UNIVERSITY OF GRONINGEN

Grounded knowledge acquisition by argumentation

An implementation for fraud detection

by

Pieter de Rooij (*s2195569*)

A thesis submitted in partial fulfilment for the
degree of Master of Science in Artificial Intelligence

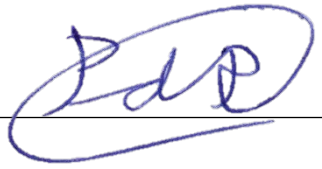
in the
Faculty of Science and Engineering
University of Groningen

August 23, 2017

Declaration of Authorship

I, Pieter de Rooij, declare that this thesis titled, 'Grounded knowledge acquisition by argumentation' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 23-08-2017

"Plans fail when there is no consultation, but there is accomplishment through many advisers."

Proverbs 15:22, *New World Translation*

UNIVERSITY OF GRONINGEN

Abstract

Faculty of Science and Engineering
University of Groningen

Master of Science in Artificial Intelligence

by Pieter de Rooij (*s2195569*)

Machine learning strives to make a system capable of autonomously achieving a level of 'understanding' of provided information. Classification is one area in which machine learning is involved. The basic problem of classification is how a novel observation ought to be labelled. Despite machine learning algorithms being capable of providing such a label, based on previous data, typical algorithms do not provide explanations for a classification. Neither does an algorithm tell how 'significant' a classification is: Should a decision maker consider this classification and act on it?

The field of argumentation can be used to yield understandable reasons for a classification. PADUA is one approach that shows how rule mining can be combined with dialogues to reason about novel observations (Wardeh et al., 2009). Bench-Capon (2003) proposed value-based argumentation frameworks that accommodate the notion that certain arguments are stronger than others.

The AGKA (Argumentative Grounded Knowledge Acquisition) architecture presented in this paper uses a decision tree, a machine learning algorithm, to learn from data. The decision tree is integrated into argumentative dialogues, similar to PADUA, to provide reasons for a classification. To rank the provided reasons by strength, expected utility is incorporated.

The architecture is evaluated in a fraud detection scenario. Results indicate that its performance is comparable to other machine learning algorithms. AGKA is also effective in finding back the rules present in the data, but only if there is a clear binary distinction between classes. This research provides insights into the connections between machine learning (finding patterns in data), argumentation (providing reasons for and against hypotheses) and decision theory (finding the best course of action in a situation).

Acknowledgements

Internal supervisor: Prof. Dr. Bart Verheij
(Institute of Artificial Intelligence and Cognitive Engineering (ALICE), University of Groningen, the Netherlands)

Second assessor: Prof. Dr. Rineke Verbrugge
(Institute of Artificial Intelligence and Cognitive Engineering (ALICE), University of Groningen, the Netherlands)

Contents

List of Figures

List of Tables

Abbreviations

AGKA	**A**rgumentative **G**rounded **K**nowledge **A**cquisition
AR	**A**ssociation **R**ule
PADUA	**P**rotocol for **A**rgumentative **D**ialogue **U**sing **A**ssociation Rules
PISA	**P**ooling **I**nformation from **S**everal **A**gents
LHS	**L**eft **H**and **S**ide
RHS	**R**ight **H**and **S**ide
Fields	
ML	**M**achine **L**earning
EU	**E**xpected **U**tility

Chapter 1

Problem description

The impact of electronic fraud is estimated to be several hundred millions for the United Kingdom and up to tens of billions worldwide (Anderson et al., 2013). The greatest contributors to these estimates are indirect costs resulting from loss of confidence. In order to diminish costs of illegitimate activity, tracking and punishing electronic criminal behaviour is required.

How can fraudulent transactions be discovered and prevented? This thesis is aimed at providing an automated system for finding typical fraudulent transactions and warn about those.

In this chapter first an illustrative scenario is sketched (Section 1.1) to familiarise the reader with detecting fraudulent electronic transactions and pinpoint the problems involved. Section 1.2 summarises how we believe the problems can be tackled, which is also the aim of this research.

1.1 Example fraud scenario

To visualise what exactly is meant by a 'fraudulent transaction', imagine the following situation. Assume Mr. Sir is called by a (credible) bank representative, who we will call Mr. Banks. Mr. Sir receives the following message:

> Mr. Sir, my name is Mr. Banks from your trusted banking company. We have detected some suspicious activity on your account. Is it true that you just transferred some money?

Before continuing the conversation, consider what procedures took place before this statement. First of all, there is a specific transaction that was flagged as suspicious. In order to flag a transaction as suspicious, there should be some ground for believing the transaction to be different from all the 'normal' ones. In order to recognise what is 'normal' and what is not, access is needed to historical data or knowledge of past transactions. Moreover, since many millions of transactions are fulfilled every day, it would be impractical to have every single one checked by a human. Hence, an automated system is required for verifying new transactions. Only after all these steps are completed a transaction may appear suspicious, prompting further investigation, possibly by a phone call as described before.

Naturally Mr. Sir is piqued by the bank representative's question, so it would be logical if he answered:

> Not sure, what are you talking about Mr. Banks?

Obviously, transferring an amount of money is a common practice for most people, so Mr. Banks' question if Mr. Sir "just transferred some money" is likely not specific enough for Mr. Sir to readily answer to. There are two ways in which the bank representative can reply:

1. Well Mr. Sir, we employ a highly sophisticated system that warned us of this specific transaction with ID # 1234567890. I cannot tell you more than that.
2. It appears you fulfilled a local transfer of €1000,- a few hours ago. Do you happen to be in Abuja, Nigeria?

Would statement 1 suffice and allow Mr. Sir to respond properly? Albeit concisely referring to a unique transaction, Mr. Sir is likely to still not have a clue as to what Mr. Banks is really referring to. Without further information, the conversation would be pointless as neither of both parties becomes any wiser. Statement 2 is a lot more appropriate, since it contains additional characteristics helping Mr. Sir to assess whether or not he authorised the transaction.

Consider what is required to go from statement 1 to statement 2. Recall an automated system is already in place for flagging suspicious transactions. Statement 1 requires the system to report a handle (here ID) to any suspicious transactions. Notice that Mr. Banks replied with 'I cannot tell you more than that'. In the worst case, the reply means only the handle is provided and Mr. Banks is incapable of accessing more

information on that specific transaction. If that is true, even a system that is correct every time would be impractical, because it is unknown why it is correct.

Suppose Mr. Banks is capable of accessing more information on the transaction based on the handle. Even then, it would be hard to pinpoint relevant specifics: With thirty or more variables for a transaction, should Mr. Banks mention ones as date and account number of the sender? Or would the transferred amount and name of the recipient be more insightful? Relevant variables depend on how much the variable contributed toward the transaction being suspicious, as well as on how much it allows Mr. Sir to relate to the transaction. Said otherwise, relevant specifics should have explanatory value. Why statement 2 can be relevant only becomes apparent with background information: Suppose Mr. Sir seldom sends amounts greater than €100,- and resides in Amsterdam, the Netherlands. Hence, it is likely he would remember a transaction of €1000,-, especially since it is transferred in a country foreign to him.

Lastly, it is preferred that Mr. Banks himself can readily access relevant information regarding a suspicious transaction. Rather than requiring a system's expert to retrieve specifics from a suspicious transaction, he ought to be able to access specifics by himself. Moreover, filtering out the relevant specifics should preferably not necessitate calling in a domain expert. Calling in experts would unnecessarily slow down the verification process, so preferably relevant information is readily available.

In summary, a system should not only provide a handle to a transaction, like in statement 1. It is preferred if it can (automatically) provide relevant information as to why a transaction is suspicious or not, which can be translated to statement 2. As a matter of fact, it is more likely Mr. Banks opens the conversation immediately with relevant information:

> Mr. Sir, my name is Mr. Banks from your trusted banking company. We have detected some suspicious activity on your account. Do you happen to have transferred €1000,- a few hours ago in Abuja, Nigeria?

Allowing Mr. Sir to immediately confirm or reject that the transaction was fulfilled with his consent. In turn Mr. Banks can immediately act upon Mr. Sir's answer. What if instead Mr. Banks would open as follows:

> Mr. Sir, my name is Mr. Banks from your trusted banking company. We have detected some suspicious activity on your account. Do you happen to have transferred €0,10 a few hours ago in Abuja, Nigeria?

Despite the description of the suspicious transaction being as specific as before, would Mr. Sir really care about that amount? It probably costs more to have Mr. Banks call Mr. Sir to verify the transaction than to consider the €0,10 as lost. The example falls flat with regard to whether action should be undertaken, as Mr. Sir probably prefers not losing any significant amount of money. Nevertheless, the point is that some cases are more important than others. It might be more worthwhile to investigate suspicious large transfers or to automatically reject small ones. Even though ordering suspicious cases by importance is not appropriate in the setting of electronic transactions, it may prove beneficial in other domains (explored further in the discussion, section 7).

Three problems should be apparent from the example:

Grounding in data Implementing an alert for potentially fraudulent electronic transactions requires an automated system. Based on historical data, the system ought to distinguish fraudulent transactions from non-fraudulent ones.

Explanation by reasons The system ought to report why a specific transaction is suspicious (or not).

Ordering by values Not every transaction is equally important. Hence, there should be a method of ordering transactions by importance.

These three themes have been studied in the research areas of machine learning, argumentation and utility theory respectively.

1.2 Research goal

The goal of this research is to solve the three problems described. A system is developed capable of solving the problems in a practical setting. We believe each problem relates to a specific research area: distinguishing fraudulent transactions can be done by machine learning, providing reasons for suspecting a transaction can be realised through argumentation, while ordering importance can be facilitated by utility theory.

In effect, a system solving the three problems is founded on an amalgamation of these three research fields. To successfully combine machine learning, argumentation and utility theory, it is imperative to establish commonalities between them, as well as to find out how one can strengthen the other.

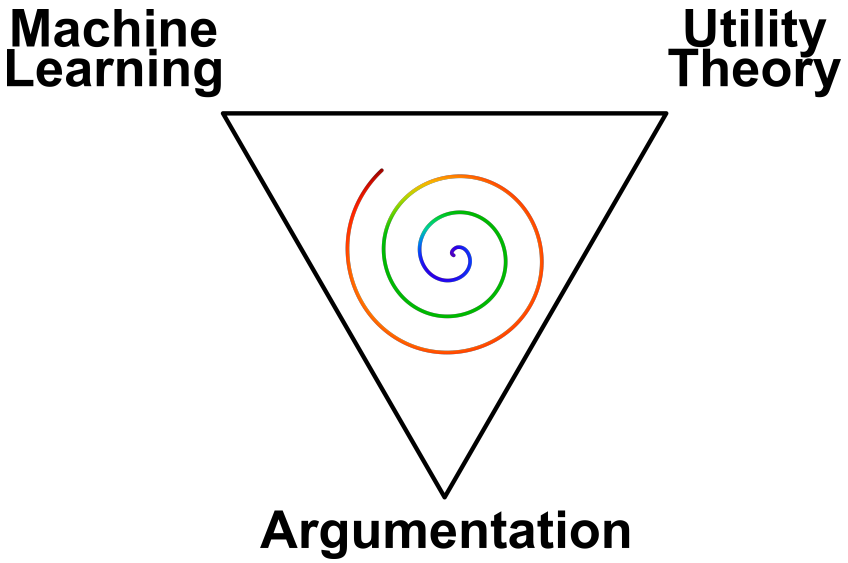


FIGURE 1.1: Combining machine learning, argumentation and utility theory.

Chapter 2

Theoretical background

The background literature comprises three main fields: machine learning (Section 2.1), argumentation (Section 2.2) and expected utility (Section 2.3). The last section mentions several approaches exhibiting a certain combination of these three main fields (Section 2.4.

2.1 Machine Learning

Extensive literature on machine learning techniques and their applications can be found in the literature, such as (Alpaydin, 2014) or (Michalski et al., 2013). This section starts with a discussion of data mining (Section 2.1.1) and continues on the subject of classification (Section 2.1.2).

2.1.1 Rule mining

Knowledge discovery in databases arose from a necessity of extracting useful information from large databases, which were becoming commonly available (Fayyad et al., 1996). The resulting techniques aimed at providing an understanding of patterns in the data, as well as scalable performance.

An example of extracting rules from a large database is (Agrawal et al., 1993). Here a large database of customer transactions is scrutinised to find rules that answer questions such as: How can the sale of Diet Coke be boosted? What impact would a discontinued sale of bagels have? Which combinations of products are likely to include some other product? To find answers to these questions, (Agrawal et al., 1993) provide a formal model for rule mining.

We follow the formal model as described in (Agrawal et al., 1993): T is a database of all transactions t, where t is a binary vector. Every $t[k]$ represents an item I_k in the complete item set I. $t[k]$ is true (or 1) if item I_k was bought, false (or 0) otherwise. It is said that t *satisfies* an item set X if for all $I_k \in X$, $t[k] = 1$.

Furthermore, an *association rule* is considered an implication of the form $X \Rightarrow I_j$, where I_j is an item which is not present in X. An association rule with confidence factor $0 \leq c \leq 1$ is satisfied in a set of transactions T if and only if at least $c\%$ of the transactions satisfying X also satisfy I_j. The notation $X \Rightarrow I_j|c$ is specified as the rule $X \Rightarrow I_j$ has a confidence factor of c.

Rules of interest inferred from a transaction base T adhere to additional constraints, which are of two different forms:

1. *Syntactic constraints.* Only a specific item (I_x) or an item set (X) is allowed to occur in the antecedent or the consequent.
2. *Support constraints.* The *support* of an association rule is defined as the fraction of transactions in T that satisfy the union of items in the antecedent and consequent of that rule. Note that this definition is different from the confidence factor of a rule.

With these definitions, rule mining can be decomposed into two sub-problems:

1. Find *large* item sets. *Large* item sets have a transaction support that is higher than a specified threshold, called *minsupport*. All other sets are called *small*.
2. Within a *large* item set, generate all rules using the items in that set.

The authors state that the solution to the second sub-problem is straightforward after having determined the *large* item sets. An algorithm is provided and elaborated on that solves the first sub-problem. It is claimed that the algorithm exhibited excellent performance on sales data obtained from a large retailing company.

2.1.2 Classification

A problem related to finding patterns in data is classification. What makes an observation a member of a certain class? Machine learning provides several techniques aimed at tackling classification problems, such as decision trees (Safavian and Landgrebe, 1991).

2.1.2.1 Decision trees

Decision trees are used in machine learning for classification. Decision trees can be attractive for several reasons (Murthy, 1998):

- Circumvents the need of acquiring knowledge from a domain expert, because knowledge can be acquired from pre-classified examples.
- Decision trees are non-parametric. This means they can model a wide range of data distributions, since only a few assumptions are made about the distribution.
- Better use of available features and more computational efficiency through the use of hierarchical decomposition.
- Tree classifiers can treat both uni- and multi-modal data the same way.
- Trees can be applied on both deterministic and incomplete problems with the same ease.
- Trees are intuitively appealing, due to the classification being performed by a sequence of simple, easy-to-understand tests.

To construct a decision tree from a given set of pre-classified training data, in general the following steps are iterated until no more splits can be made (Murthy, 1998):

1. If all training data at the current node t belongs to class C, create a leaf node with class C.
2. Otherwise, score all splits from the set of all possible splits S according to some *goodness measure*.
3. Choose the best split s^* as the test at the current node t.
4. Create a child node for every distinct outcome of s^*. The outcomes label the edges between parent and child nodes. Using s^*, partition the training data for every child node.
5. A child node is called *pure* if all training data in its partition belongs to the same class (step 1). If it is impure, steps 2 - 4 are repeated.

An example of how a decision tree is constructed can be found in (Russell and Norvig, 2010, p. 697 - 707). A popular decision tree algorithm is ID3 or its successor C4.5 (Quinlan, 1993). A different decision tree algorithm is CART (Breiman et al., 1984), which is similar to C4.5, but differs in that the output can be numerical.

2.2 Argumentation

Humans naturally engage in conversations. Depending on the purpose of a conversation, one may resort to argumentation. When trying to convince someone else or explain a matter, usually (sound) reasons are brought forward in an argument.

How arguments are structured and what types of reasoning are utilised are matters that received ample scientific attention. The focus of this section is on defeasible reasoning.

2.2.1 Defeasible reasoning

An influential paper for argumentation is the one by (Pollock, 1987). In this paper, Pollock emphasises that the philosophical notion of "defeasible reasoning" and the notion of "non-monotonic reasoning" used in AI coincide.

Pollock starts off by stating that non-deductive (defeasible) reasoning is at least as common as deductive reasoning. Standard classical logic is typically concerned with deductive reasoning. From a given set of premises, like 'Birds can fly' and 'Tweety is a bird', follow some conclusions, like 'Tweety can fly' in this example. This conclusion is valid, irrespective of additional premises. Such a logic is called *monotonic*.

Nevertheless, it is natural to deem the conclusion invalid under certain circumstances. An additional premise, like 'Tweety cannot fly', renders the conclusion invalid. This kind of reasoning is called *non-monotonic*, because an additional premise no longer warrants that the conclusion can be deduced. Non-monotonic reasoning has received interest in Artificial Intelligence for the study of reasoning and argumentation. Reiter's default logic is one system of expressing non-monotonic reasoning (Reiter, 1980). An overview of other systems is provided by (Gabbay et al., 1994).

Pollock emphasises that the concept of non-monotonic reasoning coincides with the philosophical notion of defeasible reasoning. The aim of that paper was to investigate the structure of defeasible reasoning: how a set of defeasible and non-defeasible reasons should be used in drawing conclusions. Moreover, a theory of defeasible reasoning needed be precise enough to implement in a computer program, so as to verify the theory. In subsequent publications, Pollock actually incorporated the theory into a formal system, which he named the OSCAR project (Pollock, 1995, 2008).

Pollock's theory of reasoning is based on his account of human rational architecture, which he defends in (Pollock and Cruz, 1999). According to this theory, reasoning proceeds in terms of reasons, guided by rules. Two kinds of reasons are distinguished:

Non-defeasible The conclusion (Q) is logically implied by the reason (P);

Prima facie If P is a reason to believe Q, it is called *prima facie* if there exists a condition R such that a combination of P and R is not a reason to believe Q. In this case, R is called a *defeater* of reason P for Q.

A classical example of a prima facie reason used by Pollock is "X looks red to me", as support for the conclusion "X is red". It is however conceivable that circumstances exist in which the conclusion does not hold. X might for example be illuminated by red lights, making it appear red.

Pollock distinguishes two kinds of defeaters:

Rebutting R is a rebutting defeater for P as a prima facie reason for Q if and only if R is a defeater and R is a reason for believing not Q.

Undercutting R is an undercutting defeater for P as a prima facie reason for S to believe Q if and only if R is a defeater and R is a reason for denying that P would not be true unless Q were true.

The canonical example of a rebutting defeater is about Tweety. 'Tweety is a bird and Tweety cannot fly' is a rebutting defeater to 'all birds can fly'. Not only does Tweety nullify the statement, it also leads to an opposite conclusion, namely 'not all birds can fly'. An object appearing red because it is illuminated by red lights is an example of an undercutting defeater: Even though the conclusion that the object is red is no longer warranted, it is not the case that an opposite conclusion is drawn, namely that the object is not red. The object might turn out to be actually red, even outside the presence of the red light illuminating it.

2.3 Expected utility theory

Expected utility is a way to give a value to decisions and rank those accordingly (Briggs, 2015). Expected utility theory has been used in several domains of research. For a discussion of utility theory with respect to artificial intelligence, refer to (Russell and Norvig, 2010, p. 610 - 636).

Consider a simple example: one has the option to take a pair of sunglasses along or not. Taking sunglasses along or not is an *act*. Two things can happen, either the sun is shining or it is not. These are *states* 'the world' can be in. Carrying sunglasses around

		state sun shining	no sun
act	take sunglasses	no glare, extra item	no glare, extra item
	leave sunglasses	glare, no extra item	no glare, no extra item

TABLE 2.1: Acts, states and corresponding outcomes in the sunglasses example. Since certain outcomes are preferred over others, their utility is higher.

results in added weight and can be inconvenient to wield or put away. If the sun is shining though, glare can pose a problem as it reduces visibility. Sunglasses provide protection against glare. These situations describe *outcomes*. An outcome is the result of a certain act in a certain state. The acts, states and outcomes of this example are summarised in Table 2.1.

