the progress of the analyses.

The Read-Write lock design pattern allows concurrent read access to an object while restricting the write operations on an object. This means an exclusive lock is needed when modifying data. When a writer is modifying the data of an object, all other access to the object is locked until the writer is done. This is important to ensure the data of the object is always valid and the object is consistent across different clients. An example is the previously mentioned variable storing the analysis results. As multiple requests are handled in parallel, they all want access to the same analysis results. This pattern ensures only one process has access to this variable at a time.

3 FRONT-END IMPLEMENTATION DETAILS

The front-end handles the interaction with the user. The front-end is presented as a website, which the user can load using their web-browser. This means all front-end code is executed on the machine of the user, in contrast to the code of the back-end, which is executed on a server.

The challenge of the front-end is to translate difficult analyses with a variety of input data and settings, to a comprehensible layout and flow which a user can immediately grasp. Because of this, the front-end is divided into six steps.

The tool should feel familiar to the end user. As the end user is a PwC employee, the tool is made using the PwC appkit. This appkit defines the styles for text, buttons and input fields, which are the same all over PwC.

3.1 COMPONENT HIERARCHY

The software is built into small modules with each their own purpose, so the software is maintainable, testable and comprehensible. The front-end is built on composition, rather than inheritance seen in other programming languages. This follows the React guidelines[1]. Composition means that object A 'owns' object B, and B cannot exist without A, but A can exist without B. Inheritance indicates that object B *inherits* traits from object A, like a child inherits from its parents. Composition suits this system better than inheritance, as different components are not similar, but use each other in their layouts.

The way these components interact is visualised in appendix C.7. Each component is visualised as a box with four parts. This is not a standard UML[2] class diagram. The boxes mean the following:

1. **Name** of the component.
2. **Arguments** of the component. These are the variables and functions the component gets when it is used by another component. The arguments prepended by an asterisk are required for basic functionality of the component, the others are optional.
3. **Variables** defined within the component and their type. For example: *columns: Array<String>* is a variable named columns which is an array of strings.
4. **Functions** defined within the component to carry out tasks.

The front-end is divided into four modules: Initialization, adding data, setting preferences and viewing results. In React, the code starts in the *App* component, also called the root. This is visualised in Figure 3.1. This component initializes the possibility to show notifications and includes the *StateFinder* component. The *StateFinder* decides in which step the user currently is. More information about these steps is described in Section 3.2.

To ensure every step has the same basic layout, the *Navigator* component is created. This component defines one **Next** button, one **Back** button and between them a panel with the title of the step. Every component directly included by the *StateFinder* includes the *Navigator* at its root, and specifies the layout the *Navigator* should have within the panel.

3.2 INTERFACE FLOW

The user is guided through the process of adding data, setting preferences, and following the results in six steps. The user can navigate itself through these steps by using the **Next** and **Back** buttons.

1. **Intro** (Figure C.1). The component *LandingPage* is the first page the user sees after loading the tool. The user is informed of the capabilities of the tool and

1 https://reactjs.org/docs/composition-vs-inheritance.html
2 UML: Unified Modeling Language, standard to software architecture and process visualisation.

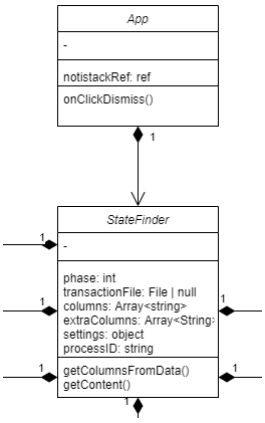


Figure 3.1: Root components of the front-end code.

what data is needed. When the user clicks *Next*, the *LandingPage* component calls the *setPhase()* argument it got from *StateFinder*. This *setPhase()* function tells *StateFinder* to go to the next step.

2. **Enter Client Data** (Figure C.2). The component *ClientData* is shown. This component gives the user the possibility to enter data about the company under investigation and to set the keys of the API's which can be used to gather more data about the company's customers. The choice was made to not provide these API keys in the back-end, because every project within PwC has its own billing, and should buy their own API key.

   The resulting client data is stored in the *StateFinder*.

3. **Upload File** (Figure C.3). The component *AddFile* handles the uploading of a file and checks if it is a correct .csv file. If so, it passes the file to the *StateFinder*, which extracts and identifies the supplied columns and separates them into known and unknown columns. The *StateFinder* passes these columns via the *AddFile* component on to the *AnalysesPrediction* component, which predicts which analyses can be done based on the identified columns, and shows these analyses in a *DropDownList*. The unknown columns are shown in a list by the *ExtraColumnsView* component.

4. **Preferences** (Figure C.4). Using the component *Settings*, the user can set the preferences for the analyses to be done. This is divided into five parts:
   - Using the *SelectBox*, the user can choose what date format (e.g. DD-MM-YYYY or MM/DD/YYYY) the fields of the data that represent a date are in.
   - The *CustomerValueRadios* allows the users to base the value of a customer either on revenue or profit, if the column *customer_value* is not specified in the data. The customer value is necessary, as a target variable in the regression analysis.
   - Which columns to take into account with segmentation and regression can be set using the *CheckColumnTable*. This shows a table with possible columns to include in these analyses, with checkboxes to add or remove them from the analyses.
   - If unknown columns are specified, and the user wants them to be used in the segmentation or regression analyses, the user must choose the data type of the field. This is done with the *ColumnsTypeTable*. It shows the name of each unknown column next to a *SelectBox* with the options: text, number, yes/no and date.
   - Next, the settings for the customer segmentation analysis are can be edited in the *SegmentationSettings*. This includes how whether to segment to a fixed number of segments, a maximum of of the segments or the columns to group on before the segmentation.
   - Lastly, the parameters for the cross-sell analysis can be edited. This is done using the *CrossSellSettings* component which shows four text inputs.

When the user clicks next, the settings are passed to the *StateFinder*.

5. **Progress** (figure C.5). The component *FollowProgress* uploads the transaction file and the settings to the back-end and receives a token from the back-end. This token identifies the process of running the analyses. *FollowProgress* uses this token to get updates about the progress of the analyses. The progress is shown to the user using loading bars. The token is remembered by the *StateFinder*. Any errors which occur are shown to the user by logging to an *ErrorList*. When the back-end has finished its analyses, the user is automatically forwarded to the last step.

6. **Results** (figure C.6). The *Results* component receives the token from the *StateFinder* and uses it to get the results from the back-end. It shows a graph of the customer concentration, the same error log as in *FollowProgress*, and shows a button to download the result excel file using a *FileButton*.

4 BACK-END IMPLEMENTATION

The back-end performs all the analyses for the customer insights. Besides the analyses, the given customer data also needs to be pre-processed, matched, and enriched with extra information. In this chapter, the steps performed in the back-end are explained and visualised.

4.1 FLOW CHART

In order to give a more detailed overview of how input data flows through the program, a flow chart is presented in Appendix D. This flow chart is similar to the flow chart in Figure 2.1, but contains additional details for each step of the back-end. Every node in the flow chart will be explained in more detail in this chapter.

4.2 CLASS DIAGRAM

A flow chart provides a detailed overview of how the data flows through the application. In order to also get a better understanding of the structure of the code base of the application, a class diagram can be found in Appendix E. This class diagram is particularly useful when the code base is ever to be extended.

4.3 API

The Application Programming Interface (API) provides a way to present the analysis service to the internet. Because the code of the front-end is executed on the user's machine and the code of the back-end on a server, these parts need to communicate with each other over the internet. The front-end requests data from the API, the API extracts this data from the back-end memory, prepares it for the front-end and responds to the front-end.

This API is implemented following the Robusness Principle (Posner, 1980), which is often reformulated as "Be conservative in what you send, be liberal in what you accept". This API accepts data in different formats, but always follows the same guidelines for what it sends back. For example the path $/api/v1/analyze$ accepts data in JSON[1] format, as well as form data[2](Masinter, 1998). In this data, most fields are allowed to be missing and will be filled in with default values. Also, more information than necessary can be given, which will be discarded. An overview of the calls this API accepts and which data it takes as input and sends back can be found in Appendix F.

4.3.1 Asynchronous design

As the analyses can take a long time to finish, the system should not wait to respond until all analyses are done. This is because of the following three problems:

- The user has no progress feedback in the meantime.
- Higher effectiveness of a Denial of Service attack. If all analyses are started synchronously and one takes long, the next cannot be executed until the previous is finished, and new requests cannot be answered.
- The browser of the user might timeout because it gets no response.

To resolve this, the API uses threads. When a request for analyses on data comes in, the API creates a new thread, copies the information from the request to the thread, starts the thread, and immediately answers a token to the user. The progress and results of each thread are identified with a token of 16 random characters[3]. This

1 JavaScript Object Notation
2 How browsers send data from a form
3 This gives a probability of $1.2564928e^{-25}$ of correctly guessing a token and illegitimately obtaining someone's analysis results. This is almost a million times less likely than finding the correct grain of sand on the Earth

thread starts running and analysing the data. Meanwhile, the front-end can request the progress and the results by specifying this token.

Results from an analysis are deleted after a day. This has three benefits: it complies with requirement B.4.3, the server does not overflow with data after a number of analyses, and no legal data management has to be done to protect the data in the server.

4.3.2 Error handling and logging

To ensure traceability of errors and stability of the system, all errors are logged to a file. The API implements a logging system that can log messages depending on their severity. From low to high severity: debug, info, warning, error or critical. To ensure the user is also informed, an error list is tracked for each token. The user can request this error list by specifying this token.

4.3.3 Authentication

To ensure only PwC employees can access this application, the API will only be accessible from the PwC VPN. To allow a specific group, and to ensure this server is allowed to process client data, users need to log in with their PwC credentials before they can access the API. The API checks if a user is logged in correctly, and if it is not, the API redirects it to the login website of PwC.

To accomplish this, the API uses the decorator design pattern. This pattern allows behaviour to be added to a function dynamically. If functions that accept requests from the front-end are prepended with *@authorisation_required*, another function will first be called, which checks if the requesting user is correctly authorised.

4.4 PREPROCESSING

After the data has been uploaded and the required columns are available, the transaction data needs to be preprocessed before it can be used in the analysis. During the preprocessing, two goals are achieved. The first goal is to remove transaction data that cannot be used in the greater part of the analysis. Each transaction must specify revenue and at least one of the four primary keys must be available. The primary keys are attributes of customers that make the customer uniquely identifiable. The standard primary keys the preprocessing algorithm looks for are:

- The kvk number of the customer.
- The vat number of the customer.
- The address of the customer.
- The name of the customer.

Transactions for which no primary key is available are filtered and put in a separate data set: unidentifiable transactions. These transactions are used to calculate the total revenue or profit a company generates, and provide an indication of the reliability of the analysis.

The second goal is to standardise the company names of the customers. The company names often contain characters that provide no actual information. Removing these characters is beneficial for the later steps of the analysis. The complete standardisation process consists of four steps:

1. Converting the name to lower case.
2. Removing uninformative characters.
3. Removing duplicate white spaces
4. Stripping the data (removing leading and trailing white spaces)

After the standardisation process, company names such as *Company B.V* and *COMPANY-BV* will be reduced to the same name: *company*.

As the preprocessing step is now finished, the data is sufficiently prepared to be used in further analysis

4.5 CUSTOMER IDENTIFICATION

The main purpose of the customer identification step is to map each transaction to a customer, where transactions belonging to the same customer are mapped

to the same customer. The result of the customer identification step is a list of unique customers that are identified in the transaction data together with the list of transactions where a new column is appended: the unique ID of the customer.

4.5.1 Combining multivariate keys

In the customer identification step, the four primary keys described in Section 4.4 are used to identify unique customers. Before the actual customer identification process begins, a customer identification specific preprocessing step is applied first. Primary keys that consist of multiple subkeys, such as the address consisting of either city and street or zip code and house number, are combined in a single column. These values are simply combined by adding a space between the subkeys and storing the result in a new column. These columns are referred to as combined keys.

After the preprocessing step, the actual customer identification can commence. The customer identification is divided into two steps. The first step is mapping customers using keys that are guaranteed to be unique, such as the vat or kvk number. The second step is mapping customers using keys for which values can be similar, such as the company name.

4.5.2 Mapping based on unique keys

The mapping based on unique keys can be further divided into two steps: Finding the value of each unique customer for each key and removing any duplicates that occurred as a result of the first step. For each key, all transactions are grouped based on that key. For example, grouping based on the kvk number of customers yields all unique kvk numbers in the transaction data. For each unique value, a new customer is created and per customer, the IDs of the transactions that belong to that customer are stored alongside additional information that can be found about that customer in the transaction data. After each key is processed, the mapped transactions are marked and the next key ignores already mapped transactions.

After this process, duplicates values can still exist. These are removed by again looking at all unique values in the customer data. If a unique key has a value that appears multiple times in the created customer data, then the corresponding customers are merged into a single customer. The lists of transactions that belong to those customers are also merged in a single list. After this step, no duplicates in the unique key columns of the customer data remain

4.5.3 Mapping based on similar keys

The next step is to map additional transactions based on keys that are not guaranteed to be unique but can be similar. For each key, the values of the remaining unmapped transactions are grouped by that key. These values are then compared to the values of existing customers using the Levenshtein string distance. The Levenshtein (Levenshtein, 1966) string distance, sometimes referred to as the edit distance (Navarro, 2001), is the number of single-character edits that are required to change one word into another. For example, the words *company* and *comp any* require one edit: the addition of a whitespace in the center. The words *companymarket* and *companyportal* have a Levenshtein distance of five, as the word *market* has to be almost completely transformed into the word *portal*. If a value has a Levenshtein distance of at most two with any existing customer value, the words are considered similar enough and the transactions belonging to that value are added to the list of transactions of a similar customer. For any value that has no similar match with existing customer values, a new customer is created.

After the mapping has concluded, each customer is given a unique ID, which is added as a new column in the corresponding transaction data set. The final result is a list of unique customers that are identified in the transaction data together with the list of transactions where for each mapped transaction a corresponding customer ID is available.

4.6 MATCHING

After unique customers have been identified, the data is ready to be matched to an external database. The purpose of the matching step is to *match* every customer in the client data to an external database, where *matched* is defined as being linked to an external database where additional data about the customer can be gathered.

4.6.1 Orbis database

Orbis is used as the external database to find additional information about each customer, such as the industries in which they operate, the number of employees, and other common characteristics of companies. Orbis is a database that tracks public information about all companies in Europe. Orbis offers two methods to retrieve information, either through their web-based interface or through the usage of an API. The initial preferred method would have been using the API, but because of budget constraints from PwC, an alternative solution had to be devised.

The alternative solution was to download company information manually from the Orbis website, convert the downloaded Excel files into a single csv file, and insert this csv file into a single table in an SQL database. This SQL database is then queried instead of the Orbis API. Apart from an upfront cost to download the information from Orbis, subsequent retrievals from the SQL database are free, in contrast to the costly Orbis AP. However, as downloading the Excel files by hand is a tedious process the decision had to be made to do this only for Dutch companies, which is a small subset of all available companies in Orbis. The rest of this report will refer to this homemade SQL database as 'the Orbis database'.

Connecting with the SQL server is done with the SQLConnector class, which is a child of the abstract Connector class. Since the SQLConnector can be configured in multiple ways, the factory method design pattern (see Section 2.1) has been applied. Using the ConnectorFactory class, any type of connector can be constructed with the same method. This allows the application to be flexible in how a connection to the Orbis database is made.

Matching with the Orbis database is done with the same four primary keys described in Section 4.4. The matching process is, again, different for keys that are guaranteed to be unique and keys that are not guaranteed to be unique.

4.6.2 Matching based on unique keys

For the unique keys, the customer data set that is generated in the customer identification process is inserted into the SQL database, using either methods from the pandas[4] library or SQL's bulk insert function if available. SQL's bulk insert method has a better performance than the methods from the pandas library (Gulutzan and Pelzer, 2003), but can sometimes be unavailable due to security constraints.

After inserting the customer data, the join operation of SQL is used multiple times to match customer values with the company database. For practical purposes, the complete joining procedure is stored as a view, for which the raw SQL can be found in Appendix G. After the matching based on unique keys is done, the customer table is removed from the SQL database to avoid leaving sensitive data on a database.

4.6.3 Matching based on similar keys

The next step is to match keys that are not guaranteed to be unique. For this purpose, the CONTAINS predicate of SQL is used to find similar matches. For every unmatched customer, a query is constructed using the Object Relation Mapping (ORM) language of SQLAlchemy[5]. The benefit of using the ORM language is that the generated queries are more robust (Chen et al., 2014), providing both query speed and safety from SQL injection attacks. If the query is successful and a match is returned, the information is appended to the data set that resulted from the previous matching step. If no match is found for a customer using a particular key, the next key is used. If no match can be found using any key, no additional information is added for that customer.

After the two matching steps have been executed, the result is a data set with additional information for each matched customer. The final step is to use this data set to enrich the original customer data. Columns that do not yet exist in the customer data, for example, newly found statistics such as industry and number of employees, are copied into the customer data. Columns that do already exist in the customer data, for example, the customer KvK number or address, are copied only if values in the original customer data are missing. The result is an enriched customer data set additional information is added and missing values are filled in using the Orbis database.

4 https://pandas.pydata.org/
5 https://www.sqlalchemy.org/

4.7 DATA COLLECTION

Four additional categories of statistics are gathered or calculated by other means. These are the distance between the client locations and the customer locations, the lifetime value of each customer, the customer purchasing behaviour, and the general industry statistics of the customers.

4.7.1 Distance

The distance can be calculated in two ways. The most straightforward way is calculating the size of a line between the two GPS locations, the haversine distance (Robusto, 1957). A more interesting distance measure is the driving time between two points. The driving time is a more accurate description of how far a customer is from a client and is therefore preferred over the haversine distance. The driving time between two points is found using the API of any Maps provider, for example, the Google Distance Matrix API [6].

4.7.2 Lifetime value

The lifetime value of a customer is a statistic that is used in the customer segmentation analysis (see Section 4.8.2), and in the customer driver analysis (see Section 4.8.3). The lifetime value of a customer can be calculated in two ways: either based on the revenue of a customer or based on the profit of a customer. If based on the revenue, the customer lifetime value is the total amount of revenue that the customer generates for the client. To calculate the total revenue each customer generates, a sum of the revenue of every transaction the client has with the customer has to be taken. If based on the profit, the transactions must also have a cost or margin available in the data set. After the profit is calculated for each transaction, the customer lifetime value is then calculated in a similar manner.

4.7.3 Customer purchasing behaviour

The general purchasing behaviour of every customer is described using four statistics. These are the number of purchased products, the number of orders placed, the average price per purchased product, and the average price per order. These statistics give an overview of the customers' purchasing information. The obtained data can also be used in the customer segmentation.

4.7.4 Industry statistics

The final set of statistics are general statistics about the industries of customers. After the matching process described in the previous section matched customers have a primary code and in certain cases secondary codes available, identifying the primary and secondary industry of their activities. These codes are classifications of economic activities, according to the specifications in 'NACE Rev. 2'[7]. Although not always perfect, these codes provide an indication of which industry the customer operates in. Four general statistics are calculated using these codes: the distribution of the industries of customers, the total revenue each industry generates for the client, the average customer lifetime value per industry, and the average number of employees the customers in each industry have. For all statistics already gathered data can be used, such as the previously calculated lifetime value of the client. Using the *group-by* function of the Pandas library together with aggregate functions, such as sum, mean, or counts, the appropriate statistics can be calculated.

The result of the data collection phase is an enriched customer data set and a new data set that contains general industry statistics. The data is now sufficiently prepared and ready for the subsequent steps of the data analysis.

4.8 CUSTOMER INSIGHTS

For the actual data analysis, four different analyses are applied to the prepared data: calculating the customer concentration, segmenting the customers into groups, determining the most important driver of the customer lifetime value, and finding the cross-sell potential of the client. The appropriate sections will discuss each analysis in detail.

6 https://developers.google.com/maps/documentation/distance-matrix/start
7 https://ec.europa.eu/eurostat/documents/3859598/5902521/KS-RA-07-015-EN.PDF

4.8.1 Customer concentration

The customer concentration is the amount of revenue held against the number of customers providing it, making it clear what percentage of revenue is made by what percentage of customers. The function takes as input a dataframe of the customers with their revenue and outputs a dataframe for the user with an easy overview of statistics and a dataframe useful for the front-end to make a graph visualisation of the customer concentration.

In the first step of calculating the customer concentration, customers are sorted in descending order based on their revenue. The analysis only needs a customer ID and the corresponding revenue. Three new columns are added during the analysis. The first additional column is the cumulative sum of the revenue. The second additional column shows the percentage of revenue made by the customer and the customers with higher revenue. The last additional column indicates the percentage of customers. The percentage of customers is a linear spread of zero to a hundred over the total amount of customers, in combination with the percentage of revenue, it gives a clear overview of what percentage of revenue is made by what percentage of customers. The front-end is given the list of cumulative revenue, reduced to one value for each percent of customers. This is done for a visualisation in the front-end, which is shown in Figure C.6.

4.8.2 Customer segmentation

The customer segmentation analysis takes all customers and groups them in different segments, based on their characteristics, causing similar customers to be put in the same group.

For the segmentation, a data set of customers is needed. The output is a summary of the formed clusters and a list of all customers and in which cluster they were put. The summary contains information on how many customers there are in each cluster and what the defining characteristics of each cluster are.

Subgrouping and one-hot encoding

Columns with categorical values already are clusters by themselves, which can make it seem controversial to use it for clustering. There are two ways these features can be used. The first option is to group the customers by their categorical value and cluster the resulting subgroups individually. For example, if there is a category with four different values, four different groups are made and the clustering is performed on every group separately. The second option is to one-hot encode the data (Gori, 2017). With this option, the feature is used for the clustering itself. For every unique categorical value, a new column is made and per customer filled in with either a one or a zero, indicating if the category belonged to the customer. By default, the ten most occurring categories are used for the clustering, but the user can change this number to their own preferences. A limit was set because sometimes categories can have over two hundred unique values, which would create two hundred new columns for all customers, which would result in categorical columns having a greater impact on the resulting clusters than non-numerical columns

Null handling

For the clustering, all data values need to be present, meaning there can be no missing values. However, the given data is almost certain to have missing values in them. It is not desired to replace missing values with self-predicted numbers since this can lead to inaccurate data. PwC does not want to give advice based on generated data, thus all customers that have incomplete information are not taken into account for the customer segmentation. These customers are removed from the data set before the clustering begins.

Normalising

The data needs to be normalised to make all columns equally important. If clustering is applied to shoe size and how tall people are, the length would outweigh the shoe size, since the sizes are such lower values. The values of each numeric column are normalised between zero and one using min-max scaling[8], to make the columns equally important for the clustering algorithm. Categorical columns already are normalised, but because every instance is either a one or a zero the

8 https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

categorical features are seen as more important by the algorithm, as normalised numerical values rarely have multiple values that reach 1. To counter this, categorical values are divided by how many new columns were made. For example, if there are five categories, every value is divided by five. Now the values are either 0 or 0.2. The user can also change this number if the categorical features need to be weighted differently.

Optimal number of clusters

Once the data is ready for clustering, the optimal number of clusters is determined by the silhouette score (Rousseeuw, 1987). For every number of clusters within a given range, the silhouette score is calculated and the number of clusters that scored the highest is used for the clustering. See Section 4.8.2 on how clusters are made. By default, this range is between 1 and 15, but the user can adjust this range to their own preferences, to speed up the process or to look at a higher number of clusters. The silhouette score looks at the distance of all data points from their own cluster data points compared to the data points of other clusters. The higher this distance is for all data points, the higher the silhouette score. Meaning every data point is as close-by its own cluster and as far away from other clusters, making it the optimal amount of clusters. The user can also give a fixed number of how many clusters should be made. If a number is given this step is skipped, else the algorithm tries to find the optimum itself.

Clustering

Since the optimal number of clusters is now established, the clustering can begin. For the clustering, the k-means cluster algorithm is used (Garbade, 2018). K-means clustering sets k random cluster centroids in the data set and puts every data point in a cluster, based on the nearest centroid. When this is done for all data points, the centers of these points are calculated and new centroids are placed on these locations. On these new centroid locations, the data points are again distributed. This is done until there are no more changes in allocating the data points to the clusters, or the maximum amount of tries is reached. When one of these two occur, the variance between clusters is calculated and the clustering is saved. The clustering now starts over from the beginning with different initial cluster centroids and tries to cluster again. After trying to cluster for a given amount of time (e.g. ten), the clustering with the highest variance is chosen and given as output for the k-means clustering. A high variance correlates to data points being as close to their own cluster and as far away from other clusters as possible.

The characteristics that were used for the clustering are now grouped per cluster, to give a summary of what every cluster contains. If there were subgroups made based on categorical features, the summaries of every subgroup are displayed separately.

4.8.3 Customer driver

To determine the most important driver of the customer value, a linear regression analysis is performed. This analysis consists of three steps: preparing the data, fitting a linear model, and comparing standardised model coefficients.

The user can customise which explanatory variables the regression model uses. This can for example consist of the number of employees or the industry. The analysis will always use the customer lifetime value as the target variable, as the relationship between the explanatory variables and the lifetime value is investigated.

Data preparation

Before a linear model can be fitted to the data, the data has to be prepared using methods similar to the data preparation that is done in the customer segmentation in Section 4.8.2. All categorical columns are converted to a numeric type, using the same one-hot encoding method. One thing to note is that the customer driver analysis uses all category values, in contrast to the customer segmentation, meaning an extensive number of columns are added to the data. After all, data has been made numeric, each feature is standardised, meaning all features have zero-mean and unit variance. This allows features of different scales to be comparable (Schroeder et al., 1986), which is necessary if model coefficients are going to be compared.

Finding the most important coefficient

After the data is sufficiently prepared, a linear model is fitted using the Scikit-learn[9] library. The advantage of this library is that it is much more efficient than any homemade implementation would be, and using the library is fairly straightforward. Using the LinearRegression class of the Scikit-learn library, a linear regression model is fit and its coefficients are determined.

Determining the most important driver of the customer lifetime value is now simple, as the standardised model coefficients can be compared. A feature with a high valued coefficient has a larger impact on the customer lifetime value than a feature with a coefficient that is low. The coefficients can be sorted based on their value descending. The first coefficient now has the largest relative impact on the customer lifetime value and is therefore the most important driver of the customer value.

The final result is an ordered list of feature coefficients, where the first coefficient is the most important driver of the customer value, and the last coefficient is the least important driver of the customer value.

4.9 CROSS-SELL ANALYSIS

The cross-sell analysis shows which products are often bought together and finds which products can be recommended to which customers. It also calculates the expected revenue increase based on these recommendations.

The implementation of cross-sell analysis is divided into four steps: Finding the baskets, using the baskets to perform market basket analysis, finalising the rule output and using this to recommend cross-sell opportunities.

4.9.1 Finding baskets

First off, the tool must preprocess the transactions in order to be able to analyse the data. All transactions with a revenue smaller than or equal to zero are removed because these serve no purpose to the analysis. Furthermore, the only transactions that occur in the last year of the data set are used in the analysis in order to have a more accurate and up-to-date result.

The transactions are entered in a way that every row represents one type of product being bought. In order to perform market basket analysis, baskets must be formed first. Baskets are a group of products that were bought by one customer at the same purchase. These baskets are needed to see which products are being bought together. Consequently, in this step the transactions are grouped either by invoice number or customer-id and date, forming baskets of products that were bought together by a customer. If both are available, invoice is preferred over customer-id and date because the latter is more error-prone as the data can contain multiple purchases on the same day from the same customer.

After grouping the baskets, duplicates are removed from the data. When a customer buys the same product basket several times, only one is kept in the data. Else, the analysis could be based on the purchasing behavior of only one customer, leading to a fixed outcome. After these steps, the data is ready for the market basket analysis.

4.9.2 Market basket analysis

The second step of cross-sell analysis is the market basket analysis. This is the most important step. During this step, the transaction data (grouped into baskets) is analysed to see which items are bought often and association rules are created. First, an algorithm is used that finds commonly bought together products (frequent itemsets) named Frequent Pattern Growth (fp growth). The output of this algorithm is an overview of all frequent itemsets in the provided transaction data with their corresponding support value, described in Equation 4.1:

$$support(x) = \frac{frequency(x)}{amount\ of\ baskets} \tag{4.1}$$

9 https://scikit-learn.org/stable/

After that, Association Rule Mining (ARM) is applied to these frequent itemsets. This algorithm calculates which itemsets are bought together and thus have the potential to be cross-sell opportunities. Rules will be formed using Equation 4.2.

$$rule : antecedentproduct(s) \rightarrow consequentproduct(s) \tag{4.2}$$

This will from now on be referred to as: $A \rightarrow B$.
To illustrate Equation 4.2: if the antecedent products are bought, it is likely that the consequents are also bought (Tan et al., 2020).

Hereafter, the lift of the rules is calculated using Equation 4.3, which calculates the dependence of consequents on antecedents. Only rules with a lift value larger than 1 are kept in the output, as then the products are dependent on each other because the antecedents and consequents appear relatively often together. The lift is calculated by dividing the support of when A and B are bought together in baskets by the support of A in any basket:

$$lift(A \rightarrow B) = \frac{support(A \text{ and } B \text{ together})}{support(A) \cdot support(B)} \tag{4.3}$$

The user has four options that they can change in this analysis:

1. The minimum frequency: How many times an itemset should occur in the data set before being frequent enough to be used for association rule mining (default value is 15).
2. The minimum confidence (Equation 4.4): the minimum threshold for confidence for association rules to be displayed. The higher the confidence, the more certain it is that the rule is often used, but it also decreases the number of rules in the output (default value is 0.2).

$$confidence(A \rightarrow B) = \frac{support(A \text{ and } B \text{ together})}{support(A)} \tag{4.4}$$

3. The maximum amount of antecedents: the maximum amount of antecedent products there should be in the association the rules (default value is 3). The more antecedent products there can be in rules, the more rules there are that can barely be applied to customers.
4. The maximum amount of consequents: the maximum of consequent products there should be in the rules (default value is 2). The more consequent products there can be in rules, the more rules there are that can barely be applied to customers.

All these options influence the output of the cross-sell analysis. An example of the output from the market basket analysis can be found in Table 4.1

Antecedents	Consequents	Support	Confidence
[Hammer]	[Nail]	0,02	0,98
[Paintbrush]	[Paint, Tape]	0,01	0,93
[Nail]	[Hammer]	0,02	0,86
[Drill]	[Screw]	0,03	0,81

Table 4.1: An example of the cross-sell market basket analysis output

4.9.3 Finalising rules

The third step of the analysis is finalising the rules from step two by adding more useful insights. The following information is added to the output per rule:

- The true confidence value is calculated using Equation 4.5 and 4.6. It is the same as confidence but it also takes into account how much the consequent products are bought without the antecedents. The true confidence is more accurate to see if a rule is correct.

$$true\ confidence(A \rightarrow B) = \frac{support(A \text{ and } B \text{ together})}{average\ support(A, B)} \tag{4.5}$$

$$average\ support(A, B) = \frac{support(A) + support(B)}{2} \tag{4.6}$$

- Revenue of the consequent products. If the revenue of the consequent product is more than 20 and twice the revenue of the antecedents, the rule is removed. Else, products can be recommended that are too expensive for the customer to buy based on the price of the antecedent products. The tool should not recommend products that are far more expensive then the products that were bought originally by the customer.

- A list of customers that buy the antecedent products and never any of the consequent products. This step is the most time consuming part of the analysis. After the first 10 rules are checked, a time estimation is given to the user containing the expectation how long the analysis still is going to take.

With all these new insights, the first output is generated and is called the cross-sell rule output. The output is sorted descendingly by the true confidence value. An example of the cross-sell rule output can be found in Table 4.2 with more insights added in Table 4.1:

Antecedents	Consequents	Support	Confidence	True confidence	Revenue	Customers
[Hammer]	[Nail]	0,02	0,98	0,96	1,10	[22, 22, 44]
[Drill]	[Screw]	0,03	0,81	0,86	2,00	[13]
[Paintbrush]	[Paint, Tape]	0,01	0,93	0,57	3,00	[11]

Table 4.2: An example of the cross-sell rule output

4.9.4 Creating recommendations output

Furthermore, another output is created with every row corresponding to a recommendation to a customer and the probability that the customer will buy the recommended product(s). The following four insights are added to this output:

1. The number of times the customer has bought the antecedent products without any of the consequent products.
2. The maximum potential revenue that could be generated from the customer, using Equation 4.7. The potential revenue is calculated by multiplying the revenue of the consequent product(s) with the amount of times the products are recommended to customer x

$$potential\ revenue(A \rightarrow B,\ x) = revenue(B) \cdot count(x) \tag{4.7}$$

3. Probability that a customer buys the consequent products after recommending it to them, using Equation 4.8. In this equation, 0.7 is the maximum probability that an existing customer buys a recommended cross-sell product (Bendle et al., 2016).

$$P(A \rightarrow B) = 0.7 \cdot true\ confidence(A \rightarrow B) \tag{4.8}$$

4. The expected revenue gain of cross-sell recommendations, as stated in Equation 4.9. The expected revenue is calculated by multiplying the probability of the rule with the potential revenue of customer x based on a single rule

$$expected\ revenue(A \rightarrow B,\ x) = P(A \rightarrow B) \cdot potential\ revenue(A \rightarrow B, x) \tag{4.9}$$

After all these new insights are added to the output, the recommendations are sorted by expected revenue to show the most promising and profitable recommendations. An example of the cross-sell recommendations output corresponding to Table 4.2 can be found in Table 4.3:

Lastly, a table is outputted to the client containing the total potential cross-sell revenue and the total expected cross-sell revenue. An example of the cross-sell revenue output corresponding to Table 4.3 can be found in Table 4.4:

Customer	Antecedent	Consequent	True Confidence	Revenue	Count	Potential revenue	Probability	Expected revenue
22	[Hammer]	[Nail]	0,96	1,10	2	2,20	0,672	1,36
13	[Drill]	[Screw]	0,81	2,00	1	2,00	0,602	1,20
11	[Paintbrush]	[Paint, Tape]	0,57	3,50	1	3,00	0,399	1,20
44	[Hammer]	[Nail]	0,96	1,10	1	1,10	0,672	0,74

Table 4.3: An example of the cross-sell recommendations output

Total potential revenue	Total expected revenue
8,30	4,50

Table 4.4: An example of the cross-sell total revenue output

4.10 OUTPUT

After all steps of the analysis have been concluded, the final step is to generate an output file for the user. As the expected users are generally skilled with Excel, the most efficient output is a single Excel file that contains the results of all analyses.

Generating this Excel file is done by using the built-in functions of the Pandas library. One of the main advantages of the Pandas library is a consistent approach to converting the underlying dataframe to different output files. Each step of the analysis implements its own function to convert the analysis result to a single or to multiple Excel worksheets. After all analyses are finished, a general output function is called that calls a customised output function for each analysis result.

The generated Excel file is then returned to the front-end, where it is available for the user to download. The file is available to download for 24 hours, after which it is removed from the server.

5 PROCESS

In this chapter, the whole development process and procedures taken in the project are gone through. The development methodology, source control, development resources, testing methodologies, and quality improvements during the project are discussed in order.

The project was carried out in twelve weeks, between April 13th and July 1st. The first two weeks consisted of doing research: getting a sense of what the project is about, setting up the requirements and giving a high-level design of the software to be implemented. In weeks 3 to 11, the software was written and tested. Week 12 consisted of preparing the presentation.

5.1 SCRUM

To have a clear structure in our project, we adopted Scrum, an agile way of working. This is a framework that helps with planning and responding quickly to the needs of the product owner (PwC). In practice, this meant that we tracked all our actions to be done in a list, called the backlog, and prioritised these actions each week. At the end of the sprint, we discussed our results of the past week and plans for the coming week with Martin Kerkhof (PwC), to make sure we were on the same page regarding priorities.

This workflow has allowed us to make quick changes in our way of work and made sure our planning was not set into stone after discussing the requirements in the first two weeks. This has proven to be valuable, as our plannings in the first five weeks were very optimistic, and we tended not to progress as fast as we had hoped.

Besides this weekly meeting, we met with Lise Gunneweg and Thijs Boevink at the start and at the end of each day, to discuss the plans and the results of the day, and the problems that arose along the way. Also, we met once a week with Prof. Dick Epema to discuss our advancements academically.

Apart from these scheduled meetings, employees within Deals Analytics were very helpful in brainstorming to solve problems we encountered and in performing user evaluations.

At the beginning of each sprint, we estimated the time to be spent on each task. At the end of each sprint, we reviewed this estimation and filled in how many hours were actually spent on each task. Table 5.1 shows how many hours were spent on which tasks.

Task	**Hours**	**Description**
Research	320	First two weeks were spent researching.
Meetings	257	Meetings within PwC and with the TUDelft supervisor.
Report	235	Writing the report & preparing the presentation.
Front-end	201	Developing the website, making it user friendly.
Customer insights	184.5	Customer segmentation, concentration and driver analyses.
Cross-sell	183	-
Code quality	167	Reviews to check each others code.
Matching	85	Matching internal customers with external company data.
Back-end	66	API: responding to front-end & providing excel output.
Customer identification	39	Extracting unique customers from transaction data.
Data collection	38	Calculating industry statistics & drive time.
Pre-processing	6	Cleaning the transaction data.
Total	**1781.5**	Hours per person: **445**

Table 5.1: Time spent on tasks

5.2 SOURCE CONTROL

This project used a repository on the GitHub website of PwC. GitHub allows for collaboration on the same code by tracking changes, hosting the code, and offering a way to check each other's code before adding it into the main branch. We adopted a workflow that made sure the software we wrote was checked twice before adding it to the main version.

In GitHub, branches are used to work on software independently in the same project. One branch is made for one feature. When that feature is finished and tested, the software is put up for review in a Pull Request (PR). Other team members review the code based on the following four metrics: if the code is up to the standards of the code already in place, if it adds a valuable feature if it is tested with at least 80% line coverage[1] and if the code is correctly formatted.

In our workflow, there were two main branches: the *master* and the *dev* branch. The *master* branch always contained a working version of the product. This branch is updated at least once a week from the *dev* branch. In the *dev* branch, changes in the current sprint were accumulated. If a feature was proposed, a new branch was made where the feature was implemented and tested. Next, the programmer creates a PR to add the new code to the *dev* branch. Nothing could be added directly to the *dev* or *master* branch, new code could only come from a separate branch using a reviewed PR.

5.3 DEVELOPMENT RESOURCES

Back-end

In the back-end, all coding is done in Python. To make coding as efficient as possible, we all picked the same IDE[2] and set it up in the same way. Because the back-end was programmed in Python, we used PyCharm[3] for writing code. The community edition was free for commercial use, and we all had experience using it. Within PyCharm, we setup **pylint**[4] to do the static analysis, and **pytest**[5] for dynamic testing.

Front-end

In the front-end, the code is a mix of JavaScript and HTML because it is supposed to run in a browser. We used the React[6] framework, which simplifies making user interfaces by dividing the code up into components, each with their own function, logic and layout. They can be separately programmed, tested and used. We used Visual Studio Code[7] as IDE, and integrated it with **ESLint**[8] and **Jest**[9] for static and dynamic testing, respectively.

5.4 TESTING

Testing is one of the most important aspects of the development process. This section will describe what testing methodologies were used and how these were applied to the project. Testing is done in two ways: statically and dynamically. Static testing consists of checking the code without running it. Static testing tries to find inconsistencies in the code, style errors or common patterns that lead to bugs, as well as the cyclomatic-[10] and interface complexity of functions.

5.4.1 Dynamic testing

Dynamic testing runs the code, tries different inputs on the system and asserts that the outcome is correct. The dynamic tests are done on three levels: unit, inte-

1 80% of all the lines in the project is tested by an automated test, one of the requirements specified in Appendix B
2 Integrated Development Environment, text editors for programming
3 https://www.jetbrains.com/pycharm/
4 https://pypi.org/project/pylint/
5 https://docs.pytest.org/en/latest/
6 https://reactjs.org/
7 https://code.visualstudio.com/
8 https://eslint.org/
9 https://jestjs.io/
10 Cyclomatic complexity, or McCabe complexity, is the number of routes the code can take through a function. This can loosely be defined as one plus the number of if-statements in a function.

gration and system. The difference in these levels is how many components they evaluate in the same test.

Back-end

During the development of the back-end, it was always made sure that unit testing was performed on new code before merging it into the development branch (as described in Section 5.2). Tests in the back-end try different inputs on the function under test and assert the function provides the correct output. The code always had to have at least 80% of the statements tested before being approved, making sure that individual components of the software work as expected. Running these tests automatically helped with locating bugs and debugging the code. This led to a line coverage of 83%.

Front-end

In the front-end, each component has its own unit tests. Groups of components have their own integration test, and two system tests make sure the flow of starting, editing, and reviewing the analyses is correct. The result of this approach is that 98% of all lines and 89% of all branches are covered in a test.

Front-end dynamic testing is done in two ways: by simulating a user automatically, and by rendering a component and verifying it looks the same each time with the same input. The testing framework renders the component under test, and the tests press buttons, enter data into textboxes or tick checkboxes and verify that the component does what it should do. The challenge in these tests is not only to cover all branches the code might go into but also to try all the unexpected actions an inexperienced user might take, to ensure the system is foolproof.

