Claims Analysis and Fraud Detection for Claims Made Using Online Claim Forms

Author: Aleksandar Saša Janjanin (s3169618)

Supervisor: dr. Rein Smedinga

External Supervisor: Tjeerd Moes

July 1, 2019
Abstract

With the increase in capabilities of automatising business procedures with the application of computing, the possibilities for new projects opened in the corresponding fields. Highly suitable for this type of automatisation are the companies which deal with problems based on the input of users. Naturally, one big group of these companies are insurances. The enormous volume of claims being submitted is burdensome and time consuming, so this thesis tried to solve these issues, and make that data efficiently scalable. In order to give the answer to the question whether the process of assessing the insurance claims and checking whether they are fraudulent be safely automated, and if so to which extent, various disciplines of Computing Science were examined. Namely, these included data science, data processing, machine learning, and web development. The data in question was obtained through a company, and all of it was submitted by real users. Due to the scope of the project, and the available resources, the outcome defined the way to approach and solve the issue of the syntactic analysis of claims, and whether they were suspicious, or they seemed consistent. However, other possible solutions were discussed as well.
Preface

The project behind this thesis came to life thanks to the collaboration with Belsimpel and people working there. The company intrigued me with the idea of this project and the possibility of conducting research and doing the development work on the large-scale platform necessary for the scope of it.

It is my belief that through incorporating this project into their business model, and through consulting them and the developers working there, I have made this project relevant and usable for both the industry and academia through many aspects. Moreover, because of the size of the entire endeavour, future studies can be based on some of the components of it in order to improve or apply new ideas which might be a better fit.
# Table of Contents

1 **Introduction** ................................................. 1  
   1.1 Belsimpel and Tulip Assist ............................ 1  
   1.2 Tulip Assist Platform .................................. 1  
   1.3 Claim Procedure ......................................... 2  

2 **Preparation Work** ........................................ 2  
   2.1 Technology Stack ........................................ 2  
   2.2 Requirements Elicitation ............................... 4  
   2.3 Related Scientific Work ............................... 4  

3 **The problem** ............................................... 4  
   3.1 Project Goals ........................................... 5  

4 **Existing Solutions** ....................................... 5  
   4.1 Flaws in Existing Solutions ......................... 6  
   4.2 Conclusion on Existing Solutions ................... 6  

5 **Data Collection** .......................................... 7  
   5.1 Relevant Data ........................................... 7  
   5.2 Collection Process ..................................... 10  
   5.3 Storing the Data ....................................... 12  
   5.4 Data Quality and Quantity ............................ 13  
   5.5 Data Usability .......................................... 15  

6 **Data Processing - Manual Processing** .................. 16  
   6.1 Collected Data ........................................... 16  
   6.2 Metric Extraction ....................................... 17  
   6.3 Data Readability ....................................... 21  
   6.4 Possibility for Machine Learning .................... 22  
   6.5 Preparing the Data For Machine Learning .......... 23  

7 **Data Processing - Machine Learning** ................... 23  
   7.1 Why Machine Learning ................................... 23  
   7.2 PHP and Machine Learning .............................. 24  
   7.3 Supervised or Unsupervised Learning ................ 25  
   7.4 Algorithm Used ......................................... 26  
   7.5 Machine Learning Outcome and Results ............. 27
1 Introduction

In this thesis I will try to answer the question "Can the process of assessing the insurance claims and checking whether they are fraudulent be safely automated, and if so to which extent?". I will do this through close collaboration with the Belsimpel company, and using the possibilities and resources provided by them. However, the methods, insights, and solutions can be useful and easily adapted to other companies in the industry, or can be used as a base for further research.

Since the described research is connected with the firm, it is worth describing it in more detail. Also, it is helpful to describe the relevant parts of their current platform, as they will be referenced throughout this document.

1.1 Belsimpel and Tulip Assist

Belsimpel is a company which has the main interest in phones and phone subscription contracts [3]. Nonetheless, over time it has spread its interest to numerous other related areas. One of which has been the product of a logical step to get into the area of selling phone insurances.

The official title of this insurance part of the company is Tulip Assist, and it provides different coverage plans for its users, based on their selection [11]. Given that it is an insurance, the customers are bound to make claims in order to start the process of obtaining the coverage to which they are entitled with respect to the terms dictated by the contract.

Overall, although Belsimpel is present in the markets of different countries, the Tulip Assist insurance is currently only available to the customers residing in The Netherlands. Moreover, at the moment, it is exclusive to the customers who decide to buy a device through Belsimpel.

1.2 Tulip Assist Platform

The main place which facilitates the Tulip Assist product is its platform, comprised out of the two main components - Frontoffice and Backoffice. First is accessible to the customers and handles the overview of their subscription, as well as all of the input from them, which is mainly the claim forms. On the other hand, the latter serves as a dashboard for the employees working with Tulip Assist. There they can see all of the users and their data, Namely, their purchases, insurances, and claims.
1.3 Claim Procedure

The current procedure for submitting a claim is to log in to the official website of Tulip Assist using one's credentials. Once logged in, the user should proceed to the claim page which would have preset questions for him to fill in. Of course, based on the different types of accidents, different questions will be asked.

After the user fills in the question, the final version of the claim form is submitted and sent to the claims department to be processed by hand. This is a time consuming process which heavily depends on the final version of the claim only, as well as the subjective assessment of the person processing the said claim. These two issues are the in the core of what this project is trying to resolve.

2 Preparation Work

Bearing in mind that the scope of the project was quite broad, the planning was the key, and one of the main points in the process. The freedom to work on the project with only the general guidelines asked for the possibilities to be analysed and decided on in a thorough and precise way. This should have in turn provided a stable foundation which enabled a path without preventable obstacles.

However, in order to achieve this, the capabilities and restrictions of the code base needed to be realised. These would dictate the approach and the decision choices.

2.1 Technology Stack

The technology stack of the Tulip Assist project was comprised of a mixture of modern and legacy technologies. This occurrence was a product of a situation where a custom framework became too complex to migrate or have all of its components modernised. Mainly due to the fact that its dependencies became quite interwoven over the time.

Because of this, when some of them stopped being updated and developed, or received a better competitor, like PHP ActiveRecord and Smarty, respectively, they could not be replaced because of the time required to do so. Moreover, the entire project would have benefited if the framework was to be updated to Laravel, the more efficient alternative which contains two modern alternatives for the aforementioned technologies - Eloquent and Blade.
On the other hand however, all of the to some extent independent components have been updated regularly and continued giving their contribution to the ease of development by enabling modern features. These are predominantly the JavaScript frameworks and libraries, as well as the languages themselves, namely PHP, JavaScript, HTML, CSS and SQL.

Given is the detailed description of the technology stack used.

- Front end:
  - JavaScript
  - CSS3
  - HTML5
  - Smarty Templating Engine
    - In the beginning of the project:
      - JQuery
      - Bootstrap
    - In the second half of the project:
      - Vue.js
      - Vuetify

- Back end:
  - PHP 7.1
  - PHP ActiveRecord

- Database:
  - PHP MyAdmin
  - MariaDB
  - MySQL

Upon getting familiar with all of these, and realising their possible advantages and disadvantages, further technical steps were possible. These included comprehending the code and its structure, requirements elicitation, looking for related scientific work, and establishing a plan with precise steps.
2.2 Requirements Elicitation

The General rule was set that the project owner was to represent the claim department. This decision saved a lot of time, as there was no overhead, and his familiarity with the way claims were handled was equal to that of those working in the said department.

As a normal consequence of this, the project owner was the person with whom the requirements solicitation was done. The results of this process are summarised in the next section in full detail. Great attention was directed towards avoiding the problems of scope, understanding, and volatility [4]. This proved as a valuable precaution which yielded great benefits in further development.

2.3 Related Scientific Work

Unfortunately, as this is a pretty focused topic, no closely related and relevant scientific papers which could help this project as a whole were found. This was understandable due to the fact that probably not a lot of research on automated claim processing was done.

However, books and papers centred on individual aspects of this work have been used and referenced throughout the paper. They contributed with new ideas, adjusted approach, and often helpful advice.

3 The problem

The most noticeable issue with the current manual way of processing claims is that it is quite difficult for a person to process the amount of claims which are submitted. Iterating through all of them and taking into account all of the factors which make a valid or invalid claim is a task difficult enough even without having to maintain constant precision. Along with the process being tedious, it is also highly time consuming. Especially when taken into consideration that the amount of time needed to process a larger amount of claims scales at least linearly with the number of claims.

Having said all of that, the process can safely be characterised as a difficult to scale and constantly perform well. However, for an always developing and ever growing company such as Belsimpel, this is an enormous issue. The unpredictability and uncertainty behind the scaling of the important process inside the company can have devastating effects if not handled properly.
3.1 Project Goals

Because of the vital importance of claim processing and claim assessment, and the impact which they have on an insurance company, Belsimpel developed a plan to automatise the process of claim assessment as much as possible. Hopefully, this would in turn save valuable time and improve the situation regarding the scalability.

As a result of this necessity this project came to life. The goal set before it was to implement some type of automatised claim assessment solution. This tool should have integrated into the company’s existing web platform, handled all of the data collection in the claim forms, processed that data, and communicated the results to the claims team. In addition to that, the tool in question should have provided an overview of all of the data collected in a well organised and comprehensible manner.

This entailed implementing a data collection for when the user is filling in the form. Afterwards, using that data it could be stated with more confidence whether the user was telling the truth. Since this data was a lot larger in quantity than the previously recorded one, and for the sake of consistency, precision, and scalability, it needed to be processed in a safe and manageable way. For this reason, the idea of incorporating a machine learning algorithm first appeared. Finally, the results of the first two parts needed to be communicated in an easy and understandable way to the claim assessment team which is responsible for the final decision.

For the scope of this project, as it was the first step towards this transition, it was agreed that it was rational to set as a goal to present the claim processing team with the information whether the claim was suspicious or not. This was to be determined by checking how many times, and how often the user changed his mind and altered his statement. This information would serve as a flag conveying the meaning that the team should focus on this flagged claims, while spending less time on the other ones.

4 Existing Solutions

As the company’s platform at the time was web-based, the first step was to search for existing technologies which could possibly be integrated. After a research was conducted, two existing tools which were a possibility were found. Namely, Mouseflow [10] and Hotjar [7]. Both of these offered similar features and functionalities, and both were paid for. Although some quite big brands were using the services of these companies, they were not suitable
for the needs of this project.

4.1 Flaws in Existing Solutions

Firstly, both focused on recording the user’s behaviour instead of the form data. This meant that they follow the mouse, the position on the screen, clicks, record the screen, etc. Using them, the intermediate text in the form fields could not be collected, at least not in a processable text form. Hence, the data could not be processed properly later on.

Secondly, these software tools were limiting since the functionality provided could not be manipulated directly, and new features could not be added with ease. The main entity responsible for doing so were the companies which developed this software, and their developers. Limitations set by this would not be easy to overcome, and the complete set of capabilities which was planned would not be met.

Thirdly, the project was bound to be closely connected with the current functionality of the Tulip Assist, as well with the current database and the people using it. This implied that it would change frequently, and possibly immensely. Hence, the need arose for a flexible solution which could keep up with the needs of both underlying software and hardware, and the people who rely on it.

Fourthly, the data in question was private and highly sensitive. When taken into consideration that it contained names, phone numbers, addresses, bank accounts, and movements, among other things, it could have been safely concluded that it needed to be completely safe and handled directly. Any contact with a third party could have caused leaks and possibly violated the laws such as GDPR [13].

4.2 Conclusion on Existing Solutions

Finally, it was easy to see that the only way to satisfy all of the arguments was to orientate towards making the software in-house. This would have provided the highly needed flexibility, while the data would not reach anyone apart from the two parties involved in the relation. On top of this, the difficulty of communicating the new requirements, reporting issues, and providing feedback to the developers would be minimised. Moreover, the optimal efficiency would be achieved with respect to the time needed for the claim department to reach them.
5 Data Collection

The final goal of the project required extensive data to be collected from the user. Bearing this in mind, it was mandatory to define its source. The logical solution for this was the claim submission form as it was the main source of input from the user. Hence, it was decided to move in the general direction of collecting the data during the filling of the form.

It was presumed that observing the user while he entered the data was the most relevant part of the communication. This was because it is a natural point in the process where most, if not all of the inconsistencies could come to the surface and be recorded. Hopefully, this should have resulted in a more precise solution being built. This decision would later on prove quite valuable because of the abundance of additional, sometimes conflicting data.

The solution was shaped and influenced by taking into account the important factors such as the ease of incorporation into the current project, the ability for it to be used in the future, the efficiency at which the data is collected, and the extent of the said data. On top of that, since the development on a website with numerous users during the day was to be a long-lasting process, there should have been no issues in terms of performance or memory.

Together, all of the aspects were satisfied by examining in detail the submission form for a new claim. Parts which it consists of were a valuable asset, as they were already specialised in various aspects of the process. Nonetheless, the data collection process could be split in the topics described hereafter.

5.1 Relevant Data

Focusing on the form, there were two main groups of data which could be monitored. These mainly deal with:

- Visit Data
- Input Data

It was important to understand the difference between these. The Visit Data is the general information regarding a visit. It was intended to capture the facts which describe an access to the form, and would always be stored for all of the accesses. On the other hand, the Input Data deals with the information directly provided by the user. This was for example what the user types in a certain input field when in the process of submitting a claim.
Furthermore, Belsimpel has access to a considerable amount of storage and processing power in respect to the needs it has. Because of this, a design decision was made to collect as much of relevant data as possible. This data, even if not used, would quite possibly be beneficial in the future research and development.

The long list of the possible parameters to track contained a lot of data for which the implementation of tracking would simply not return the value of invested time. Also, in some cases it was simply the case of it being too difficult to bridge the inconsistency between device types. In the end, in order to make the system more maintainable and longer lasting, upon looking at the list of all of the possible parameters, some of them had to be removed. The final list of Visit and Input parameters which were decided to be collected in the project was given below.

### 5.1.1 Visit Data

In order to differentiate among different visits, and also in order to gain future insight, some general data was of use. This was collected to depict the user’s behaviour over time. It was for example to help the claim processing team see if a user maybe tested the system with a certain input before the officially submitted date of the accident. Also, it could have been that the user noted a theft of a device of a same type as of that from which the claim was made. Although this did not imply anything, in case of a suspicious claim, it could represent additional evidence.

In respect to the relevance and being useful, certain factors in the data regarding the visit were shortlisted for collection. These included:

- Final version of the submitted form
- Date and time of the visit start
- Date and time of the last visit update for the user
- Device useragent; a description of the operating system, device, and web browser
- Claim for which the visit was made

Although the initial list had entries like tracking the time spent on each input field, time with keyboard/mouse/input idle, time between filling everything in and the submission, distance which the mouse or swipes traversed, the number of clicks, average time between actions, or the amount
of times the user tried to submit the form containing an error, they needed to be removed. This was for the general and already described reasons of incompatibility and inconsistency.

5.1.2 Input Data

This represented the base of the entire tool. Quality of this step and the collected data could yield improved precision in the later stages of the project when data is processed. The core of collected entries consisted of the information coming from the input fields in the form. These input fields had different properties, and as such could be observed as different classes. These might have some overlap in certain cases due to different possibilities in their representation, but the decided division of the data input types was as follows:

- Text Input
- Number Input
- Date Input
- Checkbox Input
- Radio Button Group Input
- Dropdown Menu Input

Although what the user typed in was the most important, Input Data also captures additional knowledge about how or when these changes occurred. This was in order to better portray the gathered data. Altogether, parameters tracked here were:

- Intermediate changes in text input
- Intermediate changes in number input
- Intermediate changes in date input
- Intermediate radio buttons selection
- Intermediate checkboxes checked
- Intermediate dropdown menu selection
- Type of change associated to the parameter
- Date and time of the change
- Claim visit to which the change is associated.

It is worth noting that the first six would not occur at the same time. They were relevant for different, previously defined input types, and together made a set of possible data changes for all possible form input types. This property was used in the later implementation, and it made the entire process easier. Furthermore, the conveyed information was complemented by the remaining points. These were some further descriptions of the data which could be useful in certain use cases.

5.2 Collection Process

In order to build a solution, a well defined cycle needed to be respected. This would in turn result in a well defined, well structured, loosely coupled, and highly coherent code. Here, the general structure would be outlined and represented part by part. This architecture an implementation would serve as a base for incorporating further expansions.

5.2.1 HTML Changes

The initial step was to select the HTML elements (i.e. the input fields) in the DOM tree of the claim form which need to be marked, so that they could and would collect data. This DOM tree would have most, and presumably all of the elements of the given types. However, if they were to be approached individually and without certain logic which reduces redundancy and excessive work, it would be difficult to keep track of all of the elements and possibilities. Also this would make it a lot more difficult to freely update the claim form with new questions and fields.

In order to group the mentioned elements with the same input type, a class name corresponding to the input type was attached to the element. These names were: 'listenable-text', 'listenable-number', 'listenable-radiobox', 'listenable-dropdown', 'listenable-date', and 'listenable-checkbox'. Along with this, it was important to assign to these elements an id which will be used to represent it visually. For the sake of readability, these ids were stored in a database, along with their human readable form.
5.2.2 JavaScript Listeners

The second step was to provide functionality to the elements with the relevant class attached, as well as to add the functionality to the page itself. This functionality was the possibility to capture data, or a change in the data. As this was going to happen frequently, and on certain events, we needed to utilise the Javascripts event listeners. For this, separate methods were made so that on event the change is processed and ready to be sent back to the server, with all of the corresponding data attached to it. Each of these methods corresponded to a single input field type. This was so that they can be attached to the elements and produce minimal overhead, as well as in order to avoid unorganised and cluttered code. Also, most of the listeners relevant to the visit data could be attached to the page directly.

Down the chain of events, in a JavaScript file, the HTML elements with an adequate class name attached needed to be queried by class. The results of these queries were saved in six separate arrays. This was an uncoupled solution which provides easy access to all the HTML elements. The arrays could simply be iterated through in order to attach the adequate implemented listener method to them. Finally, In these listeners, the AJAX call would make an API request to the server, thereby transferring the data about a visit or a change. Please note that appropriate listeners should also be attached to the page itself in order to obtain Visit Data.

5.2.3 API Endpoints

In the third step, upon collecting the data directly, it still needed to be sent to the server. It was certain that this was to be done by creating an API endpoint to which POST requests were to be sent. One endpoint was created to facilitate the handling of the claim form visit data, while the other was created to facilitate the claim form data changes.

As the former request was done on page load, its response could be utilised to convey the visit data. Namely, the response payload was set to contain the claim form visit id. This was then stored inside the script and used for further communication. It would be added to the payload of the later latter request to depict to which claim form visit the data change belongs to.

For both of these endpoints, upon receiving a request, the endpoint should passed the data onward to the database. There it should have been stored and saved for the future parts of the tool pipeline.
5.2.4 Communication Frequency

However, there existed two possibilities on when to send the data using the mentioned POST request. It could have been sent either on every event which was listened for, or the data could have been stored until either the claim was submitted or the tab was closed.

Former option provided the ability to make sure that at least some data was preserved upon the users encountering some unforeseen circumstances, such as battery becoming empty, loss of electricity, loss of internet, etc. Also, if the user intentionally killed the browser process this would have prevented him from being able to stop the overall transmission of data. In addition, data sent using this approach was more verbose and easily managed. Since small bits of the entire session were sent, it was affordable to capture more information. Although negligible, this path minimised the bottlenecks which might have arose when writing to the database or performing computationally intensive processing before storing the results.

If the data was sent in one batch, the scaling of the data was unpredictable. Different users would fill in different amount of details, which could end up in unbalanced decisions and inconsistent data. If small amount of data was recorded for each event, this would mean not utilising all the space in small messages. Similarly, if a lot of information was captured per event, and the message grew in size, there would be a lot of data which needed to be sent in the same batch. Another, and one of the most important issues with this approach of sending the data in groups was that it needed to be stored locally. Input stored this way was easily reachable, and more importantly easily altered. On top of that, local storage limits differed and represent and issue for a solution which aims at satisfying as much platforms and browsers as possible.

As the benefits of sending the data frequently, on each event, were noticeably bigger than those of batch transmissions, with disadvantages of the latter being quite relevant, the decision was made to implement the first option in the project. This was a crucial decision, which adhered to the security principles dictated by the company policy. Furthermore, it also benefited both the user and the server resources, and maximised the possibilities of capturing the most data.

5.3 Storing the Data

Since the Tulip Assist has existed for a longer period of time, it already had an existing, and complex database. As this was a Relational Database writ-
ten in SQL, obtained data needed to be stored inside it with that in mind. Thankfully, all of the operations were made easier by the PHPActiveRecord.

Upon examination of the database structure and the existing tables, a logical connection point arose - the claims table. However as users do not know the id of the claim which they are filling in until after they submit it, the new data will not reference anything before submitting the claim. For this reason, the two new tables were made - claimformvisits and claimformdatachanges.

The reason for splitting the data was because of the cardinality of the relation. In the base, since a user could visit the claim form page multiple times before actually submitting the form, the relation between the claims and the claimformvisits tables was a 1:N relation. Furthermore, the relation between the claimformvisits and the claimformdatachanges tables was also a 1:N relation since a claim form visit was probably going to record multiple data changes. Also, due to the JavaScript asynchronicity, the nonce column was added to the last table in order to preserve the chronological ordering of the changes. All of the relevant tables and their columns can be seen in the figure 1.

It is worth noting that all of the previous visits along with all of the corresponding changes since the last claim, or the beginning if the user has no claims, were taken as a part of the new claim. This was the case due to the difficulty of the task to draw a line between different claim form visits in order to say what belongs to a claim and what does not, and more importantly due the fact that any of the visits could have been relevant for the claim assessment.

5.4 Data Quality and Quantity
This part of the project was done early on as it was the most time-consuming. This was because the obtaining of the real data was independent from any action which could have been taken. The users made claims at their own pace, and by looking into previous data, there were on average between 15 and 20 claims per day. For the sake of consistency, it was decided that once the collection process started, there should be no changes on the data collection implementation. By implementing the data collection early, it was improved and prepared for the workload it was put through.

As it seemed that the implementation was robust and prepared for work without assistance, it was released on 19th of April. Due to overlap in dates,
Figure 1: The database model of relevant classes and fields
the official time of the start of the data collection process was set as the 20th of April at 00:00:00. After this, the tool continued to collect data all the time until the end of the project. Last testing of the data quality was done on the 21st of June of the same year. This was a total roughly two months, or sixty-two days. At that moment, there were 1113 claims in total collected from when the data collection was released.

Out of this number of claims, unfortunately, there was a small number of claims which for some undetermined reason had claim visits recorded, but no data changes. This was unusual, because of the fact that in order to submit a claim, at least one data change would be recorded. There was a 100 of these claims, and it consisted for about 8.9 percent of the total amount of claims.

Furthermore, out of these faulty claims, 14 of them, or about 1.2 percent of the total amount of claims, was registered but not accessible for some reason. It seemed as if the claim visits and claim data changes were deleted. Again, after extensive debugging and research no reason for this was found. The only observed pattern was that most of these have occurred in the beginning two weeks of the data collection.

However, besides these two issues, the data was consistent, well formed, and of good quality. Moreover, the database was well populated, and the relations between adequate claim forms and claim form data changes were made. This was determined by a manual check of a random sample, as well as by automatising the procedure and iterating through all of the entries between the two dates.

5.5 Data Usability

Even though some of the claims had issues, this was an acceptable amount. Because the data was initialised at a value which can be differentiated from the processed data. This way all of the bad claims would be marked as suspicious by default, and that the claim assessment team could pay more attention to them.

As for the good data, it was collected perfectly. It could be used in all of the algorithms in the following steps of the pipeline. This was especially true since the objects retrieved from the database could easily be translated into any other form, which was on most occasions JSON.

In order to conclude, the data collection was deemed a success. All of the technologies which were intended to consume this resource have done so without encountering issues, hence making it possible to reach the desired outcome of classifying the claims as regular or irregular.
6 Data Processing - Manual Processing

In order for the claims to be processed, the data needed to be transformed from the raw into a usable style. For this, it was necessary to decide on a proper form, and what is actually significant. This was to be decide based on the a metric which captures the specifics relevant for the use case, which led to that in some cases, certain parts of the data were omitted due to the irrelevance. Finally, upon that, this information needed to be transformed to a number representation. This represented the metric which summarised the entire collection of the data of the same type.

Due to the quite difficult task of outlining the important aspects of the input in the raw format, it was decided to first extract the metrics. This was to provide a solid foundation for the subsequent research. This task proved difficult as finding the right metric which could convey the proper and usable meaning was far from easy. Moreover, doing this before familiarising with the data key features, although the only way, made it increasingly difficult.

6.1 Collected Data

To summarise the collected data briefly, the entries in the database were observed. Unfortunately, although the form had certain checks implemented, only certain input fields were mandatory. To make situation worse, this was not observed at the time of the implementation of data collection. This resulted with some fluctuation in the data. Thankfully, there were some constants and trends which made the collected data usable.

The most interesting aspect which could be used was the necessity of the text to appear in the processing. Also, this text is one of the key places where the change in ones mind can be easily observed. Moreover, it went along especially well with the multiple choice radio buttons which hide or show certain questions depending on the choice. This was another key aspect of the data which could be observed and which could be used to the implementation’s advantage.