Table 2.1 expresses the intuition that not carrying sunglasses leaves one free from wielding additional accessories, at the risk of being bothered by sun glare. Taking sunglasses along eliminates troubles from glare at the expense of being encumbered by an item. Should one take sunglasses along or leave them in this example?

The answer to this question intuitively depends on how bothered one is by glare, as well as carrying around an additional item. Such factors affect the desirability, or *utility*, of every outcome. How often the sun is shining also matters, since that increases the risk of glare. With the utility and probability of an outcome given, the expected utility EU of taking a certain decision A can now be defined as (Equation 2.1)

$$EU(A) = \sum_{o \in O} P_A(o)U(o) \tag{2.1}$$

Where $U(o)$ is the utility of an outcome o and $P_A(o)$ is the probability of that outcome given A. $P_A(o)$ can further be defined as (Equation 2.2)

$$P_A(o) = \sum_{s \in S} P(s) f_{A,s}(o) \tag{2.2}$$

Where S is the set of possible states, $P(s)$ the prior probability of a certain state s and $f_{A,s}(o)$ is a function which is 1 if outcome o results from taking action A in s or 0 otherwise. Notice that $P(s)$ is considered to be independent from the probability of taking a certain decision A. In other words, taking a certain action does not influence the likelihood of the world being in a certain state. Formally $P(s) = P(sA) = \frac{P(s \wedge A)}{P(A)}$. Hence, the expected utility of an act implements a weighing of its possible outcomes according to the likelihood of every outcome multiplied by its desirability. Table 2.2

continues the sunglasses example with the utilities of every outcome, as well as the probability of a state given.

		state	
		sun shining ($P(s) = 0.3$)	no sun ($P(s) = 0.7$)
act	take sunglasses	$U(o) = 7$	$U(o) = 7$
	leave sunglasses	$U(o) = 2$	$U(o) = 10$

TABLE 2.2: The sunglasses example continued with added probabilities and utilities.

Using Equation 2.1, the expected utility of taking sunglasses with the valuations given in Table 2.2 can be computed as

$$\begin{aligned} EU(\text{take sunglasses}) &= 0.3 \cdot 7 + 0.7 \cdot 7 \\ &= 2.1 + 4.9 \\ &= 7 \end{aligned}$$

While leaving the sunglasses has an expected utility of

$$\begin{aligned} EU(\text{leave sunglasses}) &= 0.3 \cdot 2 + 0.7 \cdot 10 \\ &= 0.6 + 7 \\ &= 7.6 \end{aligned}$$

Since $EU(\text{leave sunglasses}) > EU(\text{take sunglasses})$ the best decision to take here is to leave the sunglasses behind. Nevertheless, should the sun shine more regularly (for instance $P(sun) = 0.7$), or carrying an additional item would be no problem ($U(\text{no glare, added weight}) = 10$, then it is preferable to take sunglasses along.

2.4 Hybrid approaches

After an introduction to the three fields machine learning, argumentation and decision theory, focus is now shifted towards approaches that combine aspects of these fields.

PADUA (2.4.1) and its successor PISA (2.4.2) are introduced as methods combining machine learning with argumentation. Value-based argumentation is discussed as a combination between argumentation and decision theory.

2.4.1 PADUA

PADUA (Protocol for Argumentation Dialogue Using Association Rules) is a protocol designed to support two agents debating a classification by offering arguments based on association rules mined from individual data sets (Wardeh et al., 2009). The data sets consist of claims for a hypothetical welfare benefit. Specifically a scenario is devised reflecting a fictional benefit Retired Persons Housing Allowance (RPHA). Several conditions have to be met before one is entitled to this benefit. Among other conditions or requirements, for instance the following two are requirements according to the putative legislation: the benefit is payable to a person who is of an age appropriate to retirement and should have an established connection with the UK labour force. In this format, it is impossible to assess whether an applicant satisfies conditions, because it raises questions as: Which age is 'appropriate to retirement'? How does one measure 'an established connection with the UK labour force'? A further interpretation or specification is required in order to answer such questions and hence be able to assess whether an applicant satisfies the conditions. The authors supposed the following interpretations to be in accordance with the desires of policy makers:

1. *Age condition:* "An age appropriate to retirement" is interpreted as pensionable age: 60+ for women and 65+ for men;
2. *Contribution condition:* "Established connection with the UK labour force" is interpreted as having paid National Insurance contributions in 3 of the last 5 years.

Every instance or record in this data set represents an applicant who was either granted the benefit or not. Information about every application consists of, but is not limited to:

- The age of the applicant;
- The country of residence;
- Whether or not National Insurance contributions were paid for each year in the past years;
- Whether a benefit has been granted.

Benefits are typically decided by a range of adjudicators working in several different offices. Across offices, different types of cases are encountered. For example, the occupation of fishermen is more common at coastal regions, but is less frequently encountered

in inland areas. Nevertheless, having that occupation can affect to which benefits one is entitled. Suppose a fisherman applies for a benefit that he is otherwise not entitled to, but his occupation is an exception to that. An adjudicator from an inland office might then decline his application, because the occupation is overlooked due to it rarely being encountered. Consequently, adjudicators become experienced on often encountered cases, but develop blind spots for others, resulting in high error rates. The PADUA protocol is designed to ameliorate errors resulting from inexperience with rare cases by integrating knowledge from several sources or data sets (here offices) through means of dialogue.

At the basis of these dialogues are association rules. With 'association rule' the authors mean "that the antecedent is a set of reasons for believing the consequent". A concrete example is the rule `contr y5 = not paid -> entitles = no`, which would read as "if in the fifth year no contribution was paid, then one is not entitled to this benefit". From the meaning of an association rule follows that an association rule consists of a premise or antecedent (`contr y5 = not paid`), a conclusion or consequent (`entitles = no`) and a confidence. Confidence is derived from a player's data set. It is defined as a percentage of cases for which the consequence holds if the condition holds as well. Suppose the example rule has a confidence of 73.14%. That would then mean that of all cases in which the contribution for the fifth year was not paid, 73.14% were not granted the benefit. These association rules are mined from their data sets using standard data mining techniques.

In a dialogue, proponent and opponent take turns, defending their proposed classification or attacking the other's proposition. To do so, during every turn a player can choose a certain move. A move consists of a speech act, or the type of that move, as well as some content. Six different speech acts are included, where *Conf* is a pre-defined confidence threshold representing the lowest acceptable confidence:

Propose rule: This speech act proposes a new rule with a confidence higher than the threshold (*Conf*), (in the case of two player games the confidence of this rule should also be higher than any other move played by the other side).

Distinguish: This act adds some new premise(s) to a previously proposed rule, such that the confidence of the new rule is lower than the confidence threshold (*Conf*).

Unwanted consequences: This speech act suggests that certain consequences (conclusions) of some rule previously played in the dialogue do not match the studied case.

Counter rule: This speech act places a new rule that contradicts the previous rule. The confidence of the proposed counter rule should be higher than the confidence of the previous rule (and higher than the threshold *Conf*).

Increase confidence: This speech act adds some new premises to a previous rule so that the overall confidence rises to some acceptable level.

Withdraw unwanted consequences: This act excludes the unwanted consequences of the rule it previously proposed, while maintaining a certain level of confidence (at least higher than the confidence threshold *Conf*).

A dialogue ends when a player fails to play a legal move in its turn, meaning this particular player loses the game while the other wins. In effect, the class proposed by the winner is the most convincing one, since the loser is unable to counter it.

2.4.2 PISA

The PISA (Wardeh et al., 2012) (Pooling Information from Several Agents) multi-agent framework is an extension to PADUA. Just like PADUA, PISA models *argument from experience*. Every agent has a background data set of past examples. This database is considered as encapsulating an agent's *experience*. Arguments are mined from this database using the same data mining techniques as used in PADUA.

A major difference between PADUA and PISA is that PISA is capable of incorporating multiple agents, while PADUA only allowed two agents to argue about the classification of novel instances. Having more than two agents presents several challenges for dialogues. For example the multi-agent argument has to be coordinated and groups may be formed between several agents promoting the same classification.

A neutral Chair Person Agent (CPA) is elected for coordinating the dialogue. Its responsibilities are:

- Starting a dialogue;
- Terminating a dialogue when a termination condition is satisfied;
- Announcing the resulting classification for the given case (once the dialogue has terminated).

If several agents advocate the same class, they are required to join forces and act as a single Group of Participants. Within this group, a leader is elected which is the agent with the greatest experience (expressed in number of records in its database). At every round, the group leader decides what move to play (if any). Group members are allowed to suggest moves if they are able to, while the leader compares all the moves and selects the best one based on confidence, if any are proposed.

The performance of PISA is evaluated against other classification approaches, including decision trees and ensemble methods. The authors conclude that performance is comparable with these other methods, but when operating groups or in noisy data, PISA outperforms these approaches.

2.4.3 Value-based argumentation

In the argumentation methods discussed before, individual reasons or arguments are considered of equal value or strength. Despite differences between the kind of arguments, like undercutting or rebutting defeaters, any individual reason is not credited differently from any other.

In real-life conditions however certain arguments may be stronger to some people than others. For instance in a debate about whether taxes should be raised or lowered. Some parties, or in general the more formal term *audiences*, will argue for taxes to be raised to promote social equality, while other parties will argue for taxes to be lowered to promote enterprise. Which side a party supports depends mainly on which norm they value more, social equality or enterprise.

(Bench-Capon, 2003) incorporates different values to different audiences by extending argumentation frameworks to value-based argumentation frameworks (VAF). In this extension, every argument is associated with an (abstract) value. Whether an argument defeats another one, depends on the audience: if argument A attacks argument B, then A defeats B for audience a if the associated value of B is not higher than the associated value of A for audience a.

2.5 Goals revisited

Recall the three problems identified in the previous chapter:

Grounding in data Implementing an alert for potentially fraudulent electronic transactions requires an automated system. Based on historical data, the system ought to distinguish fraudulent transactions from non-fraudulent ones.

Explanation by reasons The system ought to report why a specific transaction is suspicious (or not).

Ordering by values Not every transaction is equally important. Hence, there should be a method of ordering transactions by importance.

We believe the three research fields discussed in this chapter are individually capable of solving one of the aforementioned problems. With the literature in mind, it is possible to provide a more specific description of how each field provides a solution to one of the problems:

Machine learning Classify transactions as legitimate or illegitimate based on past transactions.

Argumentation Provide an understandable support by means of dialogue.

Utility theory Decide whether or not to investigate a transaction based on the expected utility of an investigation (action).

On top of that, two approaches are discussed that combine aspects from the three research fields.

(Wardeh et al., 2009) PADUA uses association rules (rule mining) inside dialogues to decide whether or not someone is entitled to a benefit. Hence, PADUA combines the fields of machine learning and argumentation.

(Bench-Capon, 2003) Value based argumentation frameworks (VAF) incorporate value, for certain audiences, into arguments. While not explicitly using expected utility, it can be said that this approach combines argumentation and utility theory. Since expected utility also offers a way of valuating, the value in a VAF might be expressed by expected utility.

Ultimately, the goal is to combine the three fields machine learning, argumentation and utility theory to solve all three problems at once. How can these three research fields be combined into one approach? This question is at the core of the next chapter.

Chapter 3

The AGKA architecture

Recall the narrative from Section 1.1, where an illegitimate transaction just came in for the system to verify. Recall that the system need not only recognise a transaction to be illegitimate, it is also desirable that it gives an understandable reason why. Suppose the incoming transaction is illegitimate because the transfer is made to a foreign account. We argue that AGKA is capable of discerning this transaction as illegitimate, including the association rule `Foreign = true` $\Rightarrow$ `illegitimate` as support. The labelling together with the support provided allows a bank representative to clearly inform a potential victim of the situation. Before showing how the label and its support are generated, first the AGKA architecture needs to be explained.

This chapter provides a description of the AGKA architecture. The aim is not to describe the details of the implementation used here. Emphasis is on the conceptual aspect of the components: what the use of a component is and, if applicable, what underlying processes support a component's result.

In this chapter we first turn toward some (abstract) data structures (Section 3.1). First the basic concept of a *construct* is introduced in Section 3.1.1. Next, a general *association rule* (AR) is formalised in Section 3.1.2. Section 3.1 concludes with the structure of an *instance* (Section 3.1.3).

After clarifying these general structures, focus is shifted to their specific implementations with regard to data generation (Section 3.2). This section is divided into two parts: *transactions* (Section 3.2.1) are a specific implementation of *instances*, while *data generation rules* (Section 3.2.2) are a specific implementation of *association rules*. Be informed that the details of generating the utilised test sets are reserved until chapter 5.3.

Section 3.3 is devoted to all the components of the AGKA architecture. Continuing on the subject of data, the database component is discussed first in Section 3.3.1. Another specific implementation of AR, called *knowledge rules* here, returns in Section 3.3.5. This section also formalises how the utilities of knowledge rules are determined. Finding knowledge rules is covered by the machine learning component, which is described in Section 3.3.2. The dialogue component, using knowledge rules, is explained in Section 3.3.3. The error component, described in Section 3.3.4, allows the system to learn from its mistakes. How the individual components of AGKA are put together to provide classifications is described in Section 3.4.

3.1 Data structures in AGKA

This section describes the general format of several structures used in the architecture. Their specific implementations are reserved for future sections. First a *construct* is defined, which specifies some relationship between two items (Section 3.1.1). Next, the format of *association rules* is formalised (Section 3.1.2), which resembles the definition of association rules as found in the background literature (e.g. (Agrawal et al., 1993) or (Wardeh et al., 2009)). Lastly, the build-up of a data point or *instance* is explicated in Section 3.1.3.

3.1.1 Construct

A *construct* is a container of any two items, with a defined *relation* between them. The first item is said to be on the *left hand side* (LHS), while the second item is said to be on the *right hand side* (RHS). The format of a construct is formulated as:

$$\begin{gathered} < \text{item 1}, \quad \text{relation}, \quad \text{item 2} > \\ \text{or} \\ < \text{LHS}, \quad \text{relation}, \quad \text{RHS} > \end{gathered} \tag{3.1}$$

Some examples of constructs are $B > A$, $A \leq 2$ or $C < D \ and \ D > B$. The latter shows how constructs may be embedded, since it is a construct consisting of two constructs, combined by the relation 'and'.

3.1.2 Association rules

An association rule (AR) is here defined as:

{
First 1
Second 2
A value
B value
49 valid
}

TABLE 3.1: An example of an instance with five variables.

$$< Cd, Cs, P, T > \tag{3.2}$$

Where:

Cd A rule's *condition*, containing a construct.

Cs The *consequence* of a rule, also containing a construct.

P Defines the conditional $(Cs|Cd)$.

T Denotes the rule type, which can either be *data generation* (Section 3.2.2) or *knowledge* (Section 3.3.5).

With this definition, an AR can be envisioned as $Cd \Rightarrow Cs$, with probability P. An example AR could be `A > 0 and B < 2` $\Rightarrow$ `X and Y`, $P = 0.7$.

The definition of an AR used here is close to that of the background literature (e.g. (Agrawal et al., 1993) or (Wardeh et al., 2009)). The condition and consequence are common. Probability is not always included, but is utilised in the literature described before. The rule type T is added to accommodate for different use cases within the AGKA architecture (such implementations are described in later sections).

3.1.3 Instances

Any data point is regarded as a collection of *variable - value pairs*, for example A (variable) - 100 (value) or Name - Bob. An instance may contain any number of such pairs, under the condition that every *variable* is unique. Due to the uniqueness condition, it is also said that an instance consists of n variables, rather than n variable - value pairs. An example of an instance with five variables is given as Table 3.1.

{

Sender	Ms. S. Stam
Send. account	NL99BANK0123456789
Recipient	Sir I. Cashalot
Rec. account	GB00AAAA9876543210
Date	12-03-2004
Time	15:50:22
Amount	12345
Pre-balance	123456
Post-balance	111111
Foreign	true
...	...

}

TABLE 3.2: An example of a transaction with some fields specified.

3.2 Data generation in AGKA

This section describes how data is generated. Data points here represent transactions, which may be considered a special case of instances in general (Section 3.1.3). The build-up of transactions is explicated in Section 3.2.1. Transactions are generated through the use of data generation rules (Section 3.2.2), which are one specific type of association rules (Section 3.1.2).

3.2.1 Transactions

In our system, a data stream of (electronic) transactions between two parties is used. The transactions are modelled as instances (Section 3.1.3). Just like an instance in general, a transaction contains a number of variable - value pairs (also called field - value pairs here). Unlike general instances though, the range a value can take depends on the variable. This range is defined by a data generation rule (Section 3.2.2).

Since the variable - value pairs depend on the defined data generation rules and these rules differ per stream, an a priori definition of the (number of) fields contained in a transaction cannot be given. Nevertheless, several fields are always present, because of what a transaction represents. These fields are *Sender name, Recipient name, Send. account, Rec. account, Amount, Date* and *Time*. *Date* and *Time* are generated independent from data generation rules (Section 3.2.3), while the other fields are required to be defined by a rule.

An example of a transaction is displayed as Table 3.2.

Variable type	**Range parameters**
Boolean	true or false
Uniform distribution	a (lower bound) and b (upper bound)
Normal distribution	μ (mean) and σ (standard deviation)
Profile	A set of profiles (names and bank account numbers)
Categorical	A set of categorical values

TABLE 3.3: Distributions and how their respective ranges may be defined in the consequence of a data generation rule.

3.2.2 Data generation rules

The data stream presented to the system is generated with the help of *data generation rules*. Using these rules ensure a desired pattern or rule is present in the stream. Data generation rules are a specific type of association rules. Recall the definition of an AR:

$$< Cd, Cs, P, T > \tag{3.3}$$

Since data generation rules are an implementation of the general AR, their definition is constrained in the following ways:

Cd The condition contains one variable of a transaction. In effect, a data generation rule operates on one specific variable - value pair of a transaction.

Cs The consequence specifies the range of a value in a variable - value pair. The specification requires both a *distribution* and *range parameters*, which can be one of the following given in Table 3.3.

P The (pre-defined) probability. Samples within the specified range with probability P and an 'alternative value' with probability $1 - P$. How values are sampled is described in Section 3.2.3.

T The rule type is in this case a data generation rule.

3.2.3 Generating a stream

A stream is generated to simulate a real-time inflow of transactions. Several parameters can be specified that affect the stream generated:

i_max The total number of iterations.

p_{il} Probability of an illegitimate transaction occurring.

Variable type	**Sample value**
Boolean	The Boolean value specified
Uniform distribution	A uniformly drawn sample within the range $[a, b]$
Normal distribution	A sample from the normal distribution (μ, σ)
Profile	One of the name and account combinations (profiles) in the specified set
Categorical	One value in the specified set

TABLE 3.4: Ranges a value may take per variable type, based on the range parameters defined in the consequence of a data generation rule.

tps Average number of transactions encountered every second.

n_prof Number of random profiles generated, consisting of a name and a bank account number.

A data stream is constructed on a per transaction basis, creating one transaction and presenting it to the system per iteration. When a transaction is constructed, it is first decided whether it will be legitimate or not, with probability p_{il} of the transaction being illegitimate. Next, the *Date* and *Time* fields are generated. *Date* is set to a random date. *Time* is set to the current time stamp. The time stamp is initialised at 00:00:00 (hh:mm:ss) when a stream starts. Every successive transaction has a chance of $\frac{1}{\text{tps}}$ to increment the time stamp by one second.