5.4.2 Regression testing

The tests can be used as a *regression test suite*. When a change is made to one part of the front-end, this should not change anything to other seemingly unrelated parts of the front-end. *Regression tests* test every part of a system and their collaboration, and fail when output differs unexpectedly. This ensures that in further development, a developer is notified when a change has unforeseen consequences to the system.

5.4.3 Usability testing

Furthermore, usability testing was performed on the project. During the development of the front-end, five possible users unrelated to the project have been interviewed to find out whether the application was easy to use. The front-end had to be tested this way, as employees of PwC are supposed to use the tool without our help when the project is finished. The developers of the project might skip some steps the average user might need to take, as the developers know their system from inside out. Five employees were invited to use the front-end without prior knowledge. Besides a lot of small changes, the following big changes were implemented:

- A **Landing page** was added to give a user the relevant information immediately when he loads the tool. The page shows what the tool does and what data it needs. This was added because the previous version did not have an explanation with a lot of unstructured text.
- The **Settings** page was extracted from **Upload File** to a separate step. In the previous version, after a file was uploaded, the analyses predictions and the settings were added to the same screen. This was too much information. In the new version, the analyses predictions are shown when a file is uploaded, but the settings are shown when the user goes to the next step. This choice was made because the analyses predictions layout depends more on the contents of the file than the layout of the settings.
- The placement of **Next** and **Back Buttons** was standardised by introducing the *Navigator*. Before this change, the users had to search for the buttons in each step.
- An existing PwC tool was shown as example, which shows icons of question marks next to fields that might be unclear. When a user hovers over such a

question mark, it gets a popup with an explanation of what to fill in. This functionality was implemented in the tool.

Overall, the number of steps a user needs to follow to perform the analyses, changed from three (enter data & preferences, progress, results) to six (landing page, enter data, add file, preferences, progress, results).

5.4.4 Performance testing

Another methodology of testing that was applied to the project was performance testing. The goal of this testing methodology was to test whether the tool performed well in real user situations, making sure that all found errors were displayed and that the program did not crash. The aim was to perform quality assurance before delivering the tool. The performance testing was conducted in three steps:

- Load testing; making sure that the tool could handle large amounts of input data.
- Stress testing; putting in unrealistic amounts of data and testing where the failure point is in the program.
- Adhoc testing; trying to break the application as much as possible when needed requirements of analyses or input variables are not met.

5.4.5 Acceptance testing

Lastly, acceptance testing was applied to the project in order to make sure the tool was ready for delivery. Employees of PwC installed the tool on their computer in order to perform the acceptance testing because at the end of the project they are the users of the tool. They checked whether the tool is in compliance with their criteria and if it meets the end user their needs. Consequently, this assured that when the tool was delivered no critical requirements were missing and it could be used without problems by the end users.

5.5 SOFTWARE IMPROVEMENT GROUP

During the project, the developed software was reviewed twice by the Software Improvement Group (SIG) in order to make sure the delivered code was up to market standards. The second ratings were on average 0.75 higher than the first. Both times ratings were given to the software between 0.5 and 5.5, with 3 being the market average. The evaluations and corresponding improvements can be found in Sections 5.5.1 and 5.5.2.

SIG focuses on six points of interest in the quality of the maintainability of the software:

1. Volume: how many lines of code (LOC) are produced, expressed in man-months. The codebase should be kept small for easier maintainability.
2. Duplication: how often code is used multiple times (copied). The code should not be duplicated but reused. Duplications are undesired since it is sensitive to bugs and intensive to maintain. When a single change needs to be made, the change needs to be done in different locations. This is hard to keep track of and there is a risk that not all locations are changed accordingly.
3. Unit size: how many lines of code every unit has. A unit should be short and for a single purpose. Units with a single responsibility are easier to test and reuse. When units have fewer lines of code, they are better understood and faster to go through.
4. Unit complexity: how many decision points in a unit are present. When a unit has a high complexity, it is harder to understand what is happening in the unit and a lot of test cases are necessary to cover the decision points (McCabe, 1976).
5. Unit interfacing: how many parameters a unit expects. When it comes to parameters, a unit should take as few as possible. A unit with many parameters is dependent on a lot of variables and hard to modify since it relies on a lot of external information.

6. Module coupling: how many incoming dependencies a module has. Modules that are coupled and dependant on each other can cause ripple effects when changes are made and make the module more complex.

5.5.1 First examination

The results from the first examination can be found in Table 5.2

Category	Rating
Volume	5.5
Duplication	5.2
Unit size	2.4
Unit complexity	2.6
Unit interfacing	3.8
Module coupling	5.0
Mean	4.1

Table 5.2: First feedback from SIG. Values between 0.5 and 5.5. Market average: 3

As can be seen, most improvements could be made in unit size and complexity. Now knowing the important points to good software quality, we made our static analysis tests stricter. The tests were adjusted on the limits of the size, complexity, and interfacing of the software. If a metric went over a limit, an error was shown. These limits were adjusted to the range in which the software is considered to be in excellent shape according to SIG. For size, the maximum lines of code per unit were set to fifteen. For complexity, the maximum number of decision points per unit was set to five. For interfacing, the maximum arguments per unit were set to three.

Before delivering our code for the second examination, we made sure the software satisfied all new restrictions. The unit size and complexity were improved by splitting units into sub-units making every unit shorter and having fewer tasks to perform. The unit interfacing was improved by putting all optional parameters in a configuration dictionary. Because optional parameters were also included, they contributed the most to the bad score for unit interfacing.

5.5.2 Second examination

The results from the second examination can be found in Table 5.3. SIG decided to split up the evaluation to two categories: front-end and back-end in order to have more detailed insights.

Category	Back-end rating	Front-end rating	Average improvement
Volume	5.5	5.5	+ 0.0
Duplication	5.5	5.2	+ 0.15
Unit size	4.6	1.3	+ 0.55
Unit complexity	5.5	4.4	+ 2.35
Unit interfacing	5.3	5.4	+ 1.55
Module coupling	3.9	5.5	- 0.3
Mean	5.1	4.6	+ 0.75

Table 5.3: Second feedback from SIG. Values between 0.5 and 5.5. Market average: 3

As can be seen above, several improvements were made after the first evaluation. There was an average improvement of 0.75 and in some categories, there were major improvements. For instance, unit complexity and unit interfacing both had an improvement of more than one point. However, module coupling had a small decrease in rating, due to the module coupling in the back-end. This was the consequence of creating a utility module with methods that were used several times by different modules. This decreased the amount of duplicated code but increased the module coupling.

Furthermore, the unit size rating in front-end is low, mostly because the new way of writing components in front-end is using functional components. Effectively are classes but are recognised by SIG as functions. React components average on about 70 to 80 lines of code. When such functional components are 70 to 80 lines it's seen as a lot, while they should be interpreted as a class. This causes the unit size rating to be very low.

5.6 LEARNING EXPERIENCE

This project has been a unique learning experience for us since it was the first time we've worked on a large project with a real-world purpose. Because we worked with PwC as a client, we also experienced how it is to work in a global company. Being an intern at PwC has taught us the dynamics of large-scale corporations and allowed us to receive training in how to deal with information leakage, corruption, insider trading, independent professionalism and other business concerns.

Since the analyses required a good understanding of the financial world, we had to ensure we also focused our research on extending our knowledge on this area. The analyses could not have been implemented without the basic economics and finance knowledge. Since we were all interested in the workings of the financial sector, this posed no problem and was actually one of the aspects that made this project interesting. When we did not have sufficient knowledge, PwC employees were ready to help us with our problems.

Besides improving our general business sense and knowledge of the financial sector, we also enhanced our project and technical skills. Working together with the client on a daily basis and experiencing regular setbacks has allowed us to develop our communication skills and our ability to react to unforeseen problems. Because this project had a relatively short time span, we also learned how to obtain the most important requirements from the client and how we should prioritise those requirements.

Working on a project during the COVID-19 epidemic has also been a new experience for all members involved. Every meeting has been held from home, and communication has been conducted solely with Google Meet[11]. As a group we were fortunate that we all knew each other before the start of the project, as that made sure the online communication was no barrier. The epidemic did unfortunately cause some more technical problems, which are discussed in section 7.1.

To conclude, we feel proud of what we learned and accomplished during the project, and are excited for any similar projects in the future.

11 https://meet.google.com/

6 EVALUATION

In this chapter, three evaluations of the final product will be done. In summary, all the *Must Have*, and most of the *Should Have* requirements are met. We have reached all three of our design goals and thus consider this project a success.

In Section 6.1, an evaluation will be done based on the *Must Have* and *Should Have* requirements. These requirements were originally set at the start of the project, but because of some unforeseen problems elaborated upon in Section 7.1, the user authentication was added as a *Should Have* requirement. In Section 6.2, the design goals set at the start of the project are evaluated. Finally, the success criteria are evaluated in Section 6.3.

6.1 EVALUATION OF REQUIREMENTS

The evaluation of the requirements is split into two parts: an evaluation of the *Must Have* requirements and an evaluation of the *Should Have* requirements. Each time a table is given, where each requirement category is listed along with an explanation of how well that category of requirements was met. An evaluation of the *Must Have* requirements can be found in Table 6.1, and an evaluation of the *Should Have* requirements can be found in Table 6.2. To summarise both tables, all *Must Have* requirements have been met and the greater part of the *Should Have* requirements have been met.

Requirement Category	**Met?**	**Explanation**
Interface	All	The user is able upload and download data, see if any errors occur, and cancel the analysis.
Data	All	The tool is able to process client data and query an external database for additional information.
Matching	All	The tool is able to remove duplicates and match the customer details to an external database.
Analysis	All	The tool is able to generate the required insights into the clients and insights into the customers of the client.
Output	All	The tool outputs the raw data and enriched data into a single Excel file.
Testing & Error handling	All	The tool remains online, despite any error in the analyses. Besides that, the final testing coverage was more than the required 80%.

Table 6.1: Evaluated *Must Have* requirements.

6.2 EVALUATION OF DESIGN GOALS

In this section, the design goals that were set at the start of the project and described in Section 2.1, are evaluated. The three design goals usability, reliability and privacy are evaluated separately. In summary, the three design goals have all been met.

- Usability - The usability of the tool consists of two parts: how well the interface is usable by a user, and how well the output data can be interpreted.

  The usability of the interface was extensively tested with the help of user evaluations, laid out in Section 5.4.3. The interface aims to provide the user with a steady stream of information, as to not overload the user. The tool is

Requirement Category	Met?	Explanation
Interface	All	The user is able to see whether compulsory columns are missing, and can be authenticated using OAuth.
Data	All	The tool specifies compulsory and optional data columns.
Matching	All	The tool is able to deal with gaps in the client data.
Analysis	Most	The tool generates the most important driver of the customer value, additional insights into the customers, cross-sell potential and customer segments. However, the tool does not identify new potential customers and does not calculate the total market potential.
Output	No	The tool does not output detailed descriptions of how the analysis was done. This final report is available publicly if a user wants more information.
Visualisation	All	The tool shows the mapping success rate and shows the progress of the analysis in real-time. The real-time progress has been split into the different steps of the analysis, such that the user is informed well on the progress.
Testing & Error handling	All	The tool corrects errors whenever possible. The analysis continues to be executed, unless a critical error does not allow any further analysis.

Table 6.2: Evaluated *Should Have* requirements.

self-explanatory, usable by PwC employees within the Deals branch, and can be executed using default settings.

The table format between input data and output data is consistent. The original structure of the client's data is kept intact, and new information is appended. The output of the tool can directly be opened in Excel and be used for interpretation.

As both parts of this design goals are considered to be implemented, this design goal is evaluated as 'reached'.

- Reliability - The reliability of the tool also consists of two parts: how reliable the conclusion of the data analysis is, and how reliable the user interface itself is.

  In the steps of the analysis that prepare that data for the actual analysis, key missing data is always either removed or entered. Useless transactions are removed during the preprocessing and customer identification phase, and missing customer information is inserted during the Orbis matching phase. The subsequent analysis steps that generate insight are also able to deal with missing values, each with the most appropriate approach for that analysis. The user is provided with some statistics, such as the rate of useful transactions or the matching success rate, in order to make the user aware of when an analysis might not be reliable.

  The user interface has been constructed such that the website does not crash on unexpected data, but rather informs the user when something has gone wrong. If any steps of the analysis fail, the subsequent steps of analysis are performed whenever the failure was not critical to the complete analysis. For example, the cross-sell analysis can be performed when a failure occurs in the customer segmentation, as those analyses are not dependent on each other,

  As both aspects of this design goals are considered to be implemented, this design goal is evaluated as 'reached'.

- Privacy - The important aspect of the privacy design goals was that any data should not be stored outside of the tool's memory. The tool does not distribute

any client data and stores the results of the analyses locally. The user only has temporary access to the files, as the files are automatically removed after 24 hours. This design goal is therefore evaluated as 'reached'.

6.3 EVALUATION OF SUCCESS CRITERIA

At the start of this project, success criteria were set to be able to evaluate the complete final product and when the project can be considered successful. The success criteria are evaluated in this section, and an explanation is given why this project can be considered a success.

As described in Section 2.3, the project can be considered a success when the system complies with the three main design goals and at least the *Must Have* requirements are implemented. Additionally, the greater part of the *Should Have* requirements should be implemented such that the tool will actually be used in the deal life cycle by PwC employees of the Deals branch. The final product contains all *Must Have* requirements, complies with the three main design goals and will actually be used in the deal life cycle. The final product can, therefore, be considered a success, according to the success criteria set at the start of the project.

7 DISCUSSION & RECOMMENDATION

Looking back at the project, there are some points worth mentioning. The encountered problems during the project, alternative approaches that would be taken if done again, advice on future work if someone would continue further development of the tool, and the ethical implications the tool brings with it are discussed in this chapter.

7.1 ENCOUNTERED PROBLEMS

This section describes the unforeseen problems that were encountered during the project and how they were solved. All problems can be categorised in two types of problems: organisational and technical problems.

7.1.1 Organisational problems

The following segment describes all problems that occurred due to organisational issues. During the project, several organisational issues arose that are discussed here.

Unclear SIG procedure

To start, a problem surfaced when deadlines were coming closer. During the project, the university demanded the code to be analysed by the Software Improvement Group (SIG) on May 29th. However, there were no instructions given on how the code should be sent. We sent an email three days before the delivery date with the question of what procedural steps were expected of us. The next day we got a reply from SIG with instructions. One point of interest was that SIG would not take responsibility for possible leakage of software and they advised us to set up a Non-Disclosure Agreement (NDA). Previously, we thought no special actions had to be taken since the course had not given any explicit instructions on how to deliver the software, this turned out otherwise. We sent an NDA the next day, which we got from the Office of the General Counsel (OGC) from PwC, to SIG. Unfortunately, this was on too short notice, since we got a reply on June 3rd about the NDA. They stated that they would not sign the NDA in its current state, because the NDA was valid for an indefinite time and a fine was given in case the software got spread outside of SIG. An updated NDA was sent and eventually approved. Due to this miscommunication, it took a week longer to submit our codebase to SIG.

We received our feedback on our the on July 9th. The next delivery date for code review from SIG was set on the 12th of July, so we put a high priority on improving our code based on the feedback, managing to resubmit the improvements by July 12th. However, on the submission date, the course sent out an email stating that the submission date (July 12th) was wrong and should have been a week later. To conclude, we had to implement all improvements in two days, while this turned out to be unnecessary.

Unavailable API keys

Another unforeseen problem was the unavailability of certain API keys. Because of COVID-19 and its resulting economic situation, the expenses of PwC had to be kept to a minimum. This meant that costly API keys, such as an Orbis API key or a Google API key, could not be made available for usage.

We were able to devise a solution for both missing API keys. The problem of the missing Orbis API keys was solved by manually downloading Orbis data sets from their website, and constructing an SQL database using those data sets. This process is described in more detail in section 4.6.

The missing Google API key was solved by giving the responsibility to the user. The user has the option to specify an API key for either the Bing or Google API. If

this API key is correct, the key will then be used by the tool for accessing the API's. The advantage of this approach is that a Bing API key can easily be generated by the user, and the Bing API has a free limit that is usually sufficiently high to performs the distance calculation.

Deployment process

[redacted]

- [redacted]
- [redacted]
- Finished Product: Overall, this means the project could not deliver a product ready to use by the intended employees of the Deals branch. To make the tool ready to use for Deal Analytics, a guide was written to install the tool locally and was installed on the PCs of Lise Gunneweg and Thijs Boevink.

PwC has indicated they would like us to continue [redacted] after our internship has ended.

7.1.2 Technical problems

There were also problems that occurred because of technical issues, instead of organisational aspects. These problems are discussed in this section.

Handling unexpected NULL values

One of the technical problems we encountered was unexpected NULL values. NULL values are undefined values and are thus unknown. A lot of data needed for analyses was initially expected to be present in the provided data sets from clients. However, most of the time these are incomplete, leaving unknown values. For the analyses, to derive the customer driver and segmentation, NULL values are not permitted. These NULL values had to be handled and removed or replaced properly. It took some extra days to decide and implement how to handle these NULL values.

Categorical features could not be transformed to numbers, because categories with a number close to each other would be seen as a relationship, while that should not be the case. The initial approach was to replace the unknown values and create a dummy feature for every feature containing NULL values, which indicated which samples were originally NULL values. The unknown values got replaced with the mean value of the present data. Besides the mean value of the data, variants were also tried with the minimum and maximum value in the data. Unfortunately, these variants were determined to not be a well fit replacement.

A logical way to handle unknown values would normally be by looking at neighbours and replacing the NULL value with something similar to the neighbours' values. However, this was not desired for our data, since PwC does not want to

give advise to their clients with 'guessed' data. Because of this, the decision was made to not take samples into account that have a missing value on one of the features that are going to be used for the segmentation. The dummy column automatically also got deleted with this decision. Unfortunately, this process took a lot of time away from implementing the actual segmentation.

Data set sizes

Hierarchical clustering was originally chosen for the clustering in the customer segmentation because this method does not use random initial weights and is, therefore, reproducible (Nielsen, 2016). However, one of the pitfalls of this method is that it cannot handle big data sets. It was thought that the data sets would not become too big for hierarchical clustering. However, the biggest available set turned out to be too big for hierarchical clustering and the cluster method was therefore changed to k-means clustering. K-means clustering is not fully reproducible, but works on every size of data set. The succeeding of the clustering on every size of the data set was considered more important than being fully reproducible.

Clustering inconsistency

The number of found clusters in the customer segmentation kept changing when the same data set, with numerical and categorical features, ran multiple times. Different methods for calculating the optimal number of clusters were tried, such as the Elbow method (Bholowalia and Kumar, 2014), Silhouette score method (Rousseeuw, 1987), and Gap Statistics method (Tibshirani et al., 2001). When these methods ran on self-generated data with clear clusters, these methods were consistent. However, when real data was used with less clear clusters and with categorical features, the optimum would just be the number of categories in the categorical feature.

Different distance metrics were attempted for better results. Besides the available metrics of the clustering algorithm, such as Euclidean and Manhattan, the Gower distance was also tried. The Gower distance is made especially for both numerical as categorical values, so one-hot encoding would not be needed anymore (Yan, 2019). Unfortunately, the Gower distance did not give the correct number of clusters on self-generated data.

Different cluster algorithms were also looked at, such as Kmeans, Kmedoids (Park and Jun, 2009), Minibatch-kmeans (Sculley, 2010), and DBSCAN (Schubert et al., 2017). All methods did not give a correct number of clusters or were not consistent, on the self-generated data with clear clusters.

By in- or decreasing the weights of categorical features, the number of clusters could be made consistent, but this meant that either the categorical feature would be really important or not at all. If the weight was tuned between these points, the number of clusters would become inconsistent again.

Eventually, a workaround was decided, by giving the option to separate the data on a categorical feature and cluster on sub-sets of the original data. For example, if there would be a category business order or private order, the data would be split into two groups, one with all business orders and one with all private orders. This workaround is unfortunately not a good method when there is a categorical feature with two hundred different categories, meaning no solution is found yet for this problem.

Input data sets were larger than expected

The last problem we encountered was that some of our code in the tool was too slow when being applied to large data sets. At first, we did not expect to receive such large data sets to analyse on. So when the tool was created this was not taken into account. When we ran the tool on these data sets, it could take more than a couple of hours and even then it was not sure how long it would take the program to finish. Consequently, a speed-up was necessary. It was found that two parts of the tool took the most time:

- Customer identification
- Cross-sell analysis

The first was customer identification. Some research into the matter concluded that the indexing of the Pandas library that was used was not performing at a high speed. In order to speed-up the customer identification, the underlying Numpy

arrays were used to allow for faster indexing. Consequently, the speed of customer identification was substantially improved.

The other performance issue was found in the cross-sell analysis. When the cross-sell analysis was performed on a large data set, the analysis would take hours in order to finish. It was found that the algorithm used for finding frequent itemsets (A-priori) was not optimal in terms of run-time. After some research, another algorithm (fp growth) was found that calculates the frequent itemsets a lot faster in terms of run-time. However, this did not resolve the problem fully. Another part of the analysis turned out to be slow on large data sets: customer recommendation. For every cross-sell rule it is needed to check whether these rules apply to which customers based on every transaction in the data set. A solution to this particular problem has not been found during the project.

7.2 ALTERNATIVE APPROACH

Looking back on the progress made and the process of the development on the tool, there are three changes we would make if we were to do the project again. These changes are discussed in this section.

Less optimistic time expectancy

To start, we would be less optimistic about the estimation of how long some tasks would take. As explained in Section 7.1, some tasks, such as customer segmentation, encountered problems causing it to take a longer time than expected. While performing the research on these topics, we expected to have it implemented fairly quickly because it did not seem that difficult. However, when implementing these features it became clear that it took longer than expected. If we would do such a project again we would reserve more time for tasks that have a higher chance of encountering problems in order to have a more accurate idea of what can be implemented in the time available.

More detailed research before starting implementation

Furthermore, we would perform more detailed research before starting implementing the chosen features. We encountered several problems (as described in Section 7.1) during the project that took a lot of time to resolve. We are of the opinion that when we would have performed more detailed research before implementing solutions, it would have been possible that these issues would have been known already. Subsequently, a solution for these issues could have been found before implementing the code causing the issue. If better research was conducted and the issue was found before implementing the features, it would have saved some time that could have been used for other possible features.

7.3 ADVICE ON FUTURE WORK

In this section, possible improvements on the tool are discussed if someone would want to continue the development.

Customer segmentation

First off, several improvements can be done in customer segmentation.

At the moment of writing, a data set can be sub-grouped based on (multiple) categorical features. However, if such a feature has a lot of categories it can be undesirable to make many sub-groups. A possible improvement could be to regroup certain sub-groups that have familiar properties. This would counter the clustering algorithm to have to run a lot of times and gives a clearer overview of the whole data.

A standard scaler works better with outliers than a linear min-max scaler. Because the clustering also needs to perform with categorical features it is hard to determine what the weights of these features should be, since the numerical features can vary outside the zero and one range. However, when clustering is performed and there are only numerical features present, a standard scaler could be used. At the moment, a linear scaler is always used, but this could be changed when only clustering on numerical features.

DBSCAN was already looked at during the implementation of the segmentation, but the epsilon was never automatised. When the epsilon is automatically deter-

mined the results might be better than with the current kmeans. Further research should have to be done for this.

Cross-sell analysis

Furthermore, cross-sell analysis can also be improved in two ways. Mostly, the probability calculation, whether a customer would buy a certain product if it was recommended with a product they already bought, can use improvement. Right now, the calculation is only based on the strength of the cross-sell association rule itself. Meaning that it does not factor in the behavior of the customer itself. For example, a regular customer that spends a lot every transaction has the same probability of buying another product than a customer that only bought one product once.

If development would be continued, it would be recommended that within the probability calculation a component is added that looks at the purchasing behavior of the customers. An example is that the customer segmentation can be used to see if other similar customers started buying the recommended product after a while or not. When this component is used within the probability calculation, the total expected revenue increase of cross-sell can be calculated more accurately.

Besides a better probability calculation, the cross-sell analysis can also be improved by looking at cross-sell possibilities for new products. The tool now only looks at existing products and transaction data, meaning that new products have no cross-sell possibilities. A feature that could be added is that products are checked for similar characteristics outside of the transaction data. Subsequently, when a new product is introduced, it can be identified as similar to already existing products. This information has the potential to create cross-sell rules for newly introduced products, causing more expected revenue of cross-selling.

New customer identification

Lastly, a new feature should be added to the tool when development would be continued: the identification of new customers. Because of all the data being available, it would be possible to derive the most important characteristics of the customers. Furthermore, this information can be used to find new potential customers having these same characteristics. These potential new customers bring along potential extra revenue, causing this feature to be interesting for companies. Furthermore, when new potential customers are identified, they can also be targeted by other analyses from the tool, such as for example cross-sell analysis. Which, in turn, will increase the expected revenue of these analyses as well.

7.4 ETHICAL IMPLICATIONS

In this section, ethical implications are discussed that are applicable to this tool.

The tool gives insight into the potential of the company, providing a better overview of the value of the company. If both sides of an acquisition deal have access to this tool, the information asymmetry between the two companies is decreased. The buyer gets a better idea of what they are buying, making sure that they are overcharged. Meanwhile, the seller can prove its market potential to the buyer, ensuring that the seller gets a fair price for the company. The tool increases the transparency of a deal as it becomes harder for both parties to sell or buy the company based on misleading information.

Furthermore, the buyer can also use this tool to see if any synergies exist between the companies. If any synergies would exist, the overall value of combining the two companies would be greater than the values of the two companies uncombined. Merging companies that both thrive off each other creating mutualism, which has a positive outcome for both companies.

In short, the tool makes it easier to make better recommendations to companies interested in other companies. By using this tool, companies have more reliable information before making decisions.

8 CONCLUSION

At the start of this project, the desire of PwC was to have a tool that could help them during the B2B deal cycle. Most of their analyses were performed manually or with limited generic data analysis tools. In order to deliver their services in larger volumes and with more valuable insights, they were looking for a tool that automatically performs their analyses and applies advanced data analytics techniques. There was no existing software solution that could be used for this purpose.

After two weeks of research and design, in nine weeks a web-based tool was fully implemented. This tool meets all the original *Must Have* requirements, and most of the *Should Have* requirements. Some *Should Have* requirements have not been met due to unforeseen circumstances, which is discussed in more detail in Section 7.1.

However, the project can be considered successful. The implemented analyses provide the user with a clear overview of the customer base of the client. It highlights the strengths, but also unexplored opportunities of a company. The complete B2B customer analysis that usually takes two or three weeks can now be performed in less than a day. The user interface has been evaluated by different PwC employees in the Deals branch and was improved such that they were able to use it properly.

The project was a unique learning experience for the team members, where project skills, technical skills, business sense, and knowledge of the financial sector have vastly been improved. The group was able to respond well to unforeseen circumstances partly created because of COVID-19, and provided efficient solutions.

When a production server is made available, the development team within PwC will port the application to be available within their network. Until then, the tool will run locally on the computers of PwC employees. As the tool saves a large amount of time within the Deals cycle, it will be used in order to provide the B2B customer analysis in larger volumes.

If there is interest in continuing development on this tool, this report should contain sufficient information to do so. We recommend five additional improvements on the cross-sell and customer segmentation analysis, and the addition of a new feature: identifying new customers, based on the data that is gathered by the tool.

POSTFACE

Dear Bas, Cherwin, Sander and Tjard,

For the last ten weeks, we have been working together with you on this project. We are amazed by the work you have accomplished in this short period of time! We are impressed by the result; a powerful tool that is highly flexible and user-friendly despite the complex algorithms used. This tool will enable us to do higher quality and more efficient diligence in the future. We will have to spend less time ourselves creating analyses and can focus more on providing more valuable insights for our clients.

Throughout the project, you faced various challenges driven by the complexity of the data, the business requirements and the IT infrastructure. You have shown a high level of flexibility in dealing with these challenges and came up with creative solutions to fix it. We have enjoyed working with you on this project very much and we learned a lot from you. You can definitely be proud of the work that you have done. We wish you all the best in the future; a great job that you will enjoy and excel at and of course we hope that this will be at PwC.

Many thanks to all of you! Lise, Martin, Thijs

BIBLIOGRAPHY

Bendle, N. T., Farris, P. W., Pfeifer, P. E., and Reibstein, D. J. (2016). *Marketing metrics: the definitive guide to measuring marketing performance*. FT Press.

Bholowalia, P. and Kumar, A. (2014). Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9).

Chen, T.-H., Shang, W., Jiang, Z. M., Hassan, A. E., Nasser, M., and Flora, P. (2014). Detecting performance anti-patterns for applications developed using object-relational mapping. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, page 1001–1012, New York, NY, USA. Association for Computing Machinery.

Garbade, D. M. J. (2018). *Understanding K-means Clustering in Machine Learning*. accesed on: 21st of April 2020.

Gori, M. (2017). *Machine Learning: A constraint-based approach*. Morgan Kaufmann.

Gulutzan, P. and Pelzer, T. (2003). *SQL Performance Tuning*. Addison-Wesley Professional.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Masinter, L. (1998). Returning values from forms: multipart/form-data. Technical report, RFC 2388, August.

McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on software Engineering*, 4(4):308–320.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Nielsen, F. (2016). *Undergraduate Topics in Computer Science*. Springer.

Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36:3336–3341.

Posner, J. (1980). Transmission control protocol. Rfc, RFC Editor. Accessed on June 16th, 2020.

Robusto, C. C. (1957). The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40.

Rousseeuw, P. (1987). Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, 20:53–65.

Schroeder, L. D., Sjoquist, D. L., and Stephan, P. E. (1986). *Understanding regression analysis: an introductory guide*. Sage.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21.

Sculley, D. (2010). Web-scale k-means clustering. *Proceedings of the 19th international conference on World wide web*.

Tan, P.-N., Steinbach, M., Karpatne, A., and Kumar, V. (2020). *Introduction to data mining*. Pearson.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society Series B*, 63:411–423.

Yan, M. (2019). *Introducing Python Package — gower*. accesed on: 4th of July 2020.

A | RESEARCH REPORT

The original research report can be found in this appendix, starting on the next page. Please note that some parts of the design have been changed throughout the project, which makes this research report only useful for comparing the original design to the final design.

Delft University of Technology & PwC

BACHELOR END PROJECT

TI3806

Research Report on B2B Customer Analysis Automation

Authors:
Sander van Leeuwen
Tjard Langhout
Bas Volkers
Cherwin Ort

Supervisors:
Dick Epema (TU Delft)
Lise Gunneweg (PwC)
Thijs Boevink (PwC)
Martin Kerkhof (PwC)

May 19, 2020

Contents

1 Introduction

This is the research report of the bachelor end project in business to business customer analytics at PwC. The project aims to apply the skills the students learned in their bachelor to build a tool for PwC, which automates the analysis of a client's growth potential, by providing insights into the value of the client and its customers, cross-selling opportunities and new customers of customer segments. This challenge is presented by the Deal Analytics group within the Deals branch of PwC Advisory. This group provides data-driven insight to improve decision making and value creation throughout the entire deal lifecycle. A part of the activities the Deal Analytics branch performs is customer analysis of companies focused on business to business (B2B) transactions. A large part of this analysis is done manually, and by automating a part of this process the efficiency of generating insights in the deal lifecycle can be improved.

This tool will not only be used by the Deal Analytics group, but also by other groups within the Deals branch. Therefore any employee working in the Deals branch who may gain productivity by using this tool to gain insight into companies and their customers will be invested in this project's success.

This tool will be developed in 12 weeks, of which the first two weeks are dedicated to research, where the goal is to find and describe the optimal way to build this tool. This report will cover the research done, and discuss the results. Firstly, the design goals and requirements of this project are described in section 2. section 3 will layout the high-level overview of this tool; which parts it consists of. Finally, the concrete implementation details are discussed in section 4.

Throughout the rest of the document, the following terminology will be used:

- Tool: Software written during this project to do business to business customer analysis
- User: The PwC employee who uses the tool
- Client: Company under advisory by the user
- Customer: Company or individual to whom the client sold goods.

2 Design Goals

In this section, the main design goals for this project will be elaborated. These design goals are considered while designing, implementing and testing the solution to the problem. The main goals set for this project are usability, reliability and privacy.

2.1 Usability

Usability is the first main design goal of the project. The tool will not only be used by the Deal Analytics group, but also by other groups within the Deals branch. It is therefore important that the user can use the software without extensive knowledge about data analysis. The tool should be self-explanatory and the analysis executable with default settings. To test the user-friendliness and guide the design of the system, two user evaluations will be held. More details are described in section 4.2.2

Another aspect of usability is a consistent data format between input and output data. The original structure of the client's input data should be kept intact. This is important because it ensures the user understands the format of the output data and can continue working with this data. As described in 4.11, there will be two output data sets: one with all collected data, and one with generated insights, cross-sell potential and newly identified customers.

2.2 Reliability

Besides usability, another key aspect of the project is reliability. Reliability has two meanings for this project. First, it answers the question of how reliable the output and conclusion of the data analysis is. If the data analysis is performed on a lacking data set, the conclusions of the analysis are not reliable. In every step of the analysis, the quality and quantity of the input data will have to be kept in mind. The user should be made aware of the quality of the uploaded data and given an indication of the reliability of the analysis.

Furthermore, another interpretation of reliability is how reliable the tool itself is. The tool must not crash on input the user gives, either through interacting with the interface itself or through uploading data. It is

therefore important to perform extensive input validation to ensure the tool stays running. Another advantage of performing extensive input validation is the increase in the security of the system. Besides the tool not crashing on user input, the tool also should not crash during any step of the analysis process. Instead, errors should be displayed to the user and steps of the analysis should be skipped if possible.

2.3 Privacy

Privacy is the last main design goal of this project. The user uploads sensitive client data, data that should not be distributed further. In order to preserve the privacy of the data, the data should only be stored temporarily in server memory. Both the input data and the results of the analysis are not stored outside of the tool's memory. An aspect of this design goal to consider is that reloading the web page removes any progress on the analysis. This aspect has to be kept in mind in designing the system.

3 High-level Overview

The application will be presented as a web-based tool. A user can surf towards a website to load the tool and upload its client's data after which the analysis of this data is done on servers. These are the main advantages of a client-server architecture in this project.

- Faster results: computation of analysis is offloaded to faster servers.
- Lower risk of spread of application outside of PwC compared to one application.
- Application can be included in existing PwC web-based tools.

The application will have two distinct parts. The first part is what the user interacts with. This is called the front-end or User Interface (UI), and runs on the computer of the user. The front-end is a website which the user can load when connected to the PwC network. In the front-end, the user can start, monitor, and review the results of the analysis. This part is elaborated upon in section 4.2.

When the user submits data of a client to be analyzed, this data is sent to the back-end which is a program running on a cluster of servers. This data consists of information about the client and its customers: the records of sales between the client and the customer, when something was sold, and for how much. This submission and the subsequent processing steps are visualised in figure 1.

The first step in analyzing this data is identifying customers. This means that the customers of sales from different records have to be compared against each other to find which sale belongs to which customer. For example: 'MediaMarkt', 'Media Markt' and 'Media Markt B.V.' should all be considered the same company. This step is called data cleaning and is explained in section 4.5.

The next step is called matching (see section 4.6) and provides extra information on these customers like their address, size, industry and their value to the client, by querying Orbis, CompanyInfo and the Google API for the identified customers.

After extracting this information, insights into the customers and their purchasing behaviour can be generated. This includes calculating the customer concentration, identifying the characteristics of valuable customers and segmenting the customers into groups, as well as analyzing the cross-sell potential by finding products commonly bought together and characteristics of companies who cross-buy. The methods used for these calculations are elaborated on in sections 4.8 and 4.9.

Lastly, new customers are identified based on earlier generated insights, improvements in sales to current customers based on cross-sell information are searched. When this is completed, the results are transferred from the back-end to the front-end and presented to the user. The user can, in turn, see the conclusions and download all data generated in the process.

To make sure the tool is always accessible by PwC employees, it will be deployed to a Kubernetes cluster provided by PwC to dynamically manage scaling.

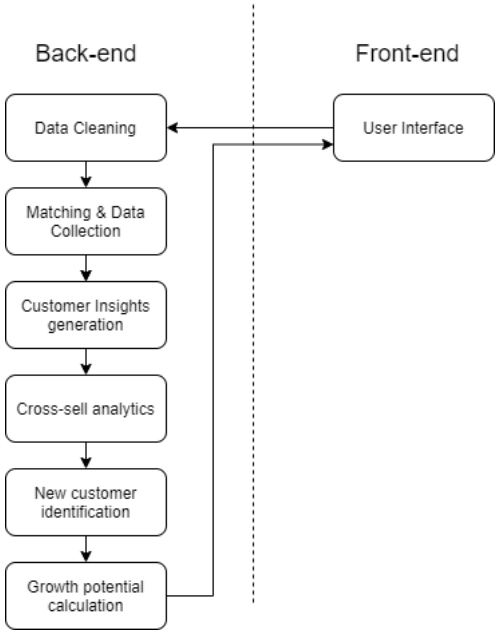


Figure 1: Flowchart at high-level

4 Implementation details

This section will provide details on how the analysis should be implemented in the front-end and back-end. The B2B customer analysis is divided into five different steps: cleaning the client data, matching internal data to external data, performing the data analysis, outputting the results into a preferred format, and providing visualisation. The actual data analysis is divided into three parts: providing insights into the customers of the client, identifying the cross-sell potential of the client, and finding new potential customers. These parts can be combined in a conclusion; a total growth potential estimate. For all steps, the implementation details are described.

4.1 Design Patterns

To ensure the architecture of the application is clear, in some way standardized, and easily maintainable, design patterns are used in the software of this application. A design pattern is a general and reusable solution to a commonly occurring problem within a given context in software design. One design pattern that will be used is the client-server pattern. The advantages of using this pattern are laid out in section 3.

Another used design pattern is Model-View-Controller. This pattern splits the logic of an application into three interconnected parts. The logic of this design pattern cannot be projected onto this application one-to-one, but the chosen design comes as close as possible. The first is the model. This is the data about the client and customers which is to be analyzed. Second is the view. This will be represented by the interface the user sees, in the form of a web-page. When the model changes, the view should change: when there are new insights found in the client data, this should be shown on the web-page. These two parts are connected by a controller. The controller handles changes made by the view and projects them onto the model. The controller communicates between front-end and back-end, and matches, expands, and analyzes the model.

4.2 Front-end

The front-end of the tool is handling the interaction with the user. The front-end is presented as a website, which the user can load using a web browser. In the front-end, the user can start, monitor and review the results of the analysis. The goal of the front-end is to make its user interface as intuitive as possible, so new users can learn to use the tool as quickly as possible, and experienced users are guided to reduce the number of mistakes. This is why the project will do two user evaluations, in which a user interface will be proposed to five users per evaluation.

4.2.1 Flow

The flow of the front-end is the same for each user. When a user visits the website, he is greeted with a quick overview of the process that is to come. Underneath this, the first step in the process is shown: uploading the customer data and setting preferences for the analysis. These preferences include:

- Which variables to include in cross-sell analytics.
- Which variables to include in customer segmentation.
- In which area to search for new customers.

After the user has supplied this information, he can click a *Start* Button. This uploads their data to the back-end, which will, in turn, start identifying the customers. Every 10 seconds, the front-end will ask the back-end if there is more information on the progress. This progress will be shown as a flowchart of the analytical process, indicating which pieces have succeeded, have failed, are in progress or are to be done. This flowchart is visible in the box 'Analysis' in appendix B.

When the analysis is done, the front-end shows the conclusions of the analysis. The user is presented with two new buttons, one to download all data which is gathered, generated insights and cross-sell opportunities found, and one button which will only contain the conclusions the back-end has found.

4.2.2 User Evaluations

The goal of the front-end is to have optimal learnability, efficiency, flexibility and user satisfaction. The search for such an intuitive design is guided by two user evaluations. In each of these user evaluations, five people will be asked to use the website to analyze a customer with data supplied by us. During this process, we will not be guiding the people through the system but will take a look at how they do on their own, what is difficult and what is unclear. Five people are enough to interview as 80% of the problems will be found (Nielsen & Landauer, 1993). In the first user evaluation, a prototype will be proposed, and the questions asked for big improvements in the user interface. Based on the responses, a user interface will be designed, which is evaluated in the second survey. In this user evaluation, the questions will be more detailed, asking for small improvements in the layout.

4.2.3 Tools & Frameworks

To ensure PwC employees are quickly familiarized with the design of the user interface, and for speeding up the development process, the PwC app kit will be used. This app kit defines the look and feel of PwC; styles for text, buttons, graphs and menu's which are also used in other PwC website based tools.

To develop the front-end logic, React[1] will be used. This is a framework with which stateful, dynamic websites can be developed using JavaScript. The main advantage of using React is reusable components. Once a component is developed, it is trivial to include it in different parts of a web-page. The members of this project already have experience creating websites with it, which speeds up the development process. The PwC app kit can be integrated by importing it to React.

4.2.4 Data Types

The application will accept an Excel file with information about the client and its customers. The information about the customers of the client should be provided in transaction format: in the first row the name of the columns, and in the rest of the rows the values of the transactions. Each row contains the information about selling one product to a customer. The user is not obliged to have any columns present in the Excel file, the tool will indicate which analyses can be done using the presented columns.

This groups the columns into two: the first part consists of *expected columns*. These columns are actively used in matching, generating customer insights and analyzing cross-sell potential. The second part is *unexpected columns*, these columns can be used to give more information about transactions or customers if other information is valuable in the future.

The following columns are expected:

- **customer_value**: The value of the given customer. If not given, the user can choose to derive the value from the revenue or profit.

[1] https://reactjs.org/

- **customer_kvk**: Kvk number of the customer. Will be used for data cleaning and finding more information about the customer.
- **customer_vat**: VAT number of the customer. Will be used for data cleaning and finding the Kvk number or name of the customer.
- **customer_address**: The total address (street, zipcode & city) of the customer, if this data was not presented by the client in separate columns. Will be used for data cleaning and finding the KvK number or name of the customer.
- **customer_address_country**: The country of the customer.
- **customer_address_town**: The town of the customer.
- **customer_address_street**: The street and street number of the customer.
- **customer_id**: Client's internal ID of the customer. Will be used for data cleaning and finding the KvK number or name of the customer.
- **customer_name**: The name of the customer. Will be used for data cleaning if no KvK or VAT number or address is present. Will also be used for finding more information about the customer.
- **invoice**: Invoice number of the order of the product. Will be used for cross-sell analysis.
- **product**: Name of the product. Will be used for cross-sell analysis.
- **revenue**: Revenue generated by this sale. Will be used for cross-sell analysis and customer insights.
- **cost**: Total costs of buying, storing and selling the product. Will be used for cross-sell analysis and customer insights.
- **date**: Order date or delivery date of sale. Will be used for customer insight. (dd-mm-yyyy)
- **industry**: Industry the client is in. Will be used for customer insights and finding new customers.
- **size**: Indicator for the size of the customer. Will be used for segmentation.

Because the project members cannot foresee what information might be useful for future projects, the application accepts more columns than this. These columns can be used for generating customer insights with the Machine Learning methods in segmentation, finding the most important drivers of customer value and discovering characteristics of most valuable customers.

4.3 Back-end

For the back-end, a programming language that provides efficient handling of large data sets is preferred. Python[2] is a good choice for this purpose. Besides being a language that is easy to write, Python provides many libraries that are useful for analysing large data sets. Packages such as NumPy[3] and Pandas[4] provide efficient and intuitive handling of data and other packages such as Scipy[5], Scikit-learn[6], and PyPI[7] provide many functions to perform machine learning and data analysis. Additionally, the built-in visualisation library Matplotlib[8] is useful for providing visualisations.

[2] https://www.python.org/
[3] https://numpy.org/
[4] https://pandas.pydata.org/
[5] https://www.scipy.org/
[6] https://scikit-learn.org/
[7] https://PyPI.org/
[8] https://https://matplotlib.org/

4.4 Work Procedure

As described, the tool will be developed in 12 weeks. Of these 12 weeks, 10 are available for programming. During the project, an agile development method called Scrum[9] will be used as the main software development technology. Scrum consists of small development cycles called sprints. A general planning is constructed and divided into weekly sprints. The main advantage of using the scrum methodology is to be able to quickly responds to changes in the planning. Before the start of each sprint, the previous sprint will be reflected upon. During this reflection, the general workflow can be assessed and unfinished tasks can be reintroduced in the new sprint.

4.5 Data Cleaning

After the data has been uploaded and the right format has been verified, the data has to be cleaned in order to be useful for the analysis. The main purpose of the data cleaning process is to map each transaction in the client data to a customer, where transactions belonging to the same customers are mapped to the same customers. The data cleaning process consists of two steps, the removal of gaps in client data and the identification of unique customers in the client data through identifying attributes and through API's.

4.5.1 Gap Removal

We define gaps as missing columns' values for a certain row. For each transaction where certain values are missing, it has to be ensured that at least one of five identifying attributes are available. Identifying attributes are customer id, KvK or VAT number, company name or company address. Each attribute on its own is enough to identify the company and be able to extract missing information later. If no identifying attribute is available, that transaction is not useful for a large part of the analysis. However, as it is an indicator of the reliability of the analysis and can be useful in some parts of the analysis, that transaction will be put in a separate data set.

4.5.2 Customer Identification through Identifying Attributes

For the first part of customer identification process, the identifying attributes described in section 4.5.1 are used. The identifying attributes are iterated based on how straightforward it is to match using those attributes. The most straightforward technique is to match either the customer id, KvK number, or VAT number. These numbers are unique for every company.

For every transaction in the client data, the first step is to check if there already exists an identified customer with such a unique number. If such an identified customer exists, the transaction can be mapped to that customer. If such a customer does not exist yet, the transaction belongs to a new customer. A newly identified customer is created, containing the information available in the transaction.

If there is no customer id, KvK number, or VAT number available, then the next step is to match a transaction to a customer based on the address. Every transaction for which an address is available can be compared to the addresses of already identified customers. This address doesn't always have to be a complete address, the only important thing is that is a unique address.

The final identifying attribute is the company name. These company names Can be compared using a string similarity algorithm. When two company names are similar enough, based on a certain predefined threshold, they can be considered to be the same customer. Comparing two strings should be done with care and factors such as the length of the strings will have to be kept in mind.

Whenever information is missing from a customer and a new transaction belonging to that customer is found that does contain the missing information, the customer's information can be updated.

4.5.3 Customer Identification through API's

If no match can be found for transactions, based on only the identifying attributes, the final step is to use the Google Custom Search API or the Google Knowledge Graph Search API. For the Google Custom Search API, a website can be found based on the company names. If a transaction leads to the same website as a known entity, they refer to the same company. The same can be applied if two entities have the same website. These two entities can be merged into a single unique customer.

A similar procedure is available for the Google Knowledge Graph Search API. This API gives for every query a small summary of company information, including the internal name the Google Knowledge Graph gives that company. To illustrate this, 'Media Markt' and 'Media-markt' will give the same website by the Google Custom

[9] https://www.scrum.org/

Search API and the same internal name by the Google Knowledge Graph search API. If the same internal name is found for different entities or transactions, they must refer to the same customer, and can therefore also be merged.

The result of the data cleaning step is a list of unique customers to which every transaction is mapped. For every customer, it will also be clear what information is available.

4.6 Matching

After the data has been cleaned and unique customers have been identified, the data is ready to be matched to external databases. Two external databases are considered, Orbis [10] and Company Info [11]. The two databases generally contain the same information. The difference is that Orbis contains information for all companies in Europe, while Company Info only contains information for companies in The Netherlands. The purpose of this process is to 'match' every customer in the client data to external a database. We define 'matched' as being linked to an external database where additional data about the customer can be gathered.

Matching the data to external databases is done in roughly the same order of identifying attributes as was used in the customer identification, although customer id cannot be used for external database matching. If a KvK number is available, Orbis can be queried using that number. If the company doesn't exist in the Orbis database, then Company Info is queried. If a company is found on Orbis, there is no need to also query Company Info for the same company.

If the company can't be found on Orbis and Company Info by KvK number or the KvK number of the company isn't available in the client data, a query by name or address will be done. Using the company name will often be enough for the Orbis or Company Info search engines to find the company. Since Orbis and Company Info also allow a search by address that is another option to consider in matching the company.

If the company name, address and KvK number are unavailable in the client data, but the VAT number of the customer is not, there is a website of the EU[12] available where a VAT number can be queried. If this VAT number exist in their database, the name and address of the company will be available.

After the company has been matched with an external database, missing information in the customer entity can be filled in using the external databases. Five additional company statistics will also be queried from the external databases. These statistics include the size, the revenue, the locations, the type and the SBI-codes associated with the company.

If the company cannot be found on Orbis and Company Info by using any of the above described methods, it is not worth the time to continue searching for the company. The company will have to be discarded. After the matching phase is done, every unique customer will contain additional relevant information for the analysis.

4.7 Data Collection

Three statistics will have to be gathered or calculated by other means. These are the distance between the client locations and the customer locations, the revenue each customer generates for the client, and the industries in which the client and customers operate.

The distance will be calculated in two ways. The most straightforward way is calculating the size of a line between the two GPS locations. To calculate this, the Haversine formula is used. This formula also takes the shape of the earth into account. A more interesting distance measure is the driving time between two points. The driving time is a more accurate description of how far a customer is from a client. The driving time between two points can be found using an API for any Maps provider, for example, the Google Distance Matrix API.

To calculate the revenue each customer generates, no external data is needed. After duplication removal in the data cleaning process, every customer has a unique identifier. Calculating the revenue each customer generates for the client is a simple sum of the profit of every transaction the client has with the customer.

Identifying the industry of the client and each customer is a task that is more challenging. There are two possible approaches, using the SBI's provided by external databases or applying Natural Language Processing on the description of a company. The simplest approach is using the SBI codes and assigning each customer to multiple categories, one for each associated SBI code. However, since many companies often have multiple SBI

[10] https://www.bvdinfo.com/orbis
[11] https://www.companyinfo.nl
[12] https://ec.europa.eu/taxation_customs/vies/

codes and the SBI codes are not always specific, it may not be clear from the SBI codes in which industry a company operates.

The more advanced approach would be to find a description of the company, for example on Orbis, Company Info, Google's Knowledge Graph Search API or the website of the company. That description can be used, in combination with some Natural Language Processing, to assign each company to a category. The second approach is preferred as it is important to be as accurate as possible in identifying the industry in which each company operates to provide customer insights.

4.8 Customer Insights

To provide customer insights three different analyses will be performed. Customer concentration, the most important driver of the most valuable customers and customer segmentation are all relevant insights to determine what kind of customers the client has and who new potential customers are.

4.8.1 Customer Concentration

The customer concentration tells the user how much profit is produced by how many customers. e.g., 80% of the profit is produced by 20% of the clients' customers. This information shows how diverged the sales of the client in prospect to its amount of customers is. The customer concentration can be established by summing all the revenue made per customer. Once the summation is completed, the customers can be sorted on their total revenue. Out of this data, it is easy to establish how many companies produce a certain amount of revenue, but also how much revenue is made by the most profitable customers of the client.

4.8.2 Customer driver

The most important driver of the most valuable customers can be a variety of things. The driver can be used to identify the characteristics of the most valuable customers. To find the driver, the characteristics of a customer have to be held against their customer value. When this is done for all the customers, a correlation has to be found between these characteristics and values. E.g., if all customers with a high customer value are located in a certain location, the driver is then determined to be the location. If all customers have a high FTE, the driver is a high headcount within the company. There are multiple independent variables however, meaning a driver could also be a combination of characteristics. To find the correlation between characteristics and customer value multiple regression analysis has to be done. It is most interesting to see which characteristics become more/less important when the customer value increases, for this a linear regression model will suffice. Another machine learning approach could be a decision tree, but secondary and tertiary drivers are harder to determine since it is not possible to sort on regression in a decision tree.

4.8.3 Customer Segmentation

Customer segmentation is the clustering of the customers into groups, to see in which branches the client's customers are active. This gives the client an overview of which branch(es) they have to target to get to their (potential) customers. Machine learning can be applied to perform the clustering of the customer. There are different types of machine learning clustering methods that are useful in this case: hierarchical and k-means clustering. Hierarchical clustering is more appropriate for customer segmentation. K-means clustering is not reproducible, while this quality is desired. Hierarchical clustering does not work with a random state and works best on smaller data. The number of customers used for the clustering will not be significantly big to cause a problem for this kind of clustering.

4.9 Cross-sell Potential

The goal of identifying cross-sell opportunities is finding customers to sell new products based on the products they already buy. This is done in three steps: identify items that are bought together by sufficiently many customers, characterize what kind of companies cross-buy, and find more customers with the same characteristics possibly interested in cross buying.

4.9.1 Finding Baskets

In order to find cross-sell potential opportunities, baskets must be formed. Baskets are sets of products that the customer bought together. For example, a customer of a store buys eggs and milk together, forming a basket. Most data sets do not group products that were bought at the same time by the same customer but

display every bought product on another row. Consequently, if cross-sell analysis would be done, baskets must be formed first. This can be done in two ways. The first is to look at an invoice number (if available) and group all rows with the same invoice number, forming a basket. However, when the invoice number is not available, rows must be grouped by the same customer and every product that they bought on the same day. If even the precise date is not available, data will be grouped by month.

4.9.2 Market Basket Analysis

When baskets are formed, they will be processed to find dependencies among products. In order to come up with products commonly bought together, association rule mining will be used.
First, the concept of frequent item-sets must be explained. The goal is to find sets of items that appear together "frequently" when being bought by the same customer. However, the term frequently is too ambiguous and is not easily defined. To find these item-sets, one must look at the support for an item-set x, which means the number of occurrences where all items in x are bought by the same customer:

$$Support(x) = \frac{frequency(x)}{N}.$$

The support is the division of the number of transactions that buy set x and the total amount of transactions (N). This only represents how many certain items were bought. However, this is far from being able to label the set as cross-sell opportunities. That is where association rule mining is necessary. Association rules are if-then rules about products. To be more specific: $\{p1, p2\} \rightarrow p3$ means that if a customer has bought $p1$ and $p2$ (the predecessors) it is likely that he will also buy or need $p3$ (successor). These rules each have their so-called confidence, which can be calculated using given sales data:

$$Confidence(\{p1, p2\} \rightarrow p3) = \frac{support(\{p1,p2,p3\})}{support(\{p1,p2\})}$$

To elaborate, the confidence of an association rule is the number of occurrences where all products are bought by the same customer divided by the number of occurrences where the first item-set is bought. However, not all high-confidence rules are interesting. For example, $\{p1, p2\} \rightarrow eggs$ can have high confidence, but that could be because eggs are a very popular product. In order to find whether an association rule is really interesting, the lift should be calculated:

$$Lift(\{p1, p2\} \rightarrow p3) = \frac{confidence(\{p1,p2\}=>p3)}{support(p3)}$$

To explain, the lift is calculated by dividing the earlier stated confidence by the support of the latter product. The outcome of the lift states the likelihood of buying all products together compared to someone just buying p3. If there is a lift of 1, it means that there is no association between the products. If the lift is greater than 1, it means that the products in the association rule are more likely to be bought together. But if the lift is smaller than 1, it means that products are unlikely to be bought together. In this case, it is only important to identify the rules containing a lift of more than 1 in order to find cross-selling opportunities.

4.9.3 A-Priori Algorithm

However, when having to perform these calculations on large sets of sales data, this could take up a lot of processing time and memory. If every possible rule would be checked, this means that when there are 10.000 products, it must be calculated for 100.000.000 possible rules. Consequently, the identifying of cross-selling opportunities would take forever and the tool would be useless. A solution to this problem is the A-Priori algorithm. A-Priori makes sure that only association rules with the potential of having a high lift are being considered in the process. This is done with the following steps:

1. A threshold is set for support and confidence
2. Only subsets having higher support than the minimum threshold are extracted
3. From these subsets, only rules will be selected with a confidence value higher than the threshold

The user will have the option to change the threshold for support and confidence if necessary. Otherwise, the standard support threshold will be based on the size of the data set. The same applies to the confidence threshold. When this algorithm is followed, only calculations will be done on the association rules that have a chance

of being a cross-sell opportunity, instead of calculating all possibilities. The association rules will be ordered from the highest lift to lowest lift (above 1) so that the association rules with the highest cross-sell potential will be shown first.

There exists a library in Python that executes the A-priori algorithm. This library will be used in the tool. There are other algorithms to perform association rule mining (such as PCY, Multistage and Multi-hash algorithm), but A-priori is the least complex one. Because of this, we chose to use it. If later on it is discovered that a speedup is necessary for the application, other algorithms could be considered.

4.9.4 Characteristics of Cross-buying Customers

By finding the characteristics of cross-buying customers, the tool can provide insight into which customers buy combinations of goods. This step will not be used in the calculation the total growth potential. These characteristics can be given by counting the values of these subgroups, e.g. which industries, fte's, location of HQ.

4.9.5 Improving Existing Customers

Once the market basket analysis is finished, customers can be found who's revenue can be improved using the cross-selling combinations. This is a matter of finding customers with predecessors but not successors. If the price is known of the successor, this can be added to create a potential revenue. After this potential revenue is generated, this can be coupled with a probability that this customer will be interested in cross-buying based on its similarity to other customers who cross-buy this combination.

4.10 New Customers

The value prediction of new customer consists of three steps: searching for new customers, analyzing their fit to the current customers, and predicting generated revenue based on similar customers. These steps use the results gathered by customer insights, segmentation, cross-sell potential and the identification of important drivers of customer values.

New customers can be sought for in two ways: using Orbis, CompanyInfo or Similarweb, competitors of valuable customers can be found, and using Google API companies in industries in regions near the client can be found. The companies returned by these queries are then analyzed for their fit to the current batch of customers using the machine learning model trained in customer segmentation. A fit is found by calculating the distance to the center of each cluster. If this distance is within one standard deviation to the centroid, it is accepted.

If a good fit is found, the potential revenue to the client can be predicted using another machine learning tool: kernel density estimation. Simply said, this will take the gaussian average, weighted by similarity to the newly identified customers, of the predicted revenue (not generated revenue, cross-sell potential is taken into account).

The kernel density estimator is a better tool for this job than the regression model generated by identification of the most important driver of customer value, as it is built upon the translation of discrete values to binary columns. This makes the model receptive to overreaction, to which kernel density estimation is less prone.

4.11 Output

After two steps of the process, intermediate results will be provided as output to the user.

After the matching phase, the list of unique customers linked to external databases together with the transactions that are mapped to customers will be made available. This data set will be useful in case the user wants to perform the data analysis manually but doesn't want to perform the time-intensive process of data cleaning and matching.

The second intermediate result that is provided as output is the output of the data collection phase. As described in section 4.7, this data contains some additional statistics about each customer. After this intermediate result is provided, the user will be asked to enter some additional data manually before the analysis can be performed.

After the analysis itself, the final result will be provided as output. Every result of every step of the analysis must be visible in this data set, such that the data set can be used for visualisation.

In order to stay consistent, the same data structure as the input data has to be maintained. Since the input data will be in table format, either Excel or CSV, the same format will have to be outputted. In python, many packages provide the functionality to convert data to table format. Libraries such as python's CSV library, openpyxl, xlxswriter or xlwt have all been considered. However, the Pandas library will be used for this purpose. The main advantage of the pandas library is that data can immediately be transformed into DataFrames, which can directly be used for executing the data analysis. After the data analysis is done the DataFrames can be converted back to table format through the use of a single function call.

4.12 Visualisation

During and after the analysis, several visualisations can be made. Either visualisations during the processing or visualisations of the output of the tool.

4.12.1 Process Visualisation

During processing, it is desirable to have an indication of what the status of the analysis is. To achieve this overview, four possible approaches are available; a progress bar, a percentage, a checklist of all the steps that are being taken during the process, or the actual flowchart of the process viewing where it is running at the moment. The latter option is the most desired because it accurately shows what part of the tool is executed and what still has to be done. The user can effortlessly see the progress of the tool. When a certain task has been done, it will be marked in the flowchart. Later on, it could be added that for every task in the flowchart the progress of each task can be shown in the form of a progress bar or percentage. Besides the progress of the tool, the mapping success rate of the client data to the extracted data from external databases should be shown. This is done so that the client knows how much of their customers were found in external databases.

4.12.2 Output Visualisation

After the analysis, the tool will have a lot of useful output to give back to the client. However, when this is returned as an Excel/CSV file, it is difficult to have a good overview quickly. Because of that, it is desirable to have visualisations of the most important outputs. The following visualisations are most desired:

- Customer concentration.
- How the revenue of the client's revenue was generated.
- Potential new customers, based on their region.
- Cross-sell opportunities.

These visualisations will be created using either React or Matplotlib. These will be the easiest to use because the tool will be programmed in Python (Matplotlib) and already uses React for the front-end as can be seen in Section 4.2.3

References

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the interact'93 and chi'93 conference on human factors in computing systems* (pp. 206–213).

A Sequence Diagram

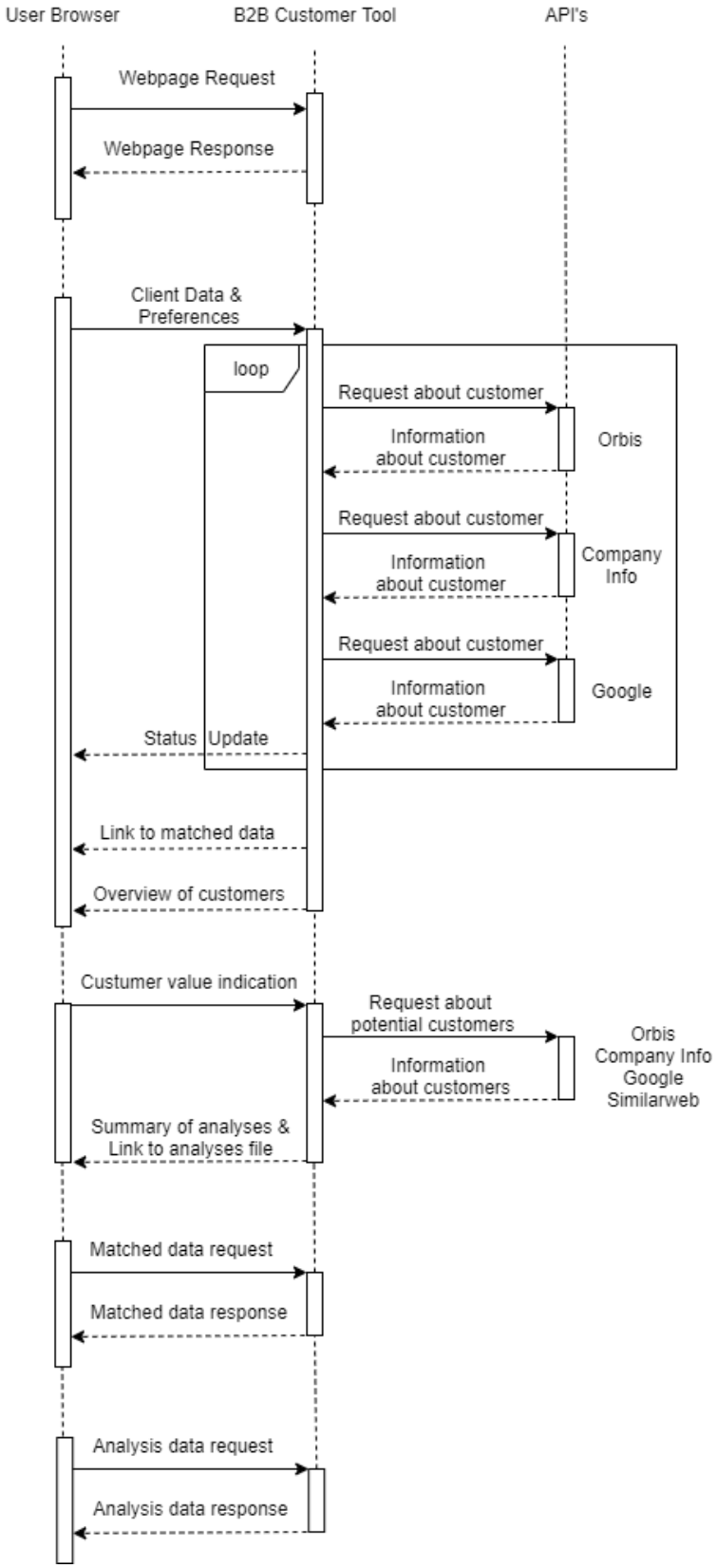
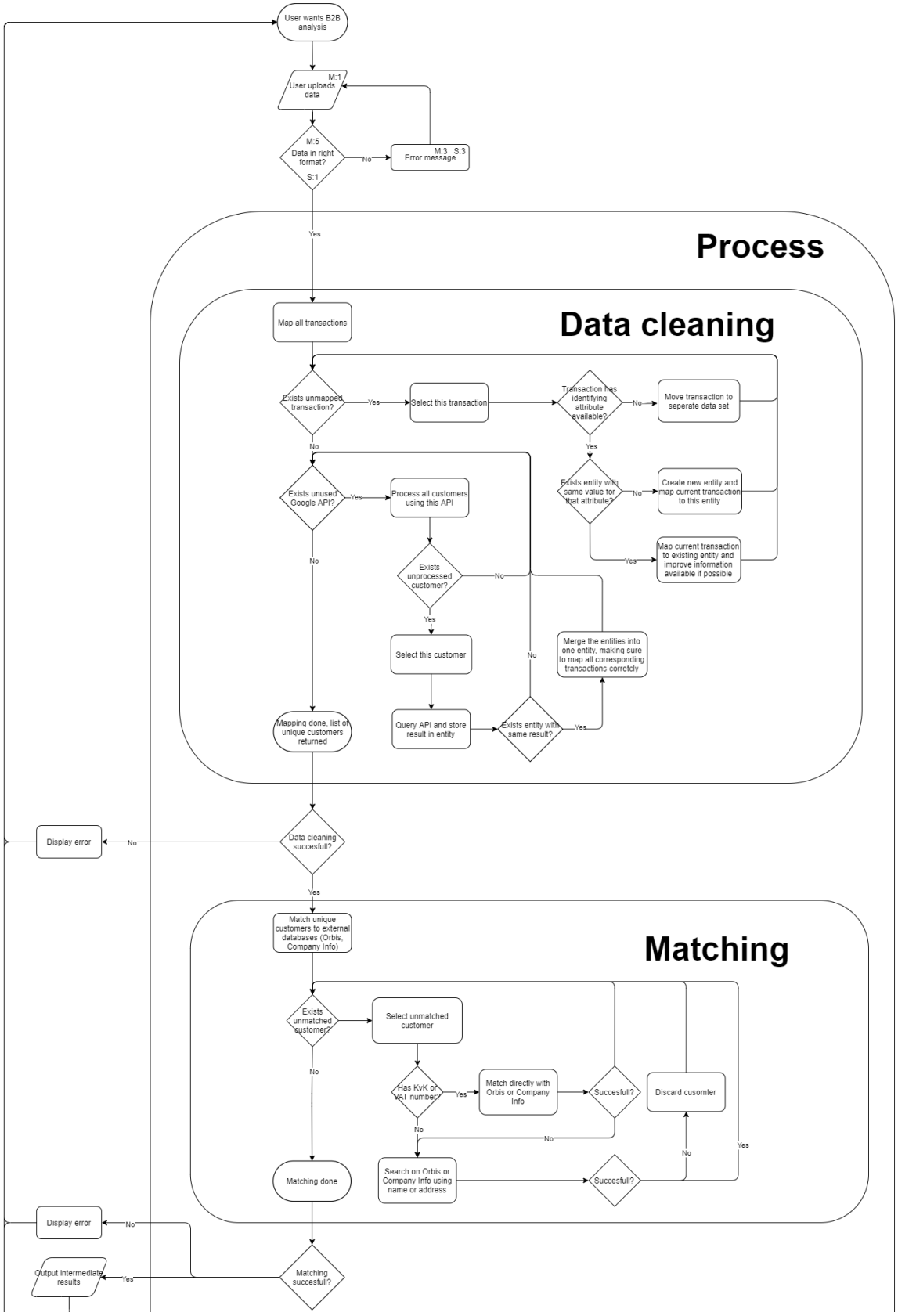
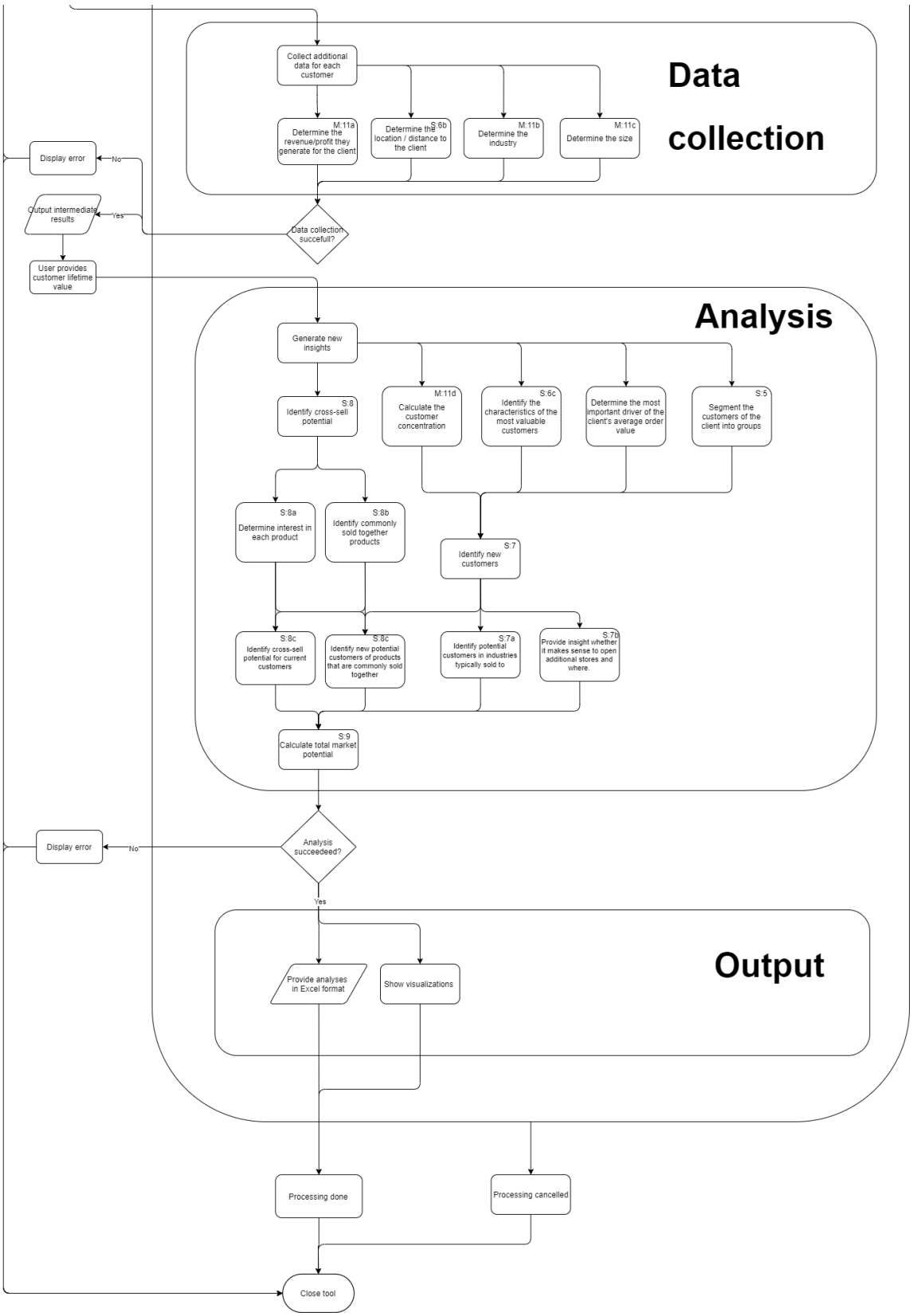


Figure 2: Sequence Diagram of component interaction. Note: Length of the bars does not indicate time.

B Flowchart


User wants B2B analysis
M:1
User uploads data
M:5
Data in right format?
S:1
No
M:3 S:3
Error message
Yes
Process
Data cleaning
Map all transactions
Exists unmapped transaction?
Yes
Select this transaction
Transaction has identifying attribute available?
No
Move transaction to seperate data set
Yes
No
Exists unused Google API?
Yes
Process all customers using this API
Exists entity with same value for that attribute?
No
Create new entity and map current transaction to this entity
Yes
Map current transaction to existing entity and improve information available if possible
No
Exists unprocessed customer?
No
Yes
Select this customer
Merge the entities into one entity, making sure to map all corresponding transactions correctly
No
Mapping done, list of unique customers returned
Query API and store result in entity
Exists entity with same result?
Yes
Data cleaning succesfull?
No
Display error
Yes
Match unique customers to external databases (Orbis, Company Info)
Matching
Exists unmatched customer?
Select unmatched customer
No
Has KvK or VAT number?
Yes
Match directly with Orbis or Company Info
Succesfull?
Discard cusomter
No
No
Yes
No
Matching done
Search on Orbis or Company Info using name or address
Succesfull?
Display error
No
Output intermediate results
Yes
Matching succesfull?


Data collection
Collect additional data for each customer
M:11a
Determine the revenue/profit they generate for the client
S:6b
Determine the location / distance to the client
M:11b
Determine the industry
M:11c
Determine the size
Display error
No
Output intermediate results
Yes
Data collection succefull?
User provides customer lifetime value
Analysis
Generate new insights
S:8
Identify cross-sell potential
M:11d
Calculate the customer concentration
S:6c
Identify the characteristics of the most valuable customers
Determine the most important driver of the client's average order value
S:5
Segment the customers of the client into groups
S:8a
Determine interest in each product
S:8b
Identify commonly sold together products
S:7
Identify new customers
S:8c
Identify cross-sell potential for current customers
S:8c
Identify new potential customers of products that are commonly sold together
S:7a
Identify potential customers in industries typically sold to
S:7b
Provide insight whether it makes sense to open additional stores and where.
S:9
Calculate total market potential
Display error
No
Analysis succeedeed?
Yes
Output
Provide analyses in Excel format
Show visualizations
Processing done
Processing cancelled
Close tool

B REQUIREMENTS

B.1 MUST HAVE

B.1.1 Interface

The tool must have a web-based user interface that lets the user perform the analysis. The interface must allow the user to

1. Upload the client data.
2. Download the raw data output
3. See if an error occurred in the importing, matching or analysis phase.
4. Cancel the analysis.

B.1.2 Data

The tool must be able to

5. Process pre-formatted client data
6. Use the API of external databases to query information.
7. Extract client and customer characteristics from external databases.

B.1.3 Matching

The tool must be able to

8. Remove duplicates in client data.
9. Match customer details to external databases.

B.1.4 Analysis

The tool must generate

10. Insights into the client
    a) Revenue of client.
    b) The industries the client is active in.
11. Insights into the customers of the client.
    a) Average order value.
    b) The revenue each customer generates.
    c) The industries the customers are active in.
    d) Number of unique customers
    e) Average customer size
    f) The concentration of the customer base (which part of revenue comes from which customers).

B.1.5 Output

The tool must output

12. Raw data: internal data matched with external data.
13. Enriched data in table format, either csv or Excel.

B.1.6 Testing and error handling

The tool must

14. Remain online, despite any errors in the analysis.
15. Have at least 80% testing line coverage.

B.2 SHOULD HAVE

B.2.1 Interface

The web interface should allow the user to

1. See if any compulsory data columns is missing.
2. Customise the analysis.
3. Authenticate itself to gain access.

B.2.2 Data

The tool should specify

4. Compulsory and optional data columns.

B.2.3 Matching

The tool must be able to

5. Deal with gaps in client data.

B.2.4 Analysis

The tool should generate

6. The most important driver of the client's average order value.
7. Additional insights into the customers of the client.
   a) Identify the locations of customers.
   b) Identify the characteristics of the most valuable customers.
8. New potential customers:
   a) Identify potential customers in industries typically sold to.
   b) Insight whether it makes sense to open additional stores and where.
9. Cross-sell potential:
   a) Identify customer interest in each product
   b) Identify products that are commonly sold together.
   c) Identify cross-sell potential for current customers
   d) Identify new potential customers of products that are commonly sold together.
10. The total market potential, a sum of current revenue, cross-sell potential, the new customers' potential and the customer insights.
11. An overview of the locations of the client.
12. Customer segments based on predefined methods.

B.2.5 Output

The tool should output

13. Detailed descriptions of how the analysis was done.

B.2.6 Visualisation

The tool should provide visualisations of

14. The mapping success rate of the client data to the extracted data from external databases.
15. The progress of the process in real-time. This includes the progress of API extraction, matching and analysis.

B.2.7 Testing and error handling

The tool should try to

16. Correct errors if they occurred in the importing, matching or analysis phase.

B.3 COULD HAVE

B.3.1 Interface

The interface could allow the user to

1. Only perform certain parts of the analysis.
2. Have basic insight into its usage.
3. See more details on the analysis.

B.3.2 Analysis

The tool could analyse

4. Webpages of the customers with Natural Language Processing to provide additional methods for customer segmentation.
5. The strategies of competitors of the client.
6. The market share of the client or the customers.
7. The general market trends in the industry the client operates.
8. Additional customer insights:
   a) Purchase frequency.
   b) Repeat purchase probability.
   c) Repeat purchase rate.
   d) Loyal customer rate.
   e) Customer retention rate.
   f) Customer churn rate.
   g) New vs returning customer rate.
   h) Group the customers by loyalty: separate returning clients from incidental clients.

B.3.3 Visualisation

The tool could provide more complex visualisations that are normally performed in off-the-shelf industry-standard visualization tools. (What kind of visualizations would this be?, more will be discussed with PwC)

9. The customer concentration
10. How the client's revenue was generated.
11. The potential customers per region, based on the area in which the user operates.
12. The cross-sell potential.
13. ...

B.3.4 Output

The tool could output

14. Intermediate result of the analysis.

B.4 WON'T HAVE

B.4.1 Interface

The web interface won't

1. Allow the user to format its data
2. Show the same page if a process has been started and the page is reloaded by the user; at every refresh of the webpage, the user is presented with the first step of the process.

B.4.2 Data

The tool won't

3. Be able to process unformatted client data.
4. Save the raw data uploaded by the user externally.

B.4.3 Output

The tool won't

5. Save the raw output externally.

B.5 NON-FUNCTIONAL REQUIREMENTS

The tool will

1. Be in English.
2. Be created using the PwC App kit. (Django, Python, Java?) (Will be discussed monday in a meeting with PwC)
3. Have source code that is hosted on a Git server of PwC.
4. Use https to provide encryption, and will not work on HTTP.
5. Come with a docker image to ease installation on a new server.
6. Tool will have a functional build on the 29th of May 2020.
7. Will have a final build before the 3rd of July 2020.
8. Perform its analysis in a reasonable timeframe, where a reasonable timeframe will depend on the size of the client's data. For the majority of the clients of PwC, the tool should performs its analysis in at most two hours.
9. Be usable by people that have limited knowledge of data analytics. An average PwC employee should be able to learn to use the tool in a maximum of two hours. After this training, the average number of errors made by experienced users should not exceed two per session.

C FRONT-END UML & SCREENSHOTS

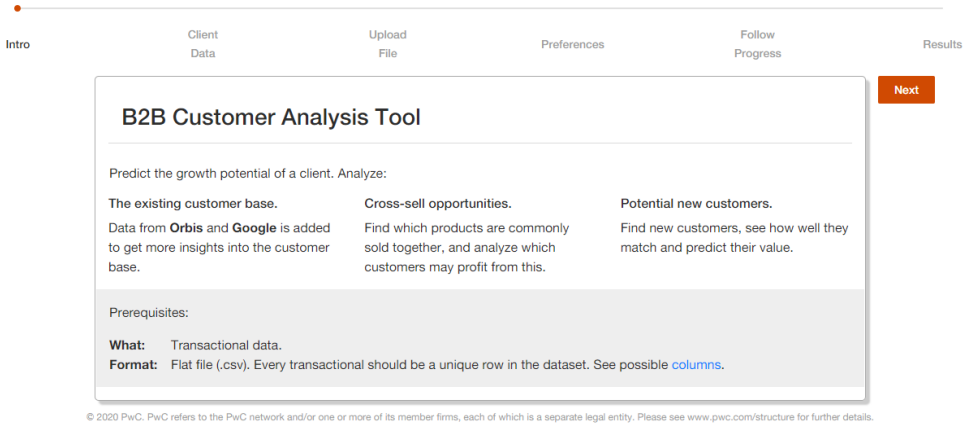


Figure C.1: First page the user sees. The user is introduced to the tool and what it can do.

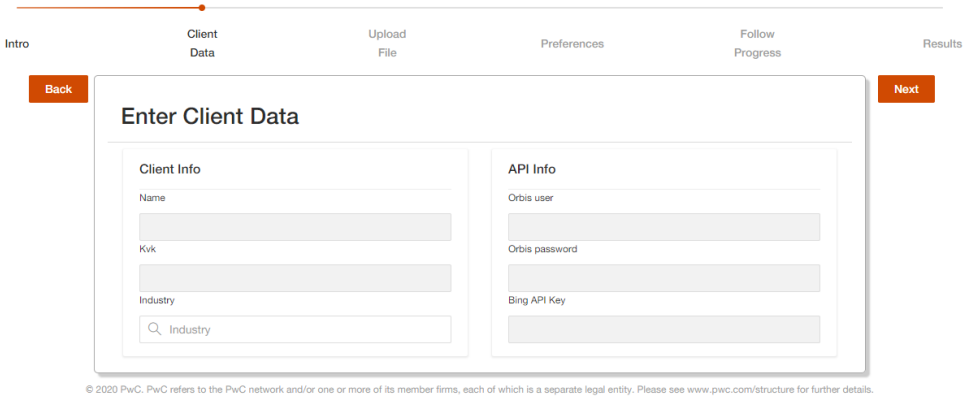


Figure C.2: Second page the user sees. The user is asked to enter information about the clients and the API's to use.