One of the ideas behind the data collection was to capture the date changes. These were to help when a person shifted the day of the accident. This change would indicate the issue with the claim, and would be a certain red flag. However, it was determined that even with the date picker available and the date format noted on the input, users used the keyboard and wrong formats. Without a mask which was to format the input in a precise way, there were endless possibilities for the date format. The outcome of this was that a part of collected input was not usable and had to be discarded.
One step further, a quite complex custom parsing method needed to be implemented.

Contrary to the previously discussed data which was at least to a certain extent helpful in behaviour depiction, some input types were not useful at all. These were the checkboxes, numbers, and dropdowns. Although these could have been helpful in the latter versions of the form, they had to be omitted for the scope of this research. These variables were simply not profitable in the sense of contribution to the final outcome.

However, all of the said input types were omitted for different reasons. The checkboxes in the claim form were used as a mean of a confirmation. As such they were mandatory, and the claim could not be submitted without them. However their collection was not useful. However, the delay between filling in of the form and actual confirmation might be useful in some future iteration of the same project. Also, When examining the data, and starting the work with it, it was proven that the numbers were a lot more easily processed as string. The operation needed to compare two numbers are more complex if they are portrayed as numbers, and not as string. Again, this might change in the future with the incorporation of a new claim form. Finally, the dropdown menu listeners were only used for hour and minute collection, and were omitted by the end of the project. Because of the inconsistency of their usage, as well as because of the time needed to process the data from these, for small benefits, they were omitted from the process of metric extraction.

6.2 Metric Extraction

Upon looking at all of the data individually, the next step was a bit more clear. However, after beginning the process, all of the hidden difficulties surfaced. These were mainly due to the need to find a way to properly convey the needed aspect of the data, or because of the lack of formality in syntax.

However, to overcome this, certain approaches were developed. These were specific to the different input types which were decided on pursuing, namely, text item, date item, and the radio group. Also, for some of them different paths were described. For all of these, advantages and disadvantages needed to be discussed, and the best option needed to be chosen.
6.2.1 Textual Data

In order to extract the data from the text, we needed to choose what was the focus. It was decided to emphasise the development of the input over time. Hence it was logical that the intermediate changes needed to be compared to each other. Furthermore, each two consecutive entries were to be compared to each other, all while respecting the chronological order. The result of this was to be some sort of a metric which would be consistent over all of the claims.

The solution was to find a distance from one sentence to the other one. This distance depicted the Levenshtein distance [8]. It represents the number of changes needed to transform one string into another one. The set of possible changes consists of insertions, replacements, and deletions. This was suitable as it respects the chronological order, and it could also assign different weights to different operations.

It was our intention to encourage insertions, and punish replacements as well as deletions. This was due to the fact that filling in of the form naturally requires insertions, but changing one’s statement requires the latter two. Hence, the weights assigned were zero, one, and one, respectively. Although it could be discussed whether this was a good weighing of the scores in general, it is completely reasonable for the scope of this project in order to find the change of opinion which could be normalised.

The first step was to remove unwanted noise. This noise was in the form of typos. It was burdensome for any of the algorithms to include these short changes. It resulted in skewed, unreliable, and incorrect metrics. To battle this problem a threshold was implemented where any distance bellow five would be disregarded. Five was chosen as a distance which is the upper bound for an edit which could be considered a typo, while not omitting to portray the changes.

Onward, there were several possible ways to describe the entire data set using this distance. In order to explain the behaviour of each of the ways, they were tested, and it resulted in valuable insight. As this was the process in the finding the proper metric, all of stated metrics were to be discussed individually.

- cumulative consecutive distance
- maximal consecutive distance
- maximal normalised consecutive distance
- normalised normalised consecutive distance
• adjusted normalised consecutive distance

On the first try, under the assumption that all of the data needs to be contained in the metric, as well as because it was the logical first step, the cumulative consecutive distance was calculated. As this prioritised that data which contained long texts and a lot of intermediate changes, another method was needed. One which could compensate for this.

In order to compensate, a way to remove the significance of the amount of intermediate changes was sought after. Although not a big improvement, by assuming that the biggest distance describes the entire data, maximal consecutive distance was tried. This was good progress in the sense that a lot of intermediate changes were not affecting the data. However this failed to show more than one change of mind in the data. On top of that, longer intermediate changes if deleted would be deemed more important than if the shorter ones were deleted.

Hence, the need for normalisation was brought up. This approach was to minimise the difference in changes over long and short intermediate changes. How it was done was to make the distance between two consecutive sentences proportional to the length of the longer one, or in other words, by dividing the distance by the greater length. In case of the insertion, as the quotient is 0, it was not important what was chosen. However, in the case of the deletion, the longer string was bound to be the first one, so in all of the cases the metric holds. Unfortunately, this solved only half of the problem. With the normalised distances between consecutive changes, the overall metric for the dataset still accumulated all of the normalised distances for the intermediate changes.

However, this issue had a really easy solution. It was to normalise the entire dataset, and to have a value which is comparable between different claims, regardless of the intermediate change size and the number of data changes - normalised normalised consecutive distance. As all of the changes are taken into account, it was only needed to divide the obtained metric with the number of changes or 1, whichever was higher. This was a nearly complete solution, but with a big issue. That was that in the case of a lot of intermediate changes, and a complete change of mind (i.e. completely deleted input) the metric would not represent that change of mind significantly enough. Issue was that due to the number of said changes, the metric would not convey the value of the change of mind after normalisation.

Finally, the most difficult task was to define how to adjust the data with the change of mind in it. It was obvious that the normalisation would be lost. However, this was acceptable due to the fact that the field type metrics,
although represented together, were not normalised because of the highly inconsistent data and low importance of certain fields. In the end, the data was adjusted in such a way that in the case of a deletion or replacement of more than 70 percent of a string, it would multiply the normalised distance by the number of intermediate data changes. In return this produces a higher metric, which is what is needed. Although sub-optimal, this was an acceptable solution.

6.2.2 Date Data

Date data had a significant flaw in it, which shaped the entire process. On the claim form, when asking for input, there was no restriction on the format. Although provided with a format and a date picker, the users could write using their keyboard whichever date they wanted. This was apparent on the inspection of the data and led to a complicated parsing method and metric system which would need to be replaced in the imminent future, as soon as the changes were made on the claim form page. Furthermore, the parser differentiated between certain and uncertain cases. Former case had no ambiguous or discarded data, while the latter did.

The date evaluation relied on the said date parser and the distance calculator. The first first replaced the separators with spaces, split the date into words, and finally begun the process. If it was still a one word it tried to split it based on the usual date format dd-mm-yyyy, and sets the uncertainty flag. On the other hand, if it was exactly three words, first it would check for the letters which would be months, then it would check if there are 4 numbers somewhere in those word, and finally if nothing works, it would assign the date in a format as close to the mentioned one. The only case here where it would say with certainty was if one of the three words was a number of length four, and one of the other two numbers was greater than 12. Otherwise, in the case of inability to parse a sting as a date, it was discarded.

There existed a built in parser function, however, the implemented algorithm adhered to the business logic and the area of The Netherlands better than it. The difference was in the readiness to handle unpredictable data. Where the PHP provided \texttt{DateTime} library would throw an error, the custom implementation would either parse or acceptably discard the input.

The return value was given as an array containing the day, month, year distances, and a certainty flag. This is afterwards forwarded to the distance calculator which utilised the \texttt{DateTime} distance method, but wrapped it with edge case checks. Here the metric was the simple weighted sum of the
distances. The implementation relied on a proportion in which the years were neglected as people have often had a typo which would skew the data. Also, the claim had to be reported in the first 14 days, making the year difference useless, apart from creating noise. Finally, this sum had to be normalised by dividing by the amount of successfully parsed dates.

6.2.3 Radio Group Data
Radio group input type represents the control of the workflow, making it perfectly suitable for further description of the text metric. As precisely separated statements were depicted by the radio buttons, it was easy to find a suitable metric. Taking into consideration that the data was in chronological order, the metric was obtained by comparing consecutive decisions, which depicted the change of mind of the user. Of course, it had to be normalised as the previous two metrics, by dividing overt the total number of changes.

6.2.4 Normalisation of Input Field Types values
Because there are several mandatory fields of a same type, a way needed to be found how to combine them in order to lower the amount of variables. Similar to the text metric, there was an issue when a large number of fields would have a metric of zero. If the combined value of fields was to be normalised, these low value fields, although to some extent insignificant, would lower the overall metric. The solution was to avoid normalising over the fields of the same type. In the end, this was perfectly acceptable because it preserved the underlying data while conveying the proper picture about the dataset.

