



University of
Zurich^{UZH}

SecRiskAI: A Machine Learning-based Tool for Cybersecurity Risk Assessment

Erion Sula
Zürich, Switzerland
Student ID: 15-718-349

Supervisor: Muriel Franco, Dr. Alberto Huertas
Date of Submission: August 1, 2021

Zusammenfassung

In den letzten Jahrzehnten hat die Anzahl und der Schweregrad von Cyberangriffen weltweit zugenommen und beeinträchtigen mit steigender Tendenz Netzwerke, Systeme, Unternehmen und Kunden. Der Schaden, den sie verursachen, nimmt jedes Jahr exponentiell zu. In jüngster Zeit haben Forscher und Unternehmen “Frameworks” für die Abschätzung von Cybersicherheitsrisiken entwickelt, um diese Risiken zu erkennen, einzuschätzen und zu priorisieren, mit dem Ziel, ihre Auswirkungen zu minimieren. Bei herkömmlichen Ansätzen ist es jedoch oft schwierig, Indikatoren für unvorhersehbare Cyberrisiken zu finden, was die Möglichkeit einer genauen Risikobewertung einschränkt. Vor diesem Hintergrund wird in dieser Arbeit die Anwendbarkeit des maschinellen Lernens auf die Bewertung von Cyberrisiken untersucht. Zu diesem Zweck wurden verschiedene Algorithmen des maschinellen Lernens auf synthetischen Datensätzen unterschiedlicher Grösse trainiert, getestet und bewertet. Darüber hinaus ist die aktuelle Version des Prototyps auch in der Lage, den Benutzer durch den Entscheidungsprozess für Cybersicherheitsinvestitionen zu unterstützen, indem MENTOR, ein System zur Empfehlung von Schutzdiensten, integriert wird. Um die Machbarkeit der vorgeschlagenen Lösung zu demonstrieren, wurden sowohl eine quantitative als auch eine qualitative Bewertung durchgeführt. Die quantitative Bewertung hat gezeigt, dass der Prototyp in der Lage ist, sehr genaue Ergebnisse zu erzielen. Andererseits hat die qualitative Bewertung die Wirksamkeit und Zuverlässigkeit der Lösung bewiesen.

Abstract

Over the last decades, the number and severity of cyber-attacks worldwide have grown, and are increasingly affecting networks, systems, businesses, and customers. The harm they cause is rising exponentially every year. Recently, researchers and companies have developed frameworks for assessing cybersecurity risk in order to identify, estimate and prioritize cyber-risks and minimize their impact. However, traditional approaches often struggle to find indicators of unpredictable cyber-risks, thus limiting the ability to perform accurate risk assessments. Taking this into account, this thesis explores the applicability of machine learning on cybersecurity risk assessment. For this purpose, various machine learning algorithms were trained, tested and evaluated on synthetic datasets of different sizes. Besides that, the current version of the prototype also capable of supporting the user through the cybersecurity investment decision process, by integrating MENTOR, a protection service recommender system. To demonstrate the feasibility of the proposed solution, a quantitative as well as a qualitative evaluation have been conducted. The quantitative evaluation showed that the prototype is able to achieve very accurate results. On the other hand, the qualitative evaluation proved the effectiveness and reliability of the solution.

Acknowledgments

This master thesis would not have been possible without the support of many people. Firstly, I would like to express my sincere gratitude to my supervisor, Muriel Franco, for his feedback, the very interesting discussions and guidance throughout this thesis.

I would also like to thank Prof. Dr. Burkhard Stiller and the members of the Communication Systems Research Group for giving me the opportunity to work on such an interesting topic.

Special thanks also to Esra Oksal for proofreading my work.

Last but not least I would like to thank my family and friends for their constant support and encouragement.

Contents

Zusammenfassung	i
Abstract	iii
Acknowledgments	v
1 Introduction	1
1.1 Motivation	1
1.2 Description of Work	2
1.3 Thesis Outline	2
2 Background	5
2.1 Risk Management	5
2.2 Artificial Intelligence (AI)	6
2.2.1 Machine Learning (ML)	8
2.2.2 Deep Learning (DL)	13
3 Related Work	17
4 The SecRiskAI Approach	23
4.1 Risk Assessment Workflow	24
4.1.1 Data Gathering	25
4.1.2 Data Processing	28
4.1.3 Multi-Class Classification Algorithms	29
4.1.4 Training, Cross Validation & Testing	34
4.2 MENTOR's API Integration	34

5	Prototype and Implementation	37
5.1	Frontend	37
5.1.1	Web-based Interface	41
5.2	Backend	43
5.2.1	Middleware	43
5.2.2	ML Classifier	46
5.3	ML Workflow	50
6	Quantitative Evaluation	55
7	Qualitative Evaluation	61
7.1	Case Study #1 - DDoS Attack	61
7.2	Case Study #2 - Recommendation of Protections	63
7.3	Case Study #3 - Phishing Scenario	66
7.4	Discussion and Limitations	67
8	Summary, Conclusions, and Future Work	69
	Bibliography	71
	Abbreviations	81
	List of Figures	81
	List of Tables	84
	Listings	85
A	Installation Guidelines	89
B	Contents of the CD	91

Chapter 1

Introduction

Cybersecurity risk assessment is the process of understanding, managing and evaluating cyber-threats. Due to increasingly complex and diverse cyberattacks, which are expected to cause 10.5 trillion US\$ annual loss by 2025 [1], the activity of assessing cyber risks has become a crucial part of any organization's risk management strategy. However, maintaining an effective cyber risk management plan is a complex and challenging task, as organizations struggle to cope with the technological advancements of cyber-adversaries and are often confronted with an increasing number of exploitable vulnerabilities [2]. According to a recent survey, only 16% of executives actually claim that their companies are well prepared to face cyber risks even though 75% of the interviewed experts consider cybersecurity to be a top priority [3]. Additionally, a report published by IBM on the cost of data breaches [4] based on more than 500 companies around the world highlights the importance of investing in effective governance, risk management and compliance programs.

1.1 Motivation

As businesses strengthen their digital dependency, they also become more vulnerable to cyber-threats. Nowadays, the biggest challenge that businesses in any industries face is keeping sensitive data private and proactively protect the infrastructure against cyber-attacks, such as Ransomware, Distributed Denial-of-Service (DDoS) attacks, etc., which known to be the main cause for both huge financial and reputational damages. Therefore, besides the need for innovation, decision-makers in cybersecurity (*e.g.*, network operator, company owner or expert team) have to be able to implement robust cybersecurity mechanisms and risk assessment frameworks to ensure a proper security level and prevent intrusions of any kind.

Despite many risk assessment standards available (*e.g.*, ISO 31000 [5], TOGAF [6], NIST SP 800-30 [7]) and multi-sector assessment frameworks proposed [8], organizations still find this activity very challenging and are often confronted with a huge volume of unstructured data, essential for finding indicators of unpredictable risks [9]. In this case,

traditional techniques often struggle at providing valuable insights and the ability of performing real-time risk assessment is limited. Moreover, there is a need for continuous risk assessment and monitoring strategy for Key Risk Indicators (KRIs) in order to identify and estimate the likelihood of unpredictable threats.

Recent studies on possible applications of Artificial Intelligence (AI) and Machine Learning (ML) algorithms have highlighted their ability to process large amounts of structured/unstructured data, extract valuable patterns, learn from historically collected records and make accurate predictions ([10, 11]). Thus, the main goal of this work is to explore the potential of ML algorithms in the field of cybersecurity risk assessment.

1.2 Description of Work

The goal of this master thesis is to design and develop a ML-based tool that provides mechanisms, where decision-makers can configure, calculate and analyze important aspects related to cybersecurity risk assessment (*e.g.*, the understanding of possible impacts of cyberattacks in the economic aspect of a business). Different metrics and information [12, 16] have to be considered in order to provide a ML model that estimates risks and provides insights to guide an adequate planning in cybersecurity [15]. Besides that, an intuitive and user-friendly interface has been developed to provide an overview of the business being analyzed and the predicted risks. Additionally, the tool has been integrated with MENTOR's recommendation API [13, 14], in order to provide a proactive recommendation of protection services based on the organization's profile and predicted risks.

An evaluation of the performance, accuracy, and limitations of the tool and the ML model is also provided, including tests to evaluate specific ML performance metrics (*e.g.*, F1 score, precision, recall). In addition, case studies have been conducted, taking into account real-world scenarios and information from different organizations. Once the interface and the case studies were defined, an evaluation was conducted to show evidences of the feasibility and accuracy of the proposed solution.

1.3 Thesis Outline

Chapter 2 comprises of two main parts. The first one provides an introduction to risk management and assessment in general. In the second section, the concept of Artificial Intelligence is explained.

Chapter 3 discusses related work in the field of risk assessment and ML.

Chapter 4 presents the approach followed in this thesis in a non-technical way. It starts by providing an architecture overview as well as a short explanation of every layer involved in the system. Next, the ML workflow designed in this work and MENTOR's API integration are presented.

Chapter 5 gives a technical overview of the approach followed in this thesis. In particular, the technical implementation of each component is presented, including the design and development of various ML models.

Chapter 6 proposes a quantitative evaluation of the ML-based tool for cybersecurity risk assessment and provides a performance comparison of the developed ML models.

In Chapter 7 a qualitative evaluation will be presented where various use cases will be proposed to discuss possible challenges/limitations of the prototype.

Finally, Chapter 8 summarizes and concludes the thesis and provides an outlook for future work in order to improve the solution.

Chapter 2

Background

This chapter provides the necessary background knowledge for the concepts and methodologies used in this work. Thus, the most important aspects of risk management is introduced as well as different AI concepts. Besides that, the ML concepts, methods, and techniques relevant for this work are presented in detail.

2.1 Risk Management

In the literature, the term “risk” is generally used to indicate a possibility of loss and/or damage. It usually involves some degree of uncertainty and the resulting outcome is challenging to predict. Depending on the context, various types of risk can be found, such as: business risk, economic risk, and safety risk [17]. On a quantitative level, a general risk (R) can be further expressed as the following triplet:

$$R = \langle s, p, c \rangle$$

This quantitative definition indicates that a risk (R) is generally a combination of a scenario (s), *i.e.*, what can actually go wrong, the probability (p) it will have and the severity of the consequence (c), *i.e.*, level of damage caused [17]. In the field of Information Technology (IT), a risk, also called cyber-risk, is usually described by a threat which is able to expose system’s vulnerability causing economic and reputation losses. A threat in this case can be represented by cybersecurity attacks, such as malware, ransomware and phishing attacks [18]. Another study [20] proposed a new way of thinking about risk in general by extending the definition with a knowledge dimension (k). More specifically, in different scenarios with equal probability, knowledge has been demonstrated to be an important factor for supporting the decision-making process.

Risk assessment is the necessary process of identifying, analyzing and evaluating risks. In the context of cybersecurity, the assessment process focuses on cyber-risks. Risk assessment is a key stage of the whole risk management lifecycle and in practice it includes

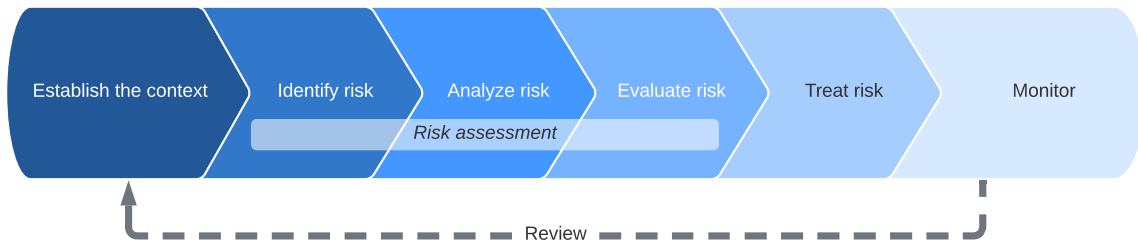


Figure 2.1: Risk management framework [19]

three main tasks: risk identification, analysis, and evaluation. Figure 2.1 gives a visual representation of a general risk management framework.

In essence, the risk management is an ongoing process of assessing, treating and monitoring risks. In the first phase, the scope for the whole risk management process and the criteria against which the risks will be assessed are set. Afterwards, the assessment phase is initiated. This stage involves the identification of possible risks followed by a comprehensive analysis and evaluation. It is important to note that putting insufficient effort in the first stage of the risk assessment process may cause potential risks being omitted from further analysis and therefore resulting in unpredictable outcomes. After being prioritized, the risks are treated based on the type, nature and priority. However, developing a good and effective treatment plan has been proven to be difficult. In such cases, having large collection of past projects along with the corresponding risks history can help developing more proactive treatment strategies [19].

Lastly, an effective risk management strategy also involves regular checking and surveillance for potential threats. Results should thus be recorded, reported both internally (*e.g.*, team, organisation), externally (*e.g.*, stakeholders) and reviewed [19].

2.2 Artificial Intelligence (AI)

In the Computer Science field, the notion of AI is generally understood to indicate the ability of any machine, ranging from common computers to robots, to perform demanding tasks which usually require human-like intelligence [21]. In this regard, AI is often associated with the process of developing systems able to make decisions, solve generalized problems, learn from past experiences and even recognize object familiar to humans [22]. In recent years, there has been growing interests in researching possible applications of AI. Nowadays, AI is in charge of supporting, or in some cases even replacing, common systems found in many different sectors, such as Healthcare, e-Commerce, Supply Chain, etc. [24]. In theory, there are four types of AI [25]:

- **Reactive Machines.** These systems are considered to be the oldest type of AI and are only able to perform basic operations. More specifically, they lack memory-based functionality and therefore do not possess the ability to learn from past interactions

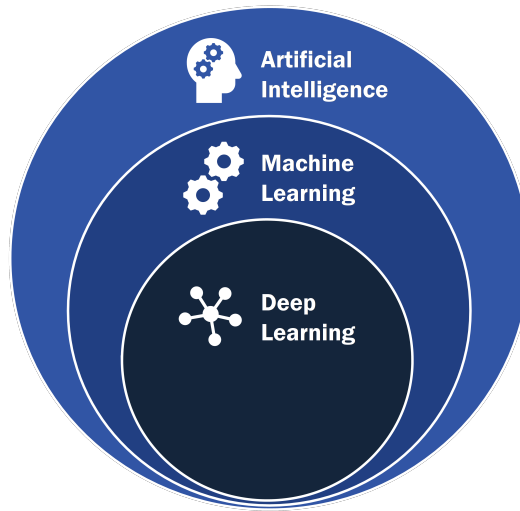


Figure 2.2: Artificial Intelligence overview [23]

or experiences. Moreover, this type of AI can only respond to a limited set or combination of inputs.

- **Limited Memory Machines.** These machines tend to have the same ability as reactive machines, but are also capable of learning from historical data. After the training phase, such machines are also capable of solving future problems by making accurate predictions. In practice, nearly all existing applications of AI come under this category.
- **Theory of Mind.** Systems that will reach this level of AI will be able to discern the emotions, needs, and thought processes of the humans it's interacting with. At the moment of writing this thesis, such systems are only in the beginning phase and can exclusively be found in self-driving cars.
- **Self-aware AI.** At this level, these types of AI systems will have become so advanced that it develops self-awareness, that is, a system which is capable of possessing emotions, needs, beliefs, and potentially desires of its own.

In the literature, AI is also considered to be an umbrella term for other disciplines such as Machine learning (ML) and Deep Learning (DL). Figure 2.2 gives an overview of the whole AI spectrum. In practice, both AI and ML are often used interchangeably. However, there are key differences between these two concepts. In broad terms, AI is defined as the science of designing and engineering machines, capable of possessing the characteristics of human intelligence. On the other hand, ML is a possible application of AI based around the idea that machines are able to learn from large amount of data. Lastly, the main difference between DL and ML relies in the way each algorithm learns [26].

2.2.1 Machine Learning (ML)

ML is a subset of AI mainly focused on developing models having the ability to learn based on previously collected data and to improve their accuracy without being explicitly programmed. In ML, algorithms are essentially able to process massive amount of data to find relevant patterns used for the training phase. After an initial training phase, when confronted with new input data, a ML model is able to make accurate predictions [27]. In general, every ML workflow can be summarized in five main steps:

1. Problem Definition & Data Collection

Before any data collection takes place, it is very important to understand the problem at hand, characterizing it and gathering all the required domain-specific knowledge and inputs from the experts. Only when the problem is clear, well-defined and understood, the data collection process actually starts. In order to fully understand a problem, experts should focus on identifying (in-)dependent variables. Independent variables, also known as *Features*, are considered to be inputs and/or stimuli for the ML Model whereas dependent variables are usually associated with the output, which in turn depends on the input, hence the name “dependent variables” [29].

After a successful problem definition, all the relevant and comprehensive data needs to be collected. The type of data needs to be defined first. Data can be categorized in 4 types: *numerical*, *categorical*, *time series* and *text*. Numerical data, also called *quantitative* data, is usually associated with numbers, which in turn can be continuous (*e.g.*, weight, salary, temperature) or discrete (*e.g.*, units of a product sold). Categorical data on the other hand are used to represent a set of characteristics. Educational level is one possible example of categorical data, as it can only have three values: “beginner”, “intermediate”, “advanced”. Time series data is characterized by collection of observations obtained through repeated measurements. Possible examples range from weather records, economic indicators and performance metrics over time [30, 31].

In addition to that, depending on the complexity of the problem and the chosen ML algorithm, the volume of data needs to be adjusted accordingly, as it may have a direct impact on the performance/accuracy of the algorithm. The size of the dataset has indicated to play a significant role during model development. Some studies found that model constructed on large datasets usually show an higher accuracy and lower error [28, 29].

2. Data Preparation

This phase, also called data pre-processing, is usually known for taking up to 80% of data scientist’s time [32]. Once the data has been successfully collected, it requires further preparation in order to be used for the training model. In the field of ML, there are several techniques to ensure that a dataset meets the requirement for an algorithm. Data scientists are usually confronted with problems such as inconsistent records, missing and/or incorrect values and even outliers that need to be handled correctly. This means, that when integrating data from different sources, each variable must be formatted correctly, and in case of missing values, the affected records can either be deleted or replaced with dummy/average values. Lastly, the

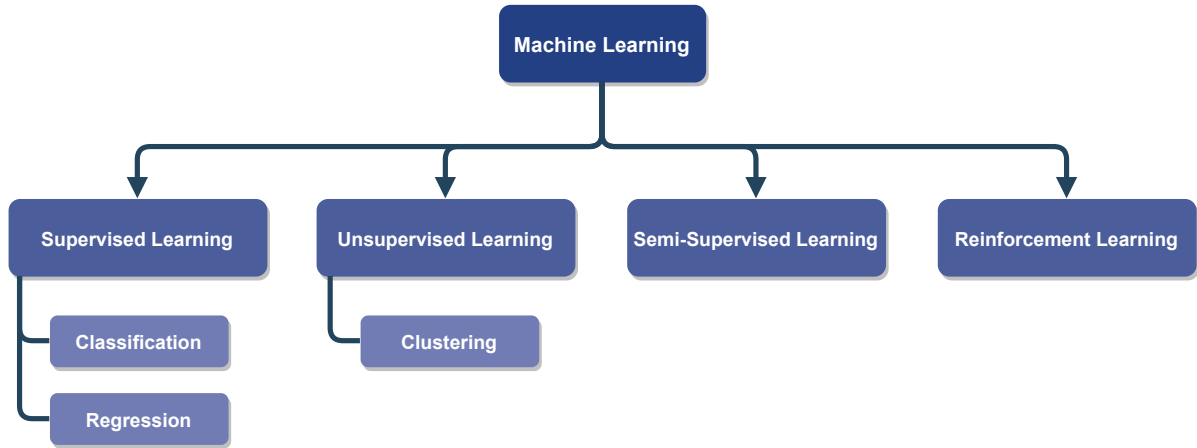


Figure 2.3: Machine Learning types (based on [34])

entire dataset is split in training and testing data, usually following the 80/20 train-test split rule (80% train data and 20% test data) depending on the original size of the dataset [33, 36].

3. Algorithm Selection

The third step consists of the choosing right algorithm for creating the model. In ML, there are many approaches aiming to support finding the most suitable solution for the defined problem. Depending on the nature of the input “signal” and the corresponding outcome, ML approaches are traditionally divided into four main categories as shown in Figure 2.3 [34].

Supervised Learning (SL). Any algorithm that falls under this category requires to be trained with labeled data. That means, the correct output of every record in the training dataset is usually known a priori. This way, as the dataset is fed to the algorithm, the model is able to learn and produce an inferred function, which is then used for making new predictions. Supervised learning uses two main techniques for developing predictive models: *Classification* and *Regression*. Classification algorithms are able to accurately assign categories, or also called classes, to test data. When confronted with previously unseen data, this type of algorithms try to label it either into two distinct classes (*i.e.*, *Binary Classification*) or more (*i.e.*, *Multi-Class Classification*) [35, 37]. On the other hand, regression algorithms are able to predict continuous values based on the input data. More specifically, such type of algorithms aim to estimate a mapping function based on both input and output variables. This technique is commonly used in financial trading but it can also find application in other areas such as: electricity, flights, products and real estate price forecasting [38].

Unsupervised Learning (UL). These kind of algorithms focus on discovering hidden patterns in the uncategorized, unlabeled data without the need of human intervention. Unsupervised learning algorithms are usually more complex and the models require huge amounts of data to be able to extract useful properties of the dataset and to group similar records in similar classes. This technique, also known as *Clustering*, is generally preferred over supervised learning algorithms when the process

of manually generating labels is considered to be laborious and expensive. This is usually the case in medical imaging where a lot of images are collected and labeling each one of them becomes an unfeasible task to accomplish [40]. Other successful applications can be found in different areas, such as: speech recognition [39], anomaly detection [41, 42] and many other fields [43].

Semi-Supervised Learning (SSL). As the name suggests, semi-supervised learning algorithms are able to deal with both labeled and unlabeled datasets. The goal of this approach is to design and develop algorithms having the ability to combine different sources of data and to analyze the effect on the learning behaviour. Semi-supervised learning algorithms are known to be extensions of either supervised or unsupervised learning strategies, thus making little adjustments to the inherited techniques [44].

Reinforcement Learning (RL). RL can be viewed as an approach which falls between SL and UL and is usually referred to as the science of decision-making. This technique introduces three main concepts: *Agent*, *Environment* and *Reward*. Initially, an agent (*e.g.*, algorithm, bot) interacts with an unknown environment and by performing actions and obtaining some kind of feedback. An action causes the current situation to change and as a consequence, the agent receives either a reward or penalty. The main goal of this kind of learning is being able to take actions in a previously unseen situation, maximizing the total rewards. Finding the optimal strategy usually takes a lot of attempts and failures [45]. There are different works highlighting the effectiveness of RL algorithms in different use cases, such as: managing resources in computer resources [46], designing traffic light control to solve congestion problems [47] and robotics [48].

4. Model Training & Validation

The process of training a ML model involves the experts to provide the previously collected and processed data to the chosen learning algorithm. In essence, the term ML model refers to the artifact which is created after the training phase. This step is also known for taking a lot of time in a typical ML workflow but nevertheless is essential for building a strong foundation for an accurate model development [49]. When training a model, it is very important to make sure that the training dataset is a subset of the initial dataset, since training a model on the whole dataset would probably lead to overfitting. More specifically, the model will pick up the detail and noises of the dataset to an extent where it negatively impacts its performance when confronted with new input data [50].

In addition to the train-test split strategy, in some scenarios it is also recommended to further extract from the train dataset a so-called validation dataset. This data is often used to provide an unbiased evaluation of the trained model while tuning the model specific properties, also known as *hyperparameters*. These parameters are essentially responsible for controlling the entire learning process and have a significant impact on the model performance. Therefore, it is extremely important to find an optimal combination of hyperparameters that provides the best results and minimizes overfitting. In ML, there are multiple ways of trying different combinations of such parameters and then evaluating the resulting configuration [51]. The most popular approaches among them are:

Grid Search. This technique is used to perform an exhaustive search to compute the optimum values of the hyperparameters. Figure 2.4a shows a visual representation of grid search with two hyperparameters, x_1 and x_2 , each with ten possible values. A total of a hundred different combinations will be evaluated and compared. As a result, the blue contours present the optimal hyperparameter combination, that is, the values of x_1 and x_2 with the highest accuracy whereas the red contours indicate values for which the performance is considered poor. One major drawback of Grid Search is that with large number of parameters, computing and evaluating every possible combination becomes a very inefficient and time-consuming task [52].

Random Search. An alternative to grid search is random search. As the name suggests, this method tries random combinations of hyperparameters in a pre-defined domain for building a model with highest accuracy. While grid search executes an exhaustive search on every possible combination, in this approach the number of search iterations is set based on available time and resources. Figure 2.4b gives a visual representation of this method, using the same parameters x_1 and x_2 as for grid search. Similarly, a hundred randomly chosen combinations are evaluated. This is also represented by the green marks which indicate that more individual values for each parameter are considered [52].

Bayesian Optimization. This method, in contrast to grid and random search, is able to keep track of the past combinations and the corresponding evaluations to form a probabilistic model based on which the parameters for the next iteration are chosen [53]. As shown in Figure 2.4c, the bayesian optimization process initially explores the space of potential parameter combination and afterwards is able to smartly select the next combination based on previous results. Various studies on this method have shown that using this approach leads to obtaining better results while maintaining a lower number of iterations when compared to the grid/random search [54].

In combination with those hyperparameter optimization techniques, an approach called *Cross-Validation (CV)* is used to evaluate the performance of the model after each iteration. The most common technique is *K-Fold CV*. During this procedure the train dataset is randomly split into independent k -folds. The parameter “ k ” can be any arbitrary number, however good standard values are five or ten. During each iteration, $k-1$ folds are used for training the model and the remaining fold is used for performance evaluation. The whole CV process is repeated “ k ” times, resulting in “ k ” number of performance estimations (PE). Lastly, the overall model performance is defined as the mean of the previously computed “ k ” PEs. Figure 2.5 illustrates the entire process for K equal to five [55].

5. Model Testing

After a successful training and validation, the final model is subjected to extensive testing. This phase is necessary to ensure that the model works according to the requirements. Furthermore, the goal of testing a ML model or software in general is to point out any unnoticed defect or flaws during the initial development. During this phase, the final model is usually evaluated using previously unseen data, also called *holdout set* [56]. Lastly, the model is deployed and made available to other services.

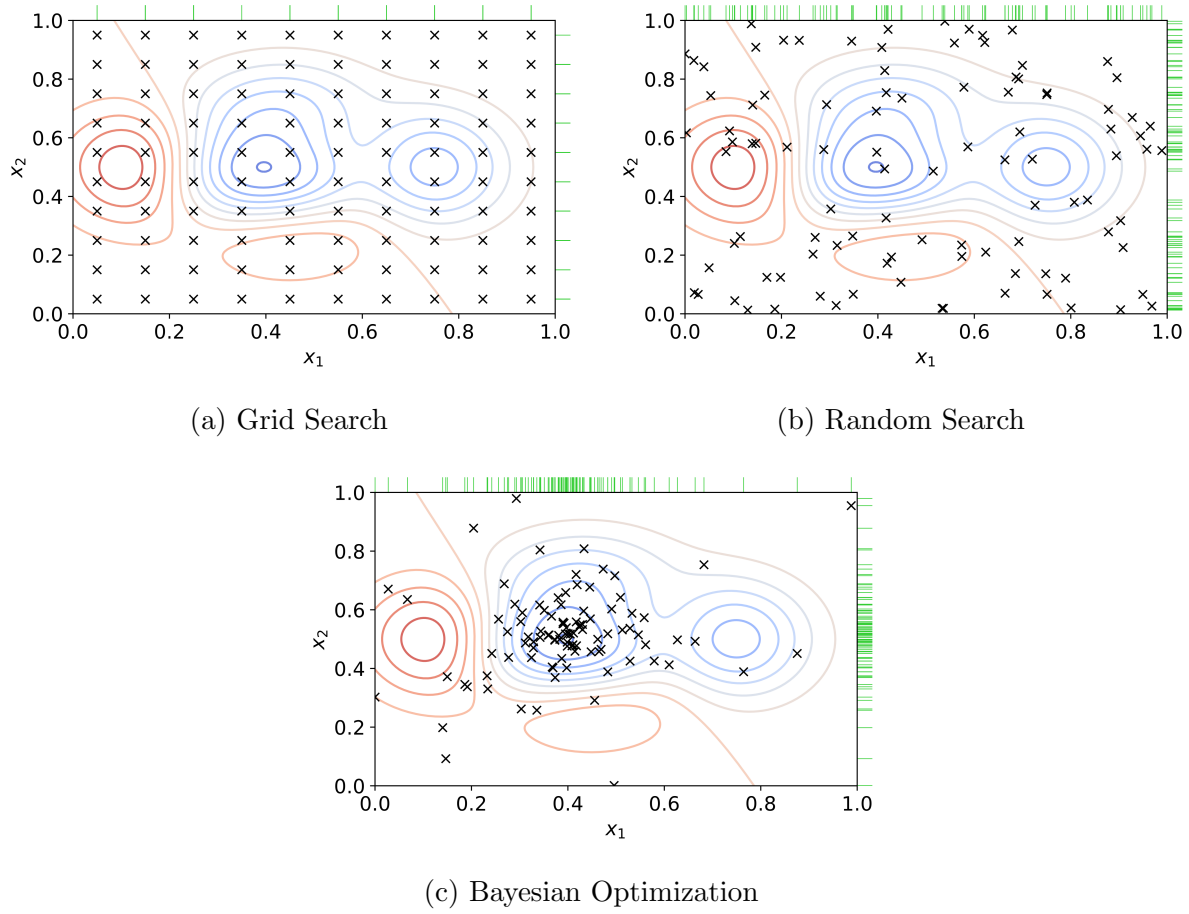


Figure 2.4: Hyperparameter optimization techniques [52]



Figure 2.5: K-Fold CV (K=5) [55]

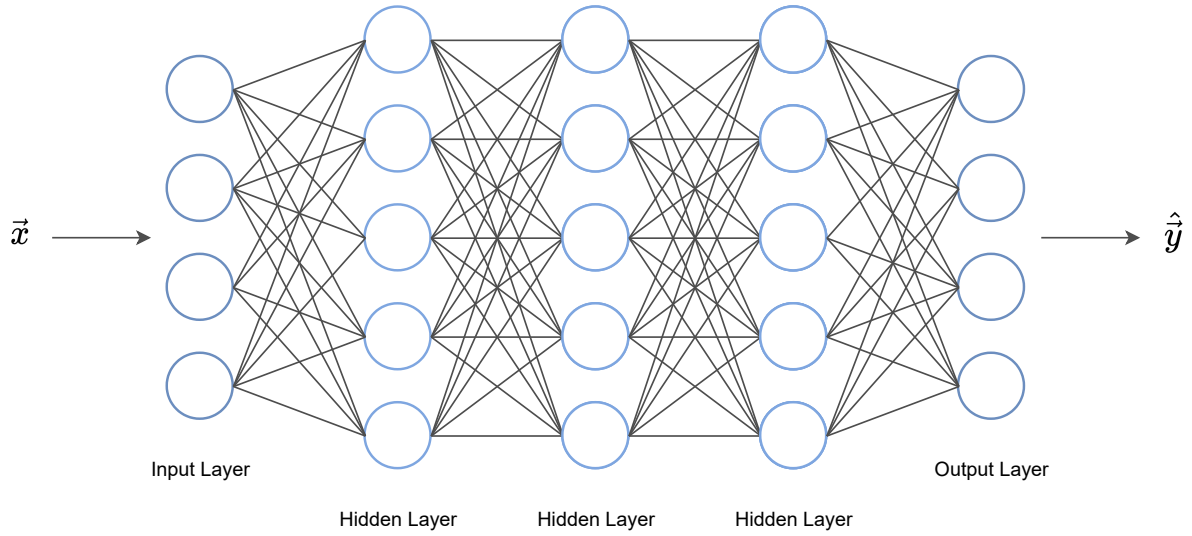


Figure 2.6: Artificial Neural Network (based on [57])

2.2.2 Deep Learning (DL)

Deep Learning (DL) is a subdomain of ML that primarily uses Artificial Neural Networks (ANNs), also known as Feed-Forward Neural Networks (FFNNs), to process large amounts of data for solving complex tasks, which usually would require human intervention. Nowadays, DL algorithms are recognized as being well-integrated into different products and services and generally they find application in different fields, ranging from law enforcement to financial services and customer service [58, 59].

ANNs are specifically designed to mimic the function of the human brain, thus offering a more generalized way of modeling and approaching problems. Similar to the structure of a nervous system, the underlying architecture of an ANN (as shown in Figure 2.6) usually consists of small computing units or nodes, known as (artificial) neurons. A typical ANN generally contains dozens or even millions of interconnected neurons arranged in several levels, called layers. There are three kind of layers: *input layer*, *hidden layer(s)* and *output layer* [60].

The first layer of a neural network, also known as input layer, is intended to receive an input vector (\vec{x}) containing initial data and forwards it to subsequent layers. It is important to note that the input vector must contain as many elements as there are neurons in the input layer. The last layer, *i.e.*, output layer, is responsible for producing the final prediction (denoted by \hat{y}). The number of neurons in the output layer represent the different prediction classes and is totally dependent on the initial problem at hand. In the case of multi-class classification problems the output layer must include as many nodes as there are possible classes. On the other hand, in a simple binary classification problem, two neurons would be sufficient [61].

However, in order to obtain the prediction vector \hat{y} , the neural network has to perform various mathematical computations. These operations are mostly done in the hidden layers through the synapses interconnecting the neurons. Figure 2.7 gives a visual representation of a single neuron, having n inputs and a single output y . As shown in Figure

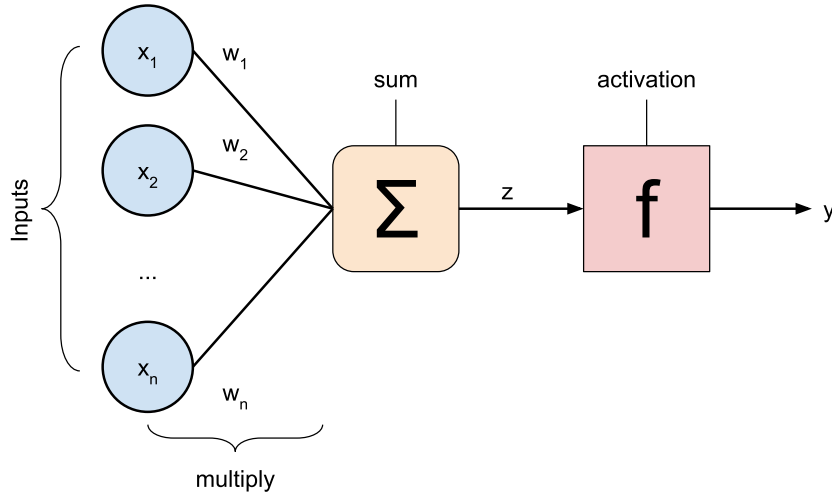


Figure 2.7: Representation of a neuron [62]

2.7, each connection has an associated strength, also called synaptic weight (W_n). In a first step, the neuron computes the summation of every input (x_n) multiplied with the corresponding weight (W_n). The result (z) is then fed to an activation function f which then produces an output signal y . The process can be summarized with the following mathematical formula (2.1) [62]:

$$y = f(z) = f\left(\sum_i w_i x_i\right) = f(w^T x) \quad (2.1)$$

The activation function f is essential for transforming an input signal into an output, which is in turn passed to the next layer. The type of activation function has also direct impact on the neural network's prediction accuracy. In the case that the activation function is not explicitly defined, the neural network behaves the same as a linear regression model. However, non-linear activation functions are generally preferred over linear ones, since they allow to model complex outputs which otherwise could not be reproduced using linear input combinations [63]. Below is a list of the most common non-linear activation functions as well as a depiction of the corresponding curves (Figure 2.8) [64].

- **Sigmoid.** The sigmoid function, also called *Logistic function*, is the most frequently used activation function. The input of the function is transformed into a value between 0 and 1. For any possible input, the sigmoid function has a characteristic S-shaped curve. The sigmoid function is defined by the following formula (2.2) [65]:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

- **Hyperbolic Tangent (tanh).** Tanh is another widely used activation function which shares some similarities with the sigmoid function. As shown in Figure 2.8, the change in output accelerates close to $x = 0$ and similar to the sigmoid function, for very large values of x , the curve approaches 1, but it never actually reaches it. However, a significant difference with the sigmoid function is evident for $x < 0$. For

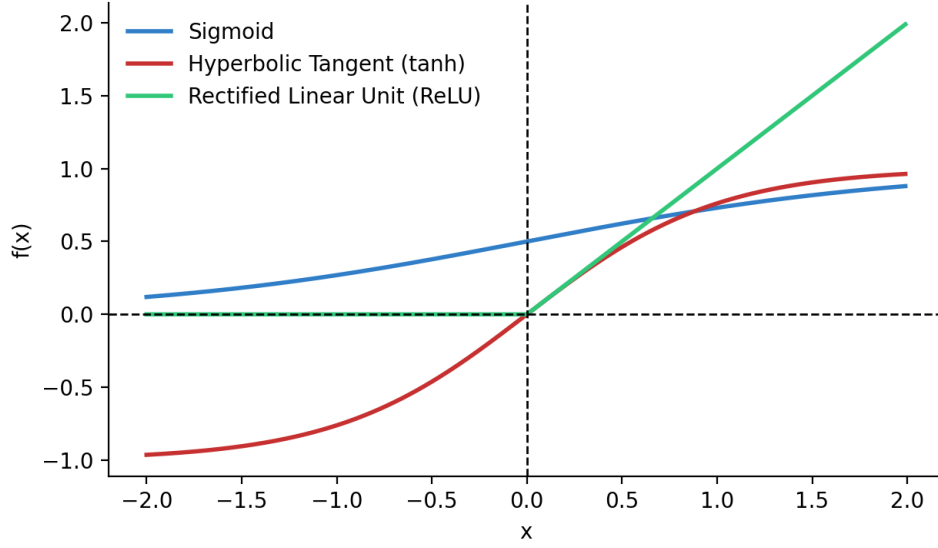


Figure 2.8: Most common non-linear activation functions

such values of x , the tanh curve approaches -1 instead of 0 [66]. The tanh activation function is described by the formula below (2.3):

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.3)$$

- Rectified Linear Unit (ReLU).** ReLU is another non-linear activation function that has gained a lot popularity in the field of DL. The main advantage of ReLU over other activation functions is that it does not activate the neurons at the same time. More specifically, in order for the neurons to be activated, the input signals must have a positive value. For any negative input value, the ReLU function returns 0 meaning that the neuron is not activated. As a result, when compared to other non-linear activation functions, ReLU is considered to be far more computationally efficient and to offer better performance [67, 68]. The ReLU activation function can be mathematically described by the following formula (2.4):

$$f(x) = \max(0, x) \quad (2.4)$$

The final step of the training phase in a FFNN is to evaluate the predicted output against the expected output. For this purpose, cost/loss functions are used to measure the resulting difference and assess the performance of the network. The main goal here is to correctly calculate and minimize the error by adjusting the weights of the network. This is usually done by combining a technique called *Backpropagation* [70] and optimisation algorithms, which based on the computed gradient, *i.e.*, the partial derivative of the cost function with respect to the weights, repeatedly adapt the synaptic weights until the global minima of the loss function is reached [69]. Lastly, the work of Shiliang Sun et al. [71] provides a survey of the different weight optimisation techniques available, including a summary of the main properties, advantages and disadvantages of each method.

Chapter 3

Related Work

This chapter provides an overview of the traditional cybersecurity risk assessment methodologies as well as recent ML approaches aiding at assessing risks in various industries. In the following paragraphs, the most popular risk assessment standards and frameworks are declared. Consequently, several scientific works proposing ML-based solutions to support risk assessment in various sectors are presented, followed by an analysis of the current state of the art in cybersecurity risk assessment. Finally, the elicited characteristics of the different approaches are summarized and compared.

Nowadays, many organizations consider cybersecurity to be a top priority [72]. This has lead to the development of many different frameworks, *i.e.*, a collection of standards, guidelines and best practices to identify and report different kind of risks. These frameworks are meant to provide decision-makers of an organization valuable information about security and to support the process of resource allocation. The **NIST CSF**¹ is a cybersecurity framework provided by the National Institute of Standards and Technology (NIST) and is the most widely used framework by US companies. Known for providing a base level of cybersecurity, the NIST CSF offers public and private organizations a detailed and structured guidance for preventing, identifying and responding to cyber-threats. It comprises of five core functions: *identify*, *protect*, *detect*, *respond* and *recover*. These functions are meant to represent the highest level of abstraction within the framework [73].

- **Identify.** This function supports the organization in developing an understanding in order to manage cybersecurity risks related to systems, people, assets, data and capabilities.
- **Protect.** A function that involves the organization to proactively develop and implement appropriate safeguards for the previously identified risks to ensure a reliable delivery of the business critical services.
- **Detect.** The third activity is about effectively detecting when a cybersecurity event is about to occur. This includes implementing continuous monitoring techniques to

¹<https://www.nist.gov/cyberframework>

identify possible threats and verify the actual protections. It is also fundamentally important to ensure that anomalies are detected and their possible impact on the infrastructure/business is reported.

- **Respond.** This function requires the organization to design and implement appropriate activities against a detected cybersecurity incident. For instance, the Organization should ensure that response plans are actually executed in the case of a cyberattack and the communication inside and outside the organization should not be hindered.
- **Recover.** Lastly, as the name implies, the recover function is about designing and implementing recovery plans for services which have been impaired by the attack. Especially for bigger companies, it is of utmost importance to ensure proper level of internal communications during the recovery process.

Overall, the NIST CSF provides a comprehensive way of standardizing and managing cybersecurity risks [74]. However, a study conducted by the Government Accountability Office (GAO) reported some challenges related to the adoption of the NIST CSF. The biggest obstacle that smaller businesses face is the deficiency of necessary resources for implementing the framework. Moreover, due to the limited resources available, smaller companies are usually forced to prioritize physical security, natural disaster response and insider threats over cybersecurity. Another important challenging aspect is the lack of knowledge and skills required to implement the NIST CSF which in turn is also the main cause of uncertainty among even medium-sized companies [75].

The **ISO/IEC 27001**² is part of a family of standards (ISO/IEC 27000) specifically developed to deal with Information Security (IS) and consists of international standards describing best practices for Information Security Management Systems (ISMS). ISO 27001 serves primarily as a guideline for organizations of any size and industries that are looking to improve IS methods and policies. Moreover, organizations can also get ISO 27001 certified once their ISMS meets the necessary requirements. However, depending on the country where the company is located, the price for the certification can potentially be very expensive and generally no implementation guidance is provided [76, 77].

The **Critical Security Controls (CIS)**³ is a popular cybersecurity framework developed by the SANS Institute. The CIS framework focuses on protecting against most prevalent cyberattacks and mitigating the effects of breaches. Even though it is considered to be not as comprehensive as NIST or ISO 27001, CIS provides a straightforward and practical framework for defense mechanisms and monitoring of high risk areas [78]. A summary of the most popular and established cybersecurity frameworks is given by Table 3.1, highlighting the goals, strengths and weaknesses of each methodology.

²<https://www.iso.org/isoiec-27001-information-security.html>

³<https://www.cisecurity.org/>

Table 3.1: Cybersecurity risk assessment frameworks overview (based on [79])

Framework	Industry	Goal	Strenghts	Weaknesses
NIST CSF	any	Align cybersecurity defenses to organizational goals, measure control maturity	<ul style="list-style-type: none"> – Comprehensive – Builds on past frameworks – Works with many compliance requirements – Freely available 	<ul style="list-style-type: none"> – Difficult to implement – Requires skills and knowledge
ISO 27001	any	Establish security management program	<ul style="list-style-type: none"> – Internationally recognized – Many compliance requirements – Integrates well with other frameworks 	<ul style="list-style-type: none"> – No implementation guidance – High certification cost
CIS	any	Automated controls to protect and monitor high risk areas	<ul style="list-style-type: none"> – Prescriptive – Easy to get started – Updated about every 2 years to address evolving risks 	<ul style="list-style-type: none"> – Not as comprehensive as other frameworks

Recently, many researchers have proposed different approaches to cybersecurity risk assessment addressing the challenges of traditional and established methodologies. DRAFT is a cybersecurity framework proposed by Soumya [80] focusing on the end-to-end Internet of Things (IoT) platform. Not only that, DRAFT also includes a Security Incident, an Event Monitoring (SIEM) tool and a cyberattack resilient framework. The DRAFT framework can be easily integrated into any IoT environment and has been thoroughly tested using different cyberattacks, such as Distributed Denial-of-Service (DDoS) attacks.

Another solution presented in [81] aims at providing a dynamic risk assessment framework in order to estimate the likelihood of cyber-threats in rapidly changing environments. This includes scenarios where companies are exposed to possibly new threats due to their global presence. This approach has been adopted by a leading global energy management

company and the authors have already received positive feedback.

CyFEr is a framework developed by [82] which addresses a major limitation of traditional risk assessment methodologies; the majority of the current frameworks do not allow the organization to prioritize the requirements necessary to reach a certain degree cybersecurity maturity. To address this issue, the authors evaluate the application of various rank-weight methods. Moreover, the solution was also tested against a blockchain cybersecurity framework (BC2F), developed using the NIST CSF.

On the other hand, SEconomy is a step-based framework described by [12]. In contrast with the previously discussed solutions, SEconomy is designed to measure the economic impact of cybersecurity threats in distributed environments integrating the NIST guidelines for assessing cyber-threats.

Pappalardo et al. [8] takes it a step further, by presenting E-MAF, a multi-sector risk assessment framework. In fact, this approach allows the evaluation of cyber-threats in trans-sectoral and inter-sectoral environments. This framework builds upon studies conducted on existing cyber-risk assessment and management frameworks, with the goal of finding possible challenges and limitations. The authors came to the conclusion that traditional risk assessment frameworks did not take into consideration the relations between different sectors and were not able to address multi-sector issues. As a result, E-MAF architecture was designed to specifically tackle this limitation.

However, as current research shows, there are still opportunities to improve and evolve traditional risk assessment processes by employing state-of-the-art and trend technologies, such as ML. In fact, over the last few years, ML algorithms have aided in solving domain specific problems in different fields, ranging from image/speech recognition to self-driving cars and online fraud detection. Recently published works have also put their attention to proposing various approaches for supporting engineering risk assessment in different industries. In the following paragraphs, some solutions will be discussed, followed by Table 3.2 showing an industry-wise segmentation of the most popular ML algorithms used to aid risk assessment. The ML algorithms taken into consideration are: Artificial Neural Network (ANN), Support-Vector Machine (SVM), Decision Tree (DT) and K-Nearest Neighbors (KNN).

In their study, Castro et al. [84] focus on investigating possible factors that have the greatest impact on car accidents. More specifically, the authors use different ML algorithms, such as DTs and ANNs, to identify relevant patterns and detect most frequent key factors involved in car accidents with the goal of exploring different factors that may lead to various injury risks.

Similarly, the authors of [85] applied different ML algorithms to classify the risk of severe injury in the automotive industry. Based on over 5,000 traffic accident records, the ML models were able to extract significant risk factors and make accurate predictions. Other works, such as [86] have applied other ML to solve similar problems.

In the healthcare industry, [87] describes an approach to identify latent risks by regularly measuring the physiological signals of patients. The authors design an ANN to automatically extract and learn features from raw, unprocessed datasets to detect possible feature anomalies.

[88] proposes an automated fall detection system, which similarly to the applications described above uses various ML models to proactively evaluate risk factors. Using several sensors, the authors collected a substantial amount of raw data for model training and testing.

Xu et al. [89] develop a transient stability assessment model for earlier detection of black-out risks. This solution addresses the main issues of past approaches, such as excessive training time, complex parameter tuning and overall inefficiency for real-time predictions. The proposed model is compared with some state-of-the-art approaches with regards to computation time and prediction accuracy. In the field of nuclear energy, [90] presents an SVM approach designed to classify transients in nuclear power plants. The data measured by the sensors constantly monitoring the power plants is used by the SVM classifier to detect anomalies and evaluate the risks of possible malfunctions of various components, *e.g.*, the feedwater system of a boiling water reactor.

In the field of cybersecurity, research has tended to focus mainly on leveraging ML to detect various types of cyberattacks and recognize breaches [91]. However, as stated by [9] and [92], ML has the potential to significantly change the cybersecurity and risk assessment landscape.

Table 3.2: Industry-wise segmentation of ML applications for risk assessment [83]

Industry	Machine Learning algorithms	Publications
Automotive	ANN	[84, 85, 86]
	KNN	-
	SVM	[86]
	DT	[84, 85]
Healthcare	ANN	[87, 88]
	KNN	[88]
	SVM	[88]
	DT	-
Energy & Nuclear	ANN	[89]
	KNN	-
	SVM	[90]
	DT	-
Cybersecurity	ANN	-
	KNN	-
	SVM	-
	DT	-

Chapter 4

The SecRiskAI Approach

This thesis proposes an approach for conducting qualitative cybersecurity risk assessment using ML techniques. In this chapter, the solution is introduced, starting with an high-level overview of the system architecture and a description of each component involved. Next, the ML-based risk assessment workflow is described and the scope of each phase clearly defined. Finally, details on the integration with the MENTOR’s recommendation API [13, 14] is provided.

Figure 4.1 gives an high-level architecture overview and highlights the system components’ interactions. In Step 1, the user is able to access the dashboard through any browser without the need of an account. The Graphical User Interface (GUI) (*i.e.*, web-based interface) was designed in a way to provide total visibility of business-related KRIs and, at the same time, increase productivity and better forecasting of important aspects related to the business security. Moreover, through the web-based interface the user is able to change both contextual information and other parameters (*e.g.*, available budget, service type and desired deployment/leasing period) required for the risk assessment and the protection service recommendations.

In order to use the information provided by the user to make risk predictions, an additional layer is required. In this approach, this task is performed by the Middleware (Step 2). More specifically, as soon as the request sent by the client is received, the *Request Processor* processes it and forwards the information to the *Profile Evaluator*, which is in charge of contacting the ML models, evaluating the prediction response, and, when specific conditions are fulfilled, establishing a connection with MENTOR (Step 6).

To perform an actual risk prediction, a request to the *Risk Classifier* is sent. The Risk Classifier is a prediction service included in the ML Classifier Layer (Step 3) and is essentially used to expose the trained ML models through the API. Additionally, the ML Classifier Layer also stores the trained ML models as well as the *Data Scalers* used to normalize the input data and increase prediction accuracy.

The process of training, validating and testing the ML models takes place in the ML workflow Layer (Step 4) and is usually carried out by data scientists/experts in the company. In summary, the *Data Generator* component is used to initialize the synthetic data generation process. Afterwards, the data is processed (*i.e.*, Data Processor) and used by the *Model Builder* for training, validating, testing, and building the models. Each phase

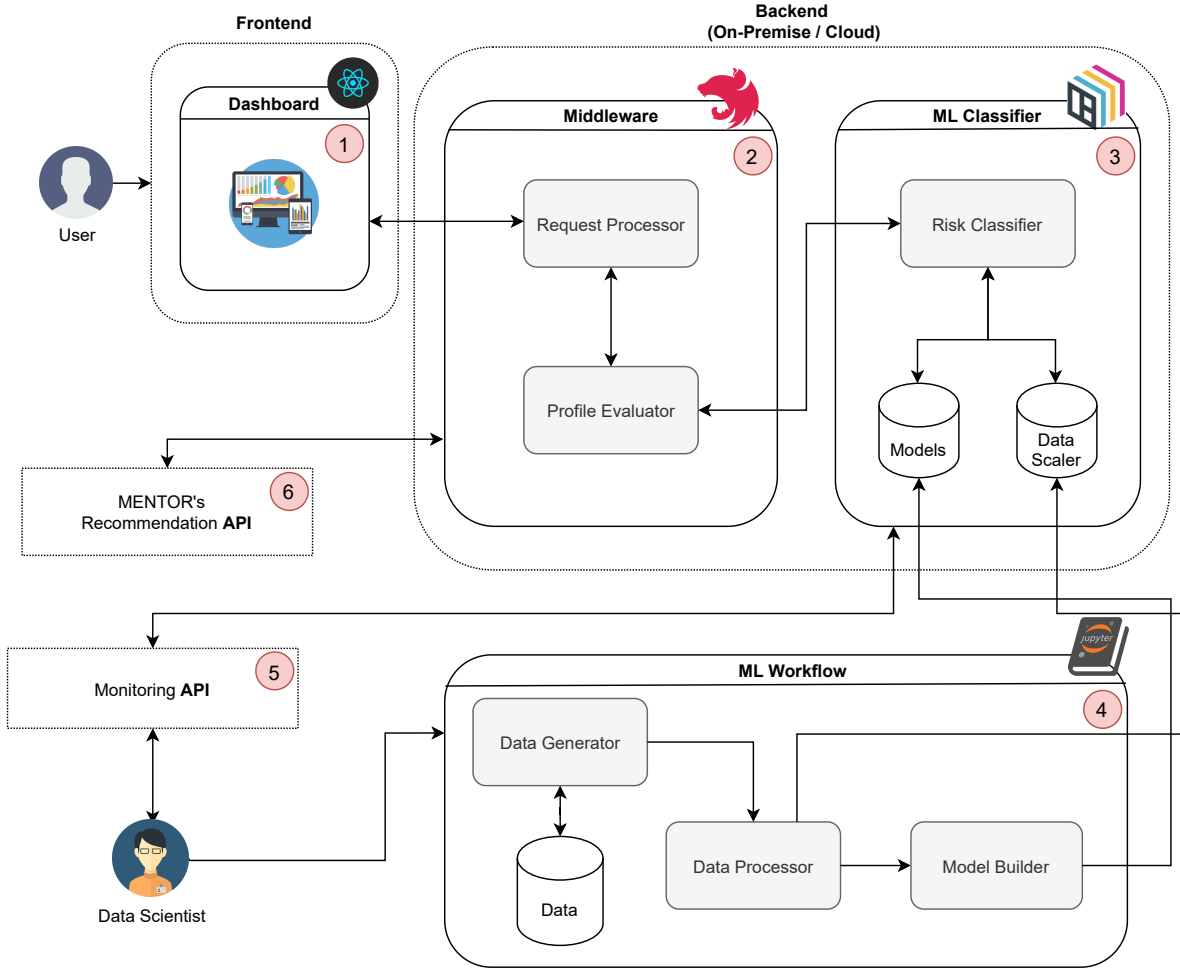


Figure 4.1: Architecture overview

of the ML Workflow is described with sufficient level of details in Section 4.1. Lastly, the interface indicated by Step 5 provides a monitoring API for checking the status of the deployed models, retrieving model-specific metadata (*e.g.*, version, creation time, accuracy) and other metrics about the prediction service (*e.g.*, request duration in seconds).

4.1 Risk Assessment Workflow

Once the opportunities of applying ML to cybersecurity risk assessment are defined and well-understood, the process of designing and developing a ML workflow (*cf.* Figure 4.1 - Step 4) begins. Figure 4.2 provides a flow chart of the supervised ML workflow implemented in this thesis, as crucial part of the solution. After the initial problem definition, the most important stage is data collection/gathering. Usually, in this phase, data is collected from sensors or other different sources and stored for further processing. However, in the field of cybersecurity risk assessment companies either do not disclose any kind of information at all [93] or in some cases they publish various reports which are often incomplete and difficult to extract meaningful and interesting results from [94, 95, 96]. To

address this issue, a synthetic data generation approach was designed and implemented.

4.1.1 Data Gathering

Synthetic data is commonly referred to as data that is created by different algorithms that try to mirror the statistical properties of the original data without revealing any actual information about the subjects [97]. After exhaustive research and analysis of different cyberattacks and corresponding companies' contextual information, the following parameters to be used as basis for this work were identified:

- **Revenue.** Referred to the income generated from normal business activities and operations, and in most cases is also used to classify businesses by providing a scale for determining their sizes [98].
- **Cybersecurity Investments.** Normally, businesses already have cybersecurity investments strategies in place to ensure a proper level of defense. This kind of information needs to be taken into consideration during the cybersecurity risk assessment, as it may have an impact on the likelihood of being targeted by a cyberattack.
- **Number of Employees and Training Level.** Similar to the revenue, information regarding the actual number of employees in a company as well as the corresponding cybersecurity training level (*e.g.*, cybersecurity basic knowledge and phishing training) represent essential contextual information required for assessing possible cyber-risks. The employee training level is measured in “*Low*”, “*Medium*” and “*High*”.
- **Successful/Failed Cyberattacks.** These parameters are meant to indicate the number of cyberattacks that the company has already experienced. This includes different attacks (*e.g.*, DDoS, Ransomware, and Phishing) that have targeted the organization's infrastructure and resulting in either a financial loss or reputational damage. Failed attempts are also taken into consideration.
- **Known Vulnerabilities.** For an effective and comprehensive risk assessment, it is essential to report any known vulnerabilities of the infrastructure. Vulnerability management is usually a key responsibility of the companies' IT security team. This phase usually involves assessing and reporting any security vulnerability present in the organization's systems [119]. There are a variety of comprehensive tools used for vulnerability scanning, such as Nmap [99], Metasploit [100] and OWASP [101]. Currently, the total number of known vulnerabilities is defined during the synthetic generation process.
- **External Cybersecurity Advisor.** In order to further strengthen their cyber resilience (*i.e.*, the ability to prepare for, respond to, and recover from cyberattacks), business are encouraged to hire external Cybersecurity Advisors (CSAs) [102]. Furthermore, CSA provide a variety of services, such as cyber preparedness, strategic messaging, working group support, partnership development, cyber assessments, incident coordination and support [102]. During the synthetic data generation phase, a binary value will be generated (either “*Yes*” or “*No*”).

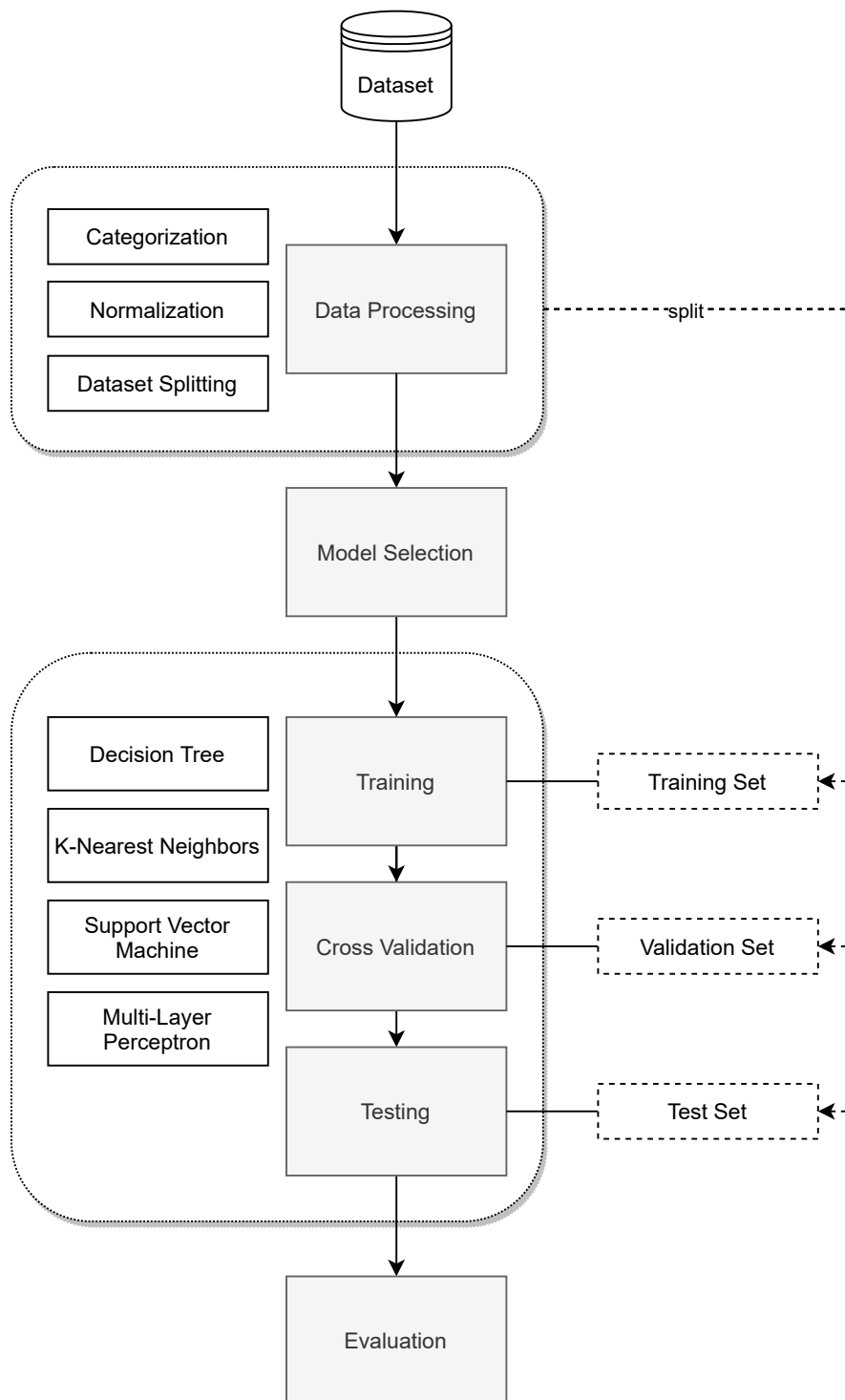


Figure 4.2: Supervised Learning ML workflow

- **Risk.** The last parameter represents the value of the qualitative risk assessment based on the previously generated parameters. Since the synthetic data generation process is designed to generate historical records of companies operating in comparable industries, the value of the risk column may be derived from past formal or tailored qualitative risk assessment techniques (*cf.* Section 3). The generated risk can have one of the following values: “*Low*”, “*Medium*” and “*High*”.

In order to generate the information mentioned above, some assumptions were made. First, upper/lower boundaries for each column were specified, so that each generated value would effectively lie in the defined range. The Table 4.1 gives an overview of the determined boundaries as well example of values for each generated information.

Table 4.1: Overview of the generated attributes

Information	ID	Range	Example
Revenue	business_value	0 to 5,000,000	2,500,000
Cybersecurity Investment(s)	invested_amount	0 to 30% * business_value	500,000
Successful Attacks	succ_attack	0 to 50	5
Failed Attacks	fail_attack	0 to 50	10
Number of Employees	nr_employees	30 to 10,000	4,450
Employee Training	employees_training	Low, Medium or High	Medium
Known Vulnerabilities	known_vuln	0 to 10	9
External Cybersecurity Advisor	external_adv	Yes or No	No
Risk	risk	Low, Medium or High	Low

It is important to note that the risk is not randomly generated, instead it is computed based on the generated attributes illustrated in Table 4.1 using the following generalized formula (4.1). For supervised learning to work, the dataset must be labeled. As a result, the *computed_risk* output is mapped to either a “Low”, “Medium” and “High” class. However, a manual labeling process would be too expensive, since the generated dataset would include thousands of records. Therefore, based on the numeric value of *computed_risk* a mapping range was defined. This means, that each *computed_risk* value is labeled using the range specified in Equation 4.2.

$$\begin{aligned}
i_r &= \frac{invested_amount}{business_value} \\
e &= \frac{nr_employees}{tot_empl} * map(employees_training) \\
att_r &= \frac{succ_attacks}{max_attacks} \\
v_r &= \frac{known_vuln}{max_known_vuln} \\
adv_i &= map(external_adv) \\
map(x) &= \begin{cases} 0, & \text{if } x = Low \\ 1, & \text{if } x = Medium \\ 2, & \text{if } x = High \end{cases}
\end{aligned}$$

$$computed_risk = i_r + e + adv_i - att_r - v_r \quad (4.1)$$

$$risk(x) = \begin{cases} High, & \text{for } x < 0 \\ Medium, & \text{for } 0 \leq x < 1 \\ Low, & \text{for } x \geq 1 \end{cases} \quad (4.2)$$

4.1.2 Data Processing

Once enough data has been successfully generated, the processing phase starts. The ML algorithms (*cf.* Section 2) require an initial processing step as they are not able to work with raw data. In a first step, any categorical variable present in the dataset is handled. Specifically, variables like “employee training level” and “external cybersecurity advisor” are mapped to numerical values, which are easier for ML algorithms to work with.

A further normalization may be necessary, depending on the selected ML algorithm [120]. Normalization is the process of scaling data into a range of $[0, 1]$. Some ML algorithms are known to be highly sensitive to features with varying degrees of magnitude, range and units. The dataset generated for this work includes features such as “revenue” and “number of employees” that have different ranges and training sensitive models on unscaled data may lead to lower performance and accuracy. In this solution, a normalization technique known as Min-Max scaling [121] was used and is defined by the following formula:

$$x_{scaled} = \frac{x - min(x)}{max(x) - min(x)} \quad (4.3)$$

The Min-Max normalization technique is applied to the entire dataset but only to “features”, namely every column except “risk” which contains the three output classes based on which future predictions will be made. The last step in the processing phase involves splitting the dataset into a training, validation and test set, as shown in Figure 4.2.

4.1.3 Multi-Class Classification Algorithms

In ML, Multi-Class Classification algorithms (MCC) aim to solve problems of classifying instances into one of three or more output classes [103]. In the model selection phase (*cf.* Fig. 4.2) popular MMC algorithms are chosen for conducting qualitative cybersecurity risk assessments. The main goal is to design and develop ML models that, based on actual contextual information, can make accurate qualitative risk assessment predictions and further monitor the organization’s infrastructure by providing continuous assessment based on input data.

Decision Tree

Decision Tree (DT) is a Supervised Learning (SL) algorithm (*cf.* Section 2) for classification used in the proposed solution [104]. This technique essentially looks at the feature values of the input dataset and categorizes them according to a specific parameter, also known as information gain. The pseudo-code below (Algorithm 1) illustrates the procedure for implementing the decision tree algorithm. In a first phase, the algorithm iterates over every feature column in the input dataset D containing organization’s historical data and computes the information gain. The goal is to find the feature column having the highest information gain which will, in turn, serve as a decision node of the tree. Next, the algorithm continues splitting the dataset on the identified decision node and performs the same search on the sub-datasets. This way, a tree structure is constructed with each node representing a feature column and the leaves indicating the output class.

Algorithm 1: DTree

```

input:  $D$ , dataset containing organization’s contextual information
  Tree = {}
  for all attributes  $\in D$  do
    Find the attribute which best divides  $D$  using information gain
     $X_{best} \leftarrow$  feature column with the highest information gain
  end for
  Tree  $\leftarrow$  Create a Decision Node that divides the dataset on  $X_{best}$ 
   $D_{sub} \leftarrow$  sub-datasets from  $D$  splitted on  $X_{best}$ 
  for all  $D_{sub}$  do
    Treesub  $\leftarrow$  DTree( $D_{sub}$ )
    Add Treesub to corresponding branch of Tree
  end for
return Tree

```

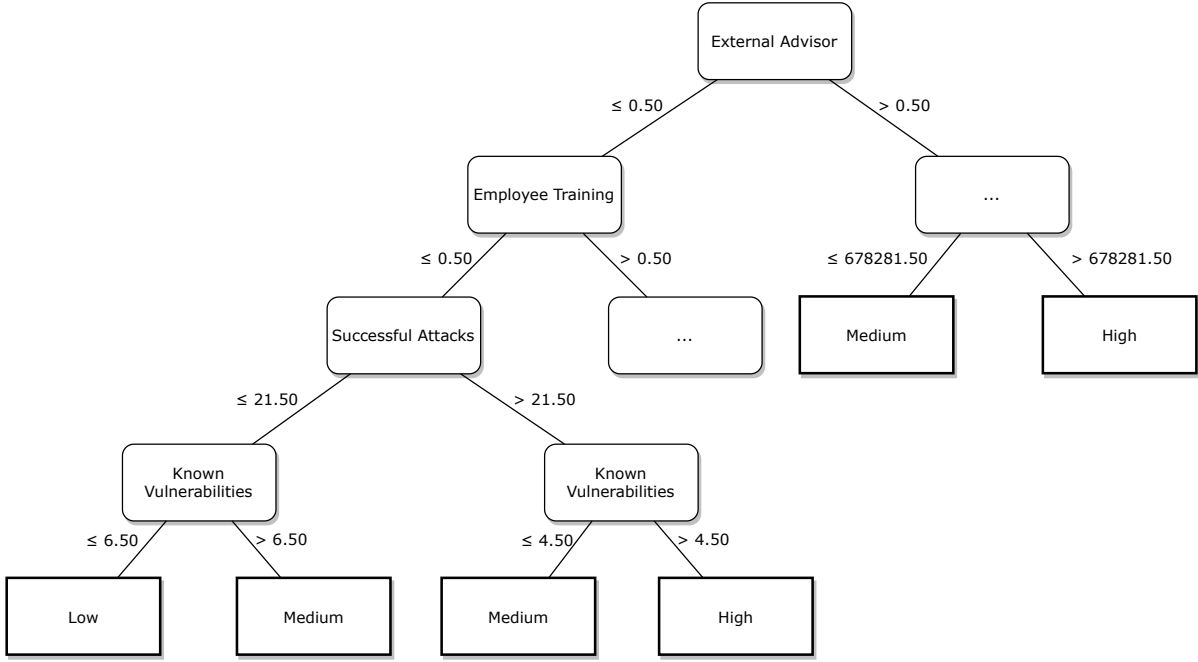


Figure 4.3: DT visualization

Besides being an easy to use and straightforward classification technique, this algorithm can be trained on historical data, without requiring extensive data pre-processing [105]. That is, compared to other classification algorithms used in this approach, the decision tree requires less effort for data preparation and the normalization step is not required. The resulting model is thus easy to understand for both technical and non-technical stakeholders. Fig. 4.3 gives a visual representation of a decision tree algorithm trained on the generated dataset. In order to make a prediction using the DT, a new sample i would traverse the tree based on each feature value and the resulting leaf value would be the output class.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is another SL algorithm (*cf.* Section 2) used to solve classification problems. More specifically, KNN is usually referred to as instance-based classifier as the main idea behind this technique is to memorize the input dataset to make future predictions [106]. As indicated by the pseudo-code (Algorithm 2) below, KNN requires three input parameters: a dataset D containing the historical information is given, a chosen number of neighbors k and x , a sample that is to be classified. The algorithm then proceeds on computing the distance between x and every record contained in D . Next, the computed distances are sorted in ascending order and k closest samples, also known as *neighbors*, to x are selected. Finally, the predicted class of x ($Class_x$) is based on the similarity with the neighbors, meaning that x is labeled following a majority voting of classes among the neighbors.

In essence, KNN calculates the probability of a sample x belonging to a specific class, based on neighbors observations. Compared to the DT, KNN requires more data pre-processing.

Algorithm 2: knn

input: D , dataset containing organization's contextual information
 k , number of nearest neighbors
 x , unclassified sample
for all $r \in D$ **do**
 Compute distance between r and x
end for
 $Neighbors \leftarrow$ Sort computed distances and select k closest samples to x
 $Class_x \leftarrow$ majority output class based on $Neighbors$
return $Class_x$

On the other hand, the training phase is definitely faster and new training data can be seamlessly added without the need of reconstructing the model. Figure 4.4 provides a visual representation of the KNN classification with k equal to seven and x being a new sample to classify. In this example, only two dimensions are taken into account (*i.e.*, cybersecurity investment(s) and # of employees). Once the k closest neighbors to x are identified, it is apparent from Fig. 4.4 that the predicted class of x is “Low”, since the majority of the neighbors belong to the “Low” class.

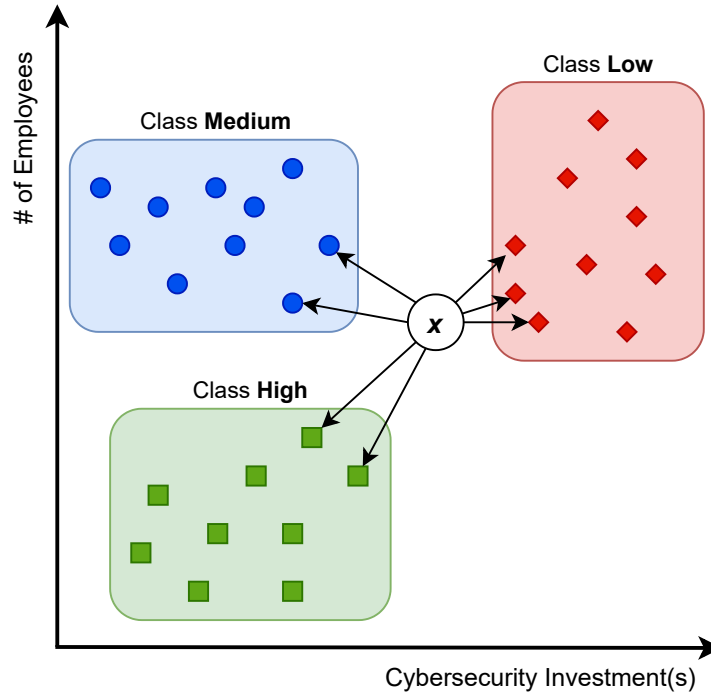


Figure 4.4: KNN visualization ($k = 7$)

Support Vector Machine

The Support Vector Machine (SVM) is the third SL classification algorithm (*cf.* Section 2) considered in this work. In contrast with DT and KNN, SVM uses a line or hyperplane

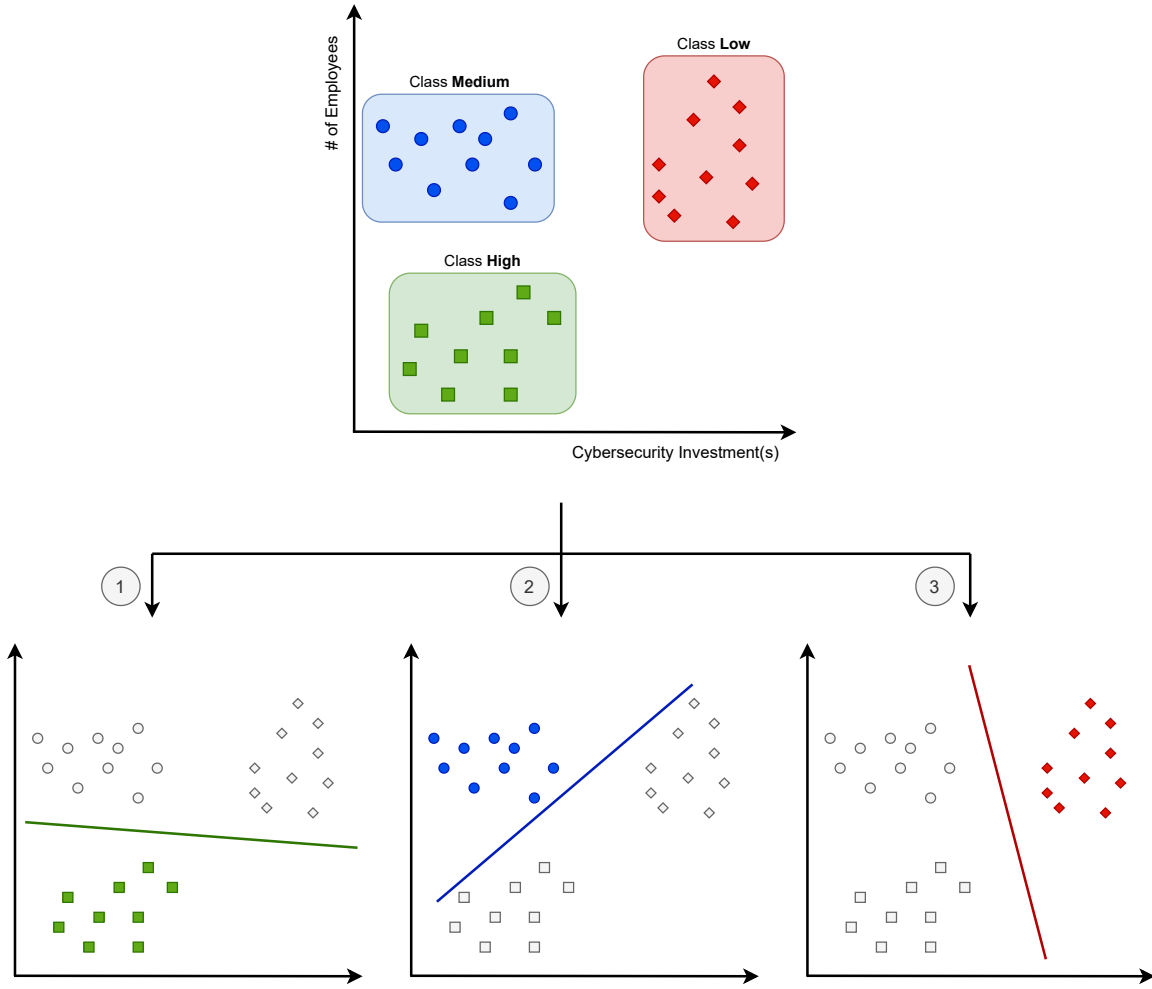


Figure 4.5: SVM visualization

to separate input data into classes. Moreover, SVM is known to be computationally less expensive than KNN but does not support MCC natively. To achieve that, a *One-vs-Rest* strategy is followed [107]. First, the multi-class dataset is broken down into multiple binary classification problems (Fig. 4.5). In this case, the following classification problems are identified:

- High vs {Low, Medium} (Fig. 4.5 - Step 1)
- Medium vs {Low, High} (Fig. 4.5 - Step 2)
- Low vs {Medium, High} (Fig. 4.5 - Step 3)

Next, a binary classifier is trained on each binary classification problem and is able to predict a class probability (P_{class}), *i.e.*, the probability of an object belonging to a specific class. After the training phase, the binary classifiers return the probability of a sample being labeled as “Low” (P_{Low}), “Medium” (P_{Medium}) and “High” (P_{High}). Finally, the model

that is able to predict the class of an unclassified sample x with the highest confidence is chosen and is represented with the following mathematical formula (4.4):

$$Class_x = \operatorname{argmax}(P_{Low}, P_{Medium}, P_{High}) \quad (4.4)$$

When dealing with larger datasets and n output classes, SVM would require the creation of n binary classifiers for each class, resulting to high computational costs. SVM does also suffer from performance issues when confronted with overlapping classes, *i.e.*, data points being not well separated [108]. On the other hand, SVM is a very flexible algorithm and allows the specification of a “kernel” function which can be linear (Fig. 4.5) but can also be of different types, such as nonlinear, polynomial, radial basis function, and sigmoid to solve many non-linear problems [109].

Multi-Layer Perceptron using Backpropagation

Multi-Layer Perceptron (MLP) is the fourth and last SL classification algorithm explored in this work. More specifically, MLP is a class of feedforward ANN, hence it inherits the characteristics of ANNs, such as input layer, hidden layer(s), output layer, perceptrons and activation functions (*cf.* Section 2). Figure 4.6 gives a simplified visual representation of the MLP model constructed in this thesis. Each node in the input layer corresponds to a specific feature of the generated dataset. Moreover, as shown by Figure 4.6, the MLP model has a total number of two hidden layers having five neurons each. Choosing the best parameters for an ANN is a very challenging task, as there are no clear rules and it really depends on the complexity of the underlying problem. For this thesis, the decision was based on the guidelines proposed by [110, 111] as well as extensive exploratory research and testing. On the other hand, the output layer was defined based on the output classes of the model (*i.e.*, Low, Medium, and High). Therefore, it consists of three neurons representing each possible classification state.

During the training phase, the MLP uses a technique called *backpropagation*. An ANN propagates the input data forward through the neurons towards the output layer, where the prediction occurs. The backpropagation algorithm [70] refers to the process of propagating the information about the prediction error backward from the output layer throughout the entire network with the goal of adjusting the weights and improve accuracy. Figure 4.6 also gives an example of a backpropagation mechanism initiated as soon as the original label (“Medium”) and predicted class (“Low”) differ. The computed error/loss is defined by the simplified formula below (Equation 4.5) and lastly is used to adjust the weights in the hidden layers [112, 113].

$$E_o = O_{actual} - O_{predicted} \quad (4.5)$$

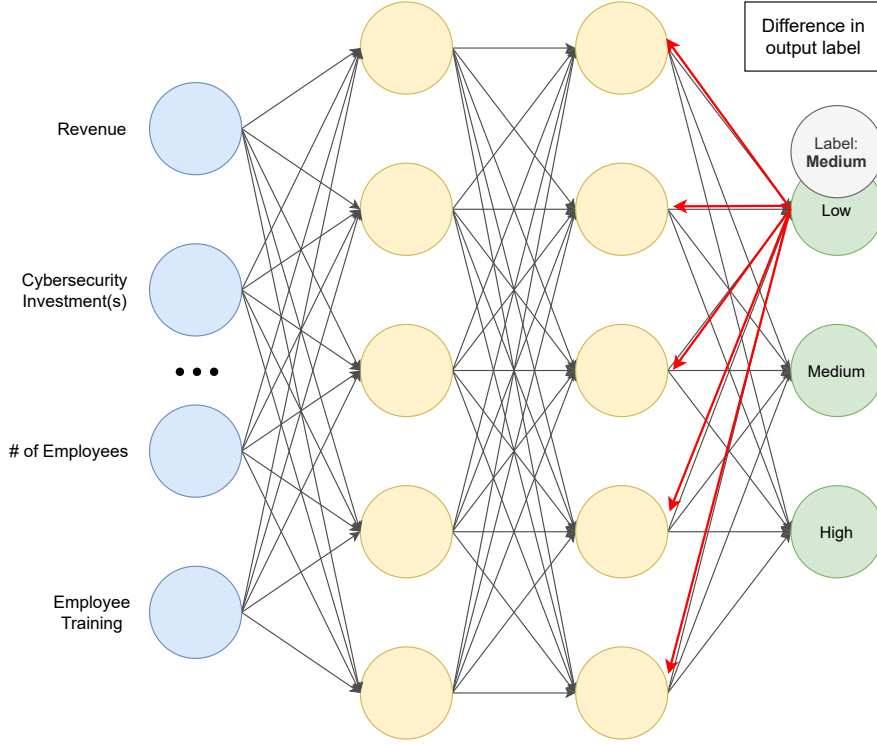


Figure 4.6: MLP visualization

4.1.4 Training, Cross Validation & Testing

Once the dataset is generated and the required ML algorithms are chosen, the training phase is initiated (*cf.* Figure 4.7). First, the dataset is split following the 80-20 train-test strategy described in Section 2. Next, the process of choosing a set of optimal hyperparameters, also called hyperparameter optimization, takes place. The main idea here is to use grid search to extensively test every combination from a pre-defined list of parameters values required by the ML algorithm for building the model. Subsequently, the performance of each model is evaluated with the help of a 5-fold CV strategy. The model with the highest accuracy is selected and tested with unseen data, *i.e.*, the test set. Lastly, the entire process is applied to each ML algorithm discussed in the previous sections.

4.2 MENTOR's API Integration

MENTOR [13] is a protection service recommender system aiming to support the cybersecurity detection/mitigation decision process. More specifically, the MENTOR system implements four different similarity measurements to recommend the adequate protection service based on customers' profiles and needs. A typical customer profile required by MENTOR would contain the following information: the region where the company operates, deployment time and leasing period of the service, and pricing. Moreover,

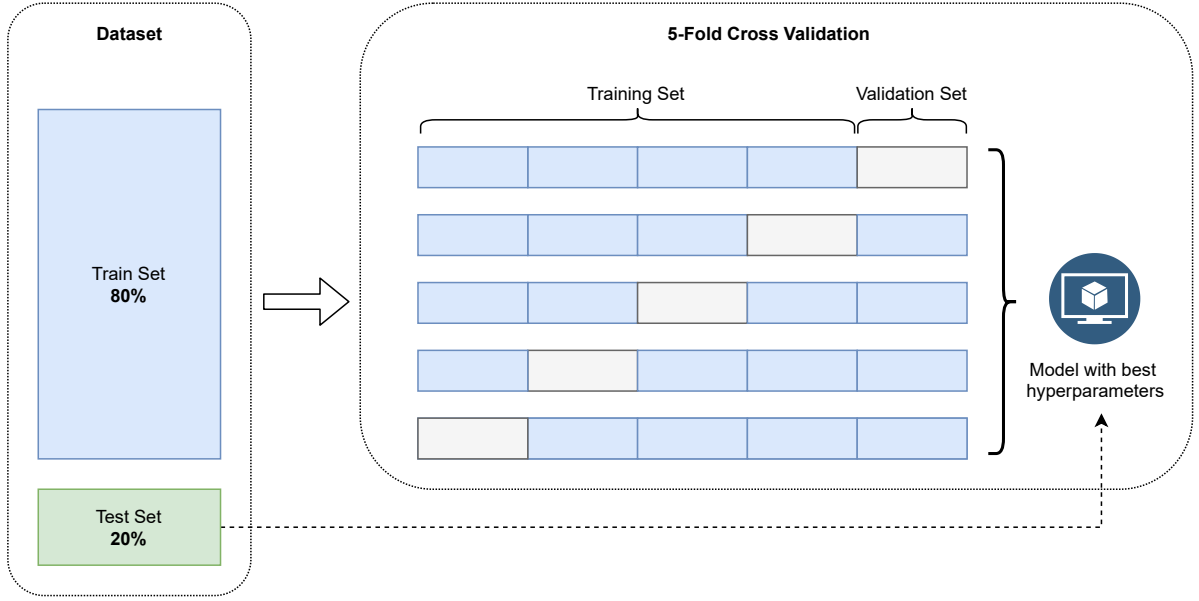


Figure 4.7: Training, CV & testing workflow

MENTOR also offers the possibility to filter out further services based on type (*i.e.*, Reactive/Proactive) and the specific attack type covered by the service [14]. Customers can also arbitrarily assign priorities for some attributes in order to receive a more tailored recommendation.

Additionally, new protection services are automatically being added from Protection Service Providers (PSPs) through a dedicated API. The recommendation engine can also be extended with other similarity measures and is designed to be loosely coupled from the other components in the system. Most importantly, MENTOR also offers an API that, based on the customer's profile (specified as JSON), returns a list of recommended services. The solution proposed in this thesis is designed to integrate the functionality of MENTOR to fully support the customer through the entire risk assessment and cybersecurity investments decision process. More specifically, the Middleware layer (Figure 4.1 - Step 2), based on the data collected by the user and the cyber-risk prediction, is in charge of retrieving the list of recommended protections. Finally, this information is used to provide, in the web-based interface of *SecRiskAI*, the best protection services against the previously identified and assessed cyber-threats.

Chapter 5

Prototype and Implementation

In this chapter, the technical implementation details of *SecRiskAI* are presented. This chapter is divided into two main sections: frontend and backend. The former provides technical details about the libraries and technologies used for building the User Interface (UI). The latter describes the technical implementation of the main layers designed and introduced in Figure 4.1, such as Middleware, ML Classifier and ML workflow. The source code of the developed solution is publicly available on GitHub¹.

Figure 5.1 gives an high-level illustration of the interaction between the user and the different components of *SecRiskAI*. The flow is triggered by the user interacting with the web-based interface. The user enters/updates the profile in the *ProfilePage*. Upon submission, a request is made to the Middleware, which processes the request and sends a POST request to the ML Classifier for prediction as well as a POST request to MENTOR, to retrieve recommendation list of protection services. The response is then constructed in the Middleware and forwarded to the frontend, more specifically the *DashboardPage*, which proceeds building the UI. In case there are any errors, such as invalid profile, a `404 bad request` is sent from the Middleware and the error message is displayed in the dashboard accordingly.

5.1 Frontend

The frontend of the *SecRiskAI* application is implemented using *React*², a popular and widely adopted JavaScript library for building user interfaces. For this particular prototype, *TypeScript*, a well known typed superset of JavaScript, was used [114]. This decision was based on the benefits that TypeScript offers over plain JavaScript, such as: static typing, readability and improved maintainability [115].

By writing reusable UI components for React applications, web developers can save time, increase productivity and make the entire application easier to develop and maintain [116]. Additionally, React is also characterized by the following benefits [117]:

¹<https://github.com/Sulasdeli/SecRiskAI>

²<https://reactjs.org/>

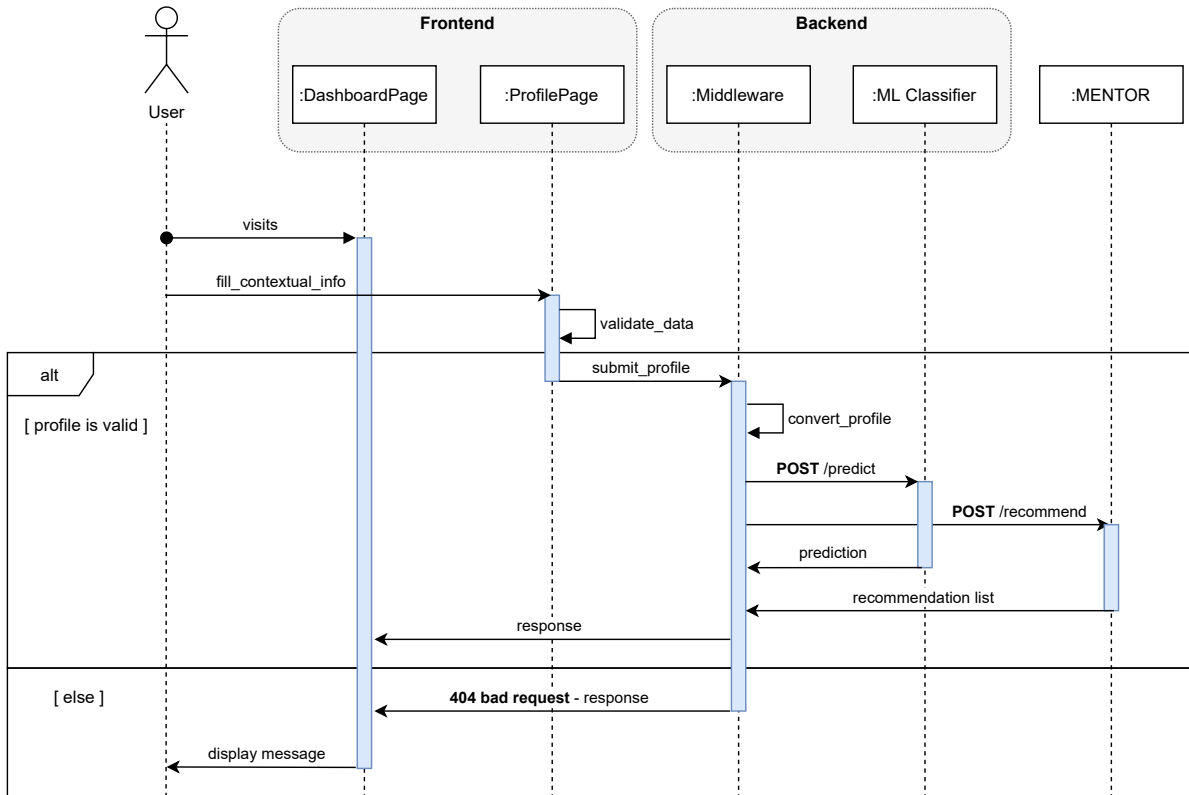


Figure 5.1: SecRiskAI sequence flow

- **Scalability & Flexibility.** React code is easier to maintain and update thanks to its modular nature. It also facilitates the creation of reusable UI components by using a particular HTML-like syntax called JSX.
- **Performance.** When the application contains a lot of user interactions or parameters that eventually result in data updates, this aspect becomes extremely important. Although modern browsers have fast performance, extensive DOM manipulation may cause slow response times and unpleasant user experiences. React directly addresses this issue by introducing a concept called “virtual DOM” along with efficient diff algorithms for identifying and effectively applying state changes to components, resulting in faster user interfaces.

Redux

In common React applications, components are usually responsible for managing the application’s state internally without requiring external tools or libraries. However, managing the states across components becomes increasingly difficult as an application scales. In this regard, an efficient state management process can be achieved by using a state management tool, such as *Redux*³, which provides a shared state across every component.

³<https://redux.js.org/>

SecRiskAI heavily relies on data tables and panels, which can be subject to changes over time and require sharing between various components. It is therefore important to have a single source of truth for data by placing it in a specific location, called “Store”. As a result, each component of the application directly accesses and modifies the data stored in the current state, which then triggers a re-rendering process of every component that is, in turn, “subscribed” to the store [118].

To update the store, Redux introduces the concept of *reducers* and *actions*. Due to the state being global and read-only, the only way of updating it is to emit an action. An action is a JavaScript object that contains useful information about a particular event [118]. The listing 5.1 below gives an example of a Redux action used in *SecRiskAI*:

```
1 export type FetchedPredictions = {
2   type: "FETCHED_PREDICTIONS";
3   predictions: Prediction;
4   loading: boolean;
5 };
```

Listing 5.1: Redux action for updating cyber-risk predictions

In general, every action contains a **type** and useful information necessary for updating the current state, such as an API call response body or user input data. In order to effectively update the global state, an action has to be dispatched. This is typically done by invoking the `dispatch()` method of the store and passing the action as parameter. Next, reducers, simple JS functions, will take the current state, the newly dispatched action and return the updated version of the state. Note that the previous state is not being overwritten. The listing below 5.2 provides implementation details of the redux state initialization used in this thesis. In lines 3-7 the global state of the application is defined. Subsequently, the reducers are combined using *combineReducers* and stored in the **reducer** constant (lines 9-13). Finally, the combined **reducer** is passed to the *createStore* function and the resulting constant (**store**) is exported (lines 15-18).

```
1 import { combineReducers, createStore } from "redux";
2
3 export type state = {
4   profile: UserProfileState,
5   predictions: PredictionState,
6   recommendations: RecommendationState
7 };
8
9 const reducer = combineReducers({
10   profile: profileReducer(),
11   predictions: predictionReducer(),
12   recommendations: recommendationReducer()
13 });
14
15 export const store = createStore(
16   reducer,
```

```

17     [...]
18 );

```

Listing 5.2: Global store initialization

As soon as an action is dispatched, the corresponding reducer will handle it properly by updating part of the state. This is typically done by the corresponding reducer, which makes use of a `switch` statement and returns the newly created state. In this prototype, actions are usually dispatched as soon as a user profile update takes place (Figure 5.4) or when the predictions are fetched from the ML Classifier service (through the Middleware) after the user navigated to the dashboard (Figure 5.2). This approach also simplifies error handling, as it allows to dispatch actions containing the error message that will in turn be displayed by the subscribed component. Such errors usually arise during external API calls, as shown by the listing 5.3 below.

For these kind of requests, *SecRiskAI* frontend uses the *Fetch API*⁴, a simple JavaScript interface for making, accessing and manipulating HTTP requests from any web browser. Fetch API enables a simpler and cleaner API design, by making use of *promises*, a concept introduced in JavaScript back in 2012 aiming to provide an alternative and easier way of working with asynchronous operations. In essence, promises are objects representing the eventual completion/failure of an asynchronous operation (*i.e.*, API calls) and can have three possible states: *pending*, *fulfilled* and *rejected*. Once created, the state of a promise is set to pending. Later, the state of the promise can change to either fulfilled (*i.e.*, the request was successful) or rejected (*i.e.*, request failed for some reason). In order to handle the resulting data/error, promises also provide a method `then()` which is triggered once the state of the promise changes.

```

1  fetch(`${getDomain()}/predict`, {...HTTP_OPTIONS(PROTOCOL_METHOD.
    POST), body: JSON.stringify(profile)})
2    .then(res => {
3      return res.json();
4    })
5    .then((response: any) => {
6      dispatch({
7        type: ActionTypes.FETCHED_PREDICTIONS,
8        predictions: response,
9        loading: false
10     });
11   })
12   .catch((error: string) => {
13     dispatch({
14       type: ActionTypes.ERROR_FETCHING_PREDICTIONS,
15       error,
16       loading: false
17     });
18   });

```

Listing 5.3: Fetching and storing risk predictions

⁴https://developer.mozilla.org/it/docs/Web/API/Fetch_API

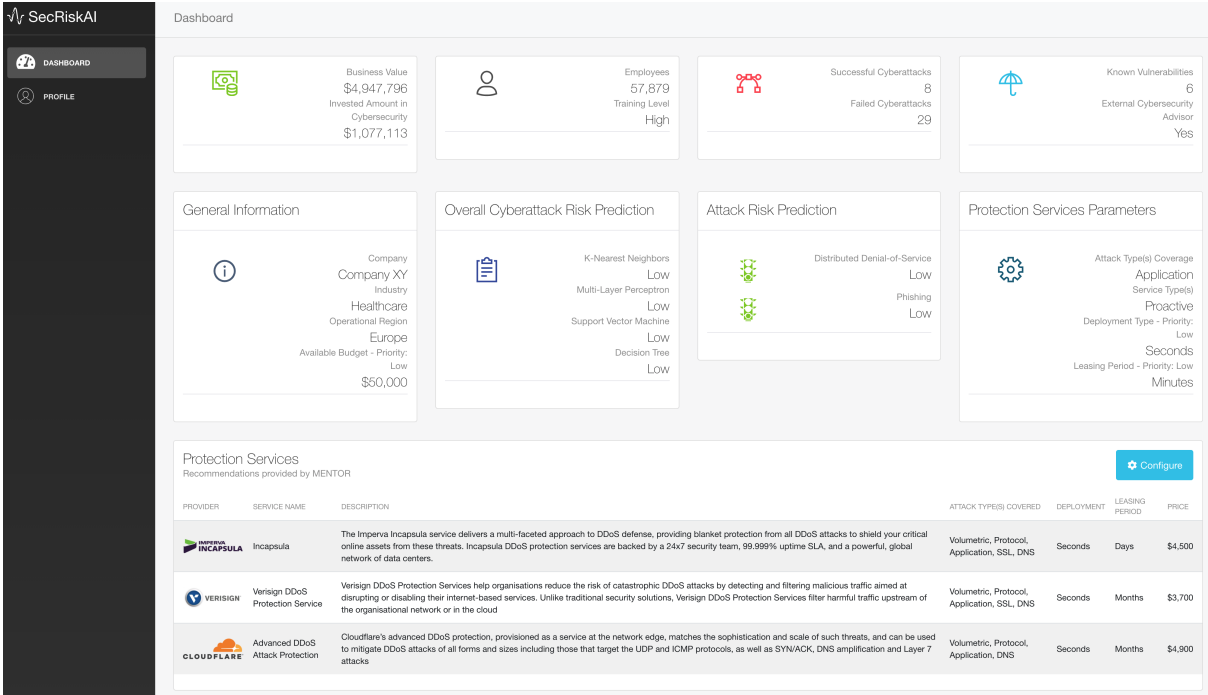


Figure 5.2: The SecRiskAI’s Dashboard

Listing 5.3 shows an example of a POST request necessary for retrieving the cyber-risk predictions from the ML Classifier. In line 3, the response is first extracted as a JSON. If the promise resolves, the predictions will be stored and the corresponding action will be dispatched (line 6-10). Otherwise, the promise will be rejected and the corresponding action containing the error message will be dispatched (line 13-17).

5.1.1 Web-based Interface

In order to visualize the different information related to cybersecurity risk assessment predictions as well as recommendations from MENTOR, an intuitive and user-friendly dashboard was designed and implemented (Figure 5.2). In the first row, the dashboard contains multiple tabs each including several pieces of contextual information, such as business value, employees, successful/failed past cyber-attacks and known vulnerabilities, already discussed in Chapter 4. Additionally, the second row gives an overview of some general information (e.g., company name, industry, operational region, etc.) as well as the cybersecurity risk assessment predictions and the desired protection services parameters used for the recommendation process.

As soon as the user accesses the dashboard, the cybersecurity risk predictions are retrieved through a POST request sent to the Middleware. As shown in Figure 5.2, *SecRiskAI* provides an overall cyberattack risk prediction, where the different ML model predictions outcomes are compared, as well as cyberattack-specific predictions, such as DDoS and phishing. It is important to note that, *SecRiskAI* is designed with flexibility in mind. This means that other cyberattack-specific ML models can be easily integrated with the

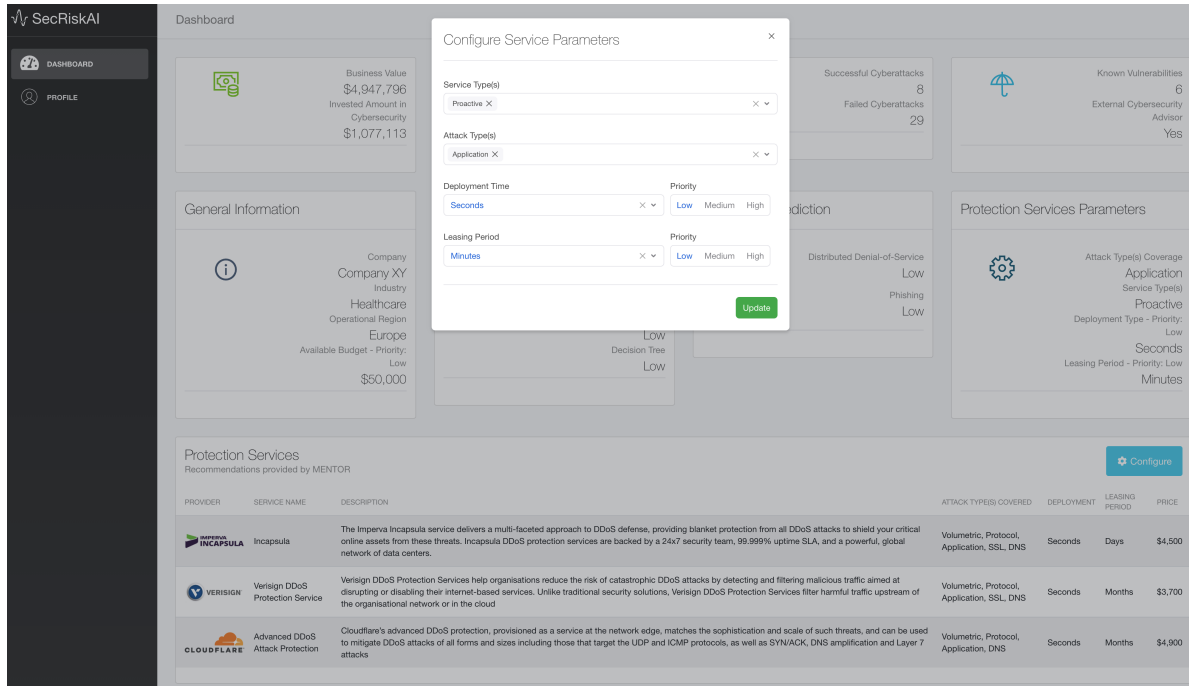


Figure 5.3: Protection service parameters

current ML Classifier service. The prediction result will then be displayed in the “Attack Risk Prediction” tab.

Additionally, this prototype also implements a table containing the recommended list of protection services most suitable for the specified profile. Each row gives an overview of a particular service with a short description, other parameters and the price. As mentioned in Chapter 4, the list is retrieved by the Middleware directly from MENTOR as soon as the dashboard is rendered. Moreover, if the user is not satisfied by the recommended services, a new recommendation process can be triggered by updating the service-specific parameters. These parameters can also be configured by clicking on the *Configure* button. Figure 5.3 shows a pop-up with the configurable parameters.

On the modal shown in Figure 5.3, the user can configure the service type (*i.e.*, proactive or reactive), desired attack types that the protection service has to cover along with the deployment time, *i.e.*, the time necessary for a service to be deployed and active, and the leasing period, *i.e.*, how long is the user willing to lease a particular service. Furthermore, a *priority* can be specified for the deployment time and leasing period in order to achieve a better personalized recommendation.

The contextual information required for the cyber-risk prediction process can be changed at any time, by clicking on the *Profile* tab on the sidebar, as shown by Figure 5.4. The user profile page is subdivided into two sections: general information and technical details. The former contains general information about the company, such as name, industry, operational region, yearly revenue, total number of employees and the available cybersecurity budget. The latter comprises information which typically would result from external tools and reports. Examples of required fields are: invested amount in cybersecurity, known

The screenshot shows the 'User profile page' in the SecRiskAI application. The interface includes a sidebar with 'DASHBOARD' and 'PROFILE' tabs. The 'PROFILE' tab is selected, leading to a 'Profile' section. This section is divided into 'General Information' and 'Technical Details'. The 'General Information' section has two tabs: 'Scenario 1 (DDoS & MITM)' and 'Scenario 2 (Phishing)'. It contains several input fields and dropdown menus for user information: Company (Company XY), Industry (Healthcare), Operational Region (Europe), Business Value (Revenue) (\$ 4,947,796), Number of Employees (57,879), Employee Training (Low, Medium, High), Cybersecurity Budget (\$ 50,000), and Priority (Low, Medium, High). The 'Technical Details' section contains fields for Invested Amount in Cybersecurity (\$ 1,077,113), Known Vulnerabilities (6), External Advisor (Yes), Successful Past Attacks (8), and Failed Past Attacks (29). An 'Update' button is located at the bottom right of the form.

Figure 5.4: User profile page

vulnerabilities, external advisor and failed/past attacks. A detailed explanation of every field present on the user profile page is found in Chapter 4.

5.2 Backend

SecRiskAI's backend comprises two components: Middleware and ML Classifier. Both the Middleware and the ML Classifier are applications designed to be deployed independently and communicate through a REST API. On the other hand, the ML workflow includes every step already discussed in Chapter 4 in an interactive computing environment, that enables the data scientists to run code, generate plots and document each step using text. The current prototype is also designed to be cloud agnostic, meaning that the entire architecture can be easily migrated to and from any on-premise infrastructure or cloud platform, regardless of the underlying operating system and other dependencies.

5.2.1 Middleware

The Middleware layer was implemented using *Nest.js*, a progressive Node.js framework for building efficient, reliable and scalable server-side applications [122]. In contrast with React, the library used for implementing the frontend, Nest.js has adopted TypeScript by default however still preserving compatibility with plain JavaScript. In essence, Nest.js aims to provide an out-of-the-box application architecture, allowing teams to design and implement highly testable, scalable and loosely coupled applications.

In this prototype, the Middleware serves the purpose of translating incoming request from the client to a format required by the ML Classifier for making predictions. Additionally, the Middleware is also in charge of requesting a list of recommended protection services from MENTOR through a dedicated REST endpoint. In order to receive and handle client-side requests, Nest.js introduces the concept of controllers. Listing 5.4 shows the technical implementation of the Middleware's controller.

```
1 @Controller()
2 export class MiddlewareController {
3     constructor(private readonly middlewareService:
4         MiddlewareService) {}
5
6     @Post("/predict")
7     async predict(@Body() profile: UserProfile): Promise<
8         PredictionResponse> {
9         let prediction = await this.middlewareService.predict(
10             profile);
11         return prediction.data
12     }
13
14     @Post("/recommend")
15     async recommend(@Body() recommendationProfile:
16         RecommendationProfile): Promise<RecommendationResponse> {
17         let recommendation = await this.middlewareService.
18             recommend(recommendationProfile);
19         return recommendation.data.recommendedServices
20     }
21 }
```

Listing 5.4: Middleware controller

In Nest.js, a controller is basically a class with specific metadata provided by decorators. To create a controller, `@Controller()` decorator is used (line 1). This allows the creation of a routing mechanism, where the framework takes care of which requests are processed by which controller, based on the path passed to the decorator. In the Middleware controller (Listing 5.4) no path was declared in the decorator, hence it is used as a default route for every incoming request.

Currently, the controller is able to handle two types of incoming POST requests having the route match one of the following patterns: `/predict` (line 5) or `/recommend` (line 11). The handler for those endpoints is created automatically by Nest.js, once the `@Post()` decorator is specified. Both routes must be explicitly bound to different functions (line 6-9 and 12-15) that will be invoked once a request from the client hits the server. The actual business logic for handling incoming request is encapsulated in a Nest class called service. The request body is then passed to the respective function of the service, as shown in 5.4 (line 7 and 13). Finally, the response data is sent back to the client (line 8 and 14).

Besides controllers, providers are also considered to be a fundamental concept in Nest. The main idea behind a provider is to *inject* dependencies, meaning that the runtime system itself takes care of managing the creation of objects along with their corresponding

relationships. This prototype makes use of services, a type of provides offered by Nest, to define logic for processing client requests. Listing 5.5 describes the technical details of the service associated with Middleware controller.

```

1  @Injectable()
2  export class MiddlewareService {
3
4      constructor(private httpService: HttpService) {}
5
6      predict(body: UserProfile): Promise<any> {
7          return this.httpService.post(`${process.env.ML_SERVER_URL}
8              /predict`, {
9              "cyberattackPredictionProfile": this.
10                 toCyberattackPredictionRequest(body),
11              "ddosPredictionProfile": this.toDDoSPredictionRequest(
12                 body)
13          }).toPromise();
14      }
15
16      recommend(body: RecommendationProfile): Promise<any> {
17          return this.httpService.post(`${process.env.MENTOR_URL}/v1
18              /recommend`, body).toPromise();
19      }
20
21      toCyberattackPredictionRequest = (body: UserProfile): string
22      => {
23          return `[${body.investedAmount}, ${body.successfulAttacks
24              }, ${body.failedAttacks}, ${body.businessValue}, ${body
25              .nrEmployees}, ${Levels[body.employeeTraining]}, ${body
26              .knownVulnerabilities}, ${Advisor[body.externalAdvisor
27              ]}]`;
28      }
29
30      toDDoSPredictionRequest = (body: UserProfile): string => {
31          return `[${Industry[body.industry]}, ${Regions[body.
32              region]}, ${body.investedAmount}, ${body.
33              successfulAttacks}, ${body.failedAttacks}, ${body.
34              businessValue}, ${body.knownVulnerabilities}, ${Advisor
35              [body.externalAdvisor]}]`;
36      }
37  }

```

Listing 5.5: Middleware service

Similar to the controller implemented in Listing 5.4, the Middleware service presented in Listing 5.5 is also implemented as a basic TypeScript class. The only difference is that it uses the `@Injectable()` decorator instead and is managed by Nest Inversion-of-Control container. The Middleware service uses the `HttpService` to make external calls to the services and it is imported using constructor injection (line 4). Before sending the prediction

request to the ML Classifier, the actual body is constructed using `toCyberattackPredictionRequest` and `toDDoSPredictionRequest` functions. This step is essential as the current implementation of ML Classifier requires a specific format as an input, where each attribute of the user's profile is used to construct a string. Listing 5.6 gives an example of the constructed request body.

```

1 {
2   cyberattackPredictionProfile: '[[1077113, 8, 29, 4947796, 57879,
3     2, 6, 1]]',
4   ddosPredictionProfile: '[[1, 0, 1077113, 8, 29, 4947796, 6, 1]]'
```

Listing 5.6: Prediction request body

Finally, a POST request including the body as JSON format is sent to the ML Classifier. The variable `ML_SERVER_URL` stores the actual base url of the ML Classifier and is typically stored in an `.env` file and can be accessed as shown in line 7 (`process.env`). To perform the actual request, the `/predict` path is attached to the base url. For the recommendation process, a POST request containing the user's profile is sent to MENTOR's API, using the base url (`MENTOR_URL`) stored in the same environment file and attaching the required path (`/v1/recommend`).

5.2.2 ML Classifier

The ML Classifier is implemented using BentoML, a flexible, high-performance framework written in Python for serving, managing and deploying ML models. More specifically, BentoML supports various ML frameworks, cloud native deployment and offers an high-performance online API serving [123]. In the proposed solution, the ML Classifier makes use of the full potential of BentoML for deploying and serving trained ML models efficiently and effectively. Listing 5.7 provides the implementation details of the prediction service (also known as BentoService) used in *SecRiskAI*.

```

1 prediction_result_mapping = {0: "LOW", 1: "MEDIUM", 2: "HIGH"}
2
3 @env(infer_pip_packages=True)
4 @artifacts([SklearnModelArtifact("knn_model"),
5     SklearnModelArtifact("mlp_model"),
6     SklearnModelArtifact("svm_model"),
7     SklearnModelArtifact("tree_model"),
8     SklearnModelArtifact("scaler"),
9     SklearnModelArtifact("ddos_mlp_model"),
10    SklearnModelArtifact("ddos_scaler")])
11 class RiskClassifier(BentoService):
12     """
13     A prediction service exposing Scikit-learn models for
14     Cybersecurity Risk Assessment
```



```

14     """
15
16     @api(input=JsonInput())
17     def predict(self, body):
18         """
19         An inference API named 'predict' with Dataframe input
20         adapter, which codifies
21         how HTTP requests or CSV files are converted to a pandas
22         Dataframe object as the
23         inference API function input
24         """
25
26         cyberattack_df = pd.DataFrame(eval(body["
27             cyberattackPredictionProfile"]))
28         normalized_cyberattack_df = self.artifacts.scaler.
29             transform(cyberattack_df)
30
31         normalized_ddos_df = self.artifacts.ddos_scaler.transform(
32             pd.DataFrame(eval(body["ddosPredictionProfile"])))
33
34         return {
35             "overall_risk_prediction": {
36                 "knn": prediction_result_mapping[self.artifacts.
37                     knn_model.predict(normalized_cyberattack_df)
38                     [0]],
39                 "mlp": prediction_result_mapping[self.artifacts.
40                     mlp_model.predict(normalized_cyberattack_df)
41                     [0]],
42                 "svm": prediction_result_mapping[self.artifacts.
43                     svm_model.predict(normalized_cyberattack_df)
44                     [0]],
45                 "dtree": prediction_result_mapping[self.artifacts.
46                     tree_model.predict(cyberattack_df)[0]]
47             },
48             "ddos_risk_prediction": prediction_result_mapping[
49                 self.artifacts.ddos_mlp_model.predict(
50                     normalized_ddos_df)[0]]
51         }
52

```

Listing 5.7: Risk prediction service

Similar to the Middleware controller discussed in the previous section, a `BentoService` is a basic class with additional decorators. The `@env()` decorator along with the `infer_pip_packages=True` option is used by BentoML to automatically figure out the PyPI packages required by the prediction service class. The `@artifacts()` decorator is used to reference and import the ML models and scalers required by the service. Once a ML model is trained and ready to be deployed, the risk prediction service is bundled with the model by invoking the `pack()` method on the newly created instance, as shown by the Listing 5.8. More specifically, the name of the declared model artifact along with the

target location, *i.e.*, where the model is currently stored (either locally or on S3⁵ bucket) and some model metadata are required (line 3-7).

```

1 risk_classifier_service = RiskClassifier()
2
3 risk_classifier_service.pack("knn_model", load("src/models/
4     KNN_classifier.joblib"), metadata = {
5         "accuracy": "96.40%",
6         "n_neighbors": 23,
7         "weights": "distance"
8     })
9 [...]

```

Listing 5.8: ML artifacts import

The name passed as a parameter in Listing 5.8 (line 3) is then used as a reference by the `@artifacts()` decorator (Listing 5.7). The current implementation of `RiskClassifier` contains only a `predict()` method. The `@api()` decorator is also needed for specifying the API entry point for client requests. In this case, any `/predict` request will invoke the `predict()` method, with the request body as a method parameter. Moreover, the option `input=JsonInput()` indicates that the input body should be a JSON. Using pandas, a fast, powerful and flexible open source data analysis/manipulation library, the request body is transformed to a DataFrame [124]. As described in Listing 5.7, this step is performed in line 24 for the overall risk prediction and line 28 for DDoS risk prediction. Next, depending on the ML model, data normalization may be required (Chapter 5). Line 25 and 27 describe the process of normalizing the DataFrame by invoking the `transform()` method of the scaler artifact. Finally, the dataframe is passed as parameter to the `predict()` method of the ML models and the prediction response is built. Listing 5.9 shows an example of a prediction response constructed by the ML Classifier. For the overall risk prediction, every model presented in Chapter 4 is implemented. On the other hand, for the DDoS risk prediction a MLP model is trained and used by default.

```

1 {
2     "overall_risk_prediction":{
3         "knn":"LOW",
4         "mlp":"LOW",
5         "svm":"LOW",
6         "dtree":"LOW"
7     },
8     "ddos_risk_prediction":"LOW"
9 }

```

Listing 5.9: Cyber-risk prediction response

⁵<https://aws.amazon.com/s3/>

Monitoring API

In addition to the prediction API, the ML Classifier is also designed to provide a basic monitoring API used by data scientists to mainly ensure that system is performing as intended. The current implementation provides an `GET /healthz` endpoint for checking the health of the system and returns an empty response with status code 200 if the system is operating correctly. Moreover, the metadata of the deployed ML models is accessible by making a `GET /metadata` endpoint. Listing 5.10 gives an example of a response containing some metadata.

```
1 {
2   "name": "RiskClassifier",
3   "version": "20210218233931_A55EA7",
4   "createdAt": "2021-06-20T21:17:00.490672Z",
5   "env": {
6     ...
7   },
8   "artifacts": [
9     {
10      "name": "knn_model",
11      "artifactType": "SklearnModelArtifact",
12      "metadata": {
13        "accuracy": "96.59%",
14        "n_neighbors": 28.0,
15        "weights": "distance"
16      }
17    }
18    ...
19  ],
20  "apis": [
21    {
22      "name": "predict",
23      "inputType": "JsonInput",
24      ...
25    }
26  ]
27 }
```

Listing 5.10: Metadata response

The metadata response contains some information about the current deployment of the ML Classifier as well as some detailed information of the served artifacts. This includes details about the accuracy and the parameter values of the deployed models (line 12-16) as well as the exposed API (line 20-26). Additionally, a `GET /metrics` endpoint is provided to retrieve useful information regarding the performance of the system, namely how long it takes for a prediction request to be processed. Besides that, data scientists are also able to provide a feedback to the prediction response by sending a request to the `POST /feedback` endpoint, including in a JSON body the id of the prediction response and the

feedback as a string. Every feedback is then stored and could be used to improve the accuracy and overall performance of the ML Classifier.

5.3 ML Workflow

The process of generating data, training and testing ML models is documented using Jupyter Notebook, an open-source web application that allows the creation of documents, also known as notebooks, which contain live code (usually Python), equations, visualizations and plain text [125]. The ML models presented in Chapter 4 were implemented using scikit-learn, a very popular open-source library for predictive data analysis and ML [126]. Table 5.1 provides an overview of the ML Classifiers and their corresponding implementation class in scikit-learn.

Table 5.1: ML Models and corresponding Scikit-learn implementation [126]

ML Model	Scikit-learn Implementation
DT	sklearn.tree.DecisionTreeClassifier [127]
KNN	sklearn.neighbors.KNeighborsClassifier [128]
SVM	sklearn.svm.SVC [129]
MLP	sklearn.neural_network.MLPClassifier [130]

As described in Chapter 4, a key requirement for training and validating ML models is having enough data at hand. The data gathering process is therefore a fundamental step. As already mentioned, collecting data directly was not feasible, therefore a synthetic data generation approach was adopted. For this reason, a data generator using Python was implemented. As shown Listing 5.11, the current implementation can be fully parameterized, by passing the required boundaries as well as standard deviation and average of other parameters, such as business value.

```

1 def generate_data(nr_entries = 1000, min_empl = 30, max_empl =
    10000, min_nr_attacks = 0, nr_attacks = 50,
2         avg_business_value = 5000000, std_business_value
    = 50000, max_invested_perc = 0.3,
    max_nr_vulnerabilities = 10):
3     df = pd.DataFrame(columns = columns)
```

```

4     for i in tqdm(range(0, nr_entries)):
5         nr_employees = random.randint(min_empl, max_empl)
6         employees_training = random.choice(LEVELS)
7
8         failed_attack = random.randrange(nr_attacks)
9         succ_attack = random.randrange(nr_attacks)
10
11        business_value = int(numpy.random.normal(loc =
12            avg_business_value, scale = std_business_value))
13        invested_perc = random.uniform(0, max_invested_perc)
14        invested_amount = int(invested_perc * business_value)
15
16        known_vulnerabilities = random.randrange(
17            max_nr_vulnerabilities)
18        external_adv = random.choice(ADVISOR)
19
20        # Risk is computed based on the generated attributes
21        computed_risk = invested_perc - (succ_attack / nr_attacks)
22            + (nr_employees / max_empl) * LEVELS.index(
23                employees_training) - (known_vulnerabilities /
24                    max_nr_vulnerabilities) + ADVISOR.index(external_adv)
25
26        df.loc[i] = [invested_amount, succ_attack, failed_attack,
27            business_value, nr_employees, employees_training,
28            known_vulnerabilities, external_adv,
29            get_categorized_risk(computed_risk)]
30    return df
31
32def get_categorized_risk(weighted_risk, upper_boundary = 1.0,
33    lower_boundary = 0.0):
34    if weighted_risk >= upper_boundary:
35        return "LOW"
36    elif weighted_risk >= lower_boundary and weighted_risk <
37        upper_boundary:
38        return "MEDIUM"
39    else:
40        return "HIGH"

```

Listing 5.11: Data generator

Furthermore, the current generator also allows to specify the number of entries to be generated, by passing a number to the corresponding `nr_entries` parameter. The list of available parameters, such as number of employees and corresponding training level, successful/failed attacks (full list can be found in Chapter 4), is then generated according to the boundaries passed as parameters to the `generate_data()` function. The resulting risk (`computed_risk`) is generated according to the formula already discussed in Chapter 4 and the technical implementation is found in line 19. Next, the computed risk (`computed_risk`) is categorized according to the range already specified in Chapter 4 and implemented by the function `get_categorized_risk()` (line 24-30). Finally, the pandas

library is used to construct a DataFrame containing the generated rows and it is stored in a `data` variable. Table 5.2 shows an example of a returned DataFrame.

Table 5.2: Generated Dataset

Invested Amount	Successful Attacks	Failed Attacks	Business Value	Number of Employees	Employee Training	Known Vulnerabilities	External Advisor	Risk
1147297	38	5	5058467	52608	MEDIUM	0	NO	MEDIUM
1320806	29	15	4909813	88397	LOW	3	YES	HIGH
719662	46	19	4944962	53330	MEDIUM	3	NO	HIGH
794910	39	39	4989944	69107	MEDIUM	5	YES	HIGH

As described in Chapter 4, once generated, the dataset is subjected to a processing phase, there each entry is categorized and, based on the ML algorithms, a normalization step may also be required. Listing 5.12 illustrates the process of categorizing each entry in the generated dataset. Initially, a mapping for each category is defined using Python dictionaries, as indicated in lines 1 & 2. Next, the values of columns “Employee Training” and “Risk” are categorized using the `replace()` function provided by pandas.

```

1 levels_mapping = { "LOW": 0, "MEDIUM": 1, "HIGH": 2 }
2 advisor_mapping = { "NO": 0, "YES": 1 }
3
4 data = data.replace({"Employee Training": levels_mapping, "Risk":
    levels_mapping, "External Advisor": advisor_mapping})

```

Listing 5.12: Dataset categorization

After the categorization phase, the dataset is then divided into training and test set, following the approach described in Chapter 4. As indicated by line 1 in Listing 5.13, the parameter defining the test size is set to `0.2`, indicating that the resulting split will be 80% training and 20% test. The current implementation uses the `train_test_split()` function imported from `sklearn.model_selection`.

```

1 X_train, X_test, y_train, y_test = train_test_split(data[features
    ].values, data["Risk"].values, test_size = 0.2)
2
3 scaler = MinMaxScaler()
4 scaler.fit(X_train)
5
6 X_train_normalized = scaler.transform(X_train)
7 X_test_normalized = scaler.transform(X_test)

```

Listing 5.13: Dataset split and normalization

In a next step, an instance of a Min-Max scaler is created (line 3) and is fitted on the train dataset (line 4) to compute the mean and standard deviation required for later scaling. Similarly, the implementation for the scaler is provided by scikit-learn and is imported

from `sklearn.preprocessing`. Since the majority of the models in this thesis require normalized data for the training process, the `transform()` method of the Min-Max scaler is invoked and both train and test data are passed as parameters (line 6 & 7). In order to be re-used in the ML Classifier, the scaler is further exported as a `.joblib` artifact and packed by the RiskClassifier service (Listing 5.8). The goal is to normalize any incoming risk prediction profile in order to increase prediction accuracy and overall reliability. Next, each ML model with the corresponding implementation described in Table 5.1 is selected and instantiated. Listing 5.14 illustrates the process of hyperparameter tuning, training and CV a DT model. It is important to note that the same workflow applies to the other ML models used in this work.

```

1 dtree = DecisionTreeClassifier()
2
3 # Define the parameter ranges that should be tested on the model
4 dtree_params = {
5     "criterion": ["gini", "entropy"],
6     "max_depth": range(1, 10),
7     "min_samples_split": range(2, 10),
8     "min_samples_leaf": range(1, 10)
9 }
10
11 # Instantiate the Grid for parameter tuning
12 dtree_grid = GridSearchCV(dtree, dtree_params, cv = 5, scoring = "
    accuracy", n_jobs = -1)
13
14 dtree = dtree_grid.fit(X_train, y_train)
```

Listing 5.14: Example of DT hyperparameter tuning, training and CV

Initially a DT is instantiated and stored in the `dtree` variable (line 1). Next, a dictionary containing the parameters required to construct a model along with a list of possible values is created and stored in the `dtree_params` variable (line 4-9). For the DT in particular, a `criterion` used to measure the quality of the data split may be either `gini` or `entropy`. The `max_depth` parameter is also required for defining the maximum depth of the tree. In the `dtree_params` grid, a sequence of numbers from 1 to 10 for `max_depth` is given. Moreover, parameters such as `min_samples_split` and `min_samples_leaf` can also be specified. The former indicates the minimum number of samples required to split an internal node and the values used in the hyperparameter tuning range from 2 to 10. The latter stands for the minimum number of samples required to be at a leaf node and similar to the `max_depth`, a range of values from 1 to 10 is predefined.

As discussed in Chapter 4, this prototype uses the grid search technique for hyperparameter tuning combined with a 5-fold CV. Using the scikit learn library, this process can be easily combined by creating an instance of `GridSearchCV`. More specifically, this class is used to execute an exhaustive search over specified parameter values of a particular ML model or estimator. As indicated in line 12, the `GridSearchCV` accepts as parameter the estimator object (`dtree`) and a list or dictionary of parameter names as keys and lists of parameter settings to try as values (`dtree_params`). Moreover, one can also specify the

`cv` parameter, determining the number of folds used in the CV strategy as well as the scoring strategy used to evaluate the performance of the ML model on the test set. In this prototype, the number of folds for the CV is set to 5 and the scoring to “accuracy”. The last parameter, that is `n_jobs`, is set to -1 meaning that the jobs are run using all processors. Subsequently, the model is then fitted on the train dataset (line 14) and evaluated on the test set. Lastly, the DT model is exported as a `.joblib` file using the `dump()` function imported from the `joblib` library.

Chapter 6

Quantitative Evaluation

This Chapter presents a quantitative evaluation of *SecRiskAI*, a platform for ML-based cybersecurity risk assessment. More specifically, various metrics required for evaluating the performance and effectiveness of the system are presented. Moreover, a performance and accuracy comparison of the ML algorithms used in this thesis is provided. The following quantitative evaluation was conducted on a machine using the Apple M1 System on a Chip (SoC) and 16 GB of RAM.

In order to design and implement *SecRiskAI*, four different ML algorithms have been trained, tuned and thoroughly tested, as described in Chapter 4 and 5. For a quantitative evaluation and comparison of the various ML models, several performance metrics were observed, such as accuracy, precision, recall and F1-Score. The generation of these metrics is also a very important step in every ML workflow for understanding the behaviour and performance of the implemented models.

The confusion matrix is the most popular technique for evaluating the correctness and accuracy of classification models [131]. In practice, confusion matrices can be used for both binary and multi-class classification problems and provide an easier way of assessing and comparing the performance of classification models. In order to compute the confusion matrix, a dataset containing 50,000 entries was generated and a 80-20 train-test split strategy was followed, as described in Chapter 4. After a training and CV phase, each ML model was tested using the remaining test set. The purpose of this phase is to test the model on previously unseen data, *i.e.*, data not used during the training phase. For each entry in the test set, the ML model is used to predict the corresponding class. Finally, the predicted labels are compared with the actual class, also called true label, and as result a confusion matrix is constructed. Figure 6.1 shows the confusion matrices generated for each ML algorithm implemented in this thesis.

A confusion matrix is essentially a summary of prediction results where each cell corresponds to the number of correct and incorrect predictions broken down by predicted/true label combination. Ideally, a best-performing classifier would result in a confusion matrix where only the diagonal is filled with values, meaning that every predicted class corresponds also to the actual label. In that case, the model would have achieved an accuracy of 100%. In other words, the accuracy of a model is calculated as the number of correctly

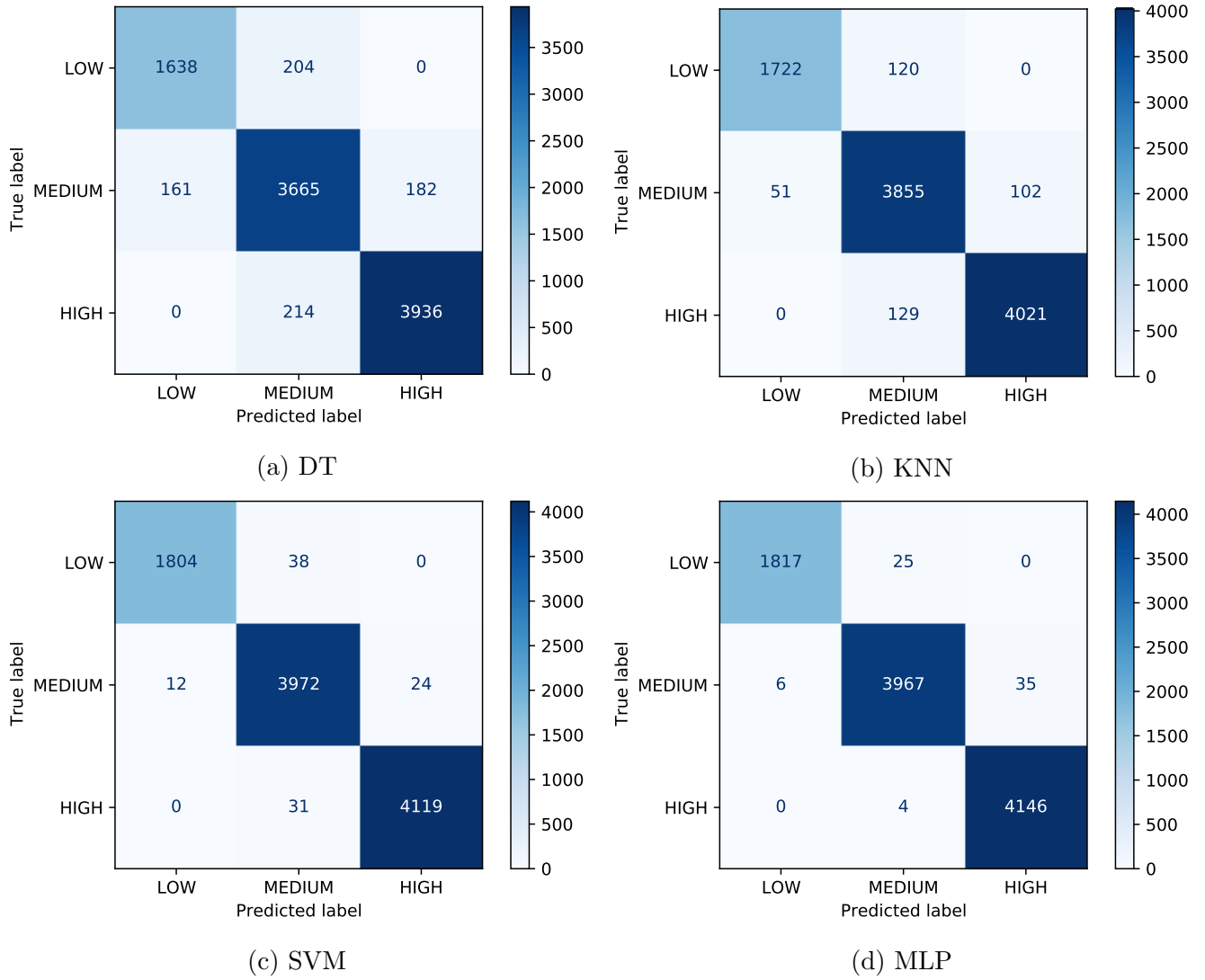


Figure 6.1: Confusion matrices

predicted classes divided by the incorrect predictions. As shown in Figure 6.1, for this particular prototype, every model was able to achieve more than 90% accuracy. However, it can be clearly observed by the cells near the diagonal that the DT and KNN scored slightly worst than SVM and MLP.

Table 6.1 shows the computed performance metrics, based on the generated dataset with 50,000 entries. Each model was trained and tuned to maximise accuracy, reduce overfitting and provide better results. SVM and MLP achieved similar accuracy scores, however in terms of computation time, the difference is substantial. As for the training phase, the SVM requires approximately half the time compared to the MLP model. The training time has also an impact on the grid search computation time, a hyperparameter technique already discussed in Chapter 4, which for the MLP model exceeds 200 seconds, as every tuned model undergoes a 5-fold CV. On the other hand, the DT and KNN had the fastest training time and KNN achieved the fastest grid search computation time of around 40 seconds.

Table 6.1: Performance metrics

ML Model	Accuracy	Training Time (s)	Grid Search Computation (s)
DT	92.64%	0.18	146.77
SVM	99.03%	5.83	149.15
KNN	95.82%	0.08	40.06
MLP	98.86%	10.53	210.55

Based on the confusion matrices presented in Figure 6.1, other important metrics such as precision, recall and F1-score can be derived as well [132]. The precision metric is used to express the proportion of units labeled by a model that actually belong to that class. As shown in Figure 6.1a, the DT was able to predict a “Low” risk for 1638 profiles out of all predicted profiles ($1638 + 161 + 0$), resulting to a precision of $1638 / 1799 \approx 91\%$.

Additionally, the recall metric quantifies a model’s predictive accuracy for a particular class, *i.e.*, it represents the ability of a model to find all entries in a dataset that belong to a particular output class. As presented in Figure 6.1a, out of 1842 ($1638 + 204 + 0$) profiles with “Low” as a true label, the DT was only able to classify 1638 correctly, resulting in a $\approx 89\%$ ($1842 / 1638$) recall.

The last performance metric considered in this evaluation is the F1-score, which ranges between 0 and 1. This metric aggregates both precision and recall by computing the so-called harmonic mean [133] and is used to compare ML models with different precision and recall in order to determine which one produces the best results. Similar to precision and recall, the F1-Score is computed for each output class. Table 6.2 gives an overview of the derived performance metrics calculated for each ML model.

Additionally, the computed performance metrics summarized in Table 6.2 reveal that the MLP, despite having a marginally lower accuracy than SVM, was able to achieve an F1-Score of 1.0 for the “High” output class. The MLP also outperformed marginally the SVM in both precision and recall scores. The small performance gain comes at cost

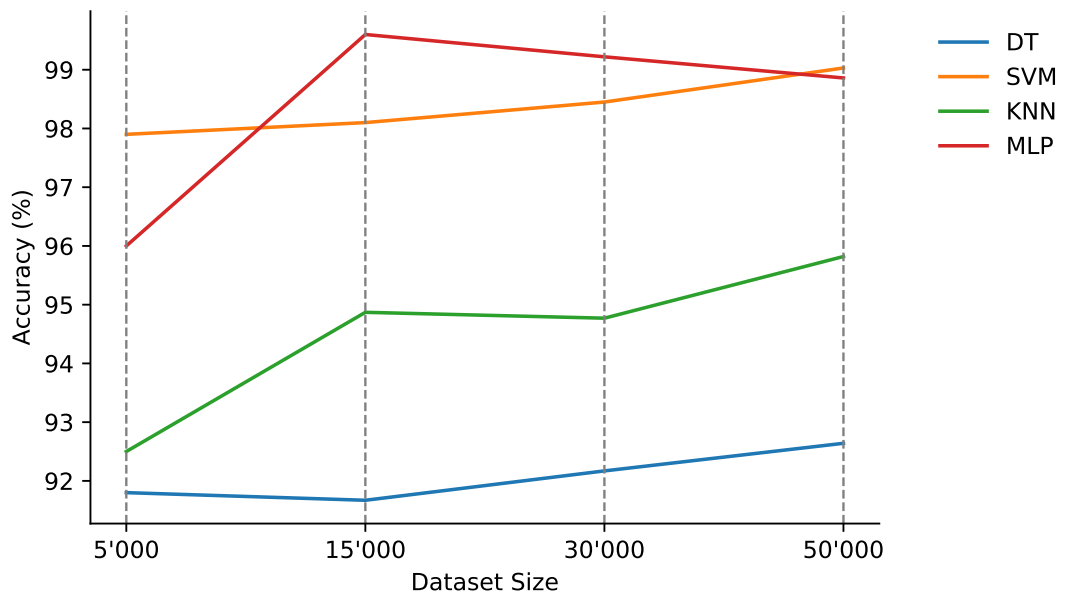
Table 6.2: Computed Precision, Recall & F1-Score for each ML model

ML Model	Class	Precision	Recall	F1-Score
DT	Low	0.91	0.89	0.90
	Medium	0.90	0.91	0.91
	High	0.96	0.95	0.95
SVM	Low	0.99	0.98	0.99
	Medium	0.98	0.99	0.99
	High	0.99	0.99	0.99
KNN	Low	0.97	0.93	0.95
	Medium	0.94	0.96	0.95
	High	0.98	0.97	0.97
MLP	Low	1.00	0.99	0.99
	Medium	0.99	0.99	0.99
	High	0.99	1.00	1.00

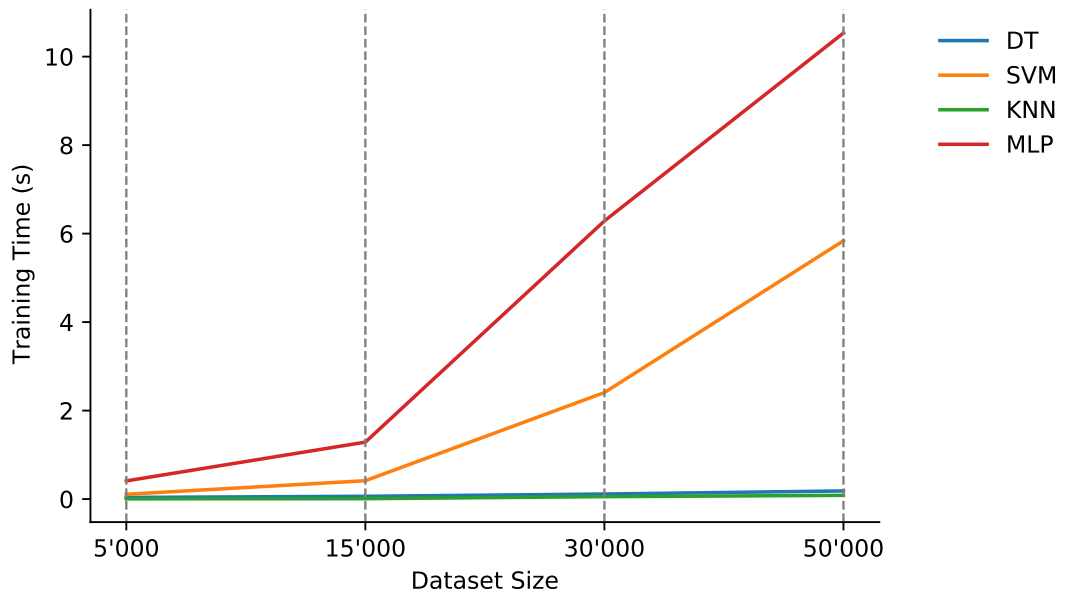
of training time which, according to Table 6.1 and also demonstrated by other studies [134, 135], is generally higher compared to SVP classifiers. This is mainly due to the higher complexity of the MLP algorithm, as described in Chapter 2. The other two ML models used by *SecRiskAI*, namely the DT and KNN, were also able to achieve still fairly high precision, recall and F1-Score. However, similar to the accuracy scores presented in Table 6.1, the metrics presented in Table 6.2 confirmed once again that the DT provided the worst performance despite having faster training times.

