A New Approach of Formulating Stereotypical Trust in Multi-Agent Systems

Romas Bieliauskas

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Master of Arts
of the
University of Aberdeen.

Department of Computing Science

2015

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: 2015

Abstract

In this paper we will be focusing on automated systems composed of multiple knowledgeable agents, able to make decisions on their own, within an environment, called *Multi-Agent Systems* (MAS). We will define what trust is between agents and the benefit of trust within a MAS by talking about different types of trust evaluation models proposed by previous researchers. Furthermore an explanation of how stereotyping can be used to help agents interact among themselves to achieve their goals will be discussed. As stereotyping offers significant improvement under certain conditions, we propose a different approach of stereotyping agents to, perhaps, allowing agents to choose a successful partner for an interaction but also know which agent is good at what task. We will focus in dept on the work of Burnett's, C., Norman's, T. J., and Sycara's, K. research [8] called, *Stereotypical Trust and Bias in Dynamic Multi-agent Systems*. We will work on their proposed simulation to implement our conceptional model of an alternative stereotype. The proposed concept stereotype will focus on connecting a particular agent with a desired task. We will test and observe if there are correlations between tasks and agent features and provide results if the concept model is capable of producing a stereotypes with an M5 Learning Decision tree. Furthermore, we will prove that our proposed concept model is capable of recognizing behavioral differences between tasks and is able to generalize to situations to different tasks in a domain. This will allow the resources needed for the completion of the task to be known and a prediction will be given based on what resources the agent contains. Furthermore, the stereotype will be able to specify at what particular task a specific agent can complete successfully and unsuccessfully. We will provide a number of future work for our proposed alternative stereotyping concept.

Acknowledgements

I would like to extend my thanks towards my honours project tutor Prof Timothy J. F. Norman for his help, detailed and insightful comments to improve this paper along with introducing me with LaTeX in a class I was a part of previous semester. I would like to thank Dr. Christopher Burnett for his help, patience and time. I would also like to thank University of Aberdeen for having such a large amount of sources accessible for students, making research that much easier.

Contents

1 Introduction **8**

1.1 Overview 8

1.2 Connection With the Real World 8

1.3 Motivation 9

1.4 The Need for Trust in Open Multi-Agent Systems 9

1.4.1 Agent's Selfishness 9

1.4.2 Dealing With Selfishness 9

1.4.3 Defining Trust 10

2 Background **12**

2.1 Trust Within Multi-Agent Systems 12

2.2 Trust Models in Multi-Agent Systems 13

2.3 Direct Trust Evaluation Models 13

2.4 Indirect/Reputation-based Trust Evaluation Models 14

2.4.1 Trust Evidence Aggregation Approaches 15

2.4.2 Testimony Filtering Approaches 16

2.5 Organizational Trust Evaluation Models 17

2.6 Socio-Cognitive Trust Evaluation Models 18

3 Framework **20**

3.1 Learning Decision Trees 20

3.2 Defining Variables 21

3.3 Stereotypical Trust Evaluation Framework 21

3.3.1 Agents and Tasks 21

3.3.2 Opinion Representation 22

3.3.3 Decision Model 25

3.4 Alternative Stereotyping Framework 25

3.4.1 Tasks and Agents 25

3.4.2 Generating Predictions 26

3.4.3 Delegation Model 26

3.4.4 Framework Summary 27

4 Related Work **28**
4.1 Stereotypical Trust . . . 28
4.2 Abstract . . . 29
4.3 Framework Summary . . . 29
4.3.1 Figure 4.1 Description . . . 29
4.4 Stereotype Model . . . 30
4.4.1 Stereotypical Reputation . . . 30
4.4.2 Learning Stereotypes . . . 31
4.4.3 Two-Phase Learning . . . 32
4.4.4 Model Tree Learning . . . 32
4.5 Stereotypical Bias . . . 33
4.6 Findings . . . 34
4.6.1 Experiment Layout . . . 34
4.6.2 Research Conclusion . . . 35

5 Conceptional Model **36**
5.1 Objective . . . 36
5.2 Requirements . . . 38
5.2.1 Functional . . . 38
5.2.2 Non-Functional . . . 38
5.3 Design . . . 38
5.4 Methodology . . . 41
5.4.1 Alternatives . . . 42
5.4.2 Problems and Difficulties . . . 42

6 Evaluation **43**
6.1 Overcoming Problems and Difficulties . . . 43
6.2 Testing Description . . . 43
6.3 Results . . . 45
6.4 Hypothesis Evaluation . . . 50

7 Conclusion **51**
7.1 Discussion . . . 51
7.2 Conclusion . . . 51
7.3 Future Work . . . 52

A Maintenance Manual **59**
A.1 Software Requirements . . . 59
A.2 List of Packages Needed . . . 59
A.3 Installation Instructions . . . 59
A.3.1 Using IDE . . . 59
A.4 Future Adaptations . . . 60

B User Manual **61**
B.1 Usage Instructions 61
B.1.1 Code Modification 77

Chapter 1

Introduction

In this chapter we provide the reader with an overview of the paper, along with motivation. A minor introduction of trust and multi-agent systems is provided for a better understanding of the topic.

1.1 Overview

In this paper we will be focusing on automated systems composed of multiple knowledgeable agents, able to make decisions on their own, within an environment, called *Multi-Agent Systems* (MAS). We will define what trust is between agents and the benefit of trust within a MAS by talking about different types of trust evaluation models proposed by previous researchers. Furthermore an explanation of how stereotyping can be used to help agents interact among themselves to achieve their goals will be discussed. As stereotyping offers significant improvement under certain conditions, we propose a different approach of stereotyping agents to, perhaps, allowing agents to choose a successful partner for an interaction but also know which agent is good at what task. We will focus in dept on the work of Burnett's, C., Norman's, T. J., and Sycara's, K. research [8] called, *Stereotypical Trust and Bias in Dynamic Multiagent Systems*. We will work on their proposed simulation to implement our conceptional model of an alternative stereotype. The proposed stereotype will focus on connecting a particular agent with a desired task. This will allow the resources needed for the completion of the task to be known and a prediction will be given based on what resources the agent contains. Furthermore, the stereotype will be able to specify at what particular task a specific agent is good at. The results will be analyzed and discussed along with proposing future work for this paper.

1.2 Connection With the Real World

In today's world, one may notice a lot of day-to-day activities that are associated with technology. These computer applications are often open distributed systems with a decentralized control regime, and constant change is present throughout the systems lifetime. Commercial transactions conducted electronically over the Internet, called *e-commerce* where MAS architecture can be used as the e-commerce design [1]. As telecommunication networks are growing, so does their complexity forcing management systems to work with incomplete data and work with uncertain situations in dynamic environments [10] proposed a *Fault Diagnosis Multi-Agent System* for a telecommunication company in Czech Republic called *Telef'onica*. As one may notice, MAS involvement in the real world has no limits. In [9] talks about the success of a production company

called *Just In Time and Lean Production* A Mas was presented that would level the workload of the workstations, production smoothing techniques, shorter manufacturing and lowered the work in process of the company. Almost any human interaction can be simulated in a MAS, demonstrating behavioral examples or simulating problems and their solutions.

1.3 Motivation

The capabilities and fascinating artificial intelligence aspect of MAS has always inspired. The fact that MAS can not only be a simulation of artificial agents interacting, but a problem solver and be applied to any field for any person or company has its opportunities. Working for a large company in the future, or perhaps even owning one, would give a lot of potential of containing knowledge about MAS. Thus improving it the agents current interaction rates could further deepen the research in MAS allowing even the system to become a part of more real life solutions.

As MAS is a part of artificial intelligence (AI), it has always been interesting to create something with its own mind. The study of AI requires one to have, at least, some knowledge in psychology, since AI systems are based on human behavior or thought process. Thus learning about MAS, one learns more than just one field of computer science and mathematics, but also acquires knowledge in other areas, such as AI and psychology. Being a person who wants to have a broad knowledge in many areas and having a, combined, business entrepreneurship and computing occupation – has truly motivated for this topic.

1.4 The Need for Trust in Open Multi-Agent Systems

Open distributed systems can be implemented as open MAS which are constructed of autonomous agents which interact with each other by using various protocols. Thus, [42] suggests that, *"interactions form the core of multi-agent systems."* Where agents often need to use services or resources which are provided by other agents to achieve their goals. If this is not controlled or dealt with can cause malfunctions, breakdowns, and effectives of MAS's and cause a threat to the well being of the system in the long run.

1.4.1 Agent's Selfishness

Agents, in open MAS, can be termed as *selfish*. As agents will use the most plausible design strategy to maximize their individual utility outcome [51]. This presents a number of challenges discussed in [42]. To begin with, if the system is open and agents can join, interact, and leave at any given time; an agent could join and promise a high quality product but deliver a low quality product. After the transaction the agent would leave the system with the money for the high quality goods, change it's identity and re-enter the system to repeat it's previous action. Thus, avoiding punishment and negative reputation from buyers or authorities. Secondly, agents can join open distributed systems with different properties such as: roles, abilities, policies, etc. This allows agents to choose from a number of interaction partners with different characteristics. Thus, a chance exists that an agent might not interact with the most efficient partner.

1.4.2 Dealing With Selfishness

To mitigate the *selfishness* of agents, protocols are introduced. A *protocol* is a series of instructions, where the moves of the agents and resource allocation are arranged for the agents. The arrangements are made in such a way, that they prevent agents from manipulating other agents for

their *selfish* goal of receiving the highest possible utility outcome. Having protocols can have its drawbacks. Works [45] and [14] talk about protocols in MASs bring up cases where protocols become subjects to trade-offs, because of the rules they create, in the act of trying to achieve their goal.

To deal with such scenarios agents should decide on their own with whom and when they should interact, without having any guarantees that in the end they will achieve their desired outcome. For this to work, agents should have full knowledge about the environment, their partners, and what is at stake. The basic idea is to allow agents to self-police the MAS by giving rating to each other on the basis of their observed behavior and through those observations make decisions with which agent to interact with in the future.

1.4.3 Defining Trust

An important factor still remains - the system, which enforces the protocol, and the agents may have limitations in their storage and computability capabilities which would restrict their performance over interactions. Furthermore, limitations of bandwidth and speed of communication channels introduce a limit to agents. Thus hindering the agent's capabilities in real-world applications. Making it practically impossible to maintain flawless information about the environment, properties of interaction partners, strategies, and interests [6]. Therefore agents are faced with an issue of uncertainty when making decisions. If such an issue is present, agents have to *trust* their interaction partner in order to reduce the uncertainty level within MAS.

Trust has been defined multiple ways, as Falcone demonstrates in his work [15]. Even though Falcone argues that trust is represented as a rich cognitive structure of beliefs, we will define trust the same way it was defined in [8] as it is relevant to this paper.

> **Trust** – "*(or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assess that another agent or group of agents will perform a particular action, both before he can in which it affects his own action... When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him. (According to the definition of Gambetta*[19])" [8]

By taking this view, we can show how the work in this article may be applicable to the extensive trust literature which shares this probabilistic view of trust (Burnett et al. 2014; Huynh et al. 2006; Jøsang and Ismail 2002; Teacy et al. 2006; want and Singh 2007).

The ability for an agent to reason with another agent about honesty, reliability, and reciprocative nature is defined through *trust models* [42]. The trust model allows agents to calculate the amount of trust they can place in other agents. The higher degree of trust put into an agent the higher change the agent will be selected as an interaction partner. Logically, the lower degree of trust is put into an agent, the lower chance that it will become an interaction partner. However, this is only the case if there exist agents with a higher degree of trust. Their choices on trusting each other involve a certain level of risk. To explain the interaction between agents we will be using the following terms:

- **Truster** – An agent that needs to rely on another agent.
- **Trustee** – An agent that provides the required resources or services to the turster agent.

Direct interaction between the truster and trustee can develop strategies to deal with dishonest and honest trustees. These strategies aim to maximize payoffs in the long run. We can further break down trust into two levels based on [42]:

- individual-level trust – Where an agent, based on it's trust model, contains a belief about the trustworthiness of agents trustees.
- system-level trust – where actors in the system are forced to be trustful because of the presented protocols or mechanisms to control the system.

The above approaches, ideally, are used together. As protocols do try to maintain trustworthiness between trusters and trustees they are not always successful at doing so without some loss in efficiency. Trust models in the individual-level aim to guide agent's decision making, while system-level aim to constrain the interaction to reduce uncertainty.

Chapter 2

Background

In this chapter we will be analyzing research which has been performed beforehand. We will discuss a hand full of trust evaluation models and how they are beneficial to our goal.

2.1 Trust Within Multi-Agent Systems

Marsh Stephen was the first to introduce Trust as a measurable property of an entity in computer science in his Ph.D[49]. Following his work, a number of computational models were introduced which focused on different areas of trust management in multi-agent system. Looking at the literature we can generalize the two main steps of the *act of trusting by an agent*, which we will define as:

1. *Trust evaluation* – Where an agent determines the trustworthiness of potential interaction partners.

2. *Trust-aware decision making* – Where an agent selects potential interaction partners based on their trust values.

A paper called, *A survey of Multi-Agent Trust Management Systems*[56], published a list of research papers on trust management which were published during the period of 2002 to 2012. Their research has clearly shown that the majority of the research effort is focused on the *trust evaluation* sub-field. Where it focuses on solving the problem of agents accurately evaluating the trustworthiness of potential interaction partners. The proposed methods for solving this problem are: *Direct trust evaluation models, Indirect/reputation-based trust evaluation models, and Socio-cognitive trust evaluation models* which we will analyze in the up coming sections of this chapter. While looking at *trust-aware decision making*, very limited amount of research has been conducted on this sub-field.

Putting these two major sub-fields aside, it is important to understand and asses the performance of proposed trust models. The *Epinions* and *Extended Epinions* data sets[37] attempt to solve certain aspects of this problem. As it is often hard to find relevant data from the real world, the effectiveness of different trust models need to be assessed under various environmental conditions and misbehaviors. To overcome this problem, in today's trust research field, simulation or synthetic data is used for assessment of the trust models.

A commonly used testbed was proposed in [18] called, *The Agent Reputation and Trust* (ART). A easy accessible testbed with a common experiment environment, to allow researchers to perform their experiments and then compare their findings to determine the most feasible solution.

Despite being commonly used, ART claims not being able to provide simulations of all experimental conditions. Therefore we notice many researchers prefer to design their own simulation environments for their trust models to asses their experiment performance outcomes.

2.2 Trust Models in Multi-Agent Systems

Trust models evaluate trustworthiness through organizational, probabilistic, and socio-cognitive techniques to allow truster agents to evaluate the potential risk of interacting with a potential trustee agent [56]. Once the trustworthiness evaluations are completed for a set of potential trustee agents, trust-aware interaction decision-making approaches help the truster agent appropriately select an appropriate trustee at a certain time. After reviewing relevant literature, we see that the main focus of the research is on improving the accuracy of trust evaluation models which can be classified as:

- Direct Trust Evaluation Models
- Indirect/Reputation-based Trust Evaluation Models
- Organizational Trust Evaluation Models
- Socio-cognitive Trust Evaluation Models

2.3 Direct Trust Evaluation Models

The way humans establish trust between each other is by observing one's behavior and remembering what outcomes occurred as a result of past interactions. Based on these outcomes one can form an evidence based method of evaluating if one should trust that interaction partner once more. This method has been widely adopted by the multi-agent trust researchers. An effective way to model trust between agents is to asses the interaction risk by looking at the probability of the trustee cheating on the truster agent.

The Beta Reputation System (BRS) is one of the first models that attempts to obtain trustworthiness value based on direct evidence proposed in [13].The model looks at the past experience of trustee agents and uses it to measure the amount of trustworthiness for that trustee. The technique used to calculate trustworthiness in BRS is through reputation. Reputation is a expectation value of positive and negative feedback on a trustee agent. The expectation value is then deducted by the uncertainty of truthfulness of feedback by belief or disbelief. The value keeps adding up, while gradually discarding the past evidence. Thus resulting in the representational value of the agent.

In BRS, the result of an interaction between agents is defined by a binary value. In other words, the outcome is defined by either a complete success (1.0) or a complete failure (.0). Jøsang and Haller introduced the Dirichlet Reputational System (DRS), which allows us to handle cases where the outcomes of the interactions are rated on a polynomial scale. [27] DRS works in the same principle as BRS, only it models historical interactions on a scale. The interaction outcome rating can be defined from 1 to 5, where 1 is the most unsatisfactory evaluation and 5 the most satisfactory evaluation. In mathematical terms, the interaction outcomes takes the value of i, where $i = \{1,...,k\}$. With this scale in use, a number of ways obtaining the reputation of a trustee agent present themselves. We can represent reputation as:

1. Evidence representation
2. Density representation
3. Multinomial probability representation
4. Point estimate representation

Representations 1 and 2 are much harder to implement in human practice than representations 3 and 4. We see that researches tend to adopt BRS more extensively than DRS.

To evaluate the performance of a trustee agent, we should analyze the quality of the services provided. A multi-dimensional trust model is proposed in [20]. The trust model allows to asses the trustees trustworthiness by looking at:

1. The probability that the interaction will produce a successful result.
2. The probability that the interaction will take place successfully with the expected budget.
3. The probability that the interaction is completed within the given time sequence.
4. The probability that the result of the interaction is of desired quality.

All four responses are taken from agents and trustworthiness of the trustee is computed by weighting the preferences of the agents and averaging them out.

Wang and Singh focused on another important aspect in evidence based trust models. They proposed a way to quantify the uncertainty which is found in trust evidence [53]. This is effective in cases where, for example, a truster agent A has interacted with trustee agent C two times. In both interactions the outcome was successful. Then truster agent B has interacted with trustee agent C 200 times and only 100 of those interactions were successful. This brings up a question; which of these two scenarios provides more uncertainty for evaluating trustee agent C's trustworthiness. Wang and Sigh [53] address the problem by calculating the uncertainty as means to evaluating a trustee's trustworthiness by looking at a set of trust evidence based on the distribution of positive and negative feedback. The approach produces a certainty value from 0 to 1, where 0 is the least certainty and 1 represents the highest certainty. By using this approach we know that certainty is high if the amount of trust provided is large and certainty is high if the amount of conflicts between feedback are low.

The methods in direct trust evaluation models allow individual truster agents to make complex decisions with more knowledge on what is at stake and if the interaction will be a success. This is possible by looking at the evidence and selecting the evidence which is most relevant for the agent's interaction. Existing-based evidence trust models often store past evidence according to the context they belong to. This does limit the evaluation of trustworthiness only within a stipulated context. When we see if an agent has high trustworthiness in web-development, for example, we would not know a lot about the agent's trustworthiness in e-commercing.

2.4 Indirect/Reputation-based Trust Evaluation Models

While direct evidence could be considered as the most applicable source of information for a truster agent to evaluate a trustee agent, there are cases where direct evidence is not available.

Especially, where there is a large number of agents in a MAS and they rarely interact. Indirect evidence is introduced to settle this issue. In indirect evidence agents receive evidence from third-party agent interactions with a trustee to estimate its trustworthiness. This method of receiving evidence rises a new type of problem – the presence of bias which can negatively affect the trust-aware interactions.

2.4.1 Trust Evidence Aggregation Approaches

Two specific sources are often used for evidence-based trust models for trustworthiness evaluation of a trustee agent. The first, direct trust evidence – direct experience from a truster agent's interaction with a trustee. Second, indirect trust – experience inherited from another agent's previous interaction with a trustee agent. When working with these two types of evaluation sources most trust models work with a weighted average approach. Where direct trust is given a weight of γ ($0 \geq \gamma \geq 1$). Indirect evidence is given a corresponding weight of (1 - γ) [56]. The aggregations of direct and indirect trust evidence can be split into static approaches and dynamic approaches. Static is where the value of γ is defined and does not change. Dynamic is where the value of γ is constantly adjusted by the truster agent.

A static γ approach is quite popular. Most researches tend to take a balanced approach and define the value of γ as 0.5 [34], [25]. However some researchers tried assigning values of 0 to γ [26], [46] or the values of 1 to γ [48] for using just one source of evidence. A paper by Barber and Kim [4] demonstrates that direct trust evidence is the most useful to a truster agent in the long run when excluding the presence of bias. Indirect evidence is most useful in the short term period as it allows to draw an accurate picture in a short period of time. Proving that ignoring one source of evidence can hinder advantages which are provided by evidence based trust models. Generally, we see that using a static value for γ is not an efficient strategy.

γ can also be adjusted to contain a dynamic value. The value of γ is diverse due to the number of direct observations, such as, how the trustee agent behaves to a available truster agent[38]. We can assume that a new truster agent, which joins a simulation, has no prior interaction experience with a trustee agent and continuously generates direct trust evidence over time. From the start, an agent fully relies on indirect trust evidence for selection of trustee agents, and over a series of interactions generates direct evidence for itself. According to the formula[38] below γ increases as the number of interactions with a trustee agent C increases:

$$\gamma = \begin{cases} \frac{N_C^B}{Nmin}, & if\, N_C^B < Nmin \\ 1, & otherwise \end{cases} \tag{2.1}$$

Where N_C^B represents the total number of direct observations of trustee's agent C's behavior by truster agent B. N_{min} is the minimum needed number of observations for a acceptable predetermined error rate ε and level of confidence ϑ. The *Chernoff Bound Theorem* is used to calculate N_{min} as: [56]

$$N_{min} = -\frac{1}{2\varepsilon^2} ln \frac{1-\vartheta}{2} \tag{2.2}$$

This method does not filter biased third-party evidence, the goal is to create enough direct trust evidence. The aim is that a truster agent can accumulate an accurate prediction of the trustworthiness of a trustee agent without looking into the indirect trustee evidence available. To achieve a low error rate and a high level in confidence one may notice N_{min} may be high. This means there might be risk when interacting with a truster agent. γ is increased to 1 thus this technique assumes that the way agents behave does not change over time. This, however, is not always true.

We take a look at [17], where an approach based on a Q-learning technique [5], suggests to select the γ value from a established static set which we will define as Γ. For Γ to maintain a set of values which are appropriate, we would need to have expert opinions about the characteristics of the system to be available. As different γ values offer different rewards for a truster agent, Q-learning chooses the γ value which provides the highest reward at each time step. Thus [18] have provided the beginning of a way to use interaction outcomes between agents to allow the truster agent to compare two sources of trust evidence. On the other hand, the method is using a predefined set of γ values. This affects the performance as the quality of expert opinions used to form the set of γ values may be off, or not sufficient.

2.4.2 Testimony Filtering Approaches

Researchers have proposed a significant amount of models which focus on third-party testimony filtering. The downside, however, is that the models usually assume that the environment has some sort of infrastructure support or maintain special characteristics.

In the *ReGreT* model [44] social relationships among the simulation members are given a assumption of credibility of witnesses. Pre-defined fuzzy rules are used to look at each witness and estimate their credibility which is then used as the trustee agent's weight. The model uses information provided by the social network of the agents, however such information is not available in many systems.

A different paper that unfair testimonies are assumed to display certain characteristics [54]. This approach is connected with the Beta Reputation System [13], where it records testimonies in the structure of successful or unsuccessful interactions with the trustee agent. All testimonies are then given equal weights to gather the most popular opinion and then each testimony is tested. The test exists to see if an opinion is outside the quartile, q and (1 - q) quartile of the majority opinion. If that is the case, the testimony is abandoned and the opinion of the majorities is updated. The model tries to prove that the majority opinion is always right, however this method is not really effective in environments where highly hostile agent witnesses have the intent to do harm.

Direct experience of a truster agent is assumed to be the most reliable source of opinion about trustworthiness of a trustee agent which is proposed in [25]. This is used as the foundation for testimony filtering before allowing them to form an evaluation of the reputation. The authors used an entropy-based method for the measurement on how much a particular testimony differs from the current truster agent before it decides weather to include it into the current used belief. This method, however, heavily depends on the amount of direct interaction experience with a trustee agent.

Another approach is looked at in [35], where a model is proposed for filtering potential unfair

testimonies by looking at behavioral data from the witnesses. To test their model, the researchers created a simulation where the data was giving biased ratings for virtual products on purpose. The idea behind their model, *TAUCA*, that combines an alternative of the cumulative sum (*CUSUM*) approach, which is taken from [40], which finds the point in time when witness behavior patterns change. This is done with a correlation analysis to remove testimonies which are suspicious. In their experiment they have demonstrated that their model can withstand *Sybil* attacks. A *Sybil* attack is where an attacker has access to multiple identities and uses them, without getting punished, to give out unfair or biased ratings.

In [34] a model is presented which allows to record interaction outcomes in multi-dimensional forms. That model is capable of doing that by receiving testimonies and clusters them twice to identify testimonies which are highly negative or positive about a certain trustee agent. If the model does not receive enough opinions which are highly negative or positive to form an opinion cluster which clearly demonstrates the majority, they are both removed and tagged as unfair testimonies. Then the testimonies which are left are used to estimate a trustees reputation. Due to the fact that this method requires an act of repeating a process until the goal is reached, the computational complexity is high, $O(mn^2)$, where m represents the amount of trustee candidates who have yet their reputations to be evaluated; n represents the amount of testimonies received for each trustee agent which is a candidate. This method is not resilient in environments which are hostile, where most of the witnesses are malicious.

