Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D3.9 Demonstrator of Visual Support for Designing Detection Models (Final version) (M30)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: Deliverable D3.9
Document version: 1.0
Document owner: Franziska Becker (USTUTT)

Work package: WP3
Task: T3.5 Visualisation support for the design of attack and anomaly detection model
Deliverable type: Demonstrator
Delivery month: M30
Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document History:

Version	Author(s)	Date	Summary of changes made
0.1	Franziska Becker (USTUTT)	2021-09-28	Initial outline
0.2	Franziska Becker (USTUTT)	2021-10-20	Added vis changes description
0.3	Franziska Becker (USTUTT)	2021-10-25	Review ready version
1.0	Franziska Becker (USTUTT)	2021-10-29	Incorporated review comments

Internal review history:

Reviewed by	Date	Summary of comments
Milan Cermak (MU)	2021-10-26	Review of content, suggestions for description improvement and extension
Arthur Drichel (RWTH)	2021-10-27	Grammar, spelling, content.

Executive Summary

This deliverable describes the final iteration of two visual analytics systems developed in the scope of task T3.5 “Visualisation support for the design of attack and anomaly detection models”. First, the context of these systems in SAPPAN is discussed. Then, updates to each of the two systems are described in detail. The first system is situated in the area of explainable artificial intelligence, as it aims to support developers of deep learning models to better understand their model results and inner workings. Its starting point is the SAPPAN use case for domain generation algorithms and the deep learning models developed to that end in task T3.3. The preparation for a user study to evaluate the system is described in close detail, followed by changes to the second visual analytics system for the visualisation of host profiles. This system illustrates network behaviour and allows its users to evaluate the performance of machine learning models that perform host classification. Finally, the deliverable discusses future work and conclusions.

Contents

1 Introduction

This deliverable is the final version of the previous deliverable D3.8, which contains a detailed description of the visualisations developed for the task T3.5 *"Visualisation support for the design of attack and anomaly detection model"*. First, we discuss the SAPPAN context of this deliverable, and then we report the modifications made to both the visualisation for domain generation algorithm classifiers and the visualisation of host profiles. For the former, we are still in the process of performing an online study to evaluate our system at the time of writing. Final results will be presented in another work package and in a scientific publication if possible. In this deliverable, we discuss the preliminary results available at this point in time.

2 SAPPAN Context

In the context of SAPPAN, the efforts described in this deliverable can be seen as part of local detection. Local detection is often carried out cooperatively by automated algorithms and trained models, while handling is performed by human operators, such as experts in security operation centres (SOCs). Before automatic solutions can be deployed to real-life scenarios, they must be thoroughly tested and understood. Being able to probe models more deeply, e.g., when they behave in a suspicious manner, is also critical in order to comply with transparency and explainability demands. The visual analytics systems developed for this part of the project aim to assist that process in two different use cases, namely DGA classification with deep learning models and network behaviour in combination with host profile analysis.

3 Visualisation of deep learning models for DGA detection

This section describes the changes we made to our visualisation system compared to the initial deliverable. For a detailed description of the system, please refer to D3.8. Subsequently, we discuss the evaluation, including the technological preparation, methodology and preliminary results.

3.1 Modifications to the visualisation system

In D3.8, we presented a web-based visual analytics system consisting of two main views, displayed in two different tabs. The first view contained a histogram matrix that showed the user the predictions, domain lengths and character occurrences on a per-class level. In addition, it contained a selection interface to define complex selections that can be analysed in subsequent visualisations. These subsequent analysis visualisations are created for all trainable layers in the model and allow the user to gain a deeper understanding of the model's behaviour. The visualisations consist of a class overview, a density-scatter plot, a cluster overview, a decision tree and a connected Voronoi diagram.

During the preparation of the evaluation and based on feedback from a preliminary test study, some modifications were made to the visualisations and interactions described in deliverable D3.8. One change that affects all components is the renaming of *category* to *class*, since that conforms to more standard terminology in the area and is thereby less likely to cause confusion. Some of the changes we implemented were also due to the different layout requirements demanded by the study setting. An illustration of the new layout is shown in Figure 1 and Figure 2.

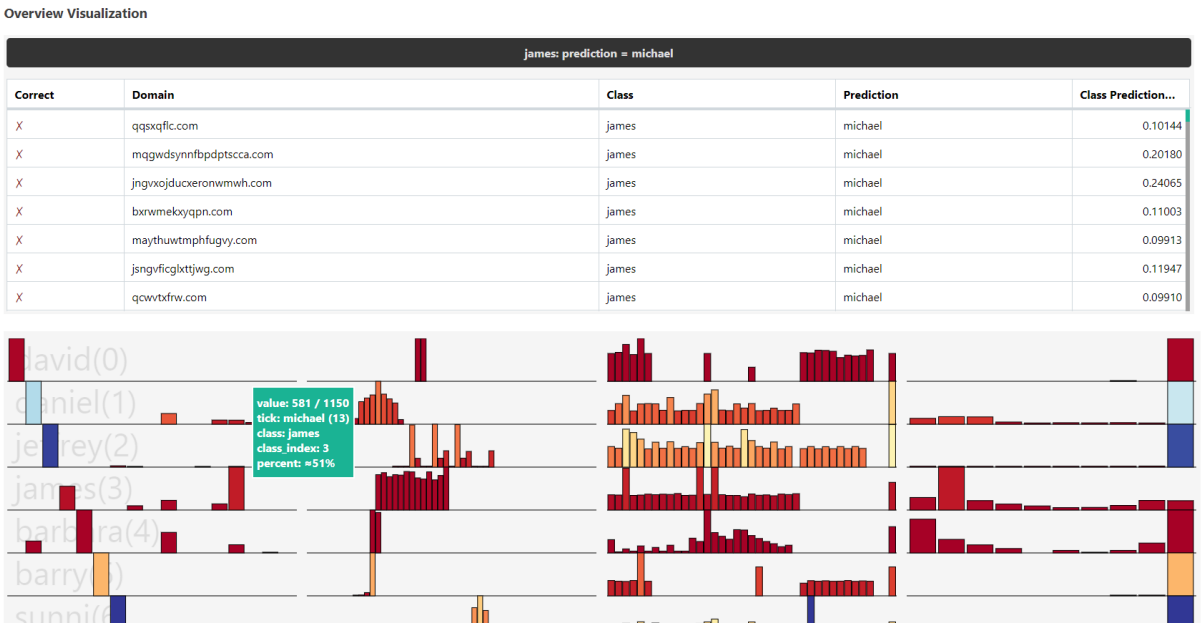

Figure 1: The (new) example table and modified histogram matrix components in the study, together with a tooltip.

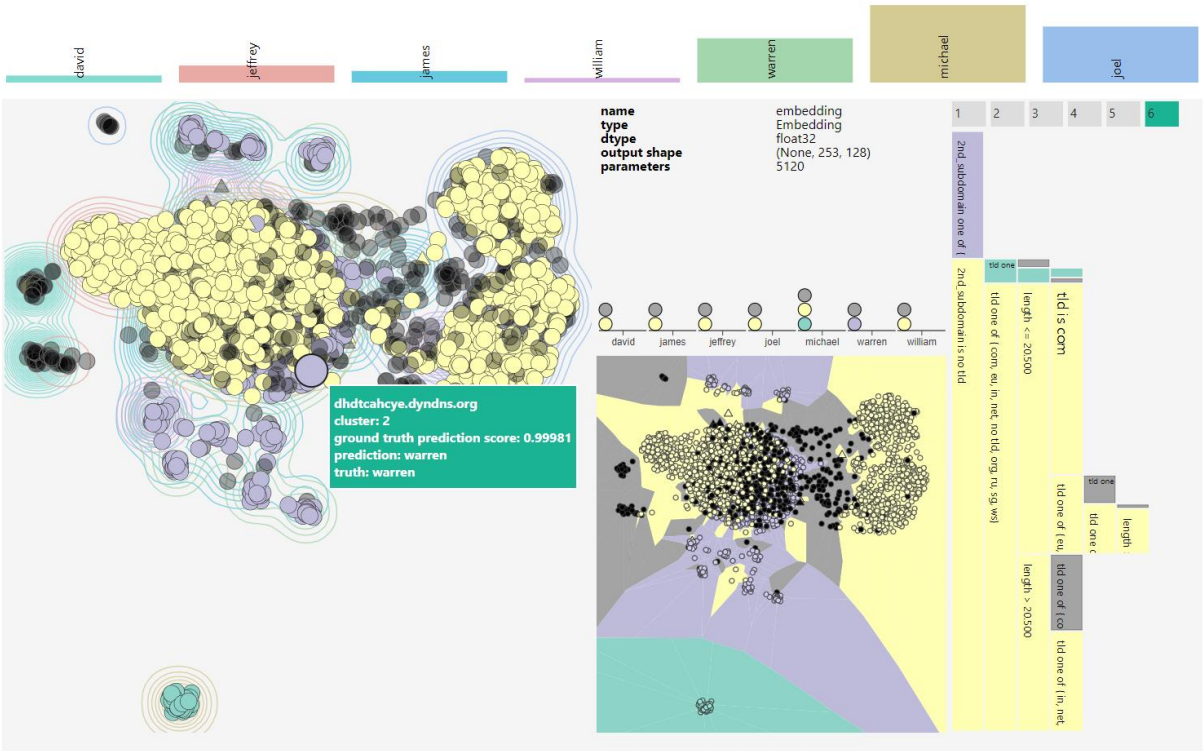

Figure 2: The modified analysis visualisations in the study, together with a tooltip.

3.1.1 Histogram Matrix

For the histogram matrix, we added another column that shows a binned version of the prediction scores that the model outputs for the instances of a particular row, as illustrated in Figure 3. For example, given a class “ruth" the prediction score column for that row would show the prediction scores the model outputs for that class, not for the predicted class. Another change we implemented was to limit the rows to the maximum value that occurs in the dataset, thereby freeing up more horizontal space. We also added the class index to the transparent label in the back of each row. Lastly, the tooltip

was updated to display all available information and the colour scale was switched to a red-to-blue colour scale that is more common than its blue-to-red counterpart is.

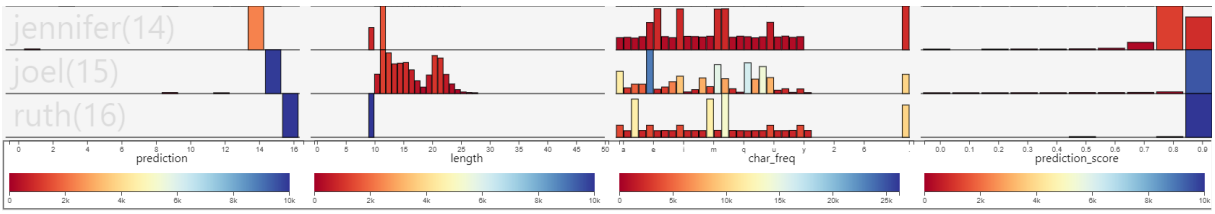

Figure 3: The reworked histogram matrix with the new prediction score column, reversed colour scale and reduced length column.

3.1.2 Example Table

Besides modifying components, we also added one: an example table, as shown in Figure 1. To help users get a better idea of the dataset without having to load the analysis visualisations, we let them view a limited number of examples. These examples can be viewed by clicking on a bar in the histogram matrix, which automatically loads examples into the table. The table contains the columns:

- **correct** – whether the model prediction is correct, indicated by a green check mark if it is correct or a red cross if it is false
- **domain** – the domain an instance represents
- **class** – the ground truth class
- **prediction** – the models prediction for that instance
- **class prediction score** – the prediction score for the ground truth class

The table can be sorted by each column, either in ascending or descending fashion, by right-clicking on the specific column. The sorting can be reset in the same manner.

3.1.3 Class Overview

The class overview was reworked to fit the layout structure of the study interface. Instead of showing it on the left on the analysis visualisation, it is now displayed above it. In addition, it is no longer simply a legend made up of coloured rectangles and the class names, but is more akin to a bar chart (cf. Figure 4). Each rectangle's height is scaled according to the number of instances from that class in the selection. The class name is additionally written across each bar, rotated by 90 degrees. Hovering over the rectangle or the name prompts the display of a tooltip that shows the class name and number of instances of that class in the selection. Clicking on a bar or class name highlights both the class contours as well as the corresponding instances in the density-scatterplot, with the addition of now also highlighting the matching element in the cluster overview.

Figure 4: The class overview in the first version (left) and the reworked version (right).

3.1.4 Density-Scatter Plot

The density-scatter plot only experienced a minor change in the form of glyph scaling. Instead of scaling the size of glyphs by the norm of their activation, all glyphs have the same size. We implemented this change to prevent users from correlating the size of a glyph with its importance and to prevent perception difficulties.

3.1.5 Cluster Overview

The cluster overview previously showed a horizontal band for each class in the selection with a small bar chart displaying all clusters for that class, where the size of the bars indicated the number of corresponding instances (cf. Figure 5 on the left). To avoid visibility problems with very small bars and in order to handle a larger number of clusters without sacrificing visibility, we decided to display equally sized coloured circles for each cluster (cf. Figure 5 on the right). This has the disadvantage of not directly showing the number of instances that belong to a cluster, but it allows for easier comparison regarding the occurrence of clusters in the different classes, in addition to the previously mentioned advantages. How the circles are arranged spatially depends on the number of clusters for a class. Whether there is more horizontal or vertical space, circles are arranged in that direction. If there are more clusters than circles can be put in a single line, we start a new line which allows for a greater number of clusters per class to be shown than in the previous design.

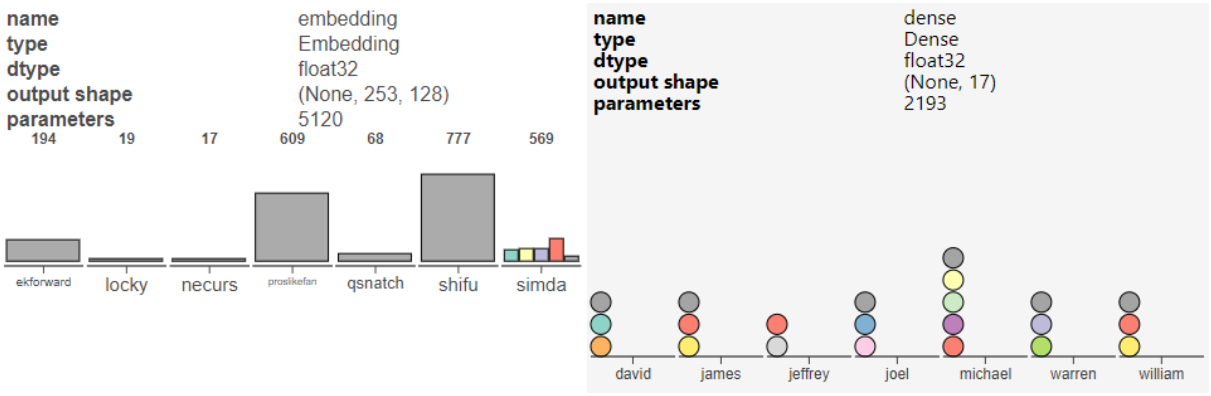

Figure 5: The cluster overview in the first version (left) and the final version (right).

In addition to the visual changes, we also incorporated more interactions. Whenever the user clicks on one of the circles, all members of that class and cluster are highlighted in the density-scatter plot. The circles themselves are also highlighted when the corresponding class is selected in the class overview.

3.1.6 Decision Tree

The decision tree was modified slightly to be more user friendly. Previously, the feature “top-level domain”, which is label-encoded for the decision tree training, was not translated to the set of top-level domains it includes at each node of the tree. This change communicates clearly which top-level domains the instances in a branch of the tree can have. An example of this change is illustrated in Figure 6.

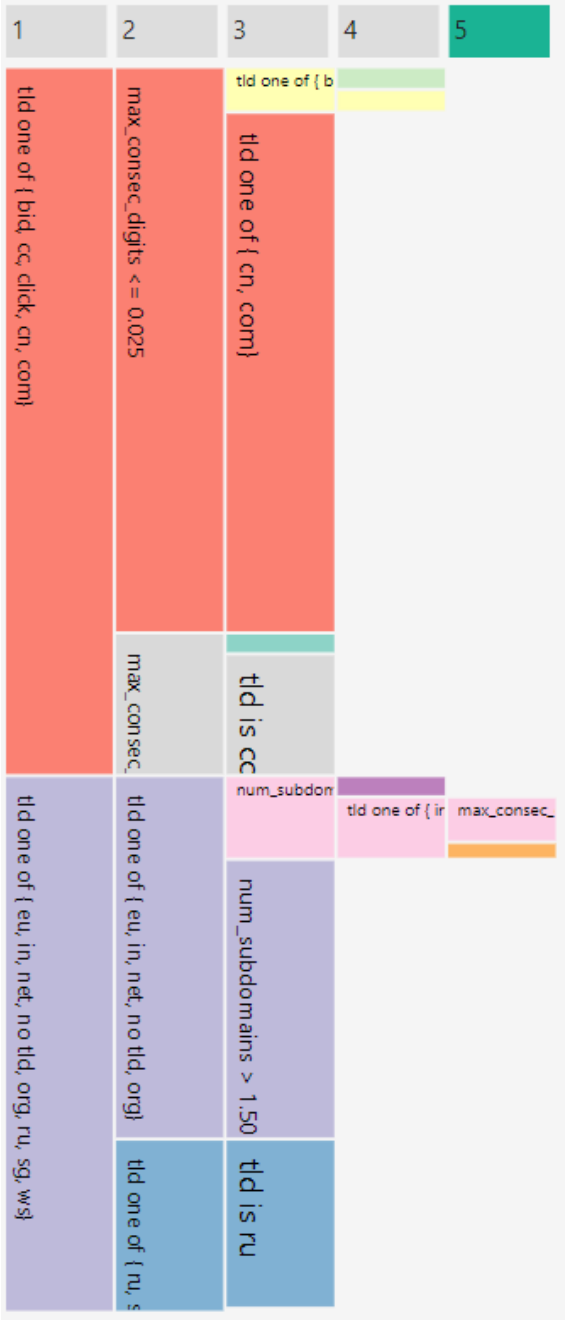

Figure 6: Decision tree with a set description of matching domains (including the top-level domain "no tld").

3.2 User Evaluation

Visualisation systems are designed to be used by people, thereby requiring some form of user-based evaluation to gauge utility, usability or gain some insight into the interactions users perform with such systems. There are, as the bare minimum, two important questions to consider for the design of a study:

- What main question(s) should the study results answer?
- Who are the main target groups that can answer these questions?

The next paragraphs discuss these questions in the context of the developed visual analytics tools for deep learning classifiers for DGA detection. There are many interesting questions to consider in regards to explainable AI (XAI) visualisations. Can the

system teach people the basic workings of a machine learning model, maybe even in a more efficient manner than other approaches? Do users more accurately judge model outputs with their help? Do users show a higher degree of appropriate trust when using the system, in contrast to established but more basic approaches? Are there differences in the system's utility for different groups of users? How well can users navigate the system, how do they interact with the system's specific design?

Who the most suited users for a study are also depends on the questions the study wants to answer. If we are to investigate performance disparities between groups that exhibit differences in demographic or some other feature such as expertise, we consequently require people from these different groups and the study design must enable these different groups to participate in the study.

For the visualisation we developed to understand deep learning models for DGA detection, we decided to investigate these key questions, also inspired by that tasks formulated by Brittany and colleagues [1]:

- Are there variations concerning performance and appropriate trust between our visualisation and an established baseline visualisation?
- Are there variations concerning performance and appropriate trust that correlate with (self-reported) expertise in machine learning?

Here, performance is defined as the recorded accuracy when making decisions regarding the classifications of a model. Depending on the particular instance for which the model makes a decision, the difficulty of judging the correctness of the model's output can vary greatly. If the data can be separated easily in the first place, gauging correctness is simpler than in cases when the data is inherently hard to separate. This should be kept in mind when choosing which data to present to participants.

Trust, in our context, generally concerns the degree to which a user trusts the output of a model. However, such trust can be misplaced or *inappropriate* when the model's output does not align with reality. Trusting a model with an accuracy of 50% as much as a model that performs with an accuracy of 99% is inappropriate in many cases and may consequently lead to a larger number of incorrect decisions. So simply trusting a model is not necessarily good, the trust needs to be justified or appropriate for the particular model. While the case of simply orienting trust at the accuracy of a model presents a simple scenario, reality may often be more complicated. For example, two different models may have the same overall accuracy but perform significantly different for a specific class. When considering an instance of that specific class, how much each model is trusted should depend on how well that model approximates the *true* definition of that class. Such differences can often go beyond easily measurable quantities: consider an image classifier that relies on hands in the image to classify a type of fish. Even if the model performs well for that class on a given set of data, it does not actually possess a truthful representation of that particular fish class and should therefore not be trusted on new data that may not incorporate human hands in combination with that class of fish. Such a scenario is an example of overtrust, i.e., trusting the model too much. The opposite scenario is when the user undertrusts the model, i.e., the user should actually trust the model but for some reason does not. These interactions in terms of appropriate trust between model decision or recommendation and the user's decision are illustrated in Figure 7.

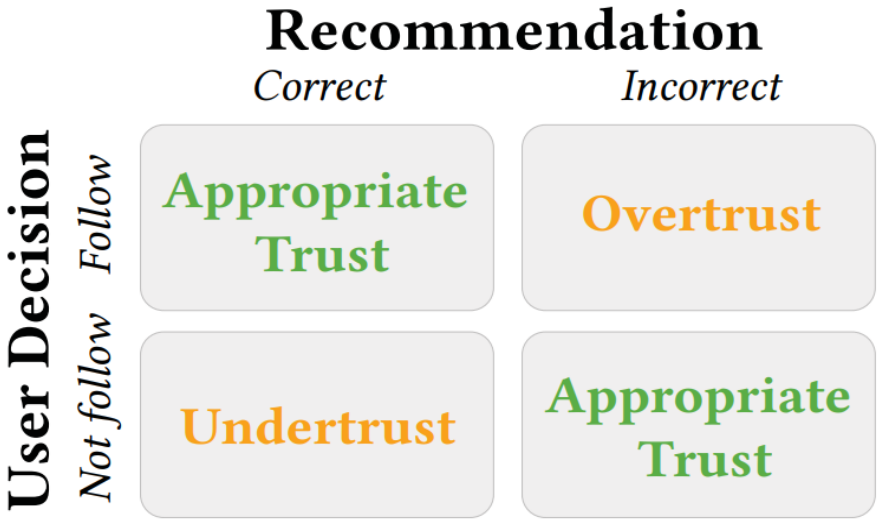

Figure 7: Scenarios of trust in the context of decision recommendations. This figure is taken from [2].

3.2.1 Study Design

A user study may be conducted in the lab, via screen sharing, completely online or via crowdsourcing (usually online). Lab studies require that enough eligible people can be found locally to conduct the study. In our case, this is unlikely and complicated by the global pandemic still underway (to some degree). In addition, it may also necessitate a large time investment when a larger number of participants should be included. The same goes for screen sharing studies where one of the study authors conducts the experiment, but lets the participant view their screen. Crowdsourcing studies have the advantage that they do not require a significant time investment on the part of the study authors. However, crowdsourcing is often reserved for studies that need a large number of participants, can be completed in a small timeframe (less than 1 hour) and do not require a narrow demographic or specific group of participants. An online study is similar to an online crowdsourcing study with the exception that the number of participants is usually smaller, it does not necessarily have to be financially compensated, may take longer to complete and is often not distributed via public platforms but through personal contacts at an organization or institution and mailing lists. We decided to conduct our study as an online study, since that will allow for easy access for all required user groups. At the same time, this frees us from having to actively conduct study sessions via screen sharing, thereby freeing time, and makes it possible to recruit a larger number of participants.