Finally, the impact of the synthetic dataset sizes on both accuracy and training time was investigated. First, datasets of different sizes were generated following the algorithm introduced in Chapter 5 (*cf.* 5.11). Each ML model was equally trained and tuned on every generated dataset using grid search followed by a 5-fold CV technique. The results were documented and visualized using line charts as shown in Figure 6.2. For small to medium sized datasets (*i.e.*, between 5,000 and 15,000), the MLP model is able to outperform every other model with an accuracy of almost 100% (Figure 6.2a). Moreover, the impact on the training time is also relatively low, with the MLP requiring approximately 2 seconds. However, the outcome is different once the size of the dataset increases. On one hand, the training time for both SVM and MLP increase drastically, which for the MLP leads to a $\approx 388\%$ increase of training time with double the dataset size. On the other hand, as highlighted by Figure 6.2b, the accuracy of the MLP model suffers a slight decrease while having the highest accuracy score among the other ML models.

Once the generated dataset size reaches over 50,000 entries, the MLP starts to perform worst than the SVM while requiring twice as much time to be trained. While this may not seem a big difference in terms of seconds, with datasets exceeding millions of entries, the gap may become even more substantial, leading to extremely slow model training and poor scalability. Furthermore, from Figure 6.2a it can be observed, that with larger dataset sizes the SVM had a minimal but constant increase in accuracy. Similarly, the KNN and DT also experienced an accuracy gain while maintaining a low training time. Therefore, based on the Figure 6.2, it is possible to conclude that MLPs really exceed



(a) Dataset size and accuracy



(b) Dataset size and training time

Figure 6.2: Dataset size evaluation

with medium-sized datasets and SVMs should be taken into consideration when dealing with large datasets.

Chapter 7

Qualitative Evaluation

In this Chapter, a qualitative evaluation of the solution is presented. More specifically, the usability and reliability of *SecRiskAI* are evaluated and documented with the help of three case studies based on real-world scenarios, aiming to cover different features and prove the effectiveness of the cybersecurity risk prediction. Also, the challenges and limitations of the proposed solution are highlighted and discussed at the end of this chapter.

7.1 Case Study #1 - DDoS Attack


In this case study, the ability of *SecRiskAI* to assess DDoS risk is investigated and analyzed. For this purpose, a ML model needs to be trained on a different dataset than the ones used in the quantitative evaluation. In fact, the synthetic data generation process (*cf.* Chapter 4) had to be adjusted in order to generate a different set of attributes that have a direct impact on the likelihood of a company being targeted by DDoS attacks. After extensive research [136, 137, 139], attributes indicating the industry and the operative region as part of contextual information were defined. Other attributes, such as number of employees and employee training, were discarded and not included in the generation step, as the impact of those on the DDoS risk could not be proved by other studies. Consequently, based on the qualitative evaluation previously presented, a dataset of 30,000 entries was generated and the MLP was chosen as a suitable model for predicting DDoS risk.

As for the contextual information, it was assumed that the company interested in assessing the DDoS risk was operating in the E-Commerce sector, buying and selling various types of goods over the internet and mostly focusing on the European market. Moreover, the number of employees amounts to nearly 10,000 and their training level, also understood as “awareness level”, for cybersecurity-related topics was classified as “low”. As shown in Figure 7.1, other general information include a business value of around 5 Million US\$ and a cybersecurity budget of just 50,000 US\$. In this particular case study, the cybersecurity budget is intended to be used in either protection services of proactive and/or reactive nature or other investments aiming to increase DDoS resiliency.

General Information

Case Study #1Case Study #2

COMPANY

 E-Shop

INDUSTRY

E-Commerce

×

▼

OPERATIONAL REGION

Europe

×


▼

BUSINESS VALUE (REVENUE)

\$

5,000,000

NUMBER OF EMPLOYEES

 10,000

EMPLOYEE TRAINING

LOW

MEDIUM

HIGH

CYBERSECURITY BUDGET

\$

50,000

PRIORITY

LOW

MEDIUM

HIGH

Technical Details

INVESTED AMOUNT IN CYBERSECURITY

\$

130,000

KNOWN VULNERABILITIES

 2

EXTERNAL ADVISOR

No

×

▼

SUCCESSFUL PAST ATTACKS

 4

FAILED PAST ATTACKS

 15

Update

Figure 7.1: E-Shop contextual information

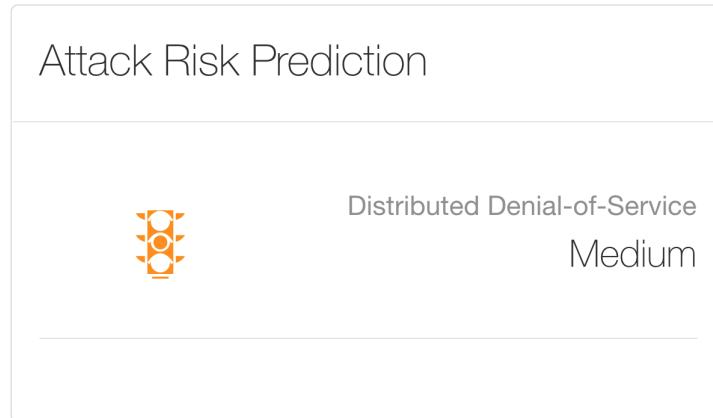


Figure 7.2: DDoS risk prediction


In order to make an accurate prediction, besides the general information, the ML classifier requires some technical details as well. More specifically, the company has to provide the amount of US\$ already invested in cybersecurity as well as any known vulnerabilities, which, as discussed in Chapter 4, may derive from third-party security tools (*e.g.*, Nmap, SecGrid and Netsparker). Additionally, the number of failed/successful past DDoS attacks must also be reported and the presence of an external cybersecurity advisor must be indicated as well.

Next, the profile is updated and submitted to the *SecRiskAI*'s backend for further processing. Through the sidebar, the user is able to navigate to the dashboard, where the contextual information is again intuitively presented and the risk prediction is automatically triggered. The company's profile is processed by the middleware and the actual prediction is delivered by the ML classifier. The prediction response is in turn rendered by the frontend and intuitively integrated into the dashboard. As shown by Figure 7.2, the overall DDoS risk prediction for the given profile is "Medium".

7.2 Case Study #2 - Recommendation of Protections

To further improve the effectiveness and reliability of the risk assessment process, *SecRiskAI* offers a seamless integration with MENTOR [13], a support tool focusing on the recommendation of cybersecurity protection services. There is an integration already placed with MENTOR to provide recommendations for DDoS attacks [14]. Thus, this case study focuses on exploring the investment options based on the assessed DDoS risk. In order to request a list of recommendations, the E-Shop is responsible for providing a desired list of parameters required for the recommendation process. Initially, *SecRiskAI* provides default parameters which can be changed by the company at any time. Figure 7.3 shows the panel encapsulating these parameters in the dashboard.

Protection Services Parameters



Attack Type(s) Coverage
 Volumetric, Application
 Service Type(s)
 Proactive
 Deployment Type - Priority: Low
 Seconds
 Leasing Period - Priority: Low
 Minutes

Figure 7.3: Protection service parameters panel




Protection Services						
Recommendations provided by MENTOR						
PROVIDER	SERVICE NAME	DESCRIPTION	ATTACK TYPE(S) COVERED	DEPLOYMENT	LEASING PERIOD	PRICE
	Incapsula	The Imperva Incapsula service delivers a multi-faceted approach to DDoS defense, providing blanket protection from all DDoS attacks to shield your critical online assets from these threats. Incapsula DDoS protection services are backed by a 24x7 security team, 99.999% uptime SLA, and a powerful, global network of data centers.	Volumetric, Protocol, Application, SSL, DNS	Seconds	Days	\$4,500
	Verisign DDoS Protection Service	Verisign DDoS Protection Services help organisations reduce the risk of catastrophic DDoS attacks by detecting and filtering malicious traffic aimed at disrupting or disabling their internet-based services. Unlike traditional security solutions, Verisign DDoS Protection Services filter harmful traffic upstream of the organisational network or in the cloud	Volumetric, Protocol, Application, SSL, DNS	Seconds	Months	\$3,700
	Advanced DDoS Attack Protection	Cloudflare's advanced DDoS protection, provisioned as a service at the network edge, matches the sophistication and scale of such threats, and can be used to mitigate DDoS attacks of all forms and sizes including those that target the UDP and ICMP protocols, as well as SYN/ACK, DNS amplification and Layer 7 attacks	Volumetric, Protocol, Application, DNS	Seconds	Months	\$4,900

Figure 7.4: MENTOR's recommendations list

The company can specify the attack type (*e.g.*, Volumetric, Application, DNS, or SSL/TLS) that have to be covered by the protection service as well as the service type, which can be either proactive (*i.e.*, service provides protection against possible future attacks) or reactive (*i.e.*, service offers protection as soon as the attack occurs). Moreover,

there is also the possibility to indicate the desired deployment time (*i.e.*, time required by the service to be deployed and active) and leasing period (*i.e.*, for how long is the company willing to contractually lease the service) followed by the assigned priority. Additionally, other parameters, such as cybersecurity budget and the operational region of the company are also fundamental parameters that are taken into consideration during the recommendation process. Similar to the cyber-risk prediction, the recommendation process for the E-Shop is triggered automatically with the default parameters (*cf.* Figure 7.3) as soon as the dashboard is loaded.

The company profile is submitted to MENTOR which returns a list of recommended services. The resulting list is then displayed in a table and is integrated in the lower section of the dashboard. Figure 7.4 shows the recommended protection services for the default parameters presented in Figure 7.3. As shown in Figure 7.4, each service provides a short description, the different attack types, the deployment time, leasing period, and the price. Additionally, it is crucial to note that the order of the services displayed is already determined by MENTOR, with the first row indicating the best-suited service to the profile provided.

As mentioned, the E-Shop is able to change the protection services parameters by clicking on the “Configure” button rendered at the top-right corner of Figure 7.4. Upon clicking this button, a modal window (*cf.* Figure 7.5) will open with all the input fields required for updating the protection service parameters and assigning a priority to the deployment time and leasing period. A click on “Update” will trigger a further recommendation process, which will, in turn, update the table of protection services.

Configure Service Parameters

Service Type(s)

Proactive

Attack Type(s)

Volumetric Application

Deployment Time

Seconds

Priority

Low Medium High

Leasing Period

Minutes

Priority

Low Medium High

Update

Figure 7.5: Protection service parameters configuration panel

7.3 Case Study #3 - Phishing Scenario

This third case study focuses on the ability of *SecRiskAI* to assess the risk of phishing, another popular and prevalent cyber-attack. With regards to this type of cyber-attack, *SecRiskAI* employs the MLP model developed on the synthetic dataset presented in Chapter 4. Furthermore, for this particular case study, it was assumed that the cyber-security risk assessment was performed by an organization operating in the financial sector. More specifically, the profile analyzed by *SecRiskAI* is the one of a well-known bank, operating mostly in North America with a yearly revenue of over 36 Million US\$. Additional information include: more than 70,000 employees with an average cyber awareness training level of “low”, a cybersecurity budget of over 750,000 US\$ and cybersecurity investments which amount to a total of 500,000 US\$. The full list of general information and technical details of Bank X are presented in Figure 7.6.

General Information

Case Study #1 Case Study #2

COMPANY	INDUSTRY
Bank X	Financial Services
OPERATIONAL REGION	BUSINESS VALUE (REVENUE)
North America	\$ 36,000,000
NUMBER OF EMPLOYEES	EMPLOYEE TRAINING
70,000	LOW MEDIUM HIGH
CYBERSECURITY BUDGET	PRIORITY
\$ 750,000	LOW MEDIUM HIGH

Technical Details

INVESTED AMOUNT IN CYBERSECURITY	EXTERNAL ADVISOR
\$ 500,000	No
KNOWN VULNERABILITIES	SUCCESSFUL PAST ATTACKS
0	2
FAILED PAST ATTACKS	
15	

Update

Figure 7.6: Bank X contextual information

Once submitted, the profile is processed by *SecRiskAI*’s backend and the resulting cyber-risk assessment is integrated in the dashboard along with the other metrics and information already discussed in previous chapters. As indicated by Figure 7.7, the resulting

phishing prediction for the bank’s profile was categorized as “High”. Possible explanation to this outcome is the average low level of employee training, which greatly increases the likelihood of data breaches caused by phishing attacks. In addition to that, as indicated by the profile in Figure 7.6, the Bank X is not employing any external cybersecurity advisor negatively impacting the organization’s overall preparation and awareness. Moreover, the assessed phishing risk is also justified by the fact that, according to a recent report published by Akamai [138], in 2020 phishing attacks on the financial sector increased almost by 45%.

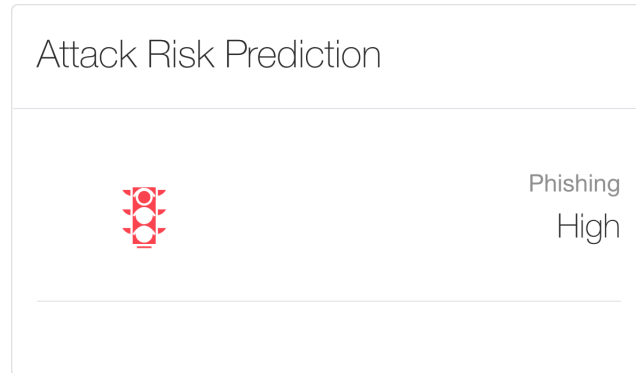


Figure 7.7: Phishing risk prediction

7.4 Discussion and Limitations

As presented in the above sections, the proposed case studies analyzed and investigated various aspects and functionalities currently offered by the *SecRiskAI* prototype. Along the user-friendly and intuitive dashboard design, as of now, *SecRiskAI* is able to provide valuable and accurate cybersecurity risk assessment for DDoS and phishing attacks while also integrating MENTOR, a tool for cybersecurity protection service recommender system. Furthermore, the quantitative evaluation of the various ML algorithms presented in Chapter 6 demonstrated that for larger datasets, SVMs achieve a slightly higher accuracy, while maintaining a lower training time when compared to MLP. Nonetheless, all four ML algorithms performed well and in most cases were able to achieve more than 90% accuracy. Moreover, the generated confusion matrices also confirmed again that the evaluated ML algorithms were able to classify most samples correctly. Other important metrics, such as precision, recall and F1-Score also provided valuable insights into the ML algorithm performance for every output class.

As of now, the biggest limitation is the lack of real-world datasets for training the ML algorithms used in this work. To partially overcome this limitation and prove the effectiveness of the prototype, a synthetic dataset generation approach was followed, as described in Chapter 4. However, while synthetic data is able to mimic various properties and aspects of real data, it is usually very challenging to generate high-quality data for complex problems. If the generated dataset does not match the behaviour and properties of the real-world dataset, this will negatively impact the performance of the trained ML models.

Lastly, *SecRiskAI* is also limited to assess the risk of only two types of cyber-attacks, namely DDoS and phishing. To address this limitation, the current prototype was designed to be easily extensible, meaning that new ML models trained specifically for different types of cyber-attacks can easily be integrated into the current solution and exposed through the same API.

Chapter 8

Summary, Conclusions, and Future Work

The main goal of this thesis was to design and implement *SecRiskAI*, a ML-based tool for supporting the cybersecurity risk assessment process. First, the overall risk assessment workflow was designed, which encapsulated various important steps such as data gathering, data processing, ML model selection and performance evaluation. In specific, this work investigated the suitability of four ML algorithms (*i.e.*, DT, SVM, KNN and MLP) for predicting and assessing the likelihood of cyber-risks. For this purpose, datasets of various sizes were generated and used during the training and testing phase of each ML model. Once tested and validated, the models were integrated in *SecRiskAI* ML classifier, which exposed various API endpoints primarily consumed by the GUI.

The implemented proof-of-concept is able to assess the risk only for a sub-set of well-known cyber-attacks, namely DDoS and phishing. For that, the prototype requires a specific set of attributes, also referred to as profile or contextual information. Based on this kind of data, *SecRiskAI* is able to predict the likelihood of being targeted by either DDoS or phishing attacks. Besides that, the current prototype also supports the integration with MENTOR to provide a list of recommended protection services based on the profile and also influenced by the calculated cyber-risk. Additionally, the integration with MENTOR was also designed to be fully configurable, meaning that the user is able at any point to update and set the priority to different profile attributes, thus triggering a new recommendation process.

Future work includes researching and investigating risk factors that may contribute in developing cyberattack-specific ML models, *i.e.*, models that are capable and fully specialized to assess the risk of specific types of cyber-attacks, allowing for a more comprehensive cybersecurity risk assessment phase. Further experimental tests are needed to estimate the behaviour and performance of the ML models currently implemented in *SecRiskAI* on real-world datasets. Additionally, the possibility of collecting prediction feedback should be explored in future studies to determine whether the performance and overall accuracy of the ML models can benefit from it. Future work should also examine the aspect of continuous risk monitoring, where key risk indicators are constantly collected and used for automated and continuous cybersecurity risk assessments in order to estimate the likelihood of unpredictable cyber-threats.

Bibliography

- [1] Louis Columbus: The Best Cybersecurity Predictions For 2021 Roundup. <https://www.forbes.com/sites/louiscolumbus/2020/12/15/the-best-cybersecurity-predictions-for-2021-roundup/>, (Last accessed April 2021).
- [2] Jon Oltsik: Cyber risk management continues to grow more difficult. <https://www.csoonline.com/article/3324363/cyber-risk-management-continues-to-grow-more-difficult.html>, (Last accessed April 2021).
- [3] Thomas Poppensieker: A new posture for cybersecurity in a networked world. <https://www.mckinsey.com/business-functions/risk/our-insights/a-new-posture-for-cybersecurity-in-a-networked-world>, (Last accessed April 2021).
- [4] IBM Security: Cost of a Data Breach Report. <https://www.ibm.com/downloads/cas/RDEQK07R>, (Last accessed April 2021).
- [5] ISO: Risk management - Guidelines. <https://www.iso.org/obp/ui/#iso:std:iso:31000:ed-2:v1:en>, (Last accessed April 2021).
- [6] The OpenGroup: The TOGAF Standard. <https://publications.opengroup.org/c182>, (Last accessed April 2021).
- [7] National Institute of Standards and Technology (NIST): SP800-30. <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-30r1.pdf>, (Last accessed April 2021).
- [8] Salvatore Marco Pappalardo, Marcin Niemiec, Maya Bozhilova, Nikolai Stoianov, Andrzej Dziech, Burkhard Stiller: Multi-Sector Assessment Framework - A New Approach to Analyse Cybersecurity Challenges and Opportunities, Springer, CCIS, pp. 1-15.
- [9] Deloitte.: Why artificial intelligence is a game changer for risk management. <https://www2.deloitte.com/content/dam/Deloitte/us/Documents/audit/us-ai-risk-powers-performance.pdf>, (Last accessed April 2021).
- [10] Maxwell W. Libbrecht, William S. Noble: Machine learning applications in genetics and genomics, Nature Reviews Genetics volume 16, pp. 321-332, 2015.

- [11] Konstantina Kourou, Themis P.Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis, Dimitrios I. Fotiadis: Machine learning applications in cancer prognosis and prediction, *Computational and Structural Biotechnology Journal* Volume 13, 2015, pp 8-17.
- [12] B. Rodrigues, M. F. Franco, G. Paranghi, B. Stiller: SEConomy: A Framework for the Economic Assessment of Cybersecurity; 16th International Conference on the Economics of Grids, Clouds, Systems, and Services (GECON 2019), Springer, Leeds, UK, pp. 1-9.
- [13] M. Franco, B. Rodrigues, B. Stiller: MENTOR: The Design and Evaluation of a Protection Services Recommender System; International Conference on Network and Service Management, Halifax, Canada, 2019, pp. 1-7.
- [14] M. Franco, E. Sula, B. Rodrigues, E. Scheid, B. Stiller: ProtectDDoS: A Platform for Trustworthy Offering and Recommendation of Protections; International Conference on Economics of Grids, Clouds, Software and Services (GECON 2020), Izola, Slovenia, September 2020, pp 1-12.
- [15] P. Radanliev: Artificial Intelligence and Cyber Risk Super-Forecasting; University of Oxford, Department of Engineering Science, Pre-Print, March 2020.
- [16] G. Moraes: Things to Consider When Calculating the Return on Security Investment. <https://securityintelligence.com/things-to-consider-when-calculating-the-return-on-security-investment>, (Last accessed April 2021).
- [17] Stanley Kaplan and B. John Garrick: On The Quantitative Definition of Risk. <https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1539-6924.1981.tb01350.x>, (Last accessed March 2021).
- [18] Linda Wilbanks: What's Your IT Risk Approach, June 2018.
- [19] Lucidchart: 5 Steps to Any Effective Risk Management Process. <https://www.lucidchart.com/blog/risk-management-process>, (Last accessed March 2021).
- [20] Terje Avena, Bodil S.Krohn: A new perspective on how to understand, assess and manage risk and the unforeseen. January 2014.
- [21] Stanford Encyclopedia of Philosophy: Artificial Intelligence. <https://plato.stanford.edu/entries/artificial-intelligence/>, (Last accessed March 2021).
- [22] IBM Cloud Education: Artificial Intelligence (AI), <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence>, (Last accessed March 2021).
- [23] Andrew Estialbo: Artificial Intelligence Overview, <https://blog.oursky.com/2020/05/07/artificial-intelligence-ai-for-businesses-what-you-need-to-know-before-starting-an-ai-project/ai-vs-ml-vs-dl/>, (Last accessed March 2021).

- [24] Michael E. Grost, Trent Jaeger: Applications of Artificial Intelligence, pp. 165-229, 1989.
- [25] Jonathan Johnson: 4 Types of Artificial Intelligence. <https://www.bmc.com/blogs/artificial-intelligence-types/>, (Last accessed March 2021).
- [26] Eda Kavlakoglu: AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: Whatâs the Difference?, (Last accessed March 2021).
- [27] IBM Cloud Education: What is Machine Learning?. <https://www.ibm.com/cloud/learn/machine-learning>, (Last accessed March 2021).
- [28] Ajiboye Abdulraheem, Ruzaini Abdullah Arshah: Evaluating the Effect of Dataset Size on Predictive Model Using Supervised Learning Technique, February 2015.
- [29] Centric: Machine Learning: A Quick Introduction and Five Core Steps. <https://centricconsulting.com/blog/machine-learning-a-quick-introduction-and-five-core-steps/>, (Last accessed March 2021).
- [30] Alina Zhang: Data Types From A Machine Learning Perspective With Examples. <https://towardsdatascience.com/data-types-from-a-machine-learningperspective-with-examples-111ac679e8bc>, (Last accessed March 2021).
- [31] Influxdata: What is time series data?. <https://www.influxdata.com/what-is-time-series-data/>, (Last accessed March 2021).
- [32] Armand Ruiz Gabernet: Breaking the 80/20 rule: How data catalogs transform data scientists' productivity. <https://www.ibm.com/cloud/blog/ibm-data-catalog-data-scientists-productivity>, (Last accessed March 2021).
- [33] Michael Chen: What Is Data Preparation and Why Is It Important?. <https://blogs.oracle.com/analytics/what-is-data-preparation-and-why-is-it-important>, (Last accessed March 2021).
- [34] Ali B. Nassie, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, Khaled Shaalan: Speech Recognition Using Deep Neural Networks: A Systematic Review. February 2019.
- [35] IBM Cloud Education: What is Supervised Learning?. <https://www.ibm.com/cloud/learn/supervised-learning>, (Last accessed March 2021).
- [36] Google: Training and Test Sets: Splitting Data. <https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data>, (Last accessed March 2021).
- [37] ScienceDirect: Supervised Learning. <https://www.sciencedirect.com/topics/computer-science/supervised-learning>, (Last accessed March 2021).

- [38] Altexsoft: Price Forecasting: Applying Machine Learning Approaches to Electricity, Flights, Hotels, Real Estate, and Stock Pricing. <https://www.altexsoft.com/blog/business/price-forecasting-machine-learning-based-approaches-applied-to-electricity-flights-hotels-real-estate-and-stock-pricing/>, (Last accessed March 2021).
- [39] Herman Kamper, Aren Jansen, Sharon Goldwater: Fully Unsupervised Small-Vocabulary Speech Recognition Using a Segmental Bayesian Model, 2015.
- [40] Sarfaraz Hussein, Pujan Kandel, Candice W. Bolan, Michael B. Wallace and Ulas Bagci: Lung and Pancreatic Tumor Characterization in the Deep Learning Era: Novel Supervised and Unsupervised Learning Approaches, August 2019.
- [41] Md. Ahsanul Kabir, Xiao Luo: Unsupervised Learning for Network Flow Based Anomaly Detection in the Era of Deep Learning, August 2020.
- [42] S. Shriram, E. Sivasankar: Anomaly Detection on Shuttle data using Unsupervised Learning Techniques, December 2019.
- [43] Muhammad Usama, Junaid Qadir, Aunn Raza, Hunain Arif, Kok-lim Alvin Yau; Yehia Elkhatib, Amir Hussain and Ala Al-Fuqaha: Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges, April 2019.
- [44] Xiaojin Zhu, Andrew Goldberg: Introduction to Semi-Supervised. Learning, 2009.
- [45] Richard S. Sutton, Andrew G. Barto: Reinforcement Learning: An Introduction, 2018.
- [46] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, Srikanth Kandula: Resource Management with Deep Reinforcement Learning, 2016.
- [47] I. Arel, C. Liu, T. Urbanik, A.G. Kohls: Reinforcement learning-based multi-agent system for network traffic signal control, 2010.
- [48] Jens Kober, J. Andrew Bagnell, Jan Peters: Reinforcement Learning in Robotics: A Survey, September 2013.
- [49] AWS: Training ML Models. <https://docs.aws.amazon.com/machine-learning/latest/dg/training-ml-models.html>, (Last accessed March 2021).
- [50] AWS: Model Fit: Underfitting vs. Overfitting. <https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html>, (Last accessed March 2021).
- [51] Jonas Benner: Cross-Validation and Hyperparameter Tuning: How to Optimise your Machine Learning Model. <https://towardsdatascience.com/cross-validation-and-hyperparameter-tuning-how-to-optimise-your-machine-learning-model-13f005af9d7d>, (Last accessed March 2021).
- [52] Wikipedia: Hyperparameter optimization. https://en.wikipedia.org/wiki/Hyperparameter_optimization, (Last accessed March 2021).

- [53] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, Si-Hao Deng: Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization, March 2019.
- [54] James Bergstra, Remi Bardenet, Yoshua Bengio, Balazs Kegl: Algorithms for Hyperparameter Optimization, 2011.
- [55] Raheel Shaikh: Cross Validation Explained: Evaluating estimator performance. <https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85>, (Last accessed March 2021).
- [56] SpringerLink: Holdout Evaluation. https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_369, (Last accessed March 2021).
- [57] Artem Oppermann: What is Deep Learning and How does it work?. <https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac>, (Last accessed March 2021).
- [58] IBM Cloud Education: Deep Learning. <https://www.ibm.com/cloud/learn/deep-learning>, (Last accessed March 2021).
- [59] IBM Developer: An introduction to deep learning. <https://developer.ibm.com/technologies/artificial-intelligence/articles/an-introduction-to-deep-learning/>, (Last accessed March 2021).
- [60] Martin Heller: What is deep learning? Algorithms that mimic the human brain. <https://www.infoworld.com/article/3397142/what-is-deep-learning-algorithms-that-mimic-the-human-brain.html>, (Last accessed March 2021).
- [61] Farhad Malik: Neural Network Layers. <https://medium.com/fintechexplained/neural-network-layers-75e48d71f392>, (Last accessed April 2021).
- [62] Nuric: Artificial Neural Networks. <https://www.doc.ic.ac.uk/~nuric/teaching/imperial-college-machine-learning-neural-networks.html>, (Last accessed April 2021).
- [63] Siddharth Sharma, Simone Sharma, Anidhya Athaiya: Activation Functions in Neural Networks, Vol. 4, Issue 12, pp. 310-316, April 2020.
- [64] Hoon Chung, Sung Joo Lee, Jeon Park: Deep neural network using trainable activation functions, July 2016.
- [65] Thomas Wood: Sigmoid Function. <https://deeptai.org/machine-learning-glossary-and-terms/sigmoid-function>, (Last accessed April 2021).
- [66] Emma Amor: Understanding Non-Linear Activation Functions in Neural Networks. <https://medium.com/ml-cheat-sheet/understanding-non-linear-activation-functions-in-neural-networks-152f5e101eeb>, (Last accessed April 2021).

- [67] DeepAI: ReLU. <https://deepai.org/machine-learning-glossary-and-terms/relu>, (Last accessed April 2021).
- [68] Gangming Zhao, Zhaoxiang Zhang, He Guan, Peng Tang, Jingdong Wang: Rethinking ReLU to Train Better CNNs, August 2018.
- [69] Scott Robinson: Introduction to Neural Networks with Scikit-Learn. <https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn/>, (Last accessed April 2021).
- [70] Thomas Wood: What is Backpropagation?. <https://deepai.org/machine-learning-glossary-and-terms/backpropagation>, (Last accessed April 2021).
- [71] Shiliang Sun, Zehui Cao, Han Zhu, Jing Zhao: A Survey of Optimization Methods from a Machine Learning Perspective, October 2019.
- [72] Marsh: Global Cyber Risk Perception Survey. <https://www.marsh.com/us/insights/research/global-cyber-risk-perception-survey.html>, (Last accessed April 2021).
- [73] NIST Cybersecurity Framework: The Five Functions. <https://www.nist.gov/cyberframework/online-learning/five-functions>, (Last accessed April 2021).
- [74] Cipher: A Quick NIST Cybersecurity Framework Summary. <https://cipher.com/blog/a-quick-nist-cybersecurity-framework-summary/>, (Last accessed April 2021).
- [75] Gwendolyn Keyes, Terese Richmond, Michael D. Farber, Darshana Singh: GAO Reports Challenges and Successes in Cybersecurity Framework Adoption. <https://www.vnf.com/gao-reports-challenges-and-successes-in-cybersecurity-framework>, (Last accessed April 2021).
- [76] 27001Academy: What is ISO 27001?. <https://advisera.com/27001academy/what-is-iso-27001/>, (Last accessed April 2021).
- [77] Jeff Petters: What is ISO 27001 Compliance? Essential Tips and Insights. <https://www.varonis.com/blog/iso-27001-compliance>, (Last accessed April 2021).
- [78] SANS: CIS Critical Security Controls. <https://www.sans.org/critical-security-controls>, (Last accessed April 2021).
- [79] Bill Gogel: Cybersecurity Frameworks . <https://www.bswllc.com/resources-articles-cybersecurity-frameworks>, (Last accessed April 2021).
- [80] Soumya Kanti Datta: DRAFT - A Cybersecurity Framework for IoT Platforms, August 2020.
- [81] Sergey Naumov, Ilya Kabanov: Dynamic framework for assessing cyber security risks in a changing environment, December 2016.
- [82] Sri Nikhil Gupta Gouriseti, Michael Mylrea, Hirak Patangia: Application of Rank-Weight Methods to Blockchain Cybersecurity Vulnerability Assessment Framework, May 2019.

- [83] Jeevith Hegde, Børge Rokseth: Applications of machine learning methods for engineering risk assessment A review, September 2019.
- [84] Yuri Castro, Young Jim kim: Data mining on road safety: factor assessment on vehicle accidents using classification models, November 2015, pp. 104-111.
- [85] Madhar Taamneh, Sharaf Alkheder, Salah Taamneh: Data-mining techniques for traffic accident modeling and prediction in the United Arab Emirates, April 2016, pp. 146-166.
- [86] Il-Hwan Kim, Jae-Hwan Bong, Jooyoung Park, Shinsuk Park: Prediction of Driver's Intention of Lane Change by Augmenting Sensor Information Using Machine Learning Techniques, June 2017.
- [87] Kai Wang, Youjin Zhao, Qingyu Xiong, Min Fan, Guotan Sun, Longkun Ma, Tong Liu: Research on Healthy Anomaly Detection Model Based on Deep Learning from Multiple Time-Series Physiological Signals, September 2016.
- [88] Ahmet Turan Özdemir, Billur Barshan: Detecting Falls with Wearable Sensors Using Machine Learning Techniques, June 2014.
- [89] Y. Xu, Z.Y. Dong, K. Meng, R. Zhang, K.P. Wong: Real-time transient stability assessment model using extreme learning machine, Volume 5, Issue 3, March 2011, pp. 314-322.
- [90] Claudio M. Rocco, Enrico Zio: A support vector machine integrated system for the classification of operation anomalies in nuclear components and systems, Volume 92, Issue 5, May 2007, pp. 593-600.
- [91] Javier Martínez Torres, Carla Iglesias Comesaña, Paulino J. García-Nieto: Review: machine learning techniques applied to cybersecurity, International Journal of Machine Learning and Cybernetics, volume 10, pp. 2823-2836.
- [92] James B. Fraley, James Cannady: The promise of machine learning in cybersecurity, April 2017.
- [93] George V. Hulme: Tackling cybersecurity threat information sharing challenges. <https://www.csoonline.com/article/3157540/tackling-cybersecurity-threat-information-sharing-challenges.html>, (Last accessed May 2021).
- [94] Derryck Coleman: Nearly 65% of Affected Public Companies Did Not Report Cybersecurity Breaches to the SEC. <https://blog.auditanalytics.com/nearly-70-of-affected-public-companies-did-not-report-cybersecurity-breaches-to-the-sec/>, (Last accessed May 2021).
- [95] NACD, CyberThreat Alliance, IHS Markit, SecurityScorecard, Diligent: The State of Cyber-Risk Disclosures of Public Companies. <https://s3.amazonaws.com/ssc-corporate-website-production/documents/resources/the-state-of-cyber-risk-disclosures-of-public-companies.pdf>, (Last accessed May 2021).

- [96] World Economic Forum: Cyber Information Sharing: Building Collective Security. http://www3.weforum.org/docs/WEF_Cyber_Information_Sharing_2020.pdf, October 2020, (Last accessed May 2021).
- [97] Markus Hittmeir, Andreas Ekelhart, Rudolf Mayer: On the Utility of Synthetic Data: An Empirical Evaluation on Machine Learning Tasks, ARES '19, August 2019.
- [98] Sangoma: SMB, SME, and Large Enterprise: Why Your Business Size Classification Matters. <https://www.sangoma.com/articles/smb-sme-large-enterprise-size-business-matters/>, (Last accessed May 2021).
- [99] Nmap: the Network Mapper - Free Security Scanner. <https://nmap.org/>, (Last accessed May 2021).
- [100] metasploit: The world's most used penetration testing framework. <https://www.metasploit.com/>, (Last accessed May 2021).
- [101] OWASP: Vulnerability Scanning Tools. https://owasp.org/www-community/Vulnerability_Scanning_Tools, (Last accessed May 2021).
- [102] Homeland Security: Cybersecurity Advisors. https://www.bu.edu/tech/files/2017/09/DHS_CSA_Fact_Sheet_2017-1.pdf, (Last accessed May 2021).
- [103] AWS: Multiclass Classification. <https://docs.aws.amazon.com/machine-learning/latest/dg/multiclass-classification.html>, (Last accessed May 2021).
- [104] IBM: Decision Tree. <https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=analytics-decision-tree>, (Last accessed May 2021).
- [105] Wikipedia: Decision Tree. https://en.wikipedia.org/wiki/Decision_tree, (Last accessed May 2021).
- [106] Machine Learning Mastery: K-Nearest Neighbors for Machine Learning. <https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning>, (Last accessed May 2021).
- [107] Machine Learning Mastery: One-vs-Rest and One-vs-One for Multi-Class Classification. <https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification>, (Last accessed May 2021).
- [108] ScienceDirect: Support Vector Machines. <https://www.sciencedirect.com/topics/neuroscience/support-vector-machines>, (Last accessed May 2021).
- [109] DataFlair: Kernel Functions-Introduction to SVM Kernel & Examples. <https://data-flair.training/blogs/svm-kernel-functions/>, (Last accessed May 2021).
- [110] Heaton Research: The Number of Hidden Layers. <https://www.heatonresearch.com/2017/06/01/hidden-layers.html>, (Last accessed May 2021).

- [111] Harpreet Singh Sachdev: Choosing number of Hidden Layers and number of hidden neurons in Neural Networks. <https://www.linkedin.com/pulse/choosing-number-hidden-layers-neurons-neural-networks-sachdev/>, (Last accessed May 2021).
- [112] Brilliant: Backpropagation. <https://brilliant.org/wiki/backpropagation>, (Last accessed May 2021).
- [113] Unite.AI: What is Backpropagation?. <https://www.unite.ai/what-is-backpropagation/>, (Last accessed May 2021).
- [114] TypeScript. <https://www.typescriptlang.org/>, (Last accessed May 2021).
- [115] Altexsoft: The Good and the Bad of TypeScript. <https://www.altexsoft.com/blog/typescript-pros-and-cons/>, (Last accessed May 2021).
- [116] React: A JavaScript library for building user interfaces <https://reactjs.org/>, (Last accessed May 2021).
- [117] Why Choose Reactjs For Your Next Project <https://easternpeak.com/blog/why-choose-reactjs-for-your-next-project>, (Last accessed May 2021).
- [118] Redux Framework: A predictable state container for Javascript Apps <https://redux.js.org/>, (Last accessed May 2021).
- [119] Bruno Rodrigues, Muriel Franco, Geetha Parangi, Burkhard Stiller: SEConomy: A Framework for the Economic Assessment of Cybersecurity; 16th Conference on the Economics of Grids, Clouds, Systems, and Services (GECON 2019), Leeds, UK, September 2019, pp 1-13.
- [120] D. Singh, B. Singh: Investigating the impact of data normalization on classification performance; Journal of Applied Soft Computing, Volume 97, Part B, December 2020, 105524.
- [121] Wikipedia: Feature scaling. https://en.wikipedia.org/wiki/Feature_scaling, (Last accessed June 2021).
- [122] Nestjs. <https://nestjs.com/>, (Last accessed June 2021).
- [123] BentoML: Model Serving Made Easy. <https://docs.bentoml.org/en/latest/>, (Last accessed June 2021).
- [124] Pandas. <https://pandas.pydata.org/>, (Last accessed June 2021).
- [125] Jupyter. <https://jupyter.org/>, (Last accessed June 2021).
- [126] scikit-learn. <https://scikit-learn.org/stable/>, (Last accessed June 2021).
- [127] scikit-learn: Decision Tree Classifier. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>, (Last accessed June 2021).

- [128] scikit-learn: K-Nearest Neighbors. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>, (Last accessed June 2021).
- [129] scikit-learn: C-Support Vector Classification. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>, (Last accessed June 2021).
- [130] scikit-learn: Multi-layer Perceptron classifier. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, (Last accessed June 2021).
- [131] Jason Brownlee: What is a Confusion Matrix in Machine Learning. <https://machinelearningmastery.com/confusion-matrix-machine-learning/>, (Last accessed June 2021).
- [132] Margherita Grandini: Metrics for Multi-Class Classification: an Overview. <https://www.arxiv-vanity.com/papers/2008.05756>, (Last accessed June 2021).
- [133] Will Koehrsen: Beyond Accuracy: Precision and Recall. <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>, (Last accessed June 2021).
- [134] E. A. Zanaty: Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. Mathematics Dept., Computer Science Section, Faculty of Science, Sohag University, Sohag, Egypt. August 2012.
- [135] André R. Goncalves, Maria A. de O. Camargo-Brunetto: Classification of Poincaré plots for temporal series of heart rate variability by using machine learning techniques. November 2010.
- [136] Kaspersky Lab: Denial of Service: How Business Evaluate The Threat of DDoS Attacks. https://media.kasperskycontenthub.com/wp-content/uploads/sites/45/2018/03/08234158/IT_Risks_Survey_Report_Threat_of_DDoS_Attacks.pdf, (Last accessed July 2021).
- [137] Gaute B. Wangen, Andrii Shalaginov, Christoffer V. Hallstensen: Cyber Security Risk Assessment of a DDoS Attack. International Conference on Information Security. September 2016.
- [138] Akamai Security Research: Financial Services Continues Getting Bombarded With Credential Stuffing And Web Application Attacks. <https://www.akamai.com/us/en/about/news/press/2021-press/akamai-soti-security-research-phishing-for-finance.jsp>, (Last accessed July 2021).
- [139] M. Franco, J. Von der Assen, L. Boillat, C. Killer, B. Rodrigues, E. J. Scheid, L. Granville, B. Stiller: SecGrid: A Visual System for the Analysis and ML-Based Classification of Cyberattack Traffic; IEEE 46th Conference on Local Computer Networks (LCN 2021), Edmonton, Canada, Virtually, October 2021, pp 1-8.

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CV	Cross-Validation
CSA	Cybersecurity Advisor
DL	Deep Learning
DT	Decision Tree
DDoS	Distributed Denial-of-Service
FFNN	Feed-Forward Neural Network
GAO	Government Accountability Office
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
ISMS	Information Security Management Systems
IS	Information Security
ISO	International Organization for Standardization
IoT	Internet of Things
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
KRI	Key Risk Indicator
ML	Machine Learning
MCC	Multi-Class Classification
MLP	Multi-Layer Perceptron
NIST	National Institute of Standards and Technology
PE	Performance Estimate
PSP	Protection Service Provider
RL	Reinforcement Learning
SL	Supervised Learning
SSL	Semi-Supervised Learning
SVM	Support-Vector Machine
UL	Unsupervised Learning

List of Figures

2.1	Risk management framework [19]	6
2.2	Artificial Intelligence overview [23]	7
2.3	Machine Learning types (based on [34])	9
2.4	Hyperparameter optimization techniques [52]	12
2.5	K-Fold CV (K=5) [55]	12
2.6	Artificial Neural Network (based on [57])	13
2.7	Representation of a neuron [62]	14
2.8	Most common non-linear activation functions	15
4.1	Architecture overview	24
4.2	Supervised Learning ML workflow	26
4.3	DT visualization	30
4.4	KNN visualization ($k = 7$)	31
4.5	SVM visualization	32
4.6	MLP visualization	34
4.7	Training, CV & testing workflow	35
5.1	SecRiskAI sequence flow	38
5.2	The SecRiskAI's Dashboard	41
5.3	Protection service parameters	42
5.4	User profile page	43
6.1	Confusion matrices	56

6.2	Dataset size evaluation	59
7.1	E-Shop contextual information	62
7.2	DDoS risk prediction	63
7.3	Protection service parameters panel	64
7.4	MENTOR's recommendations list	64
7.5	Protection service parameters configuration panel	65
7.6	Bank X contextual information	66
7.7	Phishing risk prediction	67
A.1	Terminal after docker-compose up execution	90

List of Tables

3.1	Cybersecurity risk assessment frameworks overview (based on [79])	19
3.2	Industry-wise segmentation of ML applications for risk assessment [83] . . .	21
4.1	Overview of the generated attributes	27
5.1	ML Models and corresponding Scikit-learn implementation [126]	50
5.2	Generated Dataset	52
6.1	Performance metrics	57
6.2	Computed Precision, Recall & F1-Score for each ML model	58

Listings

5.1	Redux action for updating cyber-risk predictions	39
5.2	Global store initialization	39
5.3	Fetching and storing risk predictions	40
5.4	Middleware controller	44
5.5	Middleware service	45
5.6	Prediction request body	46
5.7	Risk prediction service	46
5.8	ML artifacts import	48
5.9	Cyber-risk prediction response	48
5.10	Metadata response	49
5.11	Data generator	50
5.12	Dataset categorization	52
5.13	Dataset split and normalization	52
5.14	Example of DT hyperparameter tuning, training and CV	53

Appendix A

Installation Guidelines

This installation guideline is based on a MacOS system and may differ from the setup on a Windows or Linux system.

1. Initial setup

- (a) Download the latest version of Docker and start it: <https://docs.docker.com/docker-for-mac/install/> (MacOS), <https://docs.docker.com/docker-for-windows/install/> (Windows), <https://docs.docker.com/install/linux/docker-ce/ubuntu/> (Linux).
- (b) Create account on Github if not already existant.

2. Clone Github Repository

- (a) Download IntelliJ IDEA (or any other *IDE* of your preference): <https://www.jetbrains.com/idea/download>.
- (b) Open IntelliJ and click on *Check out from Version Control* and immediately after click *git*.
- (c) As URL copy and paste the following url: <https://github.com/Sulasdeli/SecRiskAI>.git and change the directory if necessary.

3. Starting the Application using Docker

- (a) Navigate to the following folder: **./SecRiskAI** and execute the following command:

1	<code>\$ docker-compose up</code>
---	-----------------------------------

- (b) If the projects correctly starts, the output on the terminal will show the started containers (*cf.* Figure A.1).
- (c) The *SecRiskAI*'s frontend will be running on *localhost:3001*.

```

docker-compose up
Creating secriskai_protectddos_1 ... done
Creating secriskai_ml_models_management_1 ... done
Creating secriskai_ml_decision_server_1 ... done
Creating secriskai_middleware_1 ... done
Creating secriskai_web_1 ... done
Attaching to secriskai_protectddos_1, secriskai_ml_decision_server_1, secriskai_ml_models_management_1, secriskai_middleware_1, sec
riskai_web_1
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [NestFactory] Starting Nest application...
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [InstanceLoader] HttpModule dependencies initialized +50ms
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [InstanceLoader] ConfigHostModule dependencies initialized +0ms
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [InstanceLoader] ConfigModule dependencies initialized +1ms
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [InstanceLoader] AppModule dependencies initialized +0ms
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [RoutesResolver] AppController {}: +21ms
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [RouterExplorer] Mapped {/predict, POST} route +2ms
ml_decision_server_1 | [2021-07-09 22:13:28,017] INFO - Starting BentoML proxy in production mode..
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [RouterExplorer] Mapped {/recommend, POST} route +1ms
middleware_1 | [Nest] 1 - 07/09/2021, 10:13:28 PM [NestApplication] Nest application successfully started +1ms
protectddos_1 | /usr/local/lib/python3.9/site-packages/flask_sqlalchemy/__init__.py:872: FSADeprecationWarning: SQLAlchemy
Y_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress
this warning.
protectddos_1 | warnings.warn(FSADeprecationWarning(
protectddos_1 | * Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
protectddos_1 | * Serving Flask app "engine" (lazy loading)
protectddos_1 | * Environment: production
protectddos_1 | WARNING: This is a development server. Do not use it in a production deployment.
protectddos_1 | Use a production WSGI server instead.
protectddos_1 | * Debug mode: off
protectddos_1 | * Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)

```

Figure A.1: Terminal after docker-compose up execution

Appendix B

Contents of the CD

- Abstract and Zusammenfassung:
 - *Abstract.txt* & *Zusfsg.txt*
- Thesis (PDF and L^AT_EX source code):
 - *MA_Erion_Sula.pdf* & *MA_Erion_Sula.zip*
- Midterm presentation slides:
 - *MA_Erion_Sula_midterm_presentation.pptx*
- Final presentation slides:
 - *MA_Erion_Sula_final_presentation.pptx*
- Application source code:
 - *SecRiskAI.zip*