Methods mentioned above all tend to calculate their prior experience all at once. This makes reputation-based trust evaluation models highly dependent on what experience has been shared by the witness agents. Such information in practice is obtained and maintained over time, thus making it hard to adapt these models in real life situations. To solve this problem [52] has proposed a mechanism which does not need human intervention for parameter changing. This makes the model have less uncertainty in performance and be self efficient. The model allows a truster agent to update its interaction experience on an ongoing basis. The agent considers, both, certainty and trust together which are allowed to vary and does vary when new evidence is available.

2.5 Organizational Trust Evaluation Models

Trust evaluation models introduce structure organization into multi-agent trust management approach to maintain trust within a MAS. This structure is only achieved if there is at least one trusted third-party in the MAS which can take the position of a supervisor of the transactions between agents.

One of the first papers on this topic [31] suggested a framework which consisted of three segments:

1. A well defined translational organization structurer, which would be based of three roles. (Authority, counter-party, and addressee) transactions

2. A language specification of the contract for managing the contract.

3. Using the specification language to create contract templates.

For the agents to be able to have transactions among themselves, an agent would need to register with someone from the authority, establish an agreement within the contracts by negotiating

the terms and successfully conduct the expected work while being supervised by an authority.

Other researchers have developed The Certified Reputation (CR) model in [23]. The model introduces a mechanism where a truster agent receives certified ratings about their past performance from a trustee agent. The work demonstrates that it is possible to force certified rating sharing a standard procedure of transactions between agents. Truster agents can become more efficient by only focusing their efforts on third party testimonies and filtering these testimonies as they are already provided with certified ratings.

An agent coordination mechanism is proposed in [21] which is based on the interplay of trust and proposes organizational roles for agents. Multiple transactions allow agents to see what task a trustee is good at and what role each agent can take up in their society. The organizational structure becomes dynamic and evolves. Trustee agents begin to act as references for truster agents on how to delegate tasks.

2.6 Socio-Cognitive Trust Evaluation Models

This category in multi-agent trust systems focuses on the analysis on the inherited properties of the trustee agents and the external factors which have affects on agent behavior in future interactions. In this section, trust models are mainly designed to help out evidence-based trust models in situations where there is not enough evidence for agents to make decisions.

A trust model based on the concept of *Fuzzy Cognitive Maps* (FMs) is proposed in [12]. The model works by creating a universal list of external and internal factors into FCM. This allows the truster agent to decide if a trustee agent is suitable for interaction. A system where truster agents can decide what values to give to different interaction links and share their own interaction preference. However, this causes different opinions on same links, which has an impact on the performance of the model. As opinions are subjective it is hard to say how accurate the model's outputs are.

In [3] a model is proposed which lowers the amount of analysis each agent needs to perform before interactions. The model abandons the method of using social relationships and uses market relationships which are built up through interactions. Competition, collaboration, and trade dependency are some examples of the relationships identified by the model, which are focused on by analyzing interactions through an agents point of view which is composed of agent based market model. The market model is then filtered to find the most relevant for analyzing an agents trustworthiness. Once the relevant relationships are identified, they are analyzed to determine the estimate value of the agents trustworthiness.

The model proposed in [33] is the first trust inference model which computes the confidence level based on social network information. It was given the name *SUNNY* – it maps a network of trust into a Bayesian Network, which helps a lot for probabilistic reasoning. The created Bayesian Network is used for estimating the lower and upper bounds of the confidence level for evaluating trust. The confidence level is then used as a heuristic to calculate the most accurate trustworthiness estimation of the trustee agents in the Bayesian Network.

The final Socio-Cognative trust evaluation model we will mention, will be further discussed in the upcoming parts of this paper. A bootstrapping problem was introduced and addressed in Chris Burntt's, J. T. Norman's work [7]. It deals with creating evidence-based trust models with the

assumption that there is no information, prior interaction experience nor information about social relationships that is present about trustee agents which just entered the MAS. Intrinsic properties of a trustee agent are used to echo it's trustworthiness. The model presents a way to draw stereotypes based on the features of agents using a decision tree approach. The features of a new trustee agent are then looked at and the agent is assigned to a specific stereotype determining its performance by introducing stereotypical reputation values. However, [56] states that there is a lack of suitable data, and the research did not demonstrate which features are useful and which are not when estimating a trustee's trustworthiness.

Chapter 3

Framework

3.1 Learning Decision Trees

A study [43] suggests that *Learning Decision Trees* (LDT) are one of the simplest and most successful type of learning algorithm. An input is converted into an object which is described by a set of attributes and returns a output value for the input. When a decision tree learns a discrete-valued function, it is referred as *classification learning* and when the tree learns a continuous function it is referred as a *regression* learning. For the purpose of this project we will focus on *boolean* classification where each example is classified as positive (true) or negative (false).

The decision of the tree is performed by a series of tests. The test value of one of the properties correlates to the node and the branches from that node are labeled with the possible values for the test. Usually each leaf node contains a specific value which has to be returned if that leaf is reached.

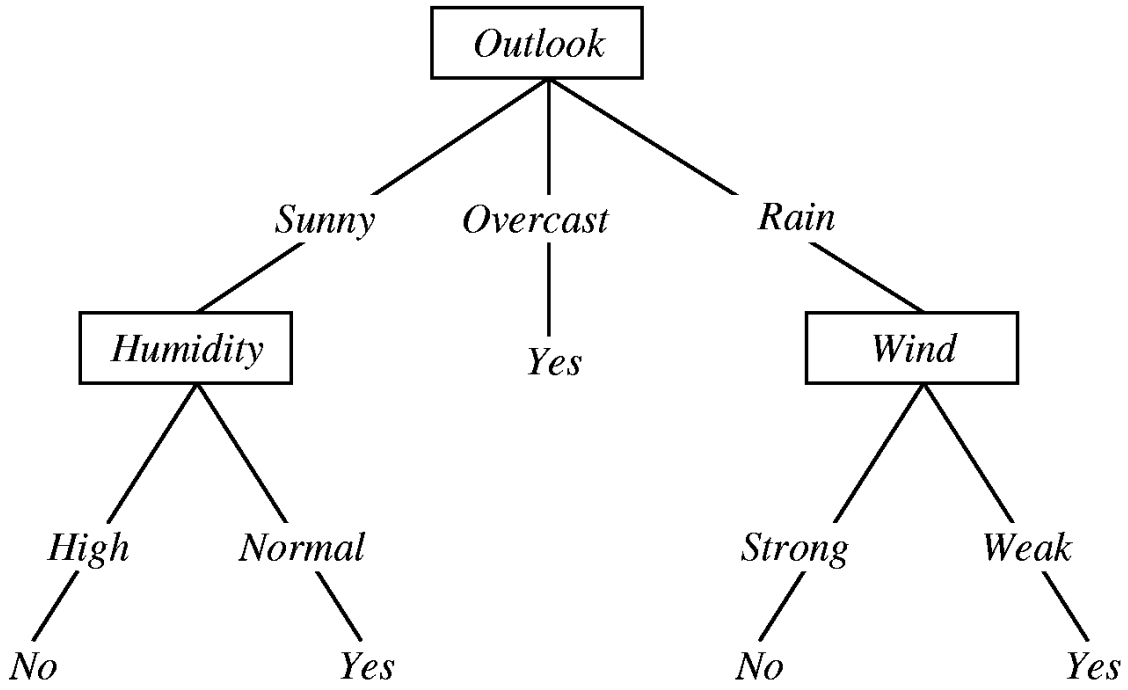


Figure 3.1 – taken from [2], displays an example of a Learning Decision Tree.

As decision trees are fully expressive, any boolean function can be written as a decision tree. However, this is where the problems arise. When short functions would construct a small

decision tree, large functions tend to construct large complex decision trees, making them good at *parity*(small) functions while bad with *majority*(large) functions.

The idea behind a LDT algorithm is to begin with the most *important* attribute first. Most important attribute is referred to an attribute which makes the most difference during the classification. Thus a smaller amount of tests would be required to achieve the wanted goal making the LDT short. Thus LDT have been modified and used in various researches in MAS.

3.2 Defining Variables

For the purpose of this better understanding of the functionality of the frameworks we will allocate mathematical variables to terms described beforehand. We will define a society of agents as A = $\{x, y_{...}\}$, where specific roles are assigned to agents. We will assign a lowercase x to represent a truster agent $x \in$ A, and a lowercase y to represent a trustee agent $y \in$ A. For tasks we create a set, $\mathcal{T} = \{\tau_1 \ ... \ \tau_n\}$ which represents a set of possible tasks for agents to be able to perform and each task $\tau \in \mathcal{T}$ has a number of possible outcomes $\mathcal{O}_\tau$.

3.3 Stereotypical Trust Evaluation Framework

In this section we will describe a framework used by Burnett's, C., Norman's, T. J., and Sycara's, K. research [8] called, *Stereotypical Trust and Bias in Dynamic Multiagent Systems* which is the inspiration of the framework used for this research. Their research is described and analyzed in chapter 4 of this paper.

3.3.1 Agents and Tasks

In order for an agent to determine whether an interaction outcome o_τ, was satisfactory or unsatisfactory, agents use their own *subjective evaluation function* for their given task [8]. The function holds representation that different agents may have a different view or expectations on what is good or bad performance in a given task. The subjective evaluation function is of an agent x for task τ as $\zeta_\tau^x : \mathcal{O} \rightarrow \{\ 0,\ 1\}$, where 1 defines task success and 0 defines task failure. This allows any agent to partition the set of possible task outcomes into those which an agent thinks in its opinion would represent a success and failure. Mathematically, $O_{x:\tau}^+$ is a positive opinion and $O_{x:\tau}^-$ is a negative opinion, "...such that $O_{x:\tau}^+ = \{o | o \in \mathcal{O}_\tau \wedge \zeta_\tau^x(o) = 1\}$, $O_{x:\tau}^- = \{o | o \in \mathcal{O}_\tau \wedge \zeta_\tau^x(o) = 0\}$, $O_{x:\tau}^+ \cup O_{x:\tau}^- = \mathcal{O}_\tau$ and $O_{x:\tau}^+ \cap O_{x:\tau}^- = \emptyset$" [8]

The global population of agents is split into, so called, *ad hoc groups* which are denoted as $\mathcal{G}$ = $\{\ G_1 \ ... \ G_n\ \}$, $\forall$ G $\in \mathcal{G}$, G $\subset$ A. The framework does not require for the ad hoc groups to cover the whole population. However, it is require that agents would be in only one group at any given time through-out the simulation, denoted as G,$G' \in (\text{G} \neq \text{G} \rightarrow \text{G}\ \ G' = \emptyset)$

Particular agent x which is allocated to a group is defined as G_x and visible recommender for that agent are denoted as $R_x \subset$ A. Similarly, candidate trustee agents y are defined as $Y_x \subset$ A. Agents were planned to be only visible to other agents only if they are in the same ad hoc group, defined as $\forall x \in$ A, $R_x = G_x \setminus \{\text{x}\}$ and $\forall \in$ A, $Y_x = G_x$. In this framework, trusters Y_x are allowed to consider delegation with themselves. With this setup the framework conforms the view of Castelfranchi and Falcone [11], whose cognitive trust model approach allows agents to form trustworthiness from their own beliefs.

Previously, encounters of each truster $x \in$ A holds a set of opinions Ops_x about trustee agents,

along with an experience base Eb_x. This is a way to hold direct experience of the agents, from which opinions can be formed, described in the upcoming section. To represent each agent's $x \in$ A evaluation for the degree of trust for a trustee $y \in$ A for a task $\tau \in \mathcal{T}$, an evaluation function is used; $E_x(y, \tau, Ops_x\ R_x, S_x)$, given the base experience of agent x, Eb_x, existing opinions held by visible recommender, R_x and existing opinions held by x, Ops_x. The evaluation function is also influenced by the stereotypical evaluations S_x The Stereotypical evaluations are further described in section 4.4.

Agents are given a binary set of features $\mathcal{F} = \{f_{1...}f_n\}$, which is a set of all possible features an agent may inherit. Each agent $x \in$ A has a feature vector which is visible and contains the values for some or all features inherited to the agent, denoted as $F_x \subseteq \mathcal{F}$. As feature set is binary, a feature vector F_x of agent x can give information on one of three things about a specific feature $f \in \mathcal{F}$. Therefore, unobserved features can be defined as $\neg\, f \in \mathcal{F}_x$, or in other words, some of the agent's x features are not in the feature vector.

3.3.2 Opinion Representation

For the framework to produce subjective opinions or beliefs about the trustworthiness of agents based on evidence, a simple trust evaluation model, based on *Subjective Logic* (SL) [29] is used.

- **Belief Representation** – An opinion about agent y performing a task τ which is held by agent x is defined as:

$$\omega^x_{y:\tau} = \left\langle b^x_{y:\tau}, d^x_{y:\tau}, u^x_{y:\tau}, a^x_{y:\tau} \right\rangle \tag{3.1}$$

$$\textbf{where } b^x_{y:\tau} + d^x_{y:\tau} + u^x_{y:\tau} = 1, \tag{3.2}$$

$$\textbf{and } a^x_{y:\tau} \in [0,1]. \tag{3.3}$$

[8]

Where $b^x_{y:\tau}$ represents the degree of belief, $d^x_{y:\tau}$ represents the degree of disbelief, $u^x_{y:\tau}$ represents the degree of uncertainty, and $a^x_{y:\tau}$ represents the base rate, or in other words a prior degree of belief. The term belief means the extent of an agent x believes that giving a task τ to agent y will end up with a satisfactory outcome.

- **Evidence Aggregation** – Agents base their opinions which are backed up with evidence.

Evidence is received by interacting with other agents and evaluating them. If no direct interactions are present beforehand, agents can receive evidence from third parties who have gathered direct interaction evidence, beforehand, on the particular agent. Opinions within the SL are formed by accumulating positive and negative evidence about a particular agent. When agent x holds evidence about another agent y is defined as, $\left\langle r^x_{y:\tau}, s^x_{y:\tau} \right\rangle$. Where $r^x_{y:\tau}$ represents a number of positive experiences of agent x about agent y and $s^x_{y:\tau}$ represents a number of negative experiences. The parameters of $r^x_{y:\tau}$ and $s^x_{y:\tau}$ are used to produce an opinion based on the equations below: [28]

$$b^x_{y:\tau} = \frac{r^x_{y:\tau}}{r^x_{y:\tau} + s^x_{y:\tau} + 2} \tag{3.4}$$

$$d^x_{y:\tau} = \frac{r^x_{y:\tau}}{r^x_{y:\tau} + s^x_{y:\tau} + 2} \tag{3.5}$$

$$u^x_{y:\tau} = \frac{r^x_{y:\tau}}{(r^x_{y:\tau} + s^x_{y:\tau} + 2)} \tag{3.6}$$

[8]

By using these equations, a simple map is created from the observed evidence about an agent's performance on a particular task to the opinion representation defined beforehand. The equation (6) is there to ensure that uncertainty decrease as more evidence is observed.

- **Probability Expectation Value** – The the probability of an expectation value for an opinion.

A single-valued trust metric can be used as an opinion's probability expectation value, which is suitable to rank potential partners. Below equation 3.7 demonstrates how the probability expectation value $P(\omega^x_{y:\tau})$ is calculated from a particular opinion $\omega^x_{y:\tau}$;

$$P(r^x_{\omega:\tau}) = b^x_{y:\tau} + a^x_{y:\tau} + u^x_{y:\tau} \tag{3.7}$$

[8]

The framework uses equations (3.7), (3.4), and (3.6) together to calculate the probability expectation value $P(\omega^x_{y:\tau})$ for a given evidence pair $\langle r^x_{y:\tau}, s^x_{y:\tau} \rangle$.

- **Base Rate** – A parameter which represents the prior degree of trust of a truster agent about a trustee agent performing a specific task.

The base rate $a^x_{y:\tau}$ is a parameter which represents the prior degree of trust agent x has about agent y performing a task τ before any observable evidence has been received and reviewed. The base rate determines the parameter's $u^x_{y:\tau}$ effect on the produced result probability expectation value.

When converting an opinion which is represented in belief representation to a *rating* (or opinion) which would be compatible with the classical probability theory, the framework faces to resolve the unassigned ambiguity in MAS. Free ambiguity belief is transferred to be either an opinion of belief or disbelief. As seen in equation (3.7) the transfer of an opinion depends on the base rate parameter, $a^x_{y:\tau}$, which, by default, holds a prior of 0.5 as it, at its current state, is uninformative as no negative $O^-_{x:\tau}$ or positive $O^+_{x:\tau}$ evidence has been received. In other words, $P(omega^x_{y:\tau}) = 0.5$, where 0.5 is the least informative value. The values of $a^x_{y:\tau} > 0.5$ will result in more of MAS ambiguity being converted to belief and $a^x_{y:\tau} < 0.5$ converted to disbelief.

Due to the framework's additivity requirement of the parameters of $b^x_{y:\tau}$, $d^x_{y:\tau}$, and $u^x_{y:\tau}$, there is no need to represent the opinions in three dimensions. Two degrees of freedom are used when plotting opinions, thus the opinion spaces of agents can be demonstrated as a ternary plot.

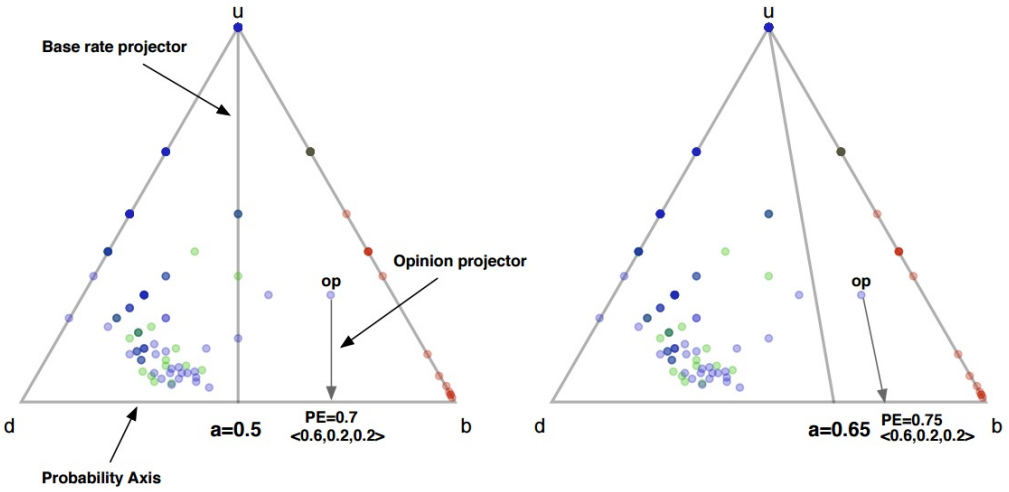


Figure 3.2 – taken from [8], displays a sample opinion space for a truster agent with a base rate projector.

The top vertex u represents the maximum ambiguity, the bottom left, d, represents maximum disbelief, and the bottom right, b, represents the maximum belief. The degree of belief is represented by the distance from the midpoint of the leftmost edge, the distance from the rightmost edge to the midpoint represents disbelief, and the degree of ambiguity in the opinion is represented as the distance from the bottom edge [8].

The probability axis, which lies on the bottom edge, identifies opinions which have no ambiguity, if an opinion lies on the bottom edge.That is where the base rate value $a^x_{y:\tau}$ is plotted. When calculating $P(\omega^x_{y:\tau})$ opinions are *projected* onto the axis, parallel to the base rate projector line, from top (u) to bottom (a).

Figure 3.2 demonstrates two different base rates. The left ternary plot has a $a^x_{y:\tau} = 0.5$ base rate opinion and the right ternary has a base rate opinion of $a^x_{y:\tau} = 0.65$. Thus, the left ternary plot would resolve any unassigned ambiguity MAS better than the right.

- **Reputation** – the opinions that are generally held about an agent's performance.

Based on [53] reputation is calculated by clustering the $r^x_{y:\tau}$ and $s^x_{y:\tau}$ parameters from agents which provide reputation. The framework clusters the set of recommender agents R as $\langle r'^x_{y:\tau}, s'^x_{y:\tau} \rangle$, where:

$$r'^x_{y:\tau} = r^x_{y:\tau} + \sum_{\rho \in R_x} r^\rho_{y:\tau} \tag{3.8}$$

$$s'^x_{y:\tau} = s^x_{y:\tau} + \sum_{\rho \in R_x} s^\rho_{y:\tau} \tag{3.9}$$

[8]

After the clustering of the evidence parameters, equations (3.4) and (3.7) are used to calculate opinions and ratings for the combined evidence.

3.3.3 Decision Model

This process involves evaluating potential partners, which differs from deciding which partner to interact with, or to interact at all. A simple decision trust model is used, where a number of possible candidates Y_x. A truster agent x will select the candidate Y_x with the highest rating for task τ, which is defined as $C_{x:\tau}$. The trusters evaluation model is defined as;

$$C_{x:\tau} = \textbf{arg max}_{y \in Y} \; E_x(y, \tau, Ops_x, R_x) \tag{3.10}$$

3.4 Alternative Stereotyping Framework

For our research, we have based our framework on Burnett's and his colleagues' work discussed above, [8]. However, we have decided to focus on tasks rather than agents. Where agents are given a specific task and are asked to choose a candidate for interaction and interact, the outcome is then split into two groups. In one, we have agents who have successfully completed the task and agents who have unsuccessfully completed the task in the other group. We investigate a new approach of investigating agent features by categorizing which, if any, have an effect on performance of a specific task and if there exists any correlation between different tasks and features of an agent.

3.4.1 Tasks and Agents

Tasks are represented as features and are given a set of binary features $\mathcal{F}_\mathcal{T} = \{f_{\tau:1 \ldots} f_{\tau:n}\}$ which are allocated to each agent, $F^x_\mathcal{T} \subseteq \mathcal{F}_\mathcal{T}$. Where the features for a task τ_n are defined as $\mathcal{F}_{\tau:n} \rightarrow \{0,1\}$, where 1 defines which task has to be attempted and 0 defines tasks that the agent does not have to perform. Each agent has to complete one task at a time, thus every agent receives the set of task features, $\mathcal{F}^x_\mathcal{T} = \{f_{\tau:n} = 0, f_{\tau:n+1} = 1\}$, where there is only a single task feature represented as one, $f_{\tau:n+1} = 1$, and all the others are given zeroes, $f_{\tau:n} = 0$.

We use [8] concept of a global society of agents A = {x,y} for which we use the same binary set of features for agents $\mathcal{F} = \{f_{1 \ldots} f_n\}$ where it represents the set of possible features for an agent may inherit denoted as $F_x \subseteq \mathcal{F}$. The interactions between agents are conducted in the same approach, of having the global society split into ad hoc groups, $\mathcal{G}$ = { G_1 ... G_n }, $\forall$ G $\in \mathcal{G}$, G $\subset$ A, with the requirement of agents to be in only one group at a time, defined as G,G‘ $\in$ (G $\neq$ G $\rightarrow$ G G‘ = $\emptyset$).

We will involve the task into the ad hoc group, as each agent is allocated with a particular task τ_n, we will denote a group to which agent $x_{\tau:n}$ belongs to as $G^x_{\tau:n}$. With the same addition of having recommender agents, $R^x_{\tau:n} \subset$ A, visible for each agent $x_{\tau:n}$ and trustee agents suitable for the task τ_n as $Y^x_{\tau:n} \subset$ A. We follow the same rule of visibility, where agents are only visible to each other if they are in the ad hoc group $\forall x_{\tau:n} \in$ A, $R^x_{\tau:n} = G^x_{\tau:n} \setminus \{x\}$ and $\forall \in$ A, $Y^x_{\tau:n} = G_x$.

3.4.2 Generating Predictions

Rather than having agents represent their subjective opinions, we focus on distinguishing which features represent good performance for a specific task and which do not. We do, however, use the same approach for receiving the base rate and reputation for each agent for a task. In our learning model we have agents with specific features and specific task features, $A_{F_x}^{\mathcal{F}_\tau}$ which perform their specific task and are split into two groups based on the outcome. If their assigned task has been successfully completed they are placed into a group denoted $V_A^{\tau:n}$ and agents where unsuccessful in completing their task as $N_A^{\tau:n}$. These groups are then investigated, where each of the agents features F_x are looked at. These features are placed into two more groups. The agent features with positive outcomes are denoted as $F_{x:t_n}^{+}$ and negative outcomes as $F_{x:t_n}^{-}$. These features are then looked at to find a correlation on which features of agent, F_x, are needed to complete a task, τ_n. This generated evidence can be then added to each task to define which features are needed to have a successful outcome $\tau_n^{F_x^+}$. The same goes for features present or not present for a unsuccessful outcome $\tau_n^{F_x^-}$. The results are then passed on to the decision model as $E_{\tau:n}^x$. Where it contains the trustworthiness evaluation for an agent to perform the specified task $x_{\tau:n}$.