The first question we aim to explore with our study, as described in the beginning of this section, suggests a between-subject study design [3]. In a between-subject design, subjects are divided into groups that test one specific case of the study. For our study, this means dividing participants into a visualisation group (*vis condition*) and a control group (*control condition*). The former performs tasks using our developed visualisations, while the latter perform the same tasks with a common baseline visualisation.

3.2.2 Methodology

This subsection describes the methodology we employed in the design and conduct of the user study.

3.2.2.1 Pilot Feedback

To test our study prototype both in regards to bugs and design, we gave our prototype to five people:

- two employees at the visualisation institute at the University of Stuttgart that work on the SAPPAN project
- two student assistants that work for the SAPPAN project at the visualisation institute at the University of Stuttgart
- one external acquaintance with basic machine learning and visualisation knowledge.

Most of the reported feedback concerned the tutorial texts and UI improvements. However, two testers also criticised a lack of clarity concerning some questions, which we addressed by revising the task questions to better reflect what we want to know from the participants.

3.2.2.2 Distribution

At the time of writing, we distributed our study call at three different institutions via mailing lists, namely at the University of Stuttgart, RWTH Aachen and Masaryk University. For the future, we aim to further circulate our study at industrial and other academic partners.

3.2.2.3 Tasks and Questions

The first task we designed was the *training* task, which participants must complete before the study can resume. The training task exists for the participant to get to know the visualisation in a low-stakes scenario and as a means of gauging their comprehension of the visualisation's design. The training task can be divided into three parts: the tutorial, the visualisation and the comprehension questions. First, we give a detailed textual explanation of the design of all visualisation components, supported by accompanying figures and short tutorial videos. The tutorial videos can be accessed at any point during the study, should the user need a refresher on how to use a particular component. Then the participant can interact with the visualisation, as they can during the actual tasks of the study. Below the visualisation(s), we show four questions that aim to test rudimentary understanding of the system:

1. For which class does the model performs the worst?
2. For which class does the model vary most in the number of different predictions?
3. Why do you think does the model fail to completely separate these two particular classes?
4. Given a specific instance, to which class do you think would the model predict it belongs?

The first and second questions are answered by choosing a class via a dropdown element, while the third and fourth questions must be answered in a free-form text input. The first question is the simplest and, as the bare minimum, requires the user to look at the different classes in the confusion matrix (control) or the histogram matrix (vis) respectively. The second question is similar to the first, but may need more careful examination of the visualisation to answer it with confidence. The third and fourth questions require, as the bare minimum, that the participant uses the example table in combination with the matrix (control) or the histogram matrix (vis). For the vis condition, it would be desirable that the user also employs the analysis visualisations for these two questions, but this is ultimately up to the user. Participants that are not able to answer

any of these questions correctly, especially when coupled with a very short study completion time, should most likely be excluded from the result analysis.

We constructed two different tasks for our study, which we denote as *decision* and *behaviour*. All tasks start with a brief textual description of the training dataset and model architecture. The decision task then presents the participant with an instance not previously seen by the model, i.e., one that is not part of the training dataset. This includes the domain, the predicted class and the prediction score (for the predicted class). For the behaviour task, we show the user an instance that is part of the training dataset and can thereby include the ground truth for that instance. For both tasks, we ask each participant about the following:

- Decision certainty on a scale of 1 to 5
- The visualisations' helpfulness on a scale of 1 to 5
- Which visualisation component affected their decision the most

The visualisations available to the participant are the same in any task. The tasks are modelled after the tasks described by Davies and Glenski in [1]. They differentiate between the following use cases for XAI experiments, as described in deliverable D3.8:

- Model Debugging and Validation
- Model Selection
- Mental Model and Model Understanding
- Human Machine Teaming
- Model Feedback, Challenging and Prescription

For the evaluation of our developed system, we focused on the use cases *model debugging and validation* as well as *mental model and model understanding* and to a lesser degree *model feedback, challenging and prescription*. These use cases consider whether users are able to verify a model's correct, diagnose problems in a model, form an adequate mental representation of the model's behaviour and display appropriate trust towards the model's outputs, which fit well to our study question. We deviate from the templates provided by Davies et al. mostly in minor ways to better fit our specific situation, except in the choice of control condition. According to their approach, the baseline for a task would be a situation with only the person and the machine learning model's output. In our view, taking such a simple baseline most likely does not accurately reflect how many developers and users of machine learning models evaluate and try to understand their models. So instead, our control condition consists of an interactive confusion matrix, a well-established tool in the artificial intelligence community that is available in many common software tools such as scikit-learn [4].

3.2.2.4 Decision Tasks

The goal of decision tasks is to evaluate the accuracy of the participant's mental proxy of the actual model; it is our take on the *mental model and model understanding* use case. In particular, it is an adaption of the task Davies and Glenski [1] briefly describe as "Given information about a model's past performance, match the model to a novel output". In our task, we do not explicitly show participants a number of instances classified by the model and then ask for the model's prediction on a new instance. Instead, our visualisation allows the user to examine data used for training and validation and how the model treats that data. We then combine this with a data instance previously unseen by the model and the prediction of the model. The task goal is to decide whether the prediction is correct or not, i.e., whether the model has a good idea of a

specific class and whether the participant has an appropriate mental representation of the model.

Dataset Generation

To generate datasets for decision tasks, we implemented a strategy where we choose a subset of all classes to include in the dataset, given a trained model and its training dataset provided by partners at RWTH Aachen. To keep the privacy of partners intact, the benign class is removed from the dataset before it is processed. First, we choose two classes that are either highly or only slightly confused with each other. We call the first class “A” and the second “B”. Then we sample a limited number of instances from all instances belonging to class A. From this selection, we take a few instances that are correctly classified by the model and change their label to class B. The other selected instances’ labels are left unchanged. This selection of instances is stored separately and not included in the final training data. Moreover, we add some of the other classes confused with “A” to our pool of included classes. This is the basis for the dataset generation. In addition, we choose some high (per-class) accuracy and some low accuracy classes from the dataset and include them as well. These choices do not necessarily translate one-to-one to the final predictions of the newly trained classifier. As a final step, we rename all classes in the dataset by generating unique names with the help of the *names*[1] python package. This package generates English first names derived from the 1990 census data in the USA, and we make sure that each class is given a unique name. The class names are changed in order to mitigate learning effects between tasks and to lessen the effect previous knowledge concerning DGAs can have on the results of the study. To keep track of our choices during the dataset generation, we store a JSON file that contains which classes were chosen to be included in the dataset, which classes were chosen for the tasks and what the original class names were.

3.2.2.5 Behaviour Tasks

The goal of behaviour tasks largely overlaps with that of the decision tasks, but the execution differs to some degree. Instead of asking participants to choose whether the model made a correct decision on a previously unseen instance, we ask the participant why they think the model made a specific decision, given an instance from the training data. This scenario differs compared to the decision task, in that it directly asks the participant for their idea of the model’s behaviour, since they can also see the ground truth in addition to the model’s prediction. This task is also similar to one of the *model debugging and validation* task by Davies et al. [1], which they summarize in the following sentence: “Given an incorrect model decision and corresponding explanation, determine the reason the model made a mistake”.

This task is exactly what we get when the instance we present to the participant happens to be one that the model classifies incorrectly. However, since we only show instances that belong to classes that are confused with others to some degree, even in cases where the model makes a correct prediction, the user can always form a more nuanced understanding of the model’s behaviour through visual analysis and communicate this understanding in the answer for this task.

[1] https://github.com/treyhunner/names

Dataset Generation

The dataset generation for the behaviour tasks follows a similar structure as for the decision tasks. Given a model and its training dataset, with the benign class removed, we build a pool of classes that will be included in the final dataset. We choose a number of classes that have at least one other class with which they are confused. Some of the correctly and some of the incorrectly classified instances are then stored, to be used as task instances later on. In contrast to the decision tasks, they are still included in the dataset and are consequently used to train the model. As we did for the decision tasks, we also include some other very high (per-class) accuracy and some very low accuracy classes in the dataset. In the same manner as for the decision tasks, we rename all classes to avoid effects from training or previous knowledge and save relevant data for the dataset generation in a JSON file. This is also necessary to adequately analyse participants answers after the study, particularly for behaviour tasks.

3.2.2.6 Appropriate Trust

Since we also aim to assess participant's trust in the visualisation, each of the decision and behaviour tasks include a question regarding certainty. More specifically, we ask participants "How certain are you regarding your previous answer", where the previous answer is whether they think the model correctly classified a new data instance, or in the case of behaviour tasks why the model made a certain decision. Participants must submit a score between 1 (uncertain) and 5 (certain) and are free to add a comment in an optional free-form text field. We chose the word "certainty" as a proxy of trust because we did not want to know how much they trust the model or the visualisations, but rather what effect the visualisation has on the trust in their own decision. Even in related literature, uncertainty is often connected to trust, e.g., when MacEachren et al. [5] describe one level of trust to be "source dependability or the confidence the user has in the information".

3.2.2.7 User Experience & Utility

In addition to the previously described questions, each decision and behaviour task includes two more questions that are related to the user's experience of the visualisations and the effect the visualisations had on their decision. For the former, we ask participants "Did the visualisations help you to assess the model's prediction?" which they must rate on a scale from 1 (no help) to 5 (a lot of help) with the option to provide further comments. For the latter, we ask "Which component or functionality of the visualisations affected your decision the most?" which must be answered in a free-form text field. Finally, at the end of the study we give the participant a last opportunity to provide any further comments or feedback in another free-form text field. Here, they can include any comments that may not have fitted previous text fields.

3.2.2.8 Connection to Established Methods and Designs

Large parts of the design of our study are inspired by the scenarios by Brittany and colleagues [1], who formulate different tasks that rely on falsifiable hypotheses to test the utility of XAI visualisations. In addition, our design can be found in other evaluation categories or patterns formulated in related work. Taking the taxonomy by Isenberg et al. [6], our study can be seen as a type of VDAR (visual data analysis and reasoning) evaluation. VDAR includes evaluations that "assess how a visualisation tool supports analysis and reasoning about data and helps to derive relevant knowledge in a given domain" [6]. This category resonates with our goal to see how well participants can understand the model we give them with the help of our visualisations, and how these

visualisations affect their trust. It also aligns with the general idea of explainable AI to generate insight into and understanding of machine learning models.

Overall, our study is a combination of a quantitative and qualitative study. It is quantitative since we record participant's completion times and accuracy for decision tasks. However, it is also qualitative since the behaviour tasks do not have a clear answer and the comments from participants cannot be analysed in quantitative manner. This is in line with Isenberg's [6] position that many interesting evaluation goals cannot necessarily be studied using just quantitative methods. Our study employs a between-subject design [3], where some participants perform the given tasks using our developed visualisations (the vis condition) and some perform the given tasks with a baseline visualisation (the control condition). This allows us to compare the utility of our approach compared to standard methods employed in the same area.

3.2.3 Technical Details

In order to conduct an online study that includes an interactive visualisation, we surveyed existing tools available and settled on SurveyJS[2] to build our study. SurveyJS is a JavaScript framework that implements many of the basic functionalities needed to construct a study. This includes defining pages, defining different types of questions, marking questions as compulsory, defining conditional questions and more. The questions are defined in JSON format and can be build using the SurveyJS Survey Creator[3], which provides a simple interface to click together the elements that should make up the study.

In order to conduct the study, we copied and adapted the previous Django app into a new one. Most notably, we added new models to the database to track and save data from participants that completed the study. The particular answers for the tasks are stored in the database, while interaction logs are stored in JSON files.

In the back end, we implemented code to decide on the condition a participant should be assigned and which particular instances the different tasks will consist of. For the former, our approach is the following: First, the participant has to fill out a form with their data concerning age, gender and expertise in machine learning as well as visualisation and then consent to the study conditions (cf. Figure 8). This information is then sent to the back end, where we count the number of participants for each condition based on their expertise. We denote anyone with a self-reported machine learning expertise greater than two, on a scale from one to five, as an expert. When we have to choose a condition, we first count the number of experts for each condition. If there are more experts for one condition than the other and the difference is less than or equal to three, then we choose the condition with less experts. Otherwise, we look at the overall counts for each condition and try to balance them. In case the counts are equal, we prefer the visualisation condition. To assign task instances, we choose instances at random (uniformly distributed) but make sure that each participant completes at least one task with an incorrectly classified instance and at least one task with a correctly classified instance. We also make sure that the ground truth classes for each task type (training, decision and behaviour) are different. We added this constraint so that knowledge gained from the first task of a particular type does not unduly affect the completion of the second one.

[2] https://surveyjs.io
[3] https://surveyjs.io/create-survey

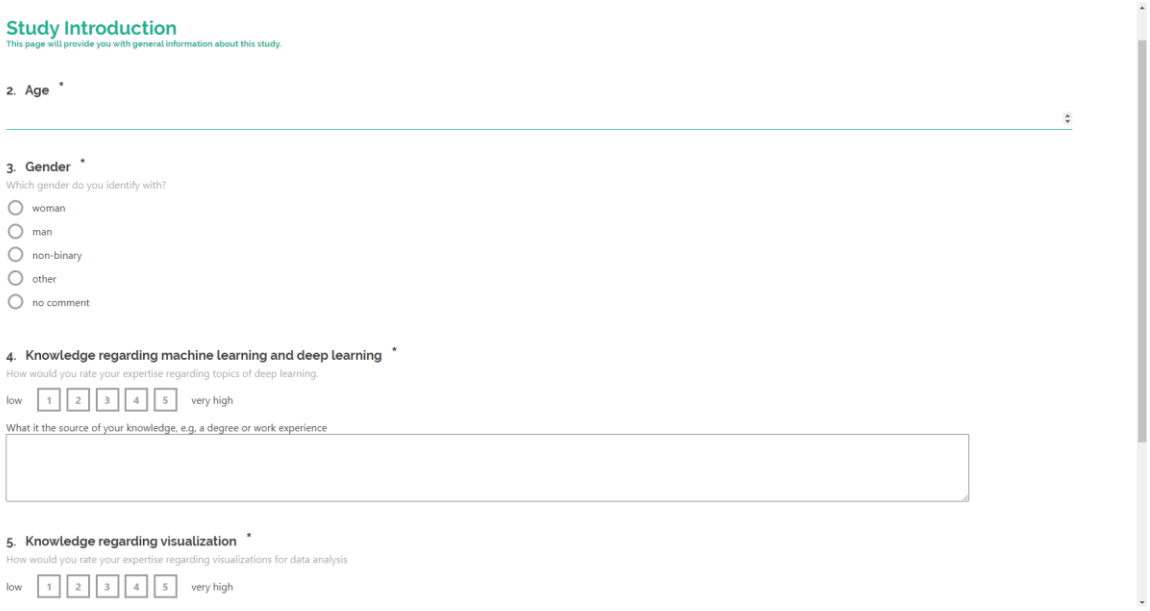

Figure 8: The study introduction where the participant has to fill out their personal information like age, gender and expertise.

3.2.3.1 Safety Measures

In most online studies, it is common to include some safety measures to ensure the integrity of the results. The simplest safety measures we included concern the completeness of information. All questions in the study are defined as required in their SurveyJS JSON file, which ensures that an answer is always provided. SurveyJS also takes care that the format of an answer matches its defined type, i.e., only numbers in the correct range can be input into an integer answer field.

Since our study is publicly available on the internet, we also require participants to possess a participant ID to take part in the study. Such an ID can be requested on the study website via a simple form that only asks for an email address. This email address is checked for validity using the Django email validator. Then, we query the database to test whether this email address has already requested a participant ID before, in which case they will not receive a new ID. If they have not already requested an ID, we automatically send an email with a newly generated participant ID to the given email address. To send these emails in an automated fashion, we created a new Gmail account for this particular purpose. The IDs we generate also have an expiration date, so they are no longer valid after two weeks. Of course, a participant ID is also no longer valid as soon as the participant completes the study.

Another safety measure we implemented concerns page reloading. While we urge participants not to reload the page during the study, since that would erase their progress, we still want to deal with that scenario in some fashion. While it would have been desirable to store the intermediate state and answers during the study, this requires considerable work and testing. Instead, we store the group and task instances chosen for a participant as soon as they are generated. This ensures that when participants accidentally reload the page, they will not see the other condition or other task instances.

Additionally, we also wanted to make sure that participants see the visualisations in a similar manner. Thus, we require participants to have a minimum resolution of 1600 by 900 pixels for the browser's inner window. This is checked whenever the size of the window changes. In case this requirement is not met, we display an alert that prevents the user from continuing until they have resized the browser to the required size. Lastly,

we also wanted to prevent users from disrupting the data-fetching process whenever they request data for the analysis visualisations in the vis condition. This is similarly achieved by showing an alert that prevents any interaction until the first batch of data has been successfully loaded.

3.2.3.2 Confusion Matrix for the Control Condition

For the control condition, we implemented an interactive confusion matrix (cf. Figure 9). The confusion matrix uses the same red-to-blue colour scale as the histogram matrix in the visualisation condition. Cells are only drawn where cell values are larger than zero, and hovering over an existing cell shows a tooltip with more detailed information concerning that cell, like the absolute and normalized values for that class. By default, the colour scale uses the normalized values for all cells. However, it can be changed to show the absolute values by using a connected dropdown element. Finally, we also show a table above the confusion matrix that has the same structure as in the visualisation condition (see Figure 1). By clicking on a cell in the confusion matrix, a limited number of instances that match that cell are loaded into this table for the user to explore.

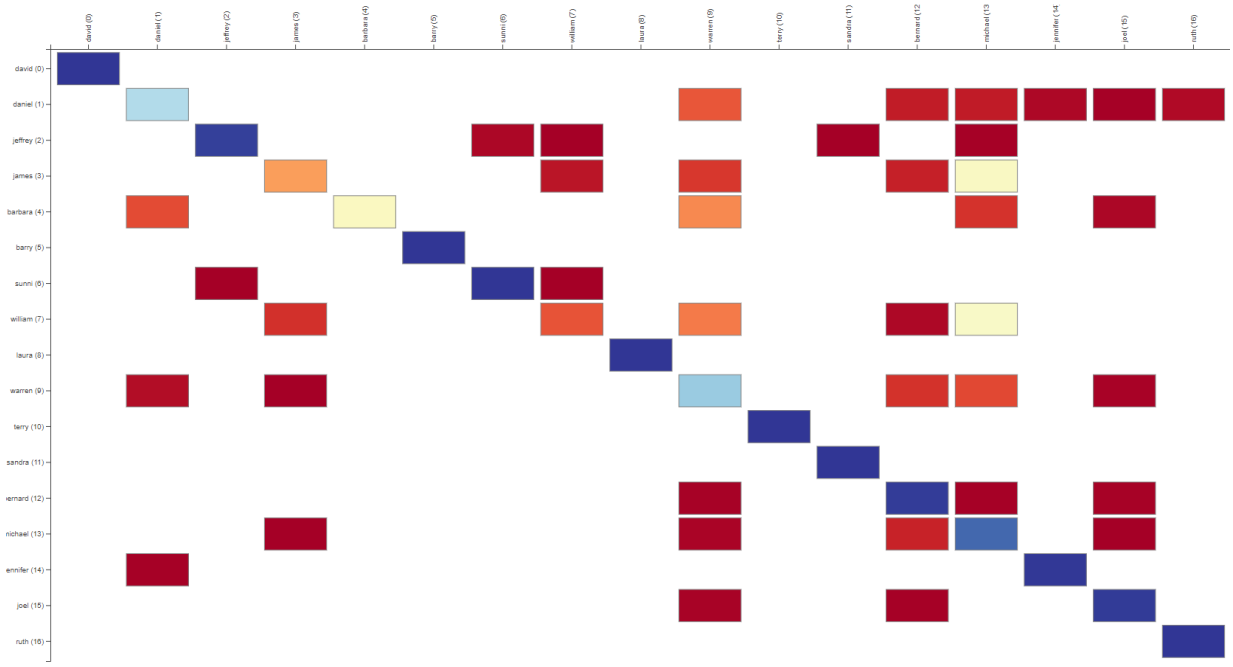

Figure 9: The interactive confusion matrix given to participants in the control condition.

3.2.4 Interaction Log

For the front end, we also added a tracking functionality such that we can record the different activities that participants perform during the study. This is limited to interactions with the study, the majority of which are related to interactions with the visualisation. This allows for more insight into the analysis process of a participant, which is especially important in an online setting where we will always have limited control over how the participants perform the study. We also record some of the HTTP requests performed by interacting with the study, mainly for debugging purposes. Table 1 and Table 2 list which kinds of interactions and requests we log during the study, split by the study condition.

Condition "control"	Description
load examples	When the user clicks on a confusion matrix cell, examples for this cell are loaded and we store which class and prediction the examples have.
change scale	The user has the option to change the colour scale to use either the normalized or the absolute value. Whenever this changes, we record the new value.

Table 1: List of interactions recorded for participants of the control condition.

Condition "vis"	Description
load examples	The user clicked on a bar in the histogram matrix and examples for that bar are loaded. We log what feature and value the bar represents.
clear selection query	The user cleared the selection query, which we log as such without additional data.
view previous layer	The user clicked on the button to show the analysis visualisations for the previous layer.
view next layer	The user clicked on the button to show the analysis visualisations for the next layer.
view specific layer	The user used the dropdown to show the analysis visualisations for a specific layer.
show decision tree level in Voronoi diagram	The user clicked on a decision tree level button at the top of the decision tree visualisation, which shows the corresponding partition as a Voronoi diagram.
highlight decision tree node in Voronoi diagram	The user clicked on a decision tree node, which highlights the corresponding cells in the Voronoi diagram.
highlight class	The user clicked on a class in the class overview and we log what class that is.
highlight cluster in class	The user clicked on a cluster in the cluster overview and we log both the class and the cluster.
selection request	The user made a selection (via the selection UI) for which we store both the request and the response data.
activations request	After the user made a selection, the automatic data loading process for all layers has begun. For each layer, the front end makes a request to receive the data for the density scatter plot. We store both the request and the response data.

decision tree request	The data for a decision tree was requested after making a selection (see *activation request*). We store both the request and the response data.

Table 2: List of interactions and requests recorded for participants of the vis condition.

3.2.5 Preliminary Results

We consider a minimum of 25 participants to be necessary to make any substantial statements about the visualisation's performance in contrast to the control condition. As of the writing of this deliverable, we have data for seven participants that was collected over the span of ten weeks. We believe that the partially long completion times and lack of monetary compensation might contribute to this lack of participation. Consequently, we can only report preliminary results in this deliverable. For the future, we plan to include some kind of monetary compensation, e.g., a chance to receive an online voucher, to motivate more people to participate.

3.2.5.1 Participants

Table 3 lists information for the participants that already completed the study, in no particular order. For easier comparison, some of that information is also displayed in Figure 10 for easier comparison.

	Age	Gender	Machine Learning Expertise	Visualisation Expertise	Condition	Familiar with DGAs
P1	23	man	2	4	vis	no
P2	33	man	2	3	control	yes
P3	24	man	4	1	control	no
P4	25	man	3	4	control	no
P5	27	man	3	1	vis	yes
P6	31	man	2	5	vis	no
P7	24	man	4	3	vis	yes

Table 3: Participant information collected at the beginning of the study.

