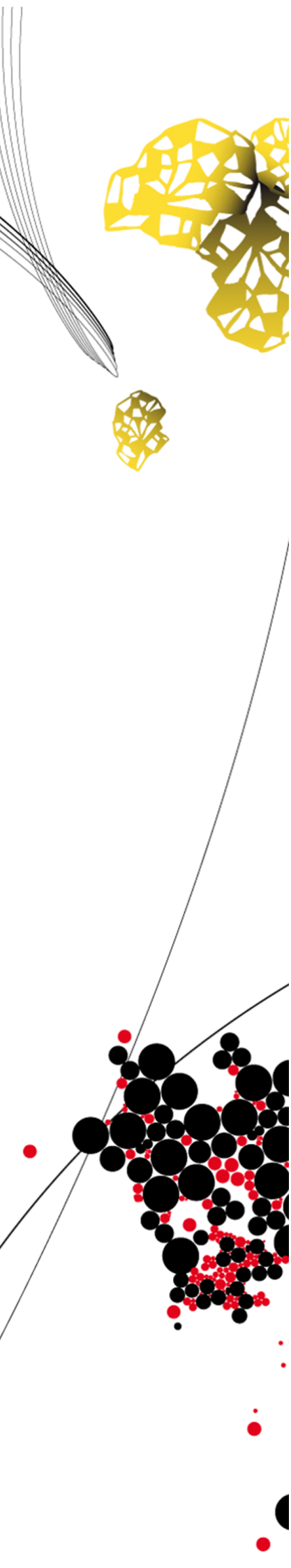


UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

An Explainable Machine Learning Approach to Risk-Adaptive Access Control

Ramon Houtsma

M.Sc. Thesis

March, 2022

Supervisors:
Dr. W. Kuijper
Dr. C. Kolb
Prof. dr. M.I.A. Stoelinga

Design and Analysis of Communication Systems Group

Faculty of Electrical Engineering, Mathematics and Computer Science

University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

Design and Analysis of Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science

M.Sc. Thesis

An Explainable Machine Learning Approach to Risk-Adaptive Access Control

Ramon Houtsma

First Supervisor Dr. Wouter Kuijper
Design and Analysis of Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente

Second Supervisor Dr. Christina Kolb
Formal Methods and Tools Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente

Chair Prof. dr. Mariëlle I.A. Stoelinga
Formal Methods and Tools Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente

March, 2022

Ramon Houtsma

An Explainable Machine Learning Approach to Risk-Adaptive Access Control

M.Sc. Thesis, March, 2022

Supervisors: Dr. Wouter Kuijper, Dr. Christina Kolb and Prof. dr. Mariëlle I.A. Stoelinga

University of Twente

Design and Analysis of Communication Systems Group

Faculty of Electrical Engineering,

Mathematics and Computer Science

P.O. Box 217

7500 AE Enschede

Abstract

Introduction. Existing Access Control Systems, such as Role-based and Attribute-based Access Control, are rigid and complex in large-scale organizations. Risk-Adaptive Access Control provides a dynamic solution but is still in a conceptual stage. This research proposes an artificial neural network as a self-learning decision engine for RAdAC through a proof-of-concept with synthetic data. **Methodology.** Synthetic data of access requests with realistic attributes is generated and then labeled with policy-based permit/deny actions and risk-based risk scores. The risk scores are modeled using attack trees. Three neural networks are trained on the data: a binary classifier to predict actions, and a multi-class classifier and a regression model to predict risk scores. **Results.** Two large datasets of 30000 access requests, a training- and test set, were generated with an 'action' class imbalance of 72/28%. Risk scores produced by the attack tree inversely correlate with the action labels, as expected. The binary classifier achieves a Matthew's correlation coefficient of 0.90; the multi-label classifier achieves a categorical accuracy of 0.90; while the regression model achieves a root mean squared error of 0.10. **Conclusion.** The proof-of-concept shows promise though the model performances are hard to compare since they were achieved on synthetic data. Neural networks appear capable of tackling the risk-adaptive access control problem, demonstration on real-world data is the next step. A probabilistic attack tree shows potential as a risk quantification method, though it remains to be demonstrated in a real-world setting. A decision model prediction explorer using the XAI technique 'LIME' could provide a solution for the explainability of the decisions. This research is one step towards the real-world adoption of RAdAC.

Contents

List of Acronyms

ABAC Attribute-based access control. 1, 4

ACS Access control system. 1, 5

ANN Artificial neural network. 3

MCC Matthew's Correlation Coefficient. 47, 49

ML Machine learning. 1–3

MSE Mean squared error. 45, 47

RAdAC Risk-adaptive access control. 1, 2, 4, 5

RBAC Role-based access control. 1

RMSE Root mean squared error. 45, 47, 49

1 Introduction

1.1 Access Control Systems

Access control systems (ACSs) regulate who gains access to what resources and are thereby essential for information protection. The current industry standard for ACS is *role-based access control* (RBAC). RBAC grants access rights to roles and assigns (multiple) roles to people. For example, the "HR" role within an organization is pre-determined to give access to payroll information; the newly hired human resources employee then gets assigned the "HR" role, thereby inheriting access to the payroll information. This generalization step eases access management. However, current implementations remain tedious to manage, are often rigid, and prone to many exceptions needing to be made as the organization grows. Since employees can have multiple roles, too many of these exceptions can lead to more roles in the organization than people, eliminating the advantage of generalization with RBAC.

The next evolution of ACS is *attribute-based access control* (ABAC). Instead of assigning roles, a set of rules or policies is applied based on attributes associated with people and resources. For example, only people with both attributes 'Manager' and 'ICT department' are allowed to access resources tagged with 'classified' and 'ICT'. ABAC may be used as an independent system or as an extra layer on top of RBAC. Typically ABAC is used complementary to RBAC, for example for determining whether 2-factor authentication is required. This system provides "fine-grained" contextual access control. However, as the quantity of attributes grows, so does the complexity of maintaining a watertight set of policies.

1.2 Risk-Adaptive Access Control

Risk-adaptive access control (RAdAC) aims to base decisions on explicit risk assessments for each decision. The goal is thereby to move towards fine-grained access control while being more dynamic than the existing role-, attribute- or policy-based solutions. No industry standards exist for RAdAC, the idea is highly conceptual. This research explores a self-learning data-driven *machine learning* (ML) decision

model as a solution for RAdAC. Such an ML model learns a dynamic policy to judge future decisions. This method has the potential for being a solution for dealing with dynamic environments, changing requirements, fine-graininess, and scalability. Furthermore, it removes the complexity of manually designing policies, though it does require management and analysis of the model. A self-learning RAdAC model has the potential to shift administration from complex policy maintenance towards monitoring of automated decision-making. First implementations typically would be used as a decision support system, keeping humans in the loop to gain trust and optimize the system. The focus of this research is on demonstrating the ability of ML models to make fine-grained access control decisions, as well as their ability to assign explicit risk scores to access requests with the use of access control data.

1.3 Thesis Outline

In Chapter 2, the research questions are proposed and an overview of the design of the research is given. The latter enables the reader to comprehend the relevance of topics discussed in Chapter 3. In Chapter 3, the state of the art in research domains relevant to this work is discussed. In Chapter 4, the methodology of synthetic data generation and access control labeling processes are explained, and the resulting data and labels are described and evaluated. Next, in Chapter 5, the data preprocessing and the training of the proof-of-concept data-driven decision models are covered. In Chapter 6, the results and limitations of the research are discussed. Finally, in Chapter 7, the research concludes with the answering of the research questions, and with pointers towards future work.

Research Design

2

2.1 Aim

This research aims to explore novel applications of machine learning in the access control domain. The goal is to predict access decisions, as well as risk scores. The prediction of binary decisions serves as a proof-of-concept of ML in access control, while the prediction of explicit risks aims to demonstrate risk-adaptiveness.

2.2 Objectives

To achieve the aim, the first objective is to leverage *artificial neural networks* (ANNs) as the ML technique of choice for the data-driven decision model. Since ANNs need data to learn from, and no suitable data is available, a subsidiary objective is to generate realistic synthetic data of access requests, together with separate methods for assigning decisions and risk score labels. The second objective is to quantify the risk scores of access requests using the attack tree [Sch99] method.

2.3 Research Questions

The following research questions were constructed following the research aim and objectives.

Research question 1. Can neural networks be leveraged to train a self-learning model to automate the decision-making in Risk-Adaptive Access Control?

Neural networks can handle problems of multi-dimensional complexity. This research explores whether a neural network is suitable to make predictions for access control problems.

(a.) What are the advantages and disadvantages of neural networks as a technique for a decision model in access control?

(b.) What data is needed to properly train a neural network for this application?

(c.) What network architecture is required?

Two access request labeling strategies are explored to explore the prediction of implicit and explicit risk assessments. The first labeling strategy is policy-based, assigning permit and deny verdicts using an ABAC system; while the second labeling strategy is risk-based, assigning explicit risk scores.

Research question 2. Can attack trees be exploited to quantify risk scores of access requests?

The aim is to assign risk scores to access requests by using attack trees as the risk quantification method. Risks and attributes will be chosen that are realistic to encounter in organizations and useful to make an access decision.

(a.) What are the advantages and disadvantages of using an attack tree as the risk quantification method for scoring access requests?

(b.) What risks should be determined?

(c.) What attack tree design is required?

2.4 Contributions

The idea of RAdAC is in a conceptual phase, with many advancements that need to be made in the areas of risk identification, legal issues, decision modeling, and integration. By focusing on the implementation of a machine learning decision model, this research contributes to solving complexity and scale challenges of access control and towards manageable fine-grained decision-making. Specifically, the goal of this research is to contribute to risk quantification using machine learning to be able to leverage them in the access control domain. Furthermore, this research studies how attack trees can be used for risk quantification in access control.

2.5 Model Structure

To clarify why certain topics are discussed in Chapter 3, this Section aims to clarify all components of the work, and how they tie together. The main themes are data generation, data labeling, and the decision model.

Figure 2.1 shows how they relate, with the inclusion of decision explanations. First, scenarios of situations where ACS is used, serve as inspiration in the generation of synthetic access requests. Next, to each access request, two labels are assigned which serve as a source of truth in the training of the decision model. The first label is a binary access decision, determined by a set of access policies. The second label is a risk score. The process of assigning an appropriate risk score involves a risk quantification method. The labeled data is then used as input for the training of the decision model, where it attempts to learn the reasoning behind the labels to make accurate predictions on not-seen-before access requests.

Initial investigations were conducted in automated model decision explanations, and a proof-of-concept prototype based on LIME [RSG16] (discussed in Section 3.4.4) was produced. Application of the concept on actual data was deferred to future work. Nonetheless, machine learning model decision explanations are envisioned as an important pillar for eventual industry adoption of RAdAC, where the ability to audit decision reasoning is required. Therefore, it is recommended to investigate further into this topic in future work.

Fig. 2.1.: The structure of the work.

3 Related Work

This section will provide an overview of the current state of the art on the topics of access control, risk quantification, machine learning, and explainable artificial intelligence.

3.1 Access Control Systems

Access Control Systems (ACS) regulate any kind of access, including but not limited to access to (digital) data, access to systems, access to locations, and access to physical objects. Widespread sharing or leaking of information in the information era has become effortless, fast, and therefore prevalent. Organizations try to control their assets by implementing formalized ACS. This section provides a brief overview of traditional ACS. The evolution of ACS shows a trend from simplistic designs where each access right is manually and explicitly defined; to generalized hierarchical systems that are more scalable and maintainable; back to fine-grained ACS through attribute-based methods, such as XACML [Fer+16] and conceptually the application of machine learning.

3.1.1 Access Control Matrix

One of the earliest methods for access control management is the access control matrix (1974) [Lam74]. In this method, the (distributed) file system reads and writes access rights to *one single* matrix, with users and files on the axes (Figure 3.1). A problem is that the matrices are sparse, leading to inefficient use of memory. Furthermore, the size of this matrix grows quadratically with the size of the organization, leading to problems with storing the matrix. These concerns are primarily an issue when computers have low amounts of memory. However, this is not an issue today due to the abundance of memory in modern computers for this type of data. The main issue is that the access control matrix does not model the underlying rules behind the rights in the matrix, thereby providing an incomplete description of the access control security policy.

	File 1	File 2	File 3	File 4	File 5	File 6
Alice	rw	r		rw	w	
Bob			w	rw		r
Charlie						rw

Fig. 3.1.: Example of an Access Control Matrix. The rows depict users whereas the columns represent files. The cells show granted access rights, where 'r' = read and 'w' = 'write'.

3.1.2 Access Control Lists and Capability Lists

Two similar solutions to the large matrices are capability lists and access control lists (ACL). A capability list is a list per user containing his access rights to files he has access to. An ACL is similar, but vice versa; an ACL specifies which users or system processes are granted access to objects as well as what operations are allowed [Shi07]. These lists resolve the problem of large access control matrices and their memory waste by truncating the empty cells. For example, in Figure 3.1, the capability list for Bob becomes just *[Bob: {File 3, [w]}, {File 4, [r,w]}, {File 6, [r]}]*. However, these solutions pose another challenge. In large enterprises, the large number of lists become difficult to manage and maintain. And, like the access control matrix, it does not model the security policies. Thus, with many lists, it becomes increasingly difficult to impose organization-wide access policies.

3.1.3 Role-based Access Control

To generalize access rights for subgroups within an organization, Role-based Access Control (RBAC) was formalized in 1992 [FK92]. RBAC abstracts centrally defined roles, where users are given one or more roles and access rights are assigned to roles (Figure 3.2). RBAC is currently the dominant form of access control [Lee+17]. This solution makes organizational access right management easier compared to ACLs because employees are categorized into groups. New employees can immediately be provided with the right set of access rights by assigning a role corresponding with their job title. Another major benefit is the possibility of a hierarchical structure, where roles can inherit access rights from parent roles. Despite its advantages, RBAC also has its disadvantages. Gathering groups of employees in roles makes it harder to provide granular access for each user. Fine-grained access *is* possible by defining new roles for exceptions. For example, a common scenario is when one member of a department needs an additional access right. In RBAC, a new role has to be created. Lots of these kinds of exceptions occur within an organization, which could lead to

role escalation when the number of roles approaches or outgrows the number of employees. Then, the benefit of using RBAC is partially lost.

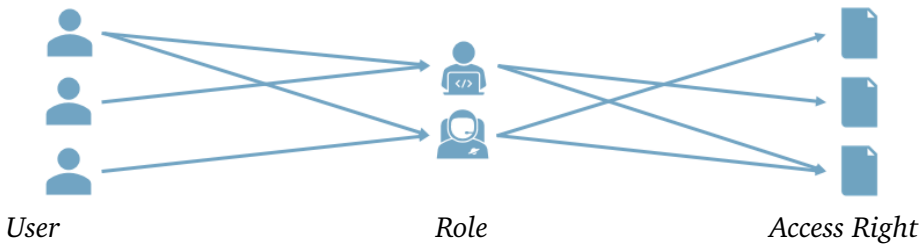


Fig. 3.2.: Visualization of Role-based Access Control. It represents the assignment of users to generalized roles (e.g. software developer) and access rights for roles to information. Not shown is the possibility for hierarchical structure, where roles inherit access rights from parent roles (e.g. 'Back-end developer' inherits 'software developer' permissions next to having back-end-specific permissions).

3.1.4 Attribute-based Access Control

The next step in the evolution of access control systems is Attribute-based Access Control (ABAC). The concept has been in development for some two decades, formalized in 2014 [Hu+14]. It is seeing significant deployment in enterprise [Lee+17]. Next to fully replacing RBAC systems, it is also frequently used as an extension to existing RBAC systems for step-up authentication, by which a user is asked for additional authentication for high-risk requests. Instead of assigning roles with permissions, permission is determined by evaluating environment conditions, subject attributes, and resource object attributes, against a predetermined set of rules or policies (see Figure 3.3). The benefit of ABAC is that it steps away from rigid roles and can dynamically evaluate fine-grained access decisions. An example of such policy is: *only users within department 'x' that are in that department for 'y' years are allowed to access documents with confidentiality level 'classified'*. Such an example works well to explain ABAC, however, it proves difficult to encapsulate all complexity in rules like this. Thus, the challenge is to configure the ACS properly: attributes and policies must be well-defined beforehand. Managing them is difficult as well. Another hindrance is that it is difficult to determine the permissions of a specific subject.

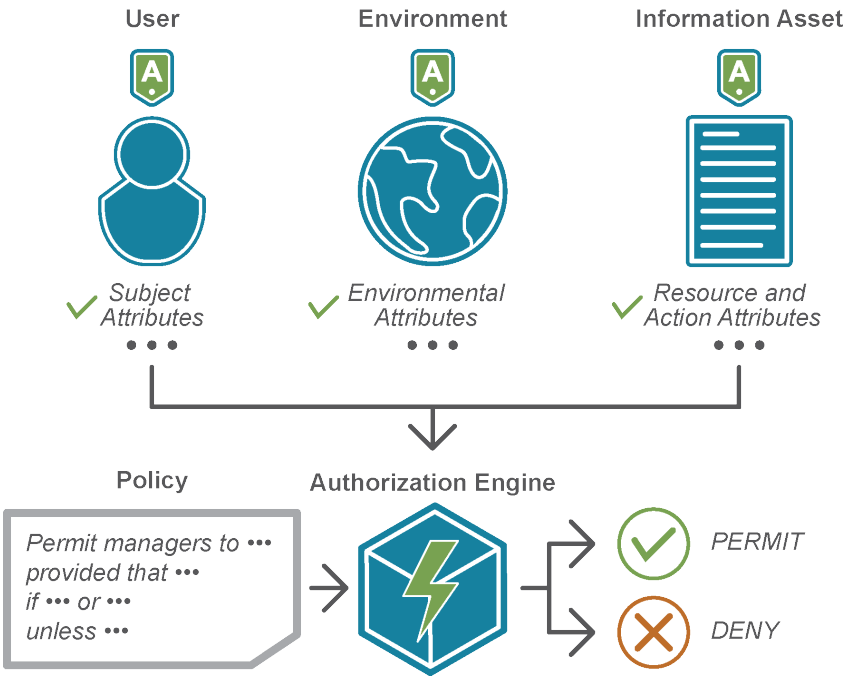


Fig. 3.3.: Scheme of Attribute-based Access Control[1].