3.4.3 Delegation Model

After receiving some evidence on the trustworthiness of agents, a decision should be made with which partner to select, or whether to delegate at all. We will use a similar, in simplicity, to [8] trust decision model approach. Having a number of candidates with potential, for interaction, candidates $Y_{\tau:n}^x$, a truster x will always choose the most relevant, in terms of successful outcome, which is denoted as:

$$C_{\tau:n}^x = \textbf{arg max}_{y \in Y} \; E_{\tau:n}^x(y, \tau_n) \tag{3.11}$$

While our decision model can hold a clear investigation of the alternative stereotyping approach, it is important to mention that it is too simple to be used in a real world models. In [11] Falcone and Castelfranchi argue integration of trust is required fr social decision making for the risks, rewards, and contexts received in a particular situation. Our approach does not offer a more mature approach nor does it focus on social decision making, and no reputation filter is used to have filtered opinions be applied to the trust evaluations. Therefore, our future work will propose different techniques which can be applied to improve the quality of trust evaluation as well as decision making based on trust in highly dynamic systems.

3.4.4 Framework Summary

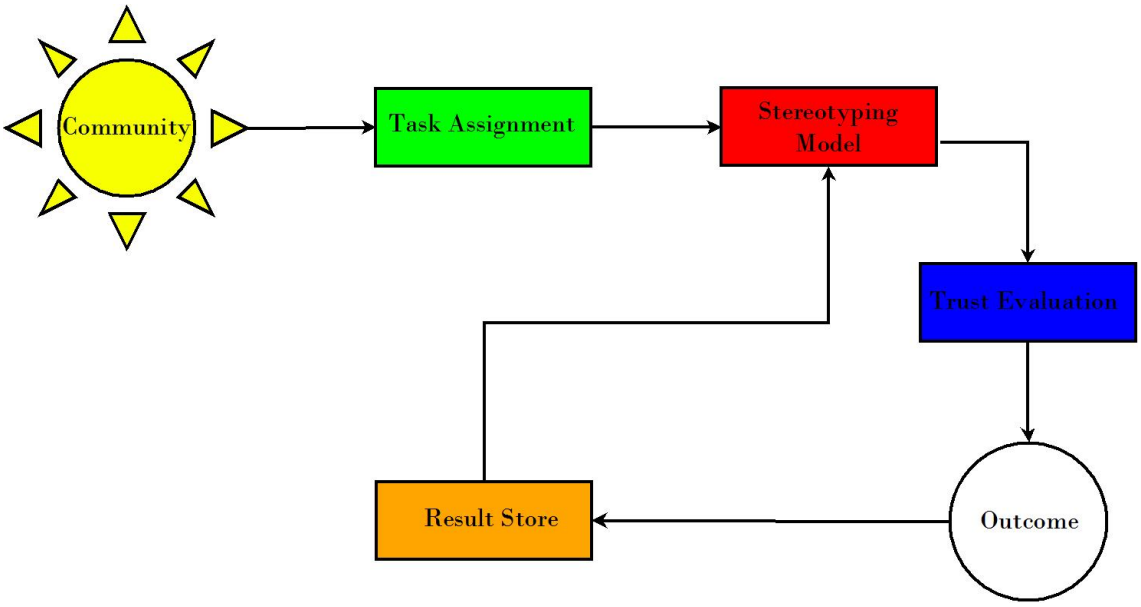


Figure 3.3 – Displays a overview of the alternative stereotypical trust evaluation framework.

Figure 3.3 provides an overview on how components described above all fit together and work as a whole to form a trust evaluation of an agent and decision mechanism. A community of agents are generated into truster x and trustee agents y. Each agent has features generated, F. Agents are then assigned a task, represented as task features, $\mathcal{F}_{\mathcal{T}}$. Where each truster agent is given one specific task to perform, $\mathcal{F}_{\mathcal{T}}^{x}$. If the stereotype model has no prior performance evidence on any task it does not interfere. If the agent has no prior experience with any other agents it may ask for other suitable recommender agents for the task, $R_{\tau:n}^{x}$, to receive a trust evaluation of the trustworthiness of trustee agents, $Y_{\tau:n}^{x}$. A truster x will choose the most reliable and suitable trustee agent for its assigned task. After the task attempt is complete the agents features and assigned task, $A_{F_x}^{\mathcal{F}_{\mathcal{T}}}$, are stored in the result store, where there are two groups for the specific task. Agents which have had a positive outcome are stored in one group, $V_{A}^{\tau:n}$, and agents which had a negative outcome are stored in $N_{A}^{\tau:n}$. The features are inspected in the result store and the features of those agents are passed on to the stereotype model. Where it holds a list of features which an agent had with a successful interaction outcome, $\tau_n^{F_x^+}$, and features an agent had with a negative outcome, $\tau_n^{F_x^-}$. When a new agent joins the community, with no prior experience, it receives a task, and the stereotype model looks at the agent features, along with the task features $A_{F_x}^{\mathcal{F}_{\mathcal{T}}}$. Then compares the agents features F_x with the allocated task outcome positive, $\tau_n^{F_x^+}$, and negative, $\tau_n^{F_x^-}$, features to produce a prediction of the trustworthiness of the agent, based on the assigned task and the features it holds.

Chapter 4

Related Work

In this chapter we will discuss Burnett's, C., Norman's, T. J., and Sycara's, K. research [8] called, *Stereotypical Trust and Bias in Dynamic Multiagent Systems*. We will discuss this particular work because our work is based on a continuation of their experiment and the fundamentals of their program is used with their full acknowledgement and agreement.

4.1 Stereotypical Trust

When diverse individuals from different backgrounds and cultures, without any previous interactions, are grouped in teams to work on a project or solve a problem, there is a form of trust which exists prior to their interactions. In [32] a theory of *swift trust* is presented. This theory characterizes this type of trust which is found prior to interactions and what process is responsible for this trust to be present. In their experiment the researchers look at the relationships formed between film studio crews, who are gathered from different and diverse organizations. The crew is usually assembled for one project and then disbanded. During the observations they noticed that crew members acted as if there was some initial trust present, despite the fact that they have just met while there was no direct or reputation opinions to inherit an idea of a person's trustworthiness. The authors found that the key process of this form of trust is by which individuals generalize their experiences with others to *categorical* experiences. The categories are the absence or presence of certain salient features.

While applying a stereotype to someone/something is often viewed in a negative way, stereotypes are able to present an accurate reflection of reality, based on the basis of rational generalizations from personal experiences [22]. In a real life situation, for example, when an employer is seeking to find employees to hire, he/she will probably produce a *short-list* of candidates based on their *curriculum vitae* (CV). The way the CV's are filtrated and "appropriate" candidates are chosen is by assessing an individual solely on the presented "features" displayed in the CV documents. Education qualifications, marks achieved, previous experience, hobbies, etc. would all be considered and the employer will use his experienced knowledge of relationships between features which may demonstrate trustworthiness and use to make a hesitant evaluation of each candidate, to select the most "suitable" candidate for the job.

4.2 Abstract

In [8] Burnett *et al.* apply stereotypes to agents by considering observable attributes of the agents as features for the stereotype. An example of which attributes could be used as features for software agents are; the trustee agent's owner, user, programmer, or version number. For agents which represent human users of a system in a e-commerce application, the attributes could be; location, nationality, age, etc. The approach taken by the researchers was to obtain a *feature* which would represent every 10 instances of a performed task by an agent. Thus, creating experience "milestones". For example, "*if an agent has performed a task τ 23 times, we may signify this with the feature "n(τ) $\geq$ 20", meaning "performed τ at least 20times".*" [8] The system would then allow a truster agents to build up stereotyping models by looking at agents interactions with different levels of experience, which are then used to predict the trustworthiness of unknown agents. Models which are created then are used to represent the notion of expected "learning curves" for the tasks, their given experience accumulates trustworthiness of the agents.

Broadly speaking, [8] presents an approach where interacting agents in MAS make use of available features in order to construct trust evaluations when no evidence exists of an agent's behavior or trustworthiness. This is done by agents being able to construct stereotypes which attempt to model their observations in the most accurate fashion. Relationships exist between an agent's features and behavior, thus, helping agents to decide which agent to interact with, without the need for exploration or random partner selection.

4.3 Framework Summary

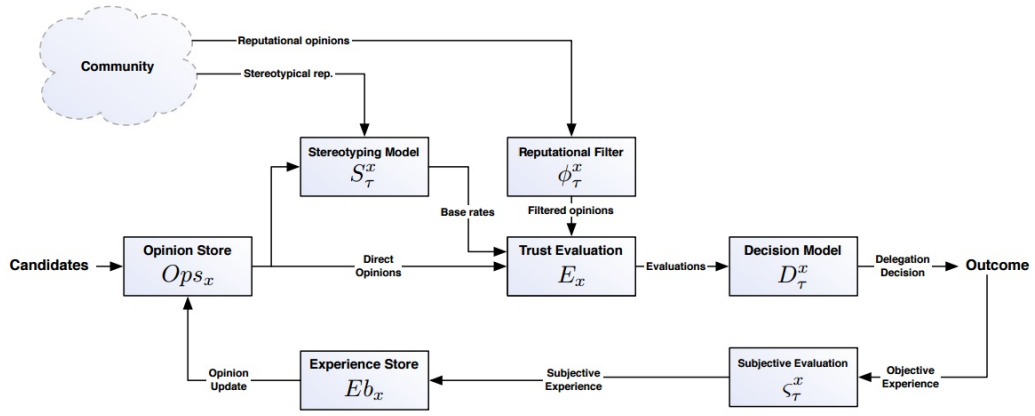


Figure 4.1 – taken from [8], displays a overview of the stereotypical trust evaluation framework.

4.3.1 Figure 4.1 Description

When agent x begins to evaluate a set of candidates, he first looks at his own opinions about the candidates from the opinion store Ops_x. These opinions are transferred to a trust evaluation function E_x. Here, reputation providers from the community may be also asked for their third-party opinions. These opinions together with direct opinions are used by a evaluation function which outputs a set of evaluations. By using the simple decision model $C_{x:\tau}$, the most trusted agent is selected for delegation. After the delegation is completed, agent x looks at the outcome and evaluates it using its own subjective evaluation function which is task-specific, ς^x_τ to produce

a subjective outcome. The new subjective outcome is then used to update the opinion of agent x about his selected candidate.

4.4 Stereotype Model

The stereotype is defined as a rule, or set of rules, where it is assumed that some prior estimate of trustworthiness for an agent is present, based on features observed which agents contain. Thus, a stereotypical evaluation procedure of an agent x may take place. This looks into the a particular task $\tau \in \mathcal{T}$ as a function $S^x_\tau : \rightarrow \mathbb{R}$ which maps possible feature vectors of agents to their initial stereotypical base rate estimates. That way all stereotypes can be evaluated in the context of general trust evaluation mechanisms. However, in cases where the function's output would not represent a base rate but an estimate for the trust metric used, such as the discrete trust value. Regardlessly, the key requirement for this stereotyping function is that it provides estimates which are compatible with the used evaluation model. When a trustee agent does not receive any evidence, the ambiguity is at a maximum state, which is; $\omega^x_{y:\tau} = (0, 0, 1, 0.5)$. Therefore $\alpha^x_{y:\tau}$ determines the value of $P(\omega^x_{y:\tau})$ and as more evidence is received the value of $u^x_{y:\tau}$ is decreased along with $a^x_{y:\tau}$ [8]. Therefore, the fundamental condition, that stereotypical assumption has to have strong evidence as acquiring that evidence, is satisfied. The formed stereotypes are *decentralized*, which means that agents do not need to agree on a common stereotype, as each agent has their own stereotypical model, which can be shared with other agents if evidence is needed.

4.4.1 Stereotypical Reputation

The formation of a stereotype requires direct experience, which is gained through multiple interactions. To deal with this issue, the method gathers *stereotypical reputation* from third-party trusters who have already gained experience through interactions and constructed their own stereotypes. Agent x will ask for indirect evidence from truster y, only if three conditions are met:

1. Agent x has had no interactions, thus having no direct experience from which the agent can evaluate truster's y trustworthiness.

2. Agent x cannot find any reputation evidence about truster y from recommenders (R_x).

3. Agent x has not directly interacted with any agents, or has not observed any features of truster y, (F_y), thus cannot form a stereotypical evaluation for y.

The way *stereotypical reputation* is calculated is:

$$SR^x_{y:\tau} = \frac{\sum_{z \in R_x} c_z a^z_{y:\tau}}{\sum_{z \in R_x} c_z} \tag{4.1}$$

Where the weighted mean of all returned evaluations of stereotypes are calculated. It takes some agent's z evaluation of agent y performing task τ. The model's accuracy is presented with Willmottet's *Root Mean Squared Error* (RMSE) [55] with a function of the differences between the predicted opinions by the model and the actual opinions. The function is presented below:

$$C_z = 1 - \sqrt{\frac{\sum_{\omega^x_{y:\tau} \in O_z} (P(\omega^x_{y:\tau}) - S_z(F_y, \tau))^2}{|O_z|}} \tag{4.2}$$

4.4.2 Learning Stereotypes

The method used to construct stereotypes from experiences are based on agents which present similar features. The ultimate goal for their stereotyping mechanism is that their stereotyping function, S^a_τ, is able to produce prior assumptions based on the observable features of candidates.

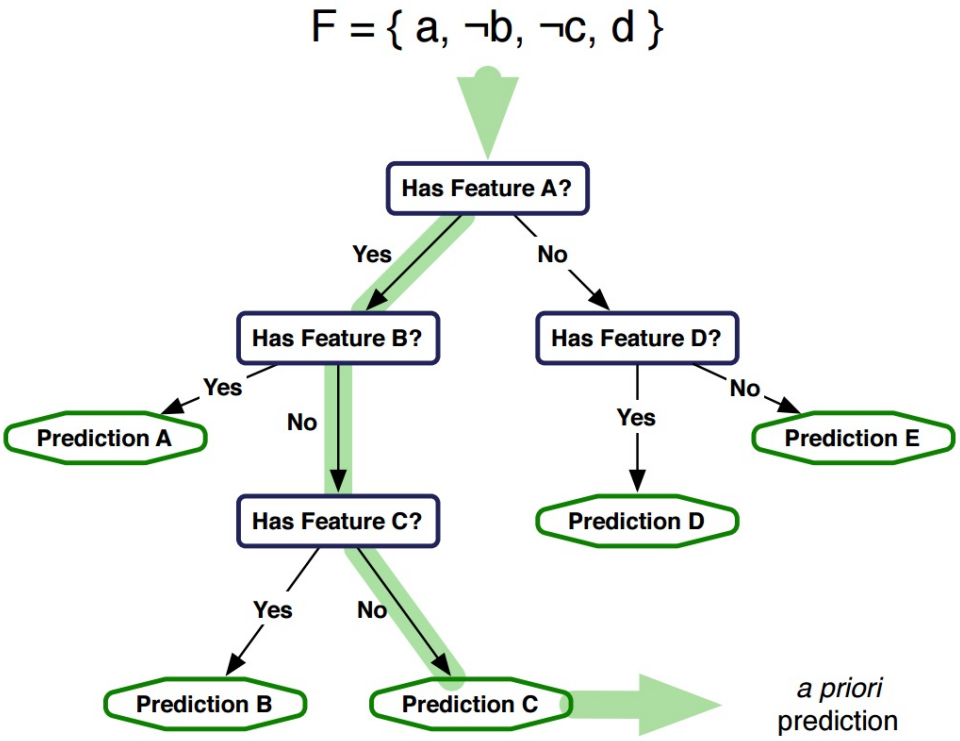


Figure 4.2 – Taken from [8], displays an example of a decision tree encoding a stereotyping function.

With a decision tree encoding a stereotyping function it is possible to enclose all of an an agent's held stereotypes about other agents concerning $\mathcal{F}$ in a compact structure. A particular feature is represented by each node and it's branches are followed based on recognized value of the feature the node represents. The base rate is represented by each leaf node of the tree. This is applied to all the leaf nodes reaching that particular node. Figure 4.2 demonstrates an agent with features a and d, thus, the path it takes results with a predicted stereotypical evaluation for that agent based on the feature vector F. This tree is used when agent y has no direct evidence, thus, giving the agent an estimate of the prior trust value for $P(\omega^x_{y:\tau})$.

As classical decision tree induction methods are not fully suitable for problems where the class value predictions which are real-valued. Burnett and his colleagues presented and evaluated two different tree-based methods for inducing stereotypes as they where not sure which method would be the most appropriate for their needs. Which is to predict a prior trust estimates.The two different methods used are:

1. Two-Phased Learning – Labels trustee agents from a set of finite discrete trust values.

2. Model Tree Learning – Learns a classifier which is capable of predicting exact trust values from a range which is continuous.

4.4.3 Two-Phase Learning

A two-phased learning approach involves obtaining a set of labels, which is done by partitioning the space of opinions in some manner. Once that is complete it goes on to tagging agents which it knows with a partition label to which they belong. Instead of using a static partitioning scheme, where the partition is divided into two groups; "good" and "bad" opinions for example, proposed in [30]. Burnett and his colleagues chose a more flexible approach proposed in [36], which involves to find identifiable *clusters*, statistically speaking, of similar opinions and defining labels as the cluster membership of the trustee agents in the secondary *classification* stage.

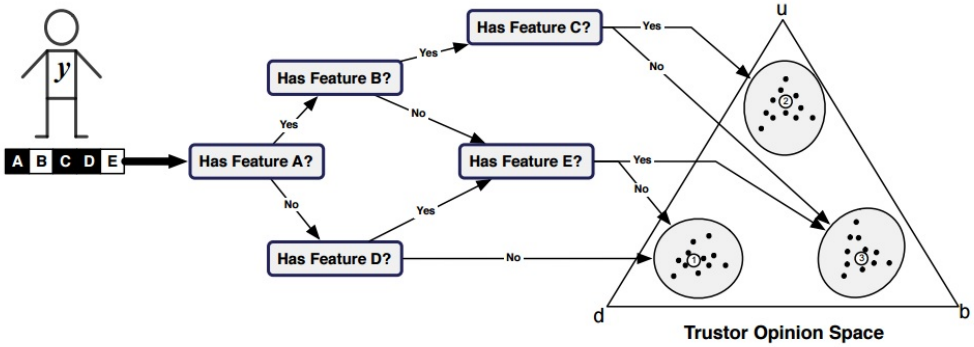


Figure 4.3 – Taken from [8], displays two-phase stereotyping in action.

Figure 4.2 demonstrates an illustration of the two-stage approach. In the first phase, a set of discrete labels are obtained by a truster. The labels are obtained by identifying clusters, using the *k-means* clustering algorithm proposed in [36], of trustee agents in its option base. The feature vector, which is each known trustee agent y, is labeled according to its membership to a cluster L_y to give a set of tuples $\langle F_y, L_y \rangle$. In the second phase, a decision tree is constructed from the produced labeled examples in the first phase. Ambiguous opinions such as, $uy : \tau^x = 1$, add no knowledge to the model. All other opinions, $\omega^x_{y:\tau} \in Ops_x$ have the example, $\langle F_y, L_y \rangle$, added to the training set. Thus, the tree is constructed. As the base-rates are wanted to be received, the class labels are converted to some continuous value which represents their corresponding clusters. That is possible by taking the mean of all opinions within the cluster if a label. The rating function (Equation number 3.7) is then used to convert the three-dimensional centroids to a single probability expectation value.

4.4.4 Model Tree Learning

The alternative to the two phased learning approach, is a decision tree induction technique, which is capable of predicting numerical values. One of them is the *M5 model tree* learning algorithm [41] , [16]. Which is similar to other decision tree approaches by recursively constructing a tree for classification.

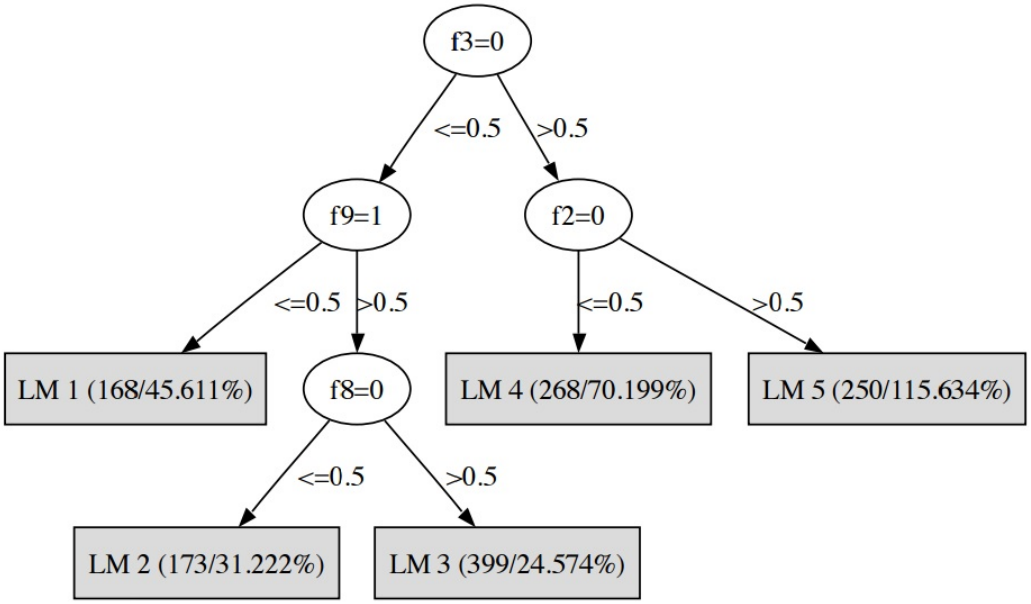


Figure 4.4 – Taken from [8], displays a example of a M5 model tree.

The difference between classical decision trees and model trees is that decision trees use leaves as class labels and model trees use regression models as leaves to estimate the target value. In Burnett's et al. work the approach differs from a traditional tree in two key induction methods.

The first, called *splitting criterion* [8] presents an algorithm which decides which attributes to use as splitting nodes, or weather it is necessary to split at all. This method is used by most tree induction approaches and involves the maximization of information gain, by which it selects attributes which are used to swiftly reduce the *impurity* of the training data subsets which are created by splitting. The second difference is after the construction of the tree is complete, linear models are computed for the tree's leaf nodes. This is computed by using standard regression techniques. A smoothing process is then applied to attempt to compensate for any sharp discontinuities which may appear between the resulting linear models.

4.5 Stereotypical Bias

To deal with other kinds of feature-behavioral correlations *bias* is addressed. Two main bias types are looked at:

1. Perceptual Bias – Trusters may perceive an outcome from a trustee in a more positive or negative way. This is because a trustee may passes or lack certain features. In Burnett's et al. case trusters use a different subjective evaluation (ς_τ^x) functions which depend on the trustee's features.

2. Behavioral Bias – Trustees with certain features might behave more positively or negatively in their interactions which depends on the features of the truster agent.

As the judgment and behavior of agents may depend on the features their partners have, not all opinions will be suitable for all trusters. By incorporating representational opinions without

acknowledging that there is a possibility of social biases, agents may make wrong decisions. To address this problem *reputation filtering* is introduced.

A simple approach is chosen for reputation filtering. Where the opinion providers which decide to be uninformative because social biases which are excluded from the reputation aggregation process. If stereotypical bias is detected in the system, agents will select a set of recommenders they interpret to be most suitable. They interpretation is formed by their own features and features of the recommenders. The filtering process of agent x for a task τ is $\phi_\tau^x : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$. When given a subset which contains the possible recommenders will return a another subset of recommenders which are filtered according to the perceived stereotype biases.

The two phase learning approach is modified in a sense that now it is attempting to find relationships between features of *reputation providers*, when the original approach looks at trusters, and the subjective opinions which they have about trustee agents. The clustering phase has been modified to be responsible for finding significant behaviors or opinions variations among the agents.

4.6 Findings

For the evaluation of the effectiveness of their approach Burnett et al. investigated the following hypotheses;

- "*Hypothesis* 1. If feature-behavioral correlations are present, then stereotyping agents will perform better than nonstereotyping agents." [8]
- "*Hypothesis* 2. The performance of stereotyping models will decrease as the strength of feature-behavioral correlations decreases." [8]
- "*Hypothesis* 3. If no feature-behavioral correlations exist, then stereotyping agents will perform no worse than nonstereotyping agents."[8]
- "*Hypothesis* 4. If either perceptual or behavioral biases are present, then reputation filtering agents will perform better than nonreputation filtering agents." [8]

4.6.1 Experiment Layout

During the experiment a simulation was created with the above discussed framework and its results investigated. 500 agents were created which played the role of trustee agents. 40 agents played the role of trusters. 20 ad hoc groups where created within the society, each containing 10 agents, which was a mixture trustee and truster agents randomly allocated to each group. Groups which would be randomly generated and only have one type of class of agent (only truster) did not engage in any interactions. An ad hoc group would exist for 5 interaction steps, after which it would be disbanded and a new group would follow into its place. The interaction probability was set to P(*interact*) = 0.8 to define the probability each truster agent would interact with a trustee agent. The probability for a trustee agent joining or leaving the society is controlled with a parameter P($\mathcal{J}l$) = 0.01. This is applied to each interaction step. If a trustee would leave, it was automatically replaced with a trustee agent with the same profile. The experiment lasted for 400 interaction steps. The trustees were taken from a series of hidden profiles which would determine their behavioral

characteristics. For each profile, 100 trustee agents were created. A global interaction outcome at each time step was used as a performance metric.