The final construction phase of transactions is governed by specified data generation rules (Section 3.2.2). Data generation rules belong to one of three categories:

- General
- Legitimate
- Illegitimate

Rules in the *general* category are applied (first) on every transaction. Depending on whether the transaction was decided to be legitimate or illegitimate, rules from the respective category are applied on the current transaction. Note that applying a data generation rule overwrites variable - value pairs if existing. Whenever a data generation rule is applied, a value is sampled from the type defined in that rule (Section 3.2.2) with specified probability P. How the samples are taken is summarised in Table 3.4.

With probability $1 - P$ a sample with an 'alternative value' is taken. The alternative values per variable type are summarised in Table 3.5. An exception is the *categorical* variable type, which has no alternative value. This type is required to have $P = 1$.

Variable type	**Alternative sample value**
Boolean	true or false (whichever is opposite the one specified)
Uniform distribution	A uniformly drawn sample within $[a, b]$ with $b - a$ added or subtracted
Normal distribution	A sample from (μ, σ) with 3σ added or subtracted
Profile	A (randomly generated) profile not appearing in the set given

TABLE 3.5: 'Alternative values' per variable type defined in the consequence of a data generation rule.

{
Date 12-03-2004
Time 15:50:22
}

TABLE 3.6: Possible legitimate transaction after filling in the default fields.

3.2.3.1 Example transaction generation

To illustrate the construction of a transaction, consider a stream for which at some point in time a legitimate transaction is generated. First, values for `Date` and `Time` are generated. After adding these variables, the transaction looks like in Table 3.6.

Suppose the stream has the following set of data generation rules specified (Table 3.7):

Category	**Rule**	**Variable type**	**Probability**
General	Sender ⇒ [(Ms. S. Stam, NL99BANK0123456789)]	Profile	1
	Recipient ⇒ [(Sir I. Cashalot, GB00AAAA9876543210)]	Profile	1
	Amount ⇒ $(\mu = 10000, \sigma = 2000)$	Normal	1
	Foreign ⇒ false	Boolean	0.8
Legitimate	Amount ⇒ $(a = 10000, b = 15000)$	uniform	1
Illegitimate	Amount ⇒ 20000	categorical	1

TABLE 3.7: A set of data generation rules as may be defined for a stream.

After generating `Date` and `Time`, the data generation rules are applied to generate other variable - value pairs. The rules in the general category are applied first. Starting off with the profile rules, the rule for the sender yields the combinations `Sender name - Ms. S. Stam` and `Send. account - NL99BANK0123456789`. Only one (name, account) combinations is given, so that combination is picked and since the rule is defined for the sender, it operates on those variables. Similarly for the recipient rule, this yields the combinations `Recipient name - Sir I. Cashalot` and `Rec. account - GB00AAAA9876543210`.

For the rule `Amount` ⇒ `Normal` $(\mu = 10000, \sigma = 2000)$, $P = 1$ a value is sampled, say 9876. Since the rule operates on the *Amount* variable, the variable - value pair `Amount`

{

Sender name	Ms. S. Stam
Send. account	NL99BANK0123456789
Recipient name	Sir I. Cashalot
Rec. account	GB00AAAA9876543210
Amount	9876
Date	12-03-2004
Time	15:50:22
Foreign	true

}

TABLE 3.8: Possible transaction after applying the data generation rules in the general category.

{

Sender name	Ms. S. Stam
Send. account	NL99BANK0123456789
Recipient name	Sir I. Cashalot
Rec. account	GB00AAAA9876543210
Amount	12345
Date	12-03-2004
Time	15:50:22
Foreign	true

}

TABLE 3.9: Possible finalised legitimate transaction generated according to the stream defined. It bears resemblance with the transaction in Table 3.2, except that this transaction has less fields.

`- 9876` is generated. Assume that for rule `Foreign` $\Rightarrow$ `Boolean (false),` $P = 0.8$ an alternative value is generated, since $P < 1$. According to Table 3.5, the alternative value for this rule would be *true*. As such, the pair Foreign - true is added to the transaction. The transaction after application of the general data generation rules is shown in Table 3.8.

Based on whether a legitimate or illegitimate transaction is generated, the data generation rules in the respective category are now applied. Since a legitimate transaction is created now, the rules in the legitimate category are applied, which is only `Amount` $\Rightarrow$ `Uniform` $(a = 10000, b = 15000)$, $P = 1$. Suppose this rule generates the pair `Amount - 12345`, overwriting the previously contained pair. The finalised transaction which is presented to the system is displayed in Table 3.9.

In a similar fashion an illegitimate transaction may be constructed. As a matter of fact, the general data generation rule `Amount` $\Rightarrow$ `Normal` $(\mu = 10000, \sigma = 2000)$, $p = 1$ is superfluous, because it will always be overridden by either the legitimate or illegitimate rule specified.

Also note the similarities between the transactions in Table 3.2 and 3.9. By adding several rules to the ones defined in 3.7, it is possible to generate the additional fields in 3.2.

3.3 AGKA components

This section focuses first on the individual components of the AGKA architecture. These components are the database (Section 3.3.1), machine learning (Section 3.3.2), dialogue (Section 3.3.3) and error (Section 3.3.4). Section 3.3.5 is devoted to *knowledge rules*, which are an implementation of the general association rules. Despite not being a component by itself of the architecture, knowledge rules play an integral role within the various components. For this reason, knowledge rules are discussed in a separate section here. The last section describes how the individual components work together to provide a classification (Section 3.4).

3.3.1 Database

The database stores encountered transactions, as described in Section 3.2.1 as well as inferred knowledge rules, which are described in Section 3.3.5. The maximum number of stored transactions is (theoretically) infinite, while the number of stored rules is limited to 10 for each class { legitimate, illegitimate}.

All transactions encountered in a stream are stored inside the database, after the true class is received by the error component (Section 3.3.4). Whenever the utility of a knowledge rule is calculated, all transactions currently contained in the database component are used.

Knowledge rules are inferred by the dialogue component (Section 3.3.3). All inferred rules offered by the dialogue component may be stored, under the condition that a rule turns out to be *useful*. *Useful* is here defined as having a lower cost to apply than to ignore. How costs are calculated is explained in Section 3.3.5.1. If one class already contains 10 rules and a useful rule is found, the least useful rule is discarded, which may be the newly found one.

3.3.2 Machine learning component

The machine learning component serves to extract meaningful regularities from previously encountered transactions contained in the database (Section 3.3.1). Regularities

are expressed as knowledge rules (Section 3.3.5), which form 'arguments' in the dialogue component (Section 3.3.3).

In order to extract knowledge rules from the database, a decision tree is used, the specifics of which are explicated in Section 3.3.2.1. After a decision tree is fit to the data, knowledge rules are extracted from its model. The extraction process is explained in Section 3.3.2.2.

3.3.2.1 Decision tree

The machine learning algorithm responsible for finding patterns in the data is a decision tree implementation, specifically an adaptation from CART (Breiman et al., 1984). The implementation originates from the Scikit-learn environment (Pedregosa et al., 2011). An advantage of the CART approach is that it supports numerical values.

Recall the discussion of decision trees in Section 2.1.2.1. A decision tree can classify novel observations by successively splitting past observations into subsets, until ideally all observations in a subset belong to one class. To determine the quality of a split, here the Gini impurity is used. Let J be the set of all classes (here {legitimate, illegitimate}), while f_i is the fraction of items belonging to class i. The Gini impurity (I_G) can then be calculated using Equation 3.4.

$$I_G(f) = \sum_{i=1}^{J} f_i(1 - f_i) = 1 - \sum_{i=1}^{J} f_i^2 \tag{3.4}$$

From Equation 3.4, it can be inferred that a subset containing just one class yields the lowest value (impurity), namely 0. Other fractions of classes yield a higher impurity.

When fitting a decision tree, the maximal depth allowed is set to 1. In other words, the model of a fitted decision tree is only allowed to consist of (at most) one node with an attribute on which the data is split. In effect, extracting rules from a fitted decision tree yields rules with (at most) one condition. This forces dialogues (Section 3.3.3) to only add one condition at every turn, instead of adding multiple conditions at once.

3.3.2.2 Rule extraction

When a tree is fit, it is possible to extract association rules from its model. Extracting association rules occurs through a recursive process, which traverses all nodes by first accessing the left child and then the right child. A knowledge rule (Section 3.3.5) is

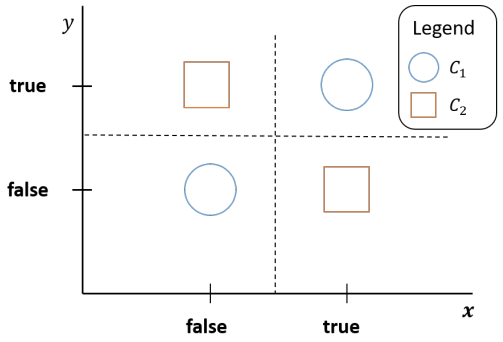


FIGURE 3.1: A data set representing the non-linearly separable XOR problem.

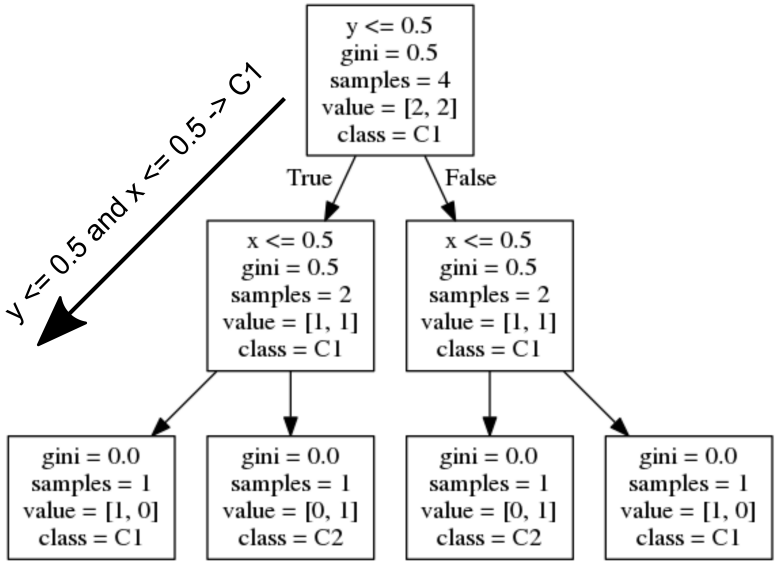


FIGURE 3.2: A potential model after fitting a decision tree to the data depicted in Figure 3.1. Every node of the tree displays the attribute and value for split, the calculated Gini impurity as well as the number of samples remaining. Leafs also display membership of the remaining samples to respective classes. A knowledge rule is depicted which can be extracted by following the left paths.

built while traversing the nodes. Once a leaf is reached, the association rule built up and until that point is stored.

Consider the data set depicted in Figure 3.1. The diligent reader may recognise this set as the XOR problem, a classical example of a set that is not linearly separable (Elizondo, 2006). In principle, it is impossible to draw one straight line separating class C_1 from class C_2. Instead a decision tree 'solves' the problem by drawing two lines or decision boundaries, also shown in Figure 3.1. One possible model of a decision tree fit on this data is shown in Figure 3.2.

Figure 3.2 also shows how the rule $\texttt{y} \leq \texttt{0.5 and x} \leq \texttt{0.5} \Rightarrow \texttt{C1}$ can be deduced or extracted from the model by following the leftmost path to a leaf node. Following all paths to all leaf nodes yields the four rules:

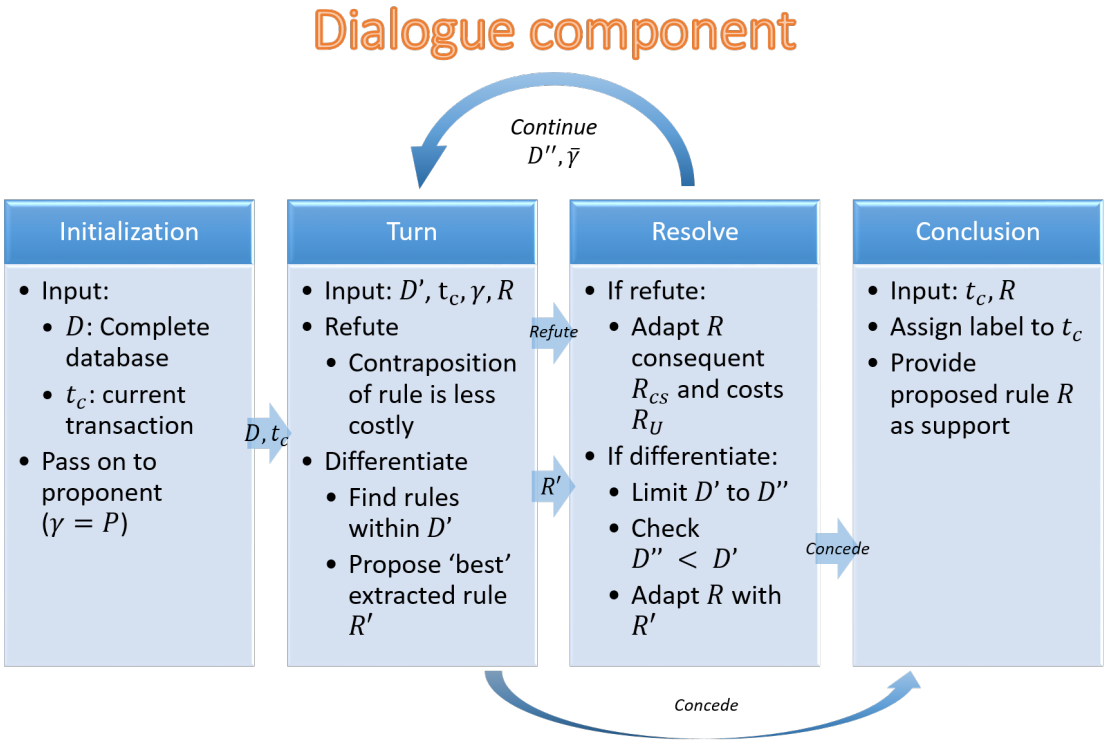


FIGURE 3.3: A visual chart of the dialogue process.

- `y` $\leq$ `0.5 and x` $\leq$ `0.5` $\Rightarrow$ `C1`
- `y` $\leq$ `0.5 and x > 0.5` $\Rightarrow$ `C2`
- `y > 0.5 and x` $\leq$ `0.5` $\Rightarrow$ `C2`
- `y > 0.5 and x > 0.5` $\Rightarrow$ `C1`

3.3.3 Dialogue component

The dialogue component serves to provide understandable support for a classification. The dialogue process is visualised in Figure 3.3. Dialogues consist of four stages: *initialisation*, *turn*, *resolve* and *conclusion*. *Initialisation* and *conclusion* occur exactly once in every dialogue (respectively at the start and the end), while *turn* and *resolve* can occur many times. There are three conditions under which a dialogue is initiated:

1. No rules in the rule base (Section 3.3.1) apply to a transaction.
2. Multiple rules from different classes in the rule base apply to a transaction.
3. An erroneous label was given (Section 3.3.4).

If any of the above three conditions apply, the *initialisation* phase commences, which receives all transactions stored D in the database (Section 3.3.1), as well as the current transaction t_c. The initialisation phase serves to pass on D and t_c to the proponent ($\gamma = P$), who pleads for an illegitimate label. This passing on starts the first turn in a dialogue, without a proposed rule yet ($R = \text{None}$).

During a *turn* phase, either of two parties (proponent $\gamma = P$, or opponent $\gamma = O$) can play one of two moves:

1. **Refute:** With the refute move, an agent denies the label provided by R (which is its consequence R_{cs}). Instead the agent proposes the label it is pleading for (γ). Refuting is allowed when turning R into favour of the current party is less costly.

2. **Differentiate:** Propose an additional condition for R. To find a suitable additional condition, association rules are mined from the provided database D' using the machine learning component 3.3.2. Useful ARs are stored in the rule base (Section 3.3.1). All found ARs are constrained to those applying on t_c. γ proposes the AR which is most in its favour R', which is the rule with the greatest difference between application cost and ignoring cost (Section 3.3.5.1). R' is provided to the *resolve* phase. If no rules can be found, a player *concedes*, leading to the *conclusion* phase.

Refuting has precedence over differentiating. That is, an agent will try to refute if possible and only differentiate if it cannot refute. Both moves lead to a *resolve* phase, with the exception of *conceding*. What happens in the resolve phase depends on the move played:

Refute: Adapt R by changing its consequence R_{cs} to γ and swapping its costs R_U. In effect R is changed to its contra-position (Section 3.3.5.1).

Differentiate: Several operations take place after a differentiate move:

1. Limit D' to transactions which satisfy the additional condition R'_{cd}, the condition of R'. The result is D''.
2. Check if $D'' < D'$. That is, D'' ought to contain less transactions than D', otherwise the move is not actually a differentiation. Should this check fail, then γ is forced to *concede*.
3. Adapt R by adding R'_{cd} as a conjunct to R_{cd}, the condition of R. R_{cs} is set to R'_{cs}, which is equal to γ since in the *turn* phase γ turned all applicable rules in its favour.

If the resolve phase turns out successfully, another turn phase is commenced with D'', t_c, $\bar{\gamma}$, which is the opponent if $\gamma = P$ and vice versa, as well as (the adapted) R. Recommencing the turn phase coincides with arrow 1 in Figure 3.3.

If anywhere during the cycle a party *concedes*, the *conclusion* phase takes place. During the conclusion phase the final label is proposed as classification for t_c. The final label is R_{cs} and R is provided as support. R is also stored in the rule base.

3.3.4 Error component

After AGKA provided a label for a novel transaction, 'external feedback' is received about the true class of that transaction. The external feedback is implemented as a function that indicates whether the given label and the true class coincide or not. The feedback is delivered to the error component, which acts upon the outcome.

If the label is correct (no error is made), no actions need to be undertaken. Whenever the system does make an error, that indicates that classification gave a wrong label to a transaction. To prevent similar errors from occurring in the future, the error component activates a process to adapt classification. This process consists of two parts:

1. Recalculate the utility of all association rules involved in the erroneous classification.
2. Initiate a dialogue to find the most applicable rule to the transaction classified wrongly.

Since beneficial inferred rules are stored in the database and these rules are not continuously reevaluated, part 1 is required: it can be the case that an inferred rule was useful in the past, but new transactions yield counter evidence. It may also hold in case a rule resulting from a dialogue is applied. Such a rule may only apply to very specific transactions, so another example may have a big impact on the utility of that specific association rule. Should the expected utility of applying a rule turn out to be lower than ignoring it, it is deleted from the database.

Initiating another dialogue, part 2 also aids in updating inferred knowledge rules. Since this transaction caused the system to make an error, it is in some respect deviating from the regularities (or rules) inferred before. That is, this transaction may contain useful information which differentiates it from others. This difference may be captured by a new rule, which can be found by means of a dialogue.

The two parts of the error process can thus be regarded as operations on the knowledge inferred: 1 *evaluates past knowledge*, while 2 *learns new information* based on the mistake made.

3.3.5 Knowledge rules

A *knowledge rule* is another type of AR. Knowledge rules are used to stipulate regularities found in encountered transactions. In addition to a general AR, knowledge rules may have an *expected utility*. Hence, the definition of a knowledge rule becomes:

$$< Cd, Cs, P, EU, T > \tag{3.5}$$

with the following specifications:

Cd A rules' condition, consisting of constructs, such as `foreign = false AND average amount < 10000`.

Cs Consequence or classification of a rule. Here it resembles the most likely classification of a transaction. ($Cs \in \{legitimate, illegitimate\}$)

P Probability of the consequence Cs being correct if the condition Cd is satisfied by an instance. This probability is calculated from the database. Suppose 1000 stored transactions satisfy a rule's condition and 990 belong to the class signified by its consequence, say `legitimate`, then P is computed to be $\frac{990}{1000} = 99\%$.

EU Expected utility of a rule, computed as a tuple of costs for applying or ignoring the rule. How the expected utility is computed is explained in section 3.3.5.1.

T In this case the rule type is a knowledge rule.

