VAIDOTAS DRUNGILAS

COLLABORATIVE DISTRIBUTED MACHINE LEARNING ON BLOCKCHAIN

DOCTORAL DISSERTATION

Kaunas
2024

KAUNAS UNIVERSITY OF TECHNOLOGY

VAIDOTAS DRUNGILAS

COLLABORATIVE DISTRIBUTED MACHINE LEARNING ON BLOCKCHAIN

Doctoral dissertation
Technological Sciences, Informatics Engineering (T 007)

2024, Kaunas

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Informatics, Department of Information Systems during the period of 2018–2023. The doctoral right has been granted to Kaunas University of Technology together with Vilnius Gediminas Technical University.

Scientific Supervisor:
Prof. Dr. Evaldas VAIČIUKYNAS (Kaunas University of Technology, Technological Sciences, Informatics Engineering, T 007).

Edited by: English language editor Dr. Armandas Rumšas (Publishing House *Technologija*), Lithuanian language editor Aurelija Gražina Rukšaitė (Publishing House *Technologija*).

The official defense of the dissertation will be held at 9 a.m. on 20 June, 2024 at the public meeting of Dissertation Defense Board of Informatics Engineering Science Field in Rectorate Hall at Kaunas University of Technology.

Address: K. Donelaičio 73-402, LT-44249 Kaunas, Lithuania.
Tel. no. (+370) 608 28 527; e-mail doktorantura@ktu.lt.

Doctoral dissertation was sent out on 20 May, 2024.
The doctoral dissertation is available on the internet http://ktu.edu and at the libraries of Kaunas University of Technology (Gedimino 50, LT-44239 Kaunas, Lithuania) and Vilnius Gediminas Technical University (Saulėtekio 14, Vilnius, LT-10223, Lithuania).

KAUNO TECHNOLOGIJOS UNIVERSITETAS

VAIDOTAS DRUNGILAS

BENDRADARBIAVIMU GRĮSTAS PASKIRSTYTAS MAŠININIS MOKYMASIS BLOKŲ GRANDINĖJE

Daktaro disertacija
Technologijos mokslai, informatikos inžinerija (T 007)

Kaunas, 2024

Disertacija rengta 2018–2023 metais Kauno technologijos universiteto Informatikos fakultete, Informacijos sistemų katedroje.

Doktorantūros teisė Kauno technologijos universitetui suteikta kartu su Vilniaus Gedimino technikos universitetu.

Mokslinis vadovas :
prof. dr. Evaldas VAIČIUKYNAS (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija, T 007).

Redagavo: anglų kalbos redaktorius dr. Armandas Rumšas (leidykla „Technologija“), lietuvių kalbos redaktorė Aurelija Gražina Rukšaitė (leidykla „Technologija“).

Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:
prof. dr. Robertas DAMAŠEVIČIUS (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija, T 007) – **pirmininkas**;
assoc. prof. dr. Razvan BOCU (Transilvania University of Brasov, Rumunija, technologijos mokslai, informatikos inžinerija, T 007);
prof. dr. Nikolaj GORANIN (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija, T 007);
prof. dr. Tomas KRILAVIČIUS (Vytauto Didžiojo universitetas, gamtos mokslai, informatika, N 009);
prof. dr. Agnė PAULAUSKAITĖ-TARASEVIČIENĖ (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009).

Disertacija bus ginama viešame informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje 2024 m. birželio 20 d. 9 val. Kauno technologijos universiteto Rektorato salėje.

Adresas: K. Donelaičio g. 73-402, 44249 Kaunas, Lietuva.
Tel. (+370) 608 28 527; el. paštas doktorantura@ktu.lt.

Disertacija išsiųsta 2024 m. gegužės 20 d.
Su disertacija galima susipažinti interneto svetainėje http://ktu.edu ir Kauno technologijos universiteto (Gedimino g. 50, 44239 Kaunas) ir Vilniaus Gedimino technikos universiteto (Saulėtekio al. 14, Vilnius, LT-10223, Lietuva) bibliotekose.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND TERMS

Abbreviations:

API – Application Programming Interface
AUC – Area Under the Curve
BCE – Binary Cross-Entropy
BFT – Byzantine Fault Tolerant
CART – Classification And Regression Tree
CDMLB – Collaborative Distributed Machine Learning on Blockchain
DApp – Distributed Application
IoT – Internet of Things
IPFS – Interplanetary File System
JSON – JavaScript Object Notation
ML – Machine Learning
PATE – Private Aggregation of Teacher Ensembles
PBFT – Practical Byzantine Fault Tolerant
PoS – Proof-of-Stake consensus algorithm
PoW – Proof-of-Work consensus algorithm
ROC – Receiver Operating Characteristic curve

Terms:

Blockchain – a decentralised ledger for recording transactions across a peer-to-peer network [1].

Blockchain network – a decentralised system of computers/participants which records and shares transactions by using a consensus algorithm [2].

Blockchain network participant – an individual involved in a decentralised system that may contribute to transaction validation through a consensus algorithm.

Smart contract – an executable digital contract written as a program code which automatically enforces predetermined rules when deployed on a blockchain [3].

Blockchain oracle – a network service that connects blockchain network to external systems or data.

Local web application – a software programme accessed and executed on a device operating without continuous connection to the Internet.

In the context of this work, the following terms are used with a specific meaning:

Trust – firm belief of a participant in the reliability of software solutions [4] and trust between collaborating organisations and individuals [5]

Partial trust – the trust required to participate in the collaboration and the decision to commit model and validation data files to the network, but not high enough to share training data outright.

Machine learning security (ML security) – the ability to protect data and model assets, model usage and development procedures against attacks [6].

Transparency – the ability to openly access and audit information [7].

Privacy preservation – ensuring non-disclosure of private information used in the machine learning model development process [8].

Collaborative machine learning – a machine learning process which has two or more participants that perform model development, model sharing or data sharing activities [9]. In the context of this dissertation, the overviewed methods and the presented solution are confined to the binary classification task.

INTRODUCTION

Motivation

The amount of data created worldwide is increasing as more and more people use network and communication technologies. To extract knowledge from such amounts of data effectively, automated data processing approaches must be developed. One such use case of data processing for various applications is machine learning. Machine learning solutions [10], [11], [12] are used successfully in computer vision, healthcare, predictive analytics and intelligent decision making, speech and pattern recognition, and other domains. With the increasing machine learning popularity, a growing number of parties try to solve similar machine learning problems, but, due to a limited quantity of high-quality data, they might not be able the achieve sufficient performance of machine learning models. Collaboration could be beneficial in all the machine learning development steps. Better performing models can be obtained by building more diverse datasets composed of multiple data sources; Model training would require less computational resources to achieve the model of a higher quality; Shared models could be reused, thus reducing the need to train new local models. Unfortunately, currently, machine learning model development is often contained in a single centralised environment with a limited amount of data.

Decentralisation could resolve such problems, but the demand for data sharing and collaboration in machine learning could lead to privacy and data or model security issues. Multi-party collaboration environments should not reveal any personal or any other sensitive information about the real-world entities. The data transfer channels should be secured, and technologies and services used should be trustworthy for the participants. A wide range of solutions have been proposed to facilitate the distributed machine learning process. The distributed learning solution could allow cooperation and improved performance. Participation in the distributed machine learning process can still suffer from lack of trust between the participants and the services, and the potential participants might lack the motivation to participate.

Distributed ledger technologies could be applied to solve trust and transparency issues. Distributed ledger technologies introduce interaction logging and might not require trusting network participants and services. Among the distributed ledger technologies, blockchain is the most prominent one. Blockchain technologies introduce data replication, which results in a system architecture that is more resistant to attacks. The trust issue is addressed by transaction logging, open access to stored data, and the immutability of blockchain networks. Every action performed on the blockchain is validated and logged, thus reducing the risk of malicious actors. An additional execution logic hosted on the blockchain can be introduced in the form of smart contracts. A smart contract enables the development of more diverse and complex distributed solutions. Smart contracts could also be extended by specialised services, which would allow distributing, reusing, and integrating the already existing solutions.

Currently, no well-established methods exist that would enable collaboration in distributed machine learning on blockchain technologies and ensure privacy

preservation, while allowing to reuse the already existing machine learning implementations.

Research objects and scope

The dissertation research object is collaboration in distributed machine learning. The scope of this research is focused on two main research areas: 1) collaborative distributed machine learning methods, their architectures, and the application of blockchain; and 2) blockchain technology-based systems methods, tools, and approaches.

Problem statement and research questions

Collaborative distributed machine learning approaches are limited by insufficient trust, limitations imposed by sensitive data, and a complex adaptation process for the currently existing machine learning solutions. To propose resolution methods for the aforementioned problems, this dissertation aims to answer to the following research questions:

RQ1. Can distributed machine learning transparency and collaboration be improved and, if so, how?

RQ2. How can blockchain technologies facilitate the collaborative distributed machine learning process?

RQ3. What modifications to blockchain technologies are required to enable collaboration in distributed machine learning?

RQ4. How can blockchain network participant data and model contributions be measured in collaborative distributed machine learning?

RQ5. Can training data privacy preservation be improved in blockchain technology-based distributed machine learning?

Aim and objectives

The aim of this thesis is to improve collaboration in distributed machine learning by using blockchain technologies. A number of objectives to reach the defined aim have been outlined:

1. Analyse distributed machine learning methods and collaboration approaches.
2. Analyse blockchain technologies, and their applications for distributed machine learning processes.
3. Propose a method for collaboration in distributed machine learning.
4. Implement the solution for collaborative distributed machine learning according to the proposed method.
5. Assess how the application of the blockchain technology affects distributed machine learning.
6. Evaluate the method capabilities to perform collaborative distributed machine learning.

Research methodology

The research was conducted by employing the constructive research method [13]. Based on the proposed method, the research was carried out in multiple stages:

- During the first stage, the research object and scope were defined, which encompass privacy-preserving distributed machine learning methods, their architectures and the application of blockchain, blockchain technology-based systems methods, tools and approaches. A research problem on collaboration in the distributed privacy-preserving machine learning process was identified.
- The second stage was dedicated to defining the research potential – how the blockchain technology could assist in resolving collaboration and privacy-preservation problems in distributed machine learning.
- The analysis of the defined problem domain was performed in the third stage. Comparative analysis of the currently available research and solutions combining privacy-preservation methods, the blockchain technology and distributed learning was conducted.
- The solution definition process was performed as step four, where the method for privacy preserving distributed machine learning was proposed, which provides means of collaboration when using private blockchain solutions and introduces methods to evaluate shared contributions.
- The fifth stage was dedicated to the implementation, experimental evaluation, and feasibility evaluation of the proposed method for potential application areas – the blockchain solution for CDMLB was implemented by using the proposed method and experimentally evaluated, by measuring the runtime and machine learning solution performance. The performance evaluation tested the applicability of the proposed method for two banking-related binary classification tasks. The performance of the developed classifiers for these datasets was experimentally tested.

Defended statements

1. The existing private blockchain architectures can be enhanced to support more diverse machine learning execution environments on a collaborative DML blockchain network.
2. Contributions of models to the collaborative DML blockchain network for individual participants can be evaluated by using the Shapley-based weighting strategy.
3. Knowledge distillation can be used to aggregate the model ensemble into a single model, without significantly reducing the performance of the classifier, in collaborative DML on a private blockchain network.

Scientific novelty

1. The proposed collaborative distributed machine learning method expands the existing capabilities of distributed machine learning on blockchain by extending the system architecture with the machine learning inference oracle service.
2. The proposed method introduces a Shapley value and performance-based ensemble weighting strategy as a solution to measure the network participant model contribution.
3. The proposed method uses the student-teacher distillation approach to increase the model privacy by enabling the aggregation of knowledge accumulated on the blockchain network.

Practical significance

1. The proposed method allows integrating the already established machine learning technologies into the blockchain architecture via local off-chain oracle services.
2. The proposed ensemble weighting strategy can be defined as a generalisation of performance weighting, which could be applied in any weighted ensemble development approach.
3. The proposed ensemble weighting strategy increases the performance of the tested binary classification tasks which used tabular data, when compared to the centralized approach or other weighting strategies.
4. The provided model usage scenarios enable privacy-preserving network knowledge extraction for individual usage or future development.
5. The provided model combination method allows combining heterogeneous model types, thereby increasing the ensemble diversity and allowing for increased collaboration opportunities.

Scientific approbation

The dissertation results have been presented in 5 academic publications: two publications in a periodical scientific journal (*MDPI Applied Sciences*, Q2) and three in proceedings of international conferences. The complete publication list with the referral information can be found in the SCIENTIFIC JOURNAL AND CONFERENCE PUBLICATIONS section.

Structure of the dissertation

The first chapter presents the results of exploratory analysis on distributed machine learning approaches, as well as the privacy preservation methods for the machine learning blockchain technology. Moreover, cooperative analysis of the currently existing blockchain-enabled collaborative distributed machine learning approaches is presented in this chapter. The second chapter defines the proposed collaborative distributed machine learning method and specifies requirements and procedures for the blockchain platform preparation, model and data deployment, and network knowledge usage parts of the method. Each individual method step is provided with a formal definition and a demonstration of its implementation. The third chapter presents empirical assessment of the proposed method. The dissertation conclusions are presented in the fourth chapter. The dissertation also provides a summary in Lithuanian, a list of references and scientific publications, and a conference list.

1. ANALYSIS OF EXISTING METHODS AND SOLUTIONS

The analysis of the currently existing research and relevant solutions is presented in this chapter. As the research consists of a combination of a few research areas, all of them are overviewed individually as well as by adopting multidisciplinary approaches. The analysis starts by describing machine learning concepts and prominent research works in the distributed machine learning field. Whereas, in the second part, an overview discusses methods for collaboration in distributed learning and privacy-preservation methods applied to distributed learning. Lastly, analysis of the blockchain technology, its architecture, and solutions associated with distributed machine learning is presented.

Currently, organisations and people that could participate in the collaborative machine learning and would benefit from the high-quality models in their workflows might not be sufficiently motivated because of multiple reasons. Due to sensitive or private data used in the processes, the parties could be limited to the amount of data they can share with other participants. Data sharing in the competitive environment could also introduce risks of providing advantage to one's competitors. This reduces the trust in the collaboration, and, if trust is not enforced by the developed system, the solution becomes not viable. Collaboration might also be hindered by security concerns, especially in the domains that contain sensitive data, such as healthcare and finances. Additional drawbacks exist for parties that are already using machine learning solutions in their domain(s), as the transfer from the local model usage to collaborative solutions might require additional development, which requires changes in the existing data management and model training pipelines. Such changes can disrupt the already developed solutions and procedures. Thus, the introduction of the collaborative process should bring significant improvements to negate the costs. Addressing these challenges and drawbacks requires a complex approach which is bound to focus on building trust among the participants, developing tools and methods to facilitate the adaptation and integration of the existing machine learning solutions into collaborative distributed machine learning environments. By addressing these challenges, collaborative distributed machine learning approaches can unlock the full potential of distributed data and resources, which would lead to more effective machine learning solutions.

1.1 Machine Learning

In the recent years, machine learning approaches have become more widely adopted [10], [11], [12] in various industries and research areas due to the ability of technologies to process large amounts of data and learn from patterns. Machine learning has been applied to image and speech recognition, with applications that are able to detect objects as well as text, and identify faces. Natural language processing is also another area of machine learning applications; research and application in this area introduced the usage of the language translation service, and chatbots of text analysis tools. Machine learning not only can make detection and classification tasks easier, but it has also been applied in such research areas as the healthcare, where it enables discovery and development of medical drugs and is even used to aid disease

diagnosis. Although some of the machine learning application areas are still developing, its usefulness and success have already been proven in such domains as finance and banking. All machine learning solutions apply some kind of a machine learning model. This model must be developed by providing data and computation resources. The machine learning model development procedure commonly consists of multiple stages (Figure 1). The machine learning model development procedure [14] consists of four main parts: data preparation, model tuning, performance evaluation, and model usage.

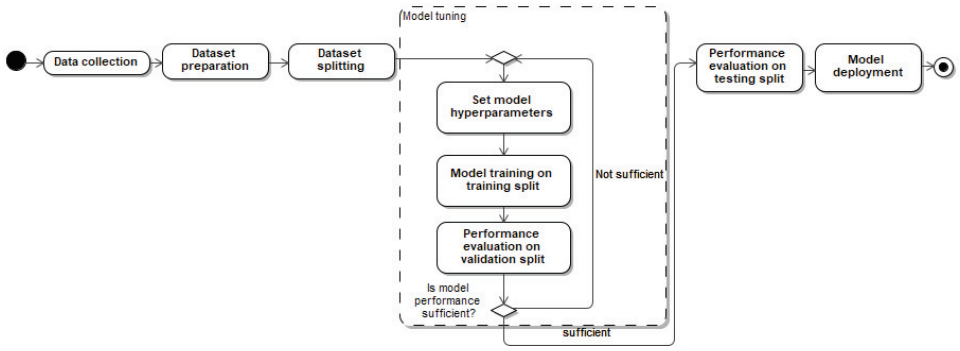


Figure 1. General machine learning model development process [14]

The data collection process is the first stage of the machine learning process. The data used in machine learning processes can be obtained by using multiple approaches [15], [16], ranging from crowd sourcing to data discovery or data synthesis. On the basis of the machine learning approach used, the collected data might require labelling. The labelling process maps the real-world objects to one or multiple labels. Labelling processes are required once the data have been collected, and the next step is to preprocess the dataset.

Multiple approaches to data preparation can be employed, and the required procedures vary depending on the dataset structure and the data themselves. In general, data preparation removes empty or erroneous data instances, cleans the data by removing duplicates, and standardises the data instances. The features of the dataset are then explored and selected. The feature exploration reveals patterns occurring in the data and relationships between the variables if such exist. If a selected dataset contains categorical data, it requires encoding to modify data formats from text to numeric features [17], while text-represented data are encoded into numerical data creating the required number of numeric classes. The data preparation process concludes with a modified dataset that is ready to be used in the model training process.

Typically, for tuning and performance evaluation, the dataset is divided into three parts: training, validation, and testing [18]. The training part is used to develop the model, validation is used in model training to tune the model hyperparameters, and the test dataset is used on the developed model to evaluate its performance.

Machine learning approaches can be divided into three categories based on the input data type and the application type. Supervised learning approaches use input data and the corresponding output data labels [19], [20]. Supervised learning approaches tune a function which maps the input data to a desired output [20], [21]. On the other hand, unsupervised learning approaches are only provided with unlabelled input data [22]. Unsupervised learning approaches develop a function that is used to describe the data structure of the input data and group them [23]. Unsupervised learning is commonly used to cluster data based on similarities and data patterns. Reinforced learning is a machine learning method that allows a program or a machine to perform actions in order to gain the maximum reward [24].

Multiple types of machine learning models [25], [26] could be used in both supervised and unsupervised settings, such as logistic regression [27], *Classification And Regression Tree* (CART) classifiers, or neural networks [28]. Based on the type of the machine learning model selected, the training procedure is performed in batches or on a larger part of the original dataset. Training approaches that use an unbatched dataset, such as the logistic regression decision tree, usually try to fit a function to detect a data instance class on newly provided data.

One of such approaches requiring data batching is the *Stochastic Gradient Descent* (SGD). The SDG algorithm minimises a loss function based on the model outputs by modifying the model parameters to follow the general trend of the negative gradient. The gradient is defined as stochastic due to the subset of data randomly selected for training. The common training procedure for SDG algorithms starts with the sampling of a batch of data to form a training dataset. Then, during training, the model predictions are calculated, and the model output is compared with the true output via the loss function. The gradient of the validation data batch is calculated, and the model parameters are adjusted repeatedly. Multiple loss functions are used to evaluate the model performance. For example, the most common metric in the binary classification is the *Area Under the Curve* (AUC) [29], [30] which is derived from the plot of the *Receiver Operating Characteristic* curve (ROC) [22] – [24]. Since the model output as a soft decision is a floating number between 0 and 1, the threshold is needed to arrive at the hard decision (predict the class – either '0' or '1'). The default threshold of 0.5 may not be optimal; therefore, the testing of all possible thresholds provides a more comprehensive evaluation. A specific threshold results in a trade-off between the accuracy of the target class (*true positive* or *hit rate*) and the error of the nontarget class (*false positive* or *miss rate*). The ROC curve summarises all possible thresholds with their corresponding trade-offs. Often it is convenient to have a single number instead of a performance curve, and, in the case of ROC, this number is AUC. The statistical interpretation of AUC is as follows: the probability that the output of the classifier will have a higher value for a randomly chosen target class instance than for a randomly chosen nontarget class instance [31]. ROC curves are not always the most viable approach when the dataset class rate is highly disbalanced, and, in that case, other loss functions may produce better results. Another comprehensive performance metric for binary classification is the *Binary Cross-Entropy* (BCE), also known as the *log-loss* [32], [33].

Finally, the validation subset of the data is used to measure the performance of the classification or regression task on data that were not used in the training process in the performance evaluation step. After training the classifier and evaluating its performance, the parameter adjustment [34], [35] can be repeated. Model parameters can be tuned manually, but such a process is time consuming and complex as machine learning models contain a large number of parameters, and the relationships between multiple parameters may be unclear to the developer. Automated approaches [36] to tune parameters are more commonly used.

When the parameter tuning has been completed, the model performance is evaluated on testing the data split. With sufficient performance, the model can be stored into a file, deployed into interactive environment, and used for the classification of regression tasks [37], [38].

In every stage of the machine learning model development process, security must be ensured by the developer. As the machine learning process deals with training data that may contain sensitive information such as pictures, names, or even social security numbers of individuals, confidential information about the business, the security and privacy of such data must be ensured [39]. During the data management stage, the model developed must ensure the integrity of the data, apply the required data anonymisation actions, and ensure that the usage of the data would not reveal sensitive information [6]. Multiple data protection methods have been proposed to secure private information such as differential privacy [40], data encryption [41], and multiple others [39], [42], [43]. The model training procedure should also ensure that any data that are used remain undisclosed to external adversaries and do not reveal information about any individual or any other entity. Part of the model training security measures are designed to protect the loaded training data in the hardware performing the training procedures [44]. Other security methods which do not rely on hardware protection have also been proposed for secure model training to ensure the integrity of the system and protect the valuable assets [6], [39], [42]. The developed model can also be used in revealing sensitive data without the proper security measures via the membership inference [41] or model inversion attacks [42].

In the context of this dissertation, the overviewed methods and the presented solutions are confined to the machine learning classification task. Some approaches show better performance when more than one classifier type is being used. Multiple classifiers can be combined into a single classifier defined as a *model ensemble* [45].

1.1.1 Ensemble learning

Ensemble learning is a technique where multiple machine learning models are combined in order to obtain a better performance model [46]. Many machine learning model types can be used to develop ensembles [45], [47]. One of the most popular approaches is ensembles developed by using CART [48] classifiers. The benefit of developed ensembles stems from the diversity of combined classifiers and how different classifiers can represent the same real-world entity in a different manner for the same training dataset. This is especially useful because decision tree classifiers are diverse in their development processes based on decisions and data partitioning.

The most popular CART classifier aggregation to ensemble methods is bagging and boosting [49]. The bagging method utilises the sampling of random subsets from the training dataset and uses these to train a single model for ensemble creation. Whereas, the boosting approach utilises iterative processes to continuously adjust weights of falsely classified samples. Another approach that uses a subset of the data to train parts of the ensemble classifier is called the random forest approach [45]. The random forest approach develops multiple CART classifiers by sampling not only subsets of data, but also a subset of features.

The ensemble creation techniques display better performance than a model developed without ensembling. They are commonly used as local not distributed algorithms, which requires centralised data for ensemble building. However, some online ensemble development techniques exist and are capable of training by using data streams. Two examples of such a technology would be the online *AdaBoost* [50] or online *Arcing* [51] approaches. The latter approach [51] is closely related to online learning; it emphasises training on more recent data batches. The approach utilises a fixed-size ensemble of classifiers and builds a new one on new data batches. This newly created classifier then replaces the worst performing one from the fixed-size ensemble. Such a replacement is performed after the validation of new data. It can be achieved in two possible methods – either directly, by using the streaming ensemble algorithm [38] by using the majority weighting, or indirectly, by using the best performance-weighted classifiers, as in the performance-weighted ensemble [52].

There are many ensemble merging techniques in existence. One such approach is the weighted averaging of ensemble predictions [53]. Multiple strategies to tune the ensemble weights are available [47], [54]. The majority voting case allows using every classifier's input [54], while the performance-based method evaluates the model performance and uses it to define weights [47], [55]. The performance-based weighting process consists of two steps: the model evaluation and the ensemble weights adjustment based on the model evaluation. Model stacking is another approach that can also be used to derive final ensemble predictions, as in the meta-learning approach [56]. In a recent development, a novel approach for achieving an optimal ensemble was introduced in [26]. The approach involves combining the tuning of hyperparameters and weights specifically tailored for regression tasks. Additionally, the utilisation of effective weights can aid in arranging classifiers for ranking-based ensemble selection, thus effectively filtering out nonuseful classifiers as part of the ensemble reduction. It should be noted that search-based approaches generally outperform ranking-based methods in terms of accuracy, as highlighted in [25]. However, due to the optimisation process required for weight tuning, search-based approaches are more computationally intensive compared to their ranking-based counterparts.

Ensemble development can be divided into two categories. Homogeneous ensembles use a single type of the machine learning model as in the random forest approach, whereas heterogeneous ensembles [45], combine multiple model types to develop an ensemble. The development of heterogeneous models is usually made up of two phases: the development of a diverse set of machine learning model types by

using the same training dataset, and the model selection for combining them into an ensemble.

Although ensemble learning is a popular approach towards improving the classifier performance, it is usually still developed by using a single dataset and using a model development environment. To better solve machine learning problems that have multiple data producers or require collaboration between multiple data producers, distributed machine learning approaches have been proposed, which commonly build upon ensemble learning approaches.

1.1.2 Distributed machine learning

Distributed machine learning differs from centralised machine learning in the sense that it has multiple parties which produce data or train multiple or single models that are later combined into a single model. Similarly to ensemble learning, there are three types of information that can be combined in the distributed machine learning setting – classifiers, classifier representations, and classifier predictions.

Based on the number of participants, the distributed machine learning solutions are divided into two categories: individual and collaborative. Solutions that are developed by a single entity by using a distributed setting that usually uses a centralised server to aggregate model or data artefacts from multiple devices or software-based solutions are defined as individual distributed machine learning. Such distributed machine learning approaches do not require defining complex collaboration processes and rules. Meanwhile, collaborative distributed machine learning combines input from multiple entities, and comprises multiple collaborating parties at any given time which would benefit from sharing data and machine learning models to receive a classifier of a higher quality. The key difference between the two approaches is that collaborative learning requires trusting the participating parties and used services as, for the individual approach, all services are naturally trusted. Collaborative distributed machine learning also requires additional security and privacy measures because other parties might be malicious, or communication between such parties could be compromised, by revealing models or data that are sent in the course of the communication.

When combining machine learning model representations, a few limitations may arise. Machine learning models may be represented in a wide range of formats that are imposed by different learning libraries and development environments. To address these discrepancies between the model file representations, model representation unification strategies are employed. In the process of applying the unification strategy, some internal model context is lost [57]. The unified model can lose so much context that the data would become unusable after unification [57].

Contrarily, when combining classifier predictions, internal model representation is not as important because the predictions need to be presented in a unified format. Predictions can be represented either in the numeric or in the categorical format. The transformation between numeric and categorical data can be achieved by setting the numeric label for categorical data.

Similar strategies that were presented in the ensemble learning section can be achieved in a distributed setting, as distributed learning relies heavily on model

combination techniques, such as boosting [58], stacking [59], or meta-learning [60]. Based on the solution implementation architecture, distributed machine learning [61] can be divided into two categories:

Centralised: it contains multiple data producers uploading raw or aggregated knowledge to aggregate information at a single location [62].

Distributed network: here, a peer-to-peer network with multiple participating members exchanges knowledge and participates in the network management [63]. Distributed networks can also operate by the collaboration of multiple participants who share data, machine learning models, or model representations. In the context of this thesis, distributed learning is defined as machine learning with multiple participants who share data and model artefacts in a peer-to-peer manner.

From the standpoint of the machine learning model aggregation and the machine learning model development, the methods can also be divided into centralised and distributed. When the computations are completed by a single machine and contain a local dataset, such an approach is defined as centralised [61]. But, due to large amounts of data, or because of data distributed over multiple storage locations, a centralised approach is not always viable. The technologies that aggregate data from multiple sources or divide computations into multi-agent environments are defined as distributed solutions. Such solutions were popularised by the map-reduce framework [64]. Such a distributed computation model inspired the development of many distributed learning frameworks and libraries, such as *PyTorch* [65], *Apache Spark* [66], and others [67], [68]. The distribution of computations allows developing more complex models by using a higher amount of data when compared to centralised approaches. The application of such a technology is also explored in the proposed method implementation by using *Apache Spark* as one of the tested model development frameworks. The *Apache Spark* framework is only tested as a technology that enables distributed model development by the method participants by using their local environments. It is not used to distributed computations when models are used in the proposed blockchain platform.

Some of the distributed learning solutions do not require the sharing of raw data [69], while others [70] rely on the sharing of small datasets for the testing or validation purposes [71]. One of the most common approaches in distributed learning is the federated learning solutions.

1.1.3 Federated learning

Federated learning represents a distributed machine learning technique which enables the training of models on a network of decentralised contributors. In contrast to the conventional machine learning methods relying on the centralisation of data for model development, federated learning works by training models on numerous decentralised devices [26]. Rather than uploading all data samples to a single server, devices retain their local data samples without sharing them with other devices [27] – [29]. Machine learning models are trained locally on edge devices, or by collaborating member environments and the produced model, which is sent to the model aggregation server. The model aggregation server receives multiple models from multiple providers and updates the global model. The model is then returned to the

network participants to use with an increased performance. Such an approach when the network contains multiple data or model information providers is defined as *horizontal federated learning* [72]. Horizontal federated learning is the most common application of federated learning, with a high amount of research interest in combining federated learning and the internet-of-things domains. Another federated learning approach is *vertical federated learning*, where, instead of providing multiple instances of the same data, the network participants provide different data features dedicated to the same machine learning problem. Due to its popularity and the dependency of the machine learning process on high quality and quantity data for machine learning, the analysis will focus on horizontal federated learning. Federated learning offers utility in these main aspects:

- **Load-balancing**: here, load balancing is used to distribute computations for machine learning. As the machine learning computations are performed in a participant environment, the computational load is naturally distributed to be merged at the central server.
- **Privacy preservation**: it allows for participation in the machine learning process without revealing the training data. The training data privacy in a federated learning process is provided by the retention of the training data in the participant devices. Similarly, model encryption and secure model aggregation can be applied to increase the privacy of the training data.

Several innovative solutions have emerged to distribute the training process [71], [73], [74]. One such example is *Google*'s implementation, where they utilise the cloud infrastructure to distribute the *Gboard* (Google keyboard) [75] model across multiple devices. The model is initially deployed and later improved with the user data. The improving process is completed via model updates from the user-provided model gradients. These model updates are then combined in the cloud-based web service and incorporated to enhance the shared model even further. The underlying principles of this approach demonstrate that decentralisation can effectively support the training of a single common machine learning model.

Nvidia presented another innovative solution for federated learning, aiming to create robust machine learning algorithms which enable different nodes to collaborate on the training model while maintaining data privacy [76]. In this approach, Nvidia employs a server-client framework where a centralised server acts as a manager/facilitator for various clients. By using the proposed architectures, the users can collaborate in the model training process while still retaining control over the local model training. The process of collaboration involves training local models on individual devices, followed by sharing local model weights, which are then used to update the global model based on the weights and historical contributions. Although the infrastructure supports the use of different models, it still relies on a centralised node to aggregate weights and update the shared model, which leads to a bottleneck in terms of resources and computational power.

As federated learning systems are composed of many data producers, the domains that contain elements distributed in the physical space and domains that

require an increased privacy are the key application areas. For example, regarding the Internet of Things (IoT) [71], [73], healthcare [77] is one of the most popular areas of its application due to the distribution and the higher demand for privacy. In the context of IoT applications, federated learning is applied to improve on data sharing, security, and crowdsensing issues. Although federated learning is applied in multiple domains, the following challenges are still present in machine learning solutions [78]:

- Federated learning solutions rely on centralised processing. Centralisation of the process may lead to the system being more susceptible to denial-of-service attacks, where negated access to the centralised server disrupts the ability to use the entire system [79], [80]. With centralisation come the performance scaling issues, as, with a greater number of data providers, the possibility of performance overload increases [79], [80].
- Federated learning approaches lack an incentive mechanism. Without sufficient motivation, the number of network participants can decrease. With sufficient incentives, the drawback could be alleviated, and new members could be motivated to join [43], [81].
- Many of the proposed solutions lack robustness and could be vulnerable to poisoning or Byzantine attacks. The results of such attacks could seriously reduce the accuracy of the system and even lead to the denial of service [82], [83].

The analysis of federated learning approaches revealed that federated learning can be applied to multiple domains, and that its approaches are popular in the Internet of things, healthcare, and other domains. Even though federated learning uses a separated network of data providers, most of the implemented solutions still rely on a centralised network architecture. This creates a weak point susceptible to attacks and might reduce performance with high user counts when comparing it to a distributed system. Furthermore, many of the proposed solutions use a single type of machine learning model or are dedicated to a single machine learning task, thus reducing the flexibility of the system, and hindering collaboration.

1.1.4 Summary of machine learning approaches

Each machine learning model training architecture could have advantages over another option based on the machine learning problem, the number of participants, and the specific requirements for the task. The local model development might be the most applicable in those cases of use where the model development can be completed with the local computational resources, and when the datasets used contain enough instances to develop the model. Multiple participants can collaborate in the local machine learning model development by merging datasets before training and sharing the developed model afterwards. Federated learning approaches are mainly applied when there are multiple data providers, and a single entity performs the model training and tuning. Federated learning mostly trains complex neural network architectures which can support a high amount of knowledge. The complex development and long

model tuning of federated learning limits the adaptability of other model types or the reusability of a single architecture for multiple machine learning problems. Distributed machine learning is capable to support both centralised and decentralised architectures and a wide range of collaboration approaches, from data sharing to model sharing. When collaborating via the data sharing, the data is merged before training as the model sharing approach requires more complex model merging strategies. The comparison of machine learning model training architectures is provided in **Table 1**.

Table 1. Comparison of the machine learning architectures and approaches

Criteria	**Local model training**	**Distributed machine learning**	**Federated learning**
Data accessibility	Offline	Online	Online
Data sharing	-	Not required / required	Not required
Complexity of support for new model types	High	High	Highest
Multiple participants	No	Yes	Yes
Possible system architectures	Centralised	Centralised or distributed	Centralised or distributed
Means of cooperation	Model and data sharing	Model and data sharing	Model gradient sharing
Limitations	Collaboration requires complete data sharing and trust between the participants	Requires model combination and may require significant development time to support new model or data types	Inclusion of a new model or a new data type to federated learning networks is complicated due to required high model training and development time

All machine learning architectures and methods, including distributed and federated learning, require security and privacy. Viable privacy preserving measures are overviewed in the next section.

1.2 Privacy Preservation Methods in Machine Learning

Privacy preservation is an important aspect in every stage of the machine learning pipeline. The importance of data privacy and security has been demonstrated by data standards [84] and legislation measures [85]. Different stages of the machine learning process deal with different privacy issues and demand distinct privacy ensuring measures. Data management and processing stages of the machine learning pipeline require secure and privacy-preserving data management to ensure that the

data are not revealed to the public or malicious adversaries. Model training procedures must ensure that neither the training process nor the developed model would reveal any sensitive training data. Model deployment should not reveal information about the training data in use and ensure that sensitive information is not accessible via derivative artefacts or model predictions.

1.2.1 Privacy preserving machine learning

The machine learning model development process utilises data representing real-world entities. The data for machine learning could represent publicly available data or private personal data. The disclosure of private data could have serious implications for people, companies, or governments. Therefore, when dealing with sensitive private data, privacy preservation measures and methods should be employed. Based on the used machine learning approach, all parts of the machine learning pipeline could require privacy preservation, ranging from private data acquisition and management to secure model deployment and usage.

Data privacy methods are important in the machine learning model development process because attacks may reveal sensitive data. Three main groups of approaches to ensure privacy in machine learning processes are: anonymisation of the data [86], [87], [88], cryptographic approaches to secure transferred information [89], [90], and architectural approaches [91], [92], [93].

The first category of data privacy preservation approaches is to anonymise the training data [86], [87]. Traditional data anonymisation techniques, such as k-anonymity [86], l-diversity [87], and t-closeness [88], define methods to select data that do not reveal identity information.

Similar approaches which are designed to select a subset of data which would not reveal the identity of any individual data points are defined as *differential privacy* [40]. The security and integrity of the data preparation step performed should not disclose any sensitive information or modify the provided dataset to introduce new features or unique conditions which would reveal the identity of any real-world entity.

The privacy preservation approach also depends on the selected model training implementation approach. When training the model in a local environment, data privacy is naturally ensured [94]. Model training privacy preservation requires the introduction of noise in a trained machine learning model so that not to reveal the identities of the training data by using model inversion attacks. Model training architectures and approaches requiring communication with external sources or an external training environment should ensure the security and privacy of the communication channels and the information that is transferred through these channels [89], [90]. Similarly, if the model training stage requires sending derivative information of the trained model, such as predictions or trained neural network gradients, the privacy of such artefacts must be preserved [68], [95]. In cases when the model developer and the user of such a model are separate entities, privacy-preserving machine learning model deployment is required. In such a model deployment approach, model compression [91], model transformation [39], or knowledge transfer approaches are usually applied [92], [93].

In the context of this dissertation, the data anonymisation and machine learning model development steps are computed individually by each participant and performed in diverse self-managed environments. The privacy of such environments depends on the owners, and, as training data and model training processes should not require any communication, their privacy should be naturally preserved [94]. The next section will overview the applicable privacy preservation approaches for the model deployment stage of the machine learning process.

1.2.2 Knowledge distillation approaches

After securing the training data and the machine learning model development processes, the solution developer should ensure privacy-preserving of the model deployment processes [40], [96]. Multiple solutions for private model deployment have been proposed [97], [98], [99]. Most of the proposed solutions apply a modification of knowledge transfer from single or multiple models to a new neural network model to secure sensitive model parameters. The propositions for such a system defined as the *Private Aggregation of Teacher Ensembles* (PATE) were suggested by [92], [93]. The PATE approach (**Figure 2**) utilises multiple subsets of sensitive data to train a number of teacher models. The proposed method divides the initial sensitive dataset into an N number of datasets and uses these subsets to develop multiple teacher classifiers. The developed teacher models are combined into a model ensemble. The ensemble predicts by voting with noise added to the voting histogram. The aggregated teacher delivers the predictions on the public dataset provided. These predictions are used to train the student model in conjunction with public unlabelled data. The PATE model security relies on access restrictions. The data preparation and aggregate teacher development stage must not be accessible to malicious actors, as these artefacts contain sensitive information. After the creation of the student model, such restrictions are not required as the model cannot reveal any sensitive information about individual models or sensitive data on which it was trained, thus increasing privacy.

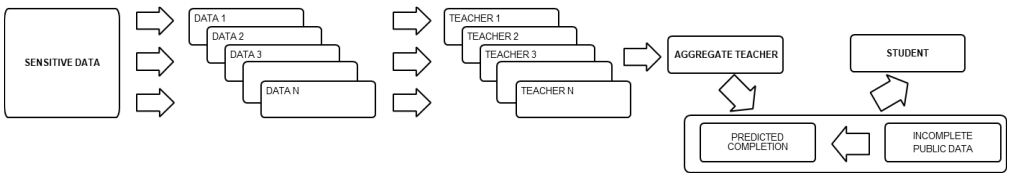


Figure 2. Overview of the PATE approach, proposed by [92]

Although the PATE approach proposes using a single dataset that is divided into subsets, a similar approach could be applied when multiple data producers are collaborating on a defined machine learning solution, while using similar data structures [100], [101], [102], [103].

Solutions that expand the original proposition of PATE or utilise its concepts in different approaches have been proposed. In the field of federated learning, the authors of [103] proposed a modification of the PATE approach. The updated knowledge transfer approach uses multiple data providers with individual datasets. The need of

unlabelled data for student model training is removed, and model distillation via a generative learning approach is defined. Distillation approaches can be divided into offline [104] and online [105] approaches.

Other modifications of the PATE approach have been proposed [106]. Some of the proposals (see, e.g., [106], [107]) also investigate the ability to model teacher knowledge as graphs which transfer information from teacher models to the student model. Other research concentrates its attention on the usage of a diverse group of teacher models for knowledge distillation [108], [109], [110]. Many other improvements and modifications exist which improve the initial proposition by shifting the research focus to data features [111], [112] or to attention [113].

Multiple research papers have proposed a combination of knowledge distillation and blockchain technology. An overview of such research is presented in the following section.

1.2.3 Knowledge distillation approaches and the blockchain technology

Research combining knowledge distillation, federated learning, and the blockchain technology has been explored in [95], [114], [115]. Combining the blockchain technology can improve the security and robustness of the proposed system [95]. The introduction of knowledge distillation allows more diverse model types to be supported in a federated learning environment [95].

The authors of [95] proposed a knowledge transfer approach which enables the support of multiple types of machine learning models in federated learning. The authors utilized blockchain to mitigate the single-point-of-failure risk and provide logging of the network participant actions to an immutable ledger. The authors developed and experimentally tested two types of smart contracts. The first one is dedicated to managing federated learning processes, whereas the second type focuses on storage management.

A novel decentralised federated learning method was proposed by [114]. This method is designed to reduce the amount of communication between the network participants and improve the network stability. The proposed method introduces a ring-based topology and uses *Ethereum* with the *Interplanetary File System* to store information about transactions and machine learning models, respectively. Knowledge distillation is also utilised to aggregate multiple models.

The continuous knowledge transfer approach for edge computing was also presented by [115]. The proposed method was implemented by using the *Hyperledger Fabric* blockchain and smart contracts. The blockchain was used to store data about the deep learning training process artefacts and network transactions. The network was experimentally evaluated the use of computer vision tasks and ensured that such solutions could be implemented in practice.

Most of the approaches that combine the blockchain technology apply it to mitigate the drawbacks of centralised federated learning approaches [114], [115]. A combination of federated learning, blockchain, and model aggregation has been proposed [70], [71], [82], [116]. But knowledge distillation is not the only privacy preservation approach that can be applied to federated learning on blockchain. In the next section an overview of approaches using other privacy preservation methods

when combining distributed learning, blockchain and privacy preservation measures is presented.

1.2.4 Blockchain enabled privacy preserving machine learning

Approaches combining blockchain with privacy preserving machine learning are overviewed in this section. The results of the compared research are presented in Table 2. The approaches are compared on the grounds of the blockchain technology usage, privacy-preserving methods, machine learning model types, the blockchain technology, and the consensus algorithm. In addition, if the proposed approach or method were to include the network participants' roles in the machine learning process, such roles are listed. Some of the overviewed approaches introduced an incentive mechanism into their process, either by providing blockchain tokens or monetary value to the network participants. Such incentive mechanisms are also considered as criteria because they could motivate participation. The application area criteria were defined if the proposed method was designed for a specialised application where single or multiple areas are listed; otherwise, the analysed approach is classified as universal. For those research propositions which do not specify the machine learning method, the definition was omitted.

Table 2. Comparison of existing blockchain enabled privacy preserving ML approaches

Criteria	ML model type	Network user and roles	ML architecture	Privacy preservation	Application area	Blockchain	Consensus	Limitations
[117]	Any	-	Mixed	Differential privacy	Healthcare	Ethereum	Proof-of-Authority	Dependency on external data storing solution, single method application area
[74]	GRU	-	-	Not applicable	Traffic flow prediction	Non-defined, Consortium	DBFT	Support for a single ML model type, limited application areas
[118]	CNN	Miners; Cooperative party	-	Not applicable	Universal	Corda	Blockwise-BA, algorand	Blockchain is used only for incentive and data logging
[119]	Logistic regression; Multinomial classification	-	-	Differential private logistic regression; Differential private multinomial classification	Universal	Hyperledger Fabric	BFT	Limited by the smart contract development language support
[70]	Neural network	Data holder; Computing node	Distributed	Differential privacy; l-	Universal	Ethereum	Proof of training quality	Requires modification and

Criteria	ML model type	Network user and roles	ML architecture	Privacy preservation	Application area	Blockchain	Consensus	Limitations
				Nearest aggregation				development of a new blockchain structure
[71]	Gradient boosting decision tree	-	-	Differential private federated learning	Industrial IOT	-	Training quality based	Requires development of a new blockchain structure and consensus algorithm
[116]	CNN	-	-	Not applicable	Universal	Custom	Committee consensus mechanism	Blockchain is only used for data logging, requires a novel blockchain structure
[82]	Logistic regression, neural network	Noiser; Verifier; Aggregator	Distributed	Differential privacy; Pre-Committed Noise to Thwart Poisoning	Universal	Custom	Proof of Federation	Security oriented, requires novel blockchain development
[120]	Neural network	Buyer; Reporter; Data processor	Distributed	Encryption of model gradients; Auditability of collaborative training	Universal	Custom	Reliability-based consensus	Only supports a single classifier type, requires custom blockchain

Criteria	ML model type	Network user and roles	ML architecture	Privacy preservation	Application area	Blockchain	Consensus	Limitations
[63]	Neural network	Miner and trainer	Distributed	Differential privacy; Random noise addition	Universal	Custom	Proof-of Work	Expensive and slow consensus algorithm, limited model type support
[121]	CNN	Parameter providers	Distributed	Zero-knowledge proofs of model gradients, encryption	Universal	Ethereum	Proof-of Work	Slow consensus algorithm, requirement to encrypt data, single model type support
CDMLB method	Universal	Data and model contributors	Distributed	Intrinsic privacy, knowledge distillation	Universal	Hyperledger Fabric	Raft (BFT)	Limited tested datasets, limited evaluation of method performance and scalability

Most of the overviewed approaches supplement the blockchain with novel consensus algorithms [70], [71], [82], [116] that are implemented in order to evaluate the performance of the distributed machine learning models. These approaches are usually implemented by using the public blockchain [117], [70] as the base technology and are mostly presented as only proof-of-concept approaches which require implementing new blockchain networks, that are complex and might be a non-feasible solution for practical implementation. With novel blockchain networks, some approaches also introduce new roles for the blockchain network participants, most commonly including roles for data contributor and data validator peers.

The most common privacy preservation measure in the compared approaches is differential privacy and the encryption of communication channels. Differential privacy measures are applied in order not to reveal sensitive information about any single data instance used in the data pool. A similar purpose in context of privacy preservation serves various noise addition techniques that are used to remove patterns in data that could reveal sensitive information. In some overviewed solutions, privacy is ensured by data encryption (Encryption of model gradients, Zero-knowledge proofs). Such approaches are commonly used in public blockchain environments to make parts of information available only to the required participants while preventing data leaks. Zero knowledge proofs are used to verify the existence of information without decryption, thus simplifying the auditing and verification purposes. The third category for privacy preservation is provided by model aggregation, such as the l-nearest aggregation and PATE-based approaches. Commonly, collaborative distributed machine learning approaches combine multiple privacy preservation methods in a single system based on the requirements. Even though part of the overviewed methods do not explicitly state the usage of encryption approach on the data, the communication channels and collaboration algorithms powering data sharing are usually encrypted due to the best practices of common security. For the method proposed in this thesis, it was defined that the most important privacy preservation drawback is the membership inference attack via the deployed model, which would enable access to training data information. Thus, the model distillation approach was selected.

The application areas for most of the proposed approaches fall into two categories: domains requiring sensitive data preservation, as in the healthcare domain [117]; multiple data creation and usage parties that are distributed in a physical space with the need to collaborate on common tasks, as in the internet of things domain [71], [74]. Universal approaches that could be applied to multiple domains have also been proposed, but most of them are considered only for deep learning [120], [121] without support for other types of machine learning models. Most of the compared approaches are only implemented on a single machine learning environment without considering heterogeneous approaches, thus reducing the usability of the developed approaches. The proposed method is also commonly confined to a limited set of supported programming languages, while only supporting a smart contract-provided set. This limits the ease of access and increases the development costs for participants with the already existing machine learning solution support. Most of the overviewed solutions either use a relatively slow consensus algorithm, such as PoW or its modifications, or

propose the usage of a custom blockchain network and solutions whose implementation would require a significant development effort. Most of the overviewed research only experimentally dealt with a small part of the proposed method while commonly not including analysis of the blockchain network performance. The common evaluation topics of the proposed methods are the security and privacy analysis and the incentive provision methods.

1.3 Blockchain Technologies for Machine Learning

Distributed Ledger Technologies (DLT) have been proposed [122] and implemented in the business and scientific communities. But only the blockchain has received the mainstream appeal, starting with the cryptocurrency called *Bitcoin* introduced by Satoshi Nakamoto [123]. The blockchain is a network composed of multiple participant peers storing the transaction information in a state database called the *ledger*. The ledger records all interactions on the blockchain and is replicated to all the participants on the network for validation and safekeeping. Transactions are cryptographically hashed and then bundled into blocks and linked to each other. The blockchain ledger is powered by the blockchain data structure and cryptography approaches, such as hashing. The hashing algorithm can transform any amount of information and produces a unique fixed length value as a result [124]. To ensure that all transactions are composed correctly, the blockchain technology also employs a Merkle tree algorithm [125] which enables simpler verification of validity for multiple hash values. Such an approach allows us to use a single value to verify that nothing in the information composition has been changed. Blockchain uses hashing algorithms to uniquely identify block information and verify its integrity. The block structure stores information about the performed transactions in a given time frame. A block is composed of two types of information: the block header and the block body (**Figure 3**). The block header stores meta information, such as the previous block hash, the version, the timestamp, the nonce, and the difficulty target. A previous block hash is a unique identifier which is used to associate the present blockchain block with the previous block, thus ensuring that the information is linked together. The timestamp is used to record the time when the block was created, and the Merkle tree root is used to verify that all the created transactions are correct. Nonce is a calculated value that is used in the block creation. The body of the block stores all information about the transactions.

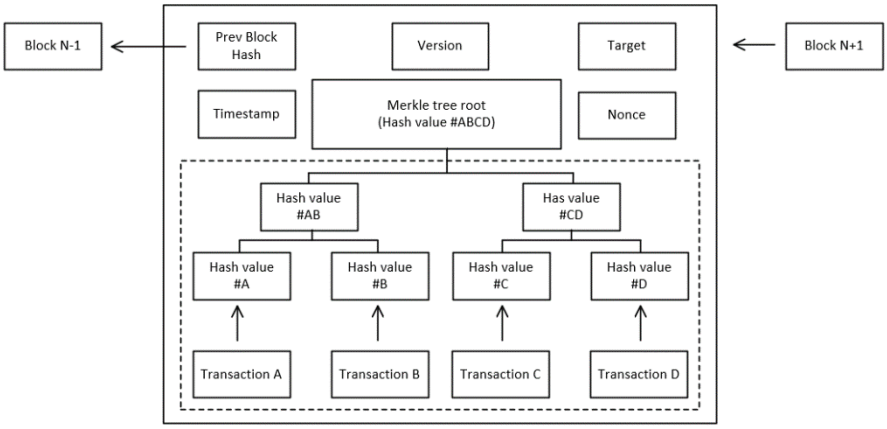


Figure 3. Blockchain block structure [2]

The Blockchain technology is designed to provide transparency, data consistency, verifiability, and data integrity [123], [126], [127]. The Blockchain technology utilises a distributed architecture, thus eliminating the need to rely on a centralised party. Distribution also increases the robustness of the solutions, which means that it is harder to deny service. Multiple copies of the ledger are distributed among the network participants for safekeeping. Such data replication ensures that the data are secure even though multiple parties can stop participating in the network.

1.3.1 Blockchain platforms

The Bitcoin cryptocurrency introduced a novel approach in the distributed ledger technologies combining transaction logging, peer-to-peer technology, and consensus algorithms into a single system [123]. Multiple other blockchain platforms were inspired and developed by the modification of the initial Bitcoin platform. The term *blockchain* [128] defines a data structure which uses cryptographic hashing algorithms to link multiple blocks of information in an immutable chain.

The process of validating a single block of transactions and agreement on the validity of the block to add to the blockchain is defined as a consensus algorithm [2]. Communities managing data deployments which store copies of immutable blocks are called *blockchain networks* [2]. Blockchain networks can be classified according to their member acceptance model and the required level of trust [129] into *public*, *private*, and *consortium* networks.

The public blockchain is openly accessible to any new members who wish to join the network. Every registered network member is able to participate in a consensus algorithm and is allowed to store a copy of the ledger. The public blockchain provides incentive mechanisms for network members who participate in the consensus algorithm by distributing tokens for contributions. The most popular examples of public blockchains are *Bitcoin* [123] and *Ethereum* [126]. As access to these networks is unrestricted, to ensure security and trust, networks require complex consensus algorithms. The two most prominent consensus algorithms used in public blockchain networks are *Proof of Work* (PoW) and *Proof of Stake* (PoS). The PoW

consensus algorithm introduces a mathematical puzzle with variable complexity that the users can solve. The puzzle solution is achieved by computing cryptographic hashes that satisfy the algorithm-dependent criteria. To calculate the result of the PoW consensus algorithm, a large amount of computation power is allocated. Such computations waste resources that could be used for other tasks and energy [109]. The Proof of Stake consensus algorithm omits the computation part of the consensus algorithm and introduces an algorithm where network participants lock a part of their owned crypto currency to verify their trust. The algorithm is defined to verify participants based on the amounts of locked currency, while trusting participants with higher amount of currency more. Other introduced consensus algorithms generally try to alleviate the drawbacks of the PoW and PoS algorithms [130].

Similarly, the private blockchain networks uses a distributed ledger and peer-to peer architecture, but restricts access to the network's participants, as generally one or more parties decide on the access rights for other members [82]. The strict access rules increase the trust of the network members participating in the blockchain, thus reducing the need for complex consensus algorithms. This increases the transaction validation speed, resulting in a network with higher performance. The most prominent private blockchain networks are *Hyperledger Fabric* [127], *Corda*, and *Ripple*. The consensus algorithms in the private blockchain perform more efficiently with less computational power wasted [131]. Although private blockchains are more efficient, the motivation to develop such a blockchain network could be hindered by the requirement to trust the collaborating parties.

The consortium blockchain [132] stands as a middle ground for public and private blockchain networks with the ability to control access to the blockchain network which operates as a public blockchain once the access has been granted. Both private and consortium blockchains are highly specialised, usually dedicated to a single application area with a limited number of participants. Private blockchains require a community of dedicated parties to set up and operate.

The private blockchain technology is most appropriate for applications requiring efficient transaction verification and having partial trust between the collaborating parties. In the context of this work, the partial trust between the collaborating parties can be defined as the trust required to participate in the collaboration and the decision to commit model and validation data files to the network, but not high enough to share training data outright. Such trust is defined as 'partial' only as the collaborating parties should not trust each other to share the model training data directly, but be willing to collaborate to gain better quality machine learning models. Public blockchains are appropriated in applications when the parties do not trust each other, and all participants are required to have equal rights.

1.3.1.1 Consensus algorithms

With the introduction of the Bitcoin crypto currency, it required a means to evaluate the participants' willingness to commit to the network, as well as to stop the double spending problem when logging transactions. To solve such problems, the Bitcoin introduced the Proof-of-Work (PoW) consensus algorithm. The purpose of consensus algorithms is to prevent alternative sources of truth in the blockchain. The

PoW consensus algorithms also provide proof that you are committed to the blockchain network and that you, as a network participant, can be trusted to validate the transactions if you complete the task. In the blockchain network, the transactions are added to each new block, and each such new addition must be validated via the consensus algorithm. To complete the task of transaction validation, network members must calculate a number which, added to the block, produces the correct one-way hash, where the hash is calculating a one-way hashing function [124], [125]. The validation of such a transaction in the Bitcoin blockchain network is called mining. The mining computation does not add any other value to the network except for ensuring that the network participants are willing to expend the required computational resources to prove their commitment to the network. The PoW consensus algorithm approach has been criticised [131], [133] for being slow, inefficient, and wasting computational power that could be allocated elsewhere. Even though this consensus algorithm is criticised, it still powers many popular blockchain networks, such as Bitcoin and other variants of cryptocurrency.

Due to a higher level of trust, private blockchains do not need to have computationally demanding consensus algorithms. The most popular consensus algorithm among the private blockchains is the *Byzantine Fault Tolerant* consensus algorithm (BFT) or similar approaches [134]. The BFT algorithm enables the network to reach consensus regardless of whether some of the nodes participating in the process fail or act maliciously. Multiple research approaches [70], [118] have been proposed to integrate the machine learning model validation process as a consensus algorithm approach, but most of them provide only the formal definition of such an approach, or the performance of such an approach is only tested without implementing it into a full network.

1.3.1.2 Smart contracts

As the popularity of the blockchain technology increased, its application expanded to a number of domains. With applications to more diverse domains, it became clear that only performing transactions to transfer funds from one network party to another is not sufficient. To support more complex solutions, smart contracts were introduced to the blockchain. Smart contracts enable the development of more complex programs which could power a wider variety of the cases of use. The smart contracts were introduced in the Ethereum [126] blockchain that included executable command operations into blockchain blocks which could be used for a wider range of applications. Smart contracts are deployed in the blockchain and are invoked by performing a transaction in a blockchain block (**Figure 4**). The smart contract code is executed in a blockchain virtual machine. As any data stored in a blockchain are immutable, the deployed smart contract cannot be removed or modified, either. Smart contracts in the Ethereum blockchain network were first introduced with proprietary programming languages, specifically, *Solidity* [135], and *Vyper* [136]. The deployment of a smart contract and the execution of its logic on most public blockchains have an associated price commonly measured in the cryptocurrency. This price is commonly based on the complexity of the smart contract and the computational resources used for the execution [137]. This not only encourages smart

contract developers to optimise their code [138] for efficient execution, but also discourages extremely complex applications. Smart contracts also have built-in limitations, as the contract is executed by multiple network peers which have to verify the execution results, the results produced by the smart contract should be deterministic [139] for successful execution. For example, any function that uses random number generation could produce nondeterministic results [139]. A smart contract in private blockchain environments does not have a cost attached to either execution, or to deployment procedures. Thus, the development of more complex solutions via smart contracts is enabled.

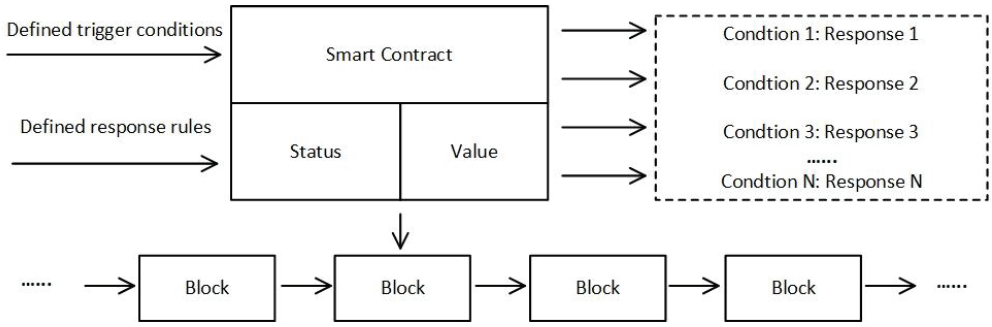


Figure 4. Smart contract deployment on public blockchain [140]

The solutions utilising smart contracts to power their business processes require to store and access business-related data. Such data could range from simple key-value pair data to large datasets with multiple features. This means that the smart contract developer evaluates how the data could be accessed by the smart contract and possible data storage solutions.

1.3.1.3 Data storage in blockchain and smart contract extensibility

The supported data storage solutions can vary depending on the type of the blockchain network selected. Public blockchains, due to the trust requirement, support less diverse data structures and data storage approaches [141], when private and hybrid blockchain types provide support for more complex data structures, and may even support existing data storage solutions [141]. Public blockchain smart contract development requires code optimisation in order to reduce the execution costs [137]. In public blockchain systems, data storage is commonly reliant on third party storage solutions, such as the *InterPlanetary File System* (IPFS) [142], [143]. Solutions, such as *Hyperledger Fabric* [127], can rely on system-provided storage that can store data in key-value pairs, or by using *CouchDB*. When adapting the blockchain storage to facilitate machine learning procedures, CouchDB provides major advantages as CouchDB enables storing data by using JSON files. This expands the possibility of storing more complex data structures in blockchain networks. Hybrid approaches, such as *Corda* [144], can utilise the currently existing database technologies to store information.

1.3.1.4 Permissioned blockchain solutions

Ripple [145] consortium blockchain was introduced as a solution for financial institutions to manage payment information. The Ripple blockchain introduced a faster consensus algorithm called the *Ripple Protocol Consensus Algorithm* (RPCA) which enables fast verifications of transactions. The Ripple blockchain was developed to include native crypto currency called *XRP*. The Ripple consortium blockchain supports custom logic via the smart contracts that can be developed by using the Python and JS programming languages. The Ripple blockchain supports blockchain oracles whose focus is dedicated to the access to centralised API that provide data to the blockchain network. The main limitation of the Ripple blockchain is its limited solution development when using smart contracts.

Corda [144] is a consortium-distributed ledger platform which uses a global state to share data between the network participants. The Corda DLT uses existing data storage technologies, such as *PostgresSQL*, *Azure SQL*, *Oracles* and others. The Corda blockchain is powered by the Java virtual machine enabling the development of dedicated Corda applications. Corda is a permissioned blockchain which enables organisations to set user roles and permissions. The user identity is provided by providing *X.509* certificates. The main drawback of the Corda DLT is that such systems lack replication between the participants, while only ensuring the complete data in the single node. Such a replication model is more common to the centralised systems in comparison to peer-to-peer blockchain networks. The Corda DLT supports local nodes and provides support for component deployment via containerised services. Smart contracts in Corda DLT can be developed by any language that can run in the Java virtual machine. The main drawbacks of Corda DLT are the limited security of the participant identity and the support for only limited transaction distribution among the participants, which removes the main benefits of a distributed system.

Hyperledger Fabric [127] is a private blockchain framework developed by the *Linux Foundation*. The modular architecture of Hyperledger Fabric allows organisations to modify the technology and customise it based on the applied requirements. The Hyperledger Fabric private blockchain supports smart contract development by using the *Go*, *Java*, and *JavaScript* programming languages. Two storage types are also supported by Hyperledger Fabric: simple storage which allows storing data in key-value pairs, and CouchDB file storage which allows using the JSON format to store more complex data. As with other modular parts of Hyperledger Fabric, it also supports alternative consensus algorithms: *BFT* implemented by *Apache Kafka* [146] and *PBFT* [147] implemented by *Raft*. Smart contracts developed for Hyperledger Fabric support calls to external oracle services. The support for such calls is important when introducing novel components, developing complex solutions, and widening support for already existing programming environments on the blockchain network architecture.

Table 3. Comparison of the supported smart contract languages and support for oracle services on blockchain platforms

	Hyperledger Fabric [127]	**Ethereum** [126]	**Corda** [144]	**Ripple** [145]
Smart contract languages	Go, JavaScript, Java	Solidity, Vyper, Yul	Java, Kotlin	C++
Network type	Private	Public	Consortium	Consortium
Consensus algorithms	BFT, RAFT (PBFT)	PoW, PoS	RAFT	Unique Node List
Supported storage technology	Key-value, CouchDB	Key-value	H2, Postgres, SQL Server, Oracle	Key-value
Native currency	No	Ether	No	XRP
Design architecture	Containerised, Modular	Node-based	Service-based	Node-based
Support for oracles	Supported	Supported	Supported	Supported
Limitations	Requires definitions of roles	Comparatively slow consensus algorithm, limited smart contract development languages	Node information is not shared with the whole blockchain network.	Low number of nodes in the blockchain network, limited smart contract functionality

A comparison of blockchain technologies having the smart contract support, and a wide range of the supported storage solutions is presented in Table 3. Hyperledger Fabric supports calls to external services from the smart contract environment [148]. Hyperledger Fabric provides a comparatively fast consensus algorithm that does not waste high amounts of computational power [2]. Hyperledger Fabric is extensible by new components that enable integration of existing machine learning environments into blockchain network [148]. Hyperledger Fabric smart contract development environment is flexible and allows to develop complex solutions unlike limited Ripple smart contracts. The Hyperledger Fabric private blockchain also have full transaction sharing between the nodes, unlike the Corda blockchain.

1.3.2 Blockchain oracles

Blockchain networks were initially designed as self-contained solutions. Smart contracts usually only access the transaction information, or the information stored in blockchain storage. Smart contracts are incapable of storing a large amount of data; therefore, they have to rely on services such as blockchain oracles for data bridging.

Solutions for trusted data feeds and side chains [149] can also become viable with the blockchain oracle usage. To introduce the ability to obtain external data from the blockchain network design, a design pattern called oracle was developed [148], [150]. The blockchain oracle can be defined as a trusted third party service which provides information to smart contract functions and external sources [150]. Similarly to smart contracts, the data or computation results obtained from blockchain oracle have to be deterministic, as they are commonly used in the smart contract logic. Blockchain oracles can be divided by four main criteria [151], [152]: data source, data direction, design pattern, and interaction. The largest and broadest overview of blockchain oracles is provided by source [152], and the taxonomy presented in the solutions is provided in **Figure 5**.

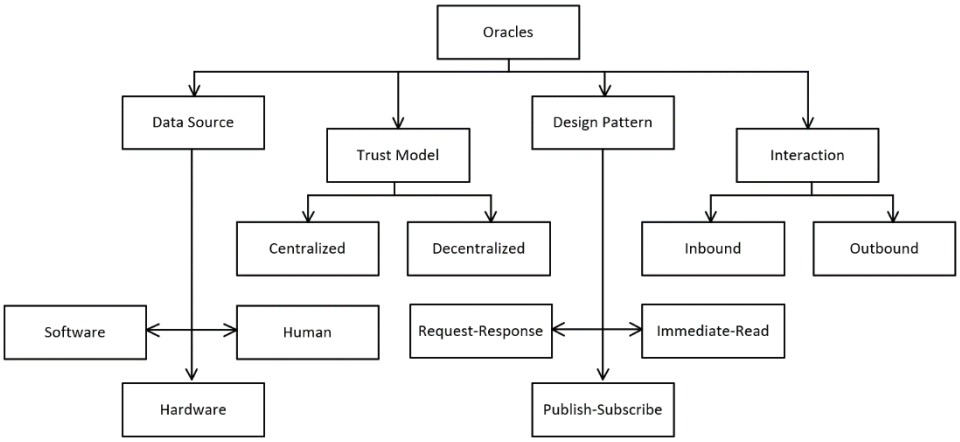


Figure 5. Taxonomy of blockchain oracles, proposed by [152]

Data sources for blockchain oracles can be software-based, where the information is provided by a third-party API service or a local network component. In a human-based solution, the information is provided manually by inputting information via software services. The data source can be produced or supplied by hardware solutions [153], [154], such as sensors or IoT devices. Such data providers would be classified as hardware oracles.

Different architecture approaches can be employed to develop blockchain oracles. The blockchain oracle can be developed as a centralised [155], [156], [157], [158] component, where a single service provides information to many users of the blockchain network. This presents multiple downsides, with the introduction of a single point of failure to a distributed system as the most crucial one. This means that the negation of access to a single centralised service could disrupt the usability of the whole blockchain network. Another drawback of centralised blockchain oracle stems from trust. Public blockchains are used for systems where users do not trust each other, even though trust is required in order to develop a trusted third-party service. Thus, introducing such a service in a public blockchain would reduce the robustness of the system and could discourage some potential network users from participating in the network. On the other side of the spectrum of centralisation, a decentralised

blockchain oracle architecture [159], [160] is proposed. Blockchain oracles can be decentralised by introducing multiple instances of third-party blockchain oracles. The use of decentralised oracles increases the reliability of the solution. Yet, trust issues persist even if the off-chain oracle is deployed on a peer-to-peer network. To increase the control of blockchain oracles, such services can be deployed as off-chain oracles on individual local network nodes. This also reduces the communication overhead as each node only communicates with a single or multiple dedicated service.

Blockchain oracles are classified according to the design patterns used. Three design patterns are the most prominent among the blockchain oracles [135]: there are the request-response, immediate read, and publish-subscribe types of oracles. The request-response design pattern defines a blockchain oracle which provides information only when requested. The publish-subscribe design pattern allows the smart contract to continuously receive information from the oracle service. This is usually reserved for information which undergoes many updates and is continuously provided. The design pattern called immediate read provides information that is small enough to keep in oracle storage and can be provided to smart contract functions on demand. Such information is usually required in a smart contract without intensive computations.

The final way to classify oracles is based on the direction of the data flow. For the data received from external data sources into the blockchain, such oracles are defined as inbound oracles. Meanwhile, those oracles which receive information from the blockchain network and share it with other services or store it in a different environment are defined as outbound. A similar proposition to classify oracle services was proposed by source [148], where oracles are divided by their means of usage. The blockchain oracles that are used only to read or provide data could be defined as data oracles. Oracles that are designed to receive data and calculate the result based on the provided data could be defined as calculation oracles.

The method combining the oracle technology with machine learning which proposes the usage of blockchain oracles is provided by source [161]. The proposed method introduces a novel architecture combining the blockchain and oracle services to perform decentralised learning on the data provided by industrial Internet-of-things devices. The proposed architecture uses multiple blockchain networks: one for data governance, and the other for data storing. The prototype network was implemented by using *XuperChain* [162].

Another method combining machine learning and blockchain oracles was proposed by source [163] which powers a novel secure voting system. The proposition utilises the Ethereum blockchain network and introduces machine learning blockchain oracles as network components which authenticate the network participants. Authentication is performed by using face recognition techniques. The research describes the architecture and does not provide any implementation details. This research also does not add oracle services into the network peer local environment and requires data to be provided from non-blockchain sources to oracle services.

Blockchain oracles have been used to perform machine learning in the blockchain technologies, but several alternative research proposals have been made on improvements to the security and trust of the approaches. Those approaches which

do not apply oracle services and combine machine learning with the blockchain technology are further discussed.

Although smart contracts enable more complex solutions on blockchain networks, they are limited by supported programming languages and can only obtain information from the blockchain network. To provide data from non-blockchain-based systems and to increase the number of the supported execution environments, smart contracts can be extended by dedicated services called blockchain oracles that provide data to the blockchain or provide an environment for computations.

1.3.3 Other blockchain technology-based machine learning solutions

Combinations of machine learning and blockchain domains have been actively researched in many research domains, especially in the *Internet-of-things* (IoT) [164], [165], healthcare, and security domains [166], especially with the combination of distributed machine learning [72], [133], and privacy-preserving machine learning, where the blockchain technology can be used to increase trust and provide means to motivate the network participants.

The blockchain technology has been proposed as a viable solution to improve the transparency, auditability [118] and security [167] of the machine learning processes. The intrusion recognition method for federated learning has been proposed by source [167]. The utilised method permissioned the blockchain to enable transparency and increased auditability of federated learning. The authors used the *MultiChain* blockchain network and trained an autoencoder with three hidden layers which, in total, contain 3000 weights. The authors of the source note that even though the performance of the model training process was reduced in the range from 5 to 15 percent, the introduction of blockchain provided beneficial transparency. For the implementation, the authors used the MultiChain blockchain, but Hyperledger Fabric and the smart contract were considered as a viable alternative.

Another approach in which the authors combined deep learning models, the blockchain technology, and federated learning was proposed by source [118]. The blockchain in this approach was used to power the incentive mechanism and compensate the network participants for the training activities of the neural network model. Additionally, a new consensus algorithm based on *Algorand* called the *blockwise-BA* protocol was proposed. The defined solution was experimentally tested by evaluating the accuracy and throughput of the trained model of the developed system. To increase the security of the network, data encryption was also employed, and, as the authors note, the introduction of the blockchain to the federated learning process introduced transparency and auditability.

Similarly, an approach to apply the blockchain technology to secure the neural network training process was proposed by source [168]. This method introduced a new blockchain block structure that would not only store blockchain operational information, but would also store cryptographic information and neural network weights. The authors describe this new structure as *DeepRing*. A novel validation and consensus approach was also defined by the authors, which relies on hashing the weights and linking them together just as blockchain blocks are linked together.

The blockchain application to increase the security of deep learning model training was proposed by source [166]. The approach employs the *Stellar* blockchain and is used to provide incentives, anonymise the identities of network participants, and log the participant activities. The introduction of multiple roles for the participants in the network such as the model contributor and the data contributor was done by utilising smart contracts.

The machine learning approach combined with Hyperledger Fabric was proposed by source [169]. A proposal to improve the Hyperledger Fabric endorsement policy by employing a machine learning approach was outlined. The implemented machine learning approach was used to detect anomalies in the validation actions of the participating nodes. The k-NN classifier was applied to detect malicious nodes, and the experiments were executed by using the *OpenMalaria* dataset. Although the authors experimented with anomalous worker detection, the research suggests that machine learning when combined with blockchain technologies is relevant and achievable.

Internet-of-things researchers applied machine learning methods to classify IoT device usage in the form of blockchain transactions [164]. Such classification could be viewed as a type of attack to identify possible IoT devices. To reduce the possibility of such an attack, the authors proposed three timestamp obfuscation approaches that utilised blockchain.

Table 4. Comparison of blockchain applications for machine learning

Reference	[168]	[169]	[164]	[166]	[167]
ML approach	Deep learning	k-NN classifier	Any classifier	Deep learning	Deep learning
Blockchain	Corda	Hyperledger Fabric	Undefined	Stella	MultiChain
Blockchain Type	Consortium	Private	Undefined	Private	Private or Public
Application area	Computer vision	Blockchain processes	IoT	Autonomous self-driving	Federated learning
Smart Contract	-	+	-	+	-
Storage	-	IBM's Cloudant	Undefined	IPFS [142]	IPFS [142]
Limitations	Limited application area, complex blockchain structure	Limited set of tested machine learning solutions	Limited application area, limited set of tested machine learning solutions	Single complex machine learning model type usage, dependency to external data storage solution	Security oriented application, dependency on external data storage solution

The majority of solutions are tailored to a specific model type, with a primary focus on enhancing the security of the system or offering incentive mechanisms through blockchain tokens.

1.3.4 Participant contribution calculation mechanisms

Systems relying on participant contribution try to encourage participation by providing monetary or other motivation to the participants called the *incentive*. This incentive should be based on the members' contributions and distribute the value gained or donated to the network fairly. Incentive can also be provided for the work completed by the network members. Incentive can be classified into two categories: *positive* and *negative*. The positive incentive motivates the network participants that provide quality contributions, whereas the negative incentive punishes misbehaving participants. Collaboration evaluation techniques were firstly studied as a part of the game theory studies such as [170], [171], [172], where the goal is to evaluate the collaborator's contribution. Such collaboration evaluation techniques are widely applied and researched in the field of economics [173], machine learning [174] and others [175]. The modified Shapley value calculation approach is used as a weight selection strategy in the proposed CDMLB method.

1.3.4.1 Shapley value

The *Shapley value* [170] was first introduced as a game theory metric to evaluate the members' contribution in a collaborative economic game according to the game theory. The Shapley value was defined as a formula, where the contribution of a coalition participant *n* is calculated (1):

$$w(\boldsymbol{v}) = \sum_{\boldsymbol{S \subseteq N \setminus \{n\}}} \frac{|\boldsymbol{S}|!\,(|\boldsymbol{N}| - |\boldsymbol{S}| - \boldsymbol{1})!}{|\boldsymbol{N}|!} \big(\boldsymbol{v}(\boldsymbol{S} \cup \{\boldsymbol{n}\}) - \boldsymbol{v}(\boldsymbol{S})\big) \tag{1}$$

where *v* is the coalition members' contribution. *N* is the total set of contributions and S is a subset out of the total contributions $S \subseteq N$ defined as a coalition. $(S \cup \{n\}) - v(S)$ represents a marginal contribution of a single coalition member.

The main benefit of the Shapley method is that it evaluates all possible permutations of a member contribution set and provides an average contribution of each member. The downside of this approach is that the computation complexity grows exponentially with each new participant. To reduce the computational complexity of this solution, many Shapley value approximation methods have been proposed [176]. The permutation sampling-based Shapley approximation method is an example of such approximation and enables the calculation of the Shapley values in linear time [176].

In the field of machine learning, Shapley values have been applied to multiple problems. The most common application is used in the dataset feature exploration [174], [177], [178]. Data importance evaluation methods where dataset sharing is considered as the coalition member contribution and the reward is defined by model performance can also employ Shapley values [179], [180]. In this case, it allows measuring the impact of the data on the performance of the model.

Similarly, the model valuation [180] in a machine learning model ensemble has been proposed, where the performance of an individual model is the contribution, and the impact to the ensemble performance is evaluated by the Shapley value.

There are many propositions to use the Shapley value for model performance evaluation or model contribution evaluation when forming a model ensemble [180]–[181]. A comparison of such propositions is presented in **Table 5**.

Table 5. Model and Data performance valuation via Shapley-based methods

Research	Measure of contribution	Approximation	Application area
Rozemberczki et al. [180]	Prediction voting	+	Model importance quantification for model ensembling
Ykhlef et al. [182]	Classifier accuracy	-	Method to select ensemble
Wang et al. [183]	Classifier accuracy	+	Data importance evaluation method
Chen et al. [181]	Generalized Shapley	-	Credit risk management with heterogeneous machine learning ensemble

The Shapley value as an incentive measure that could be used with the blockchain technology has already been proposed by [184], [185], [186]. The most complete research defining the incentive which uses Shapley as a basis for the calculations is presented in [184]. The researchers of this source presented evaluation of data shared to cloud services and defined methods how to transform this evaluation into an incentive. The participants' contribution in their denoted method was the increment of the value as the participant takes part in the coalition. The blockchain in this approach is used to control the access rights for the participants. The authors evaluated the proposed incentive measure by benchmarking its performance with varying contributor sizes and measuring the performance of the k-NN and SVM classifiers. Their experiments confirmed that the proposed incentive measure is able to provide a higher compensation to its participants, thereby proving the effectiveness of the selected approach. Even though Shapley is utilised as the classical method to evaluate contribution, many other incentive measures were proposed that were adapted for blockchain technologies.

1.3.4.2 Incentive mechanisms in the blockchain technology

The incentive in the blockchain technology was introduced with the Bitcoin cryptocurrency. This incentive encouraged the network participants to verify transactions and participate in the PoW consensus algorithm. In this incentive system, the participants are rewarded based on the number of committed blocks, which correlates to their committed computational resource input when validating a new

blockchain block. Other consensus algorithms provide similar incentives for the participation and the execution of smart contracts [130].

An example of such an incentive method is proposed by source [187] for the edge-computing-based approach. This approach allows blockchain miners to utilise edge service providers to obtain additional computational power. The authors base incentive calculations on the Stackelberg [171] game between the participants in the blockchain network and the providers of edge computing resources. The authors proved that such an incentive mechanism can be effective, and experimentally tested their proposed solution.

A similar approach for incentives for IoT and edge computing was proposed by source [188]. The authors analysed the connection between the blockchain network and the network participants and proposed an algorithm to find the Stackelberg equilibrium point. The authors experimentally tested the performance of their proposed algorithm and defined a reward pricing strategy. The research proved that Stackelberg equilibrium can be achieved by using their proposed strategy and provided the performance results of the experimentally tested methods.

Another example of an incentive developed for blockchain technology processes was presented by source [189]. This research proposed an incentive mechanism allowing to secure the verified content from the miners who are validating the blockchain blocks. The authors defined the execution cost evaluation strategy in order to address possible collusion attacks.

Another research demonstrating that incentive methods using the blockchain are viable and can be applied to data sharing was described in source [190]. The authors employed smart contracts to develop their incentive mechanism. The proposed incentive mechanism is based on evolutionary game incentives, where the participants are presented with the choice to participate in data sharing or not. The blockchain technology in this research is used to store data-sharing transaction logs and provide participation cost requirements on demand. Blockchain is also used to store the developed system logic using smart contracts. Even though many blockchain approaches employ an incentive as part of their method, in the context of this work, we are only focusing on calculating the measures of contribution. By using these contribution measures, the incentive can be defined by using personalised requirements of the participating organisations.

1.4 Summary of Analysis

The analysis of machine learning approaches revealed that the majority of the analysed approaches are developed by a single entity not employing collaboration. This reduces the diversity of the used data to develop machine learning solutions. The focus on individual development in distributed machine learning stems from security and privacy issues. Moreover, most of the distributed learning approaches found in the analysis are highly specialised, dedicated to addressing a single machine learning problem, and they typically employ only one specific type of a machine learning model. The limited machine learning model type usage restricts the possibilities for collaboration and engagement in a broader range of applications.

When analysing privacy preserving solutions that are applied to enable collaboration, the results indicated that most of the solutions concentrate their efforts on safeguarding the communication channels and the sensitive training data. As a result, most of these approaches separate the model training and deployment environments to ensure the privacy of the sensitive data. Another approach how transparency and trust could be enhanced is the introduction of the blockchain technology to the distributed machine learning pipeline. The blockchain consensus algorithm could enforce the validation of the performed computations by ensuring the correctness of the performed calculations and the provided results.

The analysis of blockchain technology applications for collaborative distributed machine learning revealed that most of the existing propositions and applications utilise the blockchain as an incentive mechanism. In such an approach, the cryptocurrency issued by the blockchain is used as an incentive measure and is provided to the participants for executing tasks or collaborating in the process. Or else, the blockchain is used as an immutable transaction log preventing the loss of information about artefacts stored in the blockchain and logging the ownership changes. These incentive solutions are frequently highly specialised, defined for a single community, or involving solutions with only limited or even no integration with smart contracts. The development of distributed machine learning applications on blockchain technology-based solutions is also hindered by the limited support of the currently existing machine learning environments and approaches.

To address this limitation, oracle services in blockchain-based distributed machine learning can be applied. Oracle services can bridge the gap between on-chain and off-chain data, thus allowing smart contracts on the blockchain to access external data sources. By using blockchain oracles, the applicable set of technology and environment for machine learning on the blockchain could be extended, thus improving the ability to develop solutions. The combination of distributed machine learning, privacy preservation methods, and blockchain technologies with blockchain oracles would provide means for solutions that improve trust and transparency and strengthen collaboration, as proven by sources [165], [166], and [191].

Based on these analysis results, it was decided to modify the thesis objectives to reflect the usage of the private blockchain technology to power the collaborative distributed machine learning. The updated research objectives are:

1. Analyse machine learning and collaboration in distributed machine learning.
2. Analyse blockchain technologies and their capabilities to be applied for distributed machine learning processes.
3. Analyse training data privacy preservation methods and implementations in machine learning.
4. Propose a method for collaboration in distributed machine learning utilising the private blockchain technology.
5. Implement the solution for collaborative distributed machine learning using blockchain technology according to the proposed method.
6. Evaluate the capabilities of the methods to perform collaborative blockchain-based distributed machine learning.

2. METHOD FOR COLLABORATIVE DISTRIBUTED MACHINE LEARNING ON BLOCKCHAIN

The Method for *Collaborative Distributed Machine Learning on Blockchain* (CDMLB) enables multiple organisations to participate in distributed machine learning and is proposed in this chapter (the method was also presented in research articles [192], [193], and [194]). The CDMLB method uses a private blockchain network and is designed for organisations which would like to collaborate in machine learning tasks but lack trust to develop a trusted third-party centralised solution. The CDMLB method utilises the already existing machine learning technologies, models, and datasets to enable collaboration on a private blockchain network. The introduction of a private blockchain network increases the transparency of the model deployment process, by allowing to access all the data and audit the activity logs of the action performed in a blockchain. Additionally, the private blockchain also enables trust in the collaboration process because the model prediction set calculation results are verified by multiple parties. The utilisation of the blockchain technology also increases the system resilience to external threats by removing a single point of failure that could arise when using a centralised third-party architecture.

The CDMLB method also provides means to ensure data privacy by defining multiple specialised environments and introducing a specialised model deployment and usage process. The proposed method encapsulates multiple environments (see **Figure 6**): a) private model development environment; b) blockchain-based model deployment environment (on-chain environment); and c) network contributor node environment (off-chain environment).

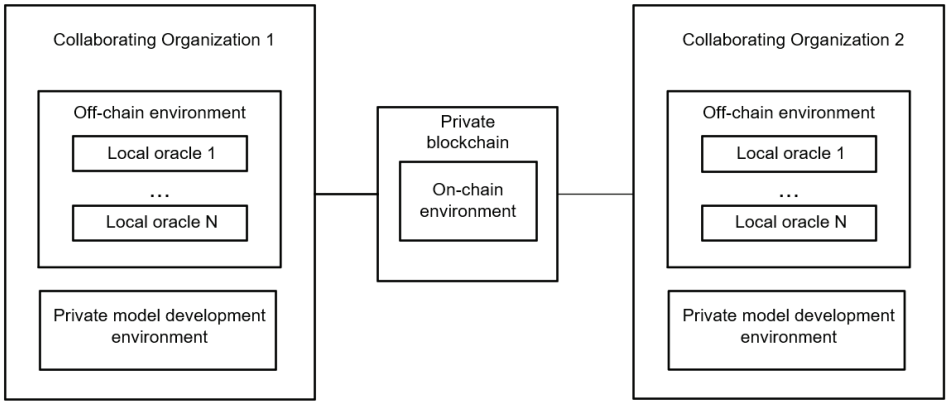


Figure 6. Environments of CDMLB private blockchain platform

The private model development environment, in the scope of the method, is solely dependent on the model developer's needs, and such an environment has no prerequisite requirements. This environment ensures that the data management and model training parts remain private and allows the user to adapt the training environment based on the individual requirements. The decision to perform model

training in a private non-blockchain environment is defined by the need to protect sensitive training data.

Model training can also require a significant amount of computational power as, based on the consensus algorithm of a selected blockchain network, it can require repeating this process for at least 51% of the network participants, thus further increasing the demand for computational resources. This would significantly increase the time to train the models. The developed smart contract must return deterministic results, and, in the model training case, such determinism is not possible without using exactly the same training data. For this reason, the proposed method model training process is not completed on the blockchain so that not to increase the number of the required computations. The CDMLB method uses blockchain in the model deployment and model usage parts of the model development process. Such a distribution of the model deployment process allows combining models from different sources, which can lead to more diverse models. Such diversity can lead to a classifier of a higher quality. The distribution of the deployment process also ensures the validity of the provided model and data as the model inference calculations are verified by multiple network participants.

The blockchain model deployment environment (henceforth referred to as *on-chain*) aims to provide a transparent, activity logging environment for model sharing. Every call to a function on the blockchain network is recorded in the distributed ledger, thus providing the ability to audit the user's actions at any given time. The data and the models uploaded to the blockchain network are distributed to all network parties and are always accessible to the network contributors. The combination of action logging and openly accessible distributed machine learning artefacts promotes the transparency for all the participating contributors. The blockchain also increases the robustness of the system by providing data replication among the network participants. The process execution of the machine learning model deployment logic is provided by the blockchain environment, which is implemented by using smart contracts.

The network contributor local node environment (henceforth referred to as *off-chain*) is introduced to support a more diverse set of machine learning programming environments. Together with the blockchain model deployment environment, it leverages a wider support of machine learning model types, as well as enables customisable model development environments. The customisation of the model development parameters allows contributors to develop more diverse models. The off-chain environment also enables the decentralisation and distribution of complex model computations using deployed oracle services that are integrated by using smart contracts.

To enable a diverse ecosystem of the currently existing machine learning programming environments using the blockchain technologies, the CDMLB method introduces a network contributor local node environment. This off-chain environment allows transferring the complex model inference computations to oracle services that are accessed by smart contract functions. Multiple instances of off-chain environments can be present in the CDMLB blockchain network architecture. This allows the network participants to select different implementation environments for

machine learning solutions, thereby giving an opportunity to tailor the environment for each task individually.

The conceptual scheme of the proposed CDMLB method is presented in **Figure 7**. In the CDMLB method, the contributor uses a dataset to develop the machine learning model. The contributor divides the collected data into training, testing, and validation datasets. The training dataset is used to develop machine learning models which use labelled data and features in the numeric format, and these models are later used for computing predictions on the testing data. In total, the CDMLB method requires the contribution of two artefacts: 1) machine learning model, and 2) validation dataset. The contributed model is used to calculate the predictions on the validation dataset. These predictions are used to evaluate the input of a contributor in the CDMLB blockchain, defined as contribution. Contributions are evaluated by measuring the performance of the model. Similarly, data contributions are evaluated by measuring the impact on the ensemble performance which contains a single best-performing model from each contributor. By using data and model contribution scores, incentives can be derived. The incentives can be used to divide monetary, or any other, value recorded by the blockchain network. The requirements for the incentive highly depend on the collaborating organisations and the described incentive calculation rules. Such rules and requirements are not part of the CDMLB method.

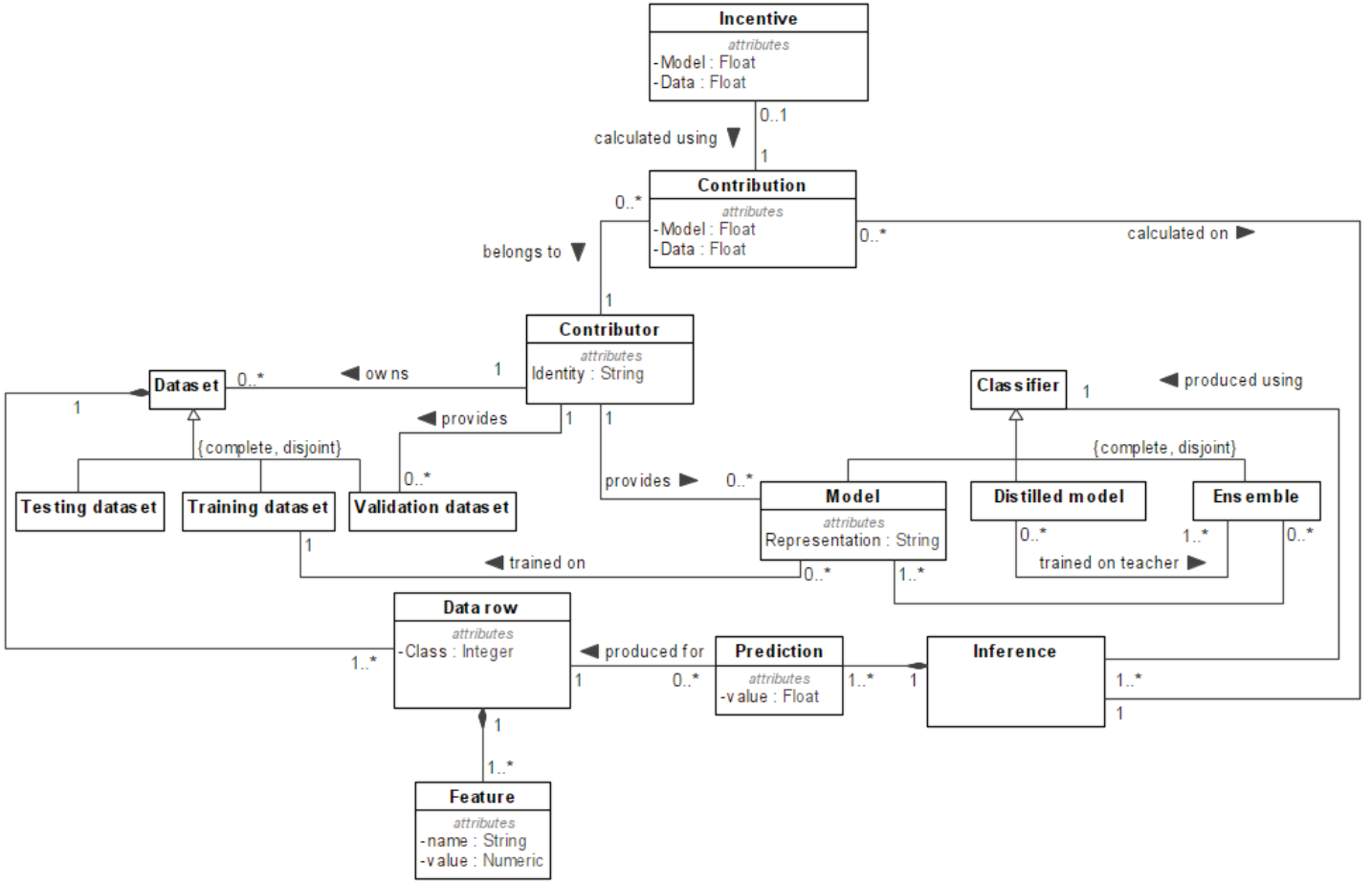


Figure 7. Concepts of CDMLB method

To utilise the CDMLB method, the collaborating organisations need to agree on the requirements for the solution implementing the method before the start of the blockchain network development.

- Select a private blockchain technology which supports API calls from the smart contract execution environment to external oracle services.
- Define the requirements for data and model structures and storage formats, which are agreed upon by all the governing organisations.
- Define the communication channels, collaboration rules, and roles for the collaborating organisations.
- Define the procedures required to implement and approve the local oracle services and smart contracts developed by the collaborating organisations.
- Define the requirement for a blockchain oracle factory service which provides smart contracts and local oracle services.
- Define the procedures for managing the developed services and the security provision of said services which should be ensured by the collaborating organisations.

Component initialisation and the required preparation processes which are necessary to be completed before the CDMLB activities could be performed are detailed in Figure 8. The collaborating parties should select the private blockchain technology with the consideration of the smart contract development environments and the consensus algorithm usability. After setting the requirements for the blockchain technology to be used, the organisations should define models and datasets that the collaboration would be based upon. This is required to start working on the smart contracts and the services which depend on the model and the data format as well as the dataset structure. Additional support to the model types and datasets can be defined after the deployment of the network as well. The private blockchain requires access management and the definition of the network roles; such role definitions should be completed before the network development starts. The two final steps of the requirements for preparation involve requiring information about the smart contract and oracle development rules. The functions provided by the blockchain oracle service in the CDMLB method are called from the smart contract, hence, the integration between two such components should be specified.

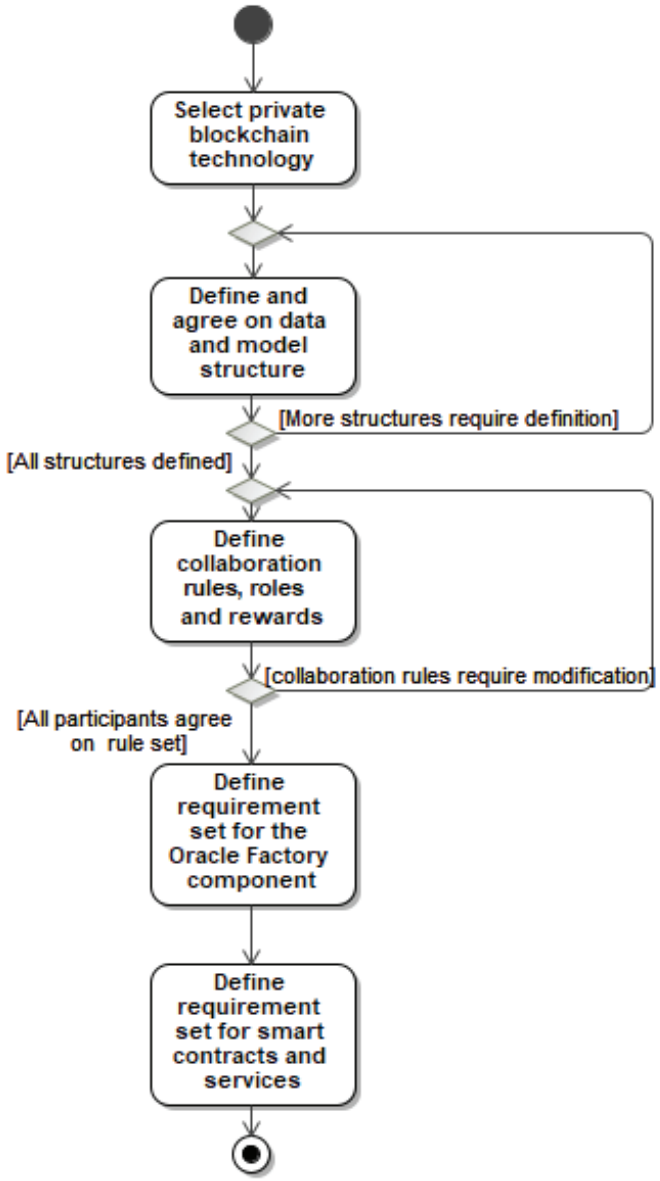


Figure 8. The required initialization procedures for the CDMLB method

The defined requirements of the smart contract and blockchain oracles are then used in their development. Both smart contracts and blockchain oracles are developed, tested, and deployed to the oracle factory component. The blockchain oracle factory component then distributes the required smart contracts and the blockchain oracle on demand to new or already existing network participants. The oracle factory component should be developed to track versions of the deployed smart contracts and oracle services, as any version mismatch could prevent the network members from

reaching the consensus. The oracle factory service should be distributed, and each participating organisation should contain such a service.

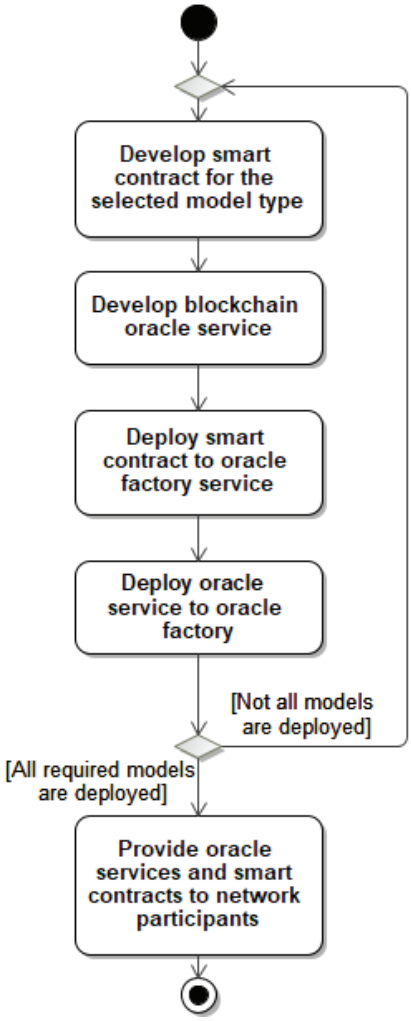


Figure 9. Development process of the proposed CDMLB method services and smart contracts

Figure 10 lists the network knowledge usage stages which are performed after all the prerequisite operations have been completed. The proposed CDMLB method consists of the CDMLB platform preparation, model and data deployment, and the network knowledge usage stages.

The method defines the procedures and requirements for the ways how the machine learning model is to be developed, deployed on the blockchain network, and then used. The method is presented from the perspective of a single network contributor. Multiple contributors must exist to successfully collaborate while using the CDMLB private blockchain platform. The proposed method can be followed to

develop multiple CDMLB solutions that are hosted on the same blockchain network. This allows contributors to submit multiple datasets or machine learning model files, as well as use the knowledge stored on the blockchain. The method keeps track of the currently best-performing models and re-evaluates all the submitted models when presented with new validation data. For every network model and data change, the contribution measures are calculated and stored in the blockchain storage. This allows monitoring the evolution of the contributor input. Only the most recent contribution measure is used when evaluating the contributors' shared model performance. The outdated contribution measures are stored in the blockchain and can be used to trace the participants' contribution history.

The CDMLB method enables usage of multiple machine learning model types simultaneously. Each new machine learning model type requires the development of a dedicated smart contract and the development of a dedicated oracle service which are distributed to all network participants. The CDMLB method enables the usage of such model types in a homogeneous ensemble or by combining different model types into heterogeneous ensembles. A single private blockchain network can contain multiple organisations collaborating by using multiple smart contracts.

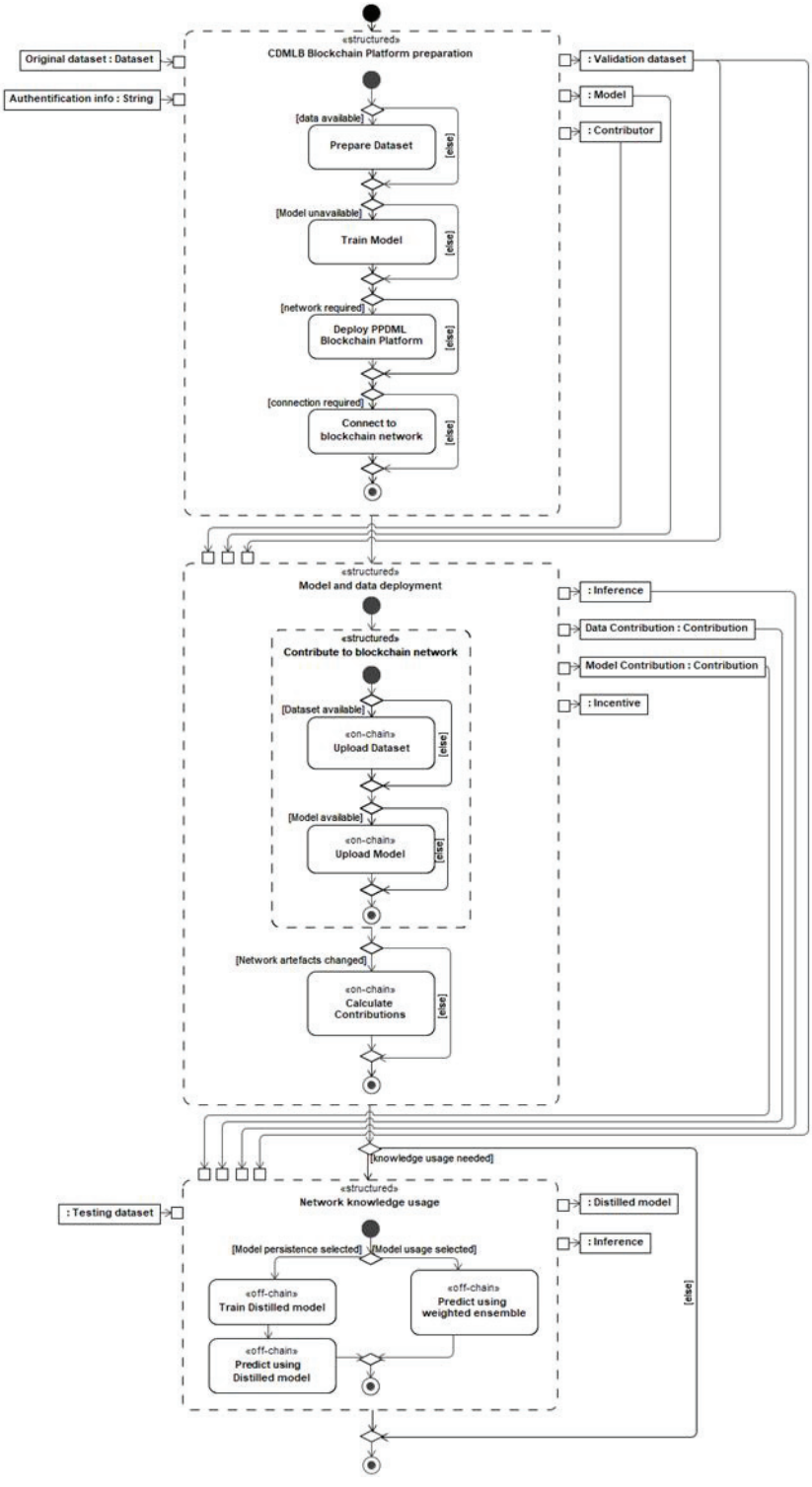


Figure 10. Process of the CDMLB method

The CDMLB blockchain platform preparation stage encapsulates processes for the development of machine learning artefacts which will be later deployed into the blockchain network. This stage also includes the processes required to deploy and connect to the CDMLB blockchain platform. The network artefact preparation starts with data preparation and is performed in a private model development environment (see **Figure 6**). The network contributor based on the agreed-upon data requirements of the collaborating organisations prepares a dataset. During the data preparation step, a training dataset is prepared, as well as validation subset that will be uploaded to the blockchain network and used for model validation. If the network contributor cannot obtain the dataset, the data preparation process can be omitted from the method. Similarly, if the contributor is not able to share the validation dataset due to any privacy or security limitations, the dataset split into separate subsets can be omitted. The prepared training dataset is used to train a classifier model. Since the method does not restrict the model training process, different model development environments can be used with various hardware and software configurations and different model development parameters. Although the model training process does not depend on the method, the resulting model representation must adhere to the model and format the requirements defined by the collaborating organisations. Once the machine learning artefact development is complete, the network contributor connects to the blockchain network; this may require the deployment of the CDMLB platform. To complete the model deployment step, the contributor must also provide a specification of the dataset required for the validation data and obtain the required permissions from the collaborating organisations. The platform preparation part of the method is concluded with the connection to the private blockchain network.

The model and data deployment stage encapsulates the processes required to share machine learning artefacts, namely, the model and the data, with the blockchain network and the procedures performed to evaluate the quality of these shared artefacts. All the actions performed in this step are implemented by using smart contracts and are called the *collaborating application*. The application users are authorised by using private blockchain identities. Once authenticated, model and data artefacts may be contributed to the network by uploading a dataset or a model to the network via the distributed application. The dataset format is transformed to save space on the blockchain storage, and, if any machine learning model exists on the blockchain storage, the uploaded dataset is used to calculate the model predictions. The model predictions are then stored in the blockchain storage. Similarly, the model upload procedure starts by uploading the machine learning model stored into a model file on the blockchain network via the distributed application. The model file information is then transformed into text-based information and stored in the blockchain storage. If the upload procedure for the model or the dataset is successful, the uploaded artefacts are replicated among the network nodes. To keep track of the input of the contributors into the overall network, the performance of each uploaded contribution is evaluated. Such contribution scores are used to derive the incentive. These incentive measures are used to define the amount of the value the contributor should be provided for their input.

The final stage of the CDMLB method is dedicated to using the model knowledge stored on the blockchain. Two different approaches are outlined on how the models can be utilised. The first approach aggregates models into a single model ensemble, and, by applying weighted averaging, produces prediction on the testing data provided by the contributor. The ensemble is produced by a specialised blockchain oracle service, and the testing data used for calculating predictions never leaves the blockchain contributor's local environment. Additionally, the produced model prediction file is stored in the contributor's local environment. Such an approach allows ensuring the privacy of the validation dataset and predictions, while still recording the transactions of the machine learning model utilisation. The second approach utilises the knowledge distillation approach to train an aggregating model on the model ensemble knowledge. The approach uses combined validation data and the model ensemble to train a new neural network model. The produced model prediction set is used as an input to the student neural network model, and its performance is validated by using combined validation data. The distilled model file is stored in the contributor's local environment, and it can be further developed individually by using the validation data, or merely used 'as is' in order to make predictions without further training. The knowledge distillation approach provides privacy preservation for machine learning models stored on the blockchain.

The introduction of the blockchain technology to the collaborative distributed learning process allows an organisation to analyse blockchain transactions and blockchain artefacts at any given time, thus enhancing transparency. The CDMLB method differs from the other currently existing solutions by enabling support to different supervised learning model types and machine learning tasks instead of developing a blockchain network for each specialised task or model type. The CDMLB method requires only a small part of the data for validation, thus removing the need to share any sensitive training data to the blockchain network. The CDMLB method also supports two network knowledge usage solutions, thus enabling privacy-preserving usage of the model. Ultimately, the CDMLB method enables the contributing organisations to evaluate contributions to the selected machine learning task.

The proposed network distillation method is designed as a countermeasure to a membership inference attack against the network. Even though most of the overviewed research applied differential privacy to defend from membership inference attacks, it is not viable in the CDMLB approach as such an approach does not send the training data to the blockchain network. The only data that are provided to the network are the anonymised validation dataset. The organisations managing the blockchain network should enable access to it only to the partially trusted parties because adversaries with an access to the blockchain ledger could perform model and feature extraction and model identification attacks. Table 6 lists threats which could be attack vectors in the course of the CDMLB process. Possible countermeasures to these attacks are described with the counter-measure provider, whether it be aspects of the blockchain or specific network knowledge usage scenarios. Most of the provided security measures are provided by the private blockchain access management as well as by the ensemble weight selection strategy.

Table 6. Countermeasures to machine learning privacy and security threats provided by the CDMLB method

No.	Threat	Possible counter measure	Countermeasure provider
1.	Model extraction	The private blockchain provides access to the model files only to the authorised participants	Private blockchain
2.	Feature extraction	Only authorised participants can access the data features in the provided data sample or the uploaded validation dataset. Even though the features are disclosed to the blockchain network participants, an attack from external sources is still a threat	Private blockchain
3.	Membership inference	Anonymisation of the provided validation dataset is recommended	Recommendation of the CDBML method
		To prevent extraction of the original training data, the CDBML method supports knowledge distillation	Knowledge distillation
4.	Training set poisoning	Training set poisoning is a viable threat to the collaboration process, but the network contributors can be blocked for repeated violations of the defined collaboration rules. The designed posShap weight selection method would also remove any non-performing machine learning models	Ensemble weight selection strategy, private blockchain
5.	Validation set poisoning	The contributions of multiple collaborators might reveal that one or more participants submit validation data which perform better on their own models. The collaborating organisations should moderate and block such adversaries from participating in the network	Private blockchain
6.	Model fitting to available data	Similarly to the validation set poisoning, model fitting to the validation data can be hard to detect and should be monitored by the collaborating organisations. Nevertheless, any detected malicious participants could be blocked from accessing the network	Private blockchain

In the next sections, the CDMLB method is presented in more detail. Each method stage is defined with an activity diagram denoting the required actions, their inputs, and the resulting outputs.

2.1 Blockchain platform preparation

The model and network preparation stage consists of four steps (see **Figure 11**): data preparation, model preparation, deployment of the CDMLB blockchain platform, and connection to the blockchain network. The data preparation and model training steps are conducted in a private model development environment, while the deployment and connection to the blockchain steps provide the network contributor with the blockchain network identity and the required services for the subsequent method stages. The data preparation or the model training steps can be omitted if the network contributor already has access to the model, or does not have any datasets that could be shared with other contributors. The blockchain platform deployment step ensures that all the required blockchain components and services are deployed and are ready to be used for collaboration purposes. Moreover, during the connection to the blockchain network, an instance of the blockchain ledger in the network contributor's local environment is created, which also deploys the required smart contracts for collaboration purposes. The smart contracts implement all the required model and upload procedures as well as calls to the required oracle services. Due to the fact that the smart contract contains model and data validation and contains calls to oracle services, new contracts should be developed when new model types or new oracle services have been added to the CDMLB platform.

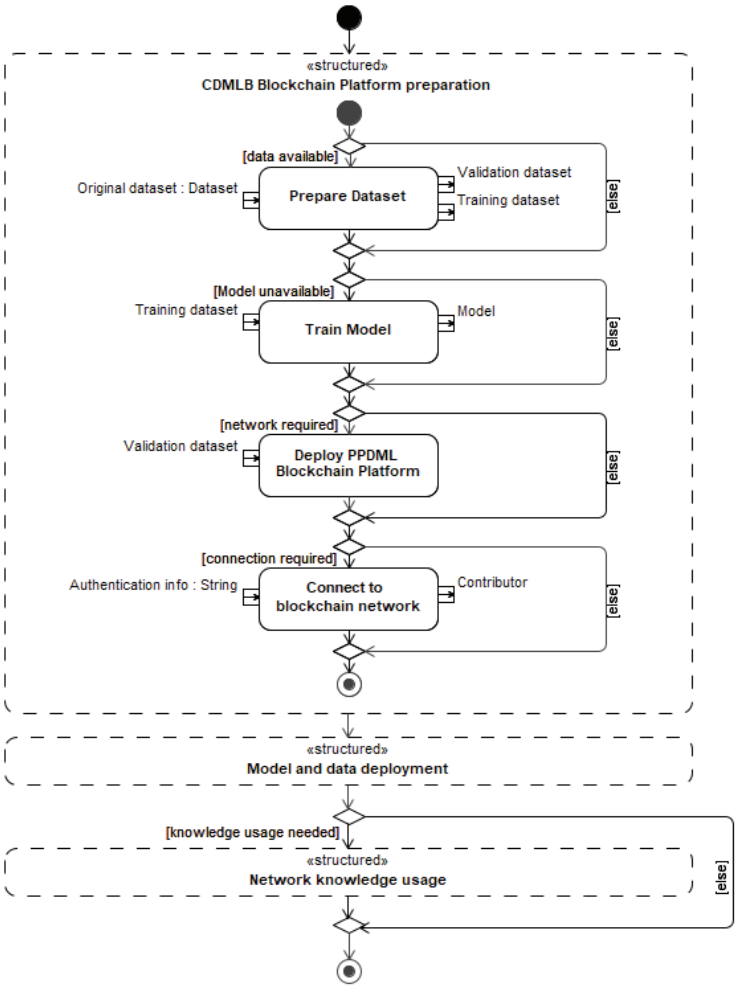


Figure 11. CDMLB blockchain platform preparation part of the proposed method

The CDMLB Blockchain platform preparation steps are further elaborated upon in the upcoming sections.

2.1.1 Dataset preparation

As in any machine learning process, the data preparation stage consists of data standardisation, data cleaning, feature selection, and data partitioning into the training, validation, and testing data subsets. The validation data subset should represent the original dataset in both class and value distribution. If the validation dataset is sensitive, the appropriate anonymisation and privacy preservation steps should be taken before the data file is deployed to the blockchain network. Currently, the CDMLB method supports tabular data only; thus, any categorical values should be transformed into numeric values by using the available encoding techniques. Additionally, it is essential that the dataset conforms to the format and structure

requirements outlined by the collaborating organisations. This includes the definitions of all data features, as well as the class label column.

The proposed CDMLB method can support multiple data structures dedicated for different machine learning problems at the same time. Smart contracts that implement data structure validation, upload and reading functions must be developed for every existing structure. Every model inference smart contract and local oracle service must be developed or updated to support new data structures. The proposed CDMLB method requires to store data in a text format, namely, in the JSON structure; thus, any data that are provided in the non-textual format should be encoded or transformed into the text format to enable the possibility to use the data in the blockchain network.

2.1.2 Model training

After the dataset has been prepared, the model training takes place, which is performed in a private model development environment. The method utilises the supervised learning approach, during which, a dataset with labelled data is used. During model training, we derive function f from the training data which labels X values of the data with the corresponding prediction. The relationship between Y and X could be defined as

$$Y = f(X) \tag{2}$$

where Y is the class label and $Y \in \{0; 1\}$, X is the d-dimensional vector $X \in \mathcal{R}^d$. The classification function f is found such that its application minimises the loss metric which is specific to the chosen classifier:

$$L(f(x; \theta), y) \tag{3}$$

where θ is a set of learnable parameters, x is a single instance of the training data, and y is its respective class label. The trained parameter set θ^* (the trained model) is obtained by minimising the loss function for a set of data instances N by using Formula (4):

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i) \tag{4}$$

The training dataset is used to train the classification function f, and the test data are used to benchmark the performance of the developed function. When provided with the novel descriptors, the function is used to predict a target value.

Machine learning methods supported by the proposed CDMLB method must be suited for the supervised learning machine learning tasks. The machine learning method needs to be able to serialise the model into a file. The serialised file formats must be compatible with the developed oracle services. The predictions of the used model should be represented as a Bayesian probability of the class membership.

2.1.3 Deploying CDMLB platform and connecting to blockchain

The connection to the blockchain network starts with obtaining the permission to join an existing private blockchain network, provided an appropriate private blockchain exists. In case no such private blockchain exists, a new private blockchain network should also be developed and deployed. The developed private blockchain network supports the execution of multiple smart contracts and the ability to deploy additional smart contracts after the network setup stage is complete. This provides the ability to develop the blockchain network for a wide range of machine learning problems without the need to create multiple instances of the blockchain network. For each smart contract used by the network, an integrated local off-chain oracle service should exist on the same network.

The CDMLB network, presented in Figure 12, is composed of collaborating organisations which issue permissions and identities to the contributors and offer access to the blockchain oracle factory component which deploys smart contracts with the required local oracle service components. The factory component providing oracles and smart contracts is required to unify the used version of the components between the contributors. Any discrepancies of the smart contract or oracle versions might prevent a contributor from participating in the network. The provided smart contracts are used to calculate the model predictions and store the model and data files. These functions power the collaboration process. Oracle services are required for the smart contract to power the model prediction calculation on different programming environments from the one provided by the smart contracts. The contributor provides multiple submissions to the collaborating application and, in turn, to the blockchain network (Figure 13). The contributor can submit datasets to the network or submit machine learning models. A single contributor can be assigned additional roles, but the minimum requirements are the submission of data and model artefacts, and the ability to use the produced ensembles or distilled models. Each individual collaborating organisation also operates an internal identity providing service.

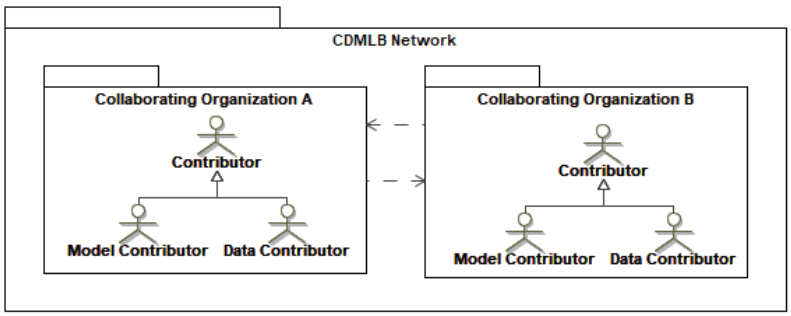


Figure 12. User roles in CDMLB network

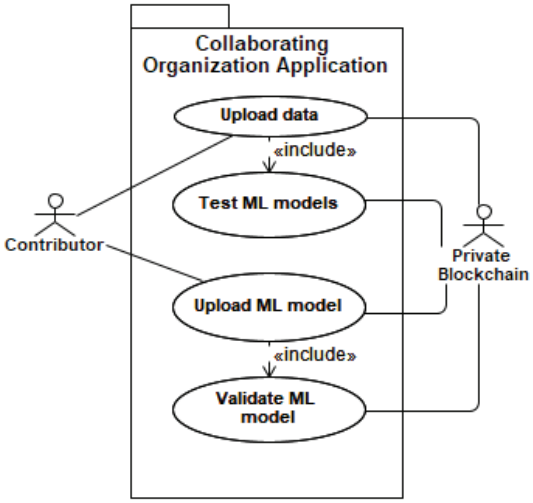


Figure 13. Minimal required functions for distributed application

To set up the CDMLB network, at least two collaborating organisations (**Figure 12**) need be present in the network, but the number of collaborating organisations can be extended to any size. The initial step of setting up a private blockchain consists of establishing organisation-based roles, communication channels like ones present in Hyperledger Fabric, developing the initial private blockchain structure, and deploying the network. Additionally, the blockchain network is initialised, and the collaborating organisations must agree on the machine learning tasks that will be supported and define the supported data and model structures. Based on data structure definitions, the collaborating organisation or multiple organisations develop smart contracts and the local oracle services.

Creating local oracles that are trusted by the blockchain network and possess a distributed architecture capable of receiving data from the blockchain network requires a custom approach. The blockchain network itself produces the data for the oracle, and the oracle functions only transform the provided data in a different programming environment which is not supported by the blockchain network.

The oracle service only applies computations and presents the results back to the smart contract. The oracle follows the request-response principle; the data and their calculations are provided after a request has been made from the smart contract. The oracle service is deployed to an individual network peer node.

Each smart contract and the respective local oracle is deployed by organisation's oracle factory service (**Figure 15**). Network contributors use this service to obtain the required smart contracts and the local oracle services. Oracle services in the proposed solution are separated from the blockchain network environment and divided into two types: data providers and computation oracles. Oracle services are replicated in each node and are deployed in the local node environment. This is the key difference from the computation oracles proposed in source [148]. In the CDMLB method, blockchain oracles are not divided into data and computation oracles; they only provide computations, whereas the validation data are obtained from the blockchain network.

Oracle services are distributed by deploying them to the local network participant node environments.

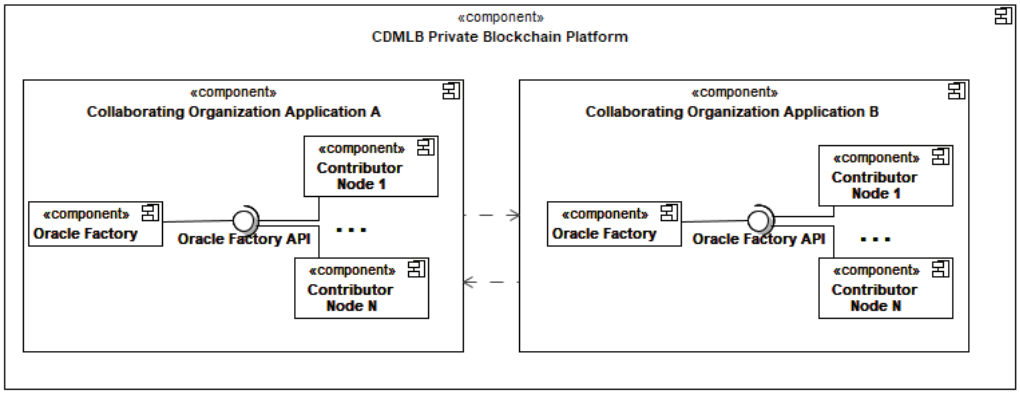


Figure 14. General component scheme of CDMLB network components

Source [150] introduces a smart contract factory pattern. Such a pattern is designed to deploy instances of smart contracts. The CDMLB method proposes a modification of the smart contract factory design pattern [150] with its component diagram presented in Figure 14. Instead of using the design pattern to deploy multiple instances of the same smart contract, this pattern is used to deploy local off-chain oracles and the required smart contracts. To keep track of the versions of the deployed smart contracts, a contract registry [150] design pattern with a smart contract factory is employed. This design pattern allows organisations to track the issued smart contracts and oracle services and allows organisations to manage the local oracle service development and maintenance. The oracle and smart contract deployment instances are tracked in the *RegistryEntry* entity, with the software version and data information. The registry entry also tracks which contributor submitted the code, by logging their identity. The oracle factory API provides functions dedicated to listing smart contracts and oracle services as well as the initialisation and update management.

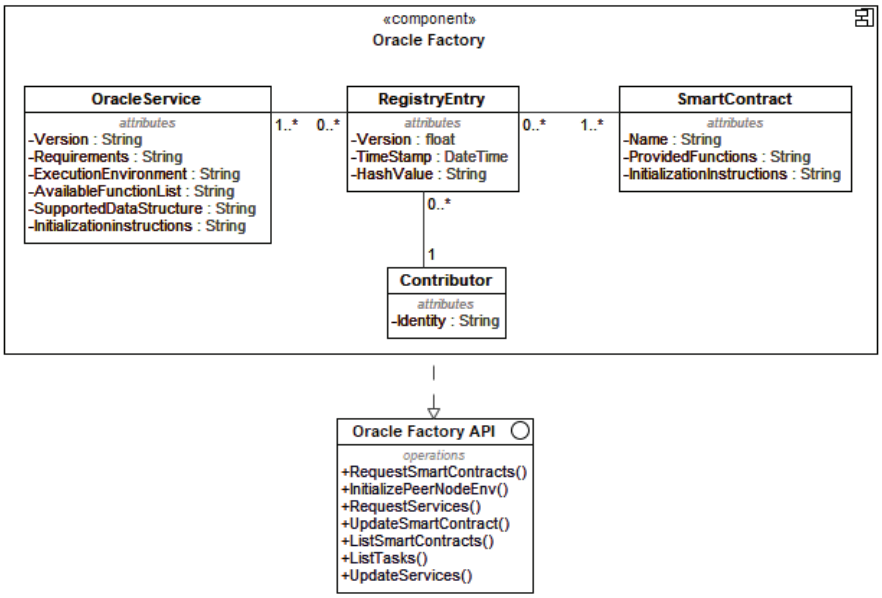


Figure 15. Concepts and operations of the oracle factory network component

After the blockchain initialisation, the user performs the smart contract deployment. The contributor node requests smart contracts and local oracle services from the oracle factory component (**Figure 15**) which downloads and compiles the requested smart contract from the network as well as the local oracle services. In the proposed method, each organisation should contain an oracle factory component and provide the oracle services to the organisation members. For every collaborating organisation, the versions of components that are being distributed must match and be synchronised. Every organisation can collaborate with multiple other organisations or participate in two blockchain networks. The organisations can collaborate with multiple other organisations simultaneously.

The provided oracle services allow the collaborating organisations and contributors to use the existing machine learning technologies such as *Python*, *R*, or other popular machine learning environments to develop their models. This reduces the development time for machine learning solutions and allows the existing models to be reused. After the initialisation, smart contracts are deployed in a local network peer environment. At the final step, the contributor tests the availability of the locally deployed oracle service operations. The setup and testing phase of the blockchain oracle deployment step is mandatory for all users joining or reconnecting to the blockchain, as any discrepancies in terms of the blockchain oracle implementation may result in a failed consensus algorithm.

The network connection stage is complete when the contributor has successfully obtained a network identity and established the required services and smart contracts (**Figure 16**). To successfully run the CDMLB method, a minimum of 4 services are required. The model inference service is dedicated for the evaluation of models and datasets after they have been contributed to the blockchain network. This model inference service also calculates the Shapley values and stores them in the blockchain

network storage. Shapley values are only calculated when new data or model artefacts are provided to the blockchain network and used from the blockchain storage if required. The blockchain knowledge usage requires two services. The ensemble usage service uses network knowledge in the form of an ensemble to calculate the model predictions on novel data and stores the results in the contributors' local environment. The alternative to the ensemble usage service is the knowledge distillation service which uses the same inputs, but provides the network contributor with a new neural network model for usage and further development.

By using the proposed method, the contributor may be part of multiple collaborative applications on the blockchain network and would only need to obtain the identity once. If the data structures used by the network differ in a number of features, or if they have multiple features represented in different formats, the network would need to join additional collaboration solutions, or an entirely new blockchain network may be developed. The provided identities must be submitted to the local distributed web application to successfully connect to the blockchain network and execute smart contracts.

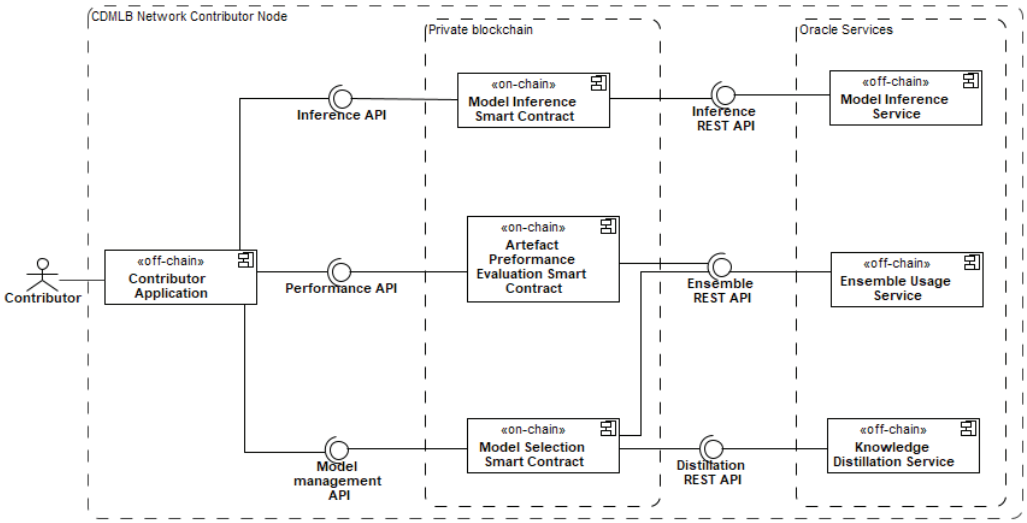


Figure 16. Components of the contributor node environment required for the CDMLB method

2.2 Model and Data Deployment

The model and data deployment to the blockchain network is performed by using the developed collaborating organisation application. The application identifies the contributor and enables the contributor to perform data or model file upload operations.

The user roles and a list of the supported functions for such an application are presented in **Figure 13**. The CDMLB method enables the data and model sharing process which is detailed in **Figure 17**. Data owners are able to share data in order to participate in evaluating the overall quality of the shared models, while the model owners are provided with means to use a larger dataset for determining the quality of

the model and with access to a distilled model developed by using the best-performing set of models. Contributors are also provided with means to use the network knowledge by predicting via the weighted ensemble.

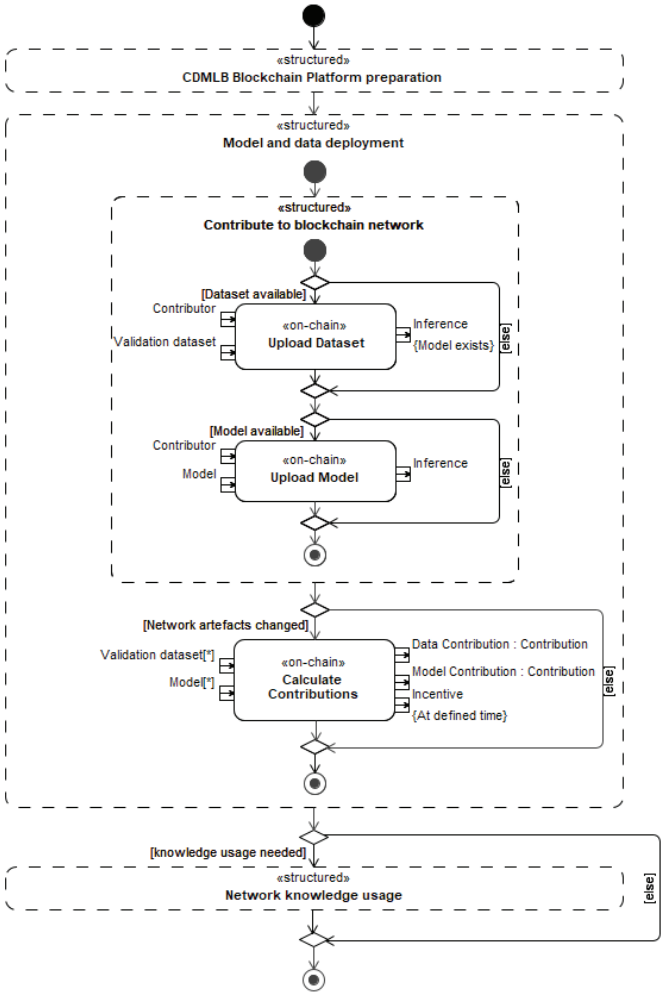


Figure 17. Model and data deployment part of the CDMLB method

2.2.1 Contribution to blockchain network

Two different contributions to the network are possible – uploading a dataset, or uploading a model. The data upload step starts with the contributor uploading a dataset file to the local web application. The local web application checks if the file is of a supported format, and checks if the uploaded dataset structure adheres to the defined structure. The blockchain oracle service then performs the testing operations with any valid model provided by the contributors. If multiple models are presented from a single contributor, the model with the highest performance will be used in testing.

Similarly, the uploaded machine learning model file is tested by the local web application, and then validated on a dataset sample to check the compatibility between the uploaded model and a data sample. Such a test ensures that the model file is compatible with the used oracle service and that it is presented in the correct file format. Both uploaded artefacts are stored on the blockchain and replicated among all the participants. The artefacts are transformed to reduce the amount of the required storage space; specifically, the dataset file is transformed by splitting the data feature and label columns and storing them into a data matrix as well as discarding any meta data contained in the data files. The ML model file is encoded into the text format, which can later be decoded by the local off-chain oracles.

To ensure the data correctness and usability, every uploaded file is validated via the dedicated function in the local oracle solution. The validation procedure also discards any mismatched classifier types or file formats. The model contribution evaluation function calculates the model's performance, and any poorly developed models are not included into the model ensemble. The data validation procedure must be designed to discard any data that include empty values and could be developed to check other data quality metrics. Similar to the model evaluation, the proposed method also evaluates the data impact on the model's performance, and any dataset not meeting the imposed requirements could be rejected by the validation. The collaborating organisations should also agree and develop functions which would define logical limitations to data features which exist in datasets. Such an approach would reduce the amount of erroneous and highly biased data form being used in the proposed method. Based on the common trust and access right management, any collaborating parties that would repeatedly provide erroneous or biased data could also be removed from the collaboration process.

Data and model contributions are stored in the respective model and data structures and are stored in the blockchain storage. For every combination of the validation data and an ML model, a dedicated off-chain blockchain oracle service calculates the model prediction set and returns the response to the smart contract which, in turn, stores these predictions in the blockchain storage. The calculated predictions are then used to obtain the model performance metrics, such as *cross-entropy* [195], or *binary cross-entropy* [39].

The proposed CDMLB method combines models by averaging their predictions. The model predictions are defined as:

$$p_i = f_i(x) \tag{5}$$

where p_i is a predicted value (output), x defines a feature vector (input), f_i is an individual classifier. To calculate an average prediction for the classifier i, the average prediction is calculated by:

$$\hat{y} = \frac{\sum_{i=1}^{n} p_i}{n} \tag{6}$$

where n is the number of classifiers in the ensemble and p_i is their predicted value. The averaging of predictions is applied for each data row in the used dataset, and it is used to calculate the classifier training loss measure.

Cross-entropy was first introduced as a measure to calculate differences between two probability distributions. It was first applied in the field of information theory and developed by using the concept of entropy [196]. In the context of machine learning, cross-entropy is usually used as a loss function [197]:

$$CE = -\sum_{i=1}^{M} y_i log(\hat{y}_i) \tag{7}$$

where M is a total number of the predicted classes, $\hat{y}$ is the predicted probability of the class, and y is the ground-truth class value, which is a one-hot encoding indicator of the true class. Such a loss function can be used for multiclass classification tasks. In the context of this work, only binary classification tasks were considered; thus, the *Binary Cross-Entropy* (BCE) measure was selected as the main loss function. To apply the proposed method for multiclass classification tasks, the BCE measure should be replaced by the CE loss function.

Binary cross-entropy was selected on the basis of its popularity in classification tasks. Binary cross-entropy quantifies the difference between the predicted class probability and the ground truth class, by representing the target class as '1' and the non-target class as '0'. Binary cross-entropy can be calculated by using the following formula:

$$BCE(y, \hat{y}) = -(y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y})) \tag{8}$$

where y characterizes the ground-truth class label and $\hat{y}$ represents the model's class probability prediction. The calculated BCE metric is appended to every instance of the model prediction set recorded to the blockchain storage.

By using the proposed CDMLB method, the evaluation of each network participant for both data and model contributions can be made. As both the data and the model are required to produce predictions, the network participant which deploys the smart contract should provide at least one instance of both artefacts.

2.2.2 Contribution calculation

The existing propositions [10], [11] suggest that data importance to the overall ensemble performance can be quantified. The CDMLB method measures an individual contributor's dataset impact φ for the network contributor *n* by evaluating the model ensemble performance subtracted from the performance of the model ensemble *N* containing all other datasets except for the dataset *n*:

$$\Delta\varphi_n = BCE_N - BCE_{N\setminus n} \tag{9}$$

The produced contribution score values are also scaled by dividing all the data contributions from the combined total sum of all scores, and thus producing the scaled contribution value ω_n:

$$\omega_n = \frac{\Delta\varphi_n}{\sum_{i=1}^{N} \Delta\varphi_i} \tag{10}$$

Those network members that produce a value higher than the limit set by the collaborating organisations could be compensated for their datasets based on the contribution. The need by the contributors to gain the most accurate machine learning solution provides motivation to keep sharing the testing data, while ensuring that the uploaded models do not lose their performance due to task shift or noisy validation data.

The CDMLB method combines best-performing models into an ensemble and employs Shapley value calculations to evaluate each model's impact to the overall ensemble performance. Reciprocal BCE is used as a performance metric to measure an ensemble member's contribution impact, and it corresponds to:

$$v(S) = \frac{1}{BCE_S} \tag{11}$$

The Shapley contribution score of a model i is calculated as follows:

$$w_i^{Shap} = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \big(v(S \cup \{i\}) - v(S)\big) - \emptyset \tag{12}$$

where N is the set of classification models, and sub-ensemble S is a subset of $S \subseteq N$, and it represents a coalition. The reciprocal BCE corresponds to the coalition contribution $v(S)$. Marginal contribution of a single coalition member i is represented by $(v(S \cup \{i\}) - v(S))$. The empty coalition $\emptyset$ measure was derived from comparing random guessing predictions of 0.5 and compared to validation data class labels (see Formula 7) For datasets containing classes in nearly equal proportions, the empty coalition measure should be exactly 0.693. A lower performance score than 0.693 would denote that the classifier performance is lower than the random guessing approach. For datasets containing classes of unequal proportions, the empty coalition value can be calculated by setting all prediction to 0.5 and calculating the BCE value. The model ensemble for evaluation is developed for an individual machine learning task. If the network participant has contributed multiple models, in order to reduce the amount required to compute the Shapley values, only a single model with the best contributor score is included. All models which do not increase the performance of the ensemble are omitted by setting their weight to zero according to Formula 12. Such an approach is defined as *positive Shapley* (posShap):

$$f(w) = \begin{cases} w, & w \geq 0 \\ 0, & w < 0 \end{cases} \tag{13}$$

where w is obtained by using Formula (12), and the posShap transformation of w is identical to the rectified linear unit activation function ReLU [198].

The alternative approach which changes the resulting prediction class based on the defined rules for the model prediction set has been proposed, and it is calculated by:

$$f(p) = \begin{cases} p, & w \geq 0 \\ 1 - p, & w < 0 \end{cases} \quad (14)$$

The alternative strategy to resolve negative Shapley values was based on the assumption that the model developer switched class labels while training. Such a switch would result in a negative Shapley value and would decrease the performance of the overall ensemble. Based on the assumptions about the developed classifier, a model prediction inversion strategy was proposed, where p is the classifier output. Such a strategy will be referred to as the *Shapley value with maximum inclusion* (maxShap).

An overview of the evaluation process is presented in **Figure 18**, which consists of 4 stages. At the first stage, contributors train their models on sensitive private data in the local private model development environment. During the second stage, the trained classifiers are validated by using the validation data deployed to the blockchain network. In the third stage, the predictions are made and used to evaluate the performance of the models by using the Shapley-based method. Shapley weights are then stored in the blockchain network. The produced Shapley-based ensemble weights and the contributed machine learning models are combined into a model ensemble during the ensemble usage stage or when applying the model distillation approach. The final weighted model ensemble using local off-chain oracle services can be used to calculate predictions when provided with validation data.

After both the data and the model contribution score have been calculated for the contributors, the incentives can be derived and distributed. The incentives may include monetary or token-based funds, thus providing motivation for the contributors to keep contributing datasets and models to the network.

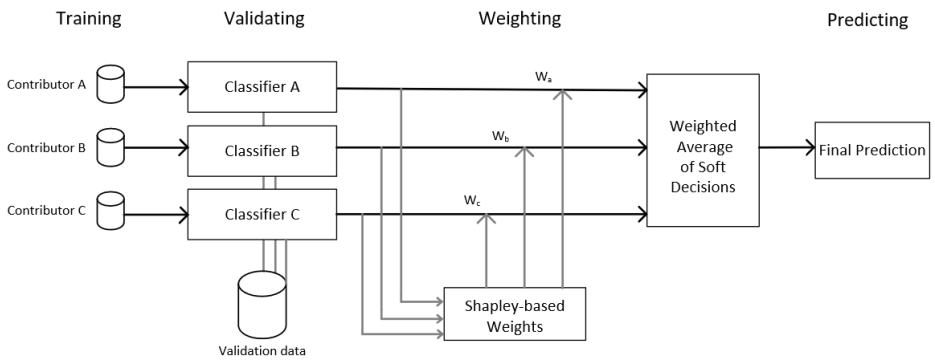


Figure 18. Overview of the proposed ensemble evaluation strategy

Contribution calculations have been implemented by using the proposed CDMLB solution, and two proposed model contribution evaluation strategies have been experimentally tested. The implementation details are presented in Section 3.

2.3 Network Knowledge Usage

The CDMLB method provides two approaches to how network knowledge can be used to model prediction calculations, with a detailed usage process being described in **Figure 19**. Both approaches calculate model predictions by using testing data. Both of the described approaches require information about the models, and their produced results are stored in the contributor's local environment.

The first approach involves aggregating multiple models into a unified ensemble. This ensemble is created by a specialised blockchain oracle service. Through weighted averaging, predictions are generated on testing the data contributed by the users without the need for the data to leave their local environments, thereby ensuring data security and privacy while recording model usage information on the blockchain. Such usage of the weighted ensemble will be defined as the *predicting using weighted ensemble approach*.

In the second approach, knowledge distillation is employed to train an aggregating model using the previously calculated model predictions. By combining the validation data and the models merged into an ensemble, a new neural network model is developed, and the distilled model file is stored locally on the contributor's node. The distillation process preserves the privacy of the training data that were used to develop contributed machine learning models on the blockchain. Such training data privacy preservation is especially important in model usage cases outside the blockchain network.

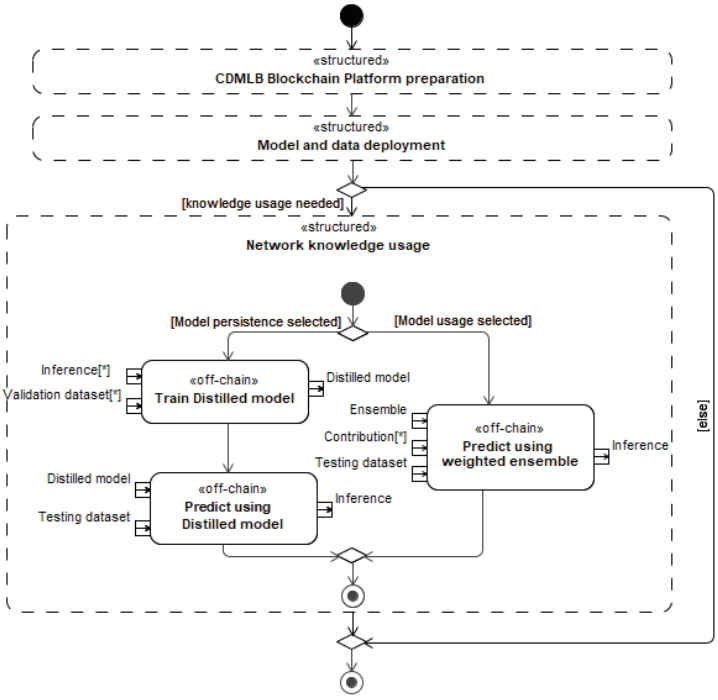


Figure 19. Proposed procedures for blockchain network knowledge usage

The prediction using weighted ensemble enables the usage of network models by providing the testing data to the specialised component. Such usage is recommended when a user is not planning to apply the results in the separate environment as the ensemble can potentially disclose information about the training data in use. For knowledge transferring to external non-blockchain environments, network usage via the distilled model is recommended.

The network usage via the distilled model allows extracting knowledge into a more simplistic representation and enables additional model tuning for further improvement. Such a distilled model can be deployed for usage in non-blockchain environments, thereby allowing for an increased amount of the possible application areas.

2.3.1 Predicting using network ensemble

The prediction using the network ensemble approach uses model files and Shapley-based weights stored in the blockchain ledger to predict the use of the testing dataset without sharing the data to the blockchain network. The approach uses a local off-chain oracle. Local oracle procedures are invoked from the model selection smart contract and are provided with the models. The model section algorithm evaluates the model performance and selects a single model for each network contributor. These models are then aggregated into an ensemble. The ensemble then predicts by combining all the model predictions with Shapley weights by using the weighted arithmetic mean:

$$X = \frac{\sum_{i=1}^{n} p_i w_i}{\sum_{i=1}^{n} w_i} \tag{15}$$

where p_i is a single model prediction, and w_i is the Shapley value for that model in an ensemble.

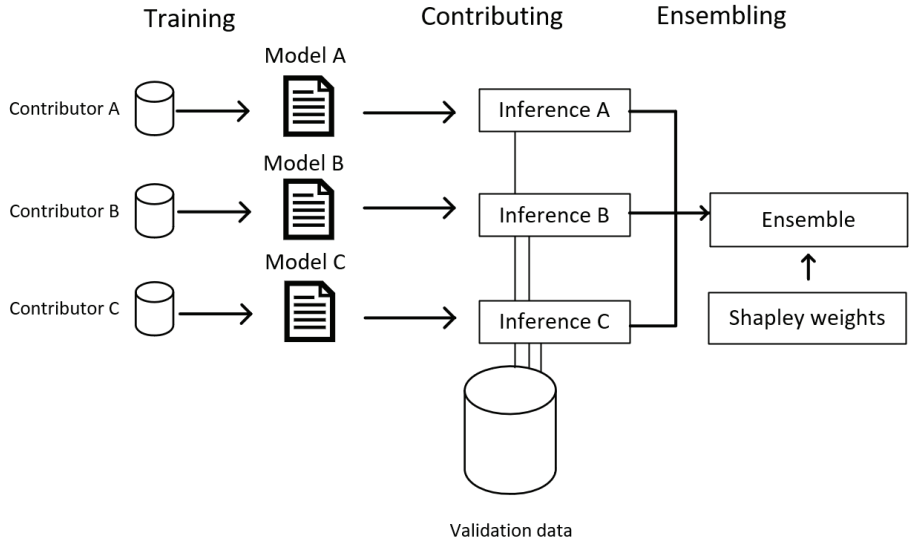


Figure 20. Ensemble creation process

After the model selection step, the contributor provides the testing dataset to the local web application and calculates predictions in the contributor's local environment by using the usage oracle This interaction with the blockchain oracle ensures that the transaction is recorded, but only the information about the access to the model file is saved. All the prediction results are stored in the user's local environment, thus preserving the user's privacy. The process should always return positive results for other network participants and allow them to only verify the model file and the validation data access fact.

The network knowledge usage approach with a weighted ensemble has been evaluated in two experiments, by comparing it with other ensemble weighting strategies, and comparing it with the other approach where network knowledge is transferred into a new neural network model. The implementation details are provided in Section 3, while the experiment results are described in Section 4.

2.3.2 Network knowledge distillation approach

To use the knowledge stored on the blockchain, the proposed method applies the knowledge distillation [199] approach on the model predictions sets stored on the blockchain ledger. The blockchain network knowledge distillation approach (**Figure 21**) allows the contributors to make predictions by using the model trained on the network. Knowledge distillation is employed to reduce the risk of revealing sensitive training data contained in the contributed machine learning models. To obtain the model prediction sets from the blockchain network, the organisation provides access and smart contracts which calculate model prediction sets. By using the smart contract, the model predictions are transferred to the local off-chain component called the *knowledge distillation service* (**Figure 16**). The service uses model predictions of every model with the positive Shapley value shared to the network and trains the neural network. The neural network uses a loss function which combines the distillation loss (the teacher loss) with the loss of the neural network predictor that is in development (the student loss).

The *Kullback-Leibler* (KL) divergence [199] is a statistical measure which evaluates the distance between two probability distributions with one being the teacher model class probabilities p, and the other being the student model class probabilities q:

$$D_{KL}(q \parallel p) = \sum_i p_i log_2 \frac{p_i}{q_i} \tag{16}$$

Class probabilities $q = \text{softmax}(Y_q)$ and $p = \text{softmax}(Y_p)$ in the experiments were calculated as softmax outputs [98]:

$$\text{softmax(Y)} = \frac{\exp(y_i/T)}{\sum_i \exp(y_i/T)} \tag{17}$$

where Y_p represents the predictions of the teacher that is being distilled and Y_q are the predictions of the student classifier. T represents the distillation temperature that

allows ensuring harder or softer probability distribution over the classes. D_{KL} reaches 0 only when the predicted class probabilities are identical ($Y_p = Y_q$).

The final loss for model tuning is calculated by introducing the α measure which allows modifying the ratio between the simple neural network model loss when $\alpha = 1$ and only using the blockchain network ensemble predictions to train the model when $\alpha = 0$

$$loss = \alpha \cdot BCE_{stud} + (1 - \alpha) \cdot D_{KL} \cdot T^2 \quad (18)$$

where BCE_{stud} represents the student loss and D_{KL} represents the distillation loss.

A model trained by using this strategy allows the network contributors to utilise the network knowledge without the need to use the model ensemble constructed on a blockchain network. This also prevents the disclosure of the knowledge to sources outside the blockchain network, as the distilled model does not provide any information about the other ML artefacts recorded in the blockchain ledger. Additionally, it allows network participants to use the network knowledge in a more versatile approach, when transferring a machine learning solution to other application environments.

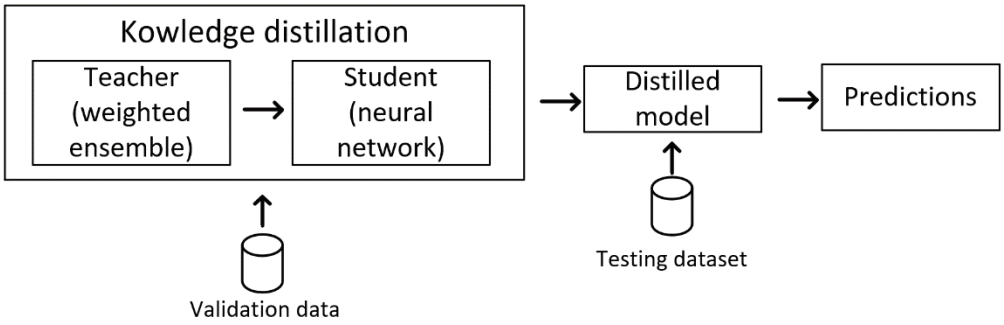


Figure 21. Knowledge distillation architecture

Due to the limitations of the blockchain technology, the blockchain network is not able to verify the result of the distillation process and will be incapable of evaluating the classifier performance or verify the correctness of such a process. The model created by the distillation process can be further developed by using private contributor data in a non-blockchain environment, outside the CDMLB method. The improved model should then be used to make predictions by using the validation data with a higher performance.

2.4 Summary

The method for collaborative distributed machine learning on the blockchain has been proposed. It enables collaboration via the model deployment and inference processes. The proposed method defines three stages in which the collaborating parties prepare the private blockchain network, participate in the collaboration, and utilise the network knowledge via the weighted ensemble usage or the distilled neural network model. The CDMLB method uses blockchain smart contracts to facilitate model and data deployment to the blockchain network and introduces the local

blockchain oracle architecture which enables local execution environments in programming languages that are not supported by the blockchain platform. The proposed CDMLB method introduces automated performance evaluation of the shared model and data artefacts. Such a model contribution is obtained by evaluating the model's performance while using binary cross entropy as a contribution measure in the Shapley value weight selection strategy. The data contribution is defined by measuring the impact on the model ensemble performance. Finally, the proposed method allows its users to utilise knowledge from the blockchain network directly, by weighted ensemble, or indirectly, by the student-teacher distillation approach.

2.5 Limitations of the CDMLB Method

The proposed CDMLB method introduces additional flexibility when compared to the smart contract-only based solutions. The method allows integrating the already existing popular machine learning environments and libraries via the application of the local off-chain oracles, although this introduces additional development efforts and complexity required to create and maintain such a solution for the collaborating organisations. This could increase the upkeep costs of the presented method and discourage possible applicants from using this method.

The introduction of new off-chain components may present additional security and software development challenges. The distribution of the blockchain oracle services by the governing organisation should be considered a priority, as the attacker of the distribution could affect multiple participants at once. The developers of the off-chain contracts should also consider possible tampering of the local services as the services will be deployed in a local participant's environment.

The motivation to use private blockchain requires partial trust between the involved parties to develop collaborative solutions. The utilisation of a private blockchain and the introduction of an organisation entity requires upfront trust between the parties. Yet, the collaborating parties should remain competitive enough to not commit into sharing the training data outright so that to make the usage of the proposed method viable. The proposed method could simply be replaced by a trusted third-party application if the collaborating parties would trust each other enough to share the training data outright.

The testing data stored on the blockchain are accessible to all the participating organisation contributors and could be used to develop adversary models trained on these data or used to generate synthetic validation data which are similar in structure. If some incentive measures exist, both of these attack vectors could be used by malicious network participants to try and exploit incentive measures for unfair gains.

The proposed CDMLB method requires defined validation data and model structures that are supported by smart contracts and local oracle components. The need to develop solutions based on defined structures may lead to delays in applying the updated data structure or new model types and may require improving or introducing additional components, such as local oracles or smart contracts.

Currently, the computational complexity of the proposed CDMLB methods is O(N!) based on the most complex exact Shapley value calculation component. The method specifies that, for every network participant, only a single best performing

model is evaluated, even though, with this precaution in mind, each new network member would significantly reduce the performance of the blockchain network. Such computation complexity could be reduced by the usage of the Shapley value approximation algorithms, but their application and effect on the proposed method have not been evaluated yet.

The scalability of the proposed CDMLB method might be limited by the modular architecture in use. In the proposed architecture, every new component is developed by using a separate runtime environment; thus, the sharing of the common runtime costs between the components is not possible. Because of this, the increased number of the local oracle components could reduce the performance and non-efficiently utilise the existing computational resources. Another drawback of scalability in the proposed method stems from the used oracle factory component that is decentralised, but only used by the members of a single organisation. It means that if the size of the participating organisation happened to grow, the computational demand for the oracle factory component would increase with the addition of each new member.

3. IMPLEMENTATION OF THE CDMLB METHOD

3.1 Dataset Preparation

For the implementation of the method, a data partitioning strategy was used, which is presented in **Figure 22**. The selected dataset was prepared by performing categorical data reduction on selected features which combined multiple categories into an aggregated category, and one-hot encoding of the categorical text-based data. The dataset was randomly shuffled, and the data were split into training, validation, and testing data. By using the strategy, 80% percent of the original dataset was dedicated for model training and 20% was assigned for testing. The testing dataset was shared to the blockchain network for model performance evaluation. Based on the individual requirements and parameters of the contributor's dataset, the proposed data partitioning strategy can use other ratios. The training data were further divided into the training and validation subsets to measure the performance of the classification models during its training stage.

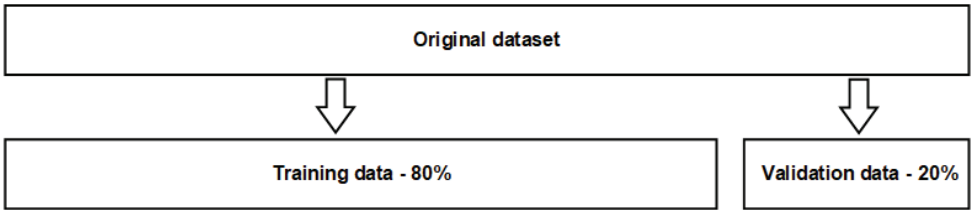


Figure 22. Proposed splitting strategy for the contributors' data

The demonstrated data splitting strategy was applied to two banking-related datasets. In the next step, the prepared training dataset is used during supervised model training.

3.2 Implemented Classifiers

The proposed method was implemented by utilising two base learners:

- *Classification And Regression Tree* (CART) [48];
- binary logistic regression [27].

The learners were selected based on their popularity in the classification tasks and their popularity in the ensemble-based machine learning approaches and the fast model inference calculation time, which allows evaluating a larger blockchain network without wasting computational resources. Binary logistic regression was also selected based on its common usage in the financial domain. The small file representation size of the selected classifiers also exerted influence on the selection of CART and the binary logistic classifier. The selected classifier training can also be completed with a relatively low amount of computational resources, thus enabling a higher possibility of collaboration. The CART classifier was also selected by its explainability. The CART classifier utilises data partitioning to develop a tree-like structure. In this classifier, the leaf nodes contain predicted values for the input vector.

The intermediate nodes contain data splitting decisions. Every decision rejects some potential sequential points and additionally discards some classification choices. The traditional decision tree classifier is developed by using a greedy split algorithm and a Gini impurity measure applied to the learning data. The following formula is used to calculate the Gini impurity:

$$\mathrm{Gini(k)} = \sum_{\mathrm{i} \subseteq \mathrm{C}} \rho_{\mathrm{i,k}} \left(1 - \rho_{\mathrm{i,k}}\right) = \ 1 - \sum_{\mathrm{i} \subseteq \mathrm{C}} \rho_{\mathrm{i,k}}^2 \tag{19}$$

where C is a list of all classes, k is a specific category after the split, and $\rho_{i,K}$ is a probability of the category k having the class i.

Decision tree classifiers differ from other machine learning models by a clear definition of data splitting decisions that can be displayed in a graph format. Such graphs can be easily analysed, thus denoting the criteria that led to the classifier decision. An example of such a graph is presented in **Figure 23**. The CART classifier is commonly used when combining multiple classifiers into ensembles [200], [201], [202].

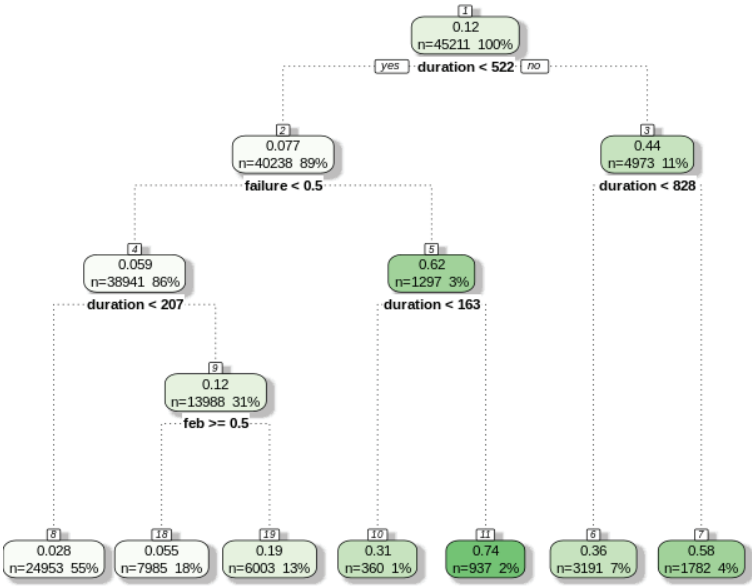


Figure 23. Visualisation of a decision tree classifier for Bank Marketing dataset, where the darker is the colour of the node, the more target class cases of the training data exist there

The second classifier empirically tested in the CDMLB method was the binary logistic regression. The binary logistic regression is one of the traditional approaches [203] to the binary classification problem. It models the linear expectation of vector $X = \{x_1, x_2, \ldots, x_n\}$ belonging to class A as $E(A|X)$, where:

$$E(A|X) = \frac{e^y}{1 + e^y} \tag{20}$$

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \cdots + \beta_n * x_n \quad (21)$$

The logistic regression model fits the weighting coefficients $\{\beta_0, \beta_1, \beta_2, \cdots, \beta_n\}$ in a manner which minimises the logistic loss (log-loss) function:

$$cost(P(A|X), X) = \begin{cases} -log\big(P(A|X)\big), if\ X\ is\ in\ class\ A \\ -log\big(1 - P(A|X)\big), if\ X\ is\ not\ in\ class\ A \end{cases} \quad (22)$$

The CDMLB method can also support other classifiers, but only the decision tree and the logistic regression classifier were implemented. The model training step is concluded by storing the machine learning model into a file.

3.3 Implemented CDMLB Blockchain Platform

The prototype blockchain network was developed by using the Hyperledger Fabric private blockchain framework. Hyperledger Fabric uses a service-based architecture and deploys services using containerisation. Such a deployment method allows us to introduce new services in the developed blockchain environment that can be replicated and developed as a separate service. The Docker container management environment was used to deploy and manage the blockchain network services and blockchain oracle components. The prototype network implementation used two organisations containing multiple network contributors. Each organisation contained an identity service to authenticate the contributor nodes. A single communication channel was deployed between the two organisations.

A model inference smart contract was developed as proof-of-concept implementation and deployed to all the network contributor nodes. By using the Docker container management service, a proof-of-concept oracle factory service was developed by deploying the required local oracle services for all the existing network contributors. Similarly, a local web application incorporating a smart contract allowed uploading the machine learning model and the data. Additional off-chain oracle services were deployed into each individual network contributor node environment.

The local replicated application was implemented as a website using the *Go* [204] programming language by using a web page templating library [205]. The developed website used the *contractapi* [206] library to access the smart contract functions. The developed prototype local web application was deployed into the contributor's node local environment while using a dedicated container service. Every network participant obtains such a local web application from their organisation. The local web applications are replicated identically for all network users and offer exactly the same functions.

The network initialisation stage contained the following steps:

- The contributor would initialise the blockchain network with the provided identity.
- The contributor would create an instance of the blockchain ledger.
- The contributor would connect to the created Hyperledger Fabric communication channel.

- The contributor would obtain the organisation-provided smart contracts, the local web application and the required oracle services from the oracle factory.
- The contributor would initialise smart contracts and oracle services.

After the initialisation, the network participants are ready to contribute data and model files via the local web application.

3.3.1 Implemented smart contracts

To implement the proposed CDMLB platform, the smart contract for collaboration was developed by using the Go programming language. The smart contract specification is provided in Figure 24. The specified functions are used by the local web application; they are used by employing Hyperledger Go API. The developed smart contract contains data structures for models, data and model inference. Each model, data and results are connected to the owner structure which defines its provider. All of these structures can also be bundled into key value pair structures that are used to combine them into addressable slices. The ensemble weights are stored in the weights data structure. The smart contract provides functions to initialise the model, the data files as well as the results calculation. The model and data validation functions are designed to test the format and the structure of the provided data and the model files. All the defined data structures can be obtained via the smart contact via the developed functions. The model inference results are calculated via the calculate predictions file. The Shapley weights are updated every time the *calculateShapleyWeight* function is called, with such calls being included in the data and model initialisation. The initDatafile, initModel file and initResults functions depend on the blockchain oracle-provided functions.

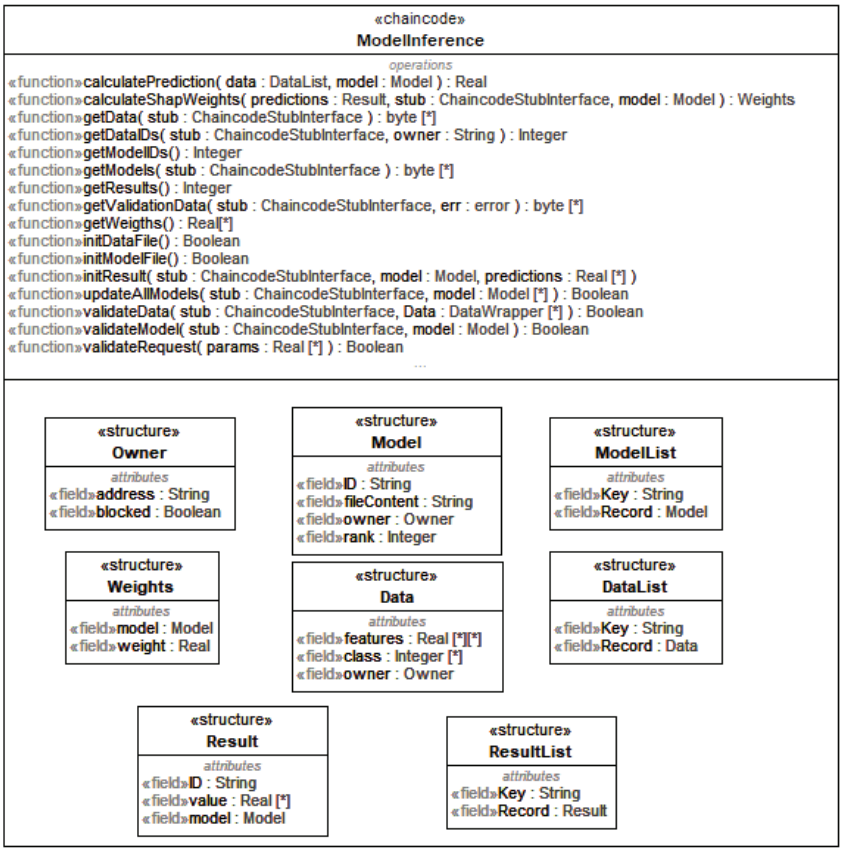


Figure 24. Smart contract implementation for Shapley weight calculation

3.3.2 Implemented oracle services

Two oracle services have been implemented to test the proof-of-concept implementation of the proposed CDMLB. The developed oracle services contained the following functions: 1) the data validation function; 2) the model file and model encoding validation function; 3) the model inference calculation function; and 4) the component setup validation function. The data validation function was designed to test if the provided data file format is correct as well as to compare the provided data batch with the example data structure. The model validation function applies similar actions by testing the model file format as well as the validation if the model can create inference with the defined sample data structure. Both of these functions are called before the smart contract calls the model inference function, thus ensuring that all the network participants are able to perform the inference calculation action. If such a test successfully completes the model inference calculation, the procedure uses the tested data and model files and returns the model inference set to the smart contract. The smart contract, after a successful call, stores the inference results in a *CouchDB*

database. These model inference results are then used in a Shapley weight calculation smart contract function on the model usage request.

The first blockchain oracle was implemented by using the *Python* programming language and with the API service implementation using the *Flask* library and machine learning procedures implemented by using the *PySpark* library. The second blockchain oracle was implemented by using the *R* language, by implementing the *API* service via the *Plumber* library and the powering machine learning functions by the *MLR3* library.

3.4 CDMLB Blockchain Platform Usage

3.4.1 Contributing to the blockchain network

By using the local web application developed by the contributing organisation, the machine learning artefacts were contributed to the blockchain network. The application allows network contributors to upload data in the CSV file format [207]. The dataset file is divided into the feature and label parts. The metadata of the presented dataset are discarded, such as the row numbers and the feature names, to save the storage space. Then, the data are stored in the JSON format [208] in the blockchain storage. The switch between the CSV format to JSON is also motivated by the JSON file format support in CouchDB on the chain storage.

The machine learning classifiers were developed by using the MLR3 and PySpark machine learning libraries and compressed into ZIP files which were deployed to the network. The aforementioned files were encoded into the text-based format by using the *BASE64* algorithm and stored in the blockchain ledger. For both data and model representations, the identification of the network contributor was also stored in a JSON-encoded structure. When this model file is being used to produce predictions, the file is decoded from the text-based structure and used to predict on new data instances.

3.4.2 Contribution evaluation

Due to the currently existing research which proposes means to evaluate the contribution of machine learning datasets [179], [180], the CDMLB method focused on developing and testing the model contribution approach.

Additionally, the method proposes an approach for the model contribution evaluation. The model contribution evaluation approach was implemented by using a machine learning ensemble of logistic regression and decision tree classifiers. The classifiers were developed by using the Python and R programming languages and the PySpark and MLR3 machine learning libraries, respectively. More than one implementation language was chosen to denote the flexible nature of the implementation environments available by using the proposed blockchain oracle architecture. The evaluation of the model performance using model ensembles was developed by using a Shapley-based ensemble weighting strategy. By applying Shapley values, they can be used to assign the contribution scores for an individual machine learning model in an ensemble. The model contributions were evaluated by

developing a decision tree and binary logistic regression classifier ensembles, as well as heterogeneous ensembles which combine the two tested classifiers into a single ensemble.

Contribution calculations with decision tree models and their combination into ensembles were implemented in a proof-of-concept blockchain network with the local web application [209]. This web application enabled network participants to upload data files, model files, and overview changes in the model performance presented in the *Area Under Curve* (AUC) measure.

3.4.3 Implemented weighted ensemble usage approach

The implemented weighted ensemble usage approach was designed to allow the network participants to use the network knowledge by providing testing data to the local web application. The ensemble usage approach was implemented by using two machine learning classifiers. The first implementation used the Python programming language and the PySpark machine learning library. The second implementation used the R programming language and the MLR3 machine learning library. Both implementations were developed for two banking-related tasks.

The ensembles are developed by using a decision tree and logistic regression classifiers. Both homogeneous ensembles were tested as along with heterogeneous ensembles mixing the two selected classifier types. The developed weighted ensemble was compared with the single model approach and other popular weight selection strategies. Moreover, the CDMLB approach was compared with the most similar Shapley-voting based strategy.

3.4.4 Implemented distilled knowledge usage approach using the three layer perceptron architecture

The knowledge distillation approach was implemented by adapting the solution proposed in source [98]. The implementation of the distilled model used the Python programming language and the *Keras* machine learning library as well as the framework provided in source [210]. The trained neural network contained two hidden layers: first with 16 hidden nodes, and second with 8 hidden nodes and a binary output layer.

The neural network was trained by using validation data as well as the used ensemble predictions as the training datasets. The distillation approach used models developed for the ensemble weight calculation experiment. A validation dataset that could be donated to the blockchain network was used to test the produced neural network's performance. The resulting neural network performance was compared to the weighted ensemble which represented the network knowledge usage approach and the single model approach. The knowledge distillation approaches were evaluated at three different levels of distillation. The first configuration $\alpha = 1$ included no blockchain network knowledge and only trained the neural network classifier on the combined validation dataset. This configuration would allow us to evaluate how the inclusion of network knowledge affects the classifier's performance. The second configuration $\alpha = 0.75$ included only a small amount of the blockchain network

knowledge. The third configuration $\alpha = 0.5$ used neural network predictions on the validation dataset and the blockchain network knowledge with the equal ratio.

3.4.5 Implemented distilled knowledge usage approach using the deep learning architecture

To test the classifier performance difference from the shallow neural network architecturedeep learning neural network architecture was implemented.. The implementation used an already existing *TabNet* [211] neural network architecture tailored to tabular data tasks. Such an architecture was selected based on its compatibility with the existing Python model development environment as well as its high performance on the experimentally tested *Bank Marketing* dataset. The implemented neural network modified the code provided in source [212], by implementing the ensemble prediction loading and knowledge distillation loss calculation functions. The developed architecture used the *TensorFlow 1.2* machine learning library to train its model. The implemented deep neural network architecture contained 4 layers, where each layer was composed of the full connection layer, proceeded with the normalisation layer, and finished with the *Generalised Linear Unit* nonlinear activation layer.

The neural network training procedure was similar to the three-layer perceptron training procedure and used the same datasets and models. The comparison with different alfa values set during the training and baseline classifiers matched the procedure performed with the three-layer perceptron classifier.

4. EXPERIMENTAL EVALUATION OF CDMLB METHOD

The experimental evaluation of the proposed method was conducted in three stages. Each stage was dedicated to testing a part of the proposed CDMLB method and evaluating the implementation of the method processes.

The first stage was dedicated to evaluating the possibility to implement the method's CDMLB blockchain platform preparation step and the system architecture based on the local oracles. An experiment defined in source [194] was conducted to test whether the local oracle approach could be a viable solution in the existing blockchain technology and how the introduction of new components into the blockchain network could affect the performance of the system (the experiment results were also published in paper [194]).

The second stage was allocated to testing the contribution calculation part of the method. The experiments concentrated efforts to develop the model evaluation part due to the already existing propositions which evaluate the data contributions [179], [180]. The experiments [192] compared two proposed ensemble weighting strategies with the existing commonly used strategies and the most similar Shapley-voting based weighting approach (the experiment results were also published in paper [192]).

The third stage evaluated the part of the network knowledge usage by measuring the performance of both method usage solutions. The experiment used models developed in the second stage and developed a distilled neural network model. The distilled model was developed by using three distinct configurations, and they were compared to the performance of the ensembles created in the second stage of the experiment.**Performance Evaluation of Model Inference via Local Off-chain Blockchain Oracles**

The model inference calculation algorithm development is a key procedure in the model deployment stage. The goal of the experiment was to evaluate the performance impact of the introduced local off-chain oracle components and to evaluate the model inference calculation algorithm implementation when using the private blockchain technology. The model inference algorithm was implemented by using smart contract and oracle services. To compare the proposed architecture with the already existing solutions, model inference was implemented by using two architectural approaches. The first approach was implemented by using only smart contracts, which covered all the logic for the model inference calculation. The second approach was implemented by using the smart contract and extended with local off-chain oracle components, which relocated the inference calculations and only provided results to the smart contract.

The Hyperledger Fabric blockchain network was used as a framework and the experiment execution environment. The Hyperledger Fabric private blockchain was chosen due to its modular architecture, the consensus algorithm which does not incur any cost attached to the smart contract execution, and the ability to call external network components from smart contract services. The Go programming language was chosen as the smart contract development language, since the Hyperledger Fabric

blockchain provides native support for this language. Two solutions implementing the model inference calculations were developed: a solution using exclusively the smart contract; and a solution using the smart contract extended with the computation transfer to the local off-chain oracle service component which was implemented as the *RESTful* microservice.

The logistic regression model type was utilised in the experiments as the classifier of choice. Such a model type was selected due to the small model representation when stored into a file, and due to the quick model inference calculation performance. The low model inference calculation time requirement was introduced to evaluate the introduced network communication overhead in more detail, rather than focusing on the model inference calculation execution time efficiency. The machine learning models used in this experiment were trained in a separate environment. This environment simulated the local model development environment, and the results of the model development stage were stored into the file format. The model training process represented the proposed model development steps presented in the CDMLB method, where no training data are exported from the local environment. The completed experiments compared the performance of only the model inference calculation part of two different implemented approaches. The model predictions in the context of the proposed CDMLB method are recalculated when a new model or new data are shared to the blockchain network.

The experiment was executed by uploading the validation dataset file to Hyperledger Fabric CouchDB by using smart contract functions called from the command line interface. The model files developed in a separate local model development environment were then uploaded to the blockchain by using the same procedure. During the model upload, the model inference calculation function was executed. The execution time of the function was measured. The model upload procedure was repeated 100 times, and the performance overhead was measured by using the Formula:

$$Overhead = 100 \times (median\left(\frac{T_{OracleAPI}}{T_{Chaincode}}\right) - 1) \tag{23}$$

where T represents a set of runtimes which were calculated over 100 iterations by using logistic regression models. The overhead was calculated by measuring the percentage of performance increase over the chaincode configuration. The model was developed by utilising the G*oML* library [38] randomly sampled data batch. The model inference execution time would start from the initial call of the model inference calculation function and would finish when the function provided results evaluating the total time spent in the model inference calculation function.

To fairly evaluate the performance of the local oracle components, the defined experiment steps were performed on a new Hyperledger Fabric network instance which did not contain any information about the previously performed experiment part. The benchmarking experiment for local off-chain oracle computations also measured the model inference execution time on a smart contract which transferred the model inference calculation logic to the separate local oracle component, and the introduced API calls to that service from the smart contract environment.

4.1.1 Experiment settings

The experiments were conducted on a server with the following configuration: 8 *Intel Xeon Silver 4114* CPU, 16 GB of RAM, SSD data storage. The experiments used the *Ubuntu 18.04* operating system, *Docker 19.03.6* containerisation environment that was running *Hyperledger Fabric 1.4.9* with *CouchDB 2.3.1* as the network database. To reach the consensus, the transactions had to be validated by all the existing network participants. All the peers were connected to a single organisation structure.

The model inference smart contract was developed by using the Go programming language. The local oracle was implemented by using the Go programming language as a RESTful service API component by utilising the Go net/http library. The local oracle implementation was identical to the smart contract developed for local computations, by only replacing the model inference calculation with external calls to the off-chain oracle service.

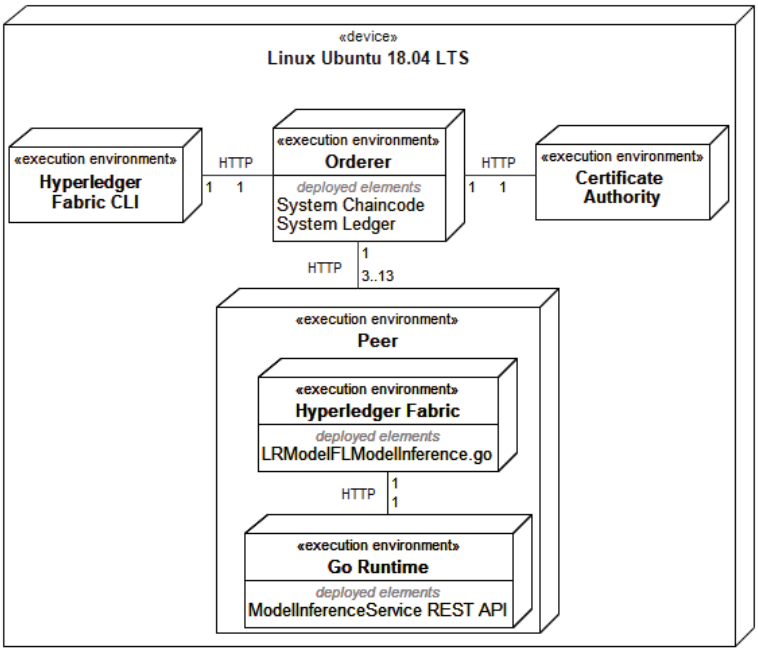


Figure 25. System configuration for the blockchain network components

The experiment prototype deployment configuration is presented in **Figure 25**. All the components were hosted on the *Ubuntu Linux* server, with all of the blockchain network components running in a Docker environment. Hyperledger Fabric CLI (command-line interface) was running in a single network peer node which was used to call the smart contract functions. The orderer and certificate authority nodes were managed by executing commands from the CLI component. The certificate authority manages the network participant access rights, and the orderer component distributes the transaction validation commands for the consensus algorithm. Every network peer

node contained a smart contract for the model inference and a copy of the distributed ledger. The local off-chain oracle service was also deployed to every network member node.

Two datasets were used to benchmark the performance of the developed model inference solutions.

1. The synthetic dataset was composed of the generated data with values representing the coordinates in a two-dimensional space resulting in two features. The values were distributed in the non-linear fashion in crescent -like shapes and were generated with the make_moons function from the scikit-learn *Python* library [213].
2. The EEG eye state dataset [214] was composed of 14 features. The dataset was developed by recording the signal outputs of 14 channels of the *EMOTIV EEG Neuroheadset*, and the test subject eyes were either open or closed during the recording.

The data sets were selected according to the suitability for the binary classification task. The synthetic dataset was selected due to its flexibility in generating any number of data instances, which allowed testing the performance drawbacks of datasets of larger sizes. The EEG eye state was selected to represent an example dataset for sensitive data that could be used in collaboration from the healthcare field. The dataset contained a sufficient amount of data instances and a balanced target against the non-target class balance.

Both datasets were divided into smaller subsets, by following the list: {1024; 2048; 4096; 8192; 16384; 32768}. The EEG dataset only contained 14980 instances; so, in order to obtain the required amount of data, the set of rows had to be expanded. This expansion was completed by bootstrapping new data based on the original dataset, by appending duplicate rows while preserving the original ratio of the positive and negative classes. The data were stored in the blockchain storage with data indexing, which enabled to speed up the reading process. The dataset record sizes when stored in the blockchain storage are presented in **Table 7**. The number of the network nodes participating in the blockchain network was set according to the following list: {3; 5; 7; 9; 11; 13}. Such amounts of network nodes were selected to represent the gradual growth of the network size, and the maximum value was set based on the available computation resources. For each data and network member count combination, the blockchain network was reset to remove any remaining artefacts from the previous configurations of the experiment.

Table 7. Tested dataset configurations and data size.

Data Records	Synthetic Dataset (kB)	EEG Eye State Dataset (kB)
1024	114	510
2048	227	1023
4096	453	2049
8192	905	4097
16 384	1809	14 446
32 768	3650	16 384

To accurately measure the performance, each operation in the experiment was repeated for 100 iterations; on each tested data and node configuration setting, the median value of all the tested iterations is presented as the experiment result.

4.1.2 Benchmarking results for the synthetic dataset

The median model inference calculation time for all the tested network member and dataset configurations are presented in **Figure 26** and **Figure 27**. The model inference calculation time depends on both the network node count and the dataset size as this is indicative in **Figure 27**. A sharp increase in the model inference execution time can be seen after the dataset size has reached more than 8192 rows. The performance of the calculation only increased in a linear fashion when datasets contained 16384 or 32768 rows.

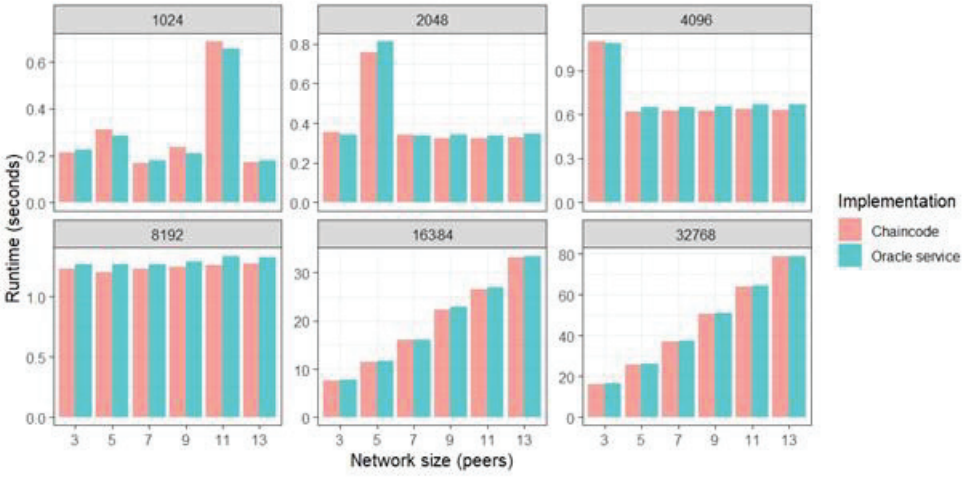


Figure 26. Model inference calculation time comparison. Synthetic dataset

Even though the two calculations of time surfaces are similar in shape (**Figure 27**) and size when separated based on the network size (**Figure 26**), a clear difference

can be seen, which confirms that a small amount of the overhead is introduced when the calculating was relocated to the local off-chain oracle service.

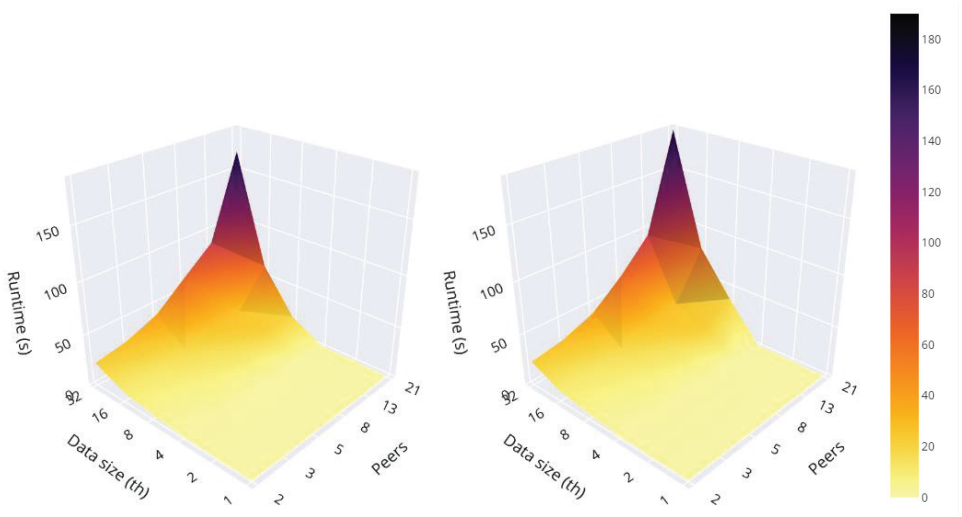


Figure 27. Model inference calculation runtime. Synthetic dataset: smart contract (*left*) and the local off-chain oracle service (*right*)

The largest tested data amount presents a linear relationship between the network node count and the model inference execution time (Figure 26), except for the outliers which were always produced from the initial run of the tested configuration. The outliers for the smart contract implementation produced a constantly quick calculation time as, for the local off-chain oracle service implementation, the outlier values were more distributed, particularly on the network configurations with the member count higher than eight.

The longest execution time was achieved with the highest amount of network nodes and the largest dataset settings. The experiment results for all iterations are presented in **Figure 28**. The statistical analysis presented in Table 8 reveals that the execution runtime central tendencies (the mean and the median) have no statistically significant difference. The extended statistical analysis of the performance results is also presented in Appendix A. The difference in the computation time might have been introduced due to the communication with the local off-chain oracle service. During this communication, data from the blockchain storage to the oracle service were being transferred as well as the result back to the smart contract. Such communication might have affected the consensus algorithm speed, thus introducing an overhead. Most of the execution time was clustered around 79 s for both implemented solutions for a network of 13 nodes with 32768 instances of synthetic data. The lowest execution times presented in this graph were both achieved as the first iteration in the experiment, as the network nodes were still free of tasks.

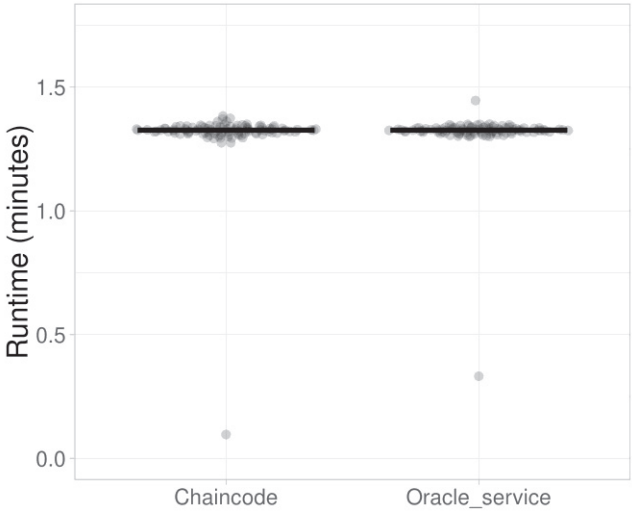


Figure 28. Distribution of the calculation time results with a network composed of 13 members and 32768 instances of synthetic data in the model validation experiment with a median calculation time of 1 minute 19 seconds for the smart contract (*left*) and 1 minute 19 seconds for the oracle service (*right*)

Table 8. Equality of central tendencies using independent sample tests: case of synthetic dataset

		Statistic	df	p
Runtime	Student's t	-0.211	198	0.833
	Mann-Whitney U	4953		0.910

Note. $H_a\ \mu_{\text{Chaincode}} \neq \mu_{\text{Oracle service}}$

The model inference calculation performance overhead presented in percentages is provided in

Table 9. these results denote that, on the synthetic dataset, the calculation overheads were both negative and positive. The total median overhead was 1.99%. The results presented that, on all the network member count configurations with low dataset sizes (1024–8192 records), the computation overhead ranged from −4.31% to 6.59%. The validation times for these settings were relatively low, whereas the variability in performance overhead was quite high. When the dataset size reaches at least 16384 records, the variability in the overhead is reduced in all the tested network configurations, and it is in the range from 0.73% to 4.28%. With the largest dataset size tested (32768 records), the variability in the model inference calculation times drops significantly, only resulting in the highest overhead of 2.79% on the network configuration that only contained three network nodes. The remaining network

configurations for this dataset size resulted in less than 1% (0.03–0.73%) overhead. The results display that the overhead diminishes once the member count and the dataset size increases. The higher variability of the model inference calculation overhead for smaller datasets was observed for all the tested network member count configurations.

Table 9. Performance overhead for the local oracle service approach when compared to the smart contract approach (in %) results for the synthetic dataset

Data records	Number of peers in the blockchain network					
	3	**5**	**7**	**9**	**11**	**13**
1024	2.66	2.56	6.28	-4.31	-1.51	6.59
2048	2.18	4.48	-0.63	3.72	4.99	5.31
4096	1.17	5.61	3.93	5.37	4.27	4.89
8192	3.16	5.01	2.85	3.79	2.96	3.59
16348	4.28	1.20	0.32	1.36	1.02	0.77
32768	2.79	0.73	0.63	0.40	0.05	0.03

The benchmarking experiment results in the tested simulated blockchain network displayed a reduction of the performance overhead in the local oracle components on larger datasets. The results for medium-sized networks (5–9 peers) displayed the performance overhead in range from 6.28% to a performance increase of 4.31%. A large network of 11 to 13 members presented a performance decrease in the range of 0.03% to 6.59%.

4.1.3 Benchmarking results for EEG eye state dataset

The median model inference calculation time for all the network member and dataset configurations tested is presented in **Figure 29** and **Figure 30**. The results present a similar pattern to the results produced on the synthetic dataset. As with the synthetic dataset, the medians presented the biggest increase in the calculation time when the dataset size reached 8192 or more. The higher number of data features and the bigger dataset size in the blockchain storage increased with the time required to calculate the model inference, thus displaying higher median values for the EEG eye state dataset. Just like in the synthetic dataset results, the linear dependency to the dataset size and the network member count to the performance was only noticeable for the two largest dataset configurations (16384 and 32768).

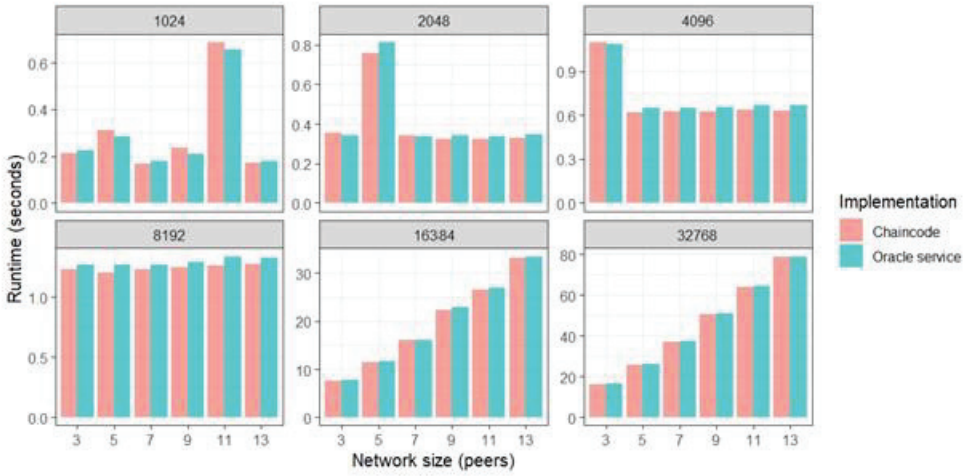


Figure 29. Model inference calculation time when using EEG eye state data

For a dataset with 32768 rows, the linear dependency of the calculation time to the network member count is apparent (Figure 29). Except for the outliers which were produced by the initial run of the defined network configuration, the outlier distribution is similar to the synthetic dataset where the outliers in the smart contract-only approach are fixed at a constant low calculation time, while the outliers of the local off-chain oracle service fluctuate.

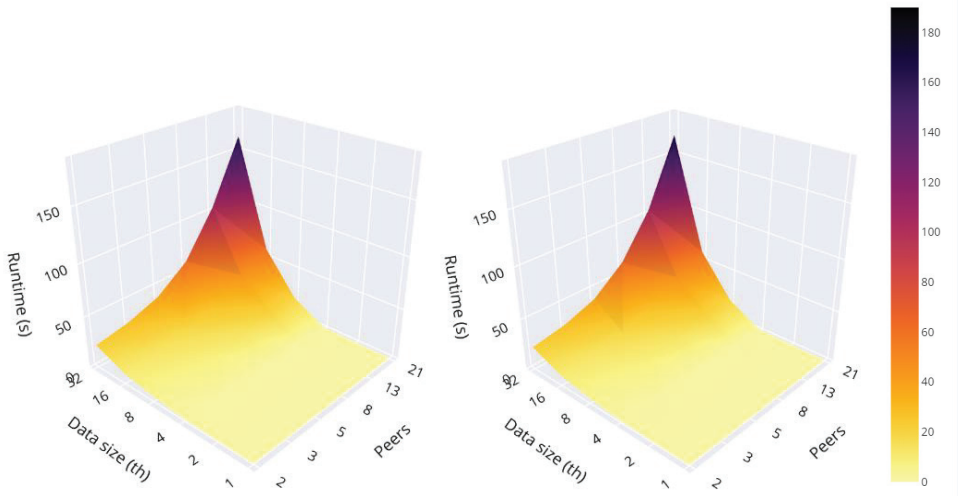


Figure 30. Model inference calculation runtime. EEG eye state dataset: smart contract (*left*) and the local off-chain oracle service (*right*)

The calculation execution time for the largest amount of data is presented in Figure 31. The statistical analysis presented in Table 10 reveals that the mean runtime has no statistically significant difference, while the median runtime shows a statistically significant difference. The extended statistical analysis of the performance results is also presented in Appendix A. The initial iteration of the

experiment presented the lowest calculation times in both approaches, thus demonstrating that the calculation overhead is introduced by the oracle service. The experiment results are clustered around 1 min 24 s mark for the smart contract validation and around 1 min 25 s for the local oracle service approach.

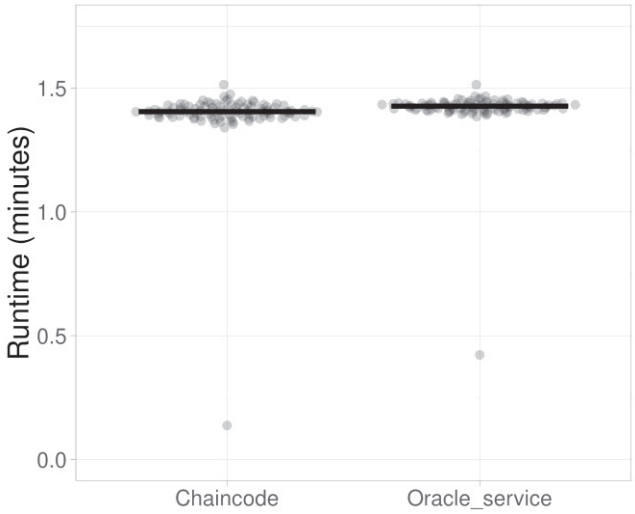


Figure 31. Distribution of the runtime results when using a network of 13 peers with the dataset of 32768 records in the model testing experiment. The median runtime for model validation was 1 minute 24 seconds for the smart contract (*left*) and 1 minute 26 seconds for the oracle service (*right*)

Table 10. Equality of central tendencies using independent sample tests: case of EEG eye state dataset

		Statistic	df	p
Runtime	Student's t	-1.37	198	0.173
	Mann-Whitney U	2677		< .001

Note. $H_a\ \mu_{\text{Chaincode}} \neq \mu_{\text{Oracle service}}$

The results of the oracle service for larger datasets and network sizes are presented in **Table 11**, where the performance overhead is presented in percentages. The oracle service increased the calculation time over all the tested data and network configurations with the total median performance overhead of 4.06%. The results demonstrate that the calculation overhead for networks with lower member counts (3–7 members) and dataset sizes (1024–32768 records) was in the range of 2.16 to 9.67%. The highest overhead was present on the configuration with the lowest dataset size (1024 records), whereas the lowest amount of overhead was observed on the highest amount of data records (32768 records). For the network sizes with 9 to 13 members,

an overhead in the range of 0.80–6.91%, the peak overhead was observed when the dataset size was of the medium size (2048–8192 records) with 6.91% for nine peers, 6.53% for 11 peers, and 6.76% for 13 peers. The lowest performance downgrade of 0.80% was detected in the network which contained 11 members, and the size of the data set was 32768. In total, the highest amount of the performance overhead ranged from 0.80% to 9.67%.

Table 11. Performance overhead for the local oracle service approach compared to the smart contract approach (in %) results for the EEG eye state dataset

Data records	Number of peers in the blockchain network					
	3	5	7	9	11	13
1024	9.32	7.69	9.67	3.52	1.93	4.61
2048	4.83	4.36	2.80	6.91	5.89	4.95
4096	2.13	6.88	5.67	6.19	6.53	5.41
8192	5.73	5.05	7.01	6.65	6.09	6.76
16384	4.06	2.97	2.35	2.34	1.54	2.69
32768	2.17	3.63	2.16	1.76	0.80	1.82

4.1.4 Summary of experimental results

The model inference calculation execution time comparison results for the synthetic dataset show a minor increase of the runtime overhead of ~2% and the mean results for individual configurations of less than 6.60%. Datasets with a lower size presented higher performance overheads due to the higher data transfer time when compared to the model inference calculation time. With higher sizes of the network member nodes and larger datasets, the performance overhead diminished because the ratio of the inference calculation execution became significantly larger than the required data transfer duration.

The model inference calculation time comparison of the smart contract and the oracle-based results for the EEG eye state dataset resulted in the total median increase of the execution overhead by ~4%. When considering the means for both tested datasets and the network member count configurations, the results were distributed from 0.8% to 9.7%. Despite the feature dimensionality of the EEG eye state dataset being 7 times larger than the synthetic datasets, and the instance count was 4 times larger over all the tested configurations, the total mean only increased by 2%, which confirms that the local oracle calculation time depends on both the calculation time allocated for the model inference rather than the dataset and the result transfer. Overall, the model inference calculation performance is not as affected by changes of the dataset size as it is affected by the increase of the network member count. The experiments revealed that each additional network member adds an additional 6 s of the calculation time when tested with the dataset of the highest size.

The results of the conducted experiment reveal that the model inference calculation time increases due to additional communication between the smart contract and the oracle services are not as significant when compared to the flexibility which the oracle services introduces. Whereas, it enables users to run model inference calculations by using the established machine learning environments and solutions.

Regardless, the model inference calculation trade-offs should be evaluated by the organisations looking to adopt the blockchain technology and oracle services to decide whether the faster execution time of an exclusively smart contract solution is sufficiently lower than the local off-chain oracle service solution which allows to adopt the already established ML solutions and components.

4.2 Shapley-based Ensemble Weighting Strategies Performance Evaluation

The experiments for the Shapley-based ensemble weighting strategy were designed to test whether the model contributions to the blockchain network could be evaluated and measured. The experiments focused on the model performance evaluation rather than on the contributed data evaluation. Ensemble-based model aggregation methods were selected for a combination of multiple model types, and they would not require the unification of the model types. The experiment tested the performance of two Shapley-based performance ensemble weight selection strategies and compared this performance to the most commonly used weighting strategies, as well as other Shapley-voting based strategies. The blockchain local off-chain oracle components using the R and Python programming languages were chosen as the experiment implementation environments. Classifiers were developed by the MLR3 machine learning library in the R language environment and the PySpark machine learning library in Python. Two banking-related datasets were used to train classifiers and evaluate their performance.

The presented CDMLB method enables the usage of multiple model types developed by using a wide range of technologies in the collaboration process. To test how a combination of model types would affect the ensemble performance, the weighting strategy experiment tested an ensemble composed of a single model type (homogeneous ensembles) and ensembles composed of multiple model types (heterogeneous ensembles). The experiment began by shuffling the dataset and dividing two selected datasets into the training, testing, and validation subsets. The training subset was further divided into a number of parts matching the number of the trained models. The divided dataset files were stored into files and were used to train models on their allocated training data on both implementations. The implementations using multiple machine learning environments allowed us to demonstrate the flexibility of the proposed method and to test how multi-environment blockchain oracles can be implemented and what changes are required to the smart contract development procedures to implement such a system.

The model predictions calculated on the testing dataset were stored into files. The ensemble validation dataset and the developed model predictions were then used to combine the models into an ensemble and measure its performance. The binary cross-entropy performance measures were used as an input for the Shapley value calculations. Based on the Shapley value results, the ensemble weights were developed. The tested ensemble weighting strategies and the monolith single model approach were then benchmarked by using the validation dataset.

4.2.1 Settings for the experiments

Experiments containing a homogeneous ensemble type were conducted with two classifiers: the decision tree (CART) [48] and the logistic regression [27]. Heterogeneous ensembles combined both model types into a single ensemble. The hyperparameters for the tested model types are presented in **Table 12**. The logistic regression model had two common parameters, a defined epsilon constant, and the number of iterations was set to 25. For the MLR3 library, the singular.ok parameter defined that the strategy to resolve singular design matrices are enabled, and the trace parameter disabled the additional information logging. For the PySpark library, the parameters set were the regularization parameter (*regParam*), the aggregation depth (*aggregationDepth*) with its default value, the prediction threshold parameter (threshold) was set to 0.5, the ElasticNet (*elasticNetParam*) mixing parameter was set to 0, and the bias inclusion into the model (*fitIntercept*) was enabled. For the decision tree classifier, the common parameters were the maximum depth of the decision tree set to 30, and the model training gain was set to 0.01. For MLR3 implementation, the minimal weights required in a node to be evaluated before splitting was set to 20, the number of competing splits (*maxcompete*) maintained in the output was set to 4, the number of surrogate splits (*maxsurrogate*) maintained in the output was set to 5, the surrogate selection method was set to 0 (*surrogatestyle*), and the surrogate usage style was set to 2 (*usesurrogate*), whereas the number of cross-validations (*xval*) was set to 10. The decision tree classifier type had the parameters (*minInstancesPerNode*), standardisation was disabled, the minimal weight (*minWeightFractionPerNode*) for each node could be obtained after the split was set to 0.0. The minimal number of instances after splitting which each node (*minInstancesPerNode*) was required to obtain was set to 1. The number of different bins (*maxBins*) to split features into was set to 32, and the implementation used the Gini (Formula 18) impurity measure which was used in deciding when to split the tree.

Table 12. Model training hyperparameter metrics for the used ML model types

Model type	**Common parameters**	**Implementation-specific parameters**
Logistic regression	E = e *10^8 iterations = 25	**MLR3**
		singular.ok = True; trace = False
		PySpark
		regParam = 0.0; aggregationDepth = 2; threshold = 0.5; elasticNetParam = 0.0; fitIntercept = True
Decision tree	maxDepth = 30 minInfoGain = 0.01	**MLR3**
		minSplit = 20; maxcompete = 4; maxsurrogate = 5; surrogatestyle = 0; usesurrogate = 2; xval = 10
		PySpark
		minInstancesPerNode = 20; Standardisation = False; minWeightFractionPerNode = 0.0; minInstancesPerNode = 1; maxBins = 32; impurity = 'gini'

Two datasets were used to train the five ensemble weighting strategies: Bank Marketing and BNG-Credit_a. The dataset parameters and sources are presented in Table 13. These datasets were selected based on the dataset compatibility with the binary classification task and a representation of the finance field which could be improved by the collaboration. Bank marketing was selected to represent datasets with a skewed target to non-target class ratio, while the BNG-Credit_a dataset was selected to test the ability of the proposed methods to perform with large datasets. In contrast to the datasets used in the performance evaluation experiment, the selected datasets contained categorical data which expanded the feature list.

The Bank Marketing dataset had to be pre-processed, as the dataset consisted of text-based categorical data. The categorical data were transformed into new dataset features by using one-hot encoding approach. The class labels for this dataset were also transformed from the text-based classes into the number-based classes.

The BNG-Credit_a (BNG) dataset also had to be pre-processed as some features contained a wide range of unique instances. To reduce the number of new features after the application of the one-hot encoding approach, the dataset features with 25,000 entries for feature A6 and 89,044 entries for feature A7 were combined into a new 'other' category. Feature A6 contained information about the customer's occupation, whereas A7 contained information about the last contact information about the customer. After successfully reducing the number of unique instances in text-based data features, the one-hot encoding approach was used to transform data instances into a new feature set

Table 13. Dataset parameters. Categorical features were obtained by using the one-hot encoding approach, thus resulting in multiple new features

Data Characteristic	**Bank Marketing** [215]	**BNG-Credit_a** [216]
Initial features	16	15
Categorical features	10	10
Total features	51	33
Total instances	45 211	1 000 000
Classes	2	2
Target class proportion	0.120	0.544

To compare the ensemble-based machine learning models with a monolith model, a two-step data-splitting strategy was implemented. The first step divided the data into training and testing subsets with a ratio of 80% to 20%, respectively. The training dataset was used to train a monolith model; it was also used to develop models which will later be combined into model ensembles. The testing data were reserved to fairly evaluate and compare two approaches: the monolith and the weighted ensemble. The second step divided the testing data subset into two new subsets, specifically, the ensemble training and the ensemble validation. The larger part of the ensemble training data was reserved for ensemble model training and using divisive split into smaller batches based on Zipf's law distribution with its exponent value set to $s = 0.2$, where the Zipf values were calculated by Function (24):

$$f(k;N,s) = \frac{1}{H_{N,s}} \cdot \frac{1}{k^s} = \frac{1}{\sum_{k=1}^{N} \frac{1}{k^s}} \cdot \frac{1}{k^s} \quad (24)$$

where $f(k;N,s)$ is the amount of data rows allocated to a specific participant, the participant's rank is k, s is an exponent (a parameter controlling the shape of distribution); N is the total number of participants; and $H_{N,s}$ is the normalisation constant (the generalised harmonic number).

Models for the ensemble were trained on an individual batch and tested on the ensemble validation dataset to measure its performance in BCE.

For each new experiment iteration, the data instances in the training data batches were randomly distributed. The baseline model for comparing homogeneous results with heterogeneous results was selected based on the best homogeneous classifier performance. The monolith model creation approach was selected based on the highest overall performance.

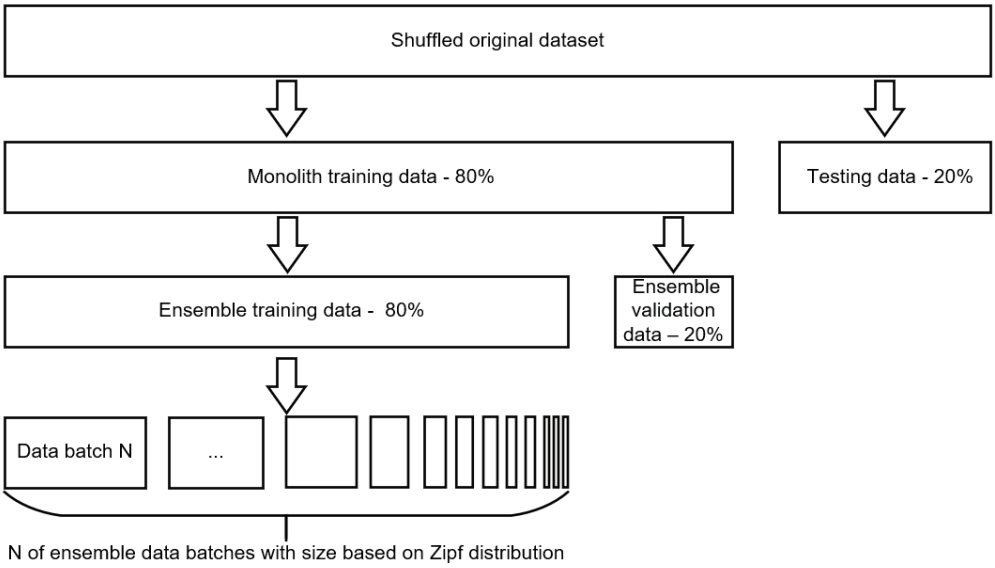


Figure 32. Data splitting strategy for ensemble and monolith development

The Python and R languages were chosen on the basis of the popularity and the amount of the machine learning solutions available in their ecosystem. MLR3 version 0.13.3 was used to develop machine learning models in the R environment. PySpark version 3.1.2 was used to develop a machine learning model in the Python environment. The unified and implementation specific model training parameters used in this experiment are described in Table 12.

The experiments compared the classifier performance of the weighting strategies defined in Section 2.2.2 as the positive Shapley (posShap) and the maximum Shapley (maxShap). These strategies were compared with the monolith approach (Mono) and the four established ensemble weighting strategies: the random weighting (Rand), the equal weighting strategy (Equal), the performance-based weighting (Perf), and Voting-based, as presented by Benedek Rozembecky and Rik Sarkar (Roz) [180].

Each experiment results are presented by using two diagrams. The first diagram presents the ensemble performance evolution based on the ensemble model size. The second diagram presents the classifier ranks based on their average performance.

The monolith (Mono) is a single model developed on the monolithic dataset approach and provides baseline capabilities of the classifier without applying ensembling or weighting strategies.

The models with a negative impact on the ensemble performance were required for the maxShap strategy, but they did not occur in the BNG_credit-a dataset; thus, the maxShap strategy is only represented in the Bank Marketing dataset results.

The critical difference diagram compared the classifier performances. It was developed by using the library, as presented in source [217]. The comparison employed the Friedman's test [218] to determine if any statistical significance exists between the model results. In case the results presented statistical significance, the proposed solution applied pairwise analysis, as described in source [219] by replacing the average rank comparison with the Wilcoxon signed-rank test [220] adjusted with Holm's alpha correction [221]. Instead of comparing the distribution difference, the Wilcoxon signed-rank test additionally adds the assumption of a symmetric distribution. Meanwhile, Holm's alpha correction adjusted the p-value from the individual test to maintain control over the family-wise error rate.

The detailed experiment results are presented in Appendix B. The following sections will compare the median experiment results and the rankings of the classifier performance.

4.2.2 Experiment results for homogeneous ensembles

Ensembles composed of logistic regression classifiers that were developed by using the BNG-Credit_a dataset are presented in Figure 33. The lowest log-loss (BCE) value was produced by the monolith model with a result of 0.326. All the tested weight selection strategies resulted in a similar loss for all the tested ensemble size configurations. The aggregated ensemble performance ranks are presented in a critical difference diagram (Figure 34). The model ranks specify that most of the weight selection strategy results were statistically similar to the Roz strategy. This indicates that the performance increase provided by weighting was low for the logistic regression classifier. The monolith approach was ranked as the best performing classifier, with the posShap approach coming second best.

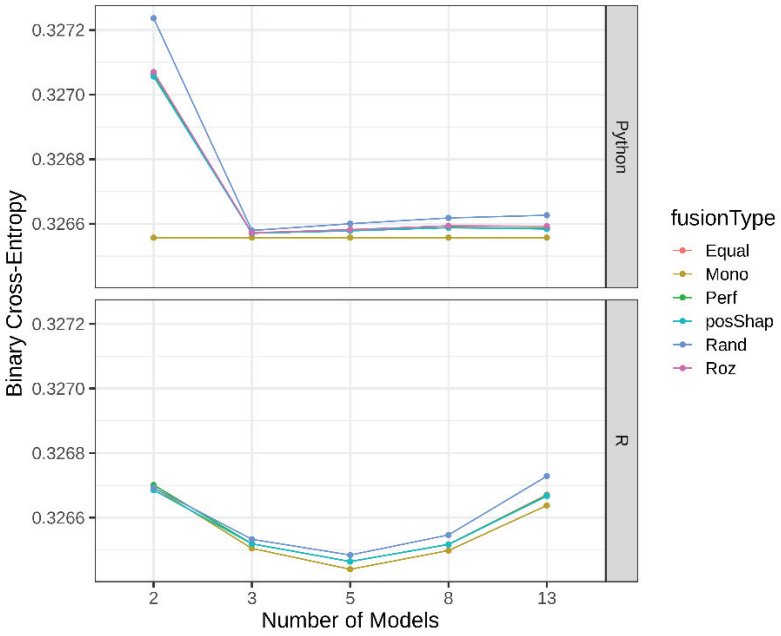


Figure 33. Performance comparison of homogeneous logistic regression ensembles developed by using BNG dataset

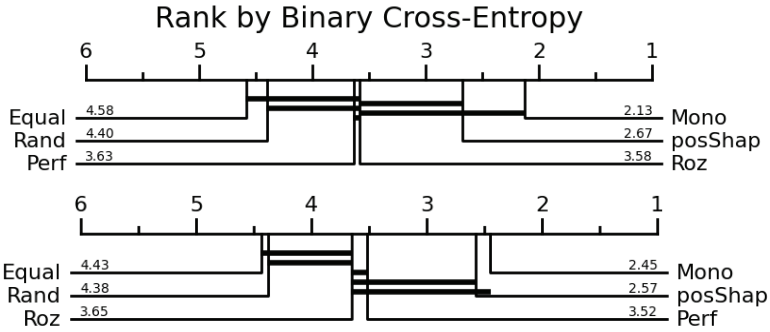


Figure 34. Ranking of homogeneous logistic regression ensembles developed by using BNG dataset for Python (top) and R (bottom) implementations

The single model monolith approach was surpassed in terms of performance by all the ensemble weighting strategies when homogenous decision tree classifiers were developed by using the BNG_credit-a dataset (**Figure 34**). Performance differences between the monolithic approach and the ensemble approaches increased with larger ensemble member counts. Compared to the posShap strategy with the highest performance, the difference ranged from 0% to 4.8%. When comparing the performance of only the ensemble weight selection strategies, the results on ensembles with 2 and 3 members revealed that there is no noticeable difference between the tested strategies. For ensembles with 5 members or more, the posShap strategy produced the best performance, resulting in 0.317 log-loss (BCE). The

performance difference is also evident in the rank-based comparison presented in (**Figure 35**) with the posShap strategy presenting the highest ranking. The Roz and Equal strategy results were statistically indifferent, as well as the Perf and Roz strategy results implemented in Python. The performance produced by the R language implementation was the highest of all the tested homogeneous configurations and implementations. When comparing the monolith approach and the largest ensemble size of 13 models, the Perf and posShap weighting strategies improved the performance by 4.1% and 4.8%, respectively.

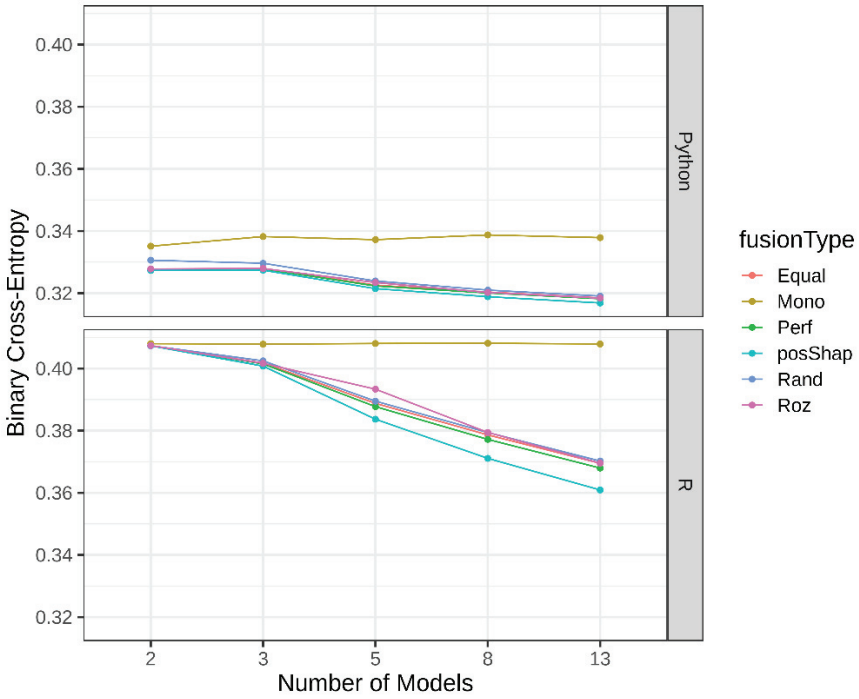


Figure 35. Performance comparison of homogeneous decision tree ensembles developed by using the BNG dataset

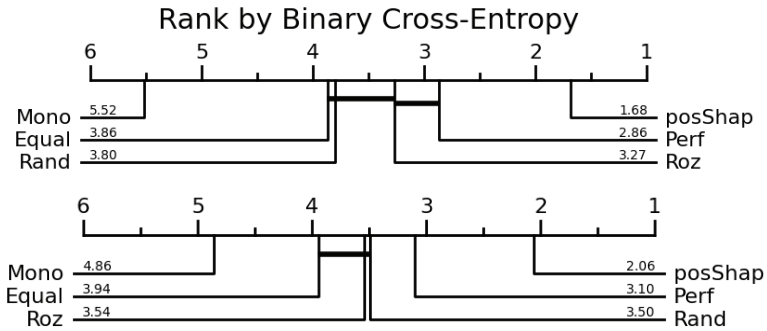


Figure 36. Ranking of homogeneous decision tree ensembles developed by using BNG dataset for Python (top) and R (bottom) implementations

The experiment results of logistic regression ensembles trained by using the Bank Marketing dataset (**Figure 37**) produced nearly identical results for ensemble sizes 2, 3 and 5 for both implementation languages. As the Bank Marketing dataset contained negative Shapley values, the maxShap strategy was applied. The performance of the maxShap value sharply decreased when the ensemble size reached 8 or more members. The distinction between the Mono and other ensemble creation strategies became apparent only when the ensemble member count reached 8 or more members. The performance difference remained small at only less than 0.1%. The highest performance of all the tested ensemble sizes and implementations by using logistic regression and the Bank Marketing dataset were a BCE value of 0.236. The ranking of the ensemble results is presented in **Figure 38**. The PosShap weighting strategy was ranked as the strategy with the best performance, while Perf produced the second-best results. Both the posShap and Perf strategies had higher ranks than the monolith approach, which indicated that the performance-based weighting produced better results for the model and dataset configuration. The resulting performance increase of 0.4% might be considered negligible, which is further highlighted by the lack of statistical significance between most of the strategies tested.

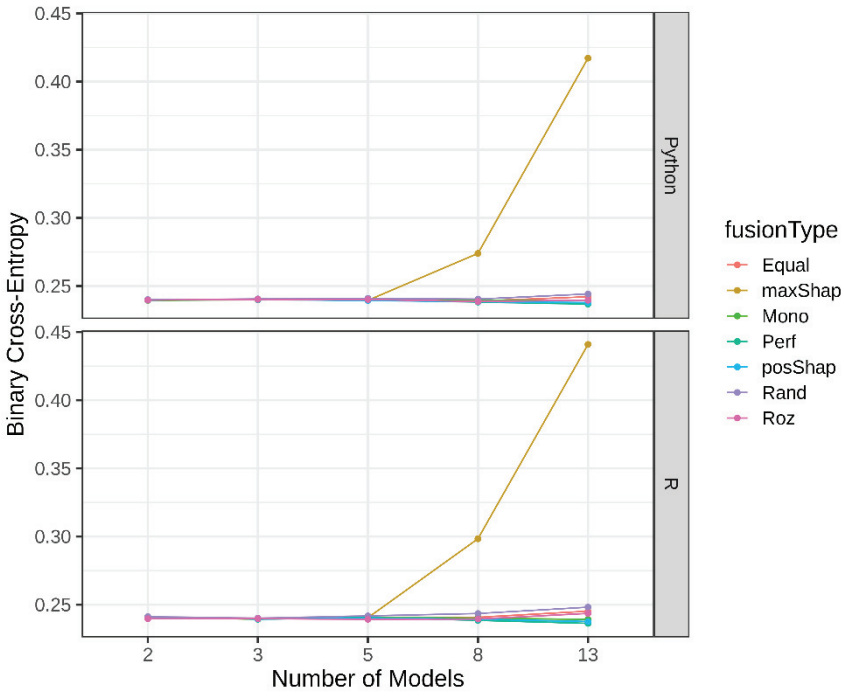


Figure 37. Performance comparison of homogeneous logistic regression ensembles developed by using Bank Marketing dataset

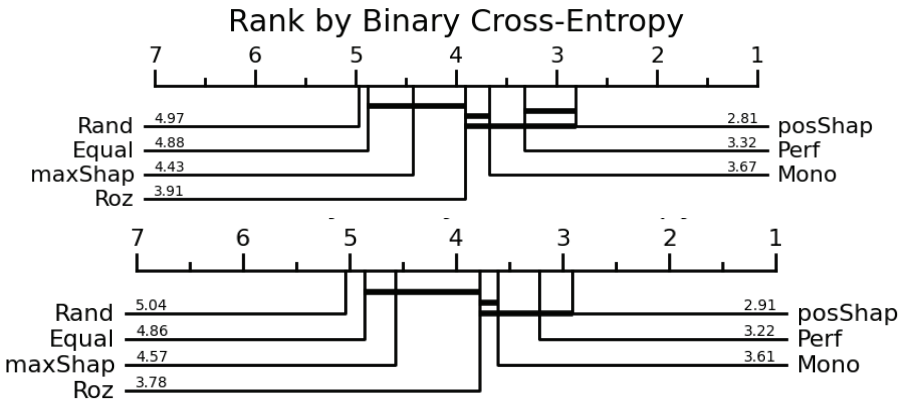


Figure 38. Ranking of homogeneous logistic regression ensembles developed by using the Bank Marketing dataset for Python (top) and R (bottom) implementations

An ensemble composed of decision tree classifiers trained on the Bank Marketing dataset results is presented in **Figure 39**. With the ensemble composed of 3 or more models, the log-loss (BCE) produced by the maxShap weighting strategy decreased and kept decreasing for higher ensemble model counts. The monolith approach was outperformed by all the ensemble weighting strategies in the Python implementation for ensemble sizes 2 to 8. For ensembles with 13 aggregated models, the posShap strategy presented the best performance, resulting in *BCE* of 0.264 for Python and 0.274 for R implementations. When using R implementation posShap, Equal and Perf weighting strategies produced lower BCE values than the monolith approach for every tested ensemble size. The best performance was reached by an ensemble composed of 8 models and resulted in *BCE* of 0.251. The rankings of the tested ensemble weighting strategies are presented in **Figure 40**. The highest-ranking approach was posShap when the ensemble was implemented in R and second-best for the Python implementation. The results presented by Roz and Random strategies show no statistical significance. A similar lack of statistical difference was also present between the posShap and Equal strategies when implemented in Python.

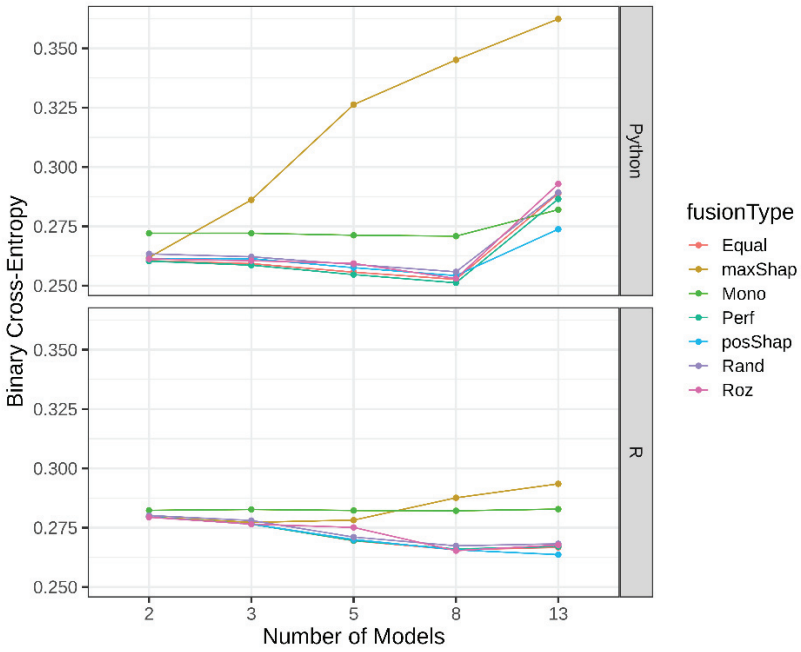


Figure 39. Performance comparison of homogeneous decision tree ensembles developed by using Bank Marketing dataset

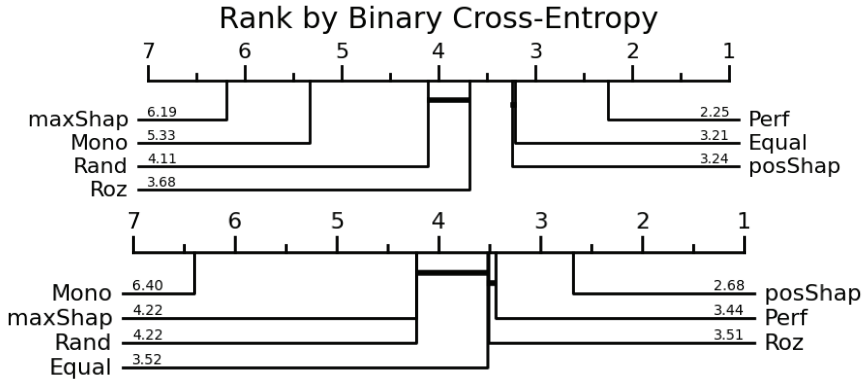


Figure 40. Ranking of homogeneous decision tree ensembles developed by using Bank Marketing dataset for Python (top) and R (bottom) implementations

4.2.3 Summary of results for homogeneous ensembles

The summary of the ranks for all implementations and datasets is provided in Table 14, and the results show that the decision tree classifier ensembles benefited more from the weighting strategies than the logistic regression classifier ensembles. The performance increase for decision tree-based ensembles when compared to the base monolith model was 1.9% for the Bank Marketing and 4.8% for the BNG_credit-

a datasets, the performance of the logistic regression results in minimal gains of 0.2% and 0.002%, respectively. Even though the weighting strategy was not as successful in increasing the performance for logistic regression classifier ensembles, the posShap strategy still performed similarly to the Perf strategy, which produced the best performance of all the tested strategies. The best performance for the Bank Marketing dataset resulted in BCE of 0.236, which was produced by the Perf strategy with the ensemble containing 13 members; with the same configuration, the posShap weighting strategy presented the result of BCE of 0.238. The results for the BNG_credit-a dataset was similar, where the posShap and Perf loss values were BCE of 0.317 and 0.318, respectively. The posShap weighting strategy was more effective in increasing the performance of the ensembles composed of a higher number of models. Contrary to the results of the posShap strategy, maxShap produced the worst results of all the tested approaches for ensemble sizes of 8 and 13. In all the tested data and implementation configurations, the posShap strategy surpassed or at least matched the performance of the Roz [180] strategy.

Table 14. Ranking position results from all the tested data and implementation configurations. Bold numbers denote the classifier with the highest rank

Model type	**Logistic regression**		**Decision tree**		**Logistic regression**		**Decision tree**	
Dataset	**BNG_credit-a dataset**				**Bank Marketing dataset**			
Implementation / Weighting strategy	Python	R	Python	R	Python	R	Python	R
Mono	**1**	**1**	6	6	3	3	6	7
Rand	5	5	4	3	7	7	5	5/6
Equal	6	6	5	5	6	6	2	4
Perf	4	3	2	2	2	2	**1**	2
Roz	3	4	3	4	4	4	4	3
MaxShap	-	-	-	-	5	5	7	5/6
PosShap	2	2	**1**	**1**	**1**	**1**	3	**1**
Model type	**Logistic regression**		**Decision tree**		**Logistic regression**		**Decision tree**	
Dataset	**BNG_credit-a dataset**				**Bank Marketing dataset**			
Implementation / Weighting strategy	Python	R	Python	R	Python	R	Python	R
Mono	**1**	**1**	6	6	3	3	6	7
Rand	5	5	4	3	7	7	5	5/6
Equal	6	6	5	5	6	6	2	4
Perf	4	3	2	2	2	2	**1**	2
Roz	3	4	3	4	4	4	4	3
MaxShap	-	-	-	-	5	5	7	5/6
PosShap	2	2	**1**	**1**	**1**	**1**	3	**1**

To test how the proposed method would work with ensembles composed of multiple classifier types, an experiment with heterogeneous ensembles was completed, and its results are presented in the next chapter.

4.2.4 Experiment results for heterogeneous ensembles

Results for heterogeneous ensembles which were developed by using the Python programming language and trained on the Bank Marketing dataset are presented in **Figure 41**. The ensemble with the highest performance utilised the equal weighting strategy and produced a BCE value of 0.233 for the highest ensemble size of 16 models. The lowest performance of all the tested weighted strategies was produced by the maxShap strategy. The maxShap strategy produced a negative impact on the performance of the model ensemble with a decrease in performance with higher ensemble sizes. The opposite result was presented by the posShap ensemble weight selection strategy, even though posShap performed similarly for ensembles with 4 and 6 models. In ensembles with 10 or more models, the performance of the posShap strategy was greater than that of the monolithic approaches. For the highest ensemble model count of 16 models, the posShap strategy also produced better performance than the Perf approach. Such an increase of performance indicates the ability of the Shapley values to measure the model contribution more precisely than merely the performance-based approach. This precision increase is enabled by evaluating the total existing permutations, evaluating the model combinations which would otherwise not be compared by the simple performance evaluation. The ensemble ranks are presented in **Figure 42**, and the results exhibit that the equal strategy outperformed the other tested approaches. The second-best performing strategy was Rand, with Perf and posShap taking ranks number three and four, respectively. The rank-based comparison also reveals that the differences between the posShap, Perf, and Roz strategies were statistically insignificant.

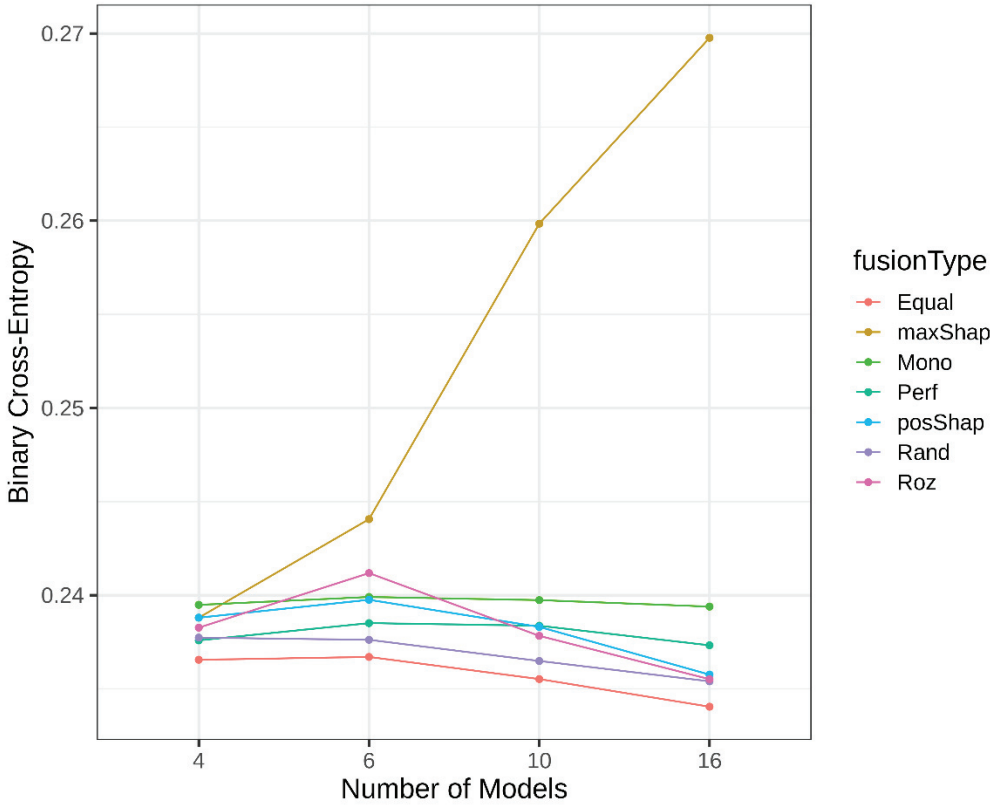


Figure 41. Heterogeneous ensemble performance comparison for a range of ensemble sizes developed by using Python implementation and Bank Marketing dataset

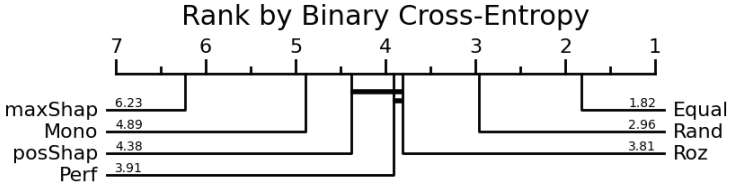


Figure 42. Rank-based comparison of heterogeneous ensemble weighting strategies for models developed in Python environment and by using Bank Marketing dataset

The heterogeneous weighted ensemble results developed by using the BNG-credit_a dataset and the Python implementation are presented in **Figure 43**. The presented results are similar to the results of the homogeneous ensembles (**Figure 35**) as both were produced by using the identical dataset. Log-loss (BCE) was further reduced with the introduction of logistic regression models and the application of weighting strategies. The best-performing weighting strategy was posShap with a median BCE of 0.312. Both the Perf and Equal strategies presented similar results. This is confirmed by the lack of statistical significance of the results presented in the critical difference diagram (**Figure 44**). Both the Perf and Equal results were also

statistically insignificant from the Roz results. The effect of weighting is clear, as all of the tested ensemble weight selection strategies outperformed the monolith model approach. For an ensemble composed of 16 models, the gain over the monolith approach was 1.3% for Equal, Perf, Rand strategies, and 1.4% for the posShap strategy.

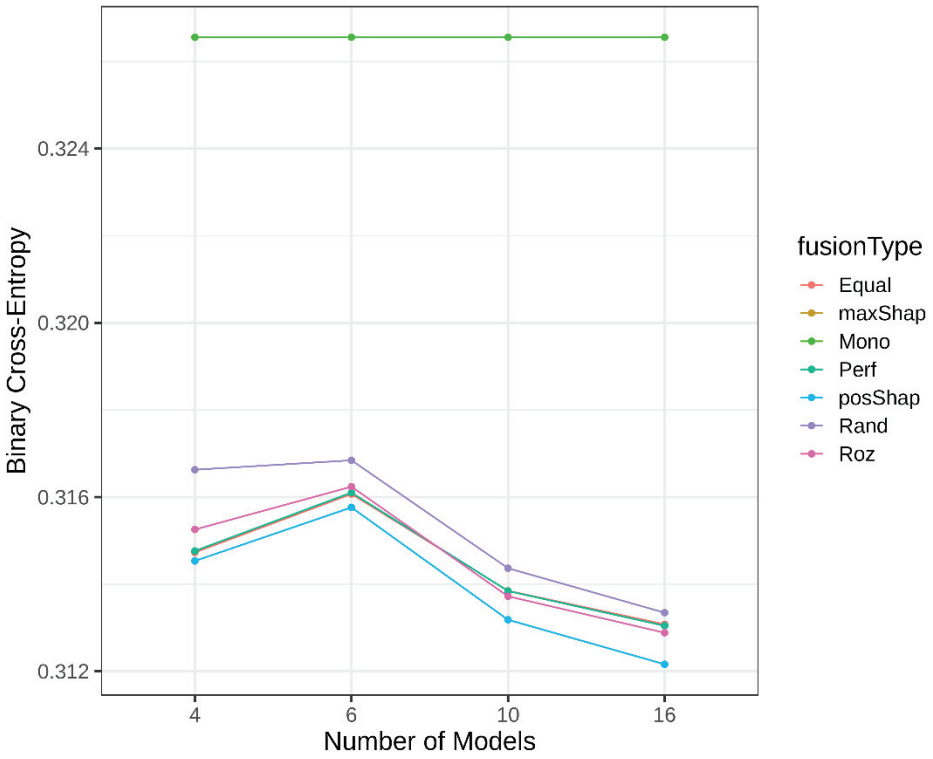


Figure 43. Heterogeneous ensemble performance comparison for a range of ensemble sizes developed by using Python implementation and BNG dataset

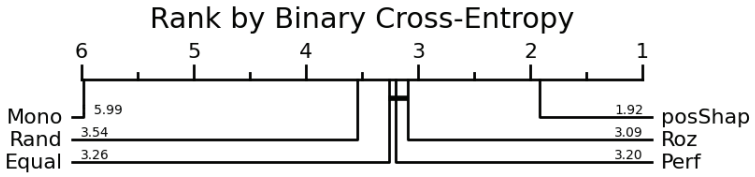


Figure 44. Rank-based comparison of heterogeneous ensemble weighting strategies for models developed in Python environment and by using BNG-credit_a dataset

The results for the R language implementation of heterogeneous ensembles for the Bank Marketing dataset are presented in **Figure 45**. The results indicate that the posShap strategy for ensembles with model counts of 10 and 16 presents better performance than the Mono single model approach. The weighting strategy with the best performance for all the tested configurations of ensembles was Equal with a BCE

value of 0.233. For ensembles with 10 or 16 models, the posShap weighting strategy presents better performance than the performance-based approach Perf, and it results in the same median loss as the random strategy with a BCE value of 0.235. Even though the ensemble weighting only results in the overall performance gains of 0.6% for all the tested weight selection strategies, except for maxShap, it still presents worse results than the monolithic approach. The ranking of the ensemble performance is presented in **Figure 46**, and it displays that, when the results are aggregated across all the sizes of ensembles, the Equal strategy exhibits the best performance. The posShap strategy was ranked similarly to the Mono approach with maxShap resulting in the lowest rank.

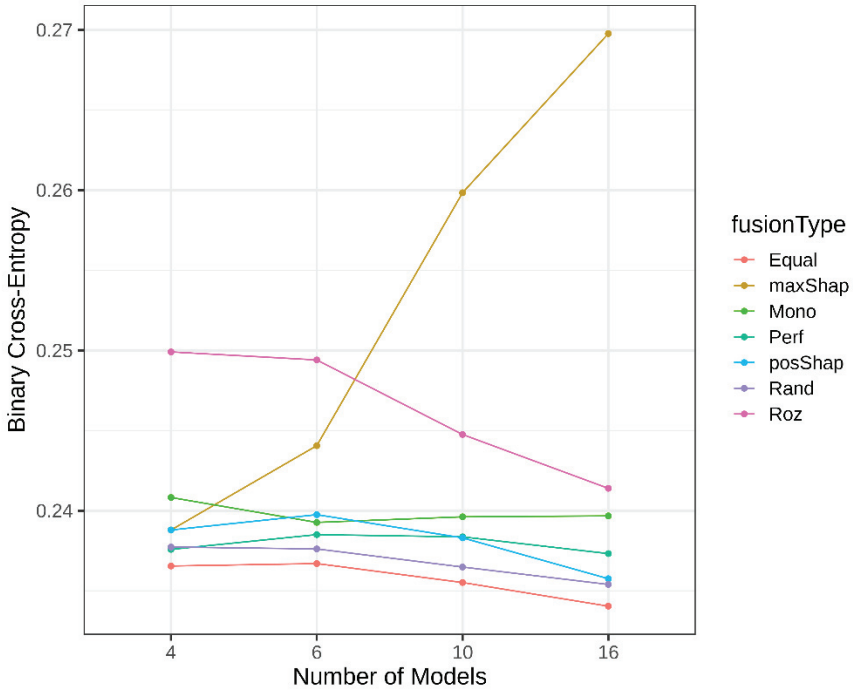


Figure 45. Comparison of heterogeneous ensemble performance for the range of ensemble sizes developed by using R implementation and BNG dataset

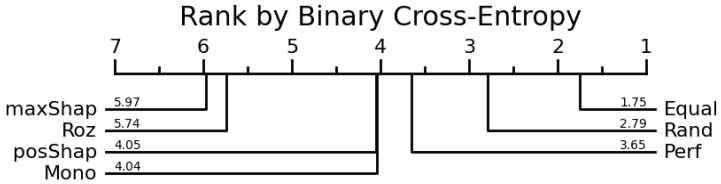


Figure 46. Rank-based comparison of heterogeneous ensemble weighting strategies for models developed in R environment and by using Bank Marketing dataset

The results of the R language implementation of heterogeneous ensembles developed by using the BNG-credit_a dataset are presented in Figure 47. The results

indicate that all the ensemble weighting techniques present a lower performance than the monolith approach. All the tested ensemble weighting approaches result in similar performance, with no statistical significance between the results. The best results for all the weighting strategies tested on any model size was reached by the Rand strategy with a *BCE* value of 0.339. The performance of the posShap strategy was reduced when the ensembles model count reached 16, but, for ensembles with 6 and 10 members, it presents the best performance. The ensemble ranking (Figure 48) shows that, between all the weighting approaches, no statistical significance was found. The monolith approach is the approach with the highest performance, with the Roz approach being ranked as the worst-performing one.

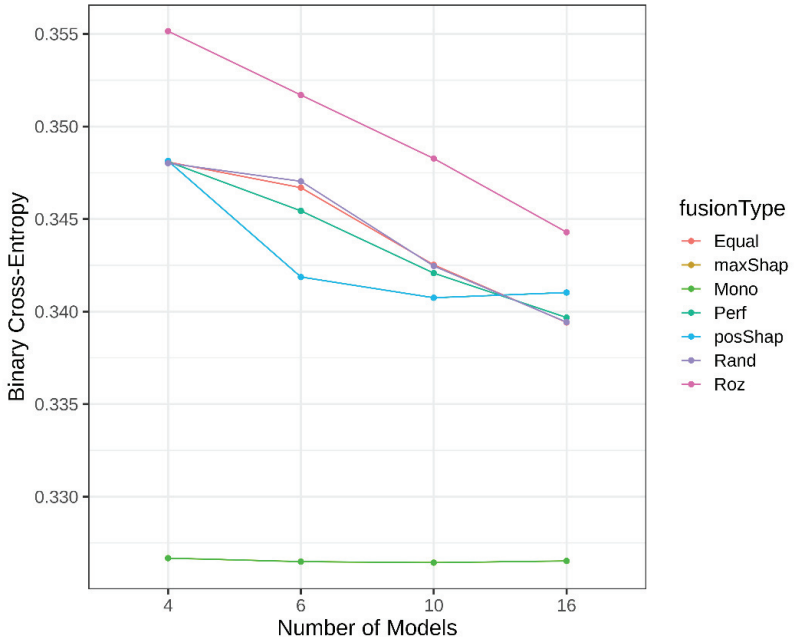


Figure 47. Heterogeneous ensemble performance comparison for a range of ensemble sizes developed by using R implementation and BNG dataset

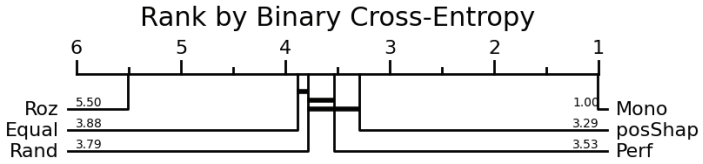


Figure 48. Rank-based comparison of heterogeneous ensemble weighting strategies for models developed in R environment and by using BNG dataset

4.2.5 Summary of results for heterogeneous ensembles

The summary of all ensemble ranks is presented in **Table 15**. The summary reveals that, for heterogeneous approaches, the selection of the correct model type is

as important as selecting the correct weighting strategy. As shown in **Figure 47**, the Mono approach outperformed all the tested weighting strategies. The performance difference between the Mono approach and the weighting strategies ranged from 1.3% to 2.2%.

Table 15. Ranking of the results of the Friedman test for the heterogeneous experiment. Bold numbers denote the classifier with the highest rank.

Dataset	**BNG_credit-a**		**Bank Marketing**	
Implementation / Weighting strategy	Python	R	Python	R
Mono	6	**1**	6	4
Rand	5	4	2	2
Equal	4	5	**1**	**1**
Perf	3	3	4	3
Roz	2	6	3	6
MaxShap	-	-	7	7
PosShap	**1**	2	5	5
Dataset	**BNG_credit-a**		**Bank Marketing**	
Implementation / Weighting strategy	Python	R	Python	R
Mono	6	**1**	6	4
Rand	5	4	2	2
Equal	4	5	**1**	**1**
Perf	3	3	4	3
Roz	2	6	3	6
MaxShap	-	-	7	7
PosShap	**1**	2	5	5

The introduced discrepancy of the performance results when comparing the Mono approach and other weighting strategies observed in two implementations for the BNG_credit-a dataset was influenced by implementation-specific hyperparameter values, as the weighting strategy performance was greater when the Python implementation was used (Figure 43), whereas the opposite was observed in the R language approach (Figure 47). The implementations were nearly identical when comparing Figure 41 and Figure 45. The results of these implementations were in the range of BCE from 0.233 to 0.269, and, in both implementations, the equal weight selection approach produced the best results. The combination of the model ensemble types increased the performance of three out of the four tested datasets and model type configurations. When compared to the highest performing ensemble in homogeneous experiments which presented a BCE value of 0.236, the heterogeneous ensembles increased the performance to BCE of 0.233. The Mono approach was surpassed in terms of performance by the ensemble weighting strategies in three out of the four tested dataset and model type configurations, thus denoting the performance benefits of the weighted ensembling approaches. The proposed posShap strategy outperformed

the Roz strategy in 3 tested heterogeneous experiment configurations, except for the Python implementation for the models developed by using the Bank Marketing dataset where both of the strategies presented identical results. The worst performance was achieved by the maxShap weighting strategy, which shows that assumptions about the prediction correction only worsen the ensemble performance. As more values were modified by the maxShap strategy in larger ensembles, the performance drawbacks became more evident.

4.2.6 Evaluation of Shapley calculation algorithm complexity

A server running *Intel Xeon Silver 4114* CPU consisting of 10 CPU cores and 2.20 GHz processor speed with 32 GB of RAM was used as the experiment environment. *Ubuntu 18.04* was used as the operating system with two model development environments: *R 4.1.3* and *Python 3.6.9*.

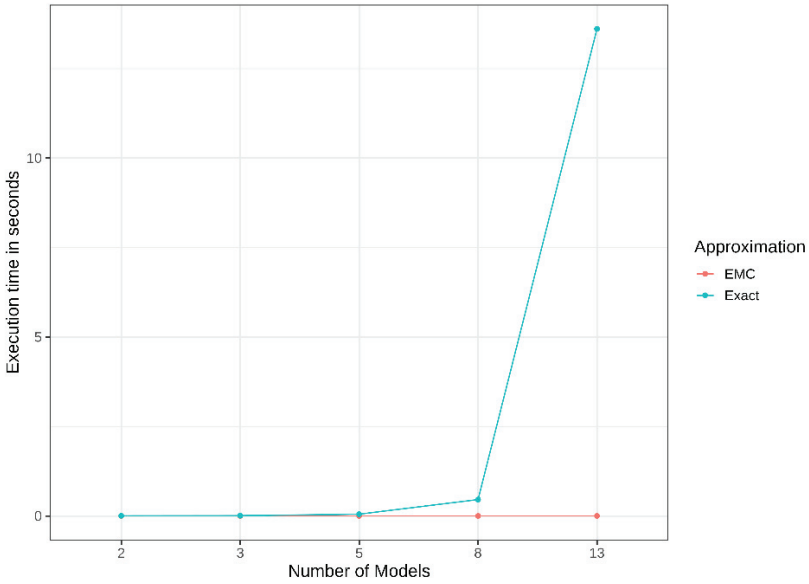


Figure 49. Shapley value calculation runtime comparison between the approximation method used by Roz [180] (EMC) and posShap (Exact) strategies

The algorithm complexity when calculating the Shapley values without any approximation measures is O(N!). The complexity of the expected marginal contributions (EMC) approximation algorithm [19] is O(N).

The Shapley value calculation time is presented in **Figure 49**. Even though the exact calculation time was nearly identical to the approximation approach for ensembles containing only 2–5 members, the computation time increased exponentially with larger sizes of the ensemble model. For ensembles sized 13, the average time for the exact Shapley calculation was 13.612 s, whereas, for the EMC approximation approach, the time was 0.002 s. The exponential increase in the Shapley calculation time could be resolved by using the currently existing

approximation algorithms [176], [222] which are similar to the one used in the Roz [180] approach, or by using model count reduction approaches before combining them into ensembles without considerably lowering the ensemble performance [199].

Nevertheless, if any approximation approach were applied, the posShap weight selection method would result in a simpler computation procedure as the method calculations would be performed with data combined into batches rather than on a single data point level as in source [180].

4.2.7 Summary of experiment results

The most prominent performance increase when compared to the monolith approach using homogeneous ensembles was achieved by the posShap weighting strategy on the ensemble which was composed of 13 models: 4.8% and 1.9% for BNG_credit-a and Bank Marketing datasets, respectively. Compared to the commonly used performance-based weight selection approach (Perf), the proposed posShap strategy increased the ensemble performance by 0.7%. The ensemble ranking revealed that the posShap strategy was ranked as the best strategy, except for a single configuration of the Python programming language, the decision tree classifier, and the Bank Marketing dataset. Similar, results for the posShap weight selection strategy were obtained in the heterogeneous ensembles, reaching a performance increase of 1.4% when compared to the Mono approach for the BNG_credit-a dataset setting. For the Bank Marketing dataset, the posShap produced gains of 0.4%, but the best performing weighting strategy was of equal weighting with an increase of performance of 0.6% when compared to the monolith approach.

Of the two proposed ensemble weighting strategies, only posShap produced positive results, as the performance of the maxShap strategy was surpassed by all the tested strategies. This reveals that the applied prediction correction methods did not improve the performance, and the exclusion of non-performing ensemble members was a more beneficial strategy. The performance of posShap varied based on the dataset and model type configurations, but the experiment results indicate that the posShap strategy surpassed or was at least similar when compared to other tested weighting strategies, including the Shapley vote-based strategy (Roz).

4.3 Performance Evaluation of Knowledge Distillation Approach Using Three Layer Perceptron

The experiment tested how the knowledge distillation approach impacts the performance of the classifier and whether it can it produce results comparable to the ensemble prediction. The distilled models were developed as a neural network model type by using blockchain ensemble predictions as the input. The goal of the knowledge distillation performance evaluation experiment was to measure the impact of model compression on the classification performance.

The experiment was conducted in two stages: the model preparation stage, and the knowledge distillation performance evaluation stage. The first stage used machine learning model predictions developed in the Shapley-based weight selection experiment as the training data for the distilled model. Only models developed by using the Python programming language were used in this experiment. The

experiment used a model ensemble containing 13 classifiers and utilised the posShap strategy to select their weights. The ensemble predictions and developed posShap weights were stored as files and used in the knowledge distillation algorithm.

The second stage consisted of neural network model development and its performance evaluation. The neural network model was developed by using weighted ensemble predictions produced on two already tested datasets: Bank Marketing and BNG_credit-a. The predictions produced from the ensemble model were used as the input to the student model. The neural network was trained iteratively with the amount of the iterations set being based on the parameters of the dataset in use.

The performance of the trained neural network models was evaluated by using the validation subset for each used dataset. To compare the performance of the distilled model, its performance was compared to the monolith single model approach and the posShap approach. The PosShap approach performance results indicate the performance of using the distilled model CDMLB methods approach step. The neural network model was developed with three different alfa parameter settings: 0.5, 0.75, and 1. The alfa parameter represents the loss ration that is calculated by using the teacher loss and the student loss, and it is used as a scaled distillation loss measure [98]. The alfa value of '1' means that the scaled loss only used the training loss of the trained neural network. Other alfa parameters denote different ratios of the neural network loss inclusion in the training process with 0.75 and 0.5 denoting 75% and 50%, respectively, of the used neural network (student) loss. The results were compared by using the Friedman ranking technique, and the classifier performance results were presented by using graphs to overview the differences in the performance distribution and median values.

4.3.1 Experiment settings

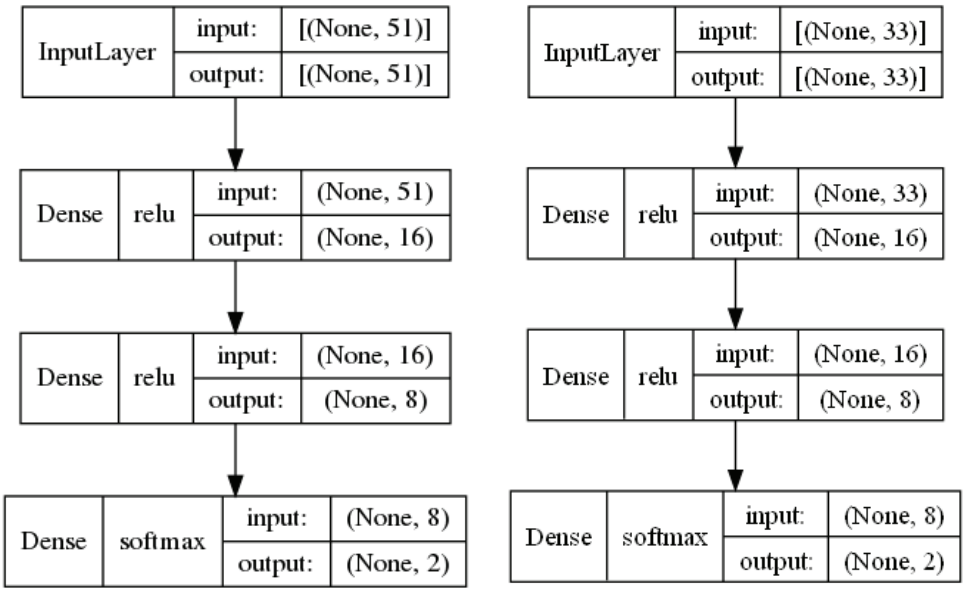
The models were developed by using the *PySpark* machine learning library version *3.1.2*. The model parameters were identical to those used in the Shapley weighting strategy performance evaluation experiment (cf. **Table 12**).

The neural network architectures are presented in **Figure 50**, with the main difference being in the configuration of the solution in terms of the number of input parameters for each network. The selected architectures were developed according to the best practices presented in source [223], with the layer count selection based on the described best practices. In the selected architectures, the first layer matched the feature set count in the respective dataset, and the following hidden layers diminished in terms of the neural network node count. The distillation model development parameters are presented in **Table 16**, where, due to the large dataset size BNG_credit-a, a lower number of training iterations and a higher training and validation batch size were used. The batch size of the validation split data was lower for the Bank Marketing dataset so that to preserve a larger amount of the training data due to the smaller amount of data instances. The experiment using the Bank Marketing dataset was computed by using 100 iterations, whereas BNG_credit-a used 50 iterations with the data split being based on a strategy defined in the Shapley-based ensemble weighting experiment presented in Section 4.2. The distillation was

implemented by using the *Keras 2.10.0* machine learning library [224], by modifying the example provided in source [210].

Table 16. Student model training parameters for the knowledge distillation approach

Parameter \ Dataset	Bank Marketing	BNG_credit-a
Dataset size	45211 × 51	1000000 × 33
Percentage of target class	0.12	0.54
Validation split	0.1	0.2
Batch size	512	4096
Number of epochs	100	200
Number of runs for BCE	100	50



(a) Bank Marketing dataset (b) BNG_credit-a dataset

Figure 50. Compact neural network architectures tested for the student model. Presented by using the keras plot_model function with the boxes representing neural network layers, with information about the used activation function and the number of neural network nodes in a layer

4.3.2 Experiment results

The experiment results for the distilled decision tree models, developed by using the Bank Marketing dataset are presented in
Figure 51. The results compare the performance between the monolith (Mono), the ensemble usage (posShap) approach proposed in Section 3.4.2 and the distilled model

with different alfa parameters (dist0.5, dist0.75, dist1). The alfa parameter indicates the amount of knowledge introduced into the loss function, with dist05 including an equal amount of the student (neural network)-to-teacher (ensemble knowledge). Meanwhile, dist1 indicates the baseline performance of the tested neural network classifier, without knowledge distillation. The results are presented by plotting the performance of the classifiers measured in BCE and the difference of the median values of the said classifiers. The baseline classifier for the differences was the Mono approach. The results indicate that the distilled model configurations performed worse than the Shap and Mono strategies when comparing the median performance values. Out of all the tested distilled model configurations, the dist0.75 model provided the best median value in BCE of 0.296. The Shap method was the best-performing approach with a BCE value of 0.253, while, in comparison, the distillation approach dist0.75 reduced the performance in BCE of 0.043.

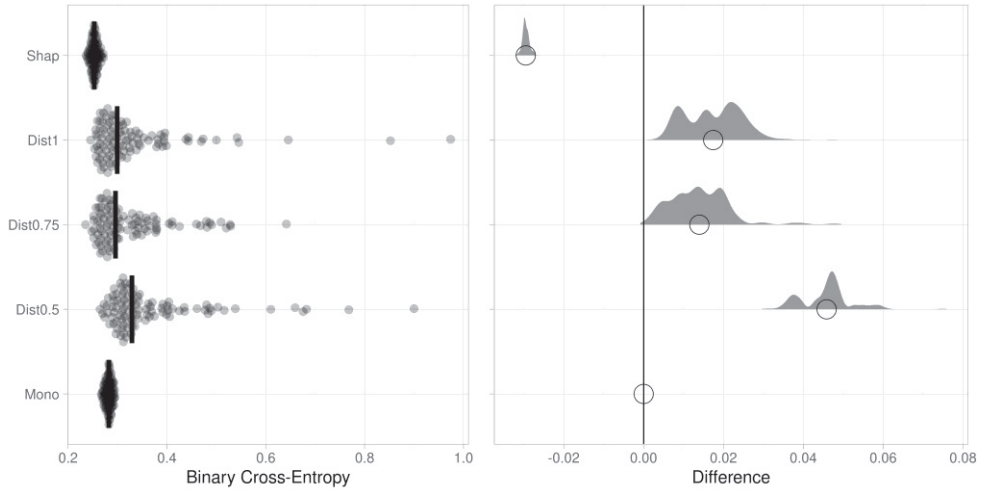


Figure 51. Prediction performance distribution of distilled decision tree classifier ensemble comparison with the posShap and Mono approaches. Bank Marketing dataset

The ensemble performance rankings (**Figure 52**) reveal that the performance of the dist05 approach yielded the lowest BCE of 0.329. The results between dist075 and dist1 were statistically insignificant. The posShap ensemble weighting strategy was the best-performing approach. The performance difference between the posShap and the Mono approaches was 16.99%.

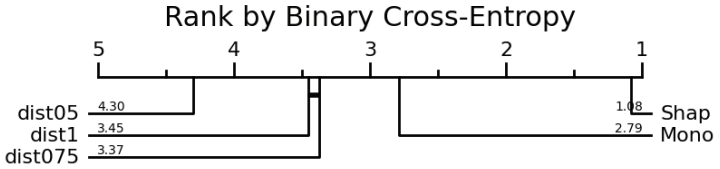


Figure 52. Performance ranking of distillation approaches and baseline models developed by using a decision tree classifier. Bank Marketing dataset

The performance results for logistic regression using the same Bank Marketing dataset (**Figure 53**) reveal that the performance loss for the distillation approaches was higher, with the median in BCE ranging from 0.294 to 0.32. The distillation approaches with the alfa values of 0.75 and 1 performed similarly with the median in BCE of 0.294 and 0.295, respectively. The results of dist0.75 are more clustered around the median when compared to dist1, which might indicate that the inclusion of predictions allows developing a more focused model.

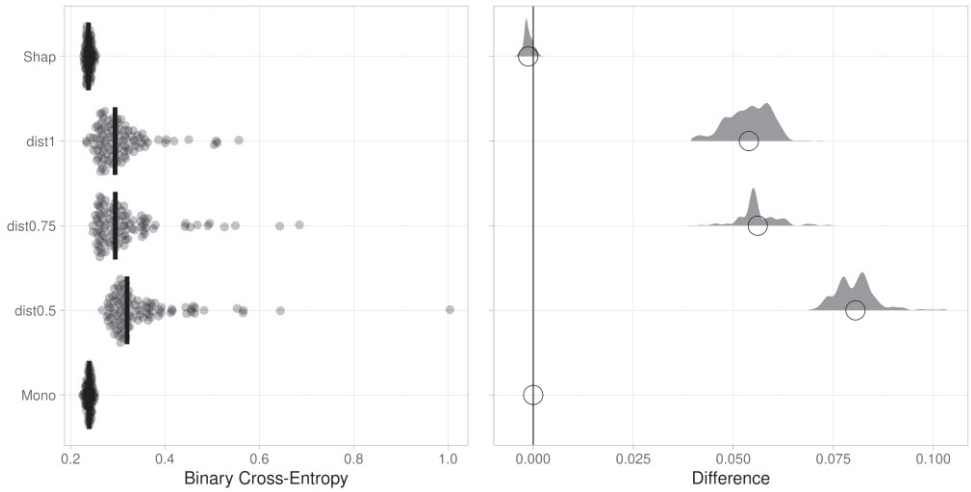


Figure 53. Prediction performance distribution of distilled logistic regression classifier ensemble comparison with the posShap and Mono approaches. Bank Marketing dataset

The classifier ranks for the approaches built by using logistic regression resulted (**Figure 54**) in a larger difference between the best-performing distilled model and the best-performing approach of BCE equal to 0.054. As the performance benefits introduced with the Shapley-based weighting were less evident in ensembles containing logistic regression, the Mono and posShap strategy results are statistically indifferent. The difference between the results of dist075 and dist1 was also statistically insignificant, which means that the introduction of the student model trained by using the predictions of the network does not significantly reduce the performance. The best-performing distillation strategy with an alfa value of 1 produced results in BCE of 0.294.

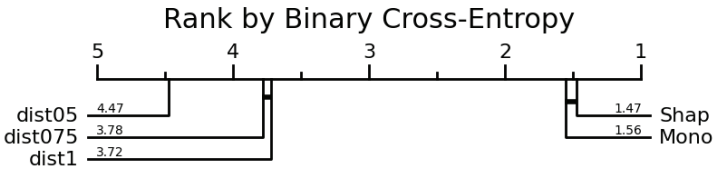


Figure 54. Performance ranking of distillation approaches and baseline models developed by using a logistic regression classifier. Bank Marketing dataset

The knowledge distillation for the BNG_credit-a dataset results (**Figure 55**) displayed a smaller performance reduction when compared to the Bank Marketing dataset with the median value of the dist1 and dist0.75 results of 0.350 and 0.344 presented in BCE, respectively. The lowest BCE was produced by using the dist0.5 approach, which resulted in a median value of 0.390. The posShap value representing the ensemble usage without knowledge distillation presented a median value in BCE of 0.317.

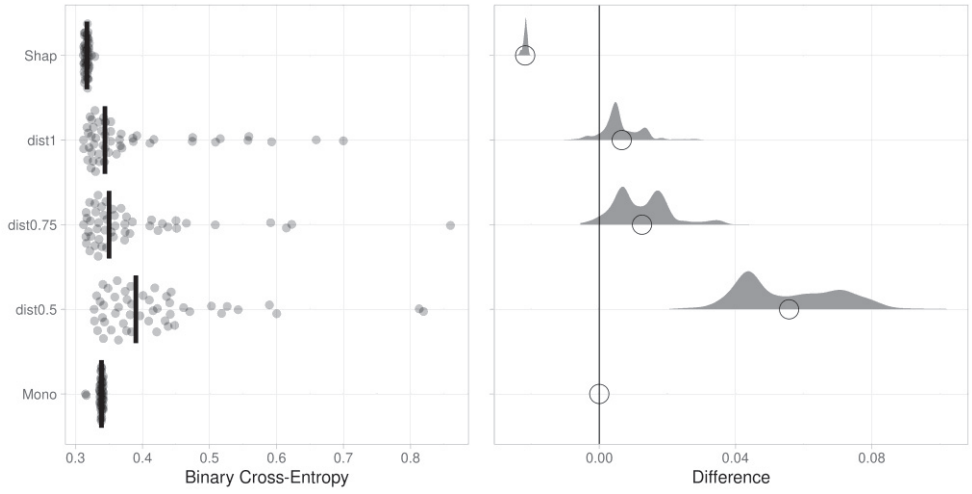


Figure 55. Prediction performance distribution of the distilled ensemble of decision tree classifiers comparison with the posShap and Mono approaches. BNG_credit-a dataset case

The rank comparison of the tested knowledge distillation and ensemble creation approaches when using the BNG_credit-a dataset and the decision tree classifier is presented in **Figure 56**. The posShap and Monolith approaches were ranked as the best and the second-best performing approaches. Similarly to other model type and dataset configurations, the differences between the results produced by dist075 and dist1 were statistically insignificant. The approach with the worst performance was the dist05 approach with a median BCE of 0.390.

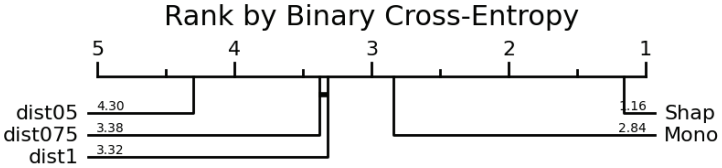


Figure 56. Performance ranking of distillation approaches and baseline models developed by using a decision tree classifier. BNG_credit-a dataset

For the same BNG_credit-a dataset, the distillation results of the logistic regression models (**Figure 57**) resulted in performance loss for all the tested distillation configurations compared to the decision tree classifier type. The median values for the best-performing distillation strategy dist1 and the second-best strategy dist075 presented in BCE were 0.366 and 0.378, respectively. In this configuration,

the Shap approach and the Mono approach presented similar results in BCE of 0.32647 and 0.32650.

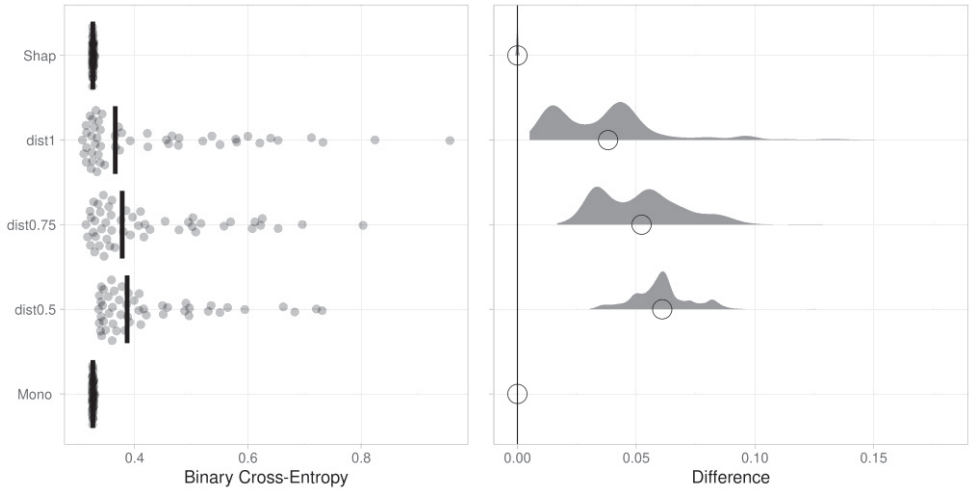


Figure 57. Prediction performance distribution of the distilled logistic regression classifier ensemble comparison with the posShap and Mono approaches. BNG_credit-a dataset case

The ranking comparison of the distillation approaches and the weighting strategies is presented in **Figure 58**. The rankings reveal that the results of the dist05 and dist1 distillation strategies did not show statistical significance, which means that the increase in the amount of the model prediction did not reduce the performance as much as in the other datasets and the model type configurations tested. Similarly, no statistical significance was discovered between the Mono and the Shapley approaches. Nevertheless, the distillation process reduces the performance levels, as the best-performing distillation strategy reduced the performance by BCE of 0.04.

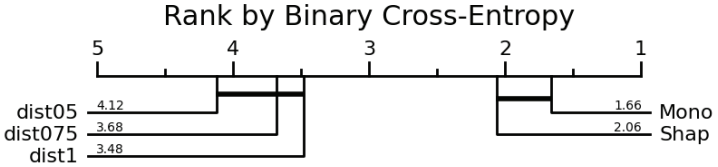


Figure 58. Performance ranking of distillation approaches and baseline models developed by using a logistic regression classifier. BNG_credit-a dataset

A summary of the results obtained in all the tested experiment configurations is presented in

Table **17** as the median values presented in BCE. The result comparison indicates that the best overall performance was achieved by the logistic regression model developed by using the bank marketing dataset and utilising the mono approach – with a BCE of 0.24. The best performing knowledge distillation strategy for the Bank Marketing dataset was achieved by the logistic regression model type and the dist1 approach – with a BCE of 0.294. In the BNG_credit-a dataset, the decision tree classifier type produced the best performance of BCE 0.344 by using dist1.

Table 17. Comparison of the median BCE results for knowledge distillation experiments. The values in bold indicate the BCE value of the best-performing classifier.

Dataset	Bank Marketing		BNG-credit-a	
Model	**Logistic regression**	**Decision tree**	**Logistic regression**	**Decision tree**
Mono	0.240	0.282	**0.326**	0.339
Shap	**0.238**	**0.253**	**0.326**	**0.317**
Dist0.5	0.320	0.329	0.387	0.390
Dist0.75	0.295	0.296	0.378	0.350
Dist1	0.294	0.299	0.366	0.344

4.3.3 Summary of experimental results

The experiment results demonstrated that the knowledge distillation approach reduces the performance of the ensemble classifier by at least 16.99% for the Bank Marketing dataset, and at least 10.41% for the BNG_credit-a dataset when comparing the posShap strategy to the dist0.75 approach. The dist0.75 approach was selected for a comparison because it retained a similar accuracy to the ensemble model while including the training approach which was chosen to comparison. The comparison of the alfa parameters revealed that the inclusion of ensemble predictions in the loss function in a higher ratio (dist05) to have a balanced variant of distillation resulted in the worst performance among all the strategies tested. The differences between the BCE medians of the distillation strategies dist075 and dist1 were statistically insignificant in 3 out of 4 experimental configurations. The similarities between the results display that a minor increase of the network ensemble prediction does not dramatically change the classifier performance. Even though the knowledge distillation reduces the performance with respect to the ensemble, it should still improve privacy according to source [106] and provide a pre-trained neural network model for further fine-tuning with individual participant data, if need be. If no further tuning is required, the usage of the Shapley-weighted ensemble for inference provides a higher accuracy for the model usage in the proposed CDMLB method.

4.4 Performance Evaluation of Knowledge Distillation with the Deep Learning Model

To complete the evaluation of knowledge distillation, an approach of the experiments with deep learning model architectures was conducted. Instead of developing a novel neural network architecture, we selected an already existing deep neural network architecture *TabNet* [211]. The TabNet architecture was selected based on the displayed performance on the *Bank Marketing* dataset [211], as well as the capability to perform binary classification tasks with high performance levels. Alternatively, any deep learning architecture that is designed to work with tabular data and can be used in the distillation process, for example, as shown in sources [225], [226]. The selected neural network architecture was developed for machine learning tasks employing tabular data. The TabNet architecture consisted of an encoder with a feature transformer, an attentive transformer, and feature masking. The implemented

deep neural network architecture contained 4 layers, where two of the layers were shared across the decision steps, whereas two of the layers were decision-dependent. Each out of the 4 layers contained a full connection layer, while proceeding with a normalisation layer, and finished with a Generalised linear unit nonlinear activation layer. A split block divides the processed representation for the attentive transformer of the subsequent step and the overall output. The feature selection mask offers interpretability, and aggregated masks provide the global feature importance. The decoder includes a feature transformer block at each step. An attentive transformer block example uses the prior scale information, whereas sparsemax serves for coefficient normalisation, which results in the sparse selection of the salient features. The implemented architecture is described in more detail in Article [211].

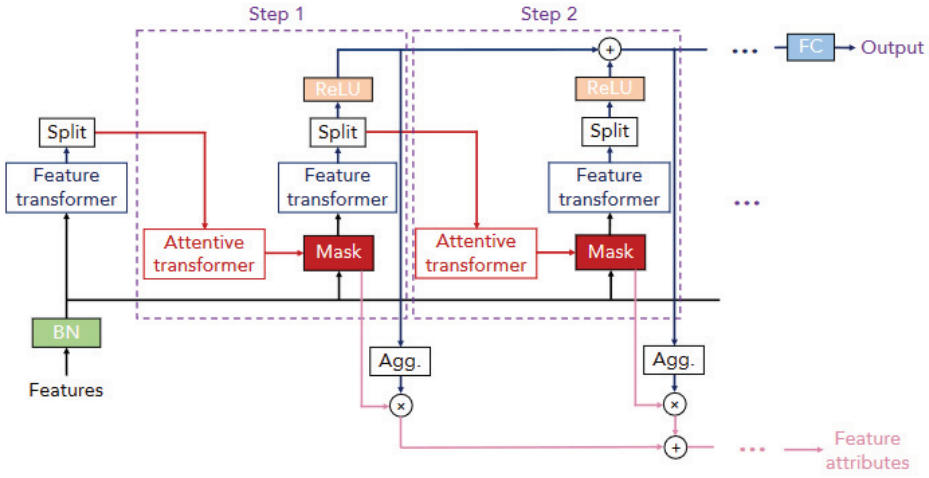


Figure 59. TabNet encoder architecture [211]

The experiment execution procedure followed the exact structure of the experiment defined in Section 4.3 Performance Evaluation of Knowledge Distillation Approach. The experiment with the deep learning model used the same base model predictions and ensemble posShap weights as the input for knowledge distillation. The model ensembles used in distillation contained 13 classifiers. The provided predictions and ensemble weights were provided to the neural network training procedure in batches.

The deep neural network training algorithm was developed by using the example provided in source [212] while adding the scaled distillation loss function introduced by source [98]. The network training parameters are introduced in **Table 18**. The dataset parameters remained the same as in the shallow neural network experiment, and the number of runs for the BNG_credit-a dataset increased from 50 to 100 iterations. Due to a lower instance count in the Bank Marketing dataset, the batch size and the feature transformation dimensionality were lower. The developed

neural network training process used the *Adam* [227] algorithm in its training optimisation.

Table 18. Deep learning model training parameters for tested datasets

Parameter \ Dataset	Bank Marketing	BNG_credit-a
Dataset size	45211 × 51	1000 000 × 33
Percentage of target class	0.12	0.54
Validation split	0.2	0.2
Batch size	256	1024
Feature transformation dimensionality (feature_dim)	64	128
Epochs	100	200
Learning rate	0,02	0,02
Decay rate	0.95	0.95
Decay every	50	50
Number of runs	100	100

4.4.1 Experiment results

The performance results of the distilled decision tree classifier into a deep learning model, developed by using the Bank Marketing dataset (**Figure 60**), provided insights that distillation improved the performance over the posShap method in both dist075 and dist1 cases. The increased performance of the dist1 case where neural network models were trained without including pre-trained model distillation demonstrated that the deep learning model was more successful than the ensembling or the monolithic approach. This indicates that the deep learning model is the most capable machine learning model type for this dataset, from all the tested model and ensemble configurations producing a median *BCE* result of 0.245. By including knowledge from the posShap decision tree ensemble, the performance increased even further to a median *BCE* result of 0.234. The distillation process when compared to Shap increased the performance of the classifier by 3.95% for dist075 configuration and decreased the performance by only 2.77% for the dist05 configuration. Such results indicate that, based on the selected distillation method, the losses could be minimal, or the performance would not decrease at all for the selected dataset.

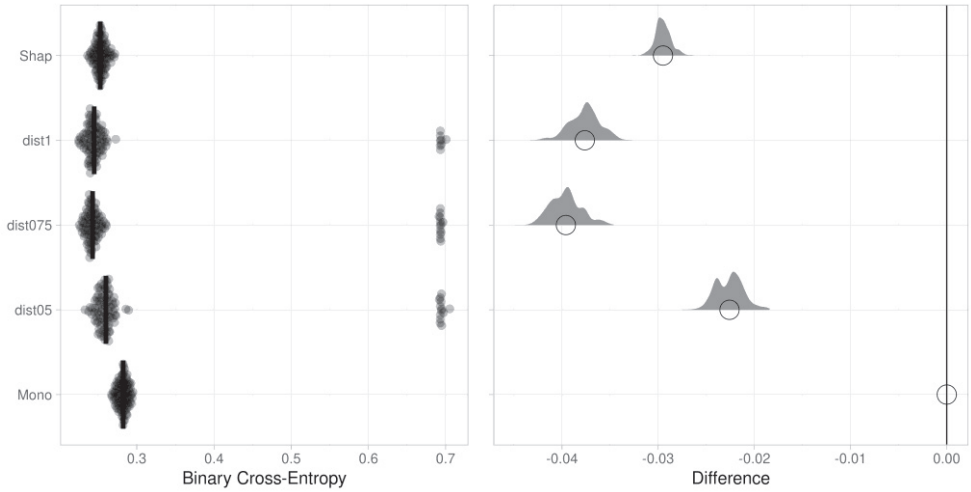


Figure 60. Performance distribution for distilled decision tree classifiers developed by using the Bank Marketing dataset. The main results clusters contained: dist1 – 92%, dist075 – 87%, dist05 – 87% out of the total results

The performance ranking of the deep learning classifiers for the Bank Marketing and decision tree classifier configuration (**Figure 60**) revealed that the dist075 approach offered the highest overall performance. The dist1 configuration was the second best, with results not statistically different from the dist075 configuration. The median decrease in BCE when compared to Shap for dist075 and dist1 configurations was 0.008 and 0.01, respectively. The dist05 configuration reduced the BCE value of the classifier by 0.007. All the tested distillation and ensemble development strategies surpassed the performance of a single model (the Mono approach).

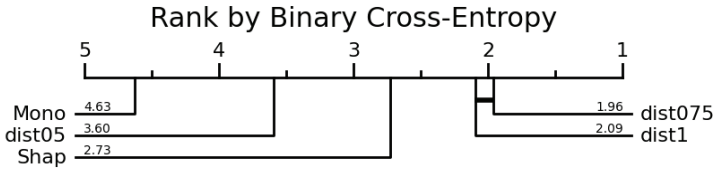


Figure 61. Ranking of the classifier performance based on the used distillation solutions Bank Marketing and decision tree base model case

The deep learning model developed by using logistic regression as its base distillation model for the same Bank Marketing dataset presented different results (Figure 62). All the tested distillation strategies, including the dist1 strategy, did not surpass the Shap and Mono classifiers in terms of performance. Similarly to the decision tree classifier case when comparing distillation configurations, dist075 presented the best performance, while dist1 was the second-best performing classifier. The highest decrease of performance over Shap was observed in the dist05 classifier of 9.2%, while the decrease was only 4.2% and 2.52% for dist1 and dist075, respectively. Such results indicate that the deep learning model was more fit to

knowledge distillation because it reduced the performance loss form at least 16% to a maximum of 9.2% for the Bank Marketing dataset.

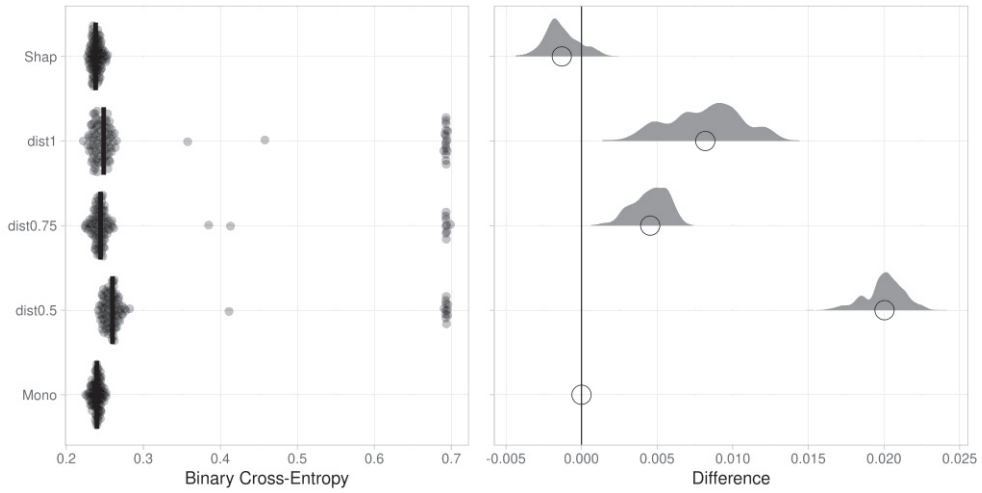


Figure 62. Performance distribution for distilled logistic regression classifiers developed by using the Bank marketing dataset. The main results clusters contained: dist1 – 84%, dist075 – 89%, dist05 – 88% out of the total results

The performance ranking results (**Figure 63**) are similar to the rankings of the tested shallow neural network architecture (**Figure 53**). The Mono and Shap classifiers produced statistically indifferent results with a median BCE value of 0.24 and 0.238, respectively. The median BCE reduction for dist075 and dist1 was 0.006 and 0.010. The performance distribution and ranking reveals that the logistic regression classifier performance results decreased after distillation.

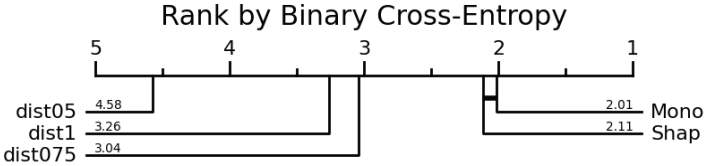


Figure 63. Ranking of the classifier performance based on the used distillation amount. Bank Marketing and logistic regression base model case

The results of the distilled decision tree models in a deep neural network case of the BNG_credi-a dataset are presented in **Figure 64**. The performance results clustered around three main points. Such clustering makes the interpretation of the results more challenging, but the performance trends which are present in the Bank Marketing dataset, such as dist075 presenting the best results with dist1 resulting in the second best and dist05 with the worst results are clear as well. The median increase of BCE for dist075 over the Shap classifier was 4.42%, while the median dist1 value

reduced the performance by 45%, even though around one third of the results for the dist1 classifier were similar to the dist075 classifier results.

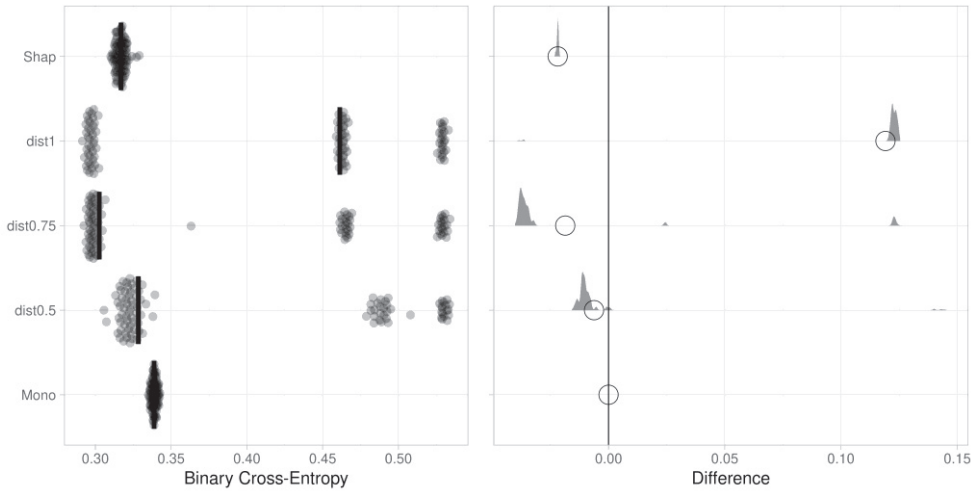


Figure 64. Performance distribution for distilled decision tree classifiers developed by using the BNG_credit-a dataset. The result cluster containing the highest performance contained: dist1 – 41%, dist075 – 57%, dist05 – 60% out of the total results

Even though the performance distribution displays a performance increase in dist075 over the Shap classifier, the performance ranking (**Figure 65**) indicates the Shap classifier as the best-performing with a median BCE of 0.317, even though the median results indicate that dist075 had the highest performance with a BCE value of 0.303. The dist075 method's overall rank is reduced by the outlier cases. The ranking test also indicates that the differences between the results of dist05, dist1, and dist075 cases were statistically insignificant, which means that the performance differences were minimal. When comparing the performance differences in terms of BCE, the median value of dist075 outperformed the Shap classifier by only -0.025, while the dist05 performance was only lower by a BCE value of 0.01. Such results indicate similarities between the median values, thus denoting that the performance loss or performance gain was minor.

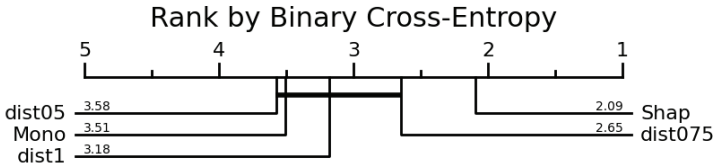


Figure 65. Ranking of the classifier performance based on the used distillation amount BNG_credit-a and the decision tree base model case

The results of the distilled logistic regression model for the BNG_credit-a dataset case are presented in **Figure 66**. Distilled logistic regression models also produced results clustered into three distinct clusters. Even though the part of the

dist05 and dist075 classifier results surpassed the Mono and Shap classifier performance, the median performance of the distilled classifiers was lower by 1.53% and 42%, respectively. Surprisingly, the median value had a higher performance over the Shap classifier by 7.975%. Similarly to the Bank Marketing dataset when comparing the logistic regression model type results to the decision tree classifier results, distillation is not as effective in preserving the classifier performance.

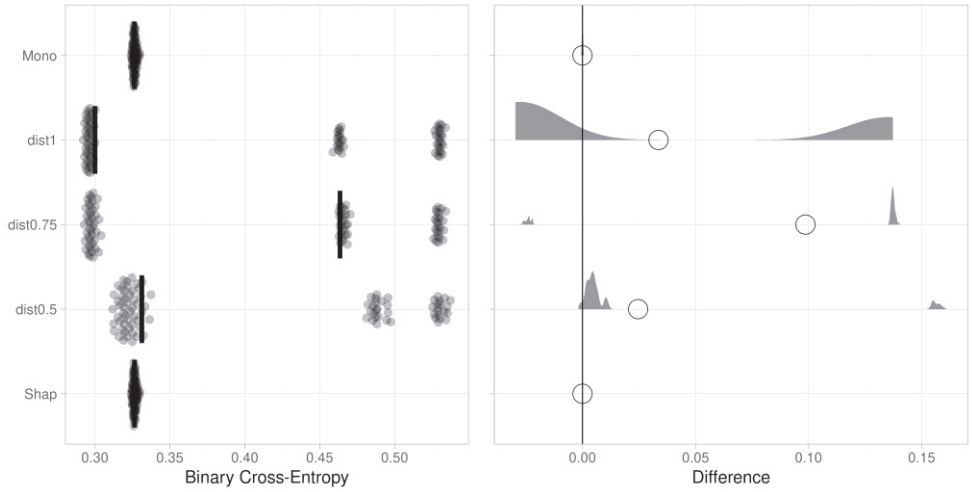


Figure 66. Performance distribution for distilled logistic regression classifiers developed by using the BNG_credit-a dataset. The result cluster containing the highest performance contained: dist1 – 66%, dist075 – 61% dist05 – 70% out of the total results

The logistic regression classifier ranking (**Figure 67**) reveals that the performance levels of all classifiers were similar. The dist1 classifier rank was similar to the Mono approach, and no statistical significance was found between all the tested distillation classifier results. This indicates that even tough Mono was the best classifier, distillation successfully produced results similar to the Shap ensemble. The median results for the Shap ensemble were 0.326, while dist05 and dist1 produced BCE values of 0.331 and 0.300, respectively.

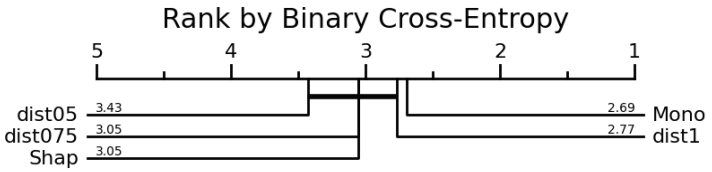


Figure 67. Ranking of the classifier performance based on the used distillation amount BNG_credit-a and the logistic regression base model case

4.4.2 Summary of experimental results

The experiment results of ensemble knowledge distillation into a classifier powered by a deep learning architecture demonstrated that the usage of such an architecture decreases the performance loss. When comparing results of a shallow neural network to deep learning for the dist075 classifier while using the Bank Marketing dataset, one can observe a performance increase from 16.99% to a performance loss of only 3.95% with part of the results, which indicates no performance loss. When comparing the distillation classifier performance the in BNG_credit-a dataset case, the differences between the results were statistically similar. The distillation performance for dist075 on the BNG_credit-a dataset ranged from a decrease of 45% to an increase of 4.41%. Such a huge margin between the results was introduced by the performance outliers, even though at least 40% to 70% of the test runs even surpassed or presented similar values to those of the Shap classifier performance. By using the deep learning neural network architecture, even the usage of the median distillation (dist05) classifier becomes viable with its performance loss ranging from 9.24% to 1.53%.

The results indicate that a model with a more complex inner model representation, such as the decision tree model type, is more suitable for distillation, as, in such a configuration, only a single classifier performed worse than the Mono classifier. It is clear that the deep learning architecture performed better than the shallow neural network architecture. This indicates that the knowledge distillation method should be used in combination with deep learning architectures to preserve ensemble performance while transferring the model from the blockchain network to other application areas.

4.5 Answers to Research Questions

The experimental results provided the following answers to the defined research questions (as presented in Section *Problem statement and research questions*):

RQ1: The distributed machine learning transparency has been improved by implementing the process using the private blockchain network and enabling network participants to read a shared ledger, which enables their ability to audit transactions, submitted models, data, and inference results. The collaboration for distributed learning has been improved by allowing the network participants to reuse the already existing machine learning solutions with a minor modification on the blockchain network, thus increasing the possibility for collaboration. Contribution evaluation has also improved the collaboration by providing means to quantify the quality of the shared models and data.

RQ2: Blockchain technologies could facilitate the collaborative distributed machine learning process by allowing the participants to deploy machine learning models and validation data via the smart contract automation and automated performance evaluation. The proposed contribution evaluation could potentially increase the motivation of the participants.

RQ3: The blockchain network should contain dedicated smart contracts which implement the collaboration process. The blockchain network technologies should also be modified to include support for the currently existing machine learning

solutions for their easier integration into the collaborative distributed machine learning environment. This can be achieved by using local blockchain oracle services which enable support for the common programming environments in blockchain networks.

RQ4: The model contributions could be evaluated by combining shared models into an ensemble and evaluating the performance of all possible variants of an ensemble by using the Shapley weight calculation strategy. Data contributions could be evaluated by measuring the drop in the ensemble performance due to new data inclusion into the shared data pool.

RQ5: The training data privacy can be improved by separating model training and model deployment environments and only sharing the already trained models to the blockchain network, which facilitates the distributed machine learning process. To further increase the privacy of the training data, an individual model is obfuscated through ensembling and further distilling with the student-teacher approach.

4.6 Threats to Validity

Threats to the performance evaluation of the model inference calculation via the local off-chain blockchain oracles experiment can be outlined as the usage of a virtual environment and the limited size of the tested network. The architecture of the local off-chain oracles was tested only in a virtual environment. This negated any latency which would be introduced into the physical system and could have affected the results of the experiment. The performance limitations were also imposed by resource sharing while performing the consensus algorithm and data distribution over network peer nodes. The highest number of blockchain network nodes tested due to the limited computational resources was 16, even though multiple network configurations were tested to evaluate the performance effect of additional network nodes. An increased number of network nodes could affect the performance of the consensus algorithm protocol as well as the speed of data replication. In order to reduce the chances of failed experiments due to the limited performance, most of the experiments were conducted in a blockchain network containing 13 peers. Such a network configuration was chosen based on the available CPU and RAM resources required to run containerised blockchain services in a virtualised environment. In addition, the experiment evaluation did not include stress testing and limitation analysis of the existing data storage approach in the selected blockchain network solution and was only using up to 20000 data instances at any given time. The performance evaluation and the scalability of the proposed system should be evaluated in the further research.

The performance of the components could have been negatively impacted as the virtual components shared the computational resources, whereas, in a physical system, the resources would be individual. The increased network size could also increase the amount of communication required to reach consensus and introduce additional latency. Although, the results of the experiment presented a gradual increase in the network participant count and tested many different configurations with the maximum configuration adhering to the known limitations.

The Shapley-based ensemble weighting performance evaluation experiment involved the following threats: a small number of the tested datasets, lack of model

tuning, single data distributed for ensemble model training, and a limited Shapley value calculation speed.

A limited number of datasets have been utilised to evaluate the performance of the ensembles developed. The experiment results could depend on the size of the dataset and the extent of the class imbalance, even though the two datasets employed were different with respect to these aspects. Bank marketing provided insight into how the proposed method would perform on medium-sized datasets with strong class imbalance. Meanwhile, the BNG_credit-a dataset displayed how the method would perform on large datasets with balanced classes.

The model training procedure did not include the model parameter tuning step of the machine learning pipeline. The model tuning step could further improve the performance of the machine learning model, but, as the model with the same parameters was being used in the experiments, the results should not be affected. The Shapley-based ensemble weighting performance evaluation experiment utilised a data splitting strategy based on the Zipf law distribution. The data distribution might affect the performance of the ensemble classifier and the effectiveness of the weighting strategies.

The selected Shapley exact calculation approach due to its computational complexity would limit the number of the model that could be combined into an ensemble. To reduce the exponential growth of Shapley value calculations, approximation measures should be introduced, which might introduce deviations to the results, thereby decreasing their performance.

Threats to performance evaluation of the knowledge distillation experiment can be outlined as lack of the neural network model parameter tuning. The experiments only tested a single neural network architecture without evaluating a different range of development parameters. The neural networks with different architectures and different model development parameters might affect the performance of the distillation approach. The neural network was only tested by using binary classification tasks.

5. CONCLUSIONS

1. The analysis of distributed machine learning approaches and methods has demonstrated that, currently, most of the proposed architectures rely on centralised components, which reduces the robustness of the architecture and requires trust among the collaborating parties. Furthermore, most of the proposed distributed learning approaches are highly specialised, dedicated to a single machine learning problem, and they usually employ a single machine learning model type, which reduces possibilities and their engagement in collaboration.
2. Analysis of the applicability of blockchain technologies for distributed machine learning purposes has indicated that most of the known approaches use blockchain to engage the network participants, as well as to store ownership and transfer information via a shared ledger. The main limitations involved are that the solutions are highly specialised. This limitation can be alleviated by the introduction of oracle services in distributed machine learning based on the blockchain technology.
3. A method for collaborative privacy-preserving distributed machine learning has been proposed, thus enabling the blockchain network participants to collaborate via the model deployment process. The proposed method quantifies the contributions of the participants and enables the usage of knowledge accumulated on the blockchain network via the weighted ensemble or the knowledge distillation approaches. Proof-of-concept implementation has been developed by using the Hyperledger Fabric private blockchain network architecture, thus demonstrating the feasibility of collaborative distributed machine learning. The architecture features local oracle-based components which enable the usage of the currently existing machine learning environments, thus eliminating the need to rely on the limited set of machine learning development environments supported by blockchain technologies.
4. The method has been experimentally evaluated by benchmarking the performance of the binary classification models in the model inference task in two approaches – an approach developed only by using smart contracts, and by combining smart contracts with blockchain oracles. The results of the experiment have demonstrated that, on average, the network slowed by 2.07% when using two datasets with the logistic regression classifier. The introduction of the oracle components increased the flexibility to use common machine learning algorithm implementations in the blockchain network.
5. The performance evaluation of the ensemble weighting strategy suggests that the contribution of each participant could be quantified and used as a basis for constructing an incentive mechanism for the motivation of the participant. The

experiment results have revealed that a Shapley-weighted ensemble increased the performance by 4.8% and 1.9% for the two tested datasets compared to using a single large model, and by 0.7% compared to a simple performance-based weighting in the form of the reciprocal of binary cross-entropy. This strategy can be used for contribution evaluation and also for obtaining weights for a simple decision-level fusion in ensemble learning with similar success or even better than other weighting strategies compared. The introduced Shapley-based strategy can also be seen as a generalisation of the performance-based weighting.

6. The results of the knowledge distillation experiment with a three-layer perceptron have proven that the application of such an approach did not present any significant improvement over directly combining the network models into an ensemble. The performance decreased from 10.41% to 23.9% with respect to the tested classifier and the used dataset. In contrast, the results of knowledge distillation into the deep learning neural network demonstrated that, for the student model, a more complex architecture outperforms a simple one and better preserves the teacher model (ensemble) knowledge. Notably, the distilled TabNet classifier (student) even surpassed the performance of a weighted ensemble (teacher) to some degree. Weak distillation (alpha=0.75) of the logistic regression ensemble deteriorated the student by 3.95% for the Bank Marketing and by 42% for the BNG-credit_a datasets, whereas the distillation of the decision tree ensemble deteriorated the student by 2.52% for the Bank Marketing and improved the student by 7.98% for the BNG-credit_a datasets. The performance results of separate runs for the Bank Marketing were distributed normally, while the results for the BNG-credit_a dataset tend to cluster at three different levels, noticeably, superior performance over the teacher was achieved for 57% and 61% of runs for the logistic regression and decision tree ensembles, respectively. Knowledge distillation was found to be beneficial, and more complex model architectures are recommended for the student model.

6. SANTRAUKA

6.1 ĮVADAS

Pasaulyje sukuriama vis daugiau informacijos, o norint prasmingai panaudoti tokį informacijos kiekį, būtina naudoti automatizuotus duomenų apdorojimo sprendimus. Vienas iš būdų apdoroti didelius duomenų kiekius yra mašininio mokymosi metodai. Mašininio mokymosi metodai [10], [11], [12] sėkmingai taikomi daugelyje sričių, tokių kaip vaizdų atpažinimas, medicina, sprendimų analizė, rekomendacinės sistemos, kalbos technologijos, bei kitur, kur taikoma atpažinimo teorija. Dėl populiarėjančio mašininio mokymosi naudotojai gali kurti tokius pat mašininio mokymosi sprendimus, tačiau nepasiekia tokios kokybės mašininio mokymosi modelio, kokios galėtų pasiekti turėdami daugiau ir įvairesnių duomenų. Šiai problemai spręsti galėtų pasitarnauti bendradarbiavimas visuose mašininio mokymosi sprendimų kūrimo etapuose: didesni ir įvairesni duomenų rinkiniai galėtų būti sukurti panaudojant skirtingus šaltinius; mašininio mokymosi modeliai galėtų būti išmokomi panaudojant mažiau skaičiavimo išteklių ir būti geresnės kokybės; pasidalinti modeliai galėtų būti pakartotinai naudojami sumažinant poreikį kurti naujus lokalius specializuotus modelius. Deja, šiuo metu mašininio mokymosi modelių kūrimas dažnai vykdomas vienoje centralizuotoje modelio kūrimo aplinkoje, panaudojant tik ribotą duomenų kiekį.

Mašininio mokymosi paskirstymas galėtų padėti spręsti šias problemas, tačiau dėl poreikio dalintis duomenims ir bendradarbiauti kuriant mašininio mokymosi sprendimus gali kilti saugumo ir privatumo užtikrinimo problemų. Naudojama bendradarbiavimo aplinka privalo neatskleisti privačios ir jautrios informacijos apie esybes ir asmenis. Duomenų perdavimo kanalai turi būti apsaugoti ir užtikrinamas pasitikėjimas naudojamomis sistemomis ir technologijomis. Šiuo metu egzistuoja daug paskirstyto mašininio mokymosi sprendimų, leidžiančių vykdyti privatumą užtikrinantį bendradarbiavimą. Paskirstyto mokymosi sprendimai turėtų įgalinti bendradarbiavimą ir pagerinti mašininio mokymosi sprendimų kokybę, tačiau šie sprendimai vis tiek yra neatsparūs atakoms ir turi pasitikėjimo problemų. Taip pat paskirstyto mokymosi proceso dalyviai gali būti nepakankamai motyvuoti dalyvauti ir dėl to pasitraukti iš proceso.

Siekiant spręsti pasitikėjimo, skaidrumo ir patikimumo problemas, gali būti taikomos paskirstytųjų duomenų technologijos (angl. *Distributed Ledger Technology*). Paskirstytųjų duomenų technologijos leidžia sistemos naudojimosi metu registruoti transakcijas į duomenų žurnalą (angl. *ledger*) ir taip padidinti pasitikėjimą tarp tinklo dalyvių ir paslaugų. Tarp paskirstytųjų duomenų technologijų blokų grandinės technologijos (angl. *Blockchain technologies*) yra populiariausios ir plačiausiai taikomos. Blokų grandinės technologijose naudojamas duomenų replikavimas leidžia kurti sistemas, atsparesnes tinklo veiklą siekiančioms sutrikdyti atakoms. Tarpusavio pasitikėjimo problemą blokų grandinės technologijos sprendžia suteikdamos galimybę naudotojams laisvai peržiūrėti transakcijų informaciją ir kitus saugomus duomenis. Kiekviena transakcija, atliekama blokų grandinėje, yra patvirtinama kelių tinklo naudotojų ir užregistruojama, taip sumažinant galimybę

kenkėjiškai veiklai tarp tinklo narių. Sudėtingesnė veiklos logika blokų grandinėse gali būti realizuota naudojant išmaniuosius kontraktus (angl. *smart contracts*). Išmanieji kontraktai taip pat gali būti papildomi specializuotomis paslaugomis, kurios leistų paskirstyti, pakartotinai naudoti ir integruoti jau sukurtus mašininio mokymosi sprendimus. Šiuo metu nėra nusistovėjusių metodų, kurie leistų vykdyti bendradarbiavimą paskirstytose mašininio mokymosi sprendimuose, sukurtuose naudojant blokų grandinės technologijas, kurios užtikrintų duomenų privatumą ir leistų pakartotinai naudoti jau sukurtas mašininio mokymosi technologijas.

Tyrimo sritis ir objektas

Šios disertacijos tyrimo objektas yra bendradarbiavimas vykdant paskirstytą mašininį mokymą. Tyrimo sritis susideda iš dviejų pagrindinių sričių:

1) bendradarbiavimu grįstu paskirstyto mašininio mokymosi metodų ir architektūrų;

2) blokų grandinės naudojimo būdų ir įrankių, skirtų mašininiam mokymuisi vykdyti.

Spendžiama problema ir tyrimo klausimai

Bendradarbiavimui skirti paskirstyto mašininio mokymosi sprendimai yra ribojami mažo dalyvių pasitikėjimo, apribojimų dėl jautrių duomenų naudojimo ir sudėtingų egzistuojančių mašininio mokymosi technologijų adaptavimo galimybių. Norint pasiūlyti metodą šioms problemoms spręsti, šios disertacijos metu buvo iškelti šie tyrimo klausimai:

1. Ar paskirstyto mašininio mokymosi proceso skaidrumas ir bendradarbiavimas gali būti patobulintas? Jei taip, kokiu būdu?
2. Kaip blokų grandinės technologija gali būti pritaikyta palaikyti bendradarbiavimui skirtus sprendimus paskirstytam mašininiam mokymuisi vykdyti?
3. Ar privatumo užtikrinimas gali būti patobulintas vykdant blokų grandinės technologijomis grindžiamą paskirstytą mašininį mokymąsi?
4. Kaip galima pamatuoti blokų grandinės tinklo nario duomenų ir modelio indėlį vykdant bendradarbiavimu grindžiamą paskirstytą mašininį mokymąsi?
5. Kaip galima patobulinti mokymo duomenų privatumo užtikrinimą vykdant paskirstytą mašininį mokymąsi blokų grandinėje?

Tyrimo tikslas ir uždaviniai

Šios disertacijos tikslas – pagerinti bendradarbiavimą vykdant paskirstytą mašininį mokymąsi panaudojant blokų grandinės technologijas

Šiam tikslui pasiekti buvo išsikelti šie uždaviniai:

1. Išanalizuoti mašininį mokymąsi, paskirstytą mašininį mokymąsi ir bendradarbiavimo būdus vykdant paskirstytą mašininį mokymąsi.
2. Išanalizuoti blokų grandinės technologijas ir galimybes jas pritaikyti paskirstytam mašininiam mokymuisi vykdyti.

3. Pasiūlyti bendradarbiavimu grindžiamą paskirstyto mašininio mokymosi metodą, naudojantį blokų grandinės technologijas.
4. Realizuoti sprendimą, įgalinantį bendradarbiavimu grindžiamą paskirstytą mašininį mokymąsi panaudojant blokų grandinės technologijas pagal pasiūlytą metodą.
5. Ištirti, kaip blokų grandinės technologijų taikymas paveikia paskirstytą mašininį mokymąsi.
6. Įvertinti pasiūlyto metodo tinkamumą vykdyti bendradarbiavimu grindžiamą paskirstytą mašininį mokymąsi blokų grandine.

Tyrimo metodika

Tyrimas buvo vykdomas konstruktyvaus tyrimo metodu [13]. Remiantis šiuo metodu, tyrimas buvo vykdomas atliekant tokius žingsnius:

- Pirmajame žingsnyje buvo apibrėžtas tyrimo tikslas ir objektas, kuris apėmė bendradarbiavimą užtikrinančio paskirstyto mašininio mokymosi metodus, jų architektūras ir taikymą blokų grandinėse bei blokų grandinės technologijomis grindžiamų įrankių ir sprendimų vertinimą. Galiausiai buvo apibrėžta tyrimo problema, vyraujanti privatumą užtikrinančiame bendradarbiavimu grįstame paskirstyto mašininio mokymosi procese.
- Antrasis žingsnis buvo skirtas apibrėžti tyrimo potencialui, įvertinant, kaip blokų grandinės technologijos gali būti taikomos spendžiant bendradarbiavimo ir bendradarbiavimo užtikrinimo problemas vykdant paskirstytą mašininį mokymąsi.
- Trečiajame žingsnyje buvo išanalizuota apibrėžta tyrimo problema. Tam atlikti buvo pasitelktas lyginamosios analizės metodas, peržvelgiant esamus sprendimus, jungiančius privatumo užtikrinimo metodus ir blokų grandinės technologijas, paskirstyto mašininio mokymo kontekste.
- Ketvirtajame žingsnyje buvo pasiūlytas metodas, skirtas bendradarbiavimu grindžiamam paskirstytam mašininiam mokymuisi, kuris taiko privačios blokų grandinės technologijas ir leidžia įvertinti dalyvių indėlių vertę.
- Penktasis žingsnis buvo skirtas realizacijai ir eksperimentiniam vertinimui bei metodo taikymo sričių paieškos galimybių tyrimui. Buvo realizuotas blokų grandinės technologijomis grindžiamas sprendimas bendradarbiavimu grindžiamam paskirstytam mašininiam mokymuisi vykdyti, remiantis pasiūlytu metodu. Sprendimo veikimas buvo įvertintas eksperimentiškai, pamatuojant sprendimo greitaveiką ir modelių kokybę (angl. *model performance*). Kokybės vertinimas buvo atliktas sprendžiant dvi su bankais susijusias klasifikavimo problemas.

Ginamieji teiginiai

1. Egzistuojančios privačios blokų grandinės gali būti patobulintos specializuotomis lokaliomis orakulų paslaugomis, leidžiančiomis palaikyti įvairesnes mašininio mokymosi aplinkas.
2. Tinklo dalyvių modelių indėliai, pateikiami blokų grandinės tinkle, skirtame bendradarbiavimu grįstam paskirstytam mašininiam mokymuisi vykdyti, gali būti įvertinti kiekvienam tinklo nariui, panaudojant *Shapley* kolektyvo svorių apskaičiavimo strategiją.
3. Žinių distiliavimas gali būti panaudotas tinke sukauptoms žinioms agreguoti iš modelių kolektyvo į vieną neuroninio tinklo modelį, smarkiau nesumažinant jo tikslumo.

Mokslinis naujumas

1. Pasiūlytas bendradarbiavimu grįstas paskirstytas mašininis mokymosi blokų grandinėje metodas (CDMLB) išplečia esamas paskirstyto mašininio mokymosi, grindžiamo blokų grandinės technologijomis, galimybes, papildant sistemos architektūrą mašininio mokymosi modelių rezultatų skaičiavimo paslauga.
2. Pasiūlytas CDMLB metodas naudoja *Shapley* reikšmėmis ir modelio tikslumu grįstą kolektyvo svorių apskaičiavimo strategiją kaip būdą pamatuoti tinklo nario pateikiamų modelių indėlį į bendrą tinkle sukauptą modelių kolektyvą.
3. Pasiūlytas CDMLB metodas naudoja mokinio ir mokytojo modelių distiliavimo sprendimą, kuris leidžia padidinti modelių privatumą, suspaudžiant modelius, kaupiamus blokų grandinės technologija grindžiamuose sprendimuose.

Praktinė reikšmė

1. Pasiūlytas CDMLB metodas leidžia sujungti dažnai naudojamas mašininio mokymosi technologijas su blokų grandinių technologijomis, panaudojant lokalias, už blokų grandinės tinklo ribų esančias orakulo paslaugas.
2. Pasiūlyta kolektyvo svorių apskaičiavimo strategija gali būti traktuojama kaip tikslumu grįsto svorių apskaičiavimo generalizacija ir gali būti taikoma visiems kolektyvams, naudojantiems svorius.
3. Pasiūlyta kolektyvo svorių apskaičiavimo strategija padidina išbandytos binarinės klasifikavimo užduoties rezultatų tikslumą, kai naudojami lentelės tipo duomenys, palyginti su centralizuotu sprendimu ar kitomis svorių apskaičiavimo strategijomis.

4. Pateikiami modelio panaudojimo scenarijai leidžia užtikrinti modelių informacijos privatumą, išgaunant suspaustą modelį iš blokų grandinės technologija grindžiamo sprendimo tolesniam naudojimui ar tobulinimui.
5. Pristatytas modelių sujungimo metodas leidžia sujungti heterogeninius modelių tipus, taip padidinant naudojamų modelių įvairovę ir įgalinant didesnę aibę bendradarbiavimo galimybių.

Rezultatų aprobavimas

Tyrimo rezultatai buvo paskelbti 5 moksliniuose leidiniuose: dvi publikacijos periodiniame moksliniame žurnale *MDPI Applied Sciences* ir trys publikacijos konferencijų leidiniuose.

Disertacijos struktūra

Disertacijos dokumento pirmajame skyriuje pateikiami tiriamosios analizės rezultatai, apibūdinantys mokslines ir taikomąsias žinias apie paskirstyto mašininio mokymosi ir privatumo užtikrinimo metodus, mašininio mokymosi, realizuoto panaudojant blokų grandinės technologijas, sprendimus. Taip pat skyriuje pateikiama lyginamoji esamų sprendimų, jungiančių blokų grandinės technologijas su privatumą užtikrinančiais metodais ir paskirstyto mašininio mokymosi metodais, analizė. Antrasis skyrius apibrėžia bendradarbiavimu grįsto paskirstyto mašininio mokymosi blokų grandinėje metodą ir aprašo reikalavimus ir procedūras blokų grandinės sistemos paruošimui, modelių ir duomenų pateikimui ir tinkle sukauptų žinių panaudojimui. Kiekvienas metodo žingsnis pateikiamas detaliu aprašu ir realizacijos pristatymu. Trečiasis skyrius pristato atliktų eksperimentinių tyrimų konfigūracijas ir rezultatus. Ketvirtajame skyriuje pateikiamos disertacijos išvados. Disertacijoje taip pat pateikiama disertacijos santrauka lietuvių kalba, šaltinių sąrašas ir mokslinių publikacijų ir konferencijų sąrašas.

6.2 EGZISTUOJANČIŲ METODŲ IR SPRENDIMŲ ANALIZĖ

6.2.1 Mašininis mokymasis

Mašininio mokymosi modelio kūrimo procesą [14] sudaro keturi pagrindiniai etapai: duomenų apdorojimas, modelio derinimas, modelio kokybės vertinimas bei modelio diegimas ir naudojimas.

Duomenų apdorojimo etapas susideda iš kelių žingsnių, kurie yra: duomenų išgavimas, duomenų paruošimas ir požymių tyrimas. Duomenų išgavimo procesas yra pirmasis mašininio mokymosi proceso etapas. Duomenys, naudojami mašininio mokymosi procese, gali būti išgaunami įvairiais būdais [15], [16] – nuo sutelktinio duomenų rinkimo iki sintetinių duomenų generavimo. Atsižvelgiant į naudojamą mašininio mokymosi metodą, surinktus duomenis gali reikėti sužymėti. Žymėjimo metu realaus pasaulio objektai priskiriami vienai ar kelioms klasėms. Žymėjimo procesas užbaigia duomenų išgavimo procesą, ir toliau vykdomas duomenų rinkinio paruošimas. Duomenims paruošti gali būti taikomi įvairūs metodai ir procedūros,

kurių taikymas gali skirtis, priklausomai nuo duomenų rinkinio. Dažniausiai duomenų paruošimo metu pašalinami tušti ar klaidingi duomenys, pašalinami dublikatai ir standartizuojami duomenų tipai. Paruošus duomenis, atliekamas duomenų rinkinio požymių tyrimas. Tiriant požymius, atskleidžiami pasitaikantys duomenų dėsningumai ir ryšiai tarp kintamųjų, jei tokie egzistuoja. Jei atrinktame duomenų rinkinyje yra kategorinių duomenų, priklausomai nuo naudojamo mašininio mokymosi sprendimo, juos gali reikti transformuoti į naujus požymius. Šios transformacijos metu duomenyse esančios kategorijos pakeičiamos iš tekstinės informacijos į naujus duomenų rinkinio stulpelius su skaitinėmis reikšmėmis [17]. Duomenų paruošimo procesas baigiamas, kai išgaunamas tinkamas duomenų rinkinys, kuris yra paruoštas naudoti mašininio mokymosi modelio mokymo procese.

Paprastai modelio derinimo ir kokybės (angl. *model performance*) vertinimo tikslais duomenų rinkinys padalijamas į tris dalis: mokymo, validavimo ir testavimo [18]. Mokymo dalis naudojama modeliui sukurti, validavimo dalis naudojama modelio mokymo metu, siekiant suderinti modelio hiperparametrus. Galiausiai, atliekant modelio kokybės vertinimo etapą, testavimo duomenų poaibis naudojamas atliekamos klasifikavimo arba regresijos užduoties rezultatų tikslumui įvertinti, kai modeliui pateikiamas testavimo duomenų rinkinys, kuris nebuvo naudojamas mokymo procese. Nepasiekus norimo modelio kokybės, modelio parametrų derinimą galima pratęsti ar jį kartoti, iki bus pasiekta tinkama kokybė. Esant pakankamai modelio kokybei, modelį galima išsaugoti faile ir naudoti regresijos ar klasifikavimo užduotims atlikti [37], [38]. Sukurti modeliai gali būti naudojami pavieniui arba modelių kolektyvuose, kur daugiau nei vienas modelis yra sujungiami, siekiant gauti tikslesnius spėjimus. Taip pat informacija apie modelius gali būti siunčiama ir naudojama kituose sprendimuose. Vienas iš būdų keletą modelių vienam mašininio mokymosi uždaviniui yra paskirstytas mašininis mokymasis.

6.2.2 Paskirstytas mašininis mokymasis

Paskirstytas mašininis mokymasis skiriasi nuo centralizuoto mašininio mokymosi tuo, kad jame dalyvauja keletas subjektų, kurie dalinasi duomenimis arba individualiais modeliais, kurie vėliau sujungiami į vieną modelį. Panašiai, kaip ir kolektyvo mokymosi atveju, paskirstyto mašininio mokymosi aplinkoje galima sujungti trijų tipų informaciją – klasifikatorius, klasifikatorių reprezentacijas ir klasifikatorių prognozes.

Federacinis paskirstytas mokymasis yra populiariausias paskirstyto mašininio mokymosi tipas, kuris naudojamas privatumui užtikrinti ir tolygiau paskirstyti skaičiavimo resursų panaudojimą. Federacinis mokymasis dažnai naudojamas kartu su kitais privatumo užtikrinimo metodais ir blokų grandinės technologijomis.

Pagal egzistuojančių dalyvių skaičių paskirstytojo mašininio mokymosi sprendimuose juos galima suskirstyti į dvi kategorijas: individualų ir bendradarbiavimu grindžiamą. Individualų paskirstytąjį mašininį mokymąsi kuria vienas subjektas, kuris paprastai naudoja centralizuotą aplinką. Šioje aplinkoje agreguojami modelio ar duomenų artefaktai iš daugelio įrenginių ar programinės įrangos sprendimų. Toks paskirstytasis mašininio mokymosi metodas nereikalauja sudėtingų bendradarbiavimo procesų ar pasitikėjimo tarp tinklo dalyvių, nes yra

valdomas vieno subjekto. Bendradarbiavimu grindžiamas paskirstytasis mašininis mokymasis jungia atskirų subjektų modelių ar duomenų įvestis. Bendradarbiavimas ir dalinimasis duomenimis ir modeliais leidžia išmokyti / gauti aukštesnės kokybės modelį. Pagrindinis skirtumas tarp šių paskirstyto mašininio mokymo kategorijų yra tas, kad bendradarbiavimu grindžiamas mokymasis reikalauja pasitikėjimo tarp tinklo dalyvių ir naudojamų paslaugų, o taikant individualų požiūrį pasitikėjimas nėra aktualus.

6.2.3 Privatumo užtikrinimo metodai

Duomenų privatumo užtikrinimas yra svarbus aspektas mašininio mokymosi modelių kūrimo procese, nes atakos, nukreiptos prieš šį procesą, gali atskleisti jautrius duomenis. Trys pagrindinės privatumo užtikrinimo mašininio mokymosi procesuose metodų grupės yra šios: duomenų nuasmeninimas [86], [87], [88], kriptografiniai metodai perduodamai informacijai apsaugoti [89], [90] ir privatumui užtikrinti skirti specializuoti sprendimai [91], [92], [93]

Vienas iš tokių privatumo užtikrinimo specializuoti sprendimų yra žinių perdavimo architektūra [92]. Ji taikoma mašininio mokymosi procese, siekiant apsaugoti modelio diegimo etapo metu naudojamą jautrią informaciją. Privatumui užtikrinti yra pasiūlyta keletas sprendimų [98]-[100]. Dauguma siūlomų sprendimų taiko žinių perkėlimo iš vieno ar kelių modelių į naują neuroninio tinklo modelį būdus, siekiant apsaugoti jautrius modelio parametrus. Sprendimai taip pat siūlo privataus mokytojų ansamblių agregavimą (angl. *Private Aggregation of Teacher Ensembles, PATE*) [104], [105]. PATE metodas naudoja jautrių duomenų rinkinius išmokyti keliems mokytojų modeliams, kurie vėliau naudojami agreguojančiam mokinio modeliui mokyti. Taip sukurtas agreguojantis klasifikavimo modelis neatskleidžia duomenų apie jam sukurti naudotus modelius, taip užtikrinant privatumą.

Privatumui užtikrinti taip pat galima taikyti ir duomenų nuasmeninimą, jų užšifravimą ar net blokų grandinės technologijas. Šioje disertacijoje koncentruotasi į paskirstytą bendradarbiavimu grindžiamą mokymą, buvo apžvelgti siūlymai ir sprendimai, jungiantys privatumo užtikrinimą, blokų grandinės technologijų naudojimą ir federacinio mokymosi metodus. Šios analizės metu buvo nustatyta, kad dauguma siūlymų papildo blokų grandinės technologijas specializuotais konsensuso algoritmais [70], [71], [82], [116], kurie siekia validuoti paskirstytų mašininio mokymosi modelių ir jų kūrimo kokybę. Šie metodai paprastai realizuojami naudojant viešąsias blokų grandines [117], [70] ir dažniausiai pateikiami tik kaip koncepcijos įrodymai, kuriems reikia realizuoti naujus blokų grandinės tinklus. Naujų tinklų kūrimas yra sudėtingas procesas, dėl šios priežasties tokių sprendimų panaudojimas praktikoje gali būti sunkiai pasiekiamas. Kuriant blokų grandinės tinklus, siūlymuose taip pat dažnai įvedami nauji duomenų teikėjo, duomenų tvirtintojo ar validavimo dalyvių vaidmenys blokų grandinės tinkle. Šie vaidmenys leidžia paskirstyti tinklo dalyvių atsakomybes ir apibrėžti vykdyto darbo atlygio apskaičiavimo būdus. Dažniausia privatumo išsaugojimo priemonė apžvelgtuose metoduose yra diferencinis privatumas ar informacijos, perduodamos ryšio kanalais, šifravimas. Diferencinio privatumo priemonės taikomos siekiant išsaugoti perduodamų jautrių duomenų privatumą. Daugumos siūlymų taikymo sritys skirstomos į dvi kategorijas:

sritys, kuriose reikia apsaugoti jautrius duomenis, kaip, pavyzdžiui, sveikatos priežiūros srityje [99]; sritys, kur yra daug fizinėje erdvėje pasiskirsčiusių duomenų ar modelių naudotojų, kurie bendradarbiauja, siekiant sukurti bendrą mašininio mokymosi sprendimą, kaip, pavyzdžiui, daiktų internetas [71], [74]. Taip pat yra siūlomi ir universalūs metodai, kuriuos būtų galima pritaikyti keletui sričių, tačiau dauguma jų skirti tik giliajam mokymuisi vykdyti [120], [121] ir nepalaiko kitų mašininio mokymosi modelių tipų. Dauguma lygintų metodų realizuojami tik vienoje mašininio mokymosi aplinkoje, neatsižvelgiant į kelių modelių tipų sujungimo metodus, todėl šių metodų pritaikomumas bendradarbiavimu grįstam paskirstytam mašininiam mokymui vykdyti yra ribotas.

6.2.4 Blokų grandinės technologijos

Blokų grandinės technologijos buvo išpopuliarintos *Bitcoin* kriptovaliutos, o augant technologijos brandai, buvo pradėtos taikyti ir kitose srityse. Terminas „blokų grandinė“ [128] apibrėžia duomenų struktūrą, kurioje, naudojant kriptografinius maišos algoritmus, įrašoma transakcijų informacija, jas sugrupuojant į blokus, o šiuos vėliau dar ir į blokų grandinę. Transakcijų validavimo ir naujo bloko įtraukimo į blokų grandinę procesas apibrėžiamas kaip konsensuso algoritmas [2]. Bendruomenės, valdančios blokų įtraukimą ir saugančios paskirstyto žurnalo kopijas, vadinamos blokų grandinės tinklais [2]. Blokų grandinės tinklus pagal jų narių prisijungimo būdus ir reikalaujamą pasitikėjimo lygį [129] galima skirstyti į viešuosius, privačius ir konsorciumo.

Išmanieji kontraktai buvo sukurti įgalinti, kurti ir vykdyti sudėtingesnėms programoms pasinaudojant blokų grandinės tinklu. Išmanieji kontraktai visų pirma buvo pristatyti *Ethereum* blokų grandinėje. Šie kontraktai, kaip ir transakcijos, įrašomi į blokų grandinėje esančius blokus ir vėliau gali būti vykdomi iškviečiant kontrakto funkcijas, o šie kvietimai užregistruojami kaip transakcijos. Kadangi į blokų grandinę įrašyti duomenys yra nekintami, į blokų grandinę įdiegti išmanieji kontraktai taip pat negali būti pašalinti ar pakeisti. *Ethereum* blokų grandinės išmanieji kontraktai yra kuriami naudojant specializuotas programavimo kalbas: *Solidity* [135], *Vyper* [136]. Taip pat išmaniųjų kontraktų diegimas ir vykdymas daugumoje viešųjų blokų grandinių yra apmokestintas. Apmokestinimo dydis priklauso nuo išmaniojo kontrakto sudėtingumo ir vykdymui reikalingų skaičiavimo išteklių kiekio [137]. Tai ne tik skatina išmaniųjų sutarčių kūrėjus optimizuoti savo kodą [138], kad jis būtų vykdomas kuo efektyviau, bet ir riboja sudėtingesnių programų kūrimą. Išmanieji kontraktai taip pat turi kūrimo apribojimų, nes programinį kodą vykdo tinklo dalyviai, kurie turi patvirtinti išmaniojo kontrakto vykdymo rezultatus, todėl, norint sėkmingai įvykdyti išmanųjį kontraktą ar jo funkcijas, gaunami rezultatai turi būti deterministiniai [139]. Tarkim, funkcija, kuri naudoja atsitiktinių skaičių generavimą, grąžina nedeterministinius rezultatus [139] ir dėl to negali būti realizuojama naudojant išmaniuosius kontraktus. Privačiose blokų grandinėse naudojamų išmaniųjų kontraktų vykdymo kaina nustatoma tinklą valdančios organizacijos, kuri gali pasirinkti juos vykdyti be jokio atlygio juos diegiant ir vykdant. Taigi, naudojant išmaniuosius kontraktus, galima kurti sudėtingesnius sprendimus be papildomų išlaidų, skirtų kodui vykdyti.

Išmanieji kontraktai nėra pritaikyti dideliems duomenų kiekiams saugoti ir apdoroti, o norint išspręsti šią problemą yra naudojamos blokų grandinės orakulo paslaugos. Orakulo paslaugos sudaro galimybę gauti arba teikti duomenis, reikalingus išmaniųjų kontraktų vykdymui, todėl jos apibrėžiamos kaip išmaniųjų kontraktų išplėtimo projektavimo šablonas [148], [150]. Blokų grandinės orakulus galima skirstyti pagal keturis pagrindinius kriterijus [151], [152]: duomenų šaltinio tipą, duomenų teikimo kryptį, orakulo projektavimo šabloną ir sąveiką su blokų grandine.

Mašininio mokymosi ir blokų grandinės sričių deriniai aktyviai tiriami daugelyje mokslinių tyrimų sričių, ypač daiktų interneto (IoT) [164], [165], sveikatos priežiūros ir saugumo srityse [166]. Dažnai blokų grandinės technologijos derinamos su federacinio mokymosi sprendimais [72], [133], kai blokų grandinės technologija naudojama siekiant padidinti pasitikėjimą ir suteikti priemonių tinklo dalyviams motyvuoti. Blokų grandinės technologija yra perspektyvus sprendimas, siekiant padidinti mašininio mokymosi procesų skaidrumą, palengvinti audito vykdymą ir padidinti saugumą.

6.3 BENDRADARBIAVIMU GRĮSTAS PASKIRSTYTO MAŠININIO MOKYMOSI METODAS BLOKŲ GRANDINĖJE

Bendradarbiavimu grįstas paskirstyto mašininio mokymosi metodas blokų grandinėje (68 pav.), naudojantis blokų grandinės technologiją (angl. *collaborative distributed machine learning on blockchain, CDMLB*), skirtas organizacijoms, kurios nori bendradarbiauti, spręsdamos mašininio mokymosi problemas, tačiau nepakankamai pasitiki kitais proceso dalyviais, kad tiesiogiai dalintųsi duomenimis ar mašininio mokymosi modeliais. CDMLB metodo tikslas yra suteikti galimybę vykdyti privatumą užtikrinantį bendradarbiavimą vykdant paskirstytą mašininį mokymąsi, panaudojant blokų grandinės technologijas. CDMLB metodas palaiko esamas mašininio mokymosi technologijas, modelius ir duomenų rinkinius, įgalindamas bendradarbiavimą privačiame blokų grandinės tinkle. Naudojant privatų blokų grandinės tinklą, padidėja modelių diegimo proceso skaidrumas. Be to, privačios blokų grandinės naudojimas padidina pasitikėjimą procesu, nes modelio spėjimų (angl. *model inference*) apskaičiavimo rezultatus tikrina keletas tinklo dalyvių. Blokų grandinės technologijos naudojimas taip pat padidina sistemos patikimumą, nes, sutrikdžius vieno tinklo komponento veikimą, nėra sutrikdomas viso tinklo darbas, kas galėtų įvykti naudojant trečiosios šalies architektūrą.

CDMLB metodas taip pat suteikia priemonių duomenų privatumui užtikrinti, nes apibrėžia kelias specializuotas aplinkas ir specializuotą modelių diegimo ir naudojimo procesą. Siūlomas metodas vykdomas keliose aplinkose: privataus modelio kūrimo aplinkoje; blokų grandinės (angl. *on-chain*) aplinkoje, naudojamoje modeliams diegti; lokalioje tinklo mazgo (angl. *off-chain*) aplinkoje.

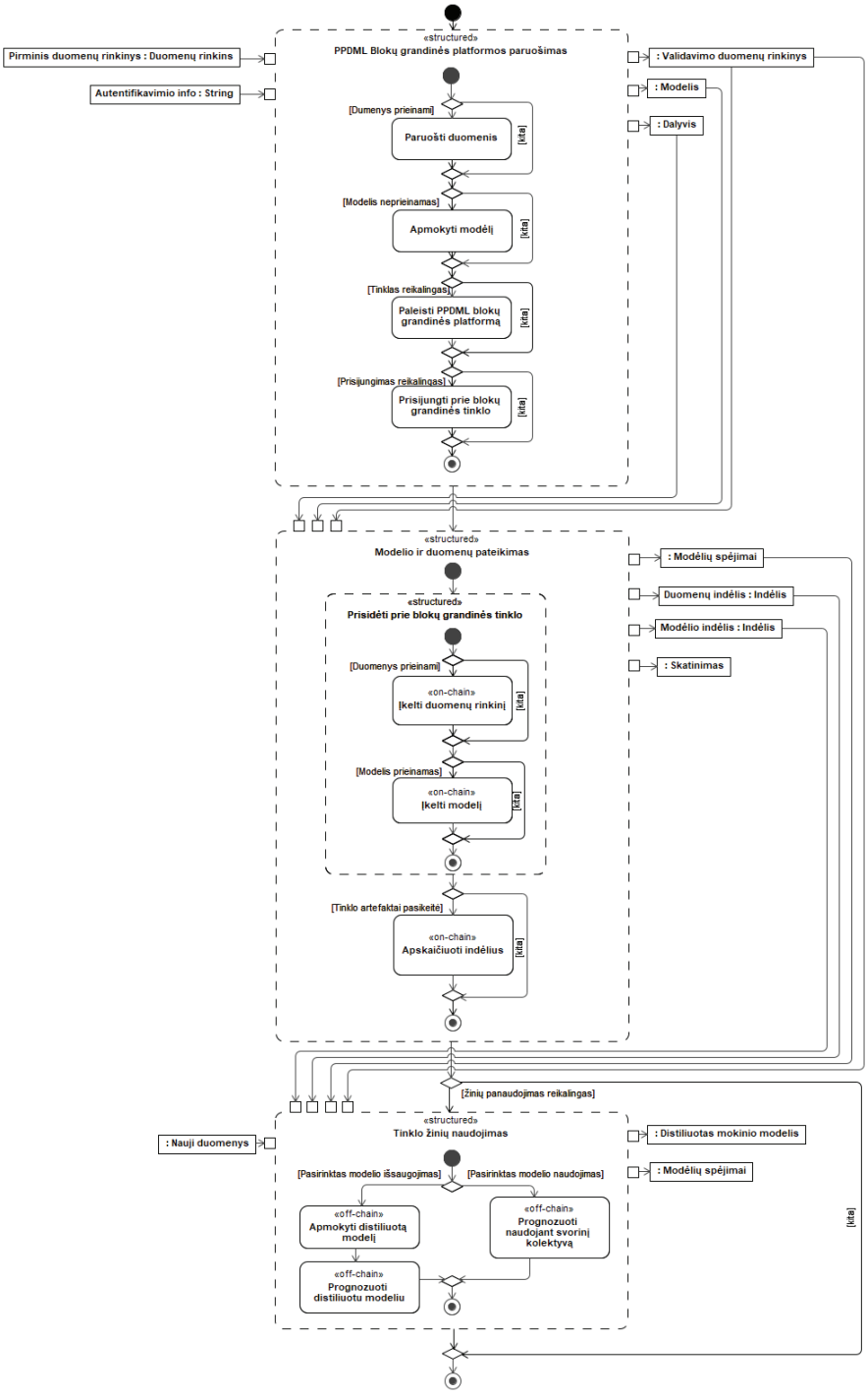
Privataus modelio kūrimo aplinka nėra tiesiogiai integruojama į metodą ir yra valdoma modelių kūrėjų. Šios aplinkos parametrai ir konfigūracijos priklauso tik nuo modelio kūrėjo poreikių. Ši aplinka užtikrina, kad mokymo duomenų valdymas ir modelio mokymas išliktų privatus, ir leidžia naudotojui organizuoti mokymą pagal individualius poreikius. Sprendimą atlikti modelio mokymą privačioje ne blokų grandinės aplinkoje lemia poreikis apsaugoti jautrius mokymo duomenis. Vieninteliai

apribojimai, taikomi šiai aplinkai, – tai gaunamų duomenų ir modelių failų formatai ir jų struktūra, kuri turi būti suderinta su bloko grandinės tinkle nurodytais pavyzdžiais ir reikalavimais.

Modeliams diegti naudojama *on-chain* aplinka, siekiant sukurti skaidrų, veiklą registruojantį procesą, skirtą dalintis mašininio mokymosi modeliais ir duomenimis. Joje atliekami veiksmai registruojami paskirstytame žurnale, leidžiant stebėti vykdymą, taip padidinant proceso skaidrumą. Šioje aplinkoje modelio diegimo procesai realizuojami išmaniaisiais kontraktais.

Lokali *off-chain* aplinka naudojama siekiant palaikyti platesnę aibę mašininio mokymosi programavimo technologijų ir sprendimų. Naudojant kartu su blokų grandinės modelių diegimo aplinka, ji leidžia palaikyti daugiau mašininio mokymosi modelių tipų, taip pat suteikia modelių kolektyvo panaudojimo galimybę bei suteikia galimybę decentralizuoti ir paskirstyti sudėtingus modelių skaičiavimus, naudojant *off-chain* orakulo paslaugas.

Pasiūlyto CDMLB metodo procese egzistuoja dalyviai, kurie valdo duomenis ir moko mašininio mokymosi modelius. Prieš prasidedant bendradarbiavimo procesui dalyviai paskirsto modelius į mokymosi, testavimo ir validavimo dalis. Pasinaudodami mokymosi duomenų imtimi tinklo dalyviai moko pasirinktus klasifikatorius, kuriuos įkelia į blokų grandinės tinklą pasitelkdami išmaniaisiais kontraktais ir juos naudojančiomis paslaugomis. Norėdami vykdyti bendradarbiavimą, dalyviai su blokų grandinės tinklu turi pasidalinti ne tik modelių failais, bet ir validavimo duomenimis. Pasinaudodami šiais dviem artefaktais, išmanieji kontraktai apskaičiuoja kiekvieno dalyvio modelių ir duomenų indėlį.



68 pav. Bendradarbiavimu grįsto paskirstyto mašininio mokymosi metodo procesas

CDMLB blokų grandinės platformos paruošimo etapas apima mašininio mokymosi artefaktų, kurie vėliau bus įdiegti į blokų grandinės tinklą, kūrimo procesus. Šis etapas taip pat apima procesus, reikalingus CDMLB blokų grandinės platformai įdiegti ir prie jos prisijungti. Tinklo artefaktų paruošimas prasideda nuo duomenų parengimo ir yra atliekamas privačioje modelio kūrimo aplinkoje. Tinklo kūrėjas, atsižvelgdamas į duomenų reikalavimus, parengtus bendradarbiaujančių organizacijų, paruošia duomenų rinkinį. Šio etapo rezultatas yra parengti mokymo ir validavimo duomenų rinkiniai. Mokymo duomenų rinkinys bus naudojamas modeliui sukurti, o validavimo – bus įkeltas į blokų grandinės tinklą ir naudojamas modelio tikslumui tikrinti. Jei tinklo dalyvis negali pateikti duomenų rinkinio, duomenų rengimo procesas gali būti praleistas. Atitinkamai, jei dalyvis negali dalintis validavimo duomenų rinkiniu dėl privatumo ar saugumo apribojimų, duomenų rinkinio skaidymo į atskirus duomenų rinkinius taip pat galima nevykdyti. Parengtas mokymo duomenų rinkinys naudojamas klasifikatoriaus modeliui mokyti. Kadangi CDMLB metodas neriboja modelio mokymo proceso, galima naudoti skirtingas modelio kūrimo aplinkas su įvairiomis aparatinės ir programinės įrangos konfigūracijomis ir skirtingais modelio kūrimo parametrais. Nors modelio mokymo procesas nepriklauso nuo metodo, gauto modelio failo formatas ir struktūra turi atitikti bendradarbiaujančių organizacijų nustatytus modelio failo formato reikalavimus. Kai mašininio mokymosi modelio ir duomenų kūrimas baigtas, tinklo dalyvis prisijungia prie blokų grandinės tinklo. Tam gali prireikti įdiegti CDMLB platformą, jei tinklo dalyvis jungiasi prie platformos pirmą kartą. Metode esanti platformos parengimo dalis baigiasi prisijungimu prie privataus blokų grandinės tinklo.

Modelio ir duomenų diegimo etapas apima procesus, reikalingus dalintis mašininio mokymosi artefaktais, t. y. modeliu ir duomenimis, blokų grandinės tinkle, ir procedūras, atliekamas siekiant įvertinti šių artefaktų kokybę. Visi šiame etape atliekami veiksmai realizuoti naudojant išmaniuosius kontraktus. Norint pradėti naudotis modelių diegimo aplinka privačioje blokų grandinėje, visų pirma reikia prisijungti, pateikiant organizacijų suteiktus prieigos duomenis. Prisijungus, modelio ir duomenų artefaktus galima įtraukti į tinklą, įkeliant duomenų rinkinį arba modelį naudojant paskirstytąją programą (angl. *Distributed application, DApp*). Siekiant sutaupyti blokų grandinės saugykloje vietos, įkelto duomenų rinkinio formatas transformuojamas ir, jei blokų grandinės saugykloje egzistuoja bent vienas mašininio mokymosi modelis, įkeltas duomenų rinkinys naudojamas modelio spėjimams apskaičiuoti. Modelio spėjimai išsaugomi blokų grandinės saugykloje. Modelio įkėlimo procedūra pradedama įkeliant išsaugotą mašininio mokymosi modelio failą į blokų grandinės tinklą, taip pat naudojantis paskirstytąja aplikacija. Modelio failo informacija transformuojama į tekstinę informaciją ir išsaugoma blokų grandinės saugykloje. Jei modelio ar duomenų rinkinio įkėlimo procedūra sėkminga, įkeltus artefaktus blokų grandinės tinklas replikuoja tarp tinklo dalyvių.

Kad būtų galima sekti tinklo dalyvių indėlį į bendrą modelių kolektyvą, vertinama kiekvieno įkelto modelio kokybė. Pasiūlytame metode modelių kokybė vertinama pasinaudojant binarinės kryžminės entropijos tikslumo įvertinimo metrika. Modelių kokybė toliau vertinama pasinaudojant *Shapley* indėlio įvertinimo funkciją, kurioje dalyvio indelio reikšmė pakeičiama atvirkštinės binarinės kryžminės

entropijos reikšme. *Shapley* formulė taip pat įvertina, ar kiekvienas modelio tikslumas buvo didesnis nei atsitiktinių spėjimų atsižvelgiant į turimą duomenų aibę, ir modeliai, pasirodantys prasčiau nei atsitiktinio spėjimo reikšmė, nėra įtraukiami į vertinimą. Pasiūlytas metodas taip pat vertina tik tokius modelius, kurie savo tikslumu pagerina kolektyvą, o ne jį pablogina, nurodant taisyklę neįtraukti dalyvių su neigiamais kolektyvo svoriais. Vertinant duomenų rinkinio indėlį, visi įkelti modeliai sujungiami į modelių kolektyvą, o patvirtinimo duomenų rinkiniai sujungiami į vieną duomenų rinkinį, neįtraukiant duomenų rinkinio, kurio našumas vertinamas. Vertinant modelio indėlį, taip pat naudojami modelių kolektyvai ir duomenų rinkiniai. Tačiau vietoj to, kad būtų vertinamas vienas duomenų rinkinys, vertinamas vienas modelis, o siekiant supaprastinti vertinimo procesą, įvertinamas tik geriausios kiekvieno tinklo dalyvio kokybės modelis. Modelio ir duomenų kokybės reikšmės gali būti naudojamos skatinimo mechanizmui sukurti. Skatinimo rodikliai gaunami paverčiant modelio ar duomenų kokybės reikšmes į santykinį dalyvio indėlį. Šis santykinis indėlis naudojamas siekiant nustatyti, kokį paskatinimą reikėtų suteikti prisidėjusiam tinklo nariui už jo dalyvavimą procese.

Paskutinis CDMLB metodo etapas skirtas blokų grandinėje saugomoms modelio žinioms naudoti. Pateikiami du skirtingi būdai, kaip galima panaudoti modelius. Pirmasis būdas modelius sujungia į kolektyvą naudojant svertinį vidurkį. Tada, naudojant naujus duomenis, kurie pateikiami tinklo dalyvių, gaunama kolektyvo prognozė. Modelių kolektyvą sukuria specializuota blokų grandinės orakulo paslauga, o nematyti duomenys, naudojami prognozei apskaičiuoti, niekada neviešinami už lokalios tinklo mazgo aplinkos ribų. Toks būdas leidžia užtikrinti nematytų duomenų rinkinio ir prognozių privatumą, kartu registruojant mašininio mokymosi modelio panaudojimą. Antrasis būdas naudoja žinių distiliavimo strategiją, skirtą agreguojančiam modeliui mokyti naudojant tinkle blokų grandinėje sukauptą modelių kolektyvą. Taikant šį būdą, naujam neuroninio tinklo modeliui mokyti naudojami sujungti visi tinkle sukaupti validavimo duomenys ir modelių kolektyvas. Sujungti modelių spėjimai naudojami kaip įvestis mokant neuroninio tinklo modelį, o jo kokybė patikrinama naudojant visus tinkle sukauptus validavimo duomenis. Distiliuoto modelio failas išsaugomas autoriaus lokalaus mazgo aplinkoje ir gali būti toliau tobulinamas naudojant individualius duomenis arba tiesiog naudojamas prognozėms atlikti be papildomo mokymo. Žinių distiliavimo strategija užtikrina blokų grandinėje saugomo mašininio mokymosi modelių privatumą.

Blokų grandinės technologijos panaudojimas privatumą išsaugančiame paskirstyto mokymosi procese suteikia galimybę bendradarbiauti organizacijoms, leidžia bet kuriuo metu analizuoti blokų grandinės transakcijas ir blokų grandinės artefaktus, taip didinant skaidrumą. CDMLB metodas skiriasi nuo esamų sprendimų tuo, kad palaiko skirtingus prižiūrimo mokymosi (angl. *supervized learning*) modelių tipus ir užduotis vietoje siūlymo kurti blokų grandinės tinklą kiekvienai specializuotai užduočiai ar modelio tipui. CDMLB metodas reikalauja nedidelio kiekio nuasmenintų duomenų modelio kokybei patvirtinti, o jautriais mokymosi duomenimis nėra dalinamasi, taip sumažinant galimus privatumo pažeidimus. CDMLB metodas taip pat palaiko du tinklo žinių panaudojimo būdus, kurie užtikrina naudojamų modelių

privatumą. Galiausiai, CDMLB metodas leidžia prisidedančioms organizacijoms įvertinti dalyvių indėlį ir panaudoti jį skatinimui apskaičiuoti.

6.4 BENDRADARBIAVIMU GRĮSTO PASKIRSTYTO MAŠININIO MOKYMOSI METODO BLOKŲ GRANDINĖJE RELIZACIJA

Pristatytas CDMLB metodas buvo realizuotas pasinaudojant *Hyperledger Fabric* privačia blokų grandinės technologija. Sistemos realizavimo ir tyrimų metu buvo naudojami sprendimų medžių ir logistinės regresijos klasifikatoriai, kurie eksperimentiškai tyrė duomenų rinkinius iš medicinos ir finansų dalykinių sričių. Realizuota CDMLB bendradarbiavimo platformoje buvo sukurti išmanieji kontraktai, leidžiantys vykdyti bendradarbiavimą pasinaudojant modelių dalinimo procesu. Šie kontraktai buvo sukurti pasinaudojant *Go* programavimo kalba ir apėmė duomenų ir modelių failų nuskaitymo, jų validavimo, modelių tikslumų vertinimo ir modelių kolektyvo panaudojimo funkciją. Taip pat esama blokų grandinės architektūra buvo išplėsta siekiant palaikyti dvi lokalių orakulų aplinkas panaudojant *Python* ir *R* programavimo kalbas. Siekiant ištirti realizuoto metodo veikimą, bus sukurti mašininio mokymosi sprendimų medžių ir logistinės regresijos klasifikatoriai, kurių sukūrimui buvo panaudotos *PySpark* ir *MLR3* mašininio mokymosi bibliotekos. Daugiau nei viena modelių realizavimo technologija buvo pasirinkta siekiant parodyti modelio lankstumą ir galimybę palaikyti. Tinklo žinių distiliavimo sprendimas buvo realizuotas pasinaudojant dviem architektūromis: negiliojo mokymosi architektūra buvo realizuota pasinaudojant *Keras* mašininio mokymosi biblioteką ir buvo sudaryta 3 lygių neuroninio tinklo. Taip pat buvo panaudota *TabNet* giliojo mokymosi architektūra.

6.5 EKSPERIMENTINIAI TYRIMAI

Eksperimentinis metodo vertinimas buvo įgyvendintas atliekant tris eksperimentus. Kiekviename eksperimente buvo vertinama siūlomo CDMLB metodo dalis ir procesus realizuojantys sprendimai [182], [183].

Pirmasis eksperimentas buvo skirtas metodo CDMLB blokų grandinės platformos parengimo etapui ir orakulo projektavimo šablono taikymui vertinti. Eksperimentas buvo atliktas siekiant patikrinti, ar lokalių orakulų projektavimo šablonas gali būti naudojamas egzistuojančiose blokų grandinės technologijose ir kaip naujų *off-chain* paslaugų pridėjimas į blokų grandinės tinklą gali paveikti veikimo greitaveiką.

Antrasis eksperimentas buvo skirtas modelio indėlio apskaičiavimo proceso daliai įvertinti. Eksperimento metu buvo orientuotasi į modelio indėlio dalies apskaičiavimą, nes siūlymų vertinti duomenų indėlį jau yra ([158], [159]), o modelio dalies apskaičiavimas nebuvo išsamiau tyrinėtas. Eksperimente buvo lyginamos pasiūlytos modelių kolektyvų svorių apskaičiavimo strategijos su dažniausiai naudojamomis strategijomis ir artimiausia pasiūlytoms strategijoms *Shapley* balsavimu grindžiama svorių apskaičiavimo strategija. Eksperimente pasiūlytos svorių apskaičiavimo strategijos buvo lyginamos su kita artima *Shapley* balsavimu grįsta strategija ir keliomis kitomis žinomesnėmis strategijomis.

Trečiajame eksperimente buvo tiriamas blokų grandinės tinkle sukauptų žinių panaudojimo procesas, vertinant prognozavimo tikslumą naudojant tiek tinklo modelių kolektyvą, tiek distiliuotą modelį. Eksperimento metu buvo naudojami antrajame eksperimente sukurti modeliai ir sukurtas distiliuotas neuroninio tinklo modelis. Distiliuotas modelis buvo realizuotas naudojant tris skirtingas konfigūracijas, jos buvo lyginamos su antrajame eksperimente sukurtų modelių kolektyvų veikimu.

6.5.1 Modelio spėjimų skaičiavimo naudojant lokalias orakulo paslaugas vertinimas

Šio eksperimento tikslas – įvertinti lokalių orakulo paslaugų poveikį greitaveikai ir įvertinti modelio spėjimų skaičiavimo algoritmą, naudojant privačios blokų grandinės technologiją. Siekiant palyginti pasiūlytą architektūrą su jau egzistuojančiais sprendimais, modelio spėjimų skaičiavimas buvo realizuotas dviem būdais. Pirmasis būdas – naudojant tik išmanųjį kontraktą, kuris padengė modelio spėjimų skaičiavimo logiką. Antrasis būdas naudojo išmanųjį kontraktą, išplėstą lokalia *off-chain* orakulo paslauga, kuri atliko modelio spėjimų skaičiavimą ir rezultatus grąžindavo išmaniajam kontraktui.

Sintetinių duomenų rinkinio modelio spėjimų apskaičiavimo algoritmo vykdymo laiko palyginimo rezultatai parodė vidutinį nedidelį ~2% greitaveikos sulėtėjimą, o atskirų tinklo ir duomenų konfigūracijų atveju – mažiau nei 6,60 %. Skaičiavimai sulėtėjo daugiau naudojant mažesnio dydžio duomenų rinkinius, o taip įvyko labiau dėl laiko, reikalingo duomenims perduoti tarp išmaniojo kontrakto ir *off-chain* orakulo paslaugos, nei dėl laiko, skirto modelio spėjimams skaičiuoti. Naudojant didesnį kiekį tinklo narių ir didesnės apimties duomenų rinkinius, skaičiavimo laikas sulėtėjo mažiau, nes laikas, skirtas modelių spėjimams skaičiuoti, tapo ilgesnis už laiką, skirtą duomenims perduoti.

Atlikto eksperimento rezultatai parodė, kad, naudojant orakulo paslaugas, algoritmo greitaveika sulėtėjo dėl papildomos komunikacijos tarp išmaniojo kontrakto ir orakulo paslaugos. Orakulo paslaugų naudojimas leidžia pakartotinai panaudoti egzistuojančias mašininio mokymosi aplinkas ir sprendimus, taigi šis sulėtėjimas nėra toks didelis, kad atsvertų lankstumą, kurį suteikia orakulų paslaugos. Nepaisant to, dėl greitaveikos ir lankstumo kompromiso turėtų spręsti organizacijos, norinčios naudoti blokų grandinės technologijas paskirstytam mašininiam mokymuisi vykdyti.

6.5.2 Shapley reikšmėmis grindžiamos kolektyvo svorių apskaičiavimo strategijos efektyvumo vertinimas

Eksperimentai, skirti įvertinti *Shapley* kolektyvo svorių apskaičiavimo strategiją, vertino jos tikslumą ir tyrė, ar galima išmatuoti modelių indėlį, juos sujungiant į blokų grandinės tinkle kaupiamą modelių kolektyvą. Modelių sujungimo į kolektyvus metodai buvo parinkti dėl galimybės sujungti kelis modelių tipus, siekiant išvengti modelių struktūros suvienodinimo. Eksperimento metu buvo tikrinama dviejų *Shapley* reikšmėmis ir tikslumu grindžiamų kolektyvo svorių

apskaičiavimo strategijų kokybė (*posShap, maxShap*), kurios palygintos su dažniausiai naudojamomis svorių apskaičiavimo strategijomis ir kita *Shapley* reikšmėmis ir balsavimu grindžiama strategija. *posShap* strategija nuo *maxShap* strategijos skyrėsi tuo, kad pašalindavo kolektyvo narius, kurie pablogina kolektyvo tikslumą, o *maxShap* pakeisdavo modelio spėjimus į atvirkštinius. Eksperimente buvo vertinamos lokalios *off-chain* orakulo paslaugos, sukurtos naudojant *R* ir *Python* programavimo kalbas. *R* kalbos aplinkoje modeliams kurti buvo panaudota *MLR3* mašininio mokymosi biblioteka, o *Python* aplinkoje - *PySpark* mašininio mokymosi biblioteka. Modeliams kurti ir vertinti buvo naudojami du su bankininkyste susiję duomenų rinkiniai. Modelio tikslumas eksperimento metu buvo matuojamas pasinaudojant binarine kryžmine entropija.

Didžiausias tikslumo pagerėjimas nustatytas lyginant p*osShap* su vieno modelio (Mono) metodu ir naudojant homogeninius kolektyvus, kuriuos sudarė 13 modelių: *BNG_credit-a* ir *Bank Marketing* duomenų rinkiniams, atitinkamai 4,8 % ir 1,9 %. Lyginant p*osShap* su dažniausiai naudojamu tikslumu pagrįstu svorių apskaičiavimo metodu (*Perf*), pasiūlyta strategija padidino ansamblio našumą 0,7 %. Atlikus kolektyvų rezultatų reitingavimą paaiškėjo, kad *posShap* strategija pasiekė didžiausią tikslumą, išskyrus vieną *Python* sprendimų medžio klasifikatoriaus realizacijos ir *Bank Marketing* duomenų rinkinio konfigūraciją. Panašūs *posShap* svorio apskaičiavimo strategijos rezultatai buvo pastebėti ir heterogeniniuose ansambliuose, kurie pasiekė 1,4 % didesnį tikslumą, palyginti su *Mono* metodu *BNG_credit-a* duomenų rinkinio konfigūracijoje. *Bank Marketing* duomenų rinkinio atveju *posShap* pasiekė 0,4 % našumo padidėjimą, tačiau kaip tiksliausia svorių apskaičiavimo strategija buvo nustatyta lygių svorių apskaičiavimo strategija (*Equal*), nes našumas, palyginti su *Mono* metodu, padidėjo 0,6 %.

Iš dviejų pasiūlytų kolektyvų svorių apskaičiavimų strategijų tik *posShap* pasižymėjo teigiamais rezultatais, o *maxShap* strategijos rezultatai buvo prastesni už visų išbandytų strategijų rezultatus. Galima pastebėti, kad taikyti spėjimų koregavimo metodai nepagerina rezultatų, o nenaudingų modelių pašalinimas, naudotas *posShap* strategijoje, pasiteisino. *PosShap* strategijos rezultatai varijuoja priklausomai nuo naudoto duomenų rinkinio ir modelio tipo, tačiau eksperimento rezultatai rodo, kad *posShap* strategija pranoko arba nebuvo prastesnė nei kitos išbandytos svorių apskaičiavimo strategijos, įskaitant *Shapley* balsavimu pagrįstą strategiją (*Roz*).

6.5.3 Žinių distiliavimo strategijos efektyvumo vertinimas

Žinių distiliavimo strategijos eksperimento tikslas buvo įvertinti poveikį klasifikavimo tikslumui, pritaikius žinių distiliavimo strategiją privatumui padidinti. Eksperimento metu buvo kuriami neuroninio tinklo modeliai, pasinaudojant kolektyvo spėjimais, sujungtais naudojant svorinį vidurkį, kaip mokymo duomenimis. Mokymas taip pat naudojo du validavimo duomenų rinkinius, gautus iš antrajame eksperimente tirtų *Bank Marketing* ir *BNG-credit_a* duomenų rinkinių. Kuriant distiliuotą modelį buvo išbandyta keletas distiliavimo lygmenų – nuo distiliavimo neįtraukimo (*alpha=1*) iki smarkaus distiliavimo (*alpha=0.5*). Eksperimento rezultatai parodė, kad žinių distiliavimo strategija sumažina modelių kolektyvo klasifikatoriaus tikslumą bent 16,99 % naudojant *Bank Marketing* duomenų rinkinį,

ir bent 10,41% naudojant *BNG_credit-a* duomenų rinkinį, lyginant *Shapey* strategiją su *alpha=0.75* strategija. Lyginant skirtingus distiliavimo lygmenis paaiškėjo, kad kolektyvo spėjimų įtraukimas į klaidos funkciją didesniu santykiu (*alpha=0.5*), siekiant subalansuoto distiliavimo varianto, lėmė prasčiausią našumą iš visų išbandytų strategijų. Skirtumai tarp distiliavimo strategijų *alpha=0.75* ir *alpha=1 BCE* medianų buvo statistiškai nereikšmingi trijose iš keturių tirtų duomenų rinkinio ir modelių tipų konfigūracijų. Rezultatų panašumai rodo, kad nežymus tinklo ansamblio prognozės padidėjimas drastiškai nekeičia klasifikatoriaus tikslumo. Nors žinių distiliavimas sumažina prognozavimo tikslumą kolektyvo atžvilgiu, pristatyta strategija pagerina privatumą, nes sukuria iš anksto išmokytą neuroninio tinklo modelį, kurį prireikus galima toliau tobulinti. Jei tolesnis modelio derinimas nėra reikalingas, *Shapley* svertinio ansamblio naudojimas užtikrina didesnį prognozavimo tikslumą pasiūlytame CDMLB metode.

6.5.4 Žinių distiliavimo efektyvumo vertinimas panaudojant giliojo mokymosi modelių architektūras

Žinių distiliavimo efektyvumo vertinimo eksperimento tikslas buvo įvertinti, ar giliojo mokymosi modelių architektūros gali pagerinti distiliavimo proceso metu sukuriamo modelio kokybę. Šio eksperimento metu buvo kuriamas modelis panaudojant TabNet [211] giliojo mokymosi modelio architektūrą. Ši architektūra pasirinkta dėl jos taikymo galimybių lentelės tipo duomenų rinkiniams. Šis eksperimentas naudojo identiškus duomenų rinkinius ir jų paruošimo sąlygas, apibūdintas žinių distiliavimo strategijos efektyvumo vertinimo eksperimente.

Šio eksperimento rezultatai parodė, kad, palyginti su žinių distiliavimo efektyvumo vertinimo eksperimente naudota architektūra, giliojo mokymosi architektūra sumažino distiliavimo metu prarandamą modelio tikslumą. Lyginant rezultatus tarp negiliosios ir giliosios neuroninio tinklo architektūrų naudojant silpną distiliavimą (dist075) modelio tikslumo praradimas sumažėjo nuo 16,99% iki 3,95% kuriant modelį naudojant *Bank Marketing* duomenų rinkiniui. Daliai klasifikatorių naudojant šią eksperimento konfigūraciją pavyko distiliuoti kolektyvą be tikslumo praradimo. Lyginant kolektyvo naudojimo rezultatus su klasifikatoriaus rezultatais nustatyta, kad *BNG_credit-a* duomenų rinkinio atveju rezultatai buvo statistiškai panašūs. Klasifikatorius, sukurtas silpno distiliavimo (dist075) metu, parodė tikslumo praradimus nuo 45% iki tikslumo pagerėjimo 4,41%. Toks didelis skirtumas tarp rezultatų buvo nustatytas dėl didelio rezultatų pasiskirstymo, tačiau, priklausomai nuo duomenų rinkinio ir modelio tipo, nuo 40% iki 70% pasirodė identiškai ar netgi aplenkė *Shapley* svorinio kolektyvo tikslumą. Atsižvelgiant į pagerintą tikslumą, net ir vidutinis distiliavimo atvejis tampa pakankamai tikslus naudoti su distiliavimo metu prarandamu tikslumu rėžyje tarp 9,24 % iki 1,53 %.

Tokie eksperimento rezultatai atskleidžia, kad, distiliuojant modelius, kurie turi sudėtingesnę vidinę reprezentaciją, tokius kaip sprendimų medžiai, distiliavimas tikslesnis. Iš rezultatų taip pat galime nustatyti, kad giliojo mokymosi architektūra buvo geresnė nei negili neuroninio tinklo mokymosi architektūra. Atsižvelgiant į tai žinių distiliavimo procesą rekomenduojama atlikti naudojant giliojo mokymosi

architektūra paremtus modelius, siekiant perkelti klasifikatorių žinias, sukauptas blokų grandinės tinkle, į kitas naudojimo aplinkas.

6.6 IŠVADOS

1. Paskirstyto mašininio mokymosi sprendimų ir metodų analizės metu buvo nustatyta, kad didžioji dalis naudojamų sprendimų architektūrų vis dar remiasi centralizuotais komponentais, kurie sumažina architektūros patikimumą ir reikalauja naudotojų pasitikėjimo. Taip pat didžioji dalis paskirstyto mašininio mokymosi sprendimų yra skirti specifinei problemai ar užduočiai spręsti ir dažnai palaiko vieną mašininio mokymosi modelio tipą, kas sumažina taikymo galimybes ir bendradarbiavimą.
2. Blokų grandinės technologijų taikymo paskirstytame mašininiame mokymesi analizė atskleidė, kad didžioji dalis sprendimų naudoja blokų grandines padidinti tinklo narių įsitraukimui bei registruoti nuosavybės ar perdavimo informaciją panaudojant paskirstytą žurnalą. Pagrindiniai siūlomų metodų ir sprendimų trūkumai kyla dėl specializuotų taikymų arba naudojimo tik proceso vykdymo eigai registruoti į blokų grandinę. Šių trūkumų galima būtų išvengti pridedant papildomas orakulų paslaugas.
3. Buvo pasiūlytas bendradarbiavimo metodas vykdant paskirstytą mašininį mokymąsi, panaudojant blokų grandinės technologijas. Metodas suteikia galimybę blokų grandinės tinklo nariams bendradarbiauti mašininio mokymosi modelių diegimo procese ir įvertinti pasidalinamų modelių ir duomenų naudą, apskaičiuojant jų indėlį kiekvienam dalyviui. Pagal pasiūlytą metodą sukurto sprendimo realizacija buvo atlikta pasinaudojant *Hyperledger Fabric* privačia blokų grandine, taip pademonstruojant galimybę vykdyti bendradarbiavimu grindžiamą paskirstytą mašininį mokymąsi. Realizuotas prototipas panaudojo lokalias *off-chain* orakulo paslaugas, kurios leido pakartotinai panaudoti įvairias mašininio mokymosi aplinkas ir technologijas, taip išplečiant ribotas blokų grandinės išmaniųjų kontraktų vykdymo aplinkas.
4. Metodas buvo eksperimentiškai įvertintas palyginant dviejų skirtingų architektūrų greitaveiką, kurių viena sukurta naudojant tik išmaniuosius kontraktus, o kita sukurta išmaniuosius kontraktus išplečiant lokalaus *off-chain* orakulo paslauga. Eksperimento rezultatai parodė, kad sprendimo greitaveika, pridėjus naująją orakulo paslaugą, sulėtėjo 2,07 %. Tačiau šių orakulo paslaugų įtraukimas leidžia pakartotinai panaudoti mašininio mokymosi aplinkas blokų grandinės tinkle, kas kompensuoja greitaveikos trūkumus.
5. Mašininio mokymosi ansamblių svorių apskaičiavimo strategijos tikslumo vertinimo eksperimento rezultatai parodė, kad galima kiekybiškai įvertinti kiekvieno nario pateiktų modelių indėlį į bendrą kolektyvą ir panaudoti šias metrikas skatinimo mechanizmui sukurti. *Shapley* reikšmėmis grįstas kolektyvo svorių apskaičiavimo metodas leido padidinti kolektyvo tikslumą 4,8 % ir 1,9 % dviem tirtiems duomenų rinkiniams, palyginti su vieno modelio

sprendimu (Mono), ir 0,7%, palyginti su tikslumu, grįstu svorių apskaičiavimu, kai modelių tikslumas buvo vertintas naudojant binarinę kryžminę entropiją. Kolektyvo svorių apskaičiavimo strategija leidžia įvertinti tinklo nario indėlį ir apskaičiuoti kolektyvo modelių svorius sprendimų sujungimui su panašiu ar net geresniu tikslumu, nei buvo pasiekta su kitomis eksperimente tirtomis strategijomis. Pasiūlyta *Shapley* reikšmėmis grįsta svorių apskaičiavimo strategija gali būti laikoma tikslumu grįstos strategijos generalizacija ir gali būti naudojama kartu su kitais modelio kokybės įverčiais.

6. Bloko grandinės tinklo žinių distiliavimo eksperimentas naudojant trijų lygių perceptroną parodė, kad distiliavimo taikymas nepagerino gauto modelio tikslumo, palyginti su modelių kolektyvu. Distiliavimo įtraukimas sumažino modelio tikslumą nuo 10,41 % iki 23,9 %, priklausomai nuo naudotojo klasifikatoriaus tipo ir duomenų rinkinio. Tačiau žinių distiliavimo naudojant giliojo mokymo architektūrą eksperimentas parodė, kad, palyginti su paprasta neuroninio tinklo architektūra, sudėtingesnė klasifikatoriaus architektūra gali tiksliau distiliuoti sukauptas kolektyvo žinias. Išmokytas TabNet klasifikatorius tikslumu net aplenkė svorinį kolektyvą esant tam tikroms sąlygoms. Lyginant kolektyvo rezultatus su distiliuotu modeliu, kai buvo distiliuotas logistinės regresijos modelis (*alpha*=0.75), jo tikslumas sumažėjo 3,95 % *Bank Marketing* duomenų rinkinio atveju ir 42 % *BNG-credit_a* duomenų rinkinio atveju. Distiliuojant sprendimų medžio klasifikatorių tikslumas sumažėjo 2,52 % *Bank Marketing* duomenų rinkinio atveju ir pagerėjo 7,98 % naudojant *BNG-credit_a* duomenų rinkinį. Rezultatų skirstinys *Bank Marketing* atveju buvo niekuo neišsiskiriantis, tačiau *BNG_credit-a* atveju susitelkė trijose grupėse. Šio duomenų rinkinio atveju viena iš grupių, kurią sudarė atitinkamai 57% ir 61% bandymo pakartojimų logistinės regresijos ir sprendimų medžių atvejų, net aplenkė kolektyvo tikslumą. Apibendrinus nustatyta, kad sudėtingesnės neuroninio tinklo architektūros naudojimas pagerino žinių distiliavimo efektyvumą ir yra rekomenduojamas naudoti naudojant šį metodą.

REFERENCES

[1] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus Inf Syst Eng*, vol. 59, no. 3, pp. 183–187, Jun. 2017, doi: 10.1007/s12599-017-0467-3.

[2] W. Wang *et al.*, "A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019, doi: 10.1109/ACCESS.2019.2896108.

[3] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016, doi: 10.1109/ACCESS.2016.2566339.

[4] K. Siau and W. Wang, "Building Trust in Artificial Intelligence, Machine Learning, and Robotics," *Cutter Business Technology Journal*, vol. 31, pp. 47–53, Mar. 2018.

[5] O. Schilke, M. Reimann, and K. S. Cook, "Trust in Social Relations," *Annual Review of Sociology*, vol. 47, no. 1, pp. 239–259, 2021, doi: 10.1146/annurev-soc-082120-082850.

[6] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, "Machine Learning Security: Threats, Countermeasures, and Evaluations," *IEEE Access*, vol. 8, pp. 74720–74742, 2020, doi: 10.1109/ACCESS.2020.2987435.

[7] J. C. S. do P. Leite and C. Cappelli, "Software Transparency," *Bus Inf Syst Eng*, vol. 2, no. 3, pp. 127–139, Jun. 2010, doi: 10.1007/s12599-010-0102-z.

[8] F. Bélanger and R. E. Crossler, "Privacy in the Digital Age: A Review of Information Privacy Research in Information Systems," *MIS Quarterly*, vol. 35, no. 4, pp. 1017–1041, 2011, doi: 10.2307/41409971.

[9] T. Hofmann and J. Basilico, "Collaborative Machine Learning," in *From Integrated Publication and Information Systems to Information and Knowledge Environments: Essays Dedicated to Erich J. Neuhold on the Occasion of His 65th Birthday*, M. Hemmje, C. Niederée, and T. Risse, Eds., in Lecture Notes in Computer Science. , Berlin, Heidelberg: Springer, 2005, pp. 173–182. doi: 10.1007/978-3-540-31842-2_18.

[10] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN COMPUT. SCI.*, vol. 2, no. 3, p. 160, Mar. 2021, doi: 10.1007/s42979-021-00592-x.

[11] K. M. J. Rahman *et al.*, "Challenges, Applications and Design Aspects of Federated Learning: A Survey," *IEEE Access*, vol. 9, pp. 124682–124700, 2021, doi: 10.1109/ACCESS.2021.3111118.

[12] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 2015, doi: 10.1126/science.aaa8415.

[13] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, no. 1, p. 6, 2008.

[14] F. Y. Osisanwo, J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[15] Y. Roh, G. Heo, and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021, doi: 10.1109/TKDE.2019.2946162.

[16] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data collection and quality challenges in deep learning: a data-centric AI perspective," *The VLDB Journal*, vol. 32, no. 4, pp. 791–813, Jul. 2023, doi: 10.1007/s00778-022-00775-9.

[17] M. K. Dahouda and I. Joe, “A Deep-Learned Embedding Technique for Categorical Features Encoding,” *IEEE Access*, vol. 9, pp. 114381–114391, 2021, doi: 10.1109/ACCESS.2021.3104357.
[18] S. Raschka, “Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning.” arXiv, Nov. 10, 2020. Accessed: Jul. 28, 2023. [Online]. Available: http://arxiv.org/abs/1811.12808
[19] D. Morgan and R. Jacobs, “Opportunities and Challenges for Machine Learning in Materials Science,” *Annual Review of Materials Research*, vol. 50, no. 1, pp. 71–103, 2020, doi: 10.1146/annurev-matsci-070218-010015.
[20] T. Hastie, R. Tibshirani, and J. Friedman, “Overview of Supervised Learning,” in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, and J. Friedman, Eds., in Springer Series in Statistics. , New York, NY: Springer, 2009, pp. 9–41. doi: 10.1007/978-0-387-84858-7_2.
[21] P. Cunningham, M. Cord, and S. J. Delany, “Supervised Learning,” in *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, M. Cord and P. Cunningham, Eds., in Cognitive Technologies. , Berlin, Heidelberg: Springer, 2008, pp. 21–49. doi: 10.1007/978-3-540-75171-7_2.
[22] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, “A systematic review on supervised and unsupervised machine learning algorithms for data science,” *Supervised and unsupervised learning for data science*, pp. 3–21, 2020.
[23] B. Mahesh, “Machine learning algorithms-a review,” *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.
[24] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
[25] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, May 1991, doi: 10.1109/21.97458.
[26] W. Du and Z. Zhan, “Building Decision Tree Classifier on Private Data”.
[27] J. E. King, “Binary logistic regression,” *Best practices in quantitative methods*, pp. 358–384, 2008.
[28] M. W. Gardner and S. R. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” *Atmospheric Environment*, vol. 32, no. 14–15, pp. 2627–2636, Aug. 1998, doi: 10.1016/S1352-2310(97)00447-0.
[29] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997, doi: 10.1016/S0031-3203(96)00142-2.
[30] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve.,” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[31] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006, doi: https://doi.org/10.1016/j.patrec.2005.10.010.
[32] Y. Ho and S. Wookey, “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling,” *IEEE Access*, vol. 8, pp. 4806–4813, 2020, doi: 10.1109/ACCESS.2019.2962617.
[33] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A Comprehensive Survey of Loss Functions in Machine Learning,” *Ann. Data. Sci.*, vol. 9, no. 2, pp. 187–212, Apr. 2022, doi: 10.1007/s40745-020-00253-5.
[34] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[35] G. Wang, J. Xu, and B. He, “A Novel Method for Tuning Configuration Parameters of Spark Based on Machine Learning,” in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Dec. 2016, pp. 586–593. doi: 10.1109/HPCC-SmartCity-DSS.2016.0088.

[36] C. Huang, Y. Li, and X. Yao, “A Survey of Automatic Parameter Tuning Methods for Metaheuristics,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 201–216, Apr. 2020, doi: 10.1109/TEVC.2019.2921598.

[37] M. E. Tipping, “Bayesian Inference: An Introduction to Principles and Practice in Machine Learning,” in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., in Lecture Notes in Computer Science. , Berlin, Heidelberg: Springer, 2004, pp. 41–62. doi: 10.1007/978-3-540-28650-9_3.

[38] G. E. P. Box and G. C. Tiao, *Bayesian Inference in Statistical Analysis*. John Wiley & Sons, 2011.

[39] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, “A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View,” *IEEE Access*, vol. 6, pp. 12103–12117, 2018, doi: 10.1109/ACCESS.2018.2805680.

[40] J. Zhao, Y. Chen, and W. Zhang, “Differential Privacy Preservation in Deep Learning: Challenges, Opportunities and Solutions,” *IEEE Access*, vol. 7, pp. 48901–48911, 2019, doi: 10.1109/ACCESS.2019.2909559.

[41] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, “Machine learning classification over encrypted data,” *Cryptology ePrint Archive*, 2014.

[42] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, Feb. 2021, doi: 10.1016/j.future.2020.10.007.

[43] S. Awan, F. Li, B. Luo, and M. Liu, “Poster: A Reliable and Accountable Privacy-Preserving Federated Learning Framework using the Blockchain,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London United Kingdom: ACM, Nov. 2019, pp. 2561–2563. doi: 10.1145/3319535.3363256.

[44] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, “Hardware for machine learning: Challenges and opportunities,” in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2017, pp. 1–8.

[45] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[46] R. E. Schapire, “The strength of weak learnability,” *Mach Learn*, vol. 5, no. 2, pp. 197–227, Jun. 1990, doi: 10.1007/BF00116037.

[47] O. Sagi and L. Rokach, “Ensemble learning: A survey,” *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018, doi: 10.1002/widm.1249.

[48] W.-Y. Loh, “Classification and regression trees,” *WIREs Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011, doi: 10.1002/widm.8.

[49] S. González, S. García, J. Del Ser, L. Rokach, and F. Herrera, “A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities,” *Information Fusion*, vol. 64, pp. 205–237, Dec. 2020, doi: 10.1016/j.inffus.2020.07.007.

[50] W. Fan, S. J. Stolfo, and J. Zhang, “The application of AdaBoost for distributed, scalable and on-line learning,” in *Proceedings of the fifth ACM SIGKDD international*

conference on Knowledge discovery and data mining, San Diego California USA: ACM, Aug. 1999, pp. 362–366. doi: 10.1145/312129.312283.
[51] A. Fern, "Online Ensemble Learning: An Empirical Study".
[52] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, San Francisco California: ACM, Aug. 2001, pp. 377–382. doi: 10.1145/502512.502568.
[53] F. Huang, G. Xie, and R. Xiao, "Research on Ensemble Learning," in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, Nov. 2009, pp. 249–252. doi: 10.1109/AICI.2009.235.
[54] L. Rokach, *Ensemble learning: pattern classification using ensemble methods*. World Scientific, 2019.
[55] A. L. Prodromidis and S. J. Stolfo, "Effective and Efficient Pruning of Meta-Classifiers in a Distributed Data Mining System".
[56] B. Pavlyshenko, "Using Stacking Approaches for Machine Learning Models," in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, Aug. 2018, pp. 255–258. doi: 10.1109/DSMP.2018.8478522.
[57] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Prog Artif Intell*, vol. 2, no. 1, pp. 1–11, Mar. 2013, doi: 10.1007/s13748-012-0035-5.
[58] A. Lazarevic and Z. Obradovic, "Boosting Algorithms for Parallel and Distributed Learning," *Distributed and Parallel Databases*, vol. 11, no. 2, pp. 203–229, Mar. 2002, doi: 10.1023/A:1013992203485.
[59] G. Tsoumakas and I. Vlahavas, "Effective stacking of distributed classifiers," in *Ecai*, 2002, pp. 340–344.
[60] A. L. Prodromidis, P. K. Chan, and S. J. Stolfo, "Meta-learning in distributed data mining systems: Issues and approaches".
[61] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A Survey on Distributed Machine Learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, Mar. 2021, doi: 10.1145/3377454.
[62] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Consistency in models for distributed learning under communication constraints," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 52–63, Jan. 2006, doi: 10.1109/TIT.2005.860420.
[63] C. Ma *et al.*, "When Federated Learning Meets Blockchain: A New Distributed Learning Paradigm," *IEEE Computational Intelligence Magazine*, vol. 17, no. 3, pp. 26–33, Aug. 2022, doi: 10.1109/MCI.2022.3180932.
[64] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[65] "PyTorch," PyTorch. Accessed: Dec. 17, 2023. [Online]. Available: https://pytorch.org/
[66] "Apache Spark™ - Unified Engine for large-scale data analytics." Accessed: Dec. 17, 2023. [Online]. Available: https://spark.apache.org/
[67] "TensorFlow." Accessed: Dec. 17, 2023. [Online]. Available: https://www.tensorflow.org/
[68] "Home," Horovod. Accessed: Dec. 17, 2023. [Online]. Available: https://horovod.ai/
[69] A. Tuladhar, S. Gill, Z. Ismail, and N. D. Forkert, "Building machine learning models without sharing patient data: A simulation-based analysis of distributed learning by ensembling," *Journal of Biomedical Informatics*, vol. 106, p. 103424, Jun. 2020, doi: 10.1016/j.jbi.2020.103424.
[70] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design," in *2018 IEEE International*

Conference on Big Data (Big Data), Dec. 2018, pp. 1178–1187. doi: 10.1109/BigData.2018.8622598.

[71] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020, doi: 10.1109/TII.2019.2942190.

[72] R. O. Ogundokun, S. Misra, R. Maskeliunas, and R. Damasevicius, "A Review on Federated Learning and Machine Learning Approaches: Categorization, Application Areas, and Blockchain Technology," *Information*, vol. 13, no. 5, Art. no. 5, May 2022, doi: 10.3390/info13050263.

[73] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated Learning for Internet of Things: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021, doi: 10.1109/COMST.2021.3075439.

[74] Y. Qi, M. S. Hossain, J. Nie, and X. Li, "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Generation Computer Systems*, vol. 117, pp. 328–337, Apr. 2021, doi: 10.1016/j.future.2020.12.003.

[75] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[76] "Federated Learning powered by NVIDIA Clara," NVIDIA Technical Blog. Accessed: Jul. 26, 2023. [Online]. Available: https://developer.nvidia.com/blog/federated-learning-clara/

[77] T.-T. Kuo and L. Ohno-Machado, "ModelChain: Decentralized Privacy-Preserving Healthcare Predictive Modeling Framework on Private Blockchain Networks." arXiv, Feb. 05, 2018. doi: 10.48550/arXiv.1802.01746.

[78] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled Federated Learning: A Survey," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–35, Apr. 2023, doi: 10.1145/3524104.

[79] Y. J. Kim and C. S. Hong, "Blockchain-based Node-aware Dynamic Weighting Methods for Improving Federated Learning Performance," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Sep. 2019, pp. 1–4. doi: 10.23919/APNOMS.2019.8893114.

[80] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain Empowered Asynchronous Federated Learning for Secure Data Sharing in Internet of Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020, doi: 10.1109/TVT.2020.2973651.

[81] U. Majeed and C. Seon, *EFLChain: Ensemble Learning via Federated Learning over Blockchain network: a framework*. 2019.

[82] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A Blockchain System for Private and Secure Federated Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021, doi: 10.1109/TPDS.2020.3044223.

[83] S. Zhou, H. Huang, W. Chen, P. Zhou, Z. Zheng, and S. Guo, "PIRATE: A Blockchain-Based Secure Framework of Distributed Machine Learning in 5G Networks," *IEEE Network*, vol. 34, no. 6, pp. 84–91, Nov. 2020, doi: 10.1109/MNET.001.1900658.

[84] M. Strobel and R. Shokri, "Data Privacy and Trustworthy Machine Learning," *IEEE Security & Privacy*, vol. 20, no. 5, pp. 44–49, Sep. 2022, doi: 10.1109/MSEC.2022.3178187.

[85] P. Voigt and A. Von dem Bussche, “The eu general data protection regulation (gdpr),” *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.
[86] L. SWEENEY, “k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, May 2012, doi: 10.1142/S0218488502001648.
[87] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “L-diversity: Privacy beyond k-anonymity,” *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, pp. 3-es, Mar. 2007, doi: 10.1145/1217299.1217302.
[88] N. Li, T. Li, and S. Venkatasubramanian, “t-Closeness: Privacy Beyond k-Anonymity and l-Diversity,” in *2007 IEEE 23rd International Conference on Data Engineering*, Apr. 2007, pp. 106–115. doi: 10.1109/ICDE.2007.367856.
[89] D. Boneh, A. Sahai, and B. Waters, “Functional Encryption: Definitions and Challenges,” in *Theory of Cryptography*, Y. Ishai, Ed., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 253–273. doi: 10.1007/978-3-642-19571-6_16.
[90] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, Bethesda MD USA: ACM, May 2009, pp. 169–178. doi: 10.1145/1536414.1536440.
[91] R. Shokri and V. Shmatikov, “Privacy-Preserving Deep Learning,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver Colorado USA: ACM, Oct. 2015, pp. 1310–1321. doi: 10.1145/2810103.2813687.
[92] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, “Scalable Private Learning with PATE.” arXiv, Feb. 24, 2018. Accessed: Jul. 24, 2023. [Online]. Available: http://arxiv.org/abs/1802.08908
[93] C. Liu, Y. Zhu, K. Chaudhuri, and Y.-X. Wang, “Revisiting model-agnostic private learning: faster rates and active learning,” *J. Mach. Learn. Res.*, vol. 22, no. 1, p. 262:11936-262:11979, Jan. 2021.
[94] M. Al-Rubaie and J. M. Chang, “Privacy-Preserving Machine Learning: Threats and Solutions,” *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, Mar. 2019, doi: 10.1109/MSEC.2018.2888775.
[95] Y. Li, J. Zhang, J. Zhu, and W. Li, “HBMD-FL: Heterogeneous Federated Learning Algorithm Based on Blockchain and Model Distillation,” in *Emerging Information Security and Applications*, J. Chen, D. He, and R. Lu, Eds., in Communications in Computer and Information Science. Cham: Springer Nature Switzerland, 2022, pp. 145–159. doi: 10.1007/978-3-031-23098-1_9.
[96] R. Xu, N. Baracaldo, and J. Joshi, “Privacy-Preserving Machine Learning: Methods, Challenges and Directions.” arXiv, Sep. 22, 2021. doi: 10.48550/arXiv.2108.04417.
[97] J. Liu, M. Juuti, Y. Lu, and N. Asokan, “Oblivious Neural Network Predictions via MiniONN Transformations,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, in CCS ’17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 619–631. doi: 10.1145/3133956.3134056.
[98] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network.” arXiv, Mar. 09, 2015. Accessed: Jul. 26, 2023. [Online]. Available: http://arxiv.org/abs/1503.02531
[99] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks,” in *2016 IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 582–597. doi: 10.1109/SP.2016.41.

[100] D. Li and J. Wang, “FedMD: Heterogenous Federated Learning via Model Distillation.” arXiv, Oct. 08, 2019. Accessed: Jul. 24, 2023. [Online]. Available: http://arxiv.org/abs/1910.03581
[101] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, “Communication-efficient federated learning via knowledge distillation,” *Nat Commun*, vol. 13, no. 1, Art. no. 1, Apr. 2022, doi: 10.1038/s41467-022-29763-x.
[102] D. Jiang, C. Shan, and Z. Zhang, “Federated Learning Algorithm Based on Knowledge Distillation,” in *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, Oct. 2020, pp. 163–167. doi: 10.1109/ICAICE51518.2020.00038.
[103] Z. Zhu, J. Hong, and J. Zhou, “Data-Free Knowledge Distillation for Heterogeneous Federated Learning,” in *Proceedings of the 38th International Conference on Machine Learning*, PMLR, Jul. 2021, pp. 12878–12889. Accessed: Jul. 24, 2023. [Online]. Available: https://proceedings.mlr.press/v139/zhu21b.html
[104] Y. Tian, D. Krishnan, and P. Isola, “Contrastive Representation Distillation,” presented at the International Conference on Learning Representations, Sep. 2019. Accessed: Jul. 26, 2023. [Online]. Available: https://openreview.net/forum?id=SkgpBJrtvS
[105] H. Chen, Y. Wang, C. Xu, C. Xu, and D. Tao, “Learning Student Networks via Feature Embedding.” arXiv, Dec. 16, 2018. doi: 10.48550/arXiv.1812.06597.
[106] J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge Distillation: A Survey,” *Int J Comput Vis*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021, doi: 10.1007/s11263-021-01453-z.
[107] Y. Hou, Z. Ma, C. Liu, T.-W. Hui, and C. C. Loy, “Inter-Region Affinity Distillation for Road Marking Segmentation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 12483–12492. doi: 10.1109/CVPR42600.2020.01250.
[108] J. Vongkulbhisal, P. Vinayavekhin, and M. Visentini-Scarzanella, “Unifying Heterogeneous Classifiers With Distillation,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 3170–3179. doi: 10.1109/CVPR.2019.00329.
[109] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data.” arXiv, Mar. 03, 2017. Accessed: Jul. 06, 2023. [Online]. Available: http://arxiv.org/abs/1610.05755
[110] F. Yuan *et al.*, “Reinforced Multi-Teacher Selection for Knowledge Distillation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, Art. no. 16, May 2021, doi: 10.1609/aaai.v35i16.17680.
[111] Y. Guan *et al.*, “Differentiable Feature Aggregation Search for Knowledge Distillation,” in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 469–484. doi: 10.1007/978-3-030-58520-4_28.
[112] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “FitNets: Hints for Thin Deep Nets.” arXiv, Mar. 27, 2015. Accessed: Jul. 26, 2023. [Online]. Available: http://arxiv.org/abs/1412.6550
[113] N. Komodakis and S. Zagoruyko, “Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer,” in *ICLR*, 2017.
[114] Y. Hu, Y. Zhou, J. Xiao, and C. Wu, “GFL: A Decentralized Federated Learning Framework Based On Blockchain.” arXiv, Apr. 13, 2021. Accessed: Jul. 24, 2023. [Online]. Available: http://arxiv.org/abs/2010.10996

[115] W. Jin, Y. Xu, Y. Dai, and Y. Xu, “Blockchain-Based Continuous Knowledge Transfer in Decentralized Edge Computing Architecture,” *Electronics*, vol. 12, no. 5, Art. no. 5, Jan. 2023, doi: 10.3390/electronics12051154.
[116] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, “A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus,” *IEEE Network*, vol. 35, no. 1, pp. 234–241, Jan. 2021, doi: 10.1109/MNET.011.2000263.
[117] K. Miyachi and T. K. Mackey, “hOCBS: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design,” *Information Processing & Management*, vol. 58, no. 3, p. 102535, May 2021, doi: 10.1016/j.ipm.2021.102535.
[118] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, “DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-Based Incentive,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438–2455, Sep. 2021, doi: 10.1109/TDSC.2019.2952332.
[119] H. Kim, S.-H. Kim, J. Y. Hwang, and C. Seo, “Efficient Privacy-Preserving Machine Learning for Blockchain Network,” *IEEE Access*, vol. 7, pp. 136481–136495, 2019, doi: 10.1109/ACCESS.2019.2940052.
[120] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, “FLChain: A Blockchain for Auditable Federated Learning with Trust and Incentive,” in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, Aug. 2019, pp. 151–159. doi: 10.1109/BIGCOM.2019.00030.
[121] Z. Mahmood and V. Jusas, “Implementation Framework for a Blockchain-Based Federated Learning Model for Classification Problems,” *Symmetry*, vol. 13, no. 7, Art. no. 7, Jul. 2021, doi: 10.3390/sym13071116.
[122] N. El Ioini and C. Pahl, “A review of distributed ledger technologies,” in *On the Move to Meaningful Internet Systems. OTM 2018 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II*, Springer, 2018, pp. 277–288.
[123] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”.
[124] R. Sobti and G. Geetha, “Cryptographic Hash Functions: A Review,” vol. 9, no. 2, 2012.
[125] R. C. Merkle, “A Digital Signature Based on a Conventional Encryption Function,” in *Advances in Cryptology — CRYPTO ’87*, C. Pomerance, Ed., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1988, pp. 369–378. doi: 10.1007/3-540-48184-2_32.
[126] D. G. Wood, “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER”.
[127] “A Blockchain Platform for the Enterprise — hyperledger-fabricdocs main documentation.” Accessed: Jul. 19, 2023. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/
[128] C. Komalavalli, D. Saxena, and C. Laroiya, “Chapter 14 - Overview of Blockchain Technology Concepts,” in *Handbook of Research on Blockchain Technology*, S. Krishnan, V. E. Balas, E. G. Julie, Y. H. Robinson, S. Balaji, and R. Kumar, Eds., Academic Press, 2020, pp. 349–371. doi: 10.1016/B978-0-12-819816-2.00014-9.
[129] F. Casino, T. K. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues,” *Telematics and Informatics*, vol. 36, pp. 55–81, Mar. 2019, doi: 10.1016/j.tele.2018.11.006.
[130] L. M. Bach, B. Mihaljevic, and M. Zagar, “Comparative analysis of blockchain consensus algorithms,” in *2018 41st International Convention on Information and*

Communication Technology, Electronics and Microelectronics (MIPRO), May 2018, pp. 1545–1550. doi: 10.23919/MIPRO.2018.8400278.
[131] S. M. H. Bamakan, A. Motavali, and A. Babaei Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Systems with Applications*, vol. 154, p. 113385, Sep. 2020, doi: 10.1016/j.eswa.2020.113385.
[132] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, and E. B. Hamida, "Consortium blockchains: Overview, applications and challenges," *Int. J. Adv. Telecommun*, vol. 11, no. 1, pp. 51–64, 2018.
[133] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and Open Research Challenges," *IEEE Access*, vol. 7, pp. 10127–10149, 2019, doi: 10.1109/ACCESS.2018.2890507.
[134] S. Gupta, J. Hellings, S. Rahnama, and M. Sadoghi, "An In-Depth Look of BFT Consensus in Blockchain: Challenges and Opportunities," in *Proceedings of the 20th International Middleware Conference Tutorials*, Davis CA USA: ACM, Dec. 2019, pp. 6–10. doi: 10.1145/3366625.3369437.
[135] "Home," Solidity Programming Language. Accessed: Jul. 19, 2023. [Online]. Available: https://soliditylang.org/
[136] "Vyper — Vyper documentation." Accessed: Jul. 19, 2023. [Online]. Available: https://docs.vyperlang.org/en/stable/
[137] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, and D. Tigano, "Design Patterns for Gas Optimization in Ethereum," in *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, Feb. 2020, pp. 9–15. doi: 10.1109/IWBOSE50093.2020.9050163.
[138] N. Masla, V. Vyas, J. Gautam, R. N. Shaw, and A. Ghosh, "Reduction in Gas Cost for Blockchain Enabled Smart Contract," in *2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON)*, Sep. 2021, pp. 1–6. doi: 10.1109/GUCON50781.2021.9573701.
[139] F. Spoto, "Enforcing Determinism of Java Smart Contracts," in *Financial Cryptography and Data Security*, M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 568–583. doi: 10.1007/978-3-030-54455-3_40.
[140] S.-Y. Lin, L. Zhang, J. Li, L. Ji, and Y. Sun, "A survey of application research based on blockchain smart contract," *Wireless Netw*, vol. 28, no. 2, pp. 635–690, Feb. 2022, doi: 10.1007/s11276-021-02874-x.
[141] J. Kalajdjieski, M. Raikwar, N. Arsov, G. Velinov, and D. Gligoroski, "Databases fit for blockchain technology: A complete overview," *Blockchain: Research and Applications*, vol. 4, no. 1, p. 100116, Mar. 2023, doi: 10.1016/j.bcra.2022.100116.
[142] Y. Psaras and D. Dias, "The InterPlanetary File System and the Filecoin Network," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, Jun. 2020, pp. 80–80. doi: 10.1109/DSN-S50200.2020.00043.
[143] N. Sangeeta and S. Y. Nam, "Blockchain and Interplanetary File System (IPFS)-Based Data Storage System for Vehicular Networks with Keyword Search Capability," *Electronics*, vol. 12, no. 7, Art. no. 7, Jan. 2023, doi: 10.3390/electronics12071545.
[144] R. G. Brown, "The corda platform: An introduction," *Retrieved*, vol. 27, p. 2018, 2018.
[145] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and Outlook," in *Trust and Trustworthy Computing*, M. Conti, M. Schunter, and I. Askoxylakis, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 163–180. doi: 10.1007/978-3-319-22846-4_10.

[146] J. Sousa, A. Bessani, and M. Vukolic, “A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform,” in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Luxembourg City: IEEE, Jun. 2018, pp. 51–58. doi: 10.1109/DSN.2018.00018.
[147] H. Sukhwani, J. Martinez, X. Chang, K. Trivedi, and A. Rindos, *Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)*. 2017, p. 255. doi: 10.1109/SRDS.2017.36.
[148] A. Beniiche, “A Study of Blockchain Oracles.” arXiv, Jul. 14, 2020. Accessed: May 22, 2023. [Online]. Available: http://arxiv.org/abs/2004.07140
[149] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, “Sidechain technologies in blockchain networks: An examination and state-of-the-art review,” *Journal of Network and Computer Applications*, vol. 149, p. 102471, Jan. 2020, doi: 10.1016/j.jnca.2019.102471.
[150] X. Xu, I. Weber, and M. Staples, “Blockchain Patterns,” in *Architecture for Blockchain Applications*, X. Xu, I. Weber, and M. Staples, Eds., Cham: Springer International Publishing, 2019, pp. 113–148. doi: 10.1007/978-3-030-03035-3_7.
[151] R. Mühlberger *et al.*, “Foundational Oracle Patterns: Connecting Blockchain to the Off-Chain World,” in *Business Process Management: Blockchain and Robotic Process Automation Forum*, A. Asatiani, J. M. García, N. Helander, A. Jiménez-Ramírez, A. Koschmider, J. Mendling, G. Meroni, and H. A. Reijers, Eds., in Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2020, pp. 35–51. doi: 10.1007/978-3-030-58779-6_3.
[152] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, “Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges,” *IEEE Access*, vol. 8, pp. 85675–85685, 2020, doi: 10.1109/ACCESS.2020.2992698.
[153] R. A. Memon, J. P. Li, M. I. Nazeer, A. N. Khan, and J. Ahmed, “DualFog-IoT: Additional Fog Layer for Solving Blockchain Integration Problem in Internet of Things,” *IEEE Access*, vol. 7, pp. 169073–169093, 2019, doi: 10.1109/ACCESS.2019.2952472.
[154] A. A. Sadawi, M. S. Hassan, and M. Ndiaye, “On the Integration of Blockchain With IoT and the Role of Oracle in the Combined System: The Full Picture,” *IEEE Access*, vol. 10, pp. 92532–92558, 2022, doi: 10.1109/ACCESS.2022.3199007.
[155] L. Chen, R. Yuan, and Y. Xia, “Tora: A Trusted Blockchain Oracle Based on a Decentralized TEE Network,” in *2021 IEEE International Conference on Joint Cloud Computing (JCC)*, Aug. 2021, pp. 28–33. doi: 10.1109/JCC53141.2021.00016.
[156] J. Park, H. Kim, G. Kim, and J. Ryou, “Smart Contract Data Feed Framework for Privacy-Preserving Oracle System on Blockchain,” *Computers*, vol. 10, no. 1, Art. no. 1, Jan. 2021, doi: 10.3390/computers10010007.
[157] Richard, M. M. Surya, and A. C. Wibowo, “Converging Artificial Intelligence and Blockchain Technology using Oracle Contract in Ethereum Blockchain Platform,” in *2020 Fifth International Conference on Informatics and Computing (ICIC)*, Nov. 2020, pp. 1–5. doi: 10.1109/ICIC50835.2020.9288611.
[158] C. Zhang, L. Zhu, C. Xu, and K. Sharif, “PRVB: Achieving Privacy-Preserving and Reliable Vehicular Crowdsensing via Blockchain Oracle,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 831–843, Jan. 2021, doi: 10.1109/TVT.2020.3046027.
[159] L. Breidenbach *et al.*, “Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks”.
[160] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, “Astraea: A Decentralized Blockchain Oracle,” in *2018 IEEE International Conference on Internet*

of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Jul. 2018, pp. 1145–1152. doi: 10.1109/Cybermatics_2018.2018.00207.

[161] Y. Lin *et al.*, "A Novel Architecture Combining Oracle With Decentralized Learning for IIoT," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3774–3785, Mar. 2023, doi: 10.1109/JIOT.2022.3150789.

[162] "XuperChain." xuperchain, Jul. 18, 2023. Accessed: Jul. 19, 2023. [Online]. Available: https://github.com/xuperchain/xuperchain

[163] A. E. Fezzazi, A. Adadi, and M. Berrada, "Towards a Blockchain based Intelligent and Secure Voting," in *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Oct. 2021, pp. 1–8. doi: 10.1109/ICDS53782.2021.9626751.

[164] A. Dorri, C. Roulin, R. Jurdak, and S. S. Kanhere, "On the Activity Privacy of Blockchain for IoT," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, Oct. 2019, pp. 258–261. doi: 10.1109/LCN44214.2019.8990819.

[165] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet of Things*, vol. 11, p. 100227, Sep. 2020, doi: 10.1016/j.iot.2020.100227.

[166] A. Fadaeddini, B. Majidi, and M. Eshghi, "Secure decentralized peer-to-peer training of deep neural networks based on distributed ledger technology," *J Supercomput*, vol. 76, no. 12, pp. 10354–10368, Dec. 2020, doi: 10.1007/s11227-020-03251-9.

[167] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study," *Applied Sciences*, vol. 8, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/app8122663.

[168] A. Goel, A. Agarwal, M. Vatsa, R. Singh, and N. Ratha, "DeepRing: Protecting Deep Neural Network With Blockchain," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2019, pp. 2821–2828. doi: 10.1109/CVPRW.2019.00341.

[169] N. K. Bore *et al.*, "Promoting Distributed Trust in Machine Learning and Computational Simulation," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 311–319. doi: 10.1109/BLOC.2019.8751423.

[170] L. S. Shapley, "17. A Value for n-Person Games," in *Contributions to the Theory of Games (AM-28), Volume II*, H. W. Kuhn and A. W. Tucker, Eds., Princeton University Press, 1953, pp. 307–318. doi: 10.1515/9781400881970-018.

[171] H. STACKELBERG, "Theory of the Market Economy (1934), trans," *AT Peacock, London, William Hodge*, 1952.

[172] P. P. Shenoy, "On coalition formation: a game-theoretical approach," *Int J Game Theory*, vol. 8, no. 3, pp. 133–164, Sep. 1979, doi: 10.1007/BF01770064.

[173] S. Béal, S. Ferrières, E. Rémila, and P. Solal, "The proportional Shapley value and applications," *Games and Economic Behavior*, vol. 108, pp. 93–112, Mar. 2018, doi: 10.1016/j.geb.2017.08.010.

[174] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowl Inf Syst*, vol. 41, no. 3, pp. 647–665, Dec. 2014, doi: 10.1007/s10115-013-0679-x.

[175] J. Lemaire, "Cooperative Game Theory and its Insurance Applications," *ASTIN Bulletin: The Journal of the IAA*, vol. 21, no. 1, pp. 17–40, Apr. 1991, doi: 10.2143/AST.21.1.2005399.

[176] J. Castro, D. Gómez, and J. Tejada, “Polynomial calculation of the Shapley value based on sampling,” *Computers & Operations Research*, vol. 36, no. 5, pp. 1726–1730, May 2009, doi: 10.1016/j.cor.2008.04.004.
[177] R. Guha, A. H. Khan, P. K. Singh, R. Sarkar, and D. Bhattacharjee, “CGA: a new feature selection model for visual human action recognition,” *Neural Comput & Applic*, vol. 33, no. 10, pp. 5267–5286, May 2021, doi: 10.1007/s00521-020-05297-5.
[178] R. Patel, M. Garnelo, I. Gemp, C. Dyer, and Y. Bachrach, “Game-theoretic Vocabulary Selection via the Shapley Value and Banzhaf Index,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 2789–2798. doi: 10.18653/v1/2021.naacl-main.223.
[179] A. Ghorbani and J. Zou, “Data Shapley: Equitable Valuation of Data for Machine Learning,” in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 2019, pp. 2242–2251. Accessed: May 16, 2023. [Online]. Available: https://proceedings.mlr.press/v97/ghorbani19c.html
[180] B. Rozemberczki and R. Sarkar, “The Shapley Value of Classifiers in Ensemble Games,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, in CIKM ’21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 1558–1567. doi: 10.1145/3459637.3482302.
[181] X. Chen, S. Li, X. Xu, F. Meng, and W. Cao, “A Novel GSCI-Based Ensemble Approach for Credit Scoring,” *IEEE Access*, vol. 8, pp. 222449–222465, 2020, doi: 10.1109/ACCESS.2020.3043937.
[182] T. Wang, J. Rausch, C. Zhang, R. Jia, and D. Song, “A principled approach to data valuation for federated learning,” *Federated Learning: Privacy and Incentive*, pp. 153–167, 2020.
[183] H. Ykhlef and D. Bouchaffra, “Induced Subgraph Game for Ensemble Selection,” *Int. J. Artif. Intell. Tools*, vol. 26, no. 01, p. 1760003, Feb. 2017, doi: 10.1142/S021821301760003X.
[184] M. Shen, J. Duan, L. Zhu, J. Zhang, X. Du, and M. Guizani, “Blockchain-Based Incentives for Secure and Collaborative Data Sharing in Multiple Clouds,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1229–1241, Jun. 2020, doi: 10.1109/JSAC.2020.2986619.
[185] L. Zhu, H. Dong, M. Shen, and K. Gai, “An Incentive Mechanism Using Shapley Value for Blockchain-Based Medical Data Sharing,” in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, May 2019, pp. 113–118. doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00030.
[186] Z. Chen, X. Sun, X. Shan, and J. Zhang, “Decentralized Mining Pool Games in Blockchain,” in *2020 IEEE International Conference on Knowledge Graph (ICKG)*, Aug. 2020, pp. 426–432. doi: 10.1109/ICBK50248.2020.00067.
[187] Z. Chang, W. Guo, X. Guo, Z. Zhou, and T. Ristaniemi, “Incentive Mechanism for Edge-Computing-Based Blockchain,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7105–7114, Nov. 2020, doi: 10.1109/TII.2020.2973248.
[188] X. Ding, J. Guo, D. Li, and W. Wu, “An Incentive Mechanism for Building a Secure Blockchain-Based Internet of Things,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 477–487, Jan. 2021, doi: 10.1109/TNSE.2020.3040446.
[189] Y. He, H. Li, X. Cheng, Y. Liu, C. Yang, and L. Sun, “A Blockchain Based Truthful Incentive Mechanism for Distributed P2P Applications,” *IEEE Access*, vol. 6, pp. 27324–27335, 2018, doi: 10.1109/ACCESS.2018.2821705.

[190] S. Xuan *et al.*, “An incentive mechanism for data sharing based on blockchain with smart contracts,” *Computers & Electrical Engineering*, vol. 83, p. 106587, May 2020, doi: 10.1016/j.compeleceng.2020.106587.
[191] J. Qiu, X. Liang, S. Shetty, and D. Bowden, “Towards Secure and Smart Healthcare in Smart Cities Using Blockchain,” in *2018 IEEE International Smart Cities Conference (ISC2)*, Sep. 2018, pp. 1–4. doi: 10.1109/ISC2.2018.8656914.
[192] V. Drungilas, E. Vaičiukynas, L. Ablonskis, and L. Čeponienė, “Shapley Values as a Strategy for Ensemble Weights Estimation,” *Applied Sciences*, vol. 13, no. 12, Art. no. 12, Jan. 2023, doi: 10.3390/app13127010.
[193] V. Drungilas, E. Vaičiukynas, and L. Čeponienė, “Towards Collaborative Privacy-preserving Machine Learning on Private Blockchain,” in *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2023, pp. 1–4. doi: 10.23919/CISTI58278.2023.10211561.
[194] V. Drungilas, E. Vaičiukynas, M. Jurgelaitis, R. Butkienė, and L. Čeponienė, “Towards Blockchain-Based Federated Machine Learning: Smart Contract for Model Inference,” *Applied Sciences*, vol. 11, no. 3, Art. no. 3, Jan. 2021, doi: 10.3390/app11031010.
[195] E. Gordon-Rodriguez, G. Loaiza-Ganem, G. Pleiss, and J. P. Cunningham, “Uses and abuses of the cross-entropy loss: Case studies in modern deep learning,” 2020.
[196] A. Rényi, “On measures of entropy and information,” in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, University of California Press, 1961, pp. 547–562.
[197] Z. Zhang and M. Sabuncu, “Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2018. Accessed: Dec. 16, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html
[198] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU).” arXiv, Feb. 07, 2019. doi: 10.48550/arXiv.1803.08375.
[199] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.
[200] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, “A Comparison of Decision Tree Ensemble Creation Techniques,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 173–180, Jan. 2007, doi: 10.1109/TPAMI.2007.250609.
[201] A. Kim, K. Oh, J.-Y. Jung, and B. Kim, “Imbalanced classification of manufacturing quality conditions using cost-sensitive decision tree ensembles,” *International Journal of Computer Integrated Manufacturing*, vol. 31, no. 8, pp. 701–717, Aug. 2018, doi: 10.1080/0951192X.2017.1407447.
[202] H. Parvin, M. MirnabiBaboli, and H. Alinejad-Rokny, “Proposing a classifier ensemble framework based on classifier selection and decision tree,” *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 34–42, Jan. 2015, doi: 10.1016/j.engappai.2014.08.005.
[203] J. W. Osborne, *Best Practices in Quantitative Methods*. SAGE, 2008.
[204] “The Go Programming Language Specification - The Go Programming Language.” Accessed: Jul. 17, 2023. [Online]. Available: https://go.dev/ref/spec
[205] “template package - html/template - Go Packages.” Accessed: Jul. 17, 2023. [Online]. Available: https://pkg.go.dev/html/template
[206] “fabric-contract-api-go/tutorials/getting-started.md at main · hyperledger/fabric-contract-api-go · GitHub.” Accessed: Jul. 17, 2023. [Online]. Available:

https://github.com/hyperledger/fabric-contract-api-go/blob/main/tutorials/getting-started.md
[207] Y. Shafranovich, “Common Format and MIME Type for Comma-Separated Values (CSV) Files,” Internet Engineering Task Force, Request for Comments RFC 4180, Oct. 2005. doi: 10.17487/RFC4180.
[208] T. Bray, “The javascript object notation (json) data interchange format,” 2014.
[209] V. Drungilas, E. Vaičiukynas, L. Ablonskis, and L. Čeponienė, “Heterogeneous Models Inference Using Hyperledger Fabric Oracles,” *Edited by Sergey Y. Yurish*, p. 83, 2022.
[210] “keras-io/examples/vision/knowledge_distillation.py at master · keras-team/keras-io,” GitHub. Accessed: Aug. 02, 2023. [Online]. Available: https://github.com/keras-team/keras-io/blob/master/examples/vision/knowledge_distillation.py
[211] S. O. Arik and T. Pfister, “TabNet: Attentive Interpretable Tabular Learning.” arXiv, Dec. 09, 2020. Accessed: Dec. 03, 2023. [Online]. Available: http://arxiv.org/abs/1908.07442
[212] “google-research/tabnet at master · google-research/google-research,” GitHub. Accessed: Dec. 03, 2023. [Online]. Available: https://github.com/google-research/google-research/tree/master/tabnet
[213] “sklearn.datasets.make_moons,” scikit-learn. Accessed: Aug. 06, 2023. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.datasets.make_moons.html
[214] D. Dua and C. Graff, *UCI machine learning repository. 2017. University of California, Irvine, School of Information and Computer Sciences*. 2017.
[215] P. Cortez, M. S. R. Laureano, and S. Moro, “Using data mining for banking direct marketing: An application of the crisp-dm methodology,” *Portugal: Institute University of Lisbon*, 2010.
[216] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, “Algorithm selection on data streams,” in *Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings 17*, Springer, 2014, pp. 325–336.
[217] H. I. FAWAZ, “Critical Difference Diagrams.” Aug. 05, 2023. Accessed: Aug. 06, 2023. [Online]. Available: https://github.com/hfawaz/cd-diagram
[218] M. Friedman, “A comparison of alternative tests of significance for the problem of m rankings,” *The annals of mathematical statistics*, vol. 11, no. 1, pp. 86–92, 1940.
[219] A. Benavoli, G. Corani, and F. Mangili, “Should we really use post-hoc tests based on mean-ranks?,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 152–161, 2016.
[220] F. Wilcoxon, “Individual comparisons of grouped data by ranking methods,” *Journal of economic entomology*, vol. 39, no. 2, pp. 269–270, 1946.
[221] S. Holm, “A simple sequentially rejective multiple test procedure,” *Scandinavian journal of statistics*, pp. 65–70, 1979.
[222] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and A. Rogers, “Bounding the estimation error of sampling-based Shapley value approximation,” *arXiv preprint arXiv:1306.4265*, 2013.
[223] G. Friedland, A. Metere, and M. Krell, “A Practical Approach to Sizing Neural Networks.” arXiv, Oct. 04, 2018. Accessed: Mar. 28, 2024. [Online]. Available: http://arxiv.org/abs/1810.02328
[224] M. Abadi *et al.*, “TensorFlow, Large-scale machine learning on heterogeneous systems.” Nov. 2015. doi: 10.5281/zenodo.4724125.
[225] B. Schäfl, L. Gruber, A. Bitto-Nemling, and S. Hochreiter, “Hopular: Modern Hopfield Networks for Tabular Data.” arXiv, Jun. 01, 2022. doi: 10.48550/arXiv.2206.00664.

[226] A. Dubey, F. Radenovic, and D. Mahajan, “Scalable Interpretability via Polynomials.” arXiv, Oct. 18, 2022. Accessed: Mar. 29, 2024. [Online]. Available: http://arxiv.org/abs/2205.14108
[227] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

SCIENTIFIC JOURNAL AND CONFERENCE PUBLICATIONS

Scientific Publications in Periodicals

DRUNGILAS, Vaidotas; VAIČIUKYNAS, Evaldas; ABLONSKIS, Linas; ČEPONIENĖ, Lina. Shapley values as a strategy for ensemble weights estimation. *Applied Science*. Basel: MDPI, 2023, vol. 13, iss. 12, art. No. 7010, pp. 1–23. ISSN 2076-3417. doi: 10.3390/app13127010

DRUNGILAS, Vaidotas; VAIČIUKYNAS, Evaldas; JURGELAITIS, Mantas; BUTKIENĖ, Rita; ČEPONIENĖ, Lina. Towards blockchain-based federated machine learning: smart contract for model inference. *Applied Sciences*. Basel: MDPI, 2021, vol. 11, iss. 3, art. No. 1010, pp. 1–21. ISSN 2076-3417. doi: 10.3390/app11031010

Scientific Publications in Conference Proceedings

DRUNGILAS, V.; VAIČIUKYNAS, E; ČEPONIENĖ, L.; Towards Collaborative Privacy-preserving Machine Learning on Private Blockchain In: *Proceedings of* CISTI'2023 – 18th Iberian Conference on Information Systems and Technologies 20–23 June 2023, Aveiro, Portugal.

DRUNGILAS, V.; VAIČIUKYNAS, E.; ABLONSKIS, L.; ČEPONIENĖ, L. Heterogeneous models inference using hyperledger fabric oracles. In: *Proceedings of the 1st blockchain and cryptocurrency conference (B2C' 2022), 9–11 November 2022, Barcelona, Spain*. Edited by Sergey Y. Yurish. Barcelona: IFSA publishing, 2022, (027), pp. 83–85. eISBN 9788409457632.

JURGELAITIS, Mantas; BUTKIENĖ, Rita; VAIČIUKYNAS, Evaldas; **DRUNGILAS, Vaidotas;** ČEPONIENĖ, Lina. Modelling principles for blockchain-based implementation of business or scientific processes. In: *CEUR workshop proceedings: IVUS 2019 international conference on information technologies: proceedings of the international conference on information technologies, Kaunas, Lithuania, April 25, 2019*. Edited by: Robertas Damaševičius, Tomas Krilavičius, Audrius Lopata, Dawid Połap, Marcin Woźniak. Aachen: CEUR-WS, 2019, vol. 2470, pp. 43–47. ISSN 1613-0073.

CURRICULUM VITAE

Vaidotas Drungilas
vaidotas.drungilas@ktu.lt

Education:

2012 – 2016	Information Systems Bachelor's studies – Kaunas University of Technology
2016 – 2018	Information System Engineering Master's studies – Kaunas University of Technology
2018 – Currently	Informatics Engineering PhD studies – Kaunas University of Technology

Professional experience:

01 Aug 2014 – 19 Sept 2017	Junior Laboratory Assistant
01 Feb 2016 – 30 Aug 2019	IT Engineer
12 Sept 2016 – 02 Sept 2018	Academic Assistant
20 Sept 2017 – 02 Feb 2020	Senior Laboratory Assistant
03 Sept 2018 – 31 Dec 2023	Lecturer
03 Feb 2020 – 31 Dec 2020	Senior Engineer
01 Jan 2021 – Currently	IT Infrastructure Engineer
01 Jan 2024 – Currently	Junior Assistant

Scientific interests:

Machine learning, distributed ledger technologies, UML modelling and reverse engineering, software development.

Scientific Publications:

1. DRUNGILAS, Vaidotas; VAIČIUKYNAS, Evaldas; ABLONSKIS, Linas; ČEPONIENĖ, Lina. Shapley values as a strategy for ensemble weights estimation. *Applied Science*. Basel: MDPI, 2023, vol. 13, iss. 12, art. No. 7010, pp. 1–23. ISSN 2076-3417. doi: 10.3390/app13127010

2. DRUNGILAS, Vaidotas; VAIČIUKYNAS, Evaldas; JURGELAITIS, Mantas; BUTKIENĖ, Rita; ČEPONIENĖ, Lina. Towards blockchain-based federated machine learning: smart contract for model inference. *Applied Sciences*. Basel: MDPI, 2021, vol. 11, iss. 3, art. No. 1010, pp. 1–21. ISSN 2076-3417. doi: 10.3390/app11031010

Scientific Conferences:

1. DRUNGILAS, V.; VAIČIUKYNAS, E.; ČEPONIENĖ, L.; Towards Collaborative Privacy-preserving Machine Learning on Private Blockchain. In: *Proceedings of CISTI'2023* – 18th Iberian Conference on Information Systems and Technologies; 20–23 June 2023, Aveiro, Portugal.

2. DRUNGILAS, V.; VAIČIUKYNAS, E.; ABLONSKIS, L.; ČEPONIENĖ, L. Heterogeneous models inference using hyperledger fabric oracles. In: *Proceedings of the 1st blockchain and cryptocurrency conference (B2C' 2022), 9–11 November 2022, Barcelona, Spain*. Edited by Sergey Y. Yurish. Barcelona: IFSA publishing, 2022, (027), pp. 83–85. eISBN 9788409457632.

3. JURGELAITIS, Mantas; BUTKIENĖ, Rita; VAIČIUKYNAS, Evaldas; DRUNGILAS, Vaidotas; ČEPONIENĖ, Lina. Modelling principles for blockchain-based implementation of business or scientific processes. In: *CEUR workshop proceedings: IVUS 2019 international conference on information technologies: proceedings of the international conference on information technologies, Kaunas, Lithuania, 25 April 2019*. Edited by: Robertas Damaševičius, Tomas Krilavičius, Audrius Lopata, Dawid Połap, Marcin Woźniak. Aachen: CEUR-WS, 2019, vol. 2470, pp. 43–47. ISSN 1613-0073.

ACKNOWLEDGMENTS

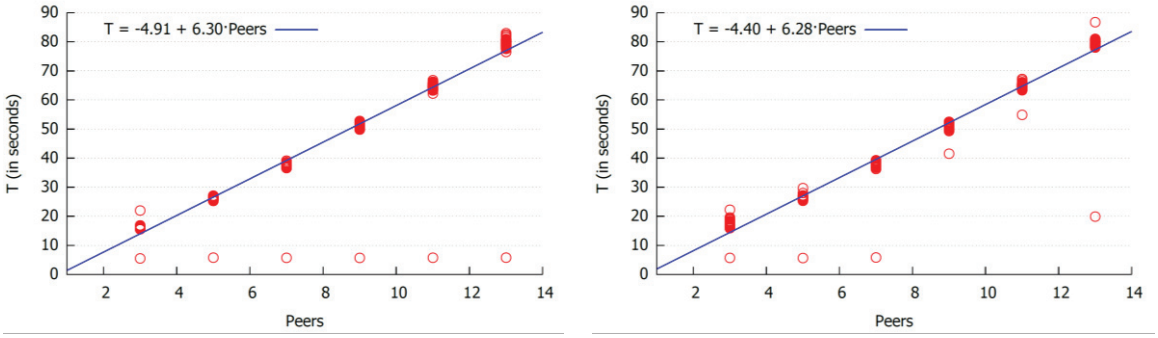
This work would not have been possible without the care and encouragement from my colleagues of the Department of Information Systems and continuous support from my family.

Thank you for being there when I needed it most.

Vaidotas Drungilas

APPENDIXES

Appendix A. Extended result analysis for Performance Evaluation of Model Inference via Local Off-chain Blockchain Oracles experiment



(a) smart contract (b) local off-chain oracle service

Figure A1. Model inference calculation time for the synthetic dataset with 32768 instances: smart contract (a) and oracle service (b) results. The increase in runtime is linear with an increase in calculation time of 6.30 s for the smart contract and 6.28s for the oracle service in a simulated blockchain network environment.

Table A1. Statistical analysis for smart contract implementations performance results synthetic dataset case. Dependent variable: T

	Coefficient	*Std. Error*	*t-ratio*	*p-value*	
const	−4.90818	0.503859	−9.741	<0.0001	***
Peers	6.29927	0.0579238	108.8	<0.0001	***

Mean dependent var	45.48601		S.D. dependent var	22.07177
Sum squared resid	14044.71		S.E. of regression	4.846251
R-squared	0.951870		Adjusted R-squared	0.951790
F(1, 598)	11826.80		P-value(F)	0.000000
Log-likelihood	−1797.285		Akaike criterion	3598.569
Schwarz criterion	3607.363		Hannan-Quinn	3601.993

Table A2. Correlation matrix for smart contract implementations performance synthetic dataset case. 5% critical value (two-tailed) = 0.0801 for n = 600. Dependent variable: T

T	Peers	
1.0000	0.9756	T
	1.0000	Peers

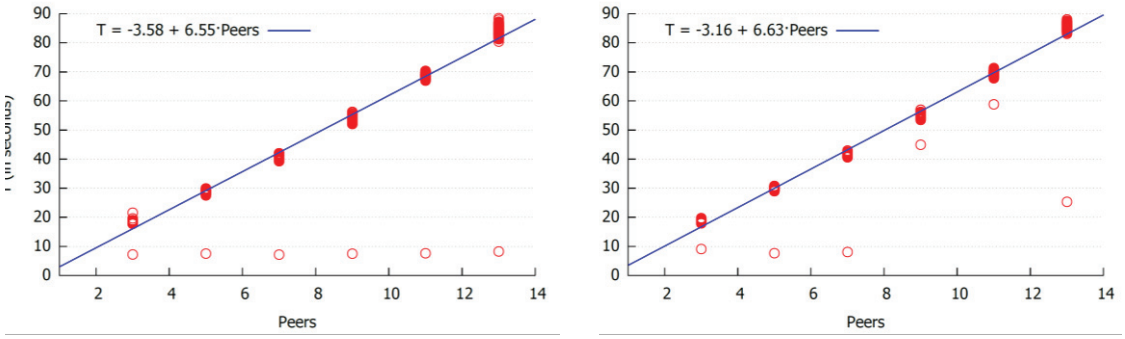
Table A3. Statistical analysis for local off-chain oracle service implementations performance results synthetic dataset case. Dependent variable: T

	Coefficient	*Std. Error*	*t-ratio*	*p-value*	
const	−4.39957	0.357455	−12.31	<0.0001	***
Peers	6.28314	0.0410931	152.9	<0.0001	***

Mean dependent var	45.86557	S.D. dependent var	21.75189
Sum squared resid	7068.662	S.E. of regression	3.438096
R-squared	0.975059	Adjusted R-squared	0.975017
F(1, 598)	23378.45	P-value(F)	0.000000
Log-likelihood	−1591.312	Akaike criterion	3186.624
Schwarz criterion	3195.418	Hannan-Quinn	3190.048

Table A4. Correlation matrix for local off-chain oracle service implementations performance synthetic dataset case. 5% critical value (two-tailed) = 0.0801 for n = 600. Dependent variable: T

T	Peers	
1.0000	0.9875	T
	1.0000	Peers



(a) smart contract (b) local off-chain oracle service

Figure A2. Model inference calculation time in relation to peer count for EEG eye state dataset with 32768 instances: smart contract (a) and oracle service (b) results. The increase in the runtime is linear with an increase in the calculation time of 6.55 s for the smart contract and 6.63s for the oracle service in a simulated blockchain network environment.

Table A5. Statistical analysis for smart contract implementations performance results EEG eye state dataset case. Dependent variable: T

	Coefficient	*Std. Error*	*t-ratio*	*p-value*	
const	−3.57527	0.528231	−6.768	<0.0001	***
Peers	6.54810	0.0607256	107.8	<0.0001	***

Mean dependent var	48.80953	S.D. dependent var	22.95308
Sum squared resid	15436.27	S.E. of regression	5.080666
R-squared	0.951086	Adjusted R-squared	0.951004
F(1, 598)	11627.52	P-value(F)	0.000000
Log-likelihood	−1825.627	Akaike criterion	3655.254
Schwarz criterion	3664.048	Hannan-Quinn	3658.677

Table A6. Correlation matrix for smart contract implementations performance synthetic dataset case. 5% critical value (two-tailed) = 0.0801 for n = 600. Dependent variable: T

T	Peers	
1.0000	0.9752	T
	1.0000	Peers

Table A7. Statistical analysis for local off-chain oracle service implementations performance results EEG eye state dataset case. Dependent variable: T

	Coefficient	*Std. Error*	*t-ratio*	*p-value*	
const	−3.15881	0.361331	−8.742	<0.0001	***
Peers	6.62539	0.0415386	159.5	<0.0001	***

Mean dependent var	49.84428	S.D. dependent var	22.91354
Sum squared resid	7222.773	S.E. of regression	3.475373
R-squared	0.977034	Adjusted R-squared	0.976995
F(1, 598)	25440.04	P-value(F)	0.000000
Log-likelihood	−1597.782	Akaike criterion	3199.565
Schwarz criterion	3208.359	Hannan-Quinn	3202.988

Table A8. Correlation matrix for local off-chain oracle service implementations performance synthetic dataset case. 5% critical value (two-tailed) = 0.0801 for n = 600. Dependent variable: T

T	Peers	
1.0000	0.9885	T
	1.0000	Peers

Appendix B. Shapley-based ensemble weighting strategies performance evaluation experiment results for every tested configuration setting. Presented in median BCE values

Table B1. Results presented in BCE for homogeneous decision tree ensembles developed by using the BNG_credit-a dataset

Programming language	Ensemble size / Weighting	2	3	5	8	13
Python	Equal	0.323	0.329	0.322	0.319	0.319
	Mono	0.338	0.339	0.339	0.339	0.339
	Perf	0.323	0.328	0.322	0.319	0.318
	posShap	0.322	0.328	0.320	0.318	0.317
	Rand	0.329	0.334	0.322	0.320	0.319
	Roz	0.328	0.328	0.323	0.320	0.318
R	Equal	0.408	0.407	0.385	0.378	0.368
	Mono	0.408	0.408	0.408	0.408	0.408
	Perf	0.408	0.407	0.384	0.376	0.367
	posShap	0.408	0.407	0.379	0.370	0.360
	Rand	0.408	0.407	0.389	0.380	0.371
	Roz	0.407	0.402	0.393	0.379	0.369

Table B2. Results presented in BCE for homogeneous logistic regression ensembles developed by using the Bank Marketing dataset

Programming language	Ensemble size / Weighting	2	3	5	8	13
Python	Equal	0.239	0.241	0.240	0.239	0.243
	maxShap	0.239	0.240	0.239	0.270	0.416
	Mono	0.239	0.240	0.239	0.239	0.240
	Perf	0.239	0.241	0.239	0.238	0.237
	posShap	0.239	0.240	0.239	0.238	0.238
	Rand	0.240	0.241	0.240	0.240	0.245
	Roz	0.240	0.240	0.240	0.239	0.240
R	Equal	0.241	0.240	0.240	0.240	0.246
	maxShap	0.241	0.239	0.240	0.288	0.441
	Mono	0.241	0.239	0.240	0.24	0.24
	Perf	0.241	0.240	0.240	0.239	0.236
	posShap	0.241	0.239	0.240	0.239	0.238
	Rand	0.241	0.240	0.241	0.241	0.247
	Roz	0.240	0.240	0.239	0.239	0.243

Table B3. Results presented in BCE for homogeneous logistic regression ensembles developed by using the BNG_credit-a dataset

Programming language	Ensemble size / Weighting	2	3	5	8	13
Python	Equal	0.327	0.326	0.326	0.326	0.326
	Mono	0.326	0.326	0.326	0.326	0.326
	Perf	0.327	0.326	0.326	0.326	0.326
	posShap	0.327	0.326	0.326	0.326	0.326
	Rand	0.327	0.326	0.327	0.326	0.327
	Roz	0.327	0.327	0.327	0.327	0.327
R	Equal	0.327	0.327	0.327	0.327	0.327
	Mono	0.327	0.327	0.326	0.327	0.327
	Perf	0.327	0.327	0.327	0.327	0.327
	posShap	0.327	0.327	0.327	0.327	0.327
	Rand	0.327	0.327	0.327	0.327	0.327
	Roz	0.327	0.327	0.327	0.327	0.327

Table B4. Results presented in BCE for homogeneous decision tree ensembles developed by using the Bank Marketing dataset

Programming language	Ensemble size / Weighting	2	3	5	8	13
Python	Equal	0.260	0.260	0.256	0.252	0.289
	maxShap	0.261	0.275	0.325	0.344	0.366
	Mono	0.272	0.273	0.272	0.270	0.282
	Perf	0.260	0.259	0.254	0.251	0.287
	posShap	0.260	0.261	0.257	0.254	0.274
	Rand	0.263	0.262	0.258	0.256	0.289
	Roz	0.261	0.261	0.259	0.253	0.293
R	Equal	0.280	0.278	0.269	0.265	0.266
	maxShap	0.280	0.278	0.272	0.281	0.291
	Mono	0.282	0.283	0.282	0.282	0.283
	Perf	0.280	0.278	0.269	0.266	0.267
	posShap	0.280	0.278	0.270	0.266	0.264
	Rand	0.280	0.279	0.271	0.267	0.268
	Roz	0.279	0.276	0.275	0.265	0.268

Table B5. Results presented in BCE for heterogeneous ensembles trained developed by using the BNG_credit-a dataset

Programming language	Ensemble size / Weighting	4	6	10	16
Python	Equal	0.313	0.318	0.314	0.313
	maxShap	0.313	0.318	0.313	0.312
	Mono	0.326	0.326	0.326	0.326
	Perf	0.313	0.318	0.314	0.313
	posShap	0.313	0.318	0.313	0.312
	Rand	0.316	0.318	0.314	0.313
	Roz	0.315	0.316	0.314	0.313
R	Equal	0.348	0.348	0.342	0.339
	maxShap	0.348	0.343	0.341	0.341
	Mono	0.327	0.326	0.326	0.326
	Perf	0.348	0.347	0.342	0.340
	posShap	0.348	0.343	0.341	0.341
	Rand	0.348	0.346	0.342	0.339
	Roz	0.355	0.352	0.348	0.344

Table B6. Results presented in BCE of heterogeneous ensembles developed by using the Bank Marketing dataset

Programming language	Ensemble size / Weighting	4	6	10	16
Python	Equal	0.236	0.237	0.235	0.233
	maxShap	0.239	0.244	0.260	0.269
	Mono	0.239	0.240	0.239	0.239
	Perf	0.237	0.239	0.238	0.237
	posShap	0.239	0.240	0.238	0.235
	Rand	0.237	0.238	0.236	0.235
	Roz	0.238	0.241	0.238	0.236
R	Equal	0.236	0.237	0.235	0.233
	maxShap	0.239	0.244	0.260	0.269
	Mono	0.241	0.239	0.239	0.24
	Perf	0.237	0.239	0.238	0.237
	posShap	0.239	0.240	0.238	0.235
	Rand	0.237	0.238	0.236	0.235
	Roz	0.250	0.249	0.245	0.241

Appendix C. Source code for the developed blockchain solutions and experiment procedures

https://github.com/HurrisLT

UDK 004.85+004.65+004.78](043.3)
SL344. 2024-05-09, 23,25 leidyb. apsk. l. Tiražas 14 egz. Užsakymas 84.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas