Learning Decision Trees from Synthetic Data Models for Human Security Behaviour

Peter Carmichael[1] and Charles Morisset[1]
p.j.carmichael@ncl.ac.uk charles.morisset@ncl.ac.uk

School of Computing, Newcastle University, Newcastle upon Tyne, UK

Abstract. In general, in order to predict the impact of human behaviour on the security of an organisation, one can either build a classifier from actual traces observed within the organisation, or build a formal model, integrating known existing behavioural elements. Whereas the former approach can be costly and time-consuming, and it can be complicated to select the best classifier, it can be equally complicated to select the right parameters for a concrete setting in the latter approach. In this paper, we propose a methodical assessment of decision trees to predict the impact of human behaviour on the security of an organisation, by learning them from different sets of traces generated by a formal probabilistic model we designed. We believe this approach can help a security practitioner understand which features to consider before observing real traces from an organisation, and understand the relationship between the complexity of the behaviour model and the accuracy of the decision tree. In particular, we highlight the impact of the norm and messenger effects, which are well-known influencers, and therefore the crucial importance to capture observations made by the agents. We demonstrate this approach with a case study around tailgating. A key result from this work shows that probabilistic behaviour and influences reduce the effectiveness of decision trees and, importantly, they impact a model differently with regards to error rate, precision and recall.

1 Introduction

Employees of organisations are known to regularly circumvent or bypass security procedures, leading to a relaxed security culture [2]. In order to identify the security culture of an organisation, a security practitioner could collect data from different sources and build a classifier model to predict the security preference of employees. For example, sources such as CCTV, interviews and physical logs (smart card data) can be used to classify employees preferences. There are three main challenges with this - 1) It is costly both in time and financially, as demonstrated by Caufield and Parkin [4]. 2) It is error prone as we rely on humans to interpret human behaviour. 3) Given the dataset, it is difficult to identify which features are relevant to build a classifier model.

To address the three challenges, we propose an assessment of classifier models known as decision trees to predict the impact behavioural elements have on the security culture of an organisation. Firstly, we generate synthetic data from parameterised models with behavioural elements. Secondly, we interpret the data to assess features based on the dataset. Finally, using traditional data mining techniques, we use cross validation to train and test each model independently. One of our results shows different parameters for human behaviour impact the accuracy of decision trees.

Our approach is inspired by online marketing techniques, where peoples behaviour is logged and suggestions are made based on purchases of others who have similar behaviour trends [19]. In the context of security, building a classifier model, an employee performing a security violation should be more substantial to the models rules than the same employee moving between locations. Identifying relevant features, indicates that human behaviour is required, in some form at least. Influences, such as *Social Proof* from Cialdini, or the *Messenger* effect from MINDSPACE can shift the culture of an organisation [7,9]. An employee in the right location at the right time may be influenced by the behaviours of others. If they observe the same action multiple times, or have an influential relationship with the instigator of the behaviour, then they may align their behaviour with this authoritative figure.

We analyse a tailgating case study simulating parameterised models, each model generates a trace we use to assess the accuracy of decision trees. We believe that a security practitioner can understand the relevant features and can begin to understand the relationship between the complexity of human behaviour and the accuracy of decision trees. As a security practitioner, acquiring knowledge where vulnerabilities are present, allows for insights towards defence strategies and interventions. For example, investing in turnstiles to reduce tailgating, or limiting the capabilities of employees who are flagged as a vulnerability.

The main contributions of this work are 1) Parameterised models to simulate and generate synthetic data with known behavioural elements. 2) The methodology for the assessment of decision trees constructed from a synthetic dataset.

The paper is split into the following Sections. In Section 2 we discuss the problem, provide an intuition for how we are approaching it and build on existing literature. In Section 3 we introduce our Multi Agent System (MAS) alongside a scenario focusing on tailgating. In Section 4 we discuss the MAS towards simulation with multiple parameters. In sections 5 and 6 we discuss our assessment methodology and analyse the case study. Section 7 is the conclusion and future work.

2 Problem Formulation

A security practitioner observes employee behaviour in an organisation and accumulates information about security incidents. Using this data, they wish to learn the security preference of employees. One possible solution to this is to use machine learning.

The data generated from observing employees forms a trace, where an entry in a trace describes who did what and when. It is similar to an intrusion detection system, where the logs of what happened are the entries, a collection of logs/entries forms a trace. The problem is, given a trace of interactions, can we use machine learning techniques to correctly identify the security preferences of agents? Let us consider a simple example, where an agents security preference

Table 1: A collection of entries forming a number of violations and preventions for four agents. Each agent is accompanied with a known security preference.

Agent	**Violations**	**Preventions**	**Security Preference**
Alice	4	1	*Usable*
Bob	2	3	*Secure*
Charlie	1	0	*Usable*
Dan	2	5	*Secure*

can be *usable* or *secure*. A *usable* agent is more likely not to follow a policy, whereas a *secure* agent is. Table 1 lists four agents, the number of violations and preventions for a policy and their security preference. Given this data, the Decision Tree in Figure 1 can be formed. It is deterministic and will resolve to a value of *usable* or *secure* dependent on the entry being evaluated. In this case, an entry is the log of violations and preventions for an agent. The resolution/decision returned is the security preference.

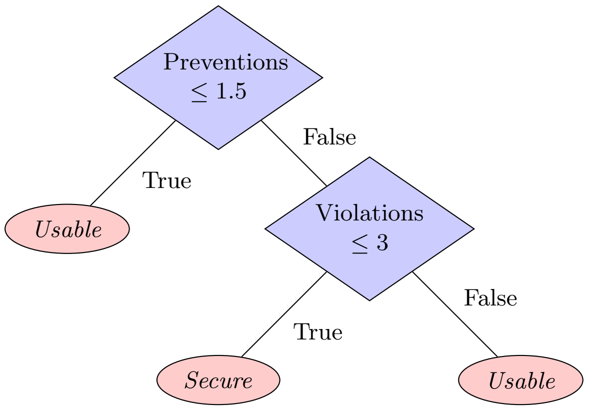

Fig. 1: Learned Decision Tree from the data in Table 1. A diamond is a decision point, an oval is a decision.

This is a small sample, however, an accurate tree has been learned with 100% accuracy for the training data. A decision tree offers predictive power and, given an agent with some information, we wish to predict their security preference. This is complex, as the decision tree from Figure 1 can be easily fooled. For example, an agent with 10 preventions and 5 violations will return as *usable*, even though they have acted securely for the most part.

Establishing which features to consider could provide meaningful results for observing the security culture of employees in an organisation. Unfortunately, the problem is of greater complexity than what we identify here, as human behaviour is complex in itself and leads us to the problem, can we use machine learning to learn from complex behaviour.

2.1 Security Culture Uncertainty

Within an organisation, a security culture exists for how individuals and groups respond to security incidents. Depending on the type of security incident and those involved, it could become a security violation or it could be prevented. We hope that individuals trained to perform tasks are security aware, but we regularly find that they circumvent organisational security policies [3].

Consider working with a company for a short period of time in order to identify the security preference of employees. We could ask them, where responses from interviews have led to popular theories such as the compliance budget [2]. Of course, respondents could lie, answer honestly but not behave consistently, or even fail to acknowledge that their behaviour is insecure.

Even if survey respondents answer honestly, it does not mean that this holds for the future. A secure employee interacting with a usable (non-secure) employee may be influenced towards usable behaviours, creating an insecure culture. Of course, this is bi-directional where secure behaviour can inform more secure behaviour.

From a security officers perspective, they only have so many tools to establish the security culture. For example, they could interview employees, then manually observe them via CCTV recordings to establish if their security preference matches their behaviour [4]. This is of course, costly and time consuming, where we would need to manually record the exact behaviour of each employee.

To add further complexity to the uncertainty of a security culture, some one who is secure may make a judgement of error causing a security incident. For example, Zhu *et. al.* showed they could get more information from people simply by providing them with information up front, exploiting a concept known as reciprocity [20]. They were able to influence people to sacrifice more information than they would usually part with.

The security culture of a company can be changed, for example, via training employees [15]. This behaviour change is one that impacts how people respond to security incidents, for example a recently trained employee may have more awareness for *spear phishing* emails, and is less likely to click suspicious links.

3 Multi Agent System

Behaviour surrounding security policies is dependent on different factors. We see three main elements which impact this security culture physical locations, observations and behaviour change. In this section, we introduce a scenario and provide formal notation to express these three elements as a Multi Agent System.

Scenario: *Agents arrive at the back of a workplace reception and there are two possibilities. Firstly, if nobody is at the front of the reception, the agent must progress to the front of the reception. Secondly, if the newly arrived agent is usable, they will attempt to tailgate. A perfectly secure agent will never attempt to tailgate. If an agent is being tailgated, they can either permit or deny the action, where a permit would allow both agents in to the main building, a deny would force the tailgater to the front of the reception. A perfectly usable agent will always permit tailgating, a secure agent will deny tailgating. The scenario runs for a working week of five days.*

3.1 Location Based Agents

Some security policies rely upon physical locations. For example, tailgating relies locations connected by some entry system such as door, corridor and so on. Furthermore, employees express unique behaviours for moving between multiple locations. In recent work, it has been shown that malicious insider behaviour can be detected by using historical data from a building access control system [6]. The data allows for suitable models to be learned surrounding movement behaviour. For our work, these physical movement models provide validation techniques. Using such techniques, Hidden Markov Models can predict with up to 92% accuracy, the next movement of someone given some historical data [12].

Decisions to enforce or circumvent security policies are individual to each employee. Typically, attitudes towards policies can be impacted by personality, past experiences and a productivity trade-off, to name a few [1]. We call these attitudes a *context*, where an employees *context* informs their decision.

Formally, in a Multi Agent System, the entities are known as agents. From here on, we refer to employees or people as agents. In general, we assume the existence of a set A of agent identifiers.

Definition 1 (Location Based Agents) *Given a set of agents A, a set of locations L, and a set of contexts C, we define the set of* location based agents *as $LBA = A \times C \times L$.*

Definition 2 (Location based actions) *Given an agent a and two locations l and l', a location based action is defined as $\mathsf{m}(a, l, l')$, indicating that a moved from l to l'. A location based action does not modify the context of the agent.*

In general, there could be many different ways to capture the concrete set of location based actions in a system, for instance by going through the actual logs of a smart card system. For the sake of simplicity, we consider here a set of links $Link \subseteq L \times L$ where (l_1, l_2) indicates a physical link between the location l_1 and l_2. Intuitively speaking, any agent can move from a location to another as long as there is a link between them. We characterise this with the following rule:

$$\frac{(l, l') \in Link}{(a, c, l) \xrightarrow{\mathsf{m}(a,l,l')} (a, c, l')} \tag{1}$$

The action 1 is defined as an inference rule at the atomic level expressing that one agent moves from one location to another, provided the two locations are connected. In general, we write $(a_1, c_1, l_1) \mid \ldots (a_n, c_n, l_n)$ for a set of location based agents. Given a set of agents $LA \subseteq LBA$, we can extend the rule above as follow

$$\frac{(a, c, l) \xrightarrow{\mathsf{m}(a,l,l')} (a, c, l')}{(a, c, l) \mid LA \xrightarrow{\mathsf{m}(a,l,l')} (a, c, l') \mid LA} \tag{2}$$

The tailgating scenario is a policy breach for an organisation. Whilst it is not the case that an agent is just *usable* or just *secure*, for now, we use these polar opposites to express our model.

We introduce two actions, $\mathsf{tgp}(a_1, a_2, l, l')$ and $\mathsf{tgd}(a_1, a_2, l, l')$, indicating that a_1 and a_2 tailgated, and that a_2 denied a tailgate from a_1, respectively. Intuitively speaking, a *usable* agent is permitted as they tailgate a *usable* agent. Secondly, a *usable* agent is denied as they tailgate a *secure* agent. Formally:

$$\frac{(a_1, c_1, l) \xrightarrow{\mathsf{m}(a_1,l,l')} (a_1, c_1, l') \quad (a_2, c_2, l) \xrightarrow{\mathsf{m}(a_2,l,l')} (a_2, c_2, l') \quad c_1 = c_2 = \mathit{usable}}{(a_1, c_1, l), (a_2, c_2, l) \mid LA \xrightarrow{\mathsf{tgp}(a_1,a_2,l,l')} (a_1, c_1, l'), (a_2, c_2, l') \mid LA} \tag{3}$$

$$\frac{(a_2, c_2, l) \xrightarrow{\mathsf{m}(a_2,l,l')} (a_2, c_2, l') \quad c_1 = \mathit{usable} \quad c_2 = \mathit{secure}}{(a_1, c_1, l), (a_2, c_2, l) \mid LA \xrightarrow{\mathsf{tgd}(a_1,a_2,l,l')} (a_1, c_1, l), (a_2, c_2, l') \mid LA} \tag{4}$$

The rules introduced so far explain how agents with a context move between locations. They can either just move, or they can tailgate and be either permitted or denied. A permit moves them into the tailgated location, the denied leaves an agent in the same location before the action occurred.

3.2 Observing Agents

In an organisation, when an action happens, people may notice. In our example, this translates to agents observing security policies being enforced or exploited. Much like in the workplace, if someone is challenged for tailgating, people within close proximity will notice. One report recorded that users security sensitivity for other peoples security behaviour was prominent in a working environment [8]. We know that people have security awareness in the workplace, particularly when policies are regularly followed.

We began with Location Based Agents, where the interactions of agents are restricted by physical locations and the security preference of each agent. We extend this notion and introduce Observing Agents:

Definition 3 (Observing Agents) *We define $OA \subseteq A \times C \times L \times \mathbb{P}(\Theta)$ as the set of observing agents where A is the set of agents, C is the set of contexts, L is the set of Locations and $\Theta \subseteq A \times Act_\theta$. We introduce a set of observable Actions Act_θ, where any act $\in Act_\theta$ can be observed by an agent.*

Let us consider that $Act_\theta = \{permit, deny\}$, which refer to the permitting or denying of tailgating. An Observing Agent during the course of their interactions, may accumulate observations of other agents permitting or denying tailgating. As it is currently defined, an agent can only observe and store one observation for each agent. At the atomic level, an inference rule for an observation would take the form:

$$\frac{}{(a, c, l, \Theta_a) \xrightarrow{\mathsf{obs}(\theta)} (a, c, l, \Theta'_a \cup \theta)} \tag{5}$$

Intuitively, an agent a_1 with the action $\mathsf{obs}(a_2, permit)$ would indicate that they have observed a_2 permitting tailgating.

For an agent observing a particular act, such as tailgating, we must consider all agents in the observable area. We provide the following definition to make use of earlier rules:

$$loc : OA \to LA$$

$$loc(a, c, l, \theta_a) = (a, c, l)$$

Therefore, the observable actions for tailgating are as follows:

$$\frac{\dfrac{\begin{array}{c}(a_1, c_1, l, \Theta_{a_1}) \xrightarrow{\mathsf{obs}(permit)} (a, c, l, \Theta_{a_1} \cup permit) \\ \forall (a', c', l, \Theta'_a) \in OA \Rightarrow (a', c', l, \Theta'_a) \xrightarrow{\mathsf{obs}(permit)} (a', c', l, \Theta'_a \cup permit)\end{array}}{\begin{array}{r}(a_1, c_1, l), (a_2, c_2, l) | loc(OA) \xrightarrow{\mathsf{tgp}(a_1, a_2, l, l')} \\ (a_1, c_1, l'), (a_2, c_2, l') | loc(OA)\end{array}}}{\begin{array}{r}(a_1, c_1, l, \theta_{a_1}), (a_2, c_2, l, \theta_{a_2}) | OA \xrightarrow{\mathsf{p}(a_1, a_2, l, l', permit)} \\ (a_1, c_1, l', \theta_{a_1}), (a_2, c_2, l', \theta_{a_2}) | OA'\end{array}} \tag{6}$$

$$\frac{\dfrac{\begin{array}{c}(a_1, c_1, l, \Theta_{a_1}) \xrightarrow{\mathsf{obs}(deny)} (a, c, l, \Theta_{a_1} \cup deny) \\ \forall (a', c', l, \Theta'_a) \in OA \Rightarrow (a', c', l, \Theta'_a) \xrightarrow{\mathsf{obs}(deny)} (a', c', l, \Theta'_a \cup deny)\end{array}}{\begin{array}{r}(a_1, c_1, l), (a_2, c_2, l) | loc(OA) \xrightarrow{\mathsf{tgd}(a_1, a_2, l, l')} \\ (a_1, c_1, l), (a_2, c_2, l') | loc(OA)\end{array}}}{\begin{array}{r}(a_1, c_1, l, \theta_{a_1}), (a_2, c_2, l, \theta_{a_2}) | OA \xrightarrow{\mathsf{d}(a_1, a_2, l, l', deny)} \\ (a_1, c_1, l, \theta_{a_1}), (a_2, c_2, l', \theta_{a_2}) | OA'\end{array}} \tag{7}$$

3.3 Behaviour Change Agents

The concept of behaviour change as a body of research contains many different models. Not all of the models are fit for our purpose. Howver, the COM-B model splits behaviour change into three elements; Capabilities, Opportunities and Motivation [13]. In this paper, we focus on the aspect of Motivation, which can be changed by influencing effects. Such effects as *Messenger* and *Social Norms* are of interest to us [10]. The former relates to those people/agents we perceive to be in a position of authority, the latter is all about those people around us in our immediate vicinity [7].

In our MAS, a behaviour change would be a change of context. A *secure* agent can become *usable* and vice versa. The following rule captures behaviour change for security preferences:

$$\frac{c \neq c'}{(a, c, l, \Theta_a) \xrightarrow{\mathsf{bchange}(c')} (a, c', l, \{\})} \tag{8}$$

Whilst we don't validate the notion of behaviour change in this work, we do hope to establish meaningful results at a later point. For example, observing and interviewing users in a social experiment where a security preference is present is one possible stream to substantiate the synthetic inference rules defined.

To the best of our knowledge, for the influencing effects *Messenger* and *Social Norms* there does not exist a strategy to quantify formally these effects. Unsurprisingly, an effect is unique to each agent. As the purpose of this work is to assess the effectiveness of decision trees, we introduce the following rules which capture effects as identical for all agents:

Definition 4 (Influencing Agents) *The set $IA \subseteq A \times A$ captures Influencing Agents, where any $(a, a') \in IA$ indicates that a' can influence a and $a \neq a'$.*

The influencing agents is for the purpose of defining inference rules for the *Messenger* effect. The following rules capture this:

$$\frac{(a, c, l, \Theta_a) \xrightarrow{\mathsf{bchange}(usable)} (a, c', l, \Theta_a) \ (\exists(a', permit) \in \Theta_a \wedge (a, a') \in IA)}{(a, c, l, \Theta_a)|IA \xrightarrow{\mathsf{messP}(usable)} (a, c', l, \Theta_a)|IA} \tag{9}$$

$$\frac{(a, c, l, \Theta_a) \xrightarrow{\mathsf{bchange}(secure)} (a, c', l, \Theta_a) \ (\exists(a', deny) \in \Theta_a \wedge (a, a') \in IA)}{(a, c, l, \Theta_a)|IA \xrightarrow{\mathsf{messD}(secure)} (a, c', l, \Theta_a)|IA} \tag{10}$$

For *Social Norms*, we care about the number of agents that have been observed for a particular action. Given a set of observations, we can establish how many agents have been observed performing a particular action. The importance of this work is not the formal definitions of influences, but rather the accuracy of decisions trees when these effects are in place. As such, we don't provide the notation for the *Social Norms* effect.

4 Multi Agent System - Simulation

For clarification, we acknowledge that at times, certain rules within the system can be executed synchronously. For example, it is possible that a behaviour change rule and an observing rule can occur synchronously. For now, we consider that a behaviour change rule takes priority. In the case of two similar rules conflicting with each other, we let the simulation tool use a random number generator to address this. We hope to improve this in later work.

4.1 Model Parameters

For the simulation, we reflect behaviours and attributes that we know exist. We understand that, whilst we will not have a model that truly reflects human behaviour, we at least can parameterise concepts that we know exist from the literature. Our parameters are as follows:

- p_1: Expected Arrival Rate - Agents arrive stochastically to the workplace reception, the arrival rate follows a normal distribution, agents can arrive at any point within some bounds. For example, if a start time for work is 9AM, we might expect some agents to turn up early, just before, just after, late or precisely on time.
- p_2: Probabilistic Decision - Assumptions have been made towards individuals as being *homo economicus*, where we make decisions based on personal gain or internal heuristics for guiding behaviour which look to maximise some reward [11]. Additionally, each day, experience is slightly different and for an agent, this could be the difference between a *secure* agent acting *usable* and vice versa, which is what we capture with our probabilistic decision, the ability for agents to act against their security preference [16].
- p_3: Social Norms Influence - Social proof, where individuals assume the actions of those they have observed in order to reflect the interpreted cultural norms is apparent in many societies [14].
- p_4: Messenger Influence - *Authority*, is influencing by social/professional status, those we perceive to be in a position of power or responsibility can influence our behaviours [9,7].
- p_5: Personality - Different personalities react differently to the same influences. We implement personality traits for agents, where different personalities are subject to different influences:
  1. *Conscientiousness* - influenced by *Messenger*.
  2. *Agreeableness* - influenced by *Social Norms* and *Messenger*.
  3. *Extroversion* - influenced by *Social Norms* [18].

In a model, there is a distinct set of actions and observations recorded for agents. A trace generated from a model records agents moving between locations, tailgating being permitted or denied, agents successfully tailgating, agents failing to tailgate and agents observing tailgating being permitted or denied. This is all public information, the private information, such as the model parameters and agent contexts are not in a trace. A trace does not contain information such as if an agent can be influenced by *Messenger* or *Social Norms*. We do this to see if decision trees can determine the underlying rules for the different parameterised models.

5 Assessment Methodology

We define a model with a set of parameters such that each model contains a different set of parameters. However, the initial state of each model is identical in terms of agent context and agent location. We then run a number of simulations for each model and generate a trace for each simulation run. A model will therefore, be associated with many traces.

A trace contains the number of entries equal to the number of agents, where an entry contains all of the features for an agent and is accompanied by the final security preference of that agent. The features of agents are the number of violations, preventions, attempts at tailgating and the number of times an agent is in close proximity when tailgating between other agents occurs.

Given all of the traces for a model, we use cross validation to construct and assess the accuracy of using decision trees for each model. The cross validation consists of a training and testing phase, where the training is inclusive for the security preference of an agent and the testing phase is exclusive of the security preference.

A prediction from a decision tree is either *usable* or *secure*. If we consider secure as our target value then a *true-positive* (tp) is a correct prediction for *secure*, *true-negative* (tn) is a correct prediction for *usable*. *False-positive* (fp) is an incorrect prediction for *secure* and *false-negative* (fn) is an incorrect prediction for *usable*. From these we can calculate the error rate, precision and recall:

$$err(fp, fn, tp, tn) = \frac{fp + fn}{tp + tn + fp + fn} \tag{11}$$

$$(tp, fp) = \frac{tp}{tp + fp} \quad r(tp, fn) = \frac{tp}{tp + fn} \tag{12}$$

The cross validation creates a number of decision trees for each parametrised model. Given a set of Decision Trees D where a set of testing traces T are present. A testing trace contains a set of entries E excluding the security preference, where a function $f : D \times E \to O$ takes a decision tree, an entry and returns an outcome O where $O = \{fp_o, fn_o, tp_o, tn_o\}$.

$$g : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times D \times \mathbb{P}(E) \to \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$$

$$g(i, j, k, l, d, E) = \begin{cases} g((i+1), j, k, l, d, (E \setminus e)), & \exists e \in E \text{ where } f(d, e) = fp_0 \\ g(i, (j+1), k, l, d, (E \setminus e)), & \exists e \in E \text{ where } f(d, e) = fn_0 \\ g(i, j, (k+1), l, d, (E \setminus e)), & \exists e \in E \text{ where } f(d, e) = tp_0 \\ g(, j, k, (l+1), d, (E \setminus e)), & \exists e \in E \text{ where } f(d, e) = tn_0 \\ (i, j, k, l) & \text{otherwise} \end{cases} \tag{13}$$

Using the function g we can calculate the number of different types of outcomes a decision tree produces. We can then use the function *calc* to assess the accuracy of a decision tree:

$$calc : D \times \mathbb{P}(P) \to [0, 1]$$
$$calc(d, E) = err(g(0, 0, 0, 0, d, E)) \tag{14}$$

Once we can calculate the error rate for one decision tree, we can then assess the accuracy of all the decision trees generated for a particular model:

$$\mu_{error}(D, E) = \frac{\sum_{d \in D} calc(d, E)}{|D|} \tag{15}$$

We do the same for the precision and recall of the decision trees, however, we do not provide the notation for this. Each model is associated with a set of decision trees. Therefore, for each model we can calculate μ_{error} to identify the accuracy of decision trees for a given set of parameters.

6 Analysis - Case Study

In this section we discuss the use of parametrised models and make remarks surrounding the results for three different cases.

The number of possible parametrised models is 2^5, we only consider 11 of these 32. The expected arrival rate is included in the majority of the parameterised models, as we do not consider too many models where all agents always arrive at the exact same time, of course this could happen, but it is very unlikely. The personality parameter is dependent upon a behaviour change parameter being present, therefore, it does not add to a model if *Social Norms* and/or the *Messenger* parameters are not included.

We used the *Julia* programming language to implement our case study and made use of the SysModels package [5,17]. We generated synthetic data on a Toshiba laptop with a 2.4 GHz i5 processor and 8GB RAM. To generate the data for 11 models with 200 agents it took 22 minutes which is roughly 2 minutes per model. Each model is generated with 10 traces each starting from an identical initial state for each model.

For the analysis we performed four test cases and used 50, 100 and 200 agents. Table 2 is the results for the 100 agents, the results for 50 and 200 agents are in the Appendix in Tables 3 and 4 respectively.

For each test case, we calculated the average error rate, the standard deviation, the precision and the recall of each parameterised model, where Table 2 shows the parameters for each model. We now make remarks regarding the results we have obtained.

Remark 1 *The average error rate for model m_1 is significantly more accurate with 50 than 100 or 200 agents.*

With regards to Remark 1, as the expected arrival time is not set, all agents arrive at the same time. The majority of agents don't ever permit, deny or attempt to tailgate, therefore, a decision tree will make inaccurate predictions for some agents, particularly when more than 50 agents are used.

Remark 2 *If the probabilistic parameter is set, then the average error rate significantly increases. In particular, it impacts more than both the Messenger and Social Norms parameters.*

Table 2: 100 Agents: p_1: Expected Arrival Rate; p_2: Probabilistic Decision; p_3: Norms Influence (Social Proof); p_4: Messenger Influence; p_5: Personality; $\mu(error)$: Average error rate of a model; $pr(s)$: The precision of the model towards *secure*; $r(s)$: The recall of the model for *secure*;

p_1	p_2	p_3	p_4	p_5	**Model**	$\mu(error)$	$\sigma(error)$	$pr(s)$	$r(s)$
					m_1	0.255	0.067	0.659	0.830
✓					m_2	0.001	0.002	1	0.997
✓	✓				m_3	0.234	0.028	0.697	0.712
✓		✓			m_4	0.073	0.019	0.963	0.953
✓			✓		m_5	0.160	0.024	0.884	0.898
✓		✓		✓	m_6	0.094	0.018	0.928	0.938
✓			✓	✓	m_7	0.114	0.016	0.904	0.910
✓	✓	✓			m_8	0.271	0.024	0.724	0.731
✓	✓		✓		m_9	0.367	0.031	0.634	0.624
✓		✓	✓	✓	m_{10}	0.027	0.012	0.975	0.969
✓	✓	✓	✓	✓	m_{11}	0.277	0.028	0.675	0.675

The use of the probabilistic parameter significantly increases the average error rate of the decision trees. Due to the uncertainty of agent behaviour, i.e. *secure* agents acting *usable* and vice versa, a *secure* agent could have always behaved as *usable*. A classifier model would always conclude that they are *usable* when they are in fact *secure*. Whilst Remark 2 is not surprising, the impact of uncertain behaviour against social influences is a useful result for a security practitioner. In the real world, some people will always be *secure* or *usable*, some hover between the two and some are slightly more *secure* or slightly more *usable*, some insight towards these numbers would allow us to calculate the impact of agents towards a model.

Remark 3 *The Messenger influence has a slightly more of an impact to the error rate, precision and recall of a model than Social Norms. It is true for all four of the test cases. They both impact the error rate, precision and recall of every model.*

The influences themselves differ in how they are implemented. The *Messenger* relies on an agent observing a behaviour of another agent that they consider to be an authoritative figure. The *Social Norms* is a cumulative influence, where the number of observations of a particular action can trigger the security preference of an agent to change. For Remark 3, the interest is that they are not probabilistic behaviours, they are private behaviours.

We have defined very simple rules for our influences. We wish to know if decision trees are capable of generating rules to deal with these simple behaviour changes. Given the data for our number of agents. We can see a slight improvement when 200 agents are present. However, the decision trees still perform poorly for these basic implementations of influences.

Remark 4 *On average, the models for 200 agents are more accurate than 50 and 100 agents.*

A trend emerged for the accuracy of models as we increased the number of agents. Whilst some of the models were more accurate for 50 agents, in general Remark 4 holds, in particular for the complex models where influences and probabilistic decisions are present. This is due to an increase number of entries to train decision trees, improving its accuracy.

Overall, we can see that with some basic aspects of human behaviour such as an uncertainty of decisions between *secure* and *usable* agents, the decision trees perform poorly. Even more so that we are just considering the polar opposites for security preferences. Whilst the influencing effects implemented are relatively simple, we believe as they increase in complexity, i.e. become heterogeneous for each agent, this would reduce the accuracy of decision trees even more. On another note, and mainly due to processing limitations, it's not clear if the accuracy can be improved by generating thousands of traces to use in the cross validation analysis.

7 Conclusion

In this paper we designed a multi agent system to generate synthetic data. Secondly, we identified features from the synthetic data. Finally, using cross validation we trained and tested many different decision trees for four test cases.

The generated decision trees showed that as the complexity of human behaviour increases, the less accurate decision trees are for predicting attributes, in this case, the security preference of employees. The remarks from Section 6 highlight the important features of the models with regards to the parameters. For example, probabilistic decisions impact the model significantly more than influences with regards to the error rate, precision and recall. Between the influences, the *Messenger* influence had a greater impact, however, this is partially down to how a security practitioner or designer implements behaviour change.

The insights towards the impact of these different parameters allows for an understanding between the limitations of decision trees and predicting security preferences. In particular, the certainty in the accuracy of the decision trees.

The three elements of the MAS allow for the bigger problem to be broken down into manageable chunks towards validation. For example, focusing on how we observe and formalising this notion will further support the work carried out here. Interviews to evaluate how our behaviour changes, again, will improve the MAS presented in this work.

The future of this work will target the unanswered questions that we can draw from this paper. Providing validation for those three elements of the multi agent system which are the physical locations, observations and behaviour change. Calculating the impact the parameters have towards error rate, precision and recall would allow a security practitioner to identify when probabilistic agents, influences or any other behaviours are present without having prior knowledge as we did. The techniques for building classifier models will be explored, for example, by considering different algorithms for building classifier trees, or sampling a range of features to assess the importance of each feature.

References

1. S. Bartsch and M. A. Sasse. How users bypass access control and why: the impact of authorization problems on individuals and the organization. 2012.
2. A. Beautement, M. A. Sasse, and M. Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 workshop on New security paradigms*, pages 47–58. ACM, 2009.
3. J. Blythe, R. Koppel, and S. W. Smith. Circumvention of security: Good users do bad things. *IEEE Security & Privacy*, 11(5):80–83, 2013.
4. T. Caufield and S. Parkin. Case study: Predicting the impact of a physical access control intervention. In *STAST (Socio-Technical Aspects of Security and Trust)*. In publication., 2016.
5. T. Caulfield and D. Pym. Improving security policy decisions with models. *IEEE Security & Privacy*, 13(5):34–41, 2015.
6. C. Cheh, B. Chen, W. G. Temple, and W. H. Sanders. Data-driven model-based detection of malicious insiders via physical access logs. In *International Conference on Quantitative Evaluation of Systems*, pages 275–291. Springer, 2017.
7. R. B. Cialdini and N. Garde. *Influence*, volume 3. A. Michel, 1987.
8. S. Das, T. H.-J. Kim, L. A. Dabbish, and J. I. Hong. The effect of social influence on security sensitivity. In *Proc. SOUPS*, volume 14, 2014.
9. P. Dolan, M. Hallsworth, D. Halpern, D. King, R. Metcalfe, and I. Vlaev. Influencing behaviour: The mindspace way. *Journal of Economic Psychology*, 33(1):264–277, 2012.
10. P. Dolan, M. Hallsworth, D. Halpern, D. King, and I. Vlaev. Mindspace: influencing behaviour for public policy. 2010.
11. R. H. Frank. If homo economicus could choose his own utility function, would he want one with a conscience? *The American Economic Review*, pages 593–604, 1987.
12. A. Gellert and L. Vintan. Person movement prediction using hidden markov models. *Studies in Informatics and control*, 15(1):17, 2006.
13. S. Michie, M. M. van Stralen, and R. West. The behaviour change wheel: a new method for characterising and designing behaviour change interventions. *Implementation science*, 6(1):42, 2011.
14. H. Rao, H. R. Greve, and G. F. Davis. Fool's gold: Social proof in the initiation and abandonment of coverage by wall street analysts. *Administrative Science Quarterly*, 46(3):502–526, 2001.
15. R. S. Shaw, C. C. Chen, A. L. Harris, and H.-J. Huang. The impact of information richness on information security awareness training effectiveness. *Computers & Education*, 52(1):92–100, 2009.
16. R. H. Thaler. From homo economicus to homo sapiens. *The Journal of Economic Perspectives*, 14(1):133–141, 2000.
17. Tristanc. Sysmodels package. `https://github.com/tristanc/SysModels`, Feb 2017. [Online; accessed 08-June-2017].
18. S. Uebelacker and S. Quiel. The social engineering personality framework. In *Socio-Technical Aspects in Security and Trust (STAST), 2014 Workshop on*, pages 24–30. IEEE, 2014.

19. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2016.
20. F. Zhu, S. Carpenter, A. Kulkarni, and S. Kolimi. Reciprocity attacks. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 9. ACM, 2011.

Appendix A Additional Results

Table 3: 50 Agents; See Table 2 for column definitions.

Model	$\mu(error)$	$\sigma(error)$	$pr(s)$	$r(s)$
m_1	0.070	0.037	0.896	0.943
m_2	0.018	0.010	0.974	0.980
m_3	0.279	0.035	0.658	0.642
m_4	0.050	0.025	0.947	0.950
m_5	0.162	0.029	0.853	0.867
m_6	0.031	0.018	0.955	0.977
m_7	0.091	0.030	0.893	0.937
m_8	0.266	0.039	0.701	0.686
m_9	0.365	0.051	0.694	0.656
m_{10}	0.017	0.012	0.976	0.986
m_{11}	0.325	0.056	0.622	0.581

Table 4: 200 Agents; See Table 2 for column definitions.

Model	$\mu(error)$	$\sigma(error)$	$pr(s)$	$r(s)$
m_1	0.201	0.135	0.861	0.912
m_2	0.006	0.003	0.996	0.998
m_3	0.170	0.013	0.910	0.888
m_4	0.014	0.006	0.993	0.993
m_5	0.050	0.009	0.976	0.972
m_6	0.024	0.007	0.984	0.990
m_7	0.047	0.011	0.976	0.973
m_8	0.140	0.021	0.933	0.912
m_9	0.277	0.029	0.833	0.812
m_{10}	0.040	0.008	0.975	0.980
m_{11}	0.161	0.016	0.920	0.892