3.3.5.1 Rule utility

The expected utility of a rule aids in deciding whether or not to use or 'believe' a rule. Once a rule is formulated, one can decide to either accept and store it, or ignore and delete it. In the framework of expected utilities (Section 2.3) these can be considered acts. The act with the highest expected utility is considered the best decision in a situation.

To calculate the expected utility of every action, priors of states and utilities of outcomes are required. With respect to association rules, the states of a rule can be that the

classification is either correct or incorrect. The prior of state 'correct' (P(correct)) is the probability of a rule's classification being correct for satisfying instances (which is the same as P in Section 3.3.5). Conversely, the prior for state 'incorrect' (P(incorrect)) is 1 minus the prior for state 'correct'.

P(correct) and P(incorrect) are computed from the database using Equation 3.6.

$$P(C) = \frac{\sum_{i=I} f_a(i) \cdot \text{equals}(C, i_C)}{\sum_{i=I} f_a(i)} \tag{3.6}$$

Here C is the classification or consequence of a rule, I the set of all instances or transactions stored, i_C the class of an instance and $f_a(i)$ a function that equals 1 if the current rule applies to instance i and 0 otherwise. In words, the probability of a rule being correct equals the amount of times its class equals the class of an instance it applies to, divided by the number of times it applies to an instance.

The actions taken in the 'correct' and 'incorrect' states lead to distinct outcomes. The acts, states and their respective outcomes are summarised in Table 3.10. If a rule leads to the correct classification and it is applied, this is called a hit, while ignoring it in such a circumstance is a miss. Should a rule's classification be incorrect then the rule can be considered as misleading or misguiding, because it steers toward a bad decision (assigning the wrong class). Should a rule be applied in such a situation, then the classification is misguided, while ignoring it can be seen as a revised classification.

		Classification	
		correct	incorrect
Action	apply	hit	misguided
	ignore	miss	revised

TABLE 3.10: Consequences for applying a rule or not while the classification is either correct or not.

Expected utility adds onto probability, because certain decisions may be more beneficial than others, even though the probability of correctness (accuracy) may be the same. This is reflected in the utility of the various outcomes.

The outcomes in Table 3.10 do have different utilities or 'desirability values'. Intuitively, it is preferable to apply a rule if it is correct and to ignore it if it is incorrect. On the other hand, applying an incorrect rule or ignoring a correct one should be avoided.

For quantifying the utilities of the outcomes in Table 3.10, a difference has to be made between rules discerning legitimate transactions and rules discerning illegitimate cases. The aim of applying legitimate rules is to diminish costs for verifying transactions (assigning the class 'legitimate' means a transaction is not checked), while the aim of

applying illegitimate rules is to diminish the amount of money lost by an illegitimate transaction. From this point forward, the utilities are referred to as costs. In this setting, it is more intuitive to talk about the outcomes incurring a cost, rather than some benefit. In reality, four different situations exist, which involve whether or not a transaction is actively verified (phone calls with the presumed victim etc.). Suppose the cost of such a verification occupies one person with one hour of labour at a gross income of κ Euro an hour. Moreover, if a fraudulent transaction occurs, the transferred amount λ is either lost or it can be recovered. The four different situations can then be described as follows:

1. A legitimate transaction is classified as such. No costs are incurred. The costs of this outcome are thus zero ($Cost(o) = 0$).
2. A legitimate transaction is classified as illegitimate. The costs of verification κ are expended, but nothing (λ) is gained nor lost, since the warning was a false positive. Hence, $Cost(o) = \kappa$.
3. An illegitimate transaction is classified as such. Again κ is expended, but the transferred amount λ may be recovered (it is not lost, this is not a cost). Similarly, $Cost(o) = \kappa$.
4. An illegitimate transaction is classified as legitimate. It is not verified, so κ is not incurred, but λ is lost. Hence, $Cost(o) = \lambda$.

In terms of costs, options 2 and 3 are similar. It is presumed that $\lambda > \kappa$, otherwise it would be more costly to prevent fraud than to let it happen. Comparing above four situations with the outcomes in Table 3.10, leads to Table 3.11 for legitimate rules and Table 3.12 for illegitimate rules.

		Classification	
		correct	incorrect
Action	apply	$Cost(o) = 0$	$Cost(o) = \lambda$
	ignore	$Cost(o) = \kappa$	$Cost(o) = \kappa$

TABLE 3.11: Incurred costs for the outcomes of applying a rule classifying regular cases or not.

		Classification	
		correct	incorrect
Action	apply	$Cost(o) = \kappa$	$Cost(o) = \kappa$
	ignore	$Cost(o) = \lambda$	$Cost(o) = 0$

TABLE 3.12: Incurred costs for the outcomes of applying a rule discerning fraudulent cases or not.

3.3.5.2 Calculation example

To clarify how cost affects the usability of association rules, consider the data set depicted in Figure 3.4. Transactions are displayed on two axes, namely *post-balance* and *amount*, respectively representing the money left on an account after the transaction and the amount of money transferred.

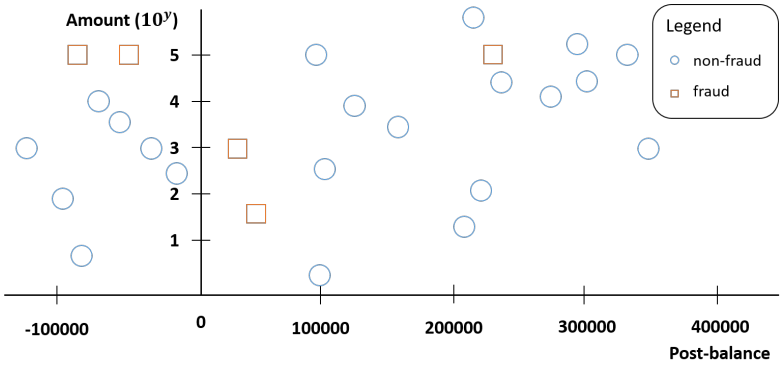


FIGURE 3.4: An example data set to calculate the utilities of association rules.

Consider the rule `post-balance` $\leq$ `0 -> illegitimate`. Since this is an illegitimate rule, Table 3.12 is applied and the cost for applying the rule can immediately be determined as k, or 5000. Of all transactions to which the rule applies, 8 are legitimate and 2 illegitimate, so the rule is correct in $\frac{2}{8+2} = 20\%$ of cases. The average of `amount` for the illegitimate transactions (consistent with the rule's classification) $\lambda = 100000$. Consequently, the cost of ignoring this rule equals $0.2 \cdot 100000 + 0.9 \cdot 0 = 20000$. Hence, applying this rule is less costly than ignoring it ($5000 < 20000$). In other words, despite the rule being inaccurate, the 'severity' of missing the illegitimate instances makes checking all applying instances worthwhile nonetheless.

Now consider the rule `post-balance` $>$ `200000 -> legitimate`. This time Table 3.11 applies and the cost of ignoring it can immediately be determined as $k = 5000$. For this rule also, $\lambda = 100000$ (inconsistent with the rule's classification). The accuracy of this rule is $\frac{9}{9+1} = 90\%$, implying the cost of applying it equals $0.1 \cdot 1000000 = 10000$. As a result, applying the rule is costlier than just ignoring it ($10000 \geq 5000$). In other words, despite the rule being accurate and `amount` being relatively high for legitimate transactions, again the 'severity' of an illegitimate transaction prompts checking all applying transactions.

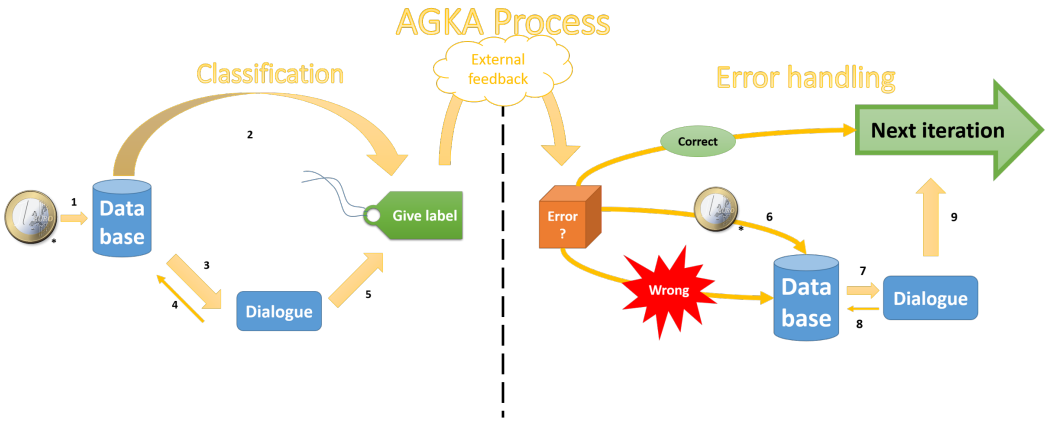


FIGURE 3.5: A visual chart of how all AGKA components are combined to provide classifications. *"*Euro Coin Transparent Background*" by *Eric* is licensed under *CC BY 2.0*

3.4 AGKA process

With each of AGKA's components explained, it is time to turn to how the system as a whole functions to classify transactions. What happens during every iteration is visually represented in Figure 3.5. The respective arrows and components depicted correspond to the following actions:

1 At the onset of an iteration, a novel transaction t_c is generated and offered to the system.

Database The database component checks whether stored knowledge rules apply.

2 At least one stored knowledge rule applies to the novel transaction. All applying rules share the same consequence or class, so that class is proposed as label.

3 There are no stored rules which apply, or multiple rules with different classes apply. These are two out of three conditions to initiate a dialogue (Section 3.3.3).

Dialogue A dialogue is initiated with the entire database and t_c.

4 Useful knowledge rules found by the dialogue component are stored in the database (rule base). These rules may have been found during the dialogue or it may be the final rule proposed.

5 The label proposed by the dialogue component is applied to t_c.

External feedback Feedback concerning the transaction's class arrives.

Error? The error component (Section 3.3.4) determines whether the system made an error (given label does not correspond with true class).

6 Regardless of the outcome, t_c is stored in the database.

Correct If there was no error, the system immediately continues to the next iteration.

Wrong If a mistake was made, the database component is notified of the error.

Database The rules responsible for the classification are reassessed in the rule base. Rules are deleted if they are no longer useful (costlier to apply than to ignore).

7 A dialogue is initiated, under the last condition that an error was made.

Dialogue Another dialogue is initiated with the entire database, this time including t_c. Again a classification is sought for t_c.

8 The dialogue component may store useful rules in the rule base in the same way as in step 4.

9 After the dialogue finishes, continue to the next iteration.

Next iteration Start the process over from step 1.

Chapter 4

Illustrative cases

In this chapter several examples are discussed, to illustrate step-by-step how the system as a whole handles detecting fraudulent transactions.

4.1 Binary decision

Suppose transactions consist of one attribute, namely `Foreign`, which states whether the recipient resides in a different country than the sender. Furthermore suppose all foreign transactions are illegitimate. In other words, the data adheres to the data generation rules `Foreign` $\Rightarrow$ `true` with $p = 1$ for the illegitimate category and `Foreign` $\Rightarrow$ `false` with $p = 1$ for the legitimate category. A data set with both a legitimate and an illegitimate transaction is depicted in Figure 4.1.

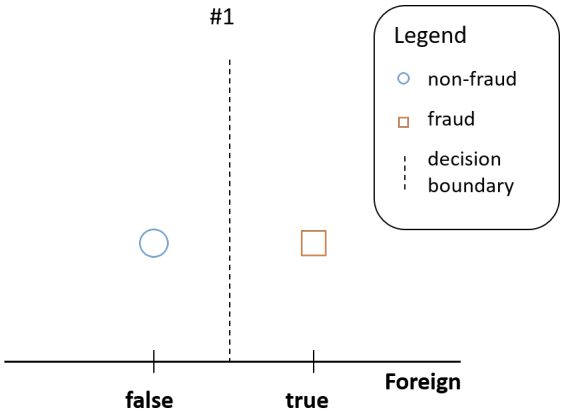


FIGURE 4.1: A set where legitimate and illegitimate transactions can be discerned based on the binary attribute `foreign`. The drawn decision boundary shows where the two classes are separated.

Keep in mind that the data is a stream in nature. In other words, the data set depicted in Figure 4.1 is accumulated over time. At the onset, there are no transactions from which inferences can be made.

Now suppose a transaction with `Foreign = false` is encountered. According to the definition, this transaction is then legitimate. For every novel transaction, the system passes the steps described in Section 3.4:

1 Generate the transaction and present it to the system. This trivial step is omitted from hereon.

Database Check the database for inferred association rules that apply to the current transaction.

3 At the onset, no rules are in the rule base yet, so a dialogue is initiated.

Dialogue (initialisation) The complete database of past transactions (none), as well as the transaction in question, is provided to the proponent, pleading for an 'illegitimate' label.

Turn The proponent (also referred to as P) cannot refute the proposal, since there is none yet. Hence, only a differentiate move is an option. Proponent tries to find rules present within the provided transactions. Since there are no transactions, finding rules fails, resulting in the proponent conceding.

Conclusion The dialogue fails, so the proposed label defaults to 'legitimate' without support.

External feedback and error? Give the labelling to the novel transaction and compare that to the 'true' label. In this case, the transaction was indeed legitimate, so the system proved to be correct.

6 Store the novel instance including its true label in the database.

Correct Act according to whether an error was made or not. Since no error was made, no adjustments are applied.

After these steps are finished, another novel transaction from the stream can be considered, starting the process over. Suppose this time an illegitimate transaction is encountered, with `Foreign = true`. The previous steps are repeated:

Database The database still does not contain inferred knowledge rules, so there are no rules applying on this transaction.

3 and dialogue A dialogue is initiated. The entire database, consisting of the previous transaction, is provided to the proponent.

Turn Again, a differentiate move is the only option for P. This time however it can find a rule. Since all encountered transactions belong to one class (legitimate), the knowledge rule ⇒ `legitimate` may be inferred. It is also useful, because it is 100% correct ($p = 1$), so it is stored in the database. However, that rule is not in favour of P, so instead it proposes the counter rule ⇒ `illegitimate` with $p = 0$.

Resolve Although this is a legal move, the fact that the database is not shrinking by this differentiation (rule applies to one instance, while the database already contained one) means P has to concede.

Conclusion Dialogue failed again, so the default label 'legitimate' is proposed.

Error? The label 'legitimate' is given to the novel transaction. However, comparing that labelling to the true class reveals that the system made an error.

6 The novel transaction including its true class is stored. Note that this step is omitted for the rest of this discussion, since it is trivial and recurring in every process.

Wrong and database The system is notified of its error, prompting it to reassess the rules that led to the wrong classification. Since the label was given by default, without support, there are no rules no reevaluate.

7 The dialogue initiated now does yield interesting information though, because the database now contains an additional, meaningful transaction:

Initialisation A dialogue is initiated and P is provided with the complete database (containing one legitimate and one illegitimate transaction) and the transaction in question (the wrongly classified illegitimate transaction).

Turn A differentiate move is still the only option for P. For finding applicable rules, fitting a decision tree would now split on the variable *Foreign* and the rule extraction procedure yields the rules `Foreign = false` ⇒ `legitimate` and `Foreign = true` ⇒ `illegitimate`. Since both turn out with $p = 1$ according to the database, both knowledge rules are stored. P will bring forward the latter rule, since it applies on the transaction in question and is best in its favour ($p = 1$).

Resolve `Foreign = true` ⇒ `illegitimate` is accepted. The database is limited to transaction on which the proposed rule applies (the single illegitimate one). The limited database, as well as the transaction in question, is passed on to the opponent (or O).

Turn and resolve O cannot refute the proposed rule, since the counter rule `Foreign = true` ⇒ `illegitimate` is never correct ($p = 0$). O cannot differentiate either, since only one transaction remains. In effect O concedes.

Conclusion P is regarded the winner of the dialogue and its pleading, 'illegitimate', is brought forward as label, with the knowledge rule `Foreign = true` ⇒ `illegitimate` as support. Since the database already contains this knowledge rule, it is not stored again.

Notice that the iteration now ended. Despite disregarding the label provided after the dialogue in the error phase (there is no need classifying the same transaction again), that dialogue did have an impact on the system: During the dialogue, the rules `Foreign = false` ⇒ `legitimate` and `Foreign = true` ⇒ `illegitimate` were stored in the database. As a matter of fact, this additional knowledge perfectly models the definition of legitimate and illegitimate transaction in the stream. Consequently, the system consolidates in a shorter process, which becomes apparent with additional transactions. Suppose the next transaction in the stream is legitimate (`Foreign = false`):

Database The stored knowledge rule `foreign = false` ⇒ `legitimate` is the only rule that applies to this transaction.

2 The label 'legitimate' is given to the transaction. This labelling does not conflict with the true class, so no error is detected and no further action is undertaken.

The run is similar for illegitimate transactions, with the exception that the other stored rule is used instead. Since all transactions are correctly classified by the inferred rules, no errors are made and no adjustments to the database are required.

4.2 Continuous values

The present discussion should however not be constrained to binary, or even multi-valued, attributes. An advantage of the CART algorithm is that it can handle continuous values. As a matter of fact, binary values are converted to 0 and 1 for respectively false and true. A split of a binary attribute is then translated into ≤ 0.5 for false and > 0.5 for true.

Suppose transactions now, instead of the binary attribute `Foreign`, consist of one continuously valued attribute `Dif. avg.`, which represents the difference between the currently transferred amount and the average amount transferred by that account. Assume there is a (hard) boundary, where the value of `Dif. avg.` for legitimate transactions never exceeds 12500, while illegitimate transactions always have a higher value.

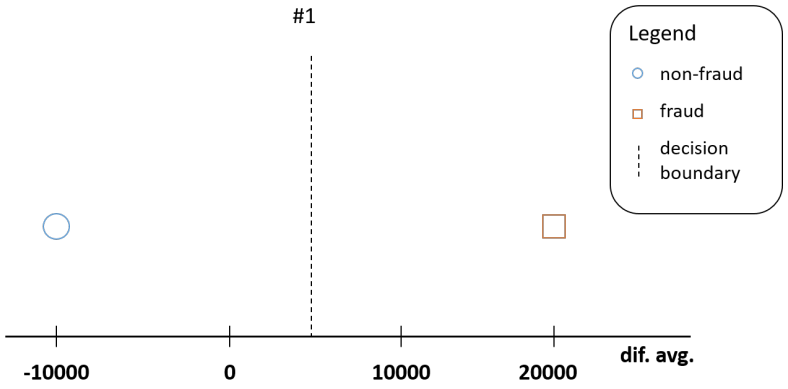


FIGURE 4.2: The first decision boundary found for the continuously valued `Dif. avg.`

For as long as only one class of transactions is encountered, the system keeps undergoing the same process as described in Section 4.1. Suppose at some point in time, the situation depicted in Figure 4.2 presents itself, where the illegitimate transaction is encountered last. The system will classify this wrongly, because it gives the default label 'legitimate', just as in Section 4.1. The error component however prompts the system, or more specifically the decision tree, to infer the boundary depicted:

Wrong and database Reevaluating responsible knowledge rules cannot be done, since the error resulted from assigning the default label 'legitimate'.

Dialogue A dialogue is initiated.

Turn P does a differentiate move. Finding rules includes fitting a decision tree, which calculates the best split to be on the attribute `Dif. avg.` at value $\frac{20000+-10000}{2} = 5000$. The extraction procedure yields the knowledge rules `Dif. avg.` $\leq 5000 \Rightarrow$ `legitimate` and `Dif. avg.` $> 5000 \Rightarrow$ `illegitimate` from the split. Both rules are stored, because both are correct all the time. P proposes the rule `Dif. avg.` $> 5000 \Rightarrow$ `illegitimate`.

Resolve The proposed rule is accepted and the database is limited to the illegitimate transaction with `Dif. avg. = 20000`, because it constitutes 'all transactions with `Dif. avg.` > 5000'.

Turn O concedes because it is not allowed to play either move.