Each profile would specify the standard deviation and the mean parameters of a *Gaussian* distribution from which simulated interaction outcomes were drawn. Each profile would specify the first 6 features from a possible set of features $\mathcal{F} = \{f_{1...}f_{18}\}$. $\mathcal{F}$ was shared by all agents of that profile. The rest features in $\mathcal{F}$ were considered as *noise* features which would be randomly assigned and did not correlate with the profile membership.Agents would be sorted into different profile groups which would represent unreliable agents and reliable agents.

For evaluating the reputation filtering model, opinions and behaviors biases were added to the profiles in addition to the *Gaussian* parameters. Profiles $p1$ and $p2$, where each profile was negatively biased were made use of towards the other, positively biased agents with similar features.

4.6.2 Research Conclusion

The approach has proved better initial trust evaluations are made when feature-behavioral correlations are present. When the correlation are present they allow agents to generalize from trust in individuals to trust in observable features. The model has proven to be robust in interactions and reputation gathering in cases where hidden feature-behavior correlations are present in the trustee agent population. When the team increased the probability of agents leaving, joining or changing identity the model proved to work well, thus, not effecting the effectiveness of the approach. The experiment results have shown that stereotyping can help agents select appropriate reputation providers when society contains several stereotypical biases. When no stereotypical bias exists in the experiment there is no drop in performance. The approach demonstrated how a relatively simple trust model can significantly improve performance. The significance of the presented stereotype model is that it learns from real-valued trust ratings which means it is compatible with any trust model which uses numerical measures for trust.

Chapter 5

Conceptional Model

5.1 Objective

Our presented concept model has been inspired by Burnett *et al.* work [8], where an issue exists of not being able to supply suitable data in demonstrating how features are useful, or related, when estimating a trustee agent's trustworthiness. The main objective of our concept model is to propose an alternative stereotyping approach to improve stereotypical trust in dynamic multi-agent systems. Thus, attempting to improve the average interaction outcome between trustee and truster agents. The attempt is based on creating evidence-based trust models which is applied to situations where there is no information, prior experience, nor information about social relations present about trustee agents.

We present a way to draw stereotypes based on an agents features who has successfully completed a certain task, through its chosen candidate interaction. The outcome can be either successful (1) or unsuccessful (0) and certain features are either present (1) for an agent or absent (0). It must be noted that the stereotype does take into consideration features an agent has which had an unsuccessful outcome for a task. Using this, we will investigate our new approach of observing particular features agents have with successful outcomes and unsuccessful outcomes.

Agent Features and Task Outcome Example

Task	Agent ID	Feature One	Feature Two	Feature Three	Feature Four	Feature Five	Feature Six	Outcome
1	1	1	1	1	0	0	0	1
1	2	0	0	0	1	1	1	0
2	2	0	0	0	1	1	1	1
2	1	1	1	1	0	0	0	0
3	3	1	0	0	1	0	1	1
3	3	1	0	0	1	0	1	0

Table 5.1: Demonstrates 3 different tasks which where attempted by agent 1, 2, and 3. Displays the feature each agent has and the interaction outcome.

It is necessary to note, that table (5.1) displays an agent *Identification Number* (ID). This is solely for the purpose of a better understanding which agent's features are displayed and are not

used in the concept model. If a feature is present in an agent it is defined as 1 and if the feature is not present, as 0. As features passed into the model demonstrate the outcomes, storing it in the concept model would not benefit the stereotyping in any way and be more demanding, storage speaking. In case the same agent is assigned the same task which it has performed in a positive outcome, and so have agents with similar features, the model will know by the agents features that the agent has high trustworthiness when completing the specific task. That is why it is important to keep track the ID of the tasks, which were presented as task features in the framework chapter of this paper.

For our model to be able to predict prior interaction trustworthiness for agents who have just entered the system with no prior experience, we introduce a learning model. The goal is to learn which features are present in successful/unsuccessful outcomes, allowing to predict the outcome of an interaction based on the features present.

Feature Analysis and Prediction Example

Task	Agent ID	Feature One	Feature Two	Feature Three	Feature Four	Feature Five	Feature Six	Outcome
1	1	1	1	1	0	0	0	1
1	2	0	0	0	1	1	1	0
1	3	1	0	1	1	0	0	1
1	4	0	0	1	0	1	1	0
1	5	1	1	1	0	0	1	1
1	6	0	1	0	0	1	1	?

Table 5.2: Demonstrates 1 task attempted by 5 agents. A prediction can be made of the success rate of the 6^{th} agent based on the features it holds.

Table (5.2) demonstrates an example where we can analyze the outcomes and if there is any relevance to the features. One may notice that agents containing *Feature One* and *Feature Three* have all had a successful outcome. The rest of the features might have an effect on the success of the outcome, however, there is not enough sufficient data to prove it. The amount of data provided in this example is surely not vast enough to conclude that having features *One* and *Two* would increase the chance of a successful outcome for *Task One*. However, as this is just an example for a better understanding of the concept model's purpose, based on the data provided, we can state that for agent six there is a higher probability of it having an unsuccessful outcome than a successful one, as it does not have features *One* and *Three*. The concept model will contain such a table, without agent IDs, for each performed task. Overall, our objective is to successfully propose a new concept model of an alternative stereotypical trust model. Where our stereotype performs, at least, as well as the one proposed by Burnett et al. in [8], if not better. To provide a demonstration of proof that the stereotype is capable to generalize in different situations with different tasks in the domain. Furthermore, to present a wide range of future work for this approach, as it could be one step closer to identifying similarities between different task performance.

5.2 Requirements

As we present a concept model of a simulation, there are not many requirements, nevertheless, a brief description is beneficial. We will cover functional and non-functional requirements for our model, with the purpose of the simulation working and producing the desired results.

5.2.1 Functional

Functional requirements are necessary to be achieved for the simulation to function and produce desired results.

- An simulation of a MAS where agents are generated with features.
- Agents are divided into a trustee and truster agent.
- Tasks are generated for the agents.
- Agents placed into ad hoc groups for interactions.
- Agent features are visible to other agents.
- Experience is obtained by agents being able to store previous interaction outcomes.
- Truster agents are able to decide with which trustee agents to interact based on experience.
- Storage of agent features for a particular task split into two groups (successful/unsuccessful).
- A working M5 learning decision tree, capable of predicting future performance.
- Generating a stereotype for each task (the required features for a successful outcome)
- Agents with no prior experience being evaluated by the stereotype.

5.2.2 Non-Functional

Non-functional requirements are not necessary for the functionality of the simulation, but more of an enhancement which could lead to producing better results.

- A probability of agents leaving the MAS.
- A probability of agents randomly failing a task with high trustworthiness.
- Reputation Opinions added to the trust evaluation.
- Modifying stereotype to find connections between an agents performance at different tasks.

5.3 Design

The way our concept is designed is to learn from previous interactions and apply a *prediction of future performance* (Px) to an agent which has no prior experience. This is performed by applying a priory score of trustworthiness to an agent from a scale of 0.0 - 1.0. This simple concept was introduced by Jøsangs [13] beta model displayed in figure (5.1)

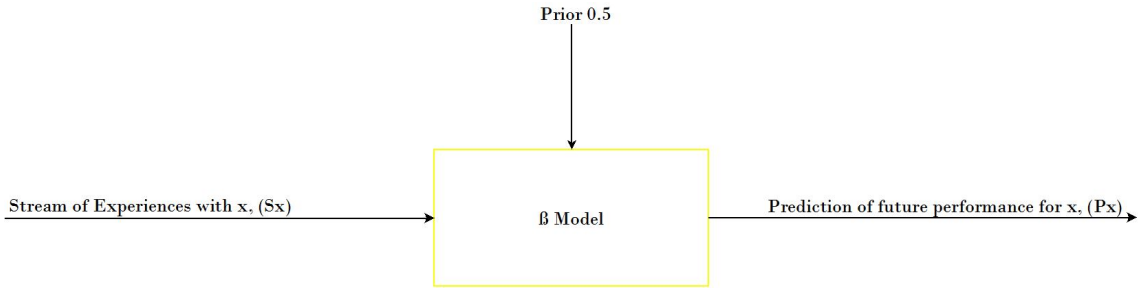


Figure 5.1 – Based on [13], displays a Beta Reputation System.

Figure 5.1 works based on probability, similar to flipping a coin. The prior evaluation is .5 (50%) where there is a (50%) chance the outcome will be successful or unsuccessful. Burnett's work has modified this BRS to have a more accurate prior rating as displayed in figure (5.2)

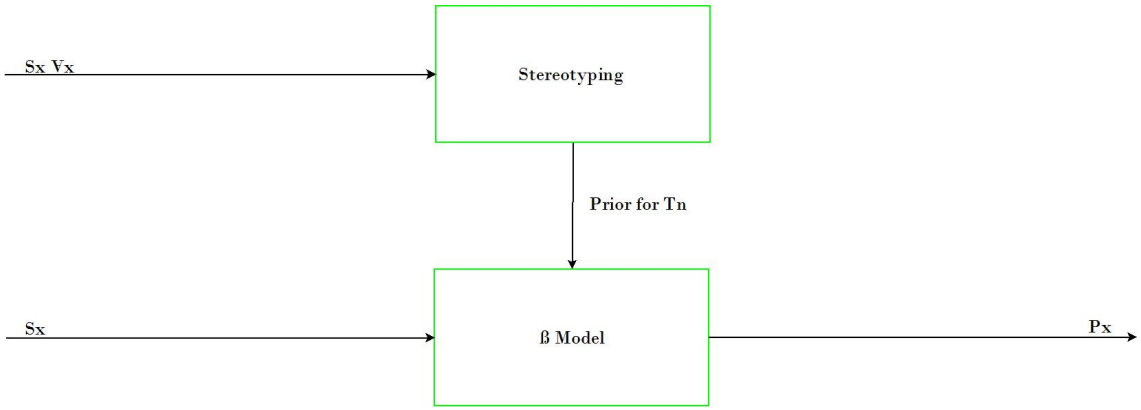


Figure 5.2 – Based on [8], displays prior modification method.

As displayed above, the priori is modified by taking the streams of experiences from every agent to build a stereotype, which then is applied for the specific agent prior attempting a specific task. Where the priori is determined by what stereotype is applied to that particular agent.

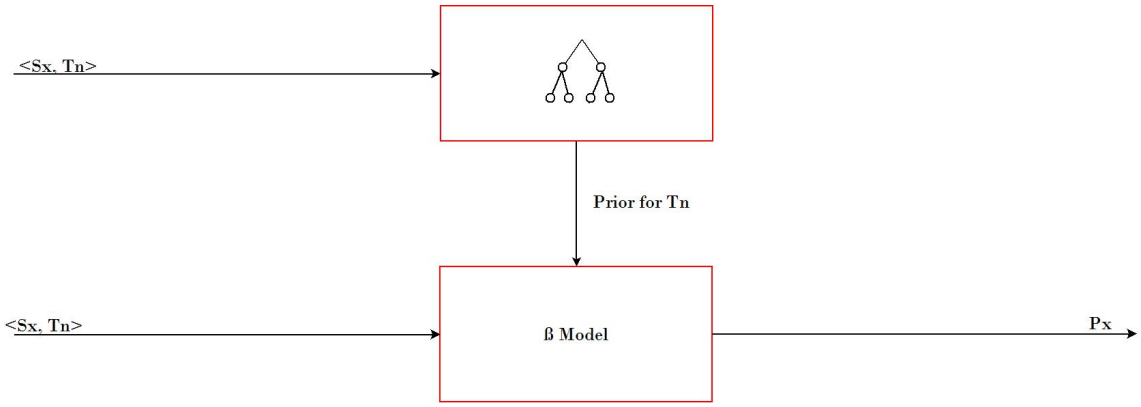


Figure 5.3 – Displays our proposed concept for prior modification.

Figure (5.3) displays our proposed prior modification. Where the stream of experiences for a specific task is run through a M5 tree to form a stereotype to modify the prior for the task. The model works by running through all experiences of agents for tasks τ_n and stores the features into two groups; one for successful outcomes, the second – for unsuccessful.

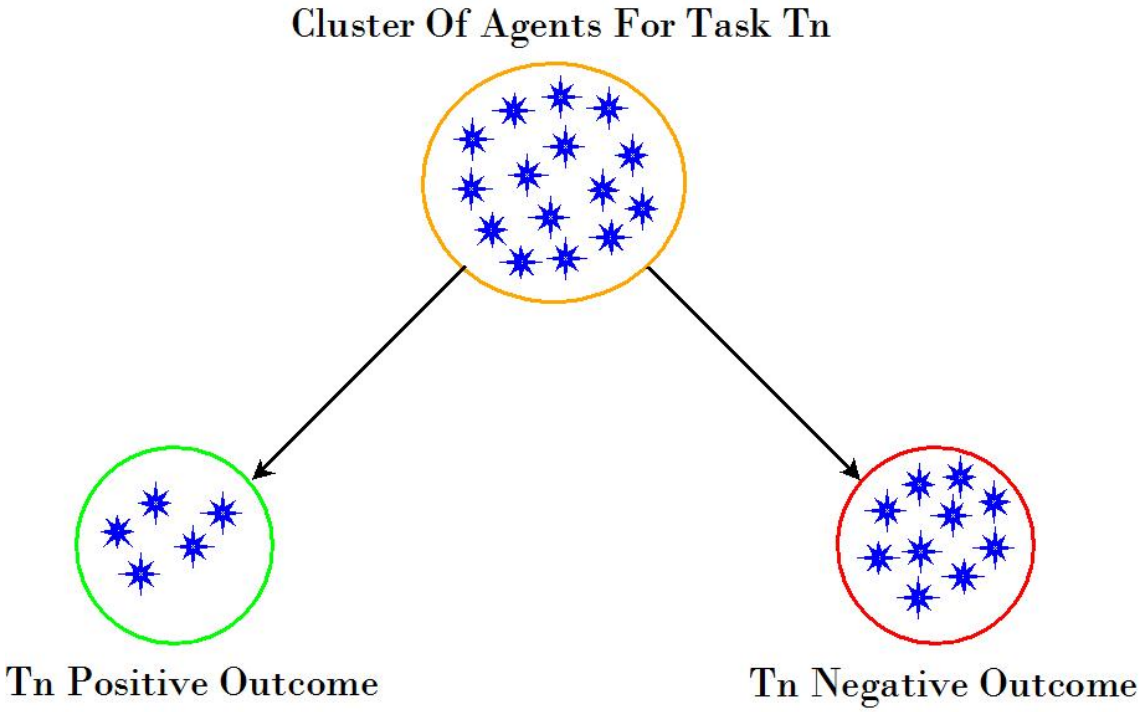


Figure 5.4 – Displays a cluster of agents which have completed task τ_n.

Figure 5.4 displays a group of agents which have been assigned task τ_n, after their completion they are grouped into two clusters. The features of the successful (positive) outcomes are stored, which are then used to predict the prior for that task. The same technique is used for the unsuccessful (negative) outcome features. We store the negative outcome features as they are used to lower priori for agents which do not have any features found in the tasks positive outcome feature section.

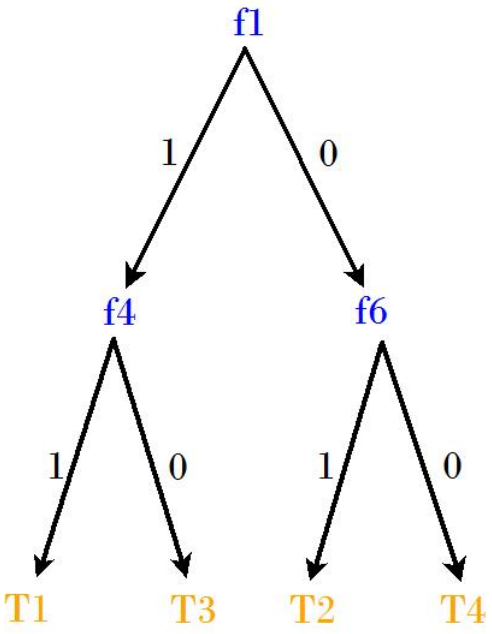


Figure 5.5 – Displays how a stereotype would be allocated to an agent.

Once sufficient evidence has been gathered the M5 decision tree would check the features for a new agent, with no prior experience, to look at its features, as it already knows which features of an agent are needed to be able to complete a task. As demonstrated in figure 5.5 if an agent would have features *One* and *Four*, it would increase its prior for task τ_1 If an agent x is assigned task τ_2 while having features displayed below:

Features of Agent *x*

0	0	1	0	1	1

Table 5.3: Demonstrates the features of agent x

Based on Figure (5.5) as x does not contain feature *One* and has *Six* this would increase the priori of x for τ_2. It is however important to state, that the examples provided are simplified and short for the purpose of easier understanding. As the model would conduct a more complex learning tree and take into account the presence/absence of more than two or three features.

5.4 Methodology

To construct our concept model we will use Burnett's, with his consent, simulation from [8] which can be downloaded online[1]. The first step was to modify the simulation for our purpose, this meant, primarily understanding how the whole simulation works, along with how classes and methods get called and work as a whole. Then, step by step, we construct the established functional requirements.

A conversion of the simulation was required as Burnett's model focuses on agents and their features. For our goal we added tasks, and represented them as features described in chapter 3. We fully removed opinion representation, as our concept focus more on a socio-cognitive approach where the stereotype is produced from observable features. We have added additional tasks for our agents which are represented as feature tasks. These tasks were then allocated for each agent to complete. The idea is to have a certain value of *cost* for each task to represent the complexity of the task. Each present feature would then contain a certain amount of *currency* which would be paid for the interaction, despite the outcome of the task completion. This would force agent decision making. A fully modified M5 decision learning tree would be added to form the stereotypes for each agent and task. This is the part where difficulties have presented themselves.

During our methodology, we have used a number of helpful tools for implementation of the concept model. We have used Java as our programming language. Eclipse IDE was used to fully import and modify the simulation which came with imported libraries. Git-hub was used as a backup, as we were dealing with large quantities of code. As our data would be represented in a *.csv* format, we used *Microsoft Excel* to be able to open and observe generated data, such as agent and task features. For further data analysis a *Weka* GUI was used.

[1] https://github.com/chrisburnett/Stereos

5.4.1 Alternatives

There are a number of possibilities that can be used to test and implement our concept model in a simulation. We have chosen to work on top of Burnett's stereotype simulation as that would allow to acquire in depth knowledge of how the simulation works. We could have attempted to create our own simulation from scratch, however, curiosity and interest in the subject has made us to choose otherwise. An alternative IDE could have been used – *Netbeans*, instead of *Eclipse*, however, being of use to the interface and shortcuts the eclipse provided an opportunity to choose otherwise. An alternative language could have been used to designed our simulation – *Ruby*. However, as we chose to work on top of the simulation which has been written in *Java*, *Ruby* has been eliminated. For data backup *Gitlab* was an option, however, *Github* has fully fulfilled our needs. Overall, there are many alternatives that could have been used, nevertheless, the programs used for this paper did not hinder the performance or outcome in any way.

5.4.2 Problems and Difficulties

The simulation we worked on was difficult to assess, since it is a simulation for a PhD thesis. It contains more classes and methods for testing various behaviors and bias, along with testing two stereotypical models presented in chapter 4. Observing and learning how each line of code works was time consuming. Removal of several unneeded classes would require the removal of methods/lines of code through-out the system which took numerous hours.

Due to the difficulties encountered and the timing constraints placed, the initial plan of constructing a working agent simulation with our presented concept model has not been achieved. Yet, a more extensive knowledge in Java has been acquired, as the code used in Burnnett's simulation is written in advanced techniques. Analysis of already established codes previously written by others has been significantly improved and mastered. To evaluate our concept model we have provided an alternative testing method presented below.

Chapter 6

Evaluation

6.1 Overcoming Problems and Difficulties

As we were unable to design a simulation with our concept stereotype model, we used another approach to test the feasibility of our proposed concept. We have designed a compact generator which generates agents, tasks and trustworthiness. The soul purpose of the model is to test if our proposed concept model would actually work and be worthy of implementation into an agent simulation.

We have tested our concept model by scientifically generating trustworthiness of each agent from a community of 400 gents. The way we represent the trustworthiness of each agent is by using the *Natural Distribution* (ND) by providing a mean value and a standard deviation. One can assume that during testing the trustworthiness (ND) has been acquired through interactions and task performance. Such values would have been generated through simulation interactions, and left within the community as their reputation value of trustworthiness. As we have proposed a conceptual model the present experiment will demonstrate if it is possible to distinguish between tasks and if it is possible to predict an agents performance in a particular task.

6.2 Testing Description

Once our generator has been complete, we have generated 2 profiles of agents. Each profile held 200 agents with a particular bend to having certain features over others. In each profile two tasks were assigned for the agents to 'complete'. What we mean by completing a task, is having a particular task assigned to 100 agents in each profile. We assume that the agents have completed the task, either, successful or unsuccessfully and have provided the agents with a value which represents their trustworthiness if they would be assigned to complete that specific task again.

Profiles of Agents

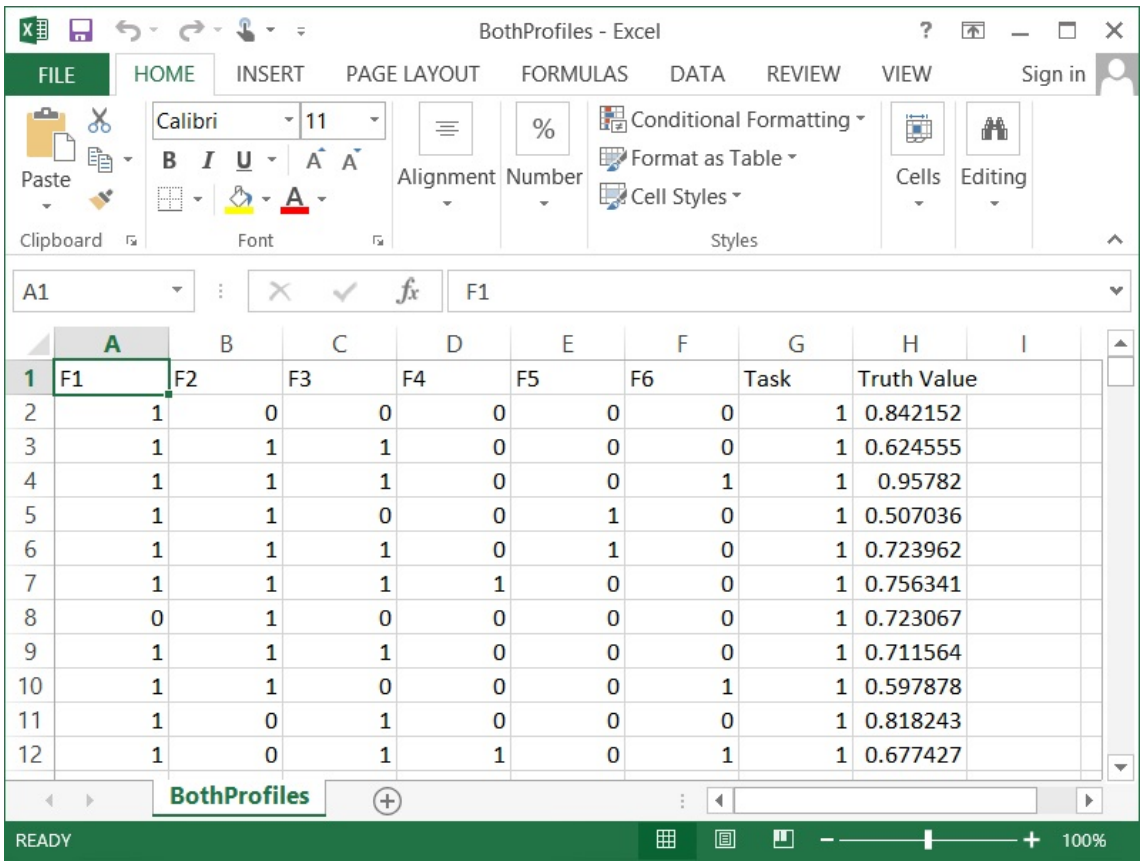
Profile	Task	Description	Mean	StDev	*f1*	*f2*	*f3*	*f4*	*f5*	*f6*
1	1	Often Good	0.7	0.15	80%	80%	80%	20%	20%	20%
1	2	Often Bad	0.3	0.15	80%	80%	80%	20%	20%	20%
2	1	Often Bad	0.3	0.15	20%	20%	20%	80%	80%	80%
2	2	Often Good	0.7	0.15	20%	20%	20%	80%	80%	80%

Table 6.1: Demonstrates 2 profiles of agents, their assigned Natural Distribution, and the likelyhood of feature inheritance.

As table (6.1) demonstrates, we have assigned 100 agents a particular task. For the sake of the experiment we assume that these simulated tasks are different, in terms of requirements, from one another. We have designed the generation to fulfill the outputs of the concept model simulation. The outputs are generated in such way that agents which perform well, in terms of having a successful outcome, on one task are less likely to perform well on the other task.

We distinguish the outcomes by applying two different normal distributions to the same profile. Where 100 of the agents have a ND where the generated trustworthiness value is often good, and the rest, (100 agents) are given a ND which is often bad. For the *often good* ND we assign a mean value of .7 and a standard deviation of .15. The ND generates values around the defined mean, where the standard deviation tells the generation how far the generated value can go under or above the given mean. In other words, approximately 68% of the values generated from a ND are within one standard deviation away from the mean, about 95% of the generated values lie within the two standard deviations, and approximately 98.7% lie within three standard deviations – called the *3-sigma rule*.

The features are represented as $f_1 \ldots f_6$. The percentages state the probability of agents to receive a particular feature. For testing purposes we have assigned an 80% chance of an agent to receive features 1 - 3 and a 20% probability chance of receiving features 4-6 for profile 1. Profile 2 contains the same probability only for opposite features. Do note that there still exists a chance of an agent containing all features or neither.



	A	B	C	D	E	F	G	H
1	F1	F2	F3	F4	F5	F6	Task	Truth Value
2	1	0	0	0	0	0	1	0.842152
3	1	1	1	0	0	0	1	0.624555
4	1	1	1	0	0	1	1	0.95782
5	1	1	0	0	1	0	1	0.507036
6	1	1	1	0	1	0	1	0.723962
7	1	1	1	1	0	0	1	0.756341
8	0	1	0	0	0	0	1	0.723067
9	1	1	1	0	0	0	1	0.711564
10	1	1	0	0	0	1	1	0.597878
11	1	0	1	0	0	0	1	0.818243
12	1	0	1	1	0	1	1	0.677427

Figure 6.1 – Displays generated features, tasks, and trustworthiness value.

As displayed in figure (6.1), we see an example, of a generated profile of agents. The figure displays the beginning of the generated list, therefore we are looking at profile 1. One may notice that agent 3 has the first three features, but a trustworthiness value of 0.5. This demonstrates the realism of a real simulation, as there can be no correlation between tasks at all, as it has not yet been proven by researchers. Even though we are generating 400 agents for our experiment, they are really representing a community of 200 agents. The number is doubled as for each profile we use the same generated features, with a different trustworthiness for each task. This is for the purpose of demonstrating if our concept model would be able to distinguish between different tasks, along with two completely different tasks which require different features to complete.

The testing generator will generate different features for values and trustworthiness for every agent each time it is run. It is important to note, when generating a data set one should *copy/move* the data set from its generated location, as if a new data set is generated it will overwrite the previous one. To scientifically evaluate the results, we have used more than one data set, demonstrated in the upcoming section.

6.3 Results

To have significant evidence to provide data to support the effectiveness of our trust model, we have produced 30 data sets. 10 data sets were generated by simply running the program and placing the generated .csv file into a separated folder ten times. The program was set with agent profile parameters demonstrated in table (6.1). Another 10 data sets were generated in a the same way as the first 10. These data sets were used as a learning set for the Weka GUI. The last 10 generated data sets were modified, where task was defined as 0 for each data set. In this way, the M5 decision learning tree would not know weather an agent is assigned task 1 or 2 and would have to predict, based on its features, what task should be assigned to the particular agent. Each data set was inserted in to Weka and classified with the M5 tree. Results are displayed below:

Cross-Validation of Data With Tasks Present

Data Set	Correlation coefficient	Mean Absolute Error	Root Mean Squared Error	Folds
1	0.7926	0.1186	0.1438	10
2	0.8	0.1117	0.1407	10
3	0.7874	0.1196	0.1471	10
4	0.7964	0.1184	0.1458	10
5	0.8091	0.1147	0.1408	10
6	0.7952	0.1156	0.1403	10
7	0.8036	0.1215	0.1473	10
8	0.8246	0.1145	0.1399	10
9	0.8186	0.1183	0.1446	10
10	0.7797	0.1206	0.1492	10
Mean	**0.80072**	**0.11735**	**0.14395**	

Table 6.2: Demonstrates the results of 10 data sets in Cross-Validation of the M5 decision learning tree.

Table (6.2) represents the gathered results from our first test of 10 data sets. The *correlation coefficient* the linear dependency of the data. In other words, how close the x and y axis are. Where 1 and -1 demonstrate a linear relation and 0 demonstrate no linear relation. As in the data sets we have provided the M5 decision tree with tasks, it demonstrates that there is a linear relation with the trustworthiness value and the task assigned. *The mean absolute error* measures the average accuracy for the continuous variables and *root mean squared error* measures the average magnitude of the error size. Logically, the higher the values – the higher amount of errors persist in the data sets. The number of folds represent the testing data set. With 10 folds 90% of the data set was used as a training set and 10% as a test set. However, tasks were still provided.

By analyzing the data one may notice that the mean absolute error average of the 10 data sets is 0.11735. This demonstrates that there is a slight amount of inaccuracy when classifying the data with an M5 decision tree. However, the amount of inaccuracy is small enough to have no impact when adjusting the prior for agents wanting to do a particular task. For example, the inaccuracy can indicate of the model adjusting the prior by .25 rather than .26 which should have no effect during agent delegations. The *root mean squared error*, on average, is 0.14395. This means the margin of error is low as well. This test was to analyze the performance of the M5 tree on our generated data sets. The results are satisfactory, therefore, we move on to our next test results.

Test Data With No Tasks Present

Training Set	Test Set	Correlation coefficient	Mean Absolute Error	Root Mean Squared Error
1	T_1	-0.0539	0.2567	0.3102
2	T_2	0.031	0.2338	0.2844
3	T_3	-0.0111	0.2353	0.2899
4	T_4	-0.057	0.2481	0.3013
5	T_5	0.0574	0.2373	0.2873
6	T_6	-0.022	0.2315	0.2864
7	T_7	0.0456	0.2364	0.2863
8	T_8	-0.0417	0.2437	0.3003
9	T_9	-0.0217	0.2477	0.3036
10	T_{10}	0.0075	0.2435	0.2925
Mean		**-0.00659**	**0.2414**	**0.29422**

Table 6.3: Demonstrates the results with a training sets containing tasks, and a test sets with no defined tasks

Table (6.3) displays the results of our 20 generated data sets, from which 10 had their tasks defined as 0. We see that the data was interpreted correctly, as the average correlation coefficient is approximately zero, meaning that there is no correlation between the x and y axis. This is true, as we have not provided the tree with tasks of agents. We see that the mean and root mean have increased. This is explainable by have the test set with no tasks assigned to agents, and therefore there is a slighter chance of the M5 tree to produce a more inaccurate prediction. However, we see that the inaccuracy has only increased by (0.2414 - 0.11735) 0.12405. Which means, the M5 tree is able to predict which agent should do which task based on their features. Further visualization will help understand the outcomes of our results.

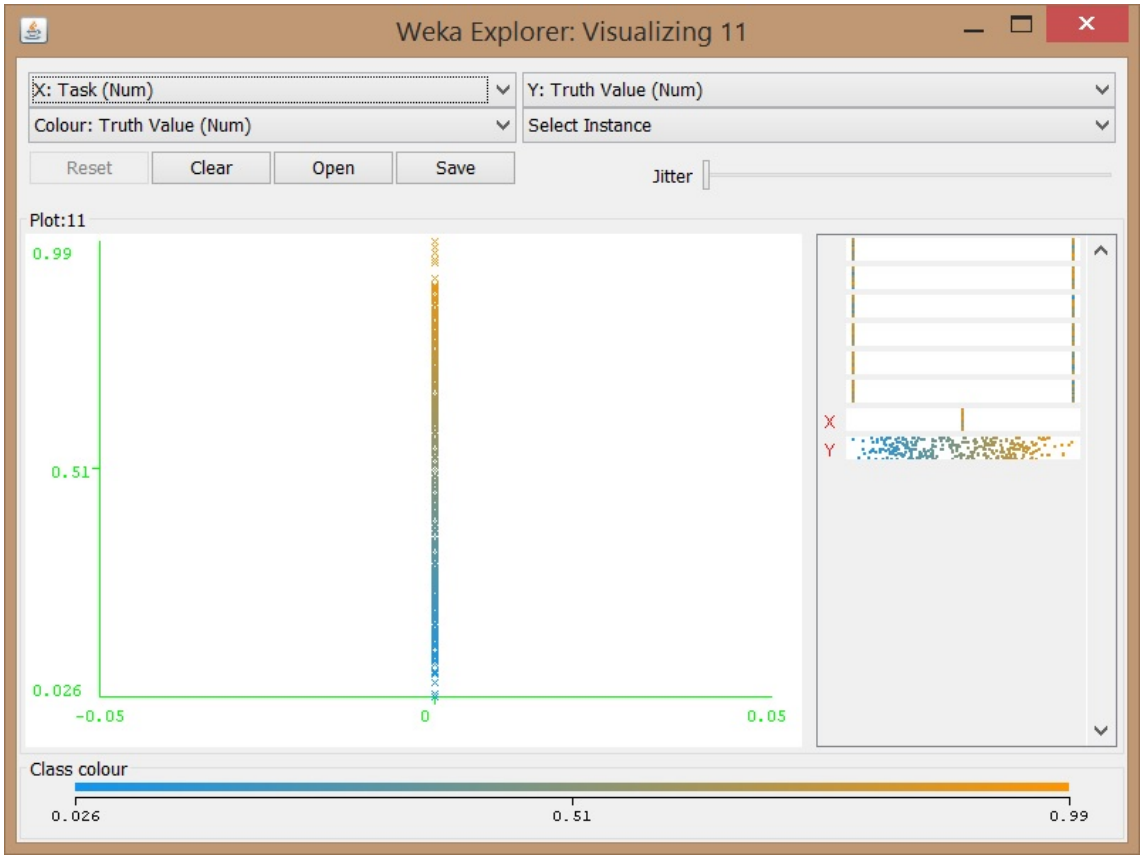


Figure 6.2 – Displays agents by their trustworthiness with no task.

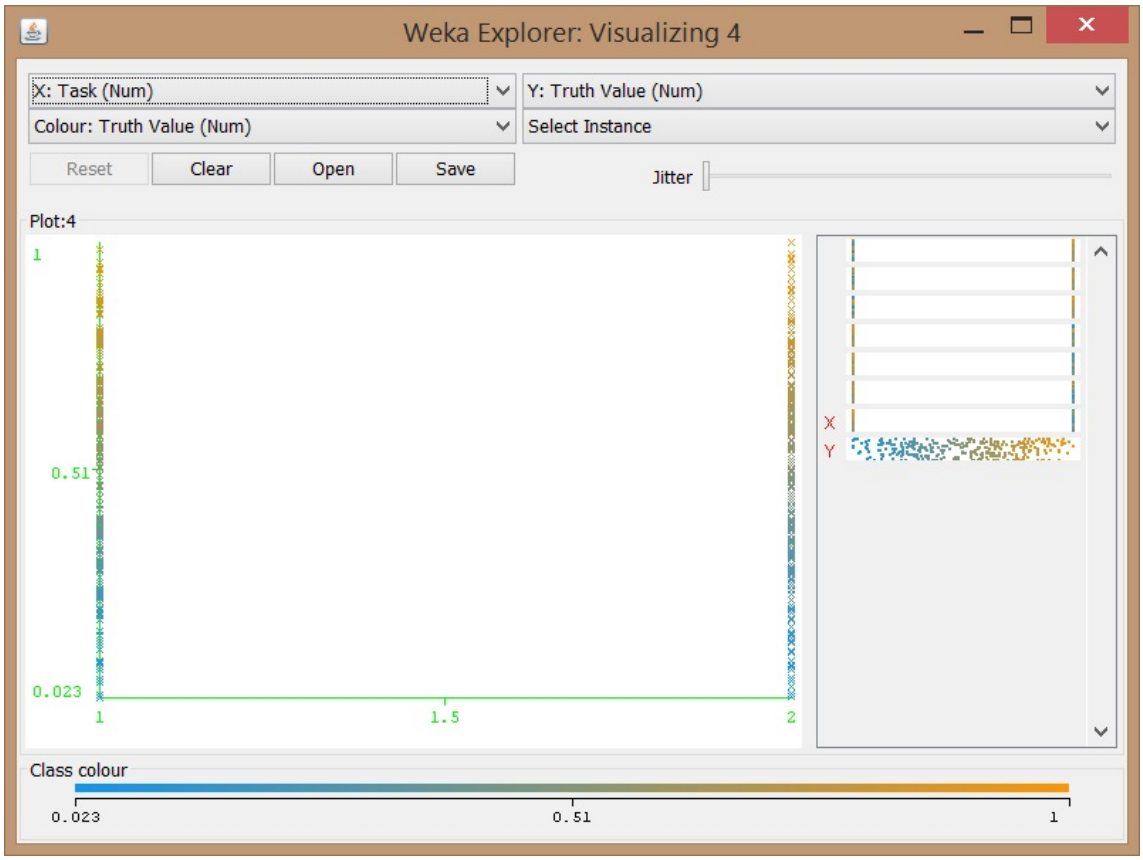


Figure 6.3 – Displays our first experiment results of classifying agents with a given task.

The above figures (6.2 - 6.3) demonstrate a visualization of agents with their trust values from our experiments. Figure 6.2 demonstrates how agents would perform a particular task, without being applied a stereotype nor having any prior experience about the trustworthiness of other agents. The blue colour represents low[1] trustworthiness and orange represents high[2] trustworthiness. Figure 6.3 demonstrates the effectiveness of our proposed concept mode, as with the provided data set tasks; it was able to categorize them, no matter the trust value. This means, that for our concept model the stereotyping mechanism would be able to store unsuccessful outcome features along with successful for better prior adjustment. However, as tasks were provided in the data set displayed by the figure, we have yet to compare it to our second experiment which is displayed in figure (6.4) below:

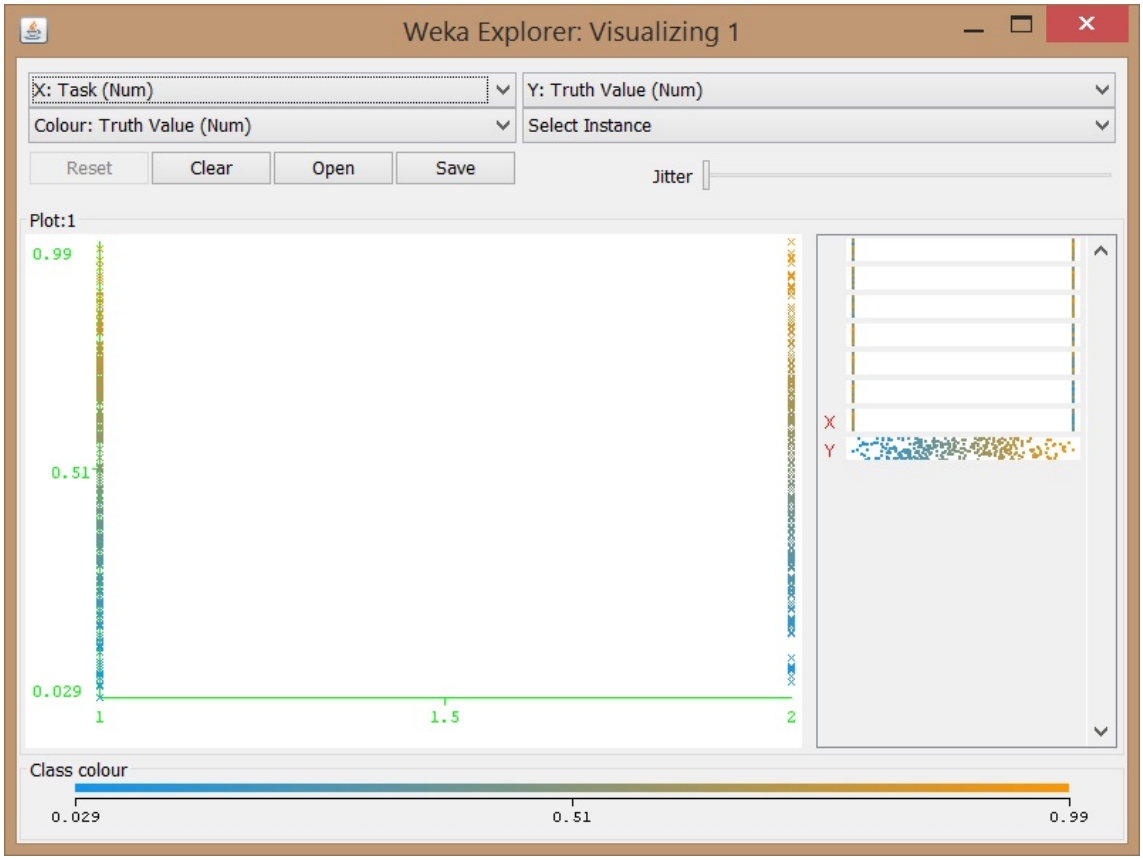


Figure 6.4 – Displays our second experiment results of classifying agents with no tasks..

Figures (6.3 & 6.4) are really similar. This is because the M5 tree is able to construct an effective learning tree and assign agents to their correct task based on their features. Leading the concept to be highly dynamic and responsive to situations where generalization is key.

[1] Most likely the task attempt was unsuccessful
[2] Most likely a successful task outcome

6.4 Hypothesis Evaluation

- *Hypothesis 1* – If there are recognizable behavioral differences between tasks stereotyping would be possible.

- *Hypothesis 2* – If the M5 decision learning tree is able to successfully classify agent features with tasks, it will be able to do so without being provided agent tasks.

Even though we were unable to provide a working stereotyping simulation, we have proven that our concept model can be implemented into a simulation and be able to recognize behavioral differences between tasks. It actually turns out to be an interesting concept how well the M5 tree is able assign agents to their particular task with the trust values they contain. Before a task would be even attempted an agent could predict the outcome – therefore lean towards the most trustful trustee agent. As our main goal was to propose an alternative stereotyping approach – we have succeeded. The presented results demonstrate, that in fact, this approach, which focuses on tasks, is able to generalize to situations and distinguish between tasks. It is important to note, that it is not a given that every MAS uses a value based representation of trust. Furthermore features may be represented in a totally different fashion. But we have proven that for real-value based simulations this concept is responsive and is able to adapt. Based on our results we can see, through a simulation, that tasks may contain behavioral differences. Along with proving hypothesis 2 where our concept stereotype would be able to generalize to situations and differentiate tasks in a domain.

Chapter 7

Conclusion

7.1 Discussion

Based on our results we can see that the provided concept model is able to generalize to situations and different tasks in a domain. Providing such evidence brings up a hypothesis that there may be relationships between tasks. Allowing one to see if an agent is good at a particular task, it could also be good at a task which has a relevant demand, such as web development and programming tasks. Thus, allowing in the future to predict the performance of an agent for a task which it has no prior evidence to.

We have discussed a number of trust models in this paper, however, there are some which address similar interests as we have in this paper. The FIRE [24] systems attempts a *role-based trust* to specific interests. In their findings they state that there might be a presence of trust even when there is no evidence available. This is an interesting concept, as it might have an impact of the concept stereotype, as it might allocate a low prior to an agent who would actually always perform that particular task with a successful outcome.

When implementing our proposed concept model, one must firmly decide weather the implementation of reputation provision should be implemented. As the general issue of deceptive reputation providers in, [47], [50] have addressed learning the trustworthiness of a reputation provider. It might be an interesting concept of combining such approach into the concept model. However, it may not be feasible to construct models of individual trustworthiness providers. As the turnover among reputation providers may be high. For this purpose we have not presented a reputation opinion approach to our concept. however, this could be implemented in the future work of the concept.

7.2 Conclusion

The foundations of trust formation in highly dynamic human societies contain an initial risk. The involvement of risk within a MAS hinders the performance of the system. We have provided with an alternative approach of formulating stereotypes to reduce the level of risk and improve performance. We have discussed numerous works on the subject and discussed that, much like humans, highly dynamic virtual societies contain a barrier when it comes to the formation of trust. We have analyzed, in depth, trust evaluations when feature-behavioral correlations are present [8] and presented an alternative approach of evaluating trust.

We have provided an alternative way an M5 tree could be used for learning the trust value of agents and being able to predict prior for an agent, which can be either high or low. As the concept

model knows the features of low trustworthiness values. We provided a concept how the features can be used to find correlations between tasks for both, high and low trustworthiness values. Our concept model has demonstrated that it is capable of being implemented into a simulation. Receiving test results which demonstrate that we can recognize behavioral differences between tasks has opened numerous possibilities for the concept model.

We have simulated a part of our stereotypical multi-agent simulation. Where a M5 learning decision tree was used to learn about agents and set trustworthiness predictions. We presented similar accuracy in predictions when tasks are not present through feature-behavioral correlations present. This was achieved by allowing the model to generalize from individual agent features. Our model has proven to be robust in situations where only one task is present; no tasks are assigned to agents; and multiple tasks are assigned to agent profiles. We have presented an alternative stereotyping concept which can be constructed with a trust model which is based upon the simple theory of probability in order to improve performance. However, it is important to note, that the concept model might behave differently in a simulated ongoing environment, as we have designed it to learn from real-valued trust ratings, if the concept is interpreted into another environment which uses a different measurement unit for trust.

7.3 Future Work

We would firstly propose a modification and integration of our generator into an agent simulation. Then, based on the concept model the simulation would be constructed and further tests could be done. Once complete, a series of tests could determine weather the simulation would produce similar results to our findings. Furthermore, additional costs could be added for task completion which would force the agents to delegate in more *selfish* manner.

Stereotypical bias can be introduced into the simulation and be observed. Agents may have behavioral correlations with their features which could impact the way they would perceive their interaction partners. Similarly, unobserved features could be introduced. As agents in MAS may attempt to hide their feature vectors or misinterpret the features of others; our proposed concept model is based on observable features. It would be interesting to examine how robust the model is to such behavior. To further test the robustness of stereotype one could create a chance of an agent joining and leaving the system. A test could be conducted with agents being highly likely to leave and join the community. A behavioral action of the methods could also be introduced which would force a particular group of agents to act in a undesired way, meaning to hide their features or to promise, but not deliver their promises.

We offer the implementation of the concept model to support reputation opinions. The effect of these opinions could be observed and compared if it effects the efficiency of the concept stereotype. The purpose of adding reputation opinions is to allow agents to pass through their opinions to other agents which have no prior experience. This would generate two priories for an agent. Thus, it might be difficult to combine the prior ratings without them overlapping or contradicting. However, the effectiveness could be tested and validated by using an approach proposed in [39].

Furthermore, the *Agent Reputation and Trust* testbed [18] could be used to create an environment where agents have to delegate among themselves to buy/sell artwork. The concept simulation

could be implemented into this testbed and would demonstrate how agents with no prior experience would be able to identify with which agents not to interact. Furthermore a *real* test could be performed to test the concept model, which would include exposing the stereotype to real-life data and observing if there are any improvements. One may also consider proposing a working concept stereotype to the international conference on autonomous agents and multi-agent systems (AAMAS).

A significantly different approach could be created by attempting to create a hybrid model. We propose a combination of two approaches can be combined to predict prior for an agent. As displayed in figure (7.1), we propose to combine Burnett's *et al.* stereotyping approach combined in some manner with our stereotyping concept.

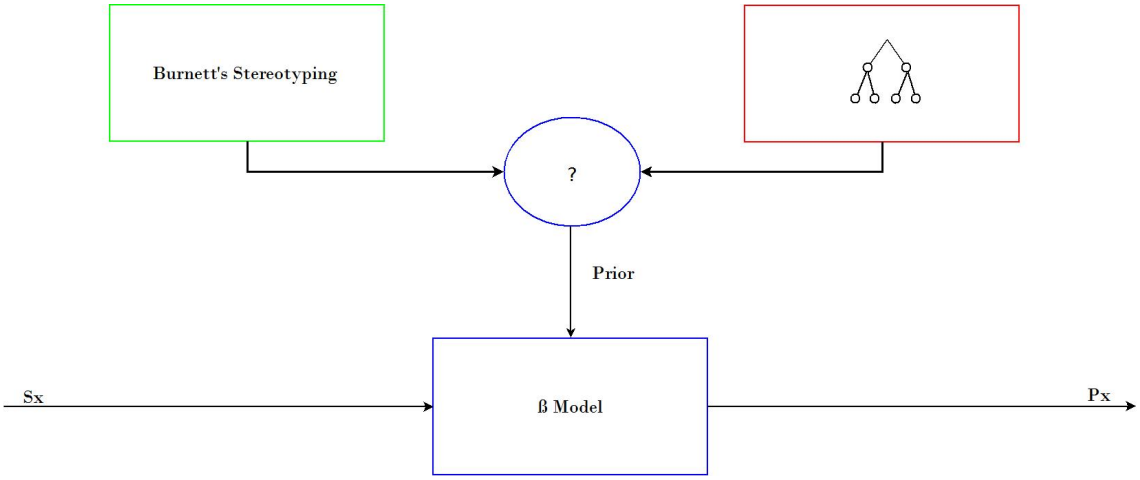


Figure 7.1 – A concept combination of Burnett's and our stereotyping models.

