UNIVERSITY OF PIRAEUS
DEPARTMENT OF DIGITAL SYSTEMS
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Causal and Contextual Information Extraction and Data Enrichment Methods with Emphasis on Artificial Intelligence-driven Optimization Algorithms for Cloud and Edge Computing

Chrysostomos G. Symvoulidis

Doctoral Thesis

Piraeus, 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

Μέθοδοι Ενίσχυσης Συνόλων Δεδομένων βάσει Εξαγωγής Αιτιακών Χαρακτηριστικών και Χαρακτηριστικών Πλαίσιου Αναφοράς με Έμφαση σε αλγόριθμους Τεχνητής Νοημοσύνης για τη Βελτιστοποίηση Υπολογιστικών Υποδομών Cloud και Edge

Χρυσόστομος Γ. Συμβουλίδης

Διδακτορική Διατριβή

Πειραιάς, 2024

UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

Causal and Contextual Information Extraction and Data Enrichment Methods with Emphasis on Artificial Intelligence-driven Optimization Algorithms for Cloud and Edge Computing

Doctoral Thesis Presented

by **Chrysostomos G. Symvoulidis**

in Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

ADVISORY COMMITTEE:

Professor Dimosthenis Kyriazis

Professor Apostolos Milionis

Professor Nikitas-Marinos Sgouros

UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

Causal and Contextual Information Extraction and Data Enrichment Methods with Emphasis on Artificial Intelligence-driven Optimization Algorithms for Cloud and Edge Computing

Doctoral Thesis Presented

by **Chrysostomos G. Symvoulidis**

in Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

APPROVED BY:

Dimosthenis Kyriazis, Professor, University of Piraeus

Apostolos Milionis, Professor, University of Piraeus

Nikitas-Marinos Sgouros, Professor, University of Piraeus

Christos Doulkeridis, Associate Professor, University of Piraeus

---

Michail Filippakis, Professor, University of Piraeus

Andriana Prentza, Professor, University of Piraeus

Georgios Kousiouris, Associate Professor, Harokopio University of Athens

Piraeus, 2024

To my grandmother Amalia and my grandfather Giorgos.

Acknowledgments

The work presented in this thesis would not have been made possible if not for some people. Therefore, this chapter is dedicated to them.

First and foremost, I would like to thank my supervisor Prof. Dimosthenis Kyriazis for his trust and guidance over the years, beginning when I was still a student and continuing until now. We got to know each other for over a decade and all I can say is that he has been a true mentor to me. In addition, I would also like to thank my advisory board member for their help over the years, Prof. Apostolos Milionis and Prof. Nikitas-Marinos Sgouros and their valuable input especially during the first years of my journey. Additionally, I would like to express my sincere gratitude to Prof. Michael Filippakis for his support over the years. Their expertise, constructive feedback, and mentorship have all been instrumental in shaping my research journey and academic growth.

Next, I would like to thank my family. I would like to thank specifically my mother Maria, my father Giorgos my sister Amalia, and Toula, for their support and tolerance towards me over the years. I know that I can sometimes be difficult to cope with, but they did a great job not showing their irritation towards me, which I am sure was not easy. Another special thanks goes to the rest of my family members, my aunts and uncles Alfi, Takis, Gianna, Sakis and Leta, my grandmother Athanasia, and my cousins Amalia, Konstantinos, Antonis, Giorgos, and Spyros.

Pursuing a PhD is no easy task. Several times, it requires having to miss out on many things, such as going out, playing games on the PC, going on vacation, or going to bed at a regular time, like a normal human being. The most difficult, yet

fun, thing I had to deal with though, was understanding these long papers full of mathematical formulas and weird symbols, and beyond that, having to think of ways a research can be expanded or identify gaps in the literature in order to give birth to an innovative, new idea. If it weren't for the people I am going to mention in the following lines, the discussions, and the waterfall thinking sessions I had with them, I am sure that I wouldn't have understood at least half of the papers I read over the years. Therefore (hoping I don't forget anyone), I need to thank my lab partners Argiro, Thanos, the two Georges, Ilias, Jean-Didier, Pepi and Giannis. The discussions we had and the time we spent together are invaluable to me.

A special spot is saved for my closest friends. As "cliché" as it may sound, these people are the family I chose. Therefore I thank them for being there for me all my life. Some of them I met while I was a little kid, and some of them I met when I was older. Regardless, I would like to sincerely thank them and to express how happy I am to have met them and for being there for me. These are my two best friends, Charis and Thodoris (in alphabetical order so that Thodoris won't get angry for being second), and my closest friends Andreas, Dimitra, Elli, Ioanna, Mary, and Tasos, along with two of the most lovely computer scientists one could ever meet Antonis and Takis.

The biggest thanks, I want to express is to my partner Alexandra. She, out of all people, is the one who has put up with me the most, and for some inconceivable reason, she still loves me and continues to be with me. For this and a whole lot more reasons, I want to say a big thank you to her.

Finally, although I know that he won't ever read this thesis, I need to thank my cat, Sotiris. He is my *son* and as a cat dad, I can only be proud of him, even though he hates everything alive, and I am pretty sure that he has compiled an elaborate scheme to destroy the whole world, just to be left alone. However, while I was writing this thesis, he was always sitting on my lap, purring, and trying to do his best to help me finish this difficult task.

Abstract

Keywords: causal discovery, context awareness, causal features generation, influence identification, cloud computing, edge computing, machine learning, deep learning

Causal Discovery and Context Awareness are central subjects in research for many years. With its origins deeply rooted in scientific research, statistical inference, and philosophical questions, causality has thrilled researchers due to its key role in revealing the relationships between variables. Its significance lies in the pursuit of understanding of not just what happens, but why it happens, and thus making it vital in many scientific fields. Context awareness on the other hand, refers to the ability of a system to understand the context and what are the factors that affect the environment in which it is being used. That being said, Causal Discovery and Context Awareness are of crucial importance when it comes to the development of Artificial Intelligence systems that can be more accurate, robust and generalizable. This thesis focuses on the design and implementation of two Contextual and Causal Extraction methods which can be used for the enrichment of datasets towards the improvement of Machine Learning models. The first method identifies the most influential instances in a dataset and utilizes them in order to generate an Influence-based dataset. The second method discovers the causal relationships that may exist in a dataset and afterwards utilizes this information to generate causal features in order to incorporate this information to the initial

dataset. In order to evaluate the methods' performance they have been applied in several, diverse frameworks and strategies used for the optimization of Cloud and Edge Computing environments. The results show that the methods can effectively extract contextual and causal information from datasets. Furthermore, the evaluation proves that the utilization of this information can significantly improve the performance of Machine Learning models. This thesis also provides potential directions for future research that could build upon the findings of the current thesis.

Περίληψη

Λέξεις κλειδιά: ανακάλυψη αιτιότητας, επίγνωση πλαισίου αναφοράς, δημιουργία αιτιακών χαρακτηριστικών, αναγνώριση επιρροής, υπολογιστικά νέφη, υπολογιστική επεξεργασία στη περιφέρεια, μηχανική μάθηση, βαθειά μάθηση

Τα πεδία Ανακάλυψης Αιτιότητας (Causal Discovery) και Επίγνωσης Πλαισίου Αναφοράς (Context Awareness) αποτελούν κεντρικά θέματα στην έρευνα εδώ και πολλά χρόνια. Με τις ρίζες της βαθιά ριζωμένες στην επιστημονική έρευνα, στα στατιστικά συμπεράσματα και σε φιλοσοφικά ερωτήματα, η Ανακάλυψη Αιτιότητας έχει ενθουσιάσει τους ερευνητές λόγω του βασικού της ρόλου στην αποκάλυψη των σχέσεων μεταξύ των μεταβλητών. Η σημασία της, έγκειται στο ότι δεν αρκείται μόνο στη κατανόηση του τι συμβαίνει, αλλά και τους λόγους για τους οποίους συμβαίνει αυτό, γεγονός που τη καθιστά απαραίτητη σε πολλά επιστημονικά πεδία. Η Επίγνωση Πλαισίου Αναφοράς από την άλλη πλευρά, αναφέρεται στην ικανότητα ενός συστήματος να κατανοεί το περιβάλλον στο οποίο χρησιμοποιείται καθώς και ποιοι είναι οι παράγοντες που το επηρεάζουν. Δεδομένων όλων των παραπάνω, τόσο η Ανακάλυψη Αιτιότητας, όσο και η Επίγνωση Πλαισίου Αναφοράς αποτελούν θέματα ζωτικής σημασίας όταν αφορούν στην ανάπτυξη συστημάτων Τεχνητής Νοημοσύνης που μπορούν να είναι πιο ακριβή, ισχυρά και γενικεύσιμα. Η παρούσα διατριβή εστιάζει στο σχεδιασμό, την υλοποίηση και την εφαρμογή δύο μεθόδων Εξαγωγής Αιτιότητας και δεδομένων σχετικών με το Πλαίσιο Α-

ναφοράς που μπορούν να χρησιμοποιηθούν για τον εμπλουτισμό των συνόλων δεδομένων με τελικό σκοπό τη βελτίωση μοντέλων Μηχανικής Μάθησης. Η πρώτη μέθοδος αναγνωρίζει τις παρατηρήσεις με τη μεγαλύτερη επιρροή σε ένα σύνολο δεδομένων με σκοπό τη δημιουργία ενός συνόλου δεδομένων βασισμένο σε αυτές. Η δεύτερη μέθοδος ανακαλύπτει τις αιτιώδεις σχέσεις που μπορεί να υπάρχουν σε ένα σύνολο δεδομένων και στη συνέχεια χρησιμοποιεί αυτές τις πληροφορίες για να δημιουργήσει αιτιακά χαρακτηριστικά τα οποία τελικά ενσωματώνονται στο αρχικό σύνολο δεδομένων. Προκειμένου να αξιολογηθεί η απόδοση των μεθόδων, έχουν εφαρμοστεί σε διαφορετικά πλαίσια και στρατηγικές που χρησιμοποιούνται για τη βελτιστοποίηση περιβάλλοντων Υπολογιστικών Νεφών (Cloud Computing) και Υπολογιστικής Επεξεργασίας στη Περιφέρεια (Edge Computing). Τα αποτελέσματα δείχνουν ότι οι μέθοδοι μπορούν να εξάγουν αποτελεσματικά πληροφορίες σχετικά με το Πλαίσιο Αναφοράς και τις Αιτιακές σχέσεις που υπάρχουν. Ακόμη αποδεικνύεται ότι η αξιοποίηση αυτών των πληροφοριών μπορεί να βελτιώσει σημαντικά την απόδοση των αλγορίθμων Μηχανικής Μάθησης. Τέλος, η παρούσα παρουσιάζει πιθανές κατευθύνσεις για μελλοντική έρευνα που θα μπορούσε να βασιστεί στα ευρήματα της τρέχουσας διατριβής.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ABA Attribute-Based Authorization

ABAC Attribute-Based Access Control

ABE Attribute-Based Encryption

AI Artificial Intelligence

ANN Artificial Neural Networks

AOMDV Ad-hoc On-demand Multi-path Distance Vector

API Application Programming Interface

ARIMA Autoregressive Integrated Moving Average

ARMA Autoregressive Moving Average

BIC Bayesian Information Criterion

CA Certification Authority

CAMEL Cloud Application Modelling and Execution Language

CausalNN Causal Neural Networks

CATE Conditional Average Treatment Effect

CBN Causal Bayesian Networks

CDT Causal Decision Trees

CH Cluster Head

COIN COordinate-based INdexing

COPA Cluster-based Optimal Path Algorithm

CPDAG Completed Partially Directed Acyclic Graph

CPU Central Processing Unit

DAG Directed Acyclic Graph

DDR3 Double Data Rate version 3

DE Differential Evolution

DE-DPSO-DPS Differential Evolution - Discrete Particle Swarm Optimization - Data Placement Strategy

DFS Distributed File System

DiD Difference-in-Differences

DirectLiNGAM Direct Linear non-Gaussian structural equation model

DL Deep Learning

DNN Deep Neural Network

DPSO Discrete Particle Swarm Optimization

DSL Domain Specific Language

DT Delaunnay Triangulation

DTM Dynamic Throughput Maximum

DTW Dynamic Time Warping Matching

D-VNF Distributed Virtual Network Function

EHR Electronic Health Record

EU European Union

FCI Fast Causal Inference

FL Federated Learning

FPGA Field Programmable Gate Array

GA-DPSO Discrete Particle Swarm Optimization algorithm with Genetic Algorithm operators

GAN Generative Adversarial Network

GDPR General Data Protection Regulation

GES Greedy Equivalence Search

GIES Greedy Interventional Equivalence Search

GLM General Linear Model

GMM Gaussian Mixture Models

GPU Graphical Processing Unit

GRED Greedy Routing for Edge Data

GRF Generalized Random Forest

HCP Healthcare Professional

HDD Hard Disk Drive

HIE Health Information Exchange

HO Healthcare Organization

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IF Impact Factor

IACR Interference-Aware Cooperative Routing

IAR Interference-Aware Routing

IIoT Industrial Internet of Things

IoT Internet of Things

ITE Individual Treatment Effect

IV Instrumental Variables

KOPA K-Means-based Optimal Path Algorithm

LiNGAM Linear non-Gaussian structural equation model

LLECP Link Lifetime and Energy Consumption Prediction

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MEC Mobile Edge Computing

MIOPA Minimum Interval-based Optimal Path Algorithm

ML Machine Learning

MAR Mean Access Rate

MSE Mean Square Error

MST Mean Service Time

NLP Natural Language Processing

NN Neural Network

P2P Peer-to-Peer

PC Peter - Clark

PCA Principal Components Analysis

PDR Patient Discharge Report

PHR Personal Health Record

QoS Quality of Service

QR Quick Response

R^2 R-squared

R2D Remote-to-Device

R2DB Remote-to-Device Backup

R2DE Remote-to-Device Emergency

RAM Random Access Memory

RCT Randomized Controlled Trials

REST Representational State Transfer

RF Random Forest

RMSE Root Mean Square Error

RNN Recursive Neural Network

SLA Service Level Agreement

SVM Support Vector Machines

TMLE Targeted Maximum Likelihood Estimation

TPU Tensor Processing Unit

UCN User Centric Networks

VM Virtual Machine

VNF Virtual Network Function

VT Voronoi Tessellation

XAI eXplainable Artificial Intelligence

Chapter 1
Introduction

Chapter Structure

This Chapter is constructed as follows:

- **Section 1.1 - Motivation**, explains the reasons behind this research, and highlights its relevance and potential impact.
- **Section 1.2 - Objectives**, outlines the goals and objectives of the research.
- **Section 1.3 - Research Questions**, outlines the main questions that the study aims to answer.
- **Section 1.4 - Research Contributions**, explains the contributions made by the research.
- **Section 1.5 - Outline of Dissertation**, presents a structural overview of the dissertation.

The origins of Artificial Intelligence (AI) can be traced back to the mid-20th century [Carbonell et al., 1983], [Fradkov, 2020], when pioneers such as Alan Turing established the fundamental principles for creating intelligent machines [Turing, 1992]. Nevertheless, it was only in the past several decades that AI and its subset, Machine Learning (ML), have experienced a substantial and rapid develop-

ment [Aggarwal et al., 2022].

This progress has been fueled by breakthroughs in both hardware and software. Computing capabilities have increased (*i.e.*, the computing power has been increased exponentially, while the storage capabilities now allow the collection of massive datasets). Furthermore, the software is now able to handle better complex algorithms. Initially focused on rule-based systems, AI has progressed towards more dynamic techniques, with ML becoming a cornerstone. The evolution from early rule-based expert systems to modern Neural Network (NN) and Deep Learning (DL) has been characterized by the development of more elaborate models that have the capacity to interpret and comprehend complex patterns.

Currently, AI and ML are everywhere, invading almost every aspect of our everyday life. Starting with the healthcare sector in which AI has been made contributions on the diagnosis of various cancer types [Kadir and Gleeson, 2018], [Kourou et al., 2015], [Lazic et al., 2022], [Raoof et al., 2020], [Yarabarla et al., 2019], or the development of individualized treatments or analyzing the survival rate of individuals in clinical trials [Marinos et al., 2022], [Marinos et al., 2021], while in the finance sector, AI is used to enhance trading tactics [Kissell, 2020] and evaluate risks [Bhatore et al., 2020].

The benefits of AI affect the transportation sector too, with the provision of autonomous vehicles [Faisal et al., 2019], while in entertainment, recommendation algorithms personalize content consumption [Pazzani and Billsus, 2007], [Phorasim and Yu, 2017]. Social media platforms exploit ML and DL for advertising [Dehghani and Tumer, 2015], Natural Language Processing (NLP) for sentiment analysis [Manias et al., 2023], while virtual assistants such as Apple's Siri or Alexa from Amazon [Brill et al., 2022] can be used to improve the overall user experience, as they are able to perform daily routines and streamline tasks at ease.

The use of AI is also heavily increased in several emerging fields in the areas of communication and services provision, from Cloud, Edge Computing, and Inter-

net of Things (IoT) [Chen, 2020], [Deng et al., 2020a], [Gill et al., 2022], to Cognitive Network Management for 5G Networks [Bega et al., 2019]. The research towards the creation of more intelligent ways to manage such diverse environments has expanded over the last years, due to the increased use of these solutions by the service providers and the facilities they provide.

Yet, as these technologies advance, important considerations and challenges arise, including issues related to bias, transparency and in general the ability of AI to comprehend the environment it is utilized in, and identify potential causal relationships among the factors that may affect it. And even though AI promises to redefine how we live and work, these issues need to be resolved, something that requires careful navigation.

As already stated, ML has made remarkable progress over the last few decades; however, one of the most significant obstacles, has to do with its inability to recognize and understand causal relationships [Stoica et al., 2017] and even though it usually excels when it comes to recognizing patterns in complex datasets or making predictions based on those correlations found in large datasets, it often fails to identify the basic or hidden causal links that exist. This important limitation prevents us from exploiting the full potential of AI. Not only that, as ML struggles to comprehend the existing cause-and-effect relationships this may lead to less accurate predictions, especially in situations where confounding factors. In order to make better decision-making systems and more advanced AI applications, closing this gap and creating methods that build a better understanding of cause and effect into ML algorithms becomes an absolute necessity.

1.1 Motivation

As described in the above, one of the key challenges of AI regards its incapacity to identify potential causal relationships among the data. For this reason, over the

last years a turn towards Causal Learning has been acknowledged [Cheng et al., 2022]. From adaptations of State-of-the-Art ML algorithms in order allowing the uncovering causal relationships in data [Cheng et al., 2022], [Li et al., 2016b], DL solutions such as Causal NN that are able to perform causal discovery on data [Wiering et al., 2002]. This shift towards the implementation of such algorithms makes it clear that developing AI models that are able to understand such relationships will lead to more accurate and robust predictions. That being said, causal information should enable ML algorithms to not just identify correlations, but rather determine cause-and-effect relationships within a dataset, thus making them capable of making more informed decisions, and generalizing better to unseen data.

This is particularly important in complex real-world scenarios where multiple factors have a crucial impact, such as in the realm of managing Cloud and Edge Computing infrastructures [Geiger et al., 2016], [Wang et al., 2021a]. In such cases, understanding causality is essential for an effective decision-making process, while also increasing the users' trust. The complexity of managing heterogeneous edge servers, IoT devices and deployed services, coordinating distributed computing resources, and managing varying workloads, creates an environment where the interactions between the factors are substantial. In these situations, understanding causality is critical to promoting effective decision-making [Afonso, 2018].

The potential benefits of integrating causal information and highlighting the context via the identification of the most effective instances into datasets are manifold. To start with, this will lead to the development of ML models that are interpretable [Moraffah et al., 2020], [Xu et al., 2020a] and transparent [Mittelstadt, 2021], [Wischmeyer, 2020]. In addition, causal understanding can facilitate on improving the overall performance of a model since it can help on reducing the bias, and enhance the model's fairness [Zhang and Bareinboim, 2018] and generalization ability [Lv et al., 2022].

However, this research is not without challenge. Existing limitations include the

difficulty in accurately utilize the identified causal relationships in data and the efficient incorporation of this knowledge in the data, in order to be exploited by the ML model during the training process, especially in diverse environments such as an edge network or a cloud computing infrastructure. Another key challenge relates to the actual data. In more detail, in general, causally-enhanced data can be difficult to obtain or gain access. Therefore alternatives in order to recognize these cause-and-effect relationships from the data that exist need to exist. Thus, there is a need for methods that enable the identification of context and the extraction of causal information in order to enhance the data and allow a ML to learn from it.

1.2 Objectives

The main objective of this thesis is to examine if there exist ways that enable the extraction of causal and contextual information from data retrieved in heterogeneous and diverse environments as a Cloud infrastructure or an Edge network, towards the creation of ML models capable of efficiently managing them. In order to achieve this objective, this thesis will present specific methods that will allow this.

In more detail, this thesis will present data enhancement methods that will enable (*i*) the identification of context within a given environment through the information that can be collected from it, (*ii*) the extraction of causal information from the collected data, (*iii*) ways to incorporate this additional information to the data, and finally (*iv*) how these methods can be applied as steps in any ML workflow.

In order to evaluate these methods, they will be utilized in different real-world scenarios and in different ML tasks, from classification, to time-series analysis, towards the advancement of existing methods used for managing Cloud and Edge infrastructures. In more detail, these tasks are related to data placement, dynamic resource allocation, optimal deployment configuration, and data prefetching. In

addition, the proposed data enhancement methods will be applied in those scenarios either individually or combined in order to evaluate their significance in both situations.

1.3 Research Questions

This thesis tries to answer the following research questions:

RQ1: "Is it possible to derive causal and contextual relationships from data, and what methods are available for accomplishing this?"

As already established based on the analysis in the previous Sections, there is a growing need for utilizing causal and contextual information which can be somehow exploited towards the enhancement of an ML model. But before getting there it is crucial to understand if such information can be extracted from the data in the first place. Thus, trying to answer this question, this thesis investigates methods related to causality and context identification from data.

RQ2: "Can the extracted causal and contextual information be utilized for improving the performance of ML or DL models used for the sufficient management of Cloud and Edge Infrastructures?"

This question arises as the subsequent step following RQ1. In more detail it elevates the process further, since if causal extraction from data is something that can be achieved, then, are there any ways that can be used in order to utilize this information in order to improve ML models that facilitate on the management of Cloud and Edge Infrastructures? And if so, what are the advantages that an AI model will have if trained on causally- and context-aware data? In order to answer to these points, this thesis evaluates the designed data enhancement methods in order to obtain results that verify their significance.

RQ3: "In which ML tasks can these causal and contextual enhancement meth-

ods be utilized?"

By asking this question, the goal is to identify in which ML tasks the proposed methods can be applied efficiently. This is particularly important, when managing multifaceted environments such as the Cloud or the Edge. To be more precise, this thesis evaluates if the proposed methods can be applied in specific ML tasks including supervised learning where the proposed methods are assessed in classification problems, such as the prediction of the users' mobility class in hybrid Cloud and Edge infrastructures, the recommendation of objects from storage Cloud services, and time-series analysis for the prediction of the resource usage of services deployed at the Edge, among others.

RQ4: Does the enhancement of the data with contextual and causal information improve an ML model's performance?

With this question, the goal is to evaluate the significance of the proposed methods. In other words, do these methods improve in any way the model? In order to answer that, the relevance of the proposed techniques will be tested by comparing the outcomes of the models when the methods are included in the training workflow and when they are absent. This direct comparison will provide the necessary information to perform this evaluation.

1.4 Research Contributions

The research that has been conducted and will be presented in this thesis can be formed in four key contributions and can be divided in these main areas: (*i*) the identification of contextual information from data, (*ii*) the discovery of causal relationships from data, and finally (*iii*) the enhancement of data exploiting the extracted contextual information and discovered relationships.

RC1: An Influence-based Instances identification method

In order to answer questions **RQ1** and **RQ2** in this thesis a novel Influence-based Dataset Generation method will be presented. This method aims at identifying the influence of any instance in a given Dataset and generating a new influence-based Dataset which should highlight the key factors that impact the decision making progress, and as a result facilitate on reducing over-fitting the AI model, and in general aid in improving the model's performance and robustness.

RC2: A Causal Features Generation method

Following an alternate approach to answer questions **RQ1** and **RQ2**, this thesis presents a Causal Features Generation method which aims at identifying causal relationships among the features in a given Dataset, generating causal features according to the previously identified causal relationships, and finally at enhancing the existing Dataset by incorporating the generated causal features. The goal of this method, is again to improve the performance of any AI model by highlighting causal relations that may exist in the data.

RC3: Design and implementation of service components used for the optimization of resources in Cloud and Edge Computing Infrastructures

Trying to answer questions **RQ3** and **RQ4** the two methods are evaluated in difference use cases and scenarios. In more detail, the Influence-based Dataset Generation has been integrated in a component of a Storage Cloud service responsible for the identification of the most popular data in order to prefetch them and thus reduce the overall transmission delays, as well as a scenario in which it is used within a component which identifies the optimal deployment configuration of services in Edge Computing infrastructures. On the other hand, the Causal Features Generation method has been applied in a scenario in which an ML model predicts the mobility of users in Edge networks towards the optimization of data placement taking into consideration the users' mobility.

RC4: Evaluation of the proposed Influence-based Dataset Generation and Causal

Features Generation methods in diverse scenarios

In order to justify the proposed methods' significance and answer question **RQ4**, as already discussed they have been applied to different use cases. In some cases the task was related to a classification problem, while in other cases it was related to time-series analysis problems. Furthermore, in order to thoroughly assess the methods importance, in all cases, a comparison evaluation is performed in which the performance of each model is measured when the methods are applied or not.

In addition, Table 1.1 demonstrates how the publications are mapped to any of the aforementioned research contributions.

1.5 Outline of Dissertation

The remainder of this dissertation is organized in the following way:

- **Chapter 2** presents the Infuence-based Dataset Generation method. The Chapter initially performs a literature review in the Instances Importance and afterwards presents in detail the architecture of the proposed Influence-based Dataset Generation method. Furthermore, in this Chapter a high-level presentation of the scenarios in which the proposed method is evaluated, setting the ground for the upcoming Chapters.
- **Chapter 3** presents the first use case used for the evaluation of the proposed Influence-based Dataset Generation method. The Chapter first analyzes and explains the necessity of the existence of a prefetching mechanism in Health Storage Clouds and the vital role such services play in the Health Information Exchange (HIE). Consequently, an overview of the proposed Health Storage Cloud is performed, showcasing the various features such a service includes, along with the adaptations that were made to the generic Influence-based Dataset Generation method. Finally, this Chapter concludes by depicting the results of the performance evaluation that was performed in order to assess

Table 1.1: List of publications related to the contributions of this thesis.

Title	Authors	Venue	Contribution
A User Mobility-based Data Placement Strategy in a Hybrid Cloud / Edge Environment using a Causal-aware Deep Learning Network	**C. Symvoulidis**, A. Kiourtis, G. Marinos. J.D. Totow Tom-Ata, G. Manias, A. Mavrogiorgou, D. Kyriazis	IEEE Transactions on Computers, 2023 (Impact Factor (IF): 3.7)	C2, C3, C4
HealthFetch: An Influence-Based, Context-Aware Prefetch Scheme in Citizen-Centered Health Storage Clouds	**C. Symvoulidis**, G. Marinos, A. Kiourtis, A. Mavrogiorgou, D. Kyriazis	Future Internet Journal, 2022 (IF: 3.4)	C1, C3, C4
[BEST PAPER AWARD] Enhancing Cloud-based Application Component Placement with AI-driven Operations	E. Paraskevoulakou, J.D. Totow Tom-Ata, **C. Symvoulidis**, D. Kyriazis	2024 14th IEEE Annual Computing and Communication Workshop and Conference (CCWC 2024)	C3
Polymorphic Cloud Application Design Assisted by Open Source Software Classification (in press)	J.D. Totow Tom-Ata, K. Kritikos, M.A. Di Girolamo, E. Paraskevoulakou, **C. Symvoulidis**, D. Kyriazis	2023 5th IEEE International Communication Engineering and Cloud Computing Conference (CECCC 2023)	C3
Dynamic Resource Allocation at the Edge: A Causal Contextually-Aware Machine Learning Approach (in press)	**C. Symvoulidis**, E. Paraskevoulakou, A. Kiourtis, A. Mavrogiorgou, D. Kyriazis	2024 10th Intelligent Systems Conference (IntelliSys 2024), Springer, Cham	C1, C2, C3, C4
[BEST PAPER AWARD] Dynamic Deployment, Prediction and Configuration in Hybrid Cloud / Edge Computing Environments using Influence-based learning	**C. Symvoulidis**, A. Kiourtis, A. Mavrogiorgou, D. Totow Tom-Ata, G. Manias, D. Kyriazis	2023 10th IEEE International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2023)	C1, C3, C4
Electronic Health Records at People's Hands Across Europe: The InteropEHRate Protocols	A. Kiourtis, A. Mavrogiorgou, K. Mavrogiorgos, D. Kyriazis, **C. Symvoulidis**, G. Bella, S. Bocca, F. Torell	2022 19th International Conference on Wearable Micro and Nano Technologies for Personalized Health (pHealth 2022)	C3
Mobile Anonymization and Pseudonymization of Structured Health Data for Research	S. Dimopoulou, **C. Symvoulidis**, K. Koutsoukos, A. Kiourtis, A. Mavrogiorgou, D. Kyriazis	2022 7th IEEE International Conference on Mobile And Secure Services (MobiSecServ 2022)	C3, C4
A Health Information Exchange Protocol Supporting Bluetooth-based Messages	A. Kiourtis, A. Graziani, A. Mavrogiorgou, **C. Symvoulidis**, K. Mavrogiorgos, D. Kyriazis	2021 11th IEEE International Conference on Information Systems and Advanced Technologies (ICISAT 2021)	C3
Emergency Health Protocols: Supporting Health Data Exchange, Cloud Storage, and Indexing	K. Koutsoukos, **C. Symvoulidis**, A. Kiourtis, A. Mavrogiorgou, S. Dimopoulou, D. Kyriazis	2022 15th International Conference on Health Informatics (HEALTHINF 2022)	C3, C4
Facilitating Health Information Exchange in Medical Emergencies	**C. Symvoulidis**, A. Mavrogiorgou, A. Kiourtis, G. Marinos, D. Kyriazis	2021 9th IEEE International Conference on e-Health and Bioengineering (EHB 2021)	C3, C4
Healthcare Provision in the Cloud: An EHR Object Store-based Cloud Used for Emergency	**C. Symvoulidis**, A. Kiourtis, A. Mavrogiorgou, D. Kyriazis	2021 14th International Conference on Health Informatics (HEALTHINF 2021)	C3, C4
Learning a generalized matrix from multi-graphs topologies towards microservices recommendations	I. Tsoumas, **C. Symvoulidis**, D. Kyriazis	2020 SAI Intelligent Systems Conference Springer, Cham	C3
Towards the identification of context in 5G infrastructures	**C. Symvoulidis**, I. Tsoumas, D. Kyriazis	2019 Intelligent Computing Conference Springer, Cham	C1, C3, C4
Modelling 5G Cloud-Native Applications by Exploiting the Service Mesh Paradigm	I. Tsoumas, **C. Symvoulidis**, D. Kyriazis	2019 European, Mediterranean, and Middle Eastern Conference on Information Systems (EMCIS 2019) Springer, Cham	C3

the service as well as the importance of the Influence-based Dataset Generation method.

- **Chapter 4** describes the second use case used for the evaluation of the proposed Influence-based Dataset Generation method. The chapter starts by performing an introduction to the topic of optimal deployment configuration on the edge. Then, an overview of the proposed deployment configu-

ration framework is presented, depicting how the Influence-based Dataset Generation method is utilized. The Chapter concludes by showcasing the results of the performance evaluation, and by discussing the lessons learnt from the first two use cases, relative to the effectiveness of the Influence-based Dataset Generation method.

- **Chapter 5** presents the second data enhancement method, which is called the Causal Features Generation method. The Chapter starts with an analysis of the State of the Art, and the discussion of relevant topics and research areas such as Causal Inference and Causal Discovery. The proposed Causal Features Generation method is then presented in detail, and the Chapter ends with a provision of a high-level view of the use case used for the evaluation of the proposed method.

- **Chapter 6** presents the use case used for the evaluation of the proposed Causal Features Generation method. At first, a literature review in the areas of Data Placement and Optimal Routing is performed. Next, an overview of the proposed User Mobility-based Data Placement Strategy is presented, along with the necessary adaptations that were made to the generic Causal Features Generation method in order to be utilized efficiently in this scenario. Moreover, the evaluation of the data placement strategy is presented, showcasing the usefulness of the Causal Features Generation method. The Chapter ends with a detailed discussion regarding the evaluation outcomes, in which the advantages of the proposed method are highlighted.

- **Chapter 7** presents a framework for Dynamic Resource Allocation at the Edge. In this use case both data enhancement methods are utilized, in order to assess whether they can be used efficiently in combination. First, a literature review in the area of Dynamic Resource Allocation at the Edge is performed. Consequently, an overview of the proposed Dynamic Resource Allocation framework is presented, highlighting the use of the two proposed data

enhancement methods. The evaluation outcomes of the proposed framework are also presented, and the significance of the two methods is also highlighted.

- **Chapter 8** summarizes the Doctoral Dissertation and its main contributions, and in addition, the open research topics and future goals as derived from the present research are described.

Chapter 2
Influence-based Dataset Generation method

Chapter Structure

This Chapter is constructed as follows:

- **Section 2.1 - Background on Instances Importance and Contextual Data Enhancement**, provides an in-depth review on the importance of the instances in a dataset and present how they can be utilized towards understanding better the outcomes of an ML model or how the information that they bring can be exploited in order to make a model more robust and contextually-aware. In addition, a review of contextual enhancement methods is performed.
- **Section 2.2 - Influence-based dataset Generation Architecture**, presents the overview of the proposed Influence-based Dataset Generation method.
- **Section 2.3 - Evaluation of the Influence-based Dataset Generation method**, provides a high-level view of the evaluation scenarios in which the proposed method has been put under.

Before describing the proposed method, it is crucial to define the advantages that training on influential instances provide:

(*i*) When training AI models with datasets that are comprised of influential instances could lead to *better generalization*, since they (*i.e.*, such instances) often capture essential characteristics of the underlying data.

(*ii*) Influential instances can make the *training process more efficient*, since the ML model can concentrate on learning from observations that carry substantial information, which can potentially reduce the need for extensive training on less informative instances.

(*iii*) *Model interpretability* can also achieved when utilizing a dataset that contains influential instances, since it can be made easier to communicate and understand, providing insights into the critical factors driving the model's predictions.

(*iv*) Influential instances may include outliers or anomalies that are important for certain applications, thus leading to more *robust models*. Training on such instances facilitates in recognizing such atypical cases from the ML model. On the other hand, by focusing on influential instances, a ML model may as well become less sensitive to irrelevant observations which do not contribute significantly to the overall predictive task. This can lead to more effective and robust models.

In order to achieve this, the proposed method has two main objectives:

(*i*) To identify the *influence* of each instance in a given dataset [Gupta and Gupta, 2017], and

(*ii*) Second, to generate an *influence-based dataset*.

The influential instances are typically used to elucidate the results of difficult-to-interpret ML or DL models. The current method, however, differs from the current way influential instances are used because it employs influential instances in a manner that will empower a ML model by providing additional data-related information. Specifically, it will provide the model with a catalog of instances that

are potentially more significant than others, which the model can use to enhance itself.

2.1 Background on Instances Importance and Contextual Data Enhancement

2.1.1 Influential Instances

The Section provides an overview of the Influential Instances. In more detail, Sub-section 2.1.1.1 introduces the term "Influential Instances". Sub-section 2.1.1.2, presents the existing available techniques which allow for the identification of influential instances and how they can be used in order to interpret the outcomes of ML or DL models, and in general how can this technique be utilized towards the creation of eXplainable Artificial Intelligence (XAI).

2.1.1.1 What are the Influential Instances

ML models are essentially constructed from the training data they are trained with. Therefore, removing a single instance from the training set can have an impact on the final model.

A training instance is "influential" if taking it out of the training data changes the predictions or parameters of an ML model substantially [Molnar, 2020]. In addition, the influence of any instance can be measured, meaning that the greater the change the instance produces to the model, the more influential it is. Furthermore, the influence of any instance in a model also depends on its target.

There exist several methods in order to detect the influence of the instances in a given dataset. Following, a detailed analysis of the most prominent methods will be performed.

2.1.1.1.1 Deletion Diagnostics

Deletion Diagnostics is a technique which can be utilized for detecting the influence any instance has in the model. The way Deletion Diagnostics work, is as follows: the process starts by training a model with a training dataset which consists of all the available instances. Once the model is trained, the model's predictions or the model's parameters can be measures using any of the available metrics, such as Cook's distance [Cook, 1977] or *DFBETA* [Adadi and Berrada, 2018], [Mi et al., 2020], etc. These metrics are defined in detail in Chapter 3.

Consequently, a repetitive process starts, where one by one, the instances of the training dataset are removed, the model is retrained, and the model's predictions along with the model's parameters are measured again in order to calculate the changes the deletion of an instance brings to the model. Understandably, the higher the changes are when compared to the original measurements, the higher the importance of the instance.

Also, an important note is that deletion diagnostics regards a *model agnostic* method, which means that it can be used in any model in order to measure the influence the instances of data have on it.

2.1.1.1.2 Influence Functions

The method of Influence functions belongs in the field of robust statistics and can be used to calculate the influence an instance has on a model [Koh and Liang, 2017]. Similar to Deletion Diagnostics, the Influence Functions identify the training instance responsible for the model parameters and predictions.

Unlike Deletion Diagnostics, rather than removing training instances, this technique calculates an approximation of the model's changes in response to an instance being upweighted in the empirical risk (which is the sum of the loss across the training data).

Influence Functions work by upweighing a training instance's loss by an extremely small step ϵ, which yields new model parameters as shown in Equation 2.1:

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta}(1-\epsilon)\frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z,\theta) \tag{2.1}$$

where θ regards the model parameter vector, $\hat{\theta}_{\epsilon,z}$ the parameter vector after upweighting z by a very small number ϵ. L represents the loss function used for the model training, z_i represents the data used for training, and finally z is the upweighted training instance. The key idea behind this method is to calculate the change in loss when a specific instance z_i is slightly upweighted from the training data, and then downweight the other data instances by the same amount (ϵ) and how the parameter vector is adjusted to optimize the new loss. This influence can be found the formula found in Equation 2.2:

$$I_{\text{up,params}}(z) = \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta}) \tag{2.2}$$

where $\nabla_\theta L(z,\hat{\theta})$ is the upweighted training instance's loss gradient relative to the model's parameters. The gradient represents the training instance's loss rate of change and illustrates how much the loss varies when the model's parameters $\hat{\theta}$ are changed. This means that when a model parameter rises and the gradient vector shows a positive entry, the loss is increased, whereas, when the parameter increases and the loss is decreased. $H_{\hat{\theta}}^{-1}$ is the inversed Hessian matrix which represents the rate of change of the gradient and can be estimated as defined in Equation 2.3:

$$H_\theta = \frac{1}{n}\sum_{i=1}^{n}\nabla_{\hat{\theta}}^2 L(z_i,\hat{\theta}) \tag{2.3}$$

Influence Functions cannot work for all ML types though, since the method re-

quires access to the loss gradient with respect to the model parameters. Hence, it can be used in models such as Artificial Neural Networks (ANN), Logistic Regression, and Support Vector Machines (SVM). On the other hand, the Influence Functions method cannot be used in tree-based models such as Random Forests.

2.1.1.1.3 Influence Sketching

Influence Sketching, proposed by Wojnowicz *et al.* [Wojnowicz et al., 2016] is an algorithm used to score the influence of the instances used for the training of ML models. The authors proposed a novel scalable version of Cook's distance, in which random projections are inserted within the influence computation. In more detail, the influence score is calculated using the randomly projected pseudo-dataset from the post-convergence General Linear Model (GLM).

This method, similar to Influence Functions and Deletion Diagnostics, requires the calculation of influence on instance level [Søgaard et al., 2021], which as already described above, happens in an iterative manner and it may require a substantial amount of time to complete, depending on the size of the training dataset.

2.1.1.1.4 TracIn

Pruthi *et al.* [Pruthi et al., 2020] proposed TracIn. TracIn method calculates the impact of a training sample on a model prediction. The goal is to track changes in the test point's loss as the training process proceeds, each time the relevant training example is used. As stated by the authors, the method can also be scaled by using (*i*) a first-order gradient approximation to the precise calculation, (*ii*) preserved checkpoints of common training processes, and finally (*iii*) picking specific layers of a Deep Neural Network (DNN).

2.1.1.2 Influential Instances for ML / DL Interpretation

Influential Instances as a method used for ML and DL interpretability is based on the idea that model's parameters and predictions can be traced back to the dataset that was used for training it.

The sense is that the algorithm responsible for producing a ML model, can be defined as a *function* that accepts the training data, which comprises features X and the target y as input, and returns the ML model as output.

In order to interpret a model using any of the Influential Instances methods the key is to detect the impact on the model parameters or predictions that would result from the exclusion of instances from the training data during the training phase. The use of Influential Instances facilitates understanding the behavior of a model as well as interpreting why individual predictions were made.

Not only that, using Influential Instances one can understand which of the samples in a dataset were the most impactful in the training process or which of those were the most consequential for a given prediction. As a result this could also lead to better understanding of the actual data, since it will be made easier to identify potential erroneous data, in which cases the model could perform worse and ultimately evaluate the robustness of the model.

2.1.2 Contextual Enhancement Methods

2.1.2.1 Outlier Detection

Outlier Detection (otherwise known as Anomaly Detection) regards a technique frequently used for contextual information extraction and retrieval [Martinez et al., 2008], [Stoimenova et al., 2006]. As also presented in [Singh and Upadhyaya, 2012], this technique is most commonly used for time-series and spatial data [Zheng et al., 2017], and allows for the detection of instances which are considered anoma-

lies, given a specific context.

There exist several methods to apply Anomaly Detection, including statistical methods, distance-based methods, density-based methods, or ensemble methods, among others. How it usually works, is that a threshold is defined (either by a domain expert, or using statistical measures) and if this threshold is not met, then a given instance is considered an outlier.

2.1.2.2 Active Learning

Active Learning is an ML-based technique which can be used for contextual information extraction [Cardellino et al., 2015], [Wu and Pottenger, 2005]. Active learning starts by initializing a model with a small amount of labeled data. Consequently, the model selects a subset of unlabeled data points to query for labels, using techniques such as uncertainty sampling, in which the data that the model is most uncertain about is selected. The labels of this data is then acquired from a human or a data source and the model is updated, according to the new labels. The above procedure is repeated until the model is trained. Active Learning can be applied for contextual information extraction, since it can be adapted in order to identify data points (*i.e.*, instances) that are most likely to contain contextual information.

2.1.2.3 Transfer Learning

Transfer Learning is a technique falling under the ML umbrella. Transfer Learning is a technique in which a model is trained in a large dataset for a specific task. Then this model is fine-tuned, by re-training it using a smaller dataset, in order to be utilized for a new task. Transfer Learning can be applied for contextual information extraction, since it can be used in order to extract relations and contextual information from a dataset, such as entities in text [Zhang and Cao, 2023].

2.1.2.4 Importance Sampling

Importance Sampling is a statistical technique similar to Influential Instances where the instances in a dataset are weighted, according to their influence on the task, through an *importance function* [Tokdar and Kass, 2010], where the higher the importance of an instance (*i.e.*, the more informative it is), the higher the weight it gets. Importance Sampling can be used for contextual information extraction [Xu et al., 2022], especially in cases where labeling the data is too expensive or the data is imbalanced, by allowing the mode to focus on the important instances while learning. Consequently,

2.1.3 How the Influential Instances are used in the Influence-based Dataset Generation method

In the current thesis, the Influential Instances method is utilized for the optimization of a model's performance. The key rationale behind this choice is that the Influential Instances provide a lot of information related to the data and their impact on a model. So, this information could somehow be utilized in order to improve the actual model if included in the training procedure. The Influential Instance method is explained thoroughly in Section 2.2.

2.2 Influence-based Dataset Generation Method Overview

This Section describes in detail the Influence-based dataset Generation method. As already described above, its objectives are two-fold; (*i*) to identify the influence of any instance in a given dataset and (*ii*) to generate a new dataset based on those instances. Creating a new dataset by selecting influential instances for training a ML model has multiple potential advantages:

- **Improved Model Performance**: It is possible to train a more effective and

accurate model by concentrating on the cases that significantly affect the model's predictions. Using only the most impactful instances when training an AI model, enhances its capacity for prediction and generalization.

- **Reduced Noise**: Eliminating irrelevant instances from the training dataset can facilitates on reducing over-fitting the AI model, which should also enhance its ability to generalize to new, unobserved data.
- **Faster Training**: Using influential instances for the training of AI result in the generation of smaller datasets which leads to shorter training periods. This is important especially when working with large datasets, as it reduces the computational requirements and accelerates model development and deployment in general.
- **Enhanced Interpretability**: Using a smaller, more focused dataset can make it easier to interpret the model's behavior, since it can be made easier to understand why the model makes specific predictions.
- **Data Quality Improvement**: The process of identifying influential instances can also reveal data quality issues. In more detail, when identifying the most influential instances and using just those for training, can lead to the elimination of inaccurate or inconsistent data which could reduce the overall quality of a dataset.
- **Reduced Bias**: In case some instances influence the model's predictions disproportionately against other instances. In such cases, training using only the influential instances can be used to reduce bias, since the generated data set should be more balanced. Hence, the probability of bias in the model's predictions may be reduced.
- **Resource Efficiency**: As already stated, using datasets that are very big in size lead to longer training periods. In addition to that, big datasets also lead to other resource-related issues. When using a smaller, more concise dataset

consisting solely of the influential instances less resources are required for storing, and thereby reducing the overall cost.

- **Improved Robustness**: Finally, when using datasets comprised of influential instances for training of an AI model, it can make it more robust against outliers or other data points that could negatively impact its performance.

However, to ensure that the resulting dataset is representative of the entire population, it is necessary to exercise caution when selecting influential examples. Incomplete or biased selection may result in the loss of vital information or unintended model behaviors. In order for a ML or DL model's decisions to be explicable and justifiable, it is essential to maintain transparency in the data selection process.

Creating a new dataset by identifying and utilizing influential instances can be a valuable strategy, but it must be executed with care and in conjunction with rigorous data analysis and domain knowledge.

The proposed method is split into two main phases; the *training* phase, and the *prediction* phase.