6.3 Data Readability
Upon extracting the metrics from the data, as well as normalising it and making the necessary decisions, further steps were needed. This implied making a decision to use Machine learning or not. Nonetheless, to make this choice an overview of the current metrics was needed.

At the first look, since the decision was made to not normalise the data over numerous types, it was a bit difficult to say what was conveyed. Moreover, the date and the radioboxes were a bit unreliable since the corresponding metrics were not reporting any changes in most of the cases.

What one could have deduced from the calculated and obtained metrics represented in a table was that the date was not the most relevant feature
due to a lot of noise, and due to the disregarded or misinterpreted input. It was fluctuating too much. On the other hand, the radioboxes provided more insight by showing the complete change in opinion. Unfortunately, based on the table, what was generally the case was that the users who changed their story may have or may have not stayed inside the boundaries of one flow. So for example, the radio button stated that it is either a theft, or a damage. On some occasions, the users would not switch between these two, while on some they would, thereby complementing and describing their story further.

However, based on the text metric, one could have assessed the state of the claim. Although it was not normalised and perfectly visible, when taken into consideration that generally two to three inputs contributed the most to the metric, one could have gotten the picture. Also, a rewrite of 70% was multiplied by the amount of intermediate data changes, this added 0.7-1 to the overall metric in response to that amount of change. Hence, anything above 1 could definitely be labelled as suspicious. This was based on an assumption that a small number of inputs contributes the most to the metric. Also, even if this was false, false positives were not the issue. Unfortunately, the problem for this approach was determining the lower bound of this. At which value did the acceptable claims stop, and the suspicious one start?

6.4 Possibility for Machine Learning

Said problems made it quite difficult to process the data visually or manually, without some sort of an algorithm aid. Here, a machine learning algorithm could have bridged the gap and helped find a consistent solution [6]. It should have also even improved on the precision over time, and the metrics would still be suitable in the current form. Although this has seemed tempting, there were some shortcomings which needed to be overcome.

The data in question was not efficiently describing all of the data and could have skewed the results of the machine learning. Moreover, the lack of information from the claims team, and time needed for the labelling of data for a precise supervised learning algorithm entailed the implementation of unsupervised learning algorithm, namely clustering.

Since both options had advantages and disadvantages, as well as possible issues in the future, it was decided to combine these two approaches in the first steps of the project. This would have gave the benefits of clustering as an advice for the validity of the claim, while providing a thorough overview of all of the metrics. Also it would give raw data and how it changed over
time. All of this was needed to improve manual assessment. Hopefully, new feedback and individual changes on the smaller parts of the framework for the project would then mean a lot more relevant advances in a smaller time frame.

### 6.5 Preparing the Data For Machine Learning

The process up to this point processed the data in good, thorough, and organised manner. This reduced the workload before the implementation of an algorithm for the unsupervised learning. The only needed thing was to adjust the format of the data so that it adhered to the input required by the algorithm.

Since the data was already stored in a database, it could be used for the testing purposes. However, before using it directly, the initial test used a simple cron job which ran PHP as a script and logged the output in a separate file or the terminal. This was a simple depiction of the clusters and the ids of the claims which belonged to them, and it helped to see that the data actually met the requirements for the algorithm input.

### 7 Data Processing - Machine Learning

Machine learning could not use the raw data for its algorithm. Luckily, the data was already processed and ready for systematic use. The only thing needed to be done was to adjust the format of the said data, which required a small amount of work.

What needed to be decided on next was how to implement the Machine Learning, which type of it, and which algorithm to use, which data to process, and to observe the results in order to see how adequate the solution was. This path proved to have a steep learning curve with surprising outcomes along the way. Moreover, even with the implemented solution, there were numerous options in order to adjust the algorithm to the use case.

#### 7.1 Why Machine Learning

The project had at its disposal a vast amount of data. It was difficult to keep track of the trends and see the correlations between the multiple metrics. It would be nearly impossible for the claim assessment team to adapt quickly and keep track of the correlations if new parameters were to be added. This would defy the purpose of the tool, and could be helpful to the efficiency only after a lot of time and additional training for the staff.
Even more importantly, data was bound to increase in size as the time progressed. This is why it was highly needed to adapt how the correlations were determined. As the assessment process was conducted at the time, it would imply to re-asses the entire dataset for all claims by hand on set interval, preferably at least every week, since there were around 100 claims submitted per week, and increasing. Considering how time-consuming and inefficient this approach was, it was not acceptable.

Finally, at this initial stage of automatising the claim assessment, the semantic meaning of the claims was not determined. The entire focus was on the syntactic differences. What this meant was that decision making was made solely on how the data changed over time, and not if the meaning it was trying to convey changed. As a possibility to make a step towards the meaning, it was possible that machine learning could find a correlation between the text input and the radio buttons which might be based on some meaning. This assumption was based on the fact that radio buttons change the flow of the claim form, i.e. different questions are asked, which should by definition have a different meaning.

Again, looking at the arguments stated, only a reasonably fast solution which increases the precision and maintains the time needed with the increase of data was required. Also it needed to be highly adaptable to a change in input parameters and their amount. For this reason, machine learning was a perfect fit as it satisfied all of the needs. It was especially a good solution since it could be trained over time to satisfy the needs even better.

7.2 PHP and Machine Learning

PHP is not a language which is widely regarded as a language of choice for data processing. This is as it was intended to serve the function of a hypertext preprocessor on the server side [2]. This is far from data science. However, it can be adapted to handle some data processing.

An important fact was that there existed an open-source library for combining machine learning and PHP [1]. It was called PHP-ML and it supported most of the machine learning algorithms, including all of the ones considered for the needs of the tool being implemented. This contributed to keeping this part of the project inside the existing technology stack, and not incorporating a new technology.
7.3 Supervised or Unsupervised Learning

In order to continue with the implementation, a decision needed to be made regarding the direction in this field. There were two possibilities to chose from, namely, supervised and unsupervised learning [6]. Moreover, the choice had to be a result of assessing the compatibility of the data and the two, as well as what was trying to be achieved.

7.3.1 Supervised Learning

The main advantages of supervised learning was that it could be very precise in assessing the outcome. [6] Possibility to train the solution based on the reviewed data gave flexibility, as well as precision for this specific use case. In the case of this solution, the claims could have been labelled as risky or not risky upon revision. This could have in turn outputted relevant data which was less likely to be wrongly classified. Moreover, this allowed for future data to affect the solution a lot more than in other approaches.

Although the process seemed like a perfect fit, it had many issues which prevented easy implementation. Firstly, the data collection took the most of the time assigned to data processing, so it left a small portion of it for labelling. Going through more than 1100 claims, all of which contain more than 15 fields, would require many work hours, which unfortunately were not available. And, without the labels, the unsupervised learning was not possible [9]. Moreover, these regression algorithms were not so suitable for discrete data [9]. And secondly, although a smaller issue since semantics were not covered, the language was an unpleasant obstacle. Majority of the claims were in Dutch, which would require someone in the team who speaks fluent Dutch, in addition to the English speaker.

7.3.2 Unsupervised Learning

Unsupervised learning was the solution in case of the unlabelled data. The biggest group of algorithms - clustering algorithms, have as the main goal to find separate groups of data points. In the case of the tool being developed, this was the answer to the absence of the possibility to manually label the data, and unlike the regression algorithms, was a perfect match [6]. Once the data was fed to an algorithm from the aforementioned group, the output would be a certain amount of groups of claims which would then be processed further. In addition, upon obtaining new data, this process would become more precise, and the border would become more fitting. Proving that this
solution, although not completely precise, is future-proof and easily scalable without the need for additional working hours put into it.

The main issues with this algorithm arose because of the precision and the differentiating of the output groups. The process was not adjustable in sense to favorise one group over the other. Hence, it could not be trained on what was an acceptable input. However, this could be adjusted to some extend by adjusting the number of clusters. On top of that, the output of the algorithm yielded unlabelled groups, which made it more difficult to define to which risk group a claim belonged. Fortunately, there was a way to overcome this. It was by keeping track of the average metric values for the separate clusters, based on which the risk values of clusters, and claims in them could be determined.

7.3.3 Decision

Having looked at both methods, the decision about the better one for the scope of the project needed to be made. Since the data obtained in the project was not labelled, and hence not compatible with the supervised learning, as well as because of the advantages of unsupervised learning and the way it addresses scalability, the latter one was decided on. This would have provided the necessary functionality while minimising the work necessary to receive the output.