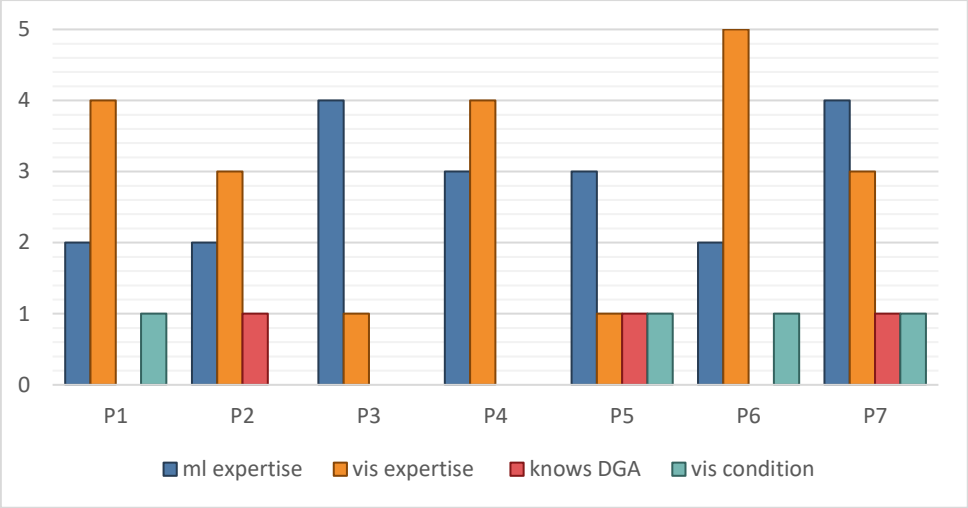

Figure 10: Participant information regarding self-reported machine learning expertise, self-reported visualisation expertise, whether they are already familiar with DGAs (1 = yes, 0 = no) and which condition they were assigned (1 = vis, 0 = control).

3.2.5.2 Results

In this subsection, we analyse the data we collected up until this point in time and consider what these findings suggest, although the small number of participants does not allow for any definite conclusions. Thus, we also do not perform any of the more sophisticated statistical analyses possible for this type of study.

Task Completion Times

First, we looked at discrepancies in task completion times, which is illustrated in Figure 11 to Figure 14. We expect participants in the vis condition to take longer than those in the control condition, since there are more visualisations that require more time to understand and more time to compute the necessary data for. In addition, the visualisations in the vis condition allow for more interaction. Therefore, we hypothesize that participants in the control condition will be faster, regardless of the accuracy of their answers. This is confirmed for the few users that already participated, with a mean completion time of approximately 7 minutes in the control condition and approximately 20 minutes in the vis condition for a single task (ref. Figure 11).

The average completion time for the training task is the largest across both conditions, which is to be expected since it contains tutorial text and the user must complete four small tasks instead of one, as is the case for the two other task types. The amount of tutorial text is also much higher for the vis condition, since the visualisations contain more concepts and algorithms that require some type of explanation to understand the resulting visualisations. This may be one of the underlying reasons why the discrepancy between the conditions is largest for the training task. While the difference between means for the decision and behaviour tasks are 8 and 13 minutes (see Figure 13 and Figure 14 respectively), it is approximately 21 minutes for the training task (see Figure 12). Another likely reason may be that a training effect occurs after the training task, and that this training effect is larger for the vis condition, since there are more concepts and interactions to learn. This would be an indicator that the training task fulfilled its function, as it is designed to let participants learn how to interpret and interact with the given visualisations.

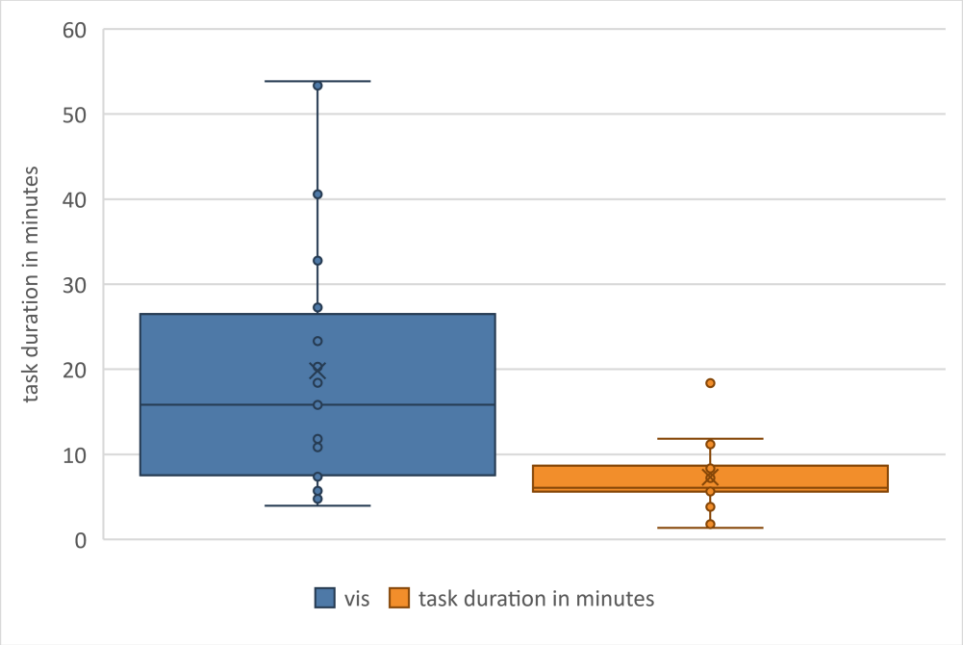

Figure 11: Completions times for each task (5 per participant, including the training task), grouped by condition.

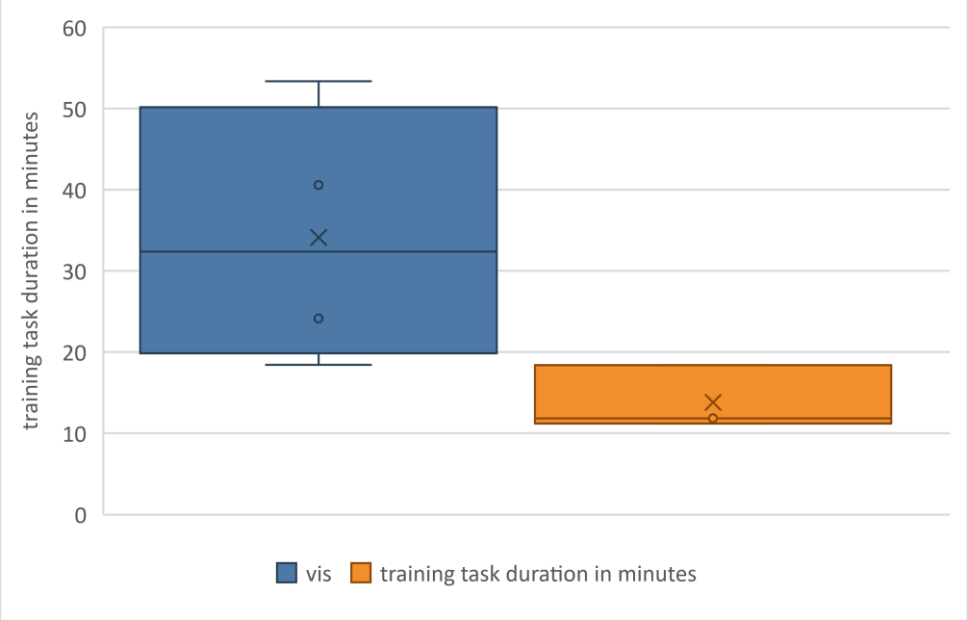

Figure 12: Completion times for the training task (1 per participant), grouped by condition.

An interesting finding to note here is that participants take longer for the behaviour tasks than for the decision tasks in the vis condition, although the decision tasks always precede the behaviour tasks. This is most likely due to the nature of the behaviour task, which requires the participant to more thoroughly understand the relationship of one instance to its ground truth and confused classes and articulate reasons for the model's decision in a free-form text. For the control condition, the mean completion times for

both task types are almost identical. This may be the result of the limited analysis capabilities provided by the confusion matrix and connected example table, whereas the vis condition gives the participant more ways of analysing the model.

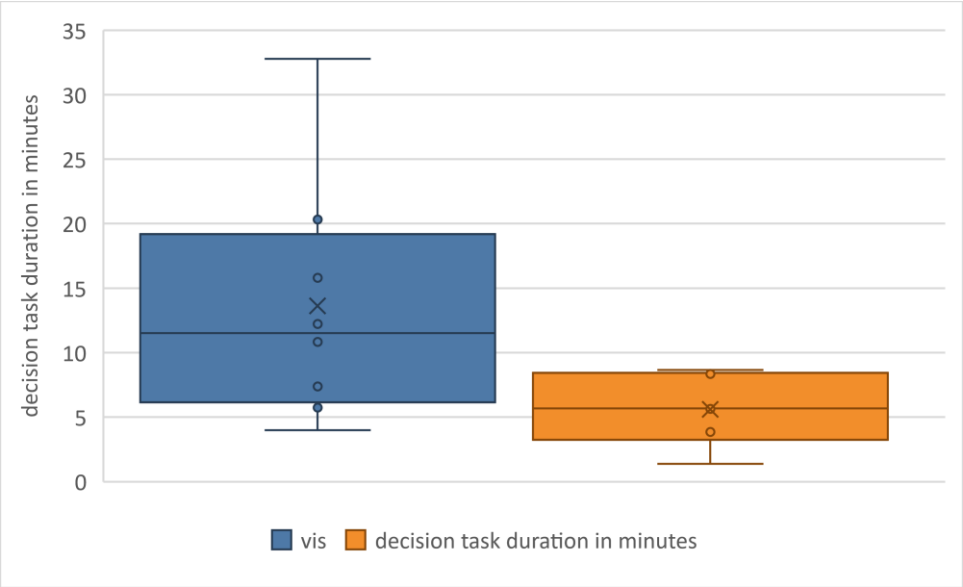

Figure 13: Completion times for each decision task (2 per participant), grouped by condition.

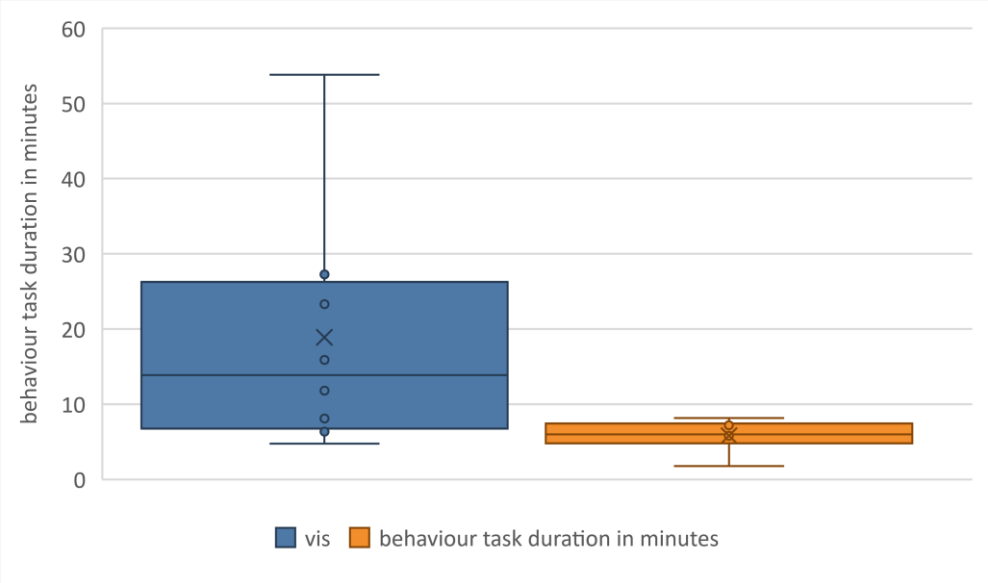

Figure 14: Completion times for each behaviour task (2 per participant), grouped by condition.

Overall Certainty and Helpfulness

Looking at the reported certainty and helpfulness across the decision and behaviour tasks grouped by condition in Figure 15, we can see that the mean certainty is very similar for both conditions, but varies much more drastically in the control condition. The same holds for the reported helpfulness, where most of the participants reported a helpfulness above three in the vis condition and only above 2 in the control condition. Overall, the control condition seems to lead to more extreme values while the vis condition more consistently leads to medium to high values.