3.1.5 Risk-Adaptive Access Control

Overview

Risk-Adaptive Access Control (RAdAC) aims to bring real-time, adaptable, risk-aware access control. Like ABAC, it uses attributes for access control. While ABAC enables more dynamic access control than RBAC, it does not explicitly factor *risk* into the decision-making. Instead, ABAC policy-makers implicitly incorporate risk into their policies. Therefore, McGraw proposed RAdAC, as an extension of ABAC, to enable explicit, dynamic risk assessment. From the white paper [McG04]:

"*RAdAC incorporates a real-time, probabilistic determination of security risk into the access control decision rather than just using a hard comparison of the attributes of the subject and object as in traditional models.*"

RAdAC goes beyond the traditional reliance on static identity credentials, attributes, and policies [FF12]. Essentially, RAdAC assesses the trade-off between *security risk* and *operational need*. [Lee+17]. Here, security risk involves the risk of *sharing*

[1] `https://www.axiomatics.com/attribute-based-access-control/`

information, whereas operational need or benefit involves the risk of *not sharing* the information.

Flexibility

RAdAC overcomes limitations of ABAC, where the use of explicit policies to make access decisions is necessary, by using dynamic decision models. In the previously mentioned ACS, it is up to the system administrators to define roles or policies. These human-defined systems lead to, inherent to humans, the inclusion of emotional decisions, thereby to flawed, biased risk assessments. Furthermore, it is highly complicated for administrators in RBAC and ABAC to define completely covering sets of roles and policies. The dynamic decision models replace the need for drafting roles or policies with the need for risk estimation. In RAdAC, the aim is to deliver a more dynamic system by automating the risk estimation for decision-making. Automating the system grants flexibility, because when risk factors change and the supervisor starts to disagree with the decision model's predictions, these discrepancies can be fed back to the model and the decision model will adapt. This flexibility is what makes the automated decision model risk-adaptive.

3.2 Risk Quantification

One of the great challenges of RAdAC is finding a suitable risk estimation method. Several risk estimation methods have been explored for RAdAC to quantify risks in access control. The inner workings of a machine learning model may explicitly represent risks. They can be used to predict human-made decisions, thereby explicitly computing implicit risk assessments that humans made for their decisions. This prospect, in addition to their flexibility as discussed in subsection 3.1.5, makes them worthy of further research in the access control domain. The following paragraph describes other risk estimation techniques in the literature.

Britton et al. define 27 risk factors to which weights are attributed [BB07]. Dos Santos et al. use an ontology-based approach to adjust the weights of each metric as they become available [dSan+16]. Another popular idea for risk estimation is to assign a subject trustworthiness score and/or object sensitivity score [Kha+13]. To weigh these risks to reach an access decision, fuzzy logic, fuzzy inference, and/or expert interviews can be used [Atl+19; AW19; Che+07]. Others use game theory models [HHR17] or market mechanisms [MCR08]. In probabilistic environments,

the Monte Carlo Simulation is a powerful method for calculating risks of as many possible scenarios as desired for realistic risk probability distribution [Atl+17].

[Sch99] came up with the idea to model threats against a system using trees. By assigning probabilities to scenarios in the tree and calculating the root probability, one can quantify the overall security risk. Such an overview helps in the design of appropriate countermeasures against threats. [Kor+13] extends attack trees with defense nodes that represent defensive measures at any level of the tree. Attack-defense trees allow the tree to formally represent and quantitatively analyze the interaction between an attacker and a defender.

3.3 Machine Learning

3.3.1 Automating Decision Making

If the number of attributes and complexity outgrows the feasibility of man-made policies, it becomes desirable to computationally determine the access control decision model. A promising technique is machine learning (ML), which is a subset of Artificial Intelligence (AI). Machine learning utilizes algorithmic models that analyze and interpret patterns in data and may thereby be used to enable decision-making without the need for human interaction. One can feed such a model historic data, which allows the model to learn which inputs lead to what outputs, and enable it to estimate future decisions for new data. Furthermore, ML is suited to be deployed dynamically as discussed in subsection 3.1.5, by feeding new information back to the model. An example of an ML model is an artificial neural network (ANN). ANNs try to replicate the neurological structure of biological brains by constructing a network of artificial neurons. These networks work surprisingly well when dealing with large amounts of high-dimensional data [ZPŠ14].

3.3.2 Applications of Machine Learning in Access Control

Machine learning has been applied to subproblems of access control, e.g. facial image recognition [SB02; LY17] or fingerprint recognition [Ali+15]. ML can also be used for anomaly detection in network traffic to detect compromised machines [Adl+13].

[Mol+12] use a Support Vector Machine (SVM) as a classifier for their risk-based security decisions. An SVM is a supervised learning method where each input is represented as a point in a k-dimensional space. It then tries to find a hyperplane that separates the two classes. The SVM is suited for two-class problems because it is a linear classifier. However, a polynomial kernel may be applied first to allow non-linear classification. [Zho+19] try automated role engineering in RBAC using SVM. With the use of context features, they assign predefined roles and permissions in a SCADA system using AdaBoost. AdaBoost is a boosting method that combines multiple weak classifiers into a single strong classifier.

[SS20] developed a RAdAC model which used features (e.g. location, time of access, emergency, data sensitivity) to predict whether access was granted in a hospital management system. To this end, they used neural networks and random forests, and in their case found higher accuracy with the random forest classifier. A neural network is made up of cells that work together to produce the desired output. It can learn by re-adjusting the weights of each cell or "neuron" based on the error in the prediction. The random forest is a classifier that outperforms a single decision tree by using majority voting with many decision trees. The individual trees differ from each other because feature randomness is used when building the trees.

Among ML applications mentioned, image recognition and anomaly detection methods are most widely used in other domains. ML specifically for RAdAC is very much unexplored: a thorough search of the relevant literature yielded that the only functioning prototypes were designed by [Mol+12; SS20]. These papers focus on the adaptability and accuracy of the decision model. For RAdAC to become a success, researchers must also focus on the *why* i.e. the explainability of the model.

3.4 Explainable Artificial Intelligence

There are many ML techniques; the inner workings of some are more transparent than others. For opaque models, it may be difficult to explain in hindsight why a certain combination of inputs generated a given output. With some applications, this may not necessarily be a concern, especially when performance is the most important requirement. For other applications like access control, however, it *is* paramount to be able to explain in hindsight why the information was or wasn't shared. The success of ML and the desire to apply ML in areas where transparency is considered important, lead to a new area of research: Explainable Artificial Intelligence (XAI).

Explainable Artificial Intelligence (XAI) was first coined as a term in 2004 by van Lent et al. [vLFM]. The number of academic papers that discuss XAI has spiked since 2016, possibly due to major improvements and successes in machine learning over the years (e.g. self-driving cars, smart personal assistants) and the introduction of the GDPR in the EU [AB18]. This legislation regulates the "right to explanation", which came into effect on May 25, 2018 [Uni16], making it necessary for certain applications to be able to explain the output of an ML model.

The application of ML has potential in RAdAC systems, because ML models may be able to efficiently use all attributes associated with information, users, and context to make classifications with high accuracy. However, with certain ML models such as neural networks, it may be complicated to trace back how classifications were determined by the model. On intuition, there appears to be a trade-off between the transparency and the performance of a model. Transparent models appear to be lesser predictors in contrast to more complex models [Bre01], an intuition caused by the fact that simple models are easier to interpret for humans. If complex models can be made explainable or interpretable, this statement would not hold. The field of XAI is about finding ways to explain these complex, black-box models or make them interpretable. [AB18] provides a good overview of the current state of literature surrounding XAI, which is discussed below.

3.4.1 Terminology

Multiple terms are used in the field of XAI which are close in meaning but do differ: explainability, interpretability, opacity, black-box, glass-box, transparency, understandability, and comprehensibility. A distinction has to be made between the notions of explainability and interpretability especially; the two are often interchanged whilst authors' interpretation of their meaning varies. In this research, it is assumed that the *explainability* of a model means the ability to explain the output of a black-box model, i.e. post hoc explanation, whereas *interpretability* of a model will mean the ability to interpret the inner workings of a model. According to literature, one refers to interpretability also with the name *comprehensibility* [Gui+18], whereas *understandability* refers specifically to the human ability to understand an explanation.

One can imagine a model as a box where certain inputs go in and one or multiple outputs come out. Then, when the model is uninterpretable, it is described as a *black-box* model, meaning it is not possible to see what's inside. Conversely, a model is called a *glass-box* model when it is fully interpretable, because one can peek through

the sides of the imaginary box like they are made of glass. *Transparency* of a model is tied to the box analogy as well, referring to the transparency of the box, meaning the extent to which a model is interpretable on a sliding scale. On the contrary, the proverbial *opacity* is inversely related to its transparency i.e. interpretability.

It should be noted that while these definitions attempt to classify algorithms into discrete categories like black-box or glass-box, the algorithms are mostly on a sliding scale between fully interpretable and completely uninterpretable, with many gray areas.

3.4.2 Interpretable Models

Interpretable models are interpretable because they are considered fully understandable by humans. Their predictions do not need further explanation as their inner workings are intuitive to humans [Gui+18]. Therefore, when an interpretable model performs equally well as an uninterpretable model, one prefers the interpretable model. Because of this property, interpretable models are used to provide global or local explanations of an uninterpretable model, as further discussed in Section 3.4.4. Examples of recognized interpretable models are *decision trees, rule-based models*, and *linear models* [Gui+18].

3.4.3 Model-specific Methods

Model-specific explanation methods try to explain the inner workings of a black-box model. For ANN's there are saliency maps for example, which visualize how heavily the inputs to a node influence its output. Another example is reconstructing feature importance.

3.4.4 Model-agnostic Methods

Model-agnostic explanation methods are methods that do not depend on the inner workings of the model. Rather, they take a black-box approach to the model and aim to explain the relationship between the outputs and the inputs. One can distinguish global and local explanation methods, as discussed in the next paragraphs.

Global Explanation Methods

Global explainability methods aim to explain the overall logic of a model. They try to understand the whole logic of a model and follow the entire reasoning leading to all possible outcomes [Gui+18]. The model-agnostic methods try to do this by learning an interpretable, glass-box model over the inputs and outputs of the underlying black-box model. In other words, they mimic a complex, opaque model with a simple, transparent model to explain its logic. An inherent flaw of this method is that the simple model of the complex model is always inaccurate to some extent. If the simple model is always in agreement with the complex model, it would not be necessary to use the complex model at all. [@Rud19] However, even when it is not a completely accurate replication of the functionality of the complex model, the interpretable model may still be useful to give insights.

Local Explanation Methods

Local explanation methods aim to explain the reason for a *specific* decision; the prediction of a black-box model for a single input instance. In 2016 the model-agnostic method 'LIME' [RSG16] was released, which approximates a black-box model locally in the neighborhood of any prediction of interest, as illustrated in Figure 3.4. These methods have great potential because they reason in a way that humans do. This kind of reasoning is called counterfactual explanation. Instead of "Why A?", people often ask "Why not B?" [Mil17].

By applying such reasoning, one does not have to explain the whole model, but rather each local decision and what the minimum conditions would be that would have led to the desired alternate decision. Such a method could apply to RAdAC since an access control decision is a binary problem, therefore the alternate decision is always the opposite class. A potential downside of local explanation methods is that they may be computationally expensive since a new local model has to be trained for each prediction requiring explanation.

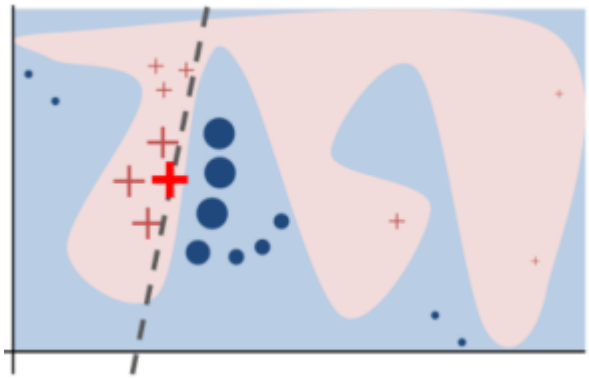


Fig. 3.4.: An example in the two-dimensional space of a local linear model within a complex decision space. Suppose the figure axes resemble earth coordinates and the two areas resemble whether an access request would be granted or not if the requester was positioned within those coordinates. Then the shortest distance vector of a single access request to the local linear model resembles the distance and angle the requester should move to change the outcome of the decision model.

4 Data Generation and Labeling

This section will provide the results of the process towards a realistic ground truth for the data-driven decision model. It will discuss data generation, access control policies, risk quantification models, implementation, and the results. The terms 'row', 'entry', 'sample', and 'access request' are used interchangeably. The terms 'column', 'variable', 'feature' and 'attribute' are used interchangeably too.

4.1 Data Generation

The data-driven decision model needs data to learn from to predict access or risks. As part of a prediction challenge, Amazon published anonymized data [@Ama13] of employee access requests to resources, labeled with *permit* and *deny* labels. However, the policy behind the labeling is not known, and for the risk-adaptive decision model preferred labels are quantified risk scores. To the best of the author's knowledge, no public data exists where access requests are labeled with a risk score. Therefore, synthetic data is generated. The goal of the generation of synthetic data is to generate features and feature values that could realistically be collected in an access control system of a real organization. The synthetic data contains hypothetical access requests for resources from employees of a fictitious organization. During generation, after each access request is generated, it is labeled. Two labels are assigned, a policy-based label and a risk-based label. These labels represent the initial truth that the model learns from. Section 4.2.2 discusses the labeling processes.

The data generation process starts with generating a set of employees with features associated with each employee. Examples of features associated with a person are nationality, age, job title, department, or clearance level. Next, a set of resources is generated with characterizing features. Examples of features associated with a resource are author, creation date, classification level, category, or department. Finally, access requests are created by randomly sampling a person, a resource, and adding contextual features like date, time, or location, with these feature values sampled from a finite set. All feature values are sampled from a probability distribution appropriate to the feature. For example, a resource is sampled from a

discrete uniform distribution; time of day is sampled from a continuous uniform distribution; while age is sampled from a realistic population distribution. As many requests as deemed necessary can be generated. The necessary number of access requests required for proper model training depends on the complexity of the data.

The formal description of the dataset of access requests is as follows: let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a finite set of attributes, and for attribute $a \in \mathcal{A}$, let $\mathcal{V}_a$ be the domain of a. Let the set of access requests be $\mathcal{Q} = \{q_1, \ldots, q_n\}$, then for $q \in \mathcal{Q}$ the access request consists of $q = \{(a_1, v_1), \ldots, (a_k, v_k)\}$ where k is the number of attributes $|\mathcal{A}|$ and $v_i \in \mathcal{V}_{a_i}$.

Feature statistics are collected, like the mean, the standard deviation for continuous variables, or frequency analyses for categorical values. These will be compared to real statistics. For example, for the feature age, the histogram is compared to the age distribution in the Netherlands to determine how realistic the feature is generated.

4.2 Data Labeling

Labels are assigned to the access requests as the source of truth for the decision models to predict. Two labeling strategies are applied to explore the prediction of implicit and explicit risk assessments respectively. The first labeling strategy which explores implicit risk prediction assigns a binary *permit* or *deny* label according to a set of policies. The idea is that behind the man-made policies are implicit risk assessments and that if the model is able to correctly predict the labels without knowing the policies, it has deducted the determined risks successfully. The second labeling strategy which explores explicit risk prediction assigns a risk score to each access request. For the risk-adaptive decision model, quantified risk scores are as preferred as ground truth, such that the model predicts risks instead of verdicts. To determine these risk scores for our generated dataset, risk quantification using attack trees is explored. Currently, the probability of events is used to determine the risk.