Notice that from this point forward, the decision boundary as depicted in Figure 4.2 is maintained, which is described by the two inferred knowledge rules. However, since we know the found boundary (5000) deviates from the true cutoff (12500), at some point in time, a novel transaction will lead the system to an error. This novel transaction will be legitimate, since it falls in the range $< 5000, 12500]$ (classified as illegitimate because

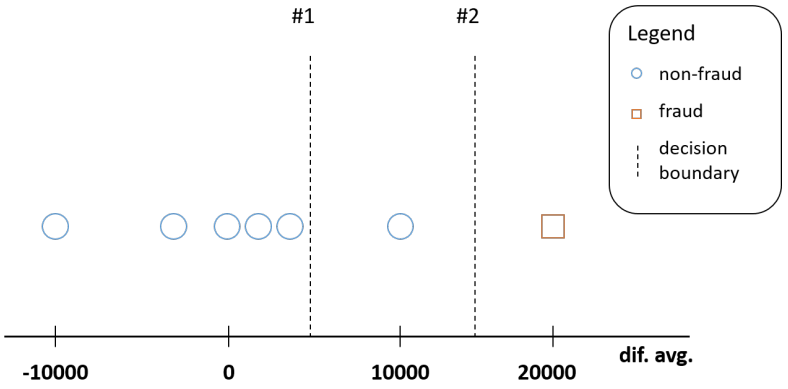


FIGURE 4.3: The second decision boundary found after making an error based on the first boundary.

the value is higher than 5000, but actually legitimate because it is lower than or equal to 12500). Suppose this legitimate transaction has the value 10000 for the `Dif. avg.` attribute. This situation is depicted in Figure 4.3. Consider what happens during the iteration in which the error is made:

2 The only applying rule is `Dif. avg.` $>$ `5000` $\Rightarrow$ `illegitimate`, so the label 'illegitimate' is given to the novel transaction.

Error? The given label 'illegitimate' does not agree with the class 'legitimate'.

Wrong and database Upon notification of the error, the system reassesses the single applied rule, which turns out to be no longer useful, as the current database shows that is incorrect half of the time. Consequently, that knowledge rule is deleted from the database.

Dialogue A dialogue commences:

Turn During P's turn, fitting a decision tree makes a new split at value $\frac{10000+20000}{2} = 15000$, yielding the rules `Dif. avg.` $\leq 15000 \Rightarrow$ `legitimate` and `Dif. avg.` $> 15000 \Rightarrow$ `illegitimate`. Both rules are stored since they are always correct. The only applicable rule left is `Dif. avg.` $\leq 15000 \Rightarrow$ `legitimate`, so P proposes its favourable form `Dif. avg.` $\leq 15000 \Rightarrow$ `illegitimate`.

Resolve The proposal is accepted and the database is limited.

Turn O refutes the proposal, because the counter rule `Dif. avg.` $\leq 15000 \Rightarrow$ `legitimate` is 100% accurate, which is much better than the proposed 0%.

Resolve The proposed rule is adapted to the counter rule. The database stays the same, because a refute move is played.

Turn P concedes as it cannot refute the proposal, nor propose a distinction.

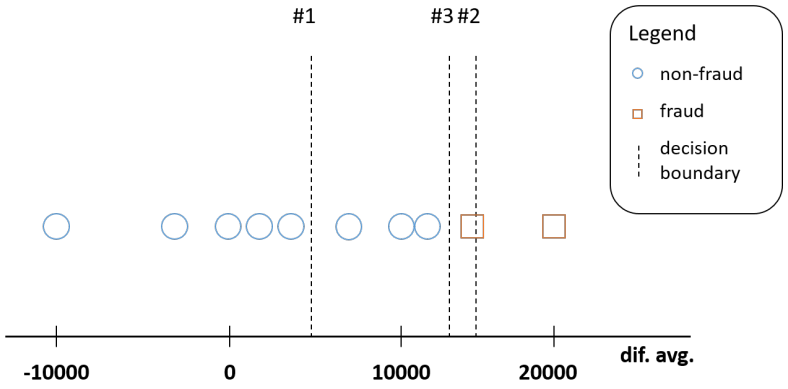


FIGURE 4.4: The third decision boundary found after the system made another error.

It can be seen that in this iteration the knowledge rule `Dif. avg.` $> 5000 \Rightarrow$ `illegitimate` is replaced in favour of `Dif. avg.` $> 15000 \Rightarrow$ `illegitimate`. We know however that this adjustment 'overshot' the true boundary of 12500. Therefore, eventually the system might make another error. Figure 4.4 depicts such a moment in time. Suppose a novel transaction contains `Dif. avg.` `= 14000`, which means it is illegitimate according to the definition. Consider the response of the error component to this mistake:

Database The knowledge rule `Dif. avg.` $\leq 15000 \Rightarrow$ `legitimate` is solely responsible for the error. The recalculated accuracy (89%) does not warrant it's deletion however.

Dialogue The dialogue in summary yields the additional rules `Dif. avg.` $\leq 13000 \Rightarrow$ `legitimate` and `Dif. avg.` $> 13000 \Rightarrow$ `illegitimate`, in a similar fashion as earlier dialogues.

The database now contains the knowledge rules:

1. `Dif. avg.` $\leq 5000 \Rightarrow$ `legitimate`
2. `Dif. avg.` $\leq 13000 \Rightarrow$ `legitimate`
3. `Dif. avg.` $> 13000 \Rightarrow$ `illegitimate`
4. `Dif. avg.` $\leq 15000 \Rightarrow$ `legitimate`
5. `Dif. avg.` $> 15000 \Rightarrow$ `illegitimate`

Arguably rule 1 and 5 have become obsolete, since rule 2 covers 1 already, while 3 covers 5. Deleting these seemingly obsolete rules would be fundamentally wrong though. For one, knowing these rules are obsolete is based on the background knowledge that there

exists a true boundary. In fact there might be 'patches' of each class within one attribute, where multiple boundaries are required to differentiate the classes.

Also notice that a transaction with `Dif. avg. = 14000` gives rise to conflict, because rule 3 and 4 both apply, which have different classifications. Deleting the older rule would however not be a viable solution. Shifting entire boundaries rather than reassessing rules may have detrimental effects in the long run. Why this is the case, and also how the system copes with 'patches' becomes apparent in the next example.

Before turning to the next example, it is worth mentioning that this example shows how the system slowly converges to the true boundary. Every successive split made shifted closer toward the true boundary.

4.3 Multiple (binary) attributes

Until now, we have assumed transactions consist of one attribute. However, transactions do not consist of a single attribute. A more realistic example is one where every transaction has three fields, namely `Foreign`, `Known` and `Night`. Respectively meaning: the recipient resides in a different country than the sender, the sender transferred money to the recipient before and the transaction took place between 22:00 and 6:00 (time zone of sender). Let illegitimate transactions be uniquely determined by the combination {`Foreign = true, Known = false, Night = true`}. Legitimate transactions may take any (binary) value for these attributes, but never in this combination.

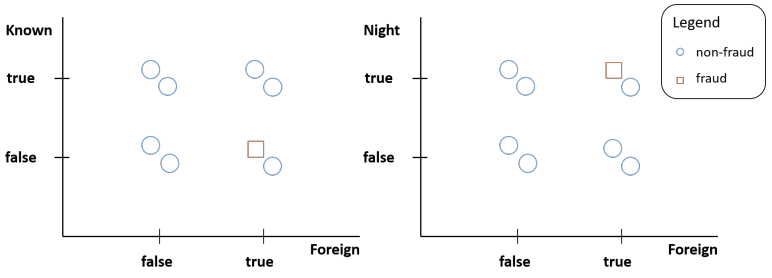


FIGURE 4.5: Example to show how multiple attributes are handled. The attributes `known` and `night` are plotted against `foreign`. Note that the scattering of transactions with the same combination of values is merely to aid visibility.

Since the processes are nearly identical as elaborated before, we'll skip forward to a point in time where all possible combinations for legitimate transactions have been populated and an illegitimate one is encountered. For convenience of calculation, presume every combination of values is encountered exactly once. This situation is depicted in Figure 4.5. The following dialogue initiated by the error component after wrongly classifying the illegitimate transaction shows how the system recovers from its error:

Turn P can only differentiate. Fitting a decision tree with depth = 1 can split at any attribute, since the quality of any split is equal. Assume a split is made at `Foreign = 0.5`, yielding the rules `Foreign` $\leq 0.5 \Rightarrow$ `legitimate` and `Foreign` $> 0.5 \Rightarrow$ `legitimate`. Both rules are saved, since they respectively carry an accuracy of 100% and 75%. The applying (counter) rule most in favour of P is `Foreign` $> 0.5 \Rightarrow$ `illegitimate` with an accuracy of 25%, which P therefore proposes.

Resolve Database is limited and the proposed rule becomes `Foreign` $> 0.5 \Rightarrow$ `illegitimate`

Turn O refutes the proposal.

Resolve The proposed rule is adapted to `Foreign` $> 0.5 \Rightarrow$ `legitimate`.

Turn P cannot refute, but proposes a further distinction. Fitting a decision tree on the limited database yields the rules `Known` $\leq 0.5 \Rightarrow$ `legitimate` and `Known` $> 0.5 \Rightarrow$ `legitimate` with respectively 50 and 100% accuracy. P proposes `Known` $\leq 0.5 \Rightarrow$ `illegitimate` with 50% accuracy.

Resolve The database is limited and the proposed rule becomes `Foreign > 0.5 and Known` $\leq 0.5 \Rightarrow$ `illegitimate`.

Turn O cannot refute, because 50% is not better than 50%. Fitting a decision tree now yields the rules `Night` $\leq 0.5 \Rightarrow$ `legitimate` and `Night` $> 0.5 \Rightarrow$ `illegitimate`, both with 100% accuracy. O proposes `Night` $> 0.5 \Rightarrow$ `legitimate` with 0% accuracy.

Resolve Database is limited and the proposed rule becomes `Foreign > 0.5 and Known` ≤ 0.5 `and Night` $> 0.5 \Rightarrow$ `legitimate`.

Turn P refutes.

Resolve The proposed rule is adapted to `Foreign > 0.5 and Known` ≤ 0.5 `and Night` $> 0.5 \Rightarrow$ `illegitimate`.

Turn O concedes.

Conclusion The label 'illegitimate' is proposed. The knowledge rule `Foreign > 0.5 and Known` ≤ 0.5 `and Night` $> 0.5 \Rightarrow$ `illegitimate` is given as support and stored.

Storing the supportive knowledge rule makes the system make no errors anymore. Remember that only the rules `Foreign` $\leq 0.5 \Rightarrow$ `legitimate` and `Foreign > 0.5 and Known` $\leq$ `0.5 and Night` $> 0.5 \Rightarrow$ `illegitimate` are saved. Nevertheless, once a transaction is encountered on which neither rule applies, P will eventually lose, since O can propose either the rule `foreign > 0.5 and known` > 0.5 `-> legitimate` or `foreign > 0.5 and night` ≤ 0.5 `-> legitimate`, both of which are always correct.

Hence the legitimate label is given, which is correct according to our definition, prompting storage of these association rules and eventually settling the system into a stable solution.

This example is similar to the example from Section 4.1, with the exception that the dialogue involves several steps. As such, the example may be extended to continuous values and even noisy data. These extensions will not be discussed here for brevity.

4.4 Rules with utility

Until now we have ignored the role of utilities. The motivation behind storing a rule or discarding it, depends on whether it is more useful to use it or not. In the previous examples, a rule is stored whenever it is correct in over 50% of applying transactions. Nevertheless, the utility of a rule encompasses more than just the prior likelihood: the desirability of outcomes is also incorporated. As described in Section 3.3.5.1, the attribute `Amount` is taken as a measure of how much impact a certain transaction has on misclassifications.

To illustrate the impact of utility on the classification process, consider the following data stream: Besides amount, transactions contain the binary attributes `Foreign` and `Known` (with the same meaning as previously). Suppose the data is scattered as depicted in Figure 4.6. The illegitimate transactions with `Foreign = true` have `Amount = 50000`, while the illegitimate transactions with `Known = true` have `Amount = 10000`. Assume the system already inferred the rules `Foreign` $> 0.5 \Rightarrow$ `illegitimate`, `Known` $> 0.5 \Rightarrow$ `illegitimate` and `Foreign` $\leq$ `0.5 and Known` $\leq 0.5 \Rightarrow$ `legitimate`. The expected utilities (EU) are respectively: {5000, 50000}, {5000, 10000} and {0, 5000}

Once a transaction with `Foreign = true` and `Known = true` is observed, we see the impact of utility by running through (part of) the process once more:

Database Both illegitimate rules, `Foreign` $> 0.5 \Rightarrow$ `illegitimate` and `Known` $> 0.5 \Rightarrow$ `illegitimate`, apply.

2 There is no conflict, so the 'best' rule `Foreign` $> 0.5 \Rightarrow$ `illegitimate` is applied. This rule is considered the 'best' since its EU ($50000 - 5000 = 45000$) is highest ($10000 - 5000 = 5000$, which is < 45000).

Suppose the novel transaction is indeed illegitimate and so are all future ones with the same combination. In that case, the latter rule will always be given as support for the

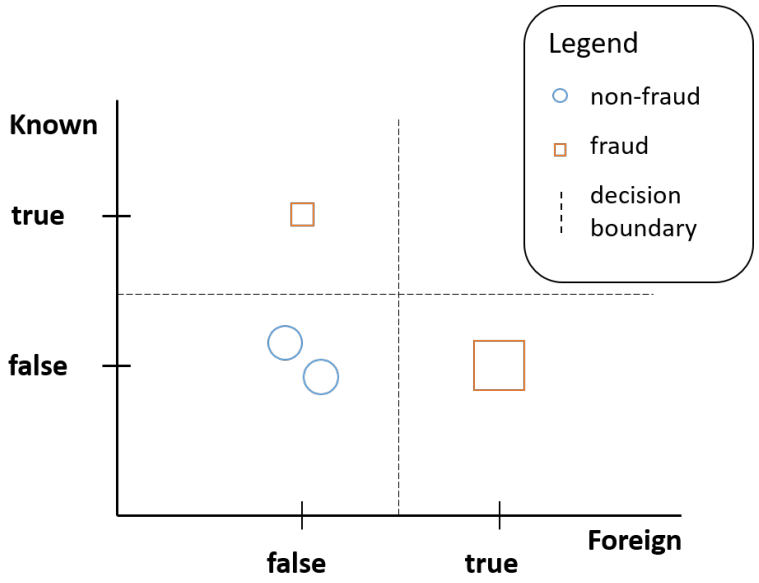


FIGURE 4.6: Data stream illustrating the effect of utility. Bigger fraud transactions contain a higher value of the attribute `amount`. The scattering is merely for the sake of visibility.

classification, rather than the 'weaker' rule `Known` $>$ 0.5 $\Rightarrow$ `illegitimate`, since its expected utility is higher.

In contrast, if the transaction and all future ones with the same combination turn out to be legitimate, utility will have an impact on the proposed and stored rules. Consider the following dialogue after the erroneous label 'illegitimate' was given.

Dialogue commences

Turn and resolve P proposes the rule `Foreign` $>$ 0.5 $\Rightarrow$ `illegitimate`, since it is still the strongest rule it may propose. Its expected utility is now $25000 - 5000 = 20000$.

Turn and resolve O will propose the distinction `Known > 0.5`. The compound association rule `Foreign > 0.5 and known` $>$ 0.5 $\Rightarrow$ `legitimate` has EU $= \{0, 5000\}$.

Turn P cannot counter the proposed rule.

Conclusion O wins the argument and the label 'legitimate' is accredited, with the association rule `Foreign > 0.5 and known` $>$ 0.5 $\Rightarrow$ `legitimate` as support. This rule is also stored.

Another way in which utility may have effect is if the classes can no longer be perfectly separated. Suppose there are also legitimate transactions with `Foreign = true` and `Known = false`. Since the rule `Foreign` $>$ 0.5 $\Rightarrow$ `illegitimate` is less discriminate, it can be calculated that its expected utility decreases (refer to Section 3.3.5.1). As a

matter of fact, should the ratio between legitimate transactions with this combination and illegitimate ones exceed 9:1, then considering all these cases as legitimate regardless would prove less costly than checking all of them.

Chapter 5

Experimental setup

In this chapter the setup for the experiments is discussed, which allow an assessment of the performance of the AGKA architecture. Section 5.1 summarises other machine learning algorithms to compare AGKA's performance to. These algorithms are integrated into the AGKA architecture. How the integration is achieved is described in Section 5.1.1. The chapter concludes with a discussion of the measures used to compare the approaches (Section 5.2).

5.1 Methods of comparison

To assess AGKA's performance, its accuracy is compared to several other machine learning algorithms.

All these algorithms have been limited to train on only the numerical attributes. This is mainly due to several algorithms, such as KNN, only being able to cope with numerical data. Incorporating categorical attributes would require defining some function(s) to turn categorical values into numerals, which is theoretically debatable.

The algorithms used are:

Legitimate regardless (Dummy). Always proposes the label 'legitimate'. Since a vast majority of transactions is not illegitimate, an algorithm that is oblivious to differences between both classes may still score very well.

k-Nearest neighbours (KNN) (Cover and Hart, 1967). KNN looks at the k data points 'closest' to the current one. Whichever class is dominant inside the closest neighbours is provided as classification.

Support vector machine (SVM) (Cortes and Vapnik, 1995). A support vector machine works by finding boundaries or support vectors between classes. These boundaries are made with as big a margin as possible. To do so, an SVM uses kernels to increase the dimensionality within the data. The SVM used here is a C-support SVM using a linear kernel.

CART (DT) (Breiman et al., 1984). Despite AGKA already using CART, a comparison with CART by itself is appropriate. Since AGKA uses (extracted) association rules stored in a database, a dialogue component as well as utilities, it is useful to see whether these added components yield any advantages over CART by itself.

Random forest (RF) (Breiman, 2001). Rather than using just one decision tree, random forest is an ensemble method that utilises a multitude of decision trees to provide classifications. The number of trees is set to 8.

Multi-layered perceptron (MLP) (Hinton, 1989). A neural network with three hidden layers, each consisting of 10 units. The activation function used is a rectified linear unit function ($f(x) = max(0, x)$) and 'adam', a stochastic gradient-based optimizer, is used for the weights (Kingma and Ba, 2014).

5.1.1 Integration into AGKA

Instead of replacing the entire AGKA architecture, only the classification part is replaced by a machine learning algorithm. Specifically the dialogue component is replaced entirely by the machine learning component, which constitutes one of the previous machine learning algorithms (Section 5.1) without rule extraction. Whenever a transaction is presented to the system, a fitted model of the algorithm in question is consulted to predict the class. If a model does not exist yet, one is fit on the existing database. If that fails, the default label 'legitimate' is proposed. Whenever an error is made, the model is refit on the entire database at that point.

This modification allows the described machine learning algorithms to operate on a stream, despite being intended to operate on data sets.

5.2 Measures of performance

To assess the performance of the algorithms used, two measures of performance are recorded that allow a comparison between all algorithms. The first, *accuracy* (Section

		Class Legitimate	Illegitimate
Label	Legitimate	True negative (tn)	False negative (fn)
	Illegitimate	False positive (fp)	True positive (tp)

TABLE 5.1: Confusion matrix to show the performance of an algorithm.

5.2.1), simply shows how often an algorithm is correct or not. As such, the first measure assesses classification performance. Such a measure is insufficient, because in this experiment costs are also included. The second measure, *costs incurred* (Section 5.2.2), is a way to assess the cost-effectiveness of a system. That is, how much money is lost due to illegitimate behaviour and verification?

Lastly, since the streams adhere to pre-defined rules, the measure *inferred rules* described in Section 5.2.3, is devised to assess the efficacy in finding those rules back. However, only AGKA explicitly extracts rules, so this measure can only be given for AGKA and not for the algorithms used for comparison.

5.2.1 Accuracy

To assess the classification performance of an algorithm, the *accuracy* is used. The accuracy is given both as a percentage and a confusion matrix. The confusion matrix displays counts in every cell as in Table 5.1.