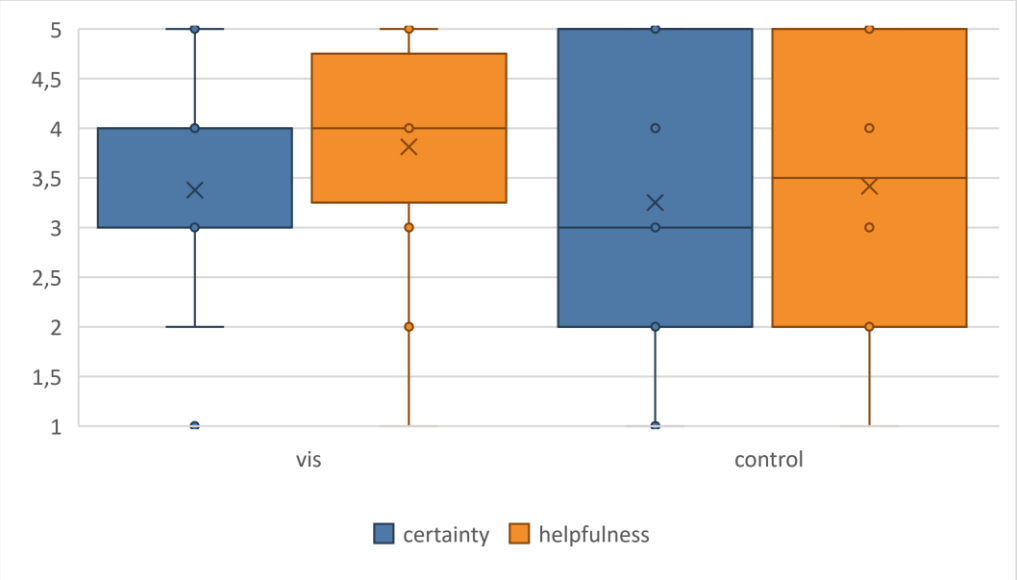

Figure 15: Overall reported certainty and helpfulness for decision and behaviour tasks, grouped by condition.

Decision Task Results

The decision task asks participant to decide, for a new data instance, whether the model made a correct or an incorrect decision. Looking only at the decision accuracy in Figure 16, it is visible that participants in the vis condition perform slightly better with an average of approximately 0.75, whereas participants of the control condition have an average accuracy of 0.5, equal to chance.

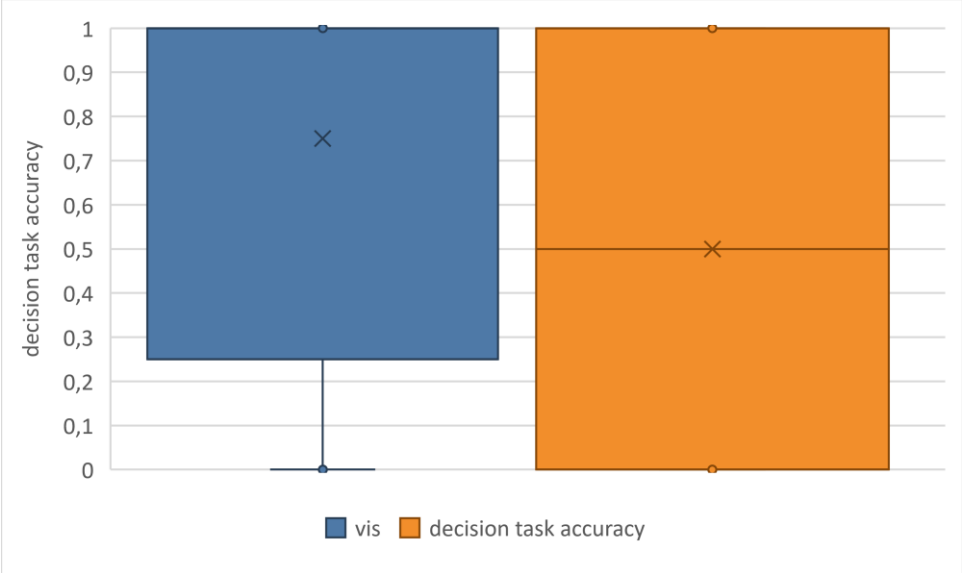

Figure 16: Accuracy for the decision tasks, grouped by condition.

In contrast to accuracy, the analysis of appropriate trust is more complex. The decision task contains instances for which the model should be trusted and those for which it should not. Therefore, we cannot simply consider the relationship between accuracy and certainty, but must investigate whether the specific task consists of an instance where the prediction should be trusted. Consequently, we labelled all task instances with a *trustworthiness* score between 0 and 1, where 0 means the model should not be trusted at all and 1 means the model should definitely be trusted. Instances where it is uncertain that the model should be trusted, i.e., in the case of two confused classes

that are inherently hard to separate, we label these instances as uncertain. In Figure 17 and Figure 18, all decision tasks that participants completed are separated into the categories *undertrust*, *overtrust*, *appropriate* and *uncertain* and we show the mean reported certainty and helpfulness for each condition.

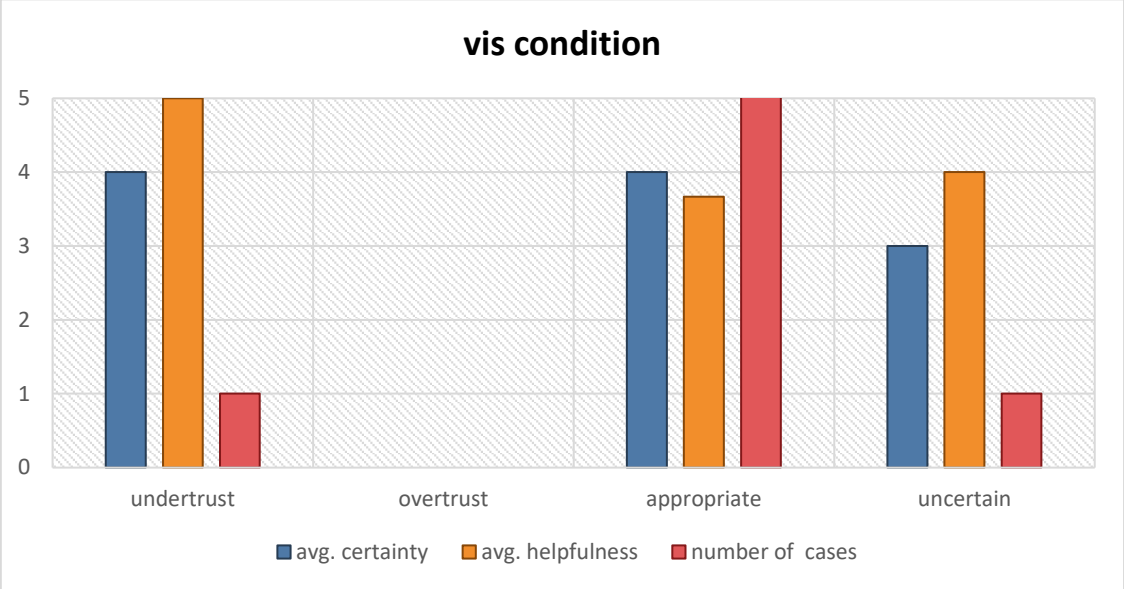

Figure 17: Different trust cases for the decision tasks in the vis condition.

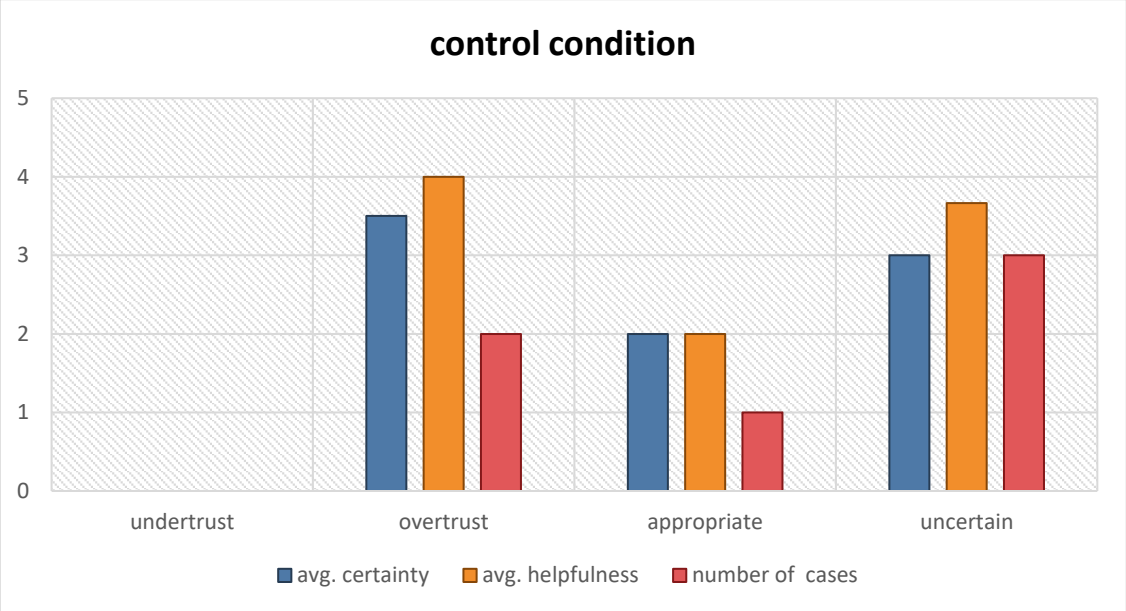

Figure 18: Different trust cases for the decision tasks in the control condition.

In Figure 18, we can see that there are two cases of overtrust where the participant should not trust the model and the trustworthiness score is larger than 0.5 in the control condition. In Figure 17, the is one case of undertrust, i.e., a case where the participant should trust the model and the trustworthiness score is larger than 0.5. In that case, the reported certainty is equal to that of appropriate trust and larger than for the uncertain case. A similar relationship exists for the control condition, although here the certainty for the case of appropriate trust is much lower. Why that is the case is difficult to interpret, since there is only one appropriate case for the control condition and only

one uncertain case for the vis condition, i.e., there is too little data to make any substantial claims.

In Figure 19 and Figure 20, we looked at the differences in reported certainty and helpfulness for the different trust cases, but also consider the combination of appropriate and uncertain cases, under the assumption that any decision in an uncertain case may be considered appropriate. There we can see that the overall reported certainty seems to be lower for appropriate and uncertain cases when participants are in the control condition, even if these two cases are combined. In cases of over- or undertrust, it seems that certainty can be very high or very low, thus we require more data to see how that changes with a larger number of participants. However, we also noticed that the participant of the undertrust case said the following in their final comment:

"I think that I misinterpreted some of the visualisation results in the previous tasks, as the selection I made did not match what I thought I had selected for analysis. I think that I misinterpreted 'length' in earlier tasks, assuming that it did not include the TLD."

This suggests that the one case of undertrust may be due to a misinterpretation of what is taken to be the length of a domain in this study. For the cases of overtrust, participants may have too easily accepted the predictions presented by the model, but this hypothesis requires further analysis.

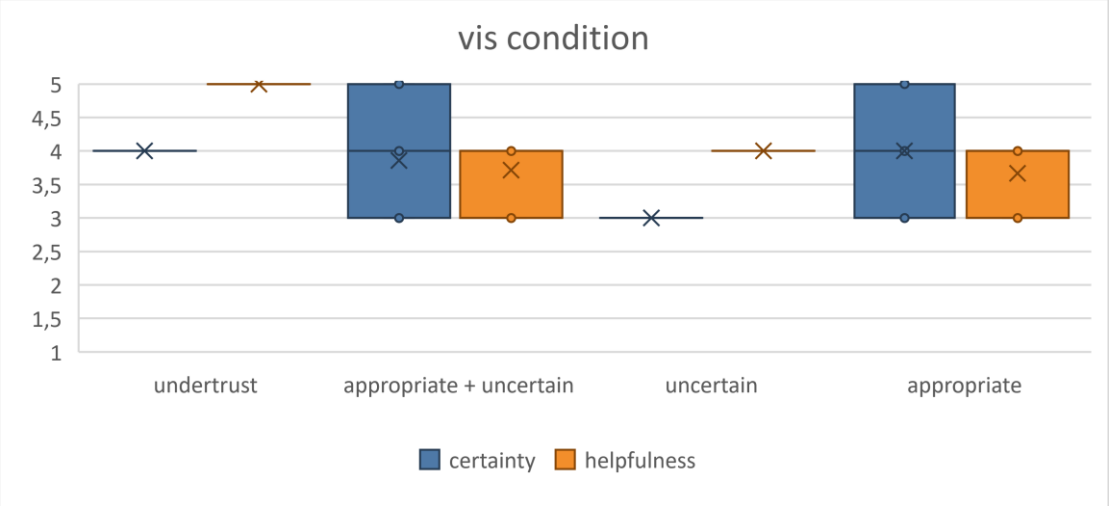

Figure 19: Certainty and helpfulness for the different trust cases in the vis condition.