4.2.1 Policy-based Labeling

A policy consists of rules to evaluate an access request. Given a policy p and an access request $q \in \mathcal{Q}$, then $d(p, q) = 1|0$ represents the decision of evaluating p against q, where 1 and 0 indicate *permit* and *deny*. Multiple policies can apply

concurrently. When applying multiple, potentially contradicting policies, a policy-combining algorithm is necessary. The chosen policy-combining algorithm is *Permit-unless-deny*, one of the official evaluation methods of XACML [@Oas20]. Let the set of policies be $\mathcal{P} = \{p_1, \ldots, p_n\}$. Then D_q is the final label after evaluating q against available policies according to the *Permit-unless-deny* policy-combining algorithm, as formalized in Equation (1). This algorithm ensures that the final outcome is 1 unless one of the policy decisions is 0, by taking the product of the policy decisions. This is a deviation from the more common *Deny-overrides* combining algorithm since it is more applicable to this use case. Where *Permit-unless-deny* has just two outcomes, *permit* and *deny*, *Deny-overrides* policies have two additional outcomes: *Indeterminate* and *NotApplicable*. However, these extra outcomes are not necessary since with each access request all attributes are present and every policy is designed such that it applies to each access request. In practice, *Permit-unless-deny* amounts to permitting the access request unless at least one of the available policies has a 'deny' decision. The evaluation method was chosen because it is practical. However, a side-effect is that the more policies are enabled, the more access requests are labeled as 0. For a balanced distribution between decisions, the set of policies is and should be carefully selected.

$$D_q = \prod_{p \in P} d(p, q) \tag{1}$$

4.2.2 Risk-based Labeling

Of the three risks in Figure 4.1, only the security risk is modeled due to time constraints. An attack tree design is chosen to model the security risk. Each node in the attack represents a security risk for the scenario, for example, the risk for data theft. Attributes and attribute values of an access request influence leaf nodes in a tailored way. For example, when the employee's nationality is not the same as the organization's host country, it can raise the security risk for data theft. Figure 4.2 shows another example of this method. The security risk is quantified as a probability, so raising the security risk means increasing the probability of the event happening. The root node determines the overall security risk associated with the access request, i.e., the label. The root probability is determined through bottom-up calculation of the tree, where each node probability is calculated according to Equations (2), depending on whether it is an AND or an OR node.

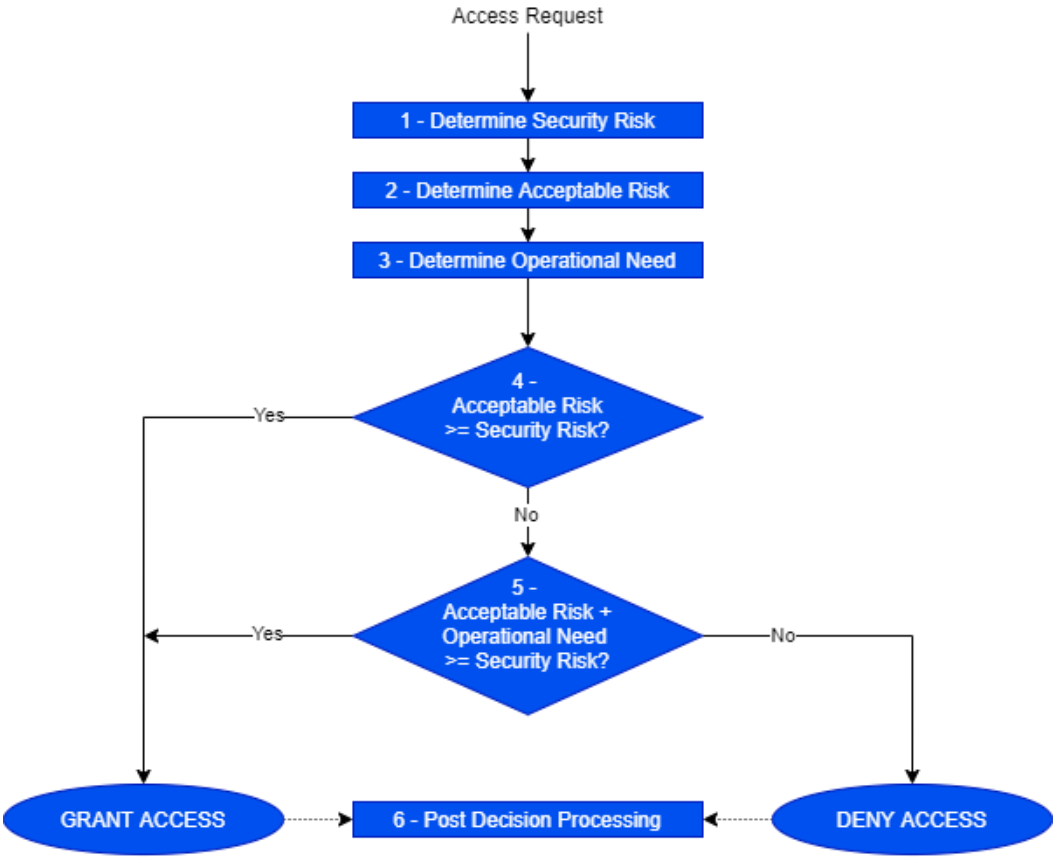


Fig. 4.1.: Flowchart for RAdAC inspired by McGraw [McG04]. The main difference with McGraw's chart is that here all risks are determined beforehand.

$$\begin{aligned} AND \text{ node:} \quad & P(A \cap B) = P(A) \cdot P(B) \\ OR \text{ node:} \quad & P(A \cup B) = P(A) + P(B) - P(A \cap B) \end{aligned} \tag{2}$$

4.3 Results

All code is available at [@Hou21].

4.3.1 Tools

The main tool used for the data generation and labeling was the Python programming language version 3.7.7 [@Pyt18]. Major Python modules were Pandas [@Pan20]

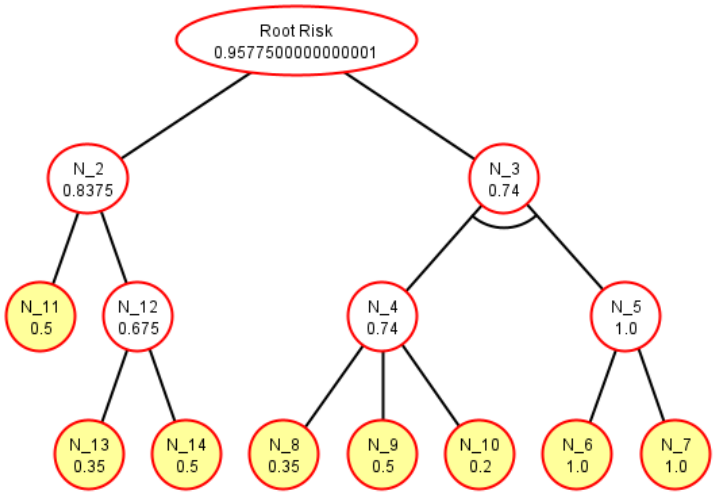


Fig. 4.2.: Example of an attack tree.

version 1.1.3 and NumPy [@Num20] version 1.19.1 for data processing; and the faker module [FO22] version 11.3.0 which provides sources of fake data.

4.3.2 Data Generation

Four data tables were generated: first, one of employees and one of resources; next, one of access requests; and last one of request labels. The employees table is used to draw resource owners from. The employees and resources tables are used to generate access requests. Finally, the labels generated by the PolicyDecisionPoint and RiskAssessmentPoint are stored in the requests table. Figure 4.3 shows the data tables with the flow of information. The choice of attributes is a set of attributes that are generally encountered within a reasonably large organization.

Sections Employees Table on Page 24, Resources Table on Page 26, and Requests Table on page Page 27 describe the attributes and their probabilities of the employees, resources, and access requests. The attributes are generated with distributions that are thought to be realistic with what one might encounter in a real organization. In that regard, the aim was to create an imbalanced dataset where the large majority of access requests are approved, with a small percent of attribute values providing cause for rejection. One of the challenges for the decision model is to correctly reject the anomalies. The attributes and attribute values are inspired by real-world attributes and used to model real-world phenomena. The organization used as inspiration for the data generation is 'TNO', a Dutch research institute.

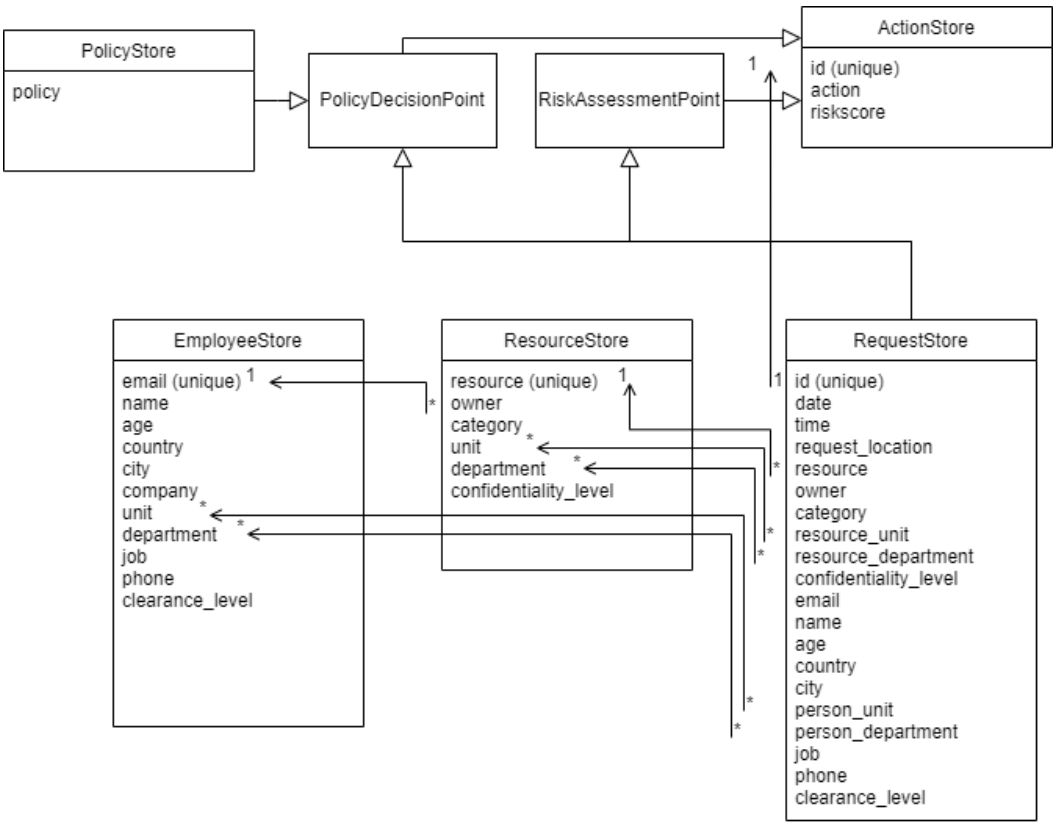


Fig. 4.3.: Overview of data tables, with *1-to-1, 1-to-many, and many-to-many* relations between attributes.

Employees Table

The employee table contains data of 2000 trusted employees, similar to the size of 'TNO'. The attributes for each employee are described in Table 4.1, while their probabilities are described in Table 4.2. The nationality distribution, unit names, department names, and job titles were inspired by TNO. The clearance levels are Dutch background check levels: the most lenient and most common background check is the 'Verklaring omtrent het gedrag' (VOG) is a declaration of decent behavior, which can be requested at the municipality of residence. The 'Verklaring van geen bezwaar' (VGB) is a security check performed by the Ministry of the Interior and Kingdom Relations, where levels 'C' to 'A' represent increasing levels of thoroughness.

Email addresses are unique identifiers, while names are generated from localized lists in the faker module. Ages are generated from the general population distribution in the Netherlands: first one of five age groups is selected with probability equal to their prevalence, then a random age is chosen from within that age group. The nationality probability within the company is inspired by TNO. The city always corresponds with

#	*Attribute*	*Description*
1	email	email address
2	name	first name and last name from country locale
3	age	years of age rounded to the nearest integer
4	country	nationality, *alpha-2* country code representation
5	city	city in country
6	company	existing company from country locale
7	unit	unit name
8	department	department name
9	job	job title
10	phone	phone number with country locale calling code
11	clearance level	level of background check thoroughness

Tab. 4.1.: Employee attribute descriptions.

the person's nationality. There is only a 0.05 probability that someone from outside the organization is allowed access to company resources: these represent trusted third parties, consultants for example. The unit and department names are names of real departments within TNO. The names are not mentioned, since there are too many and they are immediately preprocessed to numeric representations. The job titles are real common job titles within TNO. Phone numbers are random with accurate calling codes. Clearance levels are generated with decreasing probabilities as the background checks become more thorough. These declining probabilities follow the idea that the more confidential topics are within the organization, the fewer eyes see them.

#	*Attribute*	*Probability*
1	email	unique identifier: *[firstnamelastname]@[company].[tld]*
2	name	uniform random sample from a localized list
3	age	equal or greater than 18, generated according to Dutch population age distribution
4	country	50% chance of same nationality as company home country, otherwise random sample from list
5	city	random sample from country locale
6	company	95% chance of 'TNO', 5% chance of other company in country locale
7	unit	24 departments divided over the 6 units; random choice from unit departments
8	department	department name
9	job	random choice from $3 \cdot 4 = 12$ job title combinations: *[Junior, Medior, Senior][Researcher, Manager, Consultant, Scientist]*
10	phone	country locale calling code + random number
11	clearance level	30/25/20/15/10% for *'None', 'VOG', 'VGB-C', 'VGB-B', 'VGB-A'* respectively

Tab. 4.2.: Employee attribute probabilities.

Resources Table

The resources table contains 10000 unique resources, which may be requested by subjects. The attributes for each entry are described in Table 4.3, while their probabilities are described in Table 4.4. The resources are random file names with file extensions corresponding to the file category. File categories are six general categories of file types. Each resource is assigned an owner from the employees table. The resource's unit and department values are sourced from the same options as the employee's unit and department since they are created within the same company. The five confidentiality levels are English translations of typical classification levels broadly used worldwide and within TNO.

The probabilities of interest in Table 4.4 are the department and confidentiality levels. The idea is of the department probability is that an employee mostly creates resources in their own department, since most projects are worked on there. However, interdepartmental cooperation is possible, therefore there is a small chance that an owner creates files outside their department. The confidentiality levels generated

#	*Attribute*	*Description*
1	resource	file with file extension according to file category
2	owner	employee email from employees table
3	category	file type
4	unit	unit where the resource was created
5	department	department where the resource was created
6	confidentiality level	level of resource confidentiality

Tab. 4.3.: Resource attribute descriptions.

are lower or equal to the owner's confidentiality levels. The comparison between clearance and confidentiality levels is possible since both have the same number of ordinal possible values. The assumption is that an employee with clearance level 'None' can only create 'Unclassified' files, while an employee with clearance level 'VGB-B' can only create resources with 'Secret' confidentiality or lower.

#	*Attribute*	*Probability*
1	resource	unique identifier
2	owner	random choice of employees from employees table
3	category	random choice from 'audio', 'image', 'office', 'text', 'video'
4	unit	100% person's unit
5	department	95% chance of owner's department, otherwise another department from the same unit
6	confidentiality level	random choice from: *'Unclassified', 'Restricted', 'Confidential', 'Secret', 'Top Secret'*, excluding confidentiality levels that are higher than the owner's clearance level

Tab. 4.4.: Resource attribute probabilities.

Requests Table

Each access request always requests a resource from the resources table. The subject is the user requesting access to a request. Two tables of 30000 access requests are generated, one for training the decision model and one for testing the final decision model. 30000 records should provide enough samples for the data-driven decision model to learn from. Both tables draw from the same employees table and resources table, use the same attribute probability functions, therefore should have similar distributions. The only difference between the two tables is the seed with which they were generated. The access request attributes are described in Table 4.5, while their probabilities are described in Table 4.6. The contextual attributes in each access request are 'id', 'date', 'time', and 'request_location'. Since no access request is guaranteed to be unique, each access request is assigned an identification number.

These are referred to in the labels table. The date and time of an access request are sampled from January 2022, simulating all access requests within this period. The request location is added as means to simulate and detect impostors or hacking attempts.

#	*Attribute*	*Description*
1	id	request identification number
2	date	date of request
3	time	time of request
4	request_location	*alpha-2* country code
5	resource	requested resource from resources table
6	owner	requested resource owner
7	category	category of requested resource
8	resource_unit	unit where requested resource was created
9	resource_department	department where requested resource was created
10	confidentiality level	confidentiality level of requested resource
11	email	identifier of subject requesting the resource
12	name	name of subject
13	age	age of subject
14	country	*alpha-2* country code of subject
15	city	city of subject
16	person_unit	unit of subject
17	person_department	department of subject
18	job	job of subject
19	phone	phone number of subject
20	clearance_level	clearance level of subject

Tab. 4.5.: Request attribute descriptions.

Most of the attributes in Table 4.6 are associated with the subject or the resource. Here the contextual attributes and the scenarios where other default attributes differ are discussed in this paragraph. The generated dates have a 0.95 probability of being a business day of the week since realistically only a small number of access requests is expected during the weekend. The same holds for the time of day of the access request, with a 0.95 probability of the access request being generated during working hours. There is a large chance that the request location is either from the person's home country or the offices' country, as opposed to being from a different country, which would be suspicious. Resources requested are always resources from the resources table. What resource is requested depends on who does the request. There is a 95% chance that the subject is a legitimate employee from the employees table; is a small chance of the subject being a random outsider; and a very small chance of a deliberate impostor. An impostor tries to assume a legitimate employees' identity and escalate their access privilege by increasing the clearance level. If this

does not work, the age is changed to assume more authority. The impostor requests a resource from the assumed identity, to make use of the privilege escalation. When the subject is an employee, there is a high chance of requesting their own resource, a smaller chance of requesting a resource from the same department, an even smaller chance of requesting a resource from the same unit, and the smallest chance of requesting a resource outside their own unit. These probabilities are added to the data to simulate resource appetite: why would an employee want to have resources from other units?

4.3.3 Data Labeling

A fourth data table with labels associated with access requests was created in the data labeling process. The label table contains three attributes, which are described in Table 4.7. The actions are generated by the PolicyDecisionPoint using policy-based labeling, whereas the risk scores are generated by the RiskAssessmentPoint using risk-based labeling. Like the requests table this table is created twice, with the training labels associated with the training requests and the test labels associated with the test requests.

Policy-based Labeling

Due to the nature of the *permit-unless-deny* policy-combining algorithm, each additional policy that is enabled results in a smaller or equal number of requests being permitted, i.e., labeled as 1. Since the aim is an imbalanced set of labels where the large majority are labeled as 1, only ten out of twelve designed policies are active. Due to the binary outcome of each policy, each policy is practically a yes/no question. The policies created are described in Table 4.8.

The effects of each policy on the 30000 training and 30000 test access requests combined are listed in Table 4.9. If all policies were to be active, it results in a 59/41% permit/deny rate. Since the goal was to generate an imbalanced dataset with a majority of permit labels, the decision was made to disable the two policies that denied the most requests: policies 4 and 6 (see Table 4.8 for their descriptions). It was deemed that the requirement to be the owner and in the same department was too narrow, however, policy 7 covers these attributes. After dropping the two policies, and with the other ten enabled, it results in a 72/28% permit/deny rate. This is more realistic than with all policies enabled. Though more imbalanced data is more realistic, the current imbalance is considered realistic for a real-world

organization and will serve as the ground truth for data-driven machine learning to predict.

Risk-based Labeling

Four risks were defined, that serve as leaf nodes for the attack tree: 1) third party risk; 2) impostor risk; 3) hacker risk; and 4) risk-appetite risk. Figure 4.4 shows the composition of the attack tree that calculates the security risk of an access request. A flat structure using an OR-node Equation (2) was chosen since each of the risks is independent. Four risk levels were defined: these can be assigned to the leaf nodes. Since the attack tree calculates probabilities, each of the risk levels is assigned a probability: *LOW*: 0.2; *LOWMED*: 0.35; *MED*: 0.5; and *HIGH*: 1.0. Note that since high risks should result in a high root risk score, low risks receive low probabilities and high risks receive high probabilities.

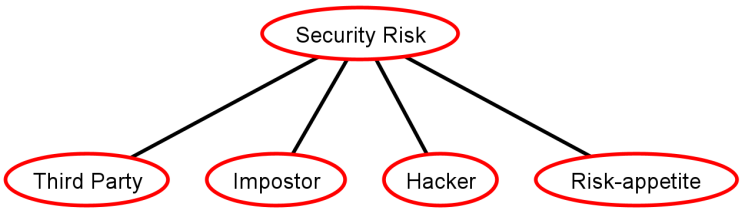


Fig. 4.4.: Attack tree design with leaf probabilities derived from access request attributes, in turn determining the root probability which is the security risk.

The third-party risk is *LOW* when the subject is in the employee table, and *MEDIUM* when it is not (and is, therefore, a third party). The rationale is that employees are already somewhat trusted. A third party has a randomly associated department and corresponding unit, job title, etc, which can be interpreted as with which department in the organization the third party is collaborating. The impostor risk is *LOW* when the subject is in the employee table; *MEDIUM* when one of the subject's attributes in the access request does not match the corresponding attribute of the subject in the employee table; and *HIGH* when that irregular attribute concerns the clearance level, since that attribute enables the impostor to request a more confidential resource. The hacker risk is *LOW* when the request is made within the business week between office hours; *LOWMED* when the request is made during the weekend; *MEDIUM* when the request is made outside of office hours; and *HIGH* when the request is not made from the subject's country or the office location. The risk-appetite risk is

LOW when the resource owner matches the subject; *LOWMED* when the resource is within the same department as the subject; *MEDIUM* when the requested resource is not in the subject's department but within their unit; and *HIGH* when the resource is outside the subject's unit. This risk is assigned to requests since in general resources should only be viewed by persons who have reason to. The further one is removed from the location the resource is created, the higher the risk that it is an unnecessary or illegitimate request.

All scenarios lead to the assigning of calculated risk scores to access requests. The risk scores are values in the interval $[0.59, 1.00]$. These risk scores are then normalized to the interval $[0, 1]$ for better interpretability to the decision model. The occurrences of risk scores are described in Table 4.10. Note that are a finite number of risk scores. This is a result of assigning probabilities through risk levels, there are no linear correlations between attribute values and leaf node probabilities. The risk scores are compared with whether the action, produced by the PolicyDecisionPoint, was permit or deny. Thereby, the perceived accuracy of the risk scores can be determined. The most important observation is that high risk scores are associated with deny verdicts, while low risk scores are associated with permit verdicts, showing a strong inverse correlation.

#	*Attribute*	*Probability*
1	id	unique sequential integer
2	date	date from January 2022, 95% of a business day versus 5% for a weekend day
3	time	24-hour notation, 95% of being between the business hours 7:00 and 19:00, 5% for outside of business hours
4	request_location	70/25/5% chance of person home country, company country, or random other country, respectively
5	resource	if not impostor or outsider: 80/15/4/1% chance of requesting random choice from own resources, from same department, from same unit, or other resource, respectively. If impostor: random choice of 'own' resources. If outsider: random resource.
6	owner	100% requested resource owner
7	category	100% category of requested resource
8	resource_unit	100% unit where requested resource was created
9	resource_department	100% department where requested resource was created
10	confidentiality level	100% confidentiality level of requested resource
11	email	95% of a legitimate employee from employees table, 0.25% of email from employees table and being an impostor, 4.75% of being an outsider
12	name	100% name of subject
13	age	>99.75% age entry associated with email in employees table, small chance of random age when subject is an impostor, and random when email is an outsider
14	country	100% country of subject
15	city	100% city of subject
16	person_unit	100% unit of subject
17	person_department	100% department of subject
18	job	100% job of subject
19	phone	100% phone number of subject
20	clearance_level	>99.75% clearance level of subject, <0.25% higher level than clearance level of subject in case of impostor

Tab. 4.6.: Request attribute probabilities.

#	*Attribute*	*Description*
1	id	request identification number indicating which access request is labeled
2	action	1 or 0, indicating a permit or deny verdict respectively. Generated by the PolicyDecisionPoint
3	riskscore	quantified risk value: a higher score indicates smaller chance of approval. Generated by the RiskAssessmentPoint

Tab. 4.7.: Labels table.

#	*Policy*	*Description*
1	Person exists policy	Is the request subject in the employee table?
2	Resource exists policy	Is the requested resource in the resource table?
3	Location policy	Is the request location where the subject lives or at the office?
4	Is owner policy	Is the subject the owner of the resource?
5	Resource in unit policy	Is the resource created in the same unit as the subject's unit?
6	Resource in department policy	Is the resource in the same department as the subject?
7	Owner or department policy	Is the resource owned by the subject or in the same department as the subject?
8	Age policy	Is the subject over 18?
9	Weekday policy	Is the request dated to a business day Mon-Fri?
10	Time policy	Is the time of day within office hours 07:00-19:00?
11	Company policy	Is the subject's company 'TNO'?
12	Clearance policy	Is the subject's clearance level higher than the resource's confidentiality level?

Tab. 4.8.: Available policies.

Policy #	*Permit #*	*Deny #*
1	57221	2779
2	60000	0
3	57035	2965
4	45297	14703
5	57109	2891
6	52064	7936
7	54342	5658
8	59996	4
9	56943	3057
10	57211	2789
11	57231	2769
12	57041	2959

Tab. 4.9.: Separate policy effectiveness on available access requests. Counts are cumulative for both the training and test requests tables.

Risk score	*Action*	*Counts*
1.00	1	27
1.00	0	8073
0.76	1	0
0.76	0	4
0.68	1	0
0.68	0	27
0.61	1	0
0.61	0	287
0.59	1	0
0.59	0	22
0.49	1	0
0.49	0	762
0.37	1	13
0.37	0	5419
0.34	1	0
0.34	0	306
0.20	1	5765
0.20	0	2085
0.00	1	37211
0.00	0	0

Tab. 4.10.: Risk score counts, by action, cumulative over the training and test request tables.

5 Data-driven Decision Model

This section will provide the results of the process towards a realistic data-driven decision model. It will discuss data pre-processing, the model architecture, hyperparameter tuning, model training and the final model performance. The terms 'row', 'entry', 'sample', and 'access request' are used interchangeably. The terms 'column', 'variable', 'feature' and 'attribute' are used interchangeably too.

A neural network is trained with the dataset. The ML model of choice should be supervised since it is a classification problem. Neural networks are interesting because they can handle high complexity and they are notoriously black-box, making them interesting for eXplainable Artificial Intelligence (XAI) analysis. Hyperparameters have been chosen through hyperparameter optimization. Neuron weights have been fine-tuned with the learning process.

This research explores a proof-of-concept of an artificial neural network as the decision model over other machine learning techniques due to its ability to handle highly complex multi-dimensional data. This advantage does not necessarily come to light in this work: since the data is not overly complex, other machine learning techniques' prediction performances would probably be similar. However, the argument can be made that the neural network technique is explored for future purposes due to its empirical success in other domains. In addition, the aim of this research is not to achieve the highest prediction performance possible; the main focus is to demonstrate the innovative application of machine learning in the access control domain. The second argument to explore neural networks in the access control domain is that they are notoriously opaque, making it a challenge to explain the predictions. If this challenge can be overcome for neural networks, then it is clear that machine learning models have a place in access control. One assumption that is made in this research, is that all information to make a decision is readily available since neural networks can't handle incomplete data.

5.1 Data Preprocessing

Data preprocessing is the process of making collected data suitable for subsequent interpretation by cleaning or transforming the data. For this research, the goal of the data preprocessing was to make the data interpretable for a neural network algorithm specifically. The methodology used was: first *cleaning* (5.1.1); next *discretization*(5.1.2); then *encoding* (5.1.3); and finally *standardizing* (5.1.4).

5.1.1 Cleaning

Cleaning of data is the process of filling in missing values or dropping entries and/or features. Entries with missing values can either be dropped, or the missing values can be filled in to preserve the data. For example, in entries where the *age* is missing, the value can be filled in by taking the mean or median of all ages in the data. Note that filling in missing data is not applicable since missing data is not modeled in the synthetic data. However, the data is still cleaned by dropping previously generated features.

Reasons for dropping entire features are to prevent unintentional discrimination or that there is simply no interpretation possible. It is up to the model owner to determine whether a feature could be wrongfully interpreted by the model, which may depend on the goal of the eventual model. For example, it would be fine to predict *mortgage values* based (partly) on income, whereas using income to assess the *risk of fraud* could lead to discrimination and stigmatization of persons with low income, as has happened with the system for risk indication “SyRi” [@Roo20]. An example of an uninterpretable feature is *unique sequential ID*. It is known beforehand that the ordinal character of this feature does not discriminate the data, since the values are 100% distinct i.e., unique. The model is therefore not able to discriminate based on the feature, thus must not attempt to and the feature should be dropped.

5.1.2 Discretization

Discretizing a feature means categorizing continuous values into a limited number of categories to learn more about categories within the data or to ease the evaluation of the data. This process can dramatically decrease the number of inputs for the model, improving prediction interpretability. Common examples of discretization are grouping ages in life phases or grouping prices in price ranges. A well-known method

of discretization is *equal width binning*, which divides the continuous variable into ranges of the same width. The width of each range is determined by Equation 3, where w is the width, $\min$ and $\max$ represent the minimum and maximum values in the range and x is the number of desired categories.

$$w = \frac{\max - \min}{x} \tag{3}$$

5.1.3 Encoding

Two methods of encoding are used to make the data interpretable by the model: *ordinal* and *one-hot* encoding. While some models can work with categorical inputs, like decision trees, a neural network cannot, thus encoding needs to take place. Ordinal encoding is applicable when discretizing a categorical variable and there is implied ordinality. An example of ordinal encoding is transforming $[\text{easy}, \text{medium}, \text{hard}] \rightarrow [0, 1, 2]$ by assigning integers to categories. A neural network is not able to understand the relationship between the words, but it does understand the ordered relationship between the integers.

Ordinal encoding imposes an ordinal relationship, however for some categorical variables no such relationship exists, e.g. with colors. In this scenario, one-hot encoding is a solution. As visualized in Table 5.1, one-hot encoding creates a new variable for each possible value in the original variable, and for each entry assigns a 0 or 1 to each new variable depending on whether the value was in the original variable for that entry. This method ensures that the model is not able to derive any ordinality where it is not meant to.

ID	Color
#1	Red
#2	Green
#3	Blue

$\rightarrow$

ID	Color_Red	Color_Green	Color_Blue
#1	1	0	0
#2	0	1	0
#3	0	0	1

Tab. 5.1.: Example of one-hot encoding.

A potential drawback that should be monitored is that the number of inputs for the model grows dramatically with the number of unique values in the variable that is one-hot encoded, which may harm model training and hurt the performance. When encountering this issue, a solution is to lower the number of categories, either by dropping entries with categories of low prevalence or discretizing the least prevalent categories into one. A frequency analysis could be performed to determine the frequency threshold for which categories should be combined.

5.1.4 Standardization

Standardization or normalization is a transformation on all values of a feature such that the mean is removed (centering) and it is scaled to unit variance (scaling). When there is a large difference in scale between input variables, the resulting model may learn large weights, resulting in low sensitivity towards small scale variables, which is not desirable. The centering and scaling transformations happen independently on each feature based on the statistics of the feature in the data. Input samples of a feature are transformed according to Equation (4), where x is the input sample, u the mean and s the standard deviation of all input samples of the feature. This transformation scales and centers any feature independently to a range of roughly $[-1, 1]$ with a mean of 0, though it does not guarantee a range since it standardizes the standard deviation. Min-max normalization Equation (5) is a more common procedure, which does guarantee a range of $[0, 1]$.

$$z = \frac{x - u}{s} \tag{4}$$

$$z = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{5}$$

5.2 Decision Model Training

5.2.1 Model Architecture

The architecture for the decision model is a feedforward neural network. In a feedforward neural network, the information must flow in one direction, as opposed to a recurrent neural network. An RNN is not needed for this application because each sample needs to be classified independent of the previous sample(s). A convolutional neural network (CNN) doesn't apply either, since this is neither a time-series problem (1-dimensional CNN), nor a computer vision problem (2D CNN), nor a 3D image or video problem (3D CNN). Therefore, the feedforward NN is most suitable for this access control problem.

A visualization of a feedforward neural network is shown in Figure 5.1. The input layer values represent all values of one sample (access request) after preprocessing, from which the network derives its prediction for one sample. An edge represents a weight factor. Hidden node h_0 is calculated from all its input vectors by $h_0 = f(x)$,

where $f(x)$ is the activation function and $x = \sum_{n=0}^{N}(i_n \cdot w_n) + b$, where N is the number of inputs, i_n the input, w_n the corresponding weight and b the bias. The activation function scales the output of the neuron. Common activation functions are *linear*, *sigmoid*, *softmax*, *Rectified Linear Unit* (ReLU), and *tanh*. Linear scales $f(x) = x$, sigmoid scales to $[0, 1]$ according to Figure 5.2, The ReLU scales to $f(x) = \max(0, x)$ and should be used for hidden layers and the tanh function to a range of $[-1, 1]$. The ReLU should normally be used for hidden layers while the others are usually for the output layer. The ReLU has become the default hidden layer activation function since it resolves the vanishing gradient problem present with tanh and sigmoid functions. The vanishing gradient problem occurs with neural networks with multiple hidden layers. With backpropagation the gradient becomes increasingly smaller due to the chain rule with backpropagation, resulting in the weights near the inputs hardly updating. The constant gradient of ReLU results in faster learning. The softmax activation function is different than the others in that it is a probability distribution among output classes, making it suitable for multi-class classification problems. For binary class prediction, two output nodes with softmax activation behave the same as one output node with sigmoid activation, since the two softmax outputs represent probabilities for opposite classes (if $p(o_1) = 0.6$, then $p(o_2) = 1 - p(o_2) = 0.4$) while the one sigmoid output is a probability for the positive class. Since behavior is the same, having one output node with sigmoid activation is preferred, due to one less node needing to be calculated and fewer weights needing updates through backpropagation (see Section 5.2.2).

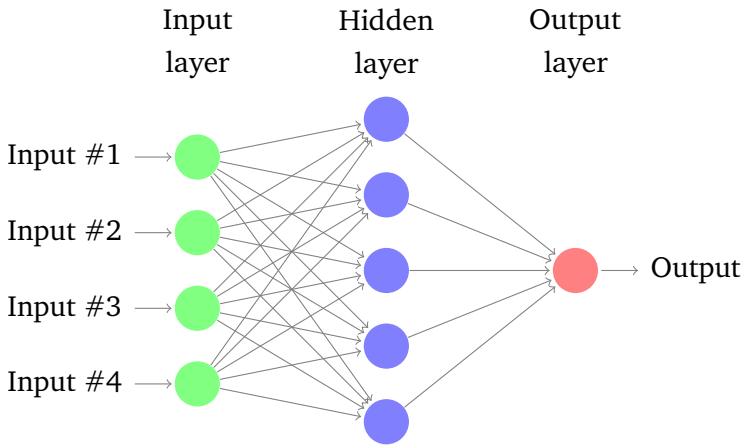


Fig. 5.1.: Depiction of a feedforward neural network.

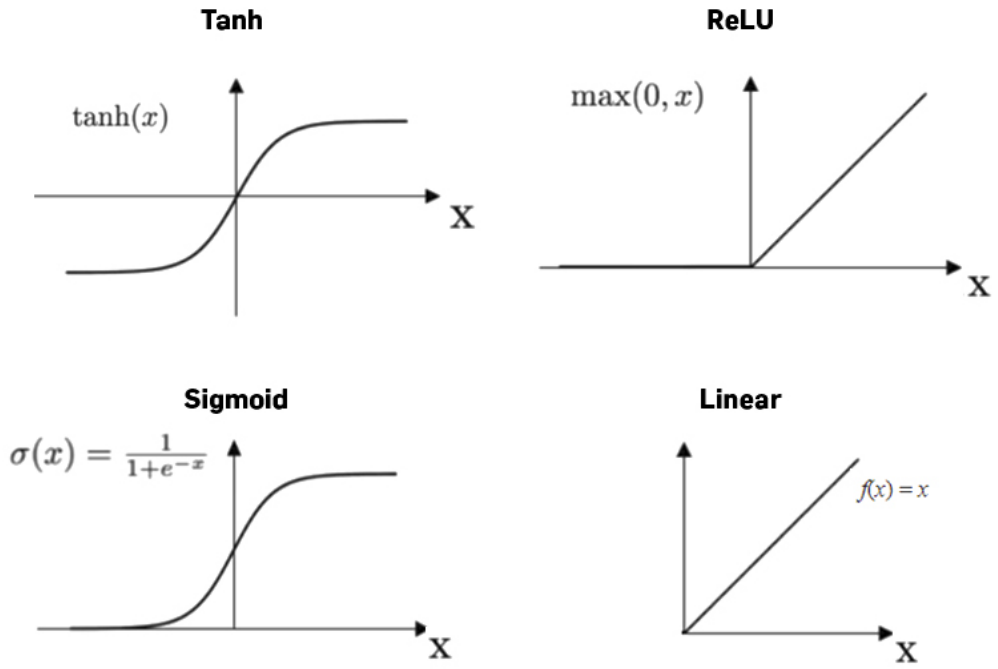


Fig. 5.2.: Common activation functions.

5.2.2 Training the Neural Network

Sampling

First, the data is split into a training, validation, and test set. The *training* set is used to train the model and update its weights; the *validation* set is used to monitor performance on unseen data *during* training to prevent overfitting; while the *test* set is used to measure performance, *after* training, on the final model. Overfitting is when the model is fitted so well to the training data that it will predict worse on unseen data since it failed to generalize. Before splitting the data, the samples are randomly shuffled to ensure unbiased evaluation by the model. Since the original dataset may be sorted in some way, after splitting the data it could cause the training set to not be similar to the validation and test sets. Without a good distribution of samples in the training set, the model will not perform well on unseen samples. Next, the dataset is split into a train set, a validation set, and a test set. Common split sizes are 80%/10%/10%, 70%/15%/15% or 60%/20%/20% for training, validation and test sets respectively. Undersampling the majority class or oversampling the minority class may help the model to give the minority more weight. For example, if the majority class is 95% of the samples, the model may be lazy enough to always predict 1 (left of Figure 5.3), since the loss is very small, and it has a 95% accuracy

(this also proves that accuracy is not a great performance indicator for imbalanced datasets). In this situation, the model hasn't learned anything, so when offered with 50% minority samples it only achieves 50% accuracy. After having oversampled the minority class in the training set, the model on the right in Figure 5.3 can predict the opposite class.

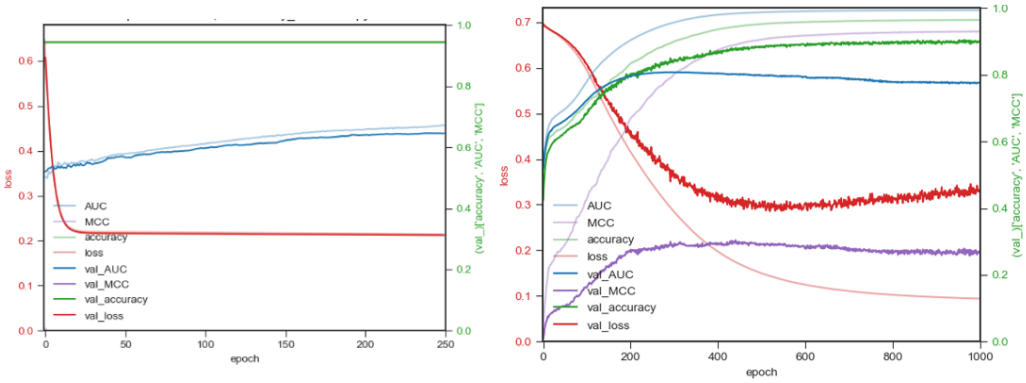


Fig. 5.3.: Model on the left always predicts 1 due to a heavily imbalanced training set (95% majority class). This problem can be overcome by oversampling the minority class in the training set as seen on the right.

Training

The network is trained by offering the neural network samples from the training set. A sample is forward propagated through the network and the network ends up with a prediction. The prediction error is the difference between the prediction and the expected output, which is measured by the loss function. Relevant loss functions in this work are the *binary cross-entropy*, the *categorical cross-entropy*, and the *mean squared error (MSE)*. Binary cross-entropy is a loss function used in binary classification tasks and is the negative average of the log of corrected predicted probabilities. The categorical cross-entropy used in multi-class classification tasks is used when the true value can be only one of multiple categories and is a measure of how similar two discrete probability distributions are. The MSE is used when doing regression, and it is calculated as the mean of the squared differences between the true and predicted values.

The samples are fed in batches. After each batch is completed, the internal parameters of the model are updated through backpropagation. The optimizer influences how the weights optimize, through the learning rate for example, which is a multiplier for the weight change. Three optimizers were explored: Stochastic Gradient

Descent (SGD), Root Mean Squared Propagation (RMSprop), and the ADAM optimizer. SGD is the most standard method for backpropagation, with a set learning rate that determines how much the weights change for a loss. Although it is possible to apply learning rate decay manually. The RMSprop and ADAM optimizers have adaptive learning rate optimization, designed specifically for machine learning to achieve local minima/maxima as quickly as possible. It computes individual learning rates for each parameter. One behavior observed with the ADAM optimizer is that it does not converge in some situations, where it jumps back and forth around one local optimum. AMSGrad is a variant that seeks to fix the convergence issue, by using the maximum of past squared gradients rather than the exponential average to update the parameters. An epoch passes when the model has processed all samples. After each epoch, performance on the validation set is measured (see Section 5.3). In addition, the average loss on the training and validation set is logged. By testing on the validation set, model performance on unseen data is modeled, enabling detection of overfitting on the training data. Figure 5.4 shows that when the validation starts increasing, the model is overfitting and training can be stopped.

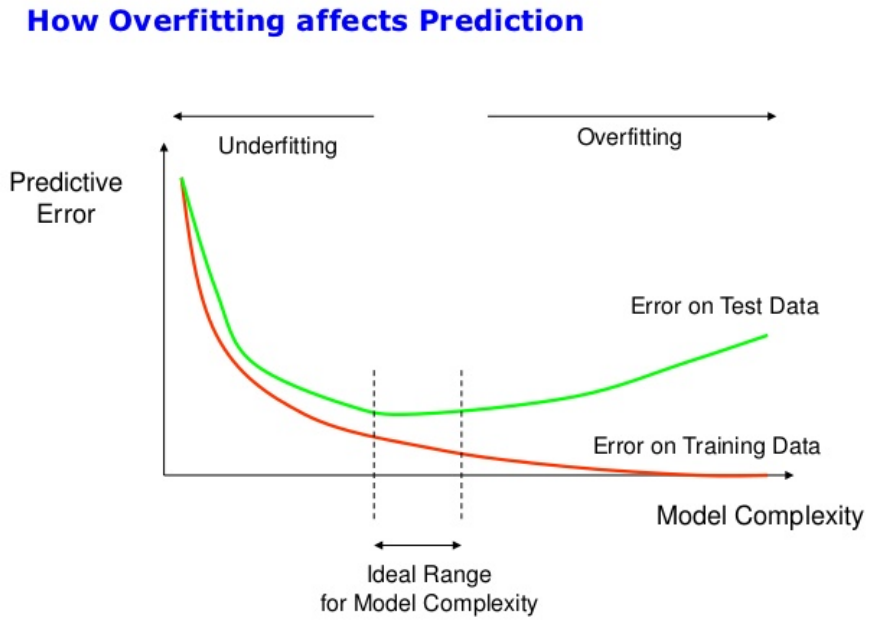


Fig. 5.4.: When the error on validation samples rises, the model is overfitting on training data.

Hyperparameter Tuning

Hyperparameter tuning is the process of comparing the performance of models with different hyperparameters to find the best configuration. There is a difference between normal parameters as discussed in Section 5.2.2 and hyperparameters. Normal parameters are the weights and biases within the network. Hyperparameters are parameters that define the architecture of the network or influence the characteristics of the training process. For example, one might want to discover whether changing the number of hidden layers or nodes within a layer improves prediction performance. One could compare different optimizers, compare training convergence speed, or efficiency with different batch sizes.

When all hyperparameters are determined, an exhaustive grid search can be leveraged to explore all possible configurations. For each configuration, the model is trained with K-fold cross-validation. K-fold cross-validation is the strategy of splitting the training data into k folds, and then training a model with the specified configuration of hyperparameters on each of them. Make sure that the test data is not used for hyperparameter tuning since that introduces data leakage and overfitting. By taking the average performance, one can determine which set of hyperparameters generalizes best without overfitting. With grid search, the number of possible configurations explodes quickly. When the number of hyperparameters and possible options is too great, other options like random search are possible. However, for this research exploring other methods was not necessary.

Final Model Training

After the best hyperparameters are determined, the model is ready for final training. Now, the model is trained extensively on all available data excluding test data, to fully optimize its internal parameters and achieve the best performance. It is acceptable to use the same data as was used in hyperparameter tuning since the aim of hyperparameter tuning was to find the hyperparameters that best generalize on this data. Last, the optimized model is tested on the unseen test set.

5.2.3 Evaluation Criteria

For binary prediction, the model predicts the positive class with a probability between 0 and 1. These predictions are rounded to end up with a prediction of 0 or 1, which can then be compared with the true labels. This allows the creation of a confusion

matrix consisting of true positives, false positives, false negatives, and true negatives. There is no perfect method to combine this table into one perfect performance score, however, solid methods are available. The most common metric is the accuracy Equation (6), which divides the sum of true positives and true negatives over the total number of predictions. Accuracy is acceptable as a metric for balanced datasets, however, with imbalanced datasets, it does not reflect well the model's ability to predict the minority class. A common method is the Area Under the Curve (AUC), which is the area Receiver-Operating-Characteristic (ROC) curve. The ROC curve plots the true positive rate (TPR) or recall ($TPR = \frac{TP}{TP+FN}$) versus the false positive rate (FPR) at different classification thresholds, see Figure 5.5 for example.

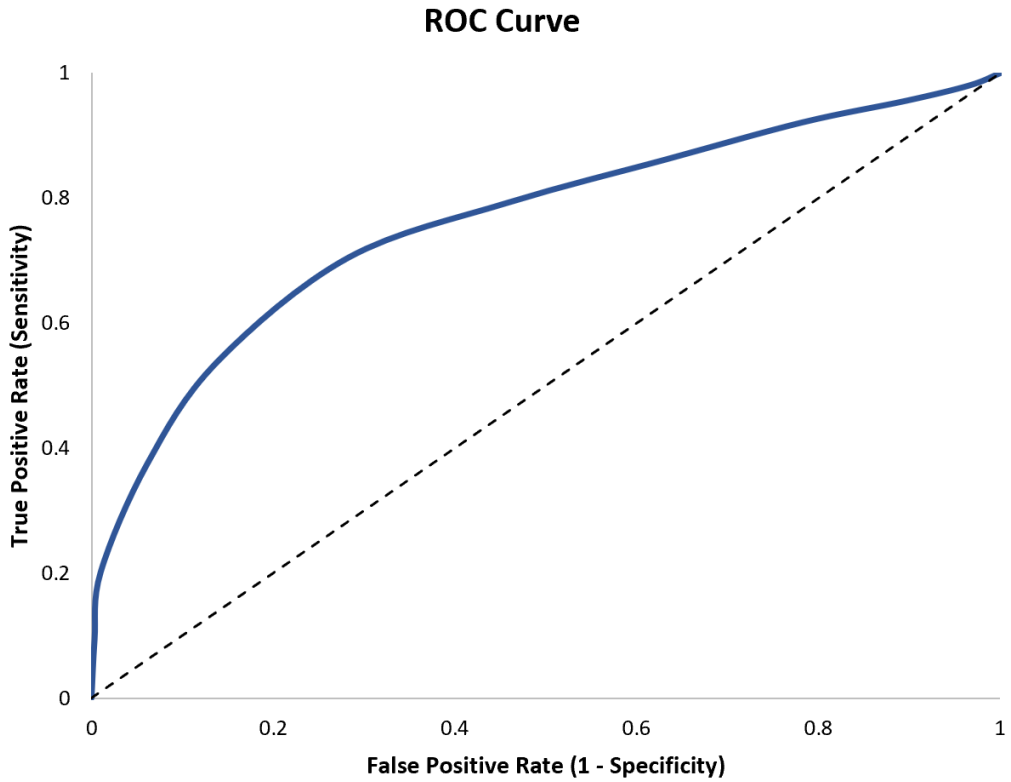


Fig. 5.5.: Example of an ROC curve.

Another upcoming metric for imbalanced datasets is the Matthews' Correlation Coefficient (MCC) Equation (7). Recent research [CWJ21] shows that this is a very good performance predictor for imbalanced datasets. It returns a value between $[-1, 1]$, where a coefficient of 1 is a perfect prediction, 0 means the model performs as if it were randomly classifying, and -1 that it predicts the opposite of expected.

For multiclass prediction, accuracy is the most useful metric as defined in Equation (8), where n_true is the number of correct predictions and n is the total number

of predictions. For regression, the *mean squared error* (MSE) and *root mean squared error* (RMSE) are common performance metrics. MSE is the mean of the squares of the differences between predicted values and target values, as shown in Equation (9). The squaring is to prevent negative predictions negating positive targets or vice versa. The RMSE is simply the root of the MSE. The RMSE is preferred since it has the same units as the dependent variable. The lower these metrics are the better. The R^2 metric is frequently used too since it is scale-independent. For this research, it is not required since the risk scores lie in the $[0, 1]$ interval.

$$\text{Accuracy(binary)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{7}$$

$$\text{Accuracy(multiclass)} = \frac{n\_true}{n} \tag{8}$$

$$\text{MSE} = \frac{\sum(y_i - \hat{y}_i)^2}{n} \tag{9}$$

$$\text{RMSE} = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n}} \tag{10}$$

5.3 Results

All code is available at [@Hou21].

5.3.1 Data Preprocessing

Tools used for data preprocessing are Pandas version 1.1.3 [@Pan20] and NumPy version 1.19.1 [@Num20]. The first step in preprocessing is dropping insignificant data. Note that features were generated that were not going to be used by the decision model. They are dropped because they don't discriminate enough to help make a decision. The feature 'name' for example: since full names are nearly unique the model cannot generalize for what a name should be to allow or deny access. If the names were kept in the dataset, they would need to be one-hot-encoded due to

being a nominal variable. One-hot-encoding creates a new feature for each unique value. This increases the number of inputs to the decision model dramatically without improving the decision-making capability. If features were going to be dropped for decision-making, why generate them in the first place? The features were generated to support the goal of generating a realistic dataset that could be encountered in a real organization. The features were inspired by real-world attributes to model a real-world organization, which doesn't imply that all available information is or should be used in the decision-making.

Due to the beforementioned argument, the following attributes are dropped from the requests table: 'id', 'resource', 'category', 'name', 'city', 'company', 'job', and 'phone'. Of these, only the company was used by the PolicyDecisionPoint and RiskAssessmentPoint. Leaving the company out reduces the number of inputs by 180, and provides a challenge to the decision model. The others were not used in the labeling process.

Next, the 'date' attribute values are binned to days of the week, and then one-hot-encoded. This reduces the number of inputs for the dates to seven. The 'time' values are processed to seconds, translating it from a nominal to a numerical attribute. The 'age' attribute doesn't require preprocessing. The 'clearance_level' and 'confidentiality_level' attributes are ordinally encoded. A new attribute is created, where the value for an access request is 1 when the 'owner' value is equal to the 'email' value, and 0 when they are not equal. This ensures that the 'owner' and 'email' values can be dropped. Dropping these attributes lowers the number of inputs to the decision model by thousands since each possible value becomes a separate input due to one-hot-encoding. The 'resource_unit', 'resource_department', 'person_unit', and 'person_department', are one-hot encoded, adding 60 attributes total. The 'request_location' and 'country' attributes are one-hot encoded, resulting in each resulting in 37 attributes.

The preprocessing brings the number of attributes to a total of 146 inputs for the decision models. Next, all values of all attributes are normalized on a per-attribute basis. This assures that they are equally weighted by the decision models. Finally, the order of access requests is shuffled to ensure that there is no order when splitting the data into training, validation, and test sets.

5.3.2 Architecture and Hyperparameter Tuning

The tool used to design, train, and predict with neural networks, is TensorFlow 2.1.0 [@Ten20]. Deterministic operations were enabled and all random generators

were seeded with the same seed (0), such that the results are reproducible. Three models were trained: one binary classifier for predicting permit/deny actions. Two models were trained for predicting risk scores: a regression model and a multi-class classifier. Since the risk labeling process resulted in 10 different risk scores, the multi-class classifier was introduced while exploring the regression model. To be able to predict the risk scores with the multi-label classifier, the risk score labels are one-hot encoded when training and predicting with the multi-label classifier. Each model takes the same inputs, and neurons in hidden layers always have ReLU as the activation function. The output layers are different: the binary classifier has one node with 'sigmoid' activation; the regression model has one node with 'linear' activation; the multi-class classifier has 10 output nodes with 'softmax' activation. The loss functions are different too: the binary classifier uses 'binary cross-entropy', the regression model MSE; and the multi-class classifier 'categorical cross-entropy'.

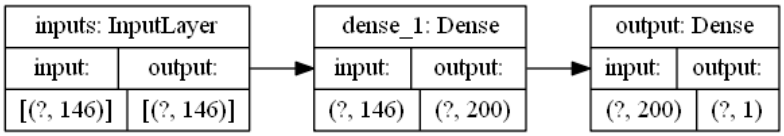
The hyperparameters which decide the learning rate, the width of the network, the depth of the network, and how often parameters are updated, are tuned separately for each model. The options are the following: *optimizer*: 'Adam', 'Root Mean Squared Propagation (RMSprop)', or Stochastic Gradient Descent (SGD) with learning rate 0.1; *hidden layers*: 1, or 2; *neurons per hidden layer*: 100, or 200; *batch size*: 512, 1024 or 4096. A grid search of these options results in 36 possible configurations for each model, or 108 configurations in total. 0 hidden layers and 3 neurons per hidden layer were trialed as well, but were performing consistently significantly worse and are therefore excluded from the grid search since they would cause the number of configurations to quadruple. The number of neurons per layer was chosen such that one option has fewer neurons than the number of inputs, and the other has more. The 100 neurons per layer function like the left-hand side of an auto-encoder, in that it learns a representation of the input.

5-fold cross-validation is applied such that each configuration's performance is generalized over the whole dataset. The 30000 training access requests are used for hyperparameter tuning. The metrics are different for each model: the binary classifier uses the *Matthew's Correlation Coefficient* (MCC); the regression model the RMSE; and the multi-class classifier the 'categorical accuracy'. The metrics are measured over the validation sets, and the average performance over the 5 folds is stored for each configuration. The 5-fold cross-validation means that $108 \cdot 5 = 540$ models were trained to find the best configurations, over the course of 7,5 hours. Each configuration has 1000 epochs to optimize the metric score. Early stopping was implemented with a patience of 50, which meant that if the model had not improved for 50 epochs, then training was stopped and it continued with the next configuration.

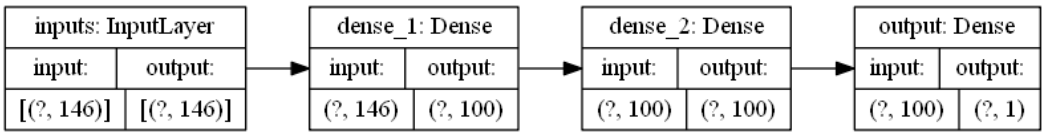
Model	*Monitor*	*Hidden layers*	*Neurons/layer*	*Batch size*	*Optimizer*
Binary	MCC	1	200	512	RMSprop
Multi-class	Cat. Acc.	2	100	512	SGD
Regression	RMSE	2	100	512	Adam

Tab. 5.2.: Hyperparameter tuning best configurations.

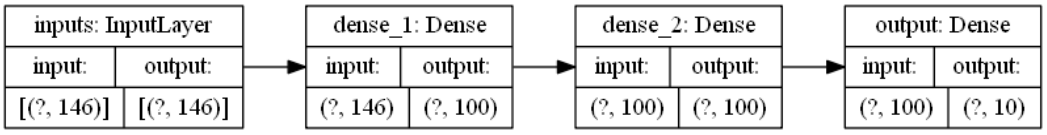
The results of the hyperparameter tuning process are shown in Table 5.2. The results show that the batch sizes of 1024 and 4096 were fast to optimize but proved too large for best optimization. Interestingly, the three models benefitted performed best different optimizers. The final model architectures are displayed in Figure 5.6.



(a) Binary classifier architecture.



(b) Regression model architecture.



(c) Multi-class classifier architecture.

Fig. 5.6.: Decision model architectures.

5.3.3 Final Models Training and Performances

First, deterministic operations were enabled and all random generators were seeded with the same seed, such that the results are reproducible. To be able to predict the risk scores with the multi-label classifier, the risk score labels are one-hot encoded when training and predicting with the multi-label classifier. Each model is initialized with the best hyperparameter configurations from Table 5.2. The 30000 training access requests and labels are used for training the final model. Each epoch, 10% of the shuffled data is used for validation. After training for 1000 epochs each model had reached its optimal weights at some point for the best validation metric score, after which it would start overfitting. The weights of the model with the best validation metric score are saved and used to compile the final model. The final

model was then measured on the test set containing 30000 not-seen-before test access requests and labels.

The binary classifier optimized after 632 epochs with an MCC score of 0.90, as seen in Figure A.1. This is a good score, not to be compared with accuracy due to the scale of $[-1, 1]$. The ROC AUC is 0.91 as seen in Figure A.2, while the accuracy is 0.89. The confusion matrix is shown in Figure A.3.

Figure B.1 and Figures B.2a to B.2g on Page 72 show the confusion matrices of the binary decision model predictions on requests that should be denied, specified per data labeling policy, sorted by performance. These results show which policies the decision model learned well from the training data, and which it didn't. '@0.5' means that the decision threshold is 0.5: the rounding threshold above which the prediction is rounded to 1 and below is rounded to 0. n is the number of access requests that were denied by the policy, thereby labeled 0.

The company policy is badly predicted, almost randomly, which is expected since the company attribute was dropped in preprocessing. The model performs worst on the clearance policy. This is a disappointing outcome since the clearance and confidentiality levels of the subject and resources should be major indicators for whether an access request should be permitted or not. This could be explained by that the model failed to learn a comparison between two of the 146 inputs, which is expectable. An new attribute could have been created in preprocessing that compared these two values. In contrast, it becomes apparent that the other policies were trivial to predict.

The multiclass classifier optimized after 452 epochs with a categorical accuracy of 0.90, as seen in Figure A.4. The confusion matrix is shown in Figure A.5. The regression model optimized after 93 epochs with an RMSE of 0.10, as seen in Figure A.6. The boxplots (presented in Figure A.7) of the predictions show that the regression model approaches the increasing trend of risk scores, though it overestimates the three lowest risk scores, and underestimates the higher risk scores. Note that the boxplots don't visualize well how many data points are in each boxplot. For example, there are only three predictions for the risk score of 0.76, while the boxplot for risk score 0.0 considers 18699 cases. See Table 4.10 and divide the occurrences by two to get an approximation of the prediction counts for each risk score.

6 Discussion and Future Work

6.1 Data Generation

The shortcoming of synthetic data is that it is arguable how well it represents the real world. In this case, it was hard to access real data since access control is mostly a topic of high confidentiality: organizations don't like to make their organizational structure or resources public. Still, being able to test on real access control data would be highly beneficial to make the case for data-driven, risk-adaptive access control.

The synthetic data in this research has not been reviewed by industry experts. Future collaboration with system administrators of a company can provide legitimacy as to the realism of the data through their expertise without revealing their company's data.

The synthetic data can be made more realistic by generating more outliers, like for example in the Kaggle Employee Access Challenge[@Ama13]. Currently, attribute values are selected by picking from a limited set of options either uniformly or with value probabilities much greater than $\frac{1}{n}$, therefore when generating n attribute values the attributes don't have outliers since even values far from the mean occur relatively frequently. Currently, the least prevalent scenario is an access request with a subject age under 18, which occurs twice in a set of 30000 access requests, which represent all requests in the timespan of a month. In reality, this would not occur this often. In addition, the decision model is still able to correctly predict this scenario consistently. A solution is to draw attribute values from a distribution, e.g. normal distribution, rather than from a limited set of options within a range.

6.2 Data Labeling

6.2.1 Policy-based Labeling

The shortcomings of the current policy-based data labeling are that it does not use a standard protocol or a standard evaluation method. This does not matter for the main goal in this research, which was to provide labels through a set of policies, to demonstrate the learning capability of neural networks in the access control domain. However, for optimal progress towards real-world implementation, it helps to approach the labeling process through proven industry standards, like XACML[@Oas20].

6.2.2 Risk-based Labeling

The shortcoming of the current risk-based labeling is that it is a small demonstration. One risk is modeled, where two or three would be more complete, since then the access decision can be made by comparing multiple predicted risks. Especially the modeling of operational need is a major challenge, due to it being extremely situationally dependent.

In this work, the risk is modeled as a continuous value. It is worth exploring whether the model is more useful if ordinal risk categories are predicted instead. It is more common in risk management to work with risk categories instead of continuous values of arbitrary range since these are more intuitive and therefore more effective for humans to act on. When the data-driven decision model is leveraged as an advice engine, then risk categories may be more effective. When the data-driven decision model is leveraged as the primary decision engine and only the decisions are monitored by humans, then continuous risk values may be more precise.

An attack tree approach was used to quantify the risk, where proponent success probabilities were modeled. This provided a solution for risk scores for the model to train on. However, in terms of realism, it is not difficult to find risk scores that are not logical. To improve, a more elaborate set of risk scenarios could be modeled with the attack tree. Modeling other factors next to probability in the tree can be explored as well, such as cost for the proponent. Last, one can compare methods discussed in Section 3.2 to conclude whether attack trees are most suited for this problem or not.

6.3 Data-driven Decision Model

In this research, one of the main drivers for working with neural networks was their ability to handle complexity, and for being opaque which is a tough problem in access control. The main goal was to demonstrate the technology in a novel domain. Achieving the best performance was not an objective. To improve performance, more experimentation with neural network architecture is advisable, e.g., more hidden layers, more neurons, or different types of neural networks replacing the feedforward nature of the network in this research. Whether neural networks are the best performing machine learning algorithm for this problem, was not tested. In future work, other machine learning algorithms like random forests may compete for the best performance.

The drawback of neural networks is that they require that all data is available. In practice, this is not always the case. This challenge could be overcome by assuming inputs based on values previously seen, or testing predictions for all possible values of missing inputs to see whether they are important. Research trends towards *imputation*, which is to replace missing inputs with values inferred from known data, or to replace the response neurons at the first hidden layer by its expected value [Śmi+18].

6.4 Explaining the Decisions

The shortcoming of machine learning techniques is that it is difficult to discover why exactly a specific decision was made. This ability is critical in the access control domain for traceability and auditability. Management must be able to explain why secret information was shared, or why mission-critical information was not shared. Less critically, in the case that access was denied, it is useful if the explanation clarifies or recommends what the requester could change to be permitted access next time. An initial prototype to explain decision model decisions has been explored using Titanic data [@Tit12], see Appendix C. The prototype explores a prediction, in this case, whether or not a passenger survived. The LIME engine suggests what features had what importance by modeling the local decision space. The dashboard then allows the user to make changes to the sample to see whether it changes the outcome. In the example shown in Figure C.1, it shows that changing your gender to female indeed makes sure you would have survived the Titanic crash. Now imagine that the scenario is an access control decision, and this dashboard shows the user what they should change regarding their access request to make

it succeed. For example, the dashboard might show that the *day of week* being a Sunday had strong influence in the request denial. When the user tries again on a Monday it is permitted. Vice versa, this dashboard allows auditors to evaluate why sensitive information was shared, and what should change to prevent that. Developments in the field of XAI may become a piece of the puzzle towards the real-world implementation of RAdAC.

7 Conclusion

Research question 1. Can neural networks be leveraged to train a self-learning model to automate the decision-making in Risk-Adaptive Access Control?

(a.) What are the advantages and disadvantages of neural networks as a technique for a decision model in access control?

(b.) What data is needed to properly train a neural network for this application?

(c.) What network architecture is required?

A proof-of-concept has been demonstrated that leveraged neural networks as a technique to automate decision-making on synthetic access requests in RAdAC. The advantage is that it can re-train with supervision, moving the policy-making complexity towards the data-driven decision model. The disadvantage is that they require complete data. Historic access request data is preferred to train such a model. Implicit risk assessments baked into policy-based decisions, as well as explicit risk scores, are both suitable as access request labels. Both achieve the goal of moving to shift the human task from policy-making towards monitoring. The quantified risk scores for this purpose are a novel approach in this domain and remain to be validated in a real-world setting. A feedforward neural network is most suitable for this application since the problem concerns tabular data.

Research question 2. Can attack trees be exploited to quantify risk scores of access requests?

(a.) What are the advantages and disadvantages of using an attack tree as the risk quantification method for scoring access requests?

(b.) What risks should be determined?

(c.) What attack tree design is required?

A proof-of-concept has been demonstrated that exploited attack trees to quantify the risk scores of access requests. The correlation of risk scores to binary permit/deny decisions has been shown in this work, though it was applied to synthetic data and remains to be validated in a real-world setting. The advantage of attack trees is that they can model interesting complexity through their tree structure. The

disadvantage is that the method may not scale well to large organizations. Useful risks to model can be the third party risk, the impostor risk, the hacker risk, and the resource appetite risk. The tree design requires at least one layer, however, multiple layers could be more interesting. Finally, modeling risk probabilities in attack trees is a workable method to quantify risk scores.

Bibliography

[AB18] Amina Adadi and Mohammed Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160 (cit. on p. 14).

[Adl+13] Aaron Adler, Michael J. Mayhew, Jeffrey Cleveland, Michael Atighetchi, and Rachel Greenstadt. “Using Machine Learning for Behavior-Based Access Control: Scalable Anomaly Detection on TCP Connections and HTTP Requests”. In: *MILCOM 2013 - 2013 IEEE Military Communications Conference*. MILCOM 2013 - 2013 IEEE Military Communications Conference. Nov. 2013, pp. 1880–1887 (cit. on p. 12).

[Ali+15] Amjad Ali, Rehanullah Khan, Irfan Ullah, Adnan Daud Khan, and Abid Munir. “Minutiae Based Automatic Fingerprint Recognition: Machine Learning Approaches”. In: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. Oct. 2015, pp. 1148–1153 (cit. on p. 12).

[Atl+17] Hany Atlam, Ahmed Alenezi, Robert Walters, and Gary Wills. “An Overview of Risk Estimation Techniques in Risk-Based Access Control for the Internet of Things”. In: *2nd International Conference on Internet of Things, Big Data and Security*. In collab. with Hany Atlam, Ahmed Alenezi, Robert Walters, and Gary Wills. INSTICC, Apr. 2017, pp. 254–260 (cit. on p. 12).

[Atl+19] Hany F. Atlam, Robert J. Walters, Gary B. Wills, and Joshua Daniel. “Fuzzy Logic with Expert Judgment to Implement an Adaptive Risk-Based Access Control Model for IoT”. In: *Mobile Networks and Applications* (Jan. 28, 2019) (cit. on p. 11).

[AW19] Hany F. Atlam and Gary B. Wills. “An Efficient Security Risk Estimation Technique for Risk-based Access Control Model for IoT”. In: *Internet of Things* 6 (June 1, 2019), p. 100052 (cit. on p. 11).

[Bre01] Leo Breiman. “Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)”. In: *Statistical Science* 16.3 (Aug. 2001), pp. 199–231 (cit. on p. 14).

[BB07] David W Britton and Ian A Brown. “A SECURITY RISK MEASUREMENT FOR THE RADAC MODEL”. In: (2007), p. 90 (cit. on p. 11).

[Che+07] P. Cheng, P. Rohatgi, C. Keser, et al. “Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control”. In: *2007 IEEE Symposium on Security and Privacy (SP ’07)*. 2007 IEEE Symposium on Security and Privacy (SP ’07). May 2007, pp. 222–230 (cit. on p. 11).

[CWJ21] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. “The Matthews Correlation Coefficient (MCC) Is More Informative Than Cohen’s Kappa and Brier Score in Binary Classification Assessment”. In: *IEEE Access* 9 (2021), pp. 78368–78381 (cit. on p. 44).

[dSan+16] Daniel Ricardo dos Santos, Roberto Marinho, Gustavo Roecker Schmitt, Carla Merkle Westphall, and Carlos Becker Westphall. “A Framework and Risk Assessment Approaches for Risk-Based Access Control in the Cloud”. In: *Journal of Network and Computer Applications* 74 (Oct. 1, 2016), pp. 86–97 (cit. on p. 11).

[FO22] Daniele Faraglia and Other Contributors. *Faker*. Jan. 27, 2022 (cit. on p. 23).

[FF12] B. Farroha and D. Farroha. “Challenges of “Operationalizing” Dynamic System Access Control: Transitioning from ABAC to RAdAC”. In: *2012 IEEE International Systems Conference SysCon 2012*. 2012 IEEE International Systems Conference SysCon 2012. Mar. 2012, pp. 1–7 (cit. on p. 10).

[FK92] David Ferraiolo and Richard Kuhn. “Role-Based Access Controls”. In: *Proceedings of the 15th National Computer Security Conference*. 15th National Computer Security Conference (NCSC); October 13-16, 1992; Baltimore, Maryland, United States. Oct. 13, 1992, pp. 554–563 (cit. on p. 8).

[Fer+16] David F. Ferraiolo, Ramaswamy Chandramouli, Vincent C. Hu, and D. Richard R. Kuhn. *A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications*. NIST SP 800-178. National Institute of Standards and Technology, Oct. 2016, NIST SP 800–178 (cit. on p. 7).

[Gui+18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, et al. “A Survey of Methods for Explaining Black Box Models”. In: *ACM Computing Surveys (CSUR)* 51.5 (Aug. 22, 2018), 93:1–93:42 (cit. on pp. 14–16).

[HHR17] Nurmamat Helil, Azhar Halik, and Kaysar Rahman. “Non-Zero-Sum Cooperative Access Control Game Model with User Trust and Permission Risk”. In: *Applied Mathematics and Computation* 307 (Aug. 15, 2017), pp. 299–310 (cit. on p. 11).

[Hu+14] Vincent C. Hu, David Ferraiolo, Rick Kuhn, et al. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. NIST SP 800-162. National Institute of Standards and Technology, Jan. 2014, NIST SP 800–162 (cit. on p. 9).

[Kha+13] Hemanth Khambhammettu, Sofiene Boulares, Kamel Adi, and Luigi Logrippo. “A Framework for Risk Assessment in Access Control Systems”. In: *Computers & Security*. 27th IFIP International Information Security Conference 39 (Nov. 1, 2013), pp. 86–103 (cit. on p. 11).

[Kor+13] Barbara Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer. “Attack-Defense Trees”. In: *J. Log. Comput.* (2013) (cit. on p. 12).

[Lam74] Butler W. Lampson. “Protection”. In: *SIGOPS Oper. Syst. Rev.* 8.1 (Jan. 1974), pp. 18–24 (cit. on p. 7).

[Lee+17] Brian Lee, Roman Vanickis, Franklin Rogelio, and Paul Jacob. “Situational Awareness Based Risk-adaptable Access Control in Enterprise Networks:” in: *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*. 2nd International Conference on Internet of Things, Big Data and Security. Porto, Portugal: SCITEPRESS - Science and Technology Publications, 2017, pp. 400–405 (cit. on pp. 8–10).

[LY17] Shih-Hsiung Lee and Chu-Sing Yang. “An Intelligent Home Access Control System Using Deep Neural Network”. In: *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*. 2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW). June 2017, pp. 281–282 (cit. on p. 12).

[McG04] Robert W McGraw. “Risk Adaptable Access Control (RAdAC)”. In: (2004), p. 13 (cit. on pp. 10, 22).

[Mil17] Tim Miller. “Explanation in Artificial Intelligence: Insights from the Social Sciences”. In: (June 22, 2017) (cit. on p. 16).

[MCR08] Ian Molloy, Pau-Chen Cheng, and Pankaj Rohatgi. “Trading in Risk: Using Markets to Improve Access Control”. In: *Proceedings of the 2008 Workshop on New Security Paradigms - NSPW ’08*. The 2008 Workshop. Lake Tahoe, California, USA: ACM Press, 2008, p. 107 (cit. on p. 11).

[Mol+12] Ian Molloy, Luke Dickens, Charles Morisset, et al. “Risk-Based Security Decisions under Uncertainty”. In: *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*. CODASPY ’12. San Antonio, Texas, USA: Association for Computing Machinery, Feb. 7, 2012, pp. 157–168 (cit. on p. 13).

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA). KDD ’16. New York, NY, USA: ACM, 2016, pp. 1135–1144 (cit. on pp. 5, 16).

[Sch99] Bruce Schneier. “Attack Trees”. In: *Dr. Dobb’s journal* 24.12 (1999), pp. 21–29 (cit. on pp. 3, 12).

[Shi07] Robert W. Shirey. *Internet Security Glossary, Version 2*. RFC Editor, Aug. 2007 (cit. on p. 8).

[Śmi+18] Marek Śmieja, Łukasz Struski, Jacek Tabor, Bartosz Zieliński, and Przemysław Spurek. “Processing of Missing Data by Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018 (cit. on p. 53).

[SS20] Kriti Srivastava and Narendra Shekokar. “Machine Learning Based Risk-Adaptive Access Control System to Identify Genuineness of the Requester”. In: *Modern Approaches in Machine Learning and Cognitive Science: A Walk-through: Latest Trends in AI*. Ed. by Vinit Kumar Gunjan, Jacek M. Zurada, Balasubramanian Raman, and G. R. Gangadharan. Studies in Computational Intelligence. Cham: Springer International Publishing, 2020, pp. 129–143 (cit. on p. 13).

[SB02] Valery Starovoitov and D Bryliuk. “Access Control by Face Recognition Using Neural Networks”. In: Sept. 1, 2002 (cit. on p. 12).

[Uni16] European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA Relevance)*. 32016R0679. May 4, 2016 (cit. on p. 14).

[vLFM] Michael van Lent, William Fisher, and Michael Mancuso. “An Explainable Artificial Intelligence System for Small-unit Tactical Behavior”. In: (), p. 8 (cit. on p. 14).

[ZPŠ14] Marijana Zekić-Sušac, Sanja Pfeifer, and Nataša Šarlija. “A Comparison of Machine Learning Methods in a High-Dimensional Classification Problem”. In: *Business Systems Research Journal* 5.3 (Sept. 1, 2014), pp. 82–96 (cit. on p. 12).

[Zho+19] Lu Zhou, Chunhua Su, Zhen Li, Zhe Liu, and Gerhard P. Hancke. “Automatic Fine-Grained Access Control in SCADA by Machine Learning”. In: *Future Generation Computer Systems* 93 (Apr. 1, 2019), pp. 548–559 (cit. on p. 13).

Webpages

[@Ama13] Amazon. *Amazon.Com - Employee Access Challenge*. 2013. URL: `https://kaggle.com/c/amazon-employee-access-challenge` (visited on Jan. 11, 2022) (cit. on pp. 19, 51).

[@Hou21] Houtsma. *General · RamonH93/Radac_model*. GitHub. 2021. URL: `https://github.com/RamonH93/radac_model` (visited on Mar. 14, 2022) (cit. on pp. 22, 45, 73).

[@Num20] NumPy. *NumPy*. 2020. URL: `https://numpy.org/` (visited on Jan. 27, 2022) (cit. on pp. 23, 45).

[@Oas20] Oasis-open. *eXtensible Access Control Markup Language (XACML) Version 3.0*. 2020. URL: `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html#_Toc325047268` (visited on Jan. 24, 2022) (cit. on pp. 21, 52).

[@Pan20] Pandas. *Pandas - Python Data Analysis Library*. 2020. URL: https://pandas.pydata.org/ (visited on Jan. 27, 2022) (cit. on pp. 22, 45).

[@Pyt18] Python. *Welcome to Python.Org*. Python.org. 2018. URL: https://www.python.org/ (visited on Dec. 31, 2018) (cit. on p. 22).

[@Roo20] Marijke Roosen. *What SyRi Can Teach Us about Technical Solutions for Societal Challenges*. Global Data Justice. Feb. 20, 2020. URL: http://globaldatajustice.org/2020-02-20-roosen-syri/ (visited on Mar. 11, 2020) (cit. on p. 36).

[@Rud19] Cynthia Rudin. *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. Sept. 21, 2019. arXiv: 1811.10154 [cs, stat]. URL: http://arxiv.org/abs/1811.10154 (visited on Feb. 10, 2020) (cit. on p. 16).

[@Ten20] TensorFlow. *TensorFlow*. TensorFlow. 2020. URL: https://www.tensorflow.org/ (visited on Mar. 13, 2022) (cit. on p. 46).

[@Tit12] Titanic. *Titanic - Machine Learning from Disaster*. 2012. URL: https://kaggle.com/c/titanic (visited on Mar. 14, 2022) (cit. on p. 53).

A Decision Models Training and Performances

Binary Classifier

Training Progression

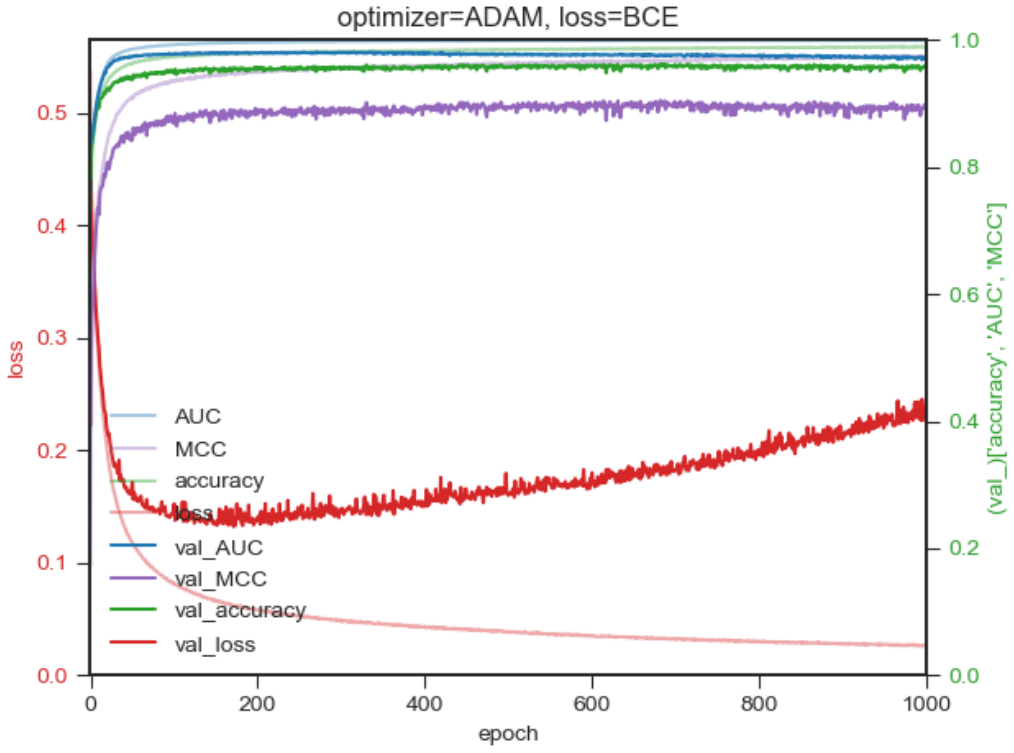


Fig. A.1.: Training progression of the binary classifier.

ROC Curve

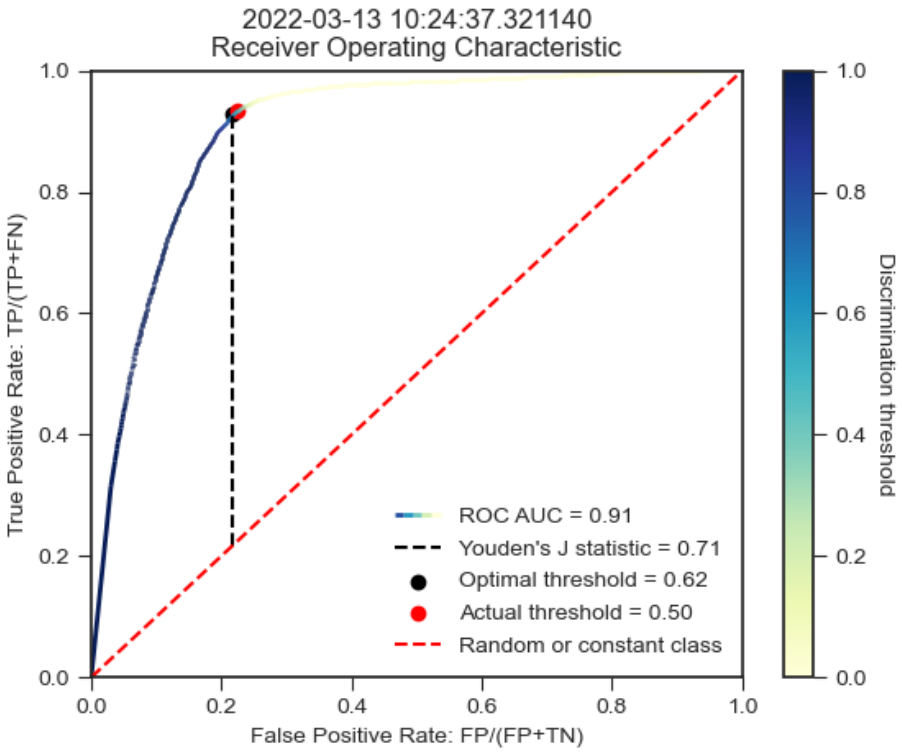


Fig. A.2.: ROC curve of the binary classifier.

Confusion Matrix

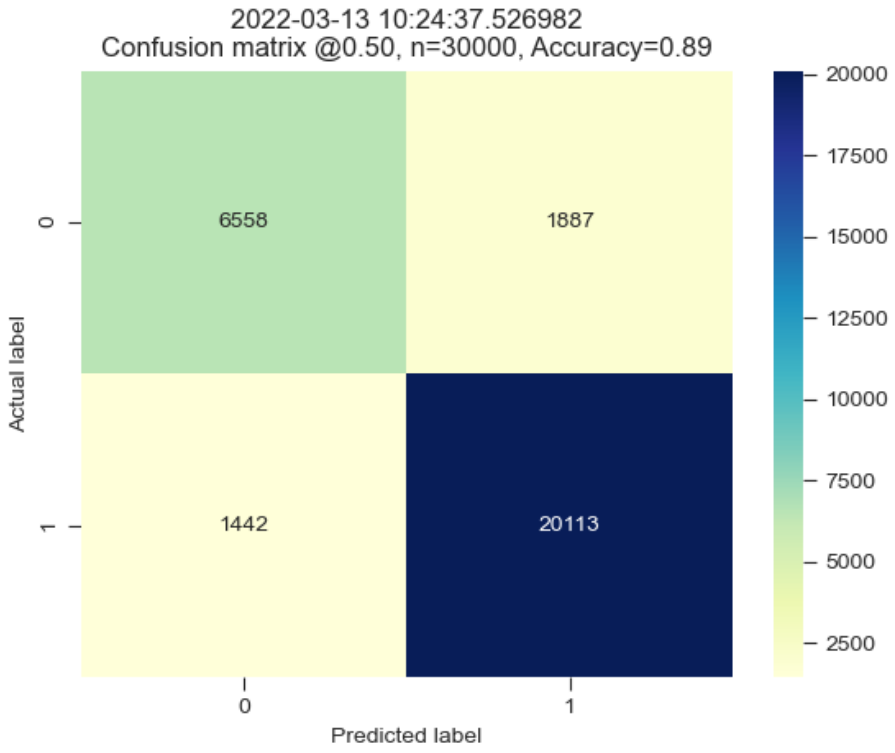


Fig. A.3.: Confusion matrix of the binary classifier predictions on the test set.

Multi-class Classifier

Training Progression

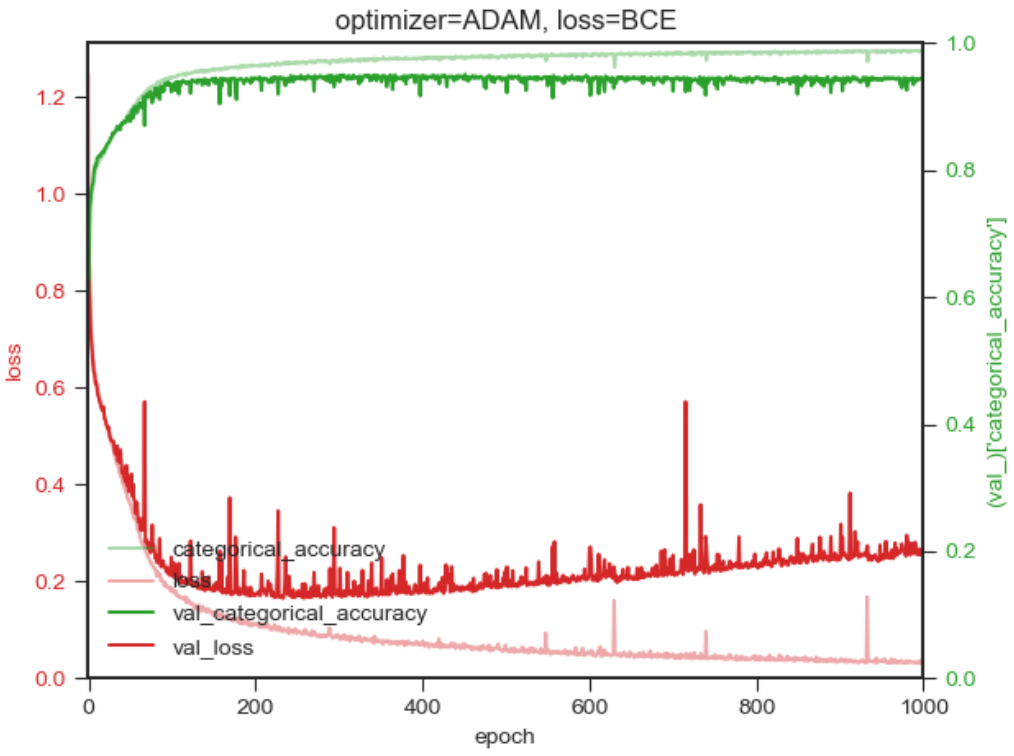


Fig. A.4.: Training progression of the multi-class classifier.

Confusion Matrix

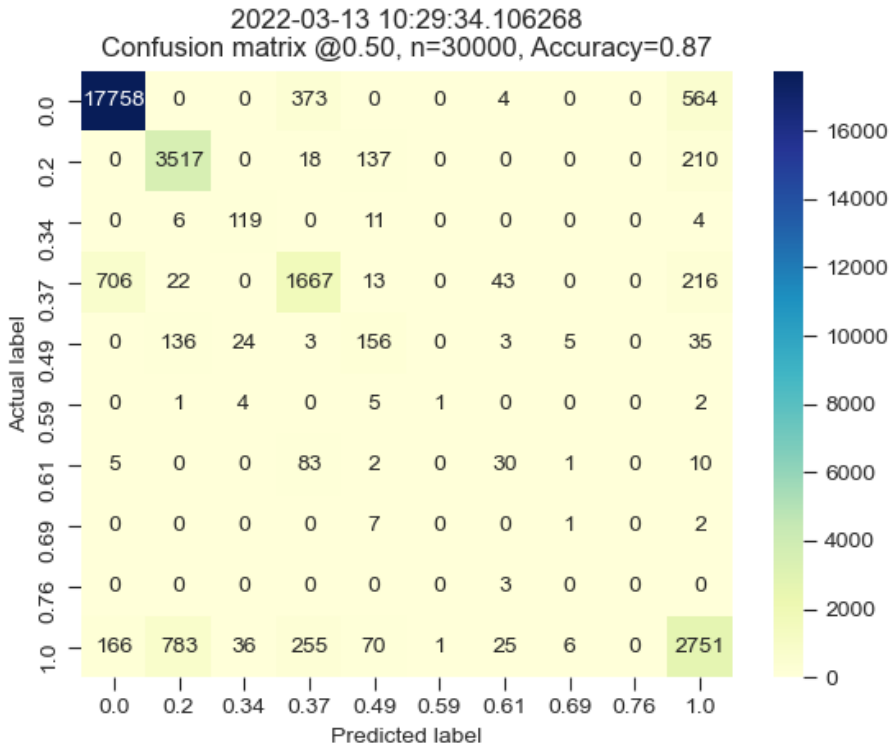


Fig. A.5.: Confusion matrix of the multi-class classifier predictions on the test set.

Regression Model

Training Progression

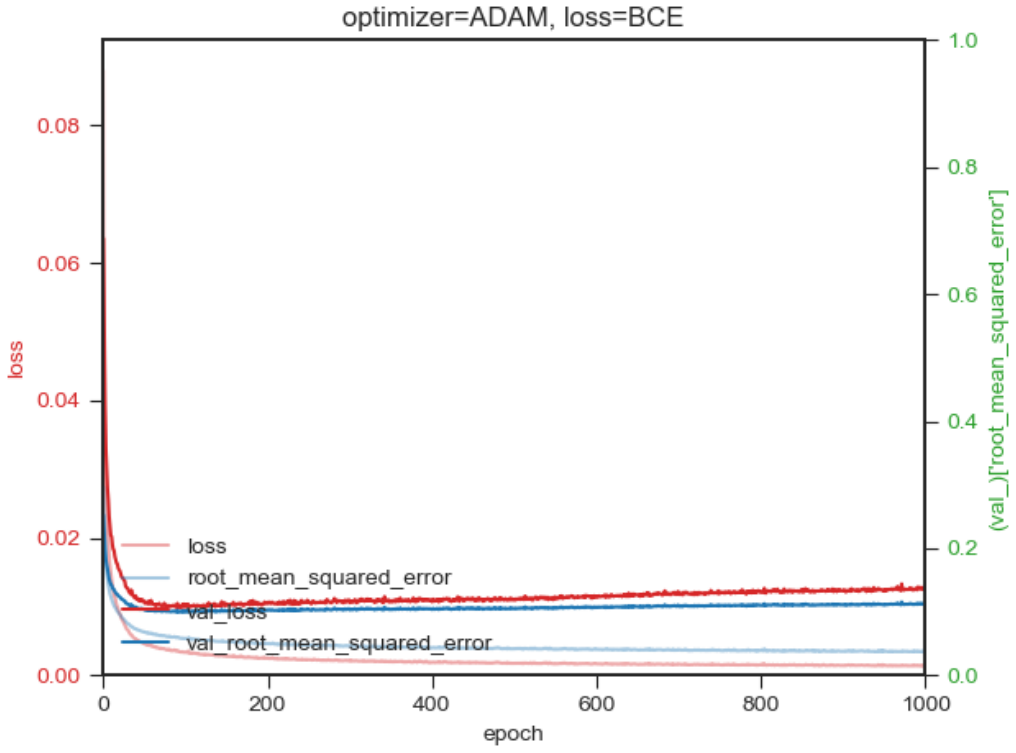


Fig. A.6.: Training progression of the regression model.

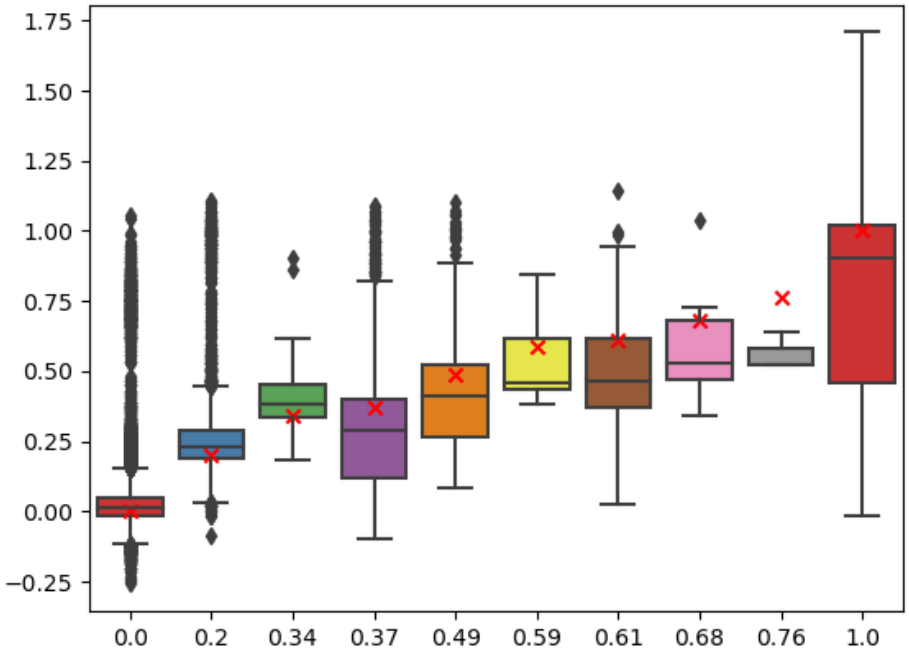


Fig. A.7.: Box plots of risk score predictions of the regression model. The x-axis shows the risk score the model tries to predict. The red 'x's mark the value of the true risk scores on the y-axis.

B

Binary Decision Model Performances on Requests denied by Policies

Figure B.1 and Figures B.2a to B.2g show the confusion matrices of the binary decision model predictions on denied requests, specified per data labeling policy, sorted by performance. These results show which policies the decision model learned well from the training data, and which it didn't. '@0.5' means that the decision threshold is 0.5: the rounding threshold above which the prediction is rounded to 1 and below is rounded to 0. n is the number of access requests that were denied by the policy, thereby labeled 0.

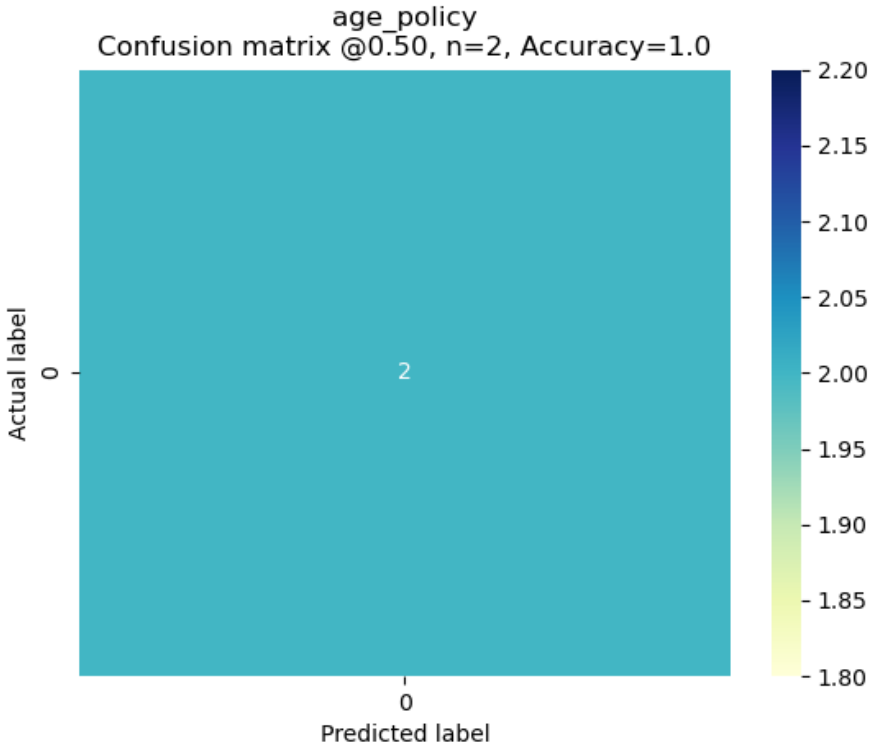
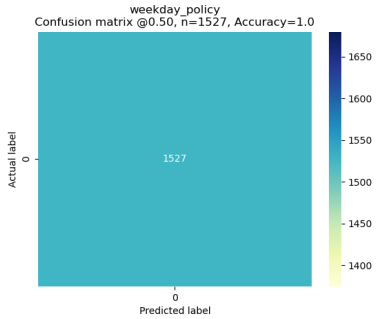
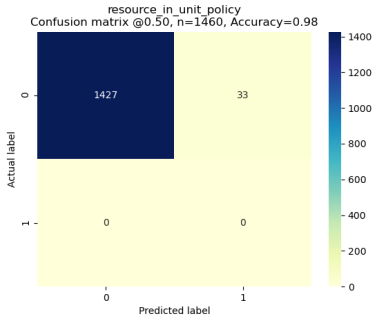


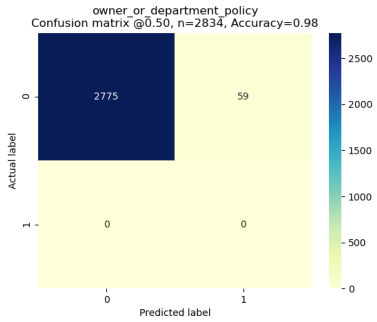
Fig. B.1.: Prediction of the binary decision model on the two minors that tried to gain access.



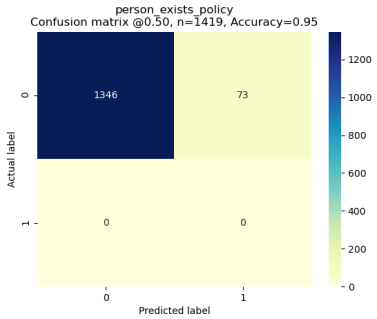
(a) Office Weekday Policy.



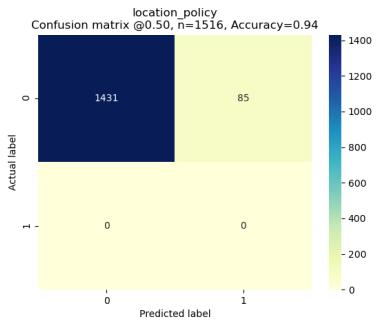
(b) Resource in same Unit as Subject Policy



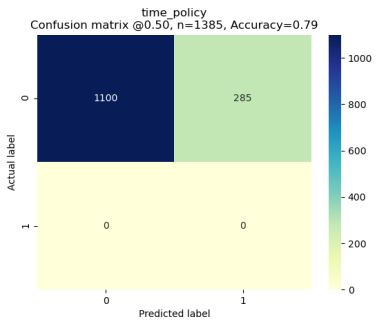
(c) Resource Owner or Same Department Policy.



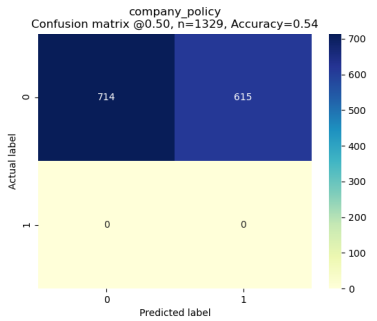
(d) Subject in Employees Table Policy.



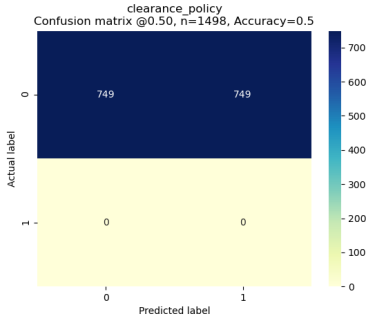
(e) Location Policy.



(f) Time within Office Hours Policy.



(g) Subject from Company Policy.



(h) Clearance Level Sufficient for Confidentiality Level Policy.

Fig. B.2.: Predictions of the binary decision model on denied requests, per policy. Sorted based on performance.

Prediction Explorer

C

All code for the prediction explorer on Page 74 is available at [@Hou21].

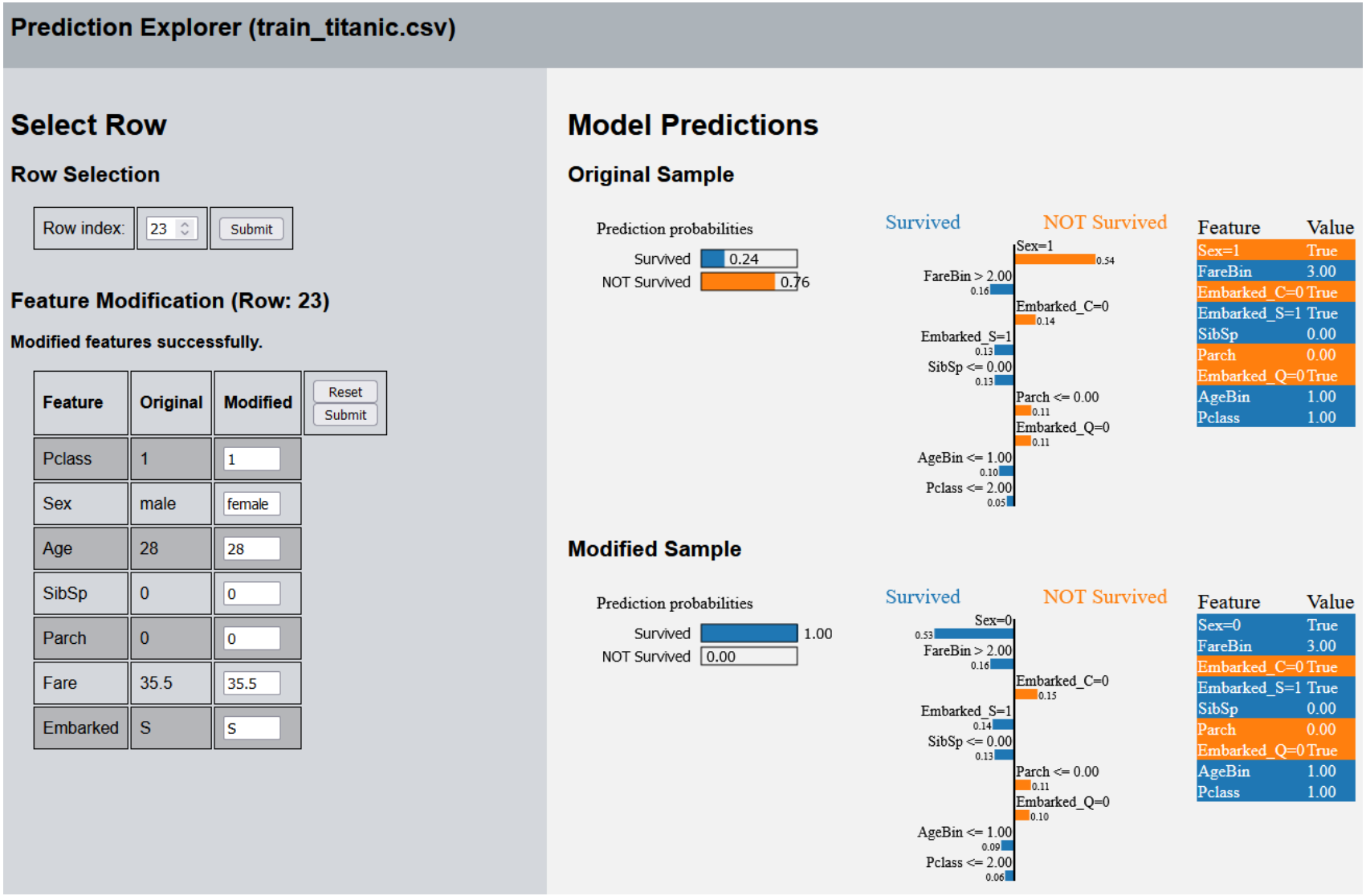


Fig. C.1.: Demo of prediction explorer with titanic dataset.