7.4 Algorithm Used

Although the group was chosen, the actual algorithm was not. As two predominant, and most adequate algorithms, k-means clustering [6], and Density-based spatial clustering of applications with noise - DBSCAN [5]. As each of them were important and could have been applied, the ease of incorporating into the project and the compatibility with it, as well as the suitability for the desired way of processing the data were the deciding factors.

The first had the advantage to the latter one as it required the number of clusters to be set beforehand. This was important in order to obtain the wanted results, in this case the regular and the irregular claims. With the second one, the algorithm tend to find the optimal amount of clusters. Which, in turn, would provide an unknown number of them. This would made it difficult to asses the regularity of the claims inside the cluster. In other words, the obtained clusters would need to be further clustered to regular and irregular ones.
Furthermore, another aspect which was different between the two was the way outliers were handled. Whereas k-means incorporated them into the clusters, the DBSCAN did not. This might have provided a bit more precise split, but the outliers needed to be classified as well. So in order to do this, additional processing might have been needed. However, this kind of processing might not be easily feasible when more variables are added, making k-means a better option.

Finally, although the latter algorithm was a adjustable by modifying the epsilon distance for classifying neighbours and the minimum neighbours for the outlier classification, the weighted advantages of k-means made it more suitable. This decision also included the fact that the basic implementations of the DBSCAN had the time complexity of $O(n^2)$ [12], which was appealing.

It was implemented having in mind that the data would grow over time, and that it needed to be recalculated every time. Also, a compromise was made that the imprecision could be compensated by adjusting the number of clusters.

### 7.5 Machine Learning Outcome and Results

The most important questions which arose were whether there was enough data to do run the algorithm properly, and if it was going to be precise enough. These questions were to mark the utilisation of machine learning in the project as a success or a failure. Moreover, in order to obtain optimal results, the k-means algorithm settings needed to be set properly. These include the number of clusters which were looked for, and the parameters which represented each claim.

As we need to split the claims into regular and irregular ones, we could have used two cluster. However, in order to minimise the Type II error and the false negatives (claims which are assessed as regular instead of being irregular), we could have looked for three clusters. These would then be labelled as regular, irregular and highly irregular. This could be acceptable if the total amount of irregular and highly irregular claims was significantly smaller than the entire amount of claims.

Also, from the three available metrics, namely text, radiobox, and the date metric, only two are adequate for machine learning. The date metric was fluctuating quite a lot and unnecessarily skewed the data. If used in the algorithm, it would have produce unwanted results and wrong output. On the other hand, text metric was present at the majority of claims, and was a good indicator of the behaviour. However, it was a question whether the radiobox would contribute to the solution. Hence, it was needed to be
determined which of them metrics are to be used.

Upon testing the given possibilities, the obtained results were summarised in the following tables. First the algorithm was run on two clusters, and afterwards on three. Also, for both of them, it was tested how the algorithm behaves when each claim was labelled with the text metric, and how when labelled with both the text and the radiobox metric. The meaning behind checking these was to keep track of the average values of the clusters.

### Table 1: Algorithm behaviour with 2 clusters and the text metric only

<table>
<thead>
<tr>
<th>Claims</th>
<th>Avg. Text Metric</th>
<th>Avg. Radiobox Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Cluster</td>
<td>860</td>
<td>0.1693</td>
</tr>
<tr>
<td>Irregular Cluster</td>
<td>153</td>
<td>1.7784</td>
</tr>
</tbody>
</table>

### Table 2: Algorithm behaviour with 2 clusters and both metrics

<table>
<thead>
<tr>
<th>Claims</th>
<th>Avg. Text Metric</th>
<th>Avg. Radiobox Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Cluster</td>
<td>875</td>
<td>0.1843</td>
</tr>
<tr>
<td>Irregular Cluster</td>
<td>138</td>
<td>1.8583</td>
</tr>
</tbody>
</table>

If we observed tables 1 and 2, we could see similar results. A lot did not change. The number of claims, and the text metric value stayed very similar. This meant that there is not a lot of difference between the two. If we observed the boxplots in the figures 2 and 3, we would the expected. On the one which contains only one metric, there was no overlap, while on the one with two metrics, there existed some overlap. However, if we examined the one of the regular claims in the overlap, we observed the issue that the claim although clearly contains a lot of irregularities which we wanted to remove. Hence, we decided on clustering with three prototypes.

K-means with three prototypes was expected to address the claims which were said to be regular, but were nonetheless irregular. This also provided the benefit of having two levels of irregular claims. They could then be handled in a different manner. The irregular cluster could contain some of
Figure 2: Boxplot depicting two clusters using 1 metric

Figure 3: Boxplot depicting two clusters using 2 metrics
the values from the regular claims, but would decrease the Type II error significantly. In the tables 3 and 4, unlike previously, we could observe the huge differences between running the algorithm with one and two metrics. Having only the text metric placed the prototypes for the irregular claims at a significantly higher values, while the one for the regular claims became a bit lower when compared to when both metrics were used. However, the clustering of the radiobox metric was interesting since the highest value has been placed in the irregular cluster, while the middle value was in the highly irregular cluster. Finally, we could also observe the increase in irregular claims when both metrics are used.

<table>
<thead>
<tr>
<th>3 Clusters 1 Metric</th>
<th>Claims</th>
<th>Avg. Text Metric</th>
<th>Avg. Radiobox Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Cluster</td>
<td>805</td>
<td>0.1278</td>
<td>0.3178</td>
</tr>
<tr>
<td>Irregular Cluster</td>
<td>159</td>
<td>1.1137</td>
<td>0.5001</td>
</tr>
<tr>
<td>Highly Irregular Cluster</td>
<td>49</td>
<td>2.8119</td>
<td>0.6898</td>
</tr>
</tbody>
</table>

Table 3: Algorithm behaviour with 3 clusters and the text metric

<table>
<thead>
<tr>
<th>3 Clusters 2 Metrics</th>
<th>Claims</th>
<th>Avg. Text Metric</th>
<th>Avg. Radiobox Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Cluster</td>
<td>641</td>
<td>0.1537</td>
<td>0.0961</td>
</tr>
<tr>
<td>Irregular Cluster</td>
<td>259</td>
<td>0.3493</td>
<td>0.9453</td>
</tr>
<tr>
<td>Highly Irregular Cluster</td>
<td>113</td>
<td>2.0242</td>
<td>0.5548</td>
</tr>
</tbody>
</table>

Table 4: Algorithm behaviour with 3 clusters and the both metrics

In order to determined how big are the overlaps in data, and to decide which one to choose for the project, we have again, resorted to examining the boxplots. Here, in figures 4 and 5 we could see that, again, when one metric was used, there was no overlap. But the interesting behaviour was observed when a second metric was introduced. The outliers for the regular claims, when two claims were used were covering the same part of the text metric values as the values for the irregular claims.
Figure 4: Boxplot depicting three clusters using 1 metric

Figure 5: Boxplot depicting three clusters using 2 metrics

It was certain that the effect of second metric per number of clusters needed to be examined before making the final decision. This was so that the overlap in the regular and irregular claims can be explained. This could
be seen in the scatter plots in figures 6 and 7.

Figure 6: 2 clusters obtained using both metrics; green - regular claims, yellow - irregular claims

Figure 7: 3 clusters obtained using both metrics; blue - regular claims, green - irregular claims, yellow - highly irregular claims
Following the examination of these figures, it was finally possible to completely understand the behaviour of the algorithm with the provided dataset. Although the outcome when using only one variable seemed to be more distinctive, the second metric provided overall precision. Furthermore, splitting the data into three clusters was immensely helpful for obtaining the more adequate result while minimising the classification type II error. Hence, it was decided that the best option for the future of the project was to use both the text and the radiogroup metric in order to obtain three clusters.

8 Data Representation

In order for the tool which was being developed to add value to the company and the project, it needed to be understandable and intuitive. Moreover, it needed to provide all the relevant information in a concise and organised manner. This implied that there should not be a lot of training needed for the claims department team before they started using it.

Also, the user interface needed to be set in such a way that it did not need to change in the future. This also took future development into consideration. This way, the time is saved, and there was no need for the claims team to adapt again, from the beginning.

8.1 Communicating the Data to the User

The data is communicated to the user through directly accessing the menu from the sidebar. This was the most relevant place to put it. It was labelled Background Data, and can be observed in figure 8.