This could be an interesting model, as it provides feature-behavior correlations combined with task correlations. This would allow to combine reputation trust to be combined with task features. Thus the intakes of an agent's trust evaluation, reputation trust, and relationships between tasks could provide the stereotype with more information and could increase the prediction of the stereotype. However, one must think about how to combine these priors together as they may overlap or open up an opportunity of agents to behave in such way to purposely increase their prior by hiding certain features or completing only one task. One must also be careful on the implementation method, as using two models could require more resources and be hard to implement in a real-life system. Our concept model has provided a handful of interesting opportunities of alternative ways to form stereotypical trust.

Bibliography

[1] Youcef Aklouf and Habiba Drias. Designing a generic marketplace architecture using multi-agent based technology. *Int. Arab J. Inf. Technol.*, 3(3):249–255, 2006.

[2] John Robert Anderson, Ryszard Stanisław Michalski, Jaime Guillermo Carbonell, and Tom Michael Mitchell. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

[3] Ronald Ashri, Sarvapali D Ramchurn, Jordi Sabater, Michael Luck, and Nicholas R Jennings. Trust evaluation through relationship analysis. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1005–1011. ACM, 2005.

[4] K Suzanne Barber and Joonoo Kim. Soft security: Isolating unreliable agents from society. In *Trust, Reputation, and Security: Theories and Practice*, pages 224–233. Springer, 2003.

[5] Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[6] Ken Binmore and Bruce Linster. *Fun and games: A text on game theory*, volume 21. DC Heath Lexington, Mass, 1992.

[7] Chris Burnett, Timothy J Norman, and Katia Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 241–248. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[8] Chris Burnett, Timothy J Norman, and Katia Sycara. Stereotypical trust and bias in dynamic multiagent systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(2):26, 2013.

[9] Maria Caridi and Andrea Sianesi. Multi-agent systems in production planning and control: An application to the scheduling of mixed-model assembly lines. *International Journal of Production Economics*, 68(1):29–42, 2000.

[10] Álvaro Carrera, Carlos A Iglesias, Javier García-Algarra, and Dušan Kolařík. A real-life application of multi-agent systems for fault diagnosis in the provision of an internet business service. *Journal of Network and Computer Applications*, 37:146–154, 2014.

[11] Cristiano Castelfranchi and Rino Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 72–79. IEEE, 1998.

[12] Cristiano Castelfranchi, Rino Falcone, and Giovanni Pezzulo. Trust in information sources as a source for trust: a fuzzy approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 89–96. ACM, 2003.

[13] Bled Electronic Commerce, Audun Jøsang, and Roslan Ismail. The beta reputation system. In *In Proceedings of the 15th Bled Electronic Commerce Conference*. Citeseer, 2002.

[14] Rajdeep K Dash, Nicholas R Jennings, and David C Parkes. Computational-mechanism design: A call to arms. *Intelligent Systems, IEEE*, 18(6):40–47, 2003.

[15] Rino Falcone and Cristiano Castelfranchi. Social trust: A cognitive approach. In *Trust and deception in virtual societies*, pages 55–90. Springer, 2001.

[16] Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes, and Ian H Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

[17] Karen K Fullam and K Suzanne Barber. Dynamically learning sources of trust information: experience vs. reputation. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 164. ACM, 2007.

[18] Karen K Fullam, Tomas B Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K Suzanne Barber, Jeffrey S Rosenschein, Laurent Vercouter, and Marco Voss. A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 512–518. ACM, 2005.

[19] Diego Gambetta. Trust: Making and breaking cooperative relations. 1988.

[20] Nathan Griffiths. Task delegation using experience-based multi-dimensional trust. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 489–496. ACM, 2005.

[21] Ramón Hermoso, Holger Billhardt, and Sascha Ossowski. Role evolution in open multi-agent systems as an information source for trust. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 217–224. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[22] James L Hilton and William Von Hippel. Stereotypes. *Annual review of psychology*, 47(1):237–271, 1996.

[23] Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. Certified reputation: how an agent can trust a stranger. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1217–1224. ACM, 2006.

[24] Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, 2006.

[25] WENG Jianshu, MIAO Chunyan, and GOH Angela. An entropy-based approach to protecting rating systems from unfair testimonies. *IEICE TRANSACTIONS on Information and Systems*, 89(9):2502–2511, 2006.

[26] Catholijn M Jonker and Jan Treur. Formal analysis of models for the dynamics of trust based on experiences. In *Multi-Agent System Engineering*, pages 221–231. Springer, 1999.

[27] Audun Jøsang and Jochen Haller. Dirichlet reputation systems. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 112–119. IEEE, 2007.

[28] Audun Jøsang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In *Proceedings of the 29th Australasian Computer Science Conference-Volume 48*, pages 85–94. Australian Computer Society, Inc., 2006.

[29] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[30] Audun Jøsang and Simon Pope. Semantic constraints for trust transitivity. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*, pages 59–68. Australian Computer Society, Inc., 2005.

[31] Martin J Kollingbaum and Timothy J Norman. Supervised interaction: creating a web of trust for contracting agents in electronic environments. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 272–279. ACM, 2002.

[32] Roderick M Kramer and Tom R Tyler. *Trust in organizations: Frontiers of theory and research*. Sage Publications, 1995.

[33] Ugur Kuter and Jennifer Golbeck. Using probabilistic confidence models for trust inference in web-based social networks. *ACM Transactions on Internet Technology (TOIT)*, 10(2):8, 2010.

[34] Siyuan Liu, Jie Zhang, Chunyan Miao, Yin-Leng Theng, and Alex C Kot. iclub: An integrated clustering-based approach to improve the robustness of reputation systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1151–1152. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[35] Yuhong Liu and Yan Sun. Anomaly detection in feedback-based reputation systems through temporal and correlation analysis. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 65–72. IEEE, 2010.

[36] David JC MacKay. *Information theory, inference, and learning algorithms*, volume 7. Citeseer, 2003.

[37] Paolo Massa and Paolo Avesani. Controversial users demand local trust metrics: An experimental study on epinions. com community. In *Proceedings of the National Conference on artificial Intelligence*, volume 20, page 121. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[38] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2431–2439. IEEE, 2002.

[39] Chris Burnett Timothy J Norman and Katia Sycara. Trust decision-making in multi-agent systems. 2011.

[40] ES Page. Continuous inspection schemes. *Biometrika*, pages 100–115, 1954.

[41] John R Quinlan et al. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. Singapore, 1992.

[42] Sarvapali D Ramchurn, Dong Huynh, and Nicholas R Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(01):1–25, 2004.

[43] Stuart Russell and Peter Norvig. Russell, stuart and norvig, peter. In *Artificial Intelligence: A Modern Approach (3rd Edition)*, pages 653–664. Prentice Hall, 2009.

[44] Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 475–482. ACM, 2002.

[45] Tuomas W Sandholm. Distributed rational decision making. *Multiagent systems: a modern approach to distributed artificial intelligence*, pages 201–258, 1999.

[46] Michael Schillo, Petra Funk, and Michael Rovatsos. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*, 14(8):825–848, 2000.

[47] Murat Şensoy, Jie Zhang, Pinar Yolum, and Robin Cohen. Poyraz: Context-aware service selection under deception. *Computational Intelligence*, 25(4):335–366, 2009.

[48] Jianqiang Shi, Gregor v Bochmann, and Carlisle Adams. Dealing with recommendations in a statistical trust model. In *Proceedings of AAMAS Workshop on Trust in Agent Societies*, pages 144–155, 2005.

[49] Marsh Stephen. Formalising trust as a computational concept. *Ph. n dissertation. University of Stirling, scotland*, 1994.

[50] WT Luke Teacy, Jigar Patel, Nicholas R Jennings, and Michael Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.

[51] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (60th Anniversary Commemorative Edition)*. Princeton university press, 2007.

[52] Yonghong Wang, Chung-Wei Hang, and Munindar P Singh. A probabilistic approach for maintaining trust based on evidence. *Journal of Artificial Intelligence Research*, 40(1):221–267, 2011.

[53] Yonghong Wang and Munindar P Singh. Formal trust model for multiagent systems. In *IJCAI*, volume 7, pages 1551–1556, 2007.

[54] Andrew Whitby, Audun Jøsang, and Jadwiga Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proc. 7th Int. Workshop on Trust in Agent Societies*, volume 6, 2004.

[55] Cort J Willmott, Steven G Ackleson, Robert E Davis, Johannes J Feddema, Katherine M Klink, David R Legates, James O'donnell, and Clinton M Rowe. Statistics for the evaluation and comparison of models. *Journal of Geophysical Research: Oceans (1978–2012)*, 90(C5):8995–9005, 1985.

[56] Han Yu, Zhiqi Shen, CYRIL Leung, Chunyan Miao, and VICTOR R Lesser. A survey of multi-agent trust management systems. *Access, IEEE*, 1:35–50, 2013.

Appendix A

Maintenance Manual

A.1 Software Requirements

To be able to run the program presented in this paper, one will need a handful of tools. For starters an IDE – *Eclipse* or *Netbeans* for example.

In case one is using a 'restricted' access account, which is not logged in as administrator one must first check if he/she has access to "C:/users/*userName*/". This is necessary for the program to run without any errors. As the *FileWriter* method uses that location for generating .csv files. If there are restrictions to "C:/users/*userName*/, one must modify lines 224, 227, and 230 of the Profile.java file to an accessible location for Java. A .csv format compatible GUI is needed in the interest of better inspecting generated data, such as *Microsoft Excel* or *Libre Office*. To further analyze the data an appropriate analyzing GUI is needed such as *Weka* or *R*.

A.2 List of Packages Needed

- Commons-math3-30.jar[1]

A.3 Installation Instructions

Once a user has downloaded the .tar file and extracted it, a folder will be created in the user's desired location. To be able to run the program one should:

A.3.1 Using IDE

1. Open IDE
2. Import....
3. Select Existing Projects into Workspace
4. Click Next
5. Go to extracted folder directory
6. Select extracted folder called "*project*"
7. Click Ok
8. Make sure the project is selected below

[1] http://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/distribution/NormalDistribution.html

9. Click Finish

The imported project should show up in the *Package Explorer* of the IDE. You can expand the folders to observe the classes. Profile.java is the main class which generates the results. A user manual is provided in Appendix B.

A.4 Future Adaptations

The design of the program, has been created with future adaptations. In the future a weka can be integrated inside the program using the weka package[2]. The way agents are created can be rewritten into a .text file, which would introduce an easier method of changing variables for the agents. Where the user would modify a line in the text file such as:

```
//  [EFFORT_LEVELS=(ID,MEAN,STAND-DEV,COST)]
p1,100,40:[(f1,1.0),(f2,1.0),(f3,1.0),(f4,0.0),(f5,0.0),(f6,1.0)]:[(t1,0.7,0.15,0),(t2,0.3,0.15,0)]
```

Which would be sent to a newly constructed which would split this string into 3 parts. The first part would define the profile and the amount of agents to generate (from which 40 would be trustee agents). Part 2 would define the probability of agents having features ([1.0] defines that an agent has a high probability of having a particular feature, [0.0] would define a low probability of the agent not have that feature.) Part 3 would be define tasks with an added feature of their cost for completion. Each of these parts would be stored in separate Array Lists, These agents could be then split into ad hoc groups by allowing which would allow them to interact with each other.

A new class could be created using the weka package to integrate the M5 learning decision tree. The M5 tree could be modified to act as the stereotype where it would take in the features of successful and unsuccessful task outcomes. With creation of new methods and classes the M5 tree could become the stereotype model which would intake new agents with no prior experience and apply the stenotype prior for a particular task. The generator is quite simple and allows easy additions and implementations into various simulations.

[2]https://weka.wikispaces.com/Use+WEKA+in+your+Java+code

Appendix B

User Manual

B.1 Usage Instructions

Running the program is simple. Once one has imported the project to an IDE such as *Eclipse* or *Netbeans*, the program is ready to be launched with its current parameters.

1. **Step One** – Launching and running the program.

   As shown in figure (B.1) simply click the green button *Run Profile* or go to *Run* → *Run* [ctrl + F11 is the hotkey for *Eclipse*]

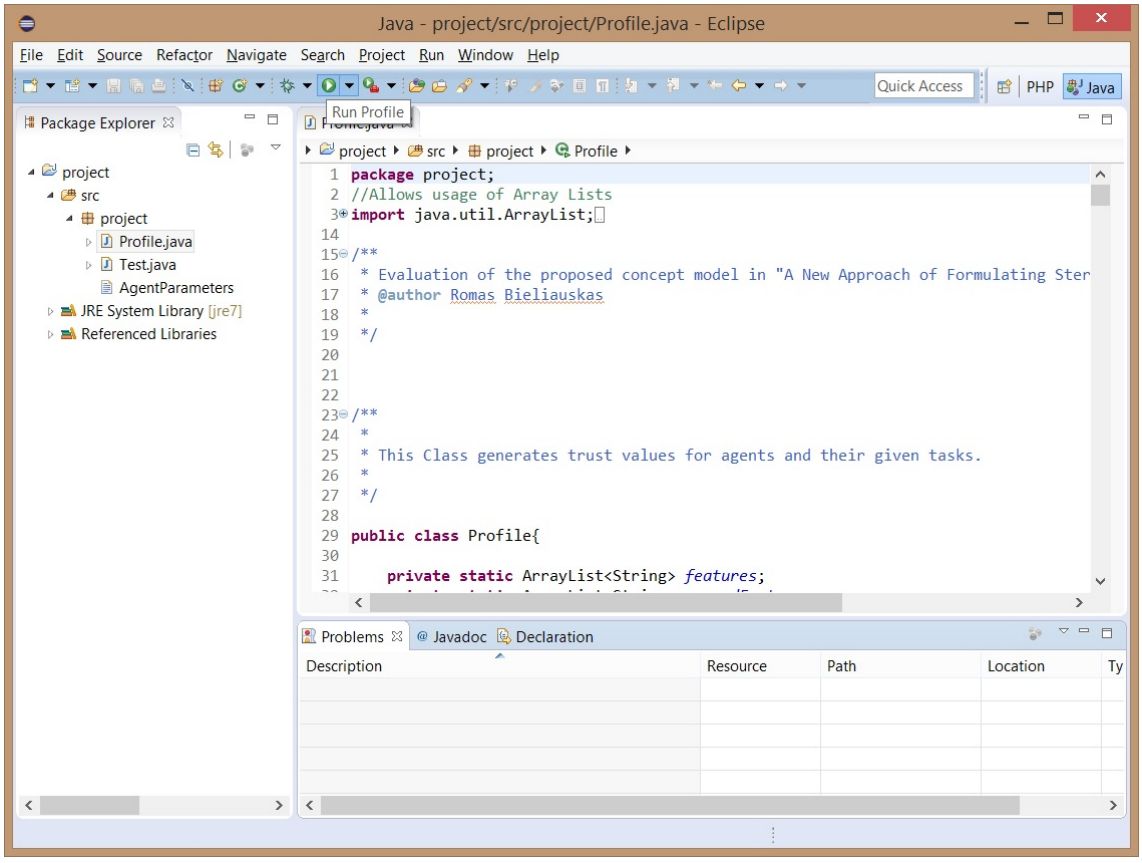


Figure B.1

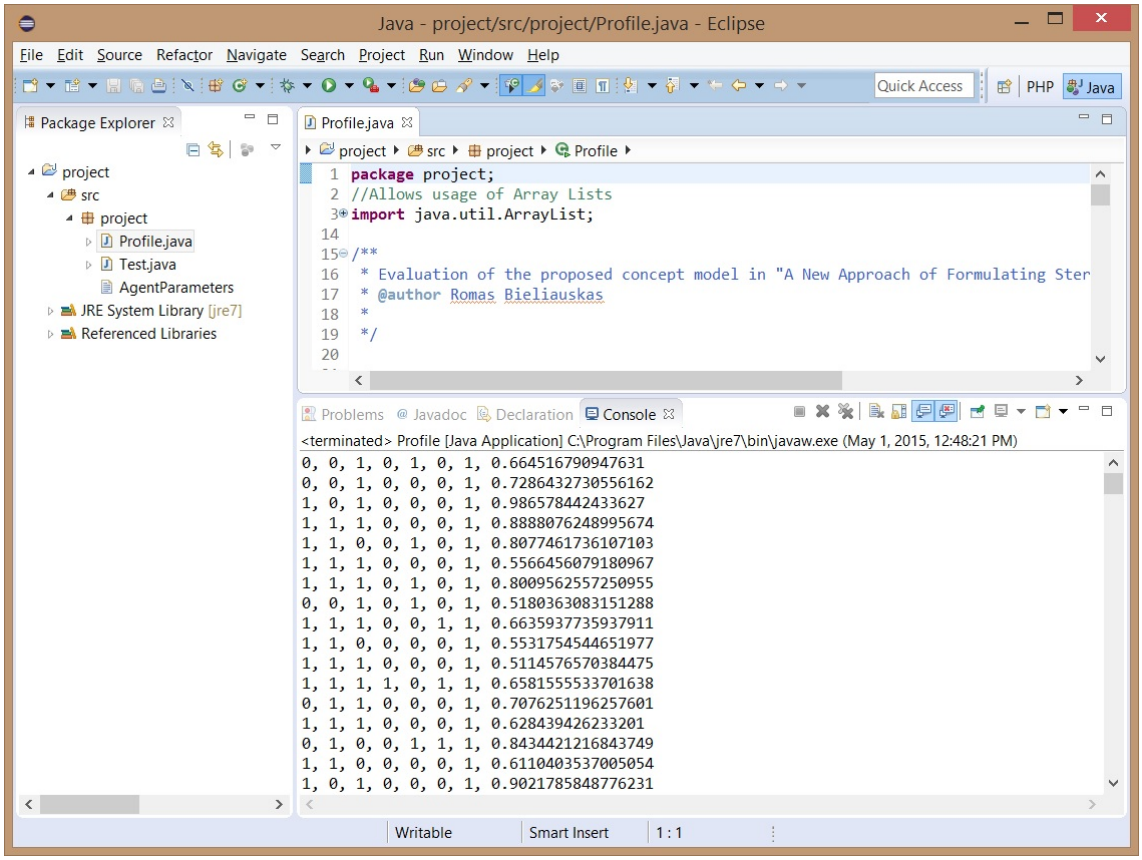


Figure B.2

The figure (B.2) above displays the user interface after the program has been run. A *console* should appear displaying features, tasks, and a normal distribution value. This means everything worked as it should and a .csv file been created

2. **Step Two** – is to find the location of the generated .csv file, which is "C:/users/*userName*/" by default.

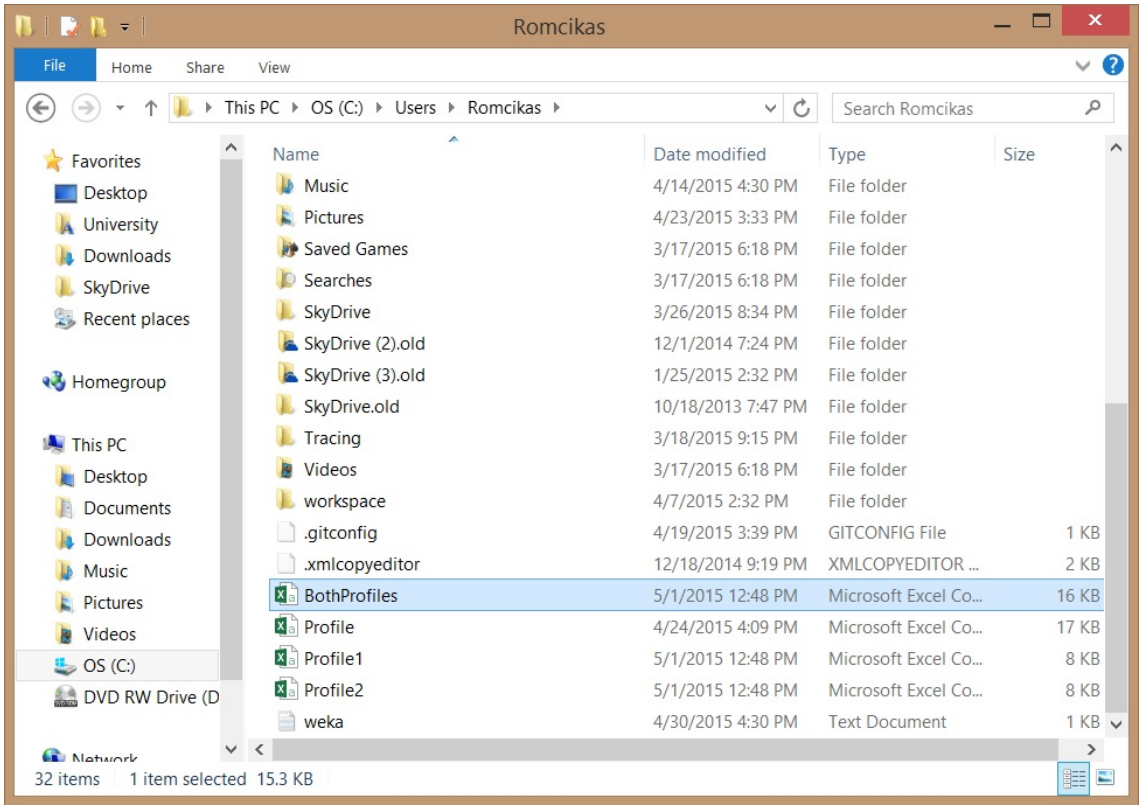


Figure B.3

For the purpose of making multiple data sets it is recommended to move the generated .csv file to a separate folder, as running the program again will overwrite the previously generated data. (Figure B.3) shows the location of the created .csv files named *"BothProfiles"*, *"Profile1"*, and *"Profile2"*. Figure (B.4), which is below, displays an example of data sets in a separate folder.

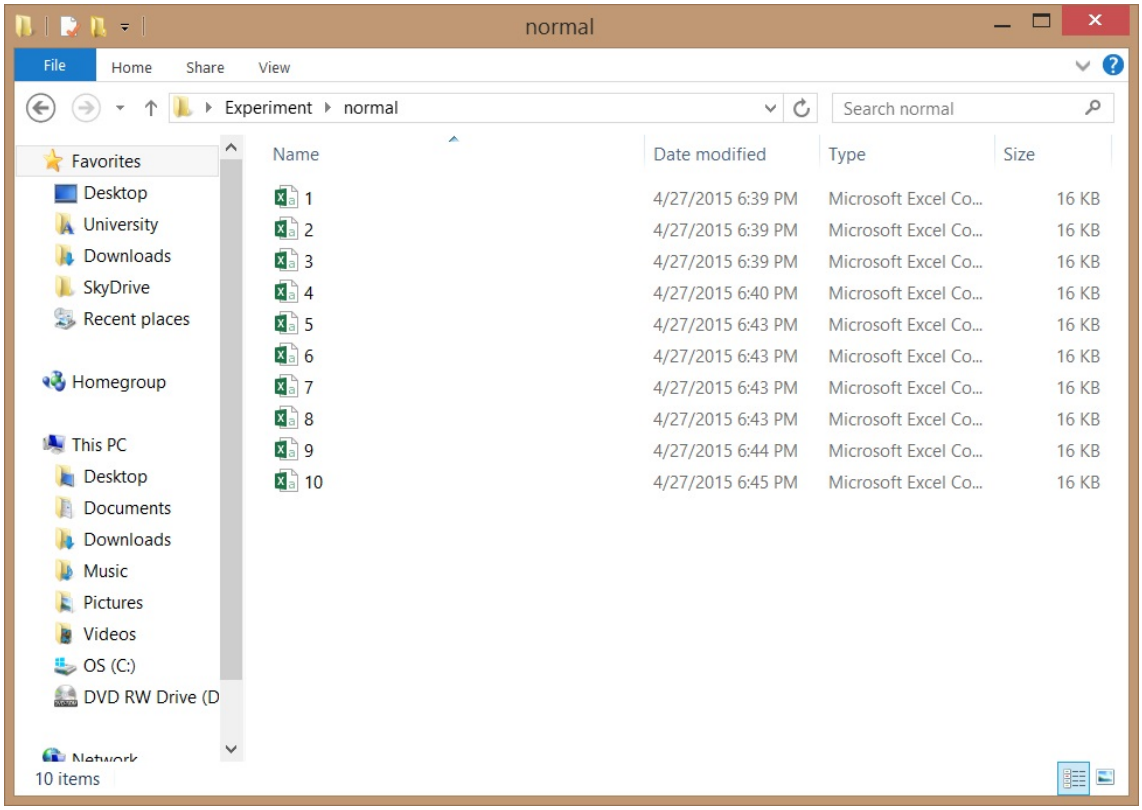


Figure B.4

3. **Step Three** – Launching Weka GUI.

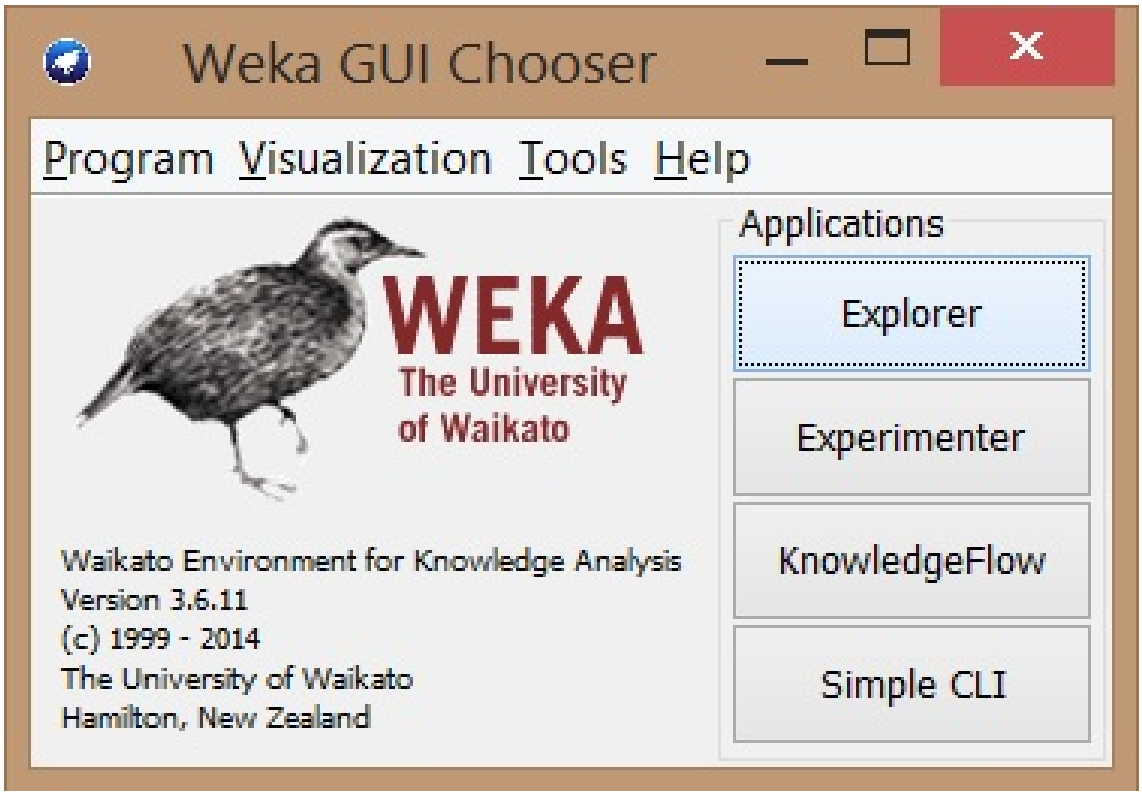


Figure B.5

Once Weka GUI is launched a GUI chooser should appear, select the *Explorer* which is highlighted in figure (B.5).

4. **Step four** – Selecting and importing data

Import the generated data set into Weka Exploerer by clicking on *Open file* which is highlighted inf figure (B.6).

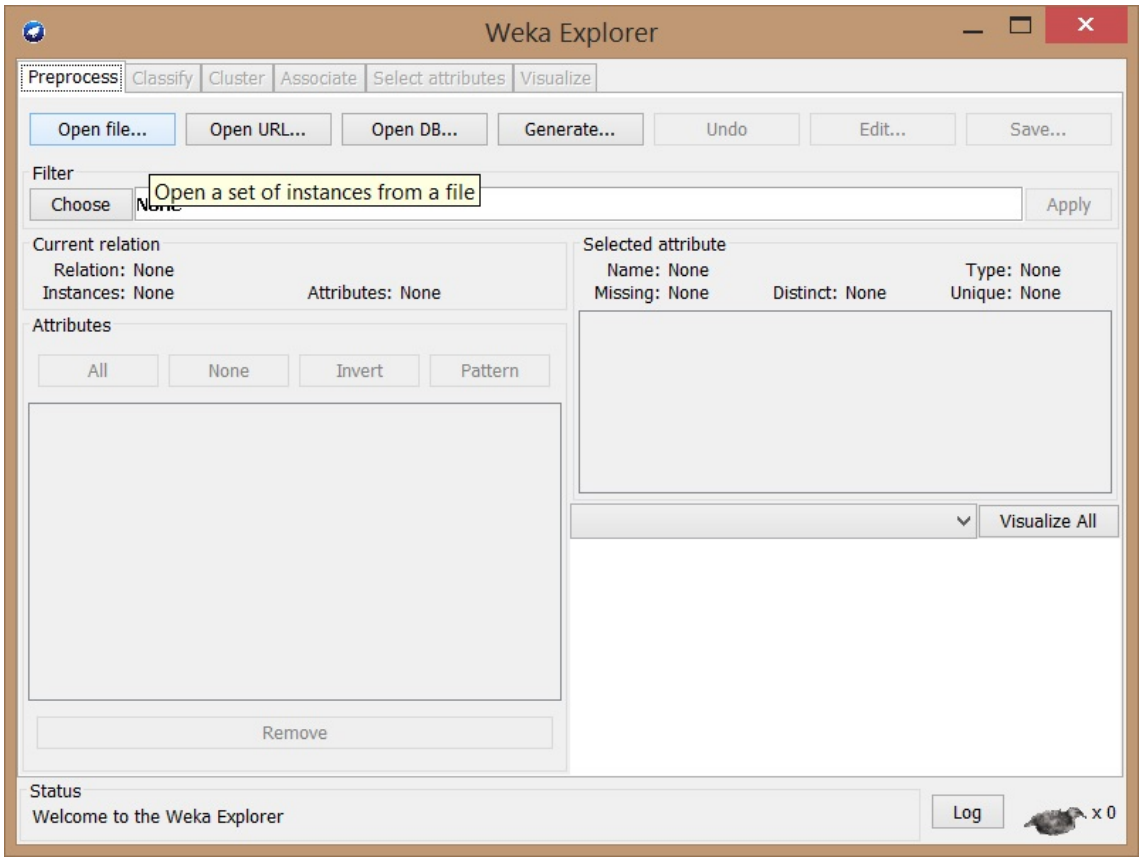


Figure B.6

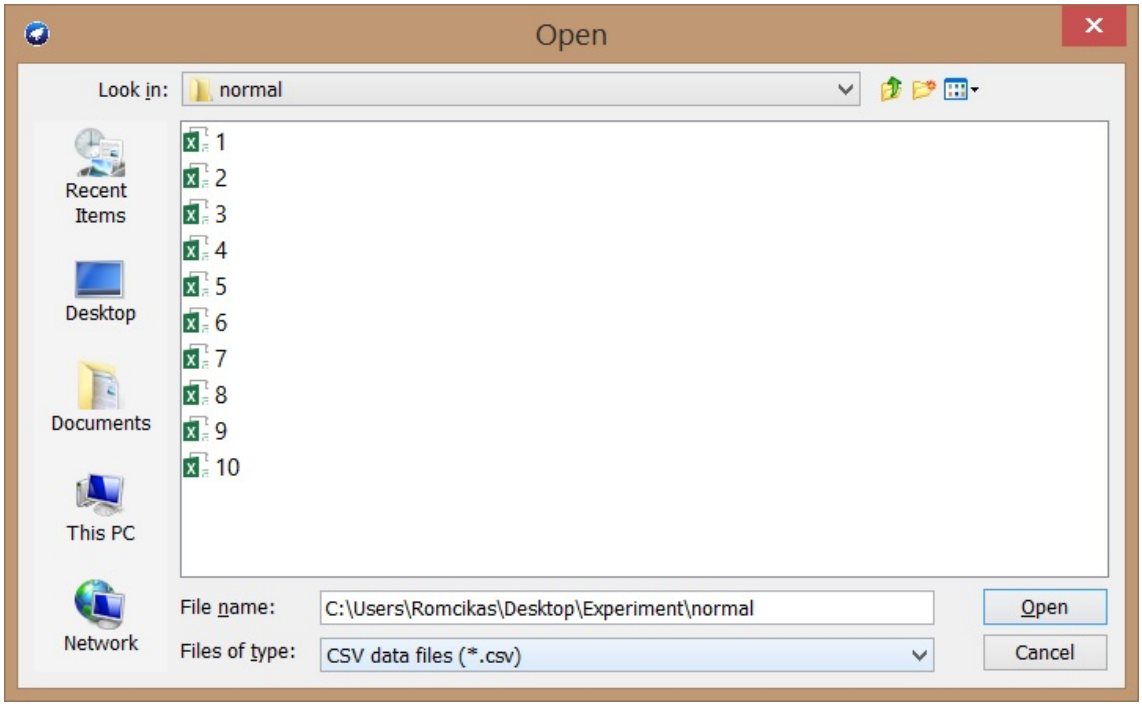


Figure B.7

It is important to note, that once you find your directory where the generated .csv file is stored it might not be seen by the GUI. Be sure to change the *Files of Type* to *CSV data files* as shown in figure (B.7) Select a file and click *Open*

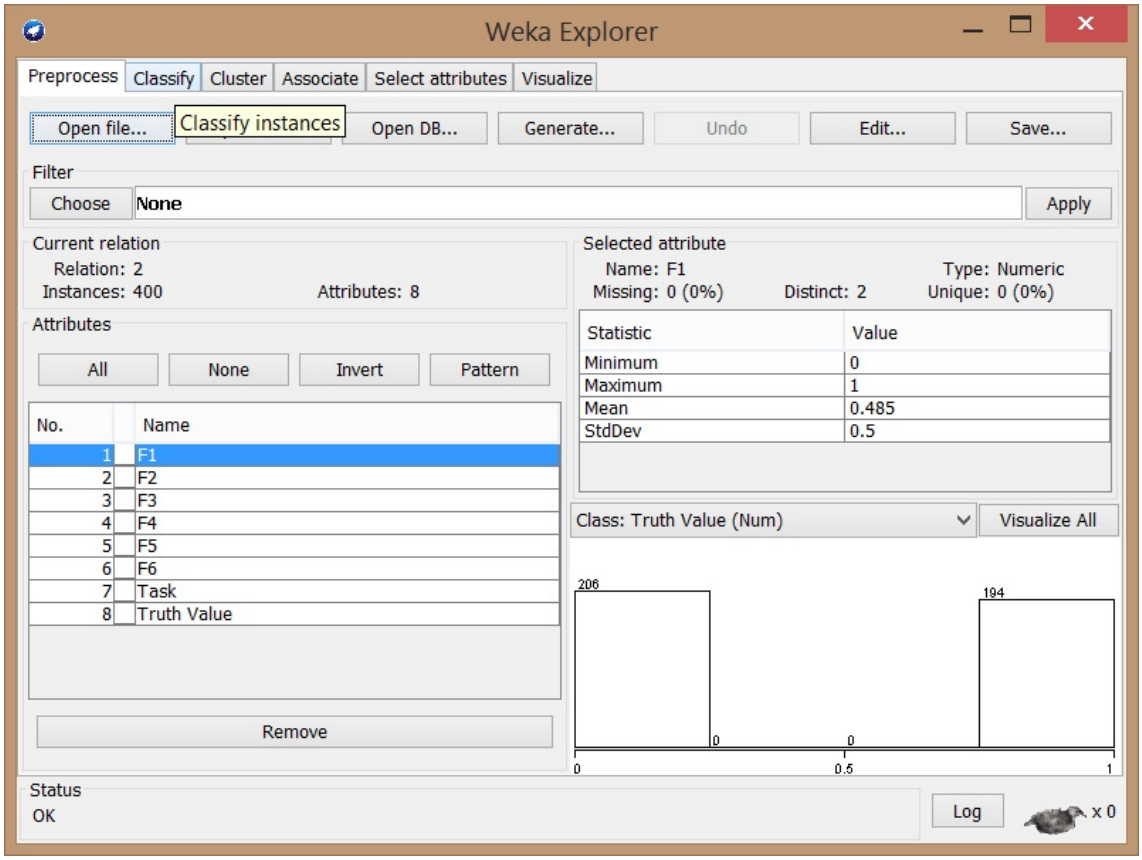


Figure B.8

Click on *Classify* as at the top left corner of your screen as highlighted in figure (B.8)

5. **Step Five** – Select a M5 Learning decision tree

Click on Choose which is highlighted in figure (B.9) to select the M5 Learning Decision tree which is located under *Trees* as demonstrated in figure (B.10)

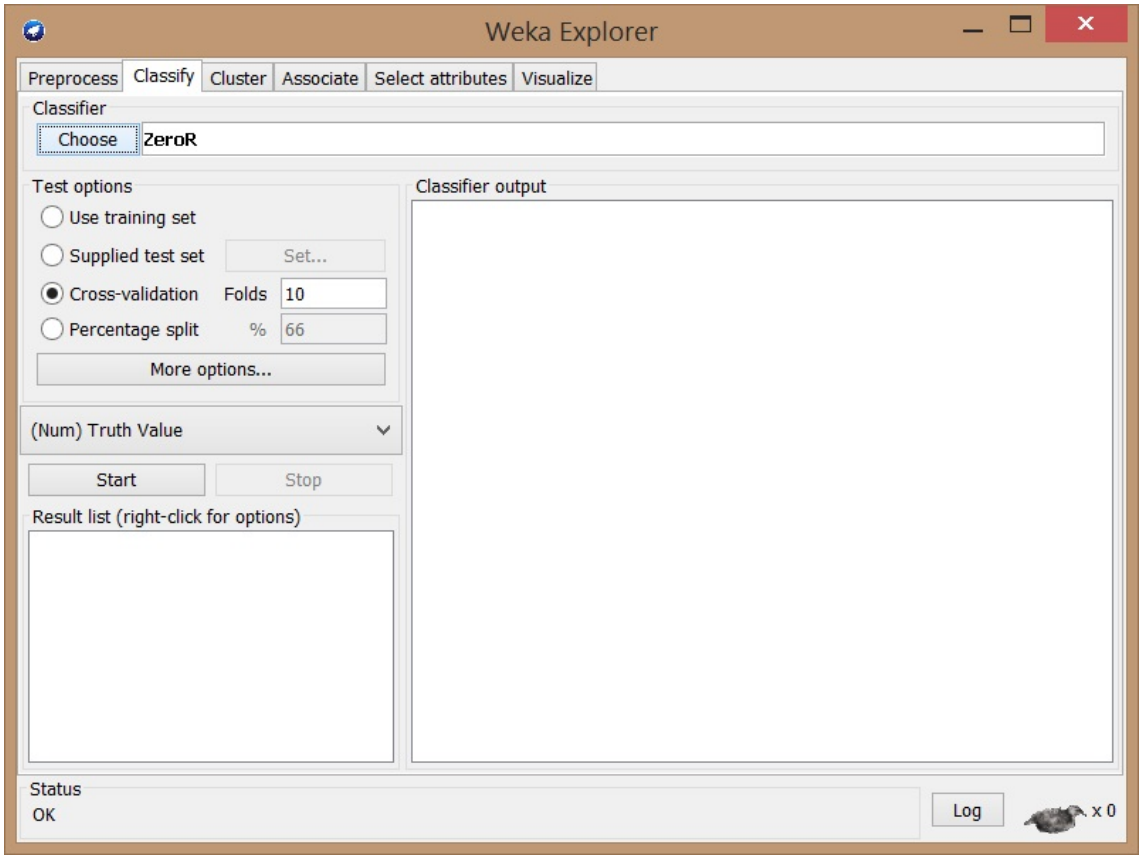

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
(Num) Truth Value
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0

Figure B.9

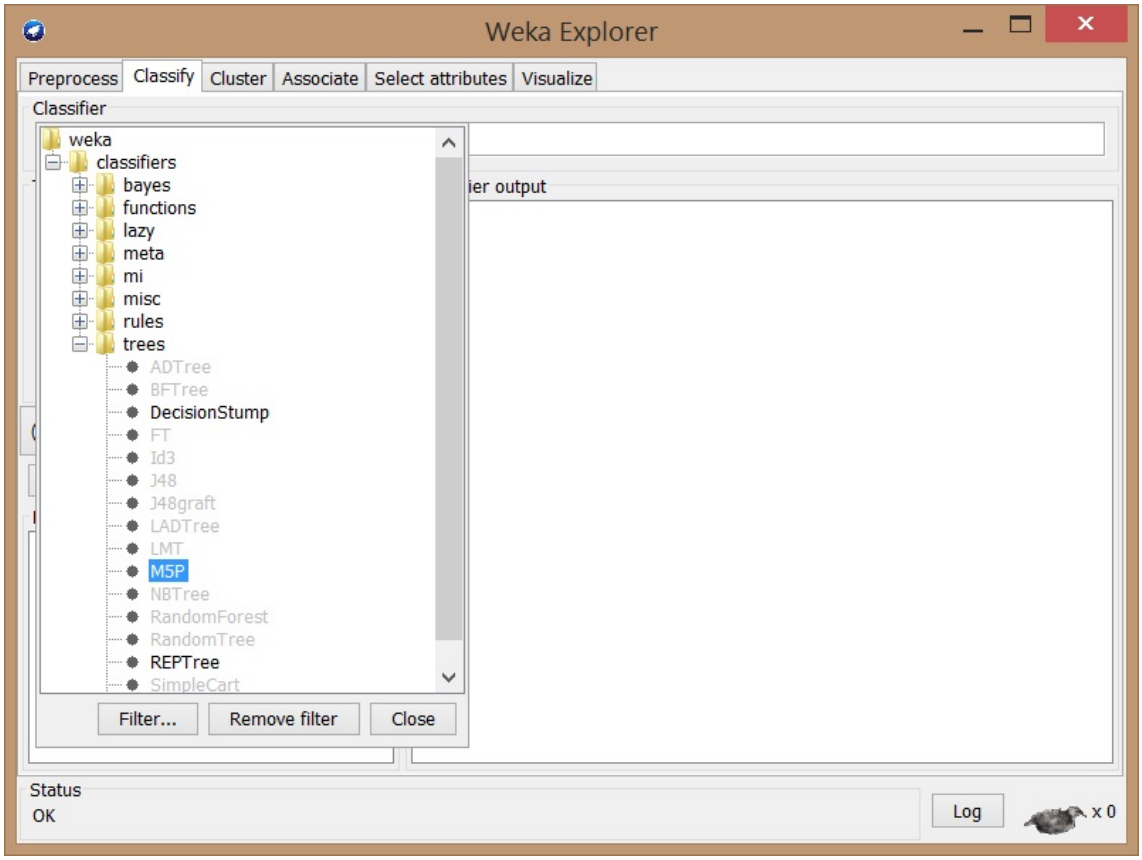


Figure B.10

6. **Step Six** – Performing the experiment

   Once you have imported the generated data and selected the M5 Decision learning tree. The default experiment is a Cross-validation with 10 folds. This is acceptable for our purpose. It will use 90% of the provided data as a training set and 10% as a test set. Simply click *Start* to run.

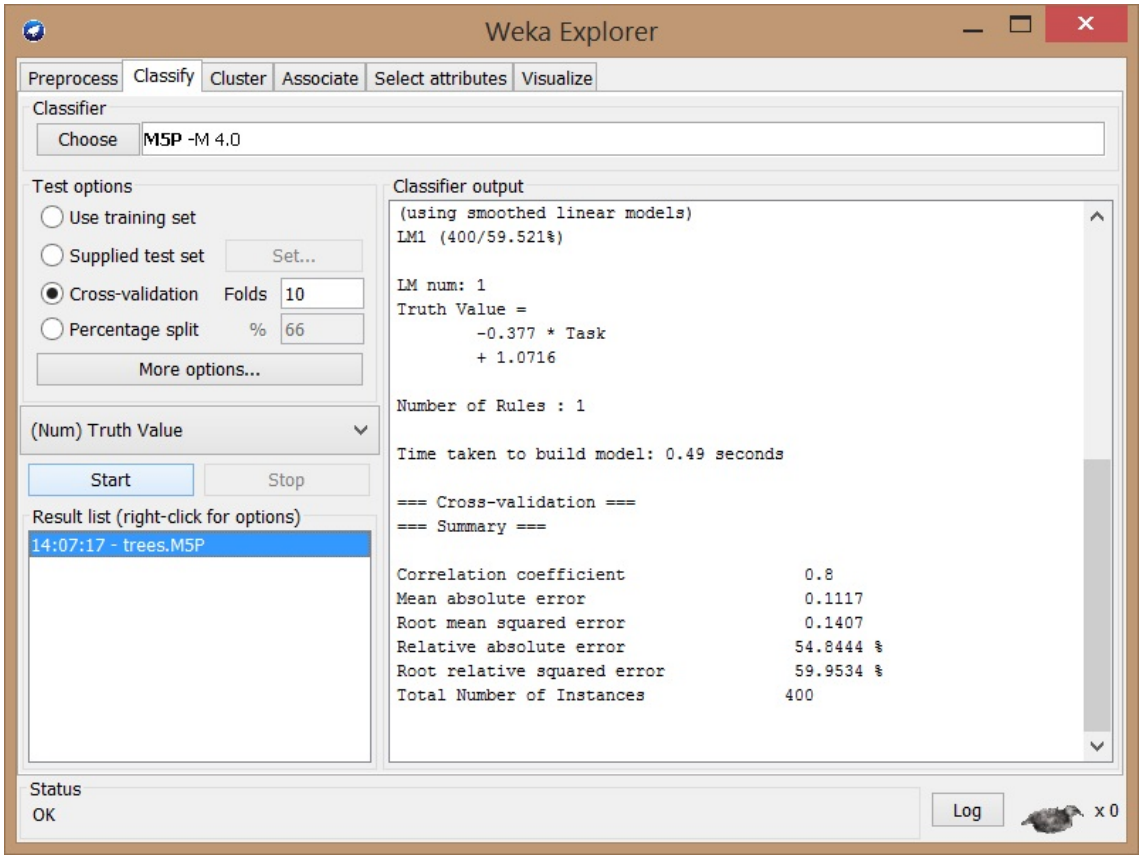


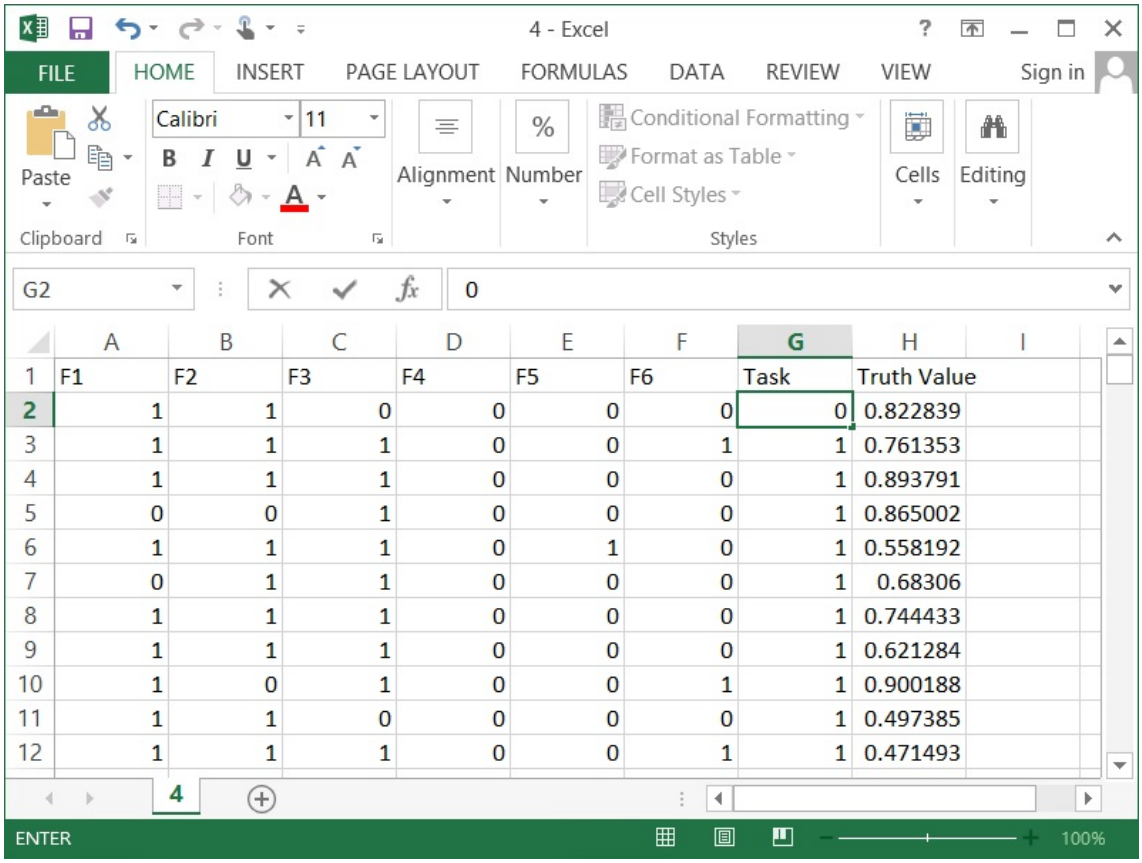
Figure B.11

Figure (B.11) highlights the *Start* button and displays an example of the presented results. However with this test we are testing agents with tasks provided.

To test the data with no presented tasks will require generating 2 .csv files of *BothProfiles*. The instructions are given at Steps 1-4. Once you have two .csv files which were generated using our provided generator follow steps below:

1. **Step 1B** – Setting tasks to zero.

   Open one of the two generated .csv files. (*it doesn't matter which one, however please be able to distinguish them*)



	A	B	C	D	E	F	G	H	I
1	F1	F2	F3	F4	F5	F6	Task	Truth Value	
2	1	1	0	0	0	0	0	0.822839	
3	1	1	1	0	0	1	1	0.761353	
4	1	1	1	0	0	0	1	0.893791	
5	0	0	1	0	0	0	1	0.865002	
6	1	1	1	0	1	0	1	0.558192	
7	0	1	1	0	0	0	1	0.68306	
8	1	1	1	0	0	0	1	0.744433	
9	1	1	1	0	0	0	1	0.621284	
10	1	0	1	0	0	1	1	0.900188	
11	1	1	0	0	0	0	1	0.497385	
12	1	1	1	0	0	1	1	0.471493	

Figure B.12

Under G select the first task which is in *row 2* change it to "0" as displayed in figure (B.12) and click and hold on the lower right square of the highlighted *G2* square and drag it down till the end of the list as displayed in figure (B.13). This will change all the values of tasks to 0. This will be our test set, where the first generated file will be our training set.

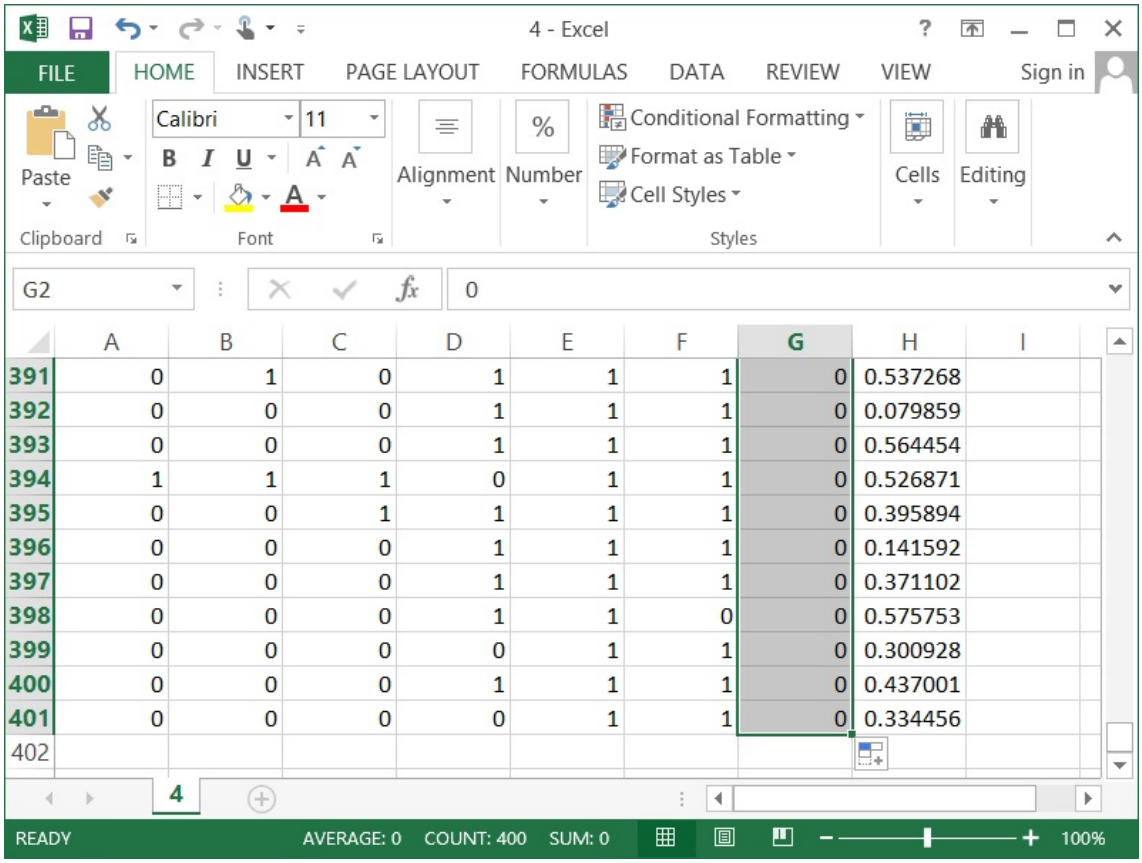


Figure B.13

Once all the tasks are set to 0 save the file while keeping the .csv format. Overwriting the file will cause no errors. Figure (B.14) displays an example of a warning that may pop up. Click *Yes* which is highlighted.

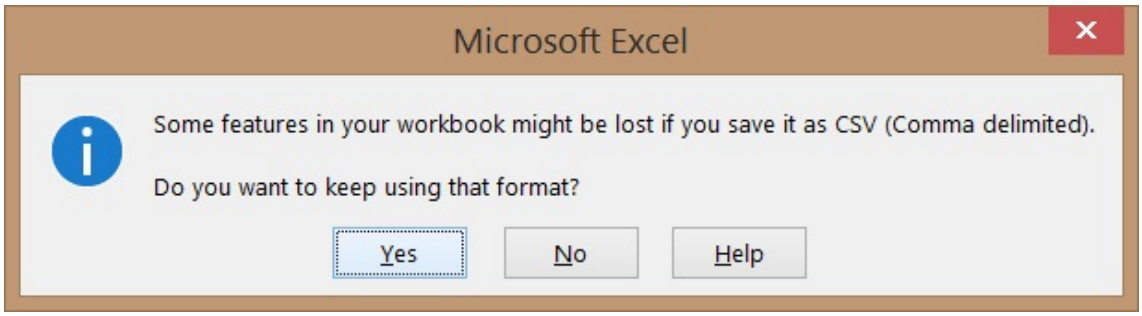


Figure B.14

2. **Step 2B** – Running the experiment with a data set for training and a test set for testing.

   Once the testing data set has been prepared open up Weka and select the M5 Learning Decision Tree. Instructions on how to open up Weka and import data sets are specified in Steps 3-5.

   For Opening a file into Weka chose the data sets **with** tasks defined. (this is the data set which you have generated without changing the task values to zero). Go into Classify and this time select *Supplied test set* as shown in figure (B.15)

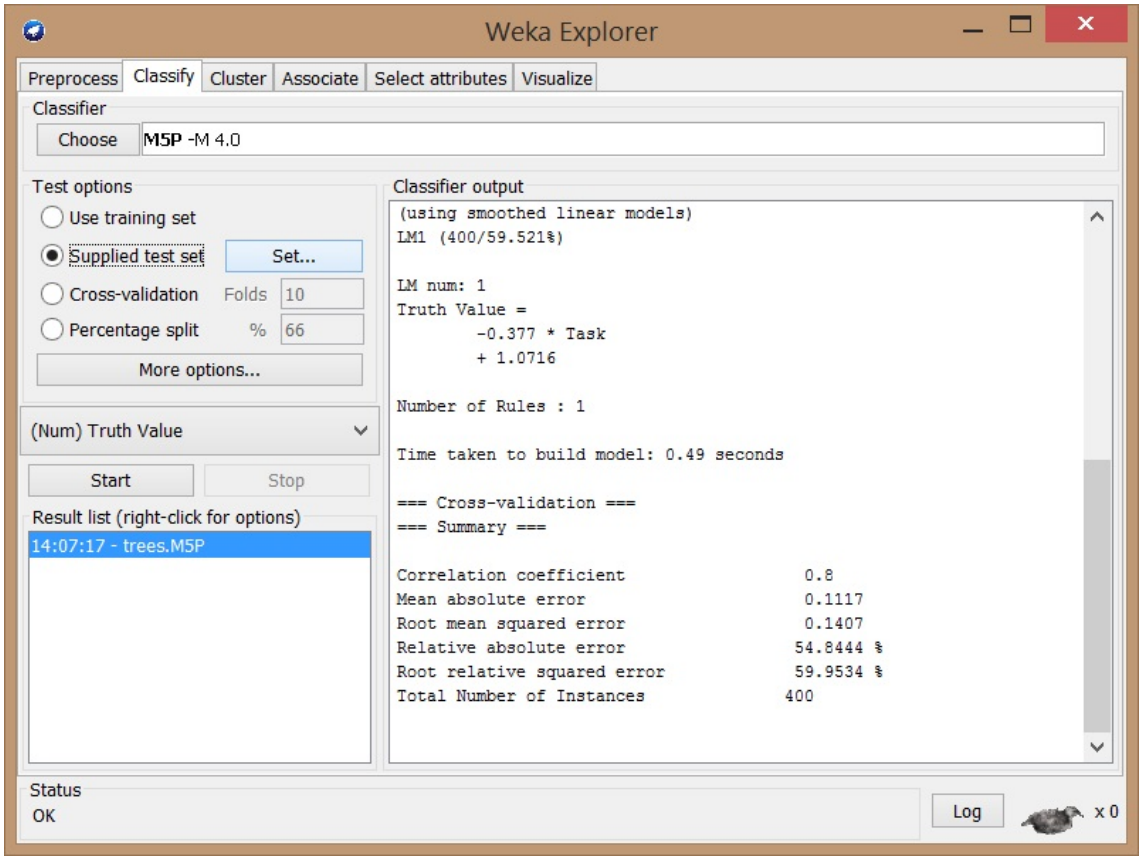


Figure B.15

Click on the highlighted *Set...* button in figure (B.15) and a small external window will pop up displayed in figure (B.16).

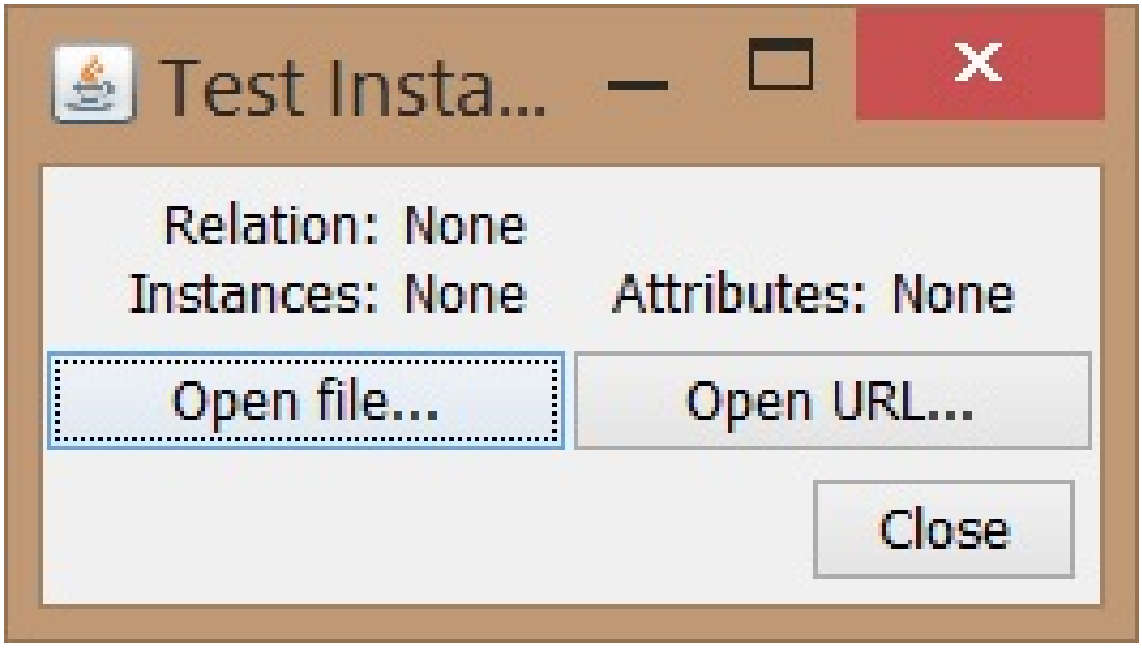


Figure B.16

Click on *Open File...* and select the data set where you have set the task values to zero. Do note that you might have to change *Files of Type* to CSV. Once you have selected the file *Relation:*, *Instances:*, and *Attributes* should change as shown in figure (B.17) **Do not** close this window, simply go back to your Weka Explorer and click *Start*.

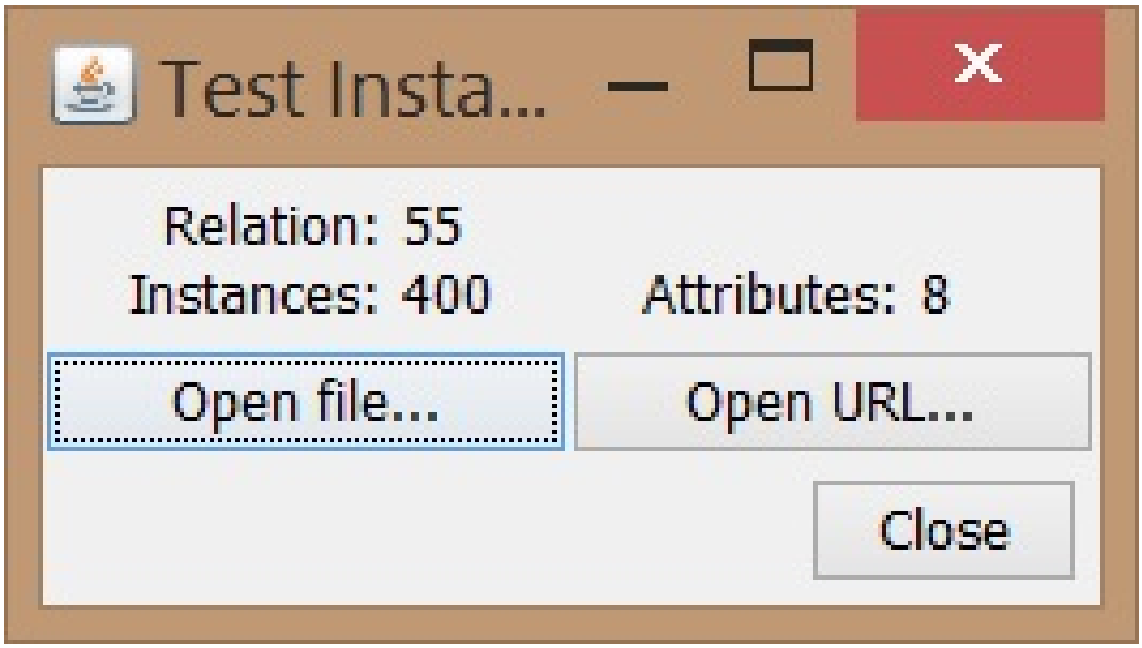


Figure B.17

This will provide you with results where the test set contains no information about the task. Figure (B.18) displays an example of the results.

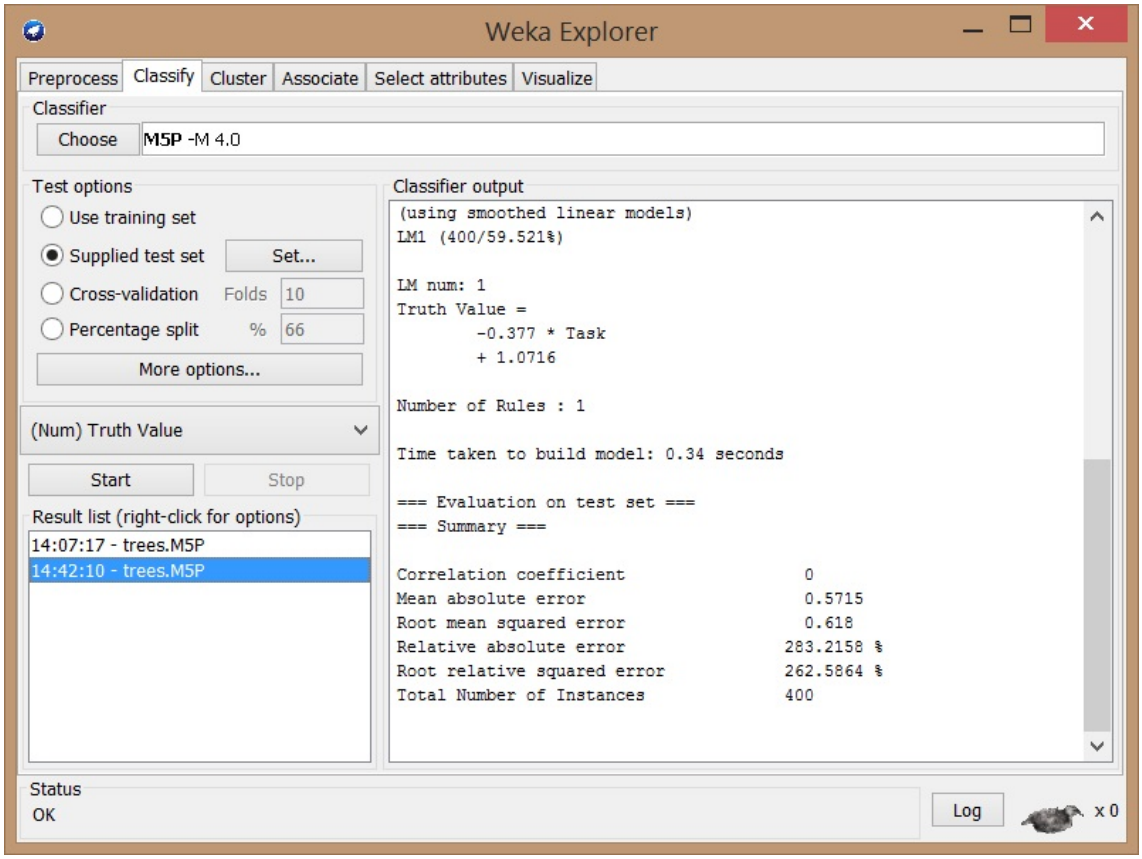


Figure B.18

You may want to click the *Visualize* tab as shown in figure (B.19), from which you can select any of the visualization squares. Do note for appropriate visualization, the x and y coordinates have to be set to *Task* and *Truth Value*.

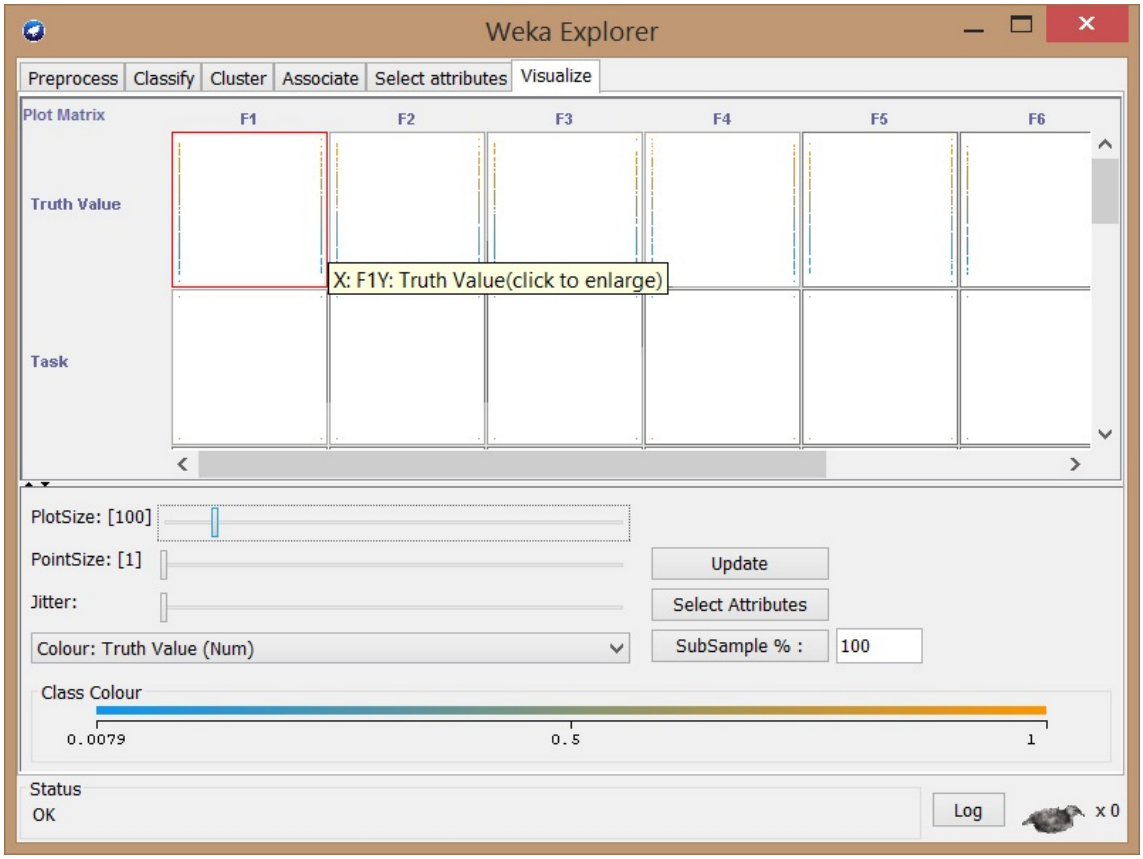


Figure B.19

This concludes the user manual. However It is possible to change variables in the provided generator to generate different results. Such as agent features and normal distribution.

B.1.1 Code Modification

The program can be modified, where variables can be changed to receive different outcomes. Below we display two methods which represent the input values for the normal distribution. They can be changed based on preferences to create a a different trustworthiness profile.

```
            // Mean and SD for First task trust values (highTrust ArrayList)
Line 104            double mean = 0.7;
Line 105            double stdev = .15;

            // Mean and SD for First task trust values (lowTrust ArrayList)
Line 108            double mean2 = 0.3;
Line 109            double stdev2 = .15;
```

As displayed above Lines 104-105 will generate high normal distribution values and lines 108-109 will generate low normal distribution values.

Below are three methods which generate which features are more likely to be selected. Modifying Lines 498, 529, and 564 will either increase or decrease the probability of a feature to be generated.

```
public int randomGenerator(){

        // Define j as an integer
        int j;

        // Define c2 as a math random method
        double c2 = Math.random();

 //Checks if the generated number is less than a provided digit
Line 498                        if (c2 < 0.5) {

                                // If true j is defined as 1
                                j =1;}

                        else {
                                // If false j is defined as 0
                                j =0;
                        }
        // j is then passed on to represent a feature in its called method
                return j;
        }
```

```
  public int randomGeneratorTwo() {

        // Define j as an integer
        int j;

        // Define c2 as a math random method
        double c2 = Math.random();

 //Checks if the generated number is less than a provided digit
Line 529                        if (c2 < 0.8){

                                // If true j is defined as 1
                                j =1;}

                        else {

                                // If false j is defined as 0
                                j =0;
                        }
```

```
    public int randomGeneratorThree() {

        // Define j as an integer
        int j;

        // Define c2 as a math random method
        double c2 = Math.random();

 //Checks if the generated number is less than a provided digit
Line 564                        if (c2 < 0.2) {

                                // If true j is defined as 1
                                j =1;}

                else {

                        // If false j is defined as 0
                        j =0;
                }
```

The methods displayed below are used in the feature generator method which is displayed below. These generators can be changed to call a different generator, depending which features you would like an agent to have. The first method generates features for Profile 1 and the second generator for Profile 2.

```
public void generateFeatures(ArrayList<String> feat){
        // Define i as 0 for loop length
        int i = 0;


        while(i < 100 ){

                        // First feature (F1)
Line 248                int first = randomGeneratorTwo();

                        // Second feature (F2)
Line 251                int second = randomGeneratorTwo();

                        // Third feature (F3)
Line 254                        int third = randomGeneratorTwo();

                        // Fourth feature (F4)
Line 257                        int fourth = randomGeneratorThree();

                        // Fifth feature (F5)
Line 260                        int fifth = randomGeneratorThree();

                        // Sixth feature (F6)
Line 263                        int sixth = randomGeneratorThree();
```

```
public void generateSecondFeatures(ArrayList<String> sfeat){

        // Define i as 0 for loop length
        int i = 0;


        while(i < 100 ){

                    // First Feature (F1)
Line 300                int first = randomGeneratorThree();

                        // Second feature (F2)
Line 303                        int second = randomGeneratorThree();

                        // Third feature (F3)
Line 306                        int third = randomGeneratorThree();

                        // Fourth feature (F4)
Line 309                        int fourth = randomGeneratorTwo();

                        // Fifth feature (F5)
Line 312                        int fifth = randomGeneratorTwo();

                        // sixth feature (F6)
Line 315                        int sixth = randomGeneratorTwo();
```