The accuracy in percentages can be calculated from the confusion matrix using Equation 5.1.

$$\text{accuracy} = \frac{tn + tp}{tn + fn + fp + tp} \cdot 100\% \tag{5.1}$$

5.2.2 Costs incurred

To measure how cost-effective an algorithm is, the *costs incurred* are also recorded. Costs are incurred whenever either one of two situations occurs (Section 3.3.5.1):

1. A transaction is flagged for verification.
2. A transaction was illegitimate, but not flagged as such.

Respectively, these situations incur a cost of κ and λ. These costs are summed and displayed after every run. A lower number is better, because it means the system proved to be less costly in operation.

5.2.3 Inferred rules

In order to assess whether the system is capable of finding back the rules present in the stream, the rules found by the system are also listed. The listed rules are those contained in the database component 3.3.1 at the end of a run. The top ten rules of both categories are listed.

Note that assessing whether the inferred rules resemble the patterns in the data is subjective. For this reason, this measure is merely given as reference. Whether the performance actually performs 'well' in this respect is left for the discussion.

5.3 Simulated data streams

In order to assess whether the system is capable of discerning explicit patterns that the data conforms to, data streams are simulated with a resemblance to actual transactions.

The simulated data streams adhere to pre-defined rules, governing what and how instances are generated. Specifically the data generation rules define the value range for every field of a transaction. A formal description, as well as the procedure of application, of these rules is given in Section 3.2.2 and Section 3.2.3. Using such a method for data generation ensures that certain 'typical' fraudulent patterns can be specified and tried.

The data generation rules are contained in three categories: general, legitimate and illegitimate. The first, general, applies to all instances. Since these are applied first however, certain affected fields can be overridden by more specific rules of the latter categories. The several simulated test streams for assessing performance of the system have different rules in these three categories. The rule sets used for every test stream are specified in Section 5.3.1.

5.3.1 Test streams

Five test streams are devised to assess whether the system is capable of finding the rule(s) governing whether or not a transaction is legitimate or not.

Since the data generation rules in the general category are common across all test streams, these general rules are first summarised in Section 5.3.2. Subsequent sections describe one test stream each. These test streams follow a similar set-up as used for Section 4 and therefore present successively harder classification problems.

5.3.2 Shared settings

All test streams share a number of settings that affect how the stream is generated. This section lists all shared settings. Where specific test streams deviate from this general setup, that is mentioned explicitly in the description of that test stream.

First of all, the stream parameters are fixed to the following default values:

i_max 100000 (one hundred thousand)

p_{il} 0.001 (one in a thousand)

tps 100

n_prof 25

Beside the stream parameters, the data generation rules in the general category are also shared among test streams. The shared data generation rules are summarised in Table 5.2.

Rule	Distribution	Probability
`Sender` $\Rightarrow$ `none`	profile	$p = 0$
`Receiver` $\Rightarrow$ `none`	profile	$p = 0$
`Pre-balance` $\Rightarrow$ $(\mu = 100000, \sigma = 20000)$	normal	$p = 1$
`Amount` $\Rightarrow$ $(a = 10000, b = 100000)$	uniform	$p = 1$
`Known` $\Rightarrow$ `true`	Boolean	$p = 0.9$
`Foreign` $\Rightarrow$ `true`	Boolean	$p = 0.1$

TABLE 5.2: Data generation rules from the general category that are shared among all test streams.

The profile rules require clarification. Essentially these rules specify that random profiles are picked for both the sender and recipient. Since the probability is set to $p = 0$, alternative values are sampled for these rules. That means a profile is sampled that does not appear in the consequence, which is `none`. Since all profiles are different from `none`, any (random) profile may be picked.

5.3.3 Binary decision

The first test stream presents the binary decision problem as described in Section 4.1 to the system. This means that the association rule `Foreign = true` $\Rightarrow$ `illegitimate`, as well as its contraposition `Foreign = false` $\Rightarrow$ `legitimate`, hold in the stream.

The data generation rules defined for this stream are given in Table 5.3.

Category	Rule	Distribution	Probability
Legitimate	`Foreign` ⇒ `false`	Boolean	1
Illegitimate	`Foreign` ⇒ `true`	Boolean	1

TABLE 5.3: Data generation rules per category for the binary decision stream.

5.3.4 Combination of binary attributes

Illegitimate transactions in this stream adhere to a specific combination of variable - value pairs, namely `Foreign = true AND previous Foreign = false AND Known = false`. Legitimate transactions can have any Boolean value for these fields, but never such that the combination holds. The presence of this combination makes this test stream more difficult than the previous one, because no field by itself is sufficient for solving the classification problem.

Category	Rule	Distribution	Probability
Legitimate	previous Foreign ⇒ true	Boolean	0.2
Illegitimate	Foreign ⇒ true	Boolean	1
	previous Foreign ⇒ false	Boolean	1
	Known ⇒ false	Boolean	1

TABLE 5.4: Data generation rules defined for the combination of binary attributes stream.

5.3.5 Continuous attribute

In previous streams illegitimate transactions may be discerned based on Boolean attributes. In this stream, the class to which a transaction belongs is decided by a cut-off in a continuous variable. The balance on the sender's account after fulfilling the transaction (`Post-balance`) is higher than -10000 for legitimate, but lower for illegitimate transactions.

Category	Rule	Distribution	Probability
Legitimate	`Post-balance` ⇒ $(a = -5000, b = 2000000)$	uniform	1
Illegitimate	`Post-balance` ⇒ $(a = -1000000, b = -15000)$	uniform	1

TABLE 5.5: Data generation rules per category for the continuous attribute stream. Illegitimate transactions can be singled out based on a cut-off (-10000) in a continuous attribute (`Post-balance`).

5.3.6 Continuous attribute with overlap

In this test stream legitimate and illegitimate transactions can no longer be uniquely determined. That is, the classes can only be discerned by one field, whose value is

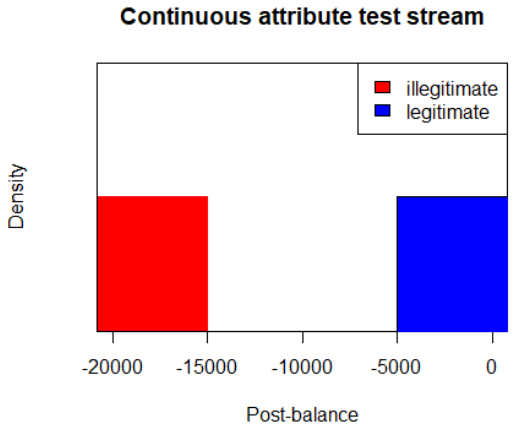


FIGURE 5.1: Distributions of values for the post-balance field in the continuous attribute stream, based on class. Notice that a 'gap' of values exists between both distributions, allowing the distributions to be perfectly separated.

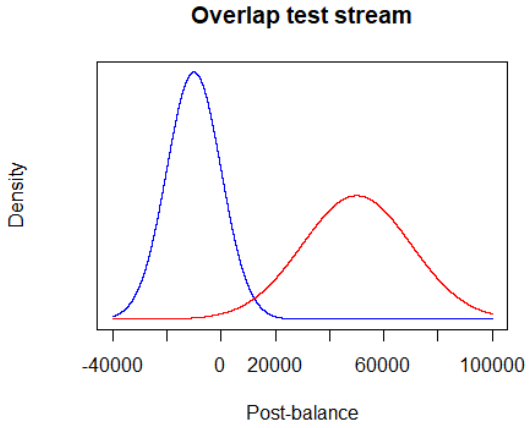


FIGURE 5.2: Distributions of values for the post-balance field in the continuous attribute with overlap stream, based on class.

normally distributed in such a way that the distributions of both classes exhibit an overlap.

The field `Post-Balance` is normally distributed with $(\mu = 50000, \sigma = 20000)$ for legitimate and $(\mu = -10000, \sigma = 10000)$ for illegitimate transactions.

Category	**Rule**	**Distribution**	**Probability**
Legitimate	`Post-balance` $\Rightarrow (\mu = 50000, \sigma = 20000)$	normal	1
Illegitimate	`Post-balance` $\Rightarrow (\mu = -10000, \sigma = 10000)$	normal	1

TABLE 5.6: Data generation rules per category for the continuous attribute with overlap stream.

5.3.7 Use of utility

To show how utilities affect the system, this stream is devised such that illegitimate transactions may easily be discerned, but it is not worthwhile to actually investigate illegitimate cases. Put differently, it costs more to retrieve the money than it costs to consider it lost.

This stream is the same as that describe in Section 5.3.3, with the addition that illegitimate rules are not worth investigating. This addition is realised by adding the data generation rule `Amount` $\Rightarrow 10$ in the illegitimate category. In summary, the data generation rules for this stream are:

Category	Rule	Distribution	Probability
Legitimate	`Foreign` $\Rightarrow$ `false`	Boolean	1
Illegitimate	`Foreign` $\Rightarrow$ `true`	Boolean	1
	`Amount` $\Rightarrow 10$	categorical	1

TABLE 5.7: Data generation rules per category for the utility stream.

5.3.8 Repetitive streams

To ensure all algorithms receive the same stream of data to classify on, first a stream of transactions is generated. After generating **max_i** transactions, one instance of the resulting database of transactions is saved. This database is then used for all algorithms, where the transactions are offered in the same sequential order. By offering each data stream in this format, the results obtained from each algorithm may be compared.

5.4 Benefits data

Another set of data used to assess AGKA's performance is the fictional benefits data as utilised in (Wardeh et al., 2009). This data set may be downloaded from `http://www.csc.liv.ac.uk/~maya/PADUA/testcases.zip`. Since this data also adheres to pre-defined rules, it can be used to evaluate the AGKA architecture on its capability of inferring those rules. The set consists of 35851 observations (proponent and opponent combined) with 11 attributes each. Since proponent and opponent contain duplicate observations (cases known to both parties), these duplicate entries are filtered, leaving 3371 observations.

The 11 attributes contained in every observation, as well as their range of values and their respective meanings can be found in Table 5.8 (For a more detailed description of

Attribute	Value	Meaning
Gender	1	male
	2	female
Age	3	age < 60
	4	≤ 60 age < 65
	5	≤ 65 age < 75
	6	≤ 75 age < 80
	7	age ≥ 80
Residency	8	UK
	9	armed forces
	10	merchant navy
	11	diplomatic services
Income	12	< 15
	13	< 20
	14	< 25
	15	≥ 25
Capital	16	< 2000
	17	< 3000
	18	< 4000
	19	≥ 4000
Contribution Year1	20	paid
	21	not paid
Contribution Year2	22	paid
	23	not paid
Contribution Year3	24	paid
	25	not paid
Contribution Year4	26	paid
	27	not paid
Contribution Year5	28	paid
	29	not paid
CLASS	30	not entitled
	31	entitled

TABLE 5.8: The 11 attributes found in every observation of the housing benefits data set. The range of (numerical) values every attribute may take are displayed, including their respective meaning.

the attributes and the pre-defined rules, consult (Wardeh et al., 2009) or (Bench-Capon, 1993)).

Two modifications to this data set are required before presenting it to AGKA. First of all an 'amount' attribute is added to each observation, which is set to the constant 10000. The attribute CLASS is regarded as 'label', with its values 'not entitled' and 'entitled' turned into respectively 'illegitimate' and 'legitimate'.

A second modification is that the data set is offered as a stream. To do so, all observations are presented in the same sequential order until all have been presented to the system.

Since the attributes are numerical values already, all algorithms may readily be applied on the set. The accuracy results obtained (Section 5.2.1) are used to compare AGKA's performance to the other algorithms. Note that the incurred costs measure is not used in the comparison. The original set is not designed to incorporate a cost measure, which makes a comparison based on the costs measure inappropriate.

Chapter 6

Results

The system is evaluated on five simulated data streams (Section 6.1), which are defined by the rules given in Section 5.3.1. Each of the subsections (Sections 6.1.1 through 6.1.5) display detailed results from each test stream. The last subsection (Section 6.1.6) lists all knowledge rules inferred by AGKA on every stream.

The system is also evaluated on the benefits data set used for PADUA, as described in Section 5.4. Results obtained from this set are given in Section 6.2.

6.1 Simulated data streams

Table 6.1 summarises the accuracy of every classifier on each of the five test streams. The highest accuracy obtained within a stream is emphasised.

For every stream the two shared performance measures described in Section 5.2, accuracy and incurred costs are provided. Section 6.1.6 reports the rules inferred by AGKA per stream.

Classifier	Binary	Combination	Continuous	Overlap	Utility
AGKA	99.99 %	99.82 %	99.99 %	99.74 %	99.89 %
DT	**99.99 %**	**99.99 %**	**99.99 %**	99.83 %	**99.99 %**
Dummy	99.90 %	99.90 %	99.91 %	**99.89 %**	99.89 %
KNN	99.90 %	99.90 %	99.99 %	99.88 %	99.98 %
MLP	99.90 %	99.90 %	99.98 %	99.88 %	99.99 %
RF	99.99 %	99.98 %	99.99 %	99.87 %	**99.99 %**
SVM	99.90 %	99.90 %	99.99 %	99.88 %	99.99 %

TABLE 6.1: Accuracy of all classifiers on all test streams, rounded down to two decimals. The highest accuracy on every stream are displayed in bold.

6.1.1 Binary decision

The binary decision data stream allows illegitimate transactions to be discerned from legitimate ones based on the value of one binary attribute. Table 6.2 lists the confusion matrices of all tested classifiers on this data stream. Table 6.3 lists the costs incurred by each classifier.

	classification	
truth	legitimate	illegitimate
legitimate	99899	4
illegitimate	1	96

(A) AGKA

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	1	96

(B) DT

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	97	0

(C) Dummy

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	97	0

(D) KNN

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	97	0

(E) MLP

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	3	94

(F) RF

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	97	0

(G) SVM

TABLE 6.2: Confusion matrices for all classifiers on the binary decision data stream.

		Incurred costs Verification	Lost	Total
Classifier	AGKA	500000	89730	589730
	DT	480000	89730	**569730**
	Dummy	0	4821514	4821514
	KNN	0	4821514	4821514
	MLP	0	4821514	4821514
	RF	470000	183689	653689
	SVM	0	4821514	4821514

TABLE 6.3: Incurred costs for the binary decision data stream.

6.1.2 Combination of binary attributes

The combination of binary attributes stream allows illegitimate transactions to be recognised by a combination of values of three attributes. Table 6.4 lists the confusion matrices

of all tested classifiers on this data stream. Table 6.5 lists the costs incurred by each classifier.

	classification	
truth	legitimate	illegitimate
legitimate	99733	171
illegitimate	5	91

(A) AGKA

	classification	
truth	legitimate	illegitimate
legitimate	99904	0
illegitimate	2	94

(B) DT

	classification	
truth	legitimate	illegitimate
legitimate	99904	0
illegitimate	96	0

(C) Dummy

	classification	
truth	legitimate	illegitimate
legitimate	99904	0
illegitimate	96	0

(D) KNN

	classification	
truth	legitimate	illegitimate
legitimate	99903	1
illegitimate	96	0

(E) MLP

	classification	
truth	legitimate	illegitimate
legitimate	99903	0
illegitimate	10	86

(F) RF

	classification	
truth	legitimate	illegitimate
legitimate	99903	1
illegitimate	96	0

(G) SVM

TABLE 6.4: Confusion matrices for all classifiers on the combination of binary attributes data stream.

		Incurred costs Verification	Lost	Total
Classifier	AGKA	1310000	238298	1548298
	DT	470000	158791	**628791**
	Dummy	0	5209862	5209862
	KNN	0	5209862	5209862
	MLP	5000	5209862	5214862
	RF	440000	375500	815000
	SVM	5000	5209862	5214862

TABLE 6.5: Incurred costs for the combination of binary attributes data stream.

6.1.3 Continuous attribute

The continuous attribute stream allows illegitimate transactions to be discerned based on one attribute that can take on a range of values. Table 6.6 lists the confusion matrices of all tested classifiers on this data stream. Table 6.7 lists the costs incurred by each classifier.

	classification	
truth	legitimate	illegitimate
legitimate	99906	4
illegitimate	3	87

(A) AGKA

	classification	
truth	legitimate	illegitimate
legitimate	99910	0
illegitimate	3	87

(B) DT

	classification	
truth	legitimate	illegitimate
legitimate	99910	0
illegitimate	90	0

(C) Dummy

	classification	
truth	legitimate	illegitimate
legitimate	99910	0
illegitimate	7	83

(D) KNN

	classification	
truth	legitimate	illegitimate
legitimate	99901	8
illegitimate	3	87

(E) MLP

	classification	
truth	legitimate	illegitimate
legitimate	99910	0
illegitimate	4	86

(F) RF

	classification	
truth	legitimate	illegitimate
legitimate	99910	0
illegitimate	4	86

(G) SVM

TABLE 6.6: Confusion matrices for all classifiers on the continuous attribute data stream.

		Incurred costs Verification	Lost	Total
Classifier	AGKA	455000	183007	638007
	DT	435000	183007	**618007**
	Dummy	0	4979850	4979850
	KNN	415000	431138	846138
	MLP	475000	231116	706116
	RF	430000	282365	712365
	SVM	430000	193401	623401

TABLE 6.7: Incurred costs for the continuous attribute data stream.

6.1.4 Continuous attribute with overlap

Just as the continuous attribute stream, the continuous attribute with overlap stream allows a distinction of illegitimate transactions based on an attribute that could take on a range of values. For this stream however, no perfect distinction between legitimate and illegitimate can be made based solely on this attribute, since values from both distributions show overlap. Table 6.8 lists the confusion matrices of all tested classifiers on this data stream. Table 6.9 lists the costs incurred by each classifier.

truth	classification: legitimate	classification: illegitimate
legitimate	99730	167
illegitimate	87	16

(A) AGKA

truth	classification: legitimate	classification: illegitimate
legitimate	99817	80
illegitimate	81	22

(B) DT

truth	classification: legitimate	classification: illegitimate
legitimate	99897	0
illegitimate	103	0

(C) Dummy

truth	classification: legitimate	classification: illegitimate
legitimate	99870	27
illegitimate	87	16

(D) KNN

truth	classification: legitimate	classification: illegitimate
legitimate	99887	10
illegitimate	102	1

(E) MLP

truth	classification: legitimate	classification: illegitimate
legitimate	99856	41
illegitimate	82	21

(F) RF

truth	classification: legitimate	classification: illegitimate
legitimate	99873	24
illegitimate	93	10

(G) SVM

TABLE 6.8: Confusion matrices for all classifiers on the continuous attribute with overlap data stream

		Incurred costs Verification	Lost	Total
Classifier	AGKA	915000	5124790	6039790
	DT	510000	4892003	5402003
	Dummy	0	6102071	6102071
	KNN	215000	5063592	5278592
	MLP	55000	6033588	6088588
	RF	310000	4814295	**5124295**
	SVM	170000	5825780	5995780

TABLE 6.9: Incurred costs for the continuous attribute with overlap data stream.

6.1.5 Use of utility

The use of utility stream allows a distinction between legitimate and illegitimate transactions based on a binary attribute. However, every illegitimate transaction is not worth an investigation. Table 6.10 lists the confusion matrices of all tested classifiers on this data stream. Table 6.11 lists the costs incurred by each classifier.

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	109	0

(A) AGKA

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	1	108

(B) DT

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	109	0

(C) Dummy

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	20	89

(D) KNN

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	7	102

(E) MLP

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	1	108

(F) RF

truth	classification: legitimate	classification: illegitimate
legitimate	99891	0
illegitimate	3	106

(G) SVM

TABLE 6.10: Confusion matrices for all classifiers on the utility data stream.

		Incurred costs: Verification	Lost	Total
Classifier	AGKA	0	1090	**1090**
	DT	540000	10	540010
	Dummy	0	1090	**1090**
	KNN	445000	200	445200
	MLP	510000	70	510070
	RF	540000	10	540010
	SVM	530000	30	530030

TABLE 6.11: Incurred costs for the utility data stream.