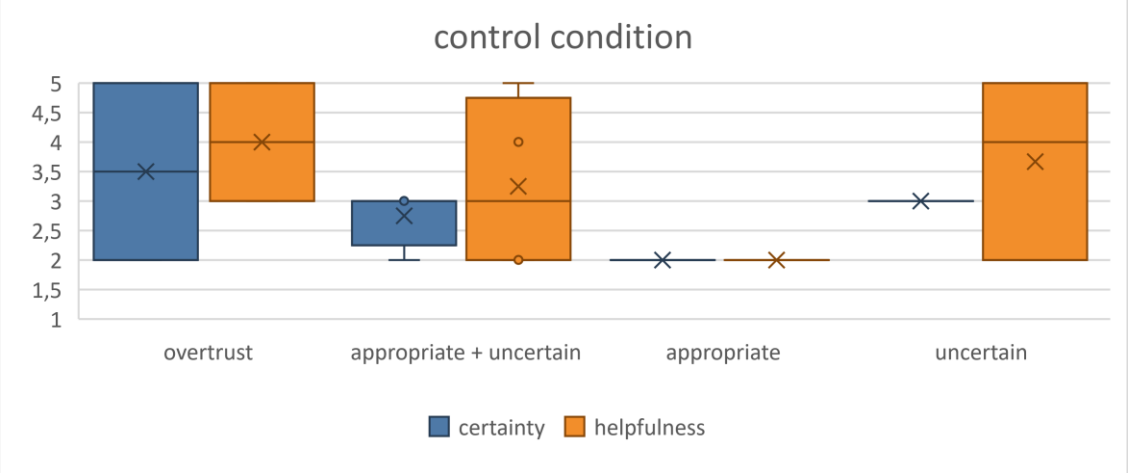

Figure 20: Certainty and helpfulness for the different trust cases in the control condition.

Behaviour Task Results

The behaviour tasks are not analysed as easily as the decision tasks, since they lack an inherently quantifiable answer. Instead, we may examine the text answers provided by the participants to investigate how much their explanation, which we treat as a proxy of their mental representation of the model, overlaps with the model's behaviour.

From looking at the participants' answers, it seems that the answers for those in the vis condition are longer on average and contain more indicators as to why the model classified the task instances as it did. For control condition participants, the answers often focus on easy to identify features such as length or top-level domain. In the following, we list some of the participants answers according to their condition.

Control Condition

"most .net domains (containing only letters) are classified as class joseph"

"the class sarah consists mostly of .ru domains consisting of letters"

"Because instances of sarah and linda are quite similar, instances of linda frequently got labeled as sarah "

"Do not know"

Vis Condition

"joseph training samples are not well predicted in general, so model does not perform well for predicting joseph anyway. spencer samples on the other hand are accurately predicted. a subset of joseph samples share very similar activations to spencer samples in lstm layer. I believe that the given sample looks similar to spencer and the model is doing well in predicting spencer-like samples."

"for the candidate classes for strings of length 21 ending in .ru the sample fits best into the linda class, since the chad and david classes contain rather natural language formed domains and doris predictions perform very badly for for this number of characters in general."

"length > 16.5, many consonants after another"

"many following consonants, size between 15 and 24"

"some character set frequency match / entropy feature learned by the lstm"

These and the other answers suggest that participants in the vis condition can make use of the additional information they can get in regards to model behaviour, with all participants mentioning some component of the analysis visualisations as the most impactful visualisation component for their decision.

Overall Differences by Expertise

Up until this point, we have mostly considered differences regarding condition, but we may also analyse how expertise in the area of machine learning or visualisation impact the results. In particular, we consider how expertise affects certainty and helpfulness, which is charted in Figure 21 and Figure 22. It should again be noted that the number of participants is still too small to form reliable conclusions based on the collected data.

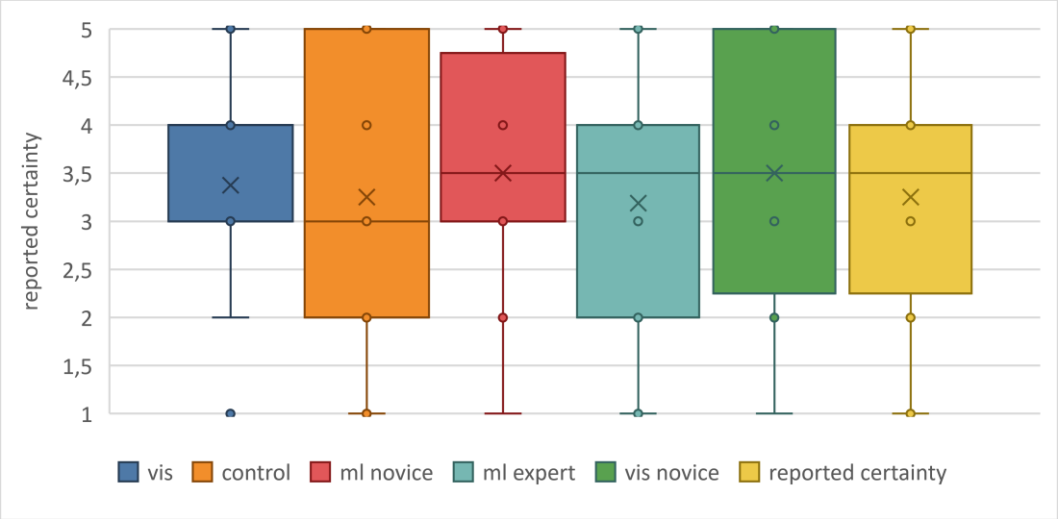

Figure 21: Reported certainty for all decision and behaviour tasks, grouped by condition and expertise.

Figure 21 illustrates the reported certainty for both the decision and behaviour tasks, which does not seem to vary significantly between any of the groups. However, the figure shows a slightly higher reported certainty for the novice groups compared to their expert counterparts. In addition, we can see that the variation is much larger for the control than the vis condition.

In regards to reported helpfulness, Figure 22 shows that the average helpfulness is a bit higher for the vis condition, almost identical for the machine learning novices and experts and higher for novices in visualisation than experts. Overall, visualisation experts seem to report the lowest helpfulness, which may be due to their experience with different kinds of visualisations and knowledge about user experience.

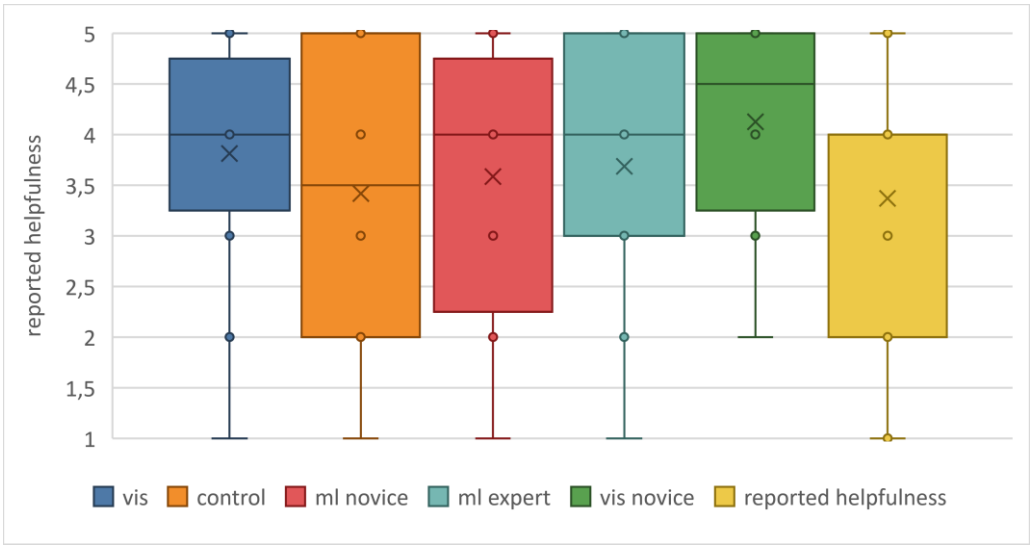

Figure 22: Reported helpfulness for all decision and behaviour tasks, grouped by condition and expertise.

Overall Findings

Since the number of participants is rather small, the results discussed up until this point must be taken with a grain of salt; they are more akin to indicators that point to where the final results may lead, rather than evidence from which conclusions can be drawn. The preliminary results seem to suggest that our visualisations may lead users to more

consistently trust in their decisions and find use in the visualisations. In behaviour tasks, participants seem to form a more complex representation of the model and often mentioned the analysis visualisations as a big impact on their decision.

However, participants in the vis condition took much longer to complete any task and required a longer training period, which most likely stems from the more complex nature of the interface in general. In regards to expertise, current data suggests that novices have greater average certainty in their decisions and explanations. Visualisation novices also report higher helpfulness than their expert counterparts do, while the difference in reported helpfulness for machine learning novices and experts is smaller.

4 Visual support for event/network flow pairs

So far, the efforts related to the visual support for host profiling have been focused on the host behaviour within their network environment. To make the whole process more effective, we have decided to add support for the analysis for the other end, the host behaviour at a process level. Such analysis though presumes a link between the hosts' behaviour exhibited on the network and processes responsible for this activity. Unfortunately, no way of establishing such a link was available and therefore we have developed a method for data correlation of network flows and host events presented in this chapter, along with a visual tool that utilizes this correlation, to provide a bottom-up understanding of the host's activity. Many analysts also expressed the need for such relation between the datasets during our requirements collection phase, as mentioned in D2.3 *"Visualisation requirements"* - state of the art section.

4.1 Data Correlation

The approach to data collection described above introduces a new kind of problem: disjoint datasets that are a result of observing the same behaviour from multiple vantage points – the network probe and endpoint monitoring. Since both of these monitoring solutions employ their metrics, it can be non-trivial to correlate the outputs of OS-level events (as captured by Endpoint Detection & Response solutions) and their corresponding network data, which were incoming/outgoing from/to the host as a result of the network operations performed by the monitored process. Establishing the link between the two is crucial for proper model verification and also provides a lot of insights on its own. In the following section, we will describe how are the data processed and correlated, as well as offer a few opportunities for improvement of the correlation mechanism.

Let us begin with network communication. Our data collection environment captures all traffic from the monitored hosts into full packet capture files (.pcap files) which contain all the network communication from outside the monitored network. While this collection approach makes sure that all of the data get captured, it is somewhat impractical to be deployed in production where full data collection is often impossible due to the sheer volume of day-to-day traffic. That is why we have focused on a more realistic scenario, where only network flows are available to the party performing the analysis. To extract information about network flow from packet capture files, we use an open-source network monitoring tool called Zeek [7] (formerly bro), but any tool capable of extracting the data will do, as long as its output format is similar to those produced by Zeek (see Listing 1).

```
{
	"ts": "1598576168.708831",
	"uid": "CppuPv30KGR6PTNqza",
	"id.orig h": "192.168.16.80",
	"id.orig_p": "64248",
	"id.resp_h": "52.114.36.2",
	"id.resp_p": "443",
	"proto": "tcp",
	"service": "ssl",
	"duration": "1.341277",
	"orig bytes": "2005",
	"resp_bytes": "4474",
	"conn_state": "SF",
	"local_orig": "-",
	"local_resp": "-",
	"missed_bytes": "2637",
	"history": "ShADdcgaFf",
	"orig_pkts": "15",
	"orig_ip_bytes": "2617",
	"resp_pkts": "9",
	"resp_ip_bytes": "2209",
	"tunnel_parents\n": "-",
	"_source": "zeek",
	"_time": 1598576168
}
```


Listing 1: Output of Zeek network flow extraction.