2.2.1 The Training Phase

Beginning with the training phase, the proposed strategy consists of five principal steps, as also depicted in Algorithm 1:

(*i*) the initial training of the ML algorithm using the original dataset,

(*ii*) the calculation of specific metrics, which determine whether an instance affects the fitted model,

(*iii*) the selection of the most influential instances as determined by the defined measures,

Algorithm 1 Influence-based Dataset Generation Generic method

Auxiliary Variables:

$\mathcal{D}$: the initial dataset,
$\mathcal{D}_{-i}$: the initial dataset, with the i-th instance removed,
$\mathcal{D}'$: the dataset consisting of all identified influential instances,
influentialInstances: the dataset constructed of the identified influential instances.

Output:

$\mathcal{D}'$

Algorithm:

```
1: fit(D)
2: influentialInstances = [ ]
3: calculate metrics
4: for i ∈ D do
5:     D_{-i} : i ∉ D
6:     fit(D_{-i})
7:     calculate metrics for D_{-i}
8: end for
9: for i ∈ D do
10:    if thresholds are met then
11:        influentialInstances ← influentialInstances ∪ i
12:        D : D − i
13:    end if
14: end for
15: kmeans.fit(influentialInstances, k = |influentialInstances|)
16: kmeans.partial_fit(D)
17: D' ←i:  for each i ∈ kmeans.clusters
18: fit(D')
```

(*iv*) the identification of all influential instances using the K-Means algorithm, the generation of a new dataset consisting of only the identified influential instances, and lastly,

(*v*) the retraining of the ML or DL model using the new dataset comprised of influential instances.

In more detail, this method entails a methodical process for enhancing the performance of an ML or DL model. The process starts with the preliminary training of the model with the original dataset. This initial training establishes the model's

baseline understanding of the data. Following this, the control metrics are computed in order to evaluate the influence of individual data points on the model. The selection of the most influential data points is thereafter determined based on these measures.

In order to further enhance the precision of the decisions, an additional step is performed in which instances which were not initially considered influential are assessed. There are several ways to perform this step, as it will be described in the following chapters. One way regards the utilization of the K-Means clustering technique in order to categorize comparable influential data points. The K-Means clustering algorithm is employed to further refine the influential instances. This algorithm combines similarly influential data elements. Clustering facilitates the reduction of redundancy and the collection of a representative sample of essential data points. Another way to evaluate the influence of such those instances, as shown in Chapter 4 could be a distance metric where an instance is considered influential if it is ”close” to an instance that is already marked as influential from the previous steps. Regardless of the method selected for the assessment of those instances, a novel dataset is finally generated, including exclusively these significant cases.

In the concluding step of the *training phase* of the proposed method, the ML or DL model is retrained using the new dataset containing influential examples. This concentrated dataset enables the model to learn from the most important data points, which can lead to enhanced performance and precision. Essentially, it fine-tunes the model to focus on what matters most in the data, which could result in a more accurate and efficient model.

2.2.2 The Prediction Phase

The *prediction phase* regards the phase where the trained model is utilized for inference. During this phase, the model is applied to new data in order to make predictions or classifications based on the knowledge it acquired during the training phase, thereby making it a valuable tool for real-world applications and decision-making.

2.3 Utilization of the Influence-based Dataset Generation method

The Influence-based Dataset Generation method has been effectively applied to three distinct use cases, each in a different domain, to demonstrate its ability to be generalized and applicability across a variety of situations. The algorithm is adapted accordingly in each case.

- In the first use case, found in Chapter 3, it was utilized as the method for training a DL network, which was used in a health storage cloud solution for the recommendation of the most used items in order to pre-fetch them and effectively reduce the transmission delay of such documents in emergency cases.
- In the second use case, which can be found in Chapter 4, the proposed method was used for the training of a ML model, used for the identification of the optimal execution context of services in Edge and Cloud infrastructures, based on the services' key characteristics and features.
- In the third use case, found in Chapter 7, the method is used in combination with the Causal Features Generation method, as it will be presented in Chapter 5. In this use case, both methods are utilized as a pre-processing step for a ML model responsible for the forecasting of the resource usage of services

deployed in Edge infrastructures.

The evaluation outcomes are described in detail for each case in the following Chapters, where the systems in which the Influence-based Dataset Generation method was adopted are also defined.

Chapter 3
Cloud Computing Optimization: An Influence-based, Prefetch Scheme in Citizen-centered Health Storage Clouds

Chapter Structure

This Chapter is constructed as follows:

- **Section 3.1 - Facilitating HIE in Medical Emergencies**, describes the concept HIE and its significance.
- **Section 3.2 - Background on Health Storage Clouds, EHR Management on the Cloud and Data Prefetching**, analyzes the State of the Art concerning Data Prefetching, the current Health Storage Cloud solutions, as well as means of Electronic Health Record (EHR) Management on the Cloud.
- **Section 3.3 - Overview of the Health Storage Cloud**, presents an overview of the proposed Health Storage Cloud and analyzes the adaptations that were made in order to apply the generic Influence-based Dataset Generation method to Prefetch Engine of the Health Storage Cloud.
- **Section 3.4 - Evaluation Results**, evaluates the proposed Health Storage Cloud

and presents the outcomes of the experiments.

This Chapter presents the evaluation of the Influence-based Dataset Generation method in the context of the implemented health storage cloud solution as it will be described below. In more detail, the proposed method, has been utilized as a pre-processing step in order to train a DL network, for the identification of the most used items (*i.e.*, health data) in order to pre-fetch them.

3.1 Facilitating HIE in Medical Emergencies

The objective of HIE [Kiourtis et al., 2021a] has received a growing amount of attention, particularly in regards to the interchange of health data between various healthcare institutions. However, the ability to exchange this data with citizens in a manner that offers them control over their data remains a crucial aspect.

To this end, a HIE strategy was designed [Symvoulidis et al., 2021b], [Kiourtis et al., 2022] consisting of two HIE protocols [Koutsoukos et al., 2022], a Storage Cloud service [Symvoulidis et al., 2022], [Symvoulidis et al., 2021a], which allowed the storage of EHR and Personal Health Record (PHR) in the cloud, and the Health Record Index [Kiourtis et al., 2021b], a service that facilitates in the real-time accessing of health data in emergency situations. Furthermore, the proposed solution allowed the backup and sharing of such data using two Remote-to-Device (R2D) internet-based protocols, namely the Remote-to-Device Backup (R2DB) and Remote-to-Device Emergency (R2DE) over Hypertext Transfer Protocol (HTTP) / Hypertext Transfer Protocol Secure (HTTPS).

In more detail, the R2DB protocol is an internet-based protocol which can be used by citizens and allows them to backup safely their EHR and PHR in a storage cloud service. This protocol specifies a set of operations used to enable all communication activities between a mobile application and an Health Storage Cloud in order to back up health data to a remote cloud service. Figure 3.1 regards a sequence dia-

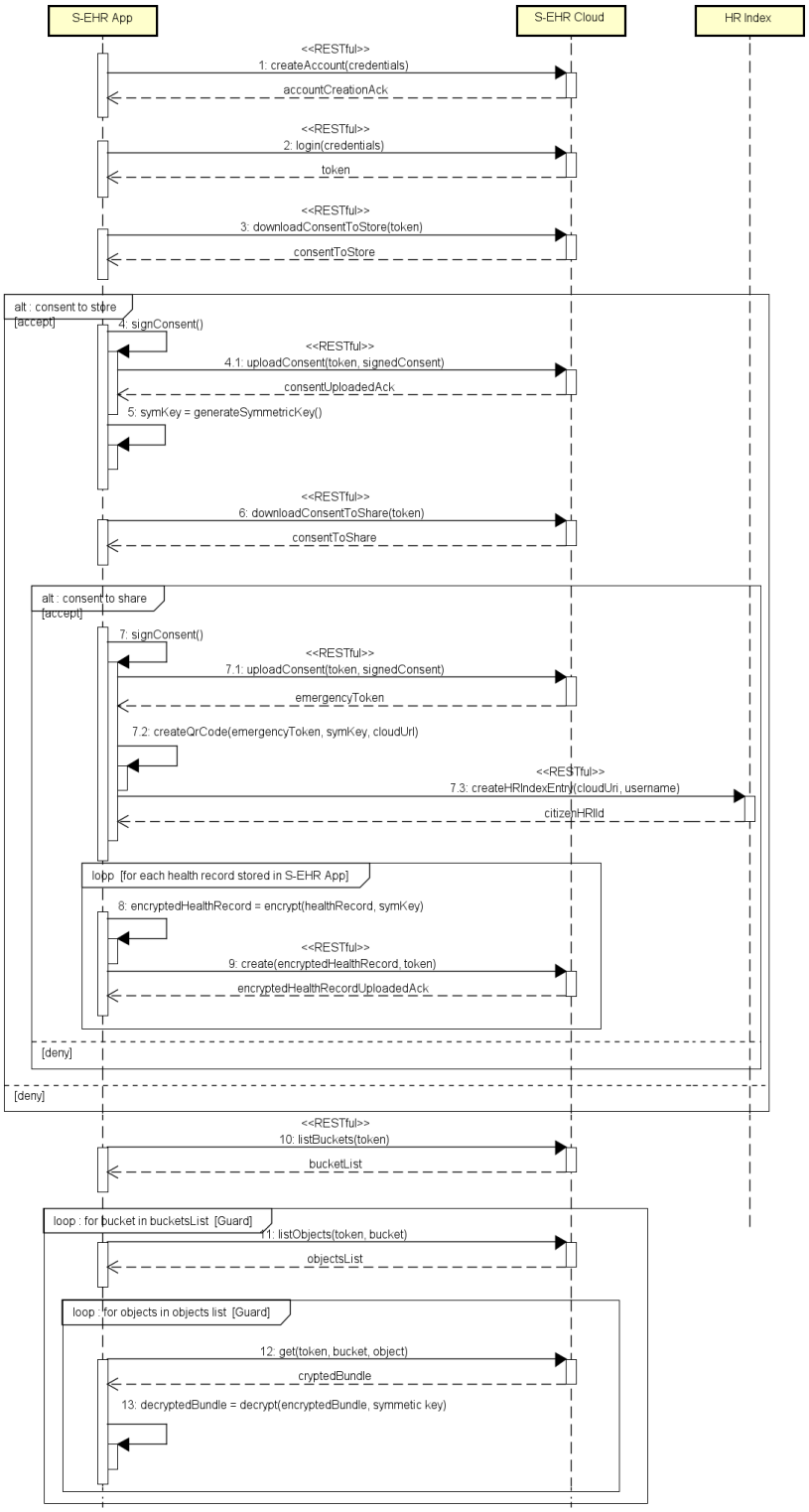


Figure 3.1: Remote-to-Device Backup Protocol Sequence Diagram.

gram depicting a comprehensive description of the R2DB protocol, which follows an algorithmic process among the stakeholders. The steps depicted in the figure can be divided into four categories:

(*i*) **Registration / Login to the Health Storage Cloud**: The user connects to their preferred Health Storage Cloud using a mobile application and registers to it, if not already done that. Otherwise, the user logs in using their credentials.

(*ii*) **Approval of consents for storing and sharing EHRs and PHRs**: In case the user utilizes the Health Storage Cloud for the first time, they are requested to accept two consent forms. The first consent allows the cloud service to store the user's encrypted health data, while the second one allows the cloud service to share the user's health data with trusted Healthcare Professional (HCP) from a trusted Healthcare Organization (HO) in case of an emergency. The consents are digitally signed by the user.

(*iii*) **Health data transfer to the Health Storage Cloud**: Once the user has agreed on the aforementioned consent forms, the backup of their EHR procedure is instantiated. The user's EHR that is already stored on the mobile device of the user is encrypted locally and then uploaded to the user's preferred Health Storage Cloud service.

(*iv*) **Restoration of previously uploaded health data on the Health Storage Cloud**: This step allows the user to download their EHR from the Health Storage Cloud back to their mobile device and synchronize in case new EHRs have been uploaded.

On the other hand, the R2DE protocol, allows HCPs from trusted HO to access a user's data in case of an emergency. The procedure is depicted in the form of a sequence diagram in Figure 3.2.

As depicted in Figure 3.2, the process consists of four main steps:

(*i*) **Scan the user's Quick Response (QR) code**: The HCP scans the user's QR

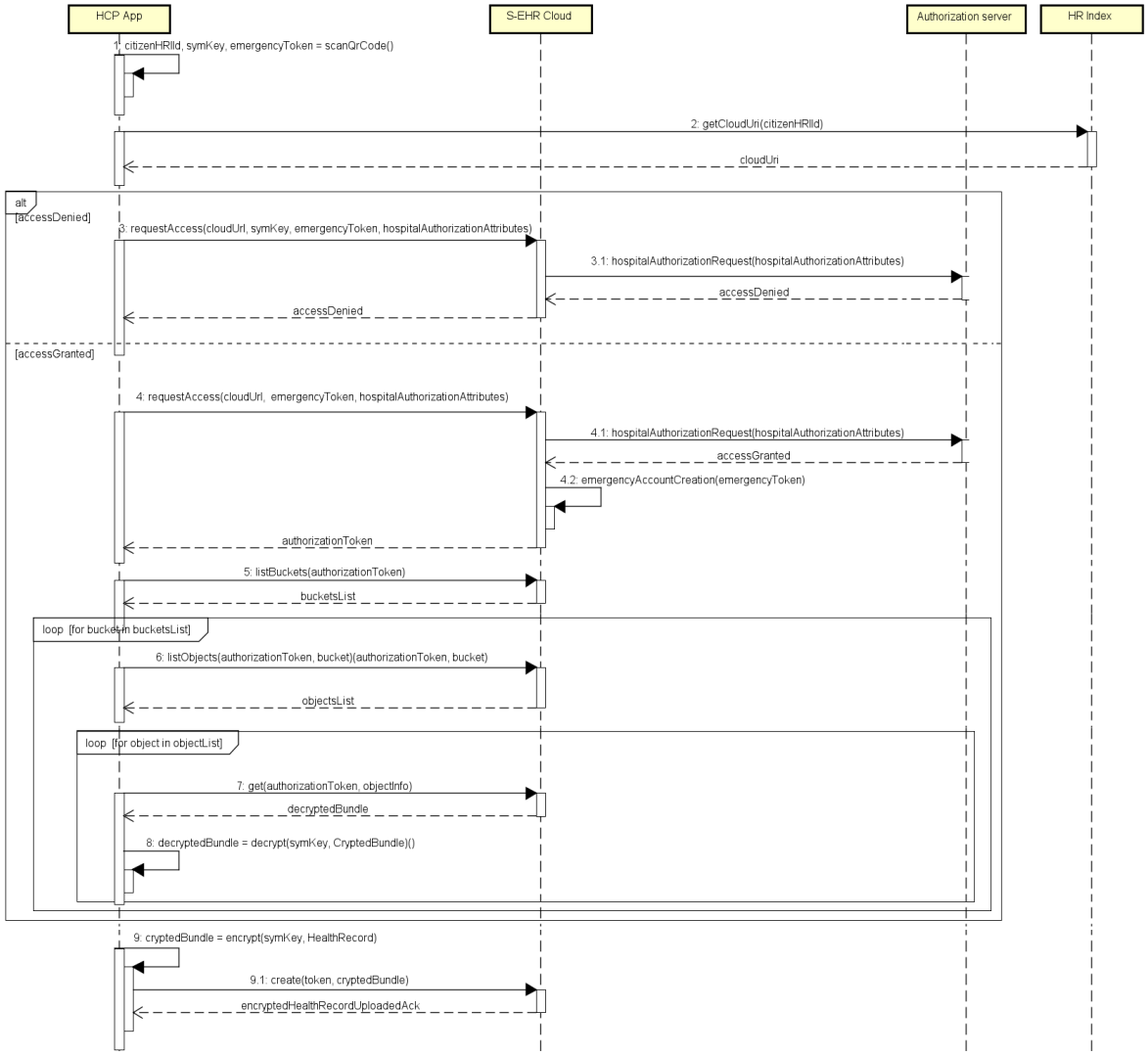


Figure 3.2: Remote-to-Device Emergency Protocol Sequence Diagram.

code which includes information related to the user's preferred Health Storage Cloud, along with the encryption key which can be used for the decryption of the health data once it is downloaded from the user's Health Storage Cloud.

(*ii*) **Request access to user's health data by HCP**: The HCP requests access to the user's selected Health Storage Cloud. In the HCP's request a set of attributes are also sent to verify that the HCP is related to a trusted HO. If the request is successful, a temporary account is created for the HO which can be used

by the HCP in order to download the user's medical information from the Health Storage Cloud.

(*iii*) **Download of user's health data from Health Storage Cloud by the HCP**: Using the temporary account, the HCP can perform request to the Health Storage Cloud in order to download the encrypted EHR and PHR of the user. This data can then be decrypted using the encryption retrieved from the QR code provided by the user.

(*iv*) **Upload of Health Data related to the emergency to the user's Health Storage Cloud**: Once the emergency is over, the HCP may upload health data related to the emergency, back to the user's Health Storage Cloud. Such data may include the Patient Discharge Report (PDR) at patient's discharge. All data uploaded by the HCP are also encrypted using the same encryption key prior to the their upload to the Health Storage Cloud service.

A comprehensive view of the emergency scenario [Symvoulidis et al., 2021b], [Kiourtis et al., 2021b] that the proposed Health Storage Cloud was used in, is depicted in Figure 3.3.

3.2 Background on Health Storage Clouds, EHR Management on the Cloud and Data Prefetching

3.2.1 Health Storage Clouds and EHR Management on the Cloud

Before proceeding with the description of the architecture of the proposed Health Storage Cloud it is important to summarize the work achieved in the field as well as the importance of people being able to own their own health data. Over the last few years and the evolution of the healthcare sector along with already established technologies such as Cloud, Edge computing and the IoT, a significant turn toward citizens' health data ownership has been acknowledged [Sonin et al., 2021], [Mc-

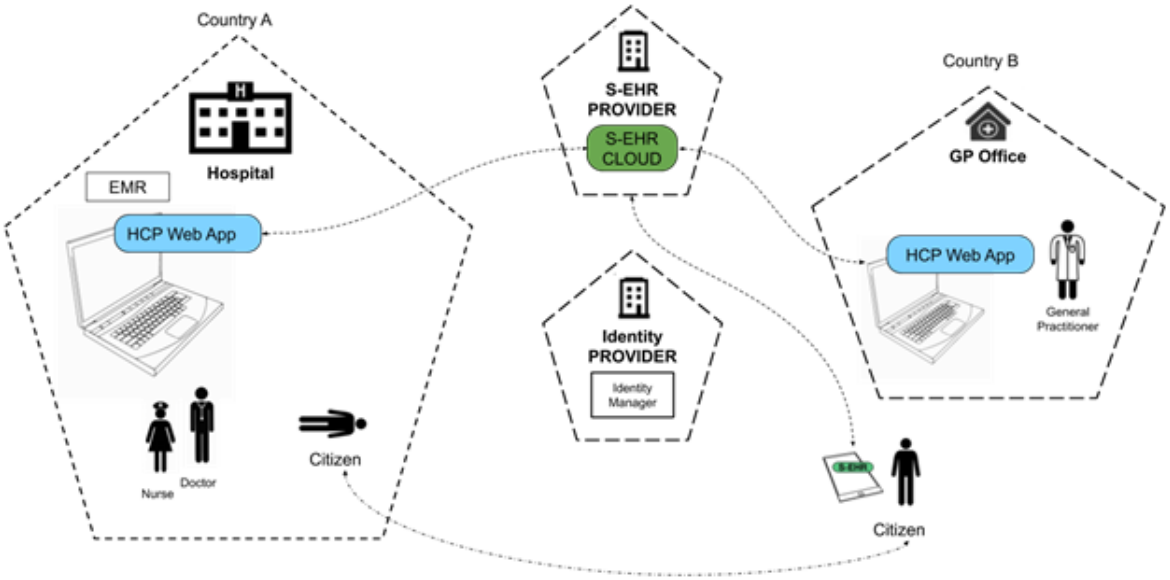


Figure 3.3: The Emergency Scenario used for the evaluation of the proposed Health Storage Cloud solution.

Curry, 2021].

For this reason, more and more technology companies have been turning in that direction [Health, 2023], [Microsoft, 2023] in order to allow their users to have full ownership over their health data, such as PHRs. Even countries have already started on the development of patient-oriented systems that provide new capabilities to the citizens. In more detail [Digital, 2023] allows Estonian citizens to combine health data from various sources and healthcare institutions, in order to provide them comprehensive view of their health status. However, while the above-mentioned initiatives certainly are steps in the right direction, there is still a significant lack of cloud solutions that are *citizen*-centered [IBM, 2023] and not HCP-centered [Salesforce, 2023], [Services, 2023].

To start with, several storage cloud services solutions have been proposed over time for the secure collection of EHRs and healthcare data in general. In [Cai et al., 2017], the authors proposed an EHR sharing scheme deployed on Cloud Computing infrastructures, in which the owner of the healthcare data can generate EHR

ciphertexts, for a user to be able to decrypt them based on transformed ciphertexts from the EHR Cloud.

Furthermore, the authors of [Cao et al., 2019] suggest a safe cloud-assisted e-Health system in which the healthcare organizations that create and maintain a patient's EHR make sure that only those with permission may see and alter that data [Kiourtis et al., 2023], without the use of a reliable third party. The latter is made possible by taking advantage of blockchain technology, which offers an invulnerable method of conducting operations, such as the exchange of EHRs, because no transaction can be approved until it is recorded on the blockchain.

Similarly, the authors in [Seol et al., 2018] proposed a cloud-based EHR model, which exploits Attribute-Based Access Control (ABAC) techniques for securing transactions between the owner of the EHR and its requestor. All transactions are digitally signed prior to their execution, while partial encryption of the EHRs is performed as well. In addition, in [Joshi et al., 2018], a centralized Attribute-Based Authorization (ABA) mechanism is introduced and in more detail Attribute-Based Encryption (ABE) is used, in order to allow medical institution to delegate the authority to healthcare providers in accessing EHRs stored in cloud-based systems. This comes in agreement with the authors of [Manoj et al., 2017] where they suggest the use of two encryption methods in a hybrid EHR system towards the optimized access control and privacy upon the health data.

Finally, the authors of [Jin et al., 2011] proposed a distributed storage model for EHR using Apache HBase [Apache, 2023b], a column-oriented database built on top of Hadoop Distributed File System (DFS) [Apache, 2023a].

3.2.2 Background on Data Prefetching

Data prefetching (otherwise known as *data caching*) is a technique which is often used in database systems [Esteves et al., 2020], [Chen et al., 2021] to improve the

execution performance by pre-storing data on memory before it is actually needed. Such methods are frequently used in database systems for the prefetching of small volumes of data, such as data in websites or data generated by IoT devices [Singh and Chitra, 2021], [Hussien and Sulaiman, 2017].

While prefetching works great with smaller-sized files, when it comes to large files that cannot be manipulated or stored even for a short time on memory or in cache, other technologies such as *data replication* [Shakarami et al., 2021] are used instead. There are several works that attempt to use this paradigm and integrate it to the Web Services and Cloud domains.

To begin with, the authors in [Mansouri and Javidi, 2018] proposed a data replication strategy which identifies the correlation of the data files stored in the cloud using historical data. The proposed strategy predicts which data are going to be requested and replicates them in order to reduce transmission delays. In more detail, the proposed data replication strategy measures the files' popularity within the cloud infrastructure, where in this case the popularity translates to the correlation a file has with the other files stored in the database, and replicates them in the requested site, thus improving both the access time and the bandwidth usage overall.

In [Mansouri, 2016] the author presented an adaptive data replication strategy aimed at minimizing the response time of applications. The data replication strategy takes into consideration several metrics, including the Mean Service Time (MST), Mean Access Rate (MAR), the load variance, the latency, etc. in order to calculate a cost function which decide which is the most appropriate server for a given data to be replicated.

Similarly, the authors in [Liang et al., 2021] proposed another correlation-aware replication strategy which is based on a rule-based management system and identifies the data to be replicated. The replication strategy identifies high-valued files (*i.e.*, files that are more likely to be requested by the users of the system), and repli-

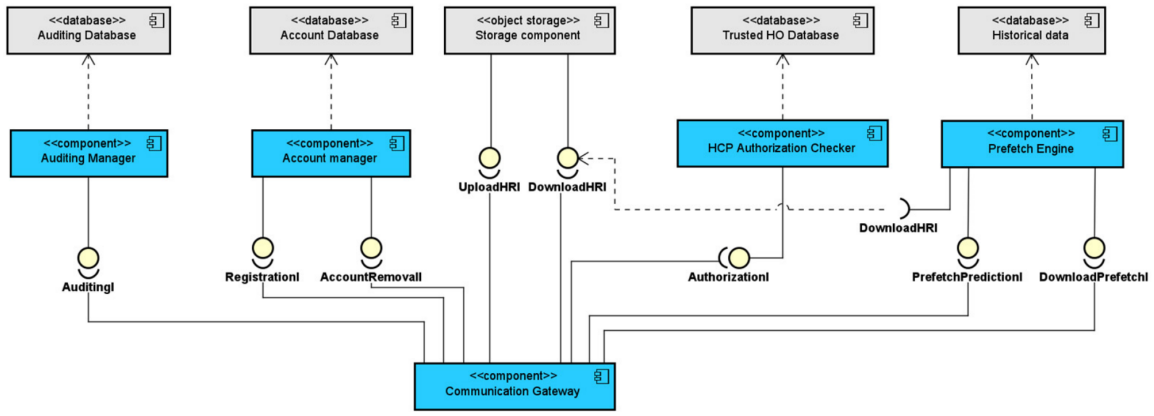


Figure 3.4: Health Storage Cloud high-level architecture.

cate them in a prefetch pool. In the case where the prefetch pool's capacity is not sufficient, a decision is made based on the above-mentioned values in order to remove files already replicated in the prefetch pool to free space for the new replicas.

3.3 Overview of the Health Storage Cloud

This Section describes the proposed Health Storage Cloud service that was implemented. A graphical depiction of the proposed service is illustrated in Figure 3.4.

3.3.1 Core Features of the Health Storage Cloud

The following features are enabled to the designed Health Storage Cloud:

- The *Communication Gateway* (see Section 3.3.2) which handles all incoming requests from the users of the service,
- The *Account Management* (see Section 3.3.3), which is responsible for the management of the users accounts and the policies applied to each user, along with the management of the HO temporary accounts which are generated in emergency situations,
- The *HCP Authorization Check* (see Section 3.3.4), which allows the authoriza-

tion of HCPs from trusted HOs and grants them access to the Health Storage Cloud service,

- The *EHR Storage* (see Section 3.3.5), which is used for the storage of the user's personal health data,

- The *Auditing management* (see Section 3.3.6), responsible for keeping logs of all actions performed by any user of the service, and finally,

- The *Data Prefetching* (see Section 3.3.7) which is accountable for determining which data requests will be made by system users and prefetching the corresponding health data.

The above listed features will be further explained in the following Sections.

3.3.2 Communication Gateway

The Communication Gateway is accountable for administering all incoming requests to the cloud storage. During an emergency, these requests may come from both citizens and physicians. It is a Flask service [Projects, 2023] that responds to HTTP requests (Figure 3.5 depicts an example HTTP request). Its primary functionalities are divided into two categories: (*i*) the functionalities that a user can perform, such as a request to create an account, a request to upload an encrypted health record, or a request to download the auditing information collected by the *Auditing manager*; and (*ii*) the functionalities performed by healthcare professionals on behalf of a healthcare institution. These capabilities include the initial request to access a user's health data, a request to obtain a specific health record, and a request to submit a discharge report once the emergency has passed and the user has been discharged from the healthcare institution.

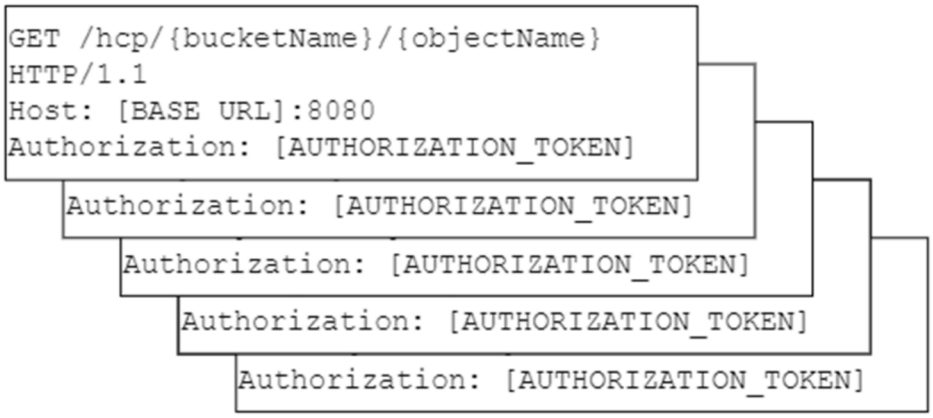


Figure 3.5: Sample HTTP requests received by the Communication Gateway.

3.3.3 Account Management

With this feature administering the accounts of users of the Health Storage Cloud can be performed, including also the temporary accounts for the healthcare professionals that were granted access during an emergency. Once a user requests to use the proposed Storage Cloud service, an account that is linked to a *bucket* [MinIO, 2023b] on the EHR Storage is created. The user stores their encrypted health data in a container, which is essentially a directory that is only accessible through their account. This bucket is also accessible to authorized healthcare professionals with transient accounts during medical emergencies. Importantly, only the user's account has both read and write permissions, while all other accounts (*i.e.*, impermanent accounts of healthcare personnel) have only read permissions. A dedicated policy, such as the one shown in Figure 3.6, is used to grant these permissions to the user's account.

3.3.4 HCP Authorization Check

The HCP Authorization Check feature allows to determine whether a healthcare institution is authorized to access a user's health data in the event of an emergency. The healthcare professionals treating the user submit a request to the cloud stor-

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "*"
        ]
      },
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads"
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:ListMultipartUploadParts",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::${aws:citizen}/*",
        "arn:aws:s3:::${aws:citizen}*"
      ]
    },
    {
       // Additional policies
    }
}
```

Figure 3.6: Sample Policy for a user's bucket.

age in order to access the user's health data via the use of a temporary account. This request should contain a collection of authorization attributes derived from a trusted third-party Certification Authority (CA). Therefore, via the HCP Authorization Check the trusted authority is contacted and the attributes are transmitted to it. The CA then evaluates these characteristics and decides whether or not to approve the request. If the request is accepted, the creation of the temporary ac-

count is authorized and the account is created. If the request is denied, a response is sent to the Communication Gateway to inform the requester that their request has been denied.

3.3.5 EHR Storage

This feature can be used for storing a user's health information [Mavrogiorgou et al., 2023]. Its purpose is to ensure that health records are not only stored securely, but are also immediately accessible, particularly in emergency situations where every second counts; any potential delays should be eliminated. The EHR Storage is based on MinIO Object Storage [MinIO, 2023a], an Object storage solution. Numerous reasons exist for choosing an object store over a conventional NoSQL-based solution such as MongoDB/GridFS [Pramukantoro et al., 2019], [Goli-Malekabadi et al., 2016], [MongoDB, 2023], [Pandey and Subbiah, 2016] or an SQL-based system. The health data stored in the storage cloud should be encrypted on both the client and storage cloud sides, so that the cloud provider does not have access to it and even if an unauthorized entity gains access to the storage cloud, they cannot access the health data. Consequently, cloud-stored data are essentially encrypted bundles [Dimopoulou et al., 2022]. This solution met the requirements because the health data stored in object storage are stored as objects. In addition, individuals can deploy the proposed cloud storage service as a self-hosted service.

In addition to being readily deployable on commodity hardware, the object storage solution that we utilized possesses another essential characteristic in that it is also a lightweight solution. Lastly, the high availability assurance and simple scalability of MinIO played a crucial role in its implementation. As previously mentioned, the Account Management feature is used to apply specific policies to the various accounts that are created and is responsible for their management.

3.3.6 Auditing Management

This feature allows the maintaining of logs of all actions performed on the encrypted health data (*i.e.*, uploads, downloads, modifications, and uploads of newer versions), requests to access the user's health data (that may be approved or denied), and the emergency temporary accounts that were created along with the downloaded or uploaded data. The information maintained by the auditing manager contains the following, as also depicted in Figure 3.7:

- Health data uploads performed by users,
- Health data downloads performed by users,
- Approved / Denies emergency access requests,
- List of HCPs and physicians who have temporary access to the storage cloud through their HO's account,
- List of declined requests and their originators,
- Health data downloads performed by HCPs during an emergency using their HO's temporary account,
- Health data uploads performed by HCPs during an emergency using their HO's temporary account

3.3.7 Data Prefetching

The Data Prefetching pertains to the implementation of the prefetch scheme's engine, as depicted in Figure 3.8. It is used for providing recommendations regarding the data an HCP may request from the Health Storage Cloud prior to the actual request, so that it can be cached and prepared for transmission once the request is made. During an emergency, HCPs will send HTTP requests, demanding the download of health data for a given user. These requests are piled in the form of a

