Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Communications and Data science

Machine learning-based anomaly detection and root cause analysis in mobile networks

Miro-Markus Nikula

Master's Thesis
2024

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Miro-Markus Nikula

Title of thesis Machine learning-based anomaly detection and root cause analysis in mobile networks

Programme Communications and Data science

Thesis supervisor Prof. Luís Manuel de Jesus Sousa Correia

Thesis advisor(s) Prof. António Manuel Raminhos Cordeiro Grilo and Prof. Petri Mähönen

Collaborative partner NOS (Portugal)

Date 7.1.2024 **Number of pages** 82 **Language** English

Abstract

The evolution of mobile networks and the arrival of 5G technology have significantly increased network size and complexity leading to challenges in network management. This thesis addresses machine learning-based anomaly detection and root cause analysis in mobile networks while providing information of various related topics, such as 4G and 5G networks, quality of experience, machine learning methods and self-organising networks. The work focuses on the limitations of typical mobile network data, mainly the lack of labels and the need for expert knowledge and develops an unsupervised machine learning-based model that can automatically detect network anomalies and perform root cause analysis on unlabelled mobile network data.

The proposed model combines DBSCAN and LSTM AE, and is trained and tested using real world 4G network data from a Portuguese telecom operator, NOS. The study shows promise in DBSCAN's ability to separate normal network traffic patterns from abnormal, and the ability of LSTM AE to learn the daily network KPI behaviour and to detect anomalies based on their reconstruction errors. The reconstruction errors also provide insight into the individual KPIs mostly contributing to the anomalies, thus the anomalies' root causes.

The research and findings in this thesis highlight the importance of self-healing in self-organising networks and how different machine learning models can be used to perform anomaly detection and root cause analysis. The obtained results show that anomalies in the available network KPIs do not always result in abnormal traffic patterns and vice versa. Consequently, it can be derived that DBSCAN based solely on traffic volumes is not an ideal method to separate normal network data from abnormal, with the goal of finding network anomalies. In addition, the results underscore the importance of high-quality data in terms of sampling rate and the number of KPIs, as well as the importance of data analysis in finding patterns on different levels of mobile networks.

Keywords Mobile network, Anomaly detection, Root cause analysis, Machine learning, DBSCAN, LSTM AE, SON, QoE

Aalto-yliopisto
Aalto-universitetet
Aalto University

Tekijä	Miro-Markus Nikula
Työn nimi	Koneoppimiseen pohjautuva poikkeamantunnistus ja juurisyyanalyysi mobiiliverkoissa
Koulutusohjelma	Communications and Data science
Työn valvoja	Prof. Luís Manuel de Jesus Sousa Correia
Työn ohjaajat	Prof. António Manuel Raminhos Cordeiro Grilo ja Prof. Petri Mähönen
Yhteistyötaho	NOS (Portugali)
Päivämäärä 7.1.2024	**Sivumäärä** 82 **Kieli** Englanti

Tiivistelmä

Mobiiliverkkojen sekä erityisesti 5G teknologian kehittyminen ovat merkittävästi kasvattaneet sekä monimutkaistaneet verkkoja, mikä on johtanut haasteisiin niiden hallinnassa. Tämä diplomityö käsittelee koneoppimiseen pohjautuvia menetelmiä poikkeamantunnistamiseen ja juurisyyanalyysiin mobiiliverkoissa sekä tarjoaa tietoa erilaisista aihepiireistä kuten 4G- ja 5G-verkoista, kokemuksen laadusta, koneoppimismenetelmistä ja itseorganisoituvista verkoista. Työ keskittyy tyypillisen mobiiliverkkodatan rajoituksiin, lähinnä syöte-tulos parien puutteeseen sekä asiantuntijatiedon tarpeeseen ja kehittää ohjaamattomaan koneoppimiseen perustuvan mallin, joka pystyy automaattisesti havaitsemaan mobiiliverkon poikkeamia ja suorittamaan niille juurisyyanalyysin datasta, joka ei sisällä syöte-tulos pareja.

Ehdotettu malli yhdistää DBSCAN- ja LSTM AE-teknologiat, ja se on koulutettu sekä testattu käyttäen portugalilaisen teleoperaattori NOS:in 4G-verkosta kerättyä dataa. Tutkimus osoittaa lupaavia tuloksia DBSCAN:in kyvystä erotella normaali tietoliikenne epänormaalista ja LSTM AE:n kyvystä oppia mobiiliverkkojen tunnuslukujen päivittäinen käyttäytyminen ja tunnistaa poikkeamia niiden rekonstruktiovirheiden perusteella. Rekonstruktiovirheet antavat myös käsityksen yksittäisten tunnuslukujen vaikutuksesta poikkeamien juurisyihin.

Tämän työn tutkimus ja tulokset korostavat itsekorjautuvuuden merkitystä itseorganisoituvissa verkoissa ja selventävät, kuinka erilaisia koneoppimismalleja voidaan käyttää poikkeamantunnistamiseen sekä juurisyyanalyysiin. Saadut tulokset osoittavat, että poikkeamat saatavilla olevissa verkon tunnusluvuissa eivät aina johda epänormaaleihin liikennemalleihin ja päinvastoin. Näin ollen voidaan päätellä, että pelkästään liikennemääriin perustuva DBSCAN ei ole ihanteellinen tapa erotella normaalia tietoliikennettä epänormaalista verkon poikkeavuuksien löytämiseksi. Lisäksi tulokset korostavat korkealaatuisen datan merkitystä, erityisesti koskien näytetaajuutta ja tunnuslukujen määrää, sekä data-analyysin merkitystä yhtäläisyyksien löytämisessä mobiiliverkkojen eri tasoilla.

Avainsanat Mobiiliverkko, Poikkeamantunnistaminen, Juurisyyanalyysi, Koneoppiminen, DBSCAN, LSTM AE, Itseorganisoituvat verkot, Kokemuksen laatu

Acknowledgements

I would like to express my gratitude to my supervisor Professor Luís M. Correia and my advisor Professor António Grilo for their unwavering support for me, the countless hours they have put into reviewing and advising this work, as well as our delightful conversations about Finnish and Portuguese cultures.

I would also like to thank NOS for providing the data used in this thesis and Ricardo Dinis, Hugo Martins and Luís Santo for their professional insights from NOS.

I would like to thank my advisor from Aalto university, Professor Petri Mähönen, who provided important insight and feedback of my thesis.

I would like to thank Aalto University and University of Lisbon for enabling the Communications and Data Science study programme that allowed me to go on an unforgettable journey to reach new heights professionally and as a person.

Special thanks to the programme's planning officer Ella Lankinen, without whose help navigating through the programme and the challenges placed by studying abroad I could not have overcome.

I am grateful for all the people I met and the friends I made during my time in Lisbon and with who I experienced and learned so much.

Finally, thanks to my family, my girlfriend and my friends in Finland, without who I would not be where I am today. Thank you for always being there when I have needed, for being role models and for supporting me and my decisions.

Vapaala, Vantaa, Finland, January 7th, 2024

Miro-Markus Nikula

Table of Contents

List of Symbols

σ_X	standard deviation of a variable X
μ	mean or subcarrier spacing (μ = [0, 1, 2, 3, 4])
F	feature space
P	probability
r	Pearson correlation coefficient
R^D	input space
$x(n)$	original sequence
$\hat{x}(n)$	reconstructed sequence
y_i	the true ith value of y
$\widehat{y_i}$	the predicted ith value of y
z	standard score (z-score)

List of Abbreviations

1G	first generation
2G	second generation
3G	third generation
4G	fourth generation
5G	fifth generation
Adam	Adaptive Moment Estimation
AE	Autoencoder
AF	Application Function
AMF	Access and Mobility Management Function
AUSF	Authentication Server Function
BS	Base Station
CART	Classification and Regression Tree
CDR	Call Detail Records
CNN	Convolutional Neural Network
C-SON	Centralised SON
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DN	Data Network
DNN	Deep Neural Network
D-SON	Distributed SON
eMBB	Enhanced Mobile Broadband
eNodeB	Evolved Node B
EPC	Evolved Packet Core
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
FDD	Frequency Division Duplexing
GAN	Generative Adversarial Network
GBR	Guaranteed Bit Rate
GCN	Graph Convolution Neural
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
HSS	Home Subscriber Server

IoT	Internet of Things
KPI	Key Performance Indicator
KQI	Key Quality Indicator
LSTM	Long Short-Term Memory
LTE	Long-Term Evolution
MEC	Multi-Access Edge Computing
MIMO	Multiple-Input Multiple-Output
MLP	Multilayer Perceptron
MME	Mobility Management Entity
mMTC	Massive Machine-Type Communications
MSE	Mean Squared Error
MSL	Mars Science Laboratory Rover
NEF	Network Exposure Function
NF	Network Function
NFV	Network Functions Virtualisation
NR	New Radio
NRF	Network Repository Function
NSSF	Network Slice Selection Function
OFDM	Orthogonal Frequency-Division Multiplexing
PCA	Principal Component Analysis
PCF	Policy Control Function
PCRF	Policy and Charging Rules Function
PDN	Packet Data Network
PDU	Protocol Data Unit
PGW	Packet Data Network Gateway
QCI	Quality Channel Indicator
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RCA	Root Cause Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network

SDN	Software-Defined Networking
SGD	Stochastic Gradient Descent
SGW	Serving Gateway
SMF	Session Management Function
SMS	Short Message Service
SOM	Self-Organising Map
SON	Self-Organising Network
SVM	Support Vector Machine
SWaT	Secure Water Treatment
TDD	Time Division Duplexing
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-reliable Low Latency Communications
VAE	Variational Autoencoder
VoIP	Voice over Internet Protocol
WADI	Water Distribution

1 Introduction

Each new generation of mobile networks has introduced numerous new technologies and enhancements to the previous ones. The first generation (1G) mobile networks introduced the first wireless cellular technology marking the transition from traditional landlines to voice only mobile networks. The second generation (2G) networks brought upon the transition from analogue to digital networks and introduced the Short Message Service (SMS). The third generation (3G) networks started to widely utilise packet switching, leading to higher data rates. The fourth generation (4G) networks enabled significant increases in data rates and introduced Voice over Internet Protocol (VoIP) service.

With the advent of the fifth generation (5G) mobile networks, the telecommunications industry is undergoing another major transformation. New mobile networks are characterised by their high-speed data transfer, low latency and reliable communications as well as their ability to simultaneously provide services to a vast number of devices and different applications. 5G networks will achieve these advancements through a myriad of revolutionary technologies and solutions. While the innovations in 5G networks will drastically reshape the landscape of mobile communications, these improvements will also increase the complexity of networks, making network management increasingly challenging.

Self-organising networks (SON) are designed to reduce the need for manual network management allowing for more effective and robust automatic management of cellular networks sometimes even in real-time [1]. One of the categories of SONs is called self-healing, to which network anomaly detection and root cause analysis (RCA) are integral parts of. Anomaly detection involves identifying unusual behaviour of the network and RCA is used to identify the underlying causes of the anomalies. There is a variety of different types of anomalies, varying from subtle, barely noticeable abnormal behaviours to anomalies severe enough to cause public safety hazards. Such is the case for instance with network-wide service outages, resulting in disconnected emergency calls. Consequently, effectively detecting anomalies and repairing their sources can result in better user experience and reduction of expenses for the network operator [1].

The development of anomaly detection and RCA methods is closely related to that of machine learning techniques, especially deep neural networks (DNN). Numerous automatic machine learning-based anomaly detectors in the context of 4G mobile networks have been developed throughout recent years. Most of them use simulation data or labelled anomaly datasets. However, a major problem with labelled mobile network datasets is their unavailability. Vast amounts of network data are continuously generated, but labelling it is an extremely laborious process often requiring the expertise of network engineers. The lack of labelled data not only hinders anomaly

detection but also poses significant challenges in RCA within mobile networks, which is a relatively scarcely studied subject. Moreover, existing solutions often depend on the insights from network experts during the system's setup phase.

The primary objective of this work is to design an unsupervised machine learning-based model with the ability to automatically detect anomalies and to conduct RCA in unlabelled mobile network data, without requiring network expert input during the implementation stage. Specifically, the anomalies of interest are the ones resulting from network errors, and not for example from natural human behaviour. The model is developed through a process of experimenting with different models and leveraging data analysis, particularly in identifying patterns and correlations within the data. Specifically, the final model consists of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and an autoencoder (AE) incorporating Long Short-Term Memory (LSTM) layers. The model is designed to be used by network operators to enhance network performance and reliability. This work is centred around a dataset containing unlabelled real-world 4G network data provided by a Portuguese telecom operator, NOS. This work aims to answer the following questions:

- How can anomaly detection and RCA be performed in the context of mobile networks?
- How can they be performed using only unlabelled data?
- How will the deployment of 5G networks change the landscape of anomaly detection and RCA in mobile networks?

The work is organised as follows: Chapter 2 provides fundamental information about essential themes of this work, such as mobile networks (especially 5G) and machine learning methods. It also presents the concepts of anomaly detection and RCA along with an overview of the existing research about them. In Chapter 3, a description and an analysis of the dataset used for this research are given, and the methodologies chosen for anomaly detection and RCA are outlined, coupled with the thought process behind their selection. The results and the means to verify them are provided in Chapter 4 along with a critical discussion of the results as well as alternative solutions. Finally, the work is concluded and possibilities for future work are considered in Chapter 5.

2 Fundamental aspects

This chapter gives an overview of the fundamental aspects of the thesis. The topics covered in this chapter are mobile networks, their services and applications, quality of experience, machine learning concepts, self-organising networks as well as anomaly detection and root cause analysis and the work related to them.

2.1 Mobile network overview

It is estimated that around 85% of the world's population have access to a 4G network, while 5G networks are projected to reach the same number already by the year 2028 [2]. 5G is an evolution of 4G designed to address the needs of an increasing number of devices and new applications in the network. While 4G networks are still being implemented and developed and will remain relevant for years to come, 5G introduces multiple new technologies and implementations. This chapter describes the architectures of both 4G and 5G networks and the central technologies enabling 5G.

2.1.1 4G system architecture

4G LTE (Long-Term Evolution) networks are packet-switched mobile networks composed of two primary components: the E-UTRAN radio access network and the Evolved Packet Core (EPC) core network [3]. The radio access network (RAN) is responsible for all radio related functionalities, such as radio resource handling, while the core network is responsible for other essential functionalities that are not radio access related, but are crucial for the operation of the network, such as user authentication. User equipment (UE) connect to the E-UTRAN, which includes eNodeB-base stations, through radio interfaces. The E-UTRAN is further connected to the EPC, which features various components depicted in Figure 1. The key components and their primary functions within the EPC are outlined below, as described in [3].

The Serving Gateway (SGW) is a gateway for incoming user traffic and is responsible for routing and forwarding user data. eNodeBs feed user plane data to the EPC using the SGW. SGW is connected to the Mobility Management Entity (MME) and it forwards user data to Packet Data Network Gateway (PGW).

The PGW serves as the connection between EPC and Packet Data Network (PDN). It receives user plane data from SGW and forwards it into the PDN.

The MME is the cornerstone of mobility, session, bearer and authentication management in the EPC. It communicates with the Home Subscriber Server (HSS) to acquire information about subscribers.

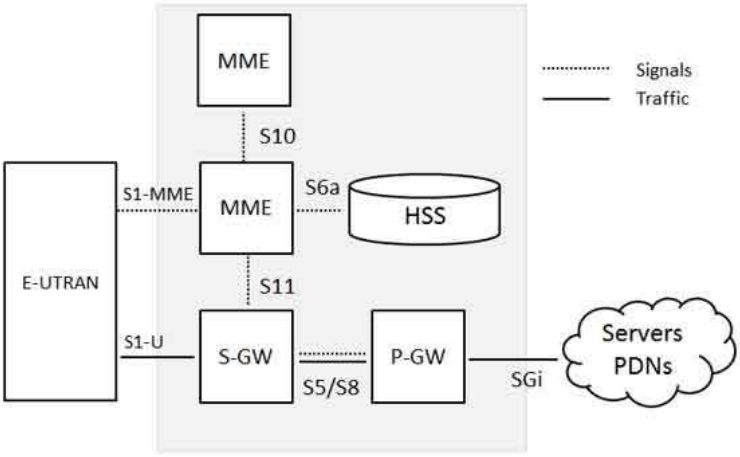


Figure 1: 4G system architecture [4].

The HSS is a database that stores information, such as authorities, of the mobile operator's subscribers.

Lastly, the Policy and Charging Rules Function (PCRF) determines decisions for policy control and flow-based charging control.

2.1.2 Standalone vs. non-standalone 5G

5G networks can be implemented using two distinct architectures: standalone and non-standalone, which are displayed in Figure 2. In non-standalone implementations, 5G is built on top of an existing 4G architecture. This architecture uses both 4G LTE base stations and 5G New Radio (NR) base stations in its RAN. However, the core network infrastructure and control mechanisms of the network are inherited from 4G, in contrast to standalone 5G, where only 5G components are used [5].

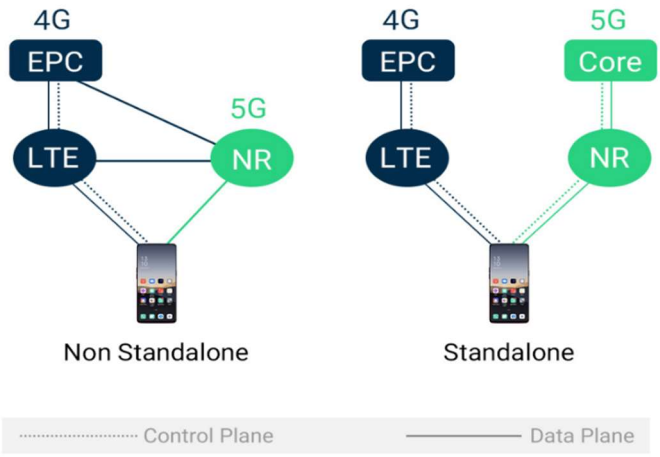


Figure 2: Non-Standalone vs. Standalone 5G [6].

Using the non-standalone implementation allows network operators to quickly roll out networks with some of the new 5G features on the RAN side, such as increased bandwidth. The non-standalone 5G network mainly improves Enhanced Mobile Broadband (eMBB) solutions. However, most of the new features of 5G are not available in the non-standalone implementation. Standalone 5G networks will feature numerous new functions and features to enable Ultra-reliable Low Latency Communications (URLLC) and Massive Machine-Type Communications (mMTC) applications such as autonomous vehicles and next-generation industrial Internet of Things (IoT) [7].

2.1.3 5G system architecture

Much like the 4G architecture, the standalone 5G architecture also consists of two main components: the RAN and the core network. The RAN of 5G is called the New Generation Radio Access Network and it consists of gNodeBs (5G base stations). The data flow is from user equipment through RAN and the User Plane Function (UPF) into the data network or in the opposite direction. Following is the description of 5G core network functions depicted in Figure 3 and their main responsibilities listed by 3GPP [8].

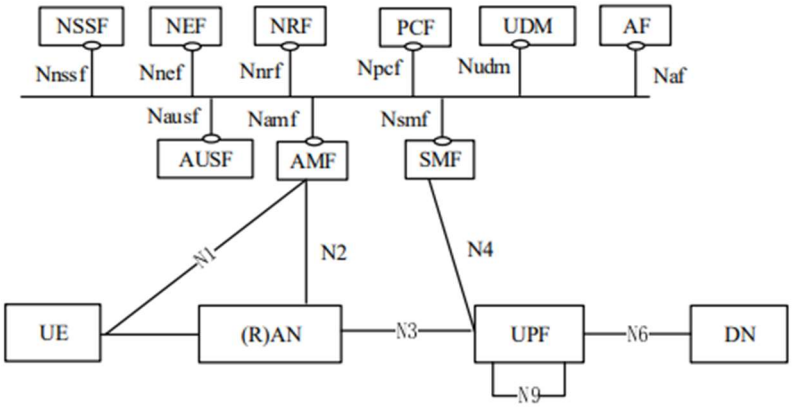


Figure 3: 5G system architecture [8].

The Access and Mobility Management Function (AMF) is responsible for the connection, registration, reachability, and mobility management.

The Session Management Function (SMF) is responsible for Protocol Data Unit (PDU) session management. A PDU session refers to the data connection between User Equipment (UE) and Data Network (DN). In 4G, the MME handles mobility and session management, but in 5G, these two features are separated into the AMF and the SMF.

The UPF works as an anchor point for traffic coming from the RAN. Among other things, it is responsible for packet routing and forwarding. It

also handles quality of service (QoS) of the user plane as well as policy enforcement.

The Unified Data Management (UDM) identifies users and authorises access based on subscription data.

The Policy Control Function (PCF) controls and manages policy rules, like for example QoS enforcement, charging and traffic routing rules.

The Application Function (AF) provides session-related information to the PCF. The PCF can then manage policy rules to match the requirements of that specific session or application.

The Authentication Server Function (AUSF) authenticates UE. When requested by the AMF, the AUSF interacts with the UDM to obtain user authentication which it then forwards to the AMF.

The Network Exposure Function (NEF) allows outside applications to communicate with the 5G network. The NEF can for example allow external requests or providing of information about events related to UE and network analytics.

The Network Repository Function (NRF) stores the functionalities of each Network Function (NF) in the network. NFs can also discover services offered by other NFs present in the network from the NRF.

The Network Slice Selection Function (NSSF) selects a set of available and appropriate network slices to accommodate a service requested by UE and sends it to the AMF.

2.1.4 Network slicing and virtualisation

Softwarisation and virtualisation are the main drivers for application-aware networks in the 5G era [9]. One of the new features of mobile networks presented in 5G is network slicing. Unlike the previous versions with one-size-fits-all type architectures, 5G divides the network into one or more isolated and independently controlled logical networks on a common underlying infrastructure [9]. Different network slices are based on the unique requirements of different services and applications. Network slicing is enabled by technologies such as Software-Defined Networking (SDN), Network Functions Virtualisation (NFV), Multi-Access Edge Computing (MEC) and cloud computing [9]. Below is an explanation of these technologies.

SDN creates a virtualised directly programmable control plane that is separated from the data plane and can enforce intelligent management decisions among network functions [9]. SDN makes networks more programmable and simplifies the management of the network. Consequently, network operators can respond to rapidly changing network conditions and easily manage, configure, and optimise network resources.

NFV means the virtualisation of network functions. This means that most of the NFs that can be seen in Figure 3 are not implemented in a network as actual physical devices, as was the case with previous network

technologies, but as software processes running on commodity off-the-shelf servers [9]. The benefits of NFV are that it makes networks much more flexible and enables more straightforward scaling of services to address changing customer demands. For example, if the NFs were physical devices, it would take a long time to update a physical AMF, but with NFs being virtual and programmable, it is a matter of minutes [9].

SDN and NFV are related topics but whereas SDN can be seen as a paradigm, NFV is an implementation. Different network slices can have distinct sets of NFs running as software processes, which are selected and programmed specifically to accommodate the use case or the service of that particular network slice.

Many use cases depend on low latency and reliable communication. These types of use cases can be accommodated to their own network slices, some of which could further benefit from having computing power closer to the data source, such as through MEC. MEC brings data processing close to where the data is generated. Edge servers are located on the edges of a network and can be embedded in 5G base stations. Processing data locally reduces the need for time-consuming data transfer. This helps to achieve ultra-low latency, higher throughput and reduction of data transfer costs. Low latency and high throughput in turn increase QoS and quality of experience (QoE) and enable applications requiring real-time data transfer [9]. In addition to local data processing, edge servers also enable local data storage and processing as well as security enhancements [10]. The role of MEC in enabling network slicing to accommodate some applications is depicted in Figure 4.

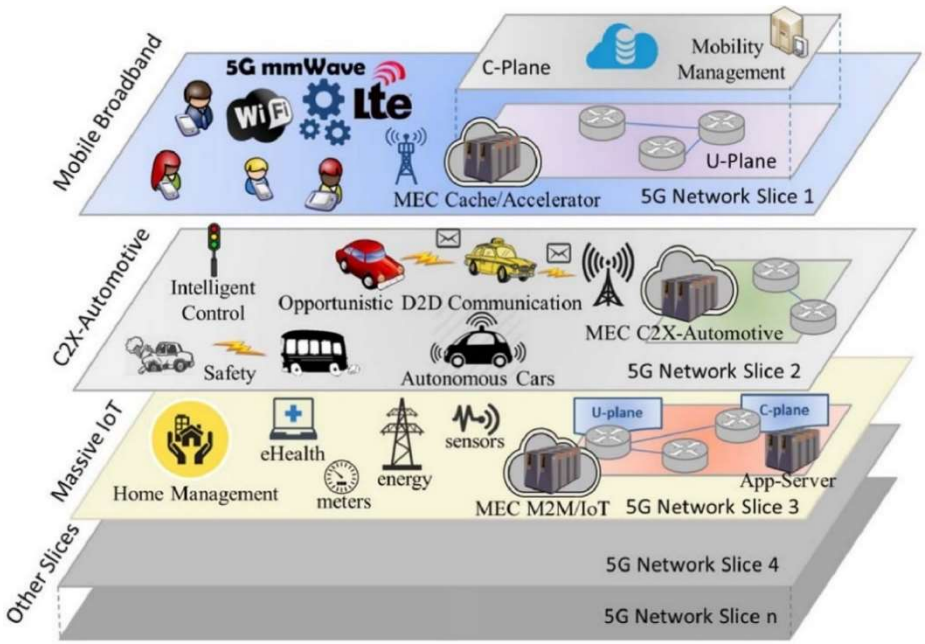


Figure 4: MEC in network slicing [9].

Each slice in Figure 4 corresponds to an individual application and each one communicates with MEC service nodes. Mobile broadband, automotive vehicles and massive IoT are just some of the applications enabled by MEC, which can be assigned to their respective network slices. MEC service nodes can operate locally inside the edge server or remotely in the cloud [9].

Not all computing can be performed on edge servers. Many applications do not depend on real-time data transfer, or they just require larger computational capacity than is available on edge servers. In these scenarios, cloud computing is necessary. Cloud computing refers to computing that is performed in a centralised data centre. Cloud servers can have substantial computational capacity, so heavy computation or permanent data storage can be performed in cloud. Fog computing takes place between cloud and edge. Fog servers offer more computing and storage resources to the edge of a network and therefore support heavier real-time computing than the edge servers [10]. An illustration of cloud, fog and edge computing hierarchy in 5G is depicted in Figure 5.

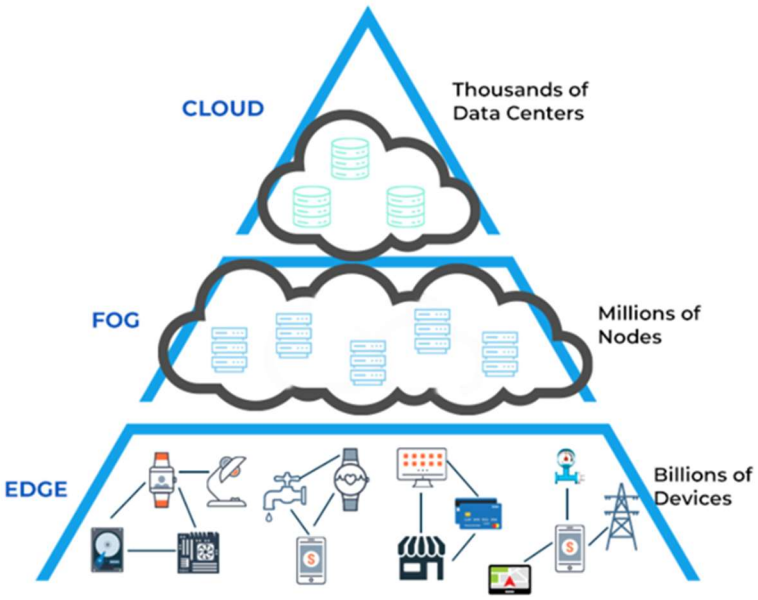


Figure 5: Cloud vs fog vs edge [11].

2.1.5 Radio interface

Radio interface schemes of LTE will remain relevant for NR as well, but with the rapid increase in data traffic volumes, the number of connected wireless devices and QoS requirements, more intelligent and adaptive solutions are required in 5G networks [12]. Many of the new features of NR radio interface aim at more efficient use of energy and frequency resources and for lower latency communication [13].

5G frequency spectrum can be divided into low (< 1 GHz), mid (1–6 GHz), and high bands (24–40 GHz) [14]. The use of low bands is

necessary in rural areas where base stations are further away from network users as low frequency waves experience smaller propagation loss, meaning that they have an increased ability to travel longer distances and pass through obstacles than waves of higher frequencies [15]. For this reason, low bands are also useful in providing deeper indoor coverage. Mid-band spectrum offers the best compromise between capacity and coverage and are seen as ideal for 5G, because mid-band waves can travel significant distances while still carrying large amounts of data [15]. High band spectrum, also known as millimetre wave spectrum, offers substantial bandwidth resulting in extremely high data rates required by some 5G applications. However, millimetre waves can only travel short distances due to higher propagation loss and poor penetration ability. To compensate for these limitations, dense deployment of 5G antennas is required. The frequency bands used may vary by country and region, but as an example, according to the Finnish radio spectrum regulator Traficom [16], the 5G bands currently used by telecom operators in Finland are the 2 GHz, 2.6 GHz, 3.5 GHz and 26 GHz bands.

Duplexing is a technique that allows continuous radio transmission and reception. In general, Frequency Division Duplexing (FDD) is used with lower frequencies while Time Division Duplexing (TDD) is used with higher frequencies [12]. FDD uses different frequencies for uplink and downlink communications while TDD uses the same frequency for uplink and downlink by having dedicated time slots for transmission and reception. In 5G, TDD is preferred over FDD for the mid and high bands due to its spectral efficiency and low-complexity implementation [12].

Unlike LTE, where each subcarrier is spaced 15 kHz apart, NR supports flexible subcarrier spacing of $2^{\mu} \times 15$ kHz, where $\mu = [0, 1, 2, 3, 4]$. Flexible subcarrier spacing allows choosing the spacing based on carrier frequency or deployment scenario. Narrow subcarrier spacing such as 15 kHz or 30 kHz can be used for lower carrier frequencies where cells are larger and the available bandwidth is scarce, whereas wider subcarrier spacing can be used with higher carrier frequencies where phase noise is a problem, as well as with latency-critical services [17].

The NR radio frame has a duration of 10 ms and is divided into 10 subframes. A subframe is divided into 2^{μ} slots, each consisting of 14 Orthogonal Frequency-Division Multiplexing (OFDM) symbols. The length of a slot is determined by the numerology. When using a 15 kHz subcarrier spacing ($\mu = 0$), an NR slot has the same structure as an LTE subframe. This is important for NR devices to work together with LTE devices. NR also introduces mini slot transmission, which allows transmission to start at any OFDM symbol and to last only for as many symbols as is needed for communication. This is an efficient approach to support low-latency communication as well as to minimise interference with other signals [17].

Another important feature of NR is the support of a massive number of steerable antenna elements in base stations. With higher frequency bands,

a technique called beamforming is used to improve spectral efficiency and to minimise interference. In beamforming, multiple antennas are used to direct signals into a specific direction. The more antennas are used, the narrower signal beam can be achieved. With lower frequencies, massive MIMO-antennas (multiple-input multiple-output) are used together with spatial separation to increase capacity and to reduce interference [18].

2.2 Services and applications

The unprecedentedly high data rates, low latency and large capacity of 5G networks open a whole new world of opportunities in comparison to 4G. The numerous 5G applications can be divided into three main categories based on their requirements: URLLC, eMBB and mMTC [19]. Figure 6 illustrates a range of 5G applications and how they are divided into the three application scenarios.

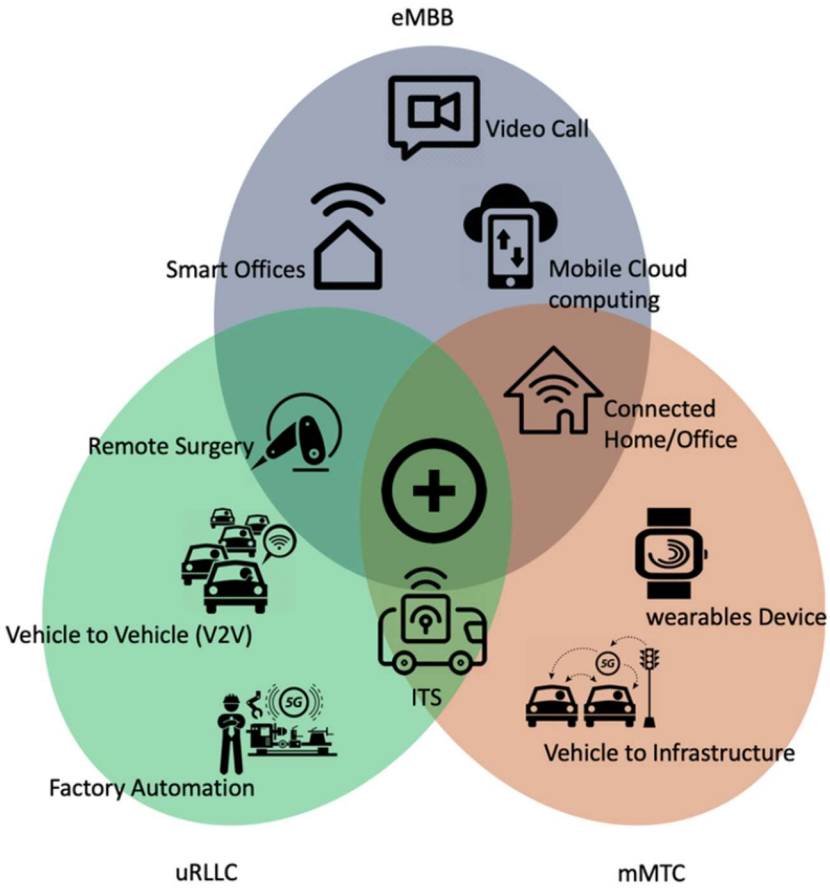


Figure 6: 5G application scenarios and some applications [20].

eMBB can be seen as an extension of the LTE broadband and it is the first stage of implementing 5G networks. eMBB enables higher data rates and

lower latency through increased bandwidth, coverage, spectral efficiency as well as massive MIMO and beamforming. eMBB use cases are for example cloud gaming and smart office. uRLLC provides ultra-reliable services with ultra-low latency to where reliable real-time communication is needed. Vehicle-to-vehicle communication in autonomous driving, as well as automated manufacturing, are examples of uRLLC applications. uRLLC is enabled by new features, such as edge computing and NFV, which can drastically reduce latency. Finally, mMCT enables connections between vast amounts of devices. In mMCT there is normally no need for high data rates or low latency, but, instead, it offers coverage and connectivity to a myriad of devices, and therefore its use cases include different IoT and smart home applications.

5G Services can be divided into guaranteed bitrate (GBR) and non-guaranteed bitrate [21]. GBR is normally used for critical services which require real-time communication, thus guaranteeing a minimum level of network performance for these services to function properly. This differentiation helps network operators to allocate resources whenever they are insufficient. To help separate different services and to prioritise them, different services are associated with Quality Channel Indicators (QCI) and priorities. QCI is used to distinguish different services by their quality of service (QoS) requirements. Lower numbers represent higher QCI classes with QCI 1 being used for reference signalling. Priority on the other hand is used to allocate resources within a communication channel. Unlike QCI, the highest number indicates the highest priority. This type of unified numbering alleviates network management and optimisation.

Resource types (GBR/Non-GBR), QCI classes, priorities as well as the maximum delays and packet error rates of different services are presented in Table 1.

Table 1: Examples of 5G services [21].

Service	Resource type	QCI	Priority	Delay (ms)	Packet error rate
Voice	GBR	1	2	100	10^{-2}
Live stream video	GBR	2	4	150	10^{-3}
Real-time gaming	GBR	3	3	50	10^{-3}
V2X-messaging	GBR	75	2.5	50	10^{-2}
Discrete automation	GBR	82	1.9	10	10^{-4}
Power distribution	GBR	85	2.1	5	10^{-5}
IMS-signalling	Non-GBR	5	1	100	10^{-6}
TCP-based services and buffered video stream	Non-GBR	8	8	300	10^{-6}
Augmented reality	Non-GBR	80	6.8	10	10^{-6}

2.3 Quality of experience

Network quality is often expressed as QoS. QoS takes network's measurable key performance indicators (KPIs) into account, such as throughput, packet loss and handover success rates. QoS is sometimes used interchangeably with QoE in the terminology, but whereas QoS can be seen from the viewpoint of a communication system, QoE refers to the experienced quality of a network user [22]. While KPIs do affect QoE, it also considers other key quality indicators (KQI) such as the expectations and other psychological factors of the user, which allow for a more comprehensive understanding of experienced quality [23].

With the recent evolution of mobile communications networks and the ever-increasing requirements of different network applications, QoE has become of more interest to researchers. Also, mobile network providers are becoming more interested in QoE aware network planning which includes measuring QoE and developing machine learning-based models for its prediction [23]. This type of QoE approach aims to maximise the experienced quality of the network users while maintaining efficient and cost-effective use of network resources.

A survey, conducted in collaboration between Swisscom and Ericsson [24] for 150 network users, shows that for network users the most important criteria of evaluation and satisfaction of the network are webpage download time and video freezing. Low throughput and high delay in the network can also cause user dissatisfaction, but the low QoE resulting from these KPIs is becoming rare. This turns out from an extensive study of the QoE of 292 mobile users in a real-world setting reported in [25]. The reason low throughput and high delay are becoming lesser problems is due to the increasing quality of evolving mobile networks and their ability to provide data to data-heavy applications. Consequently, as the traditional network KPIs' effect on QoE decreases, it becomes increasingly more affected by factors that are more difficult to measure than the typical network parameters.

Both [24] and [25] assume that technically more advanced users have higher expectations for the network and therefore their QoE might be lower than that of less advanced users in the exactly same setting. In [25], machine learning models were created to predict users' mobile QoE and the most important factor was the user's smartphone age, which could be seen as an indicator of technological advancement. In addition to this, the used application also played a large role in the experienced quality. In [23] the factors influencing QoE are divided into four classes illustrated in Figure 7: Human-, System-, Context- and Content-related, with the latter one being for video applications.

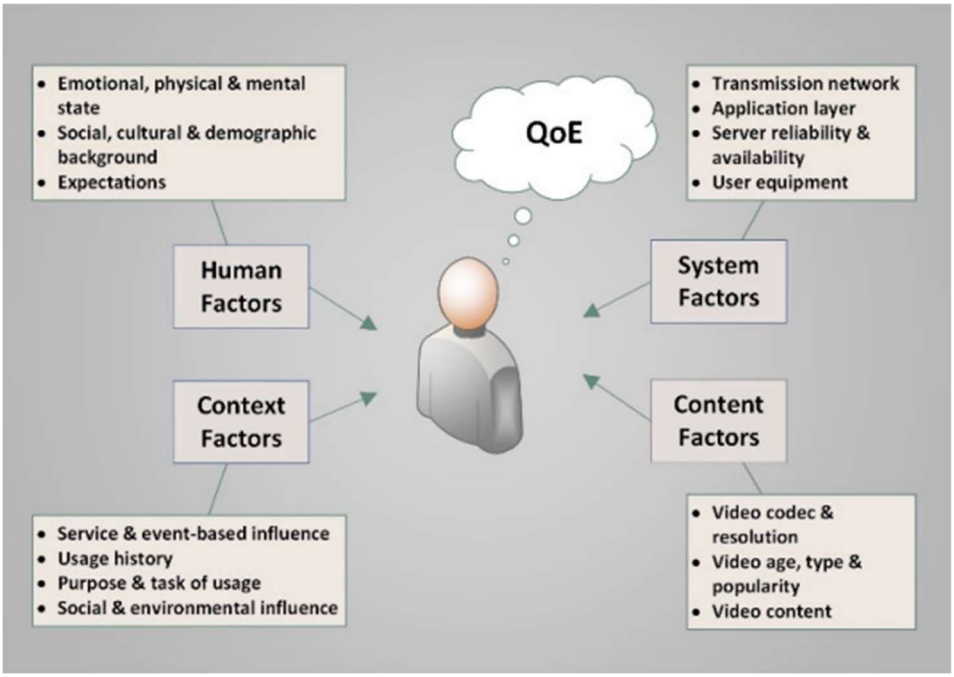


Figure 7: Factors affecting QoE. [23]

Considering the wide range of factors affecting QoE, accurate QoE assessment and especially prediction are not trivial tasks. However, according to [23], QoE assessment can be performed in two distinct manners based on the granularity of available data: subjectively or objectively. Subjective QoE needs human feedback from network users, whereas objective QoE assessment concerns only measurable QoS metrics. Subjective QoE assessment is generally much more accurate, but objective assessment has the advantages of simplicity and flexibility and is therefore much easier to implement.

2.4 Machine learning concepts

This subsection describes basic concepts of machine learning. It gives an overview of current machine learning paradigms and briefly explains some of the most common machine learning models as well as models that are important for this work.

2.4.1 Overview of machine learning

Advancements in machine learning during recent years, especially with the introduction of deep learning algorithms, have led to breakthroughs in various domains, including anomaly detection. Machine learning is an essential part of the data science process, enabling data scientists to create generalisations and to make predictions from vast and complex datasets. By leveraging

machine learning methods, data scientists can create more robust models performing more accurately than many traditional approaches.

Machine learning relies on large amounts of data, therefore, data science is an important aspect of it. Data science includes everything related to data. For example, analysis, visualisation and cleaning. A subset of data science is mining which is about extracting underlying information and useful insights from data [26]. The relationships between artificial intelligence and data science concepts are presented in Figure 8.

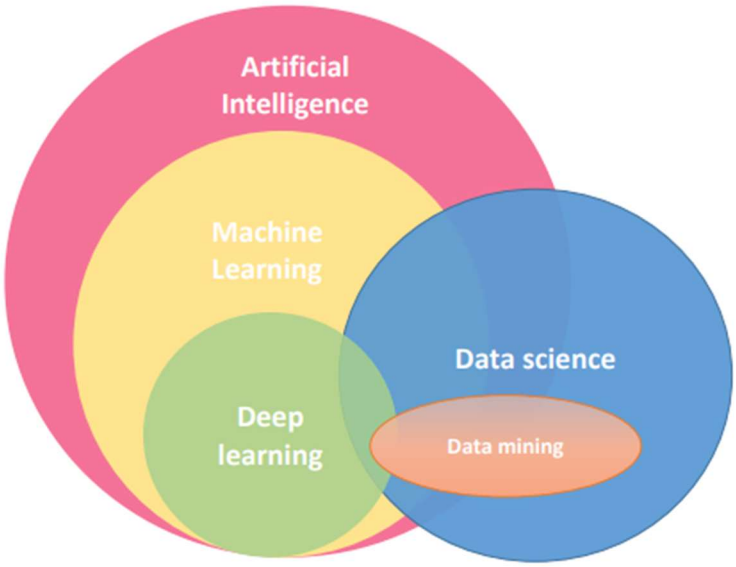


Figure 8: Data science and machine learning overview [26]

Deep learning is a subset of machine learning. Deep learning models, such as DNNs, consist of multiple processing layers and are capable of learning generalisations of data with multiple levels of abstraction. The use of traditional machine learning algorithms often requires domain knowledge, especially to determine which features of data are useful, whereas DNNs can learn complex non-linear relationships between features without any expert input in the feature engineering stage [27]. However, as a trade-off deep learning models need considerably larger amounts of data and might lose some of their interpretability as the decisions are made automatically by the model. Therefore, some DNN-based models operate as "black boxes" that offer little to no explanation of why, for example certain features were chosen over other features [27].

2.4.2 Machine learning paradigms

Machine learning can be divided into several paradigms based on the type of learning task: supervised, unsupervised, semi-supervised, self-supervised

and reinforcement learning [26]. Supervised learning involves learning from labelled data. Supervised machine learning models learn the relationships between input data (features) and output data (labels) and can be used to predict labels for new unseen data. Unsupervised learning on the other hand involves learning from unlabelled data. Unsupervised models try to find generalisations or relationships between different features of the data such as clusters of data points or outliers [26]. Semi-supervised learning involves a partially labelled dataset. It is used to improve one of the two previously mentioned tasks by either providing additional data for supervised learning or making unsupervised learning more accurate by providing additional insight about some of the data points. Self-supervised learning is often considered a subset of unsupervised learning, but it also shows characteristics of supervised learning [28]. This means that while self-supervised models do not depend on labelled data, they can create objectives or labels for themselves. Reinforcement learning is learning by trial and error. It includes an agent that receives rewards for its actions and learns to perform the actions leading to most rewards [29].

In addition to these paradigms, machine learning processes can be divided into online, offline and active learning, based on how learning is carried out [26]. Offline learning trains the model on the entire dataset at once and is especially useful when the model is not required to change its properties over time. Online learning, on the contrary, is trained incrementally, making it useful in scenarios where the most recent data points are crucial; this is often the case with time series data, for example. Active learning chooses the most useful data points to be labelled by the user and uses them for model training.

The following sections describe some of the most popular traditional supervised and unsupervised machine learning methods. In addition, a few deep learning models including both supervised and unsupervised learning are explained.

2.4.3 Supervised machine learning models

Linear regression is one of the best-known supervised learning algorithms, that finds the simplest possible relationship between a set of input features and an output, which can be seen as fitting a line to model data [26]. A simple example of linear regression can be seen in Figure 9, where two different lines are fit to describe data, each in its own way.

Logistic regression is a supervised learning algorithm used for linear classification, which divides data into different classes by separating individual data points with a linear decision boundary [26]. In simple terms logistic regression draws lines that separates data points into different classes. Logistic regression classifies data points by determining probabilities with which a data point belongs to a specific class.

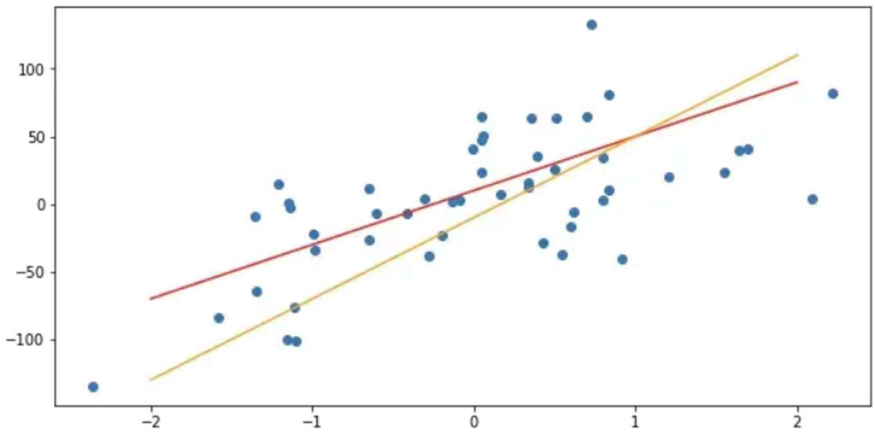


Figure 9: Linear regression [30].

Naïve Bayes classifier uses Bayes Theorem [31], which is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad (1)$$

which states the probability of an event A happening given that B has happened. By using Bayes Theorem, multiple input features can then be combined using the chain rule assuming that every feature is independent from each other [32]. Naïve Bayes is used for example in text classification and recommendation systems and its advantage is its easy implementation. However, its assumption of feature independence might be problematic in some problems that have dependent features.

Support vector machine (SVM) is a supervised learning algorithm primarily utilized for classification problems, although it that can also be used for regression. SVM can perform linear classification by maximising the margin of the hyperplane separating two classes of data points. The difference to logistic regression is that logistic regression does not try to find the largest margin to separate two classes, like on the left side of Figure 10. Another difference is that SVM can also perform non-linear classification by using the so-called kernel trick. This means that it can take non-linear inputs, and map them into a high-dimensional feature space where it is then possible to separate classes with a hyperplane [32].

Decision trees represent choices and their consequences in the form of a tree [26]. The edges of the tree represent decisions made based on the question asked in the decision nodes. Leaf nodes represent the final class or decision. Decision trees are a popular supervised learning algorithm and often used as parts of random forests. Random forests consist of multiple decision trees and base their classifying decisions on the most common decision of the decision trees. Figure 11 illustrates an example of a decision tree.

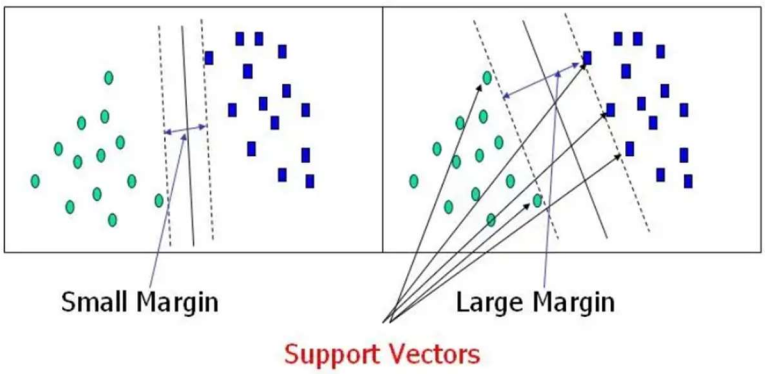


Figure 10: Support vector machine [33].

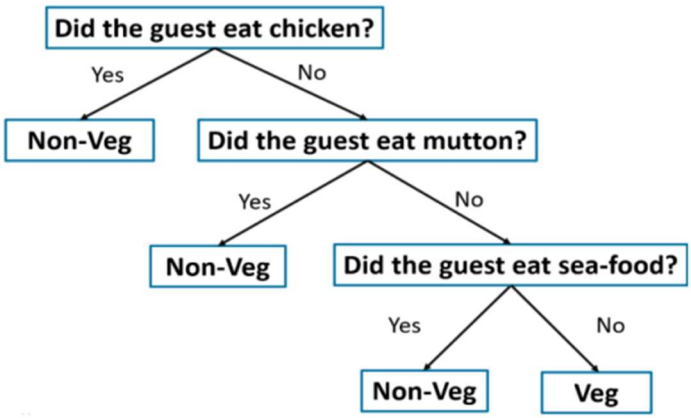


Figure 11: Decision tree example [32].

2.4.4 Unsupervised machine learning models

Principal component analysis (PCA) is a form of unsupervised learning that can be used to reduce data dimensionality. It is especially useful when there are multiple features in data and a need to reduce their number. PCA converts these possibly correlated features into a significantly smaller set of uncorrelated features. These new features are called principal components, which aim to preserve as much of the original variance as possible. PCA makes computations more efficient and guarantees that features are uncorrelated but as a trade-off, PCA reduces the interpretability of features [32].

Clustering is an unsupervised learning method, which means grouping data points into different groups, also known as clusters. Data points in the same cluster should be as similar as possible while data points in different clusters should be different from one another. Clustering is a commonly used technique in machine learning, and it can be used for example in anomaly detection. There are a variety of different clustering algorithms. The following ones are listed in [34].

One of the best-known clustering algorithms is called K-Means. In K-Means clustering, K centroids are defined and placed somewhere among the data. Each data point is then assigned to a cluster based on the Euclidian distance to the nearest centroid. After that, K new clusters are defined based on the mean of each cluster. These steps are repeated until no data point changes its cluster. K-Means clustering normally assumes roughly spherical clusters.

Sometimes the shape of data is not suitable for a classical clustering algorithm such as K-Means. This is the case for example with the two nested circles in Figure 12. In this case, another type of algorithm is required. An algorithm that could perform accurate clustering in the aforementioned situation is spectral clustering. Spectral clustering has its roots in graph theory but can also be used to cluster data. The steps of spectral clustering are as follows:

1. Create a similarity graph describing the data. A measure of data similarity can be for example the Euclidean distance between data points on a plane, but sometimes it can be a vague concept where domain knowledge often proves helpful.
2. Create a graph Laplacian from degree and weight matrices derived from the similarity matrix. The eigenvectors of the Laplacian matrix are features describing the data.
3. Based on these features, apply a classical clustering algorithm, such as K-Means clustering.

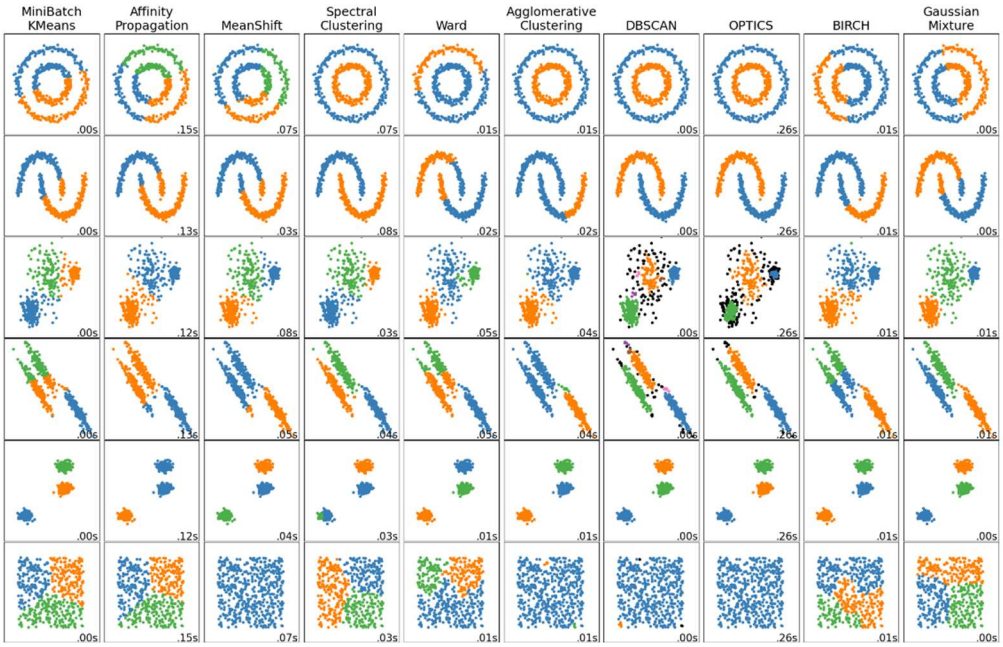


Figure 12: Clustering algorithms [35].

A common characteristic between the aforementioned clustering algorithms is that they need the number of clusters defined before the clustering. However, this can cause challenges with some data as the desired number of clusters is not always known beforehand. Nevertheless, there are alternative clustering methods capable of addressing this problem, such as hierarchical clustering. Hierarchical clustering can be executed in a bottom-up (agglomerative) or in a top-down (divisive) manner. In agglomerative clustering, each data point is declared as its own cluster. Similar clusters are then combined step by step until a suitable clustering is found. There are again numerous ways to determine cluster similarity. Some of the more common ones used in hierarchical clustering are for example minimum, maximum and average distance between data points of two clusters, as well as Ward's criterion which measures the increase of sum of squared errors in a newly formed cluster. Divisive clustering is the opposite of agglomerative clustering where there initially exists only one cluster which is then divided into smaller ones. In addition to the advantage of not requiring the number of clusters, hierarchical clustering tends to perform well regardless of the choice of the similarity metric. Even though the number of clusters is not required before the clustering, the users ultimately select the number of clusters they want to use. A disadvantage is that hierarchical clustering can be quite inefficient and computationally expensive, especially for larger datasets.

Another clustering method that does not require the number of clusters is density based DBSCAN. Instead, it requires two parameters: *MinPts*, which determines the number of data points required to form a cluster within a radius *Eps*. If a data point does not have at least *MinPts* neighbours within *Eps*, it is considered an outlier. DBSCAN is useful when the clusters have irregular shapes and when outliers or noise are expected in the data. A more detailed description of the DBSCAN algorithm can be found in Section 3.3.3.

Ultimately, the choice of clustering algorithm depends on data as well as on the shapes and sizes of the desired clusters. Some algorithms might perform very accurately on a specific type of data while performing poorly on another type. Figure 12 illustrates multiple different clustering algorithms and how they perform on data of different shapes.

2.4.5 Deep learning models

Multilayer Perceptron (MLP) is a feedforward deep neural network typically used for supervised learning. It consists of input and output layers and one or more hidden layers. The hidden layers consist of individual perceptrons. Perceptrons are neurons which combine inputs in a weighted sum [36]. In simpler terms, perceptrons are just decision-making units, and the importance of their decisions is indicated by their weights. The outputs of individual perceptrons are then determined by a special activation function such

as ReLU (Rectified linear unit), to introduce non-linearity to the network [36].

In MLP, each layer feeds forward its computed output to the next layer. Once the outputs reach the output layer, they are compared to a loss function to extract the difference between the model's predicted output and the actual one. Based on these results, the MLP updates the perceptrons' individual weights to obtain a better result. This step is called backpropagation and represents the "learning step" of the algorithm. The feed forward and backpropagation parts are repeated until each individual neuron has converged to a certain extent [36].

Convolutional Neural Network (CNN) is a special type of deep neural network that can learn spatial and temporal dependencies in data and is therefore useful for example in processing images and time series data. Much like MLPs, CNNs, displayed in Figure 13, also consist of input and output layers as well as series of hidden layers, which include convolution and pooling layers as well as one or more fully connected layers. Each of these layers have their own specific functionalities. For example, convolution layers can detect different features of the image, such as edges or colours, while pooling layers summarise this information by combining different features. Finally, fully connected layers are used for data classification. The fully connected layers form a feed-forward network, which can be for example an MLP [26].

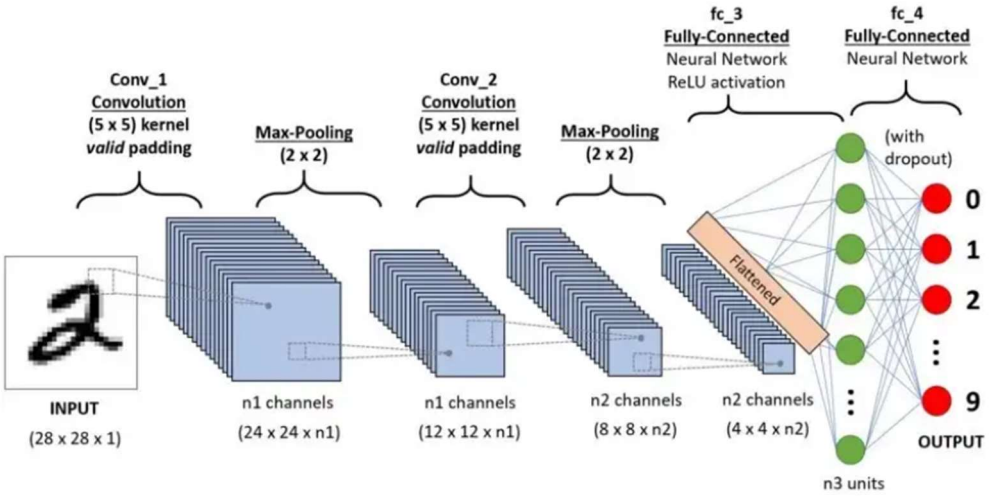


Figure 13: An example of a CNN [37].

Another type of deep neural network is the Recurrent Neural Network (RNN). RNNs are specifically useful for learning sequential data and the relationships between consequent data points. Typical feed-forward networks are designed for data points that are independent of each other, but RNNs are useful for data where a data point depends on the previous data points. Examples of these situations are natural language processing and time series

data. Much like the previously mentioned networks, RNNs also consist of input, output and hidden layers. However, in RNNs the hidden layers have recurrent connections between each other [26].

RNNs are known to suffer from the so-called vanishing gradient problem, which can make the learning of the network very slow or stop completely [38]. Another problem that sometimes arises with RNNs is the problem of capturing long-term dependencies. LSTM networks are specific types of RNNs designed to overcome these issues [38]. An alternative to LSTMs is Gated Recurrent Unit (GRU), which is known to perform even better than LSTMs for smaller datasets due to its simpler structure [39].

In recent years, Graph Neural Networks (GNNs) have become a promising method to be used for example in classification, clustering and link prediction of graph-structured data in various fields, such as natural and social sciences as well as traffic networks [40]. GNNs assume that each node is affected by its neighbours' states and therefore could be used to model time series data from multiple different sensors monitoring the same system [41]. The four steps of designing a GNN according to [40] are:

1. Finding a graph structure.
2. Specifying graph type for example whether the graph is directed and homogenous or not.
3. Designing the loss function based on the task and training type such as node-level clustering in an unsupervised setting.
4. Building the model using computational modules, such as the previously mentioned convolutional, recurrent and pooling layers.

AEs are DNNs or NNs (Neural Network), which consist of an encoder compressing data, and a decoder reconstructing the compressed data to best match the input. The difference between the reconstructed data and input data is called reconstruction error. AEs are some of the best-known deep unsupervised learning models, but due to their objective of minimising the reconstruction error, they can be considered to belong also in the category of self-supervised learning. The representations that AEs learn to model data in a compact manner are useful especially for dimensionality reduction. However, AEs can also be used for detecting anomalies by being trained only with normal data which would then lead to high reconstruction error for anomalies [42].

AEs aim to generate similar reconstructions for similar inputs, but Variational Autoencoders (VAE) reconstruct data matching the probability distribution of data, which can be useful to model for example non-stationary time series data, such as data containing seasonal variation [39]. An example of an AE is displayed in Figure 14. AEs consist of neurons and hidden layers just like other NNs, but a way to think of them is as two distinct NNs, the encoder and the decoder. Despite models like MPLs, CNNs and RNNs being typically used for supervised learning, they can also be employed in unsupervised settings, such as in the encoder and decoder of AEs.

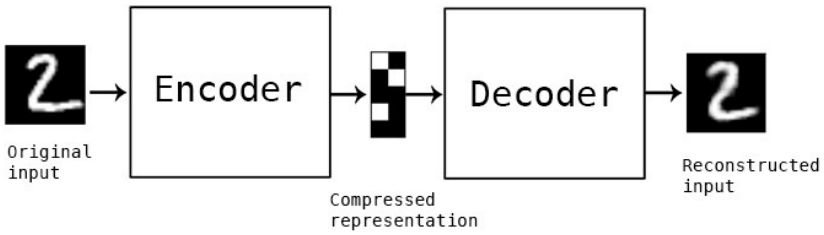


Figure 14: An example of an autoencoder [43].

2.5 Self-organising networks

As mobile networks are evolving, there is an ever-increasing need for intelligent network management solutions to accommodate for rising amounts of traffic, increasing number of devices and emergence of new applications. Traditionally, network planning and optimisation has been performed by manually adjusting thousands of BS-parameters and continuously monitoring networks. This requires skilled professionals for monitoring and fixing problems in the network, which is expensive, time-consuming and error prone. Self-organising networks (SON) are agile and autonomous networks, which are designed to reduce the need for manual network management in mobile networks.

The development of SONs is closely related to that of machine learning techniques, which can allow for more effective and robust managing of complex networks in real-time [44]. Also known as self-x functions, SONs can be divided into three categories: self-configuration, self-optimisation and self-healing [1].

Self-configuration means automatic configuring of network equipment parameters such as operational, radio and neighbour cell list parameters of base stations [1]. Self-configuring can happen when new equipment is added into an already operable network or when the network is in need of reconfiguration after recovery.

Self-optimisation is a SON function, which means continuously monitoring and adjusting network parameters to assure that the network is functioning as efficiently as possible [1]. Self-optimisation covers for example optimisation of caching commonly accessed data at base stations, addressing the trade-off between coverage and capacity, mobility prediction, and load balancing as well as optimisation of resources and handover parameters. In addition, self-optimisation might also perform coordination of different SON-functions as sometimes there might be a conflict of interests such as between minimising interference and maximising capacity.

The third function of SONs, and the focus of this thesis is self-healing. Network fault detection, fault classification and cell outage management are examples of self-healing. Self-healing procedures have traditionally been

triggered by manually monitoring the network or relying on fixed thresholds of network KPIs. With the help of new machine learning techniques, SONs can automatically detect anomalies, and in some cases, even predict them already before they lead to any more significant errors. This marks a paradigm shift from reactive to proactive self-healing [1]. After an anomaly has been detected, its root causes must be identified. This procedure is called fault classification or root cause analysis, and it is an important step in order to take the correct actions to fix the problem.

SONs can be deployed in different architectures, such as centralised SON (C-SON) and distributed SON (D-SON) depending on where the computational and decision-making processes occur. They are already being used around the world, resulting in reduced operational and capital expenses as well as improvements in network performance [44].

2.6 Anomaly detection and root cause analysis

Anomaly detection is the process of detecting unexpected behaviour from data and is often used for multivariate time series data across a myriad of different fields. Multivariate time series data refers to data that has multiple features and is collected over constant intervals of time. An example of multivariate time series data is mobile network data, which includes multiple network KPIs collected from base stations over a period of time. Anomalies in time series data can be divided into point, contextual, and collective anomalies [45]. Point anomalies are often caused by noise or errors in system operation and are single data points clearly differing from the rest of the data. Contextual anomalies do not normally stand out as clearly as point anomalies but can be labelled as anomalies based on the context, they appear in. Examples are a normal mobile network traffic pattern of a weekend on a weekday, or a traffic peak in the middle of the night. Lastly, the behaviour of individuals might not be anomalous inside their own group, but the behaviour of the group might be anomalous when compared to other groups. These types of anomalies are called collective anomalies.

Some of the more traditionally used anomaly detectors, such as statistical, threshold, linear and distance-based methods, as well as even some machine learning methods, such as SVMs, are facing challenges in detecting anomalies in complex data where dimensionality is high and labelled data points are scarce [45]. To address these challenges, new DNN-based anomaly detectors have been developed in recent years. Figure 15 shows the results of [46] comparing conventional, machine learning and DNN-based methods for anomaly detection on five well-known multivariate time series datasets, from a variety of different fields. While the F1 scores of all method groups indicated by different colours are in close proximity to each other, DNN-based methods seem to have a slight edge over the other two on most of the datasets.

While DNNs have been achieving better results recently throughout many applications including anomaly detection, it should still be kept in mind whether a DNN is viable or even needed for some problems. DNNs are known to require vast amounts of data for training, which can be computationally heavy. The five datasets used for comparison in [46] contain between 27 and 123 features, hence datasets with fewer features might turn out not to require DNNs. Also, the nature of the anomalies should be considered. From the five datasets, Water Distribution (WADI) [47] contains contextual anomalies and therefore could experience better results with the use of DNNs [46].

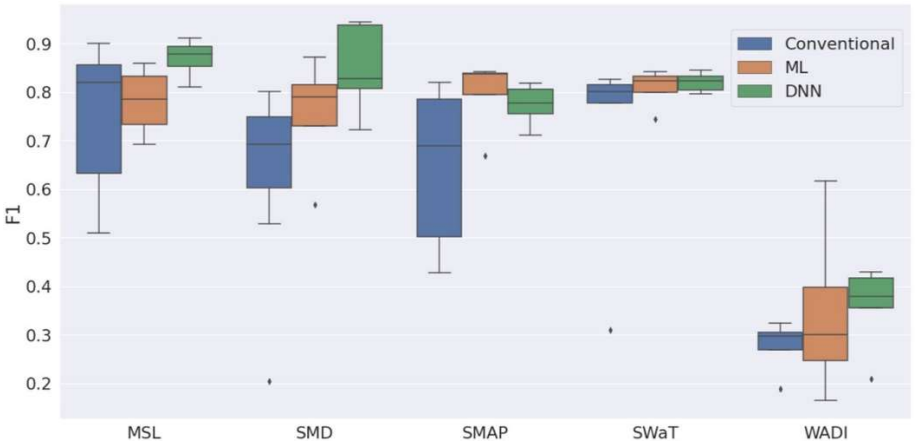


Figure 15: Comparison between F1 scores of anomaly detection method groups on five datasets [46].

When anomalies are detected, their cause should be determined. The process of discovering the specific reason behind the anomaly is called root cause analysis. Root causes are precise underlying causes that can be identified and fixed [48]. Finding the root cause is important especially in mobile networks in order to repair the fault that might affect the network used by thousands of people and to prevent the recurring happening of the fault, ultimately resulting in higher QoE for the network users.

Mobile network RCA has traditionally relied on the diagnosis of network experts examining alarms, KPIs, and configuration parameters [49]. Despite RCA being a common task performed by network operators, public datasets with network faults labelled by their root causes are rare [50]. Moreover, mobile networks are unique and might exhibit faults that are distinct from other networks, therefore emphasising the use of expert knowledge to discover the characteristic faults of a specific network [49]. These aspects make the use of supervised learning methods difficult, hence shifting the focus of automatic RCA towards unsupervised learning.

2.7 Related work

Anomaly detection is a popular research area and new solutions are continuously developed. The evolution of machine learning techniques empowers researchers with tools to detect anomalies with unprecedented speed and accuracy. In what follows, an overview of some of the most promising general machine learning-based anomaly detection models that could in theory be deployed in any field is provided. Anomaly detection models and RCA solutions designed specifically for mobile networks are discussed further.

A comparison of multiple popular DNN-based anomaly detection models is carried out in [45], by using three different anomalous multivariate time series datasets: WADI, Secure Water Treatment (SWaT) [51] and Mars Science Laboratory Rover (MSL) [52]. The datasets have different numbers of data points, features and anomaly rates. WADI has 123 features, which more than doubles the number of features in the other two datasets.

Table 2 contains the best performing anomaly detection models in terms of F1-scores from [45]. AE-based USAD [42] performs well on the MSL dataset, but weaker on WADI. GNN-based GDN [41] was more consistent on all datasets and with an F1 score of over 85 on each dataset. Noteworthy is that GRU and VAE-based OmniAnomaly [39] received the highest precision of all models on SWaT and the highest recall on WADI, but the F1-score was not among the highest for either one. This goes to show that it is difficult for a model to universally detect anomalies in different contexts, as the performance is highly dependent on the dataset used. Other above average models were RNN-based THOC [53] and GNN- and transformer-based GTA [54].

The clearest difference in the F1-scores can be seen in the WADI-dataset between the graph-based models (GTA and GDN) and the rest. The authors of [45] believe that the reason is the high dimensionality of the WADI dataset and the fact that graph-based models do not perform dimensionality reduction. The models that do, might lose some important features, that leads to lower F1-scores. The graph-based models also slightly outperform the others on the SWaT dataset. The authors state that SWaT, as well as WADI have some features that correlate with each other and therefore the graph-based models, which are able to learn the graph structure between features, are more effective.

Table 2: Anomaly detection models' F1-scores on three datasets [45].

Model	F1 on SWaT	F1 on WADI	F1 on MSL
OmniAnomaly	86.67	41.74	89.90
USAD	84.60	42.96	92.72
THOC	88.09	50.59	89.06
GTA	91.34	83.76	91.10
GDN	93.59	85.52	90.33

The models discussed above are general anomaly detection models that can be used to detect anomalies in multivariate time series datasets regardless of the field. Anomaly detection specifically in the context of mobile networks also has its fair share of literature from recent years. For example, [55] tries to capture the spatial and temporal nature of LTE RAN data by using a model that combines the strengths of capturing spatial dependencies of CNNs and capturing temporal dependencies of LSTMs. Then, together with AE-based feature selection, a decision tree model is used to classify cells into normal or anomalous classes. LSTMs are also utilised in [56] to capture temporal dependencies and combined with an AE, to detect anomalies based on the reconstruction error. Autoencoder is used also in [57], where the non-stationary nature of mobile data is considered with a VAE-based model requiring significantly fewer computing resources while compromising only two percents in the F1-score when compared to OmniAnomaly. However, the comparison was done using only one dataset. A custom NN is designed in [58] to consider the effect of more than one fault happening simultaneously in an LTE RAN. A more statistical approach is taken in [50], where the Chi-squared test is used to statistically detect anomalous network cells based on different traffic profiles (weekday, weekend, rural). Another method not relying on deep learning is experimented in [59], where a more user-centric approach is taken and a mapping between network KPIs and QoE is created. The authors use a Classification and Regression Tree (CART) to predict QoE score within base stations. If the score is below a threshold, an anomaly is detected, indicating a dysfunctional base station.

As seen before, GNN-based models have shown promise as general anomaly detection models. However, GNNs are a rather new technology, and have not been widely used in mobile anomaly detection publications. A recent paper published in 2023 by [60] uses a graph-based approach to detect anomalies from call detail records (CDR) in a supervised setting. The proposed method captures the spatial dependencies between different cells of a cellular network by using a Graph Convolution Neural (GCN) network. The GCN network is a CNN which considers the dependencies in a graph-structured cell network. The GCN module is combined with three LSTM modules. Each LSTM module operates on a different temporal scale with the goal of capturing hourly, daily and weekly dependencies of the data. Although applied only to a simple CDR dataset which does not include many typically used network KPIs, the results are promising and encourage further research on GNN-based solutions for mobile anomaly detection.

It should be noted that many research papers are using a network simulator to acquire network data, since real world operator data can be difficult to come by. This is the case in [50], [55] — [58], [49] and [60]. Even if researchers are able to obtain labelled real-world data, it is usually very unbalanced, meaning that most of the time the network is functioning normally and anomalies occur rarely [60]. The imbalance between normal and

abnormal data can make model training difficult [45]. In [60], this problem is tackled by using a generative adversarial network (GAN) to generate artificial data points similar to real-world data. In addition, data undersampling, where data points clearly far away from the time of appearance of anomalies are removed from training data, as well as penalised classification, where the classifier is additionally penalised for classifying an anomaly point as normal, are considered in [55]. Arguably the greatest challenge in real world network data is the lack of data labels, meaning the ground truth behind anomalies, for model training [54]. The lack of labels significantly limits the available approaches, especially supervised machine learning methods, and reduces the interpretability of the results. Therefore, "manually" labelling data points is used in some work. For example, in [61] data points with a Euclidean norm larger than the norm of standard deviations are labelled as anomalies. In [62], mobile network KPI dataset is labelled with user QoE values extracted from user throughput and handover success rate. This approach based solely on a few KPIs to estimate QoE might turn out not to be the most accurate, considering that QoE estimation requires lots of information as discussed in Section 2.3. The wide range of different applications with different throughput requirements also restricts the use of fixed value thresholds.

The lack of labels mainly affects supervised learning methods, making unsupervised learning particularly useful in cases with large amounts of unlabelled data [1]. Clustering techniques are used in [63] and [64] to detect anomalies based on user activity from CDR data. Both use the same dataset with one-hour and 10-minute aggregation intervals respectively. Although anomaly detection from CDR using only user activity mainly provides information about users' behaviour and might not reveal faults in the network, these papers show the effectiveness of simple clustering algorithms (agglomerative and K-means) in finding simple anomalies in aggregated time series data.

Agglomerative clustering is used also for RCA in [55], where the authors try to group anomalies caused by the same problem into the same cluster. While this approach does not require data labels, it does require a network expert to determine the root cause label for each cluster. A similar, yet more complex solution is presented in [49], where the authors use a Self-Organising Map (SOM) to transform high-dimensional KPIs into a more simplified form. Next, Ward's hierarchical clustering is used to cluster the simplified data into clusters corresponding to different root causes. The authors ensure that the different clusters represent different faults by comparing each cluster's statistical properties with the Kolmogorov–Smirnov test. Network experts are needed to label the clusters with their root cause labels in the training phase, but in the exploitation phase the system compares the KPI statistics to those labelled in the training phase and is automatically able to determine the most probable root cause.

In addition to unsupervised clustering, in [55] a supervised decision tree approach is used to classify anomalies under different root cause labels generated by network experts, including too late handover and excessive antenna tilt. Decision tree structure is used also in [57], where anomalous data points are labelled with an anomaly score calculated from their KPIs. A boosting tree classifier then uses the list of anomaly scores to classify the anomalies to match one of six different root causes. The effect of multiple faults happening simultaneously is considered in [58] with the use of supervised NNs trained to detect specific root causes. However, only three root causes were considered, two of which, excessive antenna uptilt and downtilt, cannot happen simultaneously. Another supervised method, Naïve Bayes classifier is used in [50] to express the uncertain root causes in terms of probabilities.

Example root causes considered for example in [50] and [49] are excessive antenna downtilt or uptilt, coverage hole, inter-system interference, too late handover and cell outage. Examples of network KPIs used are reference signal received power, reference signal received quality, handover success rate, signal to interference plus noise ratio, distance between user and base station, and average user throughput. All discussed papers address anomaly detection and root causes analysis in 4G LTE networks.

It is evident that there is a multitude of different approaches to conduct anomaly detection and RCA in the context of mobile networks. The choice of the approach depends on multiple factors including labels, the size and features of dataset, as well as available computational resources. Other considerations include the trade-off between missed anomalies and false alarms, whether proactive or reactive anomaly detection is required, whether the data is processed in sliding windows or incremental updates, and whether to use online or offline training [45].

3 Research material and methods

This chapter describes the implementation of a machine learning-based model to detect anomalies from unlabelled mobile network data and to automatically determine the anomalies' root causes. The used data is explained with data analysis and the specific steps to conduct anomaly detection and root cause analysis are explained under methodology.

3.1 Dataset description

The dataset used in this work includes mobile network KPIs collected during a period of over 6 months of network traffic from 25 LTE cells in the 4G radio access network of a Portuguese telecom operator, NOS. The data collected from one of the base stations is from the period of 13.9.2022 – 15.3.2023 and the data from the remaining 24 cells is from 3.12.2022 – 6.6.2023. The data from the earlier period is from a BS called CEIRA_LTE_MCO001B2. The decisions regarding the system architecture and the used models are based on the data analysis and the observed results of this BS, even though the model parameters are set individually for each of the 25 BSs. Consequently, CEIRA_LTE_MCO001B2 can be viewed as a training dataset and will be referred to as the training BS. The KPIs along with their descriptions and units are presented in Table 3.

Table 3: Network KPIs.

Abbreviation	Network KPI	Description	Unit
USERS	Average number of users	Average number of simultaneous UE.	-
PRB_DL	PRB DL average usage rate	Average physical resource block utilisation in downlink direction.	%
PRB_UL	PRB UL average usage rate	Average physical resource block utilisation in uplink direction.	%
HSR_INTRA	HSR intra frequency	Handover success rate of intra-frequency relations.	%
HSR_INTER	HSR inter frequency	Handover success rate of inter-frequency relations.	%
CELL_DL_TP	Cell DL average throughput	Average cell data throughput in downlink direction.	kbps
CELL_UL_TP	Cell UL average throughput	Average cell data throughput in uplink direction.	kbps
USER_DL_TP	User DL average throughput	Average user data throughput in downlink direction.	kbps
USER_UL_TP	User UL average throughput	Average user data throughput in uplink direction.	kbps
TVD	Traffic volume data	Traffic volume of data services.	MB
CA_TP	Carrier aggregation throughput	Total throughput using carrier aggregation.	kbps
CQI	Average CQI	Average Channel Quality Indicator.	-
TIME_ADV	Average timing advance	Distance between UE and BS	-

Individual data points in the dataset describe hourly aggregated network KPI-values. Aggregation here means the average over an hour in all the KPIs except for traffic volume data (*TVD*) and carrier aggregation throughput (*CA_TP*), where the values are the sums over an hour.

KPIs are calculated using data from multiple counters provided by network device vendors. The counters are located at the eNodeB base stations of the LTE network and are collecting data of the traffic going through BSs. Each BS consists of three 120-degree sectors, each containing three cells. Different cells represent different frequency bands. Most of the BSs use the 800, 1800 and 2100 MHz bands, but there are some exceptions, such as the 2600 MHz band being used. As a result, the dataset contains 25 (base stations) × 3 (sectors) × 3 (cells) × ~190 (days) × 24 (hours) > 1 000 000 data points. For better understanding the organisation of the dataset, a table of the cells based on sectors and frequency bands is provided in Table 4. In addition, a sample of five hours of the network KPIs is available in Table 5.

Table 4: Organisation of cells based on the sector and the frequency band.

	Sector 1	**Sector 2**	**Sector 3**
800 MHz	Cell 1	Cell 2	Cell 3
1800 MHz	Cell 4	Cell 5	Cell 6
2100 MHz	Cell 7	Cell 8	Cell 9

Table 5: Five-hour sample of the dataset.

timestamp	ltecell_name	users_avg	prb_dl_usage_rate	prb_ul_usage_rate
10/16/2022 0:00	CEIRA_MCO001N1	26.1125	58.06877056	5.996893889
10/16/2022 1:00	CEIRA_MCO001N1	24.89167	32.52273444	3.633177778
10/16/2022 2:00	CEIRA_MCO001N1	23.04583	28.27058111	1.903409444
10/16/2022 3:00	CEIRA_MCO001N1	19.08889	13.75593611	1.230778889
10/16/2022 4:00	CEIRA_MCO001N1	18.15278	12.67761333	3.197061111

hsr_intra_freq	hsr_inter_freq	cell_throughput_dl	cell_throughput_ul	user_throughput_dl
100	100	14219.4382	774.2674005	9616.047041
94.73684211	100	12499.60991	842.0447237	12016.66382
100	100	15081.95852	903.2913717	15691.89558
100	100	12130.93938	590.8063301	12421.5761
90	100	9683.548865	793.5256017	9247.721016

user_throughput_ul	traffic_volume_data	tp_carrier_aggr	cqi_avg	time_advance_avg
1016.256597	4324.145375	19547.72104	8.05444	5327.901122
1903.046637	2387.203625	28378.63933	8.29886	8484.626783
2958.047233	2259.9655	37158.02291	8.4099	6396.841882
1076.259918	1024.6125	7942.526422	9.14352	7112.94462
1382.53223	792.708625	29028.22423	8.71464	4911.903246

3.2 Data analysis

As with any data driven approach, a comprehensive data analysis is needed to understand the nature of data and the underlying patterns and dependencies between different features of it. Data visualisation, feature correlation and data grouping are methods that, when used together with domain knowledge, can help discover useful insights. These insights can further be used for first, understanding data, and secondly, to help detect anomalies.

Pearson correlation coefficients [66] between different KPIs in different sectors of a BS are calculated according to:

$$r = \frac{COV(X,Y)}{\sigma_X \sigma_Y}, \quad (2)$$

where X and Y are time series of different KPIs, $COV(X,Y)$ is the covariance between X and Y, σ_X is the standard deviation of X, and σ_Y is the standard deviation of Y. It is noteworthy that KPIs can correlate to each other differently in different sectors. For example, Figure 16 shows a correlation of 0.4 between *USERS* and *TVD* in sector 2, whereas Figure 17 shows a correlation of 0.8 between the same KPIs in sector 3 of the training BS. This means that sector 2 has more factors affecting traffic volume than just the number of users, such as for example the nature of data services used. Correlations between KPIs of sector 1 are presented in Figure 18, and they are also unique. Correlation analysis can be helpful in finding the relationships between different KPIs, which can be useful especially when performing RCA.

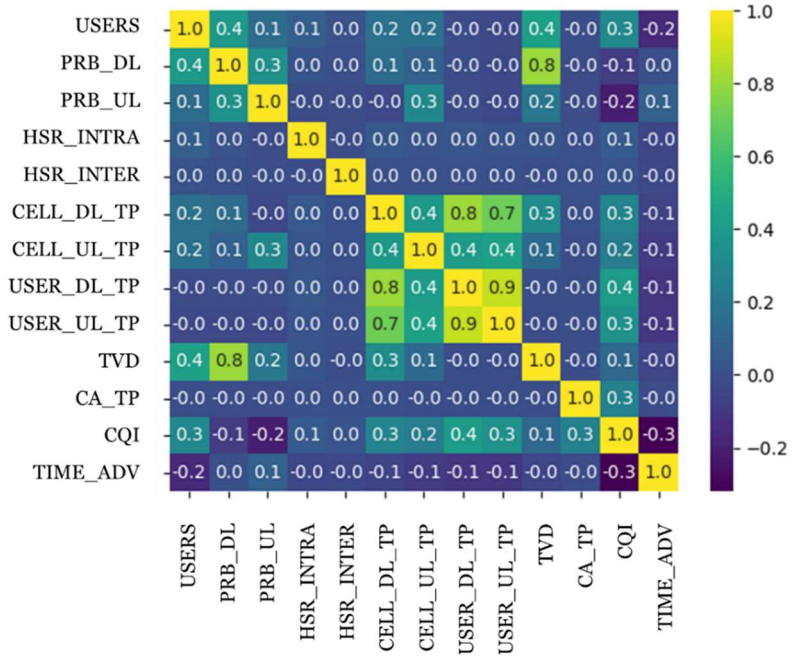


Figure 16: KPI-wise correlations in sector 2 of the training BS.

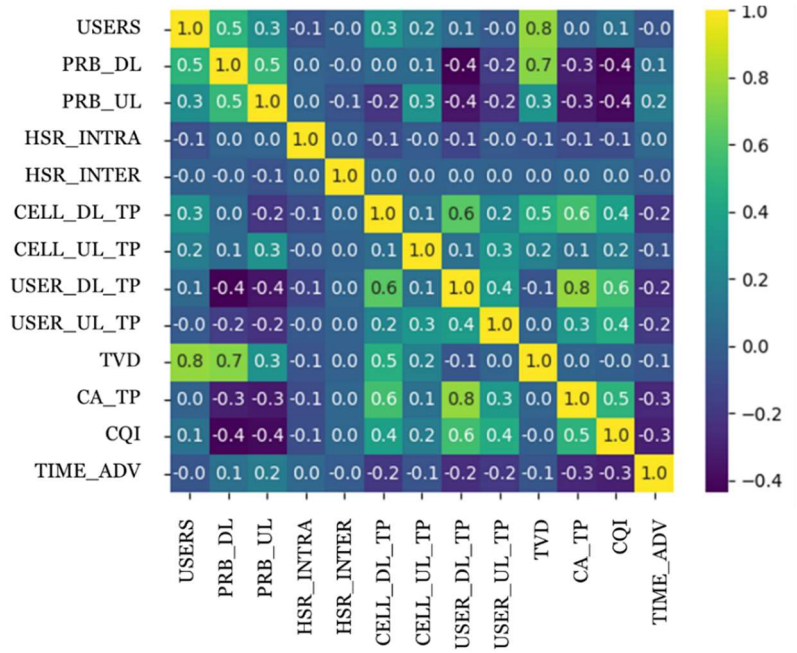


Figure 17: KPI-wise correlations in sector 3 of the training BS.

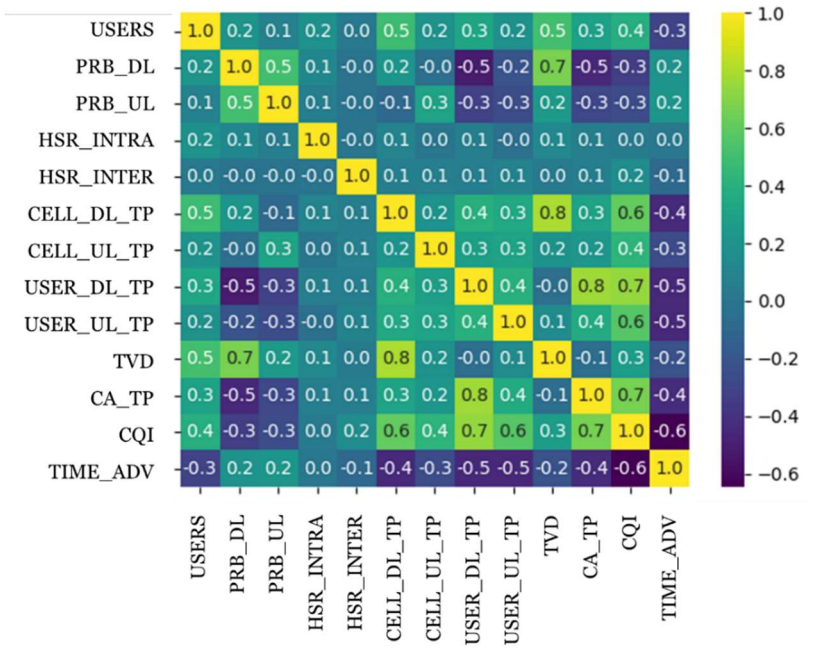


Figure 18: KPI-wise correlations in sector 1 of the training BS.

In addition to differences in correlations, different sectors can exhibit distinct characteristics such as variations in traffic patterns. These distinctions are a result of sectors facing towards different geographical regions. Several studies have investigated the characteristic behaviours based on geographical location. For example, [67] identifies five different traffic profiles

in the urban environment: residential, office, transportation, entertainment and comprehensive (for the areas with mixed profiles). Typically, the geographical areas do not fully correspond to a single profile, but rather contain characteristics of multiple profiles.

Traffic volume patterns displayed in Figure 19 show a dominant profile of a residential area in sector 1, whereas Figure 20 shows a dominant profile of an office area in sector 3. Sector 2 on the other hand, as illustrated in Figure 21, does not fully match any of the known traffic profiles, and therefore most likely consists of a mix of several of them. Notably, some sectors have completely different behaviours on working days and during weekends. Especially, the office area in sector 3 displays lots of variance between working days and weekends. As the sector is oriented towards an office area, the weekdays have a traffic peak during the day, whereas on weekends the traffic pattern resembles more that of a residential area with the peak occurring closer to midnight. While the patterns between different sectors may vary, the three cells within a sector usually exhibit similar patterns.

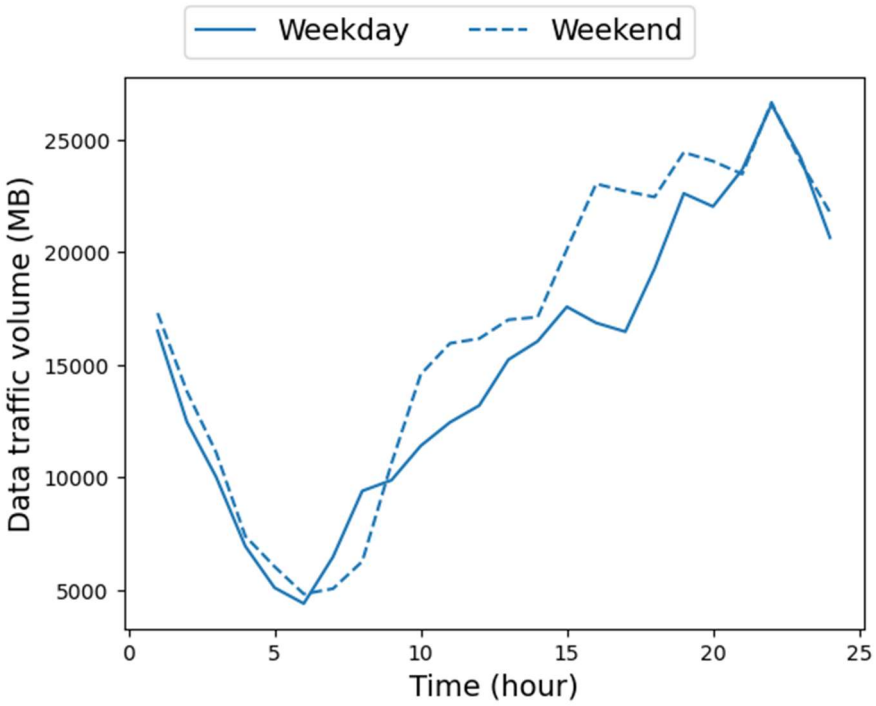


Figure 19: Average weekday and weekend traffic patterns in sector 1 of the training BS.

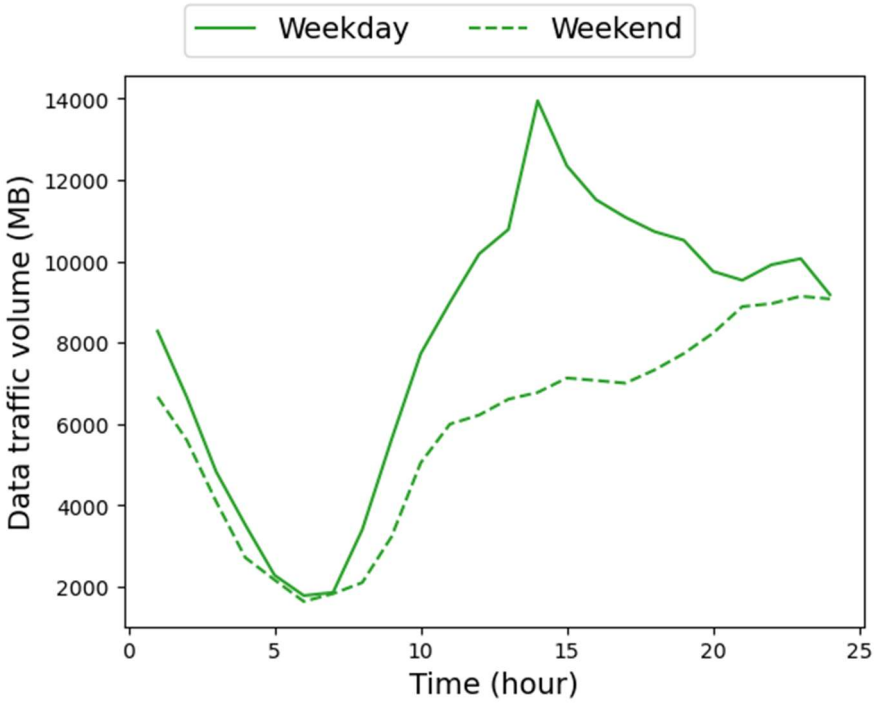


Figure 20: Average weekday and weekend traffic patterns in sector 3 of the training BS.

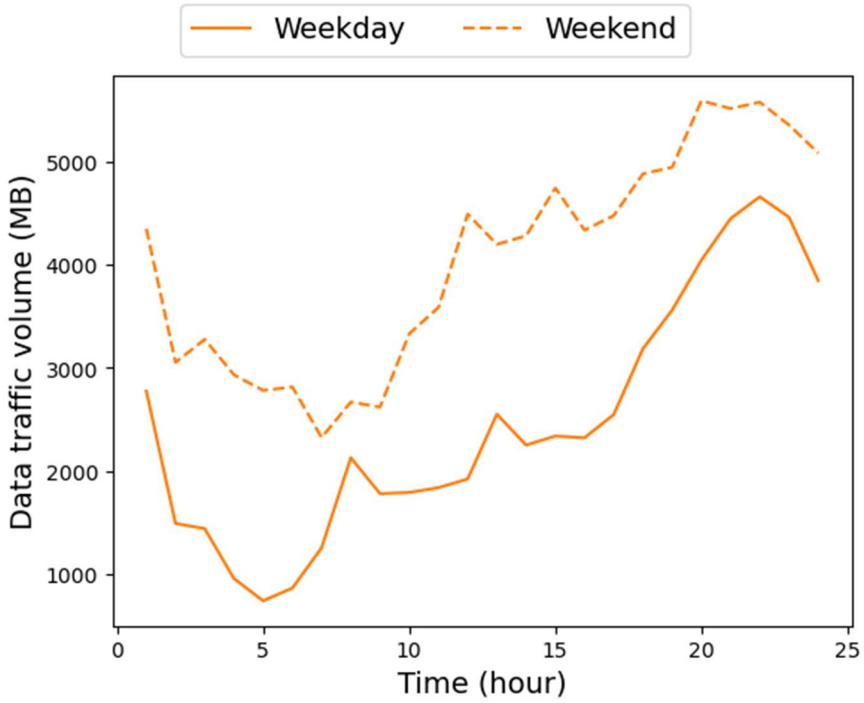


Figure 21: Average weekday and weekend traffic patterns in sector 2 of the training BS.

The variations between weekdays and weekends can also be observed from Figure 22. In addition, the figure provides a rationale for grouping the individual cells within each sector by aggregating their values. This is based on the fact that even though cells are using different frequency bands, they are still facing the same direction, and are therefore affected by the same traffic patterns. Even though the traffic patterns seem to be relatively similar, there are some differences between cells, including those in traffic volume, which can stem from the network's configuration. Various frequencies, for instance, might be allocated to different types of services and ranges. However, grouping the frequency bands within each sector simplifies matters by reducing the number of variables, making the model outlined in the subsequent sections more manageable.

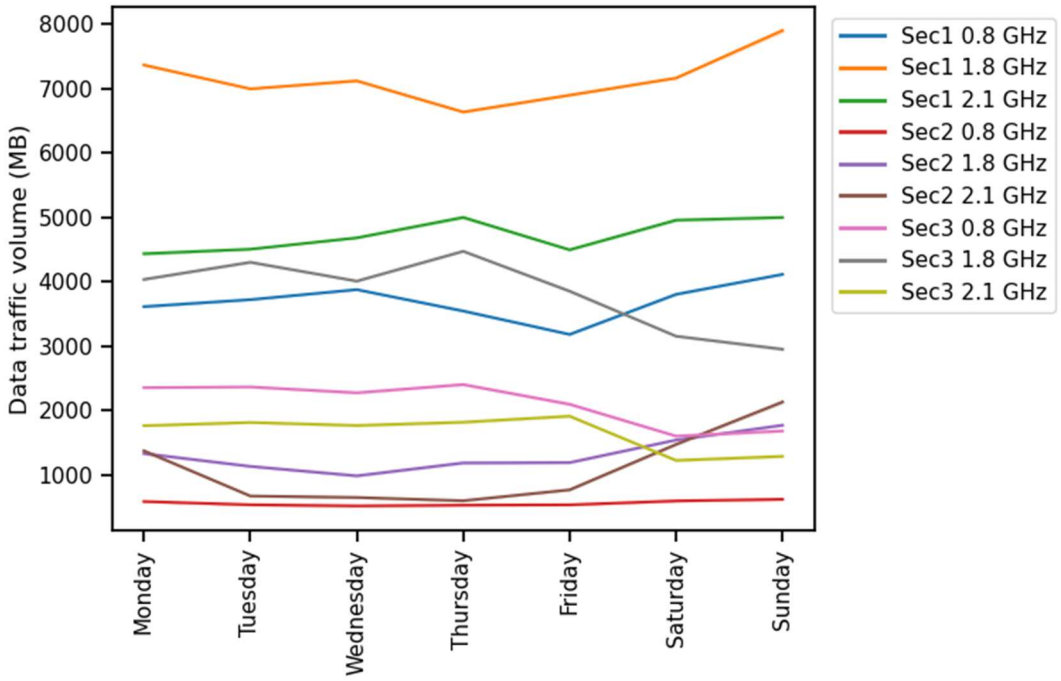


Figure 22: Average daily *TVD* of different cells of the training BS.

The key takeaways from the data analysis are the factors that significantly affect the design of the developed model. To summarise the data analysis part, the dataset can be condensed to a sector-level analysis instead of a more granular cell-level examination. Reducing the dataset any further is not feasible, given the differences in traffic patterns and KPI correlations between different sectors. Finally, the data displays large variations in patterns between weekdays and weekends.

3.3 Methodology: Anomaly detection and RCA

This subsection presents the main idea and initial considerations behind the developed model and describes the individual steps that need to be taken to build it.

3.3.1 Initial Considerations

The anomaly detection and root cause analysis system for unlabelled mobile network data consists of data preprocessing and unsupervised learning methods, DBSCAN [68] and LSTM autoencoder [69]. The system pipeline is depicted in Figure 23.

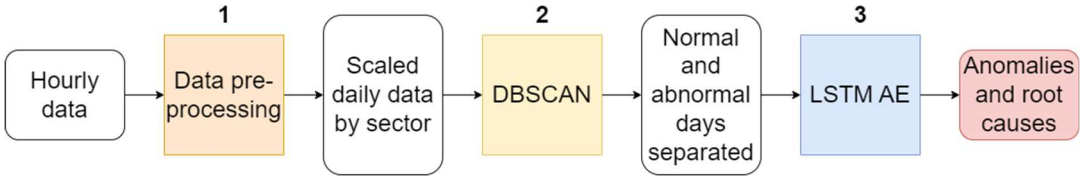


Figure 23: Anomaly detection and RCA system pipeline for unlabelled mobile network data.

The core idea is to train an LSTM autoencoder model to identify normal network behaviour. By using a labelled dataset, where normal traffic is already separated from anomalies, the training process would be very straightforward, but as the dataset in use is unlabelled, an alternative approach of separating normal data from anomalous data is required.

DBSCAN is used to separate full days (24 hours) of network data into normal and abnormal clusters, resulting in a set of days containing only typical traffic patterns. The days where the traffic pattern is atypical are considered abnormal. The abnormality in the traffic pattern can be the result of network issues, but it can also be caused by natural human behaviour, such as holidays or events, which are outside the scope of this work. For this reason, DBSCAN is not used as the final anomaly detector, but rather a step to obtain a set of normal days, which are then used to train the LSTM AE.

In addition to anomaly detection, the LSTM AE model is used also in automatic RCA. The KPI-wise reconstruction errors generated by the AE, are used to indicate the most probable root causes of each anomaly. For clarity, the term abnormal day is used to refer to the atypical days from DBSCAN, and the term anomalous day is used to refer to the days flagged by LSTM AE.

The work has been conducted using the Python programming language and its libraries. A list of the main libraries and software along with their versions is available in Table 6.

Table 6: Main libraries used.

Software	Version	Usage
Python	3.11.2	Programming language
Pandas	1.5.3	Data structure and data processing
Keras	2.12.0	LSTM AE
Tensorflow	2.12.0	Used in Keras
Scikit-learn	1.2.2	DBSCAN, NearestNeighbors and StandardScaler
Numpy	1.23.5	Mathematical operations and data structures
Kneed	0.8.2	Locating the knee point for DBSCAN
Seaborn	0.12.2	Visualisation
Matplotlib	3.7.1	Visualisation

3.3.2 Data preprocessing

Data preprocessing is an essential part of any machine learning pipeline to guarantee that the data used is clean and consistent for training models. Outlier removal is often a step performed in the preprocessing part, but with anomaly detection, outliers are the points of interest and therefore they should not be removed. The preprocessing of the data consists of the following three steps:

1. Data sampling:

   The design choices for data sampling are based on the data analysis part. Each base station is divided into three sectors. A separate dataset is created for each sector where the hourly KPIs are now aggregated from the three cells within the sector. The hourly data is divided into individual days: 24 hours from 01:00-24:00.

2. Data filtering:

   In the filtering part, if a day is not full, for example, if it is missing some hours of data, it will be filtered out because missing hours proved to have a large effect in the performance of the autoencoder. The resulting dataset contains only full days of mobile network KPIs, meaning that they include 24 hourly values of each KPI.

3. Data scaling:

   In the scaling phase each KPI is scaled individually using standard scaling [70], where the scaled value for each data point is calculated using:

$$z = \frac{x - \mu}{\sigma}, \quad (3)$$

where x is the original value, μ is the mean and σ is the standard deviation of the KPI. As a result, each KPI has zero mean and standard deviation of 1 across all the datasets. Scaling data is important as different KPIs have values in highly different magnitudes.

3.3.3 DBSCAN

In most of the related work, as discussed in Section 2.7, there is already a dataset containing only normal samples, which can directly be used to train the AE. However, since the data used in this work consists of both normal and abnormal days within the same dataset, it is essential to obtain the set of normal days by separating it from the abnormal one before training the AE.

K-Means clustering is a very commonly used clustering algorithm useful in many problems, but as the abnormal days can be very different from each other, K-Means would not place them in the same cluster, resulting potentially in multiple clusters that are difficult to interpret. DBSCAN however, as illustrated in Figure 24, places outliers (orange data points in the highlighted scenario) into the same cluster even if they are not very similar to each other. The result is a clear distinction between outliers and normal data points, which is exactly what is needed in order to train the AE.

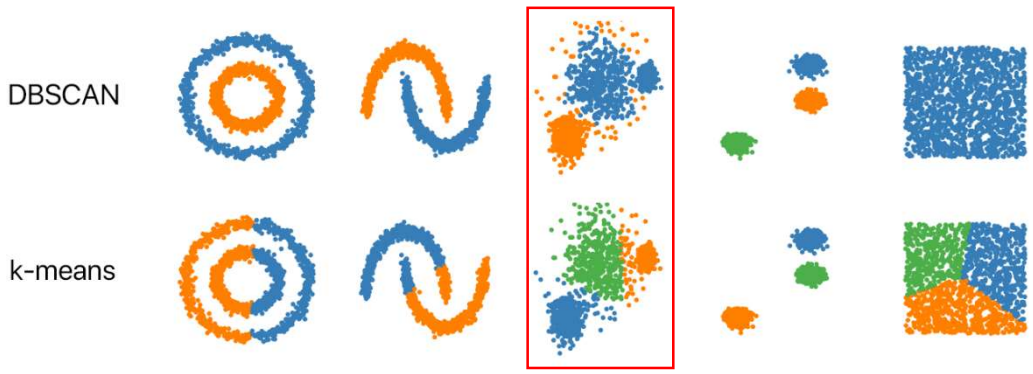


Figure 24: DBSCAN vs K-Means [71].

DBSCAN was selected due to this property as well as the fact that it is a clustering algorithm that does not require the number of clusters predefined as was mentioned in Section 2.4.4. Instead, it creates a cluster and assigns a data point and its neighbours into the cluster if the data point has at least a certain number of neighbours (*MinPts*) within a certain radius (*Eps*). If it does not, the data point is considered an outlier, or as in this case, abnormal. The pseudo code of the DBSCAN algorithm explaining the clustering process in more detail can be found in Algorithm 1. *MinPts* and *Eps* are defined by the user before clustering.

In this work, the separation of normal days from abnormal ones is based on a single KPI, traffic volume data (*TVD*). *TVD* was chosen for two reasons. Firstly, it forms a consistent pattern in most of the days and these patterns have been studied a great deal in the literature as characteristic traffic patterns of different geographical areas. To ensure the accuracy of the clustering, the patterns of the days considered normal, are compared to those from the literature by using correlation analysis and visual inspection.

```
DBSCAN (W, MinPts,Eps)
Input: a data set W, MinPts
Output: arbitrary shape clusters
for each data point p ∈ W do
    if p is not mark as 'seen' then
        Mark p as 'seen'
        Find neighborhood of data point p, NeighborPts
        if NeighborPts < minimum points then
            Mark data point as a noise
            else
                clusterid=clusterid+1
            end
        end
        for all q ∈ NeighborPts do
            Mark data points q as seen
            Find neighborhood of data point q, NeighborPts
            if NeighborPts > minimum points then
                Give data point q a clusterid
            end
        end
    end
end
```

Algorithm 1: Pseudo code of the DBSCAN algorithm [72].

Secondly, when experimenting DBSCAN with multiple KPIs in comparison to only *TVD*, a considerably larger portion of data was labelled abnormal. An explanation for this phenomenon could be the curse of dimensionality, which increases the distance between individual data points. It is further confirmed in [73], that high dimensional data makes DBSCAN unstable. In addition, it is stated by [60] that anomalies in mobile networks are rare. Therefore, to avoid classifying a significant amount of the normal data as abnormal, only *TVD* was used. As a result, the rest of the KPIs are not considered in the clustering phase.

The clustering is performed separately for weekdays and weekends because, as seen in the data analysis part in Section 3.2, there might be considerable variation between the two. In the case that the clustering was done for weekdays and weekends combined, the normal days could be placed in one or two clusters depending on how similar the traffic patterns of weekdays and weekends are to each other. By clustering weekdays and weekends separately, the system is made sure to find a single cluster containing the normal days and therefore the interpretability of the results increases.

Selecting the values for parameters *MinPts* and *Eps* plays a pivotal role in the clustering result. The selection of parameters was performed by using existing practices found in the literature as well as experimenting with different values. The value of *MinPts* determines how many data points are required to form a cluster. A very large value can result in absence of any meaningful clusters whereas the value of 1 leads to every data point forming its own cluster. An appropriate value was determined through a process of experimentation with the training BS, as well as correlation analysis elaborated upon in Section 4.1. In addition, the value of *MinPts* commonly depends on the number of features [74]. Specifically, the value of *MinPts* was set to the total number of hours in a day, which is also the number of features, totalling 24. The input data for the clustering is of shape $n \times 1 \times 24$, where n is the total number of days to be clustered, and therefore, the number of features is 24.

Selecting the value for *Eps* is slightly more complex. The value was set to be the knee point of the sorted distances to the 5th nearest neighbour of each data point. This type of approach of using the knee point is common in the DBSCAN literature and is used for example in [74]. The 5th nearest neighbour was selected via trial and error and it proved to be a good compromise between a too short radius where significantly more days would be considered abnormal, and a too long radius where next to no abnormalities were found. The distance metric for the nearest neighbours is Euclidean distance and the knee point is calculated using the KneeLocator function from Python's Kneed package. The knee point should be the point where a curve of sorted values experiences a significant change. The knee point, which is also the *Eps* value used for clustering the weekdays of sector 1 of the training BS, is 3.33 and is depicted in red in Figure 25.

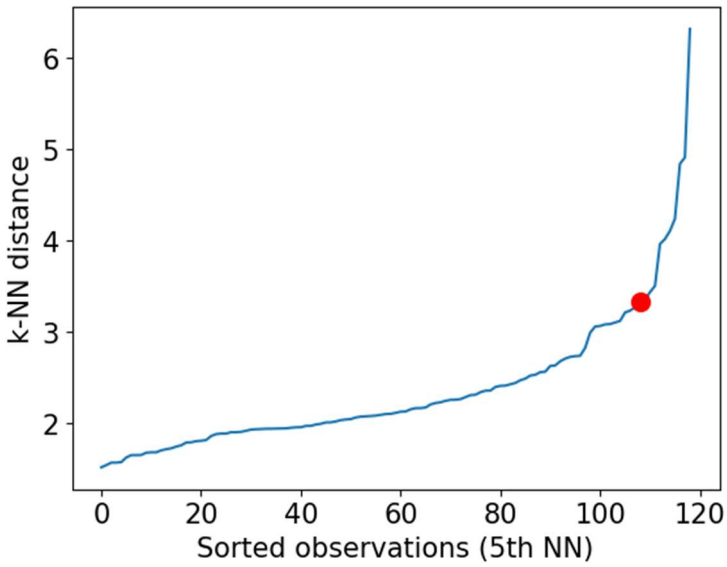


Figure 25: Knee point of weekdays of sector 1 of the training BS.

Resulting from the DBSCAN clustering is a set of labels, which in this work only contained two different values, normal and abnormal. The DBSCAN algorithm used in this work is part of Sklearn's clustering package and it assigns the label -1 to the abnormal cluster. An example of a normal day in contrast to abnormal day is displayed in Figure 26. It is noteworthy that the traffic pattern of the normal day resembles closely the average pattern of sector 1 as seen in Figure 19, and the pattern of the abnormal day seems to be unexpected, considering the dominant residential traffic profile in sector 1.

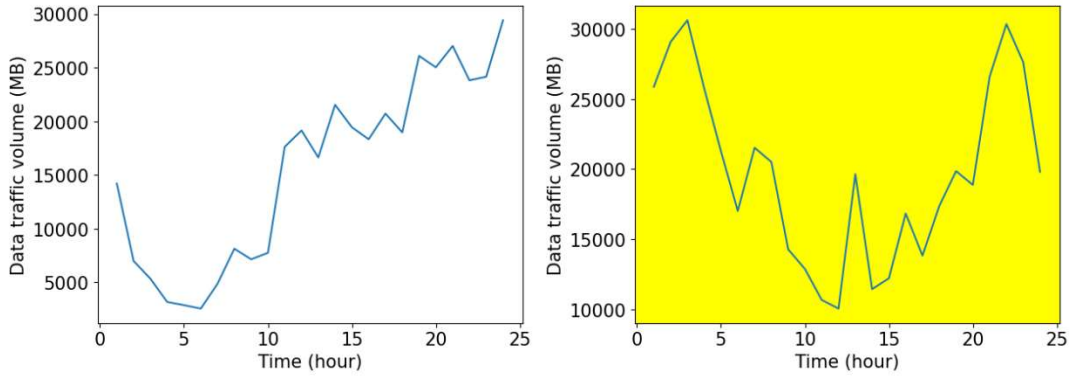


Figure 26: Normal day (a) and abnormal day (b) in sector 1 in the training BS based on *TVD*.

3.3.4 LSTM AE

As discussed in Section 2.4.5, autoencoders consist of two parts, an encoder and a decoder. They can be used to learn the normal behaviour of mobile networks by letting them learn how to first encode the original normal data into a lower-dimensional feature space and then how to decode the lower-dimensional representation to match the original data as well as possible.

Following is the general definition of an autoencoder by [56]. Let $X = R^D$ be the input space and F the feature space. An encoder is a function $\phi : X \to F$, that encodes inputs into the feature space F, while a decoder is a function $\psi : F \to X$, that tries to reconstruct the original input from the encoded representation in the feature space F. An autoencoder $\Phi_{AE} : \phi \circ \psi$, can be denoted as $\Phi_{AE}(x(n)) = \hat{x}(n)$, where $x(n)$ is the original sequence and $\hat{x}(n)$ is the reconstructed sequence. The difference between the original and the reconstructed sequences is called the reconstruction error [56]. The reconstruction error can be used to detect an anomaly as it should be low for normal data and high for anomalous data, but in this work, it is also used to determine the anomalies' root causes.

AEs which consider the temporal nature of the data (such as AEs with LSTM layers), are commonly used in the related work. In this work, LSTM-

layers are included both in the encoder and the decoder, because the AE is designed to learn the behaviour of individual days, which consist of consecutive time steps. LSTM layers are especially suitable for capturing the temporal dependency in sequential data, while having a "longer memory" than traditional RNNs and solving the vanishing gradient problem [38]. The architecture of the LSTM AE neural network can be seen in Table 7 and the three main parts of it are explained below:

Table 7: LSTM AE model architecture.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	18432
repeat_vector (RepeatVector)	(None, 24, 64)	0
lstm_1 (LSTM)	(None, 24, 64)	33024
time_distributed (TimeDistributed)	(None, 24, 7)	445
Total params	51911	
Trainable params	51911	
Non-trainable params	0	

- Encoder:

  A single LSTM layer consisting of 64 units. The number of layers and their sizes typically depend on the complexity and the amount of data available for training. The goal here is to learn the daily pattern of different mobile network KPIs. Even though there can be multiple different KPIs behaving differently throughout the day, the patterns of individual KPIs especially within a single sector are not very complex. Therefore, a single layer with 64 units proved to be sufficient. The activation function used is Rectified Linear Unit (ReLU) in order to include non-linearity in the network. The input shape for this layer is (24, *n_KPIs*), where 24 is the number of hours in a day and *n_KPIs* is the number of different KPIs in the data.

- RepeatVector:

  The RepeatVector layer is the connection between the encoder and the decoder, and it modifies the shape of the output of the encoder to be inputted to the decoder. The parameter used for the layer is 24, which means that the encoded representation is applied for each timestep of the following decoder.

- Decoder:

  The decoder is responsible for reconstructing the encoded data to match the original input. It consists of a single LSTM-layer and a time distributed dense-layer. Just like the LSTM-layer in the encoder, the

LSTM-layer of the decoder has 64 units and uses ReLU as the activation function. The network is a sequence-to-sequence model, and therefore it needs to output a sequence. Therefore, the layer has the *return_sequences*-parameter set to True. Lastly, the decoder has a TimeDistributed Dense layer with the parameter of *n_KPIs* to match the dimensionality of the original input data.

The model is compiled using the Adaptive Moment Estimation (Adam) optimiser and Mean Squared Error (MSE) is used as the loss function. Adam optimiser is a widely used one for training neural networks in various domains and it proved to converge significantly faster than for example Stochastic Gradient Descent (SGD) for the used dataset. The learning rate used is 0.001, which is the default for Adam optimiser. As the name suggests, MSE calculates the mean of the square of the errors. By squaring the errors, MSE amplifies larger errors, causing anomalies to become more prominent and easily identifiable. The MSEs [75] are calculated using:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\,, \qquad (4)$$

where n is the number of samples in the dataset, y_i is the actual value of the i-th sample of the target variable and $\hat{y}_i$ is the predicted value. To construct the LSTM-AE network, Keras was used, which is an API built on top of the TensorFlow-platform and is very straightforward to use.

The normal days identified by the DBSCAN clustering are divided into training and test sets with 90 and 10 % of the normal days, respectively. The training set is used to train the LSTM AE and the test set is used to observe its results. Because different sectors of different BSs can be very different from each other, the model is trained individually for each of the three sectors of every BS. The training dataset for each sector of each base station is of shape (*n_days*, 24, 7), where *n_days* is the number of days in the sector, 24 is the number of hours in a day and 7 is the number of potential root cause KPIs in the dataset. The model iterates through the training dataset 30 times, by setting the *epochs*-parameter of the fit-method to 30. This number of iterations was observed to be enough for the model to converge while keeping the computation time relatively low and avoiding overfitting. The convergence criterion was to monitor the reduction in the reconstruction error and to stop when it stabilised. A similar criterion is used for example in [76].

As data was separated in the clustering phase only by using the *TVD*-KPI, detecting anomalies on that KPI would point out mostly the same abnormal days as the DBSCAN, and therefore, would not add much value. Instead, anomaly detection is performed using all the KPIs besides *TVD* to detect anomalies rooting from network errors.

A day is considered an anomaly if its reconstruction error is higher than the 95th percentile of the reconstruction errors of the training set. In the majority of the work where AEs are used, the threshold is equal to the highest reconstruction error of normal data, but it should also be considered that these works utilise a labelled dataset. In this work however, the data was mixed in the beginning, and the normal set was only acquired by DBSCAN clustering. The division between normal and abnormal days is based on data density and not the ground truth, and therefore might not be definite. For this reason, the 95th percentile is used to account for any high reconstruction errors that could be present within the normal days, while not being low enough to cause misclassifications of normal days leading into false alarms.

Instead of computing a single reconstruction error for a day as is done in the anomaly detection, the reconstruction error can also be computed individually for each KPI of the anomalous days. The KPI-wise errors are then used to help determine the root cause of the anomaly in the sense that the KPIs with the highest errors are the potential root causes. In this work, if an individual KPI has an MSE larger than the mean of all the KPI-wise errors, it is considered a potential root cause. The rationale behind the definition of a root cause is based on trying to find root causes that increase the total MSE. Also, setting definite thresholds for the MSE could be problematic, considering the variance between different sectors.

Consequently, the set of KPIs does not include all KPIs in the data, but only the potential root causes of the anomaly. For instance, most throughput KPIs (*USER_DL_TP, USER_UL_TP, CELL_DL_TP, CELL_UL_TP)* can be considered more of a consequence of *TVD* rather than the cause. As the analysis is done on the sector level and data is hourly aggregated, Timing advance (*TIME_ADV*) only tells the average distance of users to the base station, and therefore would not reveal very interesting root causes. The number if the KPIs used in the system is eight. The list of KPIs and the part of the system where they are used is presented in Table 8.

Table 8: Final set of KPIs and where they are used in.

Abbreviation	KPI	Used in
TVD	Traffic volume data.	DBSCAN
USERS	Average number of users.	LSTM AE
PRB_DL	PRB DL average usage rate.	LSTM AE
PRB_UL	PRB UL average usage rate.	LSTM AE
HSR_INTRA	HSR intra frequency.	LSTM AE
HSR_INTER	HSR inter frequency.	LSTM AE
CQI	Average CQI.	LSTM AE
CA_TP	Carrier aggregation throughput.	LSTM AE

4 Results and discussion

This chapter presents and discusses the outcomes achieved by applying the developed system on the dataset featured in Section 3.1. This chapter is divided into six subsections. It starts with the specific results from using DBSCAN to differentiate normal from abnormal days. Following this, the effectiveness of LSTM AE in anomaly detection and RCA is presented, along with a section devoted to validating these outcomes. An analysis of the overall findings, including the number and nature of the finally identified anomalies is then provided. The final part critically considers the choices made, as well as alternative approaches.

4.1 Distinguishing normal from abnormal days

The performance of machine learning models is usually measured by comparing the real values of the target variable to the ones generated by the model. However, the original dataset does not have any labels or ground truth indicating the real anomalies, therefore comparing results in terms of precision, F1-score or any other commonly used metric is not possible.

Instead, some idea of the correctness of the DBSCAN clustering can be gained by comparing results with the correlations of each day to the average day. Here the *TVD* of each weekday (Mon – Fri) is compared to the average *TVD* of all weekdays, while weekends (Sat – Sun) are compared to the average weekend. The correlation between an individual day and the average day is high if they have a similar shape. As the average traffic patterns resemble the ones identified in the existing literature and the correlations can reveal the days that do not follow the pattern, this is a way to verify the validity of the clustering.

Figure 27 portrays the results of DBSCAN clustering in sector 1 of a base station called VIANA_SUL_LTE_NVC103B3. The yellow background depicts abnormal days and the white one normal days. The correlation values are depicted above each day. The red curve displays the original values of *TVD*, while the blue curve displays the smoothened values by a moving average of three hours. The x-axis is the time of the day, and the y-axis is the *TVD* value in gigabytes (GB).

So, why are the correlations not used to separate the abnormal days from the normal days in the first place? Instead, why is the computationally heavier DBSCAN used? The reason for this is that different sectors behave differently, and the daily pattern might vary more in certain sectors than in others. This makes it difficult to set a threshold for the correlation of a day to be considered normal. By using the density based DBSCAN, there is no need to set any thresholds.

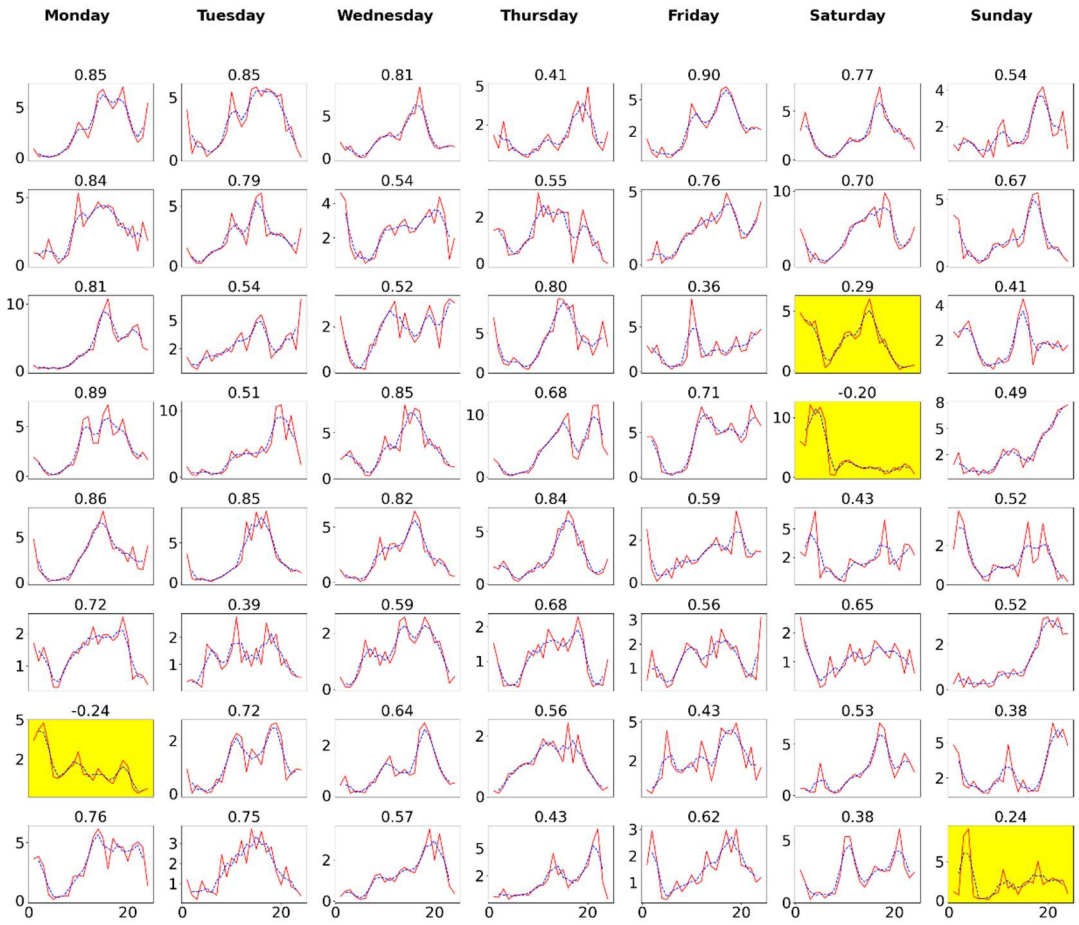


Figure 27: Sample of DBSCAN clustering result of days based on *TVD*.

In Figure 27, the clustering of days of sector 1 demonstrates promising results, in the sense that the abnormal days from the clustering also have the lowest correlations. This was also further noticed in the rest of the data for different BSs and sectors, in the cases when the dominant traffic patterns were clear and resembled the ones mentioned in Section 3.2. On the other hand, when the dominant traffic pattern was not clear and the individual days displayed varied patterns, the clustering and verifying its results were more difficult. This resulted in clustering outcomes where the normal cluster included days with lower correlation than the abnormal cluster. This would suggest that either the clustering, the verification method, or most likely both, may not be optimal for all situations.

The fact that data is sampled by the hour means that there can be significant variation in the amount of traffic even between consecutive hours. Also, if the traffic pattern of the sector is mixed and there is not one dominant pattern, it could mean that the average day is not descriptive of all days. These are challenging factors for DBSCAN but could be overcome by more accurate and uniform data.

Nevertheless, the DBSCAN clustering shows promise in being applied to an anomaly detection system for unlabelled data, not as the final anomaly detector, but rather a preprocessing step. After performing this step, the user should have at least some idea of what the normal data could be, or where the anomalies are more likely to be located in.

4.2 Anomaly detection

Let us inspect the results of the LSTM AE in sector 1 of base station VIANA_SUL_LTE_NVC103B3. The reconstruction errors of days in the training, test and abnormal sets are displayed in Figure 28. The training set (blue colour) is comprised of 90 % of the normal days from DBSCAN and was used to train the LSTM AE. The test set (green colour) comprises of the remaining 10 % of the normal days and the abnormal set (orange colour) includes the abnormal days from DBSCAN.

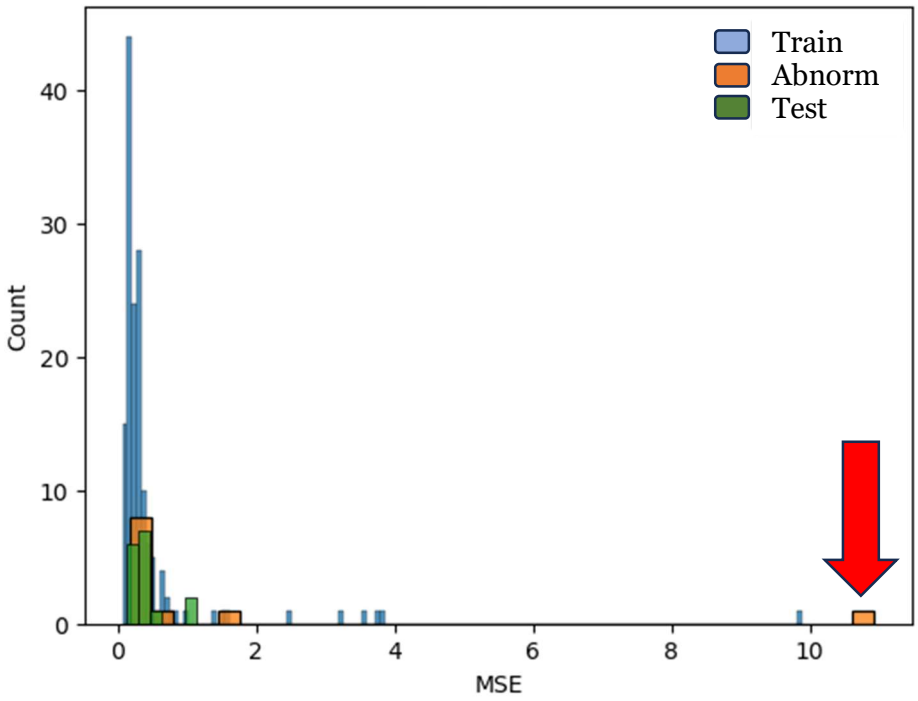


Figure 28: Histogram of reconstruction errors of training, test and abnormal in VIANA_SUL_LTE_NVC103B3, sector 1.

There is a single abnormal day, which has a reconstruction error much larger than the other abnormalities and is indicated with the red arrow. Since the AE is able to reconstruct the other abnormal days with an error even lower than some of the normal days, these days could be the ones where the abnormal traffic pattern is due to natural human behaviour. The abnormal

day, with an MSE over 10, can be considered an anomaly deriving from a problem in the network.

With visual inspection, it seems clear that the one abnormal day with the large reconstruction error is an anomaly. It was also classified as an anomaly by the LSTM AE, which calculated the 95th percentile of training MSEs to be 1.76. This means that out of all the 11 abnormal days from DBSCAN, only one is an actual anomaly deriving from unexpected behaviour in the network and is detected due to unpredictability in the remaining KPIs, not *TVD*.

In Figure 28, there is no clear distinction in the MSEs among training, test and abnormal sets. A similar behaviour was inspected also in different sectors of other BSs. In addition to the anomaly, there are multiple days in the normal training set which also have a large reconstruction error. One of them even almost as large as the anomaly. This would strongly indicate that there exist anomalies also within the normal days and that network errors do not always display abnormal traffic volume pattern. These anomalies do not display abnormal behaviour in the *TVD*-KPI but seem to show abnormal behaviour in some other KPIs regardless.

When it comes to the performance of the LSTM AE, considering that most of the days, be it in the train, test or even in the abnormal set, are within the 95 percentile of the training MSEs (1.76), it seems that the LSTM AE is able to quite accurately reconstruct the data. What is considered a small MSE depends on the data scale, but in this case, after standard scaling, most of the KPIs have a range of values of about 10 units. Comparing the MSE of 1.76 to that range, the error is not particularly high. This would indicate that the LSTM AE is learning the patterns of the majority of the days and is able to reconstruct them. In addition, the few days with significantly larger MSEs indicate that the LSTM AE is not able to reconstruct all the days, making it suitable for anomaly detection, as it is sensitive to data patterns that differ significantly from the typical or expected patterns.

4.3 Root cause analysis

As described in Section 3.3.4, the LSTM AE can be used for RCA by inspecting the reconstruction errors computed individually for each KPI. Figure 29 shows the reconstruction errors of each of the 11 days labelled as abnormal by the DBSCAN, as well as the KPI-wise reconstruction errors. The x-axis displays the average MSE of the full day considering all KPIs presented in the figure, whereas the y-axis displays the MSEs of individual KPIs. In this case, it seems that the abnormal day that was detected as an anomaly was caused by a problem in handovers, as it can be seen from the high MSE of both inter frequency (*HSR_INTER*) as well as intra frequency (*HSR_INTRA*) handover success rates.

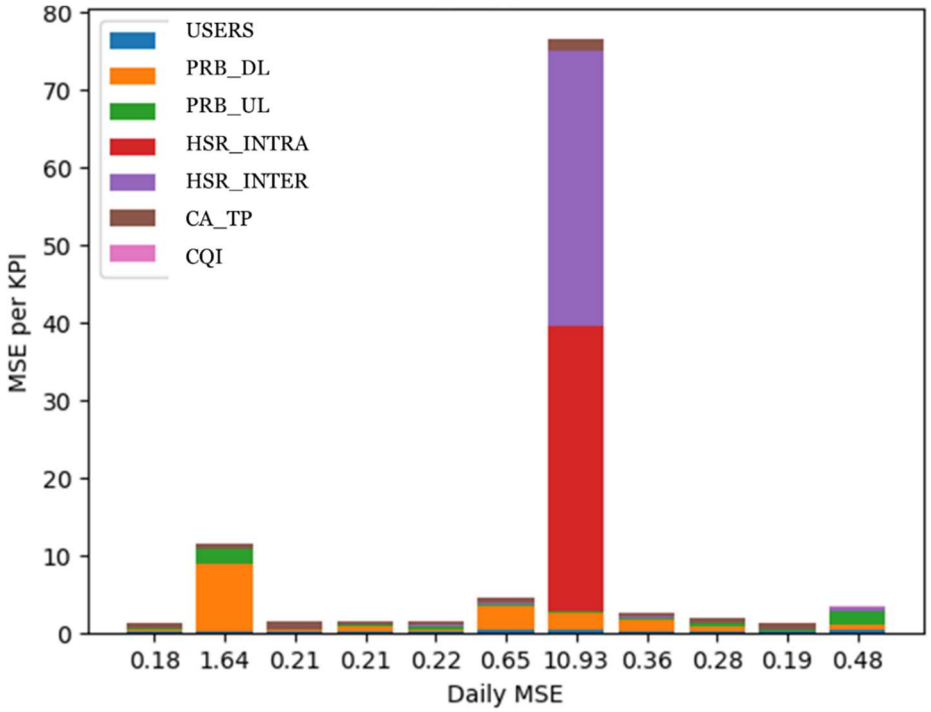


Figure 29: KPI-wise reconstruction errors of the abnormal days in sector 1 of VIANA_SUL_LTE_NVC103B3.

When plotting the scaled *HSR_INTER* pattern of all abnormal days, in Figure 30, it can be noted that all days are quite similar to each other except for one. Abnormal day number six seems to have lots of very small values during the night and morning, and is in fact, the day that was detected as an anomaly. Because the KPI in question is *HSR_INTER,* which has a range of 0 – 100 before scaling, it looks like the success rates of inter frequency handovers have been zero across the period of 00:00 – 10:00. This is a serious network problem, as it means that calls or data sessions have not successfully been transferred from one cell to another from a different frequency. This leads to dropped calls as well as disruptions in data streams. Another option is that handovers were fine, but some of the counters that keep track of the KPI have been inoperable. Nevertheless, in both scenarios the network operator should be alerted so that adequate actions can be taken to fix the issue.

Figure 31 displays the reconstructed *HSR_INTER*s of all abnormal days. The AE was able to reconstruct most of the *HSR_INTER* curves to closely match the original ones, but the reconstructed anomalous day is very different from the original in Figure 30. This is most likely due to the training set not containing enough samples like the anomaly that would have allowed the LSTM AE to learn the reconstruction of similar data. Instead, it seems that the AE tried applying a similar shape to the anomaly as to the other abnormal days. This led to a large reconstruction error and the detection of the anomaly.

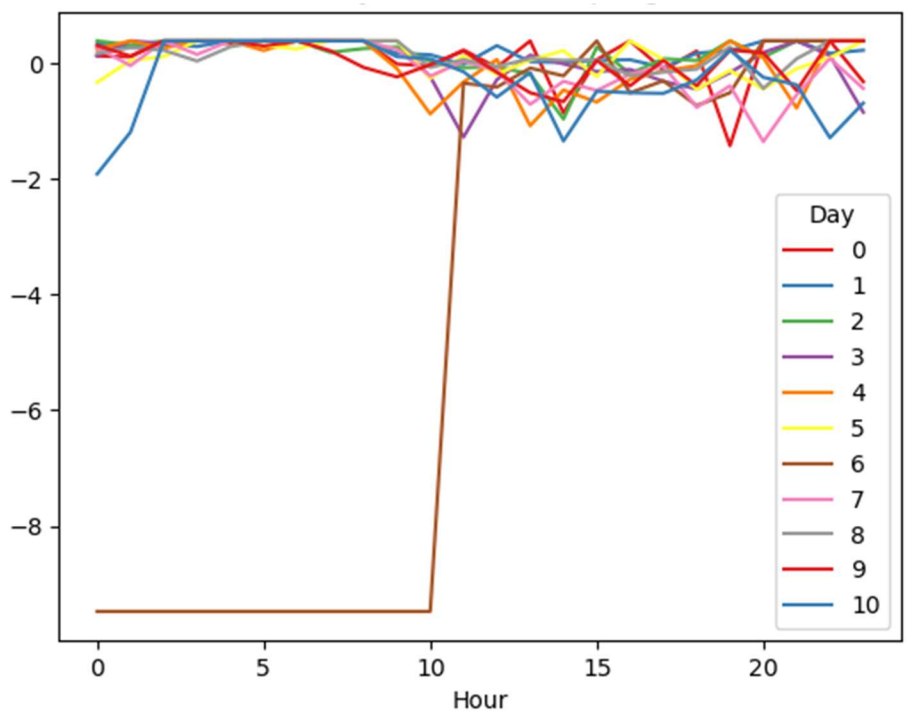


Figure 30: Original scaled inter frequency handover success rates of abnormal days in sector 1 of VIANA_SUL_LTE_NVC103B3.

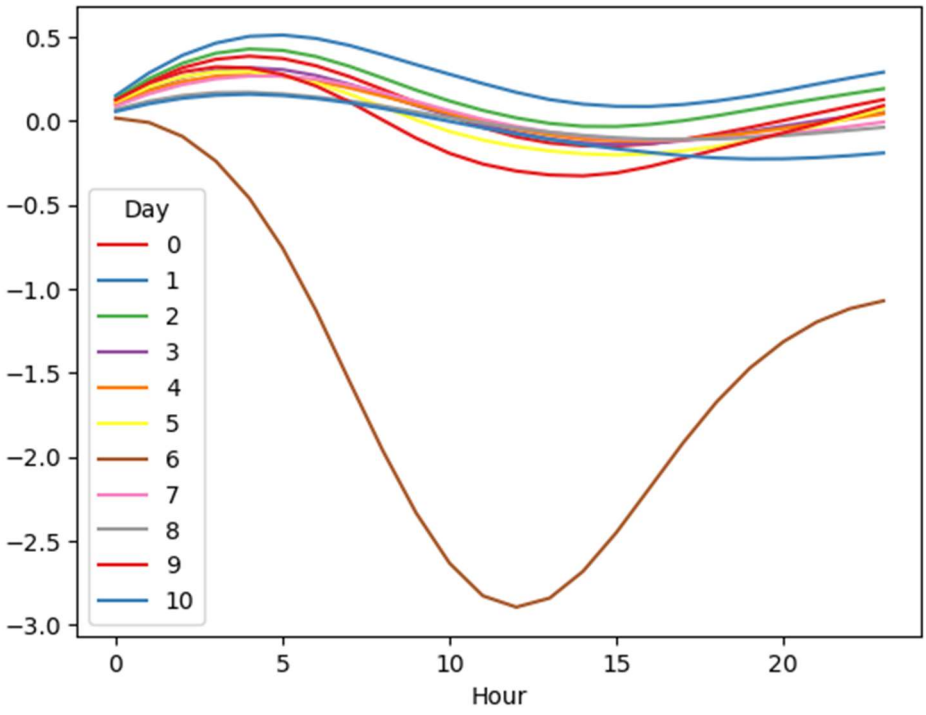


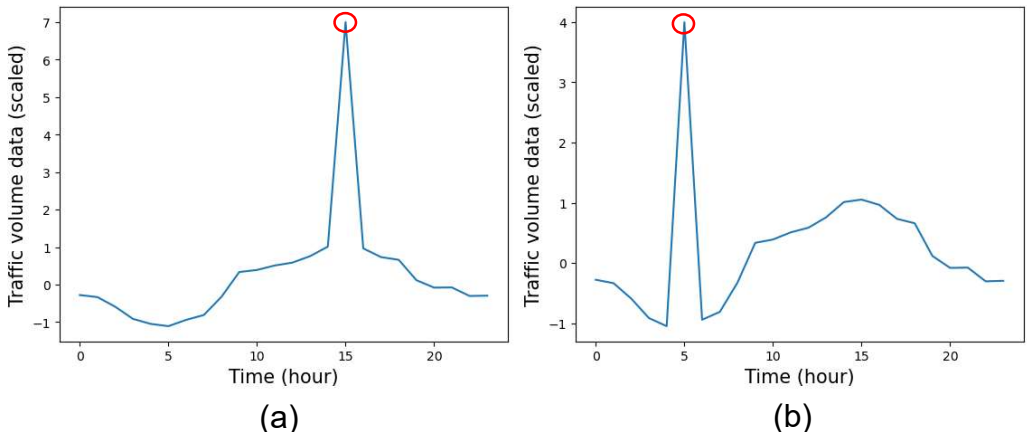
Figure 31: Reconstructed inter frequency handover success rates of anomalous days in sector 1 of VIANA_SUL_LTE_NVC103B3.

4.4 Verifying results

Because there is no labelled dataset that could be used to gain some score to measure the performance of the system, there is essentially no way of knowing if the tools are used properly and if the system does what it is supposed to. To mitigate this challenge, the normal weekdays were taken from the existing data and manually created artificial anomalous data was introduced into the dataset. The normal days here being the ones that were labelled normal by the DBSCAN and received a reconstruction error of less than 0.5 from the LSTM AE. The anomalous data was created by taking the average of all the normal days and distorting it in multiple different ways, such as inserting point anomalies and zero values as well as reversing the data to create a contextual anomaly. Results help to verify the correct operation of the system, but they also provide insight into what is anomalous enough to be considered an anomaly by the system.

- Point anomalies

  The sensitivity of the system to point anomalies was tested by taking the average of all normal days and creating a new day by inserting large values into the average day. The DBSCAN clustering phase successfully clustered each of the normal days as well as the average day into the same cluster, but the days containing large enough point anomalies were clustered into the abnormal cluster. Point anomalies were inserted into the lowest point of *TVD* (5:00) as well as into the highest point (15:00). Figure 32 displays the minimum values of the point anomalies in order for the day to be clustered into the abnormal cluster. Noteworthy is that during the peak hour of traffic volume, the point anomaly needed to be much higher than during the night.



(a) (b)

Figure 32: Point anomalies at 15.00 (a) and at 5.00 (b) clustered in the abnormal cluster by DBSCAN.

The LSTM AE on the other hand considers seven KPIs, so point anomalies in single KPIs were not as easily detected. Instead, if occurring only in one of the KPIs, an anomaly of about three times larger than the maximum value of the KPI in the training set was sufficient in most cases to be classified as an anomaly. Here, it did not make much difference whether the point anomaly happened during the time when the KPI would normally have larger values or during the time when the values are normally lower.

- Zero values

  Unlike point anomalies, a single zero value was not enough to cause the DBSCAN or the LSTM AE to detect an anomaly. Instead, multiple consecutive zeros were required to be inserted into to the average day. Figure 33 displays the minimum number of zeros to be clustered in the abnormal cluster during day and night. It can be noted that if inserted during the time when *TVD* is low on average, the number of zeros required was as high as 11, whereas, during the peak hours of *TVD* at around 15:00, three zero values were enough to cluster the day into the abnormal cluster. One of the reasons for this large difference between the two is that even though there are many zeros inserted during the night, the shape of the curve still remotely resembles the shape of the average day.

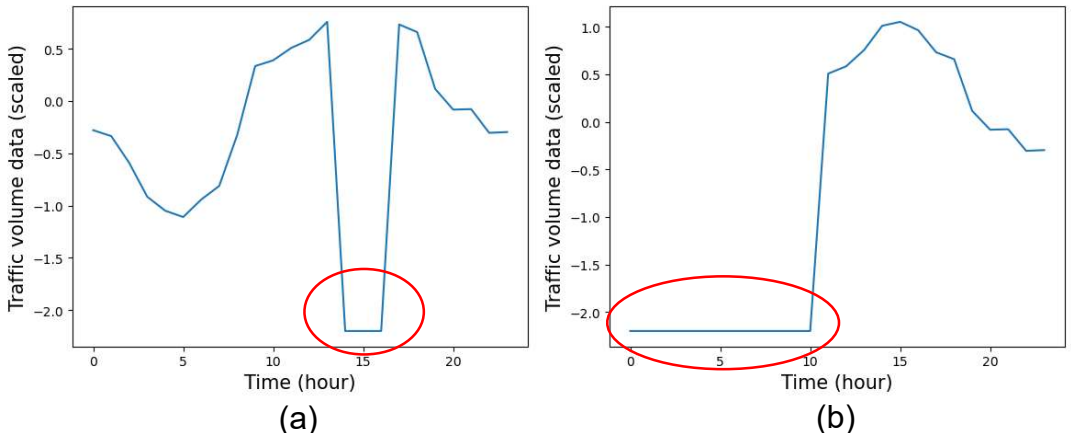


Figure 33: Zero value anomalies during day (a) and during night (b) clustered in the abnormal cluster by the DBSCAN.

The AE again was not very sensitive to anomalies in single KPIs. However, when inserting zero values into each of the seven KPIs, it turned out that only two consecutive hours of zero values were enough to detect an anomaly regardless of the time of the day.

- Contextual anomalies

  The average day of *TVD* was reversed to create a contextual anomaly where the values of the time series are within the normal range of values, but the traffic peak happens during night, and the low point during day. Consequently, DBSCAN clustering failed to classify the reversed day as abnormal. The average day and the reversed day are displayed in Figure 34.

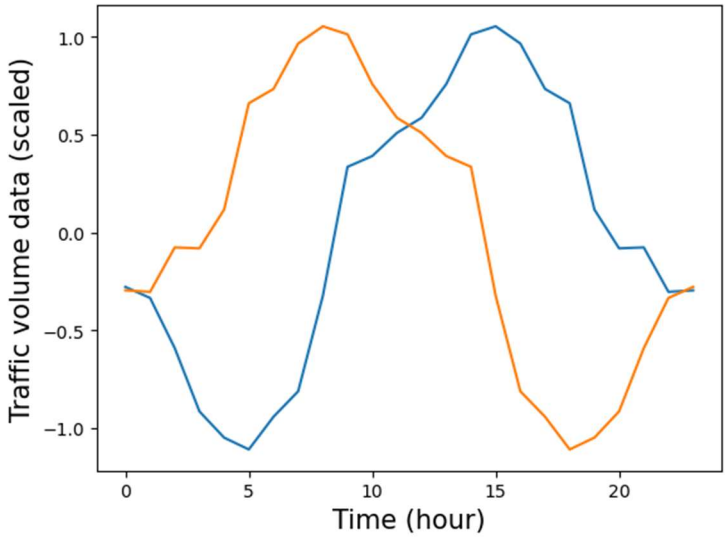


Figure 34: Average day of *TVD* (blue) and its reverse time series (orange).

Unlike DBSCAN, the AE was able to detect the contextual anomalies, but only when happening in all of the seven other KPIs simultaneously. Reversed days in individual KPIs were not enough to trigger the anomaly detector, which could cause challenges for the RCA. However, in the cases where the contextual anomaly was inserted in only one of the KPIs, the remaining six KPIs during that day followed the average day, which led to a minimal reconstruction error from those KPIs. In real-world situations the days are seldom "perfect" and therefore a contextual anomaly in a single KPI in a real-world setting would more likely be detected.

These manually created data points demonstrate that both DBSCAN and LSTM AE with the same parameters as those used for the actual data, perform reasonably as anticipated. Their sensitivity to point anomalies could be questioned but point anomalies should not cause much trouble to network operators, as they can be filtered out with simple thresholds. The real challenge is posed by the contextual anomalies, which in this experiment were not identified by DBSCAN, but were successfully detected by LSTM AE.

4.5 Overall results

The number of days classified as abnormal as well as the number of detected anomalies among the entire dataset for each sector of each BS is displayed in Figure 35. Overall, the anomaly detector detected 156 anomalies across the dataset out of 1 061 abnormal days. A few observations can be made from the figure. Firstly, each sector has abnormal days. This is no surprise, as a period of over six months can be expected to have at least a few days that are not entirely normal, whether due to anomalies or occurrences like holidays.

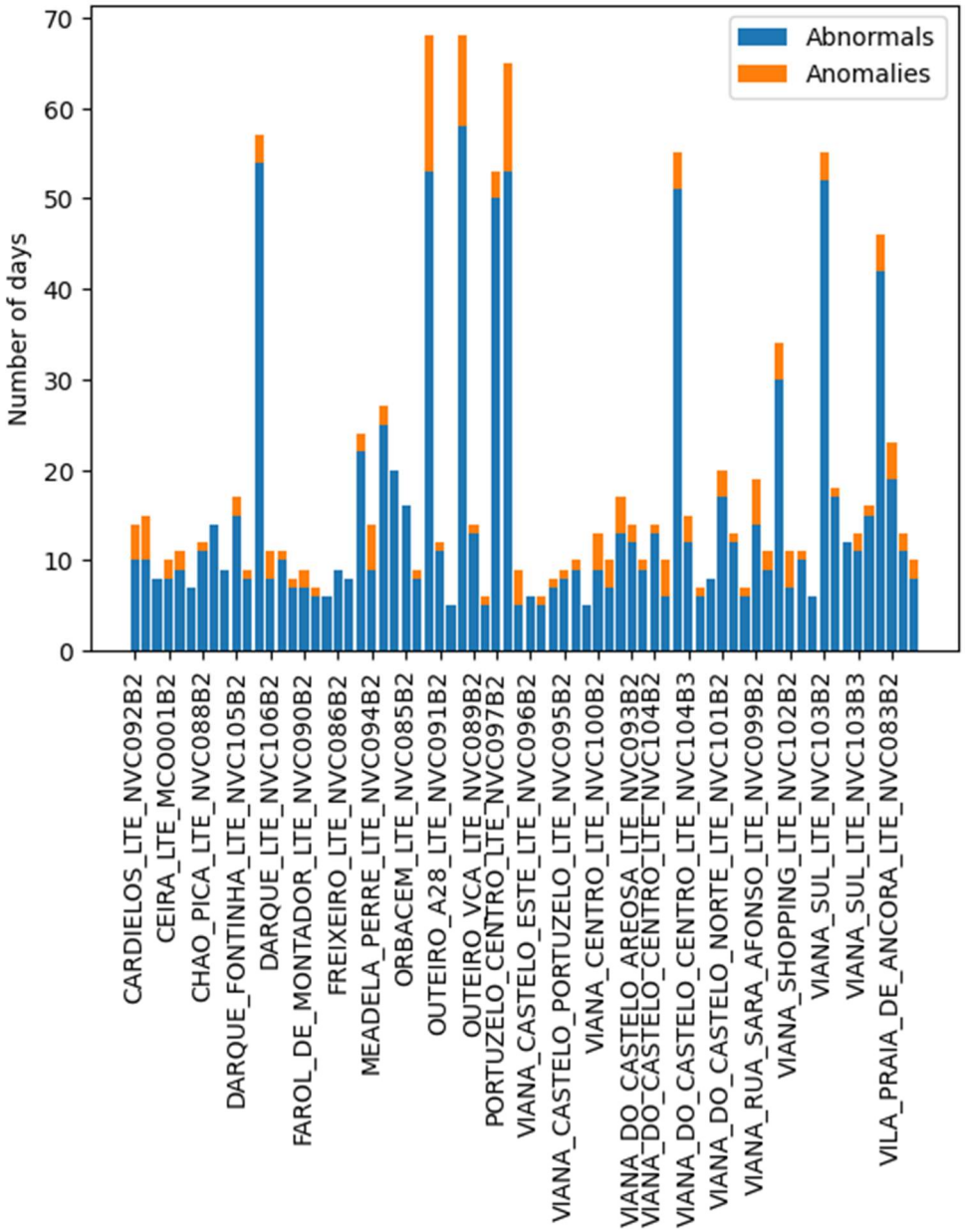


Figure 35: Bar plot of the number of abnormal and anomalous days in the sectors of all base stations.

Secondly, several sectors exhibit a high number of abnormal days, with seven sectors having 50 or more such days. A closer examination revealed that in the sectors where the most abnormal days occurred, most or even all weekends were classified as abnormal. This behaviour is of course undesirable. DBSCAN relies on data density, and if the data does not meet the density criteria set in its parameters, it may happen that all or most of the days are classified as abnormal. To address this issue, the parameters of DBSCAN could be adjusted to classify fewer days as abnormal. However, this might result in multiple normal clusters, making it harder to define normal behaviour. Notably, the problem of classifying all days as abnormal only arises on weekends, not weekdays, where there are significantly more days in the data. This would suggest that increasing the amount of data could potentially resolve this issue.

Thirdly, despite the most anomalies having been detected in the sectors with the most abnormal days, and some of the sectors with fewest abnormal days not having any anomalies, there does not seem to be any definite correlation between the number of abnormal days and anomalous days. Additionally, as mentioned before, the fact that the sectors with the most abnormal days are most likely due to undesirable behaviour of the DBSCAN in clustering of the weekends, supports this observation.

As it has been stated before, RCA is a process highly dependent on expert knowledge, and even an experienced network expert might not be able to determine root causes of anomalies happening in a network they are not familiar with. Even though the work has been conducted in collaboration with a telecom operator, specified expert input was not available for this RCA process. However, some idea of the root causes of the anomalies can be gained through analysis of the MSEs of individual KPIs. The quantities of the root causes of the 156 detected anomalies are displayed in Figure 36. Altogether, there were 327 root causes given by the model. This means that multiple anomalies were caused by a variety of distinct root causes, instead of just one. It needs to be emphasised that root causes are only potential ones given by the model and have not been verified by a network expert.

Two of the most common root causes identified were related to the usage of physical resource blocks, especially in uplink. Challenges with PRB usage rates hint at inefficient resource allocation, which might stem from issues such as interference. Even though the analysis at the KPI level is the most profound achievable with the current dataset, it could be argued that interference may be the actual underlying root cause, even deeper than the PRB KPIs themselves.

HSRs have been a frequent topic of discussion in this paper, and rightfully so by looking at the figure. HSRs, especially intra-frequency HSR, were a cause of many anomalies. HSRs are affected by poor coverage, for example resulting in connections being dropped due to the user not being close enough to a BS. Another factor contributing to low HSR is again interference.

In addition, sectors facing highways or train tracks could experience lower HSRs due to the rapid movement of devices.

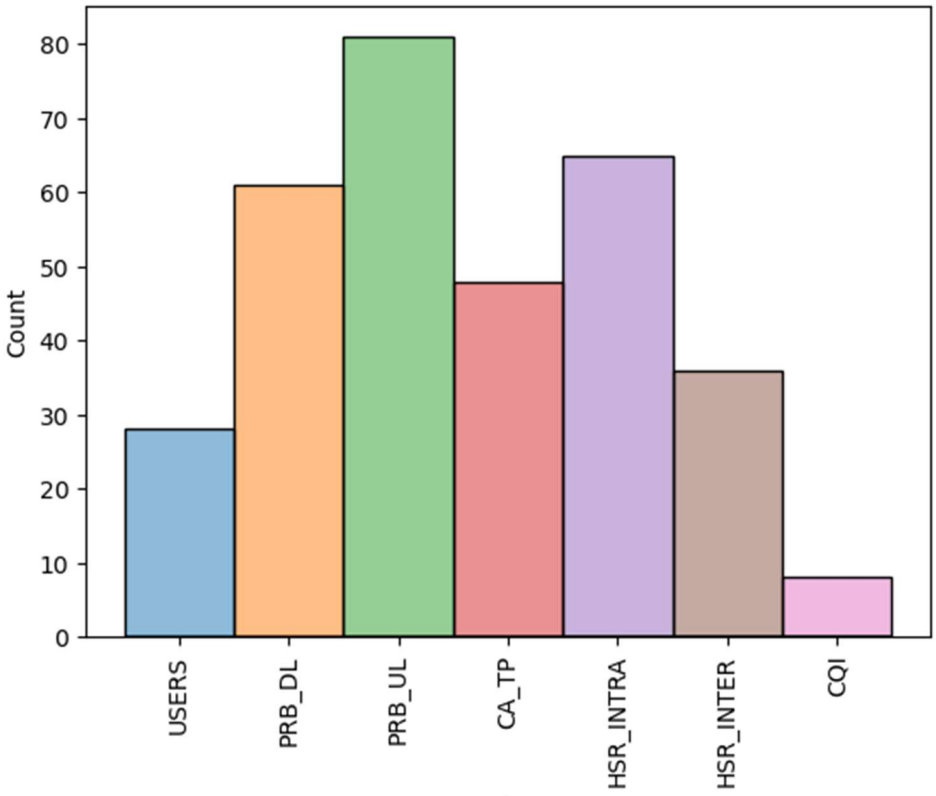


Figure 36: Root causes of the detected anomalies.

Unexpected behaviour of carrier aggregation throughput was a cause in almost 50 anomalies. This means that there might have been issues in combining multiple carriers to improve data rates and network performance. There are again a myriad of potential reasons, such as suboptimal network configuration, but carrier aggregation throughput could also be affected by PRB usage.

Unexpected behaviour in the number of users could be due to natural human behaviour, but it could also be caused by for example network unavailability. Lastly, CQI was a cause of only a few anomalies, potentially resulting from interference or propagation losses.

4.6 Discussion

In addition to the implemented system, there exist alternative approaches to perform each step of the system pipeline. This subsection contains considerations and discussion of the obtained results of data scaling, DBSCAN, LSTM

AE as well as completely different approaches for resolving the problem of anomaly detection and RCA.

4.6.1 Alternative approaches

The presented architectures of the models were selected after experimenting with multiple different approaches. Given the amount of data, its KPIs, missing values, and most importantly, the lack of labels, many of the approaches were not adequate for detecting anomalies in the dataset. The final LSTM AE model is a reactive anomaly detector, which means that it cannot predict the occurrence of anomalies beforehand, thus it is not proactive. Over the course of recent years, a paradigm shift from reactive to proactive anomaly detection has been taking place as discussed in Section 2.5, and therefore proactive approaches were also experimented.

One of the approaches to implement proactive anomaly detection was to predict the *HSR*s, because failed handovers are an especially good indication if there is something wrong with the network. An LSTM predictor was trained with different sets of KPIs, including only handovers, to predict the *HSR*s for the next hour, based on the previous n hours ($n = 1 \ldots 24$). However, this approach did not yield any promising results, mainly because the data is hourly aggregated, which makes predicting the averages of full hours extremely difficult. Handovers might show some autocorrelation over the course of short periods of time, but with the sampling rate of one hour, the prediction accuracy was very low.

Another approach was to use a random forest classifier based on the other KPIs to predict if the *HSR*s of the next hour were below 100 % or not. This approach was to try to find a connection between the *HSR*s and the other KPIs of the preceding hour, but again, the hourly nature as well as the lack of correlation between *HSRs* and other KPIs proved this task very difficult, resulting in inaccurate predictions.

After the experimentations, it became evident that proactive anomaly detection is a far more complex problem than reactive anomaly detection. For being able to use a proactive model, most likely more refined data would be required, say, data with a sampling rate of one minute instead of one hour. In addition, a larger set of KPIs could give more information of the network's operation. Nevertheless, the most important aspect missing from the dataset is most likely data labels. Prediction is typically associated with supervised learning, and without labelled data, training supervised learning models is challenging to say the least.

4.6.2 Scaling and DBSCAN considerations

In this work, standard scaling is used to scale the data into similar scales so that no single KPI would dominate in importance just due to its magnitude.

It is common to use standard scaling for data that follows a Gaussian distribution [77]. However, the distributions for different KPIs are different from each other. For example, *HSR*s are from the range of 0 – 100, with a heavily left-skewed distribution whereas *CQI* follows a more Gaussian distribution. It could have been an alternative approach to scale different KPIs with different scaling techniques based on their distributions. For example, non-Gaussian distributions, such as the *HSR*s, could have been scaled using Min-Max scaling, as suggested in [77]. However, when alternative scaling techniques were experimented, notable changes in the results in DBSCAN or anomaly detection were not observed.

The part of the system where the choice of scaling has the most effect, could ultimately be the root cause analysis. The way automatic RCA was performed in this work was by taking the mean of all KPI-wise errors. The KPIs where the error is larger than the mean are considered root causes. Scaling influences the interrelationships of the volumes of individual KPIs. With standard scaling, even though the range of the values ended up being roughly similar, there still existed some differences. In other words, the choice of scaling could result in one KPI requiring more unusual behaviour than another to be considered a root cause.

DBSCAN in an integral part of the developed system and it shows promise in time series clustering of unlabelled data. However, the usefulness of it as a part of this system could be questioned. As seen in Section 4.2, there were many anomalies within the days labelled normal by the DBSCAN, which could suggest that using DBSCAN with only *TVD* is not a definitive way of separating normal data from anomalous data in order to find network anomalies. Additionally, it shows that abnormalities in traffic volume are rarely linked to anomalies in the set of KPIs available in data. Consequently, this could mean that the set of KPIs used is not necessarily enough for performing a thorough anomaly detection. Instead, having a larger set of KPIs, using more or different KPIs in the DBSCAN stage or using a completely alternative approach could be considered.

4.6.3 LSTM AE considerations

LSTM AE is the most crucial part of the proposed system. It is responsible for anomaly detection as well as RCA. In this subsection, alternative solutions and modifications to the final LSTM AE model are considered.

The final model works with the precision of one day, meaning that it only alerts of an anomaly at the end of a day. A more reactive approach with the precision of one hour was also experimented with the LSTM AE. Here, instead of using full days starting from 01:00 and ending at 24:00, the approach was to use a sliding window of 24 hours to generate data points, with the goal of being able to alert of anomalies after the hour on which they occur. The windowed data had 24-hour sequences where the traffic peak happened

during the day as well as days where the peak was during the evening or midnight. Whereas the strength of the final model is accurately identifying daily patterns and detecting days deviating from these patterns, the sliding window-trained model was trained with data containing more versatile patterns. As a result, it became able to accurately reconstruct practically any pattern with low reconstruction errors. However, this behaviour is not desirable in AE-based anomaly detection, as it resulted in especially contextual anomalies not being detected.

Considering the precision of a day, the final model is most likely not very useful in networks which require immediate response to problems. On the other hand, the full day's precision is not necessarily a detriment, as it can allow for detecting long-lasting anomalies. If a more reactive approach is required, a solution could be to use non-overlapping data windows instead of the experimented sliding window solution with overlapping windows. This would slice the 24-hour days into chunks of, for example, 6 hours, which would not allow the LSTM AE to learn all possible patterns, while potentially increasing the precision to a six-hour interval.

The following considerations revolve around the training of the current model. One of which would make the deployment of the model much faster and lighter, as instead of training the LSTM AE for each sector of each BS, it could be trained for a single sector and then immediately used for the remaining sectors as a pre-trained model. Figure 37 displays the number of abnormal and anomalous days in the data when the model was trained only with one sector of a single BS. The final model, which was trained for each sector of each BS individually, detected 156 anomalies, but when training only with a single sector, 209 anomalies were detected. The DBSCAN phase was the same in both, thus the number of abnormal days remained the same 1061. Although, the number of anomalies detected increased by more than 33 %, visual comparison between Figure 35 and Figure 37 shows that there are numerous sectors where the number of anomalies did not increase at all. This is particularly clear in the sectors where no anomalies were detected. Out of 14 sectors where no anomalies were previously detected, 9 remained without anomalies. Altogether, the number of anomalies remained the same in 28 of 70 sectors.

The single sector that was used for the training is sector one of the training BS (Figure 19), which has a dominant traffic pattern of a residential area. With closer inspection it seems that the number of anomalies did not change at all or did not drastically change in the sectors also displaying characteristics of a residential area. The dataset used in this work is only from a period of six months, but in reality, the used dataset could contain much more data, and therefore the computational load of training the model becomes a more serious concern. In that case, based on these results, it is a possible solution to train the model not for each sector, not for only one sector, but for each traffic pattern individually. The traffic patterns could for

example be the five different ones identified by [67] and elaborated in Section 3.2. This would drastically reduce the computational load while preserving the accuracy in detecting anomalies.

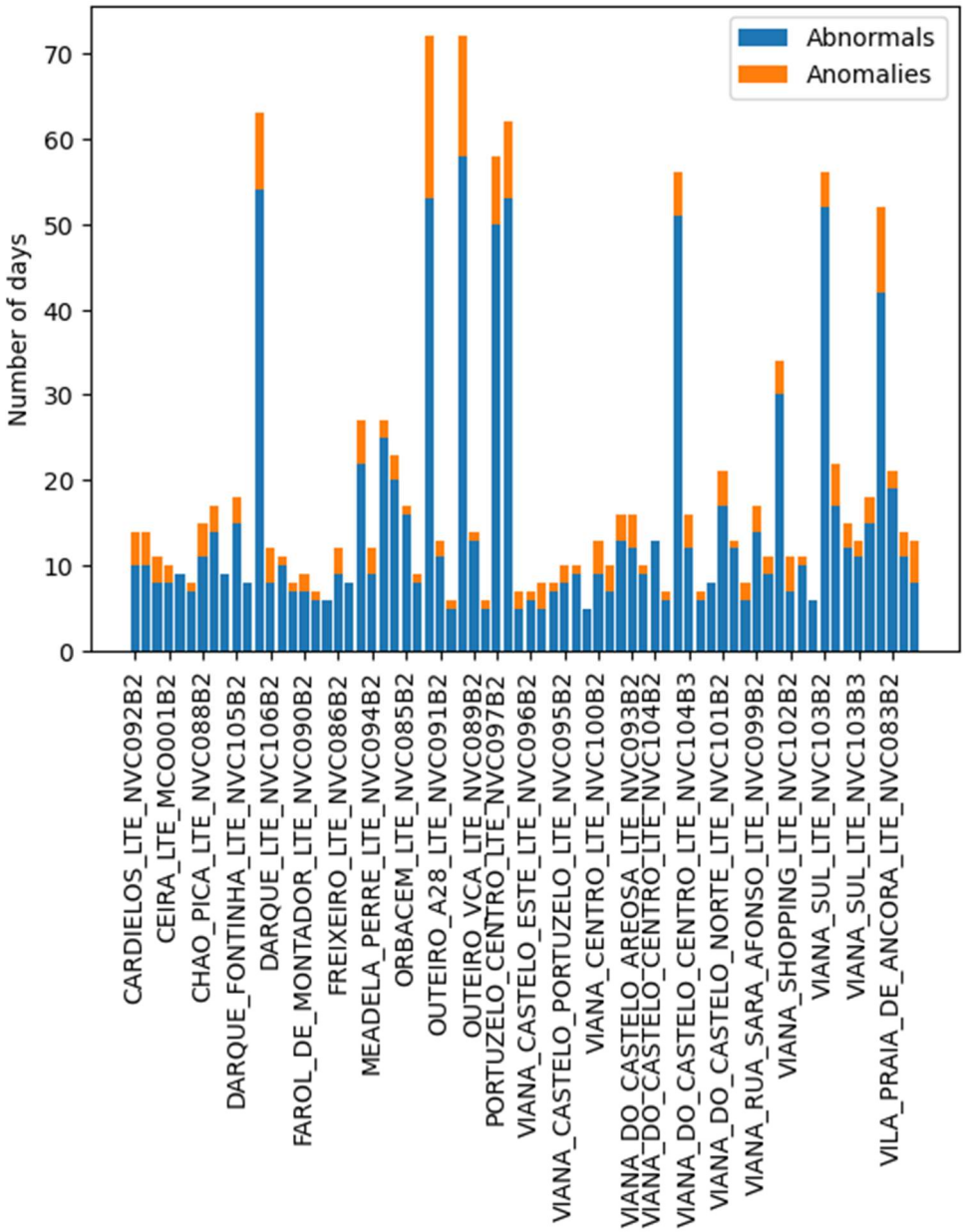


Figure 37: Bar plot of the number of abnormal and anomalous days in the sectors of all base stations when training the AE only with one sector.

As has been discussed before, the high reconstruction errors among the training sets (sets containing only normal days from DBSCAN) of some sectors, could indicate that DBSCAN is not an optimal solution to separate normal days from abnormal with the KPIs used. An alternative approach

would be to omit the DBSCAN from the system pipeline, and to use the reconstruction errors from the LSTM AE to obtain the set of normal days. Specifically, all available days would be used to train the AE, and the ones receiving the highest training error would be considered anomalies. This iterative process could be repeated until no training errors exceeded a certain threshold. If this approach would be discovered to be viable, it would significantly reduce the need for manual labelling. The downsides would be the increased computational load as well as potentially overfitting the model.

The final LSTM AE consideration is about the depth of the AE. As the used model only consists of single LSTM layers in both the encoder and the decoder, the network is not as deep as most of the ones used in the related work. Questions can be raised if the network is actually deep enough to accurately reconstruct the data. The shallower architecture has one obvious benefit in contrast to deeper models, which is the computational load. With the current architecture the model is able to learn the simple normal behaviour of the sector. However, the high training errors indicate that the model is not able to learn some of the more complex behaviours of the network, even if they are represented in the training data. Nevertheless, it is not necessarily a detriment, especially in the situations where normal days include anomalies, as it allows for learning the simple normal patterns, which results in detecting the clearest anomalies. If learning more complex normal behaviours is required, a deeper architecture could be used, but in that case, it is crucial that the normal data is clear of anomalies.

Automatic RCA in this work takes advantage of the reconstruction errors generated by the LSTM AE by comparing the MSEs of individual KPIs to the average of all the KPI MSEs within a single day. As a result, all KPIs are treated with equal importance. Depending on the characteristics of the network, it might be useful to use different weights for different KPIs. For instance, consider a KPI that even with minor deviations from the normal, is frequently linked to significant anomalies such as the ones directly impacting QoE. Contrast this with another KPI that despite exhibiting substantial variations from the normal, rarely severely influences QoE. In such scenarios, it would be reasonable to assign weights to the KPIs based on their respective significance. To determine the significance of each KPI, knowledge about the network as well as the goals of the network operator are required.

5 Conclusions and future work

The primary objective of this thesis was to develop an automatic machine learning-based system able to detect anomalies and conduct RCA in unlabelled mobile network data. The thesis also aimed to answer questions about general anomaly detection and RCA in mobile networks, performing them using only unlabelled data as well as the effect of 5G network deployment to the mobile anomaly detection and RCA landscape.

This thesis provided a comprehensive overview of 4G and 5G networks, QoE, machine learning methods, SONs and the principles of anomaly detection and RCA, along with a discussion of related work. A system was developed to detect anomalies and to determine their root causes in unlabelled 4G network data. The system (described in Chapter 3) could work as a part of the self-healing function of a self-organising network to reduce the problems and downtime of the network which is expected to lead to an increase in the objective QoE of the network users.

Data analysis was performed to find different features of the data, including similarities between different cells within sectors, differences between sectors of the same BS as well as weekend and weekday traffic patterns. These features were used to determine some of the design choices of the system. The system preprocesses the data, including splitting it into days, removing days with missing data points and scaling the KPIs. It uses DBSCAN clustering to acquire a set of normal days from both weekdays and weekends of each sector, which are then used for training of a sector-wise LSTM AE model which can be used for day-wise anomaly detection and RCA.

The results of applying the system to the available dataset and the manually created artificial data revealed promising outcomes. Specifically, DBSCAN demonstrated proficiency in being able to differentiate between normal and abnormal network traffic patterns of 24-hour time series sequences. Meanwhile, LSTM AE showcased its ability to detect anomalous days of various types based on their reconstruction errors. In addition, it could distinguish the individual KPIs mostly contributing to the anomalies, paving way for automatic RCA.

To summarise the answers for the research questions presented in Chapter 1, anomaly detection and RCA can be performed in various ways depending on the nature of the data. Traditionally used statistical and threshold-based methods often relying on expert input may be enough to address the needs of simple and smaller networks, but as networks increase in size and complexity, there is a growing trend towards machine learning-based models such as random forests and DNNs.

With mobile network data often being unlabelled, the set of suitable machine-learning methods is narrowed down. Due to their independence of labelled data, unsupervised clustering and AE-based models have been at the centre of many anomaly detectors throughout the recent years. Both can be

used also for RCA, but clustering-based models tend to depend on expert input during the setup of the model.

It is important to note that although this work uses a 4G dataset, the proposed model is not limited to any specific network technology or a set of KPIs (apart from *TVD*). The model works purely from the basis of network KPIs and therefore, in principle, could be applied to any network that generates them, including 5G. However, in the upcoming years, the increasing complexity and the number of features while moving towards non-standalone 5G networks, will set new requirements for mobile network anomaly detection. With different application scenarios, frequency bands and network slicing, a wide range of normal behaviours can be expected even within the same geographical region. Therefore, either multiple different models monitoring different parts of the network or dynamic models capable of adjusting their behaviour are needed. Fortunately, the decentralised MEC architecture of 5G will bring computing power to the edges of the network, therefore enabling real-time data processing and robust anomaly detection close to where data is generated. Real-time anomaly detection is crucial especially for URLLC applications, underlining the potential necessity for proactive anomaly detection. In addition, the sheer volume of data projected to be transferred by 5G networks makes detecting, and more importantly, reacting to every single anomaly infeasible. For this reason, not all the anomalies should be causes of alarms, but network operators should rather concentrate on narrowing down to a set of specific anomalies. Such as the ones directly impacting QoE of the users, and for which RCA can be performed and the root causes addressed.

Further aspirations about improvements of the developed model will focus on experimenting with more granular data containing more KPIs and at least some anomaly and root cause labels. This will support the fine-tuning of the model parameters, the development of a more reactive model, outputting of more specific root causes as well as verifying the results more reliably. Using GANs to generate new artificial data to increase the amount of labelled data or to characterise more versatile network conditions is also an interesting prospect. Another important topic that was not included in this thesis is the computational requirements of the proposed model. In future, the computational overhead of the developed model should be compared to other models to determine whether it is feasible to deploy the model in a distributed network architecture, for example in 5G small cells, or whether a centralised setting is more suitable. Finally, deployment of the model in an operating mobile network and evaluating its impact to the operator and users, for example in terms of operational efficiency and QoE, would serve as the ultimate test.

To conclude, machine learning-based anomaly detection and root cause analysis are complex topics with use cases not only in mobile networks but in other fields as well. The design choices of the models are affected by

the quality of the data, whether it is labelled or not, specific goals and the desired accuracy as well as available domain knowledge. The development of automatic anomaly detectors and root cause analysis solutions will be crucial activities moving forward towards fully self-organising mobile networks in 5G and beyond.

References

[1] P. V. Klaine, M. A. Imran, O. Onireti and R. D. Souza, "A Survey of Machine Learning Techniques Applied to Self Organizing Cellular Networks," *IEEE Communications Surveys & Tutorials,* vol. 19, pp. 2392-2431, 2017.

[2] Ericsson, "Network coverage outlook," Ericsson, [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/network-coverage. [Accessed 22 March 2023].

[3] S. Forconi and M. Vaser, "4G LTE architectural and functional models of Video Streaming and VoLTE services," in *Seventh International Conference on Ubiquitous and Future Networks*, Sapporo, Japan, 2015.

[4] Tutorialspoint, "LTE Network Architecture," Tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/lte/lte_network_architecture.htm. [Accessed 6 March 2023].

[5] G. Liu, Y. Huang, Z. Chen, L. Liu, Q. Wang and N. Li, "5G Deployment: Standalone vs. Non-Standalone from the Operator Perspective," *IEEE Communications Magazine,* vol. 58, no. 11, pp. 83-89, 2020.

[6] Huawei, "SA and NSA," Huawei, 28 June 2022. [Online]. Available: https://forum.huawei.com/enterprise/en/topic-discussion-sa-and-nsa/thread/896687-100305?page=1. [Accessed 14 November 2022].

[7] H. Ekström, "Non-standalone and Standalone: two standards-based paths to 5G," Ericsson, 11 July 2019. [Online]. Available: https://www.ericsson.com/en/blog/2019/7/standalone-and-non-standalone-5g-nr-two-5g-tracks. [Accessed 23 March 2023].

[8] 3GPP, "System Architecture for the 5G Systems (3GPP TS 23.501 version 15.3.0 Release 15)," ETSI, 2019.

[9] A. A. Barakabitze, A. Ahmad, R. Mijumbi and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks,* vol. 167, 2020.

[10] N. Hassan, K.-L. A. Yay and C. We, "Edge computing in 5G: A review," *IEEE Access,* vol. 7, pp. 127276-127289, 2019.

[11] A. Hossein, "Edge Computing vs Fog Computing: 10 Key Comparisons," Spiceworks, 2022. [Online]. Available: https://www.spiceworks.com/tech/cloud/articles/edge-vs-fog-computing/. [Accessed 18 November 2022].

[12] T. Akhtar, C. Tselios and I. Politis, "Radio resource management: approaches and implementations from 4G to 5G and beyond," *Wireless Networks,* vol. 27, pp. 693-734, 2020.

[13] Huawei, “5G Basics: What it's all about,” Huawei, [Online]. Available: https://seedsforthefuture.shixizhi.huawei.fr/course/1554279059139616770/application-view?courseId=2209131558556532513&appId=5036424650879508 48&appType=1&activeIndex=0&sxz-lang=en_US. [Accessed 12 October 2023].

[14] Nokia, "5G spectrum bands explained — low, mid and high band," Nokia, [Online]. Available: https://www.nokia.com/thought-leadership/articles/spectrum-bands-5g-world/. [Accessed 22 November 2022].

[15] Huawei, "5G Spectrum Public Policy Position," Huawei, 2020.

[16] Traficom, "Matkaviestinverkkojen taajuudet ja luvanhaltijat," Traficom, 2022. [Online]. Available: https://www.traficom.fi/fi/viestinta/viestintaverkot/matkaviestinverkkojen-taajuudet-ja-luvanhaltijat. [Accessed 28 January 2023].

[17] X. Lin, J. Li, R. Baldemair, J.-F. T. Cheng, S. Parkvall, D. C. Larsson, H. Koorapaty, M. Frenne, S. Falahati, A. Grovlen and K. Werner, "5G New Radio: Unveiling the Essentials of the Next Generation Wireless Access Technology," *IEEE Communications Standards Magazine,* vol. 3, pp. 30-37, 2019.

[18] S. Parkvall, E. Dahlman, A. Furuskar and M. Frenne, "NR: The New 5G Radio Access Technology," *IEEE Communications Standards Magazine,* vol. 1, no. 4, pp. 24-30, 2017.

[19] O. O. Erunkulu, A. M. Zungeru, C. K. Lebekwe and M. Mosalaosi, "5G Mobile Communication Applications: A Survey and Comparison of Use Cases," *IEEE Access,* vol. 9, pp. 97251-97295, 2021.

[20] H. R. Barzegar, N. E. Ioini, V. T. Le and C. Pahl, "Wireless Network Evolution Towards Service Continuity in 5G enabled Mobile Edge Computing," in *Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, Paris, France, 2020.

[21] Huawei, "Introduction to 5G, What is the QCI?," Huawei, 15 June 2021. [Online]. Available: https://forum.huawei.com/enterprise/en/introduction-to-5g-what-is-the-qci/thread/747701-100305. [Accessed 10 February 2023].

[22] M. Varela, L. Skorin-Kapov and T. Ebrahimi, "Quality of Service Versus Quality of Experience," in *Quality of Experience*, Springer, 2014, pp. 85-96.

[23] G. Kougioumtzidis, V. Poulkov, Z. D. Zaharis and P. I. Lazaridis, "A Survey on Multimedia Services QoE Assessment and Machine Learning-Based Prediction," *IEEE access : practical innovations, open solutions,* vol. 10, pp. 19507-19538, 2022.

[24] Ericcson, "Network performance through customers' eyes," Ericcson, [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/network-perf-customers-eyes. [Accessed 6 February 2023].

[25] E. Boz, B. Finley, A. Oulasvirta, K. Kilkki and J. Manner, "Mobile QoE prediction in the field," *Pervasive and Mobile Computing,* vol. 59, no. 101039, 2019.

[26] M. Kulin, T. Kazaz, E. De Poorter and I. Moerman, "A Survey on Machine Learning-Based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer," *MDPI Electronics,* vol. 10, no. 3, 2021.

[27] C. Janiesch, P. Zschech and K. Heinrich, "Machine learning and deep learning," *Electronic Markets,* vol. 31, p. 685–695, 2021.

[28] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang and J. Zhang, "Self-Supervised Learning: Generative or Contrastive," *IEEE Transactions on Knowledge and Data Engineering,* vol. 35, no. 1, pp. 857-876, 2023.

[29] A. S. Lampropoulos and G. A. Tsihrintzis, "Applications in recommender systems," in *Machine learning paradigms*, Swizerland, Springer, 2015, pp. 33-34.

[30] M. Miller, "The basics: Linear regression," Towards data science, October 2019. [Online]. Available: https://towardsdatascience.com/the-basics-linear-regression-2fc9f5124687. [Accessed 7 December 2022].

[31] Scikit-learn, "Naive Bayes," Scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html. [Accessed 22 September 2023].

[32] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR),* vol. 9, pp. 381-386, 2018.

[33] R. Gandhi, "Support Vector Machine — Introduction to Machine Learning Algorithms," Towards data science, June 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed 7 December 2022].

[34] C. C. Aggarwal and C. K. Reddy, Data Clustering : Algorithms and Applications, CRC Press LLC, 2013.

[35] Scikitlearn, "Clustering," Scikitlearn, [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html. [Accessed 11 December 2022].

[36] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment,* vol. 32, no. 14-15, pp. 2627-2636, 1998.

[37] Paperswithcode, "Convolutional Neural Networks,," Paperswithcode, [Online]. Available: https://paperswithcode.com/methods/category/convolutional-neural-networks. [Accessed 6 March 2023].

[38] P. Le and W. Zuidema, "Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs," *arXiv preprint,* 2016.

[39] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun and D. Pei, "Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage, USA, 2019.

[40] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Y. Z. Liu, L. Wang, C. Li and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open,* vol. 1, pp. 57-81, 2020.

[41] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI Conference on Artificial Intelligence*, Virtual, 2021.

[42] J. Audibert, P. Michiardi, F. Guyard, S. Marti and M. A. Zuluaga, "USAD: Unsupervised anomaly detection on multivariate time series," in *26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual, 2020.

[43] F. Chollet, "Building Autoencoders in Keras," The Keras Blog, 14 May 2016. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html. [Accessed 7 July 2023].

[44] J. Moysen and L. Giupponi, “From 4G to 5G: Self-organized network management meets machine learning,” *Computer Communications,* vol. 129, pp. 248-268, 2018.

[45] K. Choi, J. Yi, C. Park and S. Yoon, "Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines," *IEEE Access,* vol. 9, pp. 120043-120065, 2021.

[46] J. Audibert, P. Michiardi, F. Guyard, S. Marti and M. A. Zuluaga, "Do deep neural networks contribute to multivariate time series anomaly detection?," *Pattern Recognition,* vol. 132, 2022.

[47] C. M. Ahmed, V. R. Palleti and A. P. Mathur, "WADI: A water distribution testbed for research in the design of secure cyber physical systems," in *The 3rd international workshop on cyber-physical systems for smart water networks*, Pittsburgh, USA, 2017.

[48] J. Rooney, "Root cause analysis for beginners," *Quality progress,* vol. 37, no. 7, pp. 45-56, 2004.

[49] A. Gómez-Andrades, P. Muñoz, I. Serrano and R. Barco, "Automatic Root Cause Analysis for LTE Networks Based on Unsupervised Techniques," *IEEE Transactions on Vehicular Technology,* vol. 65, no. 4, pp. 2369-2386, 2016.

[50] H. Mfula and J. K. Nurminen, "Adaptive Root Cause Analysis for Self-Healing in 5G Networks," in *2017 International Conference on High Performance Computing & Simulation (HPCS)*, Genoa, Italy, 2017.

[51] N. O. Tippenhauer and A. P. Mathur, "SWaT: a water treatment testbed for research and training on ICS security," in *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, Vienna, Austria, 2016.

[52] K. Hundman, V. Constantinou, C. Laporte, I. Colwell and T. Soderstrom, "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in *The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, UK, 2018.

[53] L. Shen, Z. Li and J. T. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," in *Advances in Neural Information Processing Systems*, Virtual, 2020.

[54] Z. Chen, D. Chen, X. Zhang, Z. Yuan and X. Cheng, "Learning graph structures with transformer for multivariate time series," *IEEE internet of things journal,* vol. 9, no. 12, pp. 9179-9189, 2022.

[55] W. Zhang, R. Ford, J. Cho, C. J. Zhang, Y. Zhang and D. Raychaudhuri, "Self-organizing cellular radio access network with deep learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, 2019.

[56] H. D. Trinh, E. Zeydan, L. Giupponi and P. Dini, "Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach," *IEEE Access,* vol. 7, pp. 152187-152201, 2019.

[57] Y. Yuan, J. Yang, R. Duan, I. Chih-Lin and J. Huang, "Anomaly Detection and Root Cause Analysis Enabled by Artificial Intelligence," in *2020 IEEE Globecom Workshops (GC Wkshps)*, Taipei, Taiwan, 2020.

[58] K.-F. Chen, C.-H. Lin and T.-S. Lee, "Deep Learning-Based Multi-Fault Diagnosis for Self-Organizing Networks," in *ICC 2021 - IEEE International Conference on Communications*, Montreal, Canada, 2021.

[59] C. V. Murudkar and R. D. Gitlin, "QoE-driven anomaly detection in self-organizing mobile networks using machine learning," in *2019 Wireless Telecommunications Symposium (WTS)*, New York, USA, 2019.

[60] G. Y. Z. Q. Z. Y. Chen G, "A Novel Cellular Network Traffic Prediction Algorithm Based on Graph Convolution Neural Networks and Long Short-Term Memory through Extraction of Spatial-Temporal Characteristics.," *Processes,* vol. 11, no. 8, 2023.

[61] T. Zhang, K. Zhu and D. Niyato, "A Generative Adversarial Learning-Based Approach for Cell Outage Detection in Self-Organizing Cellular Networks," *IEEE Wireless Communications Letters,* vol. 9, no. 2, pp. 171-174, 2020.

[62] B. Hussain, Q. Du, S. Zhang, A. Imran and M. A. Imran, "Mobile Edge Computing-Based Data-Driven Deep Learning Framework for Anomaly Detection," *IEEE Access,* vol. 7, pp. 137656-137667, 2019.

[63] U. S. Hashmi, A. Rudrapatna, Z. Zhao, M. Rozwadowski, J. Kang, R. Wuppalapati and A. Imran, "Towards Real-Time User QoE Assessment via Machine Learning on LTE Network Data," in *IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Honololu, USA, 2019.

[64] M. S. Parwez, D. B. Rawat and M. Garuba, "Big Data Analytics for User-Activity Analysis and User-Anomaly Detection in Mobile Wireless Network," *IEEE Transactions on Industrial Informatics,* vol. 13, no. 4, pp. 2058-2065, 2017.

[65] K. Sultan, H. Ali and Z. Zhang, "Call Detail Records Driven Anomaly Detection and Traffic Prediction in Mobile Cellular Networks," *IEEE Access,* vol. 6, pp. 41728-41737, 2018.

[66] Y. Liu, Y. Mu, K. Chen, Y. Li and J. Guo, "Daily Activity Feature Selection in Smart Homes Based on Pearson Correlation Coefficient.," *Neural processing letters,* vol. 51, p. 1771–1787, 7 Jan 2020.

[67] F. Xu, Y. Li, H. Wang, P. Zhang and D. Jin, "Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment," *IEEE/ACM transactions on networking a joint publication of the IEEE Communications Society, the IEEE Computer Society, and the ACM with its Special Interest Group on Data Communication.,* vol. 25, no. 2, pp. 1147-1161, 2017.

[68] S. Learn, “Scikit Learn,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html. [Accessed 18 June 2023].

[69] F. Chollet, "Building autoencoders in Keras," The Keras blog, 14 May 2016. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html. [Accessed 18 June 2023].

[70] Scikit-learn, "sklearn.preprocessing.StandardScaler," Scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. [Accessed 22 Sep 2023].

[71] N. S. Chauhan, "DBSCAN Clustering Algorithm in Machine Learning," KDnuggets, 4 April 2022. [Online]. Available: https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html. [Accessed 7 July 2023].

[72] D. K. Kotary and S. J. Nanda, "A Distributed Neighbourhood DBSCAN Algorithm for Effective Data Clustering in Wireless Sensor Networks.," *Wireless Personal Communications,* vol. 121, p. 2545–2568, 2021.

[73] Q. Xianting and W. Pan, "A Density-Based Clustering Algorithm for High-Dimensional Data with Feature Selection," in *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, Wuhan, China, 2016.

[74] K. K. Nisa, H. A. Andrianto and R. Mardhiyyah, "Hotspot clustering using DBSCAN algorithm and shiny web framework," in *2014 International Conference on Advanced Computer Science and Information System*, Jakarta, Indonesia, 2014.

[75] Scikit-learn, "Metrics and scoring: quantifying the quality of predictions," Scikit-learn, [Online]. Available: Metrics and scoring: quantifying the quality of predictions. [Accessed 22 September 2023].

[76] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, Halifax, Canada, 2017.

[77] D. U. Ozsahin, M. T. Mustapha, A. S. Mubarak, Z. S. Ameen and B. Uzun, "Impact of feature scaling on machine learning models for the diagnosis of diabetes," in *2022 International Conference on Artificial Intelligence in Everything (AIE)*, Lefkosa, Cyprus, 2020.