![Figure 8: Background data side menu entry](image)

Upon accessing it, the users are presented with the list of all of the claims which have background data collected from the filling in stage (Figure 9). It
also stated the regularity assessment. However, if one of them was pressed, another view specific to the claim in question would appear. The idea was that it contains all of the collected data in various forms. Each of these was useful in different use cases, and intention was for them to work perfectly together, and make it feasible to confirm or disapprove the regularity of a claim. This menu consisted of the Data Changes, Data Overview, and the Suspicious Data Tab.

Figure 9: Claims overview

### 8.1.1 Data Changes

Data timeline represents a chronological ordering of the visits to the form (Figure 10). It was implemented with the intention to track if the user has accessed the form on days prior to the claim submission, and has left similar data. For example this could show that a user planned to report the claim even before the mentioned accident date.

The stem part corresponding to the leaf expanded after a click on a leaf and presented the data changes (Figure 10). These were provided as a chronological ordered list sorted by the field identifier. In addition, the flower lead to the corresponding claim. This was done since the project bares the name of Tulip Assist. Hence it was appropriate that the timeline was a tulip. Furthermore, the leaves appeared as the page was scrolled. The
user interface was as intuitive and reactive as possible. To support this, the leaves and the flower changed color to depict that an action was possible.

8.1.2 Data Overview

The Data Overview served a purpose of showing all of the changes made per field type. It differed from the timeline by the fact that all of the visits were combined into one entry and the data was sorted by input field identifiers. It can be easier to see here how the data has changes chronologically, and if applicable, how the user has changed his mind and stated a different story. This functionality can be seen in figure 11.

Furthermore, the expandable list contained a score per each field. These represented the regularity values of the corresponding field. Unlike the clusters, these were either regular or irregular. This was to prevent confusion, and to promptly communicate to the user which fields need looking in to. It was worth noting, that for the sake of thoroughness, all of the fields which
contributed to the final metric of the corresponding field type were marked as irregular.

### 8.1.3 Suspicious Data

What the previous two components could not communicate was the decision specifics. That was the reason behind this component. It contains a card for every filled in field. However, instead of representing the data, here the users could see the metrics and in which extent they have contributed to the given score. Moreover, every card contained a risk assessment, metric value for the corresponding field, precision flag, and a description. All of these can be seen in figure 12.

Using this, the users could gain deeper insight in the claim. They could also keep track of the tool and whether the clustering was functioning properly. In the case of the irregularities, they would be able to address the

![Figure 11: Data Overview](image-url)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Risk Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORM_TEXT,ITEM,DATE</td>
<td>Safe</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,DESCRIPTION</td>
<td>Increased</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,DISCOVERED</td>
<td>Increased</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,LIABILITY_NAMES</td>
<td>Safe</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,LIABILITY_REASON</td>
<td>Safe</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,WHERE</td>
<td>Increased</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,WITNESSES_NAMES</td>
<td>Increased</td>
</tr>
<tr>
<td>FORM_TEXT,ITEM,LOCATION</td>
<td>Increased</td>
</tr>
</tbody>
</table>

- Amsterdam
- Groningen
- Gronin
- Groniek
- Groningen
- FORM_TEXT,ITEM,OTHER,INSURANCE,COMPANY         | Safe        |
- FORM_TEXT,ITEM,OTHER,INSURANCE,HOLDER          | Safe        |
issue by reporting it and improving the overall precision. Also, using this feature, they could see on which field in the Data Overview section to focus the most.

Figure 12: Suspicious Data

8.1.4 Claim Assessment Summary Box

The cycle of the process of claim reviewing required the claim assessment team to open the claim overview. This page was also mentioned earlier as it was the page to which the flower of the tulip lead. In order to save time, and to group other warnings which were relevant to Claim Assessment and were scattered around the page, the Claim Assessment box was added to this page (Figures 13 and 14).

As mentioned, it consists of at most three, and at least two parts. First contains the warnings regarding the data collected from the input on the claim form, and could possibly be empty. Second represented the brief
and condensed information on how many fields are irregular. This directly corresponded to the explanation given in the Data Overview subsection. Finally, the third part contained the overall assessment of the claim, which corresponded to the cluster which it belonged to. Also, there was a button which lead back to the Background data, for the sake of saving time and making the interface more user friendly.

Figure 13: Claim assessment summary box with all fields

![Claim Assessment](image)

Figure 14: Claim assessment summary box with some fields

8.2 Complexity of the Representation

The complexity of the entire solution was intended to be as low as possible. It was built with the ease of use on mind, which was supposed to contribute to the acceptance of the tool in everyday work. The entire workflow was designed in such a way that it integrates into the existing solutions. Moreover, the design follows the steps taken when assessing a submitted form.
Because of this fact, the platform developed is easy to comprehend and use continuously.

At the moment of writing the solution was yet to go through full scale testing by the claims team. It has been tested on simulated scenarios and has performed well. In addition to this, the project owner who was representing the claim assessment team, as well as the developers working on the project have backed up the implemented interface. Taking this into consideration, the data representation, and communication of the results to the user was deemed as a success.

9 Final Product

Having implemented all of the components of the platform, they could be integrated into the existing solution, and by unifying them project gained the cohesiveness. Moreover, it became possible for the claims team to get broader information when assessing the claim. This made it easier for them, and the outcome became more precise. However, there were some flaws which were necessary within the set scope of the project, and when taking into consideration that it was the first step towards a bigger and broader goal.

In order to get the best picture of the overall project in the sense of how good it was and how well it was implemented, there were several aspects of the assessment. However, we have focused on the three different sides which have described the project from all angles. Namely topics which were discussed are, the assessment of the solution from the technical approach, the impact on the process for which it was designed, as well as the success of the project to answer the set requirements.

9.1 Technical Assessment of the Solution

The solution was implemented using an adequate approach, and the appropriate technology and algorithms. Although there was some space for improvement, the current implementation was decided on because of the circumstances in which the project was developed, and as such is highly successful. This meant that sometimes the first choice was not possible, like it was the case with supervised learning. However, although this approach was not used, the afterwards implemented solution lead to the desired outcome. It also made it possible to make the transition as easy as possible by separating the concerns of the data processing, and the machine learning specifically. This was analogue in similar cases.
Furthermore, on the side of the data collection, the main difficulty was the uncertainty of how the collected data would behave and only being to react in the end, after it was collected. The unpredictability of where the focus should be, as it turned out, had affected the solution by lowering the precision to some extent. However, through adjusting the data and the algorithms used, even this issue has been overcome, and an acceptable margin of error has been met. As it was, the solution was capable of satisfying the need of the project until another implementation in a later stage was made, where new listeners and a different systematisation of the data would be used.

Finally, in addition to the two mentioned aspects, communicating the data to the user was implemented without any issues, making all aspects of the program highly successful. All of the user interface answered the demand, and all of the implemented features kept the people who were to seek the data in those features well informed, with a lot of insight on the regularity of the claims. Together, the technical decisions in the implementation made it possible for the platform to reach the goal and provide the necessary computing and decision power to the employees working at the claims department. The technical aspect behind the project went hand in hand with the assessment process, and as such complemented it completely, making it an overall success.

9.2 Impact on the Process

As hinted, the impact on the process of assessing the claims was big in the sense that it moved some of the responsibility from the employees. On top of that, it provided them with additional information which should make it easier for them to make the proper unbiased final decision. There was a lot of space for progress, however, the core functionality needed for them was introduced.

One small uncertainty was that there was not enough time for real situation testing. Meaning that the claim reviewers did not have enough hands-on time in order to give thorough feedback. Instead, it was assessed based on the capabilities and short testing period. A contribution to this testing was given by the project owner who compared the functionality and the impact to his long-term vision. The judgement was that what the software provides perfectly aligns with the long term vision for the project.
9.3 Success of the Project to Meet the Requirements

The last thing to examine was how the built software answered the set requirements. As these were not so strict, and provided rough guidelines for where the project should be in the end, it was not easy to determine the definition of success in this sense. An exact comparison method did not exist, but was more open to interpretations. As a solution to this issue, we could assess it based on two criteria - how were the resources, mainly time and data utilised, and how are the final users and stakeholders satisfied.

The former was a broad term, again a bit vague and biased. However, it needed to be taken into consideration that a part of the time was needed for getting familiar with the code base and the technologies used, as well as the structure inside the company along with the development rules. A positive fact was that the efficiency at which the tasks were completed certainly grew as the time passed. On several occasions there were bottlenecks in the said efficiency, however they were often followed by major breakthroughs in the project. The problems which caused them were intimidating at first, and definitely challenging as they were unfamiliar, but the time was wisely and optimally managed. Moreover, the data which was obtained was thoroughly combed for useful information which was subsequently used for assessing the claims. This process left only a slim possibility for some data being not extracted and not utilised properly. However, it was also worth realising that the most important parts of the information were completely used.