```
"auditing": {
    "hr_info": {
        "uploaded": [{
            "uploaded_hr": "$hrName",
            "hr_uploaded_on": "$timestamp"
        },
        ...
        ],
        "downloaded": [{
            "downloaded_hr": "$hrName",
            "hr_downloaded_on": "$timestamp"
        },
        ...
        ]
    },
    "hco": {
        "granted_access": [{
            "healthcare_organization_name": "$hcoName",
            "granted_access_on": "$timestamp",
            "healthcare_professionals_granted_access": [
                "$hcp1Name",
                "$hcp2Name",
                ...
            ]
        },
        ...
        ],
        "downloaded": [{
            "healthcare_organization_name": "$hcoName",
            "healthcare_professional_name": "$hcpName",
            "downloaded_hr_on": "$timestamp",
            "downloaded_hr": "$hrName"
        },
        ...
        ],
        "uploaded": [{
            "healthcare_organization_name": "$hcoName",
            "healthcare_professional_name": "$hcpName",
            "downloaded_hr_on": "$timestamp",
            "downloaded_hr": "$hrName"
        },
        ...
        ]
    }
}
```

Figure 3.7: Sample Audit information logged by the Auditing Management.

sequence for each HCP, and the Prefetch Engine is activated when the sequence length exceeds a *predetermined threshold*.

Then, based on this sequence of requests, the Prefetch Engine, predicts and caches the next health data resource that should be requested by the HCP, as graphically depicted in Figure 8. If the resource to be cached is very large, it is replicated on the prefetch engine so that it can ultimately be forwarded to the Communication

Gateway. This architectural decision is made because the Prefetch Engine should be deployed by the Communication Gateway, and as a result, the transmission delay between the Prefetch Engine and the Communication Gateway is significantly lower than that between the EHR Storage and the Communication Gateway.

In both circumstances, the Communication Gateway is informed to request the resource from the prefetch engine rather than the EHR Storage. Upon completion of the user's request, the data are transmitted directly to the HCP's application from the Prefetch Engine, bypassing the EHR Storage in which they were initially stored. Additionally, if the resource is not requested by the HCP, it is removed from the Prefetch Engine in order to conserve resources. The Prefetch Engine utilizes an adapted version of the proposed Influence-based Dataset Generation method, presented in Chapter 2 for the training of the DL algorithm it encompasses which will be analyzed below.

3.3.7.1 Influence-based Dataset Generation Method Adaptation for the Prefetch Engine

In this Section, the modification of the original Influence-based dataset Generation method to fit the health data prefetching scenario at hand is reviewed. This detailed presentation of the customized adaptation demonstrates the robustness of the original method in accommodating the unique characteristics and requirements of the particular scenario in question. This demonstrates its adaptability and versatility, highlighting its potential for customization across an extensive range of application domains and challenges. In this case, not only are the specific demands addressed, but also the method's validity and potential for wider adoption are evaluated.

To start with, Algorithm 2 presents the adapted method. Similar to the core method there are two main phases; (*i*) the *training* or *online* phase, and (*ii*) the *prediction* or *online* phase, as depicted in Figure 3.9.

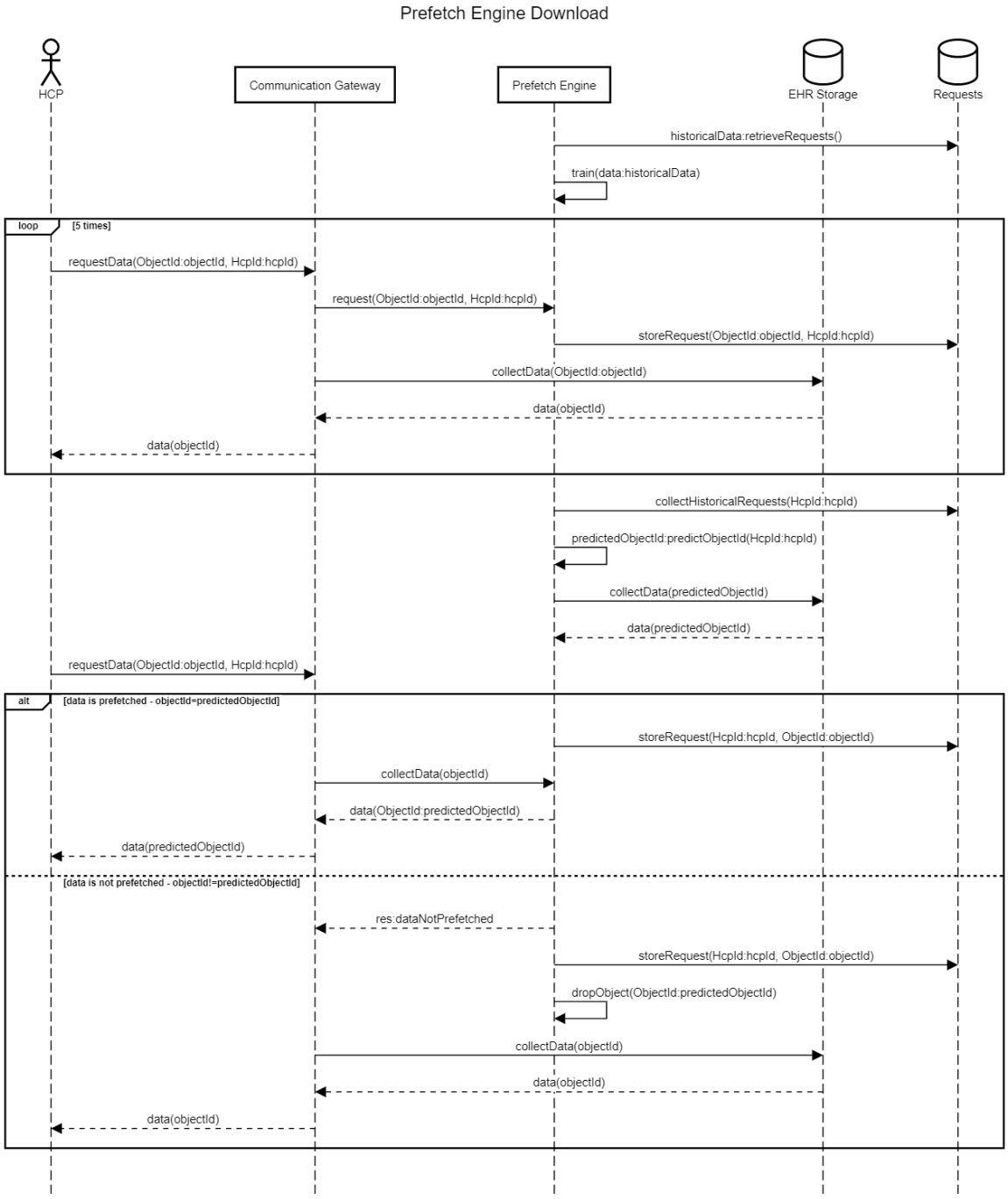


Figure 3.8: Sequence diagram illustrating the behavior of the Prefetch Engine and how health data is downloaded when prefetched versus when not prefetched.

Algorithm 2 Adaptation Influence-based dataset Generation method as it is used in the Prefetch Engine of the Health Storage Cloud service

Auxiliary Variables:

$\mathcal{D}$: the initial dataset of requests performed by HCPs,

$\mathcal{D}_{-i}$: the initial dataset, with the i-th instance removed,

$\mathcal{D}'$: the dataset consisting of all identified influential instances,

$DFBETA[i]$: the $DFBETA$s of the DL algorithm when trained with the original dataset without the i-th instance,

$F1[i]$: the $F1$ of the DL algorithm when trained with the original dataset without the i-th instance,

$influentialInstances$: the dataset constructed of the identified influential instances.

Output:

$influentialInstances$

Algorithm:

1: `fit(`$\mathcal{D}$`)` *// an LSTM model was used in this use case*
2: $GIVEN\_F1 \leftarrow 2 \times \frac{Precision \times Recall}{Precision + Recall}$
3: $influentialInstances = [\,]$
4: **for** $i \in \mathcal{D}$ **do**
5: $\quad \mathcal{D}_{-i} : i \notin \mathcal{D}$
6: $\quad$ `fit(`$\mathcal{D}_{-i}$`)`
7: $\quad DFBETA[i] \leftarrow \beta - \beta^{(-i)}$
8: $\quad F1[i] \leftarrow 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i}$
9: **end for**
10: **for** $i \in \mathcal{D}$ **do**
11: $\quad$ **if** $DFBETA[i] \geq AVG(distance, \mathcal{D})$ && $F1[i] \geq GIVEN\_F1$ **then**
12: $\quad\quad influentialInstances \leftarrow influentialInstances \cup i$
13: $\quad\quad \mathcal{D} : \mathcal{D} - i$
14: $\quad$ **end if**
15: **end for**
16: `kmeans.fit(`$influentialInstances$`,` $k = |influentialInstances|$`,` $distance = DTW$`)`
17: **if** `kmeans.predict(`$i \in \mathcal{D}$`)==cluster && dist(cluster,` $i \in \mathcal{D} \leq GIVEN\_DISTANCE$`)` **then**
18: $\quad influentialInstances \leftarrow influentialInstances \cup i$
19: **end if**
20: **return** $influentialInstances$

3.3.7.1.1 The Offline Phase

The initial phase of the algorithm is the *Offline phase*, or *training phase* as it is described in Chapter 2 - Section 2.2.1. In greater depth, the procedure begins with

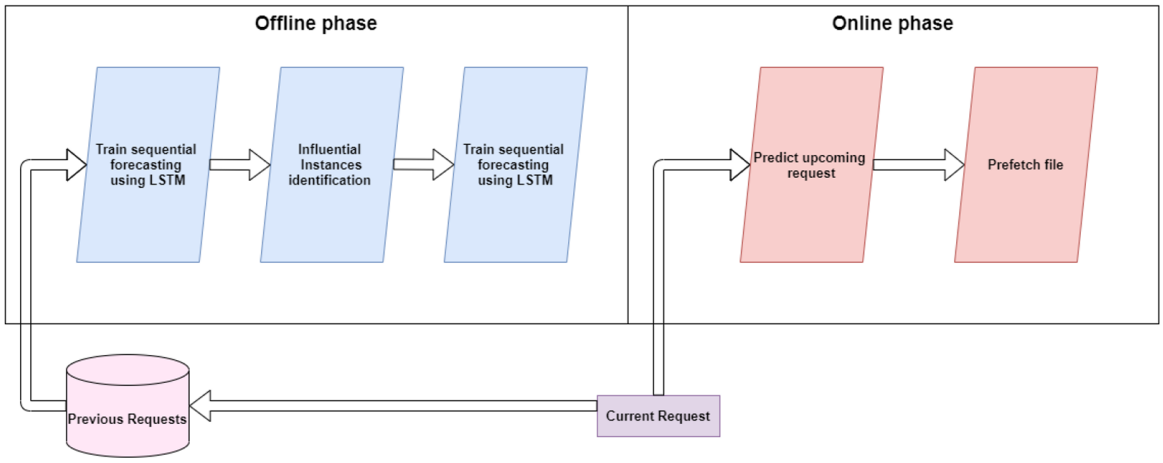


Figure 3.9: The Prefetching Engine overall workflow.

the training of the prediction model using a dataset containing queries for downloading health data, made in the past by users of the Health Storage Cloud. The incoming requests are considered elements of a sequence, where the first element of the sequence is the first request arriving from a specific user and the last element is the latest received request coming from the same user. Parallel to the process of DL model training, which will be examined in greater detail in the following paragraphs, the user's requests are continuously monitored. By observing this process, a list of requested files and a timestamp for each request are generated. Thus, the forthcoming request based on the previous data are predicted.

The DL model that was used regards an Long Short-Term Memory (LSTM) [Karim et al., 2017], since it is proved that it outperforms other statistical models such as Autoregressive Moving Average (ARMA) [Webby and O'Connor, 1996], Autoregressive Integrated Moving Average (ARIMA) [Ho and Xie, 1998], or other DL-based architectures such as Recursive Neural Network (RNN) [Zhang and Man, 1998] when it comes to time-series analysis. LSTM exceeded classical statistical techniques like ARIMA. In addition, LSTM is known to be able to better capture more long-term relationships compared to fundamental RNN models [Gers et al., 2001], while it also manages to address the problem of vanishing Gradient Descent as

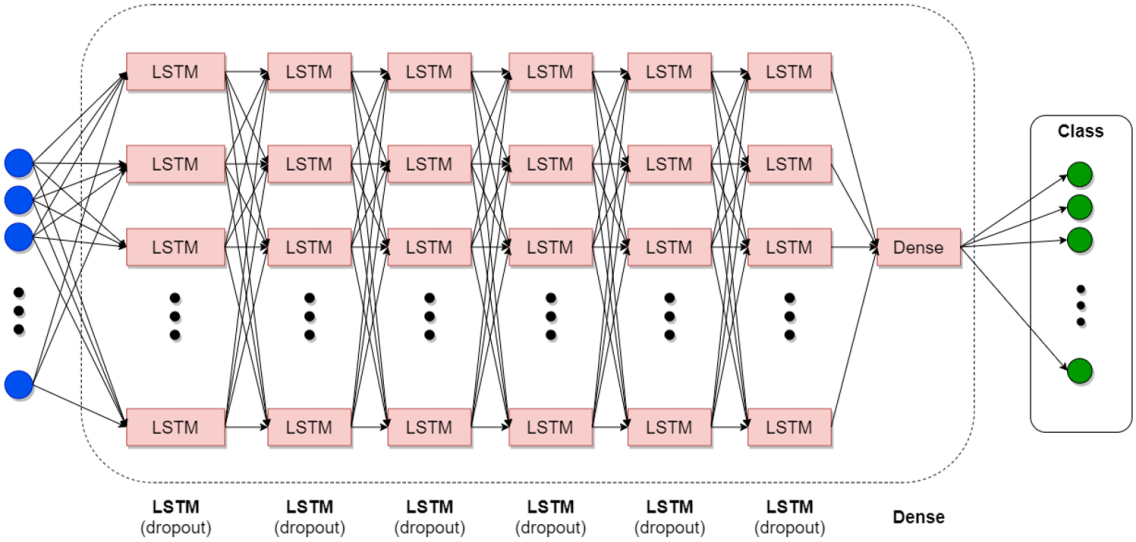


Figure 3.10: Architecture of the LSTM model used for the prediction of the upcoming data item.

opposed to RNNs [Hochreiter et al., 2001].

Regarding its architecture, this model is a seven-layered model composed of six LSTM layers and one dense layer, as depicted in Figure 3.10. To prevent overfitting, dropout layers were added after each LSTM layer with a rate of 0.75 except for the final layer, which had a rate of 0.50. Following experiments with testing datasets, these rates were selected.

Once the model has been trained, the detection of *influential instances* takes place. These constitute the instances (*i.e.*, the user's requests) that have a significant impact on the trained model, and while this method is often applied for machine learning interpretability, the purpose of this scheme is to use it to provide context-aware predictions and improve the model's performance [Karim et al., 2017], an idea that is also utilized in the field of visual analytics on electronic medical records when employing an RNN [Kwon et al., 2018].

As already mentioned in Chapter 2, an instance is considered influential if, when it is removed from the training set and the model is retrained, a significant change is established in the model, the model's parameters (*i.e.*, model weights) are dif-

ferent, and the model's predictions are also different. The greater the identified change to the model and its parameters, the more influential the instance. To determine whether an instance is influential, the model must be trained with and without the instance in question, and the results must be contrasted. In the current scenario, for instance, the LSTM model must be trained with and without an instance in order to determine whether this instance can be considered influential. This is where the adaptation of the core Influence-based dataset Generation method is done.

Typically, influential instances are used to explain the results of difficult-to-interpret machine learning or deep learning models. The current solution, however, differs from the current usage of influential instances in that it employs the influential instances in a manner that empowers the LSTM model by providing additional data-related information. Specifically, it will provide the model with a catalog of instances that are potentially more significant than others, which the model can use to enhance itself.

Usually, the influence of an instance on a model can be found using Cook's distance [Cook, 1977], which calculates the effect on the model's predictions when the model is trained with a dataset that does not include the given instance. Cook's distance is measured as it is defined in Equation 3.1:

$$D_i = \frac{\sum_{j=1}^{n} (\hat{y}_j - \hat{y}_j^{(-i)})^2}{p \times MSE} \tag{3.1}$$

However, given that in the current case a DL is selected, the effectiveness of an instance i on the model's *parameters* is preferred as the first metric to calculate the influence of the instances. In addition, Cook's distance is mostly aimed towards Linear Regression models and not DL models such as the selected one. Thus, the *DFBETA* metric [Adadi and Berrada, 2018], [Mi et al., 2020] was used instead, which is defined in Equation 3.2:

$$DFBETA_i = \beta - \beta^{(-i)} \tag{3.2}$$

where β is the weight vector of the LSTM model, while $\beta^{(-i)}$ is the weight vector of the same model when trained without the instance i.

However, the calculation of *DFBETA* alone cannot provide context regarding the impact of a particular instance on the model and its predictions. Hence, the $F1$ score is also computed in a similar manner, as defined in Equation 3.3:

$$F1[i] = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{3.3}$$

where i is the instance that is removed in order to calculate its influence.

Utilizing *DFBETA* and calculating $F1$ score for each instance, the instance's impact on the model can be determined. With this information, the instances are sorted based on significance, and the most influential ones are then retained for the next step.

The next step concerns the generation of clusters of important instances, based on the ones that were identified from the previous step, as also presented in Figure 3.11. This is accomplished using the k-means [Huang et al., 2016] algorithm. To elaborate, these instances serve as the starting points for the centroids in the k-means algorithm, thereby initiating the training of the algorithm. The default configuration of the k-means algorithm employs the Euclidean distance metric for clustering [Ghosh and Liu, 2009].

Given that the present problem entails time series data, however, the Euclidean distance metric is insufficient. In place of this, we employ the Dynamic Time Warping Matching (DTW) similarity metric [Senin, 2008] instead. In more detail, unlike the Euclidean distance, which implies a linear relationship between data points, DTW is more flexible. It is appropriate for comparing and categorizing time series

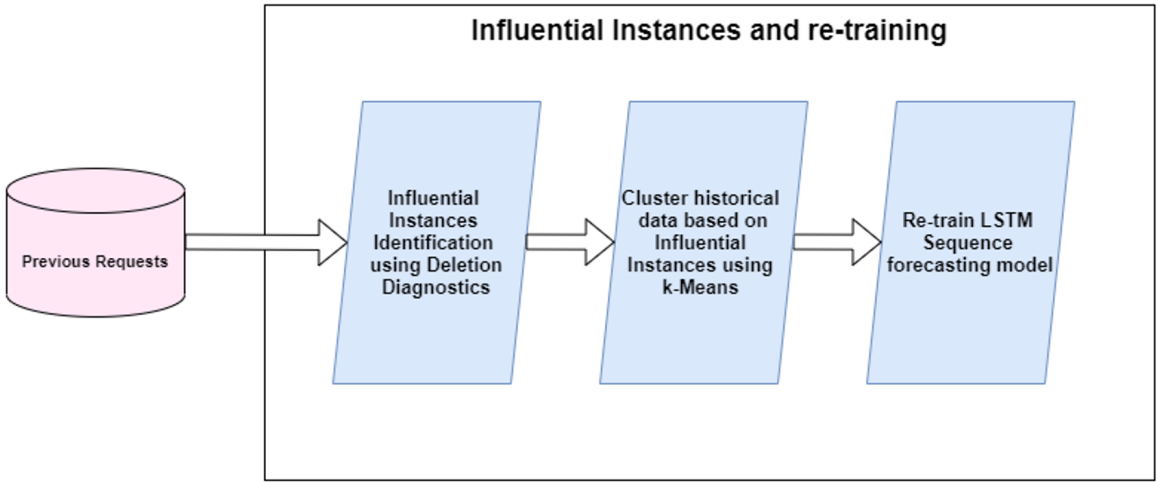


Figure 3.11: Influential instances identification workflow.

data because it accommodates for variations in the timing and rate of data points. DTW enables the comparison of sequences that may be out of sync temporally or have distinct durations. In cases involving time series data, where such variations are typical [Berndt and Clifford, 1994], DTW is used to provide a more accurate and meaningful measure of similarity between data sequences, which is essential for efficient analysis [Müller, 2007].

Consequently, the K-means algorithm is then applied to the instances that were not originally considered as influential. As also shown in Algorithm 2, every instance that is not considered influential, is assigned to its closest cluster by the K-Means algorithm. If the distance between the instance from the centroid of its predicted cluster, falls below a specified threshold, then this instance is also considered influential); otherwise, it is deemed an outlier.

3.3.7.1.2 The Online Phase

When a user sends a series of requests to the storage cloud, the *online phase* (or *prediction phase* as it is called in the generic method in Chapter 2 - Section 2.2.2) begins. Furthermore, the LSTM model anticipates the upcoming request based on this sequence and requests the Health Storage Cloud to prefetch this data in

preparation for transmission.

Upon receiving the request and validating that it matches the prefetched data, the Prefetch Engine exchanges the data directly. In contrast, if the user's request does not match the prefetched data, the prefetched data are erased in order to conserve disk space, and the data are instead retrieved from the EHR Storage. In both cases, the request sequences are archived in a specialized database that keeps a historical record of all request sequences.

3.4 Evaluation Results

The evaluation of the Prefetch Engine was conducted in two distinct phases. The first phase focused on assessing the effectiveness of the Influence-based Dataset Generation method and the training of the LSTM model, while the second phase examined whether the Prefetch Engine successfully enhanced the overall system performance, by reducing the transmission process of the requested health data.

3.4.1 Performance Evaluation of the LSTM model and the Effectiveness of the Influence-based Dataset Generation method

To evaluate the performance of the proposed prefetch strategy, a dataset consisting of 800 observations, comprising a series of HTTP requests was utilized. To evaluate the method's impact, performance experiments were conducted comparing the results when the LSTM network was trained with the original dataset against the results when trained with the enhanced dataset after the utilization of the Influence-based Dataset Generation method. In both cases, the dataset was separated into training and testing sets, with the training set of the original dataset containing 650 observations and the testing set containing 150 observations.

The metrics used for the evaluation, given that the task is considered a classifica-

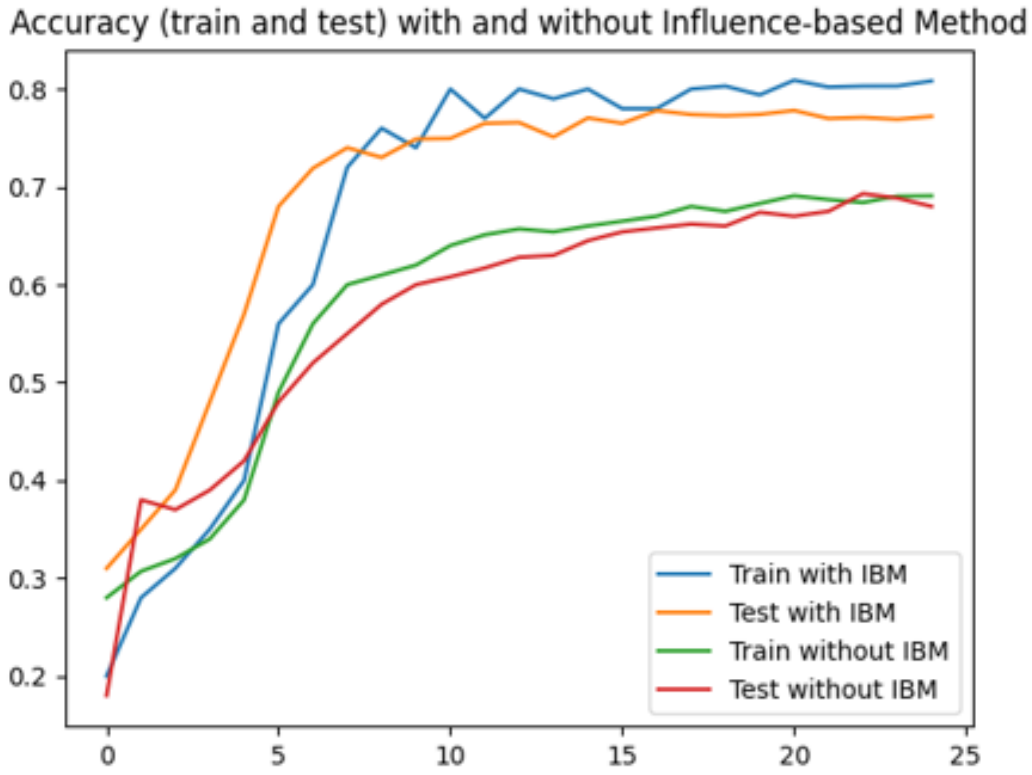


Figure 3.12: Comparison of the accuracy of the model on the train and test sets when trained with the original dataset, against the accuracy when trained with the enhanced dataset, after the utilization of the Influence-based Dataset Generation method.

tion problem where the Accuracy, Precision, Recall and F1-score, as it can also be seen in Table 3.1. Early stopping was implemented in both scenarios as a regularization technique to avoid overfitting; thus, the training phase concluded after 25 epochs. The results show an increase of $\sim 15.5\%$ when it comes to the accuracy (as also depicted in Figure 3.12) of the model. Similar results can be extracted from the other metrics as well. More precisely, when it comes to the F1-score, there is an increase of $\sim$ 17% since the model's F1-score when trained with the original dataset was equal to 0.5766, while it reached 0.6536 when trained with the enhanced dataset.

Table 3.1: Comparison of results when the Incfluence-based Dataset Generation method is utilized vs. when not utilized.

Dataset	Accuracy	Precision	Recall	F1-Score
Initial	0.6992	0.4916	0.6992	0.5766
Influence-based	0.8083	0.6536	0.8083	0.6536

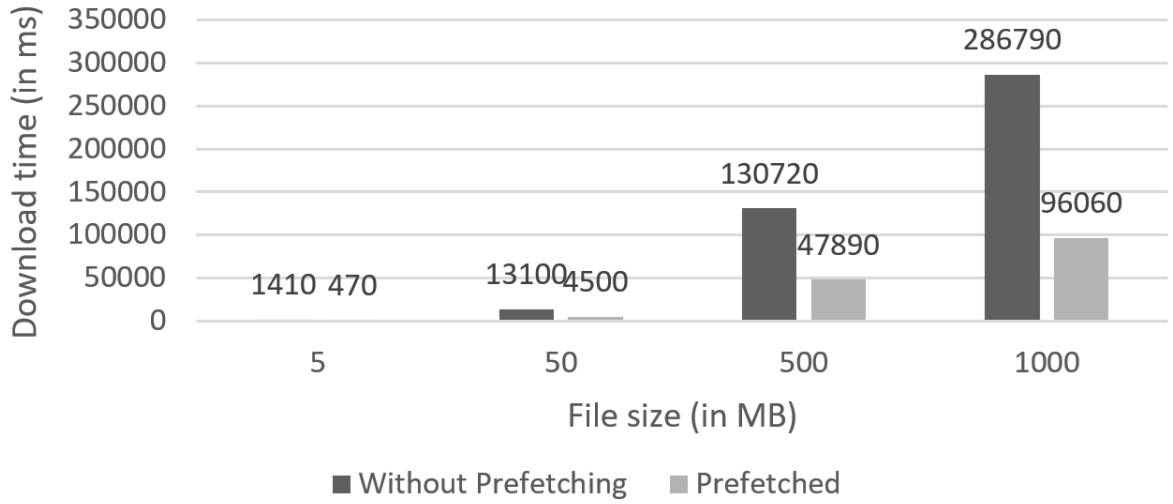


Figure 3.13: Average download times (in ms) of health records with varying sizes of either 5 MB, 50 MB, 500 MB, or 1 GB, when the Health Storage Cloud and the Prefetch Engine were deployed in infrastructures with different physical locations.

3.4.2 Evaluation of the Prefetch Engine

Further than that, the performance of the actual Prefetch Engine was also evaluated. The conducted experiments were centered on evaluating the duration of transmission for health records that were encrypted and varied in size. In more detail, the download time, which is depicted in the following figures, denotes the mean duration that elapses between the healthcare provider's side of initiating a request and the health record download's completion. It is important to point out that the time needed for file decryption was not considered into this evaluation, as it is dependent on the computational capacity of the client's device. Furthermore, regardless of whether the file was transmitted in an encrypted or decrypted state, the transmission time remained constant.

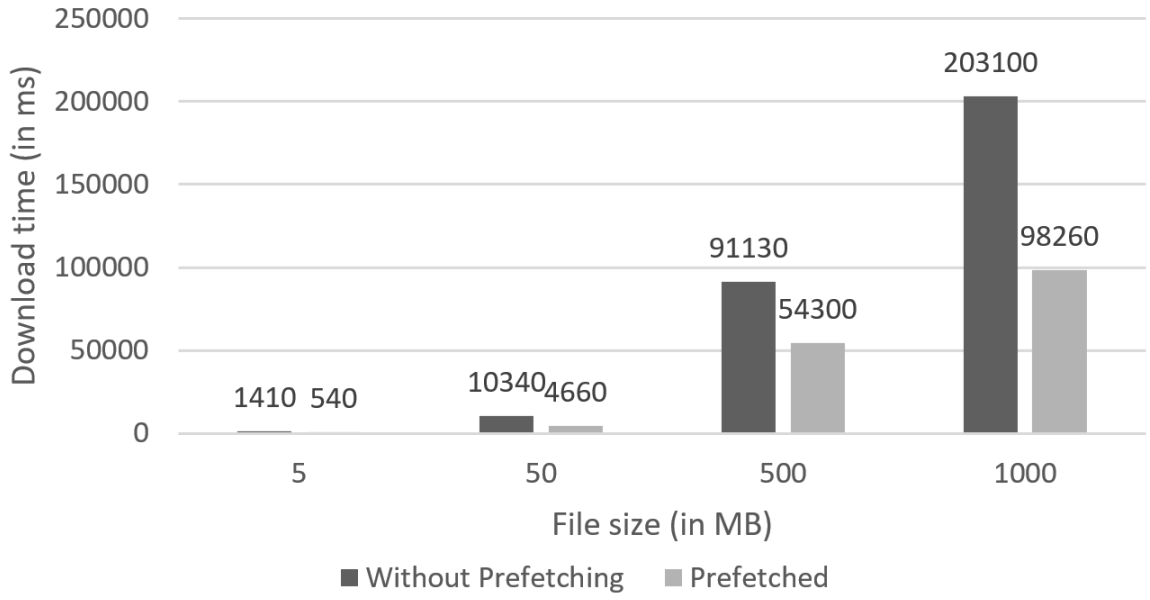


Figure 3.14: Average download times (in ms) of health records with varying sizes of either 5 MB, 50 MB, 500 MB, or 1 GB, when the Health Storage Cloud and the Prefetch Engine were deployed in the same infrastructure but in different VMs.

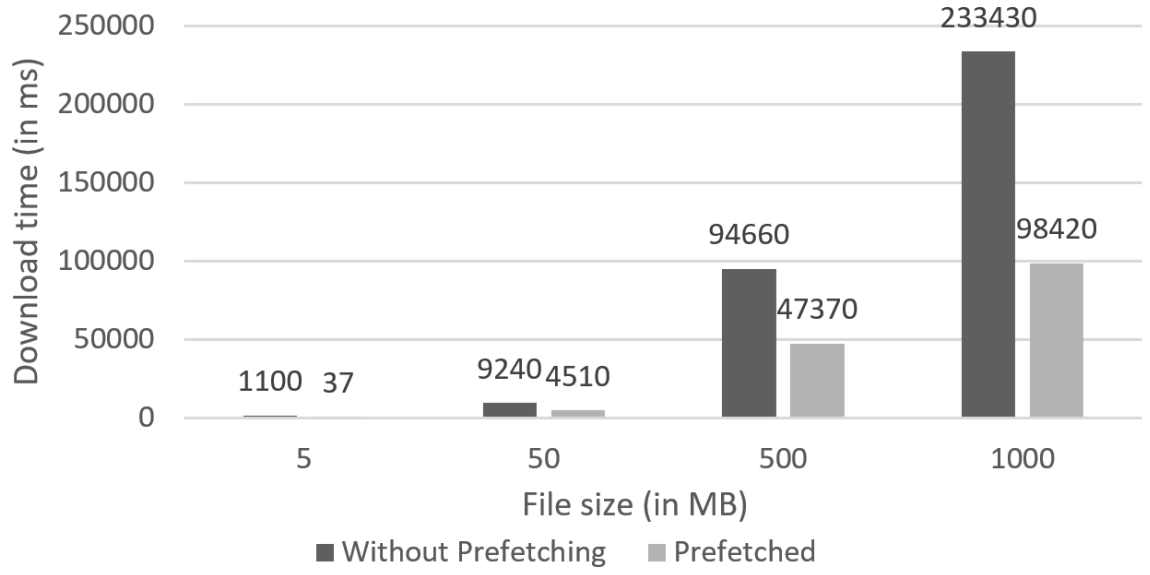


Figure 3.15: Average download times (in ms) of health records with varying sizes of either 5 MB, 50 MB, 500 MB, or 1 GB, when the Health Storage Cloud and the Prefetch Engine were deployed in the same VM.

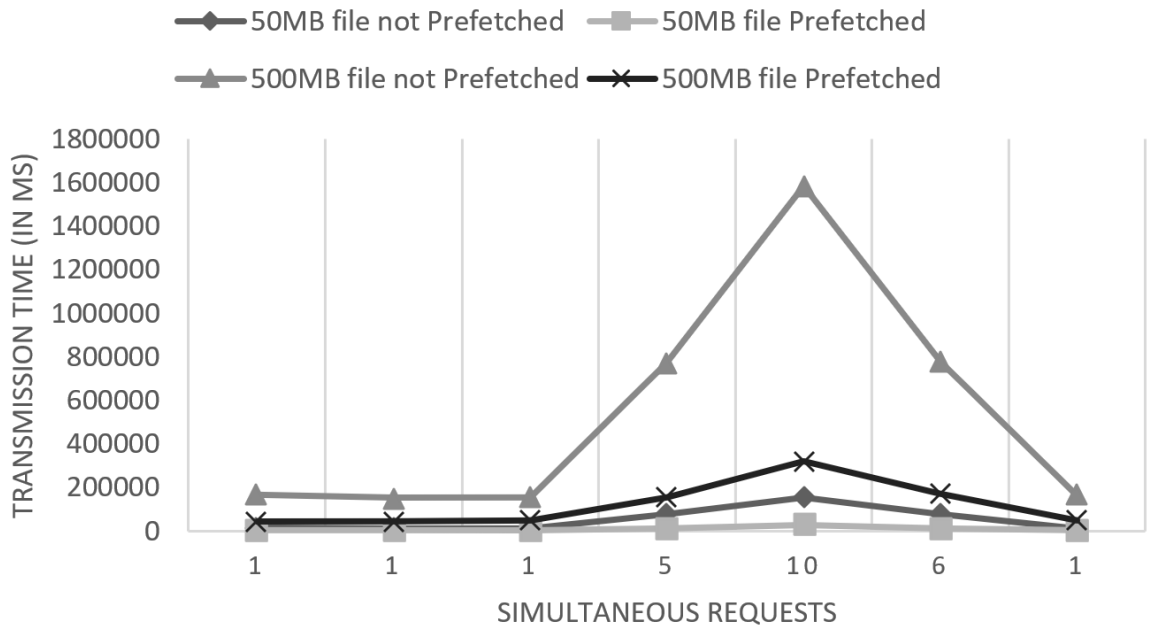


Figure 3.16: Average download times (in ms) of health records with varying sizes of either 50 MB, or 500 MB when simultaneous requests were made in the Health Storage Cloud.

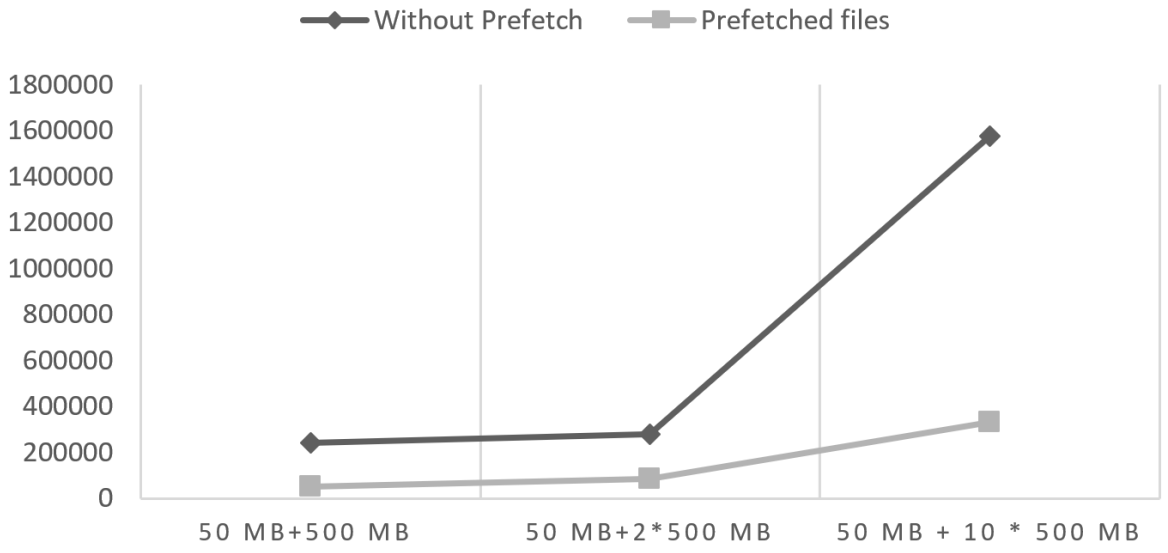


Figure 3.17: Average download times (in ms) when combination of requests for health records were made in the Health Storage Cloud.

The experiments were conducted within three distinct deployment configurations. The first setup entailed the deployment of the storage cloud solution across two VM within the same infrastructure, mirroring a standard cloud deployment with the following hardware specifications: 2x Central Processing Unit (CPU) Intel Xeon CPU E5-2620 v2 @ 2.10 GHz, 8 GB of Double Data Rate version 3 (DDR3) memory, and 60 GB of Hard Disk Drive (HDD). The second experiment involved deploying the cloud solution within a single VM, utilizing the aforementioned hardware resources, reflecting deployments in resource-constrained environments, such as edge nodes. The third configuration featured the storage deployment across two VMs, situated in different data centers, each equipped with identical resources. This particular arrangement aimed to assess the Prefetch Engine's performance in a distributed environment. As for the network capabilities in all experimental setups, the network speed remained consistent. Specifically, the download and upload speeds were both set at 150 Mbps. These varied infrastructure experiments were conducted to benchmark the Health Storage Cloud under different conditions, enabling the formulation of reliable conclusions concerning the utility of the prefetch engine.

The experiments that were performed evaluated the transfer time of a encrypted health records which varied in size. The download time, represented in the following Figures, is the mean duration between when a request is initiated on the client side (*i.e.*, the HCP's side) until the encrypted health record download is concluded. Furthermore, the Prefetch Engine's performance was tested in cases where the clients were performed *sequential* requests and in cases where the clients performed *parallel* requests.

More specifically, starting with the *sequential* requests setup, in Figure 3.13, what is presented, regards the average delay (in ms) of health records of various sizes when the Health Storage Cloud service was deployed in a multi-cloud environment. As it can be observed in the same Figure, the use of the Prefetch Engine

significantly reduces the transmission delay of a health record.

Figure 3.14 presents the average download time of the varying sized health data with the difference that the Health Storage Cloud was in that time deployed in the same cloud infrastructure but in different VMs. Likewise, a significant reduction in the transmission time of health data when the data were prefetched was detected. Adequate results in the evaluation are also observed in the case where the Health Storage Cloud engines were all deployed in the same node (*i.e.*, VM), as depicted in Figure 3.15.

The Prefetch Engine was also tested under *sequential* requests made by several the clients simultaneously. That the following experiments were only conducted in the deployment setting, where Health Storage Cloud was deployed in different computing infrastructure. As depicted in Figure 3.16, when the number of requests is very small the transmission delay is not really affected, yet even with the slightest increase in the number of clients that are connected to the service performing download request, the transmission delay is substantially affected when the Prefetch Engine is absent.

The last experiment that took place tested the performance of the Prefetch Engine in the case where clients requested different data for download at a given time. Similar to the previous experiments, the results, as depicted in Figure 3.17 show that the Prefetch Engine manages to considerably reduce the overall transmission delay.

Chapter 4
Cloud and Edge Computing Optimization: Dynamic Deployment Configuration in Hybrid Cloud / Edge Environments

Chapter Structure

This Chapter is constructed as follows:

- **Section 4.1 - Background on Deployment Configuration Optimization and Optimal Service Placement in Cloud and Edge Computing Environments**, analyzes the State of the Art on Optimal Deployment Configurations identification and Service Placement in Cloud and Edge Computing Environments.
- **Section 4.2 - Overview of the Optimal Deployment Configuration in Edge / Cloud Infrastructures Framework**, presents an overview of the proposed Deployment Configuration framework and highlights the adaptations that were made in order for the proposed Influence-based Dataset Generation method to be incorporated in the given use case.

- **Section 4.3 - Evaluation Results**, evaluates the proposed Deployment Configuration framework and presents the outcomes of this evaluation.

This Chapter presents how the Influence-based Dataset Generation method is utilized in the context of a framework for dynamic configuration of services in hybrid Edge and Computing infrastructures. In more detail, the proposed dataset enhancement method, has been utilized as a pre-processing step in order to train an ML model, responsible for the identification of the optimal deployment configuration for services deployed in Edge / Cloud infrastructures [Symvoulidis et al., 2023b].

This regards an entirely different use case when compared to the scenario presented in Chapter 3 for several reasons:

- **Different ML task and data types**: In the Prefetch Engine, the main task is to *forecast*, using time-series data, the health data that may be requested in advance. In the Deployment configuration prediction scenario, the ML has to determine the most suitable deployment configuration for services deployed in Cloud / Edge infrastructures, which is handled as a *classification* task.

- **Different ML algorithm**: In the Prefetch Engine a DL model is utilized. More precisely, an LSTM model is used, while in second scenario a ML model, and in particular a Random Forest (RF) classifier model.

- **Application focus**: The first use case primarily aims to enhance data access efficiency by prefetching the anticipated health data, thereby reducing potential latency in service requests. On the other hand, the second use case is focused on optimizing the deployment setup within edge infrastructures, with the goal of ensuring efficient resource allocation and enhancing system reliability.

4.1 Background on Deployment Configuration Optimization and Optimal Service Placement in Cloud and Edge Computing Environments

Ever since the establishment of Cloud Computing as the prominent mean of applications deployment, a major increase on the implementation of web-based services and applications which are both resource- and data-intensive has been acknowledged. In addition, Edge Computing has also played a great role on that due, to the abilities it provides. And as a result, this led to an exponential increase of produced data and need for more and more resources in order to meet the users' needs. For this reason, hybrid solutions aiming at taking advantage of both the Cloud and the Edge Computing worlds have emerged [Al Azad et al., 2021], [Yang et al., 2019], but such a transmission to a fully Cloud / Edge environment, unless the issues related to resource allocation are solved first [Warneke and Kao, 2011], [Yousafzai et al., 2017].

The Service placement problem as well as the identification of the optimal deployment configuration in such heterogeneous environments are topics, widely discussed for several reasons. To start with, applications and services deployed on the Cloud, the Edge or even in 5G infrastructures, are known for their diverse nature [Symvoulidis et al., 2019], [Tsoumas et al., 2020] in a way that they are related to large number of parameters, which should be taken into consideration when deciding on the resources which will allocated for their deployment. Furthermore, the devices (such as the IoT devices) have now the ability to process and analyze data closer to the source.

Therefore, an efficient resource allocation strategy is crucial in order to assure that the computing tasks can be performed on time and most important, cost-effectively. Not to mention that, with the establishment of $5G$ and the IoT, new challenges are expected to arise, especially when it comes to real-time adaptation

and re-allocation of resources.

To start with, the authors of [He et al., 2021] proposed a server configuration optimization strategy that handles the optimal server configuration problem as a trade-off between the deployment cost and the system performance (*i.e.*, the average response time). Zhang *et al.* [Zhang et al., 2017] on the other side, proposed an offloading strategy in which a trade-off between energy consumption and latency is considered towards the optimization of service offloading and configuration on Mobile Edge Computing (MEC) infrastructures.

In a similar manner, the authors of [Ouyang et al., 2018] proposed a mobility-aware service placement strategy for Edge Computing infrastructures. The proposed strategy makes placement decisions with the utilization of a cost model whose goal is to maintain Quality of Service (QoS) levels (in terms of latency), by considering specific costs relative to migration, and service performance taking also consideration the user's mobility. A similar solution is proposed by the authors of [Ning et al., 2020], where the service placement decision is handled as an optimization problem. The proposed strategy first utilizes the Lyapunov optimization method to split the problem into several instant optimization problems. Consequently, an approximation-based algorithm is used to estimate the expected upcoming service utilization and, based on this, perform the final decision.

The authors of [Wang et al., 2020] proposed an optimal service placement algorithm for collaborative edge applications, which are in essence applications that can be split into two main sides, the user side and the server side. the former handles the client's state as well as the computation-intensive tasks, while the latter is in charge of collecting the client's data and providing them feedback. For such services, the authors convert the problem to a cost optimization problem in which they consider costs related to placement, proximity, and activation in order to arrive at a final placement decision.

The authors of [Wang et al., 2021b] proposed a service placement strategy for 5G

Industrial Internet of Things (IIoT) MEC infrastructures, where the service placement problem is converted into a particle swarm optimization problem. In more detail, the strategy considers two main factors, the energy consumption of a service and its delay, formulates a cost function that needs to be optimized, and according to this information, selects the most appropriate server for placement at a given time.

Based on the above, it is evident that most State-of-the-Art solutions focus mostly on handling the service placement problem as a trade-off which needs to be optimized and try to figure out solutions that may not always be the best, but instead try to find the best compromise decision. This is especially problematic in cases where the factors are conflicting. The proposed optimal configuration solution on the other side, does not focus on trying to identify a middle solution but instead concentrates on finding the best placement (deployment) configuration for any given service taking into consideration its inherent features and characteristics.

4.2 Overview of the Optimal Deployment Configuration in Edge / Cloud Infrastructures Framework

As described above, the Influence-based Dataset Generation method is utilized in a novel framework responsible for the identification of the optimal deployment configuration of services in Edge and Cloud computing infrastructures.

4.2.1 Core Features of the Deployment Configuration Framework

The framework includes four features, as also depicted in Figure 4.1, which are presented in detail below:

- The *Components Analyzer*, which is responsible for identifying and extracting the core technical and resource-related characteristics of a given service,

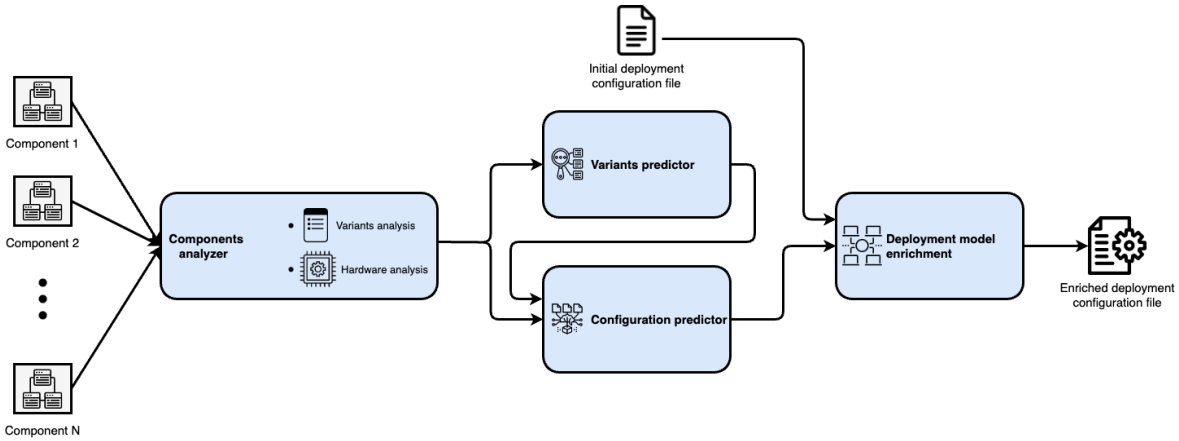


Figure 4.1: Overall architecture of the Deployment Configuration Framework.

- The *Variants Predictor*, used for the identification of the optimal deployment configuration for the given service,
- The *Configuration Predictor* which is responsible for generating the deployment configuration document for the given service based on the outcomes of the Variants Predictor component, and finally,
- The *Deployment Model Enrichment*, which produces the *enriched* deployment configuration for the given service.

4.2.2 The Components Analyzer

The Components Analyzer examines the code of a given service in order to derive valuable information concerning the service's execution context. Through the examination of the service's code base, key features that may be utilized in determining its execution context are discovered. Static code analysis is used in order to extract those key features from the code.

A bag-of-words [Qader et al., 2019] is produced through this procedure, which is comprised of a collection of modules, packages, and libraries that are imported into the code. For instance, containerized Python projects may incorporate pack-

Algorithm 3 Adaptation of the Influence-based Dataset Generation method as it used in Optimal Deployment Configuration framework

Auxiliary Variables:

$\mathcal{D}$: the initial dataset,

$\mathcal{D}_{-i}$: the initial dataset, with the i-th instance removed,

$\mathcal{D}'$: the dataset consisting of all identified influential instances,

$F1$: the F1 score of the ML algorithm when trained with the original dataset,

$F1[i]$: the F1 score of the ML algorithm when trained with the original dataset without the i-th instance,

influentialInstances: the dataset constructed of the identified influential instances,

$\kappa[i]$: Kohen's score of the ML algorithm when trained with the original dataset without the i-th instance.

Output:

influentialInstances

Algorithm:

1: `fit(`$\mathcal{D}$`)`
2: *influentialInstances* = [ ]
3: calculate $\kappa(\mathcal{D})$
4: calculate $F1(\mathcal{D})$
5: **for** $i \in \mathcal{D}$ **do**
6: $\quad \mathcal{D}_{-i} : i \notin \mathcal{D}$
7: $\quad$ `fit(`$\mathcal{D}_{-i}$`)`
8: $\quad F1[i] \leftarrow F1(\mathcal{D}_{-i})$
9: $\quad \kappa[i] \leftarrow \kappa(\mathcal{D}_{-i})$
10: **end for**
11: **for** $i \in \mathcal{D}$ **do**
12: $\quad$ **if** $\kappa[i] \leq \kappa(\mathcal{D})$ && $F1[i] \leq F1(\mathcal{D})$ **then**
13: $\quad\quad$ *influentialInstances* $\leftarrow$ *influentialInstances* $\cup\, i$
14: $\quad\quad \mathcal{D} : \mathcal{D} - i$
15: $\quad$ **end if**
16: **end for**
17: `kmeans.fit(`*influentialInstances*`,` $k = |$*influentialInstances*$|$`)`
18: **if** `kmeans.predict(`$i \in \mathcal{D}$`)==cluster && dist(cluster,` $i \in \mathcal{D} \leq$ *GIVEN_DISTANCE*`)` **then**
19: $\quad$ *influentialInstances* $\leftarrow$ *influentialInstances* $\cup\, i$
20: **end if**
21: **return** *influentialInstances*

ages such as TensorFlow [Google, 2023], NumPy [Numpy, 2023], and Pandas [Pandas, 2023], which are imported within the code of the project. Given that the appli-

cation is containerized, it would additionally incorporate Docker [Docker, 2023a] and Docker Compose [Docker, 2023b]. The dimensions of the bag-of-words may vary across different services. As a result, the zero-padding method is applied to ensure that every generated bag-of-words occupies the identical space.

The generated bag-of-words undergoes a Principal Components Analysis (PCA) [Wold et al., 1987], [Abdi et al., 2013] in order to reduce the feature space and achieve uniformity since the number of identified features, as already explained, may be different for each service. The final result of this procedure regards a vector for the given service, which is then passed as input to the *Variants Predictor*, which will be described in detail below.

4.2.3 The Variants Predictor

The Variants Predictor is responsible for the identification and prediction of the optimal execution context of a given service. The *Variants Predictor*, utilizes an adaptation of the proposed Influence-based dataset as described in Chapter 2. The adaptations that have been made will be described in detail below.

4.2.3.1 Influence-based Dataset Generation method Adaptations for the Variants Predictor

Similar to the generic method, this adaptation is again split into two main phases: the *training* phase and the *prediction* phase.

4.2.3.1.1 The Training Phase

The training phase involves 5 main steps:

(*i*) the training of the ML model with the unchanged dataset,

(*ii*) the calculation of the metrics used for the instances' influence identification, which in this case are the Cohen's Kappa score and the F1-score,

(*iii*) the selection of the most influential instances based on the outcomes of the above-mentioned metrics,

(*iv*) the selection of all influential instances, including those that were not initially identified, and finally,

(*v*) the re-training of the ML model using the new dataset, which now consists of solely influential instances.

In more detail, as also depicted in Algorithm 3, the original dataset, listed as $\mathcal{D}$, is used for the training of the ML model. The baseline Cohen's Kappa score and F1-score are then calculated for the original dataset. Consequently, the process of evaluating the influence of the individual instances in the original dataset is instantiated (*i.e.*, steps *ii* and *iii* of the aforementioned list).

In any instance i of the dataset $\mathcal{D}$, the Cohen's Kappa score (also mentioned as Cohen's Kappa coefficient) is defined as in Equation 4.1:

$$\kappa_i = \frac{p_0 - p_e}{1 - p_e} \tag{4.1}$$

where p_0 regards the agreement between the ground truth (*i.e.*, the actual values in the dataset) and the predicted values of the trained model, and p_e regards the proportion of chance agreement. The Kappa score can be a value of $0 \leq \kappa \leq 1$ [McHugh, 2012]. The closer the Kappa score is, the higher the agreement between the ground truth and the model's predictions, whereas if the score is close to 0, the less agreement there exists between the ground truth and the predictions of the ML model.

Furthermore, the F1-score for any instance i in the dataset $\mathcal{D}$ is defined as in the following Equation 4.2:

$$F1_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{4.2}$$

Upon completion of the above procedure, the most influential instances of the given dataset can be found. Any instance i is considered influential if Cohen's Kappa coefficient $\kappa[i]$ is less than the original Cohen's Kappa score $kappa(\mathcal{D})$, meaning that the agreement between the ground truth and the prediction of the model is reduced if the instance is removed from the dataset, and if the F1-score $F1[i]$ is also less than the original dataset's F1 score $F1(\mathcal{D})$, meaning that the overall accuracy of the ML is decreased if the instance is absent from the dataset.

This process produces the set of the most influential instances out of the dataset $\mathcal{D}$. Yet, the new dataset may be significantly reduced if only these instances are used for the training of the ML model, something that may essentially reduce the model's performance [Barbedo, 2018], [Symvoulidis et al., 2022]. For this reason, using the K-Means algorithm is used, or to be more precise, a generalization of the K-Means algorithm called *Gaussian Mixture Models (GMM)* [Jung et al., 2020], [Reynolds et al., 2009], [Zhang et al., 2021], to identify additional instances of the original dataset $\mathcal{D}$ which may be influential as well but were not recognized in the first phase. In more detail, the originally identified influential instances are provided as input to the K-Means algorithm for its training. Consequently, the instances of the dataset $\mathcal{D}$ which were not originally identified as influential, are passed as input to the GMM and, if clustered in any the available clusters with a high probability, they are considered influential too.

The final dataset $\mathcal{D}'$ is then generated, which is comprised of the influential instances that were initially found, along with the ones found by the K-Means algorithm. The new dataset $\mathcal{D}'$ is the one that is then used for training the ML model.

A sample of the final dataset is depicted in Table 4.1. In the *labels* column, a vector of configurations selected by the application provider or the application developer is present. In more detail, this vector's dimensions are equal to the number of application configurations available. For every service (*i.e.*, every service is an instance in the dataset), if the configuration is selected it is marked with 1, whereas

if it is not selected it is marked as 0 in the corresponding index of the array. The configurations (or characteristics) can be split into two main categories: (*i*) the application's variant, which regards the execution type (*e.g.*, it should be deployed as a Docker container, in a VM, or in a High-Performance Computing center), and (*ii*) in the hardware requirements, which regard the necessary hardware needed for the execution of the application or service (*e.g.*, an application may require Graphical Processing Unit (GPU), a Field Programmable Gate Array (FPGA), a CPU, or Tensor Processing Unit (TPU), etc. in order to be executed properly). A service or an application may have more than one configurations enabled, *e.g.*, a service utilizing a ML or a DL model may require GPUs for training, while it may require deployment as a Docker container.

Table 4.1: Features generated from Components Analyzer.

id	**feature_1**	**feature_2**	**feature_3**	**...**	**feature_100**	**labels**
1	0.8662882236289067	0.4948947564279651	0.1117908516440862	...	0.0016575690974061	[1,0,0,1,...,1]
2	0.2801278760588774	-0.3768388874846627	-0.1786334658385676	...	-0.0025667912344088	[0,0,0,1,...,1]
3	0.4584394177284303	-0.2221757257954971	0.4164187314335896	...	0.0814415288341813	[1,1,0,1,...,0]
4	0.8724889756687922	-0.3850002938928947	0.6681803513604484	...	0.0043081516691631	[1,0,0,1,...,0]
5	0.8623929920456005	0.372267125496619	0.0006769234066742	...	0.004228546139617	[0,0,0,1,...,1]
6	0.8208299621612967	-0.4191173731824708	-0.1330643380947467	...	-0.001447499256128	[0,0,0,1,...,1]
7	0.8604778764407413	0.4820758063978614	0.1132148573161234	...	0.0062417955775234	[1,0,0,1,...,1]
8	0.8816712078784504	0.3179218124276765	-0.0320968508223609	...	0.0084860448516899	[0,0,0,1,...,1]
9	0.739789842562039	0.3253980024470399	-0.0326223828797936	...	0.0066267363615557	[0,0,0,1,...,1]
...	...	...	...	...	...	...
389,000	0.1642750341691848	-0.0798884919720658	0.0539745709775505	...	0.0326708300771116	[1,0,0,0,...,0]

4.2.3.1.2 The Prediction Phase

The *prediction phase* involves the utilization of the final ML model which has been trained using the method as described in the *training* phase. More specifically, the procedure starts with the analysis of the service's code by the Components Analyzer component, which produces the vector of the given service. This vector is then passed to the Variants predictor component, and the outcome refers to the vector of predicted configurations for the given service.

4.2.4 The Configuration Predictor

The Configuration Predictor is responsible for generating the deployment model which can be used at a later time by the orchestrator for the deployment of the service. This deployment model is created using the Cloud Application Modelling and Execution Language (CAMEL) [CAMEL, 2023], a Domain Specific Language (DSL) used for specifying deployment, scalability, security, etc. requirements in multi-cloud, edge or hybrid infrastructures [Totow Tom-At et al., 2024]. The generated deployment model includes all necessary information needed for the proper deployment of a service, including instantiation information such as the number of CPU cores or the amount Random Access Memory (RAM) or the fact that it the application can be containerized. In addition, the CAMEL model includes information related to the required hardware or infrastructure (as already mentioned, a service may require the existence of GPU, TPU, or that it may require being deployed in an FPGA).

4.2.5 Deployment Model Enrichment

This is the final feature of the proposed Optimal Deployment Configuration and framework. The input for the enrichment of the deployment model is the produced deployment configuration generated by the Configuration Predictor, along with the application developer's initial configuration and the *enriched* configuration model is produced, again in CAMEL format. The *enriched* configuration is actually the final configuration which combines necessary requirements placed by the application developer and additional predicted information derived by the model generated by the Configuration Predictor component CAMEL model.

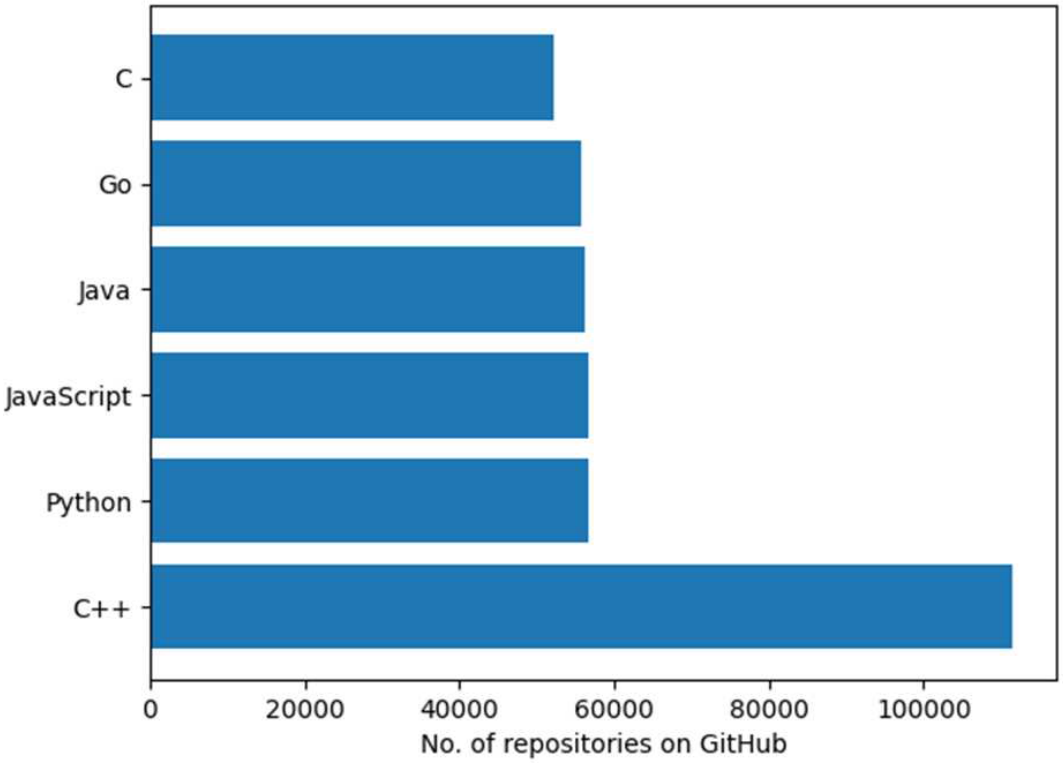


Figure 4.2: Number of repositories per programming language.

4.3 Evaluation Results

In order to evaluate the proposed Optimal Deployment Configuration framework, a dataset consisting of 389, 000 applications and services from public repositories on GitHub [GitHub, 2023a] was created. The dataset was collected using GitHub's Representational State Transfer (REST) Application Programming Interface (API) [GitHub, 2023b].

The dataset includes public repositories developed in six different programming languages (*i.e.*, C, C++, Go, Java, JavaScript, and Python). As depicted also in Figure 4.2, on average, around 50, 000 – 60, 000 repositories were collected for each programming language apart from C++, where the collected repositories were around 111, 000.

The goal of the evaluation was to examine the behavior and overall performance of the proposed framework in terms of how accurately it can identify and predict the various deployment configurations for the given services. Given that, this regards a multi-label classification, since the goal is to mainly check if the Variants Predictor can predict the necessary configurations (*i.e.*, correctly predict the configuration vector as described above). For this reason, the four key metrics used for the evaluation of classification problems were utilized; Accuracy, Precision, Recall and F1-Score, as they are defined in Equations 4.3, 4.4, 4.5, and 4.6 accordingly:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.6}$$

Table 4.2: Datasets evaluation metrics using the Initial dataset and the Influence-based dataset.

Dataset	Accuracy	Precision	Recall	F1-Score	Training time (in sec)	Dataset size
Initial	0.843	0.794	0.838	0.815	71.535	311,200
Influential instances	0.916	0.874	0.923	0.898	51.034	217,840

The selected ML model was a Random Forest Classifier with 1000 trees. Furthermore, the final predicted configurations were those that were predicted by the Random Forest Classifier with a probability greater than 50%.

80% of the dataset was utilized for training purposes (*i.e.*, 311, 200 instances), and the remaining 20% was allocated for testing (*i.e.*, 77, 800 instances). Initially, the

Random Forest Classifier was trained using the whole training set, as also depicted in Table 4.2. Using the Influence-based Dataset Generation method, a dataset consisting of only influential instances was generated, which yielded a dataset decrease of 30% (the influential instances dataset was comprised of 217,840 instances instead). Not only that, the model's performance was significantly improved since the accuracy was increased by $\simeq$ 7%, and the F1-Score by $\simeq$ 8%. In general, this evaluation proved that the proposed framework, when utilizing the adapted Influence-based Dataset Generation method, manages to successfully predict the deployment configurations for services regardless of the programming language in which they were implemented.

4.4 Discussion of the Evaluation Outcomes and the Effectiveness of the Influence-based Dataset Generation method

The Influence-based Dataset Generation method was put under evaluation in three entirely different use cases, presented in Chapters 3 and 4. In the first case, it was used as a pre-processing step towards the creation of a dataset used for the training of a DL model which predicts the data items to be pre-fetched in a Health Storage Cloud service. On the other hand, in the second case, it was utilized for the creation of a dataset used for the fitting of a ML model used for the prediction of the optimal deployment configuration of services in hybrid Cloud / Edge Computing infrastructures. In the third case, it was used in collaboration with the Causal Features Generation method as presented in Chapter 5. In this case, an ML model was designed which identified the consumption of important metrics during a service's execution such as CPU and memory consumption and based on them perform dynamic resource allocation.

Notably, the outcomes of the scenarios in which the method was utilized, revealed an overall increase in the performance of the models, independent of the model architectures, be it a DL model (such as the LSTM model for time-series classification), or a traditional ML model (such as the Random Forest Classifier).

This highlights the proposed Influence-based Dataset Generation method's ability while it shows that it is not limited to a specific model type or problem domain, yet on the contrary, it effectively addresses underlying data-related challenges that are prevalent across different modeling approaches.

In the case of the ML model for classification, the method facilitated better generalization by reducing noise and emphasizing crucial instances for decision-making, and at the same time, the method showed how flexible it was when used with the LSTM model for time series classification, where it worked well at finding and using important temporal patterns in the data.

The extracted outcomes from both scenarios highlight the robustness and versatility of the proposed method, while its ability to enhance performance regardless of the model architecture that was utilized, and the problem type highlights its significance as a valuable pre-processing step in any ML (either traditional or DL) model training pipeline.

Chapter 5

Causal Features Generation method

Chapter Structure

This Chapter is constructed as follows:

- **Section 5.1 - Background on Causality**, analyzes the State of the Art related to Causality, Causal Inference, and Causal Discovery.
- **Section 5.2 - Causal Features Generation Architecture**, presents the architecture of the proposed Causal Features Generation method.
- **Section 5.3 - Evaluation of the Causal Features Generation method**, provides a high-level view of the evaluation scenario in which the proposed method has been put under.

This Chapter describes the architecture of the proposed Causal Features Generation method. Before describing the proposed method, it is important to highlight why causal features are important, and in what ways they affect the performance of AI models. To start with, there are several reasons, which will be analyzed below:

(*i*) To improve the *interpretability* of ML models [Moraffah et al., 2020], by combining meaningful causal features with the original ones. This can be especially important, especially in scientific areas where model interpretability is

crucial. In more detail, by incorporating such features in a dataset, the cause-and-effect relationships among the features are highlighted. Thus, when such features are consolidated in a dataset that trains an ML model, it becomes easier to interpret the impact (influence) that the variables have in the model's predictions. In other words, the inclusion of such causal features in a dataset makes it easier to determine which features drive the outcomes of the model.

(*ii*) To gain deeper insights into the *causal relationships* between features (*a.k.a.*, variables) in the dataset [Guyon et al., 2007]. The added features could reveal hidden patterns or dependencies. Not only that, hidden or uncovered relationships among the features of a dataset may also be highlighted, while confounding variables can also be identified more efficiently and handled better.

(*iii*) To enhance the *performance* of ML models [Wen et al., 2021] by incorporating causal features that capture relationships and dependencies within the data. Causal features usually represent more meaningful relationships among the features, which can facilitate the ML model to better recognize how the features impact one another. Furthermore, increased performance may lead to better interpretation of the outcomes of a ML model, while it will lead to a model whose generalization ability is increased, meaning that it can perform better in unseen data.

(*iv*) To improve the *robustness* of ML models [Sun et al., 2015] by incorporating causal features that make predictions more resilient to changes in the data or underlying system. This happens, since the causal features can help a ML or DL model by making it less likely to overfit. Another point which leads to increased robustness has to do with the increased generalization of the model as it described above. Finally, when including causal features in a dataset, it allows the model to better identify the noise in the data.

To this end, the proposed Causal Features Generation method has three main goals:

(*i*) The identification and extraction of the *causal relationships* among the features of a given dataset,

(*ii*) The creation of new *causal features* according to the identified causal relationships, and finally,

(*iii*) The generation of a *dataset enhanced with causal features*, which can be used for the training of AI algorithms.

5.1 Background on Causality

This Section provides a comprehensive exploration of the foundational concepts surrounding causality, offering a solid background for understanding the nuanced aspects of causal inference and causal discovery. The following Subsections delve into various dimensions of causality, elucidating key principles and methodologies. Subsection 5.1.1 introduces the fundamental concept of causality, laying the groundwork for the discussion of the topics of Causal Inference and Causal Discovery. Subsection 5.1.2 discusses essential aspects of Causal Inference, addressing topics such as Counterfactuals, and Confounding, among others, topics of significant importance when it comes to the identification of Causal Relationships. Subsection 5.1.3 focuses on presenting Causal Discovery methodologies, tools and techniques for uncovering causal relationships.

5.1.1 What is Causality?

This Section describes the concept of Causality and sets the ground for the following Sections, where advanced methodologies and techniques towards Causal Relationships identification and Causal Discovery will be presented.

5.1.1.1 The History of Causality

In the exploration of causality, philosophical perspectives have played a pivotal role in shaping our understanding of how events unfold and intertwine. The concept of Causality originated in Ancient Greece and Plato who stated the principle: *"everything that becomes or changes must be so owning to some causes"* [Lee et al., 1971]. Aristotle was the first philosopher to give a thorough explanation of what Cause is in *Posterior Analytics*, *Physics* and *Metaphysics* [Falcon, 2006]. In more detail, Aristotle stated, for example, in his Posterior Analytics, that knowing a thing entails knowing its cause. The context always includes both a definite being and the conditions of knowledge of that being.

According to Aristotle, there are four possible answers to the question "What is this?" for any unique entity. These answers are all related to what he defined as a "cause," which is defined as *"something without which the thing would not be"*. The answers have been designated as the material cause, the efficient cause, the final cause, and the formal cause. Although a comprehensive response to the initial question would have included all four responses and consequently all four causes, Aristotle maintained that the *formal cause* was the primary and most significant one.

In the Middle Ages, Aristotle's "unmoved mover" was changed to a "creating cause of existence", when most Christian philosophers tried to bring together Aristotle's theory with the Christian belief that God created the world out of nothing [Van Huyssteen, 2003].

Forwarding in the 18th century, Isaac Newton stated that *"causes are forces or constraints that compel moving bodies to behave differently than they would have done without them"*, or in other words that cause means that objects were constrained [Smith, 2007] and is said to reject the principle of universal causation. Similarly, John Locke supported the singularist theory of causality and rejected the modern

view of causation which declared that causation involves consistency or the necessity of link between two objects.

The modern view on Causality originated from David Hume who declared that causation involves the concept of necessity, and in more detail asserted that a causal relation is characterized by three elements: (*i*) continuity of cause and effect (both in space and time), (*ii*) that the cause precedes the effect, and (*iii*) a necessary connection between the cause and the effect must exist. Also, Immanuel Kant claimed that causality was an *a priori* idea in an effort to defend it. In more detail in *Critique of Pure Reason*, Kant argued that some concepts, including causality and substance are universally true with relation to possible experience.

In the more recent years causality evolved from an abstract, philosophical concept to one that is more exact and quantifiable [Guo et al., 2020]. Data mining, ML, Statistics, and various other scientific fields are now used to look for possible cause-and-effect links in observational data and causality is regarded as a factor in the creation of events, in which a cause causes an effect, with the second occurring as a result of the first. Causality, as it will be demonstrated in the following Sections, can be further explained by two key concepts; Causal Inference (see Section 5.1.2) and Causal Discovery (see Section 5.1.3).

5.1.1.2 Correlation and Causation

In this Section, it is crucial to highlight the distinction between two concepts: correlation and causation. Causation refers to a direct cause-and-effect relationship, whereas correlation, on the other hand, represents a statistical association between two variables. Another important distinction is that even if correlation between variables exists can be proved, this does not prove the existence of a cause-and-effect relationship between those variables.

A highly discussed example to prove that correlation does not imply causation is the one regarding the positive correlation between Nobel Laureates and Chocolate

Figure 5.1: A typical example of a cause-and-effect relationship, concerns the atmospheric pressure inside an Espresso machine and the BAR indicator of an Espresso machine. There exists a causal relationship between the two, since only the first causes the second and not vice versa.
Image AI generated using Imagine's AI Art Generator tool [Imagine, 2023].

consumption per capita [Messerli, 2012], [Prinz, 2020], while there is a negative correlation between Coffee consumption and winning a Nobel Prize. Even if there is a strong correlation between these facts, there is no causal relationship among them, meaning that, eating more chocolate or drinking less coffee will not lead to winning a Nobel prize.

On the other hand, a typical cause-and-effect relationship can be found between an Espresso Machine's BAR pressure indicator and the atmospheric pressure within the machine. The BAR indicator shows the atmospheric pressure within the machine, and the higher the pressure in the machine goes, the higher this indicator points to. This is a clear cause-and-effect relationship since the cause (*i.e.*, the

atmospheric in the Espresso machine) leads to the effect (*i.e.*, the BAR indicator shoes a higher number). This relationship is not bidirectional, in the sense that, even if someone changes the BAR indicator and turn it to a higher atmospheric pressure point, the pressure in the machine will not change too.

5.1.2 Causal Inference

Causal Inference is a research topic important to many fields such as Statistics, Computer Science, Education, Economics, among others, which studies the effects that exist when changes are performed in a system.

Causal Inference involves several key concepts and elements, such as Counterfactuals and Confounding which will be further analyzed in the following Subsections. Particularly in the case of Confounding, methods for addressing it will also be highlighted.

5.1.2.1 Counterfactuals

One of the fundamental concepts of Causal Inference regards the Counterfactuals [Yao et al., 2021]. This is an element that allows one to examine what would have happened, had the conditions been different [Rubin, 2010]. These include hypothetical scenarios that never happened and can measure the causal impact of a treatment on individuals, The way to do that is to compare what actually happened (the observed outcome) against what would have happened had the circumstances be different (the counterfactual outcome).

The observed outcome is often denoted as $Y_i(1)$ for an individual i, representing the outcome when the individual was exposed to the treatment (1), while the counterfactual outcome is denoted as $Y_i(0)$ when the individual was not exposed to the treatment. Hence, the causal effect of a treatment is defined as the difference between the observed outcome and the counterfactual outcome: $Y_i(1) - Y_i(0)$.

5.1.2.2 Confounding

Confounding is a key concept in Causal Inference [Athey and Imbens, 2015]. Confounding Variables are attributes in data which distort the relationship between a cause (*i.e.*, an independent variable) and the effect (*i.e.*, the dependent variable). Confounding Variables are very important to control for, since they may lead to incorrect conclusions about the identified cause-and-effect relationships. In order to control for and address confounding, several techniques and methodologies exist, which will detailed in the following paragraphs.

5.1.2.2.1 Instrumental Variables

Instrumental Variables (IV) [Greenland, 2000], [Martens et al., 2006] are one of the most frequently used statistics tool in order to control for confounding variables in observational studies. An IV is a variable that is correlated to the independent variable (*i.e.*, the cause) but not correlated to the dependent variable (*i.e.*, the effect) and can be used as an alternative to the independent variable in order to identify the impact it has on the dependent variable.

5.1.2.2.2 Randomized Control Trials

Randomized Controlled Trials (RCT) [Stanley, 2007] is considered one of the gold standards for evaluating the causal impact of a treatment [Bhide et al., 2018], or an intervention on a particular outcome. In an RCT, participants are assigned at random to either the treatment or the control group. This randomization reduces the possibility of confounding [Jager et al., 2008] by distributing possible confounders equally across the groups, ensuring that the treatment and control groups are equivalent in terms of both observable and unobserved features.

5.1.2.2.3 Matching

Matching is a statistics technique which can be used to control for confounding

variables in observational studies by creating more proportional control and treatment groups. Matching aims at balancing the distribution of confounding variables between the two groups using some of the core aspects of RCTs.

Initially, variables that may be confounding are selected. Afterwards, some of them are selected for matching. The selected matching variables need to be relative to the outcome of the study, meaning that the should have an impact to the outcome of the study. Consequently, the participants of the two groups are matched to each other using any of the following methods. There exist different kinds of matching techniques including (*i*) One-to-One matching where every participant from the treatment group is directly matched to a participant of the control group, (*ii*) One-to-Many matching where a participant of the treatment group may be matched to more than one participants of the control group or vice versa, and finally (*iii*) Nearest-Neighbor matching where the participants of the treatment group are matched with the participant of the control group that has the most similar values to them. Finally, the results for each maching group are evaluated in order to estimate the treatment effect.

5.1.2.2.4 Propensity Score Matching

Propensity Score Matching is a specific type of matching, particularly useful in studies where multiple covariates exist and need to be considered. This technique allows to condense the information from all this variables into one variable; the propensity score [Caliendo and Kopeinig, 2008]. The propensity score measures the probability of receiving the treatment based on observed confounding variables. Propensity Score Matching follows a similar approach as in matching, but in this case, the participants of the study are matched according to the their Propensity Score. As in Matching, Propensity Score Matching helps when controlling for confounding since it enables the creation of balanced control and treatment groups with similar distributions of the identified confounding variables [Austin,

2011], [Stürmer et al., 2014].

5.1.2.2.5 Stratification

Stratification is a statistics technique used in observational studies in order to control confounders [Hoggart et al., 2003]. Stratification works by creating *strata* or subgroups of the population that participates in a study which have similar values in the identified confounding variables. This allows for more accurate comparisons between the treatment and control groups.

Similar to Matching, in Stratification potential confounders are initially identified [Tripepi et al., 2010] as well. Out of those potential confounding variables, the stratification variables are selected and according to these, strata of the study population are generated accordingly. Then, for each stratum, the control and treatment groups are compared in order to identify the treatment effect, and given that the strata include individuals with similar values in the identified confounding variables which were selected as stratification variables, this reduces the impact of the confounding variable in the stratum [Jager et al., 2008], [Kahlert et al., 2017].

5.1.2.2.6 Difference-in-Differences

Difference-in-Differences (DiD) [Goodman-Bacon, 2021] is yet another statistics technique used for the estimation causal effect of a treatment or a an intervention in observational studies, and is especially useful when randomized trials are not feasible, but there is a need for controlling for potential confounders which may exist in the treatment and control groups.

The basic idea behind DiD is that both groups are supervised over time and the changes that may exist over time are tracked and compared [Athey and Imbens, 2006]. Data from both groups are collected several times over the lifecycle of the study, and the changes that may occur are evaluated and the treatment effect is evaluated. DiD facilitate in controlling for confounding variables since it counts

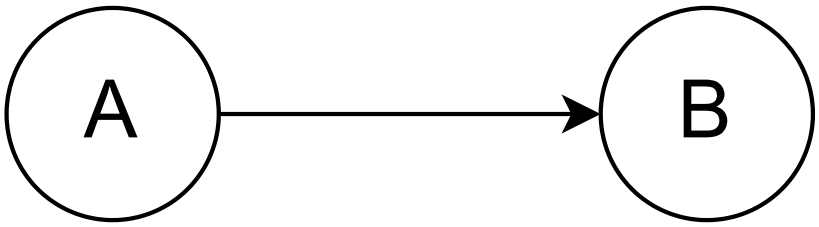


Figure 5.2: In this example of a causal graph, attribute A is causing attribute B.

on the assumption that both groups would have produced the same over time if both had received or not received the treatment. In other words, in DiD the control group can act as a Counterfactuals [Streeter et al., 2017], [Zeldow and Hatfield, 2021] which also allows for controlling for time-related confounding variables.

5.1.2.2.7 Causal Graphs

Causal Graphs or otherwise known as Causal Diagrams are Directed Acyclic Graph (DAG) which represent the causal relationships among variables in a system [Rohrer, 2018]. Causal Graphs can be particularly useful for the identification and controlling for potential confounding [Howards et al., 2012]. In a Causal Graph, the nodes of the graph represent the features of the system and the directed edges ($\rightarrow$) the causal relationship between the attributes. In addition, as their name suggests these graphs do not have any circles (acyclicity). For example, consider a graph $G = (V, E)$, where V is set of nodes of the graph which represent the attributes of the system and E the edges of the graph. As shown in Figure 5.2, variable A is a direct cause of variable B.

Causal Graphs are very useful when it comes to identifying confounding variables, since it can be done in a visible way. As also depicted in Figure 5.3, variable B is a common cause for both the independent variable A and the dependent variable B, which may lead to a *spurious* relationship (*i.e.*, wrong causal relationship between the two variables). Causal Graphs can also facilitate on addressing for confouding once identified in numerous ways [Elwert, 2013], [Law et al., 2012], the most

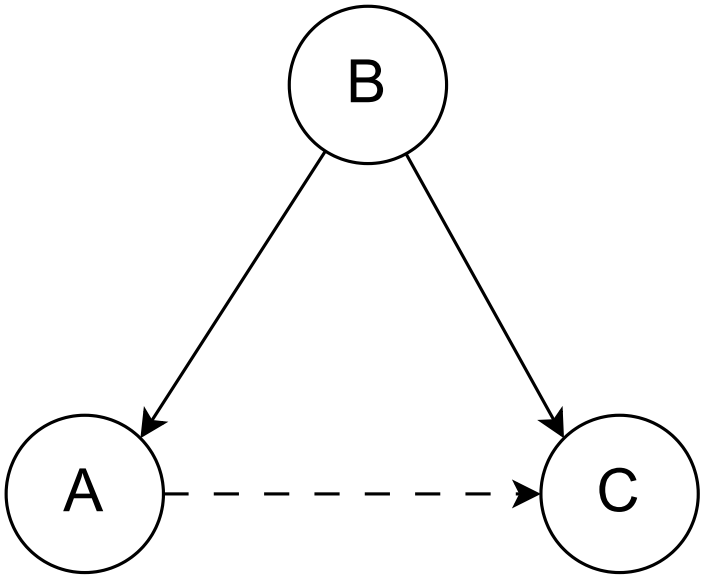


Figure 5.3: In this example of a causal graph, attribute B is a confounding variable since it influences both the independent variable A and the dependent variable C causing a *spurious* association between the two.

common being:

- blocking confounding paths,
- using back-door and front-door criteria, or
- performing sensitivity analysis.

5.1.3 Causal Discovery

Causal Discovery is a field relative to Causal Inference but its primary goal is to infer the causal relationships among the variables which often occurs through Causal Graphs. Causal discovery techniques, like Causal Inference, attempt to reveal the causal relationships between variables without the need for experimental manipulations. Following a number of Causal Discovery methods will be discussed.

5.1.3.1 Causal Bayesian Networks

Causal Bayesian Networks (CBN) are Graphical Causal Models which combine elements of Bayesian networks [Chiappa and Isaac, 2019] in order to represent causal relationships between variables in the data [Pearl, 1985]. The CBN are graphs where the nodes represent the features of the system and the directed edges the causal relationships between them.

Several CBN-based causal algorithms have been proposed in the literature, which in general can be divided to *constraint-based* and *score-based*. Constraint-based algorithms on the one hand, exploit independence tests (such as G-square, Chi-square of Fisher-z test) in order to create edge constraints on the causal graph of observational data [Spirtes et al., 2000]. Score-based algorithms on the other hand, generate Causal Graphs which are then scored using specific scores such as the Bayesian Information Criterion (BIC). They key problem with Score-based algorithms is that they are computationally expensive since they have to score all possible candidate graphs in order to produce their outcomes.

The most frequently used Constraint-based Causal Discovery algorithm is the Peter - Clark (PC) algorithm, proposed by Spirtes *et al.*. This algorithm was used as the basis for designing variants some which are presented below. To start with, *PC Stable* [Colombo et al., 2014] is a variant of PC which can deal with a common problem of the PC algorithm which regards its incapacity to handle order properly. In more detail, PC heavily depends on the order the variables are analyzed by it, while in *PC Stable* this issue solves it. For high-dimensional data, *PC Select* was proposed in [Bühlmann et al., 2010] which produces an undirected graph. Another Constrained-based algorithm regards the Fast Causal Inference (FCI) algorithm was introduced in [Spirtes, 2001] which can be applied in situations where the causal assumption cannot be met. Also, the FCI is known to tolerate confounding better than PC. Finally, Direct Linear non-Gaussian structural equation model (DirectLiNGAM) proposed in [Shimizu et al., 2006], [Shimizu et al., 2011] is

a causal discovery method that utilizes a Linear non-Gaussian structural equation model (LiNGAM) to infer causal relations between variables.

Score-based Causal Discovery algorithms include Greedy Equivalence Search (GES) and Greedy Interventional Equivalence Search (GIES) among others. GES [Chickering, 2002] on the one hand aims at identifying the structure of the Bayesian Network by searching through the space of possible graphs and determining which edges represent direct causal relationships between variables. GIES [Hauser and Bühlmann, 2012] which is an improvement of the GES algorithm due to its ability to use interventional data and provide improved identifiability of causal models.

5.1.3.2 Causal Neural Networks

Causal Neural Networks (CausalNN) [Wiering et al., 2002], [Rumelhart et al., 1985] regard a NN architecture, able of performing Causal Discovery. The way CausalNNs work is by changing the feedforward phase to resemble a Bayesian Network and thus being able to represent causal relationships. CausalNNs try to learn not only statistical connections but also the underlying cause-and-effect mechanisms [Chattopadhyay et al., 2019] that exist in the observed data during training. They also often have features for simulating interventions, which let the model guess how certain behaviors affect the system.

5.1.3.3 Causal Decision Trees

Causal Decision Trees (CDT) are Decision Trees capable of incorporating causal relationships into their modelling process [Li et al., 2016b] where each node of the developed graph can be causally interpreted. Furthermore, CDTs are able to provide a concise graphical depiction of the existing causal relationships.

5.2 Causal Features Generation Method Overview

This Section describes in detail the proposed Causal Features Generation method [Symvoulidis et al., 2023a]. As already described above, its objectives are the following: (*i*) to identify the causal relationships among the features in a given dataset; (*ii*) to generate causal features according to the previously identified causal relationships; and finally (*iii*) to generate a new causally enhanced dataset which includes also the new causal features.

The proposed Causal Features Generation method can be split into two main phases: (*i*) the discovery of the causal relationships found among the already existing variables (*i.e.*, the features) in the given dataset (described in Section 5.2.1), and (*ii*) the generation of the new causal features based on the identified relationships along with their inclusion in the original dataset (described in Section 5.2.2).

5.2.1 Causal Discovery

Algorithm 4 Causal Features Generation Generic method

Input:
$\mathcal{D}$: the dataset used for training of a ML or DL model.
Auxiliary Variables:
t: A target feature,
CF_t: List of feature t's parents.
Output:
$\mathcal{D}$
Algorithm:

1: perform causal discovery in order to extract causal relationships among the features of the dataset $\mathcal{D}$
2: store identified causal relationships in a data structure such as a list
3: **for each** of the identified causal relationships **do**
4: calculate $P(t|CF_t)$
5: $\mathcal{D} : \mathcal{D} \cup P(t|CF_t)$ *// Append the new causal feature to the existing dataset*
6: **end for**
7: **return** $\mathcal{D}$

The causal discovery phase starts, as described in Algorithm 4 with the identifica-

tion of the causal relationships of the attributes in a given dataset $\mathcal{D}$. The identification of such causal relationships is not always a straightforward procedure. In many real-world scenarios, hidden relationships, *a.k.a.* confounders exist. Confounders (otherwise known as *confounding variables* or *third variables*) are, in essence, features in a dataset that are associated with both the independent and the dependent variables [McNamee, 2003]. If not properly dealt with, confounders may lead to several issues, such as:

- **Bias in predictions:** If confounders are present in a dataset, they may lead to incorrect conclusions due to the inclusion of bias [McNamee, 2003], [Thomas, 2020].
- **Reduced model performance:** When confounding variables are not dealt with, there exist a potential to increase the variability of the predictions, and as a result reduce a model's precision and as a results its overall performance [causaLens, 2023].

This is one of the most crucial issues when it comes to causal analysis, and while it may not always be easy to deal with, there exist several strategies to address them, the common of which is the following (a detailed analysis on how to deal with the issue of identifying and managing confounding variables has been performed in Section 5.1):

- **Causal DAGs**: Causal diagrams can be used to visually represent the relationships between variables in a dataset, which as a result can facilitate on understanding and identifying potential causal relationships as well as confounding variables. Consequently, especially in the context of ML, additional measures can be taken into consideration to measure the impact of the confounding variables, and to adjust accordingly the pre-processing steps as well as the data which will be used for the training of a ML model. In addition, any of the following strategies can be also utilized in combination with the generation of causal DAGs.

- **Stratification:** This strategy involves dividing the data (in statistical analysis this is often referred to as the *study population*) into *strata* [Ding and Lu, 2017], [Frangakis and Rubin, 2002], [Gallop et al., 2009], which are essentially subgroups of the population that have common values of a key variable, which may have been identified as a potential confounder. In the context of ML, when splitting the dataset into training and testing sets, it should be ensured that each one of the two sets contains a similar distribution of the important (confounding) variables. This as a result, could ensure that a ML or a DL model is evaluated on data that is representative of different strata (*i.e.*, different subgroups of the identified confounding variable).

- **Propensity Score Matching:** Propensity Score Matching is a strategy, mostly used in observational studies, which aims at reducing the effect that potential confounding variables have over a study, by creating groups of individuals which have similar propensity scores [Austin, 2011], [Fu et al., 2019]. The propensity score measures the probability of an individual, participating in the study, of receiving the treatment, based on the observed covariates (*i.e.*, the features).

- **Other ML or Statistical methods**: Advanced ML algorithms, such as Causal Forests (or otherwise known as Generalized Random Forest (GRF)), [Athey and Wager, 2019], [Caron et al., 2022], [Suk and Kang, 2023], or statistical methods such as Targeted Maximum Likelihood Estimation (TMLE) [Luque-Fernandez et al., 2018], [Pang et al., 2016], [Schuler and Rose, 2017], can be used for causal inference, since they are capable for measuring the importance of the features in a dataset, while they can also be used in measuring metrics such as Conditional Average Treatment Effect (CATE) and Individual Treatment Effect (ITE).

In the proposed method, the identification of such causal relations is performed using either the FCI algorithm, as proposed by Spirtes *et al.* [Spirtes, 2001] or the

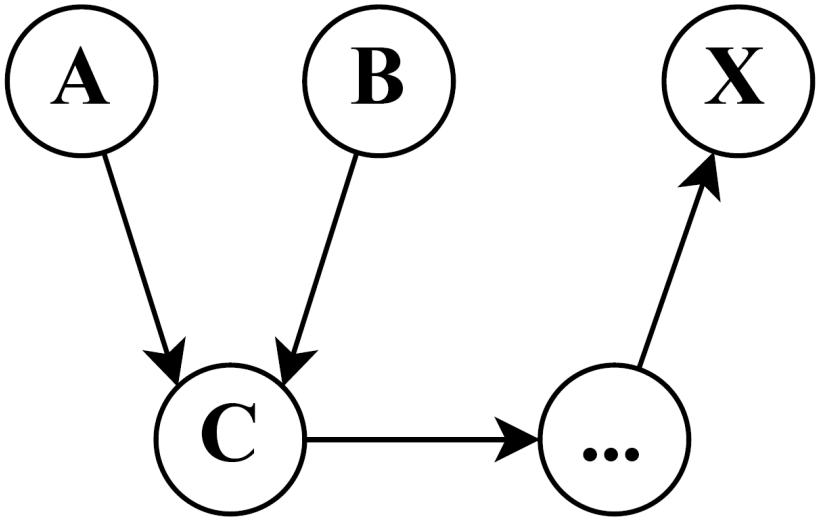


Figure 5.4: A CPDAG sample generated by the FCI algorithm. In this example graph, attribute C is caused by two features, namely attributes A, and B.

DirectLiNGAM algorithm, proposed by Shimizu *et al.* [Shimizu et al., 2011]. If the selected causal discovery algorithm is FCI, the outcome regards a Completed Partially Directed Acyclic Graph (CPDAG) $G = (V, E)$, where V regards the features of the dataset, and E the directed edges correspond to the causal relationship between one feature and another, such that $A \rightarrow B$ indicates that A causes B. As an example, in Figure 5.4, attributes A and B are both causes of attribute C. Another option would be the PC algorithm proposed by Spirtes *et al.* [Spirtes et al., 2000] [Spirtes and Glymour, 1991], yet it was proven that the FCI algorithm manages to perform better than the PC algorithm, especially when it comes to tolerating with the confounders [Spirtes, 2001]. To this end, Algorithm 4 can then be adapted as described in Algorithm 5.

On the other hand, if the selected causal discovery method is the DirectLiNGAM algorithm, the outcome is an adjacency matrix where the columns and the rows are the instances of the dataset $\mathcal{D}$. The cells in this matrix represent the causal effect a feature (column) has on a feature (row), as shown in Equation 5.1. This can be then translated into a Causal DAG, as depicted in Figure 5.5. In this case,the

outcome regards a DAG $G = (V, E)$ as the one described above, only with the difference that now the causal effect that a feature A has on feature $B$is shown on the edge. In the DirectLiNGAM case, $A \xrightarrow{x} B$ suggests that there exists a causal relationship between A and B, and x represents the effect (or strength) of the causal relationship. The strength for each edge is calculated using conventional covariance-based methods such as least squares regression.

$$\begin{bmatrix} 0 & 0 & 0 & 2.97242198 & 0 & 0 \\ 3.01647872 & 0 & 1.98821239 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5.9886615 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 7.97701605 & 0 & -1.00053324 & 0 & 0 & 0 \\ 3.98514644 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

Once the above-mentioned causal discovery step is performed, the outcome is the DAG or the CPDAG which includes the features of the dataset as nodes, and as already mentioned, the edges represent the causal relations among the features. It may also include the causal effect of each causal relationship, in case DirectLiNGAM is utilized.

The generated CPDAG is then provided as input to the next phase, as it is described in Section 5.2.2, whose purpose regards the generation of the new causal features, and finally the generation of the causally enhanced dataset.

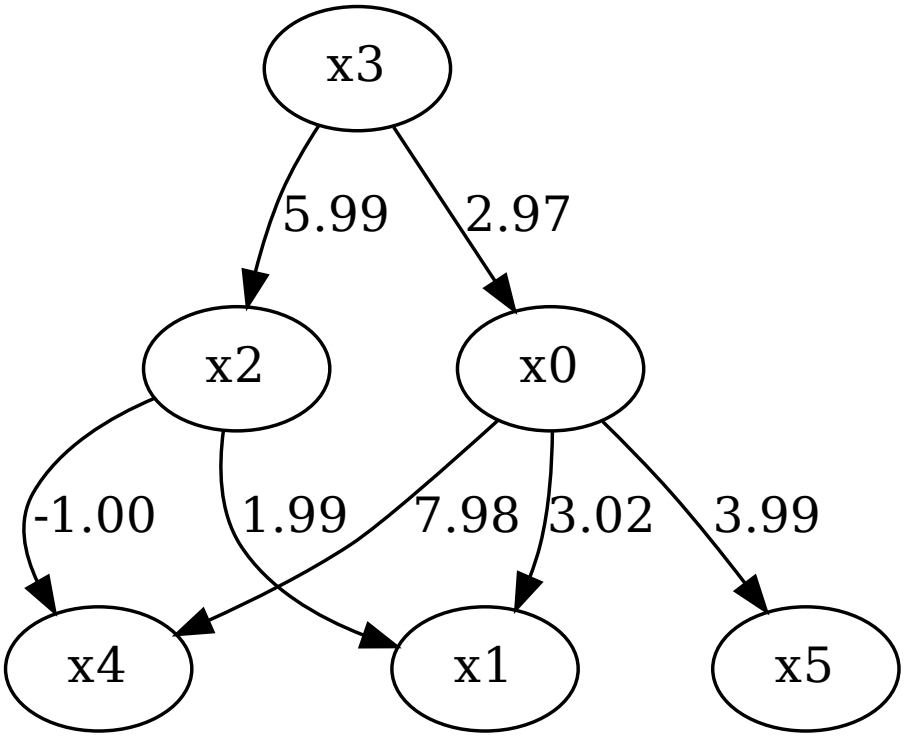


Figure 5.5: A Causal DAG sample generated by the DirectLiNGAM algorithm.

5.2.2 Generation of Causal Features based on Causal Discovery Outcomes

This phase of the proposed method is instantiated once the causal CPDAG is generated by the FCI or the DirectLiNGAM algorithms. As also described in the generic Algorithm 4, the features in which to control for their causation variables are performed, using the Parents - Children approach (p_c as it is defined in Algorithms 5 for the FCI adaptation and 6 for the DirectLiNGAM adaptation). This means that for each feature, only those features that are directly connected to it will be selected, as also shown in Figure 5.6. Additional approaches also exist with re-

Algorithm 5 Adapted Causal Features Generation method for the FCI case

Input:
$\mathcal{D}$: the dataset.
Auxiliary Variables:
fci: Output of the **fci** algorithm
pc: Output of the **p_c** algorithm,
t: A target feature,
CF_t: List of feature t's parents.
Output:
$\mathcal{D}$
Algorithm:

1: $fci \leftarrow$ **fci**($\mathcal{D}$)
2: $pc \leftarrow$ **p_c**(fci)
3: **for each** tuple in pc **do**
4: calculate $P(t|CF_t)$
5: $\mathcal{D} : \mathcal{D} \cup P(t|CF_t)$ // *Append the new causal feature to the existing dataset*
6: **end for**
7: **return** $\mathcal{D}$

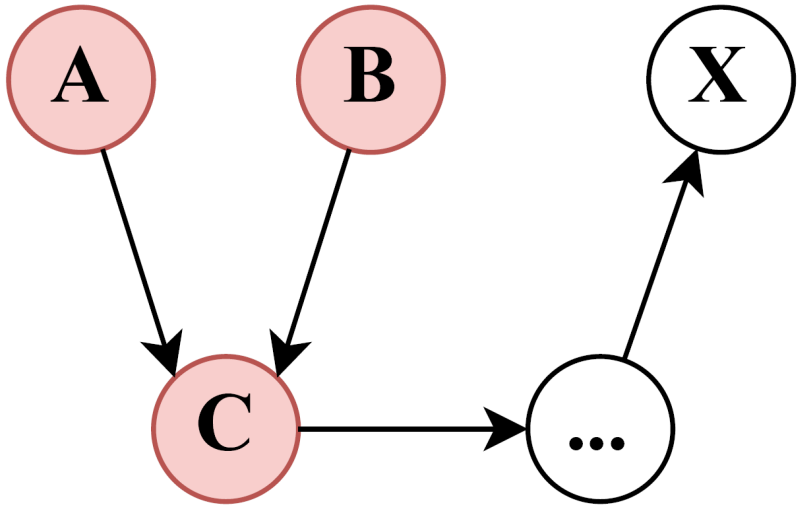


Figure 5.6: Sample of selecting causes of features using the Parents - Children approach. In this example graph, attribute C is caused by two features, namely attributes A, and B, hence both features are selected.

gards to the appropriate method of features selection, such as the Markov Blanket approach [Yu et al., 2016], [Ling et al., 2019], [Fu and Desmarais, 2010], in which the children (if any) of the target variable are also selected, along with its parent variables, however, in the current case, only the parents are causes of the target

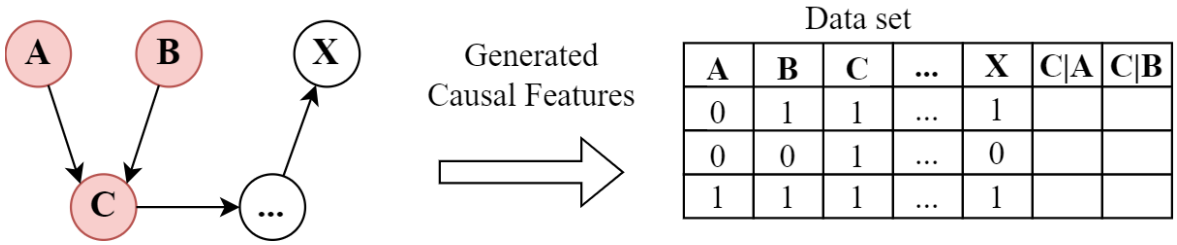
Algorithm 6 Adapted Causal Features Generation method for the DirectLiNGAM case

Input:
$\mathcal{D}$: the dataset.
Auxiliary Variables:
fci: Output of the **DirectLiNGAM** algorithm
pc: Output of the **p_c** algorithm,
t: A target feature,
CF_t: List of feature t's parents,
e_i: The causal effect of CF_t on t.
Output:
$\mathcal{D}$
Algorithm:

1: $dLiNGAM \leftarrow$ **DirectLiNGAM**($\mathcal{D}$)
2: $pc \leftarrow$ **p_c**($dLiNGAM$)
3: **for each** tuple in pc **do**
4: calculate occurrences of an instance appears in the dataset for $P(t|CF_t)$
5: **end for**
6: **for each** tuple i in pc **do**
7: $max((P(t|CF_t)) \leftarrow$ calculate max occurrences in the dataset for $P(t|CF_t)$ for any tuple in pc
8: $min(P(t|CF_t)) \leftarrow$calculate min occurrences in the dataset for $P(t|CF_t)$ for any tuple in pc
9: **end for**
10: **for each** tuple i in pc **do**
11: $causalFeature_i \leftarrow \frac{P(t|CF_t)-min(P(t|CF_t))}{max(P(t|CF_t))-min(P(t|CF_t))} \times z_i$
12: $\mathcal{D} : \mathcal{D} \cup causalFeature_i$ *// Append the new causal feature to the existing dataset*
13: **end for**
14: **return** $\mathcal{D}$

variable.

Afterwards, the generation of the causal features process is instantiated. In case the FCI is used for causal discovery, the newly generated features are in essence probabilities that measure the influence of the parent attributes have, on the value of the target attribute, and are calculated by estimating the relevant posterior probabilities as suggested by Nogueira *et al.* [Nogueira et al., 2020], as also shown in Equation 5.2, below:



A	B	C	...	X	C|A	C|B
0	1	1	...	1		
0	0	1	...	0		
1	1	1	...	1		

Figure 5.7: The new causal features are generated.

$$P(ControlF = t|CausalFs = CF_t) = \frac{\sum t \cap CF_t}{\sum CF_t} \tag{5.2}$$

Let t be the target feature and CF_t a list of all features causing t (i.e., feature t's parents) as extracted by the FCI algorithm and selected using the Parents - Children approach, such that $CF_t = [\forall p \in CF_t : p \rightarrow t]$. The calculated probabilities are in fact the newly generated causal features of the dataset which are then consolidated into the original dataset, as also depicted in Figure 5.7. A visual representation of the overall workflow is presented in Figure 5.8, starting from the creation of the causal graph, to the creation of the causal features and finally the generation of the end dataset which contains the generated causal features.

In the case where the DirectLiNGAM algorithm is used for causal discovery, the generated causal features measure the effect of the attribute CF_t on t, using the same posterior probability referred to above. The end formula to calculate the Min-Max normalized new causal feature is found in the Equation 5.3 below:

$$causalFeature_i = \frac{P(t_i|CF_{t_i}) - min(P(t|CF_t))}{max(P(t|CF_t)) - min(P(t|CF_t))} \times z_i \tag{5.3}$$

where $min(P(t|CF_t))$ and $max(P(t|CF_t))$ are the minimum and maximum occurrences where t_i has a given value given the value of CF_t, and z_i the z-score of the effect e_i accordingly.

The standard score (*i.e.*, otherwise known as Z-score) z_i for a given effect e_i. It is

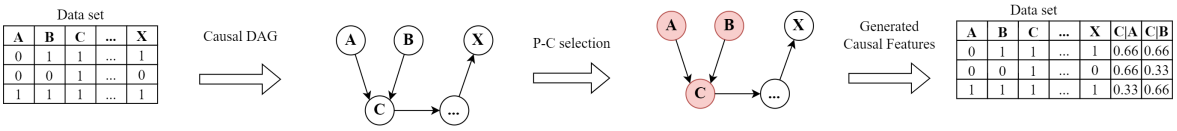


Figure 5.8: The overall causal features generation workflow starts with the creation of the causal graph, the selection of the appropriate features using the Parents - Children approach, to the generation of the new causal features.

used to standardize the effects since they may vary from very small numbers close to zero, to large or even negatives. The standard score is calculated as shown in Equation 5.4 below:

$$z_i = \frac{e_i - \mu}{\sigma} \tag{5.4}$$

where μ is the mean of all effects and σ the standard deviation of all effects.

5.3 Evaluation of the Causal Features Generation method

The Causal Features Generation method has been applied to two use cases to demonstrate its capabilities. The baseline algorithm as presented in the Chapter has been adapted accordingly. The first use case in which the proposed method was utilized in, can be found in Chapter 6 where it was utilized as a pre-processing step to the training lifecycle of a DL network responsible for classifying users of services deployed in hybrid Edge / Cloud computing infrastructures to mobility classes. The outcomes of this model was then utilized for the optimization of the data placement process.

In the second use case, the proposed data enhancement method is utilized in conjunction with the Influence-based Dataset Generation method, which can be found in Chapter 7. Similar to the first use case, both methods are part of the pre-

processing steps in the training of ML model used for the prediction of resource usage various metrics such as CPU and memory towards the dynamic resource allocation at the edge.

Chapter 6
Cloud and Edge Computing Optimization: User Mobility-based Data Placement in Hybrid Cloud / Edge Environments using Causally-aware DL

Chapter Structure

This Chapter is constructed as follows:

- **Section 6.1 - Background on Data Placement and Optimal Path identification solutions**, performs a literature review in the areas of Data Placement and Optimal Routing.
- **Section 6.2 - Mobility-based Data Placement Strategy Overview**, presents a detailed overview of the proposed User Mobility-based Data Placement Strategy and and highlights the adaptations that were made to the Causal Features Generation method in order to utilize it in the User Mobility-based Data Placement strategy.
- **Section 6.3 - Evaluation Results**, evaluates the proposed Data Placement

strategy and presents the key outcomes of this evaluation.

- **Section 6.4- Discussion on the Evaluation Outcomes and the Effectiveness of the Causal Features Generation method**, discusses the evaluation outcomes of the given scenario and highlights the advantages the proposed method brought.

This Chapter presents the results of the evaluation that the Causal Features Generation method has been put under. The proposed method is as described in Chapter 5 and has been effectively applied to a use case [Symvoulidis et al., 2023a] to demonstrate its ability. The generic algorithm is adapted accordingly to fit the use case, as it will be described in Section 6.2.

6.1 Background on Data Placement and Optimal Path identification solutions

Before proceeding with the overview of the proposed Mobility-based Data Placement Strategy, it is important to highlight the work in the fields of Data Placement and Optimal Path identification, in order to emphasize on the significance of the proposed solution.

	Key aspects taken into consideration in data placement					Additional features		
Research work	Latency	Bandwidth	Storage Capacity	Redundancy	Migration cost	Routing	Data Importance	User Mobility
Wei et al. [Wei and Wang, 2021]	✓	✓	partial	partial	✗	✓	✓	✗
Xie et al. [Xie et al., 2019a]	✓	✓	✗	✓	✗	✓	✗	✗
Du et al. [Du et al., 2020]	✗	✓	✓	✗	✗	✗	✗	✗
Shao et al. [Shao et al., 2019]	✓	✓	✓	✗	✗	✗	✗	✗
Proposed Solution	✓	✓	✓	partial	partial / indirect	✓	✗	✓

Table 6.1: Criteria-based comparison of data placement strategies on the edge.

6.1.1 Data Placement

The development of mobile-related applications has accelerated significantly in recent years which led to an increasing number of mobile devices, which are capable of generating and analyzing huge amounts of data, are developing. A typical

example is IoT devices which are regarded as one of the most significant sources of data production. In fact, according to published surveys, IoT devices will be exclusively responsible for producing over 75 zettabytes of data by 2025. In contrast, the data volume generated by such devices was a mere 13.6 zettabytes in 2019 [O'Dea, 2020]. Thus, it is reasonable to expect that the volume of information will continue to grow exponentially. Consequently, both the transmission and analysis of this data will become more complex, requiring substantial computational power and potentially causing delays when managed within conventional cloud infrastructures.

The term *Data Placement*, is associated with activities related to storing, replicating, locating and retrieving data to or from a system such as a distributed database. In general, it is a topic broadly discussed in several areas such as Cloud Computing [Agarwal et al., 2010], [Yuan et al., 2010], distributed databases and systems [Mehta and DeWitt, 1997], and Peer-to-Peer (P2P) networking [Ye et al., 2011], [Chervenak et al., 2007], among others. However, lately, there has been an increasing interest in the field of Edge Computing as well, where multiple studies are being conducted to examine this topic from different points of view.

Furthermore, despite the fact that modern ML and DL technologies have made tremendous progress recently in an attempt to address issues associated with Cloud and Edge Computing, such as orchestration [Wu, 2020], [Leng et al., 2022], runtime adaptation [Eom et al., 2013], [Sniezynski et al., 2019], etc., they are neglected in the data placement domain, since the majority of Data Placement strategies only provide specific algorithms for specified use cases and scenarios that are nearly impossible to adapt to in an exhaustive way.

To begin with, the authors of [Wei and Wang, 2021], have proposed a virtual-space method to consider the popularity of the data when storing them at the edge. In more detail, the authors proposed a methodology which involved placing the most frequently accessed data in close proximity to the network center. This placement

is intended to decrease the overall transmission latency by shortening the path needed for data retrieval. Additionally, they introduced data offloading and replication strategies, which take into consideration the data popularity. As a result, these strategies decrease the workload on the servers that are being used the most, and eventually improve the edge network's overall performance. The popularity metric defined by the authors and it measures how much a data is requested by the users in the system. The greater the value of this metric, the greater the popularity of the data; consequently, the data is stored in closer proximity to the network center.

Xie *et al.* in [Xie et al., 2019b], [Xie et al., 2021] proposed COordinate-based INdexing (COIN), a data indexing mechanism used for data sharing on the edge, in which the control plane of the edge network stores the the data indices and coordinates of the edge switches in a two-dimensional virtual space. The proposed mechanism then stores the data index to an indexing edge server, which is connected to a switch that is the closest to the coordinates of the data index. In addition, Xie *et al.* in [Xie et al., 2019a] extended their work and proposed Greedy Routing for Edge Data (GRED), a data placement and retrieval protocolwhich specifies the way data is stored and retrieved to / from the edge servers as follows. First, the position of the network switches is identified in a two-dimensional virtual space as in COIN. The an optimization of the positions is performs utilizingthe Voronoi Tessellation (VT) methodology, while a Delaunnay Triangulation (DT) multi-hop graph is constructed. Once the final graph is constructed, the data placement process is instantiated. The data is first sent to the switch that is the closest to the data and then forwarded by that switch to the closest edge server found in the two-dimensional space.

The authors of [Shao et al., 2019] proposed a data replica placement strategy for IoT workflows in hybrid Edge and Cloud environments. The data coming from IoT devices is deployed in edge servers in close proximity assuming that the data ac-

cess cost is minimum. If the data access cost exceeds a certain limit, the data is placed to another adjacent edge server or a remote data center in order to minimize the cost.

Du *et al.* in [Du et al., 2020] proposed a data placement and sharing strategy for multi-region heterogeneous Cloud / Edge environments . In more detail, the authors proposed a Discrete Particle Swarm Optimization (DPSO) with Differential Evolution (DE) data placement strategy, called Differential Evolution - Discrete Particle Swarm Optimization - Data Placement Strategy (DE-DPSO-DPS), in order to design a data placement methodology which takes into account the data transfer cost among multiple clouds, the storage capacity of an edge micro-data center, and the most significant elements that result in data transmission delays.

Lin *et al.* in [Lin et al., 2019] on the other hand, proposed Discrete Particle Swarm Optimization algorithm with Genetic Algorithm operators (GA-DPSO) which aimed at optimizing the data transmission times when performing data placement processed for a scientific workflow on the Edge and Cloud. The authors considered that the data placement problem regards a multi-parametric issue in which the data center where the data should be stored had to be considered, in combination to the actual data, a *Map*, which represents the data relocation from one data center to another, and the total transmission time during data placement.

In order to tackle these challenges, the concept of Edge Computing has been established, which leverages the computational capabilities present at the edge [Varghese et al., 2016]. Therefore, it is confident to argue that Edge Computing is emerging as a vital instrument for mitigating the costs associated with data transmission, and enhance the quality of services offered by analyzing data in smaller volumes and in closer proximity to users. However, Data Placement still remains a challenge that prevents Edge Computing from being utilized at its full potential. The latter highlights the significance of designing solutions that optimize the Data Placement process, with attention to resource management and the overall qual-

ity of service. Nevertheless, a critical concern emerges: the existing methods and techniques proposed in the literature primarily focus on data placement and optimization, with little consideration to the user and their needs, while exclusively considering the infrastructure and network facilities. As a result, these methods are policy-centric in nature, which produces solutions that lack dynamicity and fail to adapt to changing environments.

6.1.2 Optimal Routing

One of the most extensively studied subjects in computer science, the shortest path identification problem concerns discovering the shortest route from a given node in a graph to every other node. This is of significant importance within the area of Edge Computing, considering that the edge computing nodes are linked together via a network and form an edge network graph. To address this issue, several articles can be found in the literature and are described in detail below.

To start with, the authors of [Desai et al., 2022] presented an optimal IIoT, Edge-based Optimal Routing framework in order to reduce the communication latency, in which, three routing algorithms were designed. In the first algorithm, K-Means-based Optimal Path Algorithm (KOPA), the location of the edge nodes and the specified number of clusters are provided as input, and applies the K-Means algorithm in order to group the nodes according to their Euclidean distance. A Cluster Head (CH) node is selected, which is a essentially an edge server that has the capacity to directly communicate with every node in the cluster. The CH node is also directly connected to the other CHs of the other clusters. Using Dijkstra's shortest path algorithm, the distances between the various CHs are sorted and as a result, the distance between any two servers within the edge network graph is the distance between the first edge server and its CH, along with the distance between the two CHs and the distance between the second CH and the desired edge server. The second algorithm, named Cluster-based Optimal Path Algorithm (COPA),groups

the servers into clusters according to their physical coordinates. For each cluster an edge server is assigned as a CH which is in that case, the edge node closest to the centroid of the cluster. The communication following to clustering formation matches that of the KOPA algorithm. The third and final algorithm proposed by Desai *et al.* is the Minimum Interval-based Optimal Path Algorithm (MIOPA), which is in fact an extension of the KOPA and COPA algorithms. The MIOPA algorithm's objective is to achieve a balance between the amount of the clusters so as to select the CHs in an efficient manner, by arranging the clusters with respect to their maximum length of the available geographical area.

Zhang *et al.* in [Zhang et al., 2020] proposed an Ad-hoc On-demand Multi-path Distance Vector (AOMDV)-based routing protocol for mobile edge computing, called LLECP-AOMDV. The proposed protocol, utilizes the Link Lifetime and Energy Consumption Prediction (LLECP) and consists of two main phases. In the first phase, all possible paths between any two edge computing nodes are found, while in the second phase, the optimal path out of the available ones is selected. For the final selection, additional constraints are inserted, including the lifetime of the link and the energy consumption required for the data transfer.

In addition, the amount of research dedicated to the implementation of Interference-Aware Routing (IAR) protocols has risen considerably in recent years such as the researches found in [Waharte et al., 2008], [Mahmood and Cornaniciu, 2005], [Siraj and Abbasi, 2022], [He et al., 2019]. A case in point is the paper authored by Waqas *et al.* [Waqas et al., 2022]. The authors where proposed the Interference-Aware Cooperative Routing (IACR) algorithm. This algorithm determines the routing cost by utilizing a function that incorporates both the produced and received *interference* of a node within the edge network. The interference metric is referred to as a way of measuring the efficiency of an edge computing network against its energy consumption.

Following, the proposed User Mobility-based Data Placement strategy will be pro-

posed, which attempts to improve data placement and optimal routing for Edge Computing infrastructures in multiple ways. First of all, there exist none user mobility-based data placement (and data retrieval) solutions for such environments in the literature. In addition, the majority of these solutions are *rule-based*, which may function adequately in a controlled environment but could affect the robustness of the final system. In an effort to progress in that direction, a Causal-aware DL model is implemented which exploits the proposed Causal Features Generation method, as will be explained over following Sections. Additionally, the migration cost towards the selection of the most suitable servers is taken into account, which is a factor, heavily valued in service placement issues [Mao et al., 2017].

As far as the contributions of the proposed data placement strategy in the optimal routing field are concerned, we propose utilizing ML algorithms (K-Means) in order to generate clusters within a given edge network graph. The utilization of such methods is performed as a preliminary step to divide the edge network graph into several smaller sub-graphs and find optimal paths in each among the sub-graphs. This results to the reduction of the size of graph, since the route optimization (for the majority of the use cases) is only performed within a given cluster and therefore simplifies the problem considerably.

Table 6.1 presents a detailed criteria-based comparison of the proposed data placement strategy against the literature. The baseline criteria, which include aspects such as the storage capacity of the nodes, the overall latency of the network, and data redundancy, there exist specific features in the proposed strategy which are not considered before. Such factors involve (*i*) the evaluation of the migration costs that occur when transferring data from one edge node to another, (*ii*) the process of determining the optimal path connecting the user's present edge node and the edge node the contains the user's data, and finally (*iii*) the use of the *User Mobility* metric which has only been used for service placement strategies [Ouyang et al., 2018].

6.2 Mobility-based Data Placement Strategy Overview

Consider an Edge Computing infrastructure like the one depicted in Figure 6.1. Consider a typical edge computing environment, which can be formulated in the form of a graph $G(V, E)$, where G is the graph of the edge network, vertices $V = \{v_1, v_2, \ldots, v_N\}$ represent the edge servers of the network, and $E = \{e_1, e_2, \ldots, e_M\}$ represent the edges that connect the servers of the network.

The proposed data placement strategy aims at identifying the optimal server v_j (author's note: the edge server may also be referred to as *node* in the following Sections) according to the user's mobility class. In addition, the data placement strategy also defines the optimal path to upload the data from the current node the user is connected to (i.e., v_i), to the node v_j, where the data should be stored.

The information related to the optimal edge server v_j will be stored in the centralized cloud infrastructure (see Figure 6.1). Thus, a data retrieval request is made, the edge node v_i (*i.e.*, the node the user is currently connected to) must contact the cloud infrastructure, in order to retrieve information about which edge node v_j holds the user's data. Next, the optimal path between v_i and v_j needs to be created. The data needs to be first migrated to v_i. Once the data migration is done, the user can download it.

For each edge node $v_i \in V$, their maximum capacity is defined as $c_i = c(v_i)$, while $l_{ij} = l(v_i, v_j)$ defines the calculated least distance between two edge nodes v_i and the v_j. This way, all the least distances between every node set is found in the $L = \{l_{ij}\}$ distance matrix. It is important to note that, in this context, the term distance refers to the physical distance (*i.e.*, the number of edges that connect the two servers). Furthermore, the transmission cost w_{ij} between edge servers i and j is defined as the sum of all transmission delays (*i.e.*, latencies) t_{ij} for each l_{ij}. Similar to the distance matrix, matrix $W = \{w_{ij}\}$ contains all transmission costs between any two edge servers i and j and is defined as shown in Equation 6.1 below:

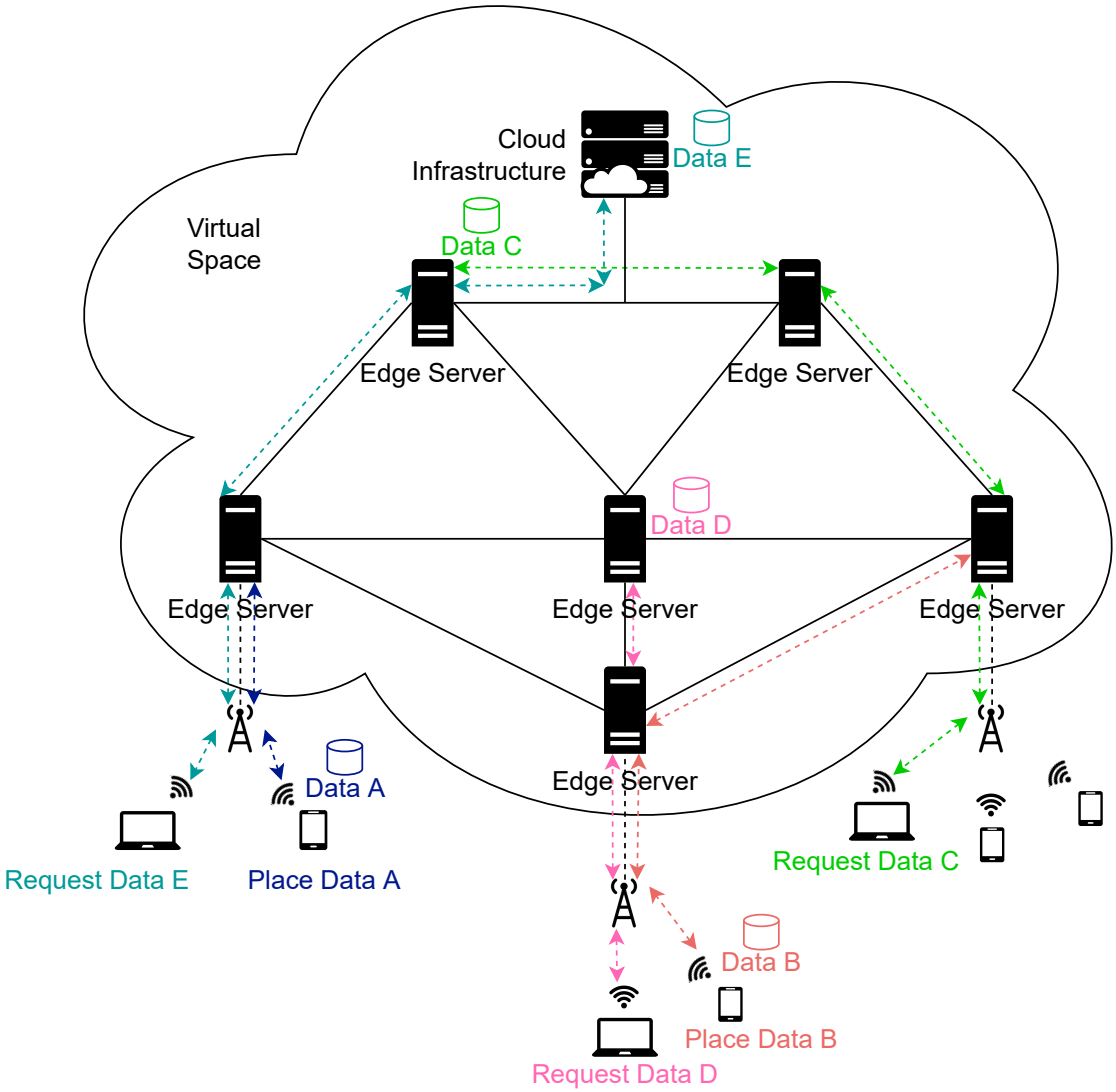


Figure 6.1: An edge computing network, including the nodes, a cloud infrastructure, and the edges connecting the servers and the cloud infrastructure, along with its users, who may perform requests to either place or request data to or from the network.

$$w_{ij} = \sum_{i,k}^{j} l_{ik} \times t_{ik} \tag{6.1}$$

The transmission delay, denoted as t_{ij} as presented in Equation 6.1 is not static

metric, yet it dynamically updated, such that it may be increased or decreased depending onthe data migrations performed over a given edge server at a certain time period. The transmission delay t_{ij} between two edge nodes i and j is defined as shown in Equation 6.2:

$$t_{ij} = \delta + \sum_{k=1}^{n} \frac{d_k}{b_{ij}} \tag{6.2}$$

δ refers to the default delay of the edge server, d_k is the size of a data to be transmitted, and b_{ij} the available bandwidth of the edge that connected the two edge nodes i and j. Hence, the transmission delay t_{ij} at a certain time period is calculated by taking into consideration the load that each data point d incorporates into the overall network when it is migrated.

Additionally, a number of user-related information are also defined. Starting with, k which represents the number of users connected to the edge network, such that $U = \{u_1, u_2, \ldots, u_k\}$ is the list of all users. $M = \{m_1, m_2, \ldots, m_k\}$ refers to the mobility of each user, where the mobility of each user is defined as $m_i = m(u_i)$. Let $T = \left[k \times q\right]$ be a matrix consisting of the edge nodes a user has visited within a sliding time window of q days (*e.g.*, for the last 30 days). Each row of T represents a user's number of visited nodes. $A = \left[k \times q\right]$ on the other hand, refers to a matrix that consists of the actions the user's made within the same sliding time window of q days, where each row of matrix A represents an individual user's activity a_i. Both matrices T and A will be used for the calculation of the users' mobility.

The high-level user mobility-based data placement strategy, starts with the creation of proximity-based clusters in the edge network, as they will be defined further in Section 6.2.1. Consequently, the user's information are exploited in order to identify their mobility patterns and classify them into mobility classes. The users may be either classified as *static, local,* or *mobile* over a predefined period of time. Thus, a user that could have been initially classified as static, as the time

progresses their mobility class might change to local, or mobile.

In any case, specific data placement algorithms have been designed for each mobility class, and the appropriate one is utilized according to the users' current class. These data placement algorithms are utilized to predict the user's optimal nodes for data placement.

Finally, as time affects the mobility class of the users, the above procedure is performed repetitively. This may lead to data migrations over time since the user might change mobility class, or another node that may be more optimal for this user may exist.

6.2.1 Proximity-based Clustering of the Edge network

The original edge network topology G is split into several clusters, utilizing the proximity-based clustering methodology, in which the K-Means algorithm is used in order to cluster the edge nodes into regions within the edge network, similar to the clusters shown in Figure 6.2.

The clustering of the edge network is based on the distances of the nodes and will be utilized during the data placement phase. In more detail, this allows the proposed data placement strategy to have greater freedom on how the data should be placed on the various edge servers, since adjacent servers can be easily identified, and as a result make the optimal node selection easier and faster.

6.2.2 User Mobility Definition

User mobility measures at what rate a user is travelling through the nodes of an edge network. It refers to a metric that has been broadly used in the cloud computing and edge computing systems. Mostly though, it is used in order to optimize service selection or placement process for the users [Wu et al., 2019], [Ouyang

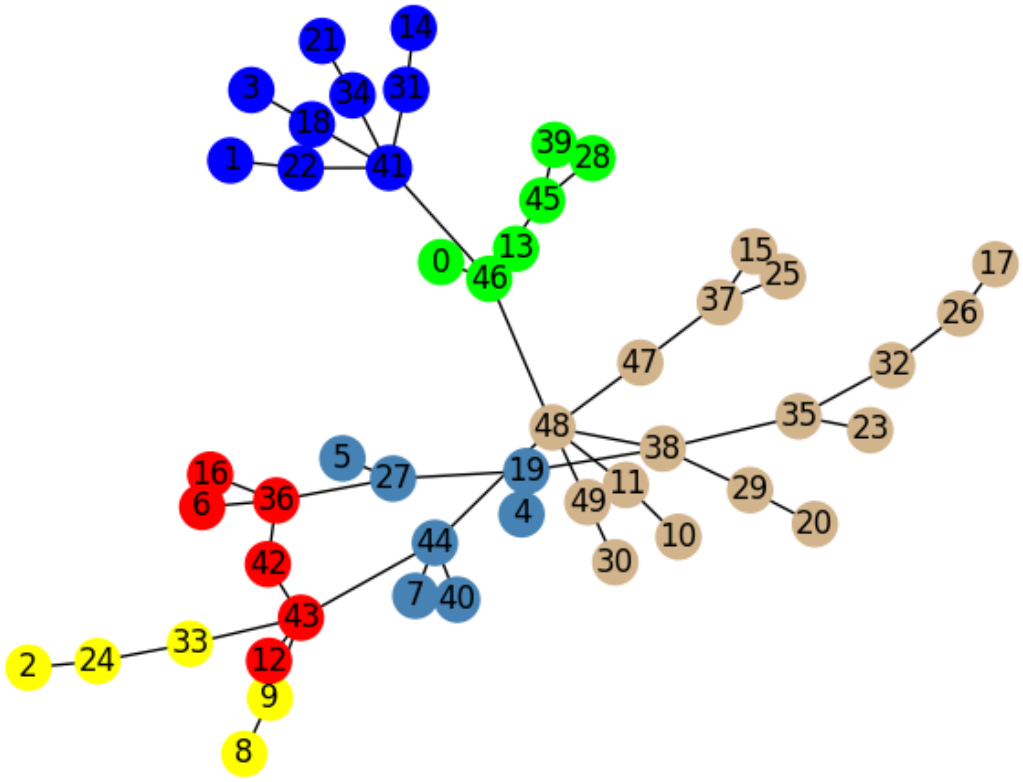


Figure 6.2: Proximity-based clustering of edge networks utilizing the K-Means algorithm.The graph depicts an edge network where the nodes (whose IDs are represented as numbers), are interconnected via a network.

et al., 2018], to assign tasks on the edge [Wang et al., 2018], to improve context-aware applications [Do et al., 2015], or to enhance and improve User Centric Networks (UCN) [Li et al., 2016a]. In this context, we utilize the user mobility in order to optimize the data placement process.

And even though it has been previously utilized in service placement strategies [Wei et al., 2020], to the best of our knowledge, it has never be exploited towards the provision a *mobile-aware data placement strategy*. Hence, it is essential to highlight how important it is, and the reasons why there is a need to perform data placement in such diverse environments [Tsoumas et al., 2021] by always taking

into consideration the user's mobility.

There are three primary reasons for designing data placement strategies that consider user mobility as a key factor in the decisions they make. To begin with, all user data will be stored on one or more edge nodes situated in close proximity to the user. As a result, the data transmission times will be considerably reduced, due to the data being in closer proximity to its owner (*i.e.*, the user that this data belongs to). Furthermore, due to the fact that the data will be distributed across a limited number of edge nodes (the data might even be stored in just one edge node), the amount of time of the data fusion procedure will be reduced as well, something that will also affect the cost [Khan et al., 2019] as well as the greenness level of the overall procedure [Chowdhury et al., 2013].

6.2.3 User Mobility Prediction

User i's mobility is defined as m_i, over a given time period of q days and is represented by an integer, as it will described further below. The higher the mobility metric m_i is, the higher the mobility of a user. In order to predict the mobility of the user, a DL network which utilizes the proposed Causal Feature Generation method as it is described in Chapter 5 is utilized. According to the mobility metric m_i, the user is then classified in either of three mobility categories:

- **The *mobile* user class:** This mobility class refers to users that are not using only one edge node, or edge nodes coming from only one proximity cluster (as the ones described in Section 6.2.1. Mobile users may are travelling along the entire edge network connecting to different edge servers every time over the given time period, and their movement pattern may be sometimes difficult to predict.

- **The *local* user class:** The local users are those that over the given time period of q days, mostly visited edge servers of a specific proximity cluster. The lo-

cal users may be linked to one or two clusters, but not more, otherwise they would be considered mobile users. The local users regard a typical case of users such as people whose home is located in a different geographic location from their office.

- **The *static* user class:** The static user category refers to those users that generally communicate with the edge network only from one specific server node. This type of user does not travel frequently even within the borders of the cluster they are linked to.

Despite the fact that these three classes may not encompass all user scenarios, users may exist who do not strictly fall into these classes. However, they comprise a wide variety of user behaviors and mobility patterns, making them suitable for a variety of edge computing applications. In addition, the results which will be presented in detail in the following Sections prove that to a great extent,the proposed data placement strategy manages to provide adequate results.

The user mobility prediction phase, utilizes the proposed Causal Features Generation method method. The adapted version of the proposed method, similar to the generic approach is divided in two steps for the causal features generation part. In this case, there exist an additional step which includes the training of the DL model which is utilized for the mobility class predictions of the user. The first phase regards the discovery of the potential causal relationships between the attributes of the data, exploiting the FCI algorithm proposed by Spirtes *et al.* [Spirtes, 2001]. During the second phase, the causal features are generated (as explained in Section 5.2.2, which are then incorporated to the initial dataset. Finally, upon completion of the aforementioned steps, the training of the dedicated DL model used for the user mobility prediction using the new causal dataset is performed.

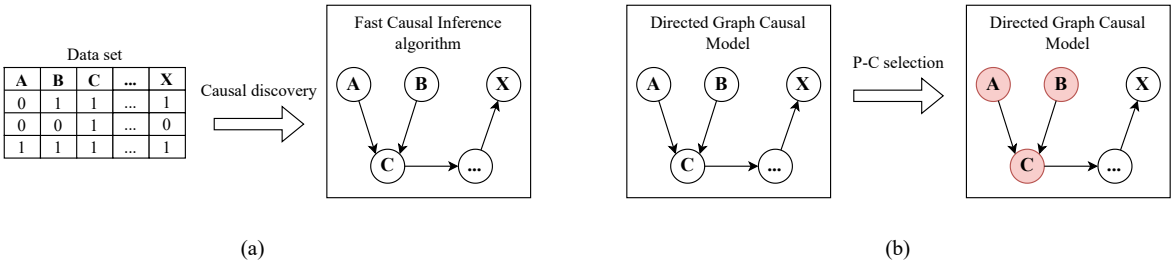


Figure 6.3: $A, B, C, \ldots, X$ are the features of a dataset. (a) Using the FCI algorithm, the causal relationships among these features are discovered and the CPDAG is generated. (b) Based on the generated CPDAG, using the Parents - Children selection approach, for each feature its parents (*i.e.*, those features that are its direct causes) are selected (if any). Consequently the new causal features are generated as described in Section 5.2.2.

6.2.3.1 Causal Discovery

Similar to the generic Causal Features Generation method, in the current case a dataset $\mathcal{D}$, used for the training of DL model which is responsible for predicting the mobility of a user, exists. Dataset $\mathcal{D}$ contains several information related to the user's interactions with the edge network, such as the number of these interactions over a time period of q days. In addition, for each interaction, the type of data requests are kept (*i.e.*, whether it was a place or a retrieve data request), as well as the size of the exchanged data, which edge server did the user interact with, and the mobility of the user. Further than that, the dataset also includes information for each edge node, and more specifically, the distances between the nodes which is useful in order to also take into consideration the movement of the user (*e.g.*, if the user visited only edge servers that are adjacent they should not be considered mobile but instead they could be considered either local or static).

The process starts with the identification of the causal relationships among the features of the dataset (as depicted also in Figure 6.3(a)). The causal relationships are identified using the FCI algorithm, since it is capable of estimating a CPDAG comprised of the features of the dataset directly connected with the features that

are causing them.

Consequently, the features along with the ones that there is a causal relationship with, are selected, based on the Parents - Children approach, as shown in Figure 6.3(b), as input to generate the causal probabilistic variables. The Parents - Children approach selects a tuple of the target feature along with its parents, i.e., the features directly connected to it, thus have a causal relationship with the child. These tuples, as also presented in Algorithm 7, are provided as input to the next phase which regards the inference of probabilities. Again, even though there exist other approaches, such as the Markov blanket approach which not only selects the parents of a target variable but also its children (if any), the use of the Parents - Children approach is favorable, since only the parents are causing the target variable.

Algorithm 7 Causal Features Generation and DL network training

Input:
$\mathcal{D}$: the dataset including all information about the users' interactions.
Auxiliary Variables:
fci: Output of the **fci** algorithm,
pc: Output of the **p_c** algorithm,
t: A target feature,
CF_t: List of feature t's parents.
Output:
causal_aware_model: The trained causal-aware Deep Learning network capable of predicting a user's mobility.
Algorithm:
1: $fci \leftarrow$ **fci**($\mathcal{D}$)
2: $pc \leftarrow$ **p_c**(fci)
3: **for each** tuple in pc **do**
4: calculate $P(t|CF_t)$
5: $\mathcal{D} : \mathcal{D} \cup P(t|CF_t)$ *// Append the new causal feature to the existing dataset*
6: **end for**
7: *causal_aware_model*.`train`($\mathcal{D}$)
8: **return** *causal_aware_model*

6.2.3.2 Ascertain Probabilities and Causal Features Generation

Next, the causal feature generation step is initiated. As mentioned in Section 5.2.2, these causal features are in fact probabilities that measure the influence of the parent attributes to the value of a target attribute and are calculated by inferring the relevant posterior probabilities, as shown in Equation 6.3:

$$P(TargetF = t|CausalFs = CF_t) = \frac{\sum t \cap CF_t}{\sum CF_t} \tag{6.3}$$

6.2.3.3 Model Training and Prediction

The **causal_aware_model** is trained as presented in Algorithm 7. This model is in fact a DL network that is capable of predicting the mobility class of a user and is trained using dataset $\mathcal{D}$ which contains generated causal features.

The selected DL model, whose high-level architecture is depicted in Figure 6.4, regards a feed-forward NN with fully connected layers followed by an output layer. Since the proposed method mostly focuses on the insertion of Causally-enhanced data the architecture of the model is not thoroughly manipulated.

As far as the output layer is concerned, it utilizes the *categorical cross-entropy* activation function in order to generate $m(u_i) \in [0, 1, 2]$, which refers the user's mobility class. The higher the output, the higher the probability metric for this specific user. Finally, the user is then categorized in the three main categories; namely *static*, *local* and *mobile* using the rule presented in Equation 6.4.

$$m_i = \begin{cases} static, & \text{if } m(u_i) = 1, \\ local, & \text{if } m(u_i) = 2, \\ mobile, & \text{if } m(u_i) = 3. \end{cases} \tag{6.4}$$

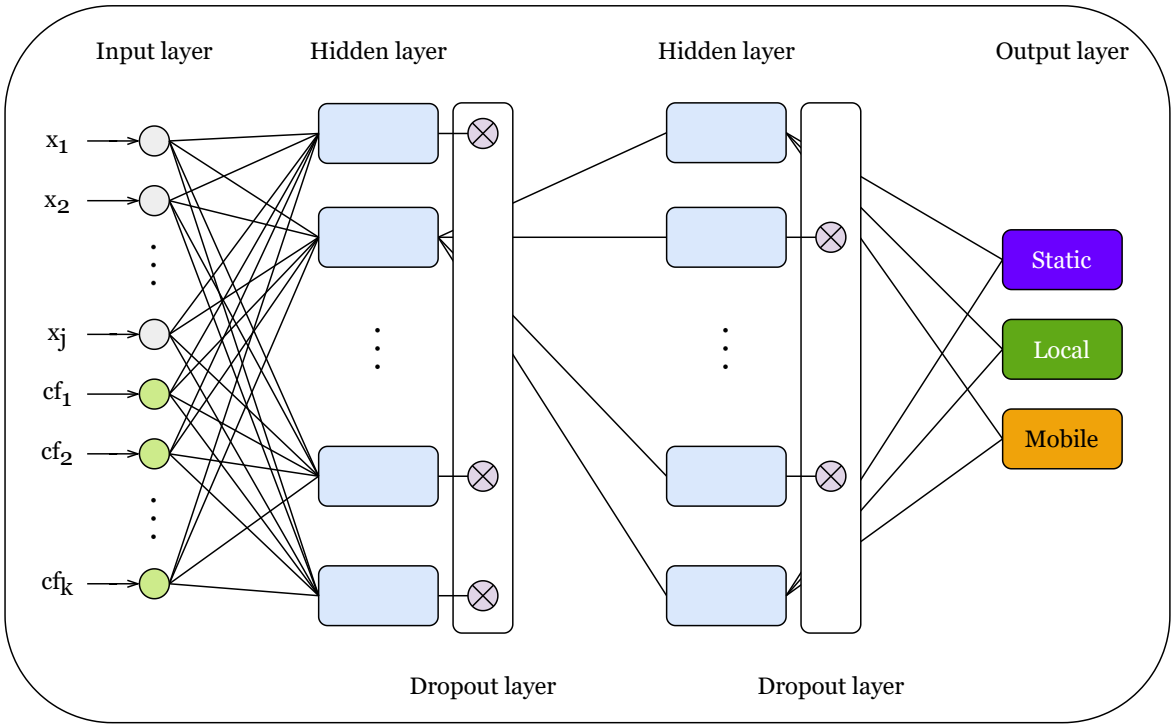


Figure 6.4: The feed-forward DL network used for the prediction of the mobility class of the users. It takes as input not only the original data features, but also the causal features as these are extracted from the previous steps, presented in Section 5.2.2. The original data features are depicted as x_i, while the generated causal features are depicted as cf_i. The output of the neural network regards the predicted mobility class of a user, i.e., they can be classified as either *static*, *local*, or *mobile*. In order to reduce over-fitting, dropout layers are inserted after each hidden layer.

6.2.4 Data Placement

As presented in Algorithm 8, the data placement process is initiated by training of the DL network with the causally-enhanced dataset $\mathcal{D}$ consisting of all users' information. This step is only performed once and may only be repeated occasionally in order to further enhance the network's results.

Consequently, the user's information over the last q days is given as input to the DL network which then performs a prediction of this user's mobility class.

In case the user is classified as *static*, then the optimal node o_i of this specific user

Algorithm 8 Data Placement process

Input:

$a_i = [k]$: User's visited nodes in the last q days.

Auxiliary Variables:

$\mathcal{D}$: the dataset including all information about the users' interactions,
m_i: user's mobility,
$m(u_i)$: user's mobility as predicted by the **causal_aware_model**,
v_i: the node the user is currently using,
o_i: user i's optimal node,
$s(d_i)$: data d_i's size,
$op_{(v_i,v_j)}$: optimal path between nodes v_i and v_j,
$cs(v_i)$: v_i's occupied storage,
$c(v_i)$: v_i's total capacity,
d_i: user's data to be placed,
n_i: nearest selected node,
R: A list including all nodes in order based on their ranking.

Output:

The Data Placement Decision based on the mobility of u_i.

Algorithm:

$m(\mathcal{D}) \leftarrow$ **causal_aware_model**$(\mathcal{D})$ *// Train the Causal DL model with the causal enriched dataset.*
$m_i \leftarrow m(u_i)$ *// Predict mobility of user* u_i.
if $m_i == 0 \,||\, m_i == 1$ **then** *// i.e., the user is classified as static or local.*
    $o_i \leftarrow max(a_i)$
    $op_{(v_i,o_i)} \leftarrow$ **optimal_path**(v_i, o_i) *// From Algorithm 9.*
    if $cs(o_i) + s(d_i) \leq c(o_i)$ **then** *// Optimal node can be used*
        return placement_decision$(d_i, v_i, o_i, op_{(v_i,o_i)})$
    else *// Optimal node cannot be used - storage limit exceeded*
        $n_i \leftarrow$ **find_nearest_neighbor**(o_i, d_i)
        $op_{(v_i,n_i)} \leftarrow$ **optimal_path**(v_i, n_i)
        return placement_decision$(d_i, v_i, n_i, op_{(v_i,n_i)})$
    end if
else *// i.e., the user is classified as mobile.*
    $h = 0$
    while true **do**:
        if $cs\,(R[h]) + s(d_i) \leq c\,(R[h])$ **then** *// The current node can be used*
            $o_i = R[h]$
            $op_{(v_i,o_i)} \leftarrow$ **optimal_path**(v_i, o_i)
            return placement_decision$(d_i, v_i, o_i, op_{(v_i,o_i)})$
        end if
        h + =1
    end while
end if

needs to be selected. This selection is performed by viewing the interactions this user had with the edge network over the last q days and the node that the user mostly interacted with, is selected as optimal. Afterwards, edge server o_i is inspected on whether it is able to store the new data d_i with respect to their size. If the edge server has the enough space to store the user's data, the optimal path between the user and the optimal node is found, by utilizing the Algorithm 9. Otherwise, the **find_nearest_neighbor** algorithm is used in order to identify the closest neighbor to o_i that can host the data. This adjacent edge node is stated as n_i. It is imporant to mention that the adjacent edge server needs to belond in the same cluster with the optimal edge node. The optimal path between the v_i and the n_i is then calculated using the **optimal_path**) algorithm, as presented in Algorithm 9.

In the case of a *local* user, the process follows a similar manner as in the static users. The proposed algorithm identifies the optimal edge server o_i and places the data there. If the edge server o_i is incapable of storing the data of the user (*i.e.*, the free storage capacity of the optimal node is not sufficient to store the user's data), then the **find_nearest_neighbor** algorithm is used once again, in order to select the closest neighbor, which as noted in the static users case, needs to be one of the servers within the same cluster of the optimal edge server. Regardless, the **optimal_path** algorithm is applied in order to calculate the least distance between the edge server that the user is currently connected to, and the one that is chosen to store the user's data. Since a local user may be linked to one or two clusters, this means that is such cases, the optimal edge server selection takes into account the nodes of both clusters.

Last but not least, if the user is classified as *mobile*, the data placement process, states that the data should be placed on the edge node or nodes that has on average the least transmission cost across the whole edge network, while also taking into consideration their capacity and how frequently they have been visited by the user. For this reason, a ranking metric has been defined which rates the edge

nodes according to the aforementioned characteristics.

The eccentricity of a node regards a metric that measures the maximum distance between an edge node i and any other node in the graph. Once the eccentricity of each node is calculated, having also each one's available capacity, the nodes' ranks are generated. The nodes are then sorted against their eccentricity along with their capacity and produce two numbers; the rank the node is when sorted by eccentricity and the rank the node is when sorted by available capacity. The centrality of any node i is produced as defined in Equation 6.5:

$$centrality_i = \frac{(M - ecc\_pos_i) \times (M - cap\_pos_i)}{M} \tag{6.5}$$

where M is the total number of nodes in the graph. The higher the position a node gets in either one of the ranks, the better the overall rank of it. For instance, consider a graph network consisting of 50 nodes. If an edge node is ranked first on both ranks (note that the position of the node ranked first is 0 and the position of the node ranked last is $M-1$), the overall rank would be equal to 50 ($M-ecc\_pos = 50$, $M-cap\_pos = 50$, divided by 50). While in the case where a node is ranked last on both ranks, the overall rank would be equal to 0.02 ($M - ecc\_pos = 1$, $M - cap\_pos = 1$, divided by 50).

Consequently, a list as defined in Equation 6.6 is produced, where it is measured how frequently each node i has been visited by each user j:

$$F[ij] = \sum_{k=1}^{n} \delta(a[j], i) \tag{6.6}$$

where a regards the list of the previously visited nodes of the user j and δ the frequency of visits of j to each node i.

The edge node that is selected for a given user is the one with the highest rank. The

ranking system is defined by solving the Equation 6.7. The node that is the highest in the *rank* metric, is selected for storing the user's data, as long as its capacity is sufficient to store the user's data and the data are placed there (as also depicted in Algorithm 8 (lines 15-22). Following, the optimal path between the v_i and the o_i (*i.e.,* the selected edge node) is calculated using the **optimal_path** algorithm.

$$rank_{ij} = \frac{(M - centrality_i) \times (M - F[ij])}{M} \tag{6.7}$$

The procedure outlined above facilitates to consistently account for the user's movements, resulting in the selection of the most appropriate edge node in the vast majority of instances, however, there are situations in which the designated edge node may not be the optimal one. In such cases, what is calculated is, (*i*)the migration cost in case the user requested the data from the node where the data is stored (*Cost*), and (*ii*) the migration cost to migrate the data to the user's current edge node plus the migration cost if the user's data were already in their current edge node and then were requested for retrieval (*Cost'*). Thus, if this difference exceeds a specific threshold, as shown in Equation 6.8, the data is migrated to the user's current edge node, otherwise, the data stays on the original edge node.

$$Cost - Cost' > threshold \tag{6.8}$$

In all cases, as also seen in Algorithm 8, the **placement_decision** (lines 7, 11, and 19) is produced, which is constituted by the information regarding the data d_i that has to be placed, the user's current edge node v_i, the edge node to which d_i will be placed to (the final edge node may either be the optimal o_i or its sufficient neighbor n_i, the current edge node v_i or its sufficient neighbor n_i), and the optimal path between the v_i and the selected edge node.

6.2.4.1 Neighbor Selection algorithm

The **find_nearest_neighbor** algorithm is responsible for identifying an alternative node (depicted in Algorithm 8 as n_i) that has the capacity to store the user's i data d_i. This requires a two-step process. As a first step, the adjacent nodes of the chosen one (either o_i or v_i) are identified and the ones whose capacity c_i is not sufficient are excluded. Next, the remaining nodes (which we will annotate as $r \in R$, where R is the list of all non-excluded nodes) are sorted by distance against the chosen node. Whichever has the minimum distance $l_{(v_i|o_i),r} = min(l((v_i|o_i), r)), \forall r \in R$ is then selected as the n_i.

Algorithm 9 Optimal Path Identification algorithm

Input:

v_i, v_j: the two edge nodes whose shortest path needs to be identified.

Auxiliary Variables:

$P[v_i, v_j]$: set of all possible paths between two nodes v_i and v_j,

$S[v_i, v_j]$: total transmission cost between two nodes v_i and v_j for each one of the paths in P,

PS_{v_i,v_j}: A tuple consisting of all available paths between v_i and v_j and their corresponding distances,

$PS\_Sorted_{v_i,v_j}$: The sorted PS_{v_i,v_j} tuple in ascending order.

Output:

l_{ij}: The shortest path between v_i and v_j.

Algorithm:

1: $P[v_i, v_j] \leftarrow$ Discover all possible paths between v_i and v_j
2: **for each** $p \in P[v_i, v_j]$ **do**:
3: Calculate transmission cost and assign it to S
4: **end for**
5: $PS_{v_i,v_j} \leftarrow (P, S)$ *// A tuple consisting of all paths and their corresponding transmission costs*
6: $PS\_Sorted_{v_i,v_j} \leftarrow$ **sort**(PS)
7: $l_{ij} \leftarrow PS\_Sorted_{v_i,v_j}[0]$
8: **return** l_{ij} *// The path with the least distance*

6.2.4.2 Optimal Path Selection algorithm

Let $P[v_i, v_j]$ be the set of all possible paths between two edge servers, v_i and v_j. The purpose of the **optimal_path** algorithm, as also presented in Algorithm 9, regards the identification of the optimal path l_{ij} between these two edge nodes. The process starts by discovering the available paths between the two edge nodes. The possible paths are then assigned to the list $I[v_i, v_j]$. Afterwards, the paths are sorted with respect to their total cost (since the $G(V, E)$ is a weighted graph, where E regards the edges between the edge servers $v \in G$, whose weight, *i.e.*, the transmission delay between the two connected edge nodes, is known) and the path with the least distance is selected as the optimal path.

6.2.5 Data Retrieval

This Section presents how a user may retrieve their data from the edge nodes. The same data retrieval process is performed when it comes to *static* or *local* users, while the data retrieval process is different when it comes to *mobile* users. In any case both processes will be presented accordingly.

6.2.5.1 Data Retrieval for a *static* or a *local* user

Consider a user whose optimal node is identified in advance, as shown in Figure 6.5. When a user performs a request to retrieve their data, this request is forwarded to the optimal edge node that was initially selected for storing their data. This edge node is responsible for transmitting the user's data back to the user.

In the case where the data of this specific user has been stored in other edge nodes as well, as also depicted in Algorithm 10, the optimal edge node is responsible for retrieving it from them as well (lines 4-6 if the user's current edge node matches the node where their data is, and lines 11-14 if the user's current edge node does not match the node where their data is) and then forwarding them to the user.

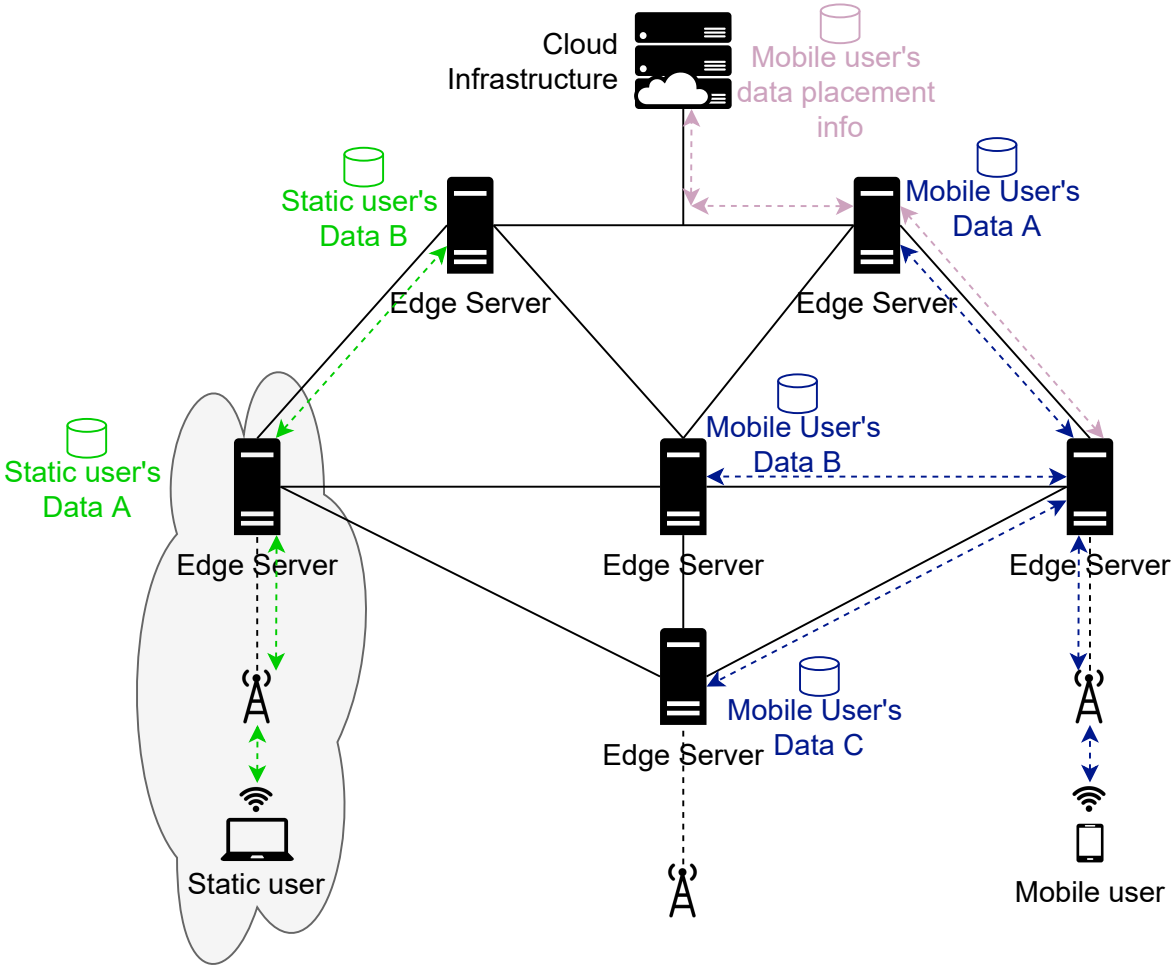


Figure 6.5: Data retrieval process for both *static* and *mobile* users. The static user's data is mostly located in one node, while the mobile user's data is placed in different nodes across the edge network.

This simplifies the retrieval process process since it is an option only occurring to static or local users whose optimal edge node did not have sufficient resources to store their data. In that case, the optimal edge node knows which adjacent edge nodes are selected for storage and performs a direct retrieval request to them. Once all data objects are collected by the optimal edge server, the actual process of transmitting the data to the user, is initiated.

Algorithm 10 Data Retrieval process for static and local users

Input:

v_i: the node the user is currently using,
o_i: the optimal node of the user.

Auxiliary Variables:

$op_{(v_i,v_j)}$: optimal path between nodes v_i and v_j,
dd_i: a Boolean value stating whether the user i's data is only stored in o_i, or not,
d_i: data to be retrieved by user.

Output:

The Data Retrieval decision for a *static* or a *local* user.

Algorithm:

1: **if** $v_i == o_i$ **then** *// The user's current node matches the user's optimal node*
2: $dd_i \leftarrow$ **user_has_data_in_other_nodes**(o_i)
3: **if** $dd_i == True$ **then**
4: d_i += **retrieve_data(**n_i, v_i, op_{o_i,v_i}**)**
5: **end if**
6: **return retrieve_data_decision**(d_i, v_i)
7: **else** *// The user is using a node different from their optimal one*
8: $op_{(v_i,o_i)} \leftarrow$**optimal_path**$(v_i, o_i)$
9: $dd_i \leftarrow$ **user_has_data_in_other_nodes**(o_i)
10: **if** $dd_i == True$ **then**
11: d_i += **retrieve_data(**n_i, o_i, op_{n_i,o_i}**)** *// Retrieve data from adjacent nodes to optimal node*
12: **end if**
13: d_i += **retrieve_data(**o_i, v_i, op_{o_i,v_i}**)**
14: **return retrieve_data_decision**(d_i, v_i)
15: **end if**

6.2.5.2 Data Retrieval for a *mobile* user

If a user is categorized as *mobile,* the data retrieval process is more complicated. As also presented in Algorithm 11, the data may be stored in several edge nodes across the edge environment and must be collected from all of them (line 4), aggregated in the edge node the user is currently connected to, and then forward it to the user (line 11).

Every time a mobile user performs a data placement, the data is placed on the current best overall-ranked edge server that has the capacity to store it. The information regarding which nodes have been used to hold the user's data is all kept in

the centralized cloud infrastructure that acts as the managing component of the edge network. Thus, whenever a data retrieval request is performed by a mobile user, the edge node the user is currently connected to, asks the centralized cloud component and retrieves this information. Consequently, individual requests are performed from the current edge node, to all the edge nodes that contain the user's data.

For each data transmission between the current edge node and any edge node that has user's data, the **optimal_path** algorithm is called, in order to identify the optimal path between the two edge nodes and assist in minimizing the transmission times. Once all data is transmitted to the user's current edge node, the actual transmission from the current edge node to the user's device is instantiated.

Algorithm 11 Data Retrieval process for mobile users

Input:
v_i: the node the user is currently using.
Auxiliary Variables:
Q: List of node hosting user i's data retrieved from the centralized cloud infrastructure,
$op_{(v_i,v_j)}$: optimal path between nodes v_i and v_j,
d_i: data to be retrieved by user.
Output:
The Data Retrieval decision for a *mobile* user.
Algorithm:

1: $Q \leftarrow$ **retrieve_nodes_list**(u_i)
2: **for** $q_i \in Q$ **do**
3: $op_{(v_i,q_i)} \leftarrow$**optimal_path**$(v_i, q_i)$
4: d_i += **retrieve_data**(q_i, v_i, op_{v_i,q_i})
5: **end for**
6: **return data_retrieval_decision**(d_i, v_i)

6.3 Evaluation Results

This Section presents the results of the simulations that were performed, in order to evaluate the proposed data placement strategy. First, the simulated envi-

ronment along with the metrics that will be used for the evaluation of both the proposed User Mobility-based Data Placement strategy, and the Causal Features Generation method will be described. Consequently, the results and the lessons learned from the experiments will be further analyzed.

6.3.1 Simulation Description and Metrics used for the Evaluation

The goal of the proposed User Mobility-based Data Placement strategy is to maximize the QoS of the enrolled users in the edge network, while optimizing the utilization of the edge network's computing resources. In order to measure the QoS the transmission cost metric is utilized, as it will be described in the following paragraphs. The edge nodes' storage capacity along with the distribution of the data items throughout the network two metrics are taken into consideration, in order to measure the resource utilization. In order to evaluate the proposed data placement strategy simulations were executed in an edge computing network whose topology is presented below.

The edge network topology regards a randomly generated network graph G, with 50 edge servers, and 100 users. Each edge node has a storage capacity which is randomly set and ranges from 500 MB to 1000 MB. Each user may have data whose size is randomly set within the range of 100 MB to 200 MB each. The graph edges (*i.e.*, the links between the edge nodes) correspond to the delay (*a.k.a.*, network cost) between the two connected edge nodes and is set randomly from 1 to 20 ms. A sample edge network graph, including the users, each connected to an edge node of the network, is depicted in Figure 6.6.

In order to evaluate the performance of the proposed data placement strategy, the following performance metrics were utilized:

- *Average path length*: The average path length is used to measure the distance between the edge node the user is currently connected to and the edge node

where the user's data are placed.

- *Average transmission cost*: The average transmission cost measures the average latency (sum of delays) for each edge between the user's current edge node and the edge nodes where the user's data is placed.

- *Distribution of data items in nodes*: The distribution of data items metric evaluates whether the data are uniformly distributed across the edge network.

- *Used storage capacity in nodes*: The storage capacity when combined with the distribution of data items in nodes in order to assure that the proposed User Mobility-based Data Placement strategy does not rely only on a small subgroup of the available edge nodes yet manages to successfully distribute the load across the whole edge network.

In addition, in order to evaluate the performance of the DL model that predicts the mobility of the users (through the evaluation of the DL model, indirectly the Causal Features Generation method is evaluated as well), the following metrics are used; Accuracy, Precision, Recall and F1-Score. Regarding Precision, Recall, and F1-Score the results for each class in Table 6.3 are presented. Furthermore, Table 6.2 depicts the confusion matrix summarizing the overall performance of the Causal-aware model when the Causal Features are included in the dataset, and comparing those results when those features are not included in the dataset used for the training of the model.

6.3.2 Simulation Execution and Discussion of Results

The simulation started with the creation of the randomly generated edge network as it was described in the previous Section. Consequently, 50 workers were deployed in order to perform random walks in the graph each one with a walk length of 100. The produced walks were then stored in a separate file which was used as

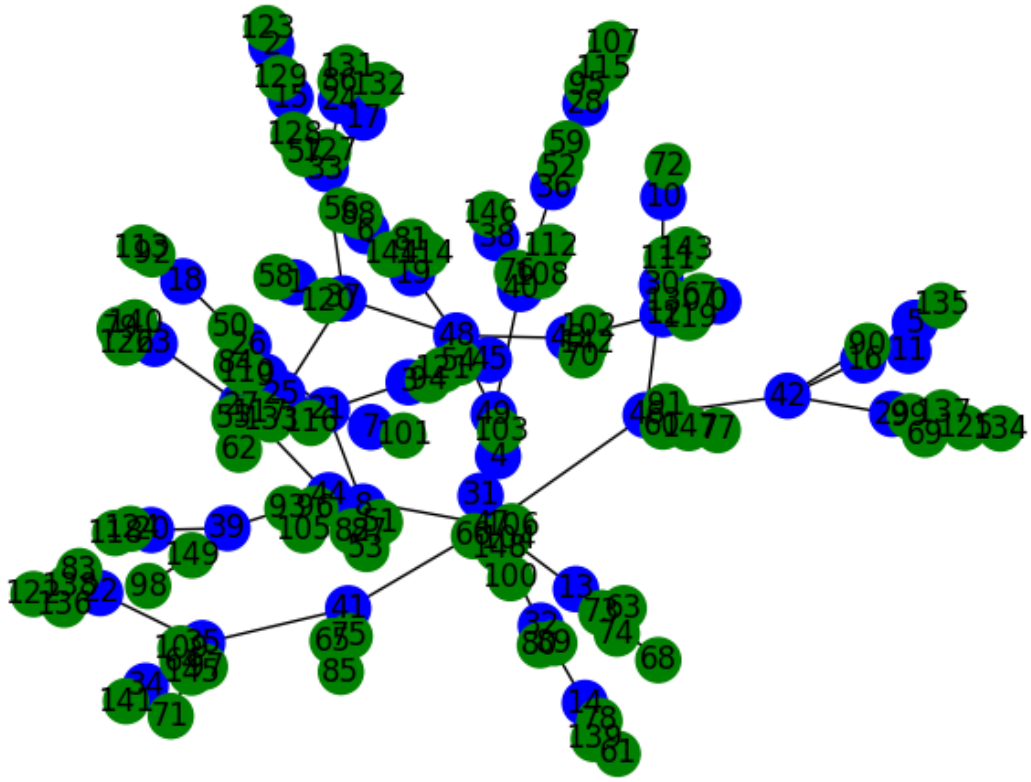


Figure 6.6: An edge network configuration and connected users. The nodes are colored in blue, while the users are colored in green with their corresponding ID numbers.

input to the K-Means algorithm in order to produce the clusters, as shown in Figure 6.2. K (*i.e.*, the number of clusters or neighbors as they were described before) was set to 6.

Starting with the average path length metric, as also seen in Figure 6.7, the proposed User Mobility-based Placement strategy performed significantly better when it comes to *static* and *local* users, which is reasonable given that for these user types, the data was placed within the cluster the users were assigned to. On the other hand, in the case of *mobile* users, the proposed strategy performed poorer in comparison, but even in the worst case scenario (*i.e.* the mobile user case), the proposed Data Placement strategy performed similarly when compared to other

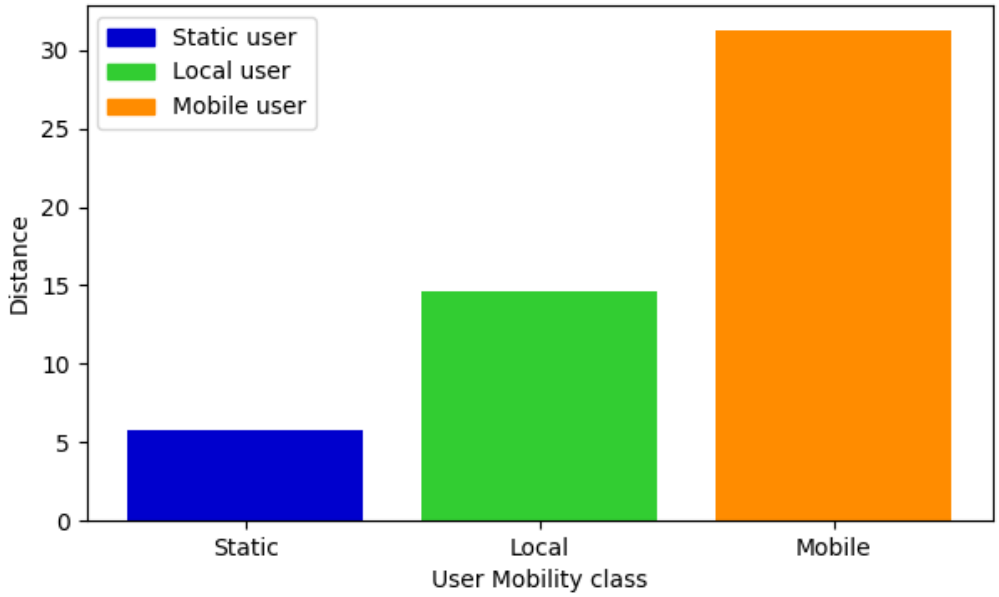


Figure 6.7: Average distance between the users and their data for each mobility class.

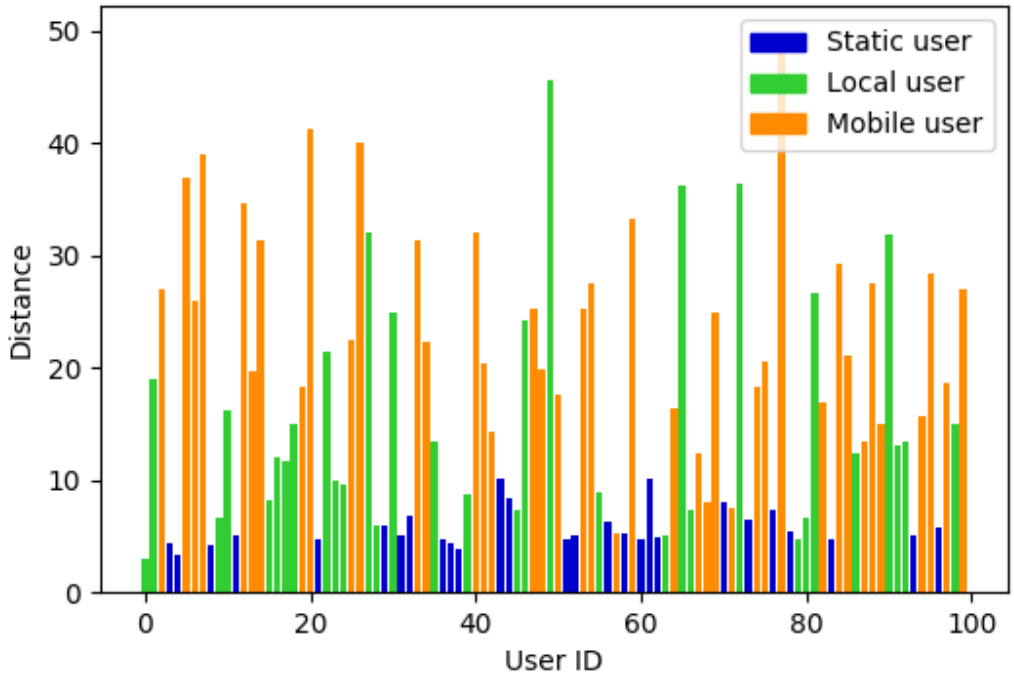


Figure 6.8: Average distance between the users and their data.

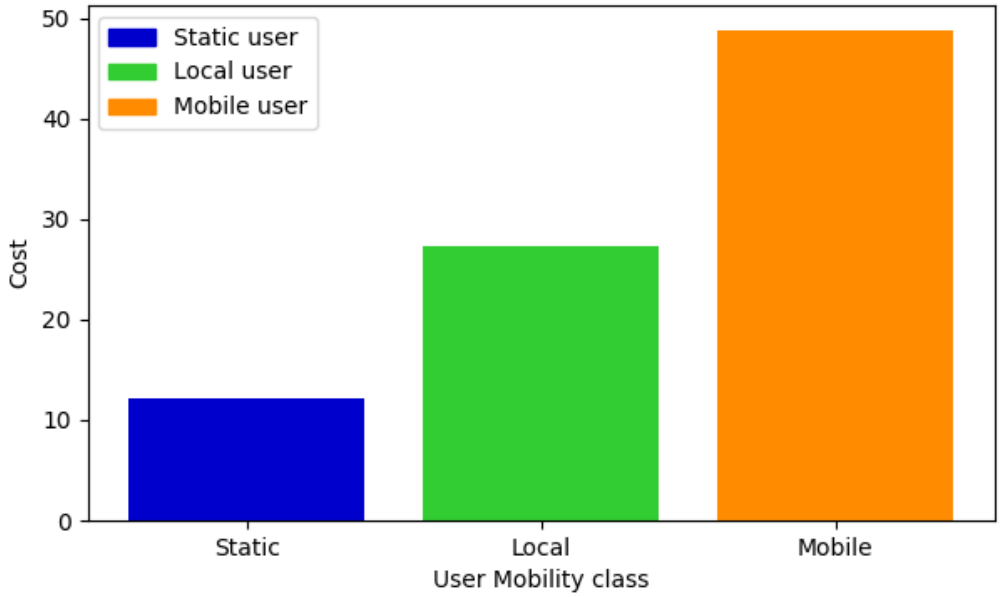


Figure 6.9: Average accessing cost between the users and their data for each mobility class.

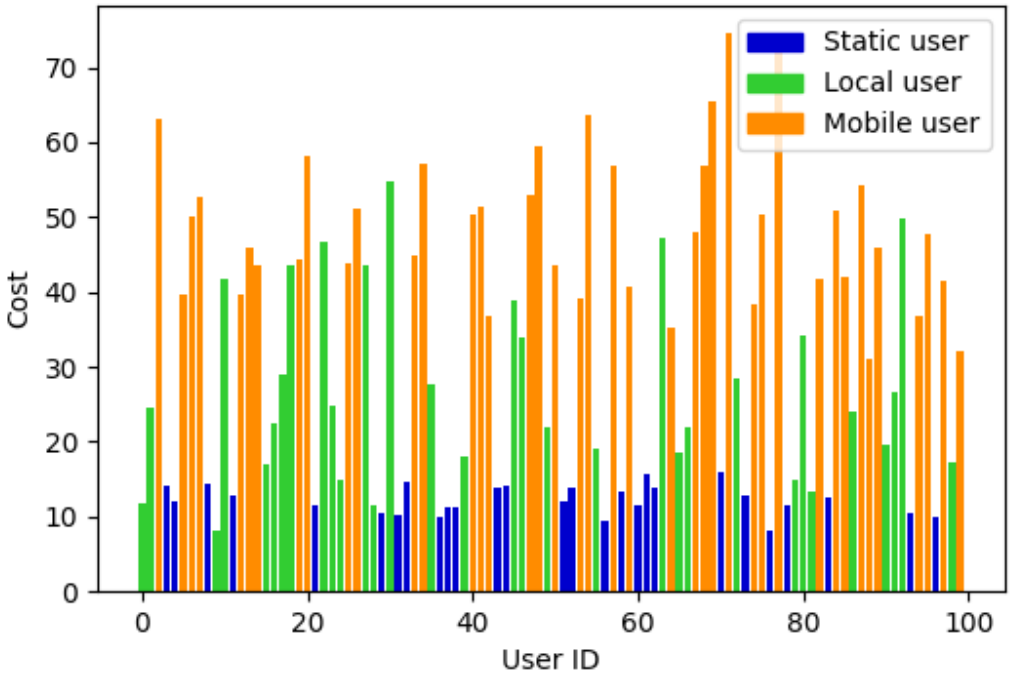


Figure 6.10: Accessing cost for every user.

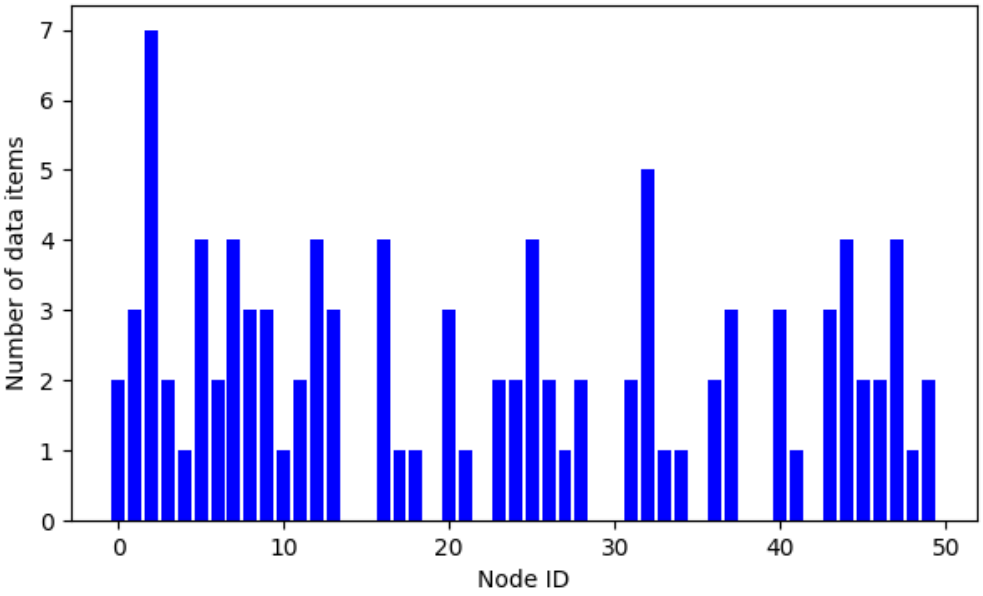


Figure 6.11: Number of data items per edge node.

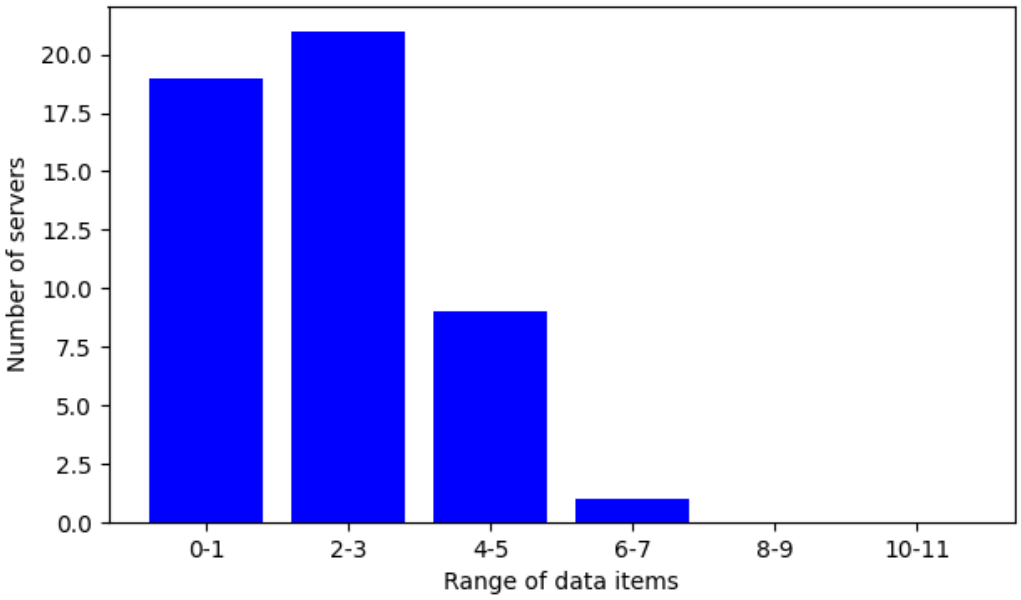


Figure 6.12: Range of stored data items per edge node.

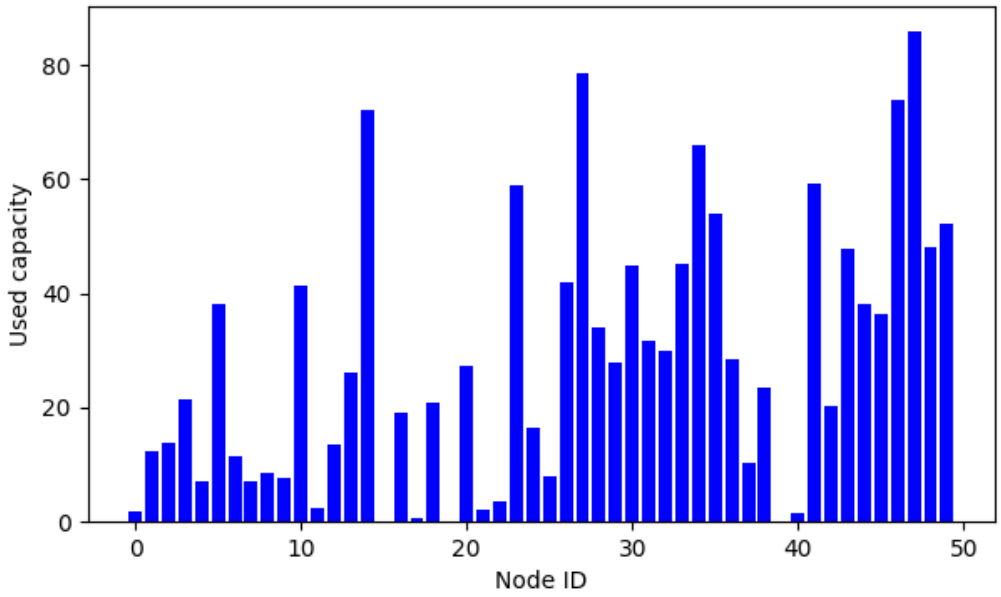


Figure 6.13: Used capacity among all edge nodes.

solutions with similar simulation configurations [Wei and Wang, 2021].

The proposed strategy significantly outperforms similar solutions, especially when it comes to static and local users, while it performs adequately when it comes to mobile users. In more detail, the average path length for all users was equal to 17.89, while specifically for static and local users, the same average path distance was equal to 5.72 and 14.67 accordingly. To compare with, when it comes to mobile users, the same metric was on average equal to 31.31.

A comprehensive view of the average distance for each user in the simulated environment is depicted in Figure 6.8.

Another metric which was used for the performance evaluation of the proposed User Mobility-based Data Placement strategy concerns the accessing (or transmission) cost for each individual user (as depicted in Figure 6.10 which presents the average cost per user in the simulation) and per mobility class as referred to, above. As depicted also in Figure 6.9, similar results to the average distance metric have

been in observed, which means that the proposed strategy successfully did manage to keep the accessing cost low in the cases of *static* and *local* users. When it comes to *mobile* users, the cost was increased when compared to the previous two mobility classes, but it was in general under acceptable limits. In more detail, on average, the accessing cost for mobile users was equal to 48.87, while for static and local users it was equal to 12.12 and 27.25 correspondingly. The overall average accessing cost was equal to 32.03.

In order to assure that the data placement strategy does not utilize only a small part of the available edge nodes, but instead utilizes the majority of them, the average used capacity among all the available edge nodes was measured, in combination to the distribution of the stored data items among the edge nodes.

The results depicted in Figure 6.11 show that there were cases where some nodes were loaded with more data items that other nodes (such as the edge node with id 2) which was loaded with 7 data items. And even though this comes in contrast to the end goal, which is to avoid cases where there are several edge nodes with many data items, that would mean that many individual users were allocated to them, while also the average capacity across the nodes was low.

Yet, the majority of the data items were *equally distributed* across the network. This can also be established when examining Figure 6.12, where we see that most edge nodes stored on average $0-3$ data items and only a few over 4 data items, and in fact, on average there were 2 data items stored per node. Another key result that can be extracted when looking at Figure 6.13 is that although there exist a few edge nodes whose used capacity reached over 60%, only a very few number of nodes had a used capacity which exceeded 80%. In fact, on average, the used capacity across all 50 deployed edge nodes was at 28.4%, while the maximum used capacity percentage was identified at node with the id 47, which had 86% of its available storage in use.

A crucial element in this evaluation concerned the assessment of the DL model

		With Causal Attributes			Without Causal Attributes		
		Actual Class					
		Static	Local	Mobile	Static	Local	Mobile
Predicted Class	Static	22	2	1	18	5	2
	Local	4	27	8	7	19	9
	Mobile	0	3	32	2	8	30
	Total	27	32	41	27	32	41

Table 6.2: Confusion matrix depicting the mobility class predictions of the 100 users in the simulations.

which performed the classification of the users to mobility classes, and through that the evaluation of the proposed Causal Features Generation method. Given that the end task regarded a multi-class classification problem, the metrics that were utilized were Accuracy, Precision, Recall and F1-Score. As it can be extracted from Table 6.2, the model performed on average very well in predicting the users' mobility accurately, while on average the accuracy is 81% when including the generated the Causal Attributes, and 67% when the Causal Features were not included in the dataset that was used for the training of the model.

In addition, as also shown in Table 6.3, the model managed to successfully separate the mobility classes. Based on the above, it can be safely said that an improvement in the model's overall performance suggests that the generated causal features played a significant role in enhancing the model's predictive capabilities. This can be also drawn by calculating the Macro-averaged F1-Score where in the case where the Causal Attributes were used, it was equal to 0.816, while when absent the same metric was equal to 0.693.

	With Causal Attributes			Without Causal Attributes		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Static	0.880	0.815	0.846	0.72	0.667	0.773
Local	0.692	0.843	0.761	0.543	0.594	0.567
Mobile	0.914	0.780	0.842	0.750	0.732	0.741

Table 6.3: Precision, Recall, and F1-Score for each user mobility class.

To sum up, the evaluation of the proposed data placement strategy led to the fol-

lowing remarks:

- The QoS requirement that was initially set was met, since the overall accessing cost for accessing the data is less when compared to other similar solutions from the the State of the Art,
- The proposed solution performs in general better when it comes to *static*, or *local* when compared to *mobile* users which is understandable, given the nature of the mobile users. Nonetheless, it still performs average when compared to the State of the Art even for the *mobile* users category.
- Initially, there was a major impact on the overall accessing cost when several static or local users were in one area (*i.e.*, a cluster of edge servers). For this reason, the strategy had to be adapted accordingly in order to make it more resilient in such cases. The adaptation that had to be made, in this scenario, allowed data placement on edge servers that were not part of the cluster the user was in but on adjacent ones as well.

In conclusion, the evaluation that it was performed proves that the proposed User Mobility-based Data Placement strategy can significantly reduce the data placement and data retrieval operations performed in an edge network, and that in most cases the optimal edge node is utilized to store the user's data. Finally, the simulated evaluation proved the ability of the proposed data placement strategy to utilize the whole edge network for data placement, rather than overloading only a few edge servers while the rest remained neglected.

6.4 Discussion on the Evaluation Outcomes and the Effectiveness of the Causal Features Generation method

The Causal Features Generation method was evaluated in two scenarios in which, a DL model was utilized for the classification of users of an edge computing envi-

ronment into mobility classes according to their behavior and their overall interaction with the system.

As the results suggested, the inclusion of the method as a preprocessing step in the training pipeline improved the model's overall performance, and the observations that can be extracted from this outcome are the following. First of all, the improved performance proves that the proposed Causal Features Generation method can make the DL model more robust by providing additional information and allowing it to generalize better to unseen data.

Another important note is that, by including these Causal-based Features in the training dataset, patterns or relationships in the data that were not evident in the original set are made easier to identify by the model.

Chapter 7

Edge Computing Optimization: A Causal Contextually-Aware Machine Learning Approach for Dynamic Resource Allocation

Chapter Structure

This Chapter is constructed as follows:

- **Section 7.1 - Background on Dynamic Resource Allocation at the Edge**, analyzes the State of the Art on Resource Allocation and Deployment Configurations on Cloud and Edge Computing Environments
- **Section 7.2 - Overview of the Dynamic Resource Allocation Framework**, presents an overview of the proposed resource allocation framework and highlights the adaptations that were made two both data enhancement methods (*i.e.*, the Causal Features Generation and the Influence-based Dataset Generation methods) in order to be incorporated in the given use case.
- **Section 4.3 - Evaluation Results**, evaluates the proposed framework and presents

the outcomes of the performed experiments.

This Chapter presents the evaluation of both data enhancement methods in the context of a framework for dynamic resource allocation Edge Computing infrastructures. In more detail, both methods have been utilized as a pre-processing step in order to train an ML model responsible for the prediction of the resource allocation of services deployed at the Edge [Symvoulidis et al., 2024].

7.1 Background on Dynamic Resource Allocation at the Edge

A new age of decentralized and distributed computing architectures has emerged in recent years with the rise of Edge Computing. With its focus on processing data closer to the source, Edge Computing brings forth exciting opportunities for both businesses and customers as well. From managing IoT devices in transporation [Zhou et al., 2021], [Lin et al., 2020], [Chavhan et al., 2022], or for the implementation of smart cities solutions [Khan et al., 2020], [Liu et al., 2019], [Lv et al., 2021].

However, there are additional challenges to overcome. One major issue involves the effective use of the resources at hand [Xiong et al., 2020], [Symvoulidis et al., 2023b], and as the demand for real-time and context-aware applications [Symvoulidis et al., 2019] increases, the demand for a strong framework for dynamic resource allocation grows. But although there exist solutions in the literature that aim at solving this matter [Tang et al., 2019a], [Wu et al., 2023], there is still work to be done, as in most cases, the proposed solutions often rely on heuristic or simplified assumptions, which do not fully encompass the dynamic and multifaceted nature of Edge Computing environments.

The authors of [Tang et al., 2019b] suggested a method for allocating resources

in hybrid cloud and edge computing systems for latency-critical applications. The proposed strategy utilizes two key algorithms, the first one being a resource scheduling algorithm and the second one being a resource matching algorithm. The resource scheduling algorithm, determines the task scheduling cost based on the data center transmission cost (*i.e.*, the cloud infrastructure part) and the edge servers. The resource matching method optimizes the resources to be allocated for the requested task on an edge server by taking into account the location of the resource, the overall cost of network transmission, and the priority of the tasks. It achieves this by considering the results of the resource scheduling algorithm.

The authors of [Deng et al., 2020b] suggested a Dynamic Throughput Maximum (DTM) method based on the *Lyapunov optimization* to optimize computation and communication resources for MEC environments with the goal of maximizing the overall throughput. Similar to this, the authors of [Plachy et al., 2016a] suggested a method for allocating resources for VM in MEC while incorporating the mobility of the users [Symvoulidis et al., 2023a]. In addition, the algorithm determines the best route, accounting for user movement, between the users and the deployed VM.

A Generative Adversarial Network (GAN) - assisted dynamic resource allocation scheme for MEC was proposed by the authors of [Gong et al., 2023], building on the work presented in [Kaur et al., 2021] and [Xu et al., 2020b]. In more detail, in order to forecast the demand for each edge node in the upcoming time slot and make allocation decisions based on this knowledge, the authors examine prior information (*i.e.*, historical data) related to user mobility.

The authors of [Xiao et al., 2012a] proposed a virtualization technology-based system which allocates the available computing resources dynamically, taking into consideration the applications' requirements. In more detail, the authors suggested using the "skewness" metric to quantify possible misuse of resources and the end goal was to minimize this metric, hence achieve optimizing resource al-

location and usage. In the same fashion, the authors of [Saraswathi et al., 2015], proposed a dynamic VM resource allocation model which can dynamically adapt the available virtual resources, based on the characteristics of the deployed services and as a result lead to an optimized use of the resources.

Gao *et al.* [Gao et al., 2022] proposed a dynamic resource allocation system for Virtual Network Function (VNF) in satellite edge clouds which aims at minimizing the services' delay while also minimizing overall network bandwidth cost. The proposed solution utilized a Distributed Virtual Network Function (D-VNF) algorithm.

Chhabra and Singh in [Chhabra and Singh, 2022] on the other hand, suggested an Service Level Agreement (SLA)-aware resource allocation scheduling method, which has two main phases. In the first phase it analyzes the resource requirements of a given service, while in the second phase it allocates the appropriate resources for the service's deployment.

The work of Yeh and Yu [Yeh and Yu, 2022] focuses on the design of a dynamic resource allocation model which can dynamically adjust the computing resources assigned to jobs in Hadoop [Apache, 2023a] towards the acceleration of their execution. The proposed model, is also capable to prioritize the execution of critical jobs by allowing these jobs to utilize more containers against the jobs with regular priority.

In a different use case, Lim *et al.* [Lim et al., 2021] raise the importance of training AI algorithms without moving the data away from their source; an important issue that is very much discussed in the existing data privacy-related policies, such as the General Data Protection Regulation (GDPR) in the European Union (EU). To deal with this matter, the authors propose the use Federated Learning (FL), but in order to exploit such techniques efficiently, several resource allocation related issues need to be resolved. To solve these issues the authors propose a two-level resource allocation and incentive mechanism, in which the resources are allocated

to workers according to the size of data they use for FL.

Plachy *et al.* [Plachy et al., 2016b] proposed a dynamic resource allocation model for MEC infrastructures, based on the users' mobility. In more detail, the authors proposed a computing and communication resources allocation algorithm where initially the user's mobility is predicted. Consequently, the most appropriate infrastructure to allocate a VM for the execution of the user's task is selected, and upon deployment of the VM, the optimal communication path is between the deployed VM and the user is identified to optimize the inter-communication between the two entities.

Given the above, it is evident that Edge Computing environments are very complex since they include multiple devices, heterogeneous workloads, and constantly shifting contextual factors [Tsoumas et al., 2021]. This is often ignored by traditional resource allocation algorithms [Xiao et al., 2012b]. Therefore, it is safe to assume that even though the research on resource allocation has progressed considerably, the subject requires further attention and should be investigated more thoroughly in order to design and develop more reliable and efficient solutions.

For this reason, in the following Sections, the proposed dynamic resource allocation framework is presented. The proposed framework is designed to perform dynamic resource adaptations on the edge servers of a given Edge Computing network, similar to the one shown in Figure 7.1.

The presented framework makes predictions using an ML model trained on a causally and contextually augmented dataset, which allows the model to generalize and adapt to dynamic contexts as an Edge Computing network. The enhancement of the dataset is performed using adaptations of the two proposed data enhancement methods (Influence-based Dataset Generation and Causal Features Generation) as they are described in Chapters 2 and 5 accordingly.

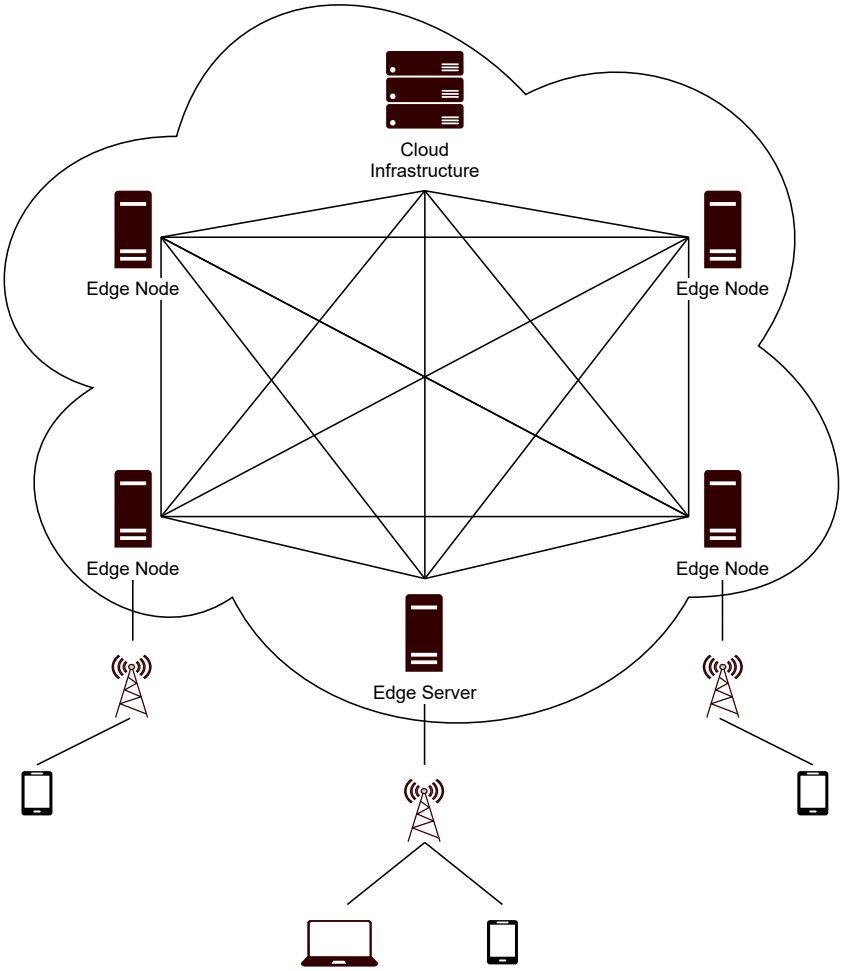


Figure 7.1: The Hybrid Cloud / Edge architecture of the system.

7.2 Dynamic Resource Allocation Framework Overview

7.2.1 Problem Formulation

In order to address the problem of dynamic resource allocation, it was formulated as presented below. First, the Edge network is considered a graph G, where $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_m\}$ are the edge nodes and $E = \{e_1, e_2, \ldots, e_n\}$ are the edges of the network connecting the nodes, as shown in Figure 7.1. This is a similar representation as in the graph network represented in Chapter 6. There are two

main goals for the proposed resource allocation framework: (*i*) to performe accurate predictions related to the resource utilization for the nodes of the given edge network, and (*ii*) to dynamically adapt the resources of any given node according to the predictions.

For each node component (*i.e.*, the servers and the links between them) of the edge network, it is considered that the following characteristics are known. To start with each server, for every server i, $v_i \in V$, $c_i = c(v_i)$ represents its storage capacity, $m_i = m(v_i)$ represents its memory usage, while $p_i = p(v_i)$ its CPU usage at any given time. In addition, $l_{ij} = l(v_i, v_j)$ contains the calculated least distance between the i-th node and any other node j, such that all least distances from node i to any node j can be found in the $L = \{l_{ij}\}$ distance matrix. The distance in this scenario refers to the physical distance of the nodes (*i.e.*, the number of edges that connect the two servers). Furthermore, the service migration cost w_{ij} between two nodes i and j is also measured.

Furthermore, the transmission delay between the node i and the node j or the user of the edge network can be found using Equation 7.1:

$$t_{ij} = \sum_{k=1}^{n} \delta + T_{proc} + T_{trans} \tag{7.1}$$

where δ is the default delay of the edge that connects i and j (measured in ms), T_{proc} the processing time of the service, and T_{trans} the transmission delay between i and j. The transmission delay can be calculated using Equation 7.2, below:

$$T_{trans} = \frac{d_k}{b_{ij}} \tag{7.2}$$

where d_k is the size of the transmitted data and b_{ij} the available bandwidth for the edge i and j.

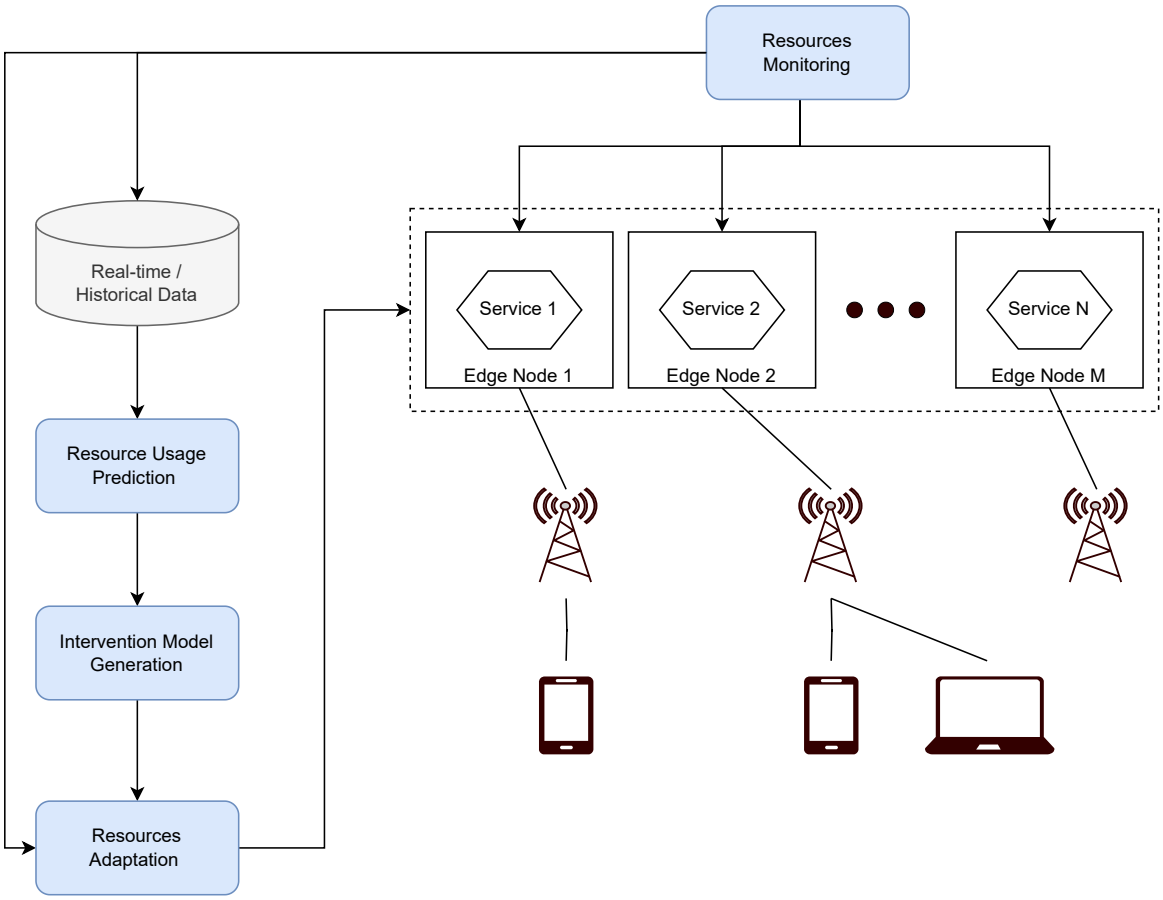


Figure 7.2: High-level Overview of the proposed Dynamic Resource Allocation platform.

Thus, the overall delay is defined as the sum of all pairs t_{ij} for any i and j, as also shown in Equation 7.3:

$$T = \sum_{i=1,j=1}^{m} t_{ij} \tag{7.3}$$

Based on the above, the proposed dynamic resource allocation framework aims at identifying the optimal actions in order to optimize resource utilization of the entire edge network, while also reducing the overall delay.

7.2.2 Core Features of the Dynamic Resource Allocation Framework

The proposed framework has four key features that enable the proper and accurate resource usage prediction and consequently the dynamic adaptation of the resources: (*i*) resources monitoring, (*ii*) resource usage prediction, (*iii*) intervention model generation, and (*iv*) resource adaptation.

7.2.2.1 Resources Monitoring

This feature allows the monitoring the resources of all computing nodes of the edge network. Essentially, it is a system that tracks critical metrics such as CPU and memory use, available disk storage, and network metrics like latency and egress and ingress bandwidth. All of the metrics are then saved in a MySQL database [Oracle, 2023].

7.2.2.2 Resource Usage Prediction

The resources prediction feature allows the prediction of the resource usage of any edge node at a given time. It utilizes both *data enhancement methods* in order to predict the utilization of two important metrics; CPU and memory. and the adaptation that are performed to the Influence-based Dataset Generation method and the Causal Features Generation method will be presented in detail below.

Prior to exploring the execution of the predictions, it is important to start with the process of data enhancement. To be more specific, the two methods are employed to improve the dataset. The process starts with the Causal Features Generation and continues with the identification of the most important instances in the dataset along with the the generation of the new *enhanced dataset*.

7.2.2.2.1 Adaptation of the Causal Features Generation method

To start with the *causal features* generation, as also shown in Algorithm 12, the process starts with the identification of the causal relationships among the features of the dataset. Only this time, instead of using the FCI algorithm, the causal discovery is performed using the DirectLiNGAM [Shimizu et al., 2011]. The outcome of the DirectLiNGAM algorithm produces an adjacency matrix $n \times n$, where n is the number of the features, as shown in Equation 7.4:

$$\begin{bmatrix} 0 & 0 & 0 & 1.534e-05 & 0 & 0 & 0 & 8.617e-09 & 2.366e-04 \\ 3.398e+07 & 0 & 0 & -5.739e+03 & 0 & 1.748e+05 & -6.874e+01 & 6.337e+00 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6.765e+00 & 7.287e-03 & 4.403e-05 & -5.556e+00 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -5.296e-04 & 0 & 0 & 1.376e+02 & 0 & 0 & -1.377+02 \\ 0 & 0 & 0 & 0 & 0 & -1.301e+04 & 8.389e+00 & 0 & -1.445e+04 \\ 0 & 0 & 0 & 0 & 0 & 9.999e-01 & 0 & 0 & 0 \end{bmatrix} \quad (7.4)$$

where each row and each column is any feature of the given dataset. The number in any position of the adjacency matrix refers to the causal effect that a feature (on the column) has on another (on the row). This can be also represented as an acyclic causal graph, in which each node represents a feature of the dataset and an edge between two nodes represents the causal relationship between these two features, similar to what is shown in Figure 7.3, where if A and B are two features of the dataset. In that case, any $A \xrightarrow{x} B$ relation shows that there exists a causal relationship between A and B, x represents how strong the causal effect is, and the direction of the arrow represents the causal influence between the two features (*i.e.*, that A causes B). In the proposed solution only the

Next, for the target variable(s) (which in this case it is the CPU and memory usage), the features that have a causal effect on them are selected using the Parents-Children approach. For each instance in the dataset, the occurrences which has the same value as the target are selected, given that they also have the same value

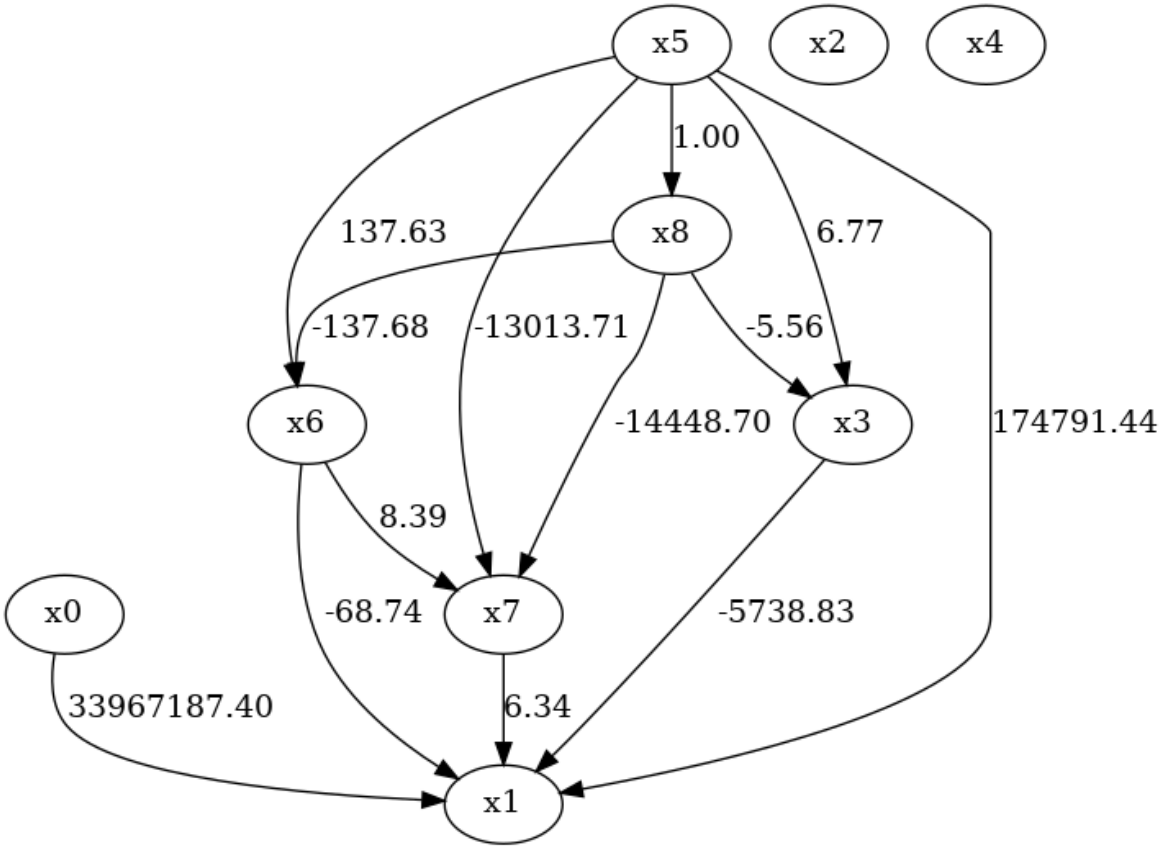


Figure 7.3: The produced Causal Diagram which was utilized for the generation of the causal features.

as the parent feature. In order to convert the value into a number in the $[0, 100]$ space, the following Equation 7.5 is utilized, which represents the newly created causal feature that is then appended to the original dataset.

$$causalFeature_i \leftarrow \frac{P(t_i|CF_{t_i}) - min(P(t|CF_t))}{max(P(t|CF_t)) - min(P(t|CF_t))} \times z_i \tag{7.5}$$

where z_i the z-score of the effect e_i as formulated in Equation 7.6 below:

$$z_i = \frac{e_i - \mu}{\sigma} \tag{7.6}$$

e_i regards the causal effect of the causal feature CF_t on the target feature t, μ the

mean of all effects, and σ the standard deviation of the effects.

Algorithm 12 Adapted Causal Features Generation method

Input:
$\mathcal{D}$: the dataset including all information about the users' interactions.
Auxiliary Variables:
dLiNGAM: Outcomes of the **DirectLINGAM** algorithm,
pc: Output of the **p_c** algorithm,
t: The target feature,
CF_t: List of feature t's parents.
e_i: The causal effect of CF_t on t,
Output:
$\mathcal{D}$: The original dataset, which now contains the causal features.
Algorithm:

1: $dLiNGAM \leftarrow$ **DirectLINGAM**($\mathcal{D}$)
2: $pc \leftarrow$ **p_c**($dLiNGAM$) **if** causal effect e_i is significant (*i.e.*, << 0 >>)
3: **for each** tuple in pc **do**
4: calculate occurrences of an instance appears in the dataset for $P(t|CF_t)$
5: **end for**
6: **for each** tuple i in pc **do**
7: $max((P(t|CF_t)) \leftarrow$ calculate max occurrences in the dataset for $P(t|CF_t)$ for any tuple in pc
8: $min(P(t|CF_t)) \leftarrow$calculate min occurrences in the dataset for $P(t|CF_t)$ for any tuple in pc
9: **end for**
10: **for each** tuple i in pc **do**
11: $causalFeature_i \leftarrow \frac{P(t|CF_t)-min(P(t|CF_t))}{max(P(t|CF_t))-min(P(t|CF_t))} \times z_i$
12: $\mathcal{D} : \mathcal{D} \cup causalFeature_i$ *// Append the new causal feature to the existing dataset*
13: **end for**
14: **return** $\mathcal{D}$

7.2.2.2.2 Adaptation of the Influence-based Dataset Generation method

The process continues with the utilization of the proposed Influence-based Dataset Generation method towards the generation of the *contextually enhanced dataset*. In order to utilize it, some adaptations to the generic method were performed, which will be explained in detail below. As also described in Algorithm 13, the selected ML algorithm is trained using the original dataset and the Mean Abso-

lute Error (MAE) and the Root Mean Square Error (RMSE) are calculated. Consequently, the instances of the dataset are evaluated with respect to their influence as follows. After an instance is removed from the original dataset, the model is retrained and the two aforementioned metrics are calculated again. In case the model's performance drops (meaning that the RMSE and MAE are both increased), then this instance is considered influential, and it is appended to the influential instances dataset, which is denoted as $\mathcal{I}$. Afterwards, the instace is re-inserted back into the original dataset, and the same process continues for all the instances in the dataset.

The next step regards the assessment of the instances that were not initially considered influential. These are then evaluated again by measuring their deviation from any of the already-identified influential instances. Every instance can be considered a vector of an X-dimensional space, where X is the number of the features, hence if the degree between the instance and any influential instance is less than a given value denoted as Θ, then this instance is also considered influential and is then appended to the influential instances dataset $\mathcal{I}$. The calculation of the degree $\theta(\vec{u}, \vec{v})$ between two vectors (*i.e.*, features) $\vec{u}$ and $\vec{v}$ is shown in Equation 7.7 below:

$$\theta(\vec{u}, \vec{v}) = \cos^{-1} \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \times \|\vec{v}\|} \tag{7.7}$$

where $\|\vec{u}\|$ is the length of the vector $\vec{u}$ which in this case should match the length of the vector $\vec{v}$, $\|\vec{v}\|$ as they are both features of the same dataset and thus have the same dimensions.

As far as the model that was selected for this task, in this case it is a Random Forest Regressor with 1000 trees. The reason behind the utilization of such a model, is because it has been proven to perform efficiently in similar tasks [Al Qassem et al., 2023], [Bei et al., 2015], [Chen et al., 2020], [De and Singh, 2016], [Kumar T et al., 2021].

Algorithm 13 Context Identification and Dataset Enhancement

Input:
$\mathcal{D}$: the original dataset.
Auxiliary Variables:
$\mathcal{I}$: the new dataset containing the influential instances,
Θ: the threshold degree.
Output:
$\mathcal{I}$.
Algorithm:

1: `fit(`$\mathcal{D}$`)`
2: $\mathcal{I} = [\,]$
3: calculate $MAE(\mathcal{D})$
4: calculate $RMSE(\mathcal{D})$
5: **for** $i \in \mathcal{D}$ **do**
6: $\mathcal{D}^{-i} : i \notin D$
7: `fit(`$\mathcal{D}^{-i}$`)`
8: calculate $MAE(\mathcal{D}^{-i})$ and $RMSE(\mathcal{D}^{-i})$
9: **if** $RMSE(\mathcal{D}^{-i}) \geq RMSE(\mathcal{D})$ && $MAE(\mathcal{D}^{-i} \geq MAE(\mathcal{D})$ **then**
10: $\mathcal{I} : \mathcal{I} \cup i$
11: $\mathcal{D} : \mathcal{D} - i$
12: **end if**
13: **end for**
14: **for** $i : i \in \mathcal{D} \wedge i \notin \mathcal{I}$ *and* $k : k \in \mathcal{I}$ **do**
15: **if** $\theta(i,k) \leq \Theta$ **then**
16: $\mathcal{I} : \mathcal{I} \cup i$
17: **end if**
18: **end for**
19: **return** $\mathcal{I}$

7.2.2.3 Intervention Model Generation

This is a feature of the proposed Dynamic Resource Allocation framework which can be used to make decisions related to the actions that need to be taken. As already specified, the goals of the proposed framework is to reduce the overall latency of the edge network and optimize the resources use. According to these goals, a set of actions are designed which can be taken, considering the resource usage predictions that were previously made.

The first goal is to optimize the average delay of the edge network so that it does

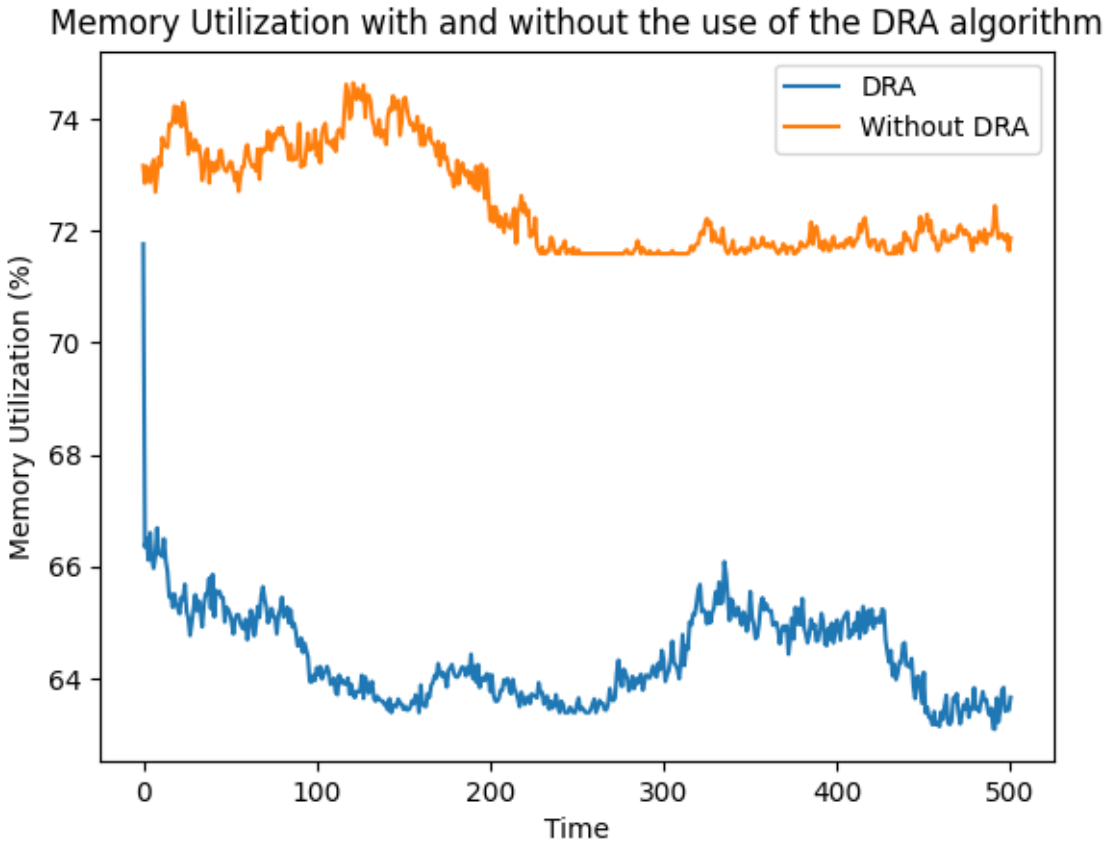


Figure 7.4: Comparison of Memory utilization when the resource allocation framework is used and is not used.

not exceed a given threshold, which is set to 10 ms. In case the average delay stays does not exceed the threshold, yet the CPU and / or the memory usage is over 80% the state is considered normal and there are no actions required.

On the other hand, if the CPU and / or the memory utilization is low ($< 80\%$), a request to free resources can be performed to the edge servers of the network that have the least CPU and memory utilization. In the event that the average delay exceeds the 10 ms threshold, a request to scale up is taken instead.

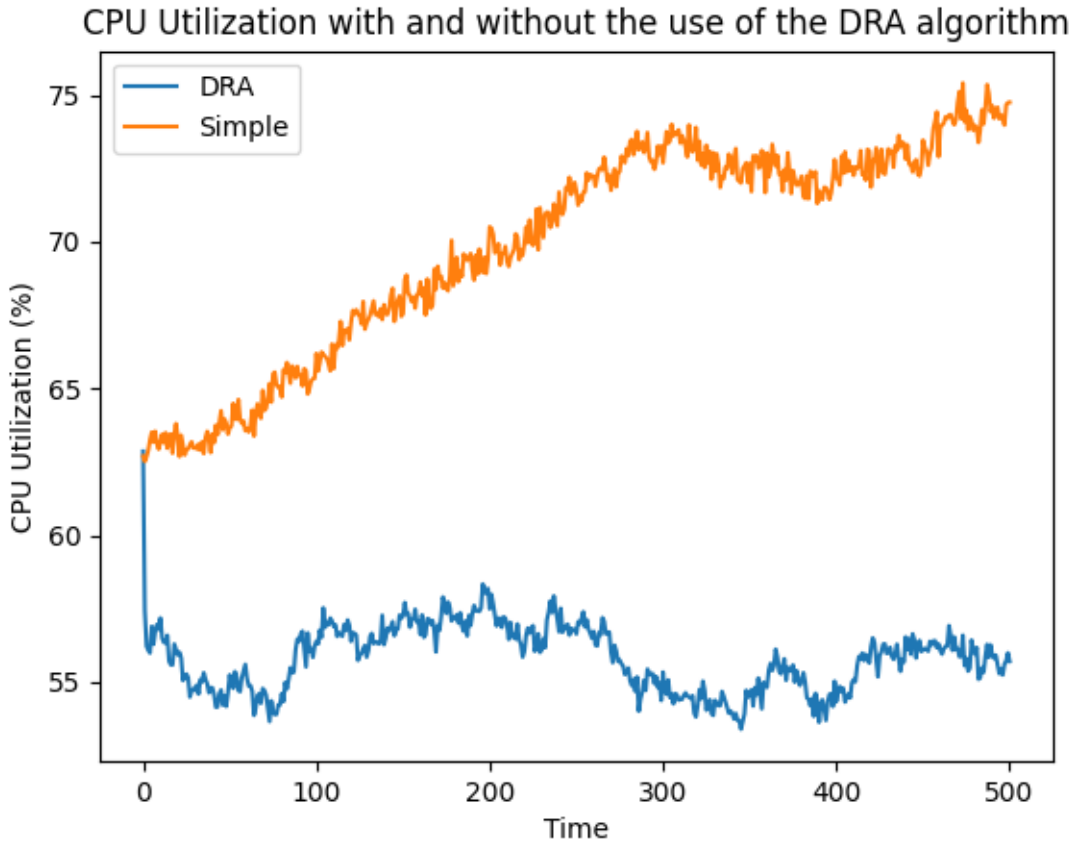


Figure 7.5: Comparison of CPU utilization when the resource allocation framework is used and is not used.

7.2.2.4 Resources Adaptation

This feature allows the decisions taken, to be actualized to the edge network. In more detail, with this feature, the proposed dynamic resource allocation framework can perform the necessary adaptations to the edge network as indicated by the decisions of the Intervention Model Generation feature.

In the case where freeing resources is requested, taking into account also the real-time monitoring information, the edge nodes with the least CPU and / or memory utilization are identified and a request to downscale is made. In contrast, the edge nodes with the highest resource utilization are scaled up in case the delay thresh-

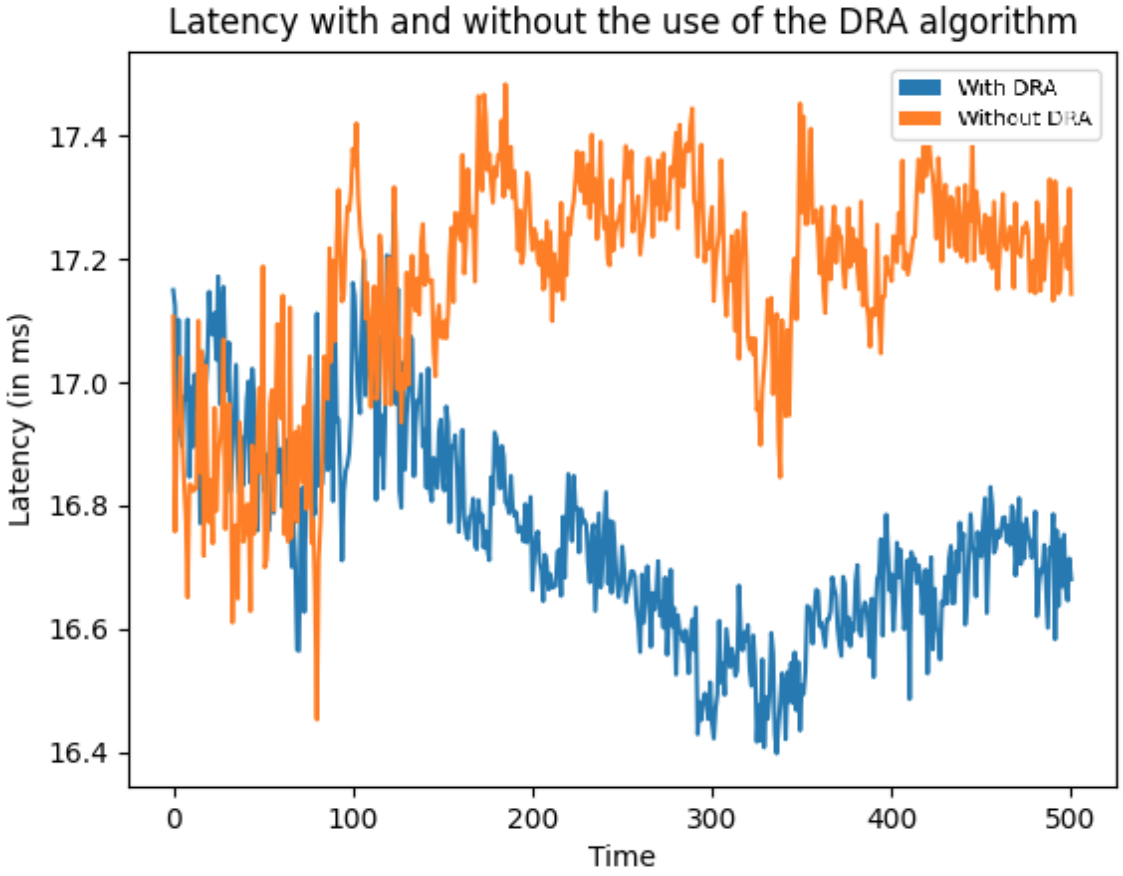


Figure 7.6: Comparison of the average total network latency over a time period when the resource allocation framework is used vs. when it is not used.

old is not met. If this adaptation is sufficient and the average delay drops below 10 ms, no other actions are required.

On the other side, if this does not improve the average latency (after a given number of iterations) the services with the highest latency are selected for re-deployment to the edge server with the least resource usage, in order to utilize those nodes better. What happens, essentially, is that the services are deployed to these servers too, given that they (the servers) have the storage capacity to host the services.

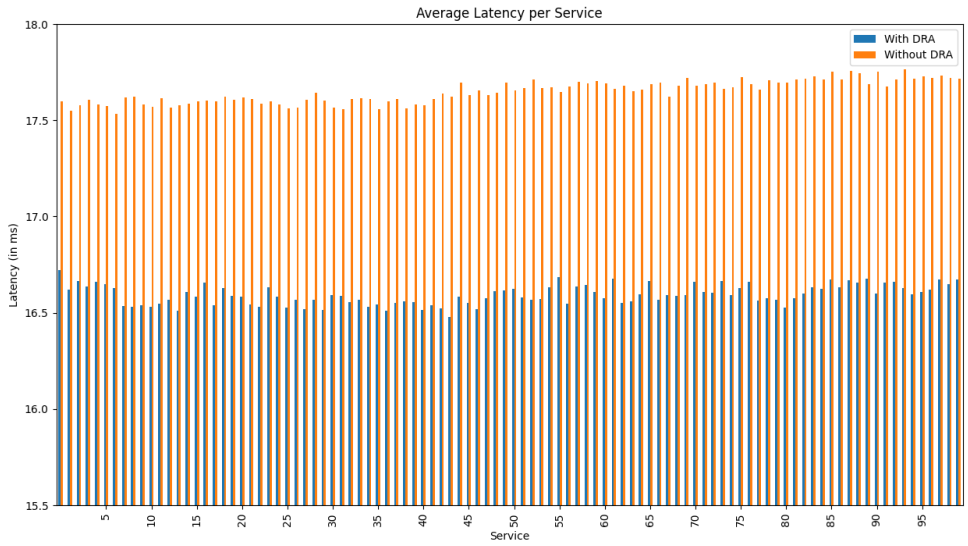


Figure 7.7: Average latency of the 100 services when the resource allocation framework is used and is not used.

7.3 Evaluation Results

Several simulations have been conducted in a simulated edge computing network in order to assess the suggested resource allocation paradigm, whose outcomes are detailed below. Starting with the network's topology, it regarded a randomly generated network graph denoted as G, where 50 edge servers were deployed, similar to the simulation of the evaluation of the User Mobility-based Data Placement strategy described in Chapter 6. The edges of the graph represented the delay (in ms) and was set in the range of $6-20$ ms for each edge node.

The difference with the simulations in Chapter 6, is that in this simulation, the required computing resources for the deployed services were taken into consideration as well. In more detail, a service was deployed randomly and required $5-10\%$ of CPU usage, $10-15\%$ of memory, and $1-5\%$ of storage, while during their initial deployment, they were allocated $10-15\%$ of the edge server's CPU, $15-20\%$ of the server's memory, and $5-10\%$ of the server's storage capacity.

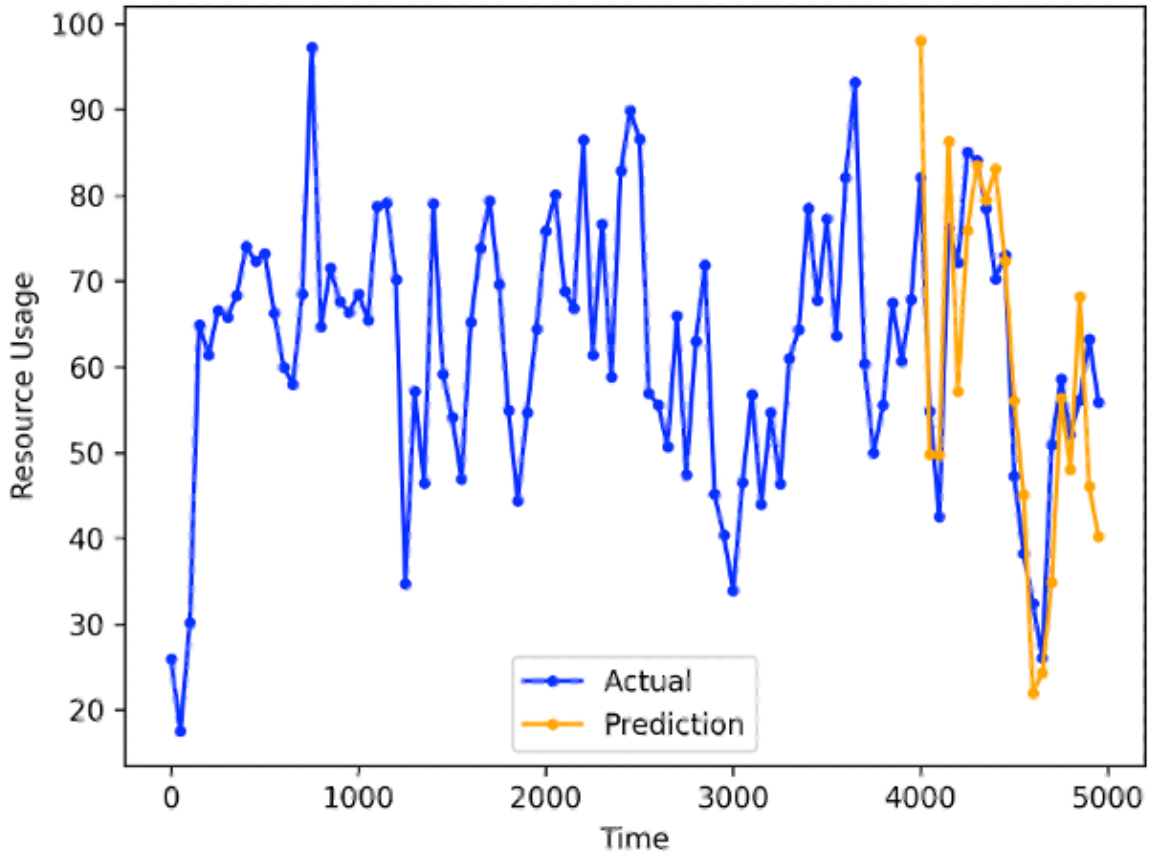


Figure 7.8: CPU Usage utilization and prediction, using the original dataset.

Once the Edge network was constructed, 1000 users were introduced to it and after every iteration, more users would inserted in the system, while others would be removed at a similar rate. They were connected to one edge server every time, and they were able to hop from one edge server to another or stay on their current one in every iteration. At least one service was deployed to each edge server, while the total number of services was set to 100. The latency between a user and the edge server they were connected to, was randomly set from 6 to 20 ms (including the default delay δ, processing time T_{proc}, and transmission delay T_{trans}). The factor that was alternating at each iteration was T_{proc} which was defined as formulated in Equation 7.8:

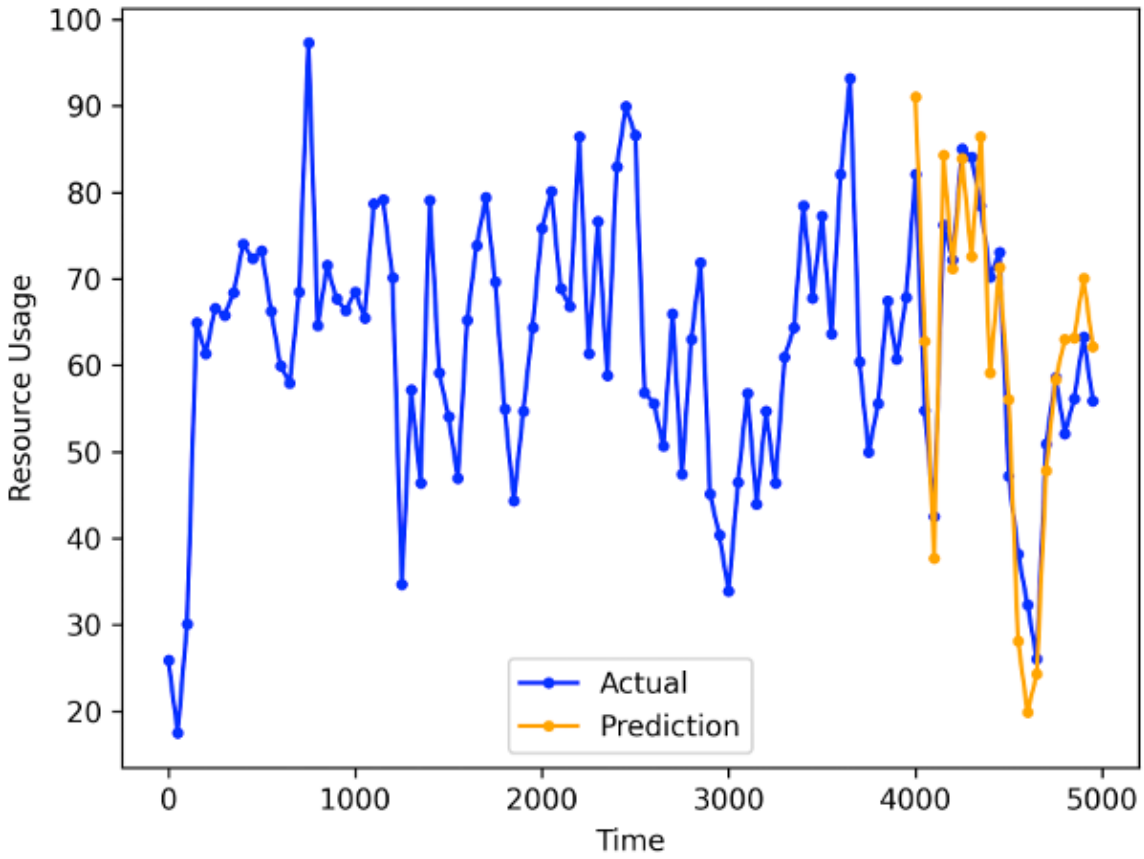


Figure 7.9: CPU Utilization and prediction, using the enhanced dataset.

$$T_{proc} = t_{proc} \times (1 + cpu\_sage) \times (0.5 \times mem\_usage) \tag{7.8}$$

where t_{proc} is the default processing time of a given service (set randomly to $3-6$ ms and was fixed), *cpu_usage* percentage of the utilized allocated CPU at a given time, and *mem_usage* percentage of the utilized allocated memory at a given time. For instance, if a service was allocated 25% of CPU and was utilizing 20%, the *cpu_usage* metric was equal to 80%.

It is important to note here that the services were also categorized as CPU- or memory-intensive. This means that for a CPU-intensive application every 10 users, required an additional 5% of CPU usage. Similarly, for memory-intensive applications every 10 users were set to require an additional 10% of memory usage.

In order to assess the proposed resource allocation framework, the average latency of the overall edge network was utilized as a key metric, while in order to evaluate the predictions of the predictions on the CPU and memory usage, given that these can be considered time-series forecasting (regression) tasks, the utilized metrics that were used are Mean Square Error (MSE), MAE, and the R-squared (R^2) score. In the second case, a different dataset derived from a real use case scenario was utilized, as will be further explained below.

Starting with the assessment of the proposed dynamic resource allocation framework, the results indicate that the overall latency of the edge network was significantly reduced when compared to the overall latency of the edge network when the framework was not exploited. As shown in Figures 7.4 and 7.5 which depict the average memory and CPU utilization of the entire edge network accordingly, it can be inferred that the use of the proposed dynamic allocation strategy has resulted in enhanced overall CPU and memory utilization in general. In more detail, on average, the average CPU utilization when utilizing the proposed strategy was equal to $\sim 55.5\%$ and the average memory utilization was equal to $\sim 64.38\%$. On the other hand, the average CPU utilization without the use of the proposed strategy was equal to $\sim 62.5\%$ and the average memory utilization was equal to $\sim 72.48\%$, resulting in a decrease in CPU utilization of approximately 7%, and a decrease in memory utilization of approximately 8.1%.

Figures 7.7 and 7.6, present a comparison of the average latency per service, and the average latency of the entire network is depicted in the latter figure when the not utilizing the proposed framework and when it is used accordingly. The average latency when utilizing the proposed strategy was equal to ~ 16.4 ms, while when the strategy was not used the average latency was equal to ~ 17.64 ms. This resulted in an average decrease of approximately 7%.

For the second part of the evaluation, which was related to the assessment of the resource usage predictions performed by the selected ML model, a dataset com-

posed of actual services incorporating real-time monitoring metrics such as CPU, memory and disk utilization, latency, egress bandwidth, ingress bandwidth, etc., was utilized. This dataset was retrieved from an actual edge network, which is comprised of 5 edge servers (every edge server is a Raspberry Pi [Foundation, 2023] with 8 GB of memory and 4 CPU cores). The monitoring data were collected using Prometheus [Prometheus, 2023]. The dataset had a total of $1,260,431$ instances. 90% of the data was used for training, while the remaining 10% was used for testing.

In order to evaluate the validity of the proposed data enhancement methods, the ML model (*i.e.*, the Random Forest Regressor) was initially trained using the original dataset, without being enhanced using the causal feature generation and the influence-based instances selection methods. Afterwards, the training was performed again, using the enhanced dataset. The results show a significant improvement in the performance of the model which can be also seen in Figures 7.8 and 7.9. Figure 7.8 depicts the predictions of the model using the original dataset, while Figure 7.9, shows the model's predictions using the causally and contextually enhanced dataset. Based on this, it is safe to assume that the model's performance was improved when trained on the enhanced dataset. Another important note is that the dataset size was also reduced, since only the influential instances were finally used (the final enhanced dataset was comprised of $930,198$ instances, *i.e.*, an $\sim 18\%$ decrease in size).

The results on the test set from the original dataset show that initially the MSE was equal to 109.374, while in the case of the enhanced dataset, the MSE was equal to 48.476 instead. This indicates that the model's predictions were closer to the actual values on average by $\sim 55.71\%$ when trained with the enhanced dataset. Adequate conclusions can be drawn when evaluating the MAE. With the original dataset the MAE was equal 8.084, while when using the enhanced dataset it was equal to 6.024, leading to an improvement of $\sim 25.51\%$ on average. Based on this,

it can be understood that the absolute differences between the predicted and the actual values were closer when utilizing the enhanced dataset.

Finally, when looking at the R^2 score, when using the original dataset it was equal to 0.631, and when using the enhanced dataset it was equal to 0.798 which means that the model could better fit on the data of the enhanced dataset. Similar results were also observed for the memory usage prediction. To sum up, this suggests that the utilization of the data enhancement method has significantly improved the performance of the ML model, leading to an increase of $\sim 26.5\%$ for the R^2 score and a decrease of the MAE by $\sim 25\%$.

Chapter 8
Conclusions and Future Work

Chapter Structure

This Chapter is constructed as follows:

- **Section 8.1 - Conclusions**, presents the concluding remarks of this thesis. This Section summarizes the key findings of the research and highlights its significance.
- **Section 8.2 - Future Work**, outlines potential directions for future research that could build upon the findings of the current thesis.

8.1 Conclusions

The main goals of this thesis was manifold. First, examine whether causal and contextual information can be efficiently extracted from any given dataset. The second main goal was to assess whether this information can be used in any way in order to ML models. Thus, this thesis focuses on the development of dataset enhancement methods that can extract such information and incorporate them in the dataset. This led to the last goal of this thesis, which was to evaluate such methods in diverse environments in order to evaluate their significance. These

methods were then exploited in approaches used for the optimization of Cloud and Edge Computing infrastructures.

In order to better structure this research, four questions were raised in which, this thesis aimed at answering. The first question, **RQ1**, raised the issue of whether causal and contextual information can be derived from data and if yes, how to do that. Trying to answer to that, in this thesis two information extraction methodologies were introduced. The first, initially described in Chapter 2, regards an Influence-based Dataset Generation method which aims at generating an Influence-based dataset by identifying the most influential instances in one. The second method described in Chapter 5 regards a Causal Features Generation method. This method first discovers causal relationships in a dataset and afterwards utilizes this information to generate causal features in order to incorporate this information to the initial dataset.

The proposed methodologies also addressed **RQ2** which wonders if the extracted causal and contextual information can be somehow utilized in order to enrich the data. Furthermore, even if the data are in enhanced, does this affect the performance of an ML model in specific tasks related to managing diverse environments such as a Cloud Computing Infrastructure or an Edge network. In order to do answer to these points, and to the points raised in **RQ3** which asks in which ML tasks these methodologies can be utilized, this thesis evaluated the proposed methodologies in various diverse scenarios and assessed their performance. The use cases were related to various ML tasks from classification, to clustering and time-series analysis.

Finally, **RQ4** doubts if the enrichment of the data with causal and contextual information actually improves in any way the performance of a ML model. To answer that, the performance of the ML models used in each scenario was evaluated when the proposed methodologies were utilized and when not and the results were compared, proving that in most cases the performance increase was substantial.

8.2 Future Work

This thesis proposed two methodologies used for the extraction of contextual and causal information for the enhancement of data towards the improvement of ML models used in various scenarios related to efficient Cloud and Edge Computing infrastructures management. Although evaluated in several distinct scenarios to justify their relevance, the proposed methodologies have some limitations which can be the starting point for future research towards more causal-aware AI.

To start with, the Influence-based Dataset Generation method manages to efficiently identify contextual information relative to the environment given the provided information, it has been tested with datasets that are of relatively modest sizes. Yet, it has been acknowledged to take significant time to operate thus making it difficult to scale when it comes to big datasets. A solution to that issue may be an adaptation of the existing method which first performs clustering on the instances and identifies influential clusters instead of individual instances.

In addition, the proposed methodologies are applied to scenarios with a controlled environment. In the future it would be interesting to utilize them in observational studies, meaning studies where there exist no manipulation of the existing variables or control over the experimental conditions. This is also a great chance to see how the proposed methodologies manage to perform in cases where it is most certain that uncontrolled confounders may exist.

Bibliography

[Abdi et al., 2013] Abdi, H., Williams, L. J., and Valentin, D. (2013). Multiple factor analysis: principal component analysis for multitable and multiblock data sets. *Wiley Interdisciplinary reviews: computational statistics*, 5(2):149–179.

[Adadi and Berrada, 2018] Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160.

[Afonso, 2018] Afonso, N. (2018). Mechanisms for providing causal consistency on edge computing. *Master's thesis*.

[Agarwal et al., 2010] Agarwal, S., Dunagan, J., Jain, N., Saroiu, S., Wolman, A., and Bhogan, H. (2010). Volley: Automated data placement for geo-distributed cloud services. In *NSDI*.

[Aggarwal et al., 2022] Aggarwal, K., Mijwil, M. M., Al-Mistarehi, A.-H., Alomari, S., Gök, M., Alaabdin, A. M. Z., Abdulrhman, S. H., et al. (2022). Has the future started? the current growth of artificial intelligence, machine learning, and deep learning. *Iraqi Journal for Computer Science and Mathematics*, 3(1):115–123.

[Al Azad et al., 2021] Al Azad, M. W., Shannigrahi, S., Stergiou, N., Ortega, F. R., and Mastorakis, S. (2021). Cledge: A hybrid cloud-edge computing framework over information centric networking. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pages 589–596. IEEE.

[Al Qassem et al., 2023] Al Qassem, L. M., Stouraitis, T., Damiani, E., and Elfadel, I. A. M. (2023). Proactive random-forest autoscaler for microservice resource

allocation. *IEEE Access*, 11:2570–2585.

[Apache, 2023a] Apache (2023a). Apache hadoop.

[Apache, 2023b] Apache (2023b). Apache hbase.

[Athey and Imbens, 2006] Athey, S. and Imbens, G. W. (2006). Identification and inference in nonlinear difference-in-differences models. *Econometrica*, 74(2):431–497.

[Athey and Imbens, 2015] Athey, S. and Imbens, G. W. (2015). Machine learning methods for estimating heterogeneous causal effects. *stat*, 1050(5):1–26.

[Athey and Wager, 2019] Athey, S. and Wager, S. (2019). Estimating treatment effects with causal forests: An application. *Observational studies*, 5(2):37–51.

[Austin, 2011] Austin, P. C. (2011). An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424.

[Barbedo, 2018] Barbedo, J. G. A. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and electronics in agriculture*, 153:46–53.

[Bega et al., 2019] Bega, D., Gramaglia, M., Fiore, M., Banchs, A., and Costa-Perez, X. (2019). Deepcog: Cognitive network management in sliced 5g networks with deep learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 280–288. IEEE.

[Bei et al., 2015] Bei, Z., Yu, Z., Zhang, H., Xiong, W., Xu, C., Eeckhout, L., and Feng, S. (2015). Rfhoc: A random-forest approach to auto-tuning hadoop's configuration. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1470–1483.

[Berndt and Clifford, 1994] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pages 359–370.

[Bhatore et al., 2020] Bhatore, S., Mohan, L., and Reddy, Y. R. (2020). Machine learning techniques for credit risk evaluation: a systematic literature review. *Journal of Banking and Financial Technology*, 4:111–138.

[Bhide et al., 2018] Bhide, A., Shah, P. S., and Acharya, G. (2018). A simplified guide to randomized controlled trials. *Acta obstetricia et gynecologica Scandinavica*, 97(4):380–387.

[Brill et al., 2022] Brill, T. M., Munoz, L., and Miller, R. J. (2022). Siri, alexa, and other digital assistants: a study of customer satisfaction with artificial intelligence applications. In *The Role of Smart Technologies in Decision Making*, pages 35–70. Routledge.

[Bühlmann et al., 2010] Bühlmann, P., Kalisch, M., and Maathuis, M. H. (2010). Variable selection in high-dimensional linear models: partially faithful distributions and the pc-simple algorithm. *Biometrika*, 97(2):261–278.

[Cai et al., 2017] Cai, Z., Yan, H., Li, P., Huang, Z.-a., and Gao, C. (2017). Towards secure and flexible ehr sharing in mobile health cloud under static assumptions. *Cluster Computing*, 20:2415–2422.

[Caliendo and Kopeinig, 2008] Caliendo, M. and Kopeinig, S. (2008). Some practical guidance for the implementation of propensity score matching. *Journal of economic surveys*, 22(1):31–72.

[CAMEL, 2023] CAMEL (2023). Camel.

[Cao et al., 2019] Cao, S., Zhang, G., Liu, P., Zhang, X., and Neri, F. (2019). Cloud-assisted secure ehealth systems for tamper-proofing ehr via blockchain. *Information Sciences*, 485:427–440.

[Carbonell et al., 1983] Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). An overview of machine learning. *Machine learning*, pages 3–23.

[Cardellino et al., 2015] Cardellino, C., Villata, S., Alemany, L. A., and Cabrio, E. (2015). Information extraction with active learning: A case study in legal text. In *Computational Linguistics and Intelligent Text Processing: 16th International Conference, CICLing 2015, Cairo, Egypt, April 14-20, 2015, Proceedings, Part II 16*, pages 483–494. Springer.

[Caron et al., 2022] Caron, A., Baio, G., and Manolopoulou, I. (2022). Shrinkage bayesian causal forests for heterogeneous treatment effects estimation. *Journal of Computational and Graphical Statistics*, 31(4):1202–1214.

[causaLens, 2023] causaLens (2023). confounders: machine learning's blindspot.

[Chattopadhyay et al., 2019] Chattopadhyay, A., Manupriya, P., Sarkar, A., and Balasubramanian, V. N. (2019). Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pages 981–990. PMLR.

[Chavhan et al., 2022] Chavhan, S., Gupta, D., Gochhayat, S. P., N, C. B., Khanna, A., Shankar, K., and Rodrigues, J. J. (2022). Edge computing ai-iot integrated energy-efficient intelligent transportation system for smart cities. *ACM Transactions on Internet Technology*, 22(4):1–18.

[Chen et al., 2020] Chen, M., Yuan, J., Liu, D., and Li, T. (2020). An adaption scheduling based on dynamic weighted random forests for load demand forecasting. *The Journal of Supercomputing*, 76:1735–1753.

[Chen, 2020] Chen, Y. (2020). Iot, cloud, big data and ai in interdisciplinary domains.

[Chen et al., 2021] Chen, Y., Zhang, Y., Wu, J., Wang, J., and Xing, C. (2021). Revisiting data prefetching for database systems with machine learning techniques. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2165–2170. IEEE.

[Cheng et al., 2022] Cheng, L., Guo, R., Moraffah, R., Sheth, P., Candan, K. S., and Liu, H. (2022). Evaluation methods and measures for causal learning algorithms. *IEEE Transactions on Artificial Intelligence*, 3(6):924–943.

[Chervenak et al., 2007] Chervenak, A., Deelman, E., Livny, M., Su, M.-H., Schuler, R., Bharathi, S., Mehta, G., and Vahi, K. (2007). Data placement for scientific applications in distributed environments. In *2007 8th IEEE/ACM International Conference on Grid Computing*, pages 267–274. IEEE.

[Chhabra and Singh, 2022] Chhabra, S. and Singh, A. K. (2022). Dynamic resource allocation method for load balance scheduling over cloud data center networks. *arXiv preprint arXiv:2211.02352*.

[Chiappa and Isaac, 2019] Chiappa, S. and Isaac, W. S. (2019). A causal bayesian networks viewpoint on fairness. *Privacy and Identity Management. Fairness, Accountability, and Transparency in the Age of Big Data: 13th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2. 2 International Summer School, Vienna, Austria, August 20-24, 2018, Revised Selected Papers 13*, pages 3–20.

[Chickering, 2002] Chickering, D. M. (2002). Learning equivalence classes of bayesian-network structures. *The Journal of Machine Learning Research*, 2:445–498.

[Chowdhury et al., 2013] Chowdhury, C. R., Chatterjee, A., Sardar, A., Agarwal, S., and Nath, A. (2013). A comprehensive study on cloud green computing: To reduce carbon footprints using clouds. *International Journal of Advanced Computer Research*, 3(8):78–85.

[Colombo et al., 2014] Colombo, D., Maathuis, M. H., et al. (2014). Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782.

[Cook, 1977] Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18.

[De and Singh, 2016] De, P. and Singh, S. (2016). Modified random forest approach for resource allocation in 5g network. *International Journal of Advanced Computer Science and Applications*, 7(11).

[Dehghani and Tumer, 2015] Dehghani, M. and Tumer, M. (2015). A research on effectiveness of facebook advertising on enhancing purchase intention of consumers. *Computers in human behavior*, 49:597–600.

[Deng et al., 2020a] Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., and Zomaya, A. Y. (2020a). Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469.

[Deng et al., 2020b] Deng, X., Li, J., Shi, L., Wei, Z., Zhou, X., and Yuan, J. (2020b). Wireless powered mobile edge computing: Dynamic resource allocation and throughput maximization. *IEEE Transactions on Mobile Computing*, 21(6):2271–2288.

[Desai et al., 2022] Desai, P. R., Mini, S., and Tosh, D. K. (2022). Edge-based optimal routing in sdn-enabled industrial internet of things. *IEEE Internet of Things Journal*.

[Digital, 2023] Digital, C. (2023). Estonian central health information system and patient portal.

[Dimopoulou et al., 2022] Dimopoulou, S., Symvoulidis, C., Koutsoukos, K., Kiourtis, A., Mavrogiorgou, A., and Kyriazis, D. (2022). Mobile anonymization and pseudonymization of structured health data for research. In *2022 Seventh International Conference On Mobile And Secure Services (MobiSecServ)*, pages 1–6. IEEE.

[Ding and Lu, 2017] Ding, P. and Lu, J. (2017). Principal stratification analysis using principal scores. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(3):757–777.

[Do et al., 2015] Do, T. M. T., Dousse, O., Miettinen, M., and Gatica-Perez, D. (2015). A probabilistic kernel method for human mobility prediction with smartphones. *Pervasive and Mobile Computing*, 20:13–28.

[Docker, 2023a] Docker (2023a). Enterprise application container platform — docker.

[Docker, 2023b] Docker (2023b). Overview of docker compose.

[Du et al., 2020] Du, X., Tang, S., Lu, Z., Wet, J., Gai, K., and Hung, P. C. (2020). A novel data placement strategy for data-sharing scientific workflows in heterogeneous edge-cloud computing environments. In *2020 IEEE International Conference on Web Services (ICWS)*, pages 498–507. IEEE.

[Elwert, 2013] Elwert, F. (2013). Graphical causal models. In *Handbook of causal analysis for social research*, pages 245–273. Springer.

[Eom et al., 2013] Eom, H., Juste, P. S., Figueiredo, R., Tickoo, O., Illikkal, R., and Iyer, R. (2013). Machine learning-based runtime scheduler for mobile offloading framework. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 17–25. IEEE.

[Esteves et al., 2020] Esteves, S., Silva, J. N., and Veiga, L. (2020). Palpatine: mining frequent sequences for data prefetching in nosql distributed key-value stores. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE.

[Faisal et al., 2019] Faisal, A., Kamruzzaman, M., Yigitcanlar, T., and Currie, G. (2019). Understanding autonomous vehicles. *Journal of transport and land use*, 12(1):45–72.

[Falcon, 2006] Falcon, A. (2006). Aristotle on causality. *Stanford Encyclopedia of Philosophy*.

[Foundation, 2023] Foundation, R. P. (2023). Raspberry pi.

[Fradkov, 2020] Fradkov, A. L. (2020). Early history of machine learning. *IFAC-PapersOnLine*, 53(2):1385–1390.

[Frangakis and Rubin, 2002] Frangakis, C. E. and Rubin, D. B. (2002). Principal stratification in causal inference. *Biometrics*, 58(1):21–29.

[Fu et al., 2019] Fu, E. L., Groenwold, R. H., Zoccali, C., Jager, K. J., van Diepen, M., and Dekker, F. W. (2019). Merits and caveats of propensity scores to adjust for confounding. *Nephrology Dialysis Transplantation*, 34(10):1629–1635.

[Fu and Desmarais, 2010] Fu, S. and Desmarais, M. C. (2010). Markov blanket based feature selection: a review of past decade. In *Proceedings of the world congress on engineering*, volume 1, pages 321–328. Newswood Ltd. Hong Kong, China.

[Gallop et al., 2009] Gallop, R., Small, D. S., Lin, J. Y., Elliott, M. R., Joffe, M., and Ten Have, T. R. (2009). Mediation analysis with principal stratification. *Statistics in medicine*, 28(7):1108–1130.

[Gao et al., 2022] Gao, X., Liu, R., Kaushik, A., and Zhang, H. (2022). Dynamic resource allocation for virtual network function placement in satellite edge clouds. *IEEE Transactions on Network Science and Engineering*, 9(4):2252–2265.

[Geiger et al., 2016] Geiger, P., Carata, L., and Schölkopf, B. (2016). Causal inference for data-driven debugging and decision making in cloud computing. *arXiv preprint arXiv:1603.01581*.

[Gers et al., 2001] Gers, F. A., Eck, D., and Schmidhuber, J. (2001). Applying lstm to time series predictable through time-window approaches. In *International conference on artificial neural networks*, pages 669–676. Springer.

[Ghosh and Liu, 2009] Ghosh, J. and Liu, A. (2009). K-means. *The top ten algorithms in data mining*, pages 21–35.

[Gill et al., 2022] Gill, S. S., Xu, M., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A., Golec, M., Stankovski, V., Wu, H., Abraham, A., et al. (2022). Ai for next generation computing: Emerging trends and future directions. *Internet of Things*, 19:100514.

[GitHub, 2023a] GitHub (2023a). Github.

[GitHub, 2023b] GitHub (2023b). Github rest api documentation.

[Goli-Malekabadi et al., 2016] Goli-Malekabadi, Z., Sargolzaei-Javan, M., and Akbari, M. K. (2016). An effective model for store and retrieve big health data in cloud computing. *Computer methods and programs in biomedicine*, 132:75–82.

[Gong et al., 2023] Gong, C., He, W., Wang, T., Gani, A., and Qi, H. (2023). Dynamic resource allocation scheme for mobile edge computing. *The Journal of Supercomputing*, pages 1–21.

[Goodman-Bacon, 2021] Goodman-Bacon, A. (2021). Difference-in-differences with variation in treatment timing. *Journal of Econometrics*, 225(2):254–277.

[Google, 2023] Google (2023). Tensorflow.

[Greenland, 2000] Greenland, S. (2000). An introduction to instrumental variables for epidemiologists. *International journal of epidemiology*, 29(4):722–729.

[Guo et al., 2020] Guo, R., Cheng, L., Li, J., Hahn, P. R., and Liu, H. (2020). A survey of learning causality with data: Problems and methods. *ACM Computing Surveys (CSUR)*, 53(4):1–37.

[Gupta and Gupta, 2017] Gupta, S. and Gupta, A. (2017). A set of measures designed to identify overlapped instances in software defect prediction. *Computing*, 99:889–914.

[Guyon et al., 2007] Guyon, I., Aliferis, C., et al. (2007). Causal feature selection. In *Computational methods of feature selection*, pages 79–102. Chapman and Hall/CRC.

[Hauser and Bühlmann, 2012] Hauser, A. and Bühlmann, P. (2012). Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 13(1):2409–2464.

[He et al., 2019] He, S., Tang, Y., Li, Z., Li, F., Xie, K., Kim, H.-j., and Kim, G.-j. (2019). Interference-aware routing for difficult wireless sensor network environment with swipt. *Sensors*, 19(18):3978.

[He et al., 2021] He, Z., Li, K., Li, K., and Zhou, W. (2021). Server configuration optimization in mobile edge computing: a cost-performance tradeoff perspective. *Software: Practice and Experience*, 51(9):1868–1895.

[Health, 2023] Health, G. (2023). Google health.

[Ho and Xie, 1998] Ho, S. L. and Xie, M. (1998). The use of arima models for reliability forecasting and analysis. *Computers & industrial engineering*, 35(1-2):213–216.

[Hochreiter et al., 2001] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[Hoggart et al., 2003] Hoggart, C. J., Parra, E. J., Shriver, M. D., Bonilla, C., Kittles, R. A., Clayton, D. G., and McKeigue, P. M. (2003). Control of confounding of genetic associations in stratified populations. *The American Journal of Human Genetics*, 72(6):1492–1504.

[Howards et al., 2012] Howards, P. P., Schisterman, E. F., Poole, C., Kaufman, J. S., and Weinberg, C. R. (2012). "toward a clearer definition of confounding" revisited with directed acyclic graphs. *American journal of epidemiology*, 176(6):506–511.

[Huang et al., 2016] Huang, X., Ye, Y., Xiong, L., Lau, R. Y., Jiang, N., and Wang, S. (2016). Time series k-means: A new k-means type smooth subspace clustering for time series data. *Information Sciences*, 367:1–13.

[Hussien and Sulaiman, 2017] Hussien, N. and Sulaiman, S. (2017). Web prefetching schemes using machine learning for mobile cloud computing. *Int. J. Adv. Soft Comput. Appl*, 9:154–187.

[IBM, 2023] IBM (2023). Watson health citizen engagement.

[Imagine, 2023] Imagine (2023). Ai art generator.

[Jager et al., 2008] Jager, K., Zoccali, C., Macleod, A., and Dekker, F. (2008). Confounding: what it is and how to deal with it. *Kidney international*, 73(3):256–260.

[Jin et al., 2011] Jin, Y., Deyu, T., and Yi, Z. (2011). A distributed storage model for ehr based on hbase. In *2011 international conference on information management, innovation management and industrial engineering*, volume 2, pages 369–372. IEEE.

[Joshi et al., 2018] Joshi, M., Joshi, K., and Finin, T. (2018). Attribute based encryption for secure access to cloud based ehr systems. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 932–935. IEEE.

[Jung et al., 2020] Jung, Y. M., Whang, J. J., and Yun, S. (2020). Sparse probabilistic k-means. *Applied Mathematics and Computation*, 382:125328.

[Kadir and Gleeson, 2018] Kadir, T. and Gleeson, F. (2018). Lung cancer prediction using machine learning and advanced imaging techniques. *Translational lung cancer research*, 7(3):304.

[Kahlert et al., 2017] Kahlert, J., Gribsholt, S. B., Gammelager, H., Dekkers, O. M., and Luta, G. (2017). Control of confounding in the analysis phase–an overview for clinicians. *Clinical epidemiology*, pages 195–204.

[Karim et al., 2017] Karim, F., Majumdar, S., Darabi, H., and Chen, S. (2017). Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669.

[Kaur et al., 2021] Kaur, I., Lydia, E. L., Nassa, V. K., Shrestha, B., Nebhen, J., Malebary, S., and Joshi, G. P. (2021). Generative adversarial networks with quantum optimization model for mobile edge computing in iot big data. *Wireless Personal Communications*, pages 1–21.

[Khan et al., 2020] Khan, L. U., Yaqoob, I., Tran, N. H., Kazmi, S. A., Dang, T. N., and Hong, C. S. (2020). Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet of Things Journal*, 7(10):10200–10232.

[Khan et al., 2019] Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., and Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235.

[Kiourtis et al., 2023] Kiourtis, A., Giannetsos, T., Menesidou, S.-A., Mavrogiorgou, A., Symvoulidis, C., Graziani, A., Kleftakis, S., Mavrogiorgos, K., Zafeiropoulos, N., Gkolias, C.-A., and Kyriazis, D. (2023). Identity management standards: A literature review. *Computers and Informatics*, 3(1):35–46.

[Kiourtis et al., 2021a] Kiourtis, A., Graziani, A., Mavrogiorgou, A., Symvoulidis, C., Mavrogiorgos, K., and Kyriazis, D. (2021a). A health information exchange protocol supporting bluetooth-based messages. In *2021 International Conference on Information Systems and Advanced Technologies (ICISAT)*, pages 1–6. IEEE.

[Kiourtis et al., 2022] Kiourtis, A., Mavrogiorgou, A., Mavrogiorgos, K., Kyriazis, D., Graziani, A., Symvoulidis, C., Bella, G., Bocca, S., and Torelli, F. (2022). Electronic health records at people's hands across europe: The interopehrate protocols. In *pHealth 2022*, pages 145–150. IOS Press.

[Kiourtis et al., 2021b] Kiourtis, A., Mavrogiorgou, A., Symvoulidis, C., Tsigkounis, C., and Kyriazis, D. (2021b). Indexing of cloud stored electronic health records

for consented third party accessing. In *2021 28th Conference of Open Innovations Association (FRUCT)*, pages 158–166. IEEE.

[Kissell, 2020] Kissell, R. (2020). *Algorithmic trading methods: Applications using advanced statistics, optimization, and machine learning techniques*. Academic Press.

[Koh and Liang, 2017] Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

[Kourou et al., 2015] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17.

[Koutsoukos et al., 2022] Koutsoukos, K., Symvoulidis, C., Kiourtis, A., Mavrogiorgou, A., Dimopoulou, S., and Kyriazis, D. (2022). Emergency health protocols supporting health data exchange, cloud storage, and indexing. In *HEALTHINF*, pages 597–604.

[Kumar T et al., 2021] Kumar T, S., Mustapha, S. D. S., Gupta, P., and Tripathi, R. P. (2021). Hybrid approach for resource allocation in cloud infrastructure using random forest and genetic algorithm. *Scientific Programming*, 2021:1–10.

[Kwon et al., 2018] Kwon, B. C., Choi, M.-J., Kim, J. T., Choi, E., Kim, Y. B., Kwon, S., Sun, J., and Choo, J. (2018). Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE transactions on visualization and computer graphics*, 25(1):299–309.

[Law et al., 2012] Law, G. R., Green, R., and Ellison, G. T. (2012). Confounding and causal path diagrams. In *Modern methods for epidemiology*, pages 1–13. Springer.

[Lazic et al., 2022] Lazic, I., Agullo, F., Ausso, S., Alves, B., Barelle, C., Berral, J. L., Bizopoulos, P., Bunduc, O., Chouvarda, I., Dominguez, D., Filos, D., Gutierrez-Torre, A., Hesso, I., Jakovljevic, N., Kayyali, R., Kogut-Czarkowska, M., Kosvyra, A., Lalas, A., Lavdaniti, M., Lonca-Turukalo, T., Martinez-Alabart, S., Michas, N., Nabhani-Gebara, S., Raptopoulos, A., Roussakis, Y., Stalika, E., Symvoulidis, C., Tsave, O., Votis, K., and Charalambous, A. (2022). The holistic perspective of the incisive project—artificial intelligence in screening mammography. *Applied Sciences*, 12(17):8755.

[Lee et al., 1971] Lee, H. D. P. et al. (1971). *Timaeus and Critias*. Penguin.

[Leng et al., 2022] Leng, J., Chen, Z., Sha, W., Ye, S., Liu, Q., and Chen, X. (2022). Cloud-edge orchestration-based bi-level autonomous process control for mass individualization of rapid printed circuit boards prototyping services. *Journal of Manufacturing Systems*, 63:143–161.

[Li et al., 2016a] Li, B., Zhang, H., and Lu, H. (2016a). User mobility prediction based on lagrange's interpolation in ultra-dense networks. In *2016 IEEE 27th annual international symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE.

[Li et al., 2016b] Li, J., Ma, S., Le, T., Liu, L., and Liu, J. (2016b). Causal decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 29(2):257–271.

[Liang et al., 2021] Liang, Y., Hu, Z., Zhang, X., and Xiao, H. (2021). Correlation-aware replica prefetching strategy to decrease access latency in edge cloud. *China Communications*, 18(9):249–264.

[Lim et al., 2021] Lim, W. Y. B., Ng, J. S., Xiong, Z., Jin, J., Zhang, Y., Niyato, D., Leung, C., and Miao, C. (2021). Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(3):536–550.

[Lin et al., 2019] Lin, B., Zhu, F., Zhang, J., Chen, J., Chen, X., Xiong, N. N., and Mauri, J. L. (2019). A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. *IEEE Transactions on Industrial Informatics*, 15(7):4254–4265.

[Lin et al., 2020] Lin, J., Yu, W., Yang, X., Zhao, P., Zhang, H., and Zhao, W. (2020). An edge computing based public vehicle system for smart transportation. *IEEE Transactions on Vehicular Technology*, 69(11):12635–12651.

[Ling et al., 2019] Ling, Z., Yu, K., Wang, H., Liu, L., Ding, W., and Wu, X. (2019). Bamb: A balanced markov blanket discovery approach to feature selection. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(5):1–25.

[Liu et al., 2019] Liu, Y., Yang, C., Jiang, L., Xie, S., and Zhang, Y. (2019). Intelligent edge computing for iot-based energy management in smart cities. *IEEE network*, 33(2):111–117.

[Luque-Fernandez et al., 2018] Luque-Fernandez, M. A., Schomaker, M., Rachet, B., and Schnitzer, M. E. (2018). Targeted maximum likelihood estimation for a binary treatment: A tutorial. *Statistics in medicine*, 37(16):2530–2546.

[Lv et al., 2022] Lv, F., Liang, J., Li, S., Zang, B., Liu, C. H., Wang, Z., and Liu, D. (2022). Causality inspired representation learning for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8046–8056.

[Lv et al., 2021] Lv, Z., Chen, D., Lou, R., and Wang, Q. (2021). Intelligent edge computing based on machine learning for smart city. *Future Generation Computer Systems*, 115:90–99.

[Mahmood and Cornaniciu, 2005] Mahmood, H. and Cornaniciu, C. (2005). Interference aware routing for cdma wireless ad hoc networks. In *MILCOM 2005-2005 IEEE Military Communications Conference*, pages 1059–1063. IEEE.

[Manias et al., 2023] Manias, G., Mavrogiorgou, A., Kiourtis, A., Symvoulidis, C., and Kyriazis, D. (2023). Multilingual text categorization and sentiment analysis: a comparative analysis of the utilization of multilingual approaches for classifying twitter data. *Neural Computing and Applications*, pages 1–17.

[Manoj et al., 2017] Manoj, R., Alsadoon, A., Prasad, P. C., Costadopoulos, N., and Ali, S. (2017). Hybrid secure and scalable electronic health record sharing in hybrid cloud. In *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 185–190. IEEE.

[Mansouri, 2016] Mansouri, N. (2016). Adaptive data replication strategy in cloud computing for performance improvement. *Frontiers of Computer Science*, 10:925–935.

[Mansouri and Javidi, 2018] Mansouri, N. and Javidi, M. M. (2018). A new prefetching-aware data replication to decrease access latency in cloud environment. *Journal of Systems and Software*, 144:197–215.

[Mao et al., 2017] Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358.

[Marinos et al., 2021] Marinos, G., Symvoulidis, C., and Kyriazis, D. (2021). Micsurv: medical image clustering for survival risk group identification. In *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART)*, pages 1–4. IEEE.

[Marinos et al., 2022] Marinos, G., Symvoulidis, C., and Kyriazis, D. (2022). K-medoids-surv: A patients risk stratification algorithm considering censored data. In *International Conference on Artificial Intelligence and Soft Computing*, pages 127–140. Springer.

[Martens et al., 2006] Martens, E. P., Pestman, W. R., de Boer, A., Belitser, S. V., and Klungel, O. H. (2006). Instrumental variables: application and limitations. *Epi-*

demiology, pages 260–267.

[Martinez et al., 2008] Martinez, R., Cebrián, M., de Borja Rodriguez, F., and Camacho, D. (2008). Contextual information retrieval based on algorithmic information theory and statistical outlier detection. In *2008 IEEE Information Theory Workshop*, pages 292–297. IEEE.

[Mavrogiorgou et al., 2023] Mavrogiorgou, A., Kiourtis, A., Manias, G., Symvoulidis, C., and Kyriazis, D. (2023). Batch and streaming data ingestion towards creating holistic health records. *Emerging Science Journal*, 7(2):339–353.

[McCurry, 2021] McCurry, C. (2021). People should have ownership of personal health data, says patients group. *Independent.ie*.

[McHugh, 2012] McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

[McNamee, 2003] McNamee, R. (2003). Confounding and confounders. *Occupational and environmental medicine*, 60(3):227–234.

[Mehta and DeWitt, 1997] Mehta, M. and DeWitt, D. J. (1997). Data placement in shared-nothing parallel database systems. *The VLDB journal*, 6(1):53–72.

[Messerli, 2012] Messerli, F. H. (2012). Chocolate consumption, cognitive function, and nobel laureates. *N Engl J Med*, 367(16):1562–1564.

[Mi et al., 2020] Mi, J.-X., Li, A.-D., and Zhou, L.-F. (2020). Review study of interpretation methods for future interpretable machine learning. *IEEE Access*, 8:191969–191985.

[Microsoft, 2023] Microsoft (2023). Cloud for healthcare.

[MinIO, 2023a] MinIO (2023a). Minio - high performance, kubernetes native object storage.

[MinIO, 2023b] MinIO (2023b). Object management - minio object storage.

[Mittelstadt, 2021] Mittelstadt, B. (2021). Interpretability and transparency in artificial intelligence. *The Oxford Handbook of Digital Ethics (online edn, Oxford Academic, 10 Nov. 2021), https://doi. org/10.1093/oxfordhb/9780198857815.013*, 20.

[Molnar, 2020] Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

[MongoDB, 2023] MongoDB (2023). Mongodb - the developer data platform.

[Moraffah et al., 2020] Moraffah, R., Karami, M., Guo, R., Raglin, A., and Liu, H. (2020). Causal interpretability for machine learning-problems, methods and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(1):18–33.

[Müller, 2007] Müller, M. (2007). Dynamic time warping. *Information retrieval for music and motion*, pages 69–84.

[Ning et al., 2020] Ning, Z., Dong, P., Wang, X., Wang, S., Hu, X., Guo, S., Qiu, T., Hu, B., and Kwok, R. Y. (2020). Distributed and dynamic service placement in pervasive edge computing networks. *IEEE Transactions on Parallel and Distributed Systems*, 32(6):1277–1292.

[Nogueira et al., 2020] Nogueira, A. R., Gama, J., and Ferreira, C. A. (2020). Improving prediction with causal probabilistic variables. In *International Symposium on Intelligent Data Analysis*, pages 379–390. Springer.

[Numpy, 2023] Numpy (2023). Numpy.

[O'Dea, 2020] O'Dea, S. (2020). Global iot connections data volume 2019 and 2025.

[Oracle, 2023] Oracle (2023). Mysql.

[Ouyang et al., 2018] Ouyang, T., Zhou, Z., and Chen, X. (2018). Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE Journal on Selected Areas in Communications*, 36(10):2333–2345.

[Pandas, 2023] Pandas (2023). Python data analysis library — pandas: Python data analysis library.

[Pandey and Subbiah, 2016] Pandey, M. K. and Subbiah, K. (2016). A novel storage architecture for facilitating efficient analytics of health informatics big data in cloud. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 578–585. IEEE.

[Pang et al., 2016] Pang, M., Schuster, T., Filion, K. B., Eberg, M., and Platt, R. W. (2016). Targeted maximum likelihood estimation for pharmacoepidemiologic research. *Epidemiology (Cambridge, Mass.)*, 27(4):570.

[Pazzani and Billsus, 2007] Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer.

[Pearl, 1985] Pearl, J. (1985). Bayesian netwcrks: A model cf self-activated memory for evidential reasoning. In *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, pages 15–17.

[Phorasim and Yu, 2017] Phorasim, P. and Yu, L. (2017). Movies recommendation system using collaborative filtering and k-means. *International Journal of Advanced Computer Research*, 7(29):52.

[Plachy et al., 2016a] Plachy, J., Becvar, Z., and Strinati, E. C. (2016a). Dynamic resource allocation exploiting mobility prediction in mobile edge computing. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE.

[Plachy et al., 2016b] Plachy, J., Becvar, Z., and Strinati, E. C. (2016b). Dynamic resource allocation exploiting mobility prediction in mobile edge computing. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE.

[Pramukantoro et al., 2019] Pramukantoro, E. S., Bakhtiar, F. A., and Bhawiyuga, A. (2019). A semantic restful api for heterogeneous iot data storage. In *2019 IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech)*, pages 263–264. IEEE.

[Prinz, 2020] Prinz, A. L. (2020). Chocolate consumption and noble laureates. *Social Sciences & Humanities Open*, 2(1):100082.

[Projects, 2023] Projects, P. (2023). Flask.

[Prometheus, 2023] Prometheus (2023). Prometheus.

[Pruthi et al., 2020] Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. (2020). Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.

[Qader et al., 2019] Qader, W. A., Ameen, M. M., and Ahmed, B. I. (2019). An overview of bag of words; importance, implementation, applications, and challenges. In *2019 international engineering conference (IEC)*, pages 200–204. IEEE.

[Raoof et al., 2020] Raoof, S. S., Jabbar, M. A., and Fathima, S. A. (2020). Lung cancer prediction using machine learning: A comprehensive approach. In *2020 2nd International conference on innovative mechanisms for industry applications (ICIMIA)*, pages 108–115. IEEE.

[Reynolds et al., 2009] Reynolds, D. A. et al. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663).

[Rohrer, 2018] Rohrer, J. M. (2018). Thinking clearly about correlations and causation: Graphical causal models for observational data. *Advances in methods and*

practices in psychological science, 1(1):27–42.

[Rubin, 2010] Rubin, D. (2010). Causal inference. In *International Encyclopedia of Education (Third Edition)*, pages 66–71. Elsevier, Oxford, third edition edition.

[Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1985). Learning internal representations by error propagation.

[Salesforce, 2023] Salesforce (2023). Health cloud.

[Saraswathi et al., 2015] Saraswathi, A., Kalaashri, Y. R., and Padmavathi, S. (2015). Dynamic resource allocation scheme in cloud computing. *Procedia Computer Science*, 47:30–36.

[Schuler and Rose, 2017] Schuler, M. S. and Rose, S. (2017). Targeted maximum likelihood estimation for causal inference in observational studies. *American journal of epidemiology*, 185(1):65–73.

[Senin, 2008] Senin, P. (2008). Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40.

[Seol et al., 2018] Seol, K., Kim, Y.-G., Lee, E., Seo, Y.-D., and Baik, D.-K. (2018). Privacy-preserving attribute-based access control model for xml-based electronic health record system. *IEEE Access*, 6:9114–9128.

[Services, 2023] Services, A. W. (2023). Aws for health.

[Shakarami et al., 2021] Shakarami, A., Ghobaei-Arani, M., Shahidinejad, A., Masdari, M., and Shakarami, H. (2021). Data replication schemes in cloud computing: a survey. *Cluster Computing*, 24:2545–2579.

[Shao et al., 2019] Shao, Y., Li, C., and Tang, H. (2019). A data replica placement strategy for iot workflows in collaborative edge and cloud environments. *Computer Networks*, 148:46–59.

[Shimizu et al., 2006] Shimizu, S., Hoyer, P. O., Hyvärinen, A., Kerminen, A., and Jordan, M. (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10).

[Shimizu et al., 2011] Shimizu, S., Inazumi, T., Sogawa, Y., Hyvarinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., Bollen, K., and Hoyer, P. (2011). Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research-JMLR*, 12(Apr):1225–1248.

[Singh and Upadhyaya, 2012] Singh, K. and Upadhyaya, S. (2012). Outlier detection: applications and techniques. *International Journal of Computer Science Issues (IJCSI)*, 9(1):307.

[Singh and Chitra, 2021] Singh, T. B. and Chitra, S. (2021). Prefetching of web objects for effective retrieval process through data mining techniques. *Preprint*.

[Siraj and Abbasi, 2022] Siraj, M. and Abbasi, Z. (2022). Impact of load balancing interference routing metric on multi channel multi radio wireless mesh network. In *ITM Web of Conferences*, volume 42. EDP Sciences.

[Smith, 2007] Smith, G. (2007). *Newton's philosophiae naturalis principia mathematica*. Library of the University of Michigan.

[Sniezynski et al., 2019] Sniezynski, B., Nawrocki, P., Wilk, M., Jarzab, M., and Zielinski, K. (2019). Vm reservation plan adaptation using machine learning in cloud computing. *Journal of Grid Computing*, 17(4):797–812.

[Søgaard et al., 2021] Søgaard, A. et al. (2021). Revisiting methods for finding influential examples. *arXiv preprint arXiv:2111.04683*.

[Sonin et al., 2021] Sonin, J., Becker, A., and Nipp, K. (2021). It's time for individuals—not doctors or companies—to own their health data. *STAT*, 12.

[Spirtes, 2001] Spirtes, P. (2001). An anytime algorithm for causal inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 278–285.

PMLR.

[Spirtes and Glymour, 1991] Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72.

[Spirtes et al., 2000] Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.

[Stanley, 2007] Stanley, K. (2007). Design of randomized controlled trials. *Circulation*, 115(9):1164–1169.

[Stoica et al., 2017] Stoica, I., Song, D., Popa, R. A., Patterson, D., Mahoney, M. W., Katz, R., Joseph, A. D., Jordan, M., Hellerstein, J. M., Gonzalez, J. E., et al. (2017). A berkeley view of systems challenges for ai. *arXiv preprint arXiv:1712.05855*.

[Stoimenova et al., 2006] Stoimenova, E., Mateev, P., and Dobreva, M. (2006). Outlier detection as a method for knowledge extraction from digital resources. *Rev. Nat. Center Digitization*, 9:1–11.

[Streeter et al., 2017] Streeter, A. J., Lin, N. X., Crathorne, L., Haasova, M., Hyde, C., Melzer, D., and Henley, W. E. (2017). Adjusting for unmeasured confounding in nonrandomized longitudinal studies: a methodological review. *Journal of clinical epidemiology*, 87:23–34.

[Stürmer et al., 2014] Stürmer, T., Wyss, R., Glynn, R. J., and Brookhart, M. A. (2014). Propensity scores for confounder adjustment when assessing the effects of medical interventions using nonexperimental study designs. *Journal of internal medicine*, 275(6):570–580.

[Suk and Kang, 2023] Suk, Y. and Kang, H. (2023). Tuning random forests for causal inference under cluster-level unmeasured confounding. *Multivariate Behavioral Research*, 58(2):408–440.

[Sun et al., 2015] Sun, Y., Li, J., Liu, J., Chow, C., Sun, B., and Wang, R. (2015). Using causal discovery for feature selection in multivariate numerical time series. *Machine Learning*, 101:377–395.

[Symvoulidis et al., 2023a] Symvoulidis, C., Kiourtis, A., Marinos, G., Tom-Ata, J.-D. T., Manias, G., Mavrogiorgou, A., and Kyriazis, D. (2023a). A user mobility-based data placement strategy in a hybrid cloud/edge environment using a causal-aware deep learning network. *IEEE Transactions on Computers*.

[Symvoulidis et al., 2021a] Symvoulidis, C., Kiourtis, A., Mavrogiorgou, A., and Kyriazis, D. (2021a). Healthcare provision in the cloud: An ehr object store-based cloud used for emergency. *Healthinf*, 1:435–442.

[Symvoulidis et al., 2023b] Symvoulidis, C., Kiourtis, A., Mavrogiorgou, A., Tom-Ata, J.-D. T., Manias, G., and Kyriazis, D. (2023b). Dynamic deployment prediction and configuration in hybrid cloud / edge computing environments using influence-based learning. In *2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 315–320.

[Symvoulidis et al., 2022] Symvoulidis, C., Marinos, G., Kiourtis, A., Mavrogiorgou, A., and Kyriazis, D. (2022). Healthfetch: An influence-based, context-aware prefetch scheme in citizen-centered health storage clouds. *Future Internet*, 14(4):112.

[Symvoulidis et al., 2021b] Symvoulidis, C., Mavrogiorgou, A., Kiourtis, A., Marinos, G., and Kyriazis, D. (2021b). Facilitating health information exchange in medical emergencies. In *2021 International Conference on e-Health and Bioengineering (EHB)*, pages 1–4. IEEE.

[Symvoulidis et al., 2024] Symvoulidis, C., Paraskevoulakou, E., Kiourtis, A., Mavrogiorgou, A., and Kyriazis, D. (in press 2024). Dynamic resource allocation at the edge: A causal contextually-aware machine learning approach. In *2024 10th Intelligent Systems Conference (IntelliSys 2024)*. Springer.

[Symvoulidis et al., 2019] Symvoulidis, C., Tsoumas, I., and Kyriazis, D. (2019). Towards the identification of context in 5g infrastructures. In *Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 2*, pages 406–418. Springer.

[Tang et al., 2019a] Tang, H., Li, C., Bai, J., Tang, J., and Luo, Y. (2019a). Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud–edge environment. *Computer Communications*, 134:70–82.

[Tang et al., 2019b] Tang, H., Li, C., Bai, J., Tang, J., and Luo, Y. (2019b). Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud–edge environment. *Computer Communications*, 134:70–82.

[Thomas, 2020] Thomas, L. (2020). Confounding variables — definition, examples and controls.

[Tokdar and Kass, 2010] Tokdar, S. T. and Kass, R. E. (2010). Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60.

[Totow Tom-At et al., 2024] Totow Tom-At, J.-D., Kritikos, K., Di Girolamo, M. A., Paraskevoulakou, E., Symvoulidis, C., and Kyriazis, D. (2024). Polymorphic cloud application design assisted by open source software classification. Accepted for publication in the proceedings of the 2024 5th IEEE International Communication Engineering and Cloud Computing Conference, in press.

[Tripepi et al., 2010] Tripepi, G., Jager, K. J., Dekker, F. W., and Zoccali, C. (2010). Stratification for confounding–part 1: The mantel-haenszel formula. *Nephron Clinical Practice*, 116(4):c317–c321.

[Tsoumas et al., 2021] Tsoumas, I., Symvoulidis, C., and Kyriazis, D. (2021). Learning a generalized matrix from multi-graphs topologies towards microservices recommendations. In *Intelligent Systems and Applications: Proceedings of the 2020 Intelligent Systems Conference (IntelliSys) Volume 2*, pages 693–702. Springer.

[Tsoumas et al., 2020] Tsoumas, I., Symvoulidis, C., Kyriazis, D., Gouvas, P., Zafeiropoulos, A., Melian, J., and Sterle, J. (2020). Modelling 5g cloud-native applications by exploiting the service mesh paradigm. In *Information Systems: 16th European, Mediterranean, and Middle Eastern Conference, EMCIS 2019, Dubai, United Arab Emirates, December 9–10, 2019, Proceedings 16*, pages 151–162. Springer.

[Turing, 1992] Turing, A. (1992). Turing's intelligent machinery. *Collected Works of AM Turing: Mechanical Intelligence. Amsterdam: Elsevier Science Publishers.*

[Van Huyssteen, 2003] Van Huyssteen, J. W. V. (2003). *Encyclopedia of science and religion.* Arba Minch University.

[Varghese et al., 2016] Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., and Nikolopoulos, D. S. (2016). Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 20–26. IEEE.

[Waharte et al., 2008] Waharte, S., Ishibashi, B., Boutaba, R., and Meddour, D. (2008). Interference-aware routing metric for improved load balancing in wireless mesh networks. In *2008 IEEE International Conference on Communications*, pages 2979–2983. IEEE.

[Wang et al., 2020] Wang, L., Jiao, L., He, T., Li, J., and Bal, H. (2020). Service placement for collaborative edge applications. *IEEE/ACM Transactions on Networking*, 29(1):34–47.

[Wang et al., 2021a] Wang, Q., Shwartz, L., Grabarnik, G. Y., Arya, V., and Shanmugam, K. (2021a). Detecting causal structure on cloud application microservices using granger causality models. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 558–565. IEEE.

[Wang et al., 2021b] Wang, T., Zhang, Y., Xiong, N. N., Wan, S., Shen, S., and Huang, S. (2021b). An effective edge-intelligent service placement technology

for 5g-and-beyond industrial iot. *IEEE Transactions on Industrial Informatics*, 18(6):4148–4157.

[Wang et al., 2018] Wang, Z., Zhao, Z., Min, G., Huang, X., Ni, Q., and Wang, R. (2018). User mobility aware task assignment for mobile edge computing. *Future Generation Computer Systems*, 85:1–8.

[Waqas et al., 2022] Waqas, A., Mahmood, H., and Saeed, N. (2022). Interference aware cooperative routing for edge computing-enabled 5g networks. *IEEE Sensors Journal*.

[Warneke and Kao, 2011] Warneke, D. and Kao, O. (2011). Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *IEEE transactions on parallel and distributed systems*, 22(6):985–997.

[Webby and O'Connor, 1996] Webby, R. and O'Connor, M. (1996). Judgemental and statistical time series forecasting: a review of the literature. *International Journal of forecasting*, 12(1):91–118.

[Wei et al., 2020] Wei, H., Luo, H., and Sun, Y. (2020). Mobility-aware service caching in mobile edge computing for internet of things. *Sensors*, 20(3):610.

[Wei and Wang, 2021] Wei, X. and Wang, Y. (2021). Popularity-based data placement with load balancing in edge computing. *IEEE Transactions on Cloud Computing*.

[Wen et al., 2021] Wen, B., Colon, L. O., Subbalakshmi, K., and Chandramouli, R. (2021). Causal-tgan: Generating tabular data using causal generative adversarial networks. *arXiv preprint arXiv:2104.10680*.

[Wiering et al., 2002] Wiering, M. A. et al. (2002). Evolving causal neural networks. In *Benelearn'02: Proceedings of the Twelfth Belgian-Dutch Conference on Machine Learning*, pages 103–108.

[Wischmeyer, 2020] Wischmeyer, T. (2020). Artificial intelligence and transparency: opening the black box. *Regulating artificial intelligence*, pages 75–101.

[Wojnowicz et al., 2016] Wojnowicz, M., Cruz, B., Zhao, X., Wallace, B., Wolff, M., Luan, J., and Crable, C. (2016). "influence sketching": Finding influential samples in large-scale regressions. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3601–3612. IEEE.

[Wold et al., 1987] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

[Wu et al., 2019] Wu, H., Deng, S., Li, W., Yin, J., Li, X., Feng, Z., and Zomaya, A. Y. (2019). Mobility-aware service selection in mobile edge computing systems. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 201–208. IEEE.

[Wu and Pottenger, 2005] Wu, T. and Pottenger, W. M. (2005). A semi-supervised active learning algorithm for information extraction from textual data. *Journal of the American Society for Information Science and Technology*, 56(3):258–271.

[Wu, 2020] Wu, Y. (2020). Cloud-edge orchestration for the internet of things: Architecture and ai-powered data processing. *IEEE Internet of Things Journal*, 8(16):12792–12805.

[Wu et al., 2023] Wu, Y., Cai, C., Bi, X., Xia, J., Gao, C., Tang, Y., and Lai, S. (2023). Intelligent resource allocation scheme for cloud-edge-end framework aided multi-source data stream. *EURASIP Journal on Advances in Signal Processing*, 2023(1):1–20.

[Xiao et al., 2012a] Xiao, Z., Song, W., and Chen, Q. (2012a). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE transactions on parallel and distributed systems*, 24(6):1107–1117.

[Xiao et al., 2012b] Xiao, Z., Song, W., and Chen, Q. (2012b). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE transactions on parallel and distributed systems*, 24(6):1107–1117.

[Xie et al., 2019a] Xie, J., Qian, C., Guo, D., Li, X., Shi, S., and Chen, H. (2019a). Efficient data placement and retrieval services in edge computing. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1029–1039. IEEE.

[Xie et al., 2019b] Xie, J., Qian, C., Guo, D., Wang, M., Shi, S., and Chen, H. (2019b). Efficient indexing mechanism for unstructured data sharing systems in edge computing. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 820–828. IEEE.

[Xie et al., 2021] Xie, J., Qian, C., Guo, D., Wang, M., Wang, G., and Chen, H. (2021). Coin: An efficient indexing mechanism for unstructured data sharing systems. *IEEE/ACM Transactions on Networking*.

[Xiong et al., 2020] Xiong, X., Zheng, K., Lei, L., and Hou, L. (2020). Resource allocation based on deep reinforcement learning in iot edge computing. *IEEE Journal on Selected Areas in Communications*, 38(6):1133–1146.

[Xu et al., 2020a] Xu, G., Duong, T. D., Li, Q., Liu, S., and Wang, X. (2020a). Causality learning: A new perspective for interpretable machine learning. *arXiv preprint arXiv:2006.16789*.

[Xu et al., 2022] Xu, W., Chen, K., Mou, L., and Zhao, T. (2022). Document-level relation extraction with sentences importance estimation and focusing. *arXiv preprint arXiv:2204.12679*.

[Xu et al., 2020b] Xu, Z., Wang, S., Liu, S., Dai, H., Xia, Q., Liang, W., and Wu, G. (2020b). Learning for exception: Dynamic service caching in 5g-enabled mecs with bursty user demands. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 1079–1089. IEEE.

[Yang et al., 2019] Yang, X., Fei, Z., Zheng, J., Zhang, N., and Anpalagan, A. (2019). Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing. *IEEE Transactions on Vehicular Technology*, 68(11):11018–11030.

[Yao et al., 2021] Yao, L., Chu, Z., Li, S., Li, Y., Gao, J., and Zhang, A. (2021). A survey on causal inference. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5):1–46.

[Yarabarla et al., 2019] Yarabarla, M. S., Ravi, L. K., and Sivasangari, A. (2019). Breast cancer prediction via machine learning. In *2019 3rd international conference on trends in electronics and informatics (ICOEI)*, pages 121–124. IEEE.

[Ye et al., 2011] Ye, F., Li, Q., and Chen, E. (2011). Benefit based cache data placement and update for mobile peer to peer networks. *World Wide Web*, 14(3):243–259.

[Yeh and Yu, 2022] Yeh, T. and Yu, S. (2022). Realizing dynamic resource orchestration on cloud systems in the cloud-to-edge continuum. *Journal of Parallel and Distributed Computing*, 160:100–109.

[Yousafzai et al., 2017] Yousafzai, A., Gani, A., Noor, R. M., Sookhak, M., Talebian, H., Shiraz, M., and Khan, M. K. (2017). Cloud resource allocation schemes: review, taxonomy, and opportunities. *Knowledge and information systems*, 50:347–381.

[Yu et al., 2016] Yu, K., Wu, X., Ding, W., Mu, Y., and Wang, H. (2016). Markov blanket feature selection using representative sets. *IEEE transactions on neural networks and learning systems*, 28(11):2775–2788.

[Yuan et al., 2010] Yuan, D., Yang, Y., Liu, X., and Chen, J. (2010). A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 26(8):1200–1214.

[Zeldow and Hatfield, 2021] Zeldow, B. and Hatfield, L. A. (2021). Confounding and regression adjustment in difference-in-differences studies. *Health services research*, 56(5):932–941.

[Zhang et al., 2020] Zhang, D.-G., Chen, L., Zhang, J., Chen, J., Zhang, T., Tang, Y.-M., and Qiu, J.-N. (2020). A multi-path routing protocol based on link lifetime and energy consumption prediction for mobile edge computing. *IEEE Access*, 8:69058–69071.

[Zhang and Bareinboim, 2018] Zhang, J. and Bareinboim, E. (2018). Fairness in decision-making—the causal explanation formula. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Zhang and Cao, 2023] Zhang, J. and Cao, M. (2023). Distant supervision for relation extraction with hierarchical attention-based networks. *Expert Systems with Applications*, 220:119727.

[Zhang et al., 2017] Zhang, J., Hu, X., Ning, Z., Ngai, E. C.-H., Zhou, L., Wei, J., Cheng, J., and Hu, B. (2017). Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet of Things Journal*, 5(4):2633–2645.

[Zhang and Man, 1998] Zhang, J. and Man, K.-F. (1998). Time series prediction using rnn in multi-dimension embedding phase space. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 2, pages 1868–1873. IEEE.

[Zhang et al., 2021] Zhang, Y., Li, M., Wang, S., Dai, S., Luo, L., Zhu, E., Xu, H., Zhu, X., Yao, C., and Zhou, H. (2021). Gaussian mixture model clustering with incomplete data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(1s):1–14.

[Zheng et al., 2017] Zheng, G., Brantley, S. L., Lauvaux, T., and Li, Z. (2017). Contextual spatial outlier detection with metric learning. In *Proceedings of the 23rd*

ACM SIGKDD international conference on knowledge discovery and data mining, pages 2161–2170.

[Zhou et al., 2021] Zhou, X., Ke, R., Yang, H., and Liu, C. (2021). When intelligent transportation systems sensing meets edge computing: Vision and challenges. *Applied Sciences*, 11(20):9680.