6.1.6 Extracted rules

Tables 6.12 through 6.16 list all knowledge rules inferred by AGKA on each data stream.

Category	Rule (condition)	P	Costs (apply, ignore)
Illegitimate	foreign > 0.5	1	(5000.0, 89730.0)
	receiver name == Mr. Daniel Caulfield and sender name == Mr. Bart Humbleton	0.25	(5000.0, 22432.5)
	time == 00:06:12 and receiver name == Mr. Daniel Caulfield	0.2	(5000.0, 17946.0)
Legitimate	time == 00:06:12 and post-balance $>$ 61834.0	1	(0.0, 5000.0)
	time == 00:06:12 and pre-balance $\leq$ 113164.0	1	(0.0, 5000.0)
	time == 00:06:12 and amount $\leq$ 79082.0	1	(0.0, 5000.0)
	sender name == Mr. Bart Humbleton and foreign ≤ 0.5	1	(0.0, 5000.0)
	receiver name == Mr. Daniel Caulfield and sender name == Mr. Bart Humbleton and foreign ≤ 0.5	1	(0.0, 5000.0)
	time == 00:06:12 and receiver name == Sir Michael Verheij	1	(0.0, 5000.0)
	time == 00:06:12 and foreign ≤ 0.5	1	(0.0, 5000.0)
	foreign ≤ 0.5	1	(0.0, 5000.0)
	receiver name == Mr. Daniel Caulfield and foreign ≤ 0.5	1	(0.0, 5000.0)
	time == 00:06:12 and pre-balance $\leq$ 118580.0	1	(0.0, 5000.0)

Table 6.12: Knowledge rules inferred for the binary decision data stream.

Category	Rule (condition)	P	Costs (apply, ignore)
Illegitimate	foreign $>$ 0.5 and known $\leq$ 0.5 and prev_foreign $\leq$ 0.5	1	(5000.0, 47659.6)
	foreign $>$ 0.5 and time == 00:02:55 and receiver name == Mrs. Betty Cashalot	0.33	(5000.0, 30787.0)
	foreign $>$ 0.5 and time == 00:01:47 and receiver name == Sir Daniel Ferguson	0.5	(5000.0, 35076.5)
	foreign $>$ 0.5 and time == 00:06:38 and post-balance ≤ -3499.0	0.33	(5000.0, 30774.0)
	foreign $>$ 0.5 and time == 00:08:26 and sender account == PO51MINE8234629224	0.5	(5000.0, 43029.0)
	time == 00:12:21 and receiver name == Mrs. Betty Cashalot	0.5	(5000.0, 36306.0)
	foreign $>$ 0.5 and time == 00:15:53 and post-balance $>$ 81377.5	0.5	(5000.0, 15890.0)
	foreign $>$ 0.5 and receiver name == Sir Jack Caulfield and post-balance ≤ -33324.5	0.33	(5000.0, 29668.0)
	foreign $>$ 0.5 and time == 00:08:37 and sender account == AA13BANK5957501967	0.33	(5000.0, 11173.67)
	foreign $>$ 0.5 and known $\leq$ 0.5 and time == 00:07:12	0.33	(5000.0, 26924.67)
Legitimate	time == 00:04:19 and pre-balance $\leq$ 111378.5	1	(0.0, 5000.0)
	time == 00:07:09 and foreign $\leq$ 0.5	1	(0.0, 5000.0)
	time == 00:04:19 and receiver name == Sir Ferdinand Bond	1	(0.0, 5000.0)
	time == 00:12:51 and sender account == XX68LOVE6635409499	1	(0.0, 5000.0)
	time == 00:12:51 and known $>$ 0.5	1	(0.0, 5000.0)
	time == 00:12:51 and prev_foreign $>$ 0.5	1	(0.0, 5000.0)
	time == 00:12:51 and pre-balance $>$ 83525.0	1	(0.0, 5000.0)
	time == 00:12:51 and post-balance $>$ 15711.0	1	(0.0, 5000.0)
	time == 00:04:19 and known $>$ 0.5	1	(0.0, 5000.0)
	foreign $\leq$ 0.5	1	(0.0, 5000.0)

Table 6.13: Knowledge rules inferred for the combination of binary attributes data stream.

Category	Rule (condition)	P	Costs (apply, ignore)
Illegitimate	post-balance ≤ -351815.5	1	(5000.0, 42093.0)
	post-balance ≤ -36590.5	1	(5000.0, 49430.5)
	pre-balance ≤ 6342.0	1	(5000.0, 49430.5)
	post-balance ≤ -36832.0	1	(5000.0, 37586.0)
	pre-balance ≤ -398.5	1	(5000.0, 47106.0)
	post-balance ≤ -14385.0	1	(5000.0, 58369.75)
	post-balance ≤ -14411.0	1	(5000.0, 59153.33)
	post-balance ≤ -10293.0	1	(5000.0, 61211.22)
	time == 00:01:41 and sender name == Sir Warren Janssen	0.5	(5000.0, 42073.0)
Legitimate	time == 00:16:21 and sender name == Ms. Lucy Bond and post-balance > 352836.5	1	(0.0, 5000.0)
	time == 00:01:41 and receiver account == ZZ34FAST5483087747	1	(0.0, 5000.0)
	time == 00:16:21 and receiver account == NL47PLAN7189721687 and known ≤ 0.5	1	(0.0, 5000.0)
	time == 00:16:21 and post-balance $>$ 98262.5	1	(0.0, 5000.0)
	time == 00:16:21 and pre-balance $>$ 133447.5	1	(0.0, 5000.0)
	time == 00:01:41 and pre-balance $>$ 580279.5	1	(0.0, 5000.0)
	time == 00:01:41 and post-balance $>$ 490184.0	1	(0.0, 5000.0)
	time == 00:01:41 and pre-balance $>$ 62338.0	1	(0.0, 5000.0)
	post-balance > -14385.0	1	(0.0, 5000.0)
	post-balance > -14411.0	1	(0.0, 5000.0)

Table 6.14: Knowledge rules inferred for the continuous attribute data stream.

Category	Rule (condition)	P	Costs (apply, ignore)
Illegitimate	time == 00:13:13 and post-balance ≤ 6701.5	1	(5000.0, 72553.0)
	time == 00:06:14 and post-balance ≤ 1984.5	1	(5000.0, 77394.0)
	time == 00:04:38 and post-balance ≤ 2046.5	1	(5000.0, 48640.0)
	time == 00:03:59 and post-balance ≤ 7788.5	1	(5000.0, 64902.0)
	time == 00:02:14 and post-balance ≤ 2170.0	0.5	(5000.0, 46881.5)
	time == 00:06:00 and post-balance ≤ 1721.0	1	(5000.0, 59589.0)
	time == 00:01:45 and post-balance ≤ −7676.0	1	(5000.0, 68875.0)
	time == 00:11:19 and post-balance ≤ 2834.0	1	(5000.0, 60613.0)
	time == 00:08:50 and post-balance ≤ 5162.5	1	(5000.0, 63473.0)
	time == 00:03:12 and foreign > 0.5	0.5	(5000.0, 27370.0)
Legitimate	time == 00:05:45 and pre-balance > 61518.5	1	(0.0, 5000.0)
	time == 00:05:45 and post-balance > −11500.5	1	(0.0, 5000.0)
	time == 00:15:13 and pre-balance > 52862.5	1	(0.0, 5000.0)
	time == 00:15:13 and post-balance > 7852.0	1	(0.0, 5000.0)
	time == 00:08:07 and pre-balance > 88288.0	1	(0.0, 5000.0)
	post-balance ≤ −4659.5 and time == 00:05:38	1	(0.0, 5000.0)
	time == 00:08:07 and pre-balance > 74546.0	1	(0.0, 5000.0)
	time == 00:08:07 and amount ≤ 57188.5	1	(0.0, 5000.0)
	post-balance ≤ −14382.0 and time == 00:06:23	1	(0.0, 5000.0)
	post-balance ≤ −14382.0 and time == 00:08:42	1	(0.0, 5000.0)

Table 6.15: Knowledge rules inferred for the continuous attribute with overlap data stream.

Category	Rule (condition)	P	Costs (apply, ignore)
Illegitimate	None	-	-
Legitimate	foreign ≤ 0.5	1	(0.0, 5000.0)
	amount > 5036.5	1	(0.0, 5000.0)
	amount > 5026.0	1	(0.0, 5000.0)
	amount > 5007.0	1	(0.0, 5000.0)
	amount > 5006.0	1	(0.0, 5000.0)
	amount > 5005.0	1	(0.0, 5000.0)

TABLE 6.16: Knowledge rules inferred for the utility data stream.

Classifier	Accuracy
AGKA	97.09 %
DT	**98.22 %**
Dummy	26.22 %
KNN	90.09 %
MLP	65.02 %
RF	96.79 %

TABLE 6.17: Accuracy of every classifier on the benefits data set. The highest accuracy is emphasised. The SVM classifier is excluded since its excessive run time does not allow it to finish classifying the set timely.

6.2 Benefits data

The classifiers used for the streams are also applied to the benefits data set. One exception is that the SVM classifier is excluded, due to excessive run time on the set (not being able to process 1000 transactions within one day of processor time).

Table 6.17 summarises the accuracy in percentages of all classifiers on the benefits data set. The respective confusion matrices for every classifier are listed in Table 6.18. The knowledge rules inferred by AGKA are summarised in Table 6.19.

	classification	
truth	legitimate	illegitimate
legitimate	871	13
illegitimate	85	2402

(A) AGKA

	classification	
truth	legitimate	illegitimate
legitimate	859	25
illegitimate	35	2452

(B) DT

	classification	
truth	legitimate	illegitimate
legitimate	884	0
illegitimate	2487	0

(C) Dummy

	classification	
truth	legitimate	illegitimate
legitimate	835	49
illegitimate	285	2202

(D) KNN

	classification	
truth	legitimate	illegitimate
legitimate	238	646
illegitimate	533	1954

(E) MLP

	classification	
truth	legitimate	illegitimate
legitimate	837	47
illegitimate	61	2426

(F) RF

TABLE 6.18: Confusion matrices for all classifiers, except SVM, on the benefits data set.

Category	Rule (condition)	P	Costs (apply, ignore)
	Capital > 17.5	1	(5000.0, 10000.0)
	Residency ≤ 8.5 and Capital > 17.5	1	(5000.0, 10000.0)
	Capital ≤ 17.5 and Residency > 8.5 and Residency ≤ 9.5 and Contr y4 > 26.5 and Age ≤ 3.5	1	(5000.0, 10000.0)
	Residency > 8.5 and Age ≤ 5.5 and Age ≤ 3.5	1	(5000.0, 10000.0)
Illegitimate	Residency > 8.5 and Age > 3.5 and Age ≤ 5.5 and Contr y4 > 26.5 and Income > 13.5	1	(5000.0, 10000.0)
(Not entitled)	Capital ≤ 17.5 and Residency > 8.5 and Age > 3.5 and Residency ≤ 9.5 and Income > 13.5	1	(5000.0, 10000.0)
	Age ≤ 3.5	1	(5000.0, 10000.0)
	Gender ≤ 1.5 and Capital ≤ 16.5 and Age ≤ 4.0	1	(5000.0, 10000.0)
	Capital ≤ 17.5 and Age > 3.5 and Contr y2 > 22.5 and Income ≤ 13.5 and Residency > 9.5	1	(5000.0, 10000.0)
	Income > 13.5	1	(5000.0, 10000.0)
	Residency ≤ 9.5 and Capital ≤ 17.5 and Income ≤ 13.5 and Age > 3.5 and Contr y2 > 22.5 and Age > 4.5 and Contr y5 ≤ 28.5 and Contr y1 ≤ 20.5 and Contr y4 > 26.5 and Contr y3 > 24.5	1	(0.0, 5000.0)
	Residency ≤ 9.5 and Capital ≤ 17.5 and Income ≤ 13.5 and Age > 3.5 and Contr y2 ≤ 22.5 and Contr y5 ≤ 28.5 and Contr y3 > 24.5 and Contr y1 > 20.5 and Contr y4 > 26.5 and Age ≤ 4.5 and Gender > 1.5	1	(0.0, 5000.0)
	Capital ≤ 17.5 and Residency ≤ 9.5 and Income ≤ 13.5 and Age > 3.5 and Contr y5 ≤ 28.5 and Contr y2 > 22.5 and Contr y1 > 20.5 and Age > 4.5 and Contr y3 > 24.5	1	(0.0, 5000.0)

Continued on next page

TABLE 6.19: Knowledge rules inferred from the benefits data set.

Table 6.19 – *Continued from previous page*

Category	**Rule (condition)**	**P**	**Costs (apply, ignore)**
	Residency $\leq$ 9.5 and Capital $\leq$ 17.5 and Income $\leq$ 13.5 and Age $>$ 3.5 and Contr y5 $>$ 28.5 and Age $>$ 4.5 and Contr y4 $>$ 26.5 and Contr y1 $\leq$ 20.5 and Contr y2 $\leq$ 22.5 and Contr y3 $>$ 24.5	1	(0.0, 5000.0)
Legitimate	Residency $\leq$ 9.5 and Capital $\leq$ 17.5 and Income $\leq$ 13.5 and Age $>$ 3.5 and Contr y5 $>$ 28.5 and Age $>$ 4.5 and Contr y4 $>$ 26.5 and Contr y1 $\leq$ 20.5 and Contr y2 $>$ 22.5	1	(0.0, 5000.0)
(Entitled)	Income $\leq$ 13.5 and Capital $\leq$ 17.5 and Residency $\leq$ 9.5 and Age $>$ 3.5 and Contr y5 $\leq$ 28.5 and Contr y2 $>$ 22.5 and Contr y1 $>$ 20.5 and Age $>$ 4.5 and Contr y4 $>$ 26.5	1	(0.0, 5000.0)
	Income $\leq$ 13.5 and Capital $\leq$ 17.5 and Residency $\leq$ 9.5 and Age $>$ 3.5 and Contr y5 $\leq$ 28.5 and Contr y2 $\leq$ 22.5 and Contr y3 $>$ 24.5 and Contr y1 $>$ 20.5 and Contr y4 $>$ 26.5 and Age $>$ 4.5	1	(0.0, 5000.0)
	Income $\leq$ 13.5 and Capital $\leq$ 17.5 and Residency $\leq$ 9.5 and Age $>$ 3.5 and Contr y5 $>$ 28.5 and Age $>$ 4.5 and Contr y4 $\leq$ 26.5 and Contr y3 $\leq$ 24.5 and Contr y1 $>$ 20.5 and Contr y2 $>$ 22.5	1	(0.0, 5000.0)
	Capital $\leq$ 17.5 and Income $\leq$ 13.5 and Residency $\leq$ 9.5 and Age $>$ 3.5 and Contr y5 $>$ 28.5 and Age $>$ 4.5 and Contr y4 $\leq$ 26.5 and Contr y1 $\leq$ 20.5 and Contr y3 $>$ 24.5 and Contr y2 $>$ 22.5	1	(0.0, 5000.0)
	Capital $\leq$ 17.5 and Income $\leq$ 13.5 and Residency $\leq$ 9.5 and Age $>$ 3.5 and Contr y5 $>$ 28.5 and Age $>$ 4.5 and Contr y4 $>$ 26.5 and Contr y2 $\leq$ 22.5 and Contr y1 $>$ 20.5	1	(0.0, 5000.0)

Table 6.19: Knowledge rules inferred from the benefits data set.

Chapter 7

Discussion

In this chapter first an analysis of the obtained results is provided (Section 7.1). Section 7.2 discusses the implications of the analysis. Section 7.3 mentions some aspects in which improvements can be made. Lastly, section 7.4 describes how the findings are relevant in a research perspective, as well as for practical applications.

7.1 Analysis

In this section the performances of the classifiers on the test streams are analysed, according to the two comparison measurements accuracy and incurred costs. The focus of this analysis lies on the performance of AGKA compared to the other algorithms.

The first measurement, the classification accuracy of an algorithm, is discussed in Section 7.1.1. The second measurement, cost-efficiency, is discussed in Section 7.1.2. The discussion of extracted rules which only applies to AGKA can be found in Section 7.1.3.

7.1.1 Classification accuracy of AGKA

As can be seen in Table 6.1 all classifiers achieve an accuracy of at least 99% on all test streams. Also apparent from Table 6.1 is that the standalone decision tree (DT) performs best overall. Only in the overlap stream does Dummy (classify everything as legitimate) outperform the other classifiers. In terms of accuracy, it would therefore be best to choose DT for finding illegitimate transactions.

First consider the percentages on the utility stream. AGKA performs just as bad as Dummy on this stream in terms of accuracy. This result is expected, since the illegitimate transactions in this stream are not worth investigating, so AGKA outputs the (wrong)

label 'legitimate'. This behaviour has a desired effect as discussed in the next section (Section 7.1.2).

Looking at the remaining streams, the accuracy of the decision tree approaches (AGKA, DT and RF) follow a similar trend. The classifiers perform relatively well on the binary, combination and continuous streams, but relatively bad on the overlap stream. This similarity in accuracy may be expected, since the approaches are founded on the same rationale. In comparison with DT and RF, AGKA appears to perform worse overall, with exceptional low accuracy on the combination and overlap streams.

Why AGKA performs worse can be discerned from the confusion matrices. For instance, Table 6.2 shows that AGKA (6.2a) outputs the label 'illegitimate' more often than DT (6.2b). This difference is also found in the other tables, especially for streams where AGKA's performance is much worse, such as for the overlap stream (Table 6.8). It can be said that AGKA employs a more 'cautious' approach compared to DT: AGKA prefers to investigate more transactions rather than risking letting an illegitimate transaction slip by. Even though this cautious approach hurts classification accuracy, there is a justification for it, which will be explained in Section (7.2.1).

Note that an accuracy of 100% is not achievable. Since the data is a stream in nature and the system starts without any previously encountered transactions, there is no other way than 'guessing' what the label of the first transactions should be. As soon as both classes, legitimate and illegitimate, contain at least one example, it becomes possible to look for differences between the classes. Nevertheless at this point, the system is bound to have made an error, because it had to guess the correct label. Even one such error will mean 100% accuracy cannot be attained.

These findings also hold for the results obtained on the benefits data (Table 6.17 and 6.18). Notice however from Table 6.18 that AGKA now employs a less cautious approach in comparison with DT. This may be attributed to the fact that all instances contain an amount equal to 10000. Since this value is equal to 2κ, AGKA can propose a 'legitimate' label as soon as one counter example is encountered.

7.1.2 Cost efficiency of AGKA

Since classification algorithms focus on achieving high classification accuracy and not on diminishing costs incurred as defined here, it is only useful to discuss how AGKA performs relative to the other classifiers on this measure.

AGKA is designed to minimise costs incurred during operation, so it should perform better than the other classifiers used in that respect. Looking at the incurred costs

tables (for example Table 6.3) shows that DT performs best overall, just like for the accuracy measure. The utility stream (Table 6.11) is a notable exception, where AGKA together with the Dummy classifier perform best. Since the illegitimate transactions in this stream are not worth investigating, AGKA simply ignores them, which leads to less incurred costs in total.

On the other streams, AGKA comes in either second or third in terms of incurred costs. The overlap stream (Table 6.9) is an exception where AGKA performs worse. This can be attributed to the 'cautious' approach mentioned before, as well as correlation between accuracy and incurred costs. Since AGKA outputs the 'illegitimate' label more often, the verification costs are higher. In combination with the lost amounts not being lower than for DT, this results in higher total incurred costs. Although a cautious approach proves to be more costly in these settings, it may still be justified as explained in Section 7.2.1.

It should be noted that classification accuracy and cost-efficiency are correlated. This can be deduced from the costs defined for every outcome (Section 3.3.5.1). Since the costs for wrong classifications are higher than for correct ones, it is logical that costs incurred by the system are lower if it is correct more often (has higher accuracy).