On the other hand, the stakeholders encouraged the project to continue. It was generally well accepted that it answered the main idea and translated it into a working software. Hence, it could be concluded that globally, the project was a success from whichever aspect it was examined. This opened new possibilities for further development and research which could continue on the set foundations.

10 Encountered Issues

During the course of the project, there were many uncertainties and issues which had to be overcame. They represented motivating checkpoints which steered the project on a specific track. Every decision was based on the knowledge gathered through the project, and as such improved over time.

Naturally, some of the issues which were encountered seemed trivial in the end. This was a good metric as it meant that a lot of progress was made and that new insights were gained. Finally, and maybe even most significantly, if the project was to be re-done now, the obtained insights on
the architecture would make it possible to evade a lot of mentioned issues.

10.1 Initial Uncertainty

As said, in the beginning, it was confusing on how to design the tool architecturally. Moreover, the steps needed to arrive to the goal were unclear and not well defined. It was difficult to find the starting point and to commence the development. The key was in splitting the tasks in separate parts.

This made it possible to overcome each of them separately, and solve the issues efficiently. Smaller problems were more manageable, and more easily put together in the end. This lesson saved a lot of time overall, and incorporated the three parts of the project which were depicted in detail - data collection, processing, and representation.

10.2 Not Enough Data

Working with data was demanding. It required prompt changes, and new solutions. At the start it was a difficult to predict how the data will look like and plan beyond. Then it was tedious to process it since it was unlabelled, and most importantly since it was not a dataset which could make a perfect distinction between regular and irregular data.

Unfortunately, these issues were the most difficult to overcome. Generally it was needed to adapt the solutions instead of the actual data. This resulted in most of the decisions during the project being made because of, or while taking into consideration the relation with the data.

10.3 Claim Department Interaction

As the entire tool was intended mainly for the claims team it was necessary to have their feedback. This would serve as the mean of getting to know more different opinions, and hear new ideas. These ideas might change the approach, or adjust the goals to those most relevant.

However, due to logistical issues, the direct communication was not possible. This was substituted by discussing the ideas and steps with the project owner who represented the claims team. It meant that it was difficult to assess precisely the direction, and that it changed often. Also, in combination with this, the lack of data until the end, which prevented the algorithm from being implemented fully earlier, prevented a large-scale testing phase. This hands-on testing was an issue as it was a source of valuable data. However, by discussing and using the software, this was to a certain extent compensated for.
10.4 Technical Issues

As smaller when compared to the previous ones, the technical issues required less thinking forward when solving them. However they were often quite demanding to track and to solve. Ones of the most interesting ones are as follows.

For the purposes of data collection, a claim id needed to be sent to the user and stored in the JavaScript script as a variable. However, because of the asynchronicity in JavaScript, on some occasions, although very rarely, it would arrive to the script after it was needed, in turn producing an error.

Also, in the claim form, a date picker was needed. The implementation relied on JQuery, and populated a text field based on the value. However, an interesting bug would occur when the built-in Google Translate for Google Chrome was translating the page. Namely, after selecting the date from the date picker, the date in the text field would be NaN-NaN-NaN. This was surprising, as it took long time to realise what was the issue. It was fixed by replacing the date input with a mask. This also benefited the data processing as the date field was now standardised.

The last issue which is worth noting were the often changes in the business decisions. It caused issues like losing time in order to adapt to the new technologies, as well as to rewrite the existing ones. This was tightly connected to switching to new listeners, and listening to new data. Both sometimes it required a significant amount of time to pass in order to obtain relevant real-world claim results.

Altogether, some issues made making progress more difficult then others. These left a trace on the project by shifting the course of the project in order to avoid them. However, all of them only moved the project onto the right track, and moved it towards completion.

11 Improvements

Although the project was completed in a successful manner, and with answering the required points, in the end it still contained certain points of possible improvement. These have varied from small and neglectful to big and significant. however, in the end based on relevance and ease of implementing them, some stood out as rational possibilities of improvement.

As this project was the first step into developing a solid solution, these will be described in detail. They could be chosen as smaller projects which utilise the existing framework in order to improve the existing functionality, or to introduce new features.
11.1 Better Naming Conventions

Although communicating the outcome to the user was a well working whole, a bit better names for the fields and variables were needed. This was so that there was no overhead for new employees who need to realise what depicts what. Currently the implementation was working perfectly well for the machine, but the users might find it in not the best format.

In order to fix this issue, it was needed to update the way fields were processed and to assign new names to them. Maybe, if proven manageable, the fields could be stored in the database with human-readable names associated to them. However, this could only be done upon agreeing on the claim form format which was uncertain before the completion of this project.

11.2 Better Database Incorporation

Unpredictability of which data to collect and assess, and the behaviour of the processed data forced a sub-optimal database design. Although this was a perfect solution at the beginning of the project, it needed some refactoring and redesign upon obtaining all of the relevant information.

Improvement of this would have saved a lot of querying, and would have made the data prepossessing a lot less computationally intensive. Finally, based on the results of the project obtained after a significant testing period, it might be even possible to store only some metrics, and not the entirety of the data. Although a possibility, this would probably not fit in the idea behind the project.

11.3 Security

Upon completing the flow of the software, some deficiencies in the security were noticed. Namely, the claim form visit id and the user id which were necessary for making the API request to store the data in the database in an adequate place. As they were obtained as a response from an API call, theoretically it was possible for them to be altered in the JavaScript of the file. However, because of the transpiling and minifying of the code, this was not something what was expected and high on the list of the priorities.

Although these matters are not urgent and crucial, they were still a bit of an issue. Along with fixing them, a security analysis of the entire platform after the integration was also advised as it could represent a good source of improvements.
11.4 Data Processing

One of the core parts of the project was the data which was obtained, along with additional information describing it. In order not to be overwhelmed by the amount of it, project focused on the most apparent significant components. It was possible that other pieces of information could be collected so that they contribute to the result a lot more than possibly seen at first sight.

This would mean that it could be used in a different way, with new metrics. Also, upon defining the format of the data, existing metric extraction could be optimised, and a workaround for an inefficient normalisation could be found. As a result of this, the data could have well extracted metrics contributing in synergy to the solution which should in turn prove more precise and more fine-tuned.

Finally, the introduction of labelled claims could be helpful to the goal of accuracy, as this would make the use of supervised machine learning algorithms possible. In combination with the aforementioned, there was even a possibility of labelling the data per input field and assigning different weights to each of them. Altogether, this was the subject with the broadest spectra of possible improvements, and as such was suggested as the next step in a subsequent project.

12 Conclusion

The project as a whole was concluded in a timely fashion, and professionally. Moreover, it has adhered to all of the requirements provided in the initial phases, and all of the stakeholders deemed it as a success. Moreover, the readiness for expansion and improvements make it future-proof, as well as a good base for further research and development. It served as a functioning framework for the automatic claim assessment, which allowed compatibility and flexibility.

The order of features developed was optimal, and served as a good lesson on how to plan the work when the tasks are interconnected and dependant on each other. Many deadlock situations were evaded and the problem of the collection of the data was solved in a way which made it possible to complete both the data collection, and data processing. This was certainly not easy as the time needed for the actual users to fill in the claim forms fell in between the two.

Furthermore, it was a great insight on how to approach a new project inside a company. This meant on understanding the other developers’ changes,
as well as the requirements of the systems, or the management. Knowledge on how to balance all of the aspects most definitely proved valuable and precious.

All of the mentioned chapters represented a part of the project. They split this thesis into separate pieces, making it readable and organised. More importantly, this systematisation made it possible to isolate each and every component and to examine it in more detail in the future.

Finally, all of the topics discussed in this thesis, as well as all of the tasks completed in the development of the project gave an answer to the research question. The process of assessing the insurance claims and checking whether they are fraudulent can be safely automated. However, the extent of that automation is still an open question. Delightfully, it received a lower bound which lays at the syntactical analysis of the input data. Hopefully, in the future, it could also be moved into the area of examining the semantics of the data. Even more optimistically, the process could possibly be fully automated in the future by feeding an Artificial Intelligence agent with the set of business rules, where it would using methods such as ones described, and new ones such as Natural Language Processing, fully replace the need for human assessment and revision.

In order to conclude, it must be said that the project was an immense pleasure to work on, as well as to complete. It felt that it was a great contribution to both scientific and industrial sector. Knowledge and experience gained while working on it will most definitely prove valuable in the future, both in the field of research, and for the personal development of those included in it.
References