When it comes to the data from the endpoint security monitoring solution, we take verbatim outputs of the F-Secure Rapid Detection and Response cloud-based endpoint monitoring solution. These originate from the monitored hosts and have already gone through several stages of data processing and enrichment inside the F-Secure cloud. While endpoint monitoring produces many different types of events, for our analysis we are interested only in events on the network communication. These are obtained from the event logging system of the underlying operating system. In the case of our environment the Microsoft Windows, but similarly usable events are also produced by other operating systems (see Listing 2).

```
{
  "event": {
    "data": {
      "process_details": {
        "fnam": "%systemroot%\\System32\\svchost.exe",
        "cmdl": "C:\\windows\\System32\\svchost.exe -k utcsvc -p",
        "sha1": "a1385ce20ad79f55df235effd9780c31442aa234",
        "gpid": "p:06ee9782a32ad7b40326a3a5e138470d",
        "onam": "svchost.exe",
        "user": "NT AUTHORITY\\SYSTEM",
        "guserid": "u:d32c80a13d21af5a3f6268db6ae3c1fb",
        "pchain": [
          "p:dc732a8e25b8f92aa10e3b3b4dcc6ef6",
          "p:f90ed52455dbf5385c9b610d4fd77c15"
        ],
        "pid": 1836,
        "exst": false,
        "elev": true,
        "path": "%systemroot%\\System32",
        "name": "svchost.exe"
      },
      "destination_host": {
        "domain name": "v10.events.data.microsoft.com"
      },
      "local_ip": "192.168.16.80",
      "local_port": "64248",
      "remote_ip": "52.114.36.2",
      "remote_port": "443",
      "direction": "out",
      "connection_volume_in": 4474,
      "connection_volume_out": 2005,
      "protocol_number": "6",
      "protocol_keyword": "TCP"
    }
  },
```


Listing 2: A sample of event data produced by the endpoint monitoring solution (some fields were omitted for clarity).

Now the correlation of the two datasets is simple in principle, but in practice, it contains a lot of caveats. The first metric on which we correlate is the IP address of the host, followed by the local port. The former allows for host identification and the latter for process identification. The ports for incoming traffic need to be known to the external hosts communicating with the monitored machine and therefore have a well-known static assignment. For the outgoing traffic, the network stack of the underlying operating system usually assigns the ports at random from a certain range. This assignment usually persists during the entire lifetime of the process and in many cases it is used for a single connection to an external host. Note the words "usually" and "in many cases", because when it comes to port allocation and usage very few assumptions can be made.

The next metric used is the timestamp. Its use mostly solves the problem in which the ports are being reused and allows for differentiation of subsequent network flows, as well as grouping of related network flows. Since the packet capture and event monitoring happens at two different instants their timestamps will differ. Again, no assumptions can be made about the timestamps, not even the order in which the events have occurred, or that the delay in between the vantage points is constant. To remediate this issue we employ a variable sliding window to determine which flows belong to which events. The further apart their timestamps are, the less likely they are to be correlated. The sliding window is user-configurable and its estimation is not automated. Another issue with timestamps was the time zone. While data from the endpoint monitoring does contain timestamps in UTC, the outputs of the network flow extraction use the local time zone. This is however an operational limitation mitigated by a configurable switch, but something to keep in mind when correlating the data nonetheless.

During the correlation itself, the script creates sets of buckets with keys based on the combination of IP address and port. These keys are then used to classify both the flows and the events, with the timestamp sliding window being another constraint the flows in the single bucket must fit into.

During our testing, we have achieved between 65% – 100% data correlation with a median of 85%. Meaning that most of the datasets were correlated 85% of flows to a corresponding host event. The lower rate of correlation was mostly caused by the way endpoint data are collected. When a process communicates over the network, the operating system and therefore the endpoint monitoring solution, logs only the establishment of the connection, no subsequent communications are logged as an event. Thus the number of flows will always be higher than the number of events. The scenario in which a 100% correlation is achieved is when the connections are closed immediately after the data have been transferred, preventing other transfers to take place on the given connection. That is an inherent limitation of the event collection method and therefore there is nothing we can do about it.

4.2 Timeline Visualisation

The data visualisation related to the flow/event correlation provides its user with a unified view of both datasets spread over time. It helps the analyst to understand: the structure of the traffic, order of the events and temporal relations in between them, periods of high and low activity, background process network communication, the domains with which the processes communicated, as well as frequency and duration of the communications.

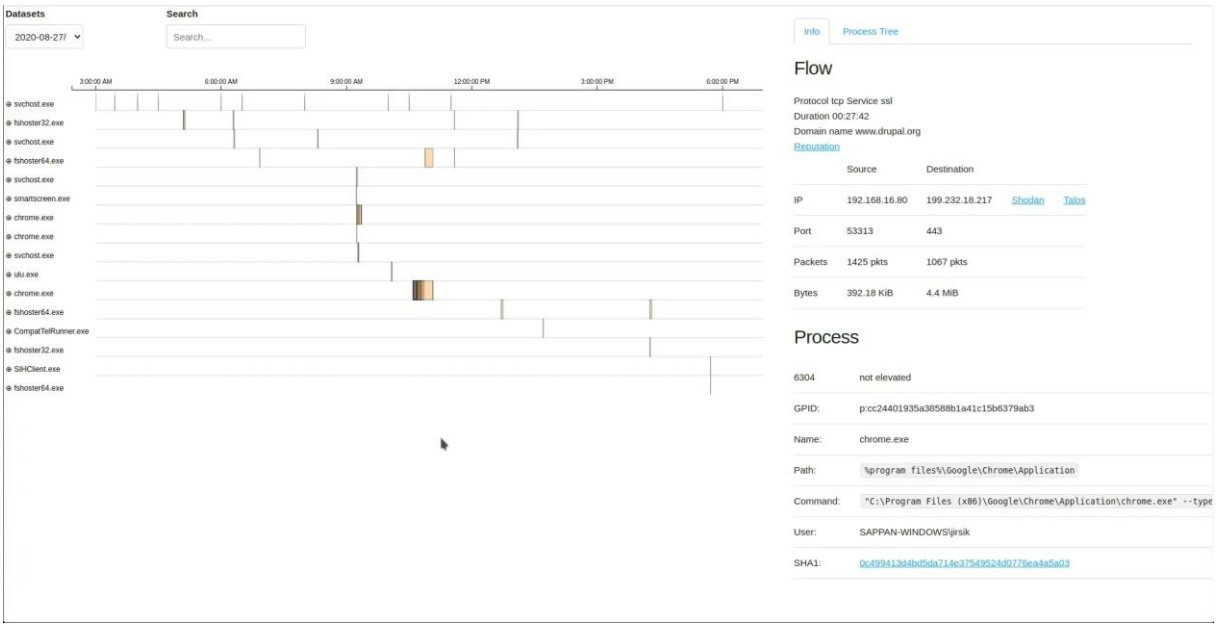

Figure 23: Overview of the timelines visualisation, with the visualisation itself on the left and an info panel on the right.

Many insights that can be derived from the visualisation are closely related to the insights provided by the set of visualisations related to the host profiling, as mentioned in D.3.8. While those allowed for host analysis and verification of models in the context of the entire network subnet, this visualisation provides a more close-up view in the context of the host and its processes. It allows to reason about the changes to the state of the host and its profile with regards to the process behaviour.

To better illustrate one of the possible uses when determining changes to the profiles due to process activity, let us suppose the following scenario: a piece of malware is exfiltrating a large number of sensitive files outside the secured network perimeter. Such activity will reflect itself as a deviation from the usual host profile by having a large number of outgoing flows with variable lengths. Some flows are long enough to trigger an automated detection designed to prevent data exfiltration. While it may be possible to identify the host on which the malicious activity is taking place, determining the process causing the behaviour is not possible without the aforementioned correlation efforts. This visualisation allows to quickly determine the offending process. Providing the domains to which data are being sent, as well as the process details and history, thus providing all the necessary information for the first remediation efforts.

The visual interface consists of three parts: the timeline visualisation, a panel with details about a selected flow/event and a process tree displaying the context in which the process was created. The timeline visualisation has processes on its discrete y-axis and time on its continuous x-axis. The user can freely drag and zoom in to any point of the visualisation. Furthermore, the processes on the y-axis can be expanded to see communication as it occurred on each separate port used by the process. In this expanded view, the domain name of the external party is also visible next to the flow. The colouring of the flow is determined by the type of communication protocol used.

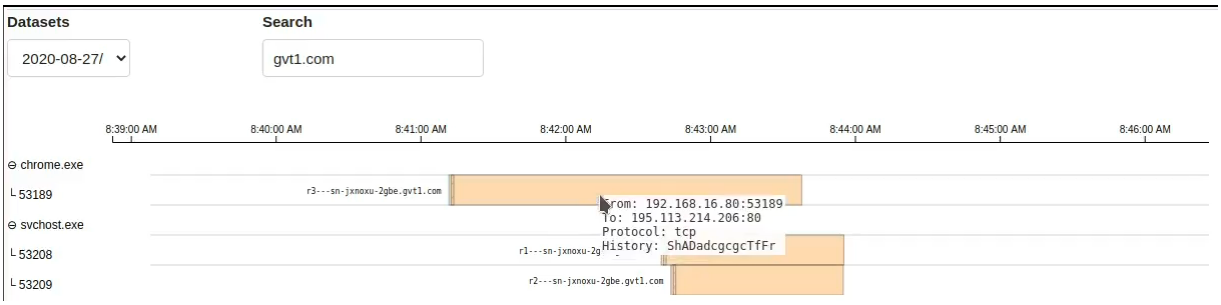

Figure 24: A detail on a specific event/flow pair, displaying the domain and contextual information.

During the analysis of a single capture, the number of events can easily grow into thousands. Our visual tool supports an efficient real-time search of the data. The user can perform partial searches on anything, from IP addresses and domains, through ports and protocol types, to process names and their unique identifiers. A negative search functionality, which excludes the matched terms from the results is also supported, allowing the user to remove well-known and harmless domains like connectivity and update checks.

Upon selecting one of the events the details view is populated with a selection of the attributes about the flow and the process. Aside from providing the basic information at a glance. The details view contains links to external tools that allow the user to check for domain reputation, information about the selected IP address and check if the hash of a selected process is known to be a malware sample. While useful, these functions are provided as a proof of concept for the potential uses when it comes to the integration with external systems.

5 Summary

This deliverable describes the final versions of both visualisation systems previously detailed in deliverable D3.8. We discussed the changes compared to the initial versions and provide an in-depth look into the preliminary results we collected in an online user study to evaluate the approach to support understanding and development of DGA classification models. Although the number of participants is too small for conclusions, the preliminary results point towards the hypothesis that our developed system can provide its users with more certainty and perceived helpfulness for tasks related to understanding and validating deep learning models.

References

[1] B. Davis and M. Glenski, “Measure Utility, Gain Trust: Practical Advice for XAI Researchers,” in *2020 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*, Salt Lake City, UT, USA, 2020.

[2] F. Yang, Z. Huang, J. Scholtz and D. L. Arendt, “How do visual explanations foster end users' appropriate trust in machine learning?,” in *International Conference on Intelligent User Interfaces, Proceedings IUI*, Cagliari, Italy, 2020.

[3] G. Charness, U. Gneezy and M. A. Kuhn, “Experimental methods: Between-subject and within-subject design,” *Journal of Economic Behavior & Organization,* vol. 81, no. 1, pp. 1-8, 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gamfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research,* pp. 2825-2830, 10 November 2011.

[5] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley and M. Gahegan, “Visual Semiotics & Uncertainty Visualization: An Empirical Study,” in *IEEE Transactions on Visualization and Computer Graphics*, 2012.

[6] T. Isenberg, P. Isenberg, J. Chen, M. Seldmair and T. Möller, “A Systematic Review on the Practice of Evaluating Visualization,” in *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[7] V. E. a. c. Paxson, *The Zeek Project.*