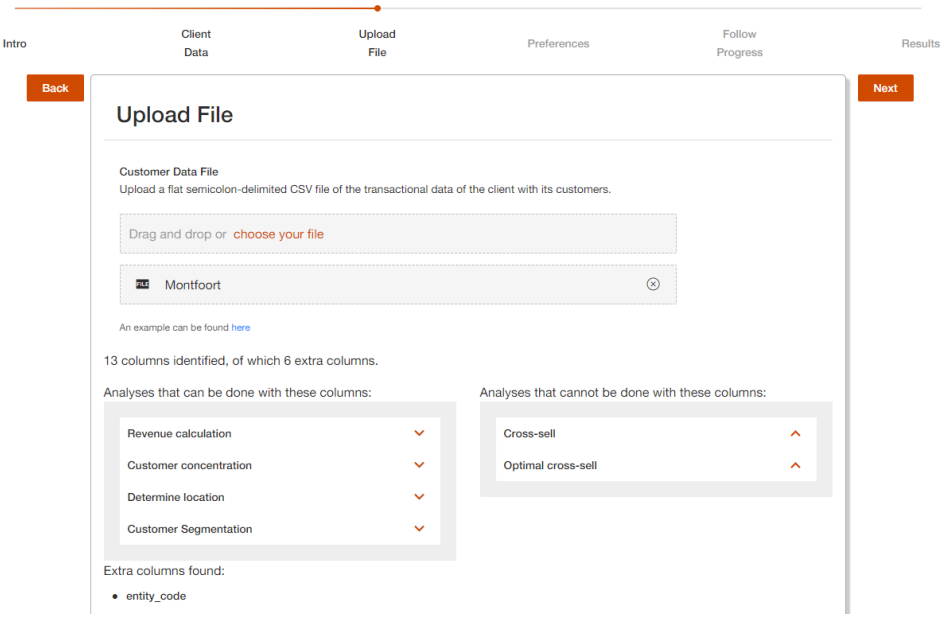


Figure C.3: Third page the user sees. The user has uploaded a file, and a prediction of the analyses which can be done based on this file is given.

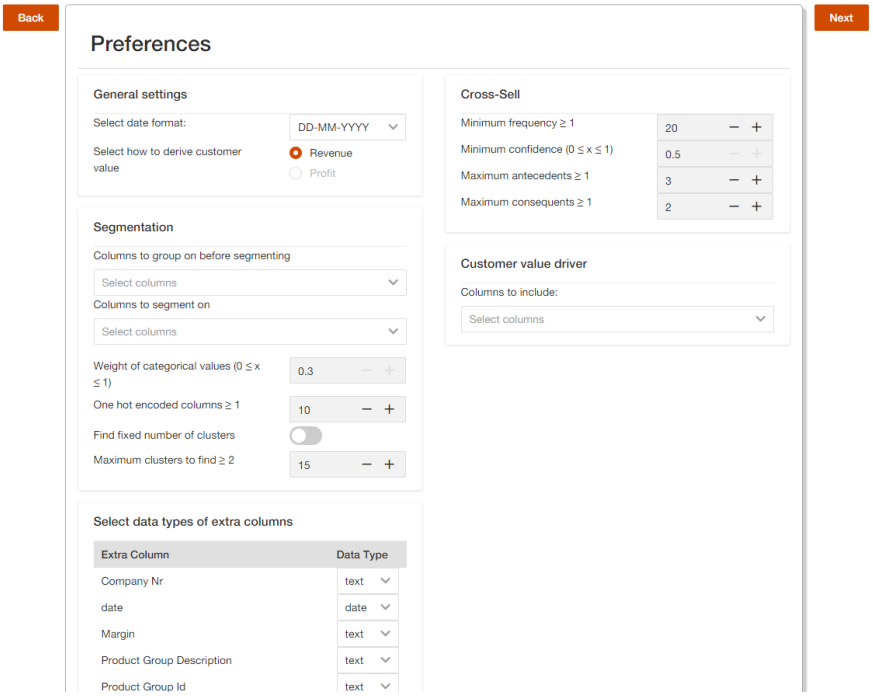


Figure C.4: Fourth page the user sees. The user can edit the settings for the analyses which are to be executed.

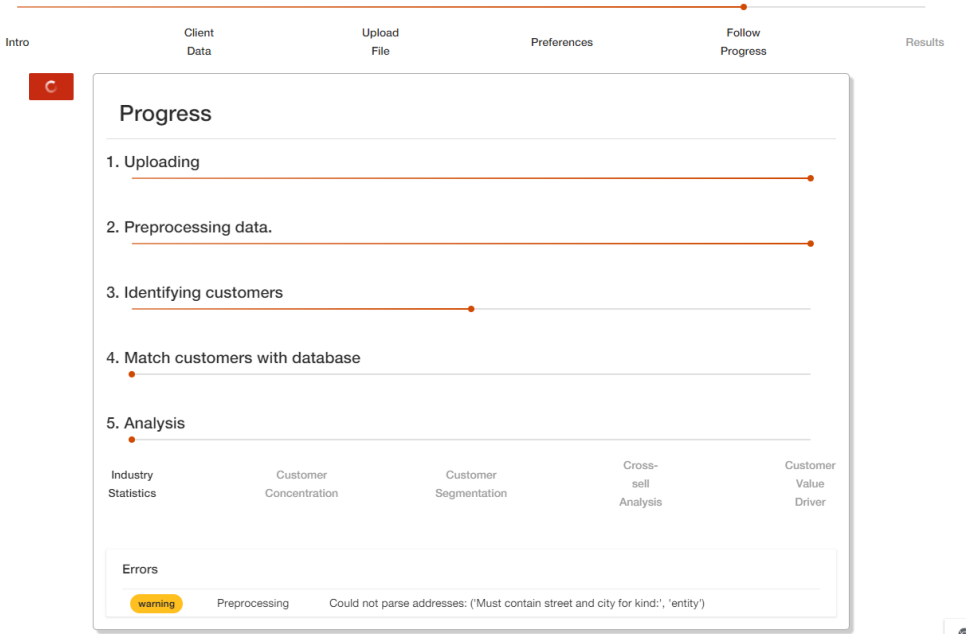


Figure C.5: Fifth page the user sees. The user can edit the settings for the analyses which are to be executed.

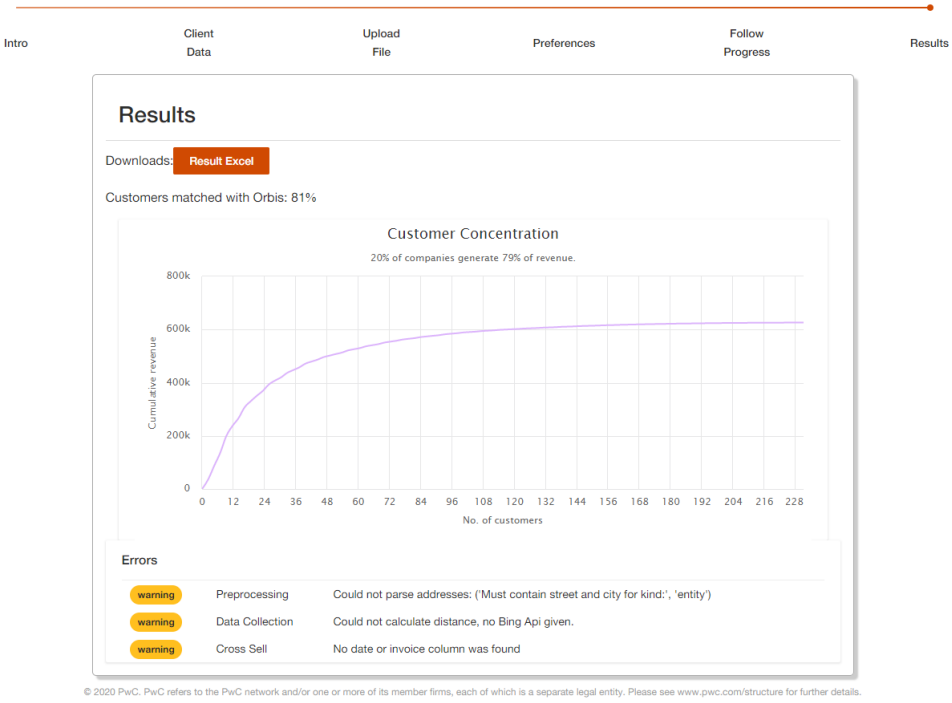


Figure C.6: Sixth and final page the user sees. The user sees information about the results of the analyses, which errors were found and a button to download the results.

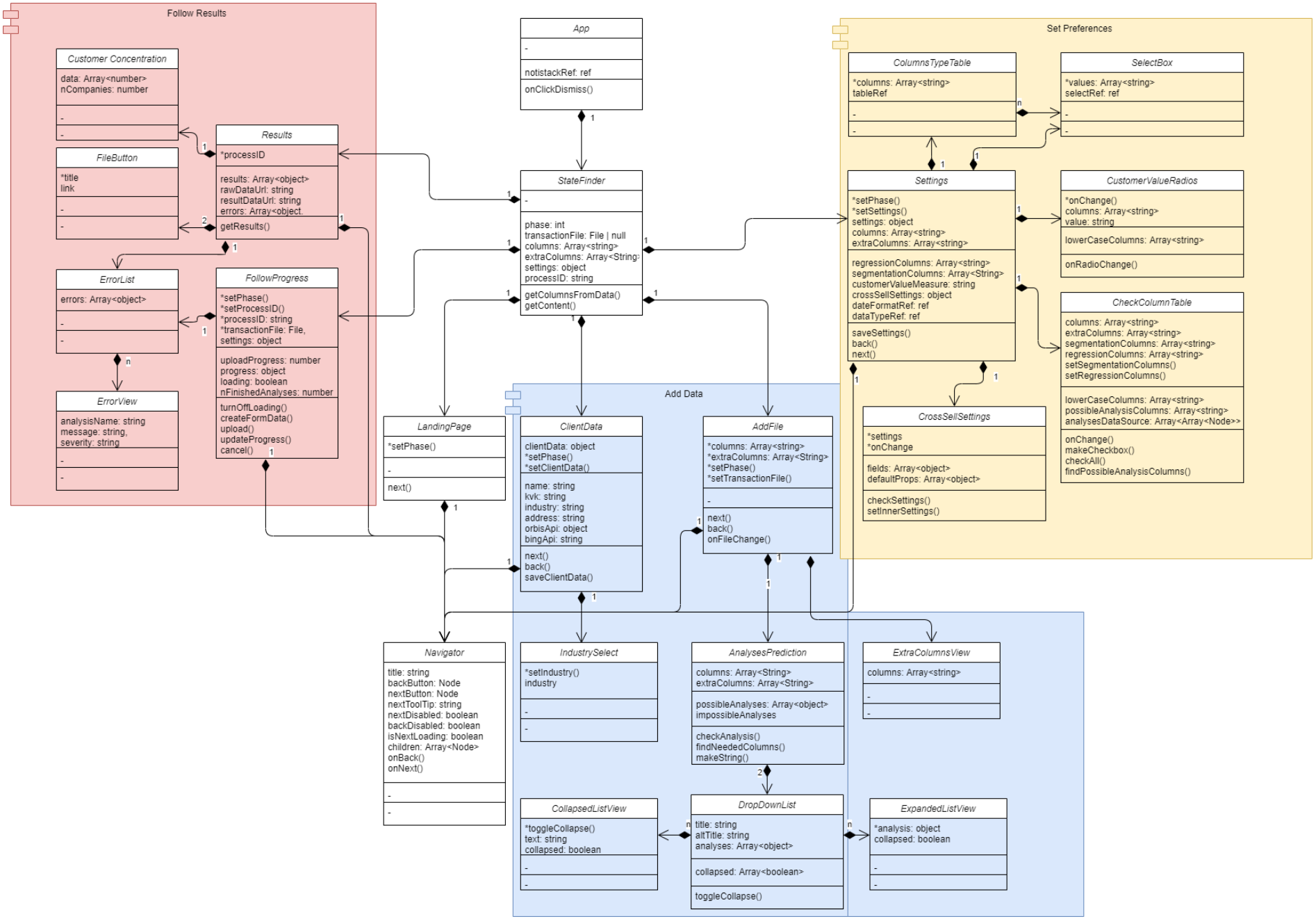
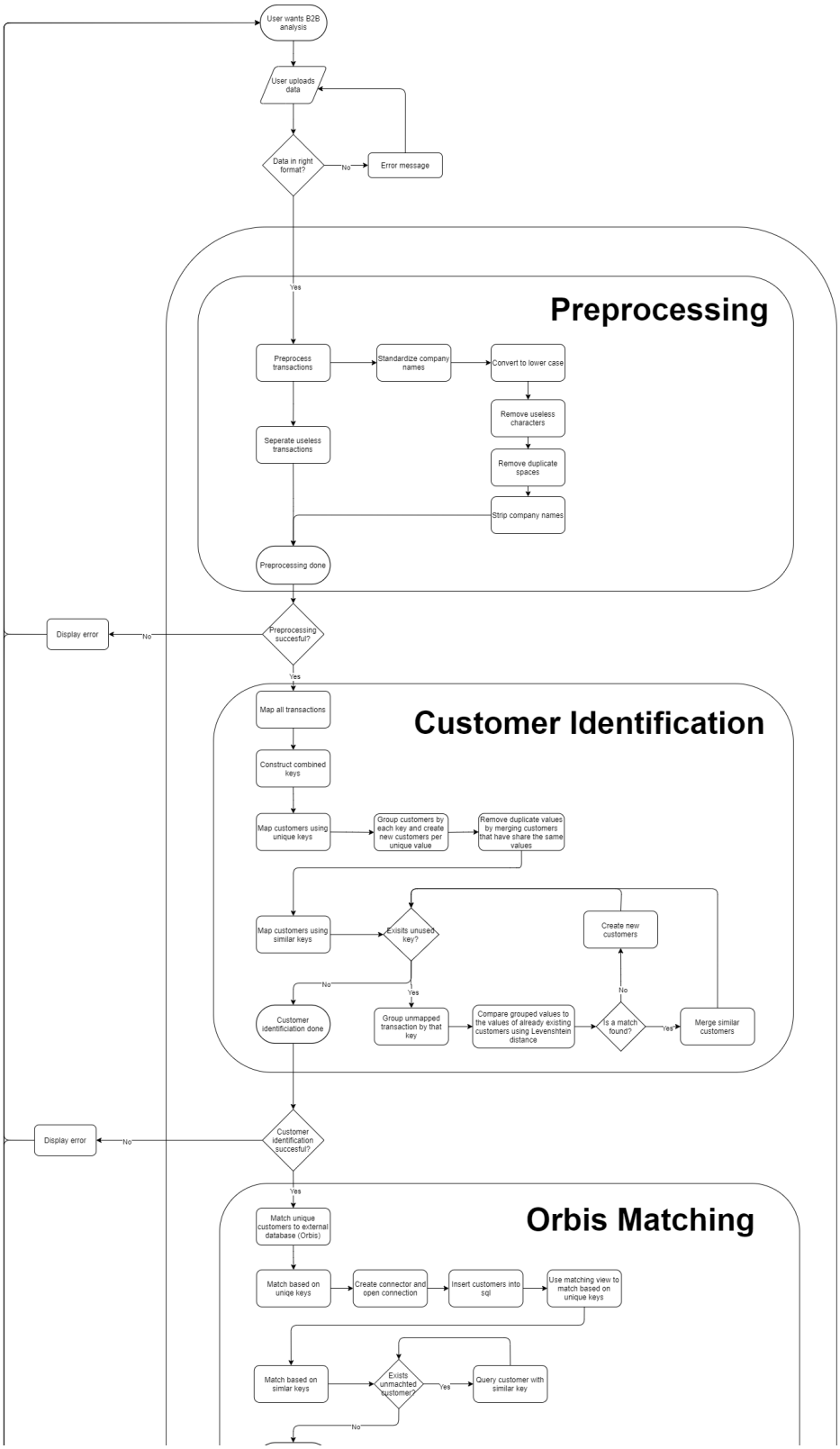
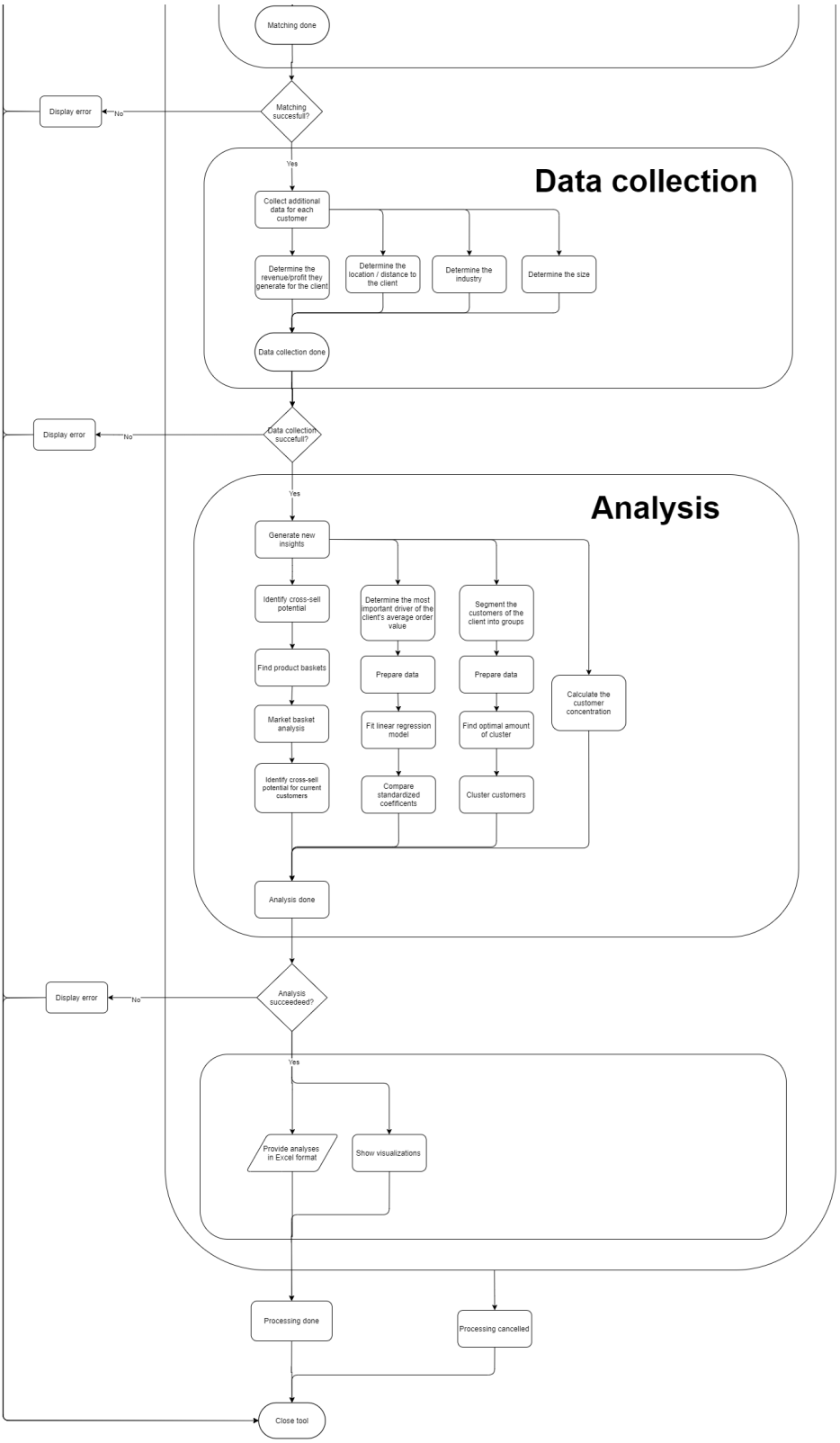


Figure C.7: UML-Layout of the Front-end. The arrows mean composition. The numbers near the arrows indicate how often the component with the black square includes the component with the arrow. Each component specifies the name, its arguments (props, arguments with a star are required, the rest optional), its variables and its functions.

D FLOWCHART


User wants B2B analysis
User uploads data
Data in right format?
No
Error message
Yes
Preprocessing
Preprocess transactions
Standardize company names
Convert to lower case
Remove useless characters
Remove duplicate spaces
Strip company names
Seperate useless transactions
Preprocessing done
Preprocessing succesful?
No
Display error
Yes
Customer Identification
Map all transactions
Construct combined keys
Map customers using unique keys
Group customers by each key and create new customers per unique value
Remove duplicate values by merging customers that have share the same values
Map customers using similar keys
Exisits unused key?
No
Yes
Create new customers
No
Customer identificiation done
Group unmapped transaction by that key
Compare grouped values to the values of already existing customers using Levenshtein distance
Is a match found?
Yes
Merge similar customers
Customer identification succesful?
No
Display error
Yes
Orbis Matching
Match unique customers to external database (Orbis)
Match based on uniqe keys
Create connector and open connection
Insert customers into sql
Use matching view to match based on unique keys
Match based on similar keys
Exists unmachted customer?
Yes
Query customer with similar key
No


Matching done
Matching succesfull?
No
Display error
Yes
Data collection
Collect additional data for each customer
Determine the revenue/profit they generate for the client
Determine the location / distance to the client
Determine the industry
Determine the size
Data collection done
Data collection succesfull?
No
Display error
Yes
Analysis
Generate new insights
Identify cross-sell potential
Find product baskets
Market basket analysis
Identify cross-sell potential for current customers
Determine the most important driver of the client's average order value
Prepare data
Fit linear regression model
Compare standardized coefficents
Segment the customers of the client into groups
Prepare data
Find optimal amount of cluster
Cluster customers
Calculate the customer concentration
Analysis done
Analysis succeedeed?
No
Display error
Yes
Provide analyses in Excel format
Show visualizations
Processing done
Processing cancelled
Close tool

E BACK-END UML

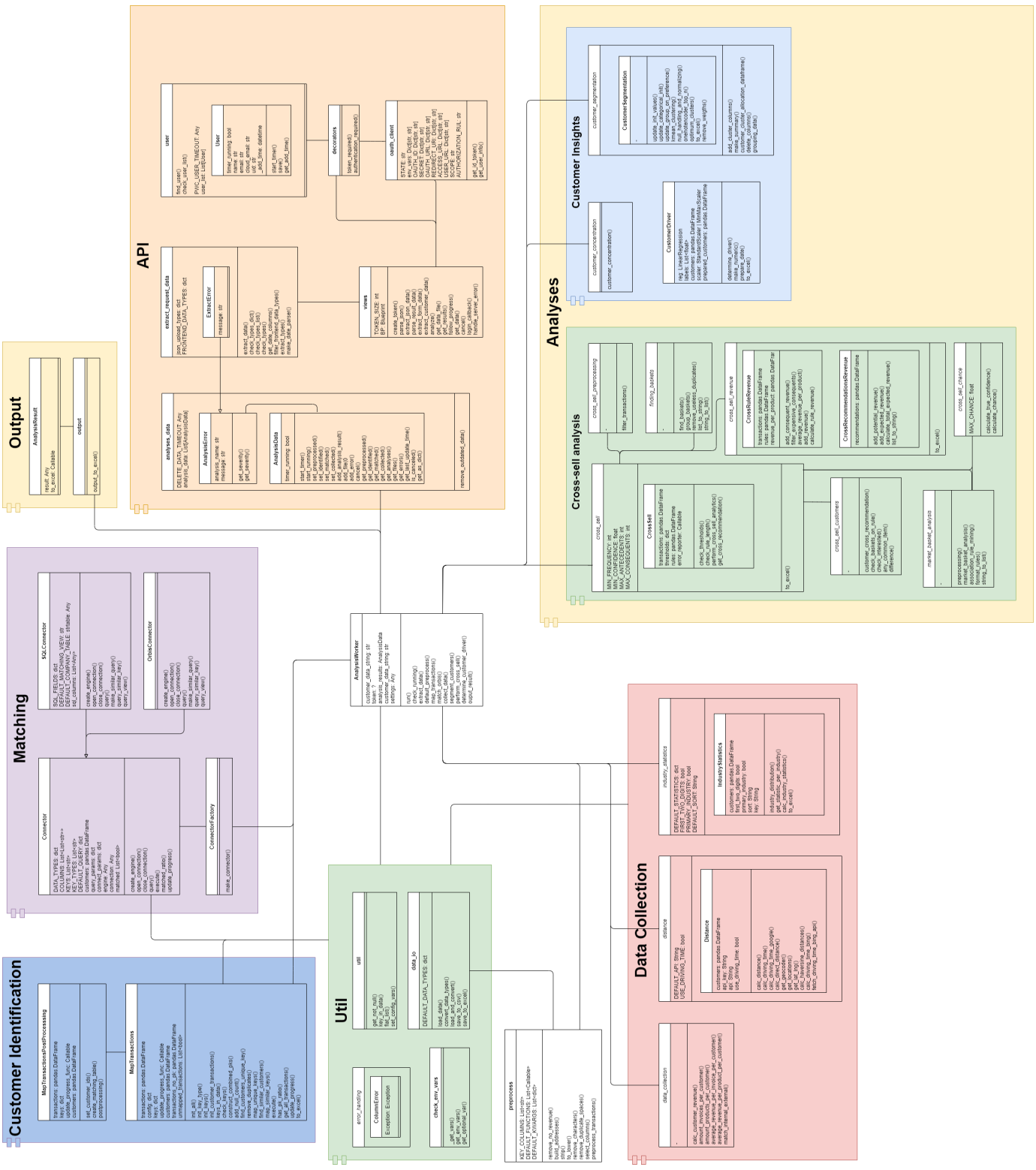


Figure E.1: UML-Layout of the back-end. The white arrow heads indicate inheritance. Each component specifies the name, its variables and its functions.

F API-ENDPOINTS

The following listing contains all the calls made between the front-end and the back-end. The front-end initiates each call, the back-end responds to these calls. The API accepts JSON or Form Data as input, and responds in JSON.

ERRORS

If an error occurred which prohibits the backend from continuing the analysis, the backend communicates this to the frontend using the following response:

```
{
    error: "<message describing the error>"
}
```

ANALYZE

Goal: Analyze the customer data.
Software function: *views.analyze*
Request URL: */api/v1/analyze*
Method: *POST*
Request data: Either JSON or Form Data[1]. Each top-level field must be a different form data input. The contents of the input should be in JSON.

```
{
  customerData: "<CSV string>", REQUIRED
  clientData: {
      name: "<name of client>",
      kvk: "<kvk number of client>",
      industry: "<industry of client>",
      city: "<city of address of client>",
      zipCode: "<zipcode of address of client>",
      street: "<street of address of client",
      houseNumber: "<house number of address of client>"
  },
  regression: {
      columns: {
          default: [array of column names],
          extra: [array of column names]
      },
  },
  crossSell: {
      minFrequency: Number,
      minConfedence: Number,
      maxAntecedents: Number,
      maxConsequents: Number,
  },
  segmentation: {
      maxClusters: Number,
      nClusters: Number,
      categoricalWeight: Number,
      oneHot: Number,
      groupColumns: {
          default: [array of column names],
          extra: [array of column names]
      },
      columns: {
```

1 How browsers send data from a form. Standard: https://tools.ietf.org/html/rfc2388

```
            default: [array of column names],
            extra: [array of column names]
        },
    customerValueMeasure: "<name of column>",
    dataTypes: [array of
      {
          column: "<column name>",
          type: "string | integer | float",
      }
    ],
    dateFormat: "<Python parseable date format (%d-%M-%Y)>",
    orbis: {
      name: "<name of orbis user>",
      password: "<password of orbis user>"
    },
    bingApi: "<API Key of Bing API>",
}
```

Response on success:

```
{
    token: "<16 random characters>"
}
```

FOLLOW PROGRESS

Goal: Get the progress of a running analysis.
Software function: *views.follow_progress*
Request URL: */api/v1/progress*
Method: *GET*
Request data: token (16-character string)
Response: JSON

```
{
    preprocessed: <percentage>,
    identified: <percentage>,
    matched: <percentage>,
    collection: <percentage>,
    results: {
        "<analysis name>": "<analysis result to show in front-end>"
        ...
    },
    errors: [array of
        {
            analysisName: "<name of analysis where error occurred>",
            message: "<text describing the error>",
            severity: "<severity of the error: info, warning or error>"
        }
    ],
    lastModified: "<timestamp of date when results with this token were last mod
}
```

GET RESULTS

Goal: Get the results and list of files of a finished analysis.
Software function: *views.get_results*
Request URL: */api/v1/results*
Method: *GET*
Request data: token (16-character string)
Response: JSON

```
{
    rawDataUrl: "<url to the raw customer data .csv file>",
    resultDataUrl: "<url to the excel file containing the analysis results>",
```

```
    results: {
        "<analysis name>": "<analysis result to show in front-end>"
        ...
    },
    errors: [array of
        {
            analysisName: "<name of analysis where error occurred>",
            message: "<text describing the error>",
            severity: "<severity of the error: info, warning or error>"
        }
    ]
    files: [list of filenames],
}
```

DOWNLOAD FILE

Goal: Download a file containing the results of an analysis. Name of file can be retrieved using */api/v1/results*
Software function: $views.get_data$ Request URL: $/api/v1/ < name > .xlsx$ or $api/v1/ < name > .csv$
Method: *GET*
Request data: token (16-character string)
Response data: bytestream with file contents.

CANCEL ANALYSIS

Goal: Cancel a running analysis.
Software function: *views.cancel*
Request URL: */api/v1/cancel*
Method: *GET*
Request data: token (16-character string)
Response data: Nothing on success.

LOGIN

Goal: Verify and remember a user after it has logged in with the PwC SSO.
Software function: $views.login_callback$
Request URL: */api/v1/callback*
Method: *GET*
Request data:

- state: number provided by the PwC SSO server.
- code: temporary code identifying the user.

Response: Redirection to */index.html* on success.

G | SQL

```
CREATE VIEW match_unique as
(
SELECT
cd.customer_id
, CASE WHEN matching_final.id IS NOT NULL THEN 1 ELSE 0 END
    ↪ AS match_in_orbis
,matching_final.company_name
,matching_final.address_line1
,matching_final.city
,matching_final.nace2_primary_code
,matching_final.nace2_secondary_code
,matching_final.guo_name
,matching_final.prma
,matching_final.opre
,matching_final.empl
,matching_final.trade_register_number
,matching_final.european_vat_number_stripped
FROM B2B_Tool.dbo.customers as cd

LEFT OUTER JOIN
(
SELECT
matching_no_duplicates.customer_id
,matching_no_duplicates.id
,matching_no_duplicates.null_count
,matching_no_duplicates.company_name
,matching_no_duplicates.address_line1
,matching_no_duplicates.city
,matching_no_duplicates.nace2_primary_code
,matching_no_duplicates.nace2_secondary_code
,matching_no_duplicates.guo_name
,matching_no_duplicates.prma
,matching_no_duplicates.opre
,matching_no_duplicates.empl
,matching_no_duplicates.trade_register_number
,matching_no_duplicates.european_vat_number_stripped
FROM
( --matching
SELECT
matching.customer_id
,matching.id
,matching.null_count
,matching.company_name
,matching.address_line1
,matching.city
,matching.nace2_primary_code
,matching.nace2_secondary_code
,matching.guo_name
,matching.prma
,matching.opre
,matching.empl
,matching.trade_register_number
,matching.european_vat_number_stripped
,ROW_NUMBER() over (partition By matching.customer_id order
    ↪ by matching.null_count  asc) as final_ranking
FROM
(
```

```
--kvk matching
SELECT
match_kvk.customer_id
,match_kvk.id
,match_kvk.null_count
,match_kvk.company_name
,match_kvk.address_line1
,match_kvk.city
,match_kvk.nace2_primary_code
,match_kvk.nace2_secondary_code
,match_kvk.guo_name
,match_kvk.prma
,match_kvk.opre
,match_kvk.empl
,match_kvk.trade_register_number
,match_kvk.european_vat_number_stripped
FROM
(
SELECT
cd.customer_id
,company.id
,company.null_count
,company.company_name
,company.address_line1
,company.city
,company.nace2_primary_code
,company.nace2_secondary_code
,company.guo_name
,company.prma
,company.opre
,company.empl
,company.trade_register_number
,company.european_vat_number_stripped
,ROW_NUMBER() over (partition By company.
    ↪ trade_register_number order by company.null_count asc)
    ↪ as ranking
FROM company
INNER JOIN B2B_Tool.dbo.customers AS cd
ON cd.customer_kvk=company.trade_register_number
)
as match_kvk
where match_kvk.ranking=1
-- end kvk matching

union all

-- vat matching
SELECT
match_vat.customer_id
,match_vat.id
,match_vat.null_count
,match_vat.company_name
,match_vat.address_line1
,match_vat.city
,match_vat.nace2_primary_code
,match_vat.nace2_secondary_code
,match_vat.guo_name
,match_vat.prma
,match_vat.opre
,match_vat.empl
,match_vat.trade_register_number
,match_vat.european_vat_number_stripped
FROM
(
SELECT
```

```
cd.customer_id
,company.id
,company.null_count
,company.company_name
,company.address_line1
,company.city
,company.nace2_primary_code
,company.nace2_secondary_code
,company.guo_name
,company.prma
,company.opre
,company.empl
,company.trade_register_number
,company.european_vat_number_stripped
,ROW_NUMBER() over (partition By company.
    ↪ european_vat_number_stripped order by company.
    ↪ null_count asc) as ranking
FROM company
INNER JOIN B2B_Tool.dbo.customers AS cd
ON cd.customer_vat=company.european_vat_number_stripped
)
as match_vat
where match_vat.ranking=1
-- end vat matching

union all

-- address matching
SELECT
match_address.customer_id
,match_address.id
,match_address.null_count
,match_address.company_name
,match_address.address_line1
,match_address.city
,match_address.nace2_primary_code
,match_address.nace2_secondary_code
,match_address.guo_name
,match_address.prma
,match_address.opre
,match_address.empl
,match_address.trade_register_number
,match_address.european_vat_number_stripped
FROM
(
SELECT
cd.customer_id
,company.id
,company.null_count
,company.company_name
,company.address_line1
,company.city
,company.nace2_primary_code
,company.nace2_secondary_code
,company.guo_name
,company.prma
,company.opre
,company.empl
,company.trade_register_number
,company.european_vat_number_stripped
,ROW_NUMBER() over (partition By company.address_line1,
    ↪ company.city order by company.null_count asc) as
    ↪ ranking
FROM company
INNER JOIN B2B_Tool.dbo.customers AS cd
```

```
ON (cd.customer_address_street=company.address_line1 AND cd.
    ↪ customer_address_city=company.city)
)
as match_address
where match_address.ranking=1
-- end address matching

union all

-- name matching
SELECT
match_name.customer_id
,match_name.id
,match_name.null_count
,match_name.company_name
,match_name.address_line1
,match_name.city
,match_name.nace2_primary_code
,match_name.nace2_secondary_code
,match_name.guo_name
,match_name.prma
,match_name.opre
,match_name.empl
,match_name.trade_register_number
,match_name.european_vat_number_stripped
FROM
(
SELECT
cd.customer_id
,company.id
,company.null_count
,company.company_name
,company.address_line1
,company.city
,company.nace2_primary_code
,company.nace2_secondary_code
,company.guo_name
,company.prma
,company.opre
,company.empl
,company.trade_register_number
,company.european_vat_number_stripped
,ROW_NUMBER() over (partition By company.company_name order
    ↪ by company.null_count asc) as ranking
FROM company
INNER JOIN B2B_Tool.dbo.customers AS cd
ON (cd.customer_name=company.company_name)
)
as match_name
where match_name.ranking=1
-- end name matching


) as matching
) as matching_no_duplicates where final_ranking=1
) as matching_final
ON cd.customer_id = matching_final.customer_id
)
```

H | PROJECT DESCRIPTION

1. WHAT YOUR COMPANY IS WORKING ON AS BACKGROUND INFORMATION?

PwC Deal Analytics is a part of the M&A department of PwC where we provide data driven insights to improve decision making and value creation throughout the entire deal lifecycle. Typical projects include:

1. Financial modelling with very large and complex data sets
2. Providing commercial, financial and operational insights
3. Creating high-impact dashboards in Tableau / PowerBI
4. Natural language processing
5. Strategic data-driven analyses and prediction such as online price benchmarking and geospatial forecast models.

PwC

PwC is a multinational professional services network of firms. PwC is organized into the following three service lines: Assurance, Tax and Advisory. Deal Analytics is part of the M&A department which is part of the Advisory service line.

2. WHAT STUDENTS WILL BE RESEARCHING / DEVELOPING FOR THE PROJECT?

The purpose of this project is to create a tool that automatically generates relevant analyses for a B2B company. In our current B2B customer analysis we provide our clients with customer insights as well as identify new customers and cross-sell potential. We want to improve and automate these analyses by creating a tool.

Problem

When a company ('target') is sold there is a short period of time during which the buyer can get familiar with the target company. If the Target is a B2B company, the buyer wants to understand the type of customers that the Target serves as well as to identify further growth opportunities. For this, the Deal Analytics team of PwC repeatedly created several analyses using self-service data analytics tools. To deliver such services in larger volume, proprietary tool support is required. Also, we think our analysis could be further improved by application of more advanced data analytics techniques that we cannot execute quickly enough when done with our more generic tool suite. Therefore, we would like to create a web-based tool, that creates more analyses in a shorter period as well as improves our existing analyses.

Design decisions necessary to solve

1. What analyses should the tool create?
2. How to create an intuitive interface overall?
3. How to automatically match customers with external databases?
4. Whether it makes sense to add a learning algorithm to the third party matching algorithms that we usually deploy (e.g. matching provides by Orbis, indirectly through the Google API)? If so, how you would present this in a user interface.

5. The choice of technology will be determined in discussion with the group at the beginning of the project. PwC uses a combination of Java, Django and the proprietary PwC App kit to implement the PwC look-and-feel. Ideally, the tool should be developed on this platform or, at least, it should be easily portable towards such a platform

Desired tool requirements

1. This tool has an intuitive user interface i.e. the user interface should be usable by people that have a limited knowledge of data analytics
2. The tool should be able to digest files with different structures although a certain level of pre-processing can be assumed. Input is likely Excel or CSV.
3. The tool should match customer details (name, Chamber of Commerce registration ID etc.) to external databases and extract client characteristics from these databases.
4. The tool should use machine learning algorithms on top of the third party matching algorithms if this adds sufficient additional matching precision
5. The tool should be able to classify customers in pre-defined categories based on their homepage using an efficient combination of keyword search and natural language procesing
6. The tool should be able to visualize several standardized analyses. However, as with ingestion of files, it can be assumed more complex visualisations is out of scope as we would generally perform these in off-the-shelf industry standard visualization tools.

Desired output (Analyses)

The tool will generate based on client data relevant insights for a B2B company, relevant insights are among others:

1. Raw output providing enriched customer data
2. Which characteristics predict which customers are most valuable?
3. What are cross-sell opportunities for the current customers? (Could be based on machine learning)
4. What are potential new valuable customers? Organisational support We will support the project team with hands-on support of a Deal Analytics team member to ensure the project team has the right data and information to work from as well as arrange access to users to test (prototypes of) the tool. On demand support will be available from a PwC internal web application design team. At least once every 2 weeks, there will be a project team meeting with a senior PwC representative to discuss progress and make priority decision where required.

3. ADDITIONAL PRACTICAL INFORMATION

1. Our office is located in Amsterdam. However, at this moment our team as well as all interns work from home due to the Covid restrictions
2. Interns will receive a PwC laptop to work and/or test the tool on during the internship as the added security features on the laptop facilitate working with more realistic data sets
3. We intend to share the results of the project publicly if the results are indeed as we hope (for instance on LinkedIn) aligned to PwC external communication policies

OTHER INFORMATION

Project is reserved for Sander van Leeuwen, Bas Volkers, Cherwin Ort, and Tjard Langhout.

I | INFO SHEET

PwC – B2B Customer Analytics
PricewaterhouseCoopers
1st of July 2020

CHALLENGE: Automating the B2B customer analyses performed by the Deal Analytics team at PwC, presented in a fully functional web-page. The analyses contain; customer matching, data collection, customer concentration, customer driver, customer segmentation, and cross-sell.

RESEARCH: The researched focused on how every analysis could be made and how the eventual software would take shape, including constructing a complete design of the system and setting the requirements.

PROCESS: Apart from the given schedule by course, we use scrum as our methodology. We had weekly sprints to keep agile, even in the short period of time for the project. Some processes took longer than expected due to authority issues or change in desire for the output of an analysis. For the authority issue, we put as much pressure on the authorities that could help us and if we were not granted access we made a workaround. For the other issue, we allocated more time to these analysis to implement the best solutions we could find. Due to COVID-19, we had to work from home, but were not obstructed in our work because of this.

PRODUCT: The product that was created is a web-based application, where client data can be submitted. The submitted data will be used for the analyses and the application will give the results of the performed analyses. The software was tested both statically as dynamically. The dynamical part includes unit, integration, and system testing. Apart from one should-have requirement, all must- and should-have requirements were satisfied.

OUTLOOK: There is still an extra feature that could be implemented, on some of the current analyses there is also room for extensions/improvements. PwC is pleased with the results of the tool and is going to use it for real recommendations to their clients.

Name	**Interest**	**Contribution**
Tjard Langhout	Robotics	back-end developer
Sander van Leeuwen	Developing business sense	front-end developer
Cherwin Ort	Data mining	back-end developer
Bas volkers	Finance	back-end developer

ADDITIONAL INFORMATION

ir. M. Kerkhof	PwC
drs. L. Gunneweg	PwC
drs. T. Boevink	PwC
prof.dr.ir. D.H.J. Epema	Delft University of Technology
Tjard Langhout	Team member
Sander van Leeuwen	Team member
Cherwin Ort	Team member
Bas Volkers	Team member