7.1.3 Extracted rules

The rules extracted by AGKA are included to assess its capability in finding back the rules governing the stream. For the binary decision and the continuous attribute streams it can be said that AGKA was capable of finding the rule(s) back. Knowledge rules stored for the binary decision stream (Table 6.12) include the rules `foreign > 0.5` $\Rightarrow$ `illegitimate` and `foreign < 0.5` $\Rightarrow$ `legitimate`, which are in accordance with the rules specified for that stream (Table 5.3).

Recall that for the continuous attribute stream a 'gap' exists between the legitimate and illegitimate values for the attribute `post-balance`. The gap constitutes values between -15000 and -5000, which implies any decision boundary between these values appropriately describes a distinction between both classes. Table 6.14 shows that AGKA indeed stored several knowledge rules which describe a decision boundary within the gap, such as the rule `post-balance > -14411` $\Rightarrow$ `legitimate`.

For the utility stream (Table 6.16) it can also be said that AGKA found the rules present, despite only the rule `Foreign` $\leq$ `0.5` $\Rightarrow$ `legitimate` being shown. Only this rule may ever be stored, because its contra-position `Foreign` $>$ `0.5` $\Rightarrow$ `illegitimate`

will be more costly to apply than to ignore. Thus the counter rule, as well as any other illegitimate rule found in this stream for that matter, is never stored.

The overlap stream does not really have a decision boundary between both classes. The deciding attribute is normally sampled for both classes and even exhibits some overlap. Defining a decision boundary between these distributions is therefore debatable. For this reason, saying whether or not the extracted rules (Table 6.15) resemble the ones present in the data is inappropriate.

For the combination of binary attributes stream it can be said that AGKA was partially able to find the rules back. As can be seen in Table 6.13, the rule `foreign` $>$ 0.5 `and known` ≤ 0.5 `and prev_foreign` $\leq 0.5 \Rightarrow$ `illegitimate` perfectly describes the combination uniquely determining illegitimate transactions. Nevertheless, for legitimate transactions ideally there would be three rules with one attribute each, namely `foreign` $\leq 0.5 \Rightarrow$ `legitimate`, `known` $> 0.5 \Rightarrow$ `legitimate` and `prev_foreign` $> 0.5 \Rightarrow$ `legitimate`. These three rules are sufficient to distinguish legitimate transactions. AGKA however only found `foreign` $\leq 0.5 \Rightarrow$ `legitimate`. The other legitimate rules contain variants of the ideal rules, as the inferred rules contain additional (irrelevant) conditions.

It is true in general that a lot of irrelevant rules are inferred for every stream. This behaviour is due to the hypothesis space for every stream: with more attributes and more examples, the number of possible splits increases exponentially, which means exponentially more conditions are possible. Whether it is a good thing to have irrelevant rules and conditions, and whether anything can be done to alleviate it is reserved for Section 7.2.3.

The same holds for the rules inferred from the benefits data (Table 6.19). The rules in the illegitimate (not entitled) category are shorter, since not meeting one condition can already be enough to reject a benefit. The rule `Capital` $> 17.5 \Rightarrow$ `illegitimate` (which translates to `Capital` $> 3000 \Rightarrow$ `illegitimate`) is one example. Possessing that amount of capital makes one not entitled to a benefit. The legitimate rules are longer, since an application has to fulfil all set conditions for a benefit to be entitled. Most rules contain some of the necessary conditions and vary in which years contribution is paid (contribution needs to be payed in at least three of the last five years). Concluding, AGKA is capable of finding back rules present in the data, with irrelevant rules and conditions among them.

Why AGKA uses a 'cautious' approach also becomes apparent from the inferred rules. Table 6.12 for example shows the system stored the knowledge rule `time == 00:06:12 and receiver name == Mr. Daniel Caulfield` $\Rightarrow$ `illegitimate`, $P = 0.2(5000, 17946)$.

Basically this knowledge rule states that the transactions to which it applies contains a very expensive illegitimate transaction or transactions, so it is better to check every transaction fulfilling the condition, despite there being more legitimate transactions ($P = 0.2$). Worst case this means 4 out of 5 transactions would be wrongly classified based on this knowledge rule, yet its expected utility shows that it is still better to apply than to ignore.

7.2 Implications

In this section, AGKA's performance is evaluated and defended (Section 7.2.1) based on the analysis in Section 7.1. Section 7.2.2 discusses whether it is appropriate to use a decision tree in the setting of fraud detection. With regard to the irrelevant information acquired by the system, Section 7.2.3 concludes with a discussion of the over-fitting issue. Lastly the effect of skewed distributions is discussed in Section 7.2.4.

7.2.1 Why use AGKA?

If a decision tree shows the best results overall, why not just use that approach? Several reasons can be formulated, which plead for using AGKA despite it having worse results in accuracy and incurred costs.

For one, AGKA incorporates reasoning and utilities. Why reasoning can be useful is apparent, since reading dialogues is much more common to humans than reading a decision tree model. Moreover, the inferred rules (Tables 6.12 through 6.16) are shown to be in accordance with the data, so it can be said that the dialogues are grounded.

The incorporation of utilities has proven to be disadvantageous with respect to accuracy and incurred costs in most streams. That does not say that using utility is unwise however. The utility stream is a notable example in that respect: AGKA did achieve the least incurred costs in this stream, thanks to using utilities. Moreover, the cautious approach may be beneficial under different circumstances. While for these streams it is beneficial to immediately disregard rules that turn out to be incorrect, in situations where such rules may become relevant again in the future, it is good to retain them. Suppose a fraudster is detected and all transactions to their account are investigated. Suppose the fraudster then switches to a different account for their illegitimate actions. If a system would forget about the first detection and immediately switch to only investigating the new account, the fraudster would be able to use the old account for illegitimate actions again.

7.2.2 Proper rationales

Using a decision tree for classification implicitly assumes that the classes present may be discerned by a number of splits based on the value of attributes. Obviously this assumption falls flat in the overlap stream, where it may be wiser to model the normal distributions and decide on the membership based on the likelihood a transactions belongs to either of the distributions. For this reason, is AGKA using a proper rationale for its classifications?

For the stream in question the answer is no, but only with additional knowledge. For a system that has to learn from data, any rationale is appropriate: it simply cannot know better than to try anything. This problem is related to over-fitting (Section 7.2.3). Only with knowledge about the structure of data, it becomes possible to choose an appropriate rationale for discerning classes. Typically it is left to the designer to find an appropriate method, which may also be a process of trial and error involving grid search, parameter sweeps and so on.

All in all, only knowledge determines what is appropriate and that shows an option for improvement: Instead of leaving the designer to find a proper rationale, the system may also try to find an optimal rationale by trying several methods.

7.2.3 Over- or under-fitting?

With any machine learning method one cannot simply bypass the problem of over-fitting. Since AGKA ultimately aims to provide an accurate classification with solid grounding, over-fitting may have an impact on both the accuracy and grounding provided. The abundance of inferred rules are an example of this. These rules can be (and have been) given as support for a classification, but they are neither accurate nor informative.

AGKA somewhat addresses the over-fitting problem by only meticulously investigating ‘difficult’ cases. The knowledge rules stored in the database provide an indication to the true class of a transaction. If there are only indications for one class, that label is immediately provided, by virtue of it being less costly to apply a rule than to ignore it. Such a case could be considered ‘easy’, because everything indicates to one label. It is possible though that there are no, or even contradicting, indications for a transaction. These cases may be considered ‘difficult’ because there is uncertainty over the true class of a transaction. Only for these cases will AGKA engage in a dialogue to find the most appropriate label and support for a transaction. Hence, AGKA responds according to the difficulty of a case in question, rather than applying a possibly over- or under-fitting model to all novel observations.

7.2.4 Skewedness

The ratio of 1 illegitimate transaction to 1000 legitimate transactions is (luckily) skewed compared to real world estimates. Nevertheless, the ratio is maintained in order to eliminate the necessity of huge data sets, merely to ensure a few illegitimate examples are encountered. With the accuracy of all classifiers being well over 99% on every stream (Table 6.1), it can be concluded that such a skewed distribution does not pose a problem to classification.

It should be noted though that even an accuracy of 99.99% still implies 1000 mistakes in 10,000,000 cases. Such a number may be undesired for real applications. Since currently it is impossible to assess the performance on larger streams (Section 7.3.1), it is interesting to know whether the accuracy holds in larger streams with even further skewed distributions, such as 1:1,000,000.

7.3 Improvements

Certain procedures and implementations in the design of AGKA can readily be improved. With regard to computational complexity, Section 7.3.1 mentions a specific issue. An improvement to the controversial behaviour of ignoring low cost cases is given in Section 7.3.2. Lastly, the notion of concept drift is discussed in Section 7.3.3 as an area to assess AGKA's performance on.

7.3.1 Optimisation

A measure that is not reported is the run time for each classifier. A design goal of AGKA is to let it operate on a stream of transactions in order to distinguish fraudulent ones. For the overlap stream, consisting of 100,000 transactions, AGKA needed to run for a full day. In real world scenarios, banks need to process many millions of transactions every day. In effect, in order to use AGKA for such scenarios, computational improvements are required.

The dialogue component is the most computationally expensive part of the system. Since a dialogue is inherently sequential in nature, the dialogue procedure cannot be parallelized. Also, during every turn a decision tree is fit on all transactions in the (subset of the) database. All candidate rules, extracted from the fitted model or applying categorical combinations, are verified (costs calculated) again according to all transactions. These operations may do the same calculations a multitude of times, because fitting a decision tree on a subset of data explores some of the possible splits it already

explored before. Verification calls on all transactions again, so it would benefit from an integration with the fitting procedure. With a large (> 100000) number of transactions to consider, these inefficiencies make holding a dialogue a lengthy, possibly infeasible operation.

7.3.2 Automatic blocking

The utility stream shows how the system decides to ignore illegitimate transactions, because it is too costly to investigate them. This behaviour is intended, despite having side effects as mentioned in Section 7.2.1.

There are several methods to change this behaviour. One option is to add a third action to knowledge rules of the illegitimate category: *block*, which means preventing a transaction being fulfilled. The new cost table is shown in Table 7.1. It doesn't cost anything to block a fraudulent transaction, because the system can instantly block it without intervention and no money is lost. However, if a legitimate transaction is blocked, this is even costlier than having it checked: A (credible) customer may furiously ask why she is unable to fulfil a transaction, which occupies a bank spokesperson with offering an explanation and possibly a system's expert to manually allow the transaction. Hence the cost of this outcome, ν, should be (far) greater than κ.

		Classification	
		correct	incorrect
Action	apply	$Cost(o) = \kappa$	$Cost(o) = \kappa$
	ignore	$Cost(o) = \lambda$	$Cost(o) = 0$
	block	$Cost(o) = 0$	$Cost(o) = \nu$

TABLE 7.1: Costs of outcomes for an illegitimate rule, including the *block* action.

7.3.3 Concept drift

The design goals for AGKA also touch the subject of *concept drift* (Widmer and Kubat, 1996; Zliobaite, 2010). Concept drift implies that the target concept depends on a hidden context which may change. For instance, the buying preferences of customers may be different during the weekend. Concept drift learners need to be able to catch such differences and adapt appropriately.

Concept drift is something that may occur within fraud detection as well. It is often the case that criminals resort to different methods if one method proves to be ineffective in accomplishing their goal. If hacking a bank account becomes impossible, perhaps copying a credit card will do the trick. Since AGKA is an implementation for fraud

detection, it is interesting to see how the system responds to concept drift. The streams used in the experiments adhere to (one) pre-defined pattern, but would it catch onto a change in this pattern?

7.4 Relevance

How the findings of this research are relevant with respect to the literature in Chapter 2, is described in Section 7.4.1. A mention of the relevance toward the topic of rule- or case-based reasoning in argumentation is given in Section 7.4.2. The section is concluded with examples of other domains to which the results may be relevant (Section 7.4.3).

7.4.1 Dialogues, decision trees, utilities?

The reason to combine the research fields of argumentation, machine learning and decision theory, was because each field individually was capable of solving a sub-problem: machine learning exploits regularities in data, argumentation provides reasons for a conclusion and decision theory aims to define the 'right' course of action. The aim of this article is to provide an architecture capable of learning from past examples and 'act appropriately' in novel circumstances, while providing logical support for its acts. The hypothesis is that this goal may be achieved by combining the aforementioned fields.

With the hypothesis and the results of AGKA in mind, this begs the question, is it useful to combine the three fields? The author believes the combination is fruitful, but there is a lot more refinement required to successfully integrate the different components. AGKA shows how a system using a classification algorithm can provide an intelligible grounding. Moreover, it is capable of diminishing incurred costs, rather than merely improving classification accuracy.

The dialogue component is restricted in the sense that it only explains the fitted decision tree. The machine learning component is specifically designed to extract knowledge rules from a fitted decision tree model, which serve as arguments for the dialogue component. Using a different machine learning algorithm breaks the interaction between the machine learning component and the dialogue component. As such, there is not yet a universal way of employing dialogues to explain machine learning results.

Moreover, the dialogue component does not add complexity to the decision tree. The only additions are that categorical variables may be used as arguments and proposals may be refuted. Refuting can however be envisioned as taking the opposite path in a tree model. This means the function of dialogues is basically reduced to wording a

decision tree approach, where every turn is simply stating what happens at a node in the tree model.

What are the commonalities between the three fields? When it comes to machine learning and argumentation, a commonality is the support for a classification: Machine learning basically provides a fitted model, while argumentation offers reasons. Although different in appearance, fitted models and reasons may be interchangeable, as shown by the conversion of a decision tree model into knowledge rules for a dialogue. A commonality between argumentation and decision theory is the strength of arguments. Some arguments are stronger than others, which may be expressed by a utility value. Conversely, the 'best' action depends on its utility, which may be conceived as an argument ("This action is better than that one, because the outcome is more preferable"). Machine learning compares to decision theory, because both are an optimisation of the result: For classification the best class needs to be found, while decision theory looks for the best action. AGKA shows how both notions can be connected, as the class is also connected to an action: Giving the illegitimate label leads to an investigation, so the label given has consequences, which should be taken into account for the classification. Can these commonalities be mathematically formulated? Possibly, but that requires further investigation.

7.4.2 Rule- or case-based reasoning?

At a first glance it may seem that AGKA utilises rule-based reasoning: Rules are extracted from the data which are used for a dialogue and rules are provided as support for a classification. It would be blunt to classify AGKA as rule-based reasoning for these matters though.

Consider the shrinking of the database during dialogues (Section 3.3.3). Conceptually, for every turn in the dialogue, the agents are consecutively limited to transactions which are more similar to the one in question. In effect, the dialogue may also be viewed as trying to find the most similar transaction(s) to the current one, in the end deciding on the label based on the properties of these similar transactions. As a matter of fact, instead of giving a rule as support, it is possible to give the final limited database as support.

Is there a clear distinction between rule- and case-based reasoning, or are the two similar? The author believes the two are merely different sides of the same coin. AGKA is a practical application that shows how the process of finding an appropriate label may be framed in terms of both kinds of reasoning. This view is supported by other work in the field of argumentation, such as (Verheij, 2017; Bench-Capon and Sartor, 2003).

7.4.3 Domains of application

Although this research is focused on the setting of fraud detection, the findings may be relevant to other domains. Relevant domains include crime prevention (Section 7.4.3.1) and emergency services (Section 7.4.3.2).

7.4.3.1 Crime prevention

Tied closely to fraud is crime in general. For fraud detection, it is necessary to figure out how fraudsters work and apply measures to prevent them from doing so. Crime could be prevented in a similar fashion.

Utility may even play a bigger role in this domain. With limited resources, it is best to prioritise the crimes that involve the greatest risks or pose the biggest threats. Expected utility could play a pivotal role here.

Moreover, preventing one type of crime may lead criminals to use other methods for achieving their goals. Having a system that adapts to such changes is beneficial over a fixed approach of preventing one type of crime.

7.4.3.2 Prioritising emergency services

Similar to prioritising which transactions should be investigated or which type of crimes should be prevented, deciding where to deploy emergency services could benefit from the approach described here.

For instance, certain traffic accidents may have bigger impacts than others. A flat tire on a country road will likely have smaller consequences than a chain collision on a highway. Therefore, sending in more units to the highway to quickly alleviate the disruption of traffic flow would be wise. Based on collected data on the effect of accidents on disruptions such as traffic jams, a system may be able to discern which accidents should be prioritised and what an appropriate response would be.

Chapter 8

Conclusion

The goal of this thesis was to integrate the research fields of machine learning, argumentation and decision theory in order to provide a system that can detect and prevent fraudulent behaviour. The system needed to be able to learn from data, be self-explanatory and to take the 'best' course of action. The AGKA architecture described in Chapter 3 provides a preliminary practical implementation that combines the three fields.

With the results obtained in regard to AGKA's performance, it can be said that the performance is 'acceptable': In terms of classification accuracy and costs incurred, AGKA falls behind the best performing algorithms in most conditions. When it comes to the rules inferred though, AGKA does find back the rules which were embedded in a stream. The lack in performance may be attributed to the design goals. Incorporating utilities either leads to a 'cautious' or 'ignorant' approach, which affects performance. Whether this effect is desired or not is debatable and depends on the circumstances.

In conclusion, it can be said that the integration of the three research fields is successful, since AGKA achieves a reasonable performance and accomplishes the requirements set out for the system. Nevertheless, more research is required to formalise how machine learning, argumentation and decision theory may be integrated.

Bibliography

R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.

E. Alpaydin. *Introduction to machine learning*. MIT press, 2014.

R. Anderson, C. Barton, R. Böhme, R. Clayton, M. J. G. Eeten, M. Levi, T. Moore, and S. Savage. *The Economics of Information Security and Privacy*, chapter Measuring the Cost of Cybercrime, pages 265–300. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-39498-0.

T. Bench-Capon. Neural networks and open texture. In *Proceedings of the 4th International Conference on Artificial Intelligence and Law*, ICAIL '93, pages 292–297, New York, NY, 1993. ACM. ISBN 0-89791-606-9.

T. Bench-Capon and G. Sartor. A model of legal reasoning with cases incorporating theories and values. *Artificial Intelligence*, 150(1):97 – 143, 2003. AI and Law.

T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. The Wadsworth statistics/probability series. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1984.

R. Briggs. Normative theories of rational choice: Expected utility. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2015 edition, 2015.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

D. Elizondo. The linear separability problem: some testing methods. *IEEE Transactions on Neural Networks*, 17(2):330–344, March 2006.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming (Vol. 3): Nonmonotonic Reasoning and Uncertain Reasoning*. Oxford University Press, Inc., New York, NY, 1994. ISBN 0-19-853747-6.

G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1):185 – 234, 1989.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Min. Knowl. Discov.*, 2(4):345–389, December 1998.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. L. Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481 – 518, 1987.

J. L. Pollock. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, Cambridge, MA, 1995. ISBN 0262161524.

J. L. Pollock. Oscar: An architecture for generally intelligent agents. *Frontiers in Artificial Intelligence and Applications*, 171:275, 2008.

J. L. Pollock and J. Cruz. *Contemporary theories of knowledge*, volume 35. Rowman & Littlefield, 1999.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993. ISBN 1558602402.

R. Reiter. Special issue on non-monotonic logic a logic for default reasoning. *Artificial Intelligence*, 13(1):81 – 132, 1980.

S. Russell and P. Norvig. *Artificial Intelligence: a modern approach (Third International Edition)*. Prentice-Hall, 2010.

S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Trans. Systems, Man, & Cybernetics*, 1991.

B. Verheij. Formalizing arguments, rules and cases. *The 16th International Conference on Artificial Intelligence and Law (ICAIL 2017). Proceedings of the Conference*, pages 199–208, 2017.

M. Wardeh, T. Bench-Capon, and F. Coenen. PADUA: a protocol for argumentation dialogue using association rules. *Artificial Intelligence and Law*, 17(3):183–215, 2009.

M. Wardeh, F. Coenen, and T. Bench-Capon. PISA: A framework for multiagent classification using argumentation. *Data & Knowledge Engineering*, 75:34 – 57, 2012.

G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

I. Zliobaite. Learning under concept drift: an overview. *CoRR*, abs/1010